BASIC CONCEPTS

IN

INFORMATION THEORY

Marc URO

CONTENTS

INTRODUCTION

Most scientists agree that information theory began in 1948 with Shannon's famous article. In that paper, he provided answers to the following questions :

- What is "information" and how to measure it?
- What are the fundamental limits on the storage and the transmission of information?

The answers were both satisfying and surprising, and moreover striking in their ability to reduce complicated problems to simple analytical forms.
Since then, information theory has kept on designing devices that reach or approach these limits.

Here are two examples which illustrate the results obtained with information theory methods when storing or transmitting information.

TRANSMISSION OF A FACSIMILE

The page to be transmitted consists of dots represented by binary digits ("1" for a black dot and "0" for a white dot). Its dimensions are 8.5×11 inches. The resolution is 200 dots per inch, that is to say 4×10^4 dots per square inch. Consequently, the number of binary digits to represent this page is $8.5 \times 11 \times 4 \times 10^4 = 3.74\ Mbits$.

With a modem at the rate of 14.4 kbps , the transmission of the page takes **4 minutes and 20 seconds**.

Thanks to techniques of coding (run length coding, Huffman coding), the time of transmission is reduced to **17 seconds**!

STORAGE OF MP3 AUDIO FILES

MP3 stands for Moving Picture Experts Group 1 layer 3. It is a standard for compressed audio files based on psychoacoustic models of human hearing. By means of masking time and frequency, limiting bandwidth and Huffman coding, it allows one to reduce the amount of information needed to represent an audio signal, as the human ear cannot distinguish between the original sound and the coded sound.

Let us consider a musical stereo analog signal. For CD quality, left channel and right channel are sampled at 44.1 Khz. The samples are quantized to 16 bits. One second of stereo music in CD format generates :

$$44.1\times10^{3}\times16\times2=1.411\,\text{Mbits}$$

By using the MP3 encoding algorithm, this value drops to 128 Kbits without perceptible loss of sound quality. Eventually, one minute of stereo music requires :

$$\frac{128\times10^{3}\times60}{8}\approx1\,\text{Mbytes}$$

A CD ROM, which has a capacity of 650 Mbytes, **can store more than 10 hours of MP3** stereo music.

DOWNLOADING MP3 FILES

Over an analogue telephone line

An analogue telephone line is made of a pair of copper wires whose bandwidth is limited to $B=4\,\text{Khz}$. Such a line transmits analogue signals with SNR (Signal to Noise Ratio) $\approx 30\,\text{dB}$ and can be modelled by a memoryless additive Gaussian noise channel. Information theory allows to compute its capacity (in bits/sec) :

$$\boxed{C=B\log_2(1+snr)\quad\text{with } SNR=10\log_{10}snr}$$

To make a long story short, the capacity is the maximum bit rate at which we can transmit information, allowing an arbitrary small probability of error, provided appropriate means are used.

Then, we obtain :

$$\boxed{C \approx 33800 \text{ bits/sec} \approx 4 \text{ Kbytes/sec}}$$

Hence, downloading a 3 minute's MP3 song with a V90 standard modem takes about :

$$\frac{1 \times 3 \times 10^3}{4} = 750 \textbf{ sec} = \textbf{12 minutes and 30 seconds}$$

At busy hours, the downloading speed may lower to 1 Kbytes/sec and the downloading time can reach **50 minutes**!

Over a digital line

As telephone signals are band limited to 4 KHz, the sampling frequency is 8 KHz. In addition, 8 bits are used for quantifying each sample. Then, the bit rate is :
$8 \times 8 = 64$ Kbits/sec $= 8$ Kbytes/sec

Thus, the downloading speed is twice as high as it is in the case of analog lines and it takes **6 minutes 15 seconds** to download a 3 minute MP3 song.

With ADSL (Asymmetric Digital Subscriber Line) modem technology

Using this technology requires having a USB (Universal Serial Bus) or an ethernet modem.

It consists of splitting the available bandwidth into three channels :

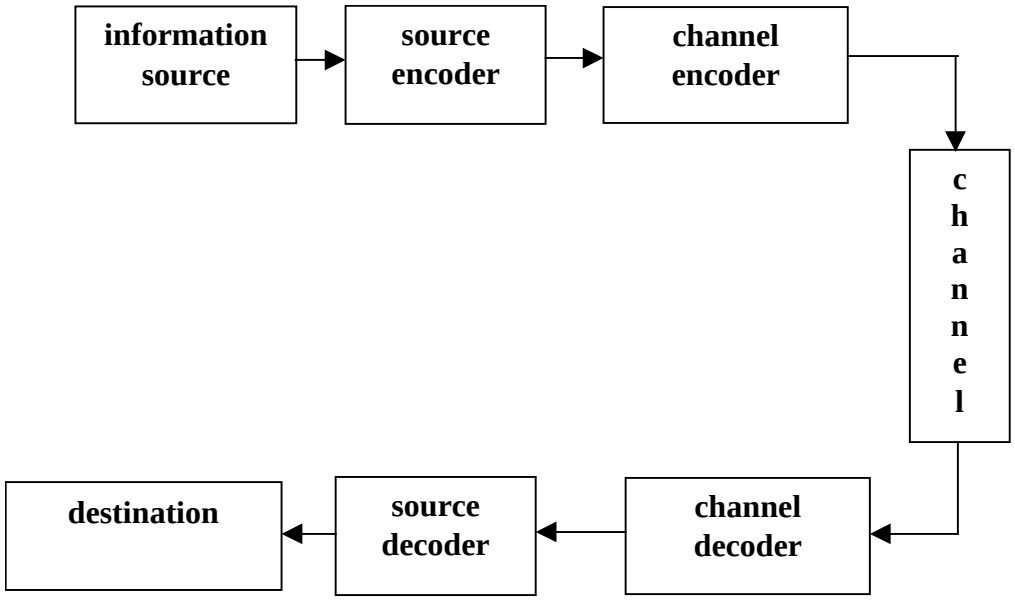
- a high speed downstream channel
- a medium speed upstream channel
- a POTS (Plain Old Telephone Service) channel

The main advantage lies in the fact that you can use your phone and be connected to the internet at the same time. With a 512 Kbits/sec modem, the downloading stream rises to 60 Kbytes/sec. Downloading a 3 minute MP3 song takes only :

$$\frac{1 \times 3 \times 10^3}{60} = \textbf{50 seconds}.$$

SHANNON PARADIGM

Transmitting a message from a transmitter to a receiver can be sketched as follows :



This model, known as the “Shannon paradigm”, is general and applies to a great variety of situations.

- An information source is a device which randomly delivers symbols from an alphabet. As an example, a PC (Personal Computer) connected to internet is an information source which produces binary digits from the binary alphabet {0, 1}.

- A channel is a system which links a transmitter to a receiver. It includes signalling equipment and pair of copper wires or coaxial cable or optical fibre, among other possibilities. Given a received output symbol, you cannot be sure which input symbol has been sent, due to the presence of random ambient noise and the imperfections of the signalling process.

- A source encoder allows one to represent the data source more compactly by eliminating redundancy : it aims to reduce the data rate.

- A channel encoder adds redundancy to protect the transmitted signal against transmission errors.

- Source and channel decoders are converse to source and channel encoders.

There is duality between “source coding” and “channel coding”, as the former tends to reduce the data rate while the latter raises it.

The course will be divided into 4 parts :

- “information measure”
- “source coding”
- “communication channel”
- “error correcting codes”

The first chapter introduces some definitions to do with information content and entropy.

In the second chapter, we will answer three questions :

Given an information source, is it possible to reduce its data rate?
If so,
By how much can the data rate be reduced?
How can we achieve data compression without loss of information?

In the “communication channel” chapter, we will define the capacity of a channel as its ability to convey information. We will learn to compute the capacity in simple situations. In our attempt to recover the transmitted signal from the received signal, Shannon’s noisy coding theorem will help us answer this fundamental question : Under what conditions can the data of an information source be transmitted reliably?

The chapter entitled “Error coding codes” deals with redundancy added to the signal to correct transmission errors. At first sight, correcting errors may seem amazing, as added symbols (the redundancy) can be corrupted too. Nevertheless we will learn to detect and correct errors, provided there are not too many of them.

INFORMATION MEASURE

We begin by introducing definitions to measure information in the case of events. Afterwards, by analogy, we will extend these notions to random variables.

SELF-INFORMATION, UNCERTAINTY

When trying to work out **the information content** of an event, we encounter a difficulty linked with the subjectivity of the information which is effectively brought to us when the event occurs.

To overcome this problem, Shannon pursued the idea of defining the information content $h(E)$ of an event E as a function which depends solely on the probability $P\{E\}$. He added the following axioms:

- $h(E)$ must be a decreasing function of $P\{E\}$: more an event is likely, the less information its occurrence brings to us.
- $h(E)=0$ if $P\{E\}=1$, since if we are certain (there is no doubt) that E will occur, we get no information from its outcome.
- $h(E \cap F)=h(E)+h(F)$ if E and F are independent.

The only function satisfying the above axioms is the logarithmic function:

$$h(E)=\log\frac{1}{P\{E\}}=-\log P\{E\}$$

As $\log\frac{1}{P\{E\}}$ represents a measure, it is expressed in different units according to the chosen base of logarithm.

logarithm base	unit
2	**bit or Sh (Shannon)**
e	**natural unit or nepers**
3	**trit**
10	**decimal digit**

The outcome (or not) of E involves an experiment. Before (respectively, after) this experiment we will think of $i(E)$ as the **uncertainty** (respectively, the **self-information**) associated to its outcome.

Example

Let us consider a pack of 32 playing cards, one of which is drawn at random.

Calculate the amount of uncertainty of the event E = {The card drawn is the king of hearts}.

As each card has the same probability of being chosen, we have

$P\{E\}=\frac{1}{32}$

and we get

$$h(E)=\log_2 32=\log_2 2^5=5\,\text{bits}$$

Since $h(E)$ is an integer, we can easily interpret the result: 5 bits are required to specify one playing card among the 32 cards:

- one bit for the colour (red or black),
- one bit for the suit (hearts or clubs if the colour is red and diamonds or spades if the colour is black)
- and so on …

At each stage, we divide the set of left cards into two subsets having the same number of elements : we proceed by dichotomy.

As the uncertainty of an event E depends on the probability of E, we can also define the **uncertainty of E knowing that another event F has occurred** by using the conditional probability $P\{E/F\}$:

$$h(E/F) = -\log P\{E/F\}$$

with $P\{E/F\} = \dfrac{P\{E \cap F\}}{P\{F\}}$

Example

The setting is the same as the previous example. What is the amount of uncertainty of E = {The card drawn is the king of hearts } knowing F = {The card drawn is a heart}?

We have :

$$P\{E/F\} = \frac{P\{E \cap F\}}{P\{F\}} = \frac{P\{E\}}{P\{F\}} \text{ as } E \subset F$$

and since $P\{E\} = \dfrac{1}{32}$ and $P\{F\} = \dfrac{1}{4}$

we obtain

$$h(E/F) = -\log_2 P\{E/F\} = \log_2 \frac{P\{E\}}{P\{F\}} = \log_2 \frac{32}{4} = \log_2 2^3 = 3 \text{ bits}$$

Interpretation:

The fact that F has occurred determines two bits: one for the colour (red) and one for the suit (hearts). Consequently, specifying one card, whatever it is, requires only 5 – 2 = 3 bits. The uncertainty of E has been reduced thanks to the knowledge of F.

Using the definition of $h(E/F)$ allows us to express $h(E \cap F)$:

$$h(E \cap F) = -\log P\{E \cap F\} = -\log[P\{E/F\} \times P\{F\}] = -\log P\{E/F\} - \log P\{F\}$$

Hence,

$$\boxed{h(E \cap F) = h(E/F) + h(F)}$$

By symmetry, as $E \cap F = F \cap E$, it follows that :

$$\boxed{h(E \cap F) = h(F \cap E) = h(F/E) + h(E)}$$

If E and F are independent, then,

$$P\{E \cap F\} = P\{E\} \times P\{F\}$$
$$P\{E/F\} = P\{E\}$$
$$P\{F/E\} = P\{F\}$$

Accordingly,

$h(E \cap F) = h(E) + h(F)$ (it is one of the axioms we took into account to define the uncertainty)

$$h(E/F) = h(E)$$
$$h(F/E) = h(F)$$

In the example of the previous page, we observed that the knowledge of F reduced the uncertainty of E. This leads us to introduce the amount of information provided by F about E, $i_{F \to E}$, as the reduction in the uncertainty of E due to the knowledge of F :

$$\boxed{i_{F \to E} = h(E) - h(E/F)}$$

Substituting for $h(E/F)$ from $h(E \cap F) = h(E/F) + h(F)$ into the previous definition, we get :

$$\boxed{i_{F \to E} = h(E) - (h(E \cap F) - h(F)) = h(E) + h(F) - h(E \cap F)}$$

As the above expression is symmetric with respect to E and F, we obtain

$$i_{F \to E} = i_{E \to F}$$

This quantity ($i_{F \to E}$ or $i_{E \to F}$) will be denoted by $i(E;F)$ and is called the "**mutual information** between E and F".

Let us return to the previous example :

The mutual information between E and F is :

$$i(E;F) = h(E) - h(E/F) = 5 - 3 = 2 \text{ bits}$$

From this example, we may hastily deduce that $i(E;F) > 0$ for all E and F. However, we have to be very careful as this property is true if and only if $h(E/F) < h(E)$, that is to say if $P\{E/F\} > P\{E\}$. Otherwise, we have $h(E/F) > h(E)$ and $i(E;F) < 0$.

Example

Two playing cards are simultaneously drawn from a pack of 32 cards. Let E (respectively F) be the event {At least one of the two drawn cards is red} (respectively { The king of spades is one of the two drawn cards}).

What is the amount of mutual information between E and F?

We have :

$$i(E;F) = h(E) - h(E/F)$$

in which

$$P\{E\} = \frac{16}{32} \times \frac{15}{31} + 2 \times \frac{16}{32} \times \frac{16}{31} = \frac{47}{62}$$

and

$$P\{E/F\} = \frac{16}{31}$$

Thus,

$$i(E;F) = \log_2 \frac{62}{47} - \log_2 \frac{31}{16} \approx -0.5546 \text{ bit}$$

In this case, knowing that F has occurred makes E less likely, and the mutual information is negative.

ENTROPY

Example

A random experiment consists of drawing one card from a pack of 32 playing cards. Let X be the discrete random variable defined as

$\{X = \sqrt{3}\} \Leftrightarrow \{\text{The drawn card is red}\}$
$\{X = 7\} \Leftrightarrow \{\text{The drawn card is a spade}\}$
$\{X = \log \pi\} \Leftrightarrow \{\text{The drawn card is a diamond}\}$

We can calculate the uncertainty associated with each of the three occurrences :

$$P\{X = \sqrt{3}\} = \frac{1}{2} \Rightarrow h(X = \sqrt{3}) = \log_2 2 = 1 \text{ bit}$$

$$P\{X = 7\} = \frac{1}{4} \Rightarrow h(X = 7) = \log_2 4 = 2 \text{ bits}$$

$$P\{X = \log \pi\} = \frac{1}{4} \Rightarrow h(X = \log \pi) = \log_2 4 = 2 \text{ bits}$$

Then, the average uncertainty becomes

$$1 \times \frac{1}{2} + 2 \times \left(\frac{1}{4} \times 2 \right) = 1.5 \text{ bit}$$

This means that the average number of bits required to represent the possible values of X is 1.5.

In more general terms, the **entropy** $H(X)$ of a discrete random variable X taking values in $\{x_1, x_2, ..., x_n\}$ with $p_i = P\{X = x_i\}$, is the average uncertainty in the outcomes $\{X = x_1\}$, $\{X = x_2\}$, …, $\{X = x_n\}$:

$$\boxed{H(X) = -\sum_{i=1}^{n} p_i \log p_i}$$

We note the following:

- n can be infinite;

- $H(X)$ depends only on the probability distribution of X, not on the actual values taken by X;

- If n is finite, the maximum of $H(X)$ is achieved if and only if X is uniformly distributed over its values (i.e $p_i = \frac{1}{n} \quad \forall i \in \{1, n\}$). Then, we have $H(X) = \log n$;

- A more formal interpretation of $H(X)$ consists in considering $H(X)$ as the expected value of $Y = \log p(X)$ with $p(X) = \sum_{i=1}^{n} 1\!\!1_{\{X=x_i\}} p_i$ where $1\!\!1_{\{X=x_i\}}$ is the indicator function of $\{X = x_i\} = \{\omega / X(\omega) = x_i\}$ i.e. $1\!\!1_{\{X=x_i\}} = \begin{cases} 1 & if \quad X = x_i \\ 0 & if \quad X \neq x_i \end{cases}$

Example

A discrete random variable X takes its values in {0, 1}. The probability distribution is given by :

$P\{X = 1\} = p = 1 - P\{X = 0\}$

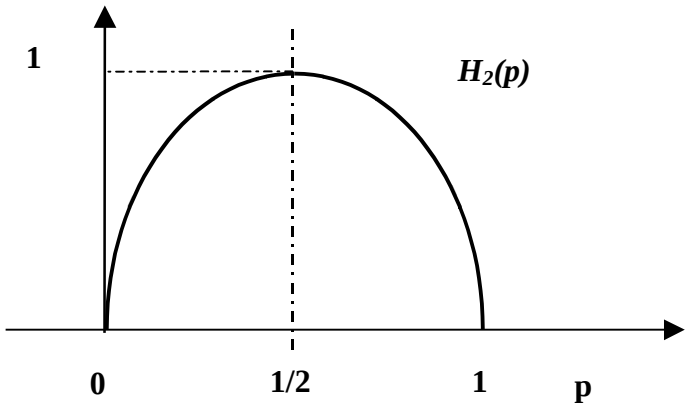
Calculate the entropy of X.

By applying the definition, we get :

$H(X) = -p \log_2 p - (1-p) \log_2 (1-p) = H_2(p)$

$H_2(p)$ is called the binary entropy function.

Sketching $H_2(p)$ versus p gives the following graph :



From the graph, we observe the following:

- the maximum is achieved when $p = \frac{1}{2}$;
- $H_2(p) = 0$ when p = 0 or $p = 1$ (there is no uncertainty);
- $H_2(p)$ is a top convex $(\cap)$ (concave) function which satisfies : $\forall p \in [0,1]\ \ H_2(p) = H_2(1-p)$. This means that $H_2(p)$ is symmetric with respect to the vertical line $p = \frac{1}{2}$.

We now extend the notions related with events to random variables.

Let us consider X (respectively Y) be a discrete random variable taking on values in $\{x_1, x_2, ..., x_n\}$ (respectively $\{y_1, y_2, ..., y_m\}$) with $p_{i.} = P\{X = x_i\}$, $p_{.j} = P\{Y = y_j\}$ and $p_{ij} = P\{X = x_i \cap Y = y_j\}$.

As the entropy depends only on the probability distribution, it is natural to define the **joint entropy** of the pair (X,Y) as

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log p_{ij}$$

Proceeding by analogy, we define:

The **conditional entropy** $H(X/Y)$:

$$H(X/Y) = \sum_{j=1}^{m} p_{.j} H(X/Y = y_j)$$

where $H(X/Y = y_j) = -\sum_{i=1}^{n} P\{X = x_i / Y = y_j\} \log P\{X = x_i / Y = y_j\}$

From this, we have :

$$H(X/Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log P\{X = x_i / Y = y_j\}$$

The **average mutual information** $I(X;Y)$ is the reduction in the entropy of X due to the knowledge of Y :

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

$I(X;Y)$ may be expressed as the expected value of the random variable $I = \log \frac{P(X,Y)}{P(X)P(Y)}$.

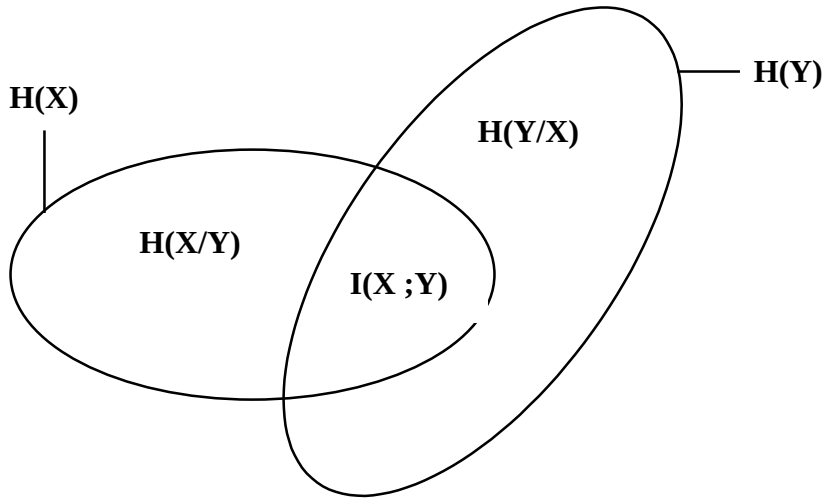
Then, we can rewrite :

$$I(X;Y) = E[I] = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log \frac{p_{ij}}{p_{i.}p_{.j}}$$

Some elementary calculations show that:

- $H(X,Y) = H(X) + H(Y/X) = H(Y) + H(X/Y)$;
- $H(X,Y) = H(X) + H(Y) - I(X;Y)$;
- $H(X/Y) < H(X)$ conditional entropy is always smaller than entropy;
- $I(X;Y) \geq 0$.

The relationship between entropy and mutual information is sketched in the Venn diagram



below:

In the case of independent random variables, the previous relations simplify to :

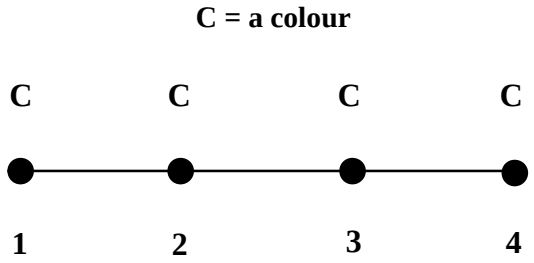
- $H(X/Y = y_j) = H(X)$;
- $H(X/Y) = H(X)$;
- $I(X;Y) = 0$;
- $H(X,Y) = H(X) + H(Y)$.

Example

In the game of mastermind, player A chooses an ordered sequence of four pieces which is concealed from player B. The pieces are of the same shape and may be of different colours. Six colours are available, so that the chosen sequence may consist of one, two, three or four colours. Player B has to guess the sequence by submitting ordered sequences of four pieces. After considering the combination put forth by B, player A tells player B the number of pieces in the correct position and the number of pieces in the wrong position, but without indicating which pieces or positions are correct.

1. What is the average amount of uncertainty in the sequence chosen by player A?

2. The first sequence submitted by player B consists of four pieces of the same colour. What is the average amount of uncertainty in the unknown sequence (the one chosen by player A) resolved by the answer given by player A to the first submitted sequence?



Solution

1. As the number of possible sequences is $6^4 = 1{,}296$, let X be a discrete random variable taking on 1,296 different values according to the chosen sequence (no matter which one). The entropy of X is the answer to the first question. And, since any sequence has the same probability of being chosen, we consider X uniformly distributed over its 1,296 values.

The average uncertainty in the unknown sequence is :

$$\boxed{H(X) = \log_2 1{,}296 = 10.34 \text{ bits}}$$

Another way to solve the problem consists in counting the needed number of bits to specify one ordered sequence.

There are four positions, and for each one, $\log_2 6$ bits are required to specify the colour. On the whole, we need $4 \times \log_2 6 = \log_2 6^4 = 10.34$ bits, which is identical to the previous result.

2. Let us represent the possible answers of player A by a discrete random variable Y. The reduction in the average uncertainty of the unknown sequence resolved by the answer given by player A is nothing but the mutual information between X and Y, which can be written as :

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

Since knowing X implies knowing Y (if the sequence chosen by player A is known, there is no doubt about the answer to be given by player A), we have :

$$H(Y/X) = 0$$

Consequently,

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

Let us evaluate the probability distribution of Y. First, we have to notice that player A cannot indicate that some pieces are in the wrong position. Accordingly, we have 5 possible answers according to the number of pieces in the correct position. Let us denote $\{Y = j\} = \{\text{j pieces are in the correct position}\}$.

- "four pieces are in the correct position"

This means that the unknown sequence is the submitted sequence. The corresponding probability is $P\{Y = 4\} = \frac{1}{1{,}296}$.

- "three pieces are in the correct position"

Let us suppose we have numbered the four different positions as 1, 2, 3, 4. If the three pieces in the right position are in position 1, 2 and 3, then there are $6 - 1 = 5$ different possible colours in position 4, which yields 5 possible sequences.

Proceeding this way for the three other possibilities according to the possible correct positions, we get :

$$P\{Y = 3\} = 4 \times \frac{5}{1{,}296} = \frac{20}{1{,}296}$$

Similar calculations lead to:

$$P\{Y = 2\} = \frac{150}{1{,}296}$$

$$P\{Y = 1\} = \frac{500}{1{,}296}$$

$$P\{Y=0\}=\frac{625}{1{,}296}$$

Eventually, we obtain :

$$H(Y)=-\frac{1}{1{,}296}\log_2\frac{1}{1{,}296}-\frac{20}{1{,}296}\log_2\frac{20}{1{,}296}-\frac{150}{1{,}296}\log_2\frac{150}{1{,}296}$$
$$-\frac{500}{1{,}296}\log_2\frac{500}{1{,}296}-\frac{625}{1{,}296}\log_2\frac{625}{1{,}296}$$

Then,

$$\boxed{I(X;Y)=H(Y)\approx 1\,\text{bit}}$$

SOURCE CODING

In this chapter, we will introduce the notion of entropy for an information source. We will be concerned with encoding the outcomes of a source so that we can recover the original data by using a minimum number of letters (for instance bits). This will lead us to study some elementary properties of codes.

The source coding theorem will exhibit the entropy of a source as the fundamental limit in data compression.

Coding procedures often used in practice as Huffman coding and the Lempel Ziv Welch algorithm, will also be described.

An **information source** is a device which delivers symbols (or letters) randomly from a set of symbols (or letters) called an alphabet. The successive symbols are chosen according to their probabilities in relation to the previous symbols.

The best examples are natural written languages such as the English language.

ENGLISH LANGUAGE

Considering a 27 symbol alphabet (26 letters and the space), Shannon studied different models of the English language. The simulations below are those of Shannon's original paper.

- **Zero-order letter model**

The symbols are chosen independently from each other and are equally likely. We can think of a box from which pieces of paper associated with letters are drawn. There are the same number of pieces of paper for each letter. A random drawing may yield a sequence as the following:

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD
QPAAMKBZAACIBZLHJQD
This compares to the result of monkeys strumming unintelligently on type writers.

- **First-order letter model**

The symbols are still independent and their numbers in the box are distributed according to their actual frequencies in the English language. The result may look like this:

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL

- **Second-order letter model**

The samples drawn consist of couples of letters. There are 27 boxes (A, B, …, Z, space), each box having pieces of paper associated with couples of letters with the same first letter (A, B, …, Z, space). The numbers of pieces of paper match the frequencies of the English language. For instance, in the box containing the couples beginning with “A”, the number of “AR” will be twice as great as the number of “AL” if we assume “AR” is twice as frequent as “AL” in English language.

The first letter of the sequence, let us say “O”, is drawn from the box described in the first-order letter model. Then, the next letter is obtained by drawing a piece of paper from the box containing the couples beginning with “O”. Let us suppose we got “ON”. We take out a couple from the “N” box; let us suppose “N space”, and so on…The result may appear as:

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

- **Third-order letter model**

We take into account the probabilities of units consisting of three successive letters, to obtain:

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE

We observe that English words begin to appear.
In the next stages, we jump to word units.

- **First-order word model**

The successive words are chosen independently from each other according to their frequencies in English language.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE

- **Second-order word model**

In addition to the previous conditions, we take into account the word transition probabilities.

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

The more sophisticated the model is, the more simulations will approach understandable English text. This illustrates that, although the sequence of letters or words in English is potentially random, certain sequences of words or letters are far more likely to occur than others, and that natural language may display transition probabilities that do not reveal themselves when monkeys strum on typewriters.

ENTROPY OF A SOURCE

In this course, we will limit ourselves to discrete stationary sources U, i.e. to discrete random process $\{U_1, U_2, ...\}$ (the successive outputs of the source U) taking on values in the same set of symbols and whose joint probability distributions are invariant under a translation of the time origin. The simplest case is the discrete memoryless source : $U_1, U_2, ...$ are independent random variables with the same probability distribution.

To take into account the memory of a source U (if the successive outputs of U are dependent), we define the entropy of U, $H_\infty(U)$, as :

$$H_\infty(U) = \lim_{L \to +\infty} H(U_L / U_{L-1}, U_{L-2}, ..., U_1)$$

Another way to estimate $H_\infty(U)$ consists in calculating the limit of $\frac{H(U_L, U_{L-1}, ..., U_1)}{L} = \frac{H_L(U)}{L}$ when L increases indefinitely. It turns out that this amounts to the same thing, since

$$\lim_{L \to +\infty} \frac{H_L(U)}{L} = \lim_{L \to +\infty} H(U_L / U_{L-1}, U_{L-2}, ..., U_1)$$

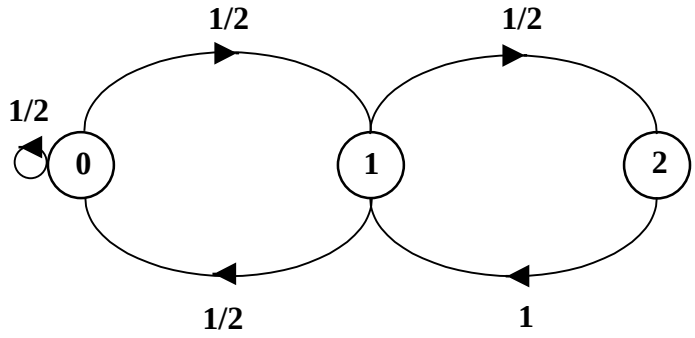
- An experiment carried out on the book “Jefferson the Virginian” by Dumas Malone resulted in 1.34 bit for the entropy of English language.

- In the special case of a memoryless source, we have $H_\infty(U) = H(U_L)$

- For a first-order Markov chain, $H_\infty(U) = H(U_L / U_{L-1})$

Example

Let us consider a Markov chain U taking on values in $\{0,1,2\}$ whose transition graph is sketched below :



The transition matrix is :

$$T = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

As there is only one class of recurrent states, U is stationary and the limiting-state probabilities x, y and z satisfy :

$(x, y, z) = (x, y, z) \times T \quad \Leftrightarrow \quad (1)$
with $x = P\{U = 0\}$ $y = P\{U = 1\}$ and $z = P\{U = 2\}$

Solving the equation (1) for x, y and z yields :

$$x = y = \frac{2}{5} \quad z = \frac{1}{5}$$

As the first row of the transition matrix corresponds to the probability distribution of U_L knowing $U_{L-1} = 0$, we get :

$$H(U_L / U_{L-1} = 0) = -\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2} = H_2\left(\frac{1}{2}\right) = 1\,\text{bit}$$

Proceeding the same way for the remaining two other rows gives :

$$H(U_L / U_{L-1} = 1) = -\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2} = H_2\left(\frac{1}{2}\right) = 1\,\text{bit}$$

$$H(U_L / U_{L-1} = 2) = 0\,\text{bit}$$

By applying the formula $H(U_L / U_{L-1}) = \sum_{i=0}^{2} P\{U_{L-1} = i\} \times H(U_L / U_{L-1} = i)$, we obtain :

$$H(U_L / U_{L-1}) = \frac{2}{5} \times 1 + \frac{2}{5} \times 1 + \frac{1}{5} \times 0 = \frac{4}{5} = 0.8\,\text{bit}$$

So, the entropy per symbol of U is :

$$\boxed{H_\infty(U) = 0.8\,\text{bit}}$$

Due to the memory of the source, this value (0.8 bit) is almost twice as small as the maximum entropy of a ternary memoryless source $(\log_2 3 = 1.585\,\text{bit})$.

Let U be an information source with memory (the successive outputs of U are dependent). Assuming U can only take on a finite number of values (N), we define the redundancy of U as:

$$\boxed{r = 1 - \frac{H_\infty(U)}{H_{MAX}} \quad \text{with} \quad H_{MAX} = \log N}$$

where $H_\infty(U)$ and H_{MAX} are expressed in the same unit (with the same base of logarithm).

- For a memoryless source U with equally likely values, there is no redundancy and $r = 0$;

- In the case of the English language, the first-order letter model leads to an entropy of $\log_2 27 = 4.75\,\text{bits}$. The estimated entropy being 1.34 bit, the redundancy is: $r = 1 - \frac{1.34}{4.75} \approx 72\%$. A possible interpretation is :

When choosing letters to write a comprehensible English text, approximately 28% of the letters can be extracted freely whereas the remaining 72% are dictated by the rules of structure of the language.

ENTROPY RATE

So far, we have been considering entropy per symbol of a source without taking into account the symbol rate of the source. However, the amount of information delivered by a source for a certain time depends on the symbol rate. This leads us to introduce the entropy rate of a source $H'(U)$ as :

$$\boxed{H'(U) = H_\infty(U) \times D_U}$$

where D_U is the symbol rate (in symbols per second)

Accordingly, $H'(U)$ may be interpreted as the average amount of information delivered by the source in one second. This quantity is useful when data are to be transmitted from a transmitter to a receiver over a communication channel.

THE SOURCE CODING PROBLEM

Example

Let U be a memoryless quaternary source taking values in {A, B, C, D} with probabilities ½, ¼, 1/8 and 1/8. 1,000 outputs of U are to be stored in the form of a file of binary digits, and one seeks to reduce the file to its smallest possible size.

First solution

There are $4 = 2^2$ symbols to be encoded. Thus, each of them can be associated with a word of two binary digits as follows:

$A \rightarrow$ "00"
$B \rightarrow$ "01"
$C \rightarrow$ "10"
$D \rightarrow$ "11"

All the codewords having the number of bits, i.e. the same length, this code is said to be a **fixed length code**.

The size of the file is : $1000 \times 2 = 2{,}000$ bits

2 bits are used to represent one symbol.

Second solution

The different symbols do not occur with the same probabilities. Therefore, we can think of a code which assigns shorter words to more frequent symbols as :

$A \rightarrow$ "1"
$B \rightarrow$ "01"
$C \rightarrow$ "000"
$D \rightarrow$ "001"

This code is said to be a **variable length code**, as the codewords do not have the same length.

From the weak law of large numbers, we deduce that in the sequence of 1,000 symbols, there are roughly :

$$1{,}000 \times \frac{1}{2} = 500 \text{ symbols of type "A"}$$
$$1{,}000 \times \frac{1}{4} = 250 \text{ symbols of type "B"}$$
$$1{,}000 \times \frac{1}{8} = 125 \text{ symbols of type "C"}$$
$$1{,}000 \times \frac{1}{8} = 125 \text{ symbols of type "D"}$$

Hence, the size of the file reduces to $500 \times 1 + 250 \times 2 + 125 \times 3 + 125 \times 3 = 1{,}750$ bits and is $\frac{2{,}000 - 1{,}750}{2{,}000} = 12.5\%$ smaller than it was in the previous solution, without loss of information (each symbol can be recovered reliably). The data have been compressed.

On average, $\frac{1{,}750}{1{,}000} = 1.75\,\text{bit}$ **are necessary to represent one symbol.**

If the symbol rate of U were 1,000 quaternary symbols per second, using the first code would result in a bit rate of 2,000 bits/sec. With the second code, the bit rate would reduce to 1,750 bits/sec.

We are now faced with three questions :

Given an information source,

- Is it possible to compress its data?

If so,

- What is the minimum average number of bits necessary to represent one symbol?
- How do we design algorithms to achieve effective compression of the data?

These three questions constitute **the source coding problem**.

Example

Let us continue the previous example with equally likely symbols A, B, C and D. We will show that there is no suitable code more efficient than the fixed length code related to “first solution”. By “more efficient, we mean that the average number of bits used to represent one symbol is smaller.

Let us consider a variable length code with :

n_A the length of the codeword associated with symbol "A"
n_B the length of the codeword associated with symbol "B"
n_C the length of the codeword associated with symbol "C"
n_D the length of the codeword associated with symbol "D"

With this code, the average number of bits used to represent one symbol in a sequence of n quaternary symbols, n being very large, is

$$\overline{n_1} = n_A \times \frac{n}{4} + n_B \times \frac{n}{4} + n_C \times \frac{n}{4} + n_D \times \frac{n}{4} = (n_A + n_B + n_C + n_D) \times \frac{n}{4}$$

As n is very large, the weak law of large numbers applies here.

By encoding this sequence with the fixed length code of the “first solution”, the average number of bits is

$$\overline{n_2} = 2 \times \frac{n}{4} + 2 \times \frac{n}{4} + 2 \times \frac{n}{4} + 2 \times \frac{n}{4} = (2 + 2 + 2 + 2) \times \frac{n}{4}$$

If we want to satisfy $\overline{n_1} < \overline{n_2}$, we can think of taking $n_A = 1$, $n_B = n_C = n_D = 2$. As the symbols are equally likely, it does not matter which one is chosen to have a codeword of length 1.

Assuming “1” (respectively “0”) is the codeword assigned to symbol “A”, as the codewords associated with B, C, D must be different (otherwise we could not distinguish between two different symbols), there is one of them which begins with “1” (respectively “0”).

For instance,

$A \rightarrow$ "1"
$B \rightarrow$ "10"
$C \rightarrow$ "00"
$D \rightarrow$ "01"

Let “10001” be an encoded sequence. Two interpretations are possible :

“ACD” or “BCA”

Such a code is not suitable to recover the data unambigously. Consequently, it is not possible to design a more efficient code than the one of fixed length 2. This is due to the fact that the probability distribution over the set of symbols {A, B, C, D} is uniform.

ELEMENTARY PROPERTIES OF CODES

A code C is a set of words c, called codewords, which result in the juxtaposition of symbols (letters) extracted from a code alphabet. We will denote b the size of the alphabet. The number of symbols $n(c)$ which comprise a codeword is its length.

The most common codes are binary codes, i.e. codes whose code alphabet is {0, 1}.

Example

In anticipation of the spread of communications and data processing technologies, the American Standard Association designed the **ASCII code** in 1963. ASCII stands for American Standard for Communication Information Interchange. Originally intended to represent the whole set of characters of a typewriter, it had to be used with teletypes, hence some special characters (the first ones listed in the table below) are now somewhat obscure. It consists of $2^7 = 128$ binary codewords having the same length (7).

Later on, additional and non printing characters were added to meet new demands. This gave birth to the extended ASCII code, a $2^8 = 256$ fixed length binary code whose the first 128 characters are common with the ASCII code.

Nowadays, keyboards still communicate to computers with ASCII codes and when saving document in “plain text”, characters are encoded with ASCII codes.

ASCII CODE TABLE

binary codes	characters	comments	binary codes	characters	comments
0000000	NUL	(Null char.)	1000000	@	(AT symbol)
0000001	SOH	(Start of Header)	1000001	A	
0000010	STX	(Start of Text)	1000010	B	
0000011	ETX	(End of Text)	1000011	C	
0000100	EOT	(End of Transmission)	1000100	D	
0000101	ENQ	(Enquiry)	1000101	E	
0000110	ACK	(Acknowledgment)	1000110	F	
0000111	BEL	(Bell)	1000111	G	
0001000	BS	(Backspace)	1001000	H	
0001001	HT	(Horizontal Tab)	1001001	I	
0001010	LF	(Line Feed)	1001010	J	
0001011	VT	(Vertical Tab)	1001011	K	
0001100	FF	(Form Feed)	1001100	L	
0001101	CR	(Carriage Return)	1001101	M	
0001110	SO	(Shift Out)	1001110	N	
0001111	SI	(Shift In)	1001111	O	
0010000	DLE	(Data Link Escape)	1010000	P	

binary codes	characters	comments	binary codes	characters	comments
0010001	DC1	(Device Control 1)	1010001	Q	
0010010	DC2	(Device Control 2)	1010010	R	
0010011	DC3	(Device Control 3)	1010011	S	
0010100	DC4	(Device Control 4)	1010100	T	
0010101	NAK	(Negative Acknowledgement)	1010101	U	
0010110	SYN	(Synchronous Idle)	1010110	V	
0010111	ETB	(End of Trans. Block)	1010111	W	
0011000	CAN	(Cancel)	1011000	X	
0011001	EM	(End of Medium)	1011001	Y	
0011010	SUB	(Substitute)	1011010	Z	
0011011	ESC	(Escape)	1011011	[	(left/opening bracket)
0011100	FS	(File Separator)	1011100	\	(back slash)
0011101	GS	(Group Separator)	1011101	]	(right/closing bracket)
0011110	RS	(Request to Send)(Record Separator)	1011110	^	(caret/cirumflex)
0011111	US	(Unit Separator)	1011111	_	(underscore)
0100000	SP	(Space)	1100000	‘	
0100001	!	(exclamation mark)	1100001	a	
0100010	"	(double quote)	1100010	b	
0100011	#	(number sign)	1100011	c	
0100100	$	(dollar sign)	1100100	d	
0100101	%	(percent)	1100101	e	
0100110	&	(ampersand)	1100110	f	
0100111	’	(single quote)	1100111	g	
0101000	(	(left/opening parenthesis)	1101000	h	
0101001	)	(right/closing parenthesis)	1101001	i	
0101010	*	(asterisk)	1101010	j	
0101011	+	(plus)	1101011	k	
0101100	,	(comma)	1101100	l	
0101101	-	(minus or dash)	1101101	m	
0101110	.	(dot)	1101110	n	
0101111	/	(forward slash)	1101111	o	
0110000	0		1110000	p	
0110001	1		1110001	q	
0110010	2		1110010	r	
0110011	3		1110011	s	
0110100	4		1110100	t	

binary codes	characters	comments	binary codes	characters	comments
0110101	5		1110101	u	
0110110	6		1110110	v	
0110111	7		1110111	w	
0111000	8		1111000	x	
0111001	9		1111001	y	
0111010	:	(colon)	1111010	z	
0111011		(semi-colon)	1111011	{	(left/opening brace)
0111100	<	(less than)	1111100	|	(vertical bar)
0111101	=	(equal sign)	1111101	}	(right/closing brace)
0111110	>	(greater than)	1111110	~	(tilde)
0111111	?	(question mark)	1111111	DEL	(delete)

(this table has been extracted from
http://www.neurophys.wisc.edu/www/comp/docs/ascii.html)

Example

Another famous code is the **Morse code**. Invented by Samuel Morse in the 1840's, it allows letters of the alphabet {a, b, …, z, "space", "full stop", "comma", …} to be sent as short electrical signals (dots) and long electrical signals (dashes).

There are different lapses of time between words, letters of a same word and dots and dashes within letters. Consequently the Morse code is a ternary code with code alphabet {**.**, **_**, **dit** (unit of time)}. The value of the unit of time depends on the speed of the operator.

- Within a letter, the space between code letters is equal to one dit.

- Between two characters in a word, the space is equal to three dits.

- The space between two words is equal to seven dits.

Morse code differs from ASCII code in the sense that shorter words are assigned to more frequent letters.

On April 15, 1912, the Titanic used the international distress call SOS "**…_ _ _…**" (sent in the correct way as one Morse symbol).

MORSE CODE TABLE

letters	Morse code	letters	Morse code
A	._	N	_.
B	_…	O	_ _ _
C	_._.	P	._ _.
D	_..	Q	_ _._
E	.	R	._.
F	.._.	S	…
G	_ _.	T	_
H	….	U	.._
I	..	V	…_
J	._ _ _	W	._ _
K	_._	X	_.._
L	._..	Y	_._ _
M	_ _	Z	_ _..

numbers	Morse code	numbers	Morse code
0	_ _ _ _ _	5	…..
1	._ _ _ _	6	_….
2	.._ _ _	7	_ _…
3	…_ _	8	_ _ _..
4	…._	9	_ _ _ _.

common punctuation	Morse code	common punctuation	Morse code
. (full stop)	._._._	- (hyphen)	_…_
, (comma)	_ _.._ _	/ (slash)	_.._.
? (question mark)	.._ _..		

special characters	Morse code
error	……..
+ (end of message)	._._.
@ (end of contact)	…_._
SOS (international distress call)	…_ _ _…

(source: http://www.wikipedia.org/wiki/Morse_code)

A code is said to be **uniquely decipherable** if any sequence of codewords can be interpreted in only one way.

Examples

- {1, 10, 11} is not uniquely decipherable as the sequence “1111” can be interpreted in “1” “11” “1” or “1” “1” “11” or …

- {1, 10} is uniquely decipherable although for any sequence, we need to consider two symbols at a time to decipher the successive codewords. In the sequence 11011, the first codeword is “1” since the following symbol is “1” whereas the second codeword is “10” since the third symbol is “0” and so on …

An **instantaneous code** is a code in which any sequence of codewords can be interpreted codeword by codeword, as soon as they are received.

Examples

- {0, 10} is instantaneous

- {1, 10} is not instantaneous. For instance, in the sequence 1110, we need to know whether the second symbol is “0” or “1” before interpreting the first symbol. This is due to the fact that a codeword (“1”) is the beginning of another codeword (“10”). It motivates the following definition.

A code is a **prefix code** if and only if no codeword is the beginning of another codeword.

Example

{1, 01, 000, 001} is a prefix code.

- A prefix code is an instantaneous code and the converse is true.

- A prefix code is uniquely decipherable but some uniquely decipherable codes do not have the prefix property.

Recovering the original codewords calls for designing uniquely decipherable codes. Kraft's theorem states the condition which the lengths of codewords must meet to be a prefix codes. It may seem restrictive to limit ourselves to prefix codes, as uniquely decipherable codes are not always prefix codes. However, McMillan's theorem will show us that we can limit our attention to prefix codes without loss of generality.

Kraft's theorem

There exists a b-ary (the size of the code alphabet is b) prefix code $\{c_1, c_2, ..., c_m\}$ with lengths $n(c_1), n(c_2), ..., n(c_m)$ if and only if :

$$\sum_{k=1}^{m} b^{-n(c_k)} \leq 1$$

This inequality is known as the **Kraft inequality**.

Example

Let us consider $C = \{10,11,000,101,111,1100,1101\}$. This code is not a prefix code as the codeword "11" is the beginning of the codewords "111", "1100" and "1101".

Nevertheless, it satisfies the Kraft inequality :

$$\sum_{c \in C} b^{-n(c)} = 2 \times 2^{-2} + 3 \times 2^{-3} + 2 \times 2^{-4} = \frac{1}{2} + \frac{3}{8} + \frac{2}{16} = 1$$

According to Kraft's theorem, there exists an equivalent binary prefix code with two codewords of length 2, three codewords of length 3 and two codewords of length 4.

To build such a code, we can use a tree made of a **root**, **nodes**, **branches** and **leaves**. A node at level i has one **parent** at level i-1 and at most b **children** at level i+1, but the root (level 0) has no parent.

- At level 0, there is only one node (the root)

- At level 1, there are at most b nodes

- At level 2, there are at most b^2 nodes

And so on …

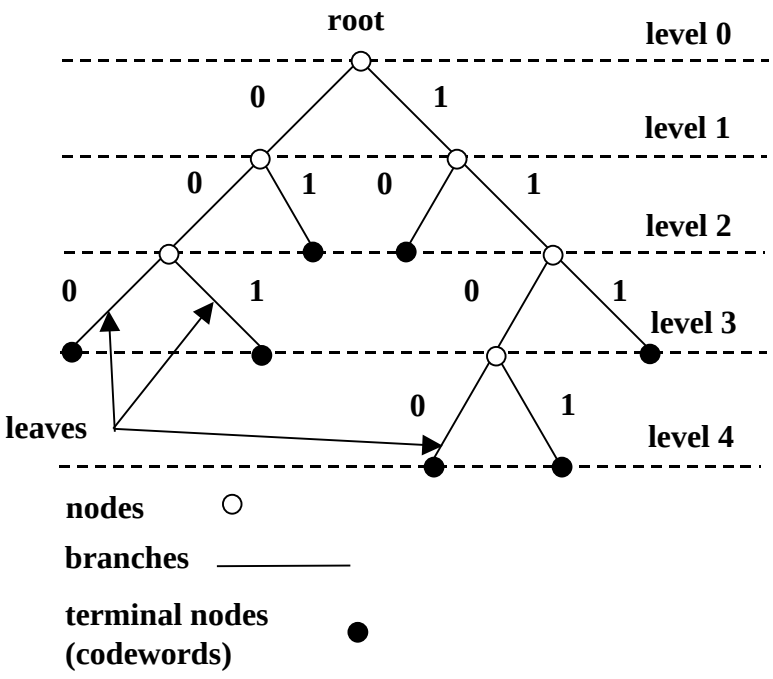
The terminal nodes (with no children) are called leaves.

A codeword is represented by a sequence of branches coming from different levels. Its length is equal to the level of its leaf.

To construct a prefix code, we have to make sure that no sequence of branches associated with a codeword is included in any sequence of branches associated with other codewords. In other words, no codeword is an ancestor of any other codeword.

In the example, the construction of a prefix code with a code tree requires that we construct

- two nodes at level 2 for the two codewords of length 2
- three nodes at level 3 for the three codewords of length 3
- two nodes at level 4 for the two codewords of length 4



Eventually we obtain the codewords listed in the table below :

codewords
01
10
000
001
111
1100
1101

McMillan's theorem

A uniquely decipherable code satisfies the Kraft inequality

Taking into account Kraft's theorem, this means that any uniquely decipherable code can be associated with an equivalent prefix code. By "equivalent", we mean "having the same length distribution".

SOURCE CODING THEOREM

This theorem states the limits which refer to the coding of the outputs of a source.

Let U be a stationary discrete source and b the size of the code alphabet.

To take into account the memory of U, we can consider the **L^{th} extension** of U. It is a source whose outputs are juxtapositions of L consecutive symbols delivered by U.

Example

U is a memoryless ternary source taking values in {0, 1, 2}. The second extension of U consists of the symbols taken two at a time, e.g., $\{00,01,02,10,11,12,20,21,22\}$.

Assuming the memory of U does not allow a "1" to follow a "0" and a "2", the second extension is : $\{00,02,10,11,12,20,22\}$ as the symbols "01" and "21" cannot occur.

To measure the ability of a code, applied to the L^{th} extension, to compress information, we define the average number of code symbols used to represent one source symbol as :

$$\overline{n(L)} = \frac{\overline{n_L}}{L} = \frac{\sum_i p_i n(c_i)}{L}$$

where c_i are the codewords assigned to the source words of the L^{th} extension of U and $p_i = P\{c_i\}$.
$\overline{n(L)}$ is also the average length of the codewords.

The smaller $\overline{n(L)}$ is, the more efficient the code is.

The **source coding theorem** consists of the two following statements :

- Any uniquely decipherable code used to encode the source words of the L^{th} extension of a stationary source U satisfies :

$$\overline{n(L)} = \frac{\overline{n_L}}{L} \geq \frac{H_L(U)}{\log b}$$

- It is possible to encode the source words of the L^{th} extension of a stationary source U with a prefix code in such way that :

$$\overline{n(L)} = \frac{\overline{n_L}}{L} < \frac{H_L(U)}{\log b} + \frac{1}{L},$$

where $H_L(U)$ and $\log b$ are expressed in the same base.

Comments :

If L tends to $+\infty$, the former inequality becomes :

$$\overline{n(\infty)} \geq \frac{H_\infty(U)}{\log b}$$

and as $H_L(U)$ is a decreasing function of L, $\frac{H_\infty(U)}{\log b}$ appears as the ultimate compression limit : we cannot find a uniquely decipherable code with $\overline{n(L)}$ smaller than $\frac{H_\infty(U)}{\log b}$.

This property provides a justification, a posteriori, of the definition of entropy. Expressing all logarithms in base 2 and taking $b = 2$, the entropy can be interpreted as the minimum average number of bits required to represent one source symbol.

$\forall \varepsilon > 0 \quad \exists L_0 \ / \ \forall L > L_0 \quad \frac{1}{L} < \varepsilon$, since $\lim_{L \to +\infty} \frac{1}{L} = 0$. Hence, for L large enough, we have :

$$\overline{n(L)} = \frac{\overline{n_L}}{L} < \frac{H_L(U)}{\log b} + \varepsilon$$

Taking the limits as L tends to $+\infty$, we obtain :

$$\overline{n(\infty)} < \frac{H_\infty(U)}{\log b} + \varepsilon$$

This means that we can achieve a prefix code to encode the outputs of U in such a way that the average number of code symbols is arbitrarily close to $\frac{H_\infty(U)}{\log b}$. This leads us to pose this question :

Is it possible to find a prefix code which satisfies $\overline{n(L)} = \frac{H_\infty(U)}{\log b}$?

Such a code exists, provided that the length of each codeword c_i is equal to the self-information of its occurrence, i.e., if $\forall i \quad n(c_i) = -\log_b P\{c_i\}$.

To meet this condition, $P\{c_i\}$ must be of the form b^{-m} with $m \in \text{IN}$, otherwise $-\log_b P\{c_i\}$ would not be an integer. The code is then said to be **optimum**.

Example

U is a memoryless source taking values in {A, B, C, D, E, F, G} with probabilities 1/3, 1/9, 1/9, 1/9, 1/9, 1/9, 1/9. The outputs of U are to be encoded with a ternary code alphabet {0, 1, 2}.

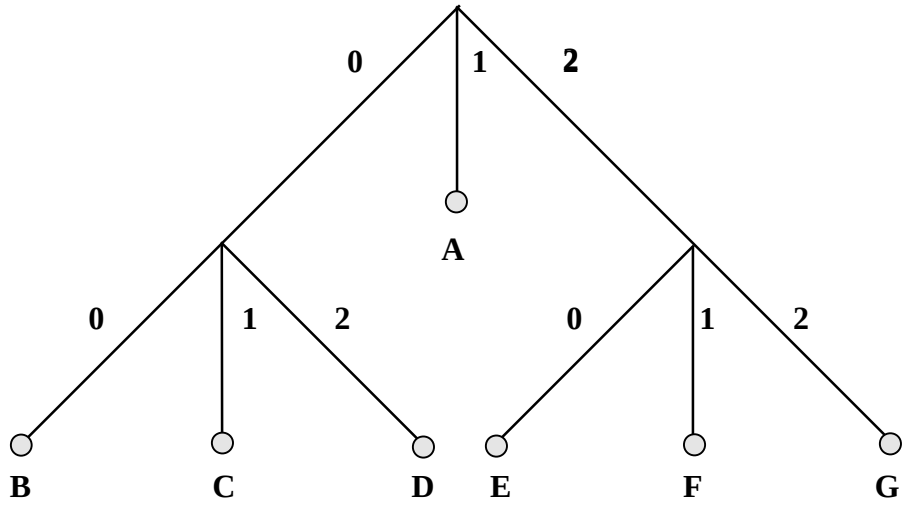
As the probabilities are negative powers of 3, which is the size of the code alphabet, we will assign codewords to the seven source symbols {A, B, C, D, E, F, G} in such a way that the length of a codeword is equal to the self-information (expressed in trits) associated with the corresponding source symbol.

source symbol	self-information (in trits)	length of the codeword
A	$-\log_3 \frac{1}{3} = 1$	**1**
B	$-\log_3 \frac{1}{9} = 2$	**2**
C	$-\log_3 \frac{1}{9} = 2$	**2**
D	$-\log_3 \frac{1}{9} = 2$	**2**
E	$-\log_3 \frac{1}{9} = 2$	**2**
F	$-\log_3 \frac{1}{9} = 2$	**2**
G	$-\log_3 \frac{1}{9} = 2$	**2**

Let us calculate $\sum_{c_i} b^{-n(c_i)}$:

$$\sum_{c_i} b^{-n(c_i)} = 3^{-1} + 6 \times 3^{-2} = 1$$

The Kraft inequality is satisfied. Consequently, there exists a prefix code with codewords having the length distribution $\{1,2,2,2,2,2,2\}$. To construct this code, we can use a ternary tree with nodes having three children as the size of the code alphabet is 3.



source symbols	codewords
A	**1**
B	**00**
C	**01**
D	**02**
E	**20**
F	**21**
G	**22**

The average length of the codewords is :

$$\bar{n} = 1 \times \frac{1}{3} + 6 \times 2 \times \frac{1}{9} = \frac{5}{3}$$

and the limit given by the source coding theorem :

$$\frac{H_1(U)}{\log_3 3} = H_\infty(U) = -\frac{1}{3}\log_3 \frac{1}{3} - 6 \times \frac{1}{9}\log_3 \frac{1}{9} = \frac{5}{3} \text{ trit}$$

(here we had to express $H_\infty(U)$ in trits, since logarithms must be expressed in the same base)

Hence, we have $\bar{n} = \frac{H_1(U)}{\log_3 3}$ and the code is optimum.

Let U be a source taking N different values. The number of b-ary symbols needed to represent one value is the smallest integer greater than or equal to $\log_b N$, denoted $\lceil \log_b N \rceil$, if we encode the outputs of U by a fixed length code. Let D_U be the symbol rate of U expressed in N-ary symbols per second. The source coding theorem states that we can find a prefix code whose the average b-ary symbols rate can be arbitrarily close to $D_U \times H_\infty(U)$ (logarithms are expressed in base b).

Using the fixed length code would result in a b-ary symbol rate equal to $D_U \times \lceil \log_b N \rceil$. If $H_\infty(U) < \log_b N$ (i.e. U has redundancy), then there exists a prefix code with a b-ary symbol rate smaller than $D_U \times \log_b N$, hence smaller than $D_U \times \lceil \log_b N \rceil$, since we can reduce the b-ary symbol rate as close to $D_U \times H_\infty(U)$ as desired.

Consequently, the source coding theorem answers the two first questions of the source coding problem :

Questions

Given an information source,

- Is it possible to compress its data?

If so,

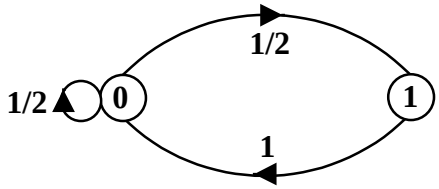
- What is the minimum average number of code symbols necessary to represent one source symbol?

Answers

- The compression, which results in a reduction in the symbol rate, is possible as long as $H_\infty(U) < \log_b N$.
- The minimum average number of code symbols required to represent one source symbol is $H_\infty(U)$.

Example

A binary source U is described by a Markov chain whose state transition graph is sketched below :



The transition matrix is :

$$\mathbf{T} = \begin{array}{c} \\ \mathbf{0} \\ \mathbf{1} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{0} & \mathbf{1} \end{array} \\ \begin{bmatrix} \mathbf{1/2} & \mathbf{1/2} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} \end{array}$$

There is only one class of recurrent states, hence U is stationary and the limiting state probabilities $x = P\{U = 0\}$ and $y = P\{U = 1\}$ satisfy :

$(x, y) = (x, y) \times T \quad \text{with} \quad x + y = 1$

$$\Leftrightarrow \begin{cases} x = \dfrac{x}{2} + y \\ y = \dfrac{x}{2} \\ x + y = 1 \end{cases}$$

Solving this system for x and y, we obtain :

$x = \frac{2}{3} \quad \text{and} \quad y = \frac{1}{3}$

Interpreting the two rows of the transition matrix, we get :

$H(U_n / U_{n-1} = 0) = H_2\left(\frac{1}{2}\right) = 1 \text{ bit}$

$H(U_n / U_{n-1} = 1) = H_2(1) = 0 \text{ bit}$

Eventually, we have :

$H_\infty(U) = H(U_n / U_{n-1}) = \frac{2}{3} \times 1 + \frac{1}{3} \times 0 = \frac{2}{3} = 0.66 \text{ bit}$

The maximum entropy for a binary source is $\log_2 2 = 1 \text{ bit}$. As $H_\infty(U) = 0.66 \text{ bit} < 1 \text{ bit}$, U has redundancy and its data can be compressed.

To take into account the memory of U, let us consider its 2nd extension which consists of the source words : “00”, “01”, “10”. “11” is not listed, since a “1” cannot follow a “1”.

Their probabilities are :

$P\{\text{"00"}\} = P\{U_{n-1} = 0 \cap U_n = 0\} = P\{U_n = 0 / U_{n-1} = 0\} \times P\{U_{n-1} = 0\} = \frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$

$P\{\text{"01"}\} = P\{U_{n-1} = 0 \cap U_n = 1\} = P\{U_n = 1 / U_{n-1} = 0\} \times P\{U_{n-1} = 0\} = \frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$

$$P\{"10"\} = P\{U_{n-1} = 1 \cap U_n = 0\} = P\{U_n = 0 / U_{n-1} = 1\} \times P\{U_{n-1} = 1\} = 1 \times \frac{1}{3} = \frac{1}{3}$$

With a **ternary code alphabet** {0, 1, 2}, we can construct the following code :

$$\begin{cases} "00" & \rightarrow & 0 \\ "01" & \rightarrow & 1 \\ "10" & \rightarrow & 2 \end{cases}$$

Then, we have $\overline{n_2} = 1$ and $\overline{n(2)} = \frac{1}{2}$.

One has to be cautious here as the second extension of U is not memoryless, since “10” cannot follow “01”. Consequently, although the distribution probability is uniform $\left(P\{"00"\} = P\{"01"\} = P\{"10"\}\right)$, the entropy of the second extension of U is smaller than 1 trit and is not equal to $\overline{n_2}$: the code is not optimum.

With a **binary code alphabet** {0, 1}, we can think of this prefix code :

$$\begin{cases} "00" & \rightarrow & 0 \\ "01" & \rightarrow & 10 \\ "10" & \rightarrow & 11 \end{cases}$$

In this case, the average number of bits necessary to represent one source symbol is :

$$\overline{n(2)} = \frac{\overline{n_2}}{2} = \frac{1}{2}\left\{2 \times \left(\frac{1}{3} + \frac{1}{3}\right) + 1 \times \frac{1}{3}\right\} = 0.83$$

Using this code results in a reduction of the source symbol rate equal to $1 - 0.83 = 17\%$, the maximum being $1 - 0.66 = 34\%$.

COMPRESSION ALGORITHMS

This section develops a systematic construction of binary codes compressing the data of a source.

SHANNON-FANO ALGORITHM

The Shannon-Fano encoding scheme is based on the principle that each code bit, which can be described by a random variable, must have a maximum entropy.

First step

We have to list the symbols, for instance from top to bottom, in order of decreasing probability.

Second step

We divide the whole set of source symbols into two subsets, each one containing only consecutive symbols of the list, in such way that the two probabilities of the subsets are as close as possible. Then, we assign "1" (respectively "0") to the symbols of the top (respectively bottom) subset.

Third step

We apply the process of the previous step to the subsets containing at least two symbols. The algorithm ends when there are only subsets with one symbol left.

The successive binary digits assigned to the subsets have to be arranged from left to right to form the codewords. This amounts to constructing a binary tree from the root to the leaves.

We should note that the Shannon-Fano encoding scheme does not always provide the best code, as the optimisation is achieved binary digit by binary digit, but not on the whole of the digits which constitute the codewords.

Example

Let U be a memoryless source taking values in $\{A, B, C, D, E, F, G\}$ with the probabilities $\{0.4, 0.2, 0.15, 0.1, 0.05, 0.05, 0.05\}$ respectively.

The entropy of U is :

$$H_\infty(U) = -0.4 \times \log_2 0.4 - 0.2 \times \log_2 0.2 - 0.15 \times \log_2 0.15 - 0.1 \times \log_2 0.1 - 3 \times 0.05 \times \log_2 0.05$$

$$H_\infty(U) \approx 2.38 \text{ bits}$$

The maximum entropy of a source taking on 7 values is $\log_2 7 \approx 2.81$ bits

Consequently, U has redundancy and its data can be compressed.

With a fixed length code

The length n has to be chosen as the smallest integer satisfying :

$2^n \geq 7$

We obtain $n = 3$

With Shannon-Fano code

Let us display the successive steps of Shannon-Fano encoding in the table below :

symbols	probabilities	1st step	2nd step	3rd step	4th step	5th step	6th step	codewords
A	**0.4**	**1**	**1**					**11**
B	**0.2**	**1**	**0**					**10**
C	**0.15**	**0**		**1**	**1**			**011**
D	**0.1**	**0**		**1**	**0**			**010**
E	**0.05**	**0**		**0**		**1**	**1**	**0011**
F	**0.05**	**0**		**0**		**1**	**0**	**0010**
G	**0.05**	**0**		**0**		**0**		**000**

The average number of bits required to represent one source symbol is :

$\bar{n} = 2 \times (0.4 + 0.2) + 3 \times (0.15 + 0.1 + 0.05) + 4 \times (0.05 + 0.05) = 2.5$

Compared to the 3 fixed length code, this Shannon-Fano code results in a reduction in the symbol rate of $\frac{3 - 2.5}{3} \approx 16\%$.

HUFFMAN ALGORITHM

This algorithm, invented in 1952 by D.A. Huffman, provides a prefix code whose construction can be achieved by a binary tree.

Here are the successive steps :

First step

We arrange the source symbols on a row in order of increasing probability from left to right.

Second step

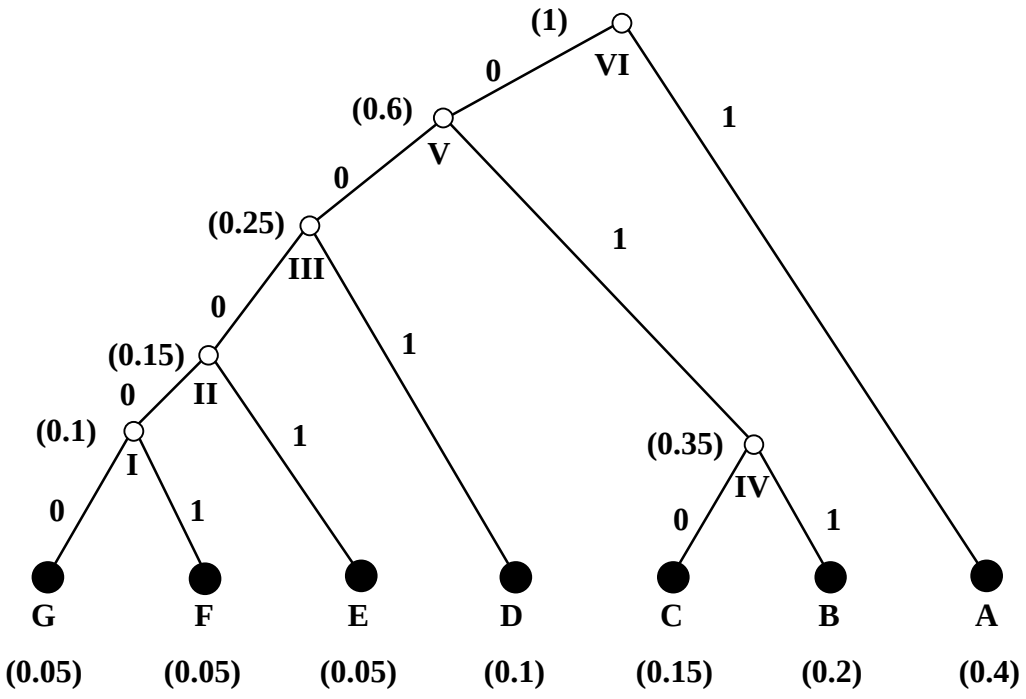
Let us denote A and B the two source symbols of lowest probabilities P_A and P_B in the list of the source words. We combine A and B together with two branches into a node which replaces A and B with probability assignment equal to $P_A + P_B$. A and B are removed from the list and replaced by the node.

Third step

We apply the procedure of the second step until the probability assignment is equal to 1. Then, the corresponding node is the root of the binary tree.

Example

Let us return to the source of the previous example. Applying the above algorithm results in the following binary tree :



symbols	probabilities	codewords
A	**0.4**	**1**
B	**0.2**	**011**
C	**0.15**	**010**
D	**0.1**	**001**
E	**0.05**	**0001**
F	**0.05**	**00001**
G	**0.05**	**00000**

The average length codewords is :

$$\bar{n} = 5\times(0.05+0.05)+4\times 0.05+3\times(0.1+0.15+0.2)+1\times 0.4 = 2.45$$

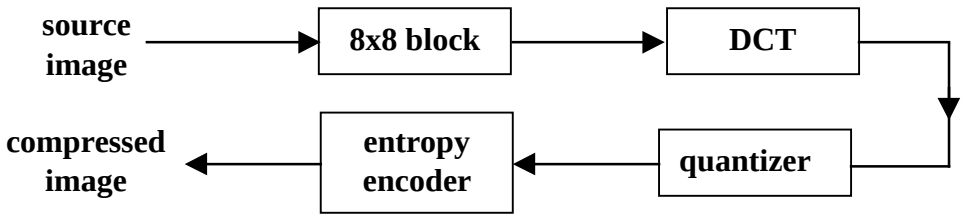
Comments

- When the source words to be encoded have the same length, the Huffman code is the most efficient among the uniquely decipherable codes.

- According to the previous comment, a Huffman code always satisfies the conditions stated in the source coding theorem.

- Applying the Shannon-Fano or Huffman algorithms requires knowledge of the probability distribution of the source words. In practice, the probabilities of the source words are unknown but, as a result of the weak law of large numbers, they may be estimated by the relative frequency of the source word outcomes in the message. As the receiver does not know these values, they have to be sent with the encoded data to allow the message to be decoded. Consequently, the efficiency of the code will be reduced.

Huffman coding is implemented in the Joint Photographic Experts Group standard to compress images. The algorithm can be sketched as follows :

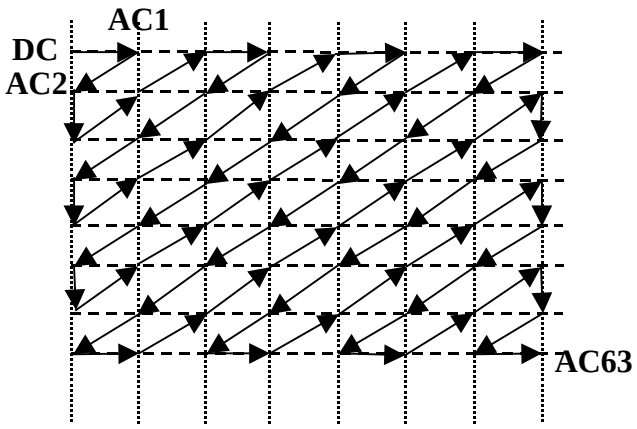


- The source image is divided into 8x8 pixel input blocks.

- Each input block can be regarded as a function of the two spatial dimensions x and y. Calculating the Discrete Cosine Transform, which is similar to the Fourier Transform, results in an 8x8 output block containing 64 elements arranged in rows and columns. The term located in the top left-hand corner is called the “DC” coefficient and the remaining 63, the “AC” coefficients. The “DC” coefficient is a measure of the average value of the 64 pixels of the input block.

- The elements of the 8x8 output blocks are quantized with a number of bits according to their locations in the block : more bits will be allocated to elements near the top left-hand corner. Quantization is lossy.

- After quantization, “DC” coefficients are encoded by difference, as there is often strong correlation between “DC” coefficients of adjacent 8x8 blocks. To facilitate the entropy coding procedure, the “AC” coefficients are ordered into a “zigzag” sequence as shown below :



Then, each nonzero coefficient is represented by two symbols : symbol-1 and symbol-2.

Symbol-1 consists of two numbers :

- the number of consecutive zero coefficients in the zigzag sequence preceding the nonzero coefficient to be encoded (RUNLENGTH).
- The number of bits used to encode the value of the amplitude of the nonzero coefficient (SIZE)

Symbol-2 is a signed integer equal to the amplitude of the nonzero coefficient (AMPLITUDE).

If there are more than 15 consecutive zero coefficients, symbol-1 is represented by (15, 0).

For both “DC” and “AC” coefficients, symbol-1 is encoded with a Huffman code, whereas symbol-2 is encoded by a variable length integer code whose codewords lengths (in bits) must satisfy :

codeword length (in bits)	amplitude
1	**-1,1**
2	**-3,-2,2,3**
3	**-7,…,-4,4,…,7**
4	**-15,…,-8,8,…,15**
5	**-31,…,-16,16,…,31**
6	**-63,…,-32,32,…,63**
7	**-127,…,-64,64,…,127**
8	**-255,…,-128,128,…,255**
9	**-511,…,-256,256,…,511**
10	**-1023,…-512,512,…,1023**

(source :
The JPEG Still Picture Compression Standard
Gregory K. Wallace
Multimedia Engineering
Digital Equipment Corporation
Maynard, Massachussetts)

LZ 78 ALGORITHM

In 1978, Jacob Ziv and Abraham Lempel wrote an article entitled “Compression of Individual Sequences via Variable Rate Coding” in the IEEE Transactions on Information Theory describing a compression algorithm known as the LZ 78 algorithm.

It consists of constructing a dictionary as the message is being read, character by character. At the beginning, the only string in the dictionary is the empty string “” in position 0.

If the juxtaposition $P \oplus c$ of the preceding string P with the last read character c is in the dictionary, the algorithm passes to the next character.
Otherwise, the couple (position of P in the dictionary, character c) is sent and the string $P \oplus c$ is added to the dictionary.

Example

The message to be transmitted is the tongue twister :

IF^STU^CHEWS^SHOES,^SHOULD^STU^CHOOSE^THE^SHOES^HE^CHEWS^ ?

("^" represents the space).

The successive steps of LZ 78 algorithm are described in the table below :

position in the dictionary	string in the dictionary	read characters	emitted couple
0	""	I	(0,I)
1	I	F	(0,F)
2	F	^	(0,^)
3	^	S	(0,S)
4	S	T	(0,T)
5	T	U	(0,U)
6	U	^C	(3,C)
7	^C	H	(0,H)
8	H	E	(0,E)
9	E	W	(0,W)
10	W	S^	(4,^)
11	S^	SH	(4,H)
12	SH	O	(0,O)
13	HO	ES	(9,S)
14	ES	,	(0,,)
15	,	^S	(3,S)
16	^S	HO	(8,O)
17	HO	L	(0,L)
18	L	D	(0,D)
19	D	^ST	(16,T)
20	^ST	U^	(6,^)
21	U^	C	(0,C)
22	C	HOO	(17,O)
23	HOO	SE	(4,E)
24	SE	^T	(3,T)
25	^T	HE	(8,E)
26	HE	^SH	(16,H)
27	^SH	O	(0,O)
28	O	ES^	(14,^)

position in the dictionary	string in the dictionary	read characters	emitted couple
29	**ES^**	**HE^**	**(26,^)**
30	**HE^**	**CH**	**(22,H)**
31	**CH**	**EW**	**(9,W)**
32	**EW**	**S^?**	**(11,?)**

If string positions are encoded with one byte, then the dictionary will contain $2^8 = 128$ strings, which is not enough. With two bytes to represent string positions, $2^8 \times 2^8 = 65536$ strings may be strored in the dictionary. If the symbols are transmitted with extended ASCII, the number of bytes used to encode the message is :

$33 \times (2+1) = 99$ Bytes

Transmitting the message by encoding the successive letters into extended ASCII code would result in a file containing 60 Bytes.

The longer the message, the more efficient the algorithm.

LZW ALGORITHM

In 1984, Terry Welch published “A technique for High performance Data Compression” in IEEE Computer. The algorithm described in this paper is an improvement of LZ 78, and it is now called the LZW algorithm.

In LZW,

- Before starting the algorithm, the dictionary contains all the strings of length one,

- As soon as the juxtaposition $P \oplus c$ is not in the dictionary, only the address of P is transmitted. Then, the string $P \oplus c$ is added to the dictionary and the character c is used to initialise the next string.

Example

Let us resume the message :

IF^STU^CHEWS^SHOES,^SHOULD^STU^CHOOSE^THE^SHOES^HE^CHEWS^?

By using the ASCII code table (page 30) to initialise the dictionary and applying the algorithm, we obtain :

position in the dictionary	string in the dictionary	initialisation of P	read characters	emitted position
0	Nul character			
…	…			
…	…			
8	Backspace			
…	…			
…	…			
44	coma			
…	…			
…	…			
67	C			
68	D			
69	E			
70	F			
…	…			
72	H			
73	I			
…	…			
…	…			
76	L			
…	…			
…	…			
79	O			
…	…			
…	…			
83	S			
84	T			
85	U			
…	…			
87	W			
…	…			
…	…			
255				
256	IF	I	F	73
257	F^	F	^	70
258	^S	^	S	8
259	ST	S	T	83
260	TU	T	U	84
261	^C	^	C	8

position in the dictionary	string in the dictionary	initialisation of P	read characters	emitted position
262	CH	C	H	67
263	HE	H	E	72
264	EW	E	W	69
265	WS	W	S	87
266	^SH	^	SH	258
267	HO	H	O	72
268	OE	O	E	79
269	ES	E	S	69
270	S,	S	,	83
271	,^	,	^	44
272	^SHO	^	SHO	266
273	OU	O	U	79
274	UL	U	L	85
275	LD	L	D	76
276	D^	D	^	68
277	^ST	^	ST	8
278	TU^	T	U^	84
279	^CH	^	CH	8
280	HOO	H	OO	72
281	OS	O	S	79
282	SE	S	E	83
283	E^	E	^	69
284	^T	^	T	8
285	TH	T	H	84
286	HE^	H	E^	72
287	^SHOE	^	SHOE	8
288	ES^	E	S^	68
289	^H	^	H	8
290	HE^C	H	E^C	72
291	CHE	C	HE	67
292	EWS	E	WS	69
293	S^	S	^	83
294	^?	^	?	8

The indexes in the dictionary may be coded with a fixed number of bits, but the algorithm is more efficient with a variable number of bits : at the beginning 9 bits are used until 256 entries are added, then 10 bits, 11 bits and so on …

To compare the performances of some codes, the compression rates have been calculated after applying different algorithms to the same 6MB set of files divided into three parts :

- text files
- binary files
- graphic files

$$\text{compression rate} = 1 - \frac{\text{size of the compressed file}}{\text{size of the original file}}$$

These files have been extracted from Dr Dobb's journal February 1991 (source Mark Nelson).

codes	graphic	binary	text	on average
Huffman	**27.22%**	**24.79%**	**40.38%**	**31.04%**
adaptative Huffman	**32.59%**	**26.69%**	**40.72%**	**33.27%**
LZW (fixed 12 bits)	**20.61%**	**15.07%**	**50.32%**	**29.20%**
LZW (variable 12 bits)	**46.78%**	**36.61%**	**54.82%**	**45.81%**
LZW (variable 15 bits)	**48.44%**	**36.15%**	**58.28%**	**47.31%**

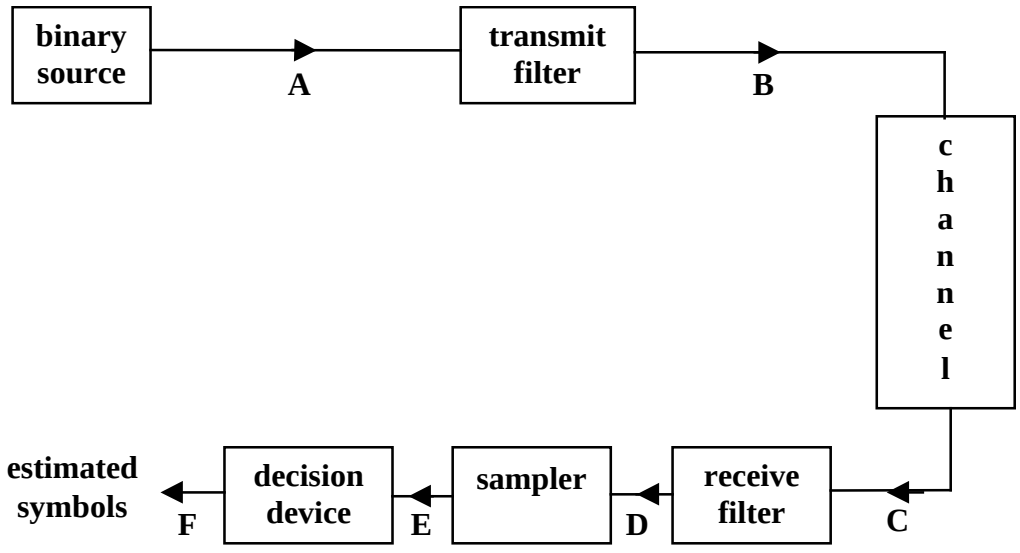
COMMUNICATION CHANNELS

This chapter deals with the transmission of information. First, we will consider the transfer of data in terms of information theory. Then, we will state the noisy channel theorem.

CHANNEL CAPACITY

Example

Let us consider a baseband digital communication system :



- binary source

A binary memoryless source with alphabet $\{-V, V\}$ (the symbols are equally likely).

- transmit filter

Its transfer function determines the shape of the power spectrum of the signal to transmit.

- channel

Optical fibres, pairs of wires, coaxial cables are channels used to link a transmitter to a distant receiver.

- receive filter

Used to select the bandwidth of the transmitted signal (generally matched to the transmit filter to optimise the signal to noise ratio).

- sampler

Converts the filtered received signal to a discrete time signal at a sample rate equal to the baud rate of the symbols emitted by the source.

- decision device

As the symbols are symmetric, equally likely and the noise process has an even probability density, the decision device compares the input sample amplitude to the threshold "0" : if it is greater (respectively smaller) than "0", the estimated symbol is V (respectively –V). We will justify this decision rule further.

If we consider the devices between A and E, the resulting device is a channel with a discrete input and a continuous output.

Between B and E, the input is continuous and the output continuous.

So, the nature, discrete or continuous, of the input and output of a channel depends on the devices it includes.

Let us denote a_k the random variable such as :

$$\{a_k = V\} = \{\text{the k}^{\text{th}} \text{symbol emitted by the source is V}\}$$
$$\{a_k = -V\} = \{\text{the k}^{\text{th}} \text{symbol emitted by the source is - V}\}$$

and B(t) the zero mean Gaussian random variable modelling the noise (its power spectrum density is $\frac{N_0}{2}$ for any value of the frequency).

Let $\frac{1}{\sqrt{T}}\Pi_T(t)$ be the impulse response of the transmit and receive filters.

Considering the channel as a perfect channel (i.e. the impulse response is $\delta(t)$), we obtain the signal in D :

$$Y_D = \left(\sum_k a_k \frac{1}{\sqrt{T}} \Pi_T (t - kT) + B(t) \right) * \frac{1}{\sqrt{T}} \Pi_T (t)$$

$$Y_D = \left(\sum_k a_k \left(\delta(t - kT) * \frac{1}{\sqrt{T}} \Pi_T (t) \right) + B(t) \right) * \frac{1}{\sqrt{T}} \Pi_T (t)$$

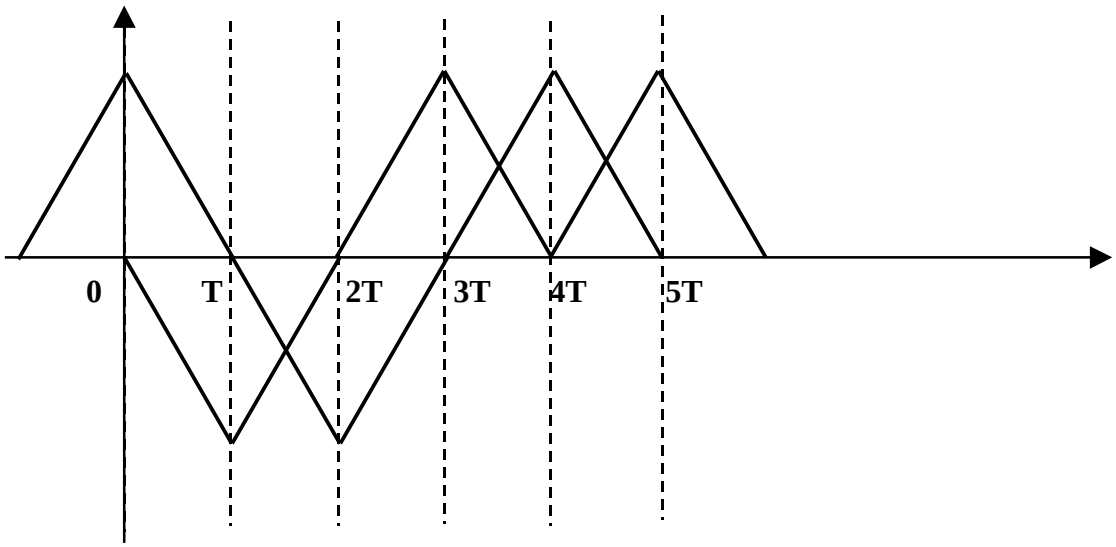
$$Y_D = \sum_k a_k \delta(t - kT) * \left(\frac{1}{\sqrt{T}} \Pi_T (t) * \frac{1}{\sqrt{T}} \Pi_T (t) \right) + B(t) * \frac{1}{\sqrt{T}} \Pi_T (t)$$

$$Y_D = \sum_k a_k \delta(t - kT) * (\Lambda_{2T}(t)) + B(t) * \frac{1}{\sqrt{T}} \Pi_T (t)$$

$$Y_D = \sum_k a_k \Lambda_{2T} (t - kT) + B(t) * \frac{1}{\sqrt{T}} \Pi_T (t)$$

In order not to have intersymbol interference, we have to sample at $(nT)_{n \in Z}$, as shown in the figure below :

$$a_0 = V, a_1 = -V, a_2 = -V, a_3 = V, a_4 = V, a_5 = V$$



Let B'(t) be $B(t) * \frac{1}{\sqrt{T}}\Pi_T(t)$ and $H_r(f)$ the transfer function of the receive filter. B'(t) is gaussian, its mean is zero and its variance σ^2 can be calculated as the power :

$$\sigma^2 = \int_{-\infty}^{+\infty} S_{B'}(f)df = \int_{-\infty}^{+\infty} S_B(f)|H_r(f)|^2 df = \frac{N_0}{2}\int_{-\infty}^{+\infty}|H_r(f)|^2 df = \frac{N_0}{2}\int_{-\infty}^{+\infty}\left|\frac{1}{\sqrt{T}}\Pi_T(t)\right|^2 df = \frac{N_0}{2}$$

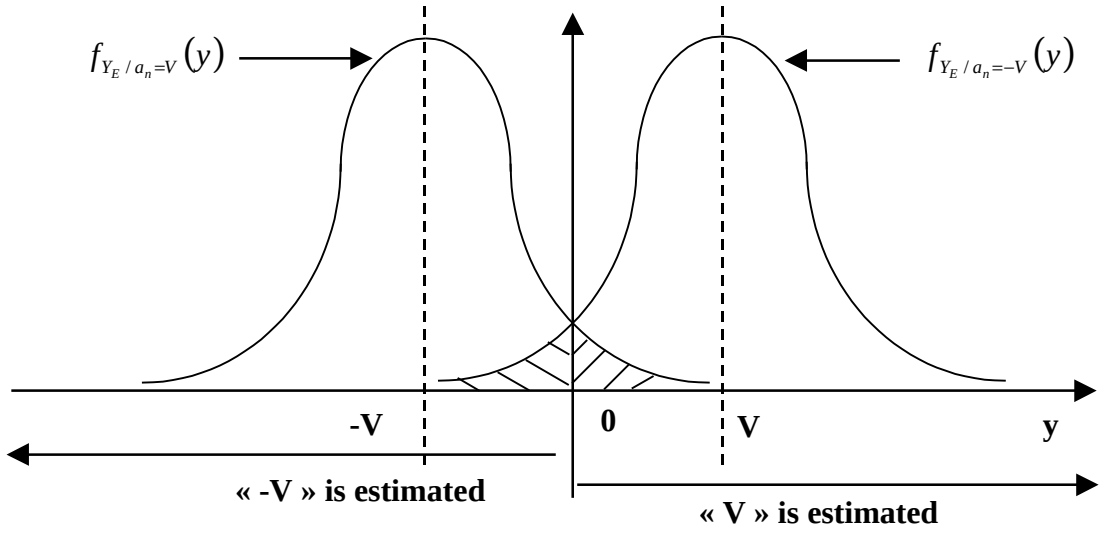
After sampling at nT, we obtain :

$$\boxed{Y_E = a_n + B'(nT)}$$

Then, we can think of a decision rule consisting in choosing the more likely of the two hypothesis $(a_n = -V$ or $a_n = V)$ based on the observation (Y_E). This is known as the maximum likelihood decision. In other terms, we have to compare the probability densities of Y_E knowing a_n :

$$\boxed{\text{if } \quad f_{Y_E/a_n=-V}(y) > (\text{respectively} <) f_{Y_E/a_n=V}(y), \text{ then the estimated symbol is - V (respectively V)}}$$

As $f_{Y_E/a_n=-V}(y)$ (respectively $f_{Y_E/a_n=V}(y)$) is a Gaussian random variable with mean –V (respectively V) and variance $\frac{N_0}{2}$, we have

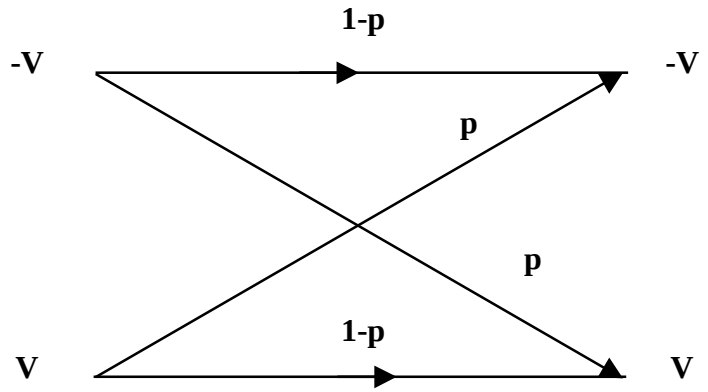


$$P\{\text{"}V\text{" is estimated / "}-V\text{" is sent}\} = P\left\{N\left(-V;\frac{N_0}{2}\right) > 0\right\} = P\left\{N(0;1) > \frac{2V}{N_0}\right\} = p$$

And, by symmetry :

$$P\{\text{"}-V\text{" is estimated / "}V\text{" is sent}\} = P\{\text{"}V\text{" is estimated / "}-V\text{" is sent}\} = p$$

Then, the model corresponding to the transmission chain between A and F can be sketched as follows :



Such a channel is called a Binary Symmetric Channel with error probability p.

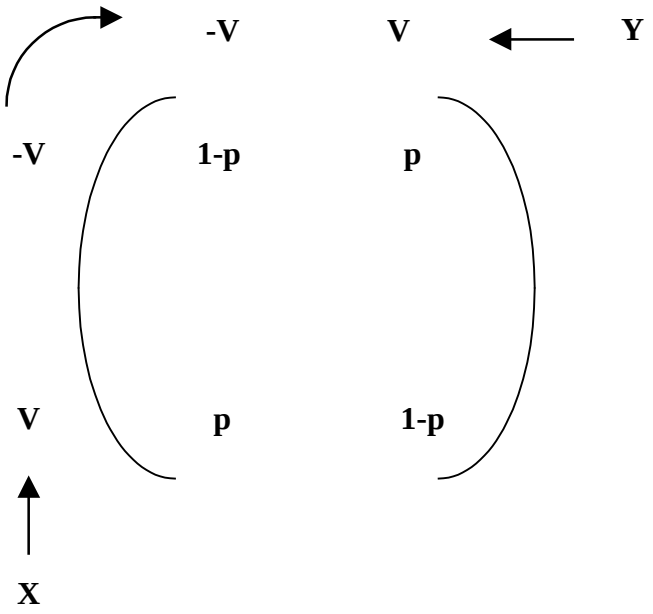
A channel can be specified by an input alphabet $\{x_1, x_2, ..., x_n\}$, an output alphabet $\{y_1, y_2, ..., y_m\}$ and a transition probability distribution :
$p_{ij} = P\{Y = y_j / X = x_i\} \quad \forall (i, j) \in [1, n] \times [1, m]$

In this course, we will limit ourselves to discrete memoryless channels, i.e., channels whose input and output alphabets are finite and for which the output symbol at a certain time depends statistically only on the most recent input symbol.

The transition probability distribution can be expressed as a transition probability matrix.

Example

Returning to the preceding example, we have the transition probability matrix



As we will attempt to recover the input symbol from the output symbol, we can consider the average mutual information between X and Y :

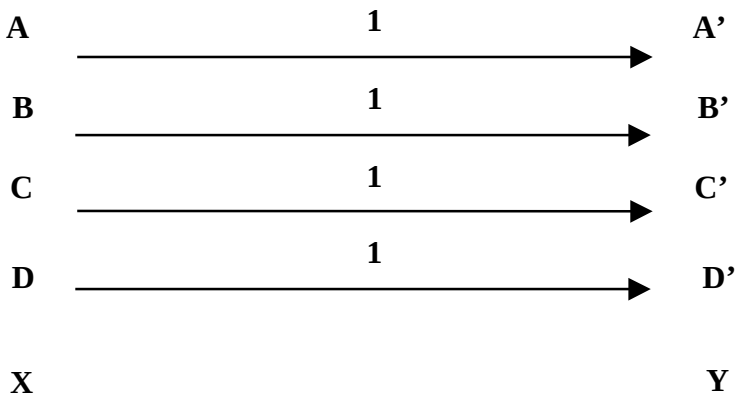
$$I(X;Y) = H(X) - H(X/Y) = H(Y) - H(Y/X)$$

This quantity, $I(X;Y)$, depends on the input probability distribution $p(X)$. Accordingly, it is not intrinsic to the channel itself. Thus, we define the capacity C as follows :

$$\boxed{C = \underset{p(X)}{Max}\, I(X;Y)}$$

Examples

A noiseless channel



We have :

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

The occurrence of Y uniquely specifies the input X. Consequently, $H(X/Y) = 0$ and

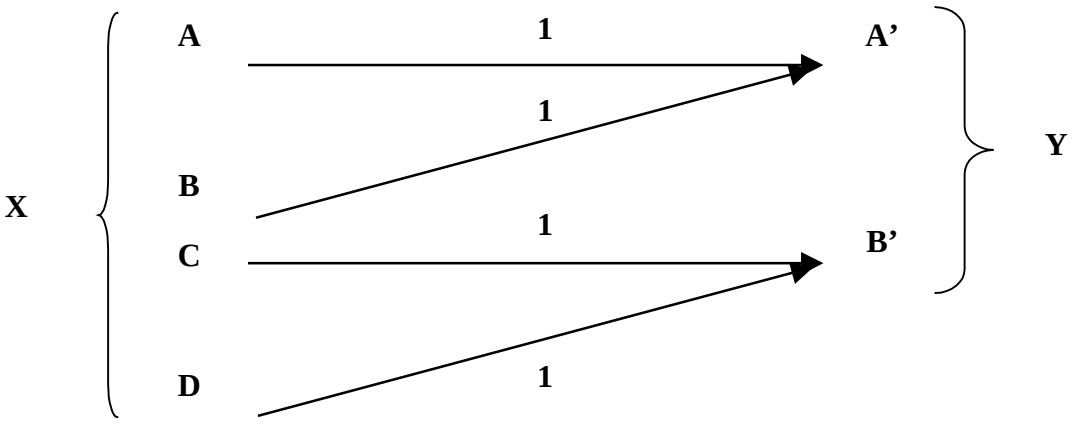
$$C = \underset{p(X)}{Max}\, H(X)$$

As X is a random variable taking on 4 different values, the maximum of $H(X)$ is $(\log_2 4)$ bits. This value is achieved for a uniform probability distribution on the input alphabet.

Finally, we get :

$$\boxed{C = \log_2 4 = 2 \text{ bits}}$$

A noisy channel



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

In this case, the input value uniquely determines the output value. Accordingly, knowing X, there is no uncertainty on Y. Then, we have :

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

And :

$$C = \underset{p(X)}{Max}\, H(Y)$$

Here, we could think of $C = 1\,\text{bit}$ as the maximum entropy of Y should be 1 bit. However, we are not sure there exists an input probability distribution such as the corresponding output probability distribution is uniform. Thus, we have to carry out the following calculations :

Let us denote :

$$p_A = P\{X = A\}$$
$$p_B = P\{X = B\}$$
$$p_C = P\{X = C\}$$
$$p_D = P\{X = D\}$$

Considering the possible transitions from the input to the output, we have :

$$P\{Y = A'\} = P\{X = A \cap Y = A'\} + P\{X = B \cap Y = A'\}$$

$$P\{Y = A'\} = P\{X = A\} \times P\{Y = A'/X = A\} + P\{X = B\} \times P\{Y = A'/X = B\}$$

$$P\{Y = A'\} = p_A + p_B$$

As Y can only take two values, we deduce :

$$P\{Y = B'\} = 1 - (p_A + p_B)$$

And :

$$H(Y) = H_2(p_A + p_B)$$

The maximum of $H_2(p_A + p_B)$, 1 bit, is achieved when $p_A + p_B = \frac{1}{2}$.

Thus, the capacity is 1 bit.

Computing the capacity of a channel may be tricky as it consists of finding the maximum of a function of $(n-1)$ variables if the input alphabet contains n symbols.

Nevertheless, when a channel is symmetric (to be defined below), there is no difficulty in calculating the capacity.

Definition

A channel is said to be **symmetric** if the set of output symbols can be partitioned into subsets in such a way that for each subset, the probability transition matrix has the following two properties

- each row is a permutation of each remaining row,
- each column (if there are more than one) is a permutation of each remaining column.

Comment

Usually the probability transition matrices related to the subsets are not stochastic matrices as some output symbols are missing. By stochastic matrix, we mean a matrix for which each row sum equals 1.

Example 1

Let us consider a channel with the probability transition matrix :

	D	**E**	**F**	**G**	**Y**
A	**0.1**	**0.5**	**0.1**	**0.3**	
B	**0.5**	**0.1**	**0.1**	**0.3**	
C	**0.1**	**0.1**	**0.5**	**0.3**	
X					

The output symbols {D, E, F, G} can be partitioned into two subsets {D, E, F} and {G}. The two probability transition matrices are :

$$T_1 = \begin{pmatrix} 0.1 & 0.5 & 0.1 \\ 0.5 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.5 \end{pmatrix} \text{ and } T_2 = \begin{pmatrix} 0.3 \\ 0.3 \\ 0.3 \end{pmatrix}$$

Each of them meets the required properties to make the channel symmetric.

Example 2

Let the probability transition matrix be :

$$\mathbf{T} = \begin{pmatrix} \mathbf{0.1} & \mathbf{0.6} & \mathbf{0.3} \\ \mathbf{0.4} & \mathbf{0.1} & \mathbf{0.5} \\ \mathbf{0.5} & \mathbf{0.2} & \mathbf{0.3} \end{pmatrix}$$

As not one of the three columns has the same value on its three rows, there is no partition containing one input symbol for which the symmetry properties are met. Neither does the global probability transition matrix meet the properties. Consequently, the channel is not symmetric.

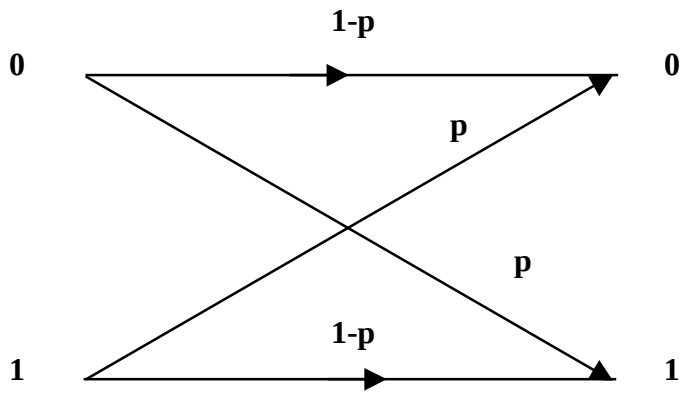
Calculating the capacity of a symmetric channel is easy by applying the following theorem :

Theorem

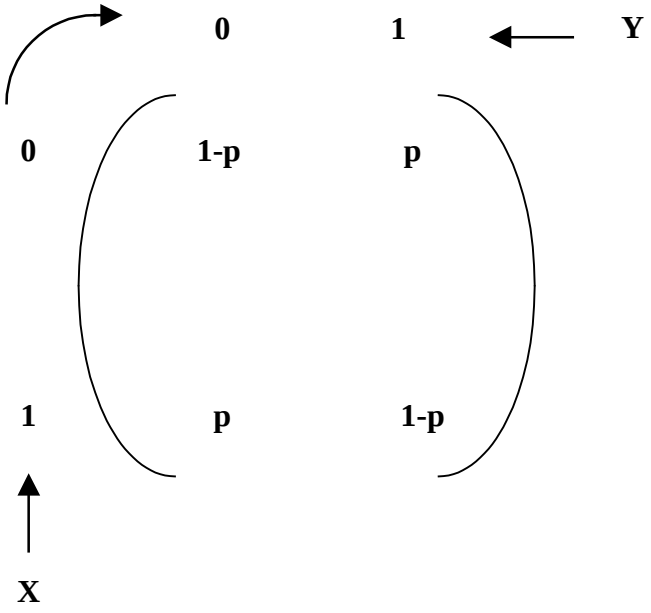
For a symmetric channel, the capacity is achieved for a uniform input probability distribution.

Example 1

Let us consider a Binary Symmetric Channel :



The probability transition matrix is :



This matrix meets the requirements to make the channel symmetric. Thus, the capacity is achieved for $P\{X=0\}=P\{X=1\}=\frac{1}{2}$

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

$$P\{Y=0\} = P\{Y=0 \cap X=0\} + P\{Y=0 \cap X=1\}$$

$$P\{Y=0\} = P\{X=0\} \times P\{Y=0/X=0\} + P\{X=1\} \times P\{Y=0/X=1\}$$

$$P\{Y=0\} = \frac{1}{2} \times (1-p) + \frac{1}{2} \times p = \frac{1}{2}$$

Thus, $P\{Y=1\} = \frac{1}{2}$ and $H(Y) = 1$ bit

Interpreting the rows of the probability transition matrix, we have :

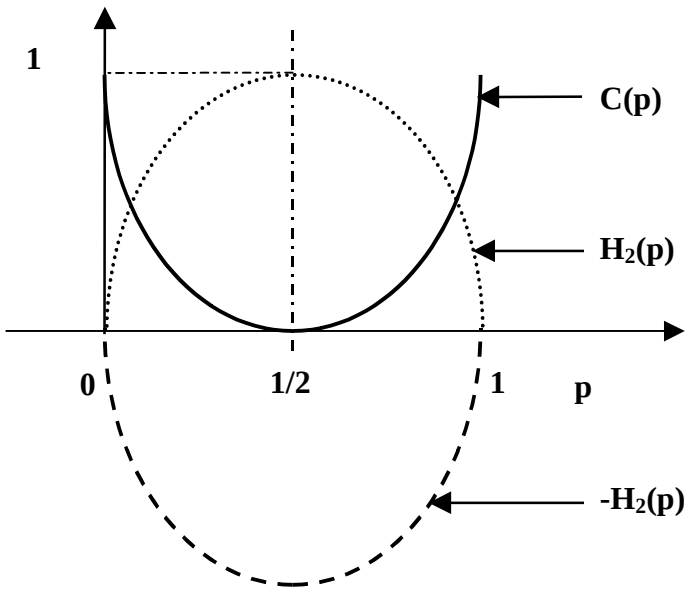
$$H(Y/X=0) = -p \times \log p - (1-p) \times \log(1-p) = H_2(p)$$

$$H(Y/X=1) = -p \times \log p - (1-p) \times \log(1-p) = H_2(p)$$

$$H(Y/X) = \frac{1}{2} \times H_2(p) + \frac{1}{2} \times H_2(p) = H_2(p)$$

Finally, we obtain :

$\boxed{C = 1 - H_2(p)}$. C can be sketched as a function of p :



Comments

- $p = \frac{1}{2}$ is an axis of symmetry for $C(p)$. Accordingly, we have $C(p) = C(1-p)$: changing p into $(1-p)$ amounts to permuting the output symbols.

- For $p = \frac{1}{2}$, $C\left(\frac{1}{2}\right) = 0$: knowing the output symbol does not provide any information about the input symbol, as the input and output random variables become independent.

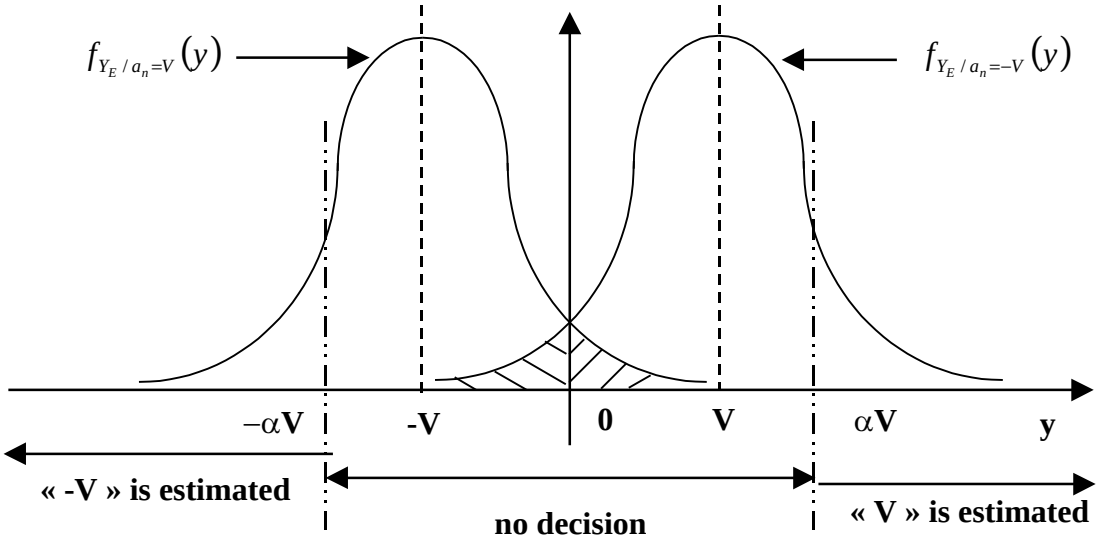
- The cases $p = 1$ or $p = 0$ can be easily interpreted : knowing the output symbol implies knowing the input symbol, which may be represented by one bit.

Example 2

The Binary Erasure Channel.

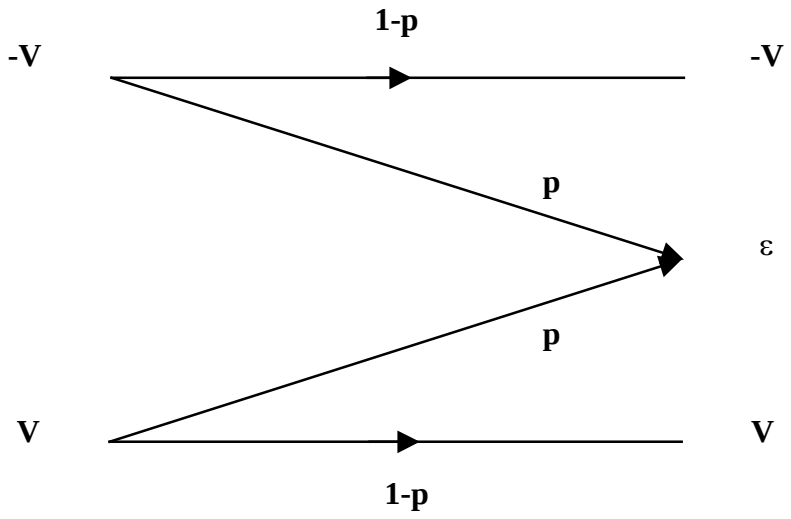
Let us resume the example of page 57.

We can think of a decision device such that the estimated symbol is V (respectively –V) if the input sample amplitude is greater (respectively smaller) than αV (respectively $-\alpha V$);, otherwise the estimated symbol is ε (an erasure symbol, i.e. neither –V nor V).



If αV is chosen such as :

$P\{Y_E < -\alpha V / a_k = V\}$ and $P\{Y_E > \alpha V / a_k = -V\}$ are negligible, then the channel can be sketched as follows :



with $p = P\left\{N\left(-V;\frac{N_0}{2}\right) > -\alpha V\right\} = P\left\{N(0;1) > \frac{(1-\alpha)V\sqrt{2}}{\sqrt{N_0}}\right\}$

After receiving ε, we may consider that the transmitted symbol is lost or ask the transmitter to re-send the symbol until the decision device delivers “-V” or “V”.

Let us write the probability transition matrix :

	-V	ε	V
-V	1-p	p	0
V	0	p	1-p

The set of output symbols can be partitioned into $\{-V,V\}$ and $\{\varepsilon\}$. The two probability transition matrices meet the requirements to make the channel symmetric. The capacity is $I(X;Y)$ with a uniform input probability distribution.

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

$$P\{Y=-V\}=P\{X=-V\cap Y=-V\}=P\{X=-V\}\times P\{Y=-V/X=-V\}=\frac{1}{2}\times(1-p)$$

By symmetry, we have :

$$P\{Y=V\}=\frac{1}{2}\times(1-p)$$

Then, we deduce :

$$P\{Y=\varepsilon\}=1-2\times\frac{1}{2}\times(1-p)=p$$

$$H(Y)=-2\times\frac{1}{2}\times(1-p)\log\left(\frac{1}{2}\times(1-p)\right)-p\log p$$

$$H(Y)=-(1-p)(-1+\log(1-p))-p\log p$$

$$H(Y)=(1-p)+H_2(p)$$

Interpreting the terms of the probability transition matrix, we obtain :

$$H(Y/X=-V)=H(Y/X=V)=H_2(p)$$

Eventually, we get :

$$\boxed{C=(1-p)\,\text{bit}}$$

For $p=0$, the channel is the noiseless channel whose capacity is one bit : when –V (respectively V) is transmitted, -V (respectively V) is estimated.

THE NOISY CHANNEL THEOREM

Let S be an information source whose the entropy per symbol is $H_\infty(S)$ and the symbol rate D_S. We want to transmit reliably the outcomes of S over a channel of capacity per use C at a symbol rate D_C. Is it possible?

The answer is given by **the noisy channel theorem** :

If the entropy rate is smaller than the capacity per time unit, i.e.:

$H_\infty'(S) = H_\infty(S) \times D_S < C \times D_C = C'$

(entropy and capacity must be expressed in the same unit)

then, $\forall\, \varepsilon > 0$, there exists a code to transmit the outcomes of S over the channel in such a way that, after decoding, we have :

$P\{\text{error}\} < \varepsilon$.

In other words, if $H_\infty'(S) < C'$, it is possible to transmit the outcomes of S over the channel with an arbitrarily low probability of error, provided appropriate means are used.

This theorem is the most important result in information theory.

Comments

- unlike the source coding theorem, the noisy channel theorem does not state how to construct the code, we only know that such a code exists,
- what is surprising is that we can transmit as reliably as desired with a noisy channel,
- a posteriori, the noisy channel theorem justifies the definition of capacity as the ability of the channel to transmit information reliably.

Example

Let S be a memoryless binary source such that

$P\{S = 0\} = 0.98$ and $P\{S = 1\} = 0.02$

The symbol rate is

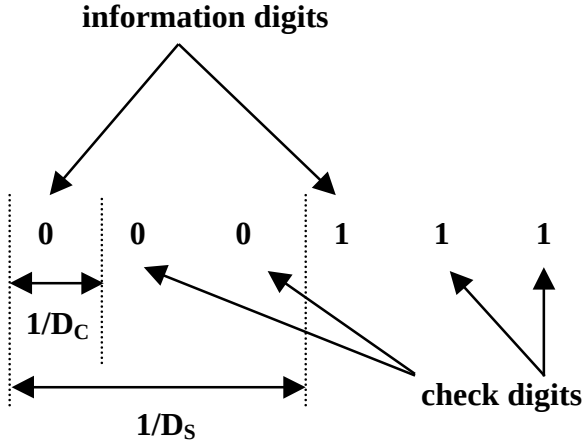
$D_S = 600\ Kbits/\sec$

To link the emitter to the receiver, we have a binary symmetric channel with crossover probability $p = 10^{-3}$. Its maximum symbol rate is $D_C = 450\ Kbits/\sec$.

We will try to answer the questions :

- Is it possible to transmit the outcomes of the source over the channel with an arbitrarily low probability of error?

- To reduce the probability of error due to the noisy channel, we can think of using the repetition code of length 3 consisting of repeating the information digit twice. In other words, each information digit is encoded into a codeword made of three digits : the information digit plus two check digits identical to the information digit. As a decision rule, we can decide “0” has been emitted if the received codeword contains at least two “0”s and “1” otherwise.



Which source coding algorithm must we use to be able to implement the repetition code without loss of information?

To answer the first question, we have to know whether the condition of the noisy channel theorem is satisfied or not.

S being memoryless, we have :

$H_\infty(S) = H_2(0.02) = 0.1414$ bit

Thus, the entropy rate is :

$$H_\infty'(S) = H_\infty(S) \times D_S = 0.1414 \times 600 \times 10^3 = 84864 \text{ bits/sec}$$

The capacity (per use) of the channel is :

$$C = 1 - H_2(10^{-3}) = 0.9886 \text{ bit}$$

And the capacity per time unit :

$$C' = C \times D_C = 0.9886 \times 450 \times 10^3 = 444870 \text{ bits/sec}$$

As $H_\infty'(S) < C'$, the answer to the first question is “yes”.

The initial symbol rate of S being greater than the maximum symbol rate of the channel, we cannot connect directly the output of S with the input of the channel. We have to reduce the symbol rate of S by applying a compression algorithm.

Taking into account the repetition code we want to use to transmit the outcomes of S over the channel, if D_S' denotes the symbol rate of the compressed source, we have to satisfy :

$$\frac{1}{D_S'} \geq 3 \times \frac{1}{D_C}$$

$$D_S' \leq \frac{D_C}{3} = \frac{450}{3} = 150 \text{ Kbits/sec} = \frac{600}{4} = \frac{D_S}{4} = 0.25 \times D_S$$

Hence, the source coding should result in an average number of code bits by source bit smaller than 0.25.

After the source coding theorem, we know that there exists a prefix code applied to the L^{th} extension satisfying :

$$H_L(S) \leq \bar{n} < H_L(S) + \frac{1}{L}$$

As S is a memoryless source, we have :

$$H_L(S) = H_\infty(S) = 0.1414 \text{ bit}$$

Thus,

$$0.1414 \le \bar{n} < 0.1414 + \frac{1}{L}$$

To make sure that $\bar{n} < 0.25$, we have to chose L so that $0.1414 + \frac{1}{L} < 0.25$ i.e.

$$L > \frac{1}{(0.25 - 0.1414)} \approx 9.2$$

Conclusion

Encoding the 10^{th} extension of S by a Huffman code will result in a reduction in the symbol rate of at least 75%. Then, the repetition code of length 3 can be implemented to transmit the outcomes of S' (the compressed source) over the channel.

ERROR CORRECTING CODES

Example

The outcomes $\{0,1\}$ of a binary memoryless source with $P\{"1"\}=q$ and $P\{"0"\}=1-q$ have to be transmitted over a binary symmetric channel whose the probability of error is p. We think of two ways to transmit the digits : directly (without coding) and using a repetition code consisting of sending three times the same information bit. We will calculate the average probability of error in both cases.

- without coding

Let ε be the event that an error occurs. Letting X (respectively, Y) denote the input (respectively, the output), we have

$$P\{\varepsilon\}=P\{(X=1\cap Y=0)\cup(X=0\cap Y=1)\}$$

$$P\{\varepsilon\}=P\{X=1\cap Y=0\}+P\{X=0\cap Y=1\}$$

and

$$P\{X=1\cap Y=0\}=P\{X=1\}P\{Y=0/X=1\}=qp$$

$$P\{X=0\cap Y=1\}=P\{X=0\}P\{Y=1/X=0\}=(1-q)p$$

$$P\{\varepsilon\}=qp+(1-q)p=p$$

- with the repetition code

There are two codewords :

$"0" \to "000"$
$"1" \to "111"$

As some errors may occur while transmitting, the possible received words are : 000, 001, 010, 011, 100, 101, 110, 111. The decoding rule (majority logic) consists of deciding “0” has been emitted if the received word contains at least two “0”, otherwise “1” is decided.

Setting $E = \{\text{at least two bits are corrupted}\}$, we have :

$$\varepsilon = \{("0" \text{ emitted}) \cap E\} \cup \{("1" \text{ emitted}) \cap E\}$$

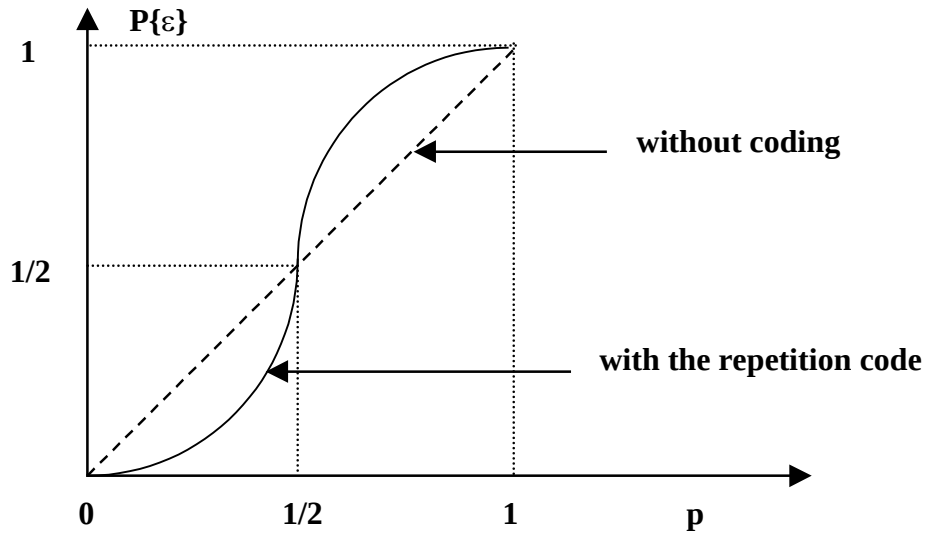
$$P\{\varepsilon\} = P\{("0" \text{ emitted}) \cap E\} + P\{("1" \text{ emitted}) \cap E\}$$

$$P\{\varepsilon\} = P\{"0" \text{ emitted}\} P\{E / "0" \text{ emitted}\} + P\{"1" \text{ emitted}\} P\{E / "1" \text{ emitted}\}$$

$$P\{\varepsilon\} = (1-q)\left(C_3^2 p^2 (1-p) + p^3\right) + q\left(C_3^2 p^2 (1-p) + p^3\right)$$

$$P\{\varepsilon\} = 3p^2(1-p) + p^3 = -2p^3 + 3p^2$$

To compare the performances, we may sketch $P\{\varepsilon\}$ versus p for both cases (without coding and using the repetition code) :



Comments

- As long as $p < \frac{1}{2}$, the probability of error resulting from using the repetition code is smaller while in the range $\frac{1}{2} < p < 1$ it is greater. To clear up this paradox, one only has to notice that for $p > \frac{1}{2}$, the outputs "0" and "1" have to be exchanged (otherwise corrupted bits are more frequent than correct bits). Then, the probability of error becomes $p' = 1 - p$ with $p' < \frac{1}{2}$.

- Error detection

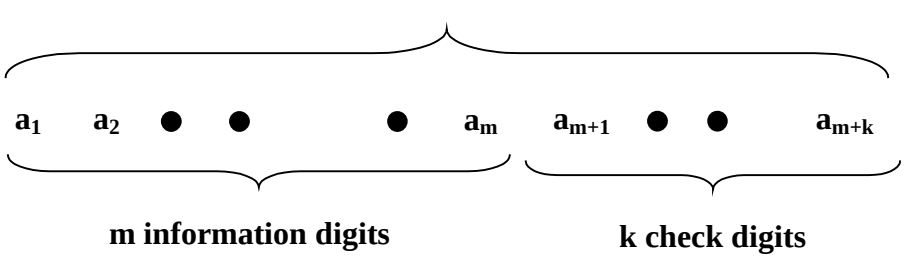
This repetition code can detect one or two errors. If one or two errors occur, the received word contains one "1" and two "0" or one "0" and two "1". However, it is not possible to know the exact number of errors (1 or 2). When three errors occur, the received word is a codeword and it is similar to the case where there is no error. The code is said to be two-error-detecting.

- Error correction

If the received word contains one error, this error is corrected by applying the decision rule (majority logic). This repetition code is said to be one-error-correcting.

CONSTRUCTION

Without loss of generality, we will limit ourselves to systematic binary codes. By "systematic binary codes", we mean codes with a binary alphabet and whose codewords consist of information bits and check bits in such a way that check bits are linear combinations of information bits, which information bits appear directly in the codeword.



Example

The repetition code is a systematic code with $m = 1$ and $k = 2$.

a_1 is the information digit

The two check digits a_2 and a_3 satisfy :

$a_2 = a_1$
$a_3 = a_1$

NEAREST NEIGHBOUR DECODING

We will assume the codewords are transmitted over a binary symmetric channel of probability of error p $\left(p < \frac{1}{2} \right)$.

The received word is denoted y and we have to decide which codeword (c) has been sent. To work out a solution to this problem, the most natural method is maximum a posteriori decoding. However, implementing this algorithm requires the knowledge of the a priori probability distribution. Consequently, we will apply maximum likelihood decoding which consists of finding the codeword c such as $P\{y \text{ received} / \text{c emitted}\}$ is as great as possible, for the received y.

Given y and c, let us suppose they differ in *l* positions. Then, we have :

$$P\{\text{y received} / \text{c emitted}\} = p^l (1-p)^{n-l} = g(l)$$

To make the calculations easier, we may consider :

$$\log(g(l)) = l \log p + (n-l) \log(1-p)$$

Differentiating $g(l)$ with respect to l gives :

$$\frac{dg(l)}{dl} = \log p - \log(1-p) = \log \frac{p}{1-p}$$

$p < \frac{1}{2}$ implies $\frac{p}{1-p} < 1$ and $\frac{d \log g(l)}{dl} < 0$

Consequently $\log(g(l))$ is a decreasing function of *l* and as $\log(g(l))$ is an increasing function of $g(l)$, $g(l)$ is a decreasing function of *l*. Also, the maximum of $g(l)$ is achieved for *l* minimum.

To summarise, we have to choose the codeword c closest to the received word y. Accordingly, maximum likelihood decoding is equivalent to minimum distance decoding.

LINEAR CODES

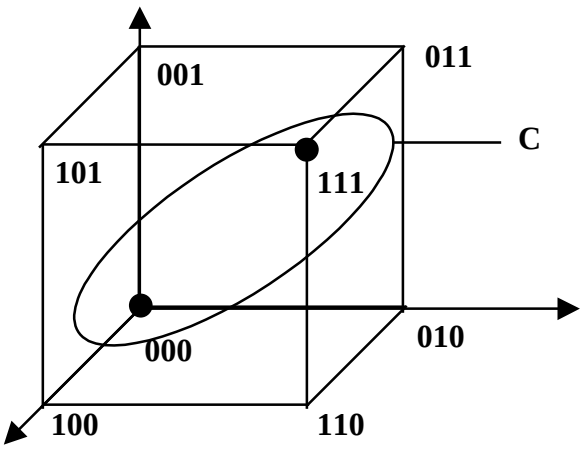
Linear codes have advantages over non linear codes: coding and decoding are easier to implement.

Let us consider $V = \{0,1\}^n$ consisting of n-tuples of binary elements. V is a vector space of dimension n over $K = \{0,1\}$. The sum of two vectors is obtained by adding (binary addition) the components in the same position: the addition table for each element follows the "modulo-2" rule: $0+0=0, \quad 1+0=0+1=1, \quad 1+1=0$.

The 2^n elements of V are the possible received words, as a word can be associated with a vector.

C is said to be a linear code if C is a subspace of V.

For instance, the repetition code is a linear code since $C = \{"000","111"\}$ is a subspace of dimension 2 of $V = \{0,1\}^3$. Each codeword is its own opposite as $0+0=1+1=0$.



V corresponds to the 8 vertices of the cube whereas C is represented by two of these vertices.

Some definitions :

The vectors u and v are possible received words which are elements of V.

Weight

The weight of u, $w(u)$, is the number of "1" in u.

$w(\text{"101"}) = 2 \quad w(\text{"000"}) = 0$

Hamming distance

The Hamming distance from u to v, $d(u,v)$, is the number of positions in which u and v differ. Consequently, $d(u,v)$ is the weight of $u+v$.

Let u and v be respectively "01011" and "00110". $d(u,v) = 3$ since the two vectors differ in three positions. Moreover, $w(u+v) = w(\text{"01101"}) = 3 = d(u,v)$.

Minimum distance

The minimum distance d_m of a code C is the minimum distance between distinct codewords of C.

And, since V is a vector space, it is a group and we have :

$$d_m = \inf_{u \neq v} d(u,v) = \inf_{u \neq v} w(u+v) = \inf_{x \in V^*} w(x)$$

Minimum distance = minimum weight, once the all-zero codeword is removed

The minimum distance is a fundamental parameter and, as we shall see, the greater the minimum distance, the more powerful the code in terms of error detecting and error correcting.

Indeed, with simple geometric considerations, the two following properties can easily be established:

Error detecting ability

A linear code C of minimum distance d_m is able to detect a maximum of $d_m - 1$ errors.

Error correcting ability

A linear code C of minimum distance d_m is able to correct a maximum of $\text{int}\left(\frac{d_m - 1}{2}\right)$ errors.

We should keep in mind that, whatever the number of supposed errors corrected, we never know the number of errors which actually occurred and the decided codeword may not be the right one. To make this clear, we will examine the different possible situations.

Let us suppose the codeword C_0 has been sent. There are several possibilities

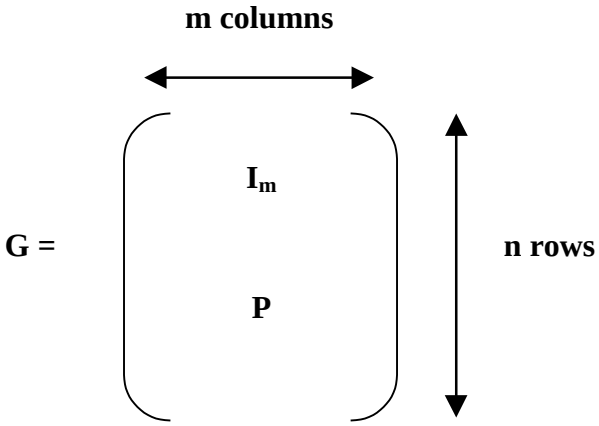
- No error occurs. C_0 is received and decided.
- Some errors occur which result in a codeword C_1 received. Then no error is corrected since C_1 is decided.
- The number of errors is smaller than or equal to $\text{int}\left(\frac{d_m - 1}{2}\right)$. Then, the received word is not a codeword but the errors are corrected and C_0 is decided.
- The number of errors is greater than $\text{int}\left(\frac{d_m - 1}{2}\right)$. If the codeword the closest to the received word is C_0 ,all the errors are corrected although their number is greater than $\text{int}\left(\frac{d_m - 1}{2}\right)$. Otherwise a codeword distinct from C_0 is decided : some errors can be corrected but others may be added too.

GENERATOR MATRIX

Let C be a linear code whose codewords consist of m information digits followed by k check digits $(n = m + k)$. As C is a subspace of $V = \{0,1\}^m$, there exists a matrix G such :

$C = \{u \in V \ / \ u = Gv \ \forall \ v \in V\}$

As the first m digits of u are identical to the m digits of v, G has the following structure :



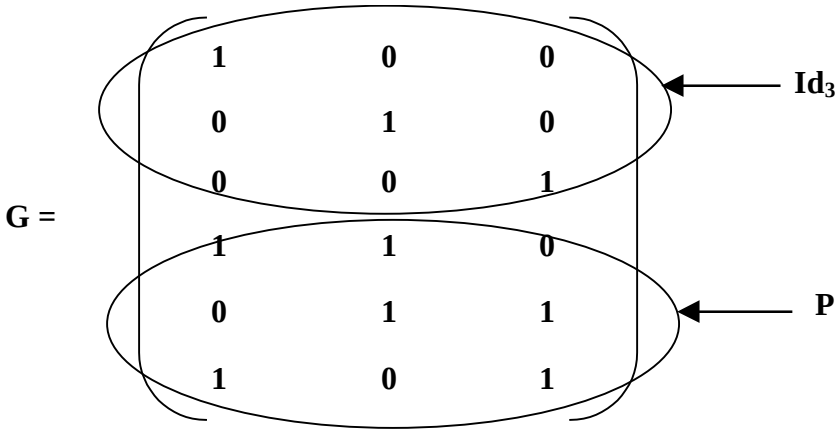
The $(n-m)$ rows of the submatrix P express the linear combinations corresponding to the check digits.

Example

$m = k = 3$. a_1, a_2, a_3 are the three information bits. The check bits a_4, a_5, a_6 satisfy :

$$\begin{matrix} a_4 = a_1 + a_2 \\ a_5 = a_2 + a_3 \\ a_6 = a_1 + a_3 \end{matrix} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

The generator matrix is :



After multiplying the generator matrix by the $2^3 = 8$ vectors consisting of the three information digits, we obtain the codewords :

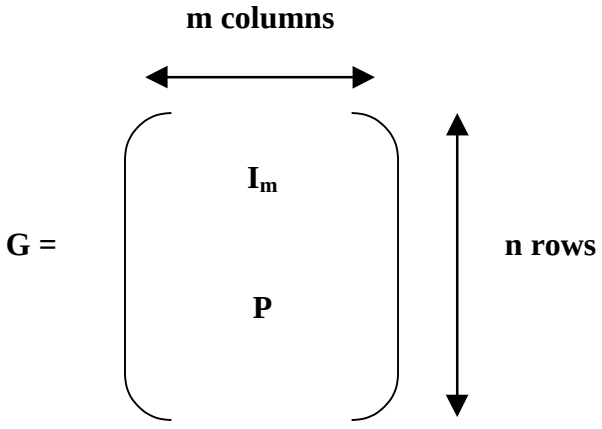
source words	codewords
000	**000000**
001	**001011**
010	**010110**
011	**011101**
100	**100101**
101	**101110**
110	**110011**
111	**111000**

The minimum weight of the codewords is 3. Therefore, the minimum distance is 3 and this code is 2-error-detecting and one-error-correcting.

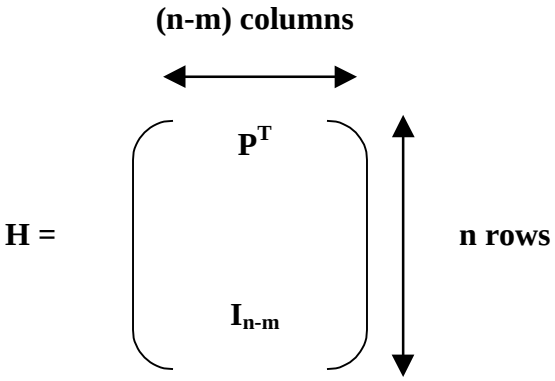
PARITY-CHECK MATRIX

Implementing maximum likelihood decoding involves finding the codeword closest to the received word.

Let us consider a systematic linear code C and its generator matrix G



Then, the orthogonal complement of C, $C^{\perp} = \{v \in V \,/\, v^T u = 0 \quad \forall\, u \in C\}$, is a linear code and its generator matrix is :



In addition, here is an important property useful for decoding :

A necessary and sufficient condition for a received word to be a codeword, is to verify :

$\boxed{H^T y = 0}$

Syndrome

The syndrome $S(y)$ associated with the received word y is defined as $\boxed{S(y) = H^T y}$.

Let us suppose the sent codeword is c and the corresponding received word y. We have $y = c + e$ where e is the error vector. The syndrome then takes the form

$S(y) = S(c+e) = S(c) + S(e) = S(e)$ since c is a codeword.

This equality can be interpreted as follows :

The syndrome of a received word depends only on the actual error. This property will help us when decoding.

Minimum distance decoding process
y is the received word.

- If $S(y) = 0$, y is a codeword and y is decided.

- If $S(y) \neq 0$, we have to find a codeword c such as $d(y,c)$ is minimum. As y is not a codeword, we have $y = c + z$ and $z = y + c$ (each vector is its own opposite).

$S(z) = S(y+c) = S(y)$.Now from $z = y + c$, we deduce $w(z) = w(y+c) = d(y,c)$. As such, finding the codeword c closest to y is the same as finding a vector z of minimum weight satisfying $S(z) = S(y)$. Then the codeword c is given by $c = z + y$.

In practice, a decoding table is constructed. It contains all the syndrome values associated with the minimum weight sequences.

Example

Let us resume the preceding code C with generator matrix G :

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

The generator matrix of the orthogonal complement of C is :

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The parity-check matrix is :

$$\mathbf{H}^T = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

The dimension of H^T is 3×6 and the sequences z have 6 components. Consequently, the syndromes are vectors with 3 components. There are $2^3 = 8$ different possible values for the syndrome.

The syndrome 000 is associated with the sequence 000000. By multiplying H^T by the sequences z, we obtain :

$S(000001) = 001$
$S(000010) = 010$
$S(000100) = 100$
$S(001000) = 011$
$S(010000) = 110$
$S(100000) = 101$

There is $8 - 6 - 1 = 1$ remaining value (111). We may associate this value with a sequence of weight equal to 2. For instance :

$S(100010) = 111$

Decoding table

syndrome values	**Sequences z (minimum weight)**
000	**000000**
001	**000001**
010	**000010**
011	**001000**
100	**000100**
101	**100000**
110	**010000**
111	**100010 (for instance)**

Using this decoding table allows us to correct one error (wherever it is) and two errors if located in the second and fifth positions.

EXERCISES

INFORMATION MEASURE EXERCISES

EXERCISE 1

Two cards are simultaneously drawn at random from a pack of 52 playing cards.

1. Calculate the uncertainty of the events :
- {the king of hearts is one of the two drawn cards}
- {at least one of the two drawn cards is a heart}

2. What is the amount of information provided by E = {at least one of the two drawn cards is a diamond} about F = {there is exactly one heart among the two drawn cards}?

3. Are the events E and F independent?

EXERCISE 2

Evaluate, in two different ways, the exact number of bits required to describe four cards simultaneously drawn from a pack of 32 playing cards.

EXERCISE 3 (after Robert Gallager)

Let $\{X = a_1\}$ denote the event that the ball in a roulette game comes to rest in a red compartment, and $\{X = a_2\}$ similarly denote a ball coming to rest in black.

We suppose that $P\{X = a_1\} = P\{X = a_2\} = \frac{1}{2}$.

The croupier of the roulette table has developed a scheme to defraud the house. After years of patient study, he has learned to partially predict the coulour that will turn up by observing the path of the ball up to the last instant that bets may be placed. By communicating this knowledge to an accomplice, the croupier expects to use his inside knowledge to gain a tidy sum for his retirement.

Let Y denote the croupier's signal : a cough, $Y = b_1$, indicates a red prediction and a blink, $Y = b_2$, indicates a black prediction. Assuming $P\{X = a_1 / Y = b_1\} = P\{X = a_2 / Y = b_2\} = \frac{3}{4}$, calculate the average information provided by Y about X.

EXERCISE 4

Suppose we have n coins, with one of them counterfeit: it is lighter than the others.

1. What is the average uncertainty associated with finding the counterfeit coin?

A Roman balance is available on which two groups of coins A and B may be compared. Each weighing results in three possibilities :

- A is heavier than B,
- A and B have the same weight,
- A is lighter than B.

2. Express the average information provided by a weighing towards finding the counterfeit coin in terms of the average uncertainty associated with the weighing. Extend this result to the case of m weighings.

3. What is the maximum average information provided by a weighing towards finding the counterfeit coin? When do we come across such a situation?

4. Let us suppose a procedure has been worked out to find the counterfeit coin with at most m weighings.

- What is the smallest value of m as function of n?

- When n is a power of 3, describe the procedure for which the average information provided by each weighing is maximum.

5. We now consider that the number of counterfeit coins is unknown : 0, 1, … or n. The balance is a weighing machine. The weight f of a counterfeit coin is smaller than the weight v of the other coins.

- Demonstrate that a weighing of coins allows us to know the number of counterfeit coins among the weighed coins.
- Let us suppose a procedure has been worked out to find the counterfeit coin (s) in at most m weighings. What is the minimum value of m as function of n?

SOURCE CODING EXERCISES

EXERCISE 1 (after Robert Gallager)

The weatherman's record in a given city is given in the table below, the numbers indicating the relative frequency of the indicated event.

	actual	
prediction	**no rain**	**rain**
no rain	**5/8**	**1/16**
rain	**3/16**	**1/8**

1. A clever student notices that he could be right more frequently than the weatherman by always predicting no rain. The student explains the situation and applies for the weatherman's job, but the weatherman's boss, who is an information theorist, turns him down. Why?

2. The weatherman's boss wants to store the predictions, M, and the actual weather, T, of 1,000 days in a computer file. How many bits are required?

3. Using a Huffman code for the value pairs (M,T), what is, approximately, the size of the file?

EXERCISE 2

Let X be a random variable taking values in $\{x_1, x_2, ..., x_n\}$ with probabilities $\left\{\frac{1}{2}, \frac{1}{2^2}, ..., \frac{1}{2^n}\right\}$.

1. Construct a Huffman code for the values of X.

2. Compare the average length of the codewords to the entropy of X. How can we explain such a result?

EXERCISE 3 (after Robert Gallager)

Let U be a discrete memoryless source taking on values in $\{a_1, a_2, ..., a_i, ...\}$ with probabilities $P\{a_1\}, P\{a_2\}, ..., P\{a_i\}, ...$. We suppose:

$\forall\, k > j \geq 1 \quad P\{a_k\} \leq P\{a_j\}$

Let us define $Q_i = \sum_{k=1}^{i-1} P\{a_k\} \quad \forall\, k > 1$ and $Q_1 = 0$ associated with the message a_i.

The codeword assigned to message a_i is formed by finding the "decimal" expansion of $Q_i < 1$ in the binary system $\left(\text{i.e. } \frac{1}{2} \to 100, \quad \frac{1}{4} \to 0100, \quad \frac{5}{8} \to 10100, ...\right)$ and then truncating this expansion to the first n_i digits, where n_i is the integer equal to or just larger than the self-information of the event $\{U = a_i\}$ expressed in bits.

1. Does this code satisfy the Kraft inequality?

2. Let $\bar{n}$ denote the average length of the codewords. Show that $\bar{n}$ satisfies the double inequality : $H(U) \leq \bar{n} < H(U) + 1$ with $H(U)$ the entropy of U

Application

3. Construct the code for a source U taking 8 values with probabilities $\frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}$.

4. Compare the average length of the codewords to the entropy of U. How can we explain this result?

EXERCISE 4

Player A rolls a pair of fair dice. After the throw, the sum of the two faces is denoted S.

1. Construct a Huffman code to encode the possible values of S.

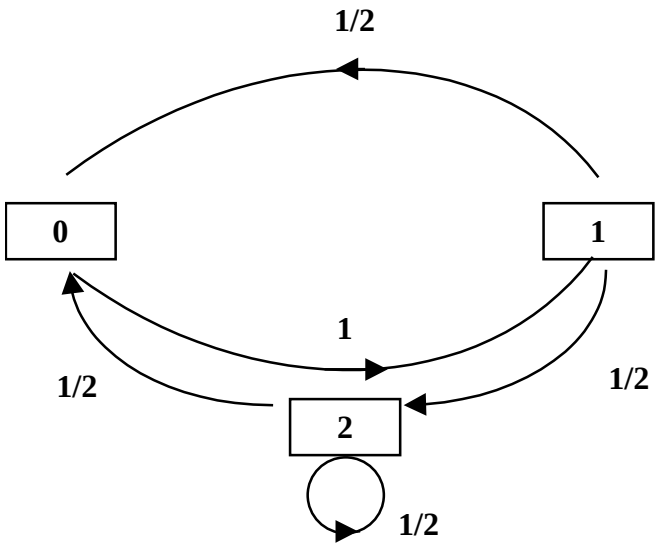
2. Player B has to guess the number S by asking questions whose answers must be "yes" or "no". We call an optimum procedure any set of successive questions which allows player B to determine S in a minimum average number of questions.

- What is the average number of questions of an optimum procedure?
- What is the first question of the optimum procedure?

- Calculate the average information provided by player A about the number S when answering the first question of an optimum procedure.

EXERCISE 5

A ternary information source U is represented by a Markov chain whose state transition graph is sketched below :



1. If U delivers 3,000 ternary symbols per second, calculate, in bits per second, the entropy rate of U.

2. Construct a Huffman code for the second extension of U. Calculate the resulting average number of bits used to represent one ternary symbol of U.

EXERCISE 6

A memoryless source S delivers symbols A, B, C, D, E, F and G with probabilities 1/16, 1/16, 1/16, 1/16,1/4,1/4 and 1/4 .

1. Construct a Huffman code for the 7 values of S. Compare the average number of bits used to represent one value of S to the entropy of S.

Let U denote the binary source obtained by the preceding coding of S.

2. Calculate the entropy of U.

3. By applying the law of large numbers, calculate the probability distribution of U.

4. What can be said about the memory of U?

5. By generalising the preceding results, give a signification to an optimum source coding.

EXERCISE 7 (after David MacKay)

A poverty-stricken student communicates for free with a friend using a telephone by selecting a positive integer n and making the friend's phone ring n times then hanging up in the middle of the nth ring. This process is repeated so that a string of symbols $n_1, n_2, ...$ is received. Let $l(n)$ denote the lapse of time necessary to transmit the integer n.

Setting $p_n = P\{\text{the integer n is emitted}\} \quad \forall n \in \text{IN}^*$ and $p = (p_1, p_2, ...)$, it can be shown that the information rate $\beta(p)$ transmitted over the telephone line is maximum when we have:
$p_n = 2^{-\beta_M l(n)} \quad \forall n \in \text{IN}^*$ with $\beta_M = \underset{p}{Max}\, \beta(p)$

Now we will suppose that $\forall n \in IN^* \quad l(n) = n$ seconds.

1. Calculate β_M and the corresponding optimum probability distribution (p).

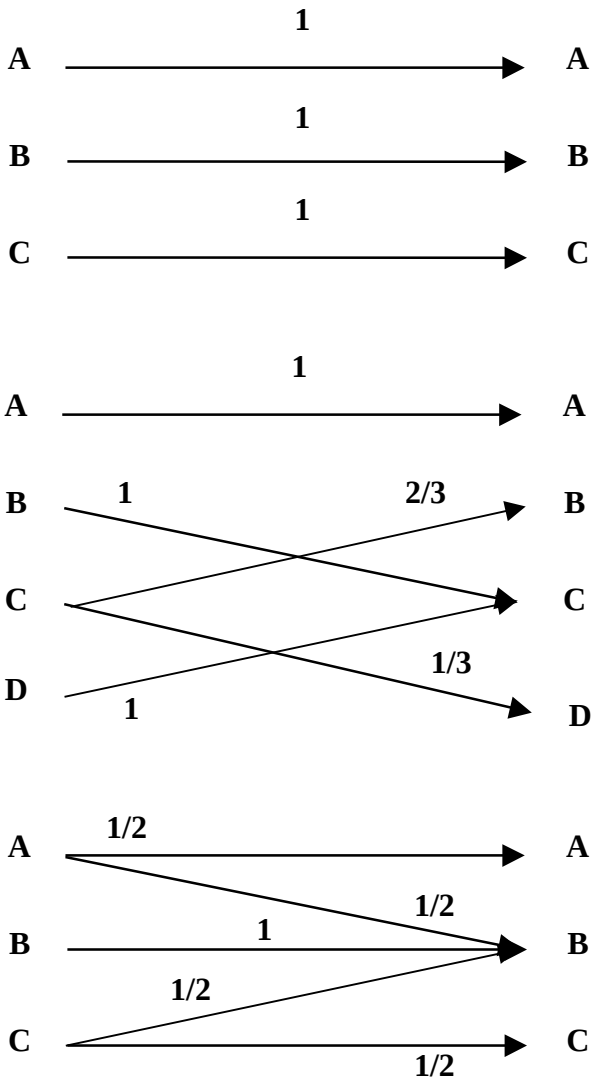
2. If p were the uniform distribution over $\{1,2\}$, what would be the value of the information rate? Compare with the value given by the preceding question.

3. Let S be a binary memoryless source whose entropy is maximum. We have to transmit the outcomes of S over the telephone line. Construct a prefix code so that the preceding procedure will achieve a maximum information rate transmitted. What is, in seconds, the average duration in transmitting a codeword? What is, in seconds, the average duration in transmitting one bit of S?

COMMUNICATION CHANNEL EXERCISES

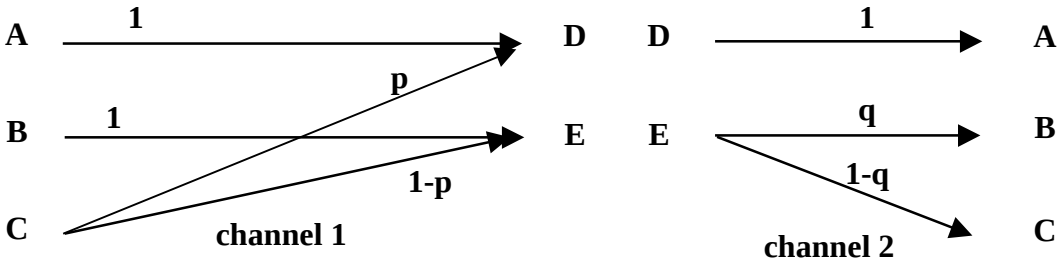
EXERCISE 1

Calculate the capacity of the channels :



EXERCISE 2

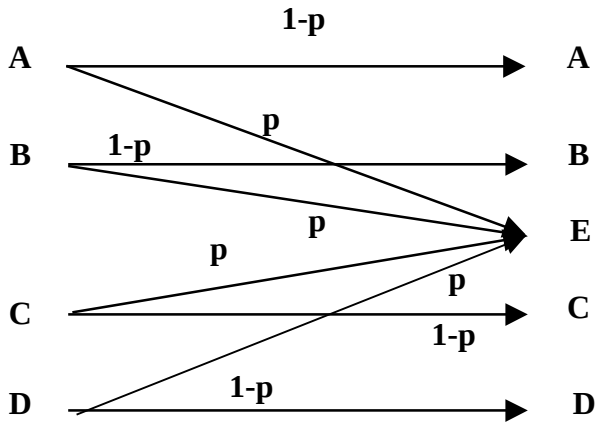
1. Calculate the capacity of the two channels :



2. Calculate the capacity of the channel obtained when the outputs D and E of channel 1 are connected with the inputs D and E of channel 2.

EXERCISE 3

We consider the channel below :



1. Calculate its capacity.

Consider transmitting the outcomes of a binary source S over this channel.

2. If we want to transmit directly (without coding) the outcomes of S, how many source symbols must be taken at a time?

3. Let us suppose S is memoryless. Calculate the probability for a source word to be received without error.

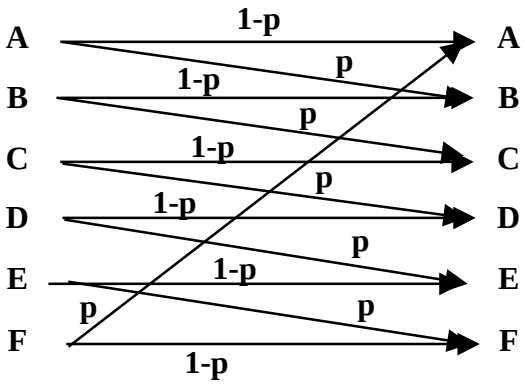
4. Suppose the source symbols equally likely. Depending on D_S, the source symbol rate, what is the maximum channel symbol rate D_U so that the probability of error is arbitrarily low, provided appropriate means are used?

EXERCISE 4

Let S be a memoryless source taking on 8 equally likely values. Its symbol rate is 1,000 symbols per second. The outcomes of S are to be transmitted over a binary symmetric channel of crossover probability equal to 0.001. The maximum channel symbol rate is 3,000 bits per second. Is it possible to transmit the outcomes of S with an arbitrarily low probability of error?

EXERCISE 5

1. Calculate the capacity of the channel :



Consider transmitting the outcomes of a source S over the channel. The possible values of S are $\{a,b,c,d,e\}$ with probabilities $\left\{\frac{1}{3},\frac{1}{3},\frac{1}{9},\frac{1}{9},\frac{1}{9}\right\}$. We suppose $p=0.01$ and the channel symbol rate equal to 5,000 (6-ary) symbols per second.

2. What is the maximum source symbol rate if we want to transmit S over the channel with an arbitrarily low probability of error?

We encode the possible values of S by a ternary code such as :

$a \rightarrow 0$
$b \rightarrow 1$
$c \rightarrow 20$
$d \rightarrow 21$
$e \rightarrow 22$

3. Is this code uniquely decipherable?

4. Calculate the average length of the codewords. Is this code optimum? Why or why not?

5. Deduce from the preceding questions a procedure to connect the source to the channel so that the probability of error is zero.

ERROR CORRECTING CODE EXERCISES

EXERCISE 1

Let us consider the following code :

source words	codewords
00	**00000**
01	**01101**
10	**10111**
11	**11010**

1. Give the generator matrix and the parity-check matrix. Construct a decoding table associated with the code.

2. We suppose the codewords are transmitted over a binary symmetric channel of crossover probability p. Calculate the probability of error by codeword when using the decoding table.

EXERCISE 2

We consider the systematic linear code :

source words	codewords
000	**?00??**
001	**00101**
0??	**010??**
011	**??1?1**
100	**1001?**
101	**101?1**
110	**1100?**
111	**111??**

1. Complete the table by replacing the question marks with the correct bits.

2. Calculate the generator matrix and the parity check matrix.

3. Is it possible to construct a decoding table able to correct one error on any of the three information bits?

4. Is it possible to construct a decoding table to correct one error on any of the check bits?

EXERCISE 3

A memoryless binary source S delivers “0” and “1” with probabilities 0.98 and 0.02 at a symbol rate of 300 Kbits/sec. A binary symmetric channel of crossover probability equal to 0.05 and whose the maximum symbol rate is 280 Kbits/sec is available.

1. Is it possible to transmit the outcomes of S over the channel with an arbitrarily low probability of error?

SOURCE CODING

2. Consider reducing the symbol rate of S by at least 50% by encoding the outputs of S with a Huffman code. What is the minimum extension of S to be encoded to meet such conditions?

3. Construct a Huffman code for the third extension of S. What is the average symbol rate of the binary source which results from this code? How many check bits have to be juxtaposed to one information digit so that the symbol rate over the channel is equal to 280 Kbits/sec?

CHANNEL CODING

4. The second extension of the binary source obtained after coding S is to be encoded by a systematic linear code whose codewords consist of two information bits and three check digits. If we want the code to correct one error by codeword, what is the smallest acceptable value of the minimum distance?

5. We consider a code that meets the preceding conditions. Construct the corresponding generator matrix.

6. List the codewords by assigning to each codeword its weight.

7. Construct a decoding table. How many patterns of two errors can be corrected?

8. Compare the probability of error resulting from using the decoding table to the probability of error when transmitting directly (without coding) the source words.

SOLUTIONS

INFORMATION MEASURE SOLUTIONS

EXERCISE 1

1. 4.70 bits and 1.18 bit.
2. –0.404 bit
3. No

EXERCISE 2

15.134 bits

EXERCISE 3

0.188 bit

EXERCISE 4

1. $\log_2 n$
2. X_i: result of the ith weighing and X random variable taking on n values with a uniform probability distribution. $I(X_i; X) = H(X_i)$
3. $\log_2 3$ bits. At the first weighing when n is a multiple of 3.
4. $\log_3 n$
5. $\dfrac{n}{\log_2(n+1)}$

SOURCE CODING SOLUTIONS

EXERCISE 1

1. The student does not provide any information about the actual weather.
2. At least $696 + 896 = 1{,}592$ bits.
3. Approximately 1,562 bits

EXERCISE 2

2. $\bar{n} = H(X)$

EXERCISE 3

1. Yes.
4. $\bar{n} = H(U)$

EXERCISE 4

2. 3.3 questions. 0.98 bit.

EXERCISE 5

1. 2 Kbits/sec
2. 1.166

EXERCISE 6

1. $\bar{n} = H_{\infty}(S)$
2. $H_{\infty}(U) = 1\,\text{bit}$
3. $P\{U = 0\} = P\{U = 1\} = \frac{1}{2}$
4. U is memoryless.

EXERCISE 7

1. $\beta_M = 1\,\text{bit/sec}$ and $p_n = 2^{-n} \quad \forall\, n \geq 1$
2. 0,666 bit/sec
3. one second.

TRANSMISSION CHANNELS SOLUTIONS

EXERCISE 1

$C_1 = C_2 = 1.585\,\text{bit}$ and $C_3 = 1.23\,\text{bit}$

EXERCISE 2

1. $C_1 = C_2 = 1\,\text{bit}$
2. C_1

EXERCISE 3

1. $2(1-p)\text{bit}$
2. Two bits at a time.
3. $(1-p)$
4. $\dfrac{D_S}{2(1-p)}$

EXERCISE 4

No

EXERCISE 5

1. $\log_2 6 - H_2(p)$
2. 5,936 symbols/sec

3. Yes

4. $\frac{4}{3}$. Yes.

5. By linking 0 with A, 1 with C and 2 with E.

ERROR CORRECTING CODES SOLUTIONS

EXERCISE 1

1. $G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$ and $H^T = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$

Decoding table

syndromes	minimum weight sequences	number of errors
000	00000	0
001	00001	1
010	00010	1
011	00011	2
100	00100	1
101	01000	1
110	00110	2
111	10000	1

2. $P\{\text{error}\} = 1 - \left\{(1-p)^5 + 5p(1-p)^4 + 2p^2(1-p)^3\right\}$

EXERCISE 2

1.

source words	codewords
000	**0**00**00**
001	00101
0**10**	010**10**
011	**01**1**1**1
100	1001**0**
101	101**1**1
110	1100**0**
111	111**01**

2. $G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $H^T = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$

3. No.

4. Yes.

EXERCISE 3

1. Yes.

2. 3.

3. 111.78 Kbits/sec. 1.5.

4. 3.

5. $G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$

6.

codewords	weight
00000	0
01011	3
10110	3
11101	4

7.

Decoding table

syndromes	minimum weight sequences
000	00000
001	00001
010	00010
011	01000
100	00100
101	11000
110	10000
111	10001

One pattern of two errors can be corrected.

8. $P\{\text{error with coding}\} = 0.018 \quad P\{\text{error without coding}\} = 0.097$

BIBLIOGRAPHY

Thomas M.Cover Joy A.Thomas	Elements of Information Theory (John Wiley & Sons)
Robert G. Gallager	Information Theory and Reliable Communication (John Wiley & Sons)
David J.C MacKay	Information Theory, Inference and Learning Algorithms
Mark Nelson	La Compression de Données (Dunod)
John G. Proakis Masoud Salehi	Communications Systems Engineering (Mac Graw Hill)

INDEX