INFORMATION THEORY

HEINRICH MATZINGER
Georgia Tech
E-mail: matzi@math.gatech.edu

April 23, 2004

Contents

1 Notations

If S is a set then S^i denotes the set consisting of all finite sequences of length i of elements of S.
We write $S = \{s_1, s_2, \ldots, s_q\}$ for the set S consisting of the elements $s_1, s_2, \ldots, s_q$.

2 Introduction

The two primary concerns of information theory are:

- How to store information so that it takes as little space as possible. Typically we have a file that we want to compress as much as possible and store in binary form.
- How to transmit information in the presence of random errors in the communication channels.

3 Codes

Imagine that we want to store some genetic code. Sequences in genetics are written using four letters:

$$A, T, C, G.$$

However, to store this information on our computer we need to "translate" it in a binary form, since all the files stored in our computer are ultimately in binary form. One possibility could be for example:

$$A \to 1, T \to 10, C \to 100, G \to 1000.$$

In this manner, an A gives a 1 in the coded text, a T is coded as 10,... With this example, we obtain that the string:

$$AATAC$$

gets encoded as:

$$11101100.$$

The rule which allows us to write every finite sequence of letters from $\{A, T, C, G\}$ as a binary sequence is called a *code*. The original alphabet $\{A, T, C, G\}$ is called the *source alphabet* whilst $\{0, 1\}$ is called the code alphabet. In other words, a code assigns to each

finite sequence of letters from the source alphabet a finite binary sequence. A code is thus a map from the set of all finite sequences of letters from the source alphabet to the set of all finite binary sequences.

Definition 3.1 *Let $S = \{s_1, \ldots, s_q\}$ be a finite alphabet (a finite set of symbols). Then a map C from the set of all finite sequences of symbols of S into the set of all finite binary sequences is called a code. In other words, a code is a rule which assigns to each finite sequence of symbols of S a finite binary sequence.*

We will denote the text we want to encode by:

$$X = X_1 X_2 \ldots X_n.$$

The text X consists of a sequence of n letters of the alphabet S and $X_i \in S$ is the i-th letter of the text X.
In the above example of a code, we had that each letter of the alphabet S was coded separately into a finite binary string. This is a special situation. The codes which satisfy this special property are called *block codes*. Not all codes are built in this way. A code which would not be a block code could be constructed in the following way for example: assign to A the digit 1, to T the digit 2, to C a 3 and to G a 4. Then write the original text X be writing each number instead of the corresponding letter. This gives an integer number written in basis 10. Rewrite this number in basis two. This sequence of operations defines a code which is not a block code. For example, to code the short text AC, we have to write the integer 13 in binary form. This gives 1101 for the encoded text.

Definition 3.2 *Let C be a code:*

$$C : \bigcup_{i=1}^{\infty} S^i \to \bigcup_{i=1}^{\infty} \{0,1\}^i.$$

Then C is called a block code if there exists a map

$$c : S \to \bigcup_{i=1}^{\infty} \{0,1\}^i$$

such that for all $n \in \mathbb{N}$ and all $X = X_1 X_2 \ldots X_n \in S^n$ we have:

$$C(X) = c(X_1)c(X_2)\ldots c(X_n).$$

An important feature a code should have is that from the encoded information we can retrieve the original information. During the encoding process we do not want to lose information. We can retrieve the full information iff there are no two different texts which are encoded into the same binary text. In the opposite case, we would be unable to tell from which original text the encoded text comes from. Thus, it would be impossible, given only the encoded text, to retrieve the full original information.

Definition 3.3 *A code C:*

$$C : \bigcup_{i=1}^{\infty} S^i \to \bigcup_{i=1}^{\infty} \{0,1\}^i$$

is called uniquely decodable iff for any two texts $X \in \bigcup_{i=1}^{\infty} S^i$ and $Y \in \bigcup_{i=1}^{\infty} S^i$ such that $X \neq Y$ we have that

$$C(X) \neq C(Y).$$

Apart from being uniquely decodable there is still another issue. Look at our first example of a block code for the genetic alphabet:

$$C(A) = 1, C(T) = 10, C(C) = 100, C(G) = 1000.$$

This is a uniquely decodable code. Every time " a new letter starts" in the binary code, this is marked by a one. Given only the encoded version of a text, we can determine what the original text was. However, if we see only part of the text, we might not be able to decode this part correctly. For example, if we know that the text X starts with 1010010, then we now that the original text started with an AC, but we don't know if the next letter is a T, a C or a G. To figure out what the last 10 means, we need to know the subsequent digits. In many practical cases, this is a situation one tries to avoid. One prefers a code which one can decode bit by bit without having to look at subsequent bits. Such a code is called an *instantaneous code*. An example of an instantaneous block code C would be given by:

$$C(A) = 1, C(T) = 01, C(C) = 001, C(G) = 0001.$$

For a word $Y = Y_1 Y_2 \dots Y_m \in \{0,1\}^m$ any word $Y_1 Y_2 \dots Y_k$ for $k \leq m$ is called a *prefix* of the code word Y. For example the word 0001 has as prefix the following words: 0,00,000 and 0001. The word "word" has as prefix the words: "w","wo","wor" and "word". If the word v is a prefix of the word w, we write:

$$v \preccurlyeq w.$$

Definition 3.4 *A code $C : \cup_{i=1}^{\infty} S^i \to \cup_{i=1}^{\infty} \{0,1\}$ is called instantaneous(also prefix code) iff for all $X, Z \in \bigcup_{i=1}^{\infty} S^i$ we have that:*

$$C(Z) \preccurlyeq C(X)$$

implies that

$$Z \preccurlyeq X.$$

Note that a code which is instantaneous is also uniquely decodable.
For a block code the binary words that code the letters of the alphabet S are called *code words*. In the example of a block code given by

$$C(A) = 1, C(T) = 01, C(C) = 001, C(G) = 0001$$

the code words are: 1,01,001 and 0001.

Lemma 3.1 *A code is instantaneous iff no complete word of the code is a prefix to another code word.*

4 The Kraft inequality and the McMillan inequality

Let us look at the following block code:

$$C(A) = 1, C(T) = 01, C(C) = 001, C(G) = 0001.$$

Here A is coded into a word of length one, T is coded into a word of length 2, C is coded into a word of length 3 and G is coded into a word of length 4 and this is an instantaneous code. Now, if we where only given those lengths, is there a simple criteria to decide if there exists an instantaneous code with these lengths for the code words? A necessary and sufficient condition for the existence of an instantaneous code with given word lengths is provided by the *Kraft inequality* (Kraft, 1945).

Lemma 4.1 *Let $S = \{s_1, s_2, \ldots, s_q\}$ denote an alphabet. Let $l_1, \ldots, l_q$ denote a sequence of natural numbers. Then, there is an instantaneous block code for the alphabet S such that symbol s_i is coded into a word of length l_i for all $i = 1, 2, \ldots, q$ iff*

$$\sum_{i=1}^{q} \left(\frac{1}{2}\right)^{l_i} \leq 1$$

Before giving the proof of the above lemma, we describe an algorithm which constructs a instantaneous block code with given lengths for the code words.

Algorithm 4.1 *Write the symbols $s_1, s_2, \ldots, s_q$ in an an order so that the corresponding lengths become increasing:*

$$l_1 \leq l_2 \leq \ldots \leq l_q.$$

1. *Choose any binary word of length l_1 as code word $C(s_1)$ for s_1.*
2. *Once the first $r < q$ code words are chosen, choose a code word of length l_{r+1} for the symbol s_{r+1}. Choose the code word $C(s_{r+1})$, so that no previously chosen code word is a prefix to it.*

Next we need to prove that when Kraft's inequality holds, that is when:

$$\sum_{i=1}^{q} \left(\frac{1}{2}\right)^{l_i} \leq 1,$$

the above algorithm always works. This means, that at each step r where $r < q$, there exists a binary word w which satisfies the two following conditions:

1. None of the preciously chosen code words is prefix to w.
2. The length of w is l_{r+1}.

It is obvious that if at each step of our algorithm we can find a binary word satisfying the two above conditions, then the resulting block code will be instantaneous. Thus, we only need to prove that when Kraft's inequality holds then at each step one can find such a binary word. This is what we do next:

Proof. We proceed by induction on r. For s_1 it is clear that we can find a word of the right length since there is no other conditions then length.

Assume now that $r < q - 1$ and that for the symbols $s_1, s_2, \ldots, s_r$ we have found code words $C(s_1), C(s_2), \ldots, C(s_r)$ of the right lengths which are not prefix to each other. Kraft's inequality

$$\sum_{i=1}^{q} \left(\frac{1}{2}\right)^{l_i} \leq 1,$$

implies

$$\sum_{i=1}^{r} \left(\frac{1}{2}\right)^{l_i} < 1.$$

Multiplying both sides of the above inequality by $2^{l_{r+1}}$ yields:

$$\sum_{i=1}^{r} 2^{(l_{r+1}-l_i)} < 2^{(l_{r+1})}. \tag{4.1}$$

Note that $2^{l_{r+1}}$ is the number of binary words of length l_{r+1}. For a word w of length $s < l_{r+1}$, we have that the number of words of length l_{r+1} having w as prefix is equal to:

$$2^{l_r - s}.$$

Thus, the sum:

$$\sum_{i=1}^{r} 2^{l_{r+1}-l_i}$$

is equal to the number of binary words of length l_{r+1} having as prefix one of the code words:

$$C(s_1), \ldots, C(s_r).$$

In other words, inequality 4.1 tells us that there are strictly more binary words of length l_{r+1}, than there are binary words of that length with a one of the words

$$C(s_1), \ldots, C(s_r)$$

as prefix. This implies that our algorithm manages to find a suitable code word for s_{r+1}. ■

The next lemma says that Kraft's inequality does not only hold for a instantaneous codes, but also for uniquely decodable codes.

Lemma 4.2 *Let $S = \{s_1, s_2, \ldots, s_q\}$ denote an alphabet. Let $l_1, \ldots, l_q$ denote a sequence of natural numbers. Then, there is a uniquely decodable block code for the alphabet S, such that symbol s_i is coded into a word of length l_i for all $i = 1, 2, \ldots, q$ iff*

$$\sum_{i=1}^{q} \left(\frac{1}{2}\right)^{l_i} \leq 1$$

Proof. An instantaneous code is automatically also uniquely decodable. Thus, if the Kraft inequality holds, there exists a instantaneous code such that $|C(s_i)| = l_i$ for all $i = 1, 2, \ldots, q$. Thus, there exists also a uniquely decodable code such that $|C(s_i)| = l_i$ for all $i = 1, 2, \ldots, q$.
So, we only need to prove the reverse side. That is we need to prove that if there exist a uniquely decodable block code C with $|C(s_i)| = l_i$ for all $i = 1, 2, \ldots, q$ then also

$$\sum_{i=1}^{q} \left(\frac{1}{2}\right)^{l_i} \leq 1.$$

Let us proceed by the absurd. Suppose on the contrary that there exists $c > 1$ such that

$$\sum_{i=1}^{q} \left(\frac{1}{2}\right)^{l_i} > c.$$

Then, for $n \geq 1$ we have:

$$\left(\sum_{i=1}^{q} \left(\frac{1}{2}\right)^{l_i}\right)^{n} > c^n. \tag{4.2}$$

Let Q designate the set of natural numbers: $Q := \{1, 2, \ldots, q\}$. Inequality 4.2 implies:

$$\sum_{k_1, \ldots, k_n \in Q} \left(\frac{1}{2}\right)^{l_{k_1} + l_{k_2} + \ldots + l_{k_n}} \geq c^n. \tag{4.3}$$

Let $Y_1, Y_2, \ldots$ be a sequence of i.i.d. Bernoulli variables such that

$$P(Y_1 = 1) = 0.5, P(Y_1 = 0) = 0.5.$$

Let w be a binary word $w = w_1 w_2 \ldots w_r$. Let A_w designate the event that the sequence $Y_1 Y_2 \ldots$ starts with w that is

$$A_w := \{Y_1 = w_1, Y_2 = w_2, \ldots, Y_r = w_r\}.$$

Note that if $|w|$ designates the length of the word w, then

$$P(A_w) = \left(\frac{1}{2}\right)^{|w|}.$$

Thus,

$$\left(\frac{1}{2}\right)^{l_{k_1}+l_{k_2}+\ldots+l_{k_n}} = P(A_w)$$

where w is the binary word:

$$w = C(s_{k_1}s_{k_2}\ldots s_{k_n}) = C(s_{k_1})C(s_{k_2})\ldots C(s_{k_n}).$$

Let $A_{k_1,\ldots,k_n}$ denote the event A_w for $w = C(s_{k_1}s_{k_2}\ldots s_{k_n})$. With this notation, inequality 4.3 can be written as:

$$\sum_{k_1,\ldots,k_n\in Q} P\left(A_{k_1,\ldots,k_n}\right) \geq c^n.$$

Ordering the terms of the sum on the left side of the last inequality according to the length of $C(s_{k_1}s_{k_2}\ldots s_{k_n})$, we get:

$$\sum_t \sum_{k\in Q^n, |C(s)|=t} P\left(A_{k_1,\ldots,k_n}\right) \geq c^n. \tag{4.4}$$

where $k := (k_1,\ldots,k_n)$ and $s := s_{k_1}s_{k_2}\ldots s_{k_n}$. Let l denote the maximum length:

$$l := \max_{i\in Q} l_i.$$

Then, $|C(s)|$ can not be longer than $n\cdot l$. This implies that inequality 4.4 can be rewritten:

$$\sum_{t=1}^{nl} \sum_{k\in Q^n, |C(s)|=t} P\left(A_{k_1,\ldots,k_n}\right) \geq c^n. \tag{4.5}$$

Note that since the code C is uniquely decodable, we have that if s and z are different strings with n letters of the alphabet S such that $|C(s)| = |C(z)|$ then $C(s) \neq C(z)$. Thus, if s is different from z, then A_w and A_v can not hold at the same time, where $w := C(s)$ and $v := C(z)$. In this case, A_w and A_v are disjoint events. This implies that for fix t, the events $A_{k_1,\ldots,k_n}$ for which $|C(s_{k_1})\ldots C(s_{k_n})| = t$ are disjoint events. The probability of the union of disjoint events is equal to the sum of their probabilities. This gives:

$$\sum_{k\in Q^n, |C(s)|=t} P\left(A_{k_1,\ldots,k_n}\right) = P\left(\bigcup_{k\in Q^n, |C(s)|=t} A_{k_1,\ldots,k_n}\right). \tag{4.6}$$

Let A_t be the event:

$$A_t := \bigcup_{k\in Q^n, |C(s)|=t} A_{k_1,\ldots,k_n}.$$

(Note that A_t is the event that the binary string:

$$Y_1Y_2\ldots Y_t$$

is exactly equal to an encoded text with n symbols from S.) Using 4.6 with 4.5, we find:

$$\sum_{t=1}^{nl} P\left(A_t\right) \geq c^n. \tag{4.7}$$

The probability of an event is always smaller equal to one, and thus:

$$P(A_t) \leq 1.$$

Using this in 4.7, yields:

$$\sum_{t=1}^{nl} 1 = l \cdot n \geq c^n.$$

For large n however the last inequality does not hold, since c^n grows much faster than $l \cdot n$. This finishes the proof and shows that if C is uniquely decodable, the Kraft inequality must hold. ■

5 Entropy and the independent case

In this section, the letters of the text we want to encode are random and independent of each other. We also assume that all the letters in the text have the same probability distribution. Thus, $X_1, X_2, \ldots$ are random letters chosen from an alphabet S each having the same probability distribution and independent of each other. The text we wish to encode is denoted by $X := X_1 X_2 \ldots X_n$.
Let E be an event. If we are told that the event E has happened, we say that we received a quantity of information equal to

$$\log_2 \left(\frac{1}{P(E)} \right).$$

Let $R \subset S$ be a subset of letters from the alphabet S. If we are told that the letter X_i is in R, we receive a quantity

$$I(R) := \log_2 \left(\frac{1}{P(X_i \in R)} \right)$$

of information. For $s \in S$, let p_s designate the probability of letter s, that is $p_s = P(X_1 = s)$. When we are told that the i-th letter of our text X_i is equal to s, we thus receive an amount of information equal to

$$\log_2 \left(\frac{1}{p_s} \right).$$

On the long run, a proportion p_s of letters are equal to s in the text X. Hence the average amount of information receive per letter when we send the message $X = X_1 X_2, \ldots X_n$ is equal to:

$$\sum_{s \in S} p_s \log_2 \left(\frac{1}{p_s} \right).$$

The above quantity is called *entropy* of the random variable X_1.

Definition 5.1 *Let Z be a random variable taking on values from a set S. Then, the number:*

$$\sum_{s \in S} P(Z = s) \log_2 \left(\frac{1}{P(Z = s)} \right)$$

is called entropy of the random variable Z.

Next we show that the entropy of two independent random variables is the sum of their entropy.

Lemma 5.1 *Let X, Y be two independent random variables taking values in S. Let Z denote the random pair $Z = (X, Y)$ which takes values in $S \times S$. Then, the entropy $H(Z) = H(X, Y)$ of Z is equal to:*

$$H(X) + H(Y).$$

Proof. For $(x, y) \in S \times S$, let $p_{(x,y)}$ denote the probability $P(X = x, Y = y)$. Let $p_x := P(X = x)$ and let $p_y := P(Y = y)$. We have that

$$H(X, Y) = \sum_{(x,y) \in S \times S} p_{(x,y)} \log \left(\frac{1}{p_{(x,y)}} \right).$$

By independence $p_{(x,y)} = p_x \cdot p_y$. Hence:

$$H(X, Y) = \sum_{(x,y) \in S \times S} p_x p_y \log \left(\frac{1}{p_x p_y} \right) = \sum_{(x,y) \in S \times S} p_x p_y \left(\log \left(\frac{1}{p_x} \right) + \log \left(\frac{1}{p_y} \right) \right).$$

Using distributivity and regrouping the terms of the sum above in different order, we find:

$$H(X, Y) = \sum_{y \in S} p_y \left(\sum_{x \in S} p_x \log \left(\frac{1}{p_x} \right) \right) + \sum_{x \in S} p_x \left(\sum_{y \in S} p_y \log \left(\frac{1}{p_y} \right) \right) = H(X) \sum_{y \in S} p_y + H(Y) \sum_{x \in S} p_x. \tag{5.1}$$

Since probabilities always add up to one, we have that

$$\sum_{y \in S} p_y = \sum_{x \in S} p_x = 1,$$

which with 5.1 implies that

$$H(X, Y) = H(X) + H(Y).$$

■

For a word w, let $|w|$ denote the number of letters in w. Thus, for example: $|\text{heinrich}| = 8$.

Take as an example an alphabet consisting of 3 letters $S = \{a, b, c\}$. Assume that the probabilities of each letter are:

$$p_a = P(X_1 = a) = 30\%, p_b = P(X_1 = b) = 40\%, p_c = P(X_1 = c) = 30\%.$$

The goal is to find a uniquely decodable code that compresses the text $X = X_1 \dots X_n$ maximally. Let C denote a block code. Wanting to compress the text X maximally means that we want to minimize the average length per letter X_i. In other words, we want to minimize the quantity:

$$E[|C(X_1)|] = l_a 30\% + l_b 40\% + l_c 30\%$$

where l_a, l_b, l_c designate the lengths of the different code words:

$$l_a = |C(a)|, l_b = |C(b)|, l_c = |C(c)|.$$

The constrain is that C is uniquely decodable, which is equivalent to McMillan's inequality:

$$\left(\frac{1}{2}\right)^{l_a} + \left(\frac{1}{2}\right)^{l_b} + \left(\frac{1}{2}\right)^{l_c} \leq 1. \tag{5.2}$$

The optimization problem we are faced with is thus: find $x, y, z \geq 0$ minimizing:

$$f(x, y, z) := x \cdot 30\% + y \cdot 40\% + z \cdot 30\%$$

under the constrain

$$g(x, y, z) = \left(\frac{1}{2}\right)^{x} + \left(\frac{1}{2}\right)^{y} + \left(\frac{1}{2}\right)^{z} \leq 1.$$

To solve this problem we use the Lagrange method. Thus, we look for x, y, z and $\lambda \neq 0$ such that $\lambda \cdot grad(f) = grad(g)$. This, then gives:

$$\lambda \cdot (p_a, p_b, p_c) = (0.5^x, 0.5^y, 0.5^z)$$

which is equivalent to:

$$(x, y, z) = (-\log_2(\lambda p_a), -\log_2(\lambda p_b), -\log_2(\lambda p_c)).$$

If we want:

$$g(x, y, z) = 1,$$

we have to put $\lambda = 1$. Hence, we find the optimal solution

$$x = \log_2\left(\frac{1}{p_a}\right), y = \log_2\left(\frac{1}{p_b}\right), z = \log_2\left(\frac{1}{p_c}\right).$$

With these optimal values, we find that the minimum possible value for the objective function f under the constrain 5.2 is equal to:

$$p_a \log_2\left(\frac{1}{p_a}\right) + p_b \log_2\left(\frac{1}{p_b}\right) + p_c \log_2\left(\frac{1}{p_c}\right)$$

which is equal to the entropy $H(X_1)$ of X_1.
However these optimal values might not be feasible because the lengths l_a, l_b, l_c need to be integers while it is possible that the optimal values x, y, z are real numbers but not integers. This implies that $H(X_1)$ is a lower bound for the minimum possible expected length per symbol $E[|C(X_1)|]$. Thus,

$$H(X_1) \leq E[|C(X_1)|]$$

for any uniquely decodable block code C. On the other hand, we can always take for the lengths l_i the first integer larger than $\log_2(1/p_i)$. This changes the quantity $x \cdot p_a + y \cdot p_b + z \cdot p_c$ by at most one. Furthermore, the thus found integers still satisfy McMillan's inequality. This means that there exist a uniquely decodable block code C such that $E[|C(X_1)|] \leq H(X_1) + 1$. We have just proved Shannon's first theorem:

Theorem 5.1 *Let C be an instantaneous block code for a text $X = X_1 X_2 \dots X_n$ with independent letters with same distribution. Then,*

$$H(X_1) \leq E[|C(X_1)|].$$

Furthermore, if C denotes a block code with shortest expected code word length then:

$$H(X_1) \leq E[|C(X_1)|] \leq H(X_1) + 1.$$

Let us look at a practical example. Let us assume that the alphabet S consists of two letters $S = \{a, b\}$ and

$$p_a = 90\%, p_b = 10\%.$$

The best a uniquely decodable block code can achieve in the case is an expected length per symbol of 1. This, means no compression at all. Such a block code is for example given by:

$$C(a) = 0, C(b) = 1.$$

A text like *abba* would then be encoded as 0110. So the encoding keeps the same length as the original text and thus there is no compression. To improve this, we can encode an extension of S instead of encoding S with a block code. Take for example the second extension S^2 of S. This is simply the Cartesian product $S \times S$, that is the set consisting of all ordered pairs of elements of S. We find that $S^2 = S \times S$ consists of four letters:

$$S^2 = S \times S = \{C = aa, D = ab, E = ba, F = bb\}.$$

We proceed as follows: we find a block code for the alphabet S^2. We first rewrite the text with letters of S^2. And then we encode this using a block code for S^2. Define for example a block code C for S^2 in the following manner:

$$C(C) = C(aa) = 1, C(D) = C(ab) = 01, C(E) = C(ba) = 001, C(F) = C(bb) = 000.$$

This is an instantaneous code since no code word is prefix to another. with this code the text *abba* gives the letter DE which is encoded as: 01001.
Note that:

$$P(aa) = 0.9 \cdot 0.9 = 0.81, P(ab) = P(ba) = 0.9 \cdot 0.1 = 0.09, P(bb) = 0.1 \cdot 0.1 = 0.01.$$

Thus, the expected length of the C-encode message per symbol of S^2 is equal to:

$$1 \cdot 0.81 + 2 \cdot 0.09 + 3 \cdot 0.09 + 3 \cdot 0.01 = 1.29.$$

Each symbol of S^2 corresponds to two symbols of S. This implies that the expected length per symbol of S is equal to $1.29/2 = 0.645$. Hence by coding the second extension of S instead of using a block code of S we can compress the text by 35.5%. We could still improve this result by coding a higher extension, for example $S^3 = S \times S \times S$.
We can rewrite Shannon's first theorem for a block code of the k-th extension S^k. Assume that the length n of the original text $X = X_1X_2 \dots X_n$ is a multiple of k so that $n = mk$. We rewrite the text X by taking blocks of k letters together, so as to obtain symbols of S^k. Written, in this form, the text X can be seen as a sequence $X = Z_1Z_2 \dots Z_m$ of m independent letters Z_i from the alphabet S^k where:

$$\begin{aligned} Z_1 &:= X_1X_2 \dots X_k, \\ Z_2 &:= X_{k+1}X_{k+2} \dots X_{2k}, \\ Z_3 &:= X_{2k+1}X_{2k+2} \dots X_{3k}, \\ &\dots \end{aligned}$$

Let C denote a block code for S^k. C can also be used to encode X by putting:

$$C(X) := C(Z_1)C(Z_2) \dots C(Z_m).$$

Let C denote a uniquely decodable block code for S^k minimizing the expected length $E[|C(Z_1)|]$ among all such block codes. Applying Shannon's first theorem yields:

$$H(Z_1) \leq E[|C(Z_1)|] \leq H(Z_1) + 1. \tag{5.3}$$

Since, $Z_1 = X_1X_2 \dots X_k$ and since the random letters $X_1, X_2, \dots$ are all independent of each other and have same distribution, we find by lemma 5.1:

$$H(Z_1) = H(X_1) + H(X_2) + \dots + H(X_k) = k \cdot H(X_1).$$

Plugging this into 5.3 and dividing by k yields:

$$H(X_1) \leq \frac{E[|C(Z_1)|]}{k} \leq H(X_1) + \frac{1}{k}.$$

Note that that $E[|C(Z_1)|]/k$ is equal to the expected code length $E[|C(X)|]/n$ per symbol of S. This gives the second version of Shannon's first theorem:

Theorem 5.2 *Let C be a uniquely decodable block code for the extension S^k with shortest possible expected code-word length $E[|C(Z_1)|]$. (Shortest possible among the uniquely decodable block codes for the extension S^k). Then:*

$$H(X_1) \leq \frac{E[|C(X)|]}{n} \leq H(X_1) + \frac{1}{k}.$$

Here, $|C(X)|$ designates the length of the encoded text and n is the number of letters in the original text X.

Summary Let us summarize this section. This is about the case when the letters in the original text are independent of each other. There are two possibilities:

1. The first possibility is when for each letter $s \in S$, the probability $p_s = P(X_1 = s)$ can be written in the form:
$$p_s = \left(\frac{1}{2}\right)^{l_s} \tag{5.4}$$
where l_s is a natural number. In this case, there exists an instantaneous block code which achieves maximum compression. The average space per encoded letter is equal to $H(X_1)$ bits. The length of the code word $C(s)$ encoding letter s, is equal to the number l_s from equation 5.4.

2. The second case is when the probabilities of the letters from s are not all equal to natural number-powers of $1/2$. In the case, maximum compression can not be reached using a block code on the alphabet S. Instead one can use a block code on an extension of S. By choosing to encode a high enough extension, one can get as close as one want to the maximum possible compression. By encoding blocks of k letters of S at a time it is possible to get as close as $1/k$ to the rate of $H(X_1)$ bits per encoded letter.

6 Compact codes

In the previous section we saw how we can get as close as we want to optimal compression by encoding large enough blocks of letters at a time. In many practical cases it is however not possible to take very large blocks of letters. Consider for example the case where S is the ascii alphabet. This alphabet consists of about 100 symbols. How many exactly? Can somebody send me an E-mail about this. If we want to encode the fifth extension S^5 of S, we are dealing if all the sequence of five letters of S. There are thus about $|S|^5 = 100^5 = 10000000000$ symbols in S^5. If we wish to find a block code for the 5-th extension S^5 we need to find a code word for each of the 10^10 symbols of S^5. In this case, it is easier to to encode a smaller extension even if this means less compression. In this situation one wishes to find the best possible compression among all block codes for a given extension. This means that one tries to find a code which among all block codes for a certain extension minimizes the expected code word length.

The general problem is thus to find an optimal instantaneous block code for a given set of symbols S. (The set of symbols could be an extension itself.) By optimal, we mean minimizing the the average length of the encoded text per symbol of the original alphabet. This is the same as to search for a code C that minimize the expected code word length $E[|C(X_1)|]$.
Let S be an alphabet. Let $\mathcal{C}$ denote the set of all possible instantaneous block codes $K : S \to \cup_{i=1}^{\infty}\{0,1\}^i$. Assume that $X_1, X_2, X_3, \ldots$ is a sequence of i.i.d. random letters taking values in S. For $s \in S$, let p_s denote the probability of letter s, i.e.

$$p_s := P(X_1 = s).$$

Definition 6.1 *Let $C : S \to \cup_i\{0,1\}^i$ be a instantaneous block code. Then, C is called a compact code if it minimizes the expected length of the encoded code word $E[|C(X_1)|] = \sum_{s\in S}|C(s)|p_s$ among all instantaneous block codes for S.*
Thus, $C : S \to \cup_i\{0,1\}^i$ is a compact code iff for all instantaneous block code $K : S \to \cup_i\{0,1\}^i$ we have:

$$\sum_{s\in S}|C(s)|p_s \leq \sum_{s\in S}|K(s)|p_s.$$

Note that a code is compact with respect to a set of probabilities $p_s, s \in S$. If we change the probabilities on the random letters a compact code might no longer be compact.
Next, we present an algorithm to construct compact codes. Let us first give two practical examples. Consider a four letter alphabet $\{a, b, c, d, \}$ and the probabilities:

$$p_a = 40\%, p_b = 30\%, p_c = 20\%, p_d = 10\%.$$

Take the code C such that:

$$C(a) = 0, C(b) = 110, C(c) = 10, C(d) = 111.$$

This is obviously a instantaneous code, but could it be compact? At first glance one can see that C can not be compact. The code word for b is longer than the code word for c although b has larger probability than c. So by exchanging code words and giving b the code word 10 and c the codeword 110 one can obtain a new instantaneous code with strictly reduced expected code word length. Hence, C can not be optimal. It is thus necessary that the length of the code words decreases as the probability of the corresponding letters increase. Let l_s denote the length of the code word $C(s)$ of letter s. For C to be compact it is thus necessary that the following condition holds:

- For $s_1, s_2 \in S$, if $p_{s_1} < p_{s_2}$ then $l_{s_1} \geq l_{s_2}$.

Let us look at another example of an instantaneous block code C:

$$C(a) = 0, C(b) = 10, C(c) = 110, C(d) = 1111.$$

Here, the condition that the code-word-lengths decrease with increasing probability of the letters, is satisfied. However, one can immediately see that C is not compact. As a matter

of fact, the code word of the letter with smallest probability d could be reduced by one bit: Instead of taking 1111 for d, we could take 111. Thus C is not compact. The reason why we can remove the last bit of the code word of d and still get an instantaneous code is the following:
there is no other code word with same length as $C(d) = 1111$ and which is identical to $C(d)$ except on its last bit. Thus, another necessary condition for a code C to be compact is:

- The two letters with smallest probabilities must have code words of the same length which differ from each other only in there last bit.

Let us replace the code word for d by the code word 111 in the last example. For the new code which we obtain, see that the above condition is satisfied. The two letters with smallest probabilities are c and d. Their code words are with the new code 110 and 111. Assume now that the code C satisfies the last condition. For example take:

$$C(a) = 0, C(b) = 10, C(c) = 110, C(d) = 111.$$

We are going to reduce the problem of finding a compact code for an alphabet with k letters to the problem of finding a compact code for an alphabet with $k-1$ letters. For this we take the two letters with smallest probability and "melt" them into one letter. The new resulting letter, has probability equal to the sum of the probabilities of the melted letters. In the example above, we replace the alphabet $\{a, b, c, d\}$ by the 3-letter alphabet: $S^* = \{a, b, \gamma\}$, where

$$p_\gamma := p_c + p_d.$$

As we will show, it turns out that we can find a compact code C for S in the following way:
1) Find a compact code K for the reduced alphabet S^*.
2) For the letters s which appear in both alphabets S and S^*, let the code word $C(s)$ of s be the same as with K, that is $C(s) := K(s)$.
3) Let r, t be the two letters in S with smallest probability. These are the letters which got "melted" down into one letter γ. Take for them the code word of $K(\gamma)$ and add a 0, resp. a 1 to get the code word for r and for t.
In the previous example, if γ has code word 11 then take 111 as code word for r and 110 as code word for t.

Lemma 6.1 *Let r, t designate the two letters with smallest probability in the alphabet S. Let S^* designate the reduced alphabet where the letters r, t were replaced by a letter γ with probability $p_\gamma := p_r + p_t$. Let K be a compact code for S^*. Let C be the code for S obtained by adding a suffix to $K(r)$ and $K(t)$:*

- $C(s) := K(s)$ *for all* $s \in S - \{r, t\}$
- $C(r) := K(r)0$
- $C(t) := K(t)1$

Then, C is a compact code for S.

Proof. Suppose on the contrary that C is not a compact code. Let $D : S \to \cup_i \{0,1\}$ be a compact code. Then, since C is not compact:

$$\sum_{s \in S} |D(s)| p_s < \sum_{s \in S} |C(s)| p_s \tag{6.1}$$

Since D is compact, it satisfies the property that the two letters of S with smallest probability must have code words which are equal except for their last bit. Define thus a code D^* on the reduced alphabet S^* in the following manner: for all $s \in S - \{r, t\}$ let $D^*(s) := D(s)$.
Let $D^*(\gamma)$ be the code word obtained from $D(r)$ by removing the last bit. We find:

$$\sum_{s \in S} |D(s)| p_s = \left(\sum_{s \in S^*} |D^*(s)| p_s \right) + p_r + p_t \tag{6.2}$$

Reducing the code C in a similar way gives the code K. Thus,

$$\sum_{s \in S} |C(s)| p_s = \left(\sum_{s \in S^*} |K(s)| p_s \right) + p_r + p_t \tag{6.3}$$

Plugging 6.2 and 6.3 into 6.1 yields:

$$\sum_{s \in S^*} |D^*(s)| p_s < \sum_{s \in S^*} |K(s)| p_s$$

It is easy to see that D^* is an instantaneous block code since D is (prefix-freeness). Hence the last inequality contradicts the fact that K is a compact code. Thus, C must be a block code. ■

7 Expected length of encoded text and entropy

Let Z_1 be a random variables which can take values in a set Ω. For $z \in \Omega$, let p_z designate the probability that Z_1 takes on value z, i.e. $p_z = P(Z = z)$. Then, the expectation is $E[Z_1]$ is defined to be equal to:

$$E[Z_1] := \sum_{z \in \Omega} z \cdot p_z.$$

Let us first recall the Law of Large Numbers from probability theory:

Theorem 7.1 *Let $Z_1, Z_2, \ldots$ be i.i.d. (= independent and identically distributed) random variables. Then, as n goes to infinity we have that:*

$$\frac{Z_1 + Z_2 + \ldots + Z_n}{n} \to E[X_1]. \tag{7.1}$$

Hence, the average of $Z_1 + \ldots + Z_n/n$ converges to the expectation $E[Z_1]$ as n goes to infinity. In a less mathematical formulation: for large n, expression 7.1 is very close to the value $E[Z_1]$.

Let us look at an example. Let Z_1 be a four sided die with sides $1, 2, 3$ and 4. Assume the die is biased and that the probabilities of each side is given by:

$$P(Z_1 = 1) = 10\%, P(Z_1 = 2) = 40\%, P(Z_1 = 3) = 10\%, P(Z_1 = 4) = 40\%.$$

The expectation $E[Z_1]$ of Z_1 is obtained by multiplying the probabilities by the corresponding values and summing over the different values:

$$E[Z_1] = 1 \cdot 0.1 + 2 \cdot 0.4 + 3 \cdot 0.1 + 4 \cdot 0.4 = 2.8$$

In this case, $(Z_1+Z_2+\ldots+Z_n)/n$ designates the value which we obtain when we throw the die n-times independently and then take the average. Hence, in the case of this example, the Law of Large Numbers tells us, that for large n the average $(Z_1 + Z_2 + \ldots + Z_n)/n$ is close to 2.8.
Assume that we are in the case of a random text X consisting of n i.i.d. letters $X_1, X_2, \ldots, X_n$ so that $X = X_1X_2 \ldots X_n$. Let C be a block code. Then the length of the encoded text $C(X) = C(X_1)C(X_2) \ldots C(X_n)$ is equal to:

$$|C(X)| = |C(X_1)| + |C(X_2)| + \ldots + |C(X_n)|.$$

For example: assume we have a three letter alphabet $S = \{a, b, c\}$ and a code C:

$$C(a) = 0, C(b) = 10, C(c) = 11.$$

Assume that $n = 4$, that is the text X has length 4. We thus throw a die 4 times to get four random letters. For example we could get $X_1 = a, X_2 = a, X_3 = b, X_4 = c$, and hence the original text would be equal to:

$$X = aabc.$$

Encoding X yields: $C(X) = C(aabc) = C(a)C(a)C(b)C(c) = 001011$. The length $|C(X)|$ of the encoded text is thus

$$|C(X)| = |C(a)| + |C(a)| + |C(b)| + |C(c)| = 1 + 1 + 2 + 2 = 6.$$

Let Z_i denote the length of the i-th encoded letter X_i of text X. Thus $Z_i = |C(X_i)|$. With this notation the length $|C(X)|$ of the encoded text equals:

$$|C(X)| = Z_1 + Z_2 + \ldots + Z_n.$$

Since, the X_i's are independent and have same probabilities so do the Z_i's which are function of the X_i's. We can thus apply the Law of Large Numbers to the Z_i's. We find that for large n, the average $(Z_1 + \ldots + Z_n)/n$ is approximately equal to $E[Z_i]$. Thus,

$$Z_1 + \ldots + Z_n \approx n \cdot E[Z_1] = n \cdot E[|C(X_1)|].$$

The conclusion is:

The length of the encoded text is approximately equal to the expected length $E[|C(X_1)|]$ of the code word times the number of letters, as soon as the length of the original text is not too small.

Probability theory (Central Limit Theorem) tells us that this approximation is very good: the difference between $Z_1 + \ldots + Z_n$ and $n \cdot E[Z_1]$ is with high probability no more than order $o(n^{\frac{1}{2}})$. Most of the time, we consider texts for which n, the number of letter is large so that $o(n^{\frac{1}{2}})$ is negligible in comparison to n. If the text X for example has length $n = 10000$, then $n^{\frac{1}{2}} = 100$. So, the encoded text length is proportional to 10000 plus minus something of the order of 100: $C(X) = 10000 \ldots E[|C(X_1)| + -o(100)$.

8 Convexity of entropy

Assume that a random letter X_1 has probabilities given by:

$$P(X_1 = a) = 20\%, P(X_1 = b) = 30\%, P(X_1 = c) = 50\%.$$

Note that we can summaries the situation in a three dimensional vector:

$$(0.2, 0.3, 0.5).$$

We write $\mathcal{L}(X_1)$ for the probability distribution of X_1. The vector specifying all the probabilities of the random letter X_1 is called the *probability distribution* of X_1. If a random letter takes values in an alphabet S with q letters, then the probability distribution of that letter can be seen as a q-dimensional vector. For example, if X_1 is a fair coin which is equal to one with probability p and equal to 0 otherwise, we have that

$$\mathcal{L}(X_1) = (p, p - 1).$$

Assume that we have two four sided dice X_1 and Y_1 with different probabilities:

$$P(X_1 = a) = 20\%, P(X_1 = b) = 30\%, P(X_1 = c) = 30\%, P(X_1 = d) = 20\%$$

and

$$P(Y_1 = a) = 25\%, P(Y_1 = b) = 25\%, P(Y_1 = c) = 25\%, P(Y_1 = d) = 25\%.$$

Assume that we have a coin where side 1 has probability p and where 0 has probability $1 - p$. We can now use the coin and the two dice to generate a new random variable: we flip the coin and then decide which coin to use based on the side of the coin. With the chosen die we generate a random letter. The letter thus obtained can be denoted by Z_1. The probabilities of Z_1 are given by

$$P(Z_1 = a) = 0.2p + 0.25(1 - p)\ ,\ P(Z_1 = b) = 0.3p + 0.25(1 - p),$$
$$P(Z_1 = c) = 0.3p + 0.25(1 - p)\ ,\ P(Z_1 = d) = 0.2p + 0.25(1 - p).$$

In vector notation, we find:

$$\mathcal{L}(Z_1) = p\ (0.2, 0.3, 0.3, 0.2) + (1-p)\ (0.25, 0.25, 0.25, 0.25) = p\mathcal{L}(X_1) + (1-p)\mathcal{L}(Y_1).$$

The distribution of Z_1 is called a *mixture* of the distributions of X_1 and Y_1. Thus, in this case $\mathcal{L}(Z_1)$ is a linear combination of $\mathcal{L}(X_1)$ and of $\mathcal{L}(Y_1)$. Note that the entropy of X_1 depends only on the "probabilities", that is of the entries in the vector $\mathcal{L}(X_1)$. Hence, the entropy $H(X_1)$ can be seen as a function of $\mathcal{L}(X_1)$. Thus, $H(X_1) = H(\mathcal{L}(X_1))$. Next we prove that entropy is a concave function.

Lemma 8.1 *Let S designate an alphabet with q symbols. Let X_1 and Y_1 be two random letters taking values in S. Let $p \in [0,1]$. Then:*

$$H\left(p\mathcal{L}(X_1) + (1-p)\mathcal{L}(Y_1)\right) \geq pH\left(\mathcal{L}(X_1)\right) + (1-p)H\left(\mathcal{L}(Y_1)\right).$$

In other words the entropy function $H(.)$ is concave.

Proof. Take for example a three letter alphabet $S = \{a, b, c\}$. Then, the entropy function H is

$$H : (p_a, p_b, p_c) \to p_a \log_2\left(\frac{1}{p_a}\right) + p_b \log_2\left(\frac{1}{p_b}\right) + p_c \log_2\left(\frac{1}{p_c}\right).$$

In other words,

$$H : (x, y, z) \mapsto x \log_2\left(\frac{1}{x}\right) + y \log_2\left(\frac{1}{y}\right) + z \log_2\left(\frac{1}{z}\right)$$

where we only consider non-negative values for x, y and z.
Let us look at the function $f(.)$:

$$f : x \mapsto x \log_2\left(\frac{1}{x}\right)$$

for $x \geq 0$. Taking the second derivative, we find:

$$\frac{d^2 f}{d^2 x} = \frac{d(\log(1/x) - 1)}{dx} = -\frac{1}{x}.$$

For $x \geq 0$ we have that $-(1/x) \leq 0$. Thus, the function f is concave. It follows that H is a sum of three concave functions: $x \log_2(1/x)$, $y \log_2(1/y)$ and $z \log_2(1/z)$. The sum of concave functions is again concave. Thus, $H(.)$ is concave ■

9 Entropy of joint variables

Theorem 9.1 *Let X_1 and Y_1 be two random variables. Then*

$$H(X_1, Y_1) \leq H(X_1) + H(Y_1)$$

with equality iff X_1 and Y_1 are independent.

Before giving a proof let us see an example. Let X_1 and Y_1 be two random letters from an alphabet $S = \{a, b, c\}$. Imagine the joint probability of X and Y is given by:

$$\begin{pmatrix} p_{a,a} & p_{a,b} & p_{a,c} \\ p_{b,a} & p_{b,b} & p_{b,c} \\ p_{c,a} & p_{c,b} & p_{c,c} \end{pmatrix} = \begin{pmatrix} 0.1 & 0.05 & 0.05 \\ 0.2 & 0.05 & 0.01 \\ 0.04 & 0.3 & 0.2 \end{pmatrix}$$

where $p_{a,a}$ designates the probability $P(X_1 = a, Y_1 = a)$ and $p_{a,b}$ designates the probability $P(X_1 = a, Y_1 = b)$ and ...
Then, (X_1, Y_1) is an ordered random pair, or call it a 2-dimensional random vector. Denote the random pair (X_1, Y_1) by Z_1. What is the entropy of Z_1? To calculate the entropy of a random object, we need to take the probability of each possible outcome and multiply it with the logarithm of the inverse of its probability. Then we sum over all possible states. For Z_1 there are in this example 9 possible states. Thus we find:

$$H(Z_1) = H(X_1, Y_1) = \sum_{r,s \in S} p_{r,s} \log\left(\frac{1}{p_{r,s}}\right).$$

In our example, thus:

$$\begin{aligned} H(Z_1) =& p_{a,a} \log\left(\frac{1}{p_{a,a}}\right) + p_{a,b} \log\left(\frac{1}{p_{a,b}}\right) + p_{a,c} \log\left(\frac{1}{p_{a,c}}\right) + p_{b,a} \log\left(\frac{1}{p_{b,a}}\right) + \\ &+ p_{b,b} \log\left(\frac{1}{p_{b,b}}\right) + p_{b,c} \log\left(\frac{1}{p_{b,c}}\right) + p_{c,a} \log\left(\frac{1}{p_{c,a}}\right) + p_{c,b} \log\left(\frac{1}{p_{c,b}}\right) + p_{c,c} \log\left(\frac{1}{p_{c,c}}\right). \end{aligned}$$

Thus:

$$H(Z_1) = 0.1 \log(1/0.1) + 0.05 \log(1/0.05) + \ldots + 0.3 \log(1/0.3) + 0.2 \log(1/0.2).$$

Let us calculate next the the entropies of X_1 and Y_1. For this we need to find what the "probabilities" for X_1 and Y_1 are.
We have that

$$P(X_1 = a) = p_{a,a} + p_{a,b} + p_{a,c} = 0.1 + 0.05 + 0.05 = 0.2$$

and

$$P(X_1 = b) = p_{b,a} + p_{b,b} + p_{b,c} = 0.2 + 0.05 + 0.01 = 0.26$$

and

$$P(X_1 = c) = p_{c,a} + p_{c,b} + p_{c,c} = 0.04 + 0.3 + 0.2 = 0.54$$

Thus, the entropy $H(X_1)$ of X_1 is equal to:

$$H(X_1) = 0.2 \log_2(1/0,2) + 0.05 \log_2(1/0.05) + 0.05 \log_2(1/0.05).$$

In a similar way we could calculate $H(Y_1)$. Let us next present a very intuitive proof of lemma 9.1. Let for this $Z_1, Z_2 = (X_2, Y_2), Z_3 = (X_3, Y_3), \ldots$ be a sequence of independent

copies of the random pair $Z_1 = (X_1, Y_1)$. Let Z be a text of n independent random letters, so that:

$$Z := Z_1 Z_2 \dots Z_n.$$

Assume that n is large. We saw that the minimum space to store a random text of i.i.d. letters is about n times the entropy of one random letter. (Here n is the length of the text.) By maximum compression without information-loss the text Z requires thus for storage:

$$H(Z_1)n$$

bits of space. Now we could also store it by taking the two text X and Y separately and compressing each of them maximally. (Here X designates the random text $X := X_1 X_2 \dots X_n$ and $Y := Y_1 Y_2 \dots Y_n$.) For text X, we need about

$$H(X_1)n$$

space and for Y, we would need about

$$H(Y_1)n$$

bits of space. Thus the total space needed if we compress X and Y separately and store them in two files is $H(X)n + H(Y)n$. Clearly if we store X in a manner to not lose any information and same thing for Y then from the two files we will be able to reconstruct the text Z. Thus, there exist a compression of Z with no information-loss which requires about $H(X_1)n + H(Y_1)n$. The optimal compression of Z can not be worse, than compressing X and Y separately. (Otherwise it would not be the optimal compression.) Thus, $H(Z_1)n$ is about less or equal to $H(X_1)n + H(Y_1)n$. Hence, $H(Z_1)n \leq H(X_1)n + H(Y_1)n$ from which it follows that

$$H(Z_1) \leq H(X_1) + H(Y_1).$$

Let us next give more mathematical proof of theorem 9.1:
Proof. Still to be written down. ■

Lemma 9.1 *Let X_1 be a Bernoulli variable with parameter p. That is $P(X_1 = 1) = p$ and $P(X_1 = 0) = 1 - p$. Then*

$$H(X_1) = p \log\left(\frac{1}{p}\right) + (1-p)\log\left(\frac{1}{1-p}\right)$$

Definition 9.1 *For $p \in [0, 1]$ we define the entropy function $H(p)$ at p in the following way:*

$$H(p) = p \log\left(\frac{1}{p}\right) + (1-p)\log\left(\frac{1}{1-p}\right)$$

At $p = 0$ and $p = 1$ define $H(p)$ by continuity to be equal to 0.

Lemma 9.2 *The function $H(p)$ reaches a maximum at $p = 1/2$.*

Lemma 9.3 *Let S be an alphabet with q symbols in it. Let X_1 be a random letter taking values in S. Then, $H(X_1)$ is maximum when each letter has same probability. When all the Q letters in the alphabet S have same probability, then $H(X_1) = \log(q)$.*

The last lemma can also be phrased in the following way: the function

$$f : (p_1, p_2, \ldots, p_q) \mapsto f(p_1, p_2, \ldots, p_q) = p_1 \log(1/p_1) + \ldots + p_q \log(1/p_q)$$

under the constrain

$$p_1, p_2, \ldots, p_q \geq 0$$

and

$$p_1 + \ldots + p_q = 1$$

is maximized when

$$p_1 = p_2 = \ldots = p_q = \frac{1}{q}.$$

10 Markov sources

In many cases, the assumption that letters in a text are independent of each other is not realistic. For example, the probability of encountering a e in a English text might be 10%. So if the letters where independent, the probability to encounter a string *eee* would have probability 0.1%. However, the string *eee* never appears in English. This shows that in an English text letters are not independent of each. Another example is: take the letters k and q. Both have probabilities to appear in an English text different from zero. However, the string qk never appears in English.
One of the main models used to approximate many real life situation is called *Markov chain*. Let $X = X_1 X_2 \ldots X_n$ designate a random text consisting of random letters $X_1, X_2, \ldots$. We say that the sequence of random letters

$$X_1, X_2, X_3, \ldots$$

has the *Markov property*, if what each letter is equal to depends only on the letter right before. By this we mean the following: assume that the random letters $X_1, X_2, \ldots$ are from an alphabet $X = \{a, b, c\}$. Then, if we have the Markov property, we can calculate the probability of what letter X_j is going to be, if we only know letter X_{j-1}. In the Markov case, for example if we know say that $X_1 = a, X_2 = b, X_3 = c$ and want to calculate what the probability $P(X_4 = a | X_1 = a, X_2 = b, X_3 = c)$ is, then we only need to know that $X_3 = c$. In other words, the conditional probability

$$P(X_4 = a | X_1 = a, X_2 = b, X_3 = c)$$

depends only on X_3. Thus, with the Markov property we have that:

$$P(X_4 = a | X_1 = a, X_2 = b, X_3 = c) = P(X_4 = a | X_3 = c).$$

Definition 10.1 *We say the sequence of random variables $X_1, X_2, \ldots$ has the Markov property iff for all (non-random) sequence $s_1, s_2, \ldots, s_{j+1}$ we have that*

$$P(X_{j+1} = s_{j+1} | X_1 = s_1, X_2 = s_2, \ldots, X_j = s_j) = P(X_{j+1} = s_{j+1} | X_j = s_j).$$

The probability to go from one letter r to the next s is called the *transition probability from state r to state s*. We are mainly interested in transition probabilities which don't change but remain fix. A Markov process which keeps always the same transition probabilities is called a *Markov chain*.

Definition 10.2 *We call the random sequence $X_1, X_2, \ldots$ a Markov chain if it has the Markov property and if for all $r, s \in S$, the transition probability*

$$P(X_{j+1} = s | X_j = r)$$

does not depend on j. The probability

$$P(X_{j+1} = s | X_j = r)$$

is called transition probability from state r into state s and is denoted by $p_{r \to s}$ or also by p_{rs}.

We usually write down the transition probabilities in form of a matrix. This matrix is then called *transition matrix*. Imagine for example a 2 letter alphabet $S = \{a, b, \}$ and the following transition matrix:

$$P = \begin{pmatrix} p_{aa} & p_{ab} \\ p_{ba} & p_{bb} \end{pmatrix} = \begin{pmatrix} 0.2 & 0.8 \\ 0.3 & 0.7 \end{pmatrix}$$

In this case we have:
after an a we have again an a with 20% probability and a b with 80% probability, after a b, we have with 30% probability an a and with 70% a b.
On important property of the i.i.d. case: all the letters $X_1, X_2, X_3, \ldots$ have the same probabilities. Thus, we have that

$$P(X_1 = a) = P(X_2 = a) = P(X_3 = a) = \ldots \tag{10.1}$$

and

$$P(X_1 = b) = P(X_2 = b) = P(X_3 = b) = \ldots \tag{10.2}$$

This property is not shared by Markov chain. However, the Markov chains we consider here (the non-cyclic irreducible ones) have the property that they quickly converge into a "stationary regime" where 10.1 and 10.2 approximately hold. When for all $s \in S$,

$$P(X_1 = s) = P(X_2 = s) = P(X_3 = s) = \ldots \tag{10.3}$$

we way that the Markov chain is in a *stationary state*. After a while most Markov chains converge into a stationary state. Since we are only interested in rather long texts in

information theory, we can always assume that our Markov texts are in a stationary state. This stationary probabilities for different $s \in S$ will be denoted by $\pi(s)$ and can be calculated by solving by the equation:

$$\begin{pmatrix} \pi(a) & \pi(b) \end{pmatrix} = \begin{pmatrix} \pi(a) & \pi(b) \end{pmatrix} \cdot \begin{pmatrix} p_{aa} & p_{ab} \\ p_{ba} & p_{bb} \end{pmatrix}$$

and $\sum_{s \in S} \pi(s)$. The stationary probability $\pi(s)$ of symbol s, represents the long term frequency of s in our Markov text X. It is also approximately is equal to the probability that the j-th letter X_j of our text is a s, provided j is not too small. Let us next look at what is the best way to encode a Markov text. Assume that we have a 3 letter alphabet $S = \{a, b, c\}$ and a transition Matrix:

$$\begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} = \begin{pmatrix} p_{aa} & p_{ab} & p_{ac} \\ p_{ba} & p_{bb} & p_{bc} \\ p_{ca} & p_{cb} & p_{cc} \end{pmatrix}.$$

In this case, if a letter is preceded by an a, then the probability that this letter is equal to :
an a is $1/2$, a b is $1/4$, a c is a $1/4$.// On the other hand, if a letter is preceded by a b, then the probability that this letter is equal to :
an a is $1/4$, a b is $1/2$, a c is a $1/4$.// Finally, if a letter is preceded by a c, then the probability that this letter is equal to :
an a is $1/4$, a b is $1/4$, a c is a $1/2$.// for each of these three cases another coding scheme is optimal:
1) In the first case, that is when the letter is preceded by an a, an optimal code D is given by:

$$D(a) = 0, D(b) = 10, D(c) = 11.$$

2) In the second case, when a letter is preceded by a b, an optimal code F is given by:

$$F(a) = 10, F(b) = 0, F(c) = 11.$$

3) If a letter is preceded by a c an optimal code G is :

$$G(a) = 10, G(b) = 11, G(c) = 0.$$

Thus, to achieve optimal compression with a Markov text we apply to each letter a different code depending on the preceding letter.
Let us illustrate this by an example of a text we encode using the above described method. Let the text to encode be equal to $X = X_1X_2X_3X_4X_5 = aabca$.
For the first letter X_1, we make the convention that it is preceded by an a. So we use code D to encode it and find $D(X_1) = D(a) = 0$.
The second letter X_2 of the text X is preceded by an a. So we use code D to encode it. We find that $D(X_2) = D(a) = 0$.
The third letter X_3 is proceeded by an a. We hence use code D. We get $D(X_3) = D(b) =$

10.
The fourth letter is preceded by a b. Thus, we use code F. We find $F(X_4) = F(c) = 11$. In a similar, way we find that the last letter X_5 needs to be encoded with code G. We find $G(X_5) = G(a) = 10$.
Eventually, when we encode the text *aabca* using this method, we get:

$$D(a)D(a)D(b)F(c)G(a) = 00101110.$$

Let us look at another example: assume the alphabet we consider is $S = \{a, b\}$ and let the transition matrix of the Markov chain is given by

$$P = \begin{pmatrix} 0.25 & 0.75 \\ 0.75 & 0.25 \end{pmatrix} = \begin{pmatrix} P_{a\to a} & P_{a\to b} \\ P_{b\to a} & P_{b\to a}. \end{pmatrix}$$

Here, a letter which is preceded by an a is equal to an a with 25% probability and to a b with 75% probability. However, "probability" 0.75 is not an integer power of 1/2. Thus, optimal compression can not be reached if we encode each letter at a time. Instead we need to take large blocks of such letters and encode them together. To do this we need to separate the letters which are proceeded by an a from those which are preceded by a b. Let us next give a few useful notations and definitions:

- Let N^a designate the total number of a's in the text $X = X_1X_2 \dots X_n$.
- Let Y_i designate the i-th letter in the text X which is preceded by an a for $i \leq N^a$.
- Let Y designate the text we obtain if we take only the letters which are preceded by an a. Hence:
$$Y = Y_1Y_2 \dots Y_{N^a}.$$
- Let N^b designate the total number of b's in the text X.
- Let Z_i designate the i-th letter of X which is preceded by a b for all $i \leq N^b$.
- Let $Z = Z_1Z_2 \dots Z_{N^b}$ designate the subsequence of letter from X, which we obtain if we take only the letters which are preceded by a b.

The most important facts about the "subtexts" Y and Z are:

- Y and Z are independent of each other.
- If we are given Y and Z, we can "reconstruct" X. (For this we always make the assumption that the first letter X_1 is a letter preceded by an a.) Hence, Y and Z uniquely determine X and thus contain the full information contained in X.

- The letters $Y_1, Y_2, \ldots$ are i.i.d. with

$$P(Y_i = a) = p_{a\to a}, P(Y_i = b) = p_{a\to b}.$$

- The letters $Z_1, Z_2, \ldots$ are i.i.d. with

$$P(Z_i = a) = p_{b\to a}, P(Z_i = b) = p_{b\to b}.$$

This solves the question of what minimum space is needed to store a Markov text X without loss of information. Since the information contained in X is equivalent to the information contained in the two text Y and Z, storing X is the same as storing Y and storing Z. Since Y and Z are independent of each other, we can not use one of them to store the other on less space. We have to encode Y and Z separately. The minimum amount of space needed to store X is equal to the space needed for Y plus the space needed for Z. But the texts Y and Z are both i.i.d. texts. Hence, their maximal compression is given by their entropy.
The minimal space for Y, is approximately $H(Y_i)$ times the number of letters in text Y. Hence, the space needed to store Y is equal to $H(Y_i)N^a$. However, N^a is approximately equal to $\pi(a)n$. Hence, the minimum amount of space needed to store Y is approximately equal to $H(Y_i)\pi(a)n$.
In a similar way, we find that the minimal space needed to store Z is approximately $H(Z_i)\pi(b)n$. Hence the total space used to store X (that is to store Y and Z separately) is approximately:

$$H(Y_i)\pi(a)n + H(Z_i)\pi(b)n.$$

This gives an amount of space of

$$H(Y_i)\pi(a) + H(Z_i)\pi(b) \tag{10.4}$$

per symbol of the text X. Expression 10.4 is called *entropy rate* of the Markov text X. We have that

$$H(Y_i) = p_{aa}\log_2\left(\frac{1}{p_{aa}}\right) + p_{ab}\log_2\left(\frac{1}{p_{ab}}\right)$$

since Y_i is equal to a with probability p_{aa} and is equal to b with probability p_{ab}. Similarly:

$$H(Z_i) = p_{ba}\log_2\left(\frac{1}{p_{ba}}\right) + p_{bb}\log_2\left(\frac{1}{p_{bb}}\right)$$

since Z_i is equal to a with probability p_{ba} and is equal to b with probability p_{bb}.

Important: For a Markov process which is not i.i.d. the entropy rate is different from the entropy $H(X_i)$ of a letter of X.
Let us explain how to encode a Markov text and reach optimal compression (or close to it).

1. Take the random text X and separate the letters which are preceded by an a from those which are preceded by a b. After separation you obtain the two text of i.i.d. letters Y and Z.

2. Chose an optimal (or close to optimal) code D to encode Y.

3. Chose an optimal (or close to optimal) code F for Z. Encode Z using F.

Of course this optimal method to encode a Markov chain works also in the general case, when the alphabet contains more than two letters.

Definition 10.3 *Let $X_1, X_2, \ldots$ be a Markov chain of letters from an alphabet S. We define the entropy rate of the Markov chain by*

$$\sum_{s \in S} H(X_i^s)\pi(s)$$

where

$$X_i^s$$

designates the i-th letter in X which is preceded by a s. Hence:

$$H(X_i^s) := \sum_{r \in S} P(X_{j+1} = r | X_j = s) \log_2 \left(\frac{1}{P(X_{j+1} = r | X_j = s)} \right) = \sum_{r \in S} p_{sr} \log_2 \left(\frac{1}{p_{sr}} \right).$$

We can also rewrite the entropy rate of a Markov chain in the following way:

$$\sum_{s,r} \pi(s) p_{sr} \log_2 \left(\frac{1}{p_{sr}} \right) = \sum_{s,r} \pi(s,r) \log_2 \left(\frac{1}{p_{sr}} \right)$$

where $\pi(s,r)$ designates the probability $\pi(s)p_{sr}$ of the string sr under the stationary distribution.

Let us give a theorem.

Theorem 10.1 *Let $X_1, X_2, \ldots$ be a Markov chain of letters from an alphabet S. Then, the entropy rate of the Markov chain is equal to the limit:*

$$\lim_{n \to \infty} \frac{H(X_1, X_2, X_3, \ldots, X_n)}{n}$$

Let us look at yet another approach to compressing Markov texts. We could encode letters or group of letters acting as if they where independent. For example we could find a block code C for the above described Markov chain. How well would that code do? The length of the encoded text is:

$$|C(a)|N^a + |C(b)|N^b$$

where

- N^a is the number of a's in the text X,
- N^b is the number of a's in the text X,

- $|C(a)|$ designates the length of the code word for a.
- $|C(b)|$ designates the length of the code word for b.

We saw that N^a is approximately equal to $\pi(a)n$. Similarly, N^b is approximately equal to $\pi(b)n$. Thus, the space needed to store the text X if we use the block code C to encode it, is approximately:

$$|C(a)|\pi(a)n + |C(b)|\pi(b)n$$

or

$$|C(a)|\pi(a) + |C(b)|\pi(b)$$

per letter of the original alphabet. This is the same space needed as if the letters would be independent and with probabilities $\pi(a)$ and $\pi(b)$.
Take now blocks of k-letters of the Markov chain at a time and encode them using a block code for the k-th extension S^k. Use the stationary distribution of those groups of letters to find a good code.
For example if you consider the 3rd extension. The probability of an *aba* in the stationary regime, is going to be equal to: the probability to have an a in the stationary regime, times the probability to go from a to b, times the probability to go from b to a. Thus, under stationary distribution the probability to have an *aba* is

$$\pi(a) \cdot p_{a\to b} p_{b\to a}.$$

We can apply Shannon's theorem for a block coding of Markov chain and find:

Theorem 10.2 *Let C be a block code on the k-th extension for a Markov chain. Assume that among such codes, C minimizes the expected length of the encoded text. Then:*

$$\frac{H(X_1, X_2, \ldots, X_k)}{k} \leq \frac{E[|C(X)|]}{n} \leq \frac{H(X_1, X_2, \ldots, X_k)}{k} + \frac{1}{k}$$

where the Markov chain $X_1, X_2, \ldots$ is taken in the stationary regime.

Assume that $X_1, X_2, \ldots$ is a Markov chain which is in its stationary regime. (If it is not, when we let the Markov chain go for a while it gets very quickly very close to the stationary regime anyhow. So the first letters which might not be in the stationary regime can be discarded.) For the stationary regime we have that all the probabilities $P(X_1 = a), P(X_2 = a), ...$ are equal to each other and are equal to $\pi(a)$.
similarly in the stationary regime, all the probabilities $P(X_1 = b), P(X_2 = b), ...$ are equal to each other and are equal to $\pi(b)$.
Why do on the long run the frequencies of a's and b's get close to the probabilities $\pi(a)$ and $\pi(b)$ just like in the i.i.d. case? Simple divide the letters $X_1, X_2, \ldots$ into r groups of letter which are r apart from each other. For example for $r = 10$, the first group will be

$$X_{10}, X_{20}, \ldots$$

the second group will be:

$$X_1, X_{11}, X_{21}, X_{31} \ldots$$

the third group will be

$$X_2, X_{12}, X_{22}, X_{32} \ldots .$$

By choosing r big enough, the letters in each group are very close to being independent of each other. So, for each group, we are very close to getting a proportion of a's, resp.b's equal to the probabilities $\pi(a)$ and $\pi(b)$. If in each group the proportion of a's and b's is very close to $\pi(a)$ and $\pi(b)$, then the overall proportion of a's and b's in X is also very close to $\pi(a)$ and $\pi(b)$. There exist also processes (i.e. sequences of random symbols) which are stationary but not necessarily Markov. Let us give a definition.

Definition 10.4 *Let $X_1, X_2, \ldots$ be a sequence of random symbols from an alphabet S. We say that the sequence $X_1, X_2, \ldots$ is* stationary *iff for all $k \in \mathbb{N}$ and all finite sequences of (non-random) symbols:*

$$s_1, s_2, \ldots, s_k$$

we have that:

$$P(X_1 = s_1, X_2 = s_2, \ldots, X_k = s_k) = P(X_{t+1} = s_1, X_{t+2} = s_2, \ldots, X_{t+k} = s_k)$$

for all $t \in \mathbb{N}$.

Note that a Markov Chain is in general not a stationary sequence of random symbols. However, if we let a Markov chain run for a little while, it usually becomes close to stationary. Also, when we start a Markov chain simulating an X_1 having the stationary distribution, then we get a stationary sequence.

11 Data transmission

Let us first look at a simple case. Assume that we want to send Matzinger's lecture notes to some relatives in Russia through a transmission channel like the Internet for example. Each bit sent costs a certain amount of money. Furthermore, during transmission, random transmission errors can occur. We assume that for each bit transmitted there is an error probability $p < 1/2$ and the errors are independent from one bit to the next. Let X_i be the i-th bit sent and let Y_i denote the i-th bit received by the receiver. We assume that:

$$\begin{aligned} P(Y_i = 1|X_i = 1) &= P(Y_i = 0|X_i = 0) = 1 - p, \\ P(Y_i = 0|X_i = 1) &= P(Y_i = 1|X_i = 0) = p \end{aligned}$$

where p does not depend on i. This model is called the *Binary Symmetric Channel* (BSC). Let us first look at an example: Assume that we have a code $D(.)$ which uses as code

word for every letter of the alphabet its rand in the alphabet expressed in binary form so that:

$$
\begin{array}{cc}
a & D(a) = 00001 \\
b & D(b) = 00010 \\
c & D(c) = 00011 \\
d & D(d) = 00100 \\
e & D(e) = 00101 \\
f & D(f) = 00110 \\
g & D(g) = 00111 \\
h & D(h) = 01000 \\
i & D(i) = 01001 \\
j & D(j) = 01010 \\
k & D(k) = 01011 \\
l & D(l) = 01100 \\
m & D(m) = 01101 \\
n & D(n) = 01110 \\
o & D(o) = 01111 \\
p & D(p) = 10000 \\
q & D(q) = 10001 \\
r & D(r) = 10010 \\
s & D(s) = 10011 \\
t & D(t) = 10100 \\
u & D(u) = 10101 \\
v & D(v) = 10110 \\
w & D(w) = 10111 \\
x & D(x) = 11000 \\
y & D(y) = 11001 \\
z & D(z) = 11010 \\
. & D(.) = 11011
\end{array}
$$

Let us now look at what happens when we transmit a text which uses the above encoding. Assume we want to transmit the message "math". We find:

$$D(math) = D(m)D(a)D(t)D(h) = 01101000011010001000$$

Let us now add a few random errors: take for example $p = 25\%$. After simulation our message reads:

$$01111001001010101001$$

If we decode this we find:

$$D^{-1}(01111001001010101001) = D^{-1}(01111)D^{-1}(00100)D^{-1}(10101)D^{-1}(01001) = odui$$

We see the message has been completely changed. Next we are going to try to transmit the same bit several times and hope in this way be able to reconstruct the original message.

Assume that our text consists of a four letter alphabet $S = \{m, a, t, h\}$. Let us use the following code:

$$D(a) = 00, D(m) = 10, D(t) = 01, D(h) = 11.$$

Let the message be:

$$math.$$

Now we encode this message:

$$D(math) = 10000111$$

Instead of transmitting it directly we are going to transmit every bit five times. This will cost more money, but in this way we increase the chances of being able to decode the message correctly: So we send:

$$1111100000000000000000000000011111111111111111.$$

Assume that the probability of an error p is 0.1. What it the probability that we can reconstruct a bit which we sent five times correctly? We work in the following way: we take groups of five bits and decide that it correspondent to the bit which appears most often: for example if we receive:

$$11110$$

as one group of five bits we assume that the bit was a 1 and that the last bit out of the five is an error. Hence we use a majority rule to try to determine what the original message was. The probability that our majority rule fails in identifying what the bit was is thus: if we take groups of five bits: the probability to have more than two errors. This is

$$\binom{5}{3}(0.1)^3(0.9)^2 + \binom{5}{4}(0.1)^4(0.9)^1 + (0.1)^5 \approx o(\frac{1}{100}$$

So we see that the probability of an error is small, about one in hundred bits only is wrong. With this strategy how much information can we transmit? How much do we have to pay to send one book.
Take the book, compress it maximally and the send each bit five times. Say the book is 200 pages. Means about, 200×1000 characters. entropy of English: about 2.71. So, encoded we get $200 \times 1000 \times 2.71$ binary bits. We send each bit five times, so the total number of bits we need is about $200 \times 1000 \times 2.71 \times 5$. The number of bits we use per information send is:

$$\frac{200 \times 1000 \times 2.71 \times 5}{200 \times 1000 \times 2.71} \frac{1}{5}$$

Thus, per actual information bit sent with the above scheme it seems like we need to pay for five transmission bits. Information sent per symbol sent in our scheme is about 1/5. This is the *information transmission rate R*. The above method to decode a received message uses the majority rule. We will see that this method is not very efficient. By this we mean, that we could send much less symbols and yet have a more reliable way for

the receiver to figure out what the original message was. Also, it should be noted, that on the long run the above described method produces a constant proportion of errors. Hence, the receiver after trying to decode the received message, still is left with a (may be small) but fix proportion of errors. We will see that we can improve this as long as the transmission rate R is not too large.

12 Conditional entropy

Assume that $Z_1, Z_2, \ldots$ is an i.i.d. sequence of two-dimensional random symbols

$$Z_1 = (X_1, Y_2), Z_2 = (X_2, Y_2), \ldots .$$

Let

$$X = (X_1, \ldots, X_n)$$

and let

$$Y = (Y_1, \ldots, Y_n).$$

Imagine the following data compression problem:
we have stored the text X. Now, we want to store Y. What is the maximum compression we can reach for Y without information-loss? Since X is already stored, we can use X to code and decode Y. In this way, we can store Y on less space than if we had to store it alone. We only need to store the information contained in Y which is not already contained in X. Let us illustrate how this is done:

Assume that the joint probabilities are given by:

$$\begin{pmatrix} P(X_i = a, Y_i = a) & P(X_i = b, Y_i = a) & P(X_i = c, Y_i = a) \\ P(X_i = a, Y_i = b) & P(X_i = b, Y_i = b) & P(X_i = c, Y_i = b) \\ P(X_i = a, Y_i = c) & P(X_i = b, Y_i = c) & P(X_i = c, Y_i = c) \end{pmatrix} = \begin{pmatrix} 0 & 1/8 & 1/8 \\ 1/8 & 1/4 & 0 \\ 1/8 & 1/8 & 1/8 \end{pmatrix}$$

Recall that the conditional probability of the event A given the event B is denoted by $P(A|B)$ and is equal to

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Thus conditional on $X_i = a$, we have that Y_i has the following probabilities:

$$\begin{aligned} P(Y_i = a|X_i = a) &= 0 \\ P(Y_i = b|X_i = a) &= \frac{P(Y_i = b, X_i = a)}{P(X_i = a)} = \frac{1/8}{1/8 + 1/8} = \frac{1}{2} \\ P(Y_i = c|X_i = a) &= \frac{P(Y_i = c, X_i = a)}{P(X_i = a)} = \frac{1/8}{1/8 + 1/8} = \frac{1}{2} \end{aligned}$$

A variable, which is equal to a with zero probability, and equal to b or c with 50% probability each can be optimally compressed by using a block code D:

$$D(b) = 0, D(c) = 1.$$

If we know that $X_i = b$ then we find:

$$P(Y_i = a|X_i = b) = \frac{P(Y_i = c, X_i = a)}{P(X_i = a)} = \frac{1/8}{1/8 + 1/8 + 1/4} = \frac{1}{4}$$
$$P(Y_i = b|X_i = b) = \frac{P(Y_i = b, X_i = b)}{P(X_i = b)} = \frac{1/4}{1/8 + 1/8 + 1/4} = \frac{1}{2}$$
$$P(Y_i = c|X_i = b) = \frac{P(Y_i = c, X_i = a)}{P(X_i = a)} = \frac{1/8}{1/8 + 1/8 + 1/4} = \frac{1}{4}$$

A random letter which is equal to a with prob. $1/4$, equal to b with prob. $1/2$ and equal to c with prob. $1/4$ can be compressed maximally by a code F:

$$F(a) = 00, F(b) = 1, F(c) = 01.$$

Eventually we have that when $X_i = c$ then

$$P(Y_i = a|X_i = c) = \frac{1}{2}$$
$$P(Y_i = b|X_i = c) = 0$$
$$P(Y_i = c|X_i = c) = \frac{1}{2}$$

A random letter which is equal to a with prob 0.5, to b with prob. 0 and equal to c with prob. 0.5 can be maximally compressed by a block code G:

$$G(a) = 1, G(c) = 0.$$

Every time $X_i = a$, we use code D to encode Y_i. When $X_i = b$, we use code F and when $X_i = c$ we use code G. Let us give an example:

$$(Z_1, Z_2, Z_3, Z_4) = ((a, b), (a, c), (c, c), (b, a))$$

where we took $n = 4$. In this case

$$X = (a, a, c, b)$$

and

$$Y = (b, c, c, a).$$

and hence $Y_1 = b$, $Y_2 = c$, $Y_3 = b$ and $Y_4 = a$. Since $X_1 = a$, we use code D to encode Y_1 and find $D(Y_1) = D(b) = 0$.
We have $X_2 = a$, and thus we use again code D to encode Y_2. We find $D(Y_2) = D(c) = 1$.

Furthermore, $X_3 = c$. Hence we use code G to encode Y_3. We find $G(Y_3) = G(c) = 0$. Eventually, $X_4 = b$. This implies that we use code F to encode Y_4. We find $F(Y_4) = F(a) = 00$. Thus the message $Y = bcba$ when encoded gives:

$$D(b)D(c)G(c)F(a) = 01000.$$

When all the codes used are instantaneous, it is easy to see that one can decode the encoded Y correctly y provided we are given X.
How much space do we need to store the text Y if we can use X for encoding and decoding purposes? To answer this question let us define:
-Let N^a, resp N^b, resp N^c be the number of a's, b's and c's appearing in the text $X = X_1 \dots X_n$.
- Let Y_j^a be the j-th Y_i for which $X_i = a$. Let Y^a be the text: $Y^a := Y_1^a \dots Y_{N^a}^a$.
- Let Y_j^b be the j-th Y_i for which $X_i = b$. Let Y^b be the text: $Y^b := Y_1^b \dots Y_{N^b}^b$.
- Let Y_j^c be the j-th Y_i for which $X_i = c$. Let Y^c be the text: $Y^c := Y_1^c \dots Y_{N^c}^c$.
It is easy to check that the three texts Y^a, Y^b and Y^c are all independent of each other. They are also all independent of X. It is easy to see that if we are given Y^a, Y^b, Y^c and X we can reconstruct Y. Thus, the minimum amount of space needed to store Y with the help of X, is equal to the minimum amount of space needed to store Y^a, Y^b and Y^c. Since the three random texts are independent of each other, the total amount of space to store (Y^a, Y^b, Y^c) is the sum of the amount of space needed for each Y^a, Y^b and Y^c.
It is easy to check that the Y_i^a's are independent of each other. So the space needed to store Y^a is equal to $H(Y_1^a)$ times the number of variables Y_i^a in Y^a. There are N^a variables Y_i^a in Y^a. By law of large number N^a is approximately equal to

$$P(X_1 = a)n.$$

Hence the space needed to store Y^a is roughly equal to

$$H(Y^a)P(X_1 = a)n.$$

In a similar fashion, the space needed for Y^b is about

$$H(Y^b)P(X_1 = b)n,$$

whilst the space for Y^c is about

$$H(Y^c)P(X_1 = c)n.$$

Thus the space needed to store (Y^a, Y^b, Y^c) is

$$n \cdot \left(H(Y^a)P(X_1 = a) + H(Y^b)P(X_1 = b) + H(Y^c)P(X_1 = c)\right). \tag{12.1}$$

This is approximately the minimum amount of space needed to store Y, if we can use X for the encoding and decoding. If we can use X for coding and decoding, what we store is only the portion of the information contained in Y which is not contained in X. Hence

expression 12.1 is a measure of the amount of information which is left in Y after we have taken out all the information which is also contained in X. If we divide expression 12.1 by the number of letters n in Y, we get the information which is contained in Y but not in X, per symbol of Y. This is then called the *conditional entropy of Y given X*.
For a letter s in the alphabet S, we also denote $H(Y_i^s)$ by $H(Y_i|X_i = s)$. This means the entropy of Y_i when we take Y_i to have a conditional probability law obtained by conditioning on $X_i = s$.

Definition 12.1 *We define the conditional entropy of Y_i given X_i to be equal to:*

$$H(Y_i|X_i) = \sum_{s \in S} H(Y_i|X_i = s)P(X_i = s).$$

13 Mutual information

How much information does X contain about Y?
We saw that the information which is contained in Y but not in X is equal to $H(Y|X)$. So the rest of the information in Y must be contained also in X. The total amount of information contained in Y is equal to the entropy $H(Y)$. Hence the information which is contained in X about Y is equal to $H(Y) - H(Y|X)$. This information is called the mutual information of X and Y. It is a measure of the amount of information shared by X and Y.

Definition 13.1 *We define the mutual information $I(X;Y)$ of X and Y to be equal to:*

$$I(X;Y) =: H(Y) - H(Y|X).$$

It is an interesting fact that the information which X contains about Y is equal to the amount of information which Y contains about X.

Lemma 13.1 *We have that*

$$I(X;Y) = I(Y;X).$$

Proof. we have that

$$I(X;Y) = H(Y) - H(Y|X) =$$

■

Proof. ■

14 Transmission channels

The model we consider goes as follows. We assume that we are able to transmit symbols from an alphabet S through a transmission channel. Let X_i denote the i-th symbol sent and let Y_i denotes the i-th symbol received by the receiver. For each symbol sent there is a positive probability of an error. Hence $P(Y_i = X_i) < 1$. We assume however

that for a given transmission channel the error probabilities remain fix during the whole transmission process. We also assume that the transmission errors are independent from one letter sent to the next. A transmission channel is defined by specifying for each pair $r, s \in S$ the probability that the receiver receives an s if we send an r. Thus We will denote this probability by p_{rs}. This is the conditional probability that $Y_i = s$ given that $X_i = r$: $p_{rs} = P(Y_i = s|X_i = r)$.
Let us give an example when we have a tree-symbol alphabet $S = \{a, b, c\}$:

$$\begin{pmatrix} p_{aa} & p_{ab} & p_{ac} \\ p_{ba} & p_{bb} & p_{bc} \\ p_{ca} & p_{cb} & p_{cc} \end{pmatrix} = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

If we send an a in this example, the probability that the receiver receives an a is 80%. On the other hand, the receiver will receive a b with 10% probability if we send an a. If we send a b the receiver gets a b with 70% probability, an a with 10% probability and a c with 20% probability.

15 Shannon's second theorem for the binary symmetric channel

The general setting is the following:

- Let $M = M_1 M_2 \dots M_k$ designate a message consisting of k symbols from an alphabet S. We assume that M is generated by a random process.
- Let D designate a code which we use to encode the message M to make it fit for transmission. Thus
$$X := D(M)$$
designates the binary string which we send through our transmission channel.
- Let n designate the number of symbols we send over the transmission channel. Hence, n is the length of the binary string X:
$$n := |X| = |D(M)|.$$
X can be written as a string of n zero's and one's':
$$X = X_1 X_2 \dots X_n$$
where X_i is the i-th bit sent over the transmission channel.
- Let $Y = Y_1 \dots Y_n$ designate the received binary string by the receiver after transmission. Thus Y_i is the i-th bit received.

- Let p designate the probability that a given bit is not transmitted correctly. Thus

$$p = P(Y_i \neq X_i).$$

- We assume that the transmission errors are independent of each other. Let A_i be the event that the i-th bit is transmitted correctly, that is:

$$A_i := \{X_i = Y_i\}.$$

  We assume that the A_i's are independent of each other for different i's.

- Since we consider the symmetric channel, we assume that the probability of an error does not dependent on what the bit which we send is equal to:

$$P(A_i|X_i = 0) = P(A_i|X_i = 1) = P(A_i) = 1 - p.$$

- Let F designate the code which we use to try to "decode" the received message Y. Thus we apply F to Y in order to try to retrieve M.

Definition 15.1 *We define the rate at which we transmit information to be the amount of information sent over the number of symbols sent. Thus:*

$$R = \frac{H(M)}{n} = \frac{H(M)}{|D(M)|},$$

where M is the message we try to transmit and $n = |X| = |D(M)|$ is the number of symbols transmitted.

Definition 15.2 *For a given channel, the channel capacity $\mathcal{C}$ is defined to be*

$$\mathcal{C} := \max I(X_i; Y_i),$$

where the maximum is taken over all possible probability laws for X_i.

We are now ready for Shannon's second theorem:

Theorem 15.1 *Assume we are trying to transmit information with a given channel with capacity $\mathcal{C} > 0$. Let $R > 0$ be a fix number. Then,*

- *If $R < \mathcal{C}$, the it is possible to send messages through that channel at rate R in a reliable way. This means that it is possible to send messages at rate R through the channel so that the receiver can with high probability correct the transmission errors. (with high probability, we mean probability as close to 100% for messages long enough.)*

- *If $R > \mathcal{C}$ it is not possible to send information in a reliable way through that channel at rate R.*

To prove that it is not possible to send information through a channel at rate $R > \mathcal{C}$ in a reliable way, we need the following lemma:

Lemma 15.1 *Let $X = X_1X_2 \dots X_n$ and $Y = Y_1Y_2 \dots Y_n$ be two random texts consisting of n symbols from an alphabet S. Let $\vec{s} = s_1s_2 \dots s_n \in S^n$ designate a (non-random) string of symbols of S. Assume that for every $\vec{s} \in S^n$ we have:*
conditional on $X = \vec{s}$, the Y_i's are independent of each other and depend only on their corresponding X_i. That is:

$$\mathcal{L}(Y|X = \vec{s}) = \bigotimes_{i=1}^{n} \mathcal{L}(Y_i|X_i = s_i). \tag{15.1}$$

Then, we have that

$$I(X;Y) \leq I(X_1;Y_1) + \ldots + I(X_n;Y_n).$$

Proof. By definition of mutual information, we have that

$$I(X;Y) = H(Y) - H(Y|X) \tag{15.2}$$

Now, we learned that the entropy of a text is smaller equal to the sum of the entropy of the letters. (See lemma 9.1.) Hence

$$H(Y) = H(Y_1Y_2 \dots Y_n) \leq H(Y_1) + \ldots + H(Y_n) \tag{15.3}$$

Furthermore, we find that

$$H(Y|X) = \sum_{\vec{s} \in S^n} H(Y|X = \vec{s})P(X = \vec{s}). \tag{15.4}$$

However conditional on $X = \vec{s}$, the letters of the text Y become independent of each other. The entropy for a text with independent letters is equal to the sum of the entropies of the letters. Hence:

$$H(Y|X = \vec{s}) = H(Y_1|X = \vec{s}) + H(Y_2|X = \vec{s}) + \ldots + H(Y_n|X = \vec{s}).$$

Conditional on $X = \vec{s}$, we have that Y_i depends only on the corresponding X_i. Hence,

$$H(Y_i|X = \vec{s}) = H(Y_i|X_i = s_i)$$

where s_i designate the i-th coordinate of the string $\vec{s}$. Hence

$$H(Y|X = \vec{s}) = \sum_{i=1}^{n} H(Y_i|X_i = s_i)$$

Plugging the last expression into 15.4, we find:

$$H(Y|X) = \sum_{\vec{s} \in S^n} \sum_{i=1}^{n} H(Y_i|X_i = s_i)P(X = \vec{s}) = \sum_{s \in S} \sum_{i=1}^{n} \sum_{s \in S} H(Y_i|X_i = s)P(X_i = s).$$

Using the last equality together with 15.3 in 15.2, we find that

$$I(X;Y) \leq (H(Y_1) + \ldots + H(Y_n)) - \sum_{i=1}^{n} \sum_{s \in S} H(Y_i|X_i = s_i)P(X_i = s) =$$

$$\left(H(Y_1) - \sum_{s \in S} H(Y_1|X_1 = s_i)P(X_1 = s) \right) + \ldots + \left(H(Y_n) - \sum_{s \in S} H(Y_n|X_n = s_i)P(X_n = s) \right) =$$

$$I(X_1;Y_1) + \ldots + I(X_n;Y_n)$$

■

Let us next show why we can only transmit information at a rate R which is below channel capacity $\mathcal{C}$. Assume that we could transmit the random message M in such a way that the receiver could decode it correctly with hundred percent probability. Then, the information of M would be contained in X as well as in Y. Hence, the mutual information between X and Y would be at least the information contained in M. The information contained in M is $H(M)$. Thus, we would have

$$H(M) \leq I(X;Y). \tag{15.5}$$

However, we can not be 100% sure that the reciever can decode the received message correctly. Instead we only are close to 100% for long messages. Thus, instead of the inequality 15.5, we have instead:

$$(1 - (\epsilon(k))H(M) \leq I(X;Y) \tag{15.6}$$

where $\epsilon(k) \to 0$ as $k \to \infty$. (Here k denotes the length of the message M.) By lemma 15.1, we find that

$$I(X;Y) \leq I(X_1;Y_1) + I(X_2;Y_2) + \ldots + I(X_n;Y_n). \tag{15.7}$$

By definition of channel capacity we have that $I(X_i;Y_i) \leq \mathcal{C}$. Applying this to 15.7 gives

$$I(X;Y) \leq n \cdot \mathcal{C} \tag{15.8}$$

Using inequality 15.8 in 15.6, we find

$$(1 - \epsilon(k))\frac{H(M)}{n} \leq \mathcal{C} \tag{15.9}$$

Now, $H(M)/n$ is equal to the rate R. Letting k go to infinity we have that $(1 - \epsilon(k)) \to 1$ and hence we obtain:

$$R \leq \mathcal{C}.$$

16 Random coding

In this section we prove that for $R < \mathcal{C}$ it is possible to transmit information at rate R in a reliable way. To prove this we use random coding. Let D designate the code used to encode the original message before transmission. In random coding the code words of D are chosen at random by flipping a fair coin. For different messages one sends one keeps however using the same list of code words. Decoding is done by comparing a received word with the list of code words. The receiver guesses that the word in the code word list which is closest to the received word is the word which has actually been sent. To compare how close words are from one another we use the *Hamming distance*. The Hamming distance for two words v and w of the same length is equal to the number of differences between v and w. For example if

$$v = 0010, w = 1011$$

then $d(v, w) = 2$ because the first and the last bit are different. In random coding the code D used to prepare the message for transmission encodes strings of l symbols at a time. Hence, D is a block code for S^l. To have the random coding work well with high probability, we need to take l large.
We can always assume that the message M we want to transmit is an i.i.d. binary message with entropy 1 per letter. This is so because any ergodic message, when it is compressed maximally and put in binary form becomes a random text of that type. We thus assume throughout this section that the symbols of the message M: $M_1, M_2, \ldots$ are i.i.d. such that

$$P(M_i = a) = P(M_i = b) = 0.5.$$

Let us define random coding at rate R.

Algorithm 16.1

- *The D code used to encode the message M before transmission is a block code on S^l.*
- *For each string of l letters $\vec{s}S^l$ chose a code word $D(\vec{s})$ at random by throwing an unbiased coin l/R times.*
- *To decode the received message, chose the code word of $D(S^l)$ which is closest to the received word.*

We want to prove that random coding works with high probability when the rate R is strictly below the capacity of the channel. Let $w_1, w_2, \ldots, w_{2^l}$ denote the code words defining our code D. Hence w_1 is the code word for the string $aaa \ldots a$ of length l and consisting only of a's. Hence w_2 is the code word for the sting $aaa \ldots ab$ of length l. We assume without loss of generality that the code word sent is the first one, that is w_1. (This assumption is made to facilitate notation.) Let v_1 denote the binary word received by the receiver. Hence v_1 is obtained from w_1 by adding approximately a percentage $|p|$ of errors. Hence, (by law of large numbers), we have that

$$\frac{d(w_1, v_1)}{|v_1|} \approx p < 0.5$$

holds with high probability when l is large enough. For a code word, which is not sent, we have another situation. Take for example the code word w_2. In this case w_2 and v_1 are independent and we have that with high probability

$$\frac{d(w_2, v_1)}{|v_1|} \approx 0.5$$

holds when l is large enough.(Law of large numbers.) Hence, the received word v_1 tends to be closer to the code word which was sent then to another code word. This is what we want to use to recognize as a receiver which code word was sent. However since there exists many code words (total number of 2^l) some of them could by chance still be close to v_1 and thus induce the receiver in error. We need to prove that with high probability, all the code words which were not sent are different enough from the received word v_1. This insures then that the receiver can decode the received word correctly with high probability. To prove that when the rate is strictly below the capacity, decoding works with high prob., we need a few definitions:
Let $\epsilon > 0$ be a very small but fix quantity. We define the event B^ϵ to be the event that the following inequality holds:

$$\left| \frac{d(w_1, v_1) - p}{|w_1|} \right| < \epsilon.$$

For all $i \in 2, \ldots, 2^l$, let A_i^ϵ denote the event that:

$$\frac{d(w_i, v_1) - p}{|w_1|} \geq \epsilon$$

holds. The event A_i^ϵ guaranties that the code word w_i is far enough from the received code word v_1.
Let A^ϵ denote the event that all the events A_i^ϵ hold simultaneously:

$$A^\epsilon = \bigcap_{i=2}^{2^l} A_i^\epsilon.$$

Let B_i denote the event that

$$\frac{d(w_i, v_1)}{|v_1|} = p.$$

Obviously when B^ϵ and A^ϵ both hold then the decoding works. By law of large number B^ϵ holds with high probability. So we just need to prove that when the rate R is strictly below the capacity of the transmission channel, then A^ϵ also holds with high probability. In other words we need to prove that the probability that A^ϵ does not hold is small. Now:

$$P\left((A^\epsilon)^c\right) = P\left((\bigcap_{i=2}^{2^l} A_i^\epsilon)^c\right) = P\left(\bigcup_{i=2}^{2^l} (A_i^\epsilon)^c\right) \leq \sum_{i=2}^{2^l} P\left((A_i^\epsilon)^c\right) = (2^l - 1) P((A_2^\epsilon)^c).$$

For small ϵ, we have that $P((A_1^\epsilon)^c)$ is very close to $P(B_2)$. Hence, we have to prove the the following expression

$$(2^l - 1)P(B_2) \tag{16.1}$$

is small. Let k denote the length of the word $|w_i|$. Hence,

$$P(B_2) = P(d(w_2, v_1) = kp) = P\left(\ BIN(k, 0.5) = k \cdot p \ \right).$$

Here, $BIN(k, 0.5)$ denotes a binomial variable with parameter k and 0.5. Hence $BIN(k, 0.5)$ is the variable which is obtained by taking the sum of five coin-tosses with unbiased coins. Now, $P(BIN(k, 0.5) = k \cdot p)$ is equal to

$$\binom{k}{kp} 0.5^k = \binom{k}{kp} p^{pk}(1-p)^{(1-p)k} \frac{0.5^k}{p^{pk}(1-p)^{(1-p)k}}.$$

However,

$$\binom{k}{kp} p^{pk}(1-p)^{(1-p)k} \tag{16.2}$$

is equal to the probability that a binomial variable with parameters k and p is equal to kp. Hence, expression 16.2 is smaller than 1. We thus find that the expression on the right side of 16.1 is smaller than

$$2^l \ \frac{0.5^k}{p^{pk}(1-p)^{(1-p)k}} = 2^{l \cdot const}$$

where the constant *const* is equal to:

$$const = 1 - \ \left[\frac{1 - p\log_2(1/p) - (1-p)\log_2(1/(1-p))}{R} \right] = 1 - \ \frac{1 - H(p)}{R}$$

(Here $H(p)$ denotes the entropy of a coin with side 1 having probability p.)
In the next section we prove that the capacity of a binary symmetric channel with error probability is equal to $1 - H(p)$. Since the rate R is supposed to be strictly smaller than the channel capacity we get:

$$R < 1 - H(p)$$

from which it follows:

$$1 < \frac{1 - H(p)}{R}$$

and finally

$$1 - \frac{1 - H(p)}{R} < 0.$$

This implies that the constant *const* is strictly negative:

$$const = 1 - \ \left[\frac{1 - p\log_2(1/p) - (1-p)\log_2(1/(1-p))}{R} \right] = 1 - \frac{1 - H(p)}{R} < 0.$$

Hence, because the constant *const* is negative we find that $2^{l \cdot const}$ is exponentially small in l and thus for l only "medium" large is already extremely small. Hence the probability that the decoding does not work correctly is extremely small.

17 Channel capacity

Let us calculate here the channel capacity for a binary channel with error-probability equal to p.
By definition the channel capacity is equal to

$$\max I(X_1; Y_1)$$

where the maximum is taken over all possible probability laws for X_1. (This means aver all possible values a and b such that $a + b = 1$, $a, b \geq 0$ and $P(X_1 = 0) = a$ and $P(X_1 = 1) = b$.) Now, we have:

$$\begin{aligned} &I(X_1; Y_1) = H(Y_1) - H(Y_1|X_1) = \\ &= H(Y_1) - (H(Y_1|X_1 = 0) \cdot P(X_1 = 0) + H(Y_1|X_1 = 1) \cdot P(X_1 = 1)) \,. \end{aligned}$$

Note, that the entropies of Y_1 conditional on $X_1 = 0$ is equal to the entropies of Y_1 conditional on $X_1 = 1$ and is equal to:

$$H(Y_1|X_1 = 0) = H(Y_1|X_1 = 1) = p \log_2 \left(\frac{1}{p}\right) + (1 - p) \log_2 \left(\frac{1}{1 - p}\right) = H(p).$$

This implies that

$$\begin{aligned} &I(X_1; Y_1) = H(Y_1) - (H(p)P(X_1 = 0) + H(p)P(X_1 = 1)) = \\ &H(Y_1) - \ \ H(p) \ (P(X_1 = 0) + P(X_1 = 1) \ ) = H(Y_1) - H(p). \end{aligned}$$

Note that $H(p)$ does not depend on $P(X_1 = 0) = a$ and $P(X_1 = 1) = b$. So, in our maximizing problem, $H(p)$ is to be treated like a constant. Hence, for the binary symmetric channel we find that the channel capacity is equal to

$$(\max H(Y_1)) - H(p) \tag{17.1}$$

where the maximum is taken over all possible probability laws for X_1. Let us find the maximum $\max H(Y_1)$. Let $q = P(Y_1 = 1)$. Then,

$$H(Y_1) = H(q) = q \log_2 \left(\frac{1}{q}\right) + (1 - q) \log_2 \left(\frac{1}{1 - q}\right).$$

Taking the derivative, we find

$$\frac{dH(q)}{dq} = -\log_2 \left(\frac{q}{1 - q}\right).$$

Putting the derivative equal to zero gives

$$\frac{q}{1 - q} = 1$$

which implies that $q = 0.5$. Hence, $H(q)$ takes it maximum value for $q = 0.5$. When we take for the probabilities of X_1 the following: $P(X_1 = 0) = P(X_1 = 1) = 0.5$ we find by law of total probability that

$$P(Y_1 = 0) = P(Y_1 = 0|X_1 = 0)0.5 + P(Y_1 = 0|X_1 = 1)0.5 = (1 - p)0.5 + p0.5 = 0.5.$$

Hence for the choice $P(X_1 = 0) = P(X_1 = 1) = 0.5$, we obtain $q = 0.5$ and thus $H(Y_1)$ finds itself maximized. For $q = 0.5$, we get

$$H(0.5) = 0.5 \log_2(2) + 0.5 \log_2(2) = 1.$$

Hence

$$\max H(Y_1)$$

is equal to 1. With 17.1 we find that for a binary symmetric channel the capacity is given by

$$1 - H(p).$$

18 Differential entropy

Definition 18.1 *Let X be a continuous random variable with probability-density function $f(.)$. Then we define the differential entropy $h(X)$ of X to be equal to:*

$$h(X) := \int f(s) \log_2 \left(\frac{1}{f(s)} \right) ds.$$

(For $f(s) = 0$, we define $f(s) \log_2(1/f(s))$ to be equal to zero.)

Lemma 18.1 *Let X be a continuous random variable with probability-density function $f(.)$. Assume that $f(.)$ is continuous. Let Y^n be a discrete approximation of X, so that:*

$$\text{when } X \in [i/n, (i+1)/n[, \text{ then } Y^n = i/n.$$

Then,

$$\lim_{n \to \infty} (H(Y^n) - \log_2(n)) = h(X).$$

In other words, if X is a continuous random variable, then for large n, $H(Y^n)$ is approximately equal to:

$$H(Y^n) := h(X) + \log_2(n).$$

Recall that when X and Y are two jointly continuous random variables, then the *conditional density* of Y conditional on $X = s$ is given by:

$$\frac{f(s, y)}{f_X(s)}$$

where $f_X(s)$ denotes the probability-density of X at the point s. The density of X can be calculated from the joint density $f(x,y)$ in the following manner:

$$f_X(s) = \int f(s,y)dy.$$

Definition 18.2 *Let X and Y be to continuous random variables having joint density equal to $f(x,y)$. Then we define the conditional differential entropy $h(Y|X=s)$ of Y given $X=s$ to be equal to the differential entropy of a random variable which has density equal to the conditional density of Y given $X=s$. Hence:*

$$h(Y|X=s) := \int \frac{f(s,y)}{f_X(s)} \log_2 \left(\frac{f_X(s)}{f(s,y)} \right) dy.$$

The conditional differential entropy of Y given X is defined:

$$h(Y|X) := \int h(Y|X=s) f_X(s) ds.$$

Compare the last formula with the definition of conditional entropy.

Lemma 18.2 *Let X and Y be two continuous random variables with a joint density function $f(x,y)$. Let (X^n, Y^n) be the discrete approximation of (X,Y) obtained by partioning $\mathbb{R}$ into intervals of length $1/n$. (By this we mean that whenever:*

$$(X,Y) \in [i/n, (i+1)/n[\times [j/n, j/(n+1)[$$

then $(X^n, Y^n) = (i/n, j/n)$.) With this notations we have:

$$\lim_{n \to \infty} I(X^n, Y^n) = h(Y) - h(Y|X).$$

Definition 18.3 *Let X and Y be two random variables having joint density $f(x,y)$. We define the mutual information $I(X,Y)$ to be equal to $I(X,Y) := h(Y) - h(Y|X)$.*

We next are explaining what the model-assumptions for a *Gaussian channel* is. We have the following notations and assumptions.

1. Let X_i designate the i-th signal sent.
2. Let Y_i designate the i-th signal received.
3. Let ϵ_i designate the error occurring during the transmission of the i-th signal. Hence, $Y_i := X_i + \epsilon_i$.
4. We assume that $\epsilon_1, \epsilon_2, \ldots$ is an i.i.d. sequence of normal variables.
5. We assume that the mean of the errors is zero on the long run:

   $$E[\epsilon_i] = 0$$

   for all $i \in \mathbb{N}$ and $VAR[\epsilon] = \sigma^2$ is given.

Lemma 18.3 *Let X be a normal random variable with variance σ^2. Then*

$$h(X) = 0.5 \log_2(2\pi e \sigma^2).$$

In the continuous case, a transmission channel consists in giving for all possible value $s \in R$, the conditional probability of Y_i given that $X_i = s$.
In many cases, when we deal with continuous transmission channels there is a power constrain. Typically, we have a requirement that

$$VAR[X_i] \leq P$$

where $P > 0$ is a given number. The next theorem is Shannon's second theorem for continuous channels.

Theorem 18.1 *Let $P > 0$ be a number. Assume that we are given a continuous transmission channel over which we are allowed to transmit below power P. (This means that we are required to transmit in such a way that $VAR[X_i] \leq P$ for all $i \in \mathbb{N}$.) Then the maximum amount of information we can send over this channel per symbol sent in a reliable way is given by*

$$\max I(X_i; Y_i)$$

where the maximum is taken over all probability distributions for X_i such that $VAR[X_i] \leq P$. (Transmission in a reliable way means as usual that the receiver can decode the message correctly and find the original message with probability close to one.)

Lemma 18.4 *Let X be a random variable such that $VAR[X] \leq P$, where $P > 0$ is a given number. Then, X has maximum differential entropy under the above power constrain iff X is normal with variance P.*

Theorem 18.2 *Assume that we transmit over a Gaussian channel with the varaince of the transmission errors given: $VAR[\epsilon_i] = \sigma > 0$ given. Then the maximum amount of information per signal sent which we can transmit in a reliable way is given by:*

$$\frac{1}{2} \cdot \log_2 \left(1 + \frac{P}{\sigma^2}\right).$$

Proof. The channel capacity for a continous channel under our power constrain is given by

$$\max I(X_i; Y_i)$$

where we take the maximum over all probability laws for X_i satisfying $VAR[X_i] \leq P$. Now in the continous case:

$$I(X_i, Y_i) = h(Y_i) - h(Y_i|X_i) = h(Y_i) - \int h(Y_i|X_i = s) f_X(s) ds.$$

Recall that in the Gaussian channel model

$$Y_i = X_i + \epsilon_i.$$

Conditional on $X_i = s$, X_i becomes a contant. Hence

$$h(Y_i|X_i = s) = h(X_i + \epsilon_i|X_i = s) = h(s + \epsilon_i|X_i = s)$$

Since adding a constant does not change the differential entropy, we have that

$$h(s + \epsilon_i|X_i = s) = h(\epsilon_i|X_i = s).$$

Since, X_i and ϵ are independent we get that

$$h(\epsilon_i|X_i = s) = h(\epsilon_i) = 0.5 \log_2(2\pi e\sigma^2)$$

where in the end we used the formula for the differential entropy of a normal random variable. Thus:

$$I(X_i, Y_i) = h(Y_i) - 0.5 \log_2(2\pi e\sigma^2).$$

Now the maximum value we get for

$$h(Y_i) = h(X_i + \epsilon_i)$$

according to lemma 18.4 is when X_i is normal with variance P. In that case Y_i is normal with variance $P + \sigma^2$. In that case we have a maximum differential entropy $h(Y_i)$ equal to

$$0.5 \log_2(2\pi e(P + \sigma^2)).$$

Hence

$$\begin{aligned} &\max I(X_i; Y_i) = (\ \max h(Y_i)\ ) - 0.5 \log_2(2\pi e\sigma^2) = \\ =&0.5 \log_2(2\pi e(P + \sigma^2)) - 0.5 \log_2(2\pi e\sigma^2) = 0.5 \log_2(2\pi e(1 + P/\sigma^2)). \end{aligned}$$

■