An Introduction to Information Theory and Applications

F. Bavaud J.-C. Chappelier J. Kohlas

version 2.04 - 20050309 - UniFr course

Contents

Notations

$\mathcal{V}_X$: set of values for random variable X.

ε: the empty string.

$E[X]$: expected valued of X.

$P(X = 3)$: general probability function.

$p_X(3)$: probability distribution of a given random variable (here X).
Notice that $p_X(3) = P(X = 3)$

$a_n \approxeq b_n$: asymptotic exponent equivalence: $\lim_{n\to\infty} \frac{1}{n} \log(\frac{a_n}{b_n}) = 0$.

$X := Y$, and $Y =: X$: equal by definition.
Both cases mean "X is by definition equal to Y".

Chapter 1

Module C1: Uncertainty and Information

by Jürg Kohlas

Learning Objectives for Chapter 1

After studying this module you should understand

- why it is important to measure the amount of *uncertainty* in a situation of choice;
- why *entropy* is an appropriate measure of uncertainty;
- how *information* and *uncertainty* are related and why therefore *entropy* plays an important role in measuring information;
- that information is always relative to a precise *question* and to *prior information*.

Introduction

Welcome to this first step into the world of *information theory*. Clearly, in a world which develops itself in the direction of an *information society*, the notion and concept of *information* should attract a lot of scientific attention. In fact, although *pragmatic* information processing in computers, in the Internet and other computer networks develops at an extremely fast pace, the theoretical and conceptual study of what information is, and how it should be treated hardly keeps up with this frantic development.

Information theory, in the technical sense, as it is used today goes back to the work of Claude Shannon and was introduced as a means to study and solve problems of communication or transmission of signals over channels. Although it is quite a narrow view of information, especially focusing on measurement of information content, it must be part of any larger theory of information. Therefore this basic module introduces

the basic elements of information theory as laid down by Shannon and his successors.

But already in this first module we try to open the view on information. We emphasize that information must always be considered with respect to precisely specified questions. The same piece of information may bear on different questions; with respect to each question its information content will be different. For some questions, the content may even be void. The amount contained in a piece of information with respect to a given question will be measured by the reduction, or more generally, the change of uncertainty regarding this question induced by the information. We follow Shannon by measuring uncertainty by the *entropy* and our approach is in the spirit of Shannon also insofar, as information is measured by change of entropy. But by expliciting the question to which the information is applied, we go beyond Shannon.

Also we stress the importance of the prior information, with respect to which information amount is to be measured. Although this is implicit in Shannons approach, expliciting it makes the concept more clear. In fact, with this view it becomes evident, that probabilities are themselves information whose content can be measured by change of entropy.

In the course of the discussions it becomes clear, that information has also an algebraic structure: information can be combined or aggregated, information must be focussed on specified questions. This important aspect however is not treated in depth. This will be reserved to other modules. The same holds true for the application of classical information theory to coding, communication and other domains.

So we wish you a lot of pleasure in studying this module.

1.1 Entropy

Learning Objectives for Section 1.1

After studying this section you should

- know how entropy is defined and what its most important properties are;
- understand why it is an appropriate measure of uncertainty.

1.1.1 Choice and Uncertainty

Learning Objectives for Subsection 1.1.1

After studying this subsection you should

- know how a *situation of choice* is formally described;
- understand how a *game of questions and answers* leads to a measure of the amount of uncertainty in a situation of choice;
- understand why this measure is a *logarithm* and what its *units* are;
- how possible choices can be *coded using the game of questions and answers.*

Any situation of uncertainty can be described as a situation where there are several possibilities, and it is unknown which one will be selected. A typical example related to computers is the question "what will be the next keystroke of a user of a computer". There are, depending on the keyboard, several dozens of possibilities, if combined strokes are allowed. A more complex situation of uncertainty arises, if a whole sequence of keystrokes in a dialog session is considered. Of course there is an abundance of situations of uncertainty in daily life, scientific inquiry, medical diagnostics, statistical inference, criminal investigations, etc. So, there is clearly much interest in studying uncertainty and in measuring the amount of uncertainty in a given situation. In fact the latter is a basic issue in communication and information theory.

We start by a *formal* description of a situation of uncertainty. Suppose there is a case where one of say m different possibilities exist. We denote these possibilities by $e_1, e_2, \ldots, e_m$. Such a set of possibilities, denoted by $S = \{e_1, e_2, \ldots, e_m\}$ is called a *(finite) scheme of choice*. The idea is that somebody, or some process or some mechanism, etc. selects one of these possibilities. The uncertainty arises, because we do not know which one of the m possibilities is selected.

How do we measure the amount of uncertainty in a scheme of choice S? Intuitively, the larger the cardinality $|S|$ of S (the number of elements), the larger the uncertainty. So much seems clear. Then, why not simply take $|S|$ as the measure of uncertainty? It is indeed a possibility. But we prefer another approach. Imagine the following game: I select a possibility of S and you can ask me questions about my choice. However I accept only questions with "yes-no" answers. For all purposes we may assume that the possibilities are represented by numbers $1, 2, \ldots, m = |S|$. So you may ask questions like: is the number you selected odd? is it less than 10? greater than 13? etc. The more questions you need, the greater is the uncertainty. So the idea is to measure the uncertainty by the number of questions you need to find my choice out.

Of course, you should ask clever questions. You may ask whether the number is 1, if no, is it 2, etc. In this way, you may need up to m questions to find my choice out. This is clearly not an optimal way to proceed. However, if you ask first, if the number is smaller than $m/2$, my answer allows you to limit your subsequent search to only half of the initial possibilities. And then you may proceed in a similar manner. So, this seems to be a clever way to find out my choice.

To get a bit more formal assume first, that m is a power of 2, $m = 2^n$. Then we may partition S with the first question (is your choice greater than 2^{n-1}?) into two halves of equal size: $\{1, \ldots, 2^{n-1}\}$ and $\{2^{n-1}+1, \ldots, 2^n\}$. Each half can be further halved by the second question. If the answer to the first question is "no", then the next question determines either $\{1, \ldots, 2^{n-2}\}$ or $\{2^{n-2}+1, \ldots, 2^{n-1}\}$. If the answer to the first question is "yes", then the next question distinguishes between $\{2^{n-1}+1, \ldots, 2^{n-1}+2^{n-2}\}$ and $\{2^{n-1}+2^{n-2}+1, \ldots, 2^n\}$. This process of questions and answers is depicted in Figure 1.1. Each question is represented by a node, starting with the first question. A question node is labelled by the set of possibilities identified so far. So the first node is labelled by the whole set S. Nodes on the first level by the two half sets, on the second level by the four quarter sets, etc. Each possible answer is indicated by an arc leaving the node. We label a "no" answer by a "0" and a "yes" answer by a "1".

The process of dividing sets of possibilities into equal halves ends eventually after exactly n steps with the actual choice. Now, the number n is nothing else than the logarithm of $m = 2^n$ in base 2, that is, $n = \log_2 m$. And this is a lot smaller than m itself.

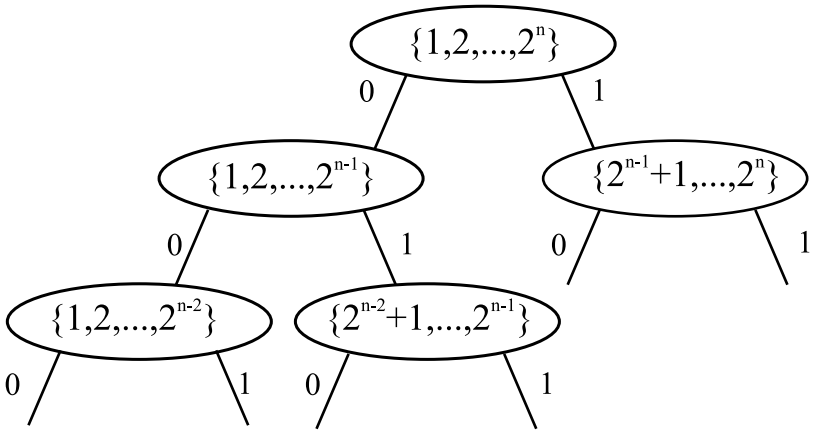


Figure 1.1: The question-answer tree for the binary search for an unknown choice among 2^n elements.

So it seems reasonable to express the amount of uncertainty in a choice system S with $|S| = 2^n$ possibilities with the logarithm in base 2. Lets denote the amount of uncertainty of a choice system S by $h(|S|)$. So we propose $h(|S|) = \log|S|$, at least if $|S|$ is a power of 2. But what about the general case, where the cardinality $|S|$ is any number? We may try the same scheme of questions. The only difference we encounter is, that some question nodes may represent sets of *odd* cardinality, say $2k+1$ for example. Then the questions partition this set into two slightly unequal sets, one with $k+1$, the other with k elements. Figure 1.2 shows this situation schematically.

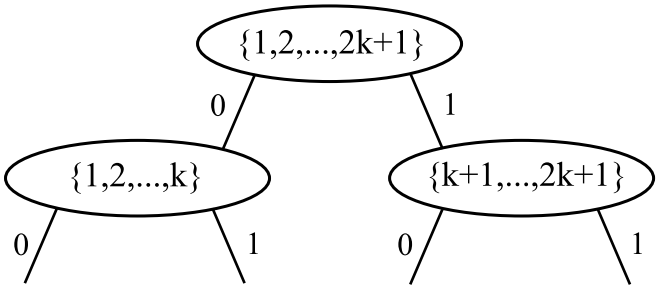


Figure 1.2: The typical node of a question-answer tree for the binary search for an unknown choice in the general case.

If the number of possibilities in the choice system S is between the two powers of 2, $2^n \leq |S| < 2^{n+1}$, then we may either remove some possibilities to get 2^n possibilities, or add some possibilities to get 2^{n+1} possibilities. In the first case we need n questions, in the latter $n+1$ to find out the actual choice. So the amount of uncertainty of S must be somewhere between these two limits. Now, we have $n \leq \log|S| < n+1$. So, we may again take $h(|S|) = \log|S|$ as a measure of the amount of uncertainty in the general case, however this time $\log|S|$ is not necessarily an integer value. We finally adopt the following definition:

Definition 1.1 *The Amount of Uncertainty of a Choice Scheme.* For a choice system S with $|S|$ possible choices, we define the amount of uncertainty $h(|S|)$ by

$$h(|S|) = \log |S|.$$

Example 1.1 (Chess Board) As a first example take an empty chess board. There are exactly $m = 64 = 2^6$ possibilities to place a piece on it. Thus the system of choice can be represented by $S = \{1, 2, \ldots, 64\}$, where each number stands for a field. The amount of uncertainty in the position of a given piece on a chessboard is

$$h(|S|) = \log 64 = 6.$$

As any definition, this one is also arbitrary to a certain degree. Its justification will have to be proved by its usefulness and beauty in applications. And this will be attempted in what follows in this module, as well as in other ones.

By the way, we may play the game of question and answer also with questions having more than 2 possible answers. Suppose the questions have $k > 2$ possible answers. Then each question allows the partition of the set of, say m possibilities into k subsets of approximately m/k elements. So, as above, if $|S| = k^n$, then we need exactly $n = \log_k |S|$ questions. This time we use the logarithm with base k. So, we might also have defined

$$h(|S|) = \log_k |S|.$$

But we have

$$\log_k |S| = \log_k(2) \cdot \log_2 |S|.$$

So, changing the base of the logarithm is like changing the unit of measurement, and thus not really essential. In the future log without indication of the base means by default base 2.

If we have a choice system S and a corresponding question tree (like Figure 1.1), then we have at the same time a *coding* of the possibilities of the choice system. In fact, concatenate the "0" and "1" on the path of the root to the possibility in question. This is a *code* of this possibility. If we use binary questions, then we have a *binary code* for the choice system. Note that the length of the code of each possibility equals either the next smaller or the next greater integer of $h(|S|) = \log |S|$. This is a first hint of the close relation between our measure of uncertainty and coding.

Example 1.2 (Binary Question Tree) A system of choice is given by $S = \{1, 2, 3, 4, 5\}$. Its amount of uncertainty is

$$h(|S|) = \log 5 \approx 2,3219 \text{ bit}.$$

A possible corresponding binary question tree is depicted in Figure 1.3. It is easy to see, that the code 001 represents the possibility $\{2\}$ and its length 3 is the next greater integer of $h(|S|)$. The possibility $\{3\}$ has code length 2, $\{4\}$ has length 2, etc.

Here are now a few very simple properties of our measure of uncertainty $h(|S|)$:

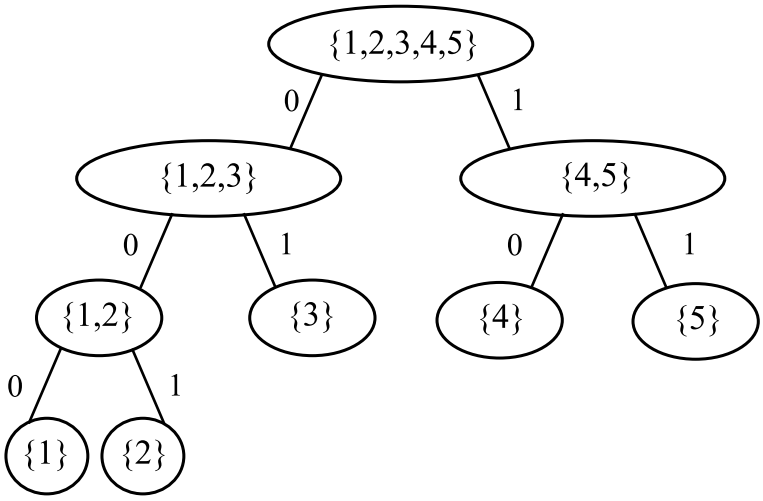


Figure 1.3: Example: A binary question tree of a 5-possibilities system.

1. If S_1 and S_2 are two choice systems and $|S_1| = |S_2|$, then $h(|S_1|) = h(|S_2|)$. Only the number of possibilities in a choice system matters, not their nature.

2. If S_1 and S_2 are two choice systems and $|S_1| < |S_2|$, then $h(|S_1|) < h(|S_2|)$, since the logarithm is a non-decreasing function. That is what we expect: the uncertainty increases with the number of possibilities of a choice.

3. If S_1 and S_2 are two choice systems and S_2 has twice as many possibilities as S_1 ($|S_2| = 2 \cdot |S_1|$), then, using base 2 for the logarithm, we obtain $h(|S_2|) = h(|S_1|)+1$. This follows from the additivity of the logarithm, and from $\log_2 2 = 1$,

$$h(|S_2|) = \log_2 |S_2| = \log_2 (2 \cdot |S_1|) = \log_2 2 + \log_2 |S_1| = 1 + h(|S_1|).$$

4. If S is a choice system with only *two* possibilities, then, with the logarithm in base 2, $h(|S|) = \log_2 2 = 1$. This unit for the measurement is called a *bit* (binary information unit). We shall see that uncertainty is closely related to information, and the latter is measured in the same units as uncertainty. Also here we enter the realm of computers. That is why binary questions are most popular.

If we have two choice systems $S_1 = \{e_{1,1}, e_{1,2}, \ldots, e_{1,n}\}$ and $S_2 = \{e_{2,1}, e_{2,2}, \ldots, e_{2,m}\}$, then the corresponding two choice possibilities can be compounded into a combined system, which contains all $n \cdot m$ pairwise combinations of possible choices,

$$\{(e_{1,1}, e_{2,1}), (e_{1,1}, e_{2,2}), \ldots, (e_{1,2}, e_{2,1}), \ldots, (e_{1,n}, e_{2,m})\}.$$

Such a set of pairs is called a *cartesian product* of the two individual sets, and is written as

$$S_1 \times S_2 = \{(e_{1,1}, e_{2,1}), (e_{1,1}, e_{2,2}), \ldots, (e_{1,2}, e_{2,1}), \ldots, (e_{1,n}, e_{2,m})\}.$$

We call this new choice system a *system of independent choices*. This expresses the idea, that the choice in each of the two system is made *independently* of the choice in the other system, to get the combined choice. How is the amount of uncertainty of such a system of independent choices related to the amount of uncertainty in each of the two choice systems? The simple answer is given in the following theorem.

Theorem 1.1 Additivity of Uncertainty. *The uncertainty of the system of independent choices is the sum of the uncertainty of both simple systems,*

$$h(|S_1 \times S_2|) = h(|S_1|) + h(|S_2|).$$

PROOF The proof is simple, since this is essentially the additivity of the logarithm. In fact,

$$\begin{aligned} h(|S_1 \times S_2|) &= \log|S_1 \times S_2| = \log(|S_1| \cdot |S_2|) \\ &= \log|S_1| + \log|S_2| = h(|S_1|) + h(|S_2|). \end{aligned}$$

This theorem is a strong justification for our definition of the measure of uncertainty: One would expect that the uncertainties in two independent situations add, when they are considered together.

Example 1.3 (Chess Board - Continuation) We return to the chess board situation of example 1.1. We saw, that the amount of uncertainty of the position of a piece on the whole board is $h(|S|) = \log 64 = 6$. In the same way, we see, that the amount of uncertainty of the position of a piece in a single row or column is $\log 8 = 3$. So we get the expected result

$$6 = \log 64 = \log 8 + \log 8.$$

Of course this can be generalized to the combination of more than two independent choices. Let $S_1, S_2, \ldots, S_m$ be m choice systems. Then the cartesian product of m-tuples

$$S_1 \times S_2 \times \cdots \times S_m = \{(e_{1,1}, e_{2,1}, \ldots, e_{m,1}), \ldots\}$$

is the corresponding system of independent choices

Corollary 1.1

$$h(|S_1 \times S_2 \times \cdots \times S_m|) = h(|S_1|) + h(|S_2|) + \cdots + h(|S_m|). \tag{1.1}$$

Example 1.4 (Dice) We throw m times a dice and assume that the throws are independent. This can be modelled by m independent systems of choices $S_1, \ldots, S_m$, where each system contains 6 possibilities. From corollary 1.1 we get that

$$h(|S_1 \times S_2 \times \cdots \times S_m|) = h(|S_1|) + h(|S_2|) + \cdots + h(|S_m|) = m \cdot \log 6.$$

Summary for Section 1.1

- We formalized situations of uncertainty by choice systems S, where one of a finite number of possibilities will be selected, but it is unknown which one;

- The uncertainty associated with a choice system is measured by the (least) number of questions to be asked in order to find out the actual choice. This leads to propose $\log |S|$ as a measure of the uncertainty. With binary questions the unit of measurement is called a *bit*;
- The question game defines a tree, which can be used to define *codes* for the possibilities of the choice system S. The lengths of these codes are approximately equal to the measure of uncertainty $\log |S|$. If binary questions are used we get *binary codes*;
- We found that the uncertainty of systems of independent choices add together.

Control Question 1

For $x > 0$, the logarithm $\log_2(x)$ is

1. always positive;
2. a non decreasing function;
3. maximal in $x = 10$;
4. equal to 0 in $x = 0$;
5. equal to 0 in $x = 1$;
6. equal to 1 in $x = 2$.

Answer

1. That is not correct. The logarithm $\log_2(x)$ takes negative numbers for all $0 < x < 1$. This will play a significant role in the next subsection, where the so-called entropy will be defined as $H(X) = -\sum p_X(x) \log p_X(x)$.
2. That is right. This is an important property of the logarithm.
3. The logarithm has no maximum, so that is wrong.
4. No, the logarithm is not even defined in $x = 0$.
5. That is correct and important. Take as an example the system of choice given by $S = \{1\}$. Since we have only one possibility, there is no uncertainty. In fact, we really obtain $h(|S|) = h(1) = \log 1 = 0$.
6. Yes, that is true. Once more this property is significant. Take as an example a system of choice S with only two possibilities. Then we get $h(|S|) = h(2) = 1$. That is why we call the unit of uncertainty "bit".

Control Question 2

If we have two choice systems $S_1 = \{e_{1,1}, e_{1,2}, \ldots, e_{1,n}\}$ and $S_2 = \{e_{2,1}, e_{2,2}, \ldots, e_{2,m}\}$, then the system of independent choices $S_1 \times S_2$ has

1. $n + m$ elements;
2. $n \cdot m$ elements.

Answer

Since the system of independent choices $S_1 \times S_2$ is given by the cartesian product of the two individual sets, that is

$$S_1 \times S_2 = \{(e_{1,1}, e_{2,1}), (e_{1,1}, e_{2,2}), \ldots, (e_{1,n}, e_{2,m})\},$$

we get $|S_1 \times S_2| = n \cdot m$.

Control Question 3

Given two choice systems S_1 and S_2 with $|S_2| = 2 \cdot |S_1|$, then $h(|S_1 \times S_2|)$ equals

1. $h(|S_1|) + h(|S_2|)$;
2. $1 + 2 \cdot h(|S_1|)$;
3. $\log(|S_1| \cdot |S_2|)$;
4. $h(|S_1|) \cdot h(|S_2|)$;
5. $1 + \frac{1}{2} \cdot h(|S_2|)$.

Answer

1. Due to the additivity of the logarithm, this is correct.
2. That is true; once more because of the additivity of the logarithm.
3. Since $|S_1 \times S_2| = |S_1| \cdot |S_2|$, this one is also correct.
4. The logarithm is additive, but not multiplicative, that means $\log(x \cdot y) \neq \log x \log y$, hence the assertion is incorrect.
5. That is wrong. But we have $h(|S_1 \times S_2|) = 1 + 2 \cdot h(|S_1|)$.

Control Question 4

We are going back to the last control question. Which of the correct assertions will remain true for arbitrary S_1 and S_2 without the property $|S_2| = 2 \cdot |S_1|$.

Answer

1. Remains true.
2. Now, the assertion becomes incorrect, because we can't expect that $h(|S_2|) = h(S_1) + 1$.
3. Remains true.

1.1.2 Choice with Known Probability

Learning Objectives for Subsection 1.1.2

After studying this subsection you should

- understand how uncertainty is affected, if *probabilities* for the possible choices are known;
- know the definition of *entropy* and some of its elementary properties;
- get a first appreciation of entropy as a *measure of uncertainty*.

There are situations, where probabilities are known for the different possibilities which may arise. For example, if we know that a user is typing an English text on the keyboard, then we know that some letters occur more often than others and thus some keys are more likely to be struck. Or, on the level of words, if we know that the user is programming, then we know that some keywords like "if", "then", "else", etc. are more likely to be typed in than most other combinations of letters. We shall see in this section how this additional knowledge of probabilities affects the amount of uncertainty in a choice system.

To start, we *formally* introduce probabilities into a choice system $S = \{e_1, e_2, \ldots, e_m\}$ by assigning probabilities p_i to the possibilities e_i for $i = 1, 2, \ldots, m$. These probabilities have to satisfy the following conditions:

$$0 \leq p_i \leq 1, \text{ for } i = 1, 2, \ldots m, \quad \sum_{i=1}^{m} p_i = 1. \tag{1.2}$$

The second condition expresses the fact that exactly one of the m possibilities must be selected. The choice system S together with the set of probabilities $P = \{p_1, p_2, \ldots, p_m\}$ forms a probabilistic choice system. Here is the formal definition:

Definition 1.2 *Probabilistic Choice System.* If S is a choice system, and P a set of probabilities on S satisfying conditions (1.2), then the pair (S, P) is called a *probabilistic choice system.* ♦

If $E \subseteq S$ is a subset of S, called an *event* in the language of probability theory, then its probability is given by

$$p(E) = \sum_{e_i \in E} p_i.$$

What is then the amount of uncertainty in a probabilistic choice system? We may attempt the same game of questions and answers as in the previous question. However, now it is no more clever to partition the set of possibilities into subsets of equal size, because this neglects the probabilities. Suppose for example that one possibility, say e_1, is much more likely than all others. Then of course we should first ask, whether this is the actual possibility. There is a big chance that we hit the actual possibility with just one question. Only if the answer is "no" do we have to continue. Let's look at an example.

Example 1.5 (Trees related to a Probabilistic Choice System) Given a probabilistic choice system (S, P) with $S = \{e_1, e_2, \ldots, e_8\}$ and $P = \{0.3, 0.2, 0.1, 0.05, 0.05, 0.1, 0.15, 0.05\}$. A corresponding binary- and an alternative, better tree of (S, P) are depicted in Figure 1.4 and Figure 1.5. A simple computation shows, that the expected word length in the binary tree is 3 and 2.75 in the better one.

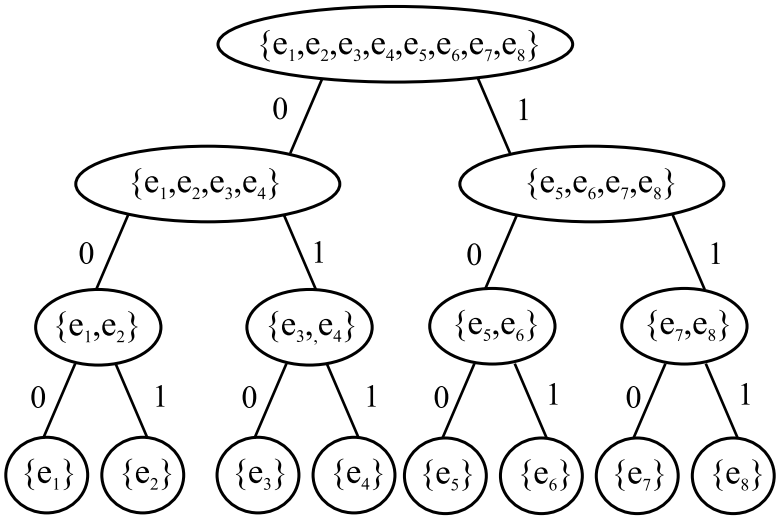


Figure 1.4: The binary tree of our example with expected worth length 3.

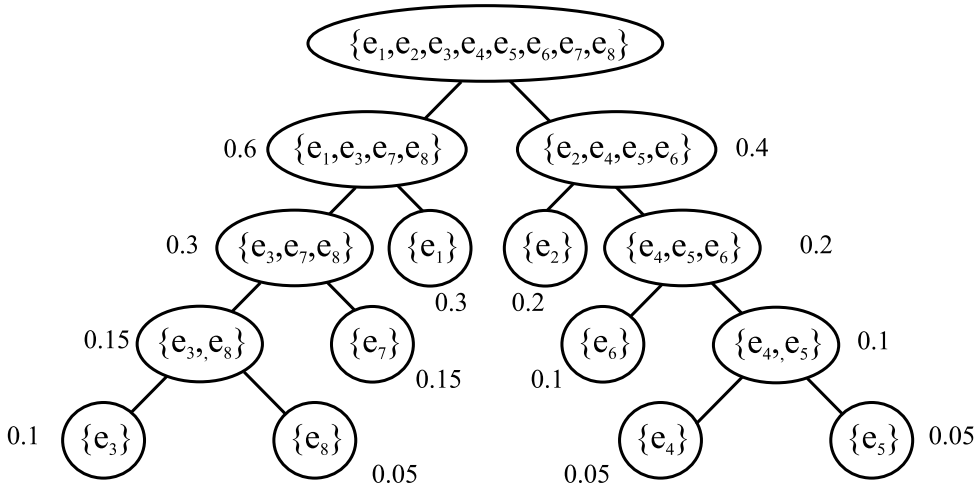


Figure 1.5: In this alternative tree the expected worth length reduces to 2.75.

As the example shows, we should try to select our questions such as to minimize the expected, or average, number of questions needed. Now, this is not a trivial task. Nevertheless, the solution to this problem is known and very much used in *coding theory* . The key idea is to partition a set not into subsets of equal cardinality, but into subsets of equal probability. It is especially known from coding theory that the expected number of questions is then approximately

$$-\sum_{i=1}^{m} p_i \log p_i.$$

This quantity is called *entropy* and we propose to use it as a measure of the amount of uncertainty in a probabilistic choice system. We do not exclude that some of the probabilities p_i vanish, $p_i = 0$. For this case we adopt the convention that $0 \log 0 = 0$, which is reasonable since $\lim_{x \to 0} x \log x = 0$.

Definition 1.3 *The Amount of Uncertainty of a Probabilistic Choice System.* Let (S, P) be a probabilistic choice system. Then we define the amount of uncertainty in (S, P) to be the *entropy*

$$H(P) = -\sum_{i=1}^{m} p_i \log p_i. \tag{1.3}$$

Again there is some amount of arbitrariness in this definition. The base k of the logarithm in the entropy corresponds, as in the previous section, to the number k of possible answers for each question in the game of questions and answers. We leave this open, as a change of the base corresponds only to a change of the unit. As before, the binary case is most popular, with the unit *bit*. This is the base we usually use in examples.

Example 1.6 (Trees related to a Probabilistic Choice System - Continuation) We continue the example 1.5 by computing the amount of uncertainty in the given probabilistic choice system, that means computing the entropy.

$$\begin{aligned} H(P) &= -\sum_{i=1}^{8} p_i \log p_i \\ &= -0.3 \cdot \log 0.3 - 0.2 \cdot \log 0.2 - 0.2 \cdot \log 0.1 \\ &\quad -0.15 \cdot \log 0.05 - 0.15 \cdot \log 0.15 \\ &\approx 2.7087 \text{ bit} \end{aligned}$$

Thus $H(P)$ is less than the expected word length in the better tree.

Entropy is a really fundamental notion of information and communication theory as the rest of this course will demonstrate. Therefore, it is worthwhile studying its properties. Note that we sometimes write

$$H(P) = H(p_1, p_2, \ldots, p_m),$$

if $P = \{p_1, p_2, \ldots, p_m\}$.

First we make the connection between the general notion of entropy as introduced here and the measure of uncertainty for choice systems without probabilities as defined in the previous subsection. We see that if P is the uniform distribution over n choices, then

$$H\left(\frac{1}{n}, \ldots, \frac{1}{n}\right) = -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} = -\log \frac{1}{n} = \log n = h(n).$$

Property 1.1 *The entropy of a uniform probability distribution over n possibilities equals the measure of uncertainty of the corresponding choice system without probability.*

This is an expression of Laplace's *principle of insufficient reason*, which says, if you know nothing other, assume equal probabilities. In this context this works nicely. Hence, it turns out that entropy also covers in this sense the particular case of choice systems without probabilities.

For a given choice system S with $|S| = n$, intuitively, we have maximal uncertainty, if we know no probabilities, or if we assume uniform (equal) probabilities over all possibilities,

$$H(P) \leq h(|S|) = \log n.$$

This is in fact true and can be proved.

In order to prove this result, we need the following lemma.

Lemma 1.1 *Let $p_1, p_2, \ldots, p_m$ and $q_1, q_2, \ldots, q_m$ be two probability distributions over the same number of m possibilities. Then*

$$\sum_{i=1}^{m} p_i \log q_i \leq \sum_{i=1}^{m} p_i \log p_i \tag{1.4}$$

and equality holds if, and only if, $p_i = q_i$.

PROOF We have $\log x = \log e \cdot \ln x$, where ln denotes the natural logarithm to the base e. $\ln x$ is a convex function, that is, its graph is below its tangent in all points (see Figure 1.6). If we take the derivative of $\ln x$ in the point $x = 1$, then this gives us $\ln x \leq x - 1$ with equality if, and only if, $x = 1$. Therefore, we have

$$\ln \frac{q_i}{p_i} \leq \frac{q_i}{p_i} - 1,$$

hence

$$\sum_{i=1}^{m} p_i \ln \frac{q_i}{p_i} \leq \sum_{i=1}^{m} q_i - \sum_{i=1}^{m} p_i = 1 - 1 = 0.$$

From this we conclude that

$$\sum_{i=1}^{m} p_i \log q_i - \sum_{i=1}^{m} p_i \log p_i = \log e \sum_{i=1}^{m} p_i \ln \frac{q_i}{p_i} \leq 0.$$

This shows that (1.4) holds with equality if, and only if, $q_i/p_i = 1$ for all i, i.e. $p_i = q_i$. ■

We apply this lemma now,

$$H(P) - \log m = -\sum_{i=1}^{m} p_i \log p_i + \sum_{i=1}^{m} p_i \log \frac{1}{m} \leq 0,$$

with equality if, and only if, $p_i = 1/m$. We have proved the following theorem:

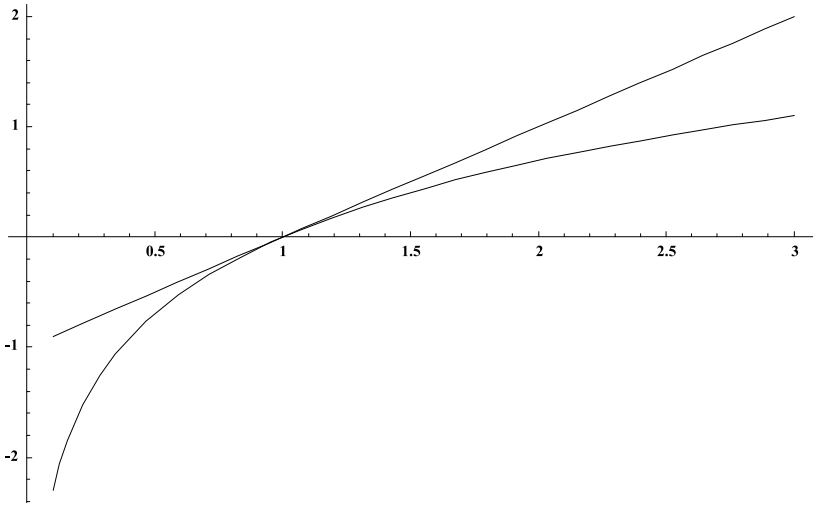


Figure 1.6: Convexity of the function $y = \ln x$ and the tangent at the point $x = 1$.

Theorem 1.2

$$\max H(P) = \log m, \tag{1.5}$$

where the maximum is taken over all probability distributions P for m possibilities. This maximum is reached only for the equiprobable distribution.

We add further elementary properties of entropy.

1. If (S_1, P_1) and (S_2, P_2) are two probabilistic choice systems with $|S_1| = |S_2|$ and $P_1 = P_2$, then $H(P_1) = H(P_2)$. This says that the entropy depends only on the probability distribution, but not on the nature of the possibilities e_i in the choice system. This follows directly from the definition of the entropy (1.3) which depends only on the probabilities p_i.

2. We have

$$H(p_1, p_2, \ldots, p_n) = H(p_1, p_2, \ldots, p_n, 0).$$

   This comes from the convention $0 \cdot \log 0 = 0$. It says that possibilities with vanishing probability are irrelevant to the amount of uncertainty. This is reasonable, since we may be sure that such possibilities are never selected.

3. Consider a two stage scheme as depicted in Figure 1.7. In the first stage one of two possibilities are selected with probabilities p and $q = 1 - p$. If in the first stage the first possibility is selected, then in the second stage one of n possibilities is selected with probabilities p_i/p. If the second possibility is selected in the first stage, then one of m possibilities is selected in the second stage with probabilities q_i/q. Here it is assumed that

$$\sum_{i=1}^{n} p_i = p, \qquad \sum_{i=1}^{m} q_i = q.$$

Note that this implies

$$p_1 + p_2 + \cdots p_n + q_1 + q_2 + \cdots q_m = 1,$$

i.e. $\{p_1, \ldots, p_n, q_1, \ldots, q_m\}$ is a probability distribution on $n+m$ elements. Then we have the following equality between entropies of the two stages:

$$\begin{aligned} & H(p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_m) \\ &= H(p, q) + pH(\frac{p_1}{p}, \frac{p_2}{p}, \ldots, \frac{p_n}{p}) + qH(\frac{q_1}{q}, \frac{q_2}{q}, \ldots, \frac{q_m}{q}). \end{aligned}$$

This can be verified, by the definition of entropy.

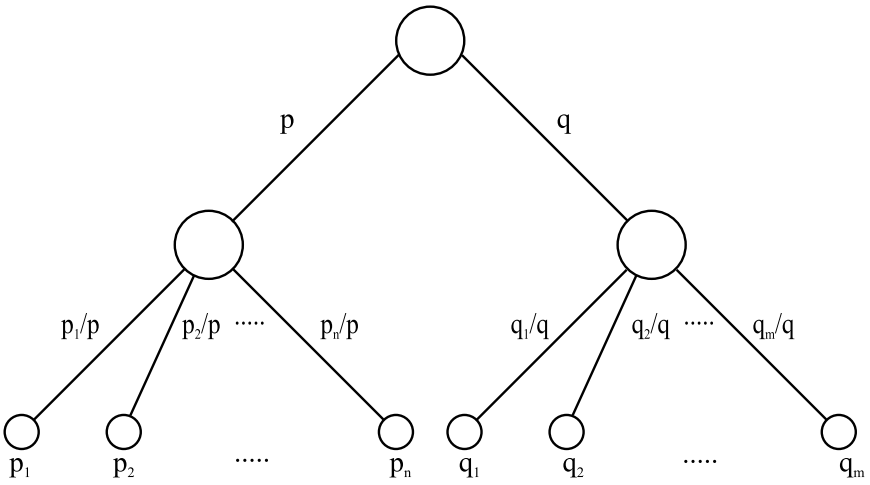


Figure 1.7: A two-stage probabilistic choice where in the first stage one of two possibilities are selected with probabilities p and $q = 1 - p$ and in the second stage either one of n possibilities with probability p_i/p or one of m possibilities with probability q_i/q.

We add a number of more technical properties of entropy.

Proposition 1.1 *1.* $H(p_1, p_2, \ldots, p_n) = H(p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(n)})$ *for every permutation* π.

2. $H(p_1, p_2, \ldots, p_n)$ *is continuous in all its variables.*

3. We have the equation

$$\begin{aligned} H(p_1, \ldots, p_n) &= H(p_1 + p_2, p_3, \ldots, p_n) \\ &\quad + (p_1 + p_2) H(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2}), \end{aligned}$$

for every probability distribution $p_1, \ldots, p_n$ *with* $n \geq 2$.

4. $H(\frac{1}{n}, \ldots, \frac{1}{n})$ *is monotone increasing with* n.

These propositions are very easy to prove. Their importance resides in the fact, that they are *characterizing* properties for entropy. That, is, when we impose these four

reasonable conditions on a measure of uncertainty, then we necessarily get the entropy for this measure. We shall return to this interesting point in section 1.1.4.

PROOF (1) Follows directly from the definition of entropy and commutativity of addition.

(2) Follows from the fact that the logarithm is a continuous function.

(3) Needs some simple computations,

$$\begin{aligned}
&H(p_1, p_2, \ldots, p_n) \\
&\quad = \quad -p_1 \log p_1 - p_2 \log p_2 - \sum_{i=3}^{n} p_i \log p_i \\
&\quad = \quad -(p_1 + p_2) \log(p_1 + p_2) - \sum_{i=3}^{n} p_i \log p_i \\
&\qquad +p_1 \log(p_1 + p_2) - p_1 \log p_1 + p_2 \log(p_1 + p_2) - p_2 \log p_2 \\
&\quad = \quad H(p_1 + p_2, p_3, \ldots, p_n) - (p_1 + p_2) \left(\frac{p_1}{p_1 + p_2} \log \frac{p_1}{p_1 + p_2} + \frac{p_2}{p_1 + p_2} \log \frac{p_2}{p_1 + p_2} \right) \\
&\quad = \quad H(p_1 + p_2, p_3, \ldots, p_n) + (p_1 + p_2) H(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2}).
\end{aligned}$$

(4) Follows since $H(\frac{1}{n}, \ldots, \frac{1}{n}) = \log n$ and the logarithm is monotone increasing. ■

For further reference we introduce an alternative notion. A probabilistic choice scheme (S, P) may also be represented by a *finite random variable* X, which takes values e_i from $S = \{e_1, e_2, \ldots, e_m\}$. The probability that $X = e_i$ is then $p_X(e_i)$ and P is called the *probability* density of X. Inversely, each finitely-valued random variable gives rise to probabilistic choice systems. Formally, a random variable with values in S is an application of some sample space Ω into S. A probability distribution in S is then induced by

$$p_X(x) = \sum_{\omega \in \Omega : X(\omega) = x} p(\omega),$$

if $\{p(\omega) : \omega \in \Omega\}$ are the probabilities defined in the finite sample space Ω. The set $\{p_X(x) : x \in S\}$ then defines the probabilities on the choice space S. So we may as well speak of random variables instead of probabilistic choice systems, and define accordingly the entropy of random variable X with values in S by

$$H(X) = -\sum_{x \in S} p_X(x) \log p_X(x).$$

This then measures the uncertainty associated with random variable X. In what follows this will often be a more convenient way to look at things.

Example 1.7 (Bernoulli) Let X be a binomial random variable representing n Bernoulli trials, that is with

$$p_X(x) = \binom{n}{x} p^x (1 - p)^{n-x}.$$

The entropy of X is given by

$$H(X) = -\sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} \cdot \log\left(\binom{n}{i} p^i (1-p)^{n-i}\right).$$

Let us take $n = 4$ and $p = q = 0.5$. Hence

$$\begin{aligned} H(X) &= -\sum_{i=0}^{4} \binom{4}{i} (0.5)^i \cdot (0.5)^{4-i} \cdot \log\left(\binom{4}{i} (0.5)^i \cdot (0.5)^{4-i}\right) \\ &= -0.1250 \log 0.0625 - 0.5 \log 0.25 - 0.375 \log 0.375 \\ &\approx 2.0306 \text{ bit.} \end{aligned}$$

To conclude this section, we consider a random variable X associated with a probabilistic choice situation (S, P). Assume that an event $E \subseteq S$ is observed. How does this affect the uncertainty? This event induces a new probabilistic choice situation (E, P_E). Here P_E refers to the conditional probabilities

$$p_{X|E}(x) = \frac{p_X(x)}{p_X(E)}, \text{ for all } x \in E.$$

The uncertainty related to this new situation is

$$H(X|E) = -\sum_{x \in E} p_{X|E}(x) \log p_{X|E}(x).$$

This is also called the conditional entropy of X given E.

Example 1.8 (Conditional Entropy) Let X be a random variable related to the probabilistic choice situation (S, P) given by $S = \{1, 2, 3, 4\}$, $P = \{0.5, 0.25, 0.125, 0.125\}$, and $E = \{1, 3\}$ an event. Thus

$$\begin{aligned} H(X) &= -0.5 \log 0.5 - 0.25 \log 0.25 - 0.125 \log 0.125 - 0.125 \log 0.125 \\ &= 1.75\text{bit.} \end{aligned}$$

With $p_X(E) = 0.625$, $p_{X|E}(1) = p_X(1)/p_X(E) = 0.8$ and $p_{X|E}(3) = p_X(3)/p_X(E) = 0.2$ we obtain

$$\begin{aligned} H(X|E) &= -p_{X|E}(1) \log p_{X|E}(1) - p_{X|E}(3) \log p_{X|E}(3) \\ &= -0.8 \log 0.8 - 0.2 \log 0.2 \\ &\approx 0.7219 \text{ bit.} \end{aligned}$$

Summary for Section 1.1

- We have seen that for a probabilistic choice, represented by a *probabilistic choice system*, a different strategy in the game of questions and answer must be used: rather than partitioning the set of possibilities into subsets of equal cardinality, we partition it into subsets of nearly equal probability. This leads to the *entropy* as approximately the expected number of questions and thus as an appropriate measure of uncertainty;

- We have seen that the uncertainty of a choice system equals the entropy of a probabilistic choice system with equal probabilities, that is a *uniform probability distribution*. This corresponds to Laplace's principle of insufficient reason. So the concept of entropy also covers the case of non probabilistic choice systems;
- In fact, equal probabilities, or choice systems without known probabilities represent, for a set S of a given cardinality, the largest uncertainty;
- Entropy depends only on the probability distribution of a choice system, but not on the nature of the possibilities;
- We saw some simple properties of entropy which characterize the concept entropy.

Control Question 5

Given a probabilistic choice system (S, P) by $S = \{e_1, e_2, \ldots, e_n\}$ and $P = \{p_1, p_2, \ldots, p_n\}$. Then, $H(P)$

1. $= -\sum_{i=1}^{n} p_i \log p_i$
2. $= h(n)$
3. $\leq \log n$
4. $\leq h(|S|)$
5. > 0

Answer

1. This is correct. It is simply the definition of the amount of uncertainty of a probabilistic choice system, also known as the entropy.
2. In general, this assertion is wrong. But it becomes correct if P is the uniform probability distribution, that is if $p_1 = \ldots = p_n = 1/n$.
3. We saw, that if we assume uniform probabilities for all possibilities, we have maximal entropy, so this is correct.
4. Since $h(|S|) = \log n$, this case equals the case above. Thus, this assertion is also true.
5. Take as an example the probability distribution $p_1 = 1$, $p_2, \ldots, p_n = 0$. The entropy is then $H(P) = \log 1 = 0$. This counter-example shows, that the proposition is wrong. However, $H(P) \geq 0$ holds for all probability distributions P.

Control Question 6

Given the binary tree as depicted in figure 1.8. Compute

1. the expected word length;
2. the entropy.

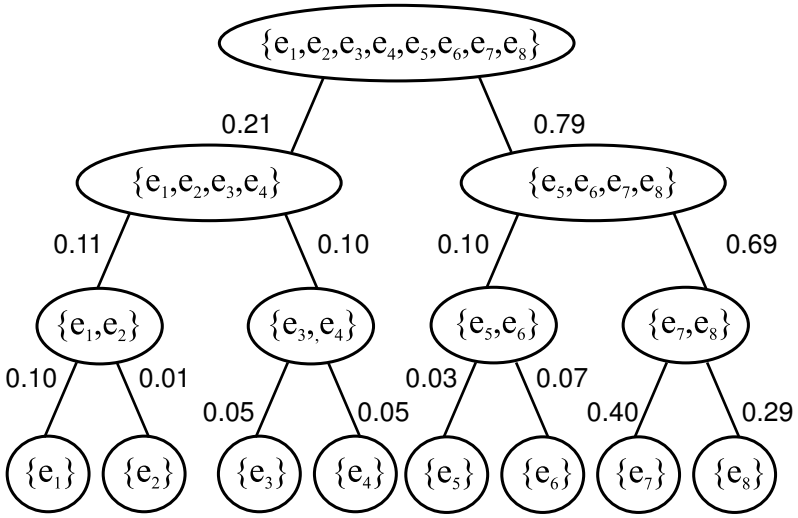


Figure 1.8: Compute the expected word length and the entropy in this binary tree.

Answer

1. The expected word length is equal to

$$3 \cdot (0.1 + 0.01 + 0.05 + 0.05 + 0.03 + 0.07 + 0.4 + 0.29) = 3.$$

   This is not a surprise, since the tree is equilibrated.

2. For the entropy we get

$$-0.1 \log 0.1 - 0.01 \log 0.01 - 0.1 \log 0.05$$
$$-0.03 \log 0.03 - 0.07 \log 0.07 - 0.4 \log 0.4 - 0.29 \log 0.29 \approx 2.2978.$$

Control Question 7

Given a probabilistic choice system (S, P) by $S = \{e_1, e_2, \ldots, e_n\}$ and $P = \{p_1, p_2, \ldots, p_n\}$. Then, $H(p_1, p_2, \ldots, p_n)$

1. $= H(0, p_1, \ldots, p_n)$;
2. $= H(p_1 + p_2, p_3, \ldots, p_n) + (p_1 + p_2) H(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2})$;
3. $= H(p_1 + p_n, p_2, \ldots, p_{n-1}) + (p_1 + p_n) H(\frac{p_1}{p_1+p_n}, \frac{p_n}{p_1+p_n})$;
4. $= \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$.

Answer

1. Follows directly from the convention $0 \cdot \log 0 = 0$.
2. That is correct (see proposition 1.1).
3. Same case as above (use the permutation property).
4. Since we have $\log \frac{1}{p_i} = \log 1 - \log p_i = -\log p_i$, the assertion is correct.

Control Question 8

Let X be a random variable related to a probabilistic choice situation (S, P) and E an event $E \subseteq S$. Then $H(X|E) \leq H(X)$. Is this assertion correct?

Answer

No, the assertion is incorrect. Here is a counter-example:

Let $S = \{e_1, e_2, e_3\}$, $p_1 = 0.99$, $p_2 = 0.005$, $p_3 = 0.005$ and $E = \{e_2, e_3\}$. Hence we get $H(X) = -0.99 \log 0.99 - 0.01 \log 0.005 \approx 0.0908$ and $H(X|E) = -\log 0.5 = 1$.

1.1.3 Conditional Entropy

Learning Objectives for Subsection 1.1.3

After studying this section you should

- know how the entropy of compound choice systems or multidimensional variables is related to the entropy of the components or single variables;
- understand how then knowledge of the choice in one component or the value of one variable affects the uncertainty of the remaining components or variables.

We start by considering two choice systems S_1 and S_2 and the associated system of independent choices $S_1 \times S_2 = \{(e_{1,1}, e_{2,1}), (e_{1,1}, e_{2,2}), \ldots, (e_{1,n}, e_{2,m})\}$. By affecting probabilities $p_{i,j}$ to the compound choice $(e_{1,i}, e_{2,j})$, we extend the system of independent choices to a compound probabilistic choice system $(S_1 \times S_2, P)$, where $P = \{p_{i,j}; i = 1, 2, \ldots, n; j = 1, 2, \ldots, m\}$. We must have

$$0 \leq p_{i,j}, \qquad \sum_{i=1}^{n} \sum_{j=1}^{m} p_{i,j} = 1.$$

This is a two-dimensional probability distribution. We may compute the two marginal distribution $P_1 = \{p_1^{(1)}, p_2^{(1)}, \ldots, p_n^{(1)}\}$, and $P_2 = \{p_1^{(2)}, p_2^{(2)}, \ldots, p_m^{(2)}\}$, defined by

$$p_i^{(1)} = \sum_{j=1}^{m} p_{i,j}, \quad p_j^{(2)} = \sum_{i=1}^{n} p_{i,j}. \tag{1.6}$$

This gives us then two associated probabilistic choice systems (S_1, P_1) and (S_2, P_2).

We shall introduce a random variable for each probabilistic choice system as explained at the end of the previous subsection. So let X be associated with the system (S_1, P_1) and Y with the system (S_2, P_2). The pair of variables (X, Y) is then associated with the compound probabilistic system $(S_1 \times S_2, P)$. We have the two-dimensional probability distribution $p_{(X,Y)}(e_{1,i}, e_{2,j}) = p_{i,j}$ for the pair of random variables (X, Y). Variable X has the marginal distribution $p_X(e_{1,i}) = p_i^{(1)}$ and Y the marginal distribution $p_X(e_{2,j}) = p_j^{(2)}$. We remind you that two probabilistic choice systems, or two random variables X and Y are called *independent*, if, and only if,

$$p_{X,Y}(x, y) = p_X(x) \cdot p_Y(y), \text{ for all pairs } (x, y) \in S_1 \times S_2.$$

We have three different entropies associated with the three probabilistic choice systems: The two single variables X and Y and the two-dimensional variable (X, Y),

$$\begin{aligned} H(X,Y) &= -\sum_{x\in S_1}\sum_{y\in S_2} p_{X,Y}(x,y)\log p_{X,Y}(x,y), \\ H(X) &= -\sum_{x\in S_1} p_X(x)\log p_X(x), \\ H(Y) &= -\sum_{y\in S_2} p_Y(y)\log p_Y(y). \end{aligned}$$

Example 1.9 (Compound Probabilistic Choice System) Given a compound system of independent choices

$$\begin{aligned} S_1 \times S_2 &= \{(e_{1,1}, e_{2,1}), (e_{1,1}, e_{2,2}), (e_{1,2}, e_{2,1}), (e_{1,2}, e_{2,2})\}, \\ P &= \{0.5, 0.1, 0.3, 0.1\}, \end{aligned}$$

and two random variables (X, Y) associated with $(S_1 \times S_2, P)$. It is easy to identify the 'single' choice systems

$$S_1 = \{e_{1,1}, e_{1,2}\}, \quad S_2 = \{e_{2,1}, e_{2,2}\}.$$

Applying (1.6) gives us the two marginal distributions $P_1 = \{0.6, 0.4\}$ and $P_2 = \{0.8, 0.2\}$. We are now able to compute the entropies

$$\begin{aligned} H(X,Y) &= -\sum_{x\in S_1}\sum_{y\in S_2} p_{X,Y}(x,y)\log p_{X,Y}(x,y) \\ &= -0.5\cdot\log 0.5 - 0.1\cdot\log 0.1 - 0.3\cdot\log 0.3 - 0.1\cdot\log 0.1 \\ &\approx 1.6855 \text{ bit}, \\ H(X) &= -\sum_{x\in S_1} p_X(x)\log p_X(x) = -0.6\log 0.6 - 0.4\log 0.4 \\ &\approx 0.9710 \text{ bit}, \\ H(Y) &= -\sum_{y\in S_2} p_Y(y)\log p_Y(y) = -0.8\log 0.8 - 0.2\log 0.2 \\ &\approx 0.7219 \text{ bit}. \end{aligned}$$

The question arises, how the three entropies above are related. The answer is contained in the following theorem

Theorem 1.3 *For any pair of random variables X and Y, we have*

$$H(X,Y) \leq H(X) + H(Y). \tag{1.7}$$

Equality holds if, and only if, X and Y are independent *random variables.*

PROOF This theorem is proved by straightforward calculation and using lemma 1.1:

$$\begin{aligned} & H(X) + H(Y) \\ = & -\left(\sum_x p_X(x) \log p_X(x) + \sum_y p_Y(y) \log p_Y(y) \right) \\ = & -\left(\sum_x \sum_y p_{X,Y}(x,y) \log p_X(x) + \sum_x \sum_y p_{X,Y}(x,y) \log p_Y(y) \right) \\ = & -\left(\sum_x \sum_y p_{X,Y}(x,y) \log p_X(x) \cdot p_Y(y) \right). \end{aligned}$$

Now, lemma 1.1 gives us the following inequality,

$$\begin{aligned} & -\left(\sum_x \sum_y p_{X,Y}(x,y) \log p_X(x) \cdot p_Y(y) \right) \\ & \geq -\left(\sum_x \sum_y p_{X,Y}(x,y) \log p_{X,Y}(x,y) \right) \\ & = H(X,Y). \end{aligned}$$

This proves the inequality (1.7). According to lemma 1.1 we have equality in the last inequality, if, and only if, $p_{X,Y}(x,y) = p_X(x) \cdot p_Y(y)$, which means, that X and Y are independent. ■

This theorem tells us, that entropies of two variables add up to the entropy of the compound, two-dimensional variables, only if the variables are independent. Otherwise there is less uncertainty in the compound situation than in the two simple choice systems. The reason is, that the dependence between the variables (their correlation) accounts for some "common" parts of uncertainty in both single variables.

Example 1.10 (Compound Probabilistic Choice System - Continuation) In example 1.9 we had the random variables X and Y. Check yourself that $H(X,Y) < H(X) + H(Y)$.

Theorem 1.3 generalizes easily to more than two variables. Let $\mathbf{X}$ in a general setting denote the vector $(X_1, X_2, \ldots, X_m)$ of m random variables X_i. This vector random variable has the probability distribution $p_{\mathbf{X}}(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ and each variable X_i has the marginal distribution

$$p_{X_i}(x_i) = \sum_{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_m} p_{\mathbf{X}}(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_m).$$

The random variables $X_1, X_2, \ldots, X_m$ are called (mutually) independent, if, and only

if,

$$p_{\mathbf{X}}(\mathbf{x}) = p_{X_1}(x_1) \cdot p_{X_2}(x_2) \cdots p_{X_m}(x_m)$$

holds. The common entropy of the multidimensional variable $\mathbf{X}$ is defined by

$$H(\mathbf{X}) = -\sum_{\mathbf{x}} p_{\mathbf{X}}(\mathbf{x}) \log p_{\mathbf{X}}(\mathbf{x}).$$

Then theorem 1.3 can be generalized as in the following corollary.

Corollary 1.2 *For any multidimensional random variable* $\mathbf{X} = (X_1, X_2, \ldots, X_m)$ *we have*

$$H(\mathbf{X}) \leq \sum_{i=1}^{m} H(X_i).$$

Equality holds if, and only if, the variables $X_1, X_2, \ldots, X_m$ *are mutually independent.*

PROOF Goes by induction over m. The corollary holds for $m = 2$ according to theorem 1.3. Suppose it holds for m. Then consider the pair of random variables $\mathbf{X}_m = (X_1, X_2, \ldots, X_m)$ and X_{m+1}, such that $\mathbf{X}_{m+1} = (\mathbf{X}_m, X_{m+1})$. Again by theorem 1.3 and by the assumption of induction, we have

$$H(\mathbf{X}_{m+1}) \leq H(\mathbf{X_m}) + H(X_{m+1}) \leq \sum_{i=1}^{m} H(X_i) + H(X_{m+1}) = \sum_{i=1}^{m+1} H(X_i).$$

■

Example 1.11 (Independence) Let $X_1, \ldots, X_n$ be independent random variables supplying the result 0 with probability 0.5 and 1 with probability 0.5, which means

$$p_{X_i}(0) = 0.5, \quad p_{X_i}(1) = 0.5, \quad \text{for } i = 1, \ldots, n.$$

Hence $H(X_1, \ldots, X_n) = n \cdot H(X_1) = n$.

We come back to the case of two variables X and Y. Suppose, we observe the value of one variable, say $Y = y$. How does this affect the uncertainty concerning variable X? We remark that this observation changes the distribution $p_X(x)$ to the *conditional distribution* $p_{X|y}(x, y)$ defined as

$$p_{X|y}(x, y) = \frac{p_{X,Y}(x, y)}{p_Y(y)}.$$

Therefore, we obtain the *conditional entropy* of X, given $Y = y$,

$$H(X|Y = y) = -\sum_{x} p_{X|y}(x, y) \log p_{X|y}(x, y).$$

To simplify the notation we often abbreviate $H(X|Y = y)$ by $H(X|y)$. So, the observation that $Y = y$ changes the uncertainty regarding X from $H(X)$ to $H(X|y)$.

As the following example shows, the new entropy or uncertainty may be smaller or larger than the old one. A particular observation may increase or decrease uncertainty. Note however, that if the two random variables are independent, then we have $p_{X|y}(x,y) = p_X(x)$ for every x and y. In this case we see that

$$H(X|y) = -\sum_x p_X(x) \log p_X(x) = H(X).$$

The uncertainty in X does not change, when a variable Y which is independent of X is observed.

Example 1.12 (Conditional Entropy) Given $p_{X,Y}(0,1) = p_{X,Y}(1,0) = p_{X,Y}(0,0) = \frac{1}{3}$, $p_{X,Y}(1,1) = 0$, $p_X(0) = p_Y(0) = \frac{2}{3}$ and $p_X(1) = p_Y(1) = \frac{1}{3}$. Hence

$$\begin{aligned}
H(X|Y=0) &= -p_{X|y}(0,0)\log p_{X|y}(0,0) - p_{X|y}(1,0)\log p_{X|y}(1,0) \\
&= -\frac{p_{X,Y}(0,0)}{p_Y(0)}\log\frac{p_{X,Y}(0,0)}{p_Y(0)} - \frac{p_{X,Y}(1,0)}{p_Y(0)}\log\frac{p_{X,Y}(1,0)}{p_Y(0)} \\
&= -0.5\log 0.5 - 0.5\log 0.5 \\
&= 1, \\
H(X|Y=1) &= -\frac{p_{X,Y}(0,1)}{p_Y(1)}\log\frac{p_{X,Y}(0,1)}{p_Y(1)} - \frac{p_{X,Y}(1,1)}{p_Y(1)}\log\frac{p_{X,Y}(1,1)}{p_Y(1)} \\
&= 0, \\
H(X) = H(Y) &= -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \approx 0.9183.
\end{aligned}$$

So we get that $H(X|Y=1) < H(X) < H(X|Y=0)$.

In addition to the conditional entropy of X given a particular observation $Y = y$ we may consider the *expected* conditional entropy of X given Y, which is the expected value of $H(X|y)$ relative to y,

$$H(X|Y) = \sum_y p_Y(y) H(X|y).$$

We emphasize the difference between $H(X|y)$, $H(X|E)$ and $H(X|Y)$. In the former case we understand the conditional entropy of the variable X given an observed event $Y = y$ or E. In the latter case we denote the expected conditional entropy.

Note that

$$p_{X,Y}(x,y) = p_Y(y) p_{X|y}(x,y).$$

Therefore, we can develop the expected conditional entropy as follows,

$$\begin{aligned}
H(X|Y) &= \sum_y p_Y(y) H(X|y) \\
&= -\sum_x \sum_y p_Y(y) p_{X|y}(x,y) \log p_{X|y}(x,y) \\
&= -\sum_x \sum_y p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_Y(y)} \\
&= -\sum_x \sum_y p_{X,Y}(x,y) \left(\log p_{X,Y}(x,y) - \log p_Y(y)\right) \\
&= H(X,Y) - H(Y).
\end{aligned}$$

Here we have proven the following very important theorem.

Theorem 1.4 *For a pair of random variables X and Y we always have*

$$H(X,Y) = H(Y) + H(X|Y). \tag{1.8}$$

This theorem tells us, that we can always consider the uncertainty of a pair of variables as the result of a chaining, where we start with the uncertainty of one of the variables, say Y, and add the (expected) conditional uncertainty of the second one, given the first one. Of course, we may start with any of the two variables. Thus,

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

also holds true.

Example 1.13 (Expected Conditional Entropy) We continue example 1.12 by computing the expected conditional entropy of X given Y and the compound entropy of (X,Y).

$$\begin{aligned} H(X|Y) &= p_Y(0)H(X|Y=0) + p_Y(1)H(X|Y=1) = \frac{2}{3} \text{ bit} \\ H(X,Y) &= -\log\frac{1}{3} \approx 1.5850 \text{ bit.} \end{aligned}$$

As you can check $H(X,Y) = H(Y) + H(X|Y)$.

In communication theory, channels for the transmission of signals are considered. Suppose that at the input random signs from a certain choice system I appear with known probabilities. This then defines a random variable X. During a transmission an input sign can be changed into some output sign from an output choice system O. Of course there must be some dependence between the sign at the input and the sign at the output. If we denote the output sign by the variable Y, then this dependence is described by the conditional probabilities $p_{Y|x}(y,x)$, where

$$0 \le p_{Y|x}(y,x) \text{ for all } x \in I, y \in O, \quad \sum_y p_{Y|x}(y,x) = 1 \text{ for all } x \in I.$$

This is called the *transmission matrix*. Figure 1.9 shows this channel system. Then the equation

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

says that the whole uncertainty in the system is composed of the uncertainty of the input signal $H(X)$ and the transmission uncertainty over the channel $H(Y|X)$.

Figure 1.9: Transmission channel.

Example 1.14 (Symmetric Binary Channel) A simple symmetric binary chan-

nel with random variables X for the input and Y for the output is given by the following transmission matrix:

$$\mathbf{P} = \begin{pmatrix} p_{Y|x}(0,0) & p_{Y|x}(1,0) \\ p_{Y|x}(0,1) & p_{Y|x}(1,1) \end{pmatrix} = \begin{pmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{pmatrix}$$

Thus, the probability of a transmission error is ε. Let $p_X(0) = p$ and $p_X(1) = q = 1 - p$. Hence $H(X) = -p\log p - (1-p)\log(1-p)$ and

$$\begin{aligned} p_Y(0) &= p_{Y|x}(0,0)\cdot p_X(0) + p_{Y|x}(0,1)\cdot p_X(1) \\ &= (1-\varepsilon)p + \varepsilon(1-p), \\ p_Y(1) &= p_{Y|x}(1,0)\cdot p_X(0) + p_{Y|x}(1,1)\cdot p_X(1) \\ &= \varepsilon p + (1-\varepsilon)(1-p). \end{aligned}$$

With

$$\begin{aligned} H(Y|0) &= -p_{Y|x}(0,0)\log p_{Y|x}(0,0) - p_{Y|x}(1,0)\log p_{Y|x}(1,0) \\ &= -(1-\varepsilon)\log(1-\varepsilon) - \varepsilon\log\varepsilon, \\ H(Y|1) &= -p_{Y|x}(0,1)\log p_{Y|x}(0,1) - p_{Y|x}(1,1)\log p_{Y|x}(1,1) \\ &= -\varepsilon\log\varepsilon - (1-\varepsilon)\log(1-\varepsilon) = H(Y|0), \end{aligned}$$

we obtain

$$\begin{aligned} H(Y|X) &= p_X(0)H(Y|0) + p_X(1)H(Y|1) = pH(Y|0) + (1-p)H(Y|0) \\ &= H(Y|0) = H(Y|1). \end{aligned}$$

This is not a surprise, since the channel is symmetric. With a certain amount of effort you can show that $H(X,Y) = H(X) + H(Y|X)$. Let us consider now a numerical example. Given $\varepsilon = 0.1$, $p_X(0) = p = 0.2$ and $p_X(1) = q = 0.8$. Thus $H(X) = -0.2\log 0.2 - 0.8\log 0.8 \approx 0.7219$ bit,

$$\begin{aligned} p_Y(0) &= 0.9\cdot 0.2 + 0.1\cdot 0.8 = 0.26, \\ p_Y(1) &= 0.1\cdot 0.2 + 0.9\cdot 0.8 = 0.74, \end{aligned}$$

and $H(Y|X) = H(Y|0) = -0.9\log 0.9 - 0.1\log 0.1 \approx 0.4690$ bit. Since

$$\begin{aligned} p_{X,Y}(0,0) &= p_{Y|x}(0,0)p_X(0) = 0.9\cdot 0.2 = 0.18, \\ p_{X,Y}(1,0) &= p_{Y|x}(0,1)p_X(1) = 0.1\cdot 0.8 = 0.08, \\ p_{X,Y}(0,1) &= p_{Y|x}(1,0)p_X(0) = 0.1\cdot 0.2 = 0.02, \\ p_{X,Y}(1,1) &= p_{Y|x}(1,1)p_X(1) = 0.9\cdot 0.8 = 0.72, \end{aligned}$$

we obtain finally $H(X,Y) = -0.18\log 0.18 - 0.08\log 0.08 - 0.02\log 0.02 - 0.72\log 0.72 \approx 1.1909$ bit.

Theorem 1.4 again generalizes easily to a sequence of more than two variables.

Contrary to the conditional entropy of X, given an observation $Y = y$, the expected conditional entropy of X given Y is always smaller or at most equal to the entropy of X. So, on the average, an observation of Y does decrease the uncertainty of X.

Corollary 1.3 *For any pair of random variables X and Y, we have*

$$H(X|Y) \leq H(X). \tag{1.9}$$

Equality holds, if, and only if, X and Y are independent.

PROOF To prove inequality (1.9) we use the chaining rule and theorem 1.3.

$$H(X|Y) = H(X,Y) - H(Y) \leq H(X) + H(Y) - H(Y) = H(X).$$

With equality if and only if X and Y are independent. ■

If X and Y are independent, then observing any one of these two variables does not change the uncertainty of the other one. That is, intuitively in this case, one variable can not give information about the other one.

Corollary 1.4 *Let $X_1, X_2, \ldots, X_m$ be random variables. Then*

$$\begin{aligned} &H(X_1, X_2, \ldots, X_m) \\ &\quad = H(X_1) + H(X_2|X_1) + \cdots + H(X_m|X_1, X_2, \ldots, X_{m-1}). \end{aligned} \tag{1.10}$$

PROOF The proof is by induction. It holds for $m = 2$ from theorem 1.4. Suppose it holds for some m. Then, let $\mathbf{X}_m = (X_1, X_2, \ldots, X_m)$. From theorem 1.4 and the assumption of induction, we obtain that

$$\begin{aligned} &H(X_1, X_2, \ldots, X_m, X_{m+1}) = H(\mathbf{X}_m, X_{m+1}) \\ &\quad = H(\mathbf{X}_m) + H(X_{m+1}|\mathbf{X}_m) = H(X_1, X_2, \ldots, X_m) + H(X_{m+1}|X_1, X_2, \ldots, X_m) \\ &\quad = H(X_1) + H(X_2|X_1) + \cdots \\ &\qquad \cdots + H(X_m|X_1, X_2, \ldots, X_{m-1}) + H(X_{m+1}|X_1, X_2, \ldots, X_m). \end{aligned}$$

So equation (1.10) holds for $m + 1$, hence for all m. ■

(1.10) is called the (generalized) *chaining rule*. It is especially very important for communication theory.

Summary for Section 1.1

- We found that the joint entropy of several random variables is always less or equal to the sum of the entropies of the individual variable. It equals the sum only if the variables are *independent*.
- The conditional entropy measures the uncertainty of a variable, when the value of another variable is observed. This uncertainty may, depending on the observation, increase or decrease. The expected conditional entropy however is always smaller than the original entropy. Conditional entropy equals unconditional entropy, if the random variables are independent.

Control Question 9

Relate

1. $H(X|y)$;
2. $H(X|Y)$;
3. $H(X|E)$;

with

a. expected conditional entropy;

b. entropy conditioned on an observed event.

Answer

$H(X|y)$ and $H(X|E)$ are denoting the entropy conditioned on an observed event. Whereas $H(X|Y)$ is the expected conditional entropy.

Control Question 10

$H(X)$ may be

1. $< H(X|Y)$;
2. $< H(X|y)$;
3. $> H(X|y)$;
4. $= H(X|y)$;
5. $= H(X|Y)$.

Answer

The first assertion is incorrect, since for any pair of random variables X and Y we have $H(X) \geq H(X|Y)$. The second and third proposition are indeed correct (see example 1.12). If X and Y are independent random variables, we always have $H(X) = H(X|Y) = H(X|y)$, hence the fourth and fifth assertions are also true.

Control Question 11

Relate, if possible,

1. $H(X, Y)$;
2. $H(X|Y)$;

with

a. $\leq H(X)$;

b. $\leq H(Y)$;

c. $\leq H(X) + H(Y)$;

d. $= H(Y) + H(X|Y)$;

e. $= H(X) + H(Y|X)$;

f. $= H(Y) + H(Y|X)$;

g. $= \sum_{x,y} p_{X|Y}(x, y) \log p_{X|Y}(x, y)$;

h. $= H(X, Y) - H(X)$.

Answer

For $H(X, Y)$ we have

c. $\leq H(X) + H(Y)$;

d. $= H(Y) + H(X|Y)$;

e. $= H(X) + H(Y|X)$.

For $H(X|Y)$ we have

a. $\leq H(X)$;

c. $\leq H(X) + H(Y)$.

1.1.4 Axiomatic Determination of Entropy

Learning Objectives for Subsection 1.1.4

After studying this section you should understand

- that entropy is characterized by some simple conditions;
- how entropy is derived from these conditions.

We introduce here four simple conditions, which should be satisfied by a measure of uncertainty related to a probability over a finite choice system. If $S = \{e_1, e_2, \ldots, e_n\}$ is a finite choice system and $p_1 = p(e_1), p_2 = p(e_2), \ldots, p_n = p(e_n)$ a probability distribution over it, then the measure of uncertainty of this system is assumed to be a function $H(p_1, p_2, \ldots, p_n)$ of $p_1, p_2, \ldots, p_n$ only. But at this point the form this function should take is undefined. In particular, H does *not* denote the entropy here, but some unknown function. Rather than to define H by some formula, we impose the following conditions on H:

(H1) $H(p_1, p_2, \ldots, p_n) = H(p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(n)})$ for any *permutation* π of $1, 2, \ldots, n$.

(H2) $H(p_1, p_2, \ldots, p_n)$ is a continuous function in all variables.

(H3) The equation

$$H(p_1, p_2, \ldots, p_n) = H(p_1 + p_2, p_3, \ldots, p_n) + (p_1 + p_2) \cdot H(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2})$$

holds for all probability distributions $p_1, p_2, \ldots, p_n$.

(H4) For a uniform probability distribution $p_i = \frac{1}{n}$ for $i = 1, \ldots, n$, $H(p_1, \ldots, p_n) = H(\frac{1}{n}, \ldots, \frac{1}{n})$ as a function of $n \geq 1$ is monotone increasing.

These are reasonable conditions to impose on a measure of uncertainty of a probability distribution. (H1) says that the measure does not depend on the numbering of the possible choices. (H2) requires that small changes in the probabilities should only provoke small changes in the measures of uncertainty. Condition (H3) is more technical, it is a simple form of a chaining formula like theorem 1.4. (H4) expresses the idea that with uniform distribution (equivalent to choice without probabilities), the uncertainty should increase with the number of possibilities.

Entropy as defined in subsection 1.1.2 indeed satisfies these conditions as is stated in proposition 1.1. In this subsection we shall prove that H *must* be the entropy, if (H1) to (H4) are required.

Theorem 1.5 *(H1), (H2), (H3) and (H4) are satisfied if, and only if,*

$$H(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log p_i. \tag{1.11}$$

The logarithm may be taken to any base.

The "if" part of the theorem has already been proved in proposition 1.1. The "only if" part remains to be proved. That is, we assume conditions (H1) to (H4) and derive (1.11). We do this in three steps. In each step, we prove a lemma, each of which is also interesting in itself.

We start by showing that (H3) essentially already contains a general form of chaining. We consider a probability distribution $p_1, p_2, \ldots, p_n$. But instead of selecting one of the possibilities directly according to these probabilities, we use a two-stage choice scheme as represented by a tree in in Figure 1.10. In the first stage one of several arcs is selected with probability $p_1 + \cdots + p_{i_1}, p_{i_1+1} + \cdots + p_{i_2}, \ldots$. In the second stage, depending on the choice in the first stage, a second arc is selected. For example, if in the first stage the left most arc has been selected, then the next selection is according to the probabilities

$$\frac{p_1}{p_1 + \cdots + p_{i_1}}, \frac{p_2}{p_1 + \cdots + p_{i_1}}, \ldots, \frac{p_{i_1}}{p_1 + \cdots + p_{i_1}}.$$

We see, that with this two-stage scheme we finally select one of the n possibilities with the original probabilities $p_1, p_2, \ldots, p_n$.

We have now the following lemma.

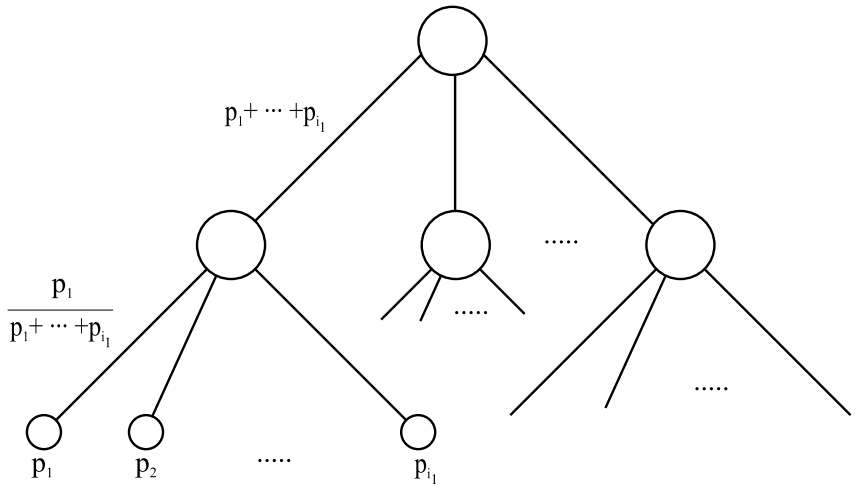


Figure 1.10: A two-stage probabilistic choice system.

Lemma 1.2 *(H1) and (H3) imply that*

$$
\begin{aligned}
&H(p_1, p_2, \ldots, p_{i_1}, p_{i_1+1}, \ldots, p_{i_2}, \ldots, p_{i_{s-1}+1}, \ldots, p_{i_s}, p_{i_s+1}, \ldots, p_n) = \\
&H(p_1 + \cdots + p_{i_1}, p_{i_1+1} + \cdots + p_{i_2}, \ldots, p_{i_{s-1}+1} + \cdots + p_{i_s}, p_{i_s+1} + \cdots + p_n) \\
&\quad + (p_1 + \cdots + p_{i_1}) \cdot H(\frac{p_1}{p_1 + \cdots + p_{i_1}}, \ldots, \frac{p_{i_1}}{p_1 + \cdots + p_{i_1}}) \\
&\quad + (p_{i_1+1} + \ldots + p_{i_2}) \cdot H(\frac{p_{i_1+1}}{p_{i_1+1} + \ldots + p_{i_2}}, \ldots, \frac{p_{i_2}}{p_{i_1+1} + \ldots + p_{i_2}}) \\
&\quad \ldots \\
&\quad + (p_{i_{s-1}+1} + \ldots + p_{i_s}) \cdot H(\frac{p_{i_{s-1}+1}}{p_{i_{s-1}+1} + \ldots + p_{i_s}}, \ldots, \frac{p_{i_s}}{p_{i_{s-1}+1} + \ldots + p_{i_s}}) \\
&\quad + (p_{i_s+1} + \ldots + p_n) \cdot H(\frac{p_{i_s+1}}{p_{i_s+1} + \ldots + p_n}, \ldots, \frac{p_n}{p_{i_s+1} + \ldots + p_n}).
\end{aligned}
$$

PROOF First we prove that

$$
\begin{aligned}
&H(p_1, \ldots, p_i, p_{i+1}, \ldots, p_n) \\
&\quad = \quad H(p_1 + \cdots + p_i, p_{i+1}, \ldots, p_n) \\
&\qquad + (p_1 + \cdots + p_i) \cdot H(\frac{p_1}{p_1 + \cdots + p_i}, \ldots, \frac{p_i}{p_1 + \cdots + p_i}). \qquad (1.12)
\end{aligned}
$$

This is proved by induction over i. It holds for $i = 2$ by (H3). Suppose (1.12) holds

for i. Then, applying the formula for $i = 2$ to (1.12), we obtain

$$\begin{aligned}
&H(p_1, \ldots, p_i, p_{i+1}, \ldots, p_n) \\
&\quad = \quad H(p_1 + \cdots + p_i, p_{i+1}, \ldots, p_n) \\
&\qquad +(p_1 + \cdots + p_i) \cdot H(\frac{p_1}{p_1 + \cdots + p_i}, \ldots, \frac{p_i}{p_1 + \cdots + p_i}) \\
&\quad = \quad \{H((p_1 + \cdots + p_i) + p_{i+1}, p_{i+2}, \ldots, p_n) \\
&\qquad + ((p_1 + \cdots + p_i) + p_{i+1}) \cdot H(\frac{p_1 + \cdots + p_i}{p_1 + \cdots + p_i + p_{i+1}}, \frac{p_{i+1}}{p_1 + \cdots + p_i + p_{i+1}})\} \\
&\qquad +(p_1 + \cdots + p_i) \cdot H(\frac{p_1}{p_1 + \cdots + p_i}, \ldots, \frac{p_i}{p_1 + \cdots + p_i}).
\end{aligned}$$

Since (1.12) holds for i, we conclude that

$$\begin{aligned}
&H(\frac{p_1}{p_1 + \cdots + p_i + p_{i+1}}, \ldots, \frac{p_i}{p_1 + \cdots + p_i + p_{i+1}}, \frac{p_{i+1}}{p_1 + \cdots + p_i + p_{i+1}}) \\
&\quad = \quad H(\frac{p_1 + \cdots + p_i}{p_1 + \cdots + p_i + p_{i+1}}, \frac{p_{i+1}}{p_1 + \cdots + p_i + p_{i+1}}) \\
&\qquad + \frac{p_1 + \cdots + p_i}{p_1 + \cdots + p_i + p_{i+1}} \cdot H(\frac{p_1}{p_1 + \cdots + p_i}, \ldots, \frac{p_i}{p_1 + \cdots + p_i}).
\end{aligned}$$

If we substitute this above, we obtain

$$\begin{aligned}
&H(p_1, \ldots, p_i, p_{i+1}, \ldots, p_n) \\
&\quad = \quad H(p_1 + \cdots + p_i + p_{i+1}, p_{i+2}, \ldots, p_n) \\
&\qquad +(p_1 + \cdots + p_i + p_{i+1}) \cdot H(\frac{p_1}{p_1 + \cdots + p_i + p_{i+1}}, \ldots, \frac{p_{i+1}}{p_1 + \cdots + p_i + p_{i+1}}).
\end{aligned}$$

So (1.12) holds for every $i = 2, \ldots, n$.

Now, by (H1) we then also have

$$\begin{aligned}
&H(p_1, \ldots, p_{i-1}, p_i, p_{i+1}, \ldots, p_{j-1}, p_j, p_{j+1}, \ldots, p_n) \\
&\quad = \quad H(p_1, \ldots, p_{i-1}, p_i + \cdots + p_j, p_{j+1}, \ldots, p_n) \\
&\qquad +(p_i + \cdots + p_j) \cdot H(\frac{p_i}{p_i + \cdots + p_j}, \ldots, \frac{p_j}{p_i + \cdots + p_j}),
\end{aligned}$$

and this for $1 \leq i < j \leq n$. If we apply this successively to

$$H(p_1, p_2, \ldots, p_{i_1}, p_{i_1+1}, \ldots, p_{i_2}, \ldots, p_{i_{s-1}+1}, \ldots, p_{i_s}, p_{i_s+1}, \ldots, p_n),$$

then Lemma follows. ■

In the next step of our overall proof, we consider the case of uniform probability distributions, or the case of choice without probabilities. That is, we put $p_i = 1/n$ for $i = 1, \ldots, n$. We define

$$h(n) = H(\frac{1}{n}, \ldots, \frac{1}{n}).$$

Then we prove the next lemma:

Lemma 1.3 *(H1), (H3) and (H4) imply*

$$h(n) = c \cdot \log n$$

for some constant $c > 0$ and all integers n.

PROOF Let $n = m \cdot l$ for some integers n, m, l. By lemma 8 we have

$$\begin{aligned} h(m \cdot l) &= H(\frac{1}{m \cdot l}, \ldots, \frac{1}{m \cdot l}) \\ &= H(\frac{1}{l}, \ldots, \frac{1}{l}) + l \cdot \frac{1}{l} \cdot H(\frac{1/ml}{1/l}, \ldots, \frac{1/ml}{1/l}) \\ &= h(l) + h(m). \end{aligned} \tag{1.13}$$

This fundamental equation has the solution $h(n) = c \cdot \log n$. We show that this is the only solution. If m and l are integers, select an integer N and determine n such that

$$l^n \leq m^N < l^{n+1}.$$

Then, from (H4) we conclude that

$$h(l^n) \leq h(m^N) < h(l^{n+1}).$$

But by the fundamental equation (1.13) $h(l^n) = n \cdot h(l)$ and $h(m^N) = N \cdot h(m)$, hence

$$n \cdot h(l) \leq N \cdot h(m) < (n+1) \cdot h(l).$$

We note that for $l = 1$ we have $h(l^n) = h(l) = n \cdot h(l)$, hence $h(l) = 0$. For $l > 1$, (H4) implies that $h(l) > 0$. Thus suppose $l > 0$. Then

$$\frac{n}{N} \leq \frac{h(m)}{h(l)} < \frac{n+1}{N}.$$

But, we also have $\log l^n \leq \log m^N < \log l^{n+1}$ by the monotonicity of the logarithm. Hence

$$n \cdot \log l \leq N \cdot \log m < (n+1) \cdot \log l,$$

or

$$\frac{n}{N} \leq \frac{\log m}{\log l} < \frac{n+1}{N}.$$

These inequalities imply

$$\mid \frac{h(m)}{h(l)} - \frac{\log m}{\log l} \mid < \frac{1}{N}.$$

Since this holds for all integers N, we conclude that

$$\frac{h(m)}{h(l)} = \frac{\log m}{\log l}.$$

This in turn is valid for all integers m, l. Therefore,

$$\frac{h(m)}{\log m} = \frac{h(l)}{\log l} = c,$$

for a certain constant c. Thus we have $h(m) = c \cdot \log m$ for $m > 1$. But for $m = 1$ we have both $h(m) = 0$ and $c \log 1 = 0$. Thus $h(m) = c \cdot \log m$ for all $m \geq 1$. Since by (H4) $h(n)$ is monotone increasing, we must have $c > 0$. ■

In the third step we use the results obtained so far to prove (1.11) for *rational* probabilities. We formulate the corresponding lemma:

Lemma 1.4 *(H1), (H3) and (H4) imply for rational probabilities $p_1, \ldots, p_n$ that*

$$H(p_1, \ldots, p_n) = -c \cdot \sum_{i=1}^{n} p_i \log p_i. \tag{1.14}$$

PROOF Assume that

$$p_1 = \frac{q_1}{p}, \ldots, p_n = \frac{q_n}{p}$$

for some integers $q_1, \ldots, q_n$ and p such that

$$q_1 + \cdots + q_n = p.$$

We have by definition

$$\begin{aligned} h(p) &= H(\frac{1}{p}, \ldots, \frac{1}{p}) \\ &= H(\underbrace{\frac{1}{p}, \ldots, \frac{1}{p}}_{q_1}, \underbrace{\frac{1}{p}, \ldots, \frac{1}{p}}_{q_2}, \ldots, \underbrace{\frac{1}{p}, \ldots, \frac{1}{p}}_{q_n}), \end{aligned}$$

where the first group contains q_1, the second q_2, and the last q_n arguments. From lemmas 8 and 1.3 we then obtain, using this grouping of variables, that

$$\begin{aligned} h(p) &= H(\frac{q_1}{p}, \ldots, \frac{q_n}{p}) \\ &\quad + \frac{q_1}{p} \cdot H(\frac{1}{q_1}, \ldots, \frac{1}{q_1}) + \cdots + \frac{q_n}{p} \cdot H(\frac{1}{q_n}, \ldots, \frac{1}{q_n}) \\ &= H(p_1, \ldots, p_n) + c \cdot p_1 \log q_1 + \cdots + c \cdot p_n \log q_n. \end{aligned}$$

This implies that

$$\begin{aligned} & H(p_1, \ldots, p_n) \\ &= c \cdot \log p - c \cdot p_1 \log q_1 - \cdots - c \cdot p_n \log q_n \\ &= c \cdot (p_1(\log p - \log q_1) + \cdots + p_n(\log p - \log q_n)) \\ &= -c \cdot (p_1 \log p_1 + \cdots + p_n \log p_n). \end{aligned}$$

This proves (1.14). ■

Now, we are nearly at the end of the overall proof of theorem 1.5. If p_i are arbitrary probabilities, not necessarily rational ones, then they can be approximated by a sequence of rational ones, converging to p_i. Since (1.11) holds for all rational probability distributions, the required continuity (H2) of $H(p_1, \ldots, p_n)$ and the continuity of the right hand side of (1.14) then implies that (1.11) holds for any probability distribution. This concludes the overall proof of theorem 1.5.

This theorem tells us, that we may select the base of the logarithm, as well as the constant $c > 0$ arbitrarily. These choices only determine the measurement unit for the measure of uncertainty.

Summary for Section 1.1

- In this subsection we proved that the elementary requirements, that the function

  $$H(p_1, \ldots, p_n)$$

  be continuous, does not depend on the ordering of the probabilities and satisfies a simple decomposition property, together with the requirement that $h(n) = H(\frac{1}{n}, \ldots, \frac{1}{n})$ be monotone increasing with n imply that H must be the entropy;
- The proof was done in three steps: In the first one a more general decomposition property was derived. In the second step, it was proved that $h(n)$ is essentially a logarithm. In the third step it was shown that H is the entropy, if the probabilities are *rational* numbers. The theorem follows then from the requirement of continuity.

Control Question 12

Which conditions do not characterize the entropy H?

1. $H(p_1, p_2, \ldots, p_n) = H(p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(n)})$ for exactly one *permutation* π of $1, 2, \ldots, n$;
2. $H(p_1, p_2, \ldots, p_n)$ is a continuous function in all variables;
3. $H(p_1, \ldots, p_n) = H(\frac{1}{n}, \ldots, \frac{1}{n})$ as a function of $n \geq 1$ is monotone decreasing.

Answer

1. Since $H(p_1, p_2, \ldots, p_n) = H(p_{\pi(1)}, p_{\pi(2)}, \ldots, p_{\pi(n)})$ must hold true for **all** *permutations* π of $1, 2, \ldots, n$, this condition does not characterize the entropy H, thus the answer is correct.
2. That is indeed a characterization of the entropy.
3. The uncertainty should increase with the number of possibilities, hence this is not a characterization of H.

1.2 Information And Its Measure

Learning Objectives for Section 1.2

After studying this section you should understand

- how information is measured;
- that the measure of information is always relative to a precise question and also relative to previous information;
- that information and questions have a natural algebraic structure;
- other quantities, related to the measure of information, like the mutual information, the divergence and the degree of surprise, together with their properties and the relations among them.

1.2.1 Observations And Events

Learning Objectives for Subsection 1.2.1

After studying this subsection you should understand

- that an observation of a random variable or an event related to a random variable is information;
- that the amount of information gained by observing the value of a variable or an event is measured by the resulting change of uncertainty;
- that therefore entropy and measure of information are intimately related.

What is information and how is it measured? We start studying this question in this subsection. The basic idea is that information is something that changes uncertainty, preferably decreasing it. Accordingly, we propose to measure the amount of information by the amount of change of uncertainty. This idea will be developed in this section step by step, progressing from very simple to more involved situations. The emphasis in this section will be on the measurement of information content, and less on the representation of information and other properties it may have besides quantity.

To start let's consider a probabilistic choice system (S, P) represented by a random variable X taking values $x \in S$ with probabilities $p_X(x)$ (see subsection 1.1.2). This random variable describes a certain experiment where the outcome is uncertain. The uncertainty of this situation is measured by the entropy

$$H(X) = -\sum_{x \in S} p_X(x) \log p_X(x). \tag{1.15}$$

When the experiment is carried out, a certain value $x \in S$ of the random variable is observed. There is no uncertainty left. So the previous uncertainty $H(X)$ is reduced to the posterior uncertainty 0. The difference $H(X) - 0 = H(X)$ is the amount of information gained by performing the experiment. So the entropy of a random variable measures the amount of information gained by observing the actual value of the variable.

This idea calls for two important remarks:

- Since information is a change of entropy it is measured in the same unit as entropy, i.e. bits, if base 2 is selected for the logarithm.
- The amount of information gained by an observation is the same for all possible observations. In particular, it is the same whether the probability of the actual observation is small or large. We return to this point in subsection 1.2.4.

Example 1.15 (Binary Random Variable) If, for a binary variable X the outcome $'0'$ arises with probability p and $'1'$ with probability $q = 1 - p$, then observing the outcome of this binary experiment results in a gain of information $H(X) = -p \log p - q \log q$. In particular, in the case of a fair coin, observing the outcome of a throw gives 1 bit of information.

Let's now slightly generalize the situation. We still consider a random variable X related to a probabilistic choice situation (S, P). The associated uncertainty is still $H(X)$. But this time, we carry out the experiment only partially. We do not observe the exact value of X, but only some event $E \subseteq S$. Obviously this is also information. But what is its amount? The observation of the event E changes the random variable X to the conditional variable $X|E$ related to the new probabilistic choice situation (E, P_E). Here P_E denotes the conditional probabilities

$$p_{X|E}(x) = \frac{p_X(x)}{p_X(E)}, \text{ for all } x \in E.$$

This new situation, created by the observation of the event E, has the uncertainty which corresponds to the conditional entropy $H(X|E)$ (see subsection 1.1.2),

$$H(X|E) = -\sum_{x \in E} p_{X|E}(x) \log p_{X|E}(x).$$

So, the observation of E changes the uncertainty from $H(X)$ to $H(X|E)$. The amount of information gained is thus $H(X) - H(X|E)$. We shall see in a moment, that this is not always really a *gain* of information, since $H(X|E)$ may be *greater* than H(X), such that the observation of event E *increases* uncertainty, which corresponds according to our definition of *negative* information.

Example 1.16 (Murderer) Assume that we have n suspected murderers, but one of them (say number 1) is a lot more suspect than the other $n - 1$ ones. We may represent the probability that suspect 1 is the murderer by $p_X(1) = 1 - \epsilon$, which is nearly one. The probabilities that one of the other suspects could be the murderer is only $p_X(i) = \epsilon/(n-1)$. The entropy is then

$$H(X) = -(1 - \epsilon)(\log 1 - \epsilon) - \epsilon \log \frac{\epsilon}{n-1}. \tag{1.16}$$

If ϵ is small, then this entropy will be very small. This reflects the fact, that we are pretty sure that no. 1 is the murderer. But suppose now that all of a sudden no. 1 produces an alibi. We are then forced to exclude no. 1 from the list of suspects. This corresponds to the event E that $X \in \{2, \ldots, n\}$. The conditional distribution of X given E is then $p_{X|E}(i) = 1/(n-1)$ for $i = 2, \ldots, n$. The corresponding new

uncertainty is $H(X|E) = \log(n-1)$. This can be much larger than $H(X)$. So, the new information, i.e. the alibi of no. 1, changes (unexpectedly) a clear and neat situation into a very uncertain, messy situation. The information is therefore *negative*. This example should convince that negative information is a reality.

We introduce now some new notation. First we denote the conditional random variable X given an event $E \subseteq S$ by X_E. It corresponds, as noted above, to the probabilistic choice situation (E, P_E), where P_E is the set of conditional probabilities $p_{X|E}(x)$ for $x \in E$. Then, we denote the amount of information of the event E with respect to the random variable by $i(E/X)$. So we have

$$i(E/X) = H(X) - H(X_E). \tag{1.17}$$

If, in particular, the event E corresponds to the observation of a precise value x of the random variable, $E = \{x\}$, then we write the corresponding amount of information $i(x/X)$. And we have $H(X|x) = 0$, hence, as already noted above

$$i(x/X) = H(X). \tag{1.18}$$

In this sense, and only in this sense, entropy is a measure of information.

If we are interested whether a particular event E takes place or not, we are confronted with a new choice situation $(\{E, E^c\}, P)$. Associated with it is a new random variable Y with the following probability distribution

$$p_Y(E) = p_X(E) = \sum_{x \in E} p_X(x), \quad p_Y(E^c) = p_X(E^c) = \sum_{x \in E^c} p_X(x).$$

What will be the expected information, when we learn whether E takes place or not? It is

$$\begin{aligned} & I(X|Y) \\ &= p_Y(E) i(E/X) + p_Y(E^c) i(E^c/X) \\ &= H(X) - (p_Y(E) H(X|E) + p_Y(E^c) H(X|E^c)) \\ &= H(X) - H(X|Y). \end{aligned} \tag{1.19}$$

But we know (see corollary 1.3) that $H(X|Y) \leq H(X)$. So, the expected measure of information gained by observing whether some event takes place or not, is *never negative*, i.e. $I(X|Y) \geq 0$. We shall get back to the important notion of expected information in subsection 1.2.3.

Example 1.17 (Murderer - Continuation) Let's return to the murder example 1.16 above. Suppose somebody announces that he will produce proof of the guilt or innocence of no. 1 (by examining DNA for example). We expect with probability $1 - \epsilon$ that we will prove the guilt of no. 1. This represents event E^c in the notation of example 1.16. The resulting uncertainty in this case will be 0 and the information obtained $H(X)$ (see (1.16)). With probability ϵ we expect that the innocence of no. 1 will be proved (event E). The remaining uncertainty is then, as seen in example 1.16 $\log(n-1)$ and the information obtained $H(X) - \log(n-1)$. So, in this particular case, the expected information to be gained by this proof is equal to

$$\begin{aligned} & (1-\epsilon) H(X) + \epsilon (H(X) - \log(n-1)) \\ &= H(X) - \epsilon \log(n-1) \\ &= -(1-\epsilon)\log(1-\epsilon) - \epsilon \log \epsilon \geq 0. \end{aligned}$$

The last equation is obtained using (1.16). Note that this is exactly the amount of information when we learn whether suspect no. 1 is guilty or not.

Suppose now that information, in the form of events observed, comes in successive steps. First we observe an event $E_1 \subseteq S$, and then we get more precise information from an event $E_2 \subseteq E_1$. Since event E_1 changes the random variable X to the conditional random variable X_{E_1}, the information gained by E_2 with respect to the former information E_1 is $i(E_2/X_{E_1}) = H(X_{E_1}) - H(X_{E_2})$. The next theorem shows then that we can add the information gained in each step to get the full information.

Theorem 1.6 *Let X be a random variable associated with a probabilistic choice situation (S, P) and E_1, E_2 two events, $E_2 \subseteq E_1 \subseteq S$. Then*

$$i(E_2/X) = i(E_1/X) + i(E_2/X_{E_1}). \tag{1.20}$$

PROOF The proof is straightforward, using the definition of information

$$\begin{aligned} & i(E_2/X) \\ &= H(X) - H(X_{E_2}) \\ &= H(X) - H(X_{E_1}) + H(X_{E_1}) - H(X_{E_2}) \\ &= i(E_1/X) + i(E_2/X_{E_1}). \end{aligned}$$

We want at this step to stress the following important aspect of information:

- An amount of information is always *relative to prior information.* So, the amount of information of the event E_2 relative to the original variable X is generally not the same as its amount relative to the information given by the event E_1. That is, in general $i(E_2/X) \neq i(E_2/X_{E_1})$. The notation we use underlines this: $i(E_2/X)$ is the amount of information contained in the event E_2 relative to the prior information or prior probability distribution of X, whereas $i(E_2/X_{E_1})$ is the amount of information of the *same* event E_2 relative to the prior information or probability distribution of X_{E_1}.

Example 1.18 (Relativity to prior Information) This remark can be illustrated by the special case of choice without probabilities. Thus, assume S to be a deterministic choice system. If $E \subseteq S$ is an observed event, then we may denote its amount of information relative to the prior information S by $i(E/S)$. Then we have

$$i(E/S) = \log |S| - \log |E| = \log \frac{|S|}{|E|}.$$

If, as in the theorem 1.6, we have $E_2 \subseteq E_1 \subseteq S$, then, once E_1 is observed, we have a new choice system E_1. If we next observe E_2, we gain information $i(E_2/X_{E_1})$ with

respect to the former information E_1. Thus,

$$\begin{aligned} i(E_1/S) &= \log|S| - \log|E_1|, \\ i(E_2/S) &= \log|S| - \log|E_2|, \\ i(E_2/X_{E_1}) &= \log|E_1| - \log|E_2|. \end{aligned}$$

Of course, in this case we also have $i(E_2/S) = i(E_1/S) + i(E_2/X_{E_1})$.

We note that we obtain exactly the same results, if we do not assume a choice system S without probabilities, but a probabilistic choice system (S, P), where P is the *uniform* probability distribution over S.

This discussion seems to indicate that not only *events* represent information, but also random variables or rather that their associated *probability distributions*, are information. This is indeed so. Subsection 1.2.5 will discuss probabilities as information.

Of course theorem 1.6 generalizes to more than two events.

Corollary 1.5 *If $E_m \subseteq E_{m-1} \subseteq \ldots \subseteq E_1 \subseteq S$, then*

$$i(E_m/X) = i(E_1/X) + i(E_2/X_{E_1}) + \cdots + i(E_m/X_{E_{m-1}}). \tag{1.21}$$

Example 1.19 (Fair Die) Let X be the random variable associated with the toss of a fair die. Then we have $H(X) = -\log\frac{1}{6} = \log 6$ bit. Someone is telling us, that $X \neq 1$. So let E_1 be the event $X \neq 1$. Thus

$$i(E_1/X) = H(X) - H(X_{E_1}) = \log 6 - \log 5 = \log\frac{6}{5} \text{ bit},$$

since $H(X_{E_1}) = \log 5$ bit. A litte while later, we receive the information, that $X \neq 1$ and $X \neq 2$ and we associate the event E_2 with it. So

$$i(E_2/X) = H(X) - H(X_{E_2}) = \log 6 - \log 4 = \log\frac{3}{2} \text{ bit},$$

since $H(X_{E_2}) = \log 4$ bit. Finally we compute

$$i(E_2/X_{E_1}) = H(X_{E_1}) - H(X_{E_2}) = \log 5 - \log 4 = \log\frac{5}{4} \text{ bit}.$$

We verify that indeed

$$i(E_2/X) = \log\frac{3}{2} = \log\frac{6}{5} + \log\frac{5}{4} = i(E_1/X) + i(E_2/X_{E_1}).$$

We may also have the situation where two different sources of information report two events $E_1, E_2 \subseteq S$ relative to a probabilistic choice situation (S, P) and an associated random variable X. These two pieces of information can be combined into the event $E_1 \cap E_2$. We assume that $E_1 \cap E_2$ is not empty, since this would represent *contradictory* or *incompatible* information. The amount of the combined information is then $i(E_1 \cap E_2/X)$. By theorem 1.6 we see that

$$i(E_1 \cap E_2/X) = i(E_1/X) + i(E_1 \cap E_2/X_{E_1}) = i(E_2/X) + i(E_1 \cap E_2/X_{E_2}).$$

It does not matter in which sequence the two pieces of information are combined. In both cases we get the same result. Here we observe that information may come in pieces and can then be combined. This points to a certain algebraic structure of information, besides its quantitative aspect.

Example 1.20 (Fair Die - Continuation) Once again we are tossing a fair die (random variable X). As before $H(X) = \log 6$ bit. We observe, that the result is an even number (event E_1). Since $H(X_{E_1}) = \log 3$ bit, we get that

$$i(E_1/X) = H(X) - H(X_{E_1}) = \log 6 - \log 3 = \log \frac{6}{3} = \log 2 = 1 \text{ bit.}$$

We observe next, that the result is smaller than 4 (event E_2). So, with $H(X_{E_2}) = \log 3$ bit,

$$i(E_2/X) = H(X) - H(X_{E_2}) = \log 6 - \log 3 = 1 \text{ bit.}$$

Note that $E_1 \cap E_2 = \{2\}$. Since $H(X_{E_1 \cap E_2}) = 0$ bit, we finally obtain

$$\begin{aligned} i(E_1 \cap E_2/X) &= \log 6 \text{ bit,} \\ i(E_1 \cap E_2/X_{E_1}) &= \log 3 \text{ bit,} \\ i(E_1 \cap E_2/X_{E_2}) &= \log 3 \text{ bit.} \end{aligned}$$

And we see that

$$i(E_1 \cap E_2/X) = \log 6 = i(E_1/X) + i(E_1 \cap E_2/X_{E_1}) = 1 + \log 3.$$

Summary for Section 1.2

- In this subsection we have seen that events or, more particularly, observations of values of random variables are information.
- The amount of information gained by an event is measured by the change of uncertainty, that is, the entropy. So information is measured in bits, like the entropy.
- The amount of information gained by observing an event may be negative, that it, uncertainty may be increased.
- In case the exact value of a random variable is observed, the amount of information gained equals the entropy of the variable, which is always non-negative.
- The amount of information gained is *relative* to the prior information. That is, an event has not an *absolute* amount of information, but the amount depends on what was known before - the prior probability distribution.
- If information, represented by events, comes in successive steps, then the total information gained is the sum of the information gained in each step relative to

the previous step.

Control Question 13

What is the relation between entropy and the measure of information?

1. There is no relation.
2. Since a probability distribution P represents information, then, in this sense, and only in this sense, $H(P)$ measures at the same time entropy and information.
3. Information is defined by the change of entropy.
4. Entropy and Information are both measured in bits.

Answer

1. That is wrong (see below).
2. It is correct that a probability distribution represents information. But this is not a relation between entropy and information and, additionally, the assertion is nonsense.
3. Yes, that is the main idea of an information measure.
4. That is indeed correct, but this is due to the relation between the entropy and information. Hence, the answer is wrong.

Control Question 14

$i(E/X)$ is always positive, since

1. $H(X|Y) \leq H(X)$;
2. $H(X|E) \leq H(X)$;
3. $H(X) \geq 0$;
4. information is always relative to a prior information;
5. the assertion is wrong; information can be negative.

Answer

1. It is correct, that $H(X|Y) \leq H(X)$, but this has nothing to do with the assertion.
2. We can't expect that $H(X|E) \leq H(X)$, hence this is wrong.
3. It is correct, that $H(X) \geq 0$, but this has nothing to do with the assertion.

4. It is correct, that information is always relative to a prior information, but this has nothing to do with the assertion.

5. Yes, information can also be negative, see example 1.16.

Control Question 15

Let X be a random variable related to the probabilistic choice situation (S, P) and $E_2 \subseteq E_1 \subseteq S$. Then $i(E_2/X)$

1. $= 0$, if E_2 corresponds to the observation of a precise value x of X;
2. $= H(X) - H(X|E_2)$;
3. $= i(E_1/X) + i(E_2/X_{E_1})$;
4. $= -\log \frac{|E_2|}{|S|}$ if X is uniform distributed.

Answer

1. That is wrong. If E_2 corresponds to the observation of a precise value x of X, we get $i(E_2/X) = H(X)$.
2. This is the definition of $i(E_2/X)$, hence it is correct.
3. Since $i(E_2/X) = H(X) - H(X_{E_1}) + H(X_{E_1}) - H(X_{E_2})$ it is correct.
4. Correct. Follows directly from $-\log \frac{|E_2|}{|S|} = \log \frac{|S|}{|E_2|} = \log |S| - \log |E_2|$.

1.2.2 Information and Questions

Learning Objectives for Subsection 1.2.2

After studying this subsection you should understand

- that information may relate to different questions;
- that the amount of an information can only be measured relative to a precisely specified question, as well as relative to prior information;
- that information and questions exhibit an algebraic structure;
- what independent information means and that the total amount of independent information is the sum of the amounts of individual information.

We start by considering the simple case of a compound probabilistic choice system $(S_1 \times S_2, P)$ and the corresponding pair of random variables X and Y (we refer to subsection 1.1.3 for these notions). This simple situation will serve as a model to

study how information pertains to different questions, and why therefore, the content of information can only be measured relative to a specified question. Also this simple case serves to exhibit the important fact that information and questions possess an inherent algebraic structure.

Assume we observe the value of one of the two variables, say $Y = y \in S_2$. As we have seen in the previous section this reduces the uncertainty regarding Y to zero, and the observation contains $i(y/Y) = H(Y)$ bits of information relative to the prior information Y. But this observation also changes the prior probability distribution of X, which becomes a conditional random variable $X|y$ with probability distribution

$$p_{X|y}(x, y) = \frac{p_{X,Y}(x, y)}{p_Y(y)}, \text{ for all } x \in S_1.$$

This means that the uncertainty regarding the variable X changes too. So the observation $Y = y$ also contains information relative to X. But its amount is not the same, as that relative to Y. In fact, the change in entropy of X, which measures the amount of information of y relative to X, is as follows:

$$i(y/X) = H(X) - H(X|y).$$

We saw in subsection 1.1.3, example 1.12, that the entropy $H(X|y)$ may be smaller or greater than $H(X)$. In the latter case, we get a negative amount $i(y/X)$ of information. So, this is another case where negative information may arise.

The observation y also changes the common entropy of the two variables also. That is, we have

$$i(y/X, Y) = H(X, Y) - H(X, Y|y).$$

We note that

$$p_{X,Y|y}(x, y') = \frac{p_{X,Y}(x, y')}{p_Y(y)}$$

for all $x \in S$, if $y' = y$. And we have $p_{X,Y|y}(x, y') = 0$, if $y' \neq y$. But this shows that $p_{X,Y|y}(x, y) = p_{X|y}(x)$. So, we conclude that $H(X, Y|y) = H(X|y)$ and hence,

$$i(y/X, Y) = H(X, Y) - H(X|y). \tag{1.22}$$

Using theorem 1.4, we also find

$$i(y/X, Y) = H(X) - H(X|y) + H(Y|X) = i(y/X) + H(Y|X). \tag{1.23}$$

The information gained by y with respect to both variables X and Y equals the information gained by y with respect to variable X plus the expected uncertainty remaining in Y, when X is observed.

So, the same simple observation $Y = y$ is information, which has a different measure relative to different references or questions. $i(y/Y)$ is the amount of information with respect to the question of the value of Y, $i(y/X)$ the amount of information regarding the unknown value of X and finally $i(y/X, Y)$ the amount of information with respect to the unknown common value of the two variables X and Y together. So, we emphasize, the amount of an information is to be measured relative to the question to be considered. This is a second *principle of relativity*. The first one was introduced in the previous subsection, and says that the amount of information is always measured relative to the prior information. Let's summarize the two basic principles of relativity of information:

- *Relativity regarding the question*: The amount of information is to be measured relative to a specified question. Questions are so far represented by random variables or choice situations. The question is, what is the value of the random variable, or the element selected in a choice situation?

- *Relativity regarding the prior information*: The amount of information is to be measured relative to the prior information. Prior information is represented by the prior probability distribution of the variable or the probabilistic choice situation.

If X and Y are independent random variables, then $H(X|y) = H(X)$ and thus $i(y/X) = 0$. The information $Y = y$ has no content relative to X, it does not bear on X. And $i(y/X,Y) = i(y/Y)$ from (1.23), since $H(Y|X) = H(Y)$ in this case.

Example 1.21 Assume $p_{X,Y}(0,1) = p_{X,Y}(1,0) = p_{X,Y}(0,0) = \frac{1}{3}$, $p_{X,Y}(1,1) = 0$, hence $p_X(0) = p_Y(0) = \frac{2}{3}$ and $p_X(1) = p_Y(1) = \frac{1}{3}$ like in examples 1.12 and 1.13. Since

$$p_{X,Y|Y=0}(0,0) = \frac{p_{X,Y}(0,0)}{p_Y(0)} = p_{X,Y|Y=0}(1,0) = \frac{p_{X,Y}(1,0)}{p_Y(0)} = \frac{1}{2},$$
$$p_{X,Y|Y=0}(0,1) = p_{X,Y|Y=0}(1,1) = 0,$$

it follows, that $H(X,Y|Y=0) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = -\log\frac{1}{2} = 1$ bit. With $H(X,Y) = -\frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} = -\log\frac{1}{3}$ bit, we obtain

$$i(Y=0/X,Y) = H(X,Y) - H(X,Y|Y=0) = -\log\frac{1}{3} + \log\frac{1}{2} = \log\frac{3}{2} \text{ bit.}$$

Example 1.22 (Compound Choice Situation without Probabilities) Consider a compound choice situation $S_1 \times S_2$ without probabilities. Then any observation of $y \in S_2$ yields the amount of information

$$i(y/S_1 \times S_2) = \log|S_1 \times S_2| - \log|S_1| = \log|S_1| + \log|S_2| - \log|S_1| = \log|S_2| \text{ bit.}$$

This is not a surprise, since there is no uncertainty left in S_2, and, as we have seen, the information gained is then the uncertainty in S_2.

Example 1.23 (Symmetric Binary Channel - Continuation) The communication channel is an important example (see example 1.14 in subsection 1.1.3). X refers to an uncertain input signal and Y to the output signal. In practice one observes the output signal $Y = y$ and would like to infer about the unknown input signal, which was transformed into y. Then $i(y/X) = H(X) - H(X|y)$ is the information content of the information y relative to this question. If there is no distortion of the input signal during the transmission, then $X = Y$ and $H(X|y) = 0$ bit. In this case $i(y/X) = H(X)$, that is, y contains all the information about X. But in general, we have $H(X|y) > 0$ and hence $i(y/X) < H(X)$. There is a loss of information associated with the transmission. It is also of interest to look at the expected

information at the output regarding the input,

$$I(X|Y) = \sum_{y \in S_2} p_Y(y) i(y/X).$$

This is a very important quantity in communication theory. We shall come back to this in subsection 1.2.3.

We now generalize the discussion above by considering events related to the two variables X and Y. We notice that there may be events E related to the variable X, that is $E \subseteq S_1$, events related to Y, i.e. $E \subseteq S_2$ and finally events related to both variables, $E \subseteq S_1 \times S_2$. We *label* an event E by $d(E)$ to indicate to what domain it belongs. So $d(E) = \{x\}$ means $E \subseteq S_1$, and $d(E) = \{x, y\}$ means $E \subseteq S_1 \times S_2$. If $d(E) = \{x, y\}$, then we define the *projection* of E to domain x by

$$E^{\downarrow x} = \{x \in S_1 : \text{there is a } y \in S_2 \text{ such that } (x, y) \in E\}.$$

The projection $E^{\downarrow y}$ is defined similarly. We refer to figure 1.11 for a geometric picture of projections.

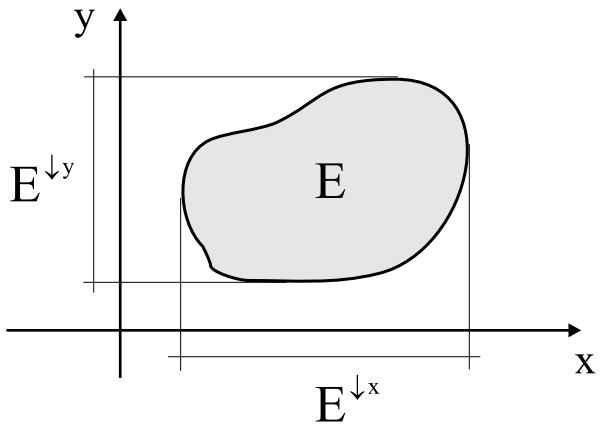


Figure 1.11: The projection of an event to a smaller domain, illustrated in the case of two dimensions.

We start by considering an event E with $d(E) = \{x, y\}$ (see figure 1.11). This is clearly information relative to X, to Y and to (X, Y). The corresponding measures are

$$\begin{aligned} i(E/X) &= H(X) - H(X|E), \\ i(E/Y) &= H(Y) - H(Y|E), \\ i(E/X,Y) &= H(X,Y) - H(X,Y|E). \end{aligned}$$

At this point we need to clarify conditional random variables such as $X|E$, if E is an event relative to $S_1 \times S_2$, that is, $E \subseteq S_1 \times S_2$. Clearly, the conditional random variables $X, Y|E$ has the probability distribution

$$p_{X,Y|E}(x, y) = \frac{p_{X,Y}(x, y)}{p_{X,Y}(E)}, \text{ for } (x, y) \in E.$$

Otherwise, that is, if $(x, y) \notin E$, we have $p_{X,Y|E}(x, y) = 0$. The conditional variable $X|E$ has now the *marginal* probability distribution of $p_{X,Y|E}(x, y)$, that is

$$p_{X|E}(x) = \sum_{y \in S_2} p_{X,Y|E}(x, y) \text{ for all } x \in E^{\downarrow x}.$$

For $x \notin E^{\downarrow x}$, we have $p_{X|E}(x) = 0$. Similarly, we have

$$p_{Y|E}(y) = \sum_{x \in S_1} p_{X,Y|E}(x,y) \text{ for all } y \in E^{\downarrow y},$$

and $p_{Y|E}(y) = 0$ for $y \notin E^{\downarrow y}$. This clarifies how $H(X|E)$ and $H(Y|E)$ have to be computed,

$$\begin{aligned} H(X|E) &= -\sum_x p_{X|E}(x) \log p_{X|E}(x), \\ H(Y|E) &= -\sum_y p_{Y|E}(y) \log p_{Y|E}(y). \end{aligned}$$

Again, one or several of these information measures may be negative in the general case.

Example 1.24 (Fair Coin) Consider a fair coin which is thrown twice. X is the result of the first throw, Y the result of the second one. Since the coin is fair, we have $p_{X,Y}(x,y) = 1/4$ for all four possible results. Therefore $H(X,Y) = \log 4 = 2$ bit. If we learn that the coin did not show both times heads, then we know that the event $E = \{(0,0),(0,1)(1,0)\}$ took place assuming that 0 indicates "tails" and 1 means "heads". So we obtain that $p_{X,Y}(x,y) = 1/3$ for the three remaining possible results in E. Thus, $H(X,Y|E) = \log 3$ and we gained the amount of information $i(E/X,Y) = 2 - \log 3 = \log 4/3$ bit. Regarding the first throw, we obtain

$$\begin{aligned} p_{X|E}(0) &= p_{X,Y}(0,0) + p_{X,Y}(0,1) = \frac{2}{3}, \\ p_{X|E}(1) &= p_{X,Y}(1,0) = \frac{1}{3}. \end{aligned}$$

So the remaining uncertainty is

$$H(X|E) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} = (\log 3) - \frac{2}{3} \text{ bit.}$$

The information obtained relative to the first throw is therefore

$$i(E/X) = H(X) - H(X|E) = 1 - (\log 3 - \frac{2}{3}) = \log\frac{2}{3} + \frac{2}{3} \approx 0.0817 \text{ bit}$$

Due to the symmetry of the situation, the same amount of information is also gained relative to the second throw.

Next, we look at the case when an event E relative to the second variable Y is observed, i.e. $d(E) = \{y\}$. As usual, we have in this case $i(E/Y) = H(Y) - H(Y|E)$. But what can be said about the information relative to X,Y or to X alone. To answer these questions, we need to extend the event relative to E to an event relative to X,Y, but without adding information. We define

$$E^{\uparrow\{x,y\}} = E \times \mathcal{V}_X.$$

This is the so-called *cylindric extension* of E to the domain $\{x,y\}$ (see figure 1.2.2). A look at fig. 1.12 shows that no information relative to the variable X has been added, which is not already contained in E.

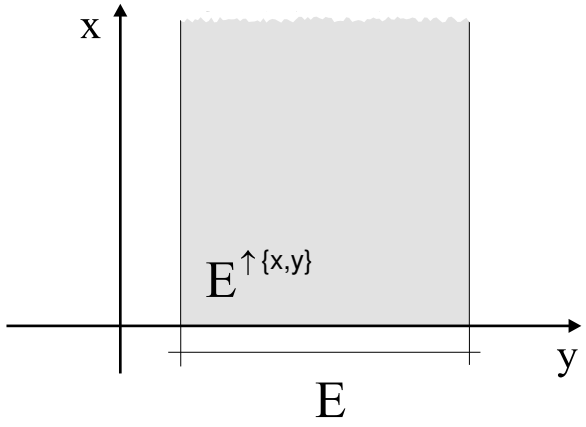


Figure 1.12: The cylindric extension of an event to a larger domain, illustrated in the case of two dimensions

First, we define

$$i(E/X,Y) = i(E^{\uparrow\{x,y\}}/X,Y) = H(X,Y) - H(X,Y|E^{\uparrow\{x,y\}}).$$

That is, we consider that E and $E^{\uparrow\{x,y\}}$ represent the same information relative to X, Y. Note that

$$\begin{aligned} & p_{X,Y|E^{\uparrow\{x,y\}}}(x,y) \\ = \; & \frac{p_{X,Y}(x,y)}{p_{X,Y}(E^{\uparrow\{x,y\}})} = \frac{p_{X,Y}(x,y)}{p_Y(E)}, \text{ for all } (x,y) \in E^{\uparrow\{x,y\}}, \end{aligned}$$

because

$$p_{X,Y}(E^{\uparrow\{x,y\}}) = \sum_{(x,y)\in E^{\uparrow\{x,y\}}} p_{X,Y}(x,y) = \sum_{x\in S_1, y\in E} p_{X,Y}(x,y) = p_Y(E).$$

In the same way, we define

$$i(E/X) = i(E^{\uparrow\{x,y\}}/X) = H(X) - H(X|E^{\uparrow\{x,y\}}).$$

Remembering that

$$p_{X|E^{\uparrow\{x,y\}}}(x) = \sum_{y\in E} p_{X,Y|E^{\uparrow\{x,y\}}}(x,y) \tag{1.24}$$

and

$$H(X|E^{\uparrow\{x,y\}}) = -\sum_{x\in S_1} p_{X|E^{\uparrow\{x,y\}}}(x) \log p_{X|E^{\uparrow\{x,y\}}}(x). \tag{1.25}$$

We notice that this information is in general different from zero. Thus, even an event relating to variable Y carries information relative to variable X. This results from the correlation between the two variables X and Y. Indeed, assuming that X and Y are stochastically independent, i.e. $p_{X,Y}(x,y) = p_X(x)p_Y(y)$, then

$$p_{X,Y|E^{\uparrow\{x,y\}}}(x,y) = \frac{p_X(x)p_Y(y)}{p_Y(E)} = p_X(x)p_{Y|E}(y).$$

Thus in this case we obtain

$$p_{X|E^{\uparrow\{x,y\}}}(x) = \sum_{y \in E} p_{X,Y|E^{\uparrow\{x,y\}}}(x,y) = p_X(x).$$

Thus, we have that $H(X|E^{\uparrow\{x,y\}}) = H(X)$ and therefore $i(E/X) = 0$.

Of course, by symmetry, a similar analysis can be carried out for an event related to variable Y.

Example 1.25 (Fair Coin - Continuation) We are referring to the example 1.24. But now, we observe, that the second throw resulted in "heads". This is represented by the event $E_Y = \{1\}$. We easily see, that $p(E) = 0.5$. To compute $i(E_Y/X,Y)$ we need the cylindric extension of E_Y given by

$$E_Y^{\uparrow\{x,y\}} = \{(0,1),(1,1)\}.$$

Regarding both throws, we obtain

$$p_{X,Y|E_Y^{\uparrow\{x,y\}}}(0,1) = p_{X,Y|E_Y^{\uparrow\{x,y\}}}(1,1) = \frac{1}{2}$$

and 0 otherwise. So the information obtained by E_Y relative to both throws is

$$\begin{aligned} i(E/X,Y) &= H(X,Y) - H(X,Y|E_Y^{\uparrow\{x,y\}}) \\ &= 2 + \frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2} \\ &= 1 \text{ bit.} \end{aligned}$$

Assume next, that $E = \{(x,y)\}$, that is that an exact value is observed for both random variables X and Y. Then, we obtain the following measures of information,

$$\begin{aligned} i(x,y/X) &= i(x/X) = H(X), \\ i(x,y/Y) &= i(y/Y) = H(Y), \\ i(x,y/X,Y) &= H(X,Y) \leq i(x/X) + i(y/Y). \end{aligned} \tag{1.26}$$

The first two equalities hold true since $H(X|x,y) = H(X|x) = 0$ and $H(Y|x,y) = H(Y|y) = 0$. The last inequality is nothing else than (1.7) (subsection 1.1.3). In this case, clearly all three information measures are non-negative. Equality holds in the last inequality, when X and Y are *independent*. Then the individual pieces of information, bearing on the variables X and Y respectively, add to the total information bearing on both variables simultaneously.

The condition for the additivity of information can be generalized as shown in the following theorem.

Theorem 1.7 *Let X and Y be* independent *random variables relative to a probabilistic choice situation $(S_1 \times S_2, P)$. If $E = E^{\downarrow\{x\}} \times E^{\downarrow\{y\}}$, then*

$$i(E/X,Y) = i(E/X) + i(E/Y). \tag{1.27}$$

PROOF Since X and Y are independent, we have $H(X,Y) = H(X)+H(Y)$ (see (1.7) in subsection 1.1.3). Furthermore, the conditional variables $X|E^{\downarrow\{x\}}$ and $Y|E^{\downarrow\{y\}}$ are still independent, since, for $x \in E^{\downarrow\{x\}}$ and $y \in E^{\downarrow\{y\}}$ we have

$$\begin{aligned} & p_{X,Y|E}(x,y) \\ = & \frac{p_{X,Y}(x,y)}{p_{X,Y}(E)} \\ = & \frac{p_X(x)p_Y(y)}{p_X(E^{\downarrow\{x\}})p_Y(E^{\downarrow\{y\}})} \\ = & p_{X|E^{\downarrow\{x\}}}(x)p_{Y|E^{\downarrow\{y\}}}(y). \end{aligned}$$

Therefore, again from (1.7), we also have $H(X,Y|E) = H(X|E^{\downarrow\{x\}}) + H(Y|E^{\downarrow\{y\}})$. Thus, we obtain

$$\begin{aligned} & i(E/X,Y) \\ = & H(X,Y) - H(X,Y|E) \\ = & (H(X) - H(X|E^{\downarrow\{x\}})) + (H(Y) - H(Y|E^{\downarrow\{y\}})) \\ = & i(E^{\downarrow\{x\}}/X) + i(E^{\downarrow\{y\}}/Y). \end{aligned}$$

Example 1.26 (Fair Coin - Continuation) We are referring to the example 1.25. Since we know the result of the second throw, we get $i(E_Y^{\uparrow\{x,y\}}/Y) = H(Y) = \log 2 = 1$ bit. The cylindric extension of the event E_Y does not add information relative to X, thus $i(E_Y^{\uparrow\{x,y\}}/X) = 0$ bit. So we get the expected result

$$i(E_Y^{\uparrow\{x,y\}}/X,Y) = i(E_Y^{\uparrow\{x,y\}}/X) + i(E_Y^{\uparrow\{x,y\}}/Y) = 1 \text{ bit}.$$

Two events E_1 and E_2 which each bear on one of the variables X and Y, i.e. $d(E_1) = \{x\}$ and $d(E_2) = \{y\}$, are called *independent*. Theorem 1.7 says that the information content of their combination $E_1 \times E_2$ is the sum of the information contents of each event, provided that the random variables X and Y are independent. This carries the addition theorem from entropies over to information measures.

In the previous sections, we showed that a choice situation without probabilities has the same entropy as the probabilistic choice situation with uniform probability distribution over the possible choices. This corresponds to Laplace's *principle of insufficient reason*. In the next example, we want to draw attention to the danger of an unreflected application of this principle.

Example 1.27 (Fair Coin - Continuation) We refer back to example 1.24 where a fair coin is thrown twice and the event E reported that heads did not turn out twice. We now drop the assumption that the coin is fair. The coin can be anything. So we do not have a uniform distribution over the four possible outcomes. In fact we have no probability distribution at all. That is, we have a scheme of choice $S = \{(0,0),(0,1),(1,0),(1,1)\}$ without probabilities. The uncertainty $h(|S|)$ of this compound choice scheme is 2 bit as before; here Laplace's principle of insufficient reason still applies. Also $h(|E|) = \log 3$ bit as before.

But what is the uncertainty regarding the first throw? We insist that we have no

reason to assume a uniform distribution over the four possibilities before the event E is reported and over the three remaining possibilities, after event E is reported. We then simply have a choice scheme $E = \{(0,0),(0,1),(1,0)\}$ *without probabilities*. So, when we regard the first throw, we have the two possibilities given by $E^{\downarrow\{x\}} = \{0,1\}$ and hence the corresponding choice situation without probabilities. So the remaining uncertainty is $h(|E^{\downarrow\{x\}}|) = 1$ bit as before the event E became known. Hence the information is null, $i(E/X) = 0$ bit.

The fact is, knowing that the coin is fair is *information*, represented by the uniform distribution. This additional information together with the event E yields information on the first throw. If we do not know that the coin is fair, or if we have no reason to assume that it is so (like if the coin is ordinary), then we do not have this information and the inference would be biased, if we simply assumed it without reason.

What this example shows, is that a probability distribution over a set of possible choices is *information*. And assuming a *uniform* distribution is replacing *ignorance* by *information*. Sometimes this does not matter, but in general it does! This reinforces the *relativity principle* of information stating that a measure of information is always relative to prior information.

In particular, this example shows that we need to treat choice schemes without probabilities and those with probabilities differently. In fact, it will be possible to join both cases in an appropriate unique formalism. But this goes beyond this introductory chapter and is postponed until chap. 1.2.5.

Another thing, we saw with the model case of two variables, is that events or information may relate to different questions, but still carry information to other questions. In fact there is some order between questions: The question related to variable X or domain $\{x\}$, and represented by the set of possible values $\mathcal{V}_X$ is somehow coarser, than the question related to both variables X, Y or domain $\{x,y\}$, and represented by the set $\mathcal{V}_X \times \mathcal{V}_Y$. And an information related to some domain (say $\{x,y\}$) also carries information for the coarser domains $\{x\}$ and $\{y\}$.

So already this first very simple model case of two variables hints to a lot of the structure of information and questions; structure which has to be formalized, if information in all its generality is to be understood. We shall treat these aspects in a more general case in the next subsection.

Summary for Section 1.2

- Events refer to one or the other or to both variables. In any case they represent information. But its amount is to be measured with respect to a specified variable or question.
- The second principle of relativity says that the amount of information in an observed event is also measured relative to a prior information. Prior information is represented by the prior probability distribution (before observation of the event) of the considered question, that is the probabilistic choice situation or random variable.
- Independent events with respect to independent random variables carry information which sum up to the total information represented by the two events

simultaneously. This is the addition theorem for independent information.

- Since probability distributions also represent information, choice situations without probabilities must be carefully distinguished from choice situations with probabilities.

Control Question 16

Relativity regarding the

1. posterior information;
2. question;
3. maximal gain;
4. expected change of entropy;

is a basic principle of information.

Answer

1. No, one of the basic principles of information is relativity regarding **prior** information.
2. That is correct; information is always relative to a question.
3. We have never seen a relativity principle called "maximal gain".
4. We have never seen a relativity principle called "expected change of entropy".

Control Question 17

Someone is telling you that the weather will be fine tomorrow. Is this information a gain for you?

1. Yes, of course.
2. No!

Answer

Both possibilities are wrong, because information is always measured relative to a specified question. Hence it depends on the question if the information is a gain or a loss.

Control Question 18

What are conditions on the random variables X and Y relative to a probabilistic choice system $(S_1 \times S_2, P)$ and an event E, such that the additivity of information holds, that is $i(E/X,Y) = i(E/X) + i(E/Y)$:

1. X and Y are independent.
2. P is the uniform probability distribution.
3. $E = E^{\downarrow\{x\}} \times E^{\downarrow\{y\}}$.
4. $H(X,Y) = H(X) + H(Y)$.

Answer

1. That is correct. X and Y have to be independent.
2. No, there is no limitation on P.
3. That is indeed an important condition.
4. This is equivalent to the condition "X and Y independent"; since X and Y have to be independent, this condition is correct.

Control Question 19

Given a choice system $(S_1 \times S_2)$ with the corresponding random variables X, Y and an event $E = \{y\} \subseteq S_2$. Then

1. $i(E/Y) = 0$;
2. $i(E/X) = 0$;
3. $i(E/X,Y) = 0$.

Answer

1. Information is definded by the change of entropy, hence, here we have $i(E/Y) = H(Y) - H(Y|E) = H(Y)$. So this assertion is wrong.
2. Since the observation of an event $E \subseteq S_2$ can also affect the random variable X, this proposition is wrong.
3. $i(E/X,Y) = H(X,Y) - H(X,Y|E)$ and we can't expect that $H(X,Y) = H(X,Y|E)$. It follows that this assertion is also untrue.

1.2.3 Mutual Information and Kullback-Leibler Divergence

Learning Objectives for Subsection 1.2.3

After studying this subsection you should understand

- the notion of mutual information and its relation to entropy and information;
- the notion of informational distance (Kullback-Leibler divergence) and its relation to mutual information.

We look once more at a compound probabilistic choice situation $(S_1 \times S_2, P)$ and the associated random variables X and Y. If a value y is observed for Y, then as seen in subsection 1.2.2, we get the amount of information $i(y/X) = H(X) - H(X|y)$ with respect to X. But rather than looking at a particular observation y, we look at the *expected* amount of information relative to X gained by observing Y. This value is

$$I(X|Y) = \sum_y p_Y(y) i(y/X) = \sum_y p_Y(y)(H(X) - H(X|y)) = H(X) - H(X|Y). \quad (1.28)$$

$I(X|Y)$ is called the *mutual information* between X and Y. It is an important notion in information theory, but it is not an information, strictly speaking. It is the expected or average amount of information obtained on X by observing Y. In the corollary 1.3 in subsection 1.1.3, we have seen that always $H(X|Y) \leq H(X)$ and equality holds only, if X and Y are independent. This implies that the following property holds.

Theorem 1.8

$$I(X|Y) \geq 0, \quad (1.29)$$

and $I(X|Y) = 0$ if, and only if, X and Y are independent.

So, although in a particular case the information $i(y/X)$ may be negative, on average we expect a positive amount of information about X from observing Y.

Example 1.28 (Communication Channel) Consider a communication channel, with X the input source and Y the output. By observing the output Y we expect a positive amount of information regarding the input X. Although in particular transmissions, uncertainty about the input is increased, on average the observation of the output decreases uncertainty about the input. This is of course extremely important for reasonable communication.

Furthermore, from (1.8) we obtain

$$I(X|Y) = H(X) + H(Y) - H(X, Y).$$

Because this formula is *symmetric* in X and Y, we conclude that $I(X|Y) = I(Y|X)$. We expect as much information on X by observing Y than on Y by observing X.

Theorem 1.9

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

That is a remarkable result. So mutual information between X and Y is the same as between Y and X. This symmetry is also evident from the following formula for the mutual information, which we obtain from (1.28), if we introduce the definition of the

entropies appearing there,

$$\begin{aligned} I(X|Y) &= -\sum_x p_X(x)\log p_X(x) + \sum_{x,y} p_Y(y)p_{X|Y}(x,y)\log p_{X|Y}(x,y) \\ &= \sum_{x,y} p_{X,Y}(x,y)(\log\frac{p_{X,Y}(x,y)}{p_Y(y)} - \log p_X(x)) \\ &= \sum_{x,y} p_{X,Y}(x,y)\log\frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}. \end{aligned} \tag{1.30}$$

Finally, we conclude from $I(X|Y) \geq 0$ and $H(X|Y) \geq 0$ that $I(X|Y) \leq H(X) = i(x/X)$. By symmetry we have also $I(Y|X) \leq H(Y) = i(y/Y)$. The expected information on either variable obtained by observing the other one, is always less than the information gained by directly observing either of the variables. Now, of course in a transmission system, it is not possible to observe the input directly. That is why a loss of information has to be expected when transmitting information.

Example 1.29 (Statistical Inference) Let X and Y be binary random variables representing a throw of a coin. The coin may be fair or not. This is represented by a choice system or a random variable Q. Its probability distribution is given by $p_Q(\text{fair}) = p_Q(\text{unfair}) = 0.5$. We know, that if the coin is fair, we have

$$p_{X|Q}(0|\text{fair}) = p_{X|Q}(1|\text{fair}) = p_{Y|Q}(0|\text{fair}) = p_{Y|Q}(1|\text{fair}) = 0.5$$

and if it is unfair,

$$\begin{aligned} p_{X|Q}(0|\text{unfair}) &= p_{Y|Q}(0|\text{unfair}) = 0.9, \\ p_{X|Q}(1|\text{unfair}) &= p_{Y|Q}(1|\text{unfair}) = 0.1. \end{aligned}$$

We further assume that X and Y are independent, thus, the conditional probability distribution of $(X,Y|Q)$ is given by

$$\begin{aligned} &p_{X,Y|Q}(0,0|\text{fair}) = p_{X,Y|Q}(0,1|\text{fair}) = p_{X,Y|Q}(1,0|\text{fair}) = p_{X,Y|Q}(1,1|\text{fair}) = 0.25, \\ &p_{X,Y|Q}(0,0|\text{unfair}) = 0.81, \\ &p_{X,Y|Q}(0,1|\text{unfair}) = p_{X,Y|Q}(1,0|\text{unfair}) = 0.09, \\ &p_{X,Y|Q}(1,1|\text{unfair}) = 0.01. \end{aligned}$$

Since

$$\begin{aligned} p_{X,Y}(0,0) &= p_{X,Y|Q}(0,0|\text{fair})p_Q(\text{fair}) + p_{X,Y|Q}(0,0|\text{unfair})p_Q(\text{unfair}) \\ &= 0.25\cdot 0.5 + 0.81\cdot 0.5 = 0.53, \\ p_{X,Y}(0,1) &= p_{X,Y|Q}(0,1|\text{fair})p_Q(\text{fair}) + p_{X,Y|Q}(0,1|\text{unfair})p_Q(\text{unfair}) \\ &= 0.25\cdot 0.5 + 0.09\cdot 0.5 = 0.17, \\ p_{X,Y}(1,0) &= p_{X,Y|Q}(1,0|\text{fair})p_Q(\text{fair}) + p_{X,Y|Q}(1,0|\text{unfair})p_Q(\text{unfair}) \\ &= 0.25\cdot 0.5 + 0.09\cdot 0.5 = 0.17, \\ p_{X,Y}(1,1) &= p_{X,Y|Q}(1,1|\text{fair})p_Q(\text{fair}) + p_{X,Y|Q}(1,1|\text{unfair})p_Q(\text{unfair}) \\ &= 0.25\cdot 0.5 + 0.01\cdot 0.5 = 0.13, \end{aligned}$$

we obtain

$$\begin{aligned} H(X,Y) &= -p_{X,Y}(0,0)\log p_{X,Y}(0,0) - p_{X,Y}(0,1)\log p_{X,Y}(0,1) \\ &\quad -p_{X,Y}(1,0)\log p_{X,Y}(1,0) - p_{X,Y}(1,1)\log p_{X,Y}(1,1) \\ &= -0.53\log 0.53 - 0.17\log 0.17 - 0.17\log 0.17 - 0.13\log 0.13 \\ &\approx 1.7373 \text{ bit.} \end{aligned}$$

An finally, with

$$\begin{aligned} H(X,Y|\text{fair}) &= -\sum_{x,y} p_{X,Y|Q}(x,y|\text{fair})\log p_{X,Y|Q}(x,y|\text{fair}) \\ &= -p_{X,Y|Q}(0,0|\text{fair})\log p_{X,Y|Q}(0,0|\text{fair}) \\ &\quad -p_{X,Y|Q}(0,1|\text{fair})\log p_{X,Y|Q}(0,1|\text{fair}) \\ &\quad -p_{X,Y|Q}(1,0|\text{fair})\log p_{X,Y|Q}(1,0|\text{fair}) \\ &\quad -p_{X,Y|Q}(1,1|\text{fair})\log p_{X,Y|Q}(1,1|\text{fair}) \\ &= \log 4 = 2 \text{ bit,} \\ H(X,Y|\text{unfair}) &= -0.81\log 0.81 - 0.09\log 0.09 - 0.09\log 0.09 - 0.01\log 0.01 \\ &\approx 0.938 \text{ bit,} \\ H(X,Y|Q) &= \sum_q p_Q(q)H(X,Y|q) \\ &= p_Q(\text{fair})H(X,Y|\text{fair}) + p_Q(\text{unfair})H(X,Y|\text{unfair}) \\ &\approx 0.5\cdot 2 + 0.5\cdot 0.938 \\ &\approx 1.469 \text{ bit,} \end{aligned}$$

we have

$$I(X,Y|Q) = H(X,Y) - H(X,Y|Q) \approx 1.7373 - 1.469 \approx 0.2683 \text{ bit.}$$

This is the expected information about the question whether the coin is fair, if we observe two throws of the coin.

Example 1.30 (Symmetric Binary Channel - Continuation) In a communication channel we must expect to get less information on the input by observing the output, than by directly observing the input. For example consider the symmetric binary channel from example 1.14 with $\varepsilon = 0.1$, $p_X(0) = 0.2$ and $p_X(1) = 0.8$. We saw, that $p_Y(0) = 0.26$ and $p_Y(1) = 0.74$, thus

$$H(Y) = -0.26\log 0.26 - 0.74\log 0.74 \approx 0.8267 \text{ bit,}$$

and we obtain

$$\begin{aligned} I(X|Y) &= I(Y|X) = H(Y) - H(Y|X) \\ &= -0.26\log 0.26 - 0.74\log 0.74 + 0.9\log 0.9 + 0.1\log 0.1 \\ &\approx 0.3578 \text{ bit.} \end{aligned}$$

But $H(X) = -0.2\log 0.2 - 0.8\log 0.8 \approx 0.7219$ bit $> I(X|Y)$. That means that the expected loss of information is $H(X) - I(X|Y) \approx 0.7219 - 0.3578 = 0.3641$ bit.

We introduce another important notion of information theory. Consider two probabilistic choice systems, with the *same* choice set S, but different probability distributions. If X and Y are the corresponding random variables, then we define

$$K(P_X, P_Y) = \sum_{x \in S} p_X(x) \log \frac{p_X(x)}{p_Y(x)}. \tag{1.31}$$

This is called the *Kullback-Leibler divergence* between X and Y. It is a kind of distance between the probability distributions of X and Y, although $d(P_X, P_Y) \neq d(P_Y, P_X)$. But

Theorem 1.10

$$K(P_X, P_Y), K(P_Y, P_X) \geq 0,$$

and

$$K(P_X, P_Y) = K(P_Y, P_X) = 0$$

if, and only if, $P_X = P_Y$.

PROOF Indeed, we have

$$K(P_X, P_Y) = \sum_{x \in S} p_X(x) \log p_X(x) - \sum_{x \in S} p_X(x) \log p_Y(x).$$

Lemma 1.1 tells us then that $K(P_X, P_Y) \geq 0$ and equal to 0 only if $p_X(x) = p_Y(x)$. ■

Example 1.31 (Symmetric Binary Channel - Continuation) Once more we consider the symmetric binary channel from the examples 1.14 and 1.30. Here we have

$$\begin{aligned}
K(P_X, P_Y) &= p_X(0) \log \frac{p_X(0)}{p_Y(0)} + p_X(1) \log \frac{p_X(1)}{p_Y(1)} \\
&= 0.2 \log \frac{0.2}{0.26} + 0.8 \log \frac{0.8}{0.74} \\
&\approx 0.0143 \text{ bit}, \\
K(P_Y, P_X) &= p_Y(0) \log \frac{p_Y(0)}{p_X(0)} + p_Y(1) \log \frac{p_Y(1)}{p_X(1)} \\
&= 0.26 \log \frac{0.26}{0.2} + 0.74 \log \frac{0.74}{0.8} \\
&\approx 0.0152 \text{ bit}.
\end{aligned}$$

Consider a compound probabilistic situation $(S \times S, P)$ and the associated pair of random variables X and Y, which have both the same set of values S. Denote by (S, P_X) and (S, P_Y) the probabilistic choice situations related to the two variables X and Y. That is, p_X and p_Y are the marginal distributions of X and Y, given by

$$p_X(x) = \sum_y p_{X,Y}(x, y), \quad p_Y(y) = \sum_x p_{X,Y}(x, y).$$

Furthermore $P_X \cdot P_Y$ denotes the probability distribution with values $p_X(x) \cdot p_Y(y)$. Then (1.30) shows that

$$I(X|Y) = I(Y|X) = K(P, P_X \cdot P_Y). \tag{1.32}$$

In this view, $I(X|Y)$ measures to what degree the common probability distribution of the pair (X, Y) diverges from the case of *independence*. That is, why the mutual information is also sometimes considered as a measure of dependence between two random variables with the same set of values.

Finally we have (see 1.30)

$$\begin{aligned} I(X|Y) &= \sum_{x,y} p_{X,Y}(x,y) \log \frac{p_{Y|x}(x,y)}{p_Y(y)} \\ &= \sum_{x,y} p_X(x) p_{Y|x}(x,y) \log \frac{p_{Y|x}(x,y)}{p_Y(y)} \\ &= \sum_x p_X(x) K(P_{Y|x}, P_Y) \\ &= \sum_x p_X(x) i(x/Y). \end{aligned} \tag{1.33}$$

$i(x/Y) = H(Y) - H(Y|x)$ measures the information gained on Y by observing $X = x$. Thus, although in general $K(P_{Y|x}, P_Y) \neq i(x/Y)$, there is equality on average over x between this information and the Kullback-Leibler divergence $K(P_{Y|x}, P_Y)$. By symmetry we also have

$$I(Y|X) = \sum_y p_Y(y) K(P_{X|y}, P_X) = \sum_y p_Y(y) i(y/X).$$

Example 1.32 (Kullback-Leibler Divergence and Mutual Information) Consider $S = (e_1, e_2)$, $S \times S = \{(e_1,e_1), (e_1,e_2), (e_2,e_1), (e_2,e_2)\}$ and $P = \{0.5, 0.1, 0.3, 0.1\}$. Let X and Y be the random variables associated to the compound probabilistic situation $(S \times S, P)$. Thus, the marginal distributions of X and Y are given by

$$\begin{aligned} p_X(e_1) &= p_{X,Y}(e_1,e_1) + p_{X,Y}(e_1,e_2) = 0.5 + 0.1 = 0.6, \\ p_X(e_2) &= p_{X,Y}(e_2,e_1) + p_{X,Y}(e_2,e_2) = 0.3 + 0.1 = 0.4, \\ p_Y(e_1) &= p_{X,Y}(e_1,e_1) + p_{X,Y}(e_2,e_1) = 0.5 + 0.3 = 0.8, \\ p_Y(e_2) &= p_{X,Y}(e_1,e_2) + p_{X,Y}(e_2,e_2) = 0.1 + 0.1 = 0.2. \end{aligned}$$

We define the distribution $P_X \cdot P_Y$ by $p(x,y) = p_X(x) p_Y(y)$. Let us now compute some entropies. With

$$
\begin{aligned}
p_{X|Y=e_1}(e_1,e_1) &= \frac{p_{X,Y}(e_1,e_1)}{p_Y(e_1)} = \frac{0.5}{0.8} = 0.625,\\
p_{X|Y=e_1}(e_2,e_1) &= \frac{p_{X,Y}(e_2,e_1)}{p_Y(e_1)} = \frac{0.3}{0.8} = 0.375,\\
p_{X|Y=e_2}(e_1,e_2) &= \frac{p_{X,Y}(e_1,e_2)}{p_Y(e_2)} = \frac{0.1}{0.2} = 0.5,\\
p_{X|Y=e_2}(e_2,e_2) &= \frac{p_{X,Y}(e_2,e_2)}{p_Y(e_2)} = \frac{0.1}{0.2} = 0.5,
\end{aligned}
$$

we obtain

$$
\begin{aligned}
H(X) &= -0.6\log 0.6 - 0.4\log 0.4 \approx 0.9710 \text{ bit},\\
H(X|e_1) &= -p_{X|Y=e_1}(e_1,e_1)\log p_{X|Y=e_1}(e_1,e_1) - p_{X|Y=e_1}(e_2,e_1)\log p_{X|Y=e_1}(e_2,e_1)\\
&= -0.625\log 0.625 - 0.375\log 0.375\\
&\approx 0.9544 \text{ bit},\\
H(X|e_2) &= -p_{X|Y=e_2}(e_1,e_2)\log p_{X|Y=e_2}(e_1,e_2) - p_{X|Y=e_2}(e_2,e_2)\log p_{X|Y=e_2}(e_2,e_2)\\
&= -0.5\log 0.5 - 0.5\log 0.5\\
&= 1 \text{ bit},\\
H(X|Y) &= p_Y(e_1)H(X|e_1) + p_Y(e_2)H(X|e_2)\\
&\approx 0.8\cdot 0.9544 + 0.2\cdot 1 = 0.9635 \text{ bit}.
\end{aligned}
$$

Hence $I(X|Y) = H(X) - H(X|Y) \approx 0.9710 - 0.9635 \approx 0.0074$ bit. Since

$$
\begin{aligned}
K(P, P_X\cdot P_Y) &= \sum_{(x,y)\in S\times S} p_{X,Y}(x,y)\log\frac{p_{X,Y}(x,y)}{p(x,y)}\\
&= p_{X,Y}(e_1,e_1)\log\frac{p_{X,Y}(e_1,e_1)}{p(e_1,e_1)} + p_{X,Y}(e_1,e_2)\log\frac{p_{X,Y}(e_1,e_2)}{p(e_1,e_2)}\\
&\quad + p_{X,Y}(e_2,e_1)\log\frac{p_{X,Y}(e_2,e_1)}{p(e_2,e_1)} + p_{X,Y}(e_2,e_2)\log\frac{p_{X,Y}(e_2,e_2)}{p(e_2,e_2)}\\
&= 0.5\log\frac{0.5}{0.6\cdot 0.8} + 0.1\log\frac{0.1}{0.6\cdot 0.2} + 0.3\log\frac{0.3}{0.4\cdot 0.8} + 0.1\log\frac{0.1}{0.4\cdot 0.2}\\
&\approx 0.0074 \text{ bit},
\end{aligned}
$$

we verify that $I(X|Y) = K(P, P_X\cdot P_Y)$. And finally, with

$$
\begin{aligned}
p_{Y|X=e_1}(e_1,e_1) &= \frac{p_{X,Y}(e_1,e_1)}{p_X(e_1)} = \frac{0.5}{0.6} = \frac{5}{6},\\
p_{Y|X=e_1}(e_2,e_1) &= \frac{p_{X,Y}(e_1,e_2)}{p_X(e_1)} = \frac{0.1}{0.6} = \frac{1}{6},\\
p_{Y|X=e_2}(e_1,e_2) &= \frac{p_{X,Y}(e_2,e_1)}{p_X(e_2)} = \frac{0.3}{0.4} = \frac{3}{4},\\
p_{Y|X=e_2}(e_2,e_2) &= \frac{p_{X,Y}(e_2,e_2)}{p_X(e_2)} = \frac{0.1}{0.4} = \frac{1}{4},
\end{aligned}
$$

and

$$\begin{aligned} H(Y) &= -0.8\log 0.8 - 0.2\log 0.2 \approx 0.7219 \text{ bit}, \\ i(e_1/Y) &= H(Y) - H(Y|e_1) \approx 0.7219 + \frac{5}{6}\log\frac{5}{6} + \frac{1}{6}\log\frac{1}{6} \approx 0.0719 \text{ bit}, \\ i(e_2/Y) &= H(Y) - H(Y|e_2) \approx 0.7219 + \frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4} \approx -0.00894 \text{ bit}, \end{aligned}$$

we obtain

$$\begin{aligned} I(X|Y) &= \sum_x p_X(x) i(x/Y) = p_X(e_1) i(e_1/Y) + p_X(e_2) i(e_2/Y) \\ &\approx 0.6 \cdot 0.0719 - 0.4 \cdot 0.0894 \approx 0.0074 \text{ bit}. \end{aligned}$$

It is interesting to note that $i(e_2/Y)$ is negative.

Summary for Section 1.2

- We have defined *mutual information* $I(X|Y)$ between two random variables X and Y as the expected gain in information on one variable, obtained if the other one is observed. Remarkably, this value is symmetric, $I(X|Y) = I(Y|X)$.
- Mutual information is also the difference between the sum of the individual entropies of the two variables and the actual entropy of the pair. This shows that mutual information is always non-negative and vanishes exactly, if the two variables X and Y are independent. Thus it also measures the reduction of uncertainty of the actual situation with respect to the case of independent variables.
- The *Kullback-Leibler divergence* $K(P_X, P_Y)$ of two probability distributions P_X and P_Y measures the "distance" from P_X to P_Y. However, in general $K(P_X, P_Y) \neq K(P_Y, P_X)$. Nevertheless we have $K(P_X, P_Y) = K(P_Y, P_X) = 0$ if $P_X = P_Y$.
- The mutual information $I(X|Y)$ equals the Kullback-Leibler divergence from the common distribution of the pair (X, Y) to the product of their marginal distributions. This is another measure of the distance of the actual situation and the assumed case of independence.
- The information gained on a variable Y by observing another variable X equals on average the Kullback-Leibler divergence between the conditional distribution of Y given $X = x$ and the unconditional marginal distribution of Y.

Control Question 20

What can we say about the mutual Information between X and Y?

1. The mutual information is the expected amount of information relative to X by observing Y.
2. In some cases the mutual information can be negative.

3. The mutual information can be zero; even if X and Y are independent.

4. The mutual information between X and Y equals the mutual information between Y and X.

Answer

1. That fits our definition of the mutual information exactly, hence it is correct.

2. Incorrect, since $I(X|Y) = H(X) - H(X|Y)$ and $H(X|Y) \leq H(X)$ it follows, that $I(X|Y) \geq 0$.

3. The mutual information is equal to zero, if, and only if, X and Y are independent, hence the assertion is wrong.

4. We have $I(X|Y) = H(X) + H(Y) - H(X,Y) = H(X) + H(Y) - H(Y,X) = I(Y|X)$, hence the proposition is correct.

Control Question 21

1. $K(P_X, P_Y) = K(P_Y, P_X)$, since K is a kind of distance.

2. $K(P_X, P_Y) = 0$, if X and Y are independent.

3. $I(X|Y) = K(P_X \cdot P_Y, P)$, if X and Y are the random variables associated to the compound probabilistic situation $(S \times S, P)$.

Answer

1. We said that K is a kind of distance, however, $K(P_X, P_Y) \neq K(P_Y, P_X)$.

2. $K(P_X, P_Y) = 0$ if, and only if, $P_X = P_Y$, hence the assertion is false.

3. It is true, that $I(X|Y) = K(P, P_X \cdot P_Y)$, but, since $K(P_X, P_Y) \neq K(P_Y, P_X)$, the proposition is wrong.

1.2.4 Surprise, Entropy and Information

Learning Objectives for Subsection 1.2.4

After studying this subsection you should understand

- the notion of degree of surprise associated with an event;
- its relation to entropy, measures of information and mutual information.

If $E \subseteq S$ is a rare event in a probabilistic choice situation (S, P), then we may be very surprised when we actually observe it. So, it may be interesting to *measure* the degree of unexpectedness or of surprise of an event. Let's denote the degree of unexpectedness of the event E by $s(E)$. The following characteristics seem reasonable to assume for it:

- $s(E)$ should only depend on the *probability* $p(E)$ of the event E, i.e.
$$s(E) = f(p(E)).$$
- $s(E)$ should be a monotone decreasing function of its probability $p(E)$. The greater the probability, the smaller the unexpectedness of the event; the smaller the probability, the greater the surprise.
- If two events E_1 and E_2 are independent, then the degree of surprise of their common occurrence should be the *sum* of their individual degrees of surprise,
$$s(E_1 \cap E_2) = s(E_1) + s(E_2).$$

The logarithm is the only function satisfying these requirements,

$$s(E) = -\log p(E).$$

If we (arbitrarily) require in addition that the degree of surprise of an event with probability $1/2$ equals 1, $f(1/2) = 1$, then the logarithm must be taken to the base 2. So $-\log_2 p(E)$ is considered as a measure of the degree of surprise or unexpectedness of an event E.

Some authors define $-\log p(E)$ to be the "information contained in E" or the "self-information of E". We disagree with this view: An information is a, possibly partial, answer to a precise question and it is information in so far as it changes the uncertainty about the possible answer to this question. If we consider a single event E, then no specific question is associated with it. On the other hand, the degree of unexpectedness is associated with an *event* and not a specified *question*.

Example 1.33 (Lottery) Suppose you win in a lottery where odds are 1 to say 10^{15} or something like that. You will be (agreeably) surprised. If you do not win, you will not be surprised at all. The amount of information that you win relative to the question "win or not?" is the same, as it is if you find out that you do not win. The amount of the *same* information (i.e. the number drawn in the lottery) relative to the question "which number is drawn" is much larger. The degree of surprise associated with the number drawn on the other hand does not depend on the question asked.

Example 1.34 (Swiss-Lotto) We are playing Swiss-Lotto, that means choosing 6 numbers out of 45. There are exactly

$$\binom{45}{6} = \frac{45!}{6! \cdot (45-6)!} = 8\text{'}145\text{'}060$$

possibilities, so, the degree of unexpectedness $s(\text{win}) = \log 8\text{'}145\text{'}060 \approx 22.9575$. Let X denote the random variable whether we win or not. Thus

$$\begin{aligned} p(X = \text{win}) &= \frac{1}{8\text{'}145\text{'}060}, \\ p(X = \text{not win}) &= 1 - \frac{1}{8\text{'}145\text{'}060}, \end{aligned}$$

and hence

$$\begin{aligned} H(X) &= -\frac{1}{8\text{‘}145\text{‘}060}\log\frac{1}{8\text{‘}145\text{‘}060} - \left(1-\frac{1}{8\text{‘}145\text{‘}060}\right)\log\left(1-\frac{1}{8\text{‘}145\text{‘}060}\right) \\ &\approx 2.9957\cdot 10^{-6} \text{ bit.} \end{aligned}$$

The uncertainty whether we win or not is really small, because it is almost sure that we will not win.

To further clarify the difference between surprise and information, consider a probabilistic choice situation where the possible choices S are only E or not E (that is E^c) with probabilities p and $1-p$ for the two possible cases. The uncertainty associated with this situation is

$$-p\log p - (1-p)\log(1-p).$$

In figure 1.13, the graph of this uncertainty and the degree of surprise $-\log p$ as a function of p is drawn. Uncertainty is maximal for $p = 1/2$. This is also the maximum information obtained by observing E with respect to the question whether E happens or not. Uncertainty is null if either $p = 0$ or $p = 1$. Surprise however is maximal (infinite in fact), if $p = 0$. In this case we do not at all expect E to happen. Surprise is null on the other hand, if $p = 1$. In this case we are sure that E will happen, and are not surprised at all, if E indeed occurs.

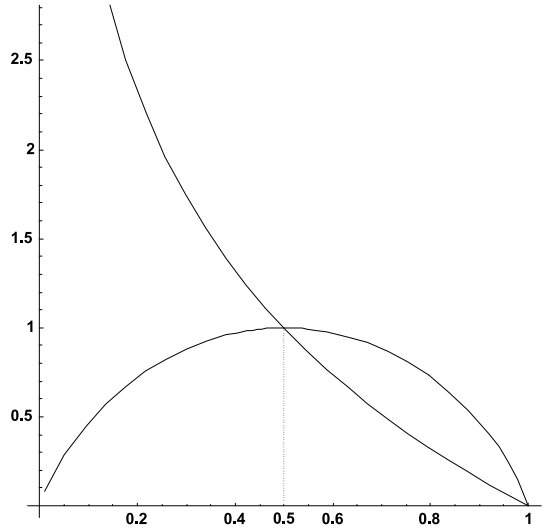


Figure 1.13: Information contained in event E relative to the question "E or not E" versus degree of surprise in E.

If we look at the entropy

$$H(X) = -p_X(x_1)\log p_X(x_1) - p_X(x_2)\log p_X(x_2) - \cdots - p_X(x_m)\log p_X(x_m)$$

of a random variable X, then we see that it is the *expected value* of the unexpectedness $-\log p_X(x_i)$ of the events $X = x_i$. So, surprise, entropy and information are closely related, but different concepts. The amount of information (of a value of a random variable observed) is the average or expected degree of surprise of the possible events.

There is even another relation. Let X and Y be two random variables associated with the compound probabilistic choice situation $(S_1 \times S_2, P)$. How does the unexpectedness

of the event $X = x$ change, if $Y = y$ is observed? Originally, the degree of surprise of the event $\{X = x\}$ is $-\log p_X(x)$. Once $Y = y$ is observed it changes to

$$-\log p_{X|y}(x|y) = -\log \frac{p_{X,Y}(x,y)}{p_Y(y)}.$$

So, the change of surprise is

$$-\log p_X(x) + \log \frac{p_{X,Y}(x,y)}{p_Y(y)} = \log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}.$$

From this we see that the mutual information

$$I(X|Y) = \sum_{x,y} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}$$

is the expected change of surprise for a value of X given that a value of Y is observed. Or, in other words, the expected amount of information gained with respect to X by observing Y equals the expected change of unexpectedness of observing X given that Y is observed. So, this is another link between amount of information and degree of surprise.

Summary for Section 1.2

- We defined the degree of unexpectedness or of surprise of an event as the negative of the logarithm of its probability. It grows as the probability of the event decreases, is zero for a probability of one, and infinite for a probability of zero.
- Surprise is not to be confused with information. The former is simply associated with an event, the latter is measured with respect to a question. That is, an event always has the same unexpectedness. However the amount of information it carries depends on the question considered.
- However surprise and information are related: Entropy of a variable , hence information, equals the expected value of surprise of the possible values of the random variable. And mutual information, i.e. expected information with respect to one variable, given the observation of the other, equals the expected change of surprise in the first variable, given the observation of the second one.

Control Question 22

Let E be an event with $p(E) = 0$. Then

1. $s(E) = 0$, since we are sure that E can not happen;
2. $s(E) = 1$, since $s(E)$ is maximal if $p(E) = 0$.

Answer

Surprise is infinite (and hence maximal) if $p(E) = 0$. So both answers are wrong.

Control Question 23

The degree of surprise $s(E)$

1. is monotone decreasing (as a function of $p(E)$);
2. is continuous;
3. can be zero;
4. is measured with respect to a specified question.

Answer

1. The greater the probability, the smaller the surprise, hence it is correct. One can also argue that the logarithm is monotone increasing and $\log x \leq 0$ if $0 < x \leq 1$.
2. Follows directly from the fact that the logarithm is a continuous function.
3. If $p(E) = 1$, there is no surprise left, and we indeed get $s(E) = 0$.
4. The degree of surprise is associated with an event and not, like information, to a specified question. It is important to understand this difference.

1.2.5 Probability as Information

Learning Objectives for Subsection 1.2.5

After studying this subsection you should understand

- that not only events or observations of values of random variables represent information, but that a probability distribution represents information too;
- how the Kullback-Leibler divergence measures in a special, but important case, the amount of information of a probability distribution.

We look at a choice situation S without probability. As we know, its uncertainty is measured by $h(|S|) = \log |S|$. But now we also understand, that we may consider the probabilistic choice situation (S, p_V) too, where $p_V = 1/|S|$ is the uniform distribution over S. We have $H(V) = h(|S|) = \log |S|$. The set of possible choices is all we know in a given situation.

But, now suppose, that some source of information tells us, that in fact, not all elements of S are equally probable, but that we rather have a probabilistic choice situation (S, P), where P represents some non uniform probability distribution over S. Let X be the random variable associated with this choice situation. We know that $H(X) < H(V)$. Thus, we really have new *information* with respect to the initial choice situation without probabilities, namely information in our sense, that changes the measure of

uncertainty. The amount of information contained in the probability distribution P, relative to the prior uniform distribution, is expressed by the change of entropy,

$$\begin{aligned} i(X/V) &= H(V) - H(X) \\ &= \log|S| + \sum_x p_X(x) \log p_X(x) \\ &= \sum_x p_X(x) \log(|S| p_X(x)) \\ &= K(P_X, P_V). \end{aligned}$$

So, the information content of P relative to the original uniform distribution equals the Kullback-Leibler divergence between the probability distribution P and the original uniform distribution. We notice that

$$0 \leq i(X/V) \leq \log|S|.$$

The first inequality holds since $H(X) \leq H(V) = \log|S|$ and the second one because $H(V) - H(X) \leq H(V)$, since $H(X) \geq 0$. So, the amount of information $i(X/V)$ is always positive, and the maximal amount $\log|S|$ is obtained, when the event $X = x$ is observed, that is $H(X|x) = 0$. In fact, the last case corresponds to a degenerate random variable Y with probability distribution $p_Y(x) = 1$, and $p_Y(y) = 1$, if $y \neq x$.

So, here we go one step further and consider not only events or values of random variables which are observed or reported as information, but more generally specifications of probability distributions over choice sets as well. This is in fact more general, also in the sense that an observed event may be represented by the conditional probability distribution it induces. So, in fact, if we start with the uniform random variable V over S and observe event $E \subseteq$S, then this amounts to considering the conditional random variable $V_E = V|E$, which has a uniform distribution over E,

$$p_{V|E}(x) = \left\{ \begin{array}{ll} \frac{1}{|E|} & \text{if } x \in E, \\ 0 & \text{otherwise.} \end{array} \right\}$$

Then we obtain

$$i(E/V) = i(V_E/V) = H(V) - H(V_E) = \log|S| - \log|E| = \log\frac{|S|}{|E|}.$$

Note that $|E|/|S| = p(E)$ and according to the last result,

$$i(E/V) = -\log p(E).$$

Here we see that the *surprise is* a measure of information, but a very particular one. It is the amount of information gained by an event E relative the prior information represented by the uniform random variable V, that is, the choice without probability.

We stress once more, that in this more general setting as well, the amount of an information is still to be measured relative to a) a *precisely specified question* and b) a specified *prior information.*

In order to emphasize these remarks we consider the case of compound choice situations. So let $S_1 \times S_2$ be a compound choice situation. We associate with it two uniform random variables V_1 over S_1 and V_2 over S_2. Then in the pair (V_1, V_2) with the uniform distribution over $S_1 \times S_2$, the two variables V_1 and V_2 are independent. Consider now

any pair of random variables (X, Y), associated with a probabilistic choice situation $(S_1 \times S_2, P)$. Then this represents an information relative to (V_1, V_2) of amount

$$i(X, Y/V_1, V_2) = \log|S_1| + \log|S_2| - H(X, Y).$$

We have (use theorem 1.3):

$$\begin{aligned} & i(X, Y/V_1, V_2) \\ & \quad \geq \quad (\log|S_1| - H(X)) + (\log|S_2| - H(Y)) \\ & \quad = \quad i(X/V_1) + i(Y/V_2). \end{aligned} \tag{1.34}$$

Equality holds exactly, if X and Y are independent. The probability distribution P over $S_1 \times S_2$ of course contains also information with respect only to S_1 (or S_2). In fact, knowing the distribution P gives an uncertainty of $H(X)$ with respect to S_1. Hence, we have

$$i(X, Y/V_1) = H(V_1) - H(X) = i(X/V_1).$$

Similarly we obtain $i(X, Y/V_2) = i(Y/V_2)$.

Example 1.35 (Symmetric Binary Channel - Continuation) Given a compound choice situation $S_1 \times S_2 = \{(0,0), (0,1), (1,0), (1,1)\}$ (symmetric binary channel). With it we associate two uniform random variables V_1 and V_2 as described above. This means that we know nothing about the channel and its input/output. But now consider the pair (X, Y) of random variables associated with the compound choice situation $(S_1 \times S_2, P)$ given in the examples 1.14 and 1.30. This is new information describing how the channel works. We have seen, that $H(X, Y) \approx 1.1909$ bit, $H(X) = 0.7219$ bit and $H(Y) = 0.8267$ bit. Hence

$$\begin{aligned} i(X, Y/V_1, V_2) &= \log|S_1| + \log|S_2| - H(X, Y) \\ &= 2\log 2 - H(X, Y) \approx 0.8091 \text{ bit}, \\ i(X/V_1) &= \log|S_1| - H(X) \approx 0.2781 \text{ bit} \\ i(Y/V_2) &= \log|S_2| - H(Y) \approx 0.1733 \text{ bit}. \end{aligned}$$

Note that $0.4514 \text{ bit} \approx i(X/V_1) + i(Y/V_2) < i(X, Y/V_1, V_2) \approx 0.8091$ bit.

A particular case arises, if one of the two random variables X or Y is degenerate. Suppose for example that $Y = y$. Then we have $p_{X,Y}(x, y') = p_X(x)$ if $y' = y$ and $p_{X,Y}(x, y') = 0$, if $y' \neq y$. In such a case we have

$$i(X, y/V) = \log|S_1| + \log|S_2| - H(X, y).$$

But

$$H(X, y) = -\sum_{x,y} p_{X,Y}(x, y) \log p_{X,Y}(x, y) = -\sum_{x} p_X(x) \log p_X(x) = H(X).$$

Thus we see that, in correspondence to (1.34)

$$i(X, y/V) = (\log|S_1| - H(X)) + \log|S_2| = i(X/V_1) + i(y/V_2)$$

If both variables X and Y are degenerate, that is $X = x$ and $Y = y$, then the last result implies that

$$i(x, y/V) = i(x/V_1) + i(y/V_2).$$

This is of course the same information as observing the values x and y for the possible choices in S_1 and S_2.

Another particular case is that only a probabilistic information relative to S_1 is known, whereas relative to S_2 we have a choice without probabilities. This situation can be represented by a pair of random variables X and Y such that

$$p_{X,Y}(x,y) = \frac{p_X(x)}{|S_2|},$$

such that X has the marginal probability distribution $p_X(x)$ and Y the uniform distribution $p_Y(y) = 1/|S_2|$, representing a choice without probabilities. Then we find that

$$\begin{aligned} i(X,Y/V) &= \log|S_1| + \log|S_2| - H(X,Y) \\ &= \log|S_1| + \log|S_2| - \sum_{x,y} \frac{p_X(x)}{|S_2|} log \frac{p_X(x)}{|S_2|} \\ &= \log|S_1| - \sum_x p_X(x) \log p_X(x) \\ &= i(X/V_1). \end{aligned}$$

So, if nothing is known about the choice in S_2, then the amount of information relative to the question V is the same as the information relative to V_1.

Example 1.36 (Probability as Information) Consider $S_1 = \{0,1\}$, $S_2 = \{0,1\}$ and the random variables X associated with S_1 and Y associated with S_2. Let $p_Y(0) = 0.1$ and $p_Y(1) = 0.9$. Relative to S_1 we have a choice without probabilities. Thus

$$\begin{aligned} p_{X,Y}(0,0) &= \frac{p_Y(0)}{|S_1|} = \frac{0.1}{2} = 0.05, \\ p_{X,Y}(0,1) &= \frac{p_Y(1)}{|S_1|} = \frac{0.9}{2} = 0.45, \\ p_{X,Y}(1,0) &= \frac{p_Y(0)}{|S_1|} = \frac{0.1}{2} = 0.05, \\ p_{X,Y}(1,1) &= \frac{p_Y(1)}{|S_1|} = \frac{0.9}{2} = 0.45, \end{aligned}$$

and, with $H(X,Y) = -0.05\log 0.05 - 0.45\log 0.45 - 0.05\log 0.05 - 0.45\log 0.45 \approx 1.469$ bit, we obtain

$$\begin{aligned} i(X,Y/V) &= \log|S_1| + \log|S_2| - H(X,Y) \\ &= 2\log 2 - H(X,Y) \approx 0.531 \text{ bit}. \end{aligned}$$

Since we have a choice without probabilites relative to S_1 we get that $H(X) = -0.5\log 0.5 - 0.5\log 0.5 = \log 2 = 1$ bit. Since $H(Y) = -0.1\log 0.1 - 0.9\log 0.9 \approx 0.469$ bit we finally obtain

$$\begin{aligned} i(X/V_1) &= H(V_1) - H(X) = \log|S_1| - H(X) = 1 - 1 = 0 \text{ bit}, \\ i(Y/V_2) &= H(V_2) - H(Y) = \log|S_2| - H(Y) \approx 1 - 0.469 \approx 0.531 \text{ bit}. \end{aligned}$$

Note that $i(Y/V_2) = i(X,Y/V)$.

To conclude we prove a theorem, which is to information what theorem 1.4 is to entropy. In order to formulate the theorem, we introduce a new concept. If X and Y are a pair of variables associated with the compound probabilistic choice situation $(S_1 \times S_2, P)$, then let Y_x denote the conditional random variable Y given that $X = x$. We may consider the amount of information carried by this variable, or rather its probability distribution, with respect to the question V_2,

$$i(Y_x/V_2) = \log|S_2| - H(Y|x).$$

It is the measure of the information contained in the pair of random variables (X, Y) and the observation $X = x$ with respect to V_2. We consider the expected value of this information

$$I(Y|X/V_2) = \sum_x p_X(x) i(Y_x/V_2) = \log|S_2| - H(Y|X)$$

Now we have the following theorem.

Theorem 1.11 *Let X and Y be a pair of random variables associated with the probabilistic choice situation $(S_1 \times S_2, P)$. Then*

$$i(X, Y/V) = i(X/V_1) + I(Y|X/V_2).$$

PROOF The proof is straightforward, using theorem 1.4,

$$\begin{aligned} & i(X, Y/V) \\ &= \log|S_1| + \log|S_2| - H(X, Y) \\ &= (\log|S_1| - H(X)) + (\log|S_2| - H(Y|X)) \\ &= i(X/V_1) + I(Y|X/V_2). \end{aligned}$$

This theorem says that the measure of information contained in X, Y relative to the choice situation $S_1 \times S_2$ without probabilities, is the sum of the information contained in X with respect to S_1 and the expected information of Y given an observation of X with respect to S_2.

Example 1.37 (Symmetric Binary Channel - Continuation) We continue example 1.35. Since $H(X|Y) \approx 0.4690$ (see example 1.14), we obtain

$$\begin{aligned} I(Y|X/V_2) &= \log|S_2| - H(Y|X) \\ &= \log 2 - 0.4690 \approx 0.5310 \text{ bit.} \end{aligned}$$

Thus 0.8091 bit $\approx i(X, Y/V) = i(X/V_1) + I(Y|X/V_2)$.

We have seen that both observations of random variables or events, as well as probability distributions represent information. In many cases it seemed that events could be replaced by probability distributions, i.e. the conditional distribution, to obtain the same amount of information. Also choice without probability turned out to be equivalent to probabilistic choice with a uniform probability distribution. So, finally, it seems

that probability distributions are the ultimate form of information. This however is a premature conclusion, as the following example shows.

Example 1.38 (Probability Distributions and Events as Information) Suppose a coin is thrown two times and it is reported that the result was not heads both times. How much information does this provide for the question whether the first throw was heads or tails?

If we know nothing about the coin, in particular, if we cannot assume that it is a fair coin, then we are faced with a choice situation without probability. In fact $S = \{(h,h),(h,t),(t,h),(t,t)\}$, if h stands for heads and t for tails. The observation is the event $E = \{(h,t),(t,h),(t,t)\}$ and the amount of information gained with respect to the original choice situation is $\log 4/3$ bit. The event E projected to the choice situation of the first throw is $\{h,t\}$, that is both results are equally possible. Hence, the information gained by E with respect to this question is 0 bit.

Assuming an uniform probability distribution over S means to assume that the coin is *fair*. Then, event E conditions the probability distribution to 1/3 for the three possible outcomes $\{(h,t),(t,h),(t,t)\}$. The marginal distribution with respect to the first throw assigns probability 1/3 to heads and 2/3 to tails. Thus, this time the information gained with respect to the first throw, is $\log 4 + 1/3 \log 1/3 + 2/3 \log 2/3$ bit, which is different from 0. But note that in this case we have used the *additional information* that the coin is *fair*.

So, probabilistic choice situations with uniform distributions and choice situations without probability are *not* always identical and equivalent. In general, we have to carefully distinguish between probability distributions as information and non-probabilistic information such as events.

This concludes our introductory discussion of the measure of information.

Summary for Section 1.2

- Probability distributions on choice situations represent information relative to the choice without probability. Its amount is measured, as usual, by the reduction of uncertainty that is achieved.
- Observed events or values also represent information which can be subsumed into the conditional probability distributions they induce, relative to the prior probability distribution.
- As before, in the case of compound choice situations, probability distributions carry information with respect to different questions, and in general the respective amounts are different. The relativity of information with respect to specified questions is still valid in this more general framework.

Control Question 24

Consider the probabilistic choice situation (S, p_V) with $p_V = 1/|S|$ and let X be a random variable associated with (S, P). Then $i(X/V)$

1. $= H(V) - H(X)$;
2. $= K(P_X, P_V \cdot P_X)$;
3. can be negative.

Answer

1. This is correct, since information is defined by the change of entropy.
2. The relation between the Kullback-Leibler divergence and the information content of P relative to the uniform distribution is given by $K(P_X, P_V)$, hence the assertion is false.
3. That is not possible. $i(X/V) \geq 0$ because $H(X) \leq H(V)$ if V is uniformly distributed.

Control Question 25

Let X and Y be a pair of random variables associated with $(S_1 \times S_2, P)$. Then, $i(X, Y/V) = i(X/V_1) + I(Y|X/V_2)$, if, and only if,

1. X and Y are independent;
2. X and Y have a uniform probability distribution;
3. $p_X = p_Y$.

Answer

All answers are wrong, since $i(X, Y/V) = i(X/V_1) + I(Y|X/V_2)$ holds for all random variables X and Y.

Summary for Chapter 1

- We have seen that *entropy* is used to measure the uncertainty in a *probabilistic choice situation* or in a *random variable*. Entropy is related to the game of (binary) questions and answers in the sense, that it indicates the expected number of questions to find out the actual element selected in the choice situation or the actual value of the random variable. Uncertainty is essentially measured in *bits*.
- *Information* is given by an observed event or an observed value of a random variable. A probability distribution specified on a choice situation also represents information relative to the situation without known probabilities. The *amount of information* is measured by the change in the uncertainty between the situation before the information is obtained and afterwards. Like uncertainty, information is measured in *bits*.
- The amount of information is *relative* to a *specified question*, which is indicated

or represented by a choice situation. It is also *relative* to the *prior information* which is given by the *prior* probability distribution. Information leads to a *posterior* probability distribution. The amount of information is the difference between the entropies of these two distributions.

- *Mutual information* between two random variables is the *expected* amount of information gained on one variable, if the other one can be observed. It is *symmetric*: the expected gain of information is the same for both variables.
- The *degree of surprise* of an event can also be measured. It turns out, that entropy is the expected degree of surprise in a choice situation. So surprise, uncertainty and information are closely linked, but *different* concepts.

OutLook

This module contains an elementary introduction into the basics of information theory. However there are still two questions that need to be looked at briefly:

1. What are the applications of this theory?
2. In what directions is this theory to be further developed?

The main application of information theory is in *coding theory*. This theory studies how to code most economically the signals or the information coming from some sources of specified characteristics. It also addresses the question of coding signals or information for transmission over noisy channels, where information is lost. The problem is to introduce enough redundancy into the code such that information loss can be compensated. These problems are studied in modules C2, C3 and C4.

But coding is only a relatively narrow aspect of information. There are many more questions related to information:

1. How is information processed, that is, how is information from different sources combined and focussed on the questions of interest? This relates to *information processing* in general, and on the computer in particular.
2. What does it mean that information is *uncertain*?
3. How is information represented and how can consequences be deduced or inferred from it? This brings *logic* and general *inference methods*, for example from *statistics* into the game.

These questions raise many issues of research. Subsequent modules treat these questions in some depth.

Chapter 2

Module C2: Efficient Coding of Information

by J.-C. Chappelier

Learning Objectives for Chapter 2

In this chapter we present:

1. the basics of coding a discrete information source for compressing its messages;
2. what conditions a code needs to meet in order to compress efficiently;
3. the fundamental limit for the compression of data;
4. and methods for producing efficient compression codes.

Introduction

In the preceding chapters, we introduced Shannon's measure of uncertainty and several of its properties. However, we have not yet experienced how useful this measure can be in practice. In this chapter, we exhibit the first practical problem which benefits from Shannon's measure. This problem is the problem of coding a discrete information source into a sequence of symbols. We also develop some efficient methods for performing such codings, and study under which conditions it can be done. In this framework, entropy appears to be the fundamental limit to data compression; i.e. related to the minimal coding expected length.

But what are the general reasons for *coding* after all? There are basically three different purposes:

- coding for compressing data, i.e. reducing (on average) the length of the messages. This means trying to remove as much redundancy in the messages as possible.

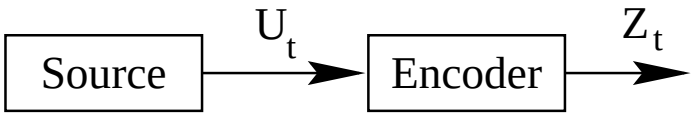


Figure 2.1: Basic schema for source coding: the source symbol U_t at time t is transformed into the corresponding codeword Z_t.

- coding for ensuring the quality of transmission in noisy conditions.
  This requires adding redundancy in order to be able to correct messages in the presence of noise.
- coding for secrecy: making the message impossible (or hard) to read for unauthorized readers. This requires making the access to the information content of the message hard.

This chapter focuses on the first aspect of coding: coding for *efficiency*.

The current chapter addresses the problem of coding a discrete information source. But what does all this mean? The following section answers that question. The next section then focuses on the efficiency of coding for compressing data. Finally, the last section provides an algorithm to actually construct an efficient code.

2.1 Coding a Single Random Variable

Learning Objectives for Section 2.1

After studying this section you should know:

1. what "*coding a discrete memoryless information source*" means;
2. what the general properties of a code are;
3. what the relation between codes and trees is;
4. under which conditions certain codes may exist.

An *information source** is a *message** generator, i.e. a generator of sequences of symbols. A *symbol* is simply an element of a set, called an *alphabet*. In this course, only finite alphabets will be addressed. When the alphabet is finite, the information source is said to be *discrete*; and the size of the alphabet is called the *arity** of the source. Finally, only messages of finite length will be considered.

For instance, a newspaper can be regarded as a discrete information source whose messages are the texts contained in the newspaper, the symbols simply being the letters of the usual alphabet (including the whitespace and other punctuation marks).

The messages from the source come then into a *encoder* which transforms them into a sequence of *codewords**. The basic schema of such a coding framework is given in figure 2.1.

A *codeword** is simply a (non empty) sequence of symbols taken into the coding alphabet, another alphabet used by the encoder. Coding therefore maps source symbols to

codewords, each of which may consist of several code symbols.

More formally, an *information source** U_t, and more generally U when the time index is not pertinent, is a random process on a given alphabet $\mathcal{V}_U$; i.e. a sequence of random variables on $\mathcal{V}_U$.

Each symbol as a probability $P(U_t = u_i)$ to be emitted by the source at time t. The source is said to be memoryless if the probability for a symbol u_i to be emitted does not depend on the past emitted values; i.e. if

$$\forall t \geq 1 \; \forall u_i \in \mathcal{V}_U \quad P(U_t = u_i | U_1 ... U_{t-1}) = P(U_t = u_i).$$

Furthermore, only *stationary sources**, i.e. sources for which $P(U_t = u_i)$ does not depend on t, are addressed in this chapter. In such a case, and when no ambiguity is possible, $P(U = u_i)$ will be denoted by p_i. We assume that $p_i \neq 0$ for all considered symbol u_i in the source alphabet; that is, we do not bother with those symbols of the source whose probability to occur is zero.

Regarding the coder, only the simplest case where one single codeword is associated to each source symbol, is considered. Technically speaking, the encoding process $Z := f(U)$ is a *mapping* from the source alphabet $\mathcal{V}_U$ to the set of codewords $\mathcal{V}_Z$.

Denoting $\mathcal{Z}^*$ the set of all finite length sequences of symbols from the code alphabet $\mathcal{Z}$, the set of codewords $\mathcal{V}_Z$ is a subset of $\mathcal{Z}^*$ not containing the empty string (i.e. the unique sequence of length 0).

Furthermore, we focus on such codes where different source symbols map to different codewords. Technically speaking, the mapping f is *injective*.
Such codes, where $Z = f(U)$ is an injective mapping, are called *non-singular codes**

Definition 2.1 (Non-singular Code) A code of a discrete information source is said to be *non-singular* when different source symbols maps to different codewords.
Formally, denoting z_i the codeword corresponding to source symbol u_i, we have:

$$u_i \neq u_j \implies z_i \neq z_j.$$

All the codes considered in the rest of this chapter are non-singular.

Since there is no reason to create codewords which are not used, i.e. which do not correspond to a source symbol, the mapping f from $\mathcal{V}_U$ to $\mathcal{V}_Z$ is *surjective*, and is therefore a *one-to-one mapping*.

Example 2.1 (Finite Source Encoding) A very common example of code is given by the Morse code. This code is used for coding letters. It uses essentially of two symbols: a dot (·) and a dash (–).[1]For instance, the letters `E`, `A` and `K` are respectively coded "·", "· –" and "– · –".

As we are interested in coding messages (i.e. sequences of symbols) and not only single symbols alone, we focus on codes such that each encoded message can be uniquely

[1]Actually four symbols are used in the Morse code: a letter space and a word space code are also used.

decoded. Such codes are called *non-ambiguous codes**A code for which any sequence of codewords uniquely corresponds to a single source message.

Definition 2.2 (Non-Ambiguous Codes) A code of a discrete source is said to be *non-ambiguous* if and only if each (finite length) sequence of codewords uniquely corresponds to a single source message. ♦

More formally, a code is said to be non-ambiguous if and only if the trivial extension $\hat{f}$ of the coding mapping f to the set of messages $\mathcal{V}_U^*$, taking its value into the set of finite length sequences of codewords $\mathcal{V}_Z^*$ ($\hat{f} : \mathcal{V}_U^* \to \mathcal{V}_Z^*$), is a one-to-one mapping.

Example 2.2 (Ambiguous Code) Consider the source consisting of the three symbols `a`, `b` and `c`. Its messages can be any sequence of these symbols; for instance "`aabca`" is a message of this source.

Then, the following encoding of this source:

`a` $\mapsto$ 1 `b` $\mapsto$ 00 `c` $\mapsto$ 11

is *ambiguous*.

For instance, there is no way, once encoded, to distinguish the message "`aaaa`" from the message "`cc`". Indeed, both are encoded as "1111".

Example 2.3 (Non-Ambiguous Code) Keeping the same source as in the previous example, consider now the following code:

`a` $\mapsto$ 1 `b` $\mapsto$ 00 `c` $\mapsto$ 10

It can be show that this code is *non-ambiguous*. For instance the sequence "10000" decodes into "`abb`" and the sequence "1000" into "`cb`".

2.1.1 Prefix-Free Codes

Among the class of non-ambiguous codes, certain are of some particular interest. These are the *prefix-free* codes. Before defining what such a code is, we need to introduce the notion of *prefix**.

A sequence z of length n ($n \geq 1$) is said to be a *prefix** of another sequence z' if and only if the first n symbols of z' form exactly the sequence z. For instance, `abba` is a prefix of `abbabc`. Note that any sequence is trivially a prefix of itself.

Definition 2.3 (Prefix-Free Code) A code of a discrete source is said to be *prefix-free* when no codeword is the prefix of another codeword.
More formally, a code Z, the alphabet of which is $\mathcal{Z}$ and the set of codewords of which is $\mathcal{V}_Z$, is said to be prefix-free if and only if

$$\forall z \in \mathcal{V}_Z \qquad \forall y \in \mathcal{Z}^* \qquad (zy \in \mathcal{V}_Z \implies y = \varepsilon)$$

(ε denoting the empty (i.e. 0-length) string). ♦

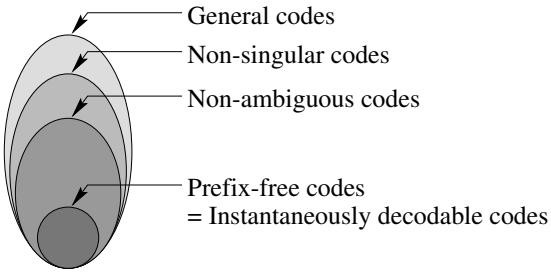


Figure 2.2: How the different types of codes are related.

Example 2.4 (Prefix-Free Code) Consider the source consisting of the three symbols `a`, `b` and `c`.

The following encoding of this source:

$$\texttt{a} \mapsto 0 \qquad \texttt{b} \mapsto 10 \qquad \texttt{c} \mapsto 11$$

is *prefix-free*.

On the other hand, the following encoding:

$$\texttt{a} \mapsto 1 \qquad \texttt{b} \mapsto 00 \qquad \texttt{c} \mapsto 10$$

is not prefix-free as 1 (the codeword for `a`) is a prefix of 10 (the codeword for `c`).

Why focusing on prefix-free codes? The answer lies in the following next two properties (2.1 and 2.2), which emphasize their interest.

Property 2.1 *Any prefix-free code is non-ambiguous.*

It is important to notice however that there exists some non-ambiguous code which are not prefix-free as the example 2.3 shows.

Let us now come to the second interesting property of prefix-free codes.

Definition 2.4 A code is said to be *instantaneously decodable** if and only if each codeword in any string of codewords can be decoded as soon as its end is reached. ♦

Property 2.2 *A code is instantaneously decodable if and only if it is prefix-free.*

This definition ensures that there is no need to memorize the past codewords nor to wait for the following ones to achieve the decoding. Such a code saves both time and space in the decoding process of a encoded message.

We up to now have encountered many different types of codes: non-singular, non-ambiguous, prefix-free. How these different types are related one to another is summarized in figure 2.2.

Control Question 26

Consider some information source U, the symbols of which are $u_1 = 1$, $u_2 = 2$, $u_3 = 3$, and $u_4 = 4$, with the following probability distribution:

u_i	1	2	3	4
$P(U = u_i)$	0.5	0.25	0.125	0.125

Consider then the following encoding of it (where z_i is the codeword for u_i):

z_1	z_2	z_3	z_4
0	10	110	111

1. Is this code non ambiguous?
2. Encode the message 1423312.
3. Decode the sequence `1001101010`.

Answer

1) In this code no codeword is prefix of another codeword. Therefore this code is *prefix-free*. Since any prefix-free code is non-ambiguous, this code **is non-ambiguous**.

2) 1423312 → `011110110110010`

3) `1001101010` → 21322

Control Question 27

Are these codes prefix-free? non-ambiguous? Instantaneously decodable?

a. z_1=`00`, z_2=`10`, z_3=`01`, z_4=`11`

b. z_1=`0`, z_2=`1`, z_3=`01`

c. z_1=`1`, z_2=`101`

Answer

First of all, any prefix-free code is instantaneously decodable and, conversely, any instantaneously decodable is prefix-free. Therefore the answer to the first question is the same to the answer to the last one.

The code defined in **a-** is prefix-free: indeed no codeword is prefix of another one. It is therefore also a non-ambiguous code.

The code defined in **b-** is not prefix-free as z_1 is prefix of z_3. It is furthermore ambiguous as, for instance, 01 corresponds to both sequences "$z_1\, z_2$" and "z_3".

The code defined in **c-** is <u>not</u> prefix-free since z_1 is prefix of z_2. However, this code <u>is</u> non ambiguous.

Indeed, the only way to get 0 is with z_2. Therefore, when receiving a 1, waiting for one bit more (or the end of the message) can decode the message: if the next bit is 1

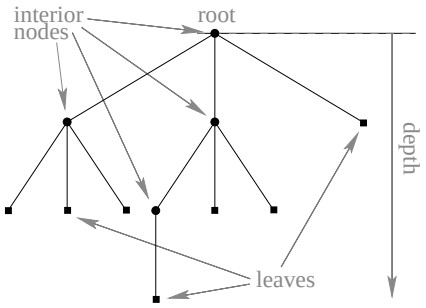


Figure 2.3: Summary of the terms related to trees.

then we decode into $u_1\ u_1$; and if we received a 0 instead, we decode as u_2, swallowing meanwhile the next digit which is surely a 1.

For instance, `1011111011011` is to be understood as:

$$z_2\ z_1\ z_1\ z_1\ z_2\ z_2\ z_1$$

and decoded into

$$u_2\ u_1\ u_1\ u_1\ u_2\ u_2\ u_1.$$

Notice: *This is an example of a code which is non-ambiguous although it is not prefix-free.*

2.1.2 n-ary Trees for Coding

In order to study more deeply the properties of prefix-free codes, we now need to introduce some more definitions and formulate some theorems. Among them, the most useful tool for the study of prefix-free codes are *n-ary trees*.

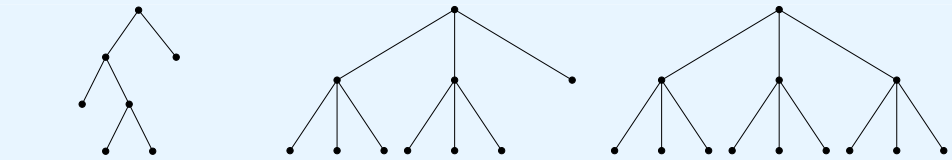
Let us first very briefly summarize the concept of tree and related terms (cf figure 2.3). A tree is basically a graph (nodes and arcs) which begins at a *root* node (simply "the root"). Each node in the graph is either a *leaf* or an *interior node*.[2] An interior node has one or more *child* nodes and is called the *parent* of its child nodes. The *arity** of a node is the number of its child nodes. A *leaf* node is a node without child, i.e. a node of arity 0.

As opposed to real-life tree, the root is usually depicted at the top of the figure, and the leaves are depicted at the bottom. The *depth* of a node in a tree the number of arcs to go from the root to this node. By convention the depth of the root is null. The depth of a tree is the maximum depth of its leaves, i.e. the maximum number of arcs to go from the root to a leaf. Finally, a node n_1 is said to *cover* another node n_2 if the path from the root to n_2 contains n_1. Notice that a node covers at least itself.

Definition 2.5 (n-ary tree) A *n-ary tree* ($n \geq 1$) is a tree in which each interior node has arity n, i.e. has exactly n child nodes.
A full n-ary tree is a n-ary tree in which all the leafs have the same depth. ♦

[2]Notice that by this definition, the root is also an interior node.

Example 2.5 (*n*-ary tree)

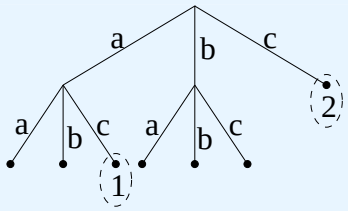


Binary tree ($n = 2$) Ternary tree ($n = 3$) Full Ternary tree

Property 2.3 *In the full n-ary tree of depth $d \geq 0$, each node at depth δ ($0 \leq \delta \leq d$) covers exactly $n^{d-\delta}$ leaves.*

Definition 2.6 (Coding Tree) A *coding tree* is a n-ary tree, the arcs of which are labeled with letters of a given alphabet of size n, in such a way that each letter appears at most once out of a given node.
The codewords defined by such a tree correspond to sequences of labels along paths from the root to a leaf. ♦

Example 2.6 (Coding Tree)



A ternary coding tree: the codeword represented by leaf 1 is "ac" and leaf 2 represents the codeword "c".

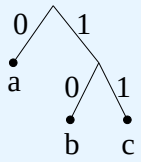
Definition 2.7 (*n*-ary code) A code with an alphabet of size n is called a *n-ary code.* ♦

Property 2.4 *For every n-ary prefix-free code, there exists at least one n-ary coding tree such that each codeword corresponds to the sequence of labels of an unique path from the root to a leaf.*
Conversely, every coding tree defines a prefix-free code. The codewords of this prefix-free code are defined as the sequences of labels of each path from the root to each leaf of the coding tree.

Shortly speaking, prefix-free codes and coding trees are equivalent.

Example 2.7 The coding tree corresponding to the prefix-free code of example 2.4

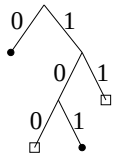
$(\{0, 10, 11\})$ is



As a standard display convention, the leaves are labeled by the source symbol, the codeword of which is the path from the root.

Notice that when representing a (prefix-free) code by a tree, it can occur that some leaves do not correspond to any codeword. Such leaves are called "*unused leaves*".

For instance, the binary coding tree corresponding to the code { 0, 101 } has two unused leaves as shown on the right.



Indeed, neither 11 nor 100 correspond to codewords. They are useless.

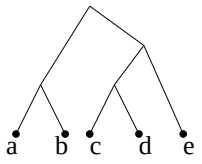
Definition 2.8 (Complete Code) When there is no unused leaf in the corresponding n-ary coding tree, the code is said to be a *complete code**. ♦

Control Question 28

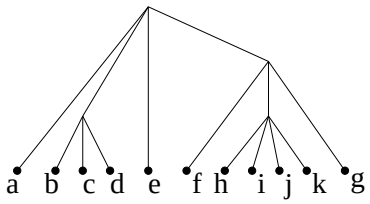
For all the trees below, with the convention that all the branches out of a same node are labeled with different labels (not displayed here), tell if it is a coding tree or not; and, in the case it is a coding tree, tell

1. what is its arity,
2. if the corresponding code is complete,
3. the length of the codeword associated to message "c".

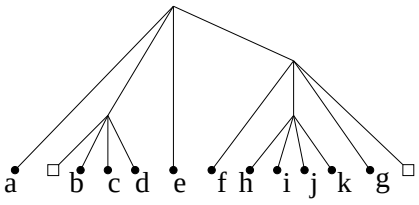
1)



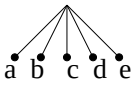
2)



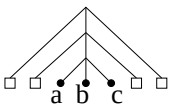
3)



4)



5)



Answer

1) yes, arity=2, complete, length for codeword coding `c` is 3
2) no, this tree is not a coding tree since the arity of its nodes is not constant (sometimes 3, sometimes 4)
3) yes, arity=4, not complete, length for codeword coding `c` is 2
4) yes, arity=5, complete, length for codeword coding `c` is 1
5) yes, arity=3, not complete, length for codeword coding `c` is 3

2.1.3 Kraft Inequality

We are now looking at the conditions that must hold for a prefix-free code to exist. The "Kraft inequality" appears as a necessary and sufficient condition.

Theorem 2.1 (Kraft Inequality) *There exists a D-ary prefix-free code of N codewords and whose codeword lengths are the positive integers $l_1, l_2, \ldots, l_N$ if and only if*

$$\sum_{i=1}^{N} D^{-l_i} \leq 1. \tag{2.1}$$

When (2.1) is satisfied with equality, the corresponding prefix-free code is complete.

Example 2.8 For the binary ($D = 2$) complete prefix-free code of example 2.4 ($\{0, 10, 11\}$), the sum $\sum_{i=1}^{N} D^{-l_i}$ is $2^{-1} + 2^{-2} + 2^{-2}$, i.e. $\frac{1}{2} + \frac{1}{4} + \frac{1}{4}$ which is indeed equal to 1.

Similarly, Kraft inequality tells us that there exists at least one ternary prefix-free code whose codeword lengths are 1, 2, 2 and 4. Indeed

$$3^{-1} + 3^{-2} + 3^{-2} + 3^{-4} = \frac{46}{81} \simeq 0.57 < 1.$$

Such a code would not be complete.

e-pendix: Kraft Inequality

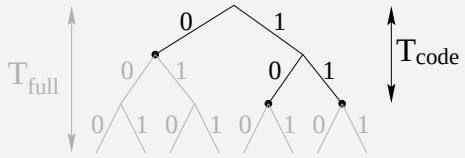
Warning! A classical pitfall to avoid with this theorem is the following: the theorem only tells us *when* a prefix-free code may exists, but it does not at all answers the question if a given code (with such and such lengths for its codewords) is indeed prefix-free.

For instance, the first code given in example 2.4 ($\{1, 00, 10\}$) is not prefix-free. However the corresponding sum $\sum D^{-l_i}$ is $2^{-1} + 2^{-2} + 2^{-2} = 1$. The pitfall to avoid is that Theorem 2.1 does not tell us that this code is prefix-free, **but** that there exists a prefix-free code with the same codeword lengths. Indeed such a code is given in the second part of example 2.4 ($\{0, 10, 11\}$).

Let us now proceed with the proof of Theorem 2.1.

PROOF

$\boxed{\Longrightarrow}$ Suppose first that there does exist a D-ary prefix-free code whose codeword lengths are $l_1, l_2, \ldots, l_N$. Let $L := \max_i l_i + 1$. Consider constructing the corresponding coding tree T_{code} by pruning[3] the *full* D-ary tree of depth L, T_{full}, at all nodes corresponding to codewords.



Because of the prefix-free condition, no node corresponding to a codeword can be below another node corresponding to another codeword. Therefore each node corresponding to a codeword prunes its own subtree. Looking at the i^{th} codeword and applying property 2.3 to l_i (which is $< L$), T_{code} has, for this node only, D^{L-l_i} leaves less than T_{full}.

Considering now the whole code, T_{code} has $\sum_{i=1}^{N} D^{L-l_i} = D^L \left(\sum_{i=1}^{N} D^{-l_i}\right)$ leaves less than T_{full}.

However at most D^L leaves can be removed since T_{full} has precisely D^L leaves. Therefore

$$D^L \left(\sum_{i=1}^{N} D^{-l_i}\right) \leq D^L,$$

i.e.

$$\sum_{i=1}^{N} D^{-l_i} \leq 1.$$

Furthermore, in the case where the considered code is complete, all the leaves correspond to a codeword; therefore all the corresponding subtrees in T_{full} have been "removed", and therefore all the D^L leaves of T_{full} have been "removed". This means that $\sum_{i=1}^{N} D^{L-l_i} = D^L$, i.e. $\sum_{i=1}^{N} D^{-l_i} = 1$.

$\boxed{\Longleftarrow}$ Conversely, suppose that $l_1, l_2, \ldots, l_N$ are positive integers such that (2.1) is satisfied. Let L be the largest of these numbers $L := \max_i l_i$, and n_j be the number of these l_i that are equal to j $(1 \leq j \leq L)$.

Inequality (2.1) can then be written as $\sum_{j=1}^{L} n_j\, D^{-j} \leq 1$, i.e. $n_L \leq D^L - \sum_{j=1}^{L-1} n_j\, D^{L-j}$.

Since $n_L \geq 0$, we have:

$$D\, n_{L-1} \leq D^L - \sum_{j=1}^{L-2} n_j\, D^{L-j},$$

i.e.

$$n_{L-1} \leq D^{L-1} - \sum_{j=1}^{L-2} n_j\, D^{L-j-1}.$$

And since all the n_j are non-negative, we successively get, for all $0 \leq k \leq L-1$

$$n_{L-k} \leq D^{L-k} - \sum_{j=1}^{L-k-1} n_j \, D^{L-j-k}.$$

These inequalities constitute the key point for constructing a code with codeword lengths $l_1, l_2, \ldots, l_N$:

1. start with a single node (the root)
2. for all k from 0 to L do
   (a) assign each codeword such that $l_i = k$ to a node of the current depth (k). These n_k nodes becomes therefore leaves of the coding tree.
   (b) extend all the remaining nodes of current depth with D child nodes.

Doing so, the number of nodes which are extended at step (2b) is $D^k - \sum_{j \leq k} n_j D^{k-j}$ leading to $D^{k+1} - \sum_{j \leq k} n_j D^{k+1-j}$ new nodes for next step. Because of the former inequalities, this number is greater than n_{k+1}, leaving therefore enough nodes for next step (2a).

The algorithm can therefore always assign nodes to codewords; and finally construct the whole coding tree for the code.

Therefore, if the l_i satisfy inequality (2.1), we are able to construct a prefix-free code with the corresponding codeword lengths.

Furthermore, in the case where $\sum_i D^{-l_i} = 1$, the number of nodes remaining after step (2a) when $j = L$ is

$$D^L - \sum_{j \leq L} n_j D^{L-j} = D^L - \sum_{i=1}^{N} D^{L-l_i} = D^L (1 - \sum_i D^{-l_i}) = 0,$$

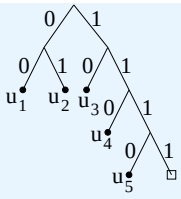
which means that all the nodes have been affected to a codeword, i.e. that the code is complete. ■

Note that this proof of the Kraft inequality actually contains an effective algorithm for constructing a D-ary prefix-free code given the codeword lengths (whenever such a code exists).

Example 2.9 Does a binary prefix-free code with codeword lengths $l_1 = 2$, $l_2 = 2$, $l_3 = 2$, $l_4 = 3$, and $l_5 = 4$ exist?

The answer is "yes" since $\sum_{i=1}^{5} 2^{-l_i} = 1/4 + 1/4 + 1/4 + 1/8 + 1/16 = 15/16 < 1$. An example of such a code can be:

[3] i.e. removing the whole subtree at given node



U	u_1	u_2	u_3	u_4	u_5
Z	00	01	10	110	1110

Example 2.10 Does a binary prefix-free code with codeword lengths 1, twice 2, 3, and 4 exist?

The answer is "no", since $\sum_{i=1}^{5} 2^{-l_i} = 1/2 + 1/4 + 1/4 + 1/8 + 1/16 = 19/16 > 1$.

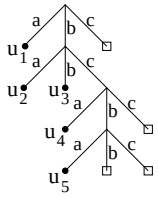
Control Question 29

Does there exists a ternary prefix-free code with codeword lengths 1, 2, 2, 3 and 4?

Answer

Since $\sum_{i=1}^{5} 3^{-l_i} = 1/3 + 1/9 + 1/9 + 1/27 + 1/81 = 49/81 < 1$, we know that such a prefix-free code exists.

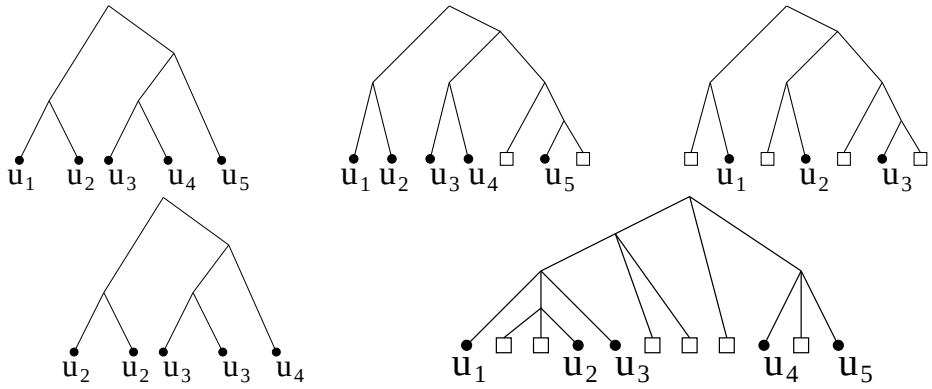
Here is one example, as a matter of illustration:



U	u_1	u_2	u_3	u_4	u_5
Z	a	ba	bb	bca	bcba

Control Question 30

Which of the following trees (maybe several) has for its codeword lengths: 2, 2, 3, 3 and 4.



Answer

Trees 2 and 5. Notice that tree 4 is even not a conding tree, since in a coding tree one message must correspond to one leaf and one leaf only.

Summary for Chapter 2

Codes: prefix-free $\Longrightarrow$ non-ambiguous $\Longrightarrow$ non-singular

Prefix-Free Codes:
- no codeword is prefix of another
- equivalent to instantaneously decodable codes
- equivalent to coding trees

Kraft Inequality: $\exists$ prefix-free D-ary code $\Longleftrightarrow$ $\sum_i D^{-l_i} \leq 1$

2.2 Efficient Coding

Learning Objectives for Section 2.2

In this section, you should:

1. understand what "*efficient*" means for a compression code;
2. learn how prefix-free codes and probabilized n-ary trees are related;
3. learn what the universal bound on "efficiency" is for memoryless source coding;
4. see an example of efficient codes.

2.2.1 What Are Efficient Codes?

It is now time to start addressing the question of original interest, namely coding for efficiency. Our goal is to code the information source so as to minimize the average code length; i.e. the average length of a sequence of codewords.

Provided that the source has certain general properties (which almost often hold) minimizing the average code length is equivalent to minimizing the *expected code length**.

Definition 2.9 (Expected code length) Formally, recalling that source symbol u_i $(1 \leq i \leq N)$ has a probability p_i to be emitted, and denoting l_i the length of the corresponding codeword, the *expected code length** $E[L]$ is the expected value of the length of a codeword, i.e.

$$E[L] = \sum_{i=1}^{N} p_i \, l_i \tag{2.2}$$

When precision is required, the expected length of the code Z will be denoted by $E[L_Z]$.

We are therefore looking for (prefix-free) codes such that $E[L]$ is as small as possible.

From (2.2), it is obvious that we should assign the shorter codewords to the most probable values of U. Indeed, if $p_i > p_j$ and $l \geq l'$ then $p_i \, l + p_j \, l' \geq p_i \, l' + p_j \, l$.

But how do we know what codeword lengths to use? And what is the smallest $E[L]$ that can be targeted?

We will address these questions shortly, but we first have to look more accurately at the properties of coding trees within this perspective.

Control Question 31

Consider some information source U, the symbols of which are $u_1 = 1$, $u_2 = 2$, $u_3 = 3$, and $u_4 = 4$, with the following probability distribution:

u_i	1	2	3	4	5
$P(U = u_i)$	0.125	0.3	0.125	0.25	0.2

Consider then the following encoding of it (where z_i is the codeword for u_i):

z_1	z_2	z_3	z_4	z_5
1110	110	10	1111	0

What is the expected code length?

Answer

By definition the expected code length is the expected value of the length of the codewords, i.e., denoting l_i the length of z_i:

$$E[L] = \sum_{i=1}^{4} p(Z = z_i) \cdot l_i = \sum_{i=1}^{4} p(U = u_i) \cdot l_i = 0.125 \cdot 4 + 0.3 \cdot 3 + 0.125 \cdot 2 + 0.25 \cdot 4 + 0.2 \cdot 1 = 2.85$$

2.2.2 Probabilized n-ary Trees: Path Length and Uncertainty

Recall that a prefix-free code defines a n-ary tree in which each codeword corresponds to a leaf (through its path from the root). On the other hand, the probability distribution of the source to be coded assigns probabilities to the codewords, and hence to the leaves of the corresponding n-ary tree. By convention, a probability 0 is assigned to any unused leaf (i.e. that does not correspond to a codeword).

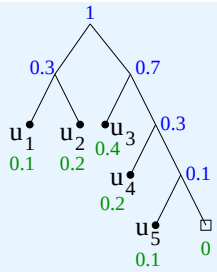
The probability assignment can further be extended to interior nodes by recursively assigning them, from the leaves to the root, a probability equal to the sum of the probabilities of the child nodes.

Doing so we create a *probabilized n-ary tree*.

Definition 2.10 (Probabilized n-ary tree) A probabilized n-ary tree is a n-ary tree with nonnegative numbers between 0 and 1 ("probabilities") assigned to each node (including the leaves) in such a way that:

1. the root is assigned with a probability 1, and
2. the probability of every node (including the root) is the sum of the probabilities of its child nodes.

Example 2.11 Taking $p_1 = p_5 = 0.1$, $p_2 = p_4 = 0.2$ and $p_3 = 0.4$ for the binary prefix-free code in example 2.9, page 92, results in the following binary tree with probabilities:



Notice that, in a probabilized n-ary tree, the sum of the probabilities of the leaves must be one.

Lemma 2.1 (Path Length Lemma) *In a probabilized n-ary tree, the average depth of the leaves is equal to the sum of the probabilities of the interior nodes (i.e. excluding the leaves but including the root).*

PROOF The probability of each node is equal to the sum of the probabilities of the leaves of the subtree stemming out from that node. Therefore the sum of the probabilities of interior nodes is a sum over leaf probabilities.

Furthermore, a leaf probability appears in this sum exactly as many times as the depth d of the corresponding leaf. Indeed, a leaf at depth d is covered by exactly d interior nodes: all these nodes that are on the path from the root to that leaf.

Thus, the sum of the probabilities of all the interior nodes equals the sum of the products of each leaf probability and its depth. This latter sum is precisely the definition of the average depth of the leaves.

More formally, let ν_i, $1 \leq i \leq M$ be the M interior nodes and λ_j, $1 \leq j \leq N$ be the N leaves. Let furthermore P_i be the probability of interior node ν_i and p_j the probability of leaf λ_j. Finally, let $\delta(\lambda_j)$ be the depth of leaf λ_j and let us denote by $\nu_i \geq \lambda_j$ the fact that interior node ν_i covers the leaf λ_j.

Then the the sum of the probabilities of interior nodes is equal to:

$$\sum_{i=1}^{M} P_i = \sum_{i=1}^{M} \sum_{j:\nu_i \geq \lambda_j} p_j = \sum_{j=1}^{N} \sum_{i:\nu_i \geq \lambda_j} p_j = \sum_{j=1}^{N} p_j \left(\sum_{i:\nu_i \geq \lambda_j} 1 \right).$$

Moreover, $\sum_{i:\nu_i \geq \lambda_j} 1$ in nothing but the number of interior nodes covering leaf λ_j. Therefore

$$\sum_{i:\nu_i \geq \lambda_j} 1 = \delta(\lambda_j)$$

and

$$\sum_{i=1}^{M} P_i = \sum_{j=1}^{N} p_j \, \delta(\lambda_j) =: E\left[\delta\right].$$

Example 2.12 (Path Length Lemma) In the preceding example, the average

depth of the leaves is $1 + 0.3 + 0.7 + 0.3 + 0.1 = 2.4$ by the Path Length Lemma. As a check, note that (definition of the expected code length) $2 \cdot 0.1 + 2 \cdot 0.2 + 2 \cdot 0.4 + 3 \cdot 0.2 + 4 \cdot 0.1 = 2.4$.

We now consider some entropy measures on a probabilized n-ary tree.

Definition 2.11 (Leaf Entropy of a Probabilized n-ary Tree) Let N be the number of leaves of a probabilized n-ary tree and $p_1, p_2, \ldots, p_N$ their probabilities.
The leaf entropy of such a tree is defined as

$$H_{\text{leaf}} = -\sum_i p_i \log p_i \tag{2.3}$$

Property 2.5 *For the probabilized n-ary tree corresponding to the prefix-free coding tree of an information source U, we have:*

$$H_{leaf} = H(U) \tag{2.4}$$

PROOF Let Z be the prefix-free code under consideration. By definition (of a coding tree), p_i is the probability of the i^{th} codeword, and therefore $H_{\text{leaf}} = H(Z)$.

Furthermore, since the code is non-singular ($Z = f(U)$ is injective), $H(Z) = H(U)$.

Therefore $H_{\text{leaf}} = H(U)$. ■

Definition 2.12 Let M be the number of interior nodes of a probabilized n-ary tree and $P_1, P_2, \ldots, P_M$ their probabilities. Let furthermore $q_{i1}, q_{i2}, \ldots, q_{in_i}$ be the probabilities of the n_i child nodes (including leaves) of the interior node whose probability is P_i. The branching entropy H_i at this node is then defined as

$$H_i = -\sum_{j=1}^{n_i} \frac{q_{ij}}{P_i} \log \frac{q_{ij}}{P_i}, \tag{2.5}$$

Notice that, because of the second property of the definition of a probabilized n-ary tree (definition 2.10, page 96), we have

$$P_i = \sum_{j=1}^{n_i} q_{ij}.$$

Example 2.13 Suppose that the $M = 5$ nodes for the tree of examples 2.9, page 92, and 2.11, page 2.11, are numbered in such a way that $P_1 = 1$, $P_2 = 0.3$, $P_3 = 0.7$, $P_4 = 0.3$ and $P_5 = 0.1$.

Then

$$H_{\text{leaf}} = -\sum_{i=1}^{5} p_i \log p_i \simeq 2.122 \text{ bit}.$$

We have $n_1 = 2$ and $q_{11} = 0.3$ and $q_{12} = 0.7$, thus

$$H_1 = -0.3 \log 0.3 - 0.7 \log 0.7 \simeq 0.881 \text{ bit.}$$

Similarly, $n_2 = 2$ and $q_{21} = 0.1, q_{22} = 0.2$, thus

$$H_2 = -\frac{0.1}{0.3} \log \frac{0.1}{0.3} - \frac{0.2}{0.3} \log \frac{0.2}{0.3} \simeq 0.918 \text{ bit.}$$

It is left as an exercise to show that $H_3 \simeq 0.985$ bit, $H_4 \simeq 0.918$ bit, $H_5 = 0$.

Theorem 2.2 (Leaf Entropy Theorem) *The leaf entropy of a probabilized n-ary tree equals the sum over all interior nodes (including the root) of the branching entropy of that node weighted by its probability. Using the above defined notations:*

$$H_{leaf} = \sum_{i=1}^{M} P_i \, H_i \tag{2.6}$$

Example 2.14 Continuing Example 2.13, we calculate H_{leaf} by (2.6) to obtain

$$\begin{aligned} H_{leaf} &= 1 \cdot H_1 + 0.3 \cdot H_2 + 0.7 \cdot H_3 + 0.3 \cdot H_4 + 0.1 \cdot H_5 \\ &\simeq 0.881 + 0.3 \cdot 0.918 + 0.7 \cdot 0.985 + 0.3 \cdot 0.918 + 0 \text{ bit} \\ &\simeq 2.122 \text{ bit.} \end{aligned}$$

in agreement with the direct calculation made in example 2.13.

Theorem 2.3 *For any two prefix-free codes of the same information source, the code which has the shortest expected code length has the highest symbol entropy rate.*

Shortly speaking, compressing the data increases the symbol entropy.

2.2.3 Noiseless Coding Theorem

We now use the results of the previous sections to obtain a fundamental lower bound on the expected code length of a prefix-free code of some information source.

Lower Bound on the Expected Codeword Length for Prefix-Free Codes

Theorem 2.4 (Shannon Coding Theorem, Part 1) *For any discrete memoryless information source of entropy $H(U)$, the expected code length $E[L]$ of* any *D-ary prefix-free code for this source satisfies:*

$$E[L] \geq \frac{H(U)}{\log D}, \tag{2.7}$$

The bound (2.7) could possibly have been anticipated on intuitive grounds. It takes $H(U)$ bits of information to specify the value of U. But each D-ary digit of the codeword can, according to Theorem 1.2 and mutual information definition (equation (1.28)), provide at most $\log D$ bits of information about U. Thus, we surely will need at least $H(U)/\log D$ code digits, on the average, to specify U.

Control Question 32

Consider some information source U, the symbols of which are $u_1 = 1$, $u_2 = 2$, $u_3 = 3$, and $u_4 = 4$, with the following probability distribution:

u_i	1	2	3	4
$P(U = u_i)$	0.5	0.25	0.125	0.125

Consider then the following encoding of it (where z_i is the codeword for u_i):

z_1	z_2	z_3	z_4
0	10	110	111

1. What is the expected code length?
2. Is the code considered an efficient code, i.e. optimal from the expected code length point of view?

Answer

By definition the expected code length is the expected value of the length of the codewords, i.e., denoting l_i the length of z_i:

$$E[L] = \sum_{i=1}^{4} p(Z = z_i) \cdot l_i = \sum_{i=1}^{4} p(U = u_i) \cdot l_i = 0.5 \cdot 1 + 0.25 \cdot 2 + 0.125 \cdot 3 + 0.125 \cdot 3 = 1.75$$

Let us compare the expected code length of that code to the source entropy: it can easily be computed that $H(U) = 1.75$ bit.

We know, by the part 1 of Shannon Noiseless Coding Theorem, that for any prefix-free code Z' of U we must have $H(U) \leq E[L']$ (here $\log D = 1$ since we have a binary code ($D = 2$)). Therefore, since we have $E[L] = H(U)$, for any prefix-free code Z' of U we have $E[L] \geq E[L']$. This means that, from the point of view of expected code length, **the proposed code is optimal**: there cannot exists another prefix-free code with a strictly shorter expected code length.

The above theorem is our first instance where the answer to a technical question turns out to be naturally expressed in terms of Shannon's entropy. However, this is not a full justification yet of the use of entropy, since only a *lower bound* has been specified. For instance, the value "1" is trivially another valid lower bound for the expected code length, but we would not claim this bound as a justification for anything! Only when the given lower bound is, in some sense, the best possible one, it can be used as a justification. To show that the bound expressed in the above theorem is indeed the best possible one, we need to show that there exist some codes whose expected code length can be arbitrarily close to it.

Shannon-Fano Prefix-Free Codes

We now show how to construct "efficient" prefix-free codes, although non-optimum in general, but close enough to the lower bound on the expected code length.

The key idea is to use as codeword for u_i, a code whose length is

$$l_i = \left\lceil -\frac{\log p_i}{\log D} \right\rceil,$$

where $\lceil x \rceil$ denotes for any x the only integer such that $x \leq \lceil x \rceil < x + 1$.

Such a code is called a *Shannon-Fano code* since the technique is implicit in Shannon's 1948 paper but was first made explicit by Fano.

But does such a prefix-free code always exists? The answer is "yes" due to the Kraft inequality.

Indeed, since by definition $l_i \geq -\frac{\log p_i}{\log D}$, we have

$$\sum_i D^{-l_i} \leq \sum_i D^{\frac{\log p_i}{\log D}} = \sum_i D^{\log_D p_i} = \sum_i p_i = 1.$$

Let us now measure how "good" such a code is in terms of its expected code length. By definition of l_i we have:

$$l_i < -\frac{\log p_i}{\log D} + 1. \tag{2.8}$$

Multiplying both side by p_i and summing over i gives:

$$\sum_i p_i \, l_i < \frac{-\sum_i p_i \log p_i}{\log D} + \sum_i p_i, \tag{2.9}$$

i.e.

$$E[L] < \frac{H(U)}{\log D} + 1. \tag{2.10}$$

We see that the Shannon-Fano code has an expected code length which is within one symbol of the lower bound (2.7) satisfied by *all* prefix-free codes. This code is thus quite good. Indeed, we know, from the first part of the coding theorem previously seen, that no prefix-free code can beat the expected code length of the Shannon-Fano code by more than one symbol. Therefore, when the entropy of the encoded source $H(U)$ is large, Shannon-Fano coding is nearly optimal. But when $H(U)$ is small, we can generally do much better than Shannon-Fano coding, as discussed in the next section.

Let us now foccus on the second part of Shannon's First Coding Theorem.

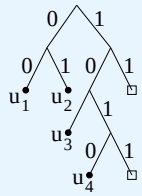
Theorem 2.5 (Shannon Coding Theorem, Part 2) *For any discrete memoryless information source of entropy $H(U)$, there exists at least one D-ary prefix-free code of it whose expected code length $E[L]$ satisfies:*

$$E[L] < \frac{H(U)}{\log D} + 1. \tag{2.11}$$

Example 2.15 Consider binary ($D = 2$) Shannon-Fano coding for the 4-ary source U for which $p_1 = 0.4$, $p_2 = 0.3$, $p_3 = 0.2$ and $p_4 = 0.1$. Such a coding will have as codeword lengths (since $\log_2(2) = 1$)

$$l_1 = -\lceil \log_2 0.4 \rceil = 2, \qquad l_2 = -\lceil \log_2 0.3 \rceil = 2,$$
$$l_3 = -\lceil \log_2 0.2 \rceil = 3, \text{ and} \qquad l_4 = -\lceil \log_2 0.1 \rceil = 4.$$

We then construct the code by the algorithm given in the proof of the Kraft inequality, page 92, to obtain the code whose binary tree is



By the Path Length Lemma, we have:

$$E[L] = 1 + 0.7 + 0.3 + 0.3 + 0.1 = 2.4,$$

and a direct calculation gives:

$$H(U) = 0.4 \log 0.4 + 0.3 \log 0.3 + 0.2 \log 0.2 + 0.1 \log 0.1 \simeq 1.8 \text{ bit}.$$

We see indeed that (2.11) is satisfied.

Notice, however, that this code is clearly non-optimal. Had we simply used the 4 possible codewords of length 2, we would have had a shorter code ($E[L] = 2$).

e-pendix: Shannon Noiseless Coding Theorem

Control Question 33

Consider a source U, the entropy of which is 2.15 bit. For the following values (2.75, 2.05, 3.25, 2.15), do you think a binary binary prefix-free codes of U with such a expected code length could exist? Do you think a better code, i.e. another binary prefix-free codes of U with a shorter expected code length, can exist? (yes, no, or maybe)

expected code length	could exist?	does a better code exist?		
		no	maybe	yes, for sure
2.75				
2.05				
3.25				
2.15				

Answer

expected code length	could exist?	does a better code exist?		
		no	maybe	yes, for sure
2.75	yes		**X**	
2.05	no	**X**		
3.25	yes			**X**
2.15	yes	**X**		

From the first part of Shannon Noiseless Coding Theorem, we know that no prefix-free binary code of U can have an expected length smaller than 2.15. Therefore 2.05 is impossible, and there is no better prefix-free code than a code having an expected length of 2.15.

The second part of the theorem tells us that we are sure that there exists at least one code whose expected code length is smaller than $H(U) + 1$ i.e. 3.15. Therefore we are sure that there exist a better code than a code having an expected length of 3.25.

For the other aspect of the question, the theorem doesn't say anything, neither that such code exist nor that they cannot exist. So maybe such codes could exist, according to our knowledge up to now.

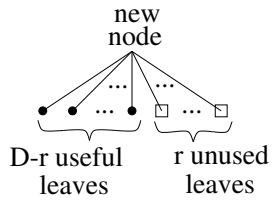
2.2.4 Huffman Codes

Huffman Coding Algorithm

We now show how to construct an optimum D-ary prefix-free code for a discrete memoryless information source U with n symbols. The algorithm for constructing such an optimal code is the following:

1. Start with n nodes (which will finally be the leaves of the coding tree) corresponding to the n symbols of the source $u_1, u_2, \ldots, u_n$.
   Assign the probability p_i to node u_i for all $1 \leq i \leq n$.
   Mark these n nodes as "active".
   Compute the remainder r of $1 - n$ divided by $D - 1$. Notice that, although $1 - n$ is negative, r is positive by definition of a remainder.
   Notice also that in the binary case ($D = 2$), r is always null.

2. Group together, as child nodes of a newly created node, the $D - r$ least probable active nodes together with r unused nodes (leaves):



   Mark the $D - r$ active nodes as "not active" and the newly created node as "active".
   Assign the newly created node a probability equal to the sum of the probabilities of the $D - r$ nodes just deactivated.

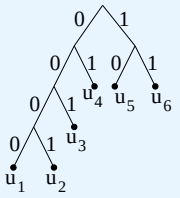
3. If there is only one active node, then stop (this node is then the root of the coding tree). Otherwise, set $r = 0$ and go back to step 2.

The prefix-free code resulting from such a coding algorithm is called a *Huffman code**, since the simple algorithm here described was discovered by D. Huffman in the fifties.

Example 2.16 (Binary Huffman Coding) Consider the information source U such that

U	u_1	u_2	u_3	u_4	u_5	u_6
p_i	0.05	0.1	0.15	0.27	0.20	0.23

One Huffman code for U is given by:



z_1	z_2	z_3	z_4	z_5	z_6
0000	0001	001	01	10	11

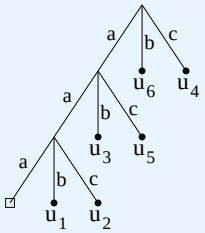
The probabilities associated to the interior nodes are the following:

$v_1 = u_1 \oplus u_2$	$v_2 = v_1 \oplus u_3$	$v_3 = u_5 \oplus u_6$	$v_4 = v_2 \oplus u_4$	$v_5 = v_4 \oplus v_3$
0.15	0.30	0.43	0.57	1

Finally, notice that $E[L] = 2\ (0.2 + 0.23 + 0.27) + 3\ (0.15) + 4\ (0.1 + 0.05) = 2.45$ (or by the Path Length Lemma: $E[L] = 1 + 0.57 + 0.43 + 0.30 + 0.15 = 2.45$), and $H(U) = -\sum_{i=1}^{6} p_i \log p_i = 2.42$ bit.

Example 2.17 (Ternary Huffman Coding) For the same source U as in the previous example and using a ternary code ($D = 3$), we have for the remainder of $1 - n := 1 - 6 = -5$ by $D - 1 := 2$: $r = 1$. Indeed, $-5 = -3 \cdot 2 + 1$. Therefore one unused leaf has to be introduced.

The ternary Huffman code is in this case:



z_1	z_2	z_3	z_4	z_5	z_6
aab	aac	ab	c	ac	b

The probabilities associated to the interior nodes are the following:

$v_1 = u_1 \oplus u_2$	$v_2 = v_1 \oplus u_3 \oplus u_5$	$v_3 = v_2 \oplus u_6 \oplus u_4$
0.15	0.50	1

Finally, notice that $E[L] = 1 + 0.5 + 0.15 = 1.65$ (by the Path Length Lemma), and

$\frac{H(U)}{\log 3} = \frac{2.42}{1.59} = 1.52$.

Control Question 34

Consider the unfair dice with the following probability distribution:

1	2	3	4	5	6
0.17	0.12	0.10	0.27	0.18	0.16

The purpose of this question is to build one binary Huffman code for this dice. For this code, we will use the convention to always give the label `0` to the least probable branch and the label `1` to the most probable branch. Furthermore, new nodes introduced will be named 7, 8, etc..., in that order.

1. What are the first two nodes to be regrouped? What is the corresponding probability?
2. What are then the next two nodes that are regrouped? What is the corresponding probability?
3. Keep on giving the names of the two nodes to be regrouped and the corresponding probability.
4. Give the Huffman code found for this source:

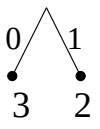
$u_i =$	1	2	3	4	5	6
$z_i =$						

Answer

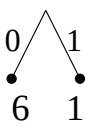
Huffman coding consists in iteratively regrouping the least two probable values. Let us then first order the source messages by increasing probability:

3	2	6	1	5	4
0.10	0.12	0.16	0.17	0.18	0.27

1) The first two values to be regrouped are then 3 and 2, leading to a new node "7" whose probability is $0.10 + 0.12 = 0.22$.

The corresponding subtree (useful for building the whole tree) is 

using the convention defined in the question.

2) The next iteration of the algorithm now regroups 6 and 1: 

and the corresponding probability is 0.33.

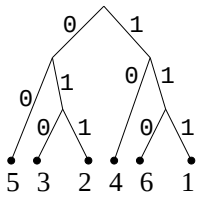
The new set of values is therefore:

5	7	4	8
0.18	0.22	0.27	0.33

3) Here are the next iterations:

9 is made with 5 and 7 , its probability is 0.4
10 is made with 4 and 8 , its probability is 0.6
11 is made with 9 and 10 , its probability is 1.0

4) The corresponding Huffman code (i.e. fullfiling the convention) is therefore:



i.e.

$u_i =$	1	2	3	4	5	6
$z_i =$	111	011	010	10	00	110

---

e-pendix: Huffman Coding

e-pendix: Efficient Coding

Optimality of Huffman Coding

We now want to demonstrate that Huffman coding is an optimal coding in the sense that no other prefix-free code can have an expected code length strictly shorter than the one resulting from the Huffman coding.

It is however important to remember that there are many optimal codes: permuting the code symbols or exchanging two codewords of the same length will give another code with the same expected length. The Huffman algorithm constructs only one such optimal code.

Before proving the optimality of Huffman codes, we first need to give a few properties of optimal codes in general.

A code is optimal if $\sum_{i=1}^{n} p_i \, l_i$ is minimal among all possible prefix-free code of the same source, denoting l_i the length of the codeword corresponding to the symbol u_i.

Lemma 2.2 *For an optimal code of some information source with n possible symbols we have:* $\forall i(1 \leq i \leq n)\ \forall j(1 \leq j \leq n)\ p_i > p_j \implies l_i \leq l_j$.

PROOF Let Z be an optimal code of the considered source. For a given i and a given j, consider the code Y in which the codewords z_i and z_j are swapped, i.e. $y_j = z_i$,

$y_i = z_j$ and $y_k = z_k$ for $k \neq i$, $k \neq j$. Then

$$\begin{aligned} E[L_Y] - E[L_Z] &= p_j\, l_i + p_i\, l_j - (p_i\, l_i + p_j\, l_j) \\ &= (p_i - p_j)\,(l_j - l_i). \end{aligned}$$

Because Z is optimal, $E[L_Y] \geq E[L_Z]$. Therefore, if $p_i > p_j$, $l_j - l_i$ has to be non-negative. ■

Lemma 2.3 (Node-Counting Lemma) *The number of leaves in a D-ary tree is $1 + k \cdot (D-1)$ where k is the number of interior nodes (including the root).*

PROOF Each interior node has D child nodes, therefore the total number of nodes in the tree which are a child of another node is $k \cdot D$. The only node in the tree which is not a child of another nodes is the root. The total number of nodes in the tree is then $k \cdot D + 1$.

But there are by definition k interior nodes, therefore the number of leaves (i.e. the number of nodes which are not interior nodes) is

$$k \cdot D + 1 - k = 1 + k \cdot (D-1).$$

Lemma 2.4 *For a given information source U, in the coding tree of an optimal prefix-free D-ary code of U, there are at most $D-2$ unused leaves and all these unused leaves are at maximum depth.*
Moreover, there is an optimal D-ary code for U in which all the unused leaves are child nodes of the same parent node.

PROOF If there is at least one unused leaf which is not at the maximum length, the expected code length could be decreased by transferring one of the codewords at maximum length to this unused leaf. The original code would therefore not be optimal.

Moreover, if there are more than D unused leaves at the maximum length, at least D of these unused nodes can be regrouped as child nodes of the same node and replaced by this only one unused node, which is at a depth shorten by 1. Therefore if there are more than D unused leaves, the code cannot be optimal.

Finally, if there are precisely $D-1$ unused leaves at the maximum length, they can be regrouped as child nodes of the same parent node, which also has one used leaf as its last child node. But the code can then be shorten by simply removing this last useless digit. Indeed, this last digit is not discriminative as all its sibling nodes are useless nodes. ■

Lemma 2.5 *The number of unused leaves in the tree of an optimal D-ary prefix-free code for a discrete information source U with n possible symbols, is the (positive) remainder of the division of $1-n$ by $D-1$.*

PROOF Let r be the number of unused leaves. Since U has n different symbols, we have:

$$r = \begin{bmatrix}\text{number of leaves in the} \\ D\text{-ary coding tree}\end{bmatrix} - n.$$

It follows then from the node-counting lemma that

$$r = [1 + k(D-1)] - n,$$

or

$$1 - n = -k(D-1) + r.$$

Moreover, from lemma 2.4, we know that, if the code is optimal, $0 \leq r < D-1$. It follows then, from Euclid's division theorem for integers, that r is the *remainder* of the division of $1-n$ by $D-1$ (the quotient being $-k$). ■

Lemma 2.6 *There exists an optimal D-ary prefix-free code for a discrete information source U with n different symbols ($n \geq 2$) such that the $D-r$ least probable codewords differ only in their last digit, with r the remainder of the division of $1-n$ by $D-1$ (therefore $D-r \geq 2$).*

PROOF First notice that not all optimal codes are claimed to satisfy this property, but by rearranging an existing optimal code, we can find at least one optimal code that satisfy the property.

Let us consider an optimal D-ary prefix-free code for U (this exists since the number of D-ary prefix-free codes for U is finite). By lemma 2.5, we know that there are r unused leaves, which by lemma 2.4 are all at maximal depth. Let us consider the $D-r$ siblings of these unused leaves. They are among the longest length codewords (since they are at maximal depth).

Let us now build the code where we exchange these $D-r$ longest codewords with the $D-r$ less probable ones. Due to lemma 2.2 this does not change the expected length (otherwise the considered code would not have been optimal). Therefore the resulting code is also an optimal code. But we are sure for this latter code that the $D-r$ least probable codewords differ only in their last digit. ■

Due to lemma 2.6, it is sufficient to look for an optimal code in the class of codes where the $D-r$ least probable codewords differ only in their last digit.

It is now time to establish the optimality of Huffman coding.

Theorem 2.6 *Huffman coding is optimal: if Z is one Huffman code of some information source U and X is another non-ambiguous code for U, then $E[L_X] \geq E[L_Z]$.*

PROOF We prove this theorem by induction on the number of codewords (i.e. the number of source symbols).

It is trivial to verify that for any source with less than D symbols, the Huffman code is optimal.

Suppose now that the Huffman coding procedure is optimal for any source with at most $n-1$ symbols and consider a source U with n symbols ($n > D$).

Let r be the remainder of the division of $1-n$ by $D-1$: $1-n = q(D-1)+r$.

Without loss of generality, let $u_{n-(D-r)+1}$, ..., u_n be the $D-r$ least probable source symbols.

By construction a Huffman code Z for U is made of an Huffman code Y for the source V whose $n-(D-r)+1$ different symbols are $v_1 = u_1$, ..., $v_{n-(D-r)} = u_{n-(D-r)}$ and $v_{n-(D-r)+1}$, with probabilities $q_1 = p_1$, ..., $q_{n-(D-r)} = p_{n-(D-r)}$ and $q_{n-(D-r)+1} = p_{n-(D-r)+1} + \cdots + p_n$.

Indeed the number of unused leaves introduced for Y is the remainder of the division of $1-[n-(D-r)+1]$ by $D-1$, which is 0 since $1-[n-(D-r)+1] = 1-n-r+D-1 = q(D-1)+(D-1) = (q+1)(D-1)$. This shows that Y indeed corresponds to the code built in the second and following steps of the building of Z.

Z appears then as an extension of Y in the codeword $y_{n-(D-r)+1}$: $z_1 = y_1$, ..., $z_{n-(D-r)} = y_{n-(D-r)}$ and $y_{n-(D-r)+1}$ is the prefix of all the codewords $z_{n-(D-r)+1}$,..., z_n which all differ only by the last symbol.

Then, denoting by l_i the length of z_i and by l'_i the length of y_i:

$$\begin{aligned} E[L_Z] &:= \sum_{i=1}^{n} p_i\, l_i = \sum_{i=1}^{n-D+r} p_i\, l_i + \sum_{i=n-D+r+1}^{n} p_i\, l_i \\ &= \sum_{i=1}^{n-D+r} p_i\, l'_i + \sum_{i=n-D+r+1}^{n} p_i\, (l'_i+1) \\ &= \sum_{i=1}^{n} p_i\, l'_i + \sum_{i=n-D+r+1}^{n} p_i \\ &= E[L_Y] + \sum_{i=n-D+r+1}^{n} p_i \end{aligned}$$

Since $\sum_{i=n-D+r+1}^{n} p_i$ is independent of the coding process (it only depends on the source U), due to lemma 2.6 and to the fact that Y, by induction hypothesis, is optimal for V (which has less than n symbols), we then conclude that Z is optimal for U (i.e. that $E[L_Z]$ is minimal). ■

Summary for Chapter 2

Prefix-Free Codes:

- no codeword is prefix of another
- prefix-free $\Longrightarrow$ non-ambiguous $\Longrightarrow$ non-singular
- prefix-free $\equiv$ instantaneously decodable

Kraft Inequality: $\exists$ prefix-free D-ary code $\Longleftrightarrow \sum_i D^{-l_i} \leq 1$

Entropy bound on expected code length:

$$E[L] = \sum_i p_i \, l_i \geq \frac{H(U)}{\log D}$$

Shannon-Fano code:

$$l_i = \left\lceil -\frac{\log p_i}{\log D} \right\rceil$$

$$E[L] = \sum_i p_i \, l_i \leq \frac{H(U)}{\log D} + 1$$

Huffman code:

1. introduce $1 - n \mod (D-1)$ unused leaves with probability 0
2. recursively regroup least probable nodes
3. is optimal (regarding the expected code length) among non-ambiguous codes

Historical Notes and Bibliography

Shannon Noiseless Coding Theorem	1948
Kraft Inequality	1949
Huffman Coding	1952

OutLook

For further details on coding for compression, refer to [9].

Chapter 3

Module C3: Entropy rate of stationary processes. Markov chains

by F. Bavaud

Introduction

Consider a doubly discrete system X_t which, at each time $t \in \mathbb{Z} := \{\ldots, -2, -1, 0, 1, 2, \ldots\}$, is in some state $x_t \in \Omega := \{\omega_1, \ldots, \omega_m\}$. In the description we adopt, the *state space* Ω is finite with $m = |\Omega|$ states; also, the set of times $t \in \mathbb{Z}$ is also discrete but bi-infinite at both extremities.

Definition 3.1 A *stochastic process* is a bi-infinite sequence

$$X_{-\infty}^{\infty} = \ldots X_{-2}X_{-1}X_0X_1X_2\ldots = \{X_t\}_{-\infty}^{\infty}$$

of random variables X_t indexed by an integer t. ♦

Example 3.1 Let X_t be the local temperature of the t-th day at noon. The sequence of successive X_t constitutes a stochastic process, which can be modelled by variously sophisticated models: for instance, the distribution of each X_t is independent of the other $X_{t'}$, or depends on X_{t-1} only, etc. The observation of a numerical series of values $\ldots x_{-2}x_{-1}x_0x_1x_2\ldots$ constitutes a realization of the stochastic processs.

Let $x_t \in \Omega$ represent the observed value of variable X_t. The stochastic process is completely determined provided all the joint (= multivariate) probabilities of the form

$$p(x_l \ldots, x_n) = p(x_l^n) := P(X_l = x_l, X_{l+1} = x_{l+1}, \ldots, X_n = x_n)$$

are defined, for all l and n with $l \leq n$. Note that sequence x_l^n contains $n - l + 1$ elements. To be consistent, the probability measure must add up to one:

$$\sum_{x_l^n \in \Omega^{n-l+1}} p(x_l^n) = 1$$

Also, the probability of a subsequence contained in a sequence must obtain by summing over all the values of the states of the sequence not appearing in the subsequence, that is, for instance

$$\sum_{(x_2\ x_3\ x_5)\ \in \Omega^3} p(x_1\ x_2\ x_3\ x_4\ x_5) = p(x_1\ x_4)$$

The probabilistic dynamics of a stochastic process might or might not depend explicitely of the time i at which it is observed. In the second case, the process is said to be stationary:

Definition 3.2 A stochastic process is *stationary* if the joint probability of a subsequence is invariant with respect to an overall shift in time, i.e.

$$\begin{aligned} & P(X_{l+T} = x_l, X_{l+T+1} = x_{l+1} \ldots X_{n+T} = x_n) \\ = \quad & P(X_l = x_l, X_{l+1} = x_{l+1} \ldots X_n = x_n) \end{aligned}$$

for any $T \in \mathbb{Z}$. ♦

3.1 The entropy rate

Learning Objectives for Section 3.1

After studying this section you should

- be familiar with the concept of *entropy rate*, in relation to the concepts of *typical set*, *redundancy* and *compression*
- be able to assess whether or not a given compression scheme (diminishing the length and/or the number of categories of the message) is feasible in principle

Consider a stationary process. The joint entropy of the sequence $X_1, X_2, \ldots, X_k = X_1^k$, measuring the total uncertainty in the joint outcome of the sequence, is

$$H(X_1^k) = -\sum_{x_1^k \in \Omega^k} p(x_1^k)\ \log p(x_1^k)$$

$H(X_1^k)$ is increasing in k: adding more arguments increases uncertainty. One defines the *entropy rate* $\widehat{h_\infty}$ of the process as the limit (if existing)

$$\widehat{h_\infty} := \lim_{k \to \infty} \frac{1}{k} H(X_1^k) \tag{3.1}$$

Also, consider the conditional entropy h_k, for $k = 1, 2, 3 \ldots$, measuring the uncertainty in X_k conditionnaly to $X_1 \ldots, X_{k-1}$, as well as its limit when $k \to \infty$:

$$h_k := H(X_k|X_1^{k-1}) \overset{(a)}{=} H(X_1^k) - H(X_1^{k-1}) \qquad h_\infty := \lim_{k\to\infty} h_k \tag{3.2}$$

where (a) follows from $H(Y|Z) = H(Y, Z) - H(Z)$. The quantity $\widehat{h_\infty}$ in 3.1 measures the uncertainty *per variable* in an infinite sequence, and the quantity h_∞ in 3.2 measures the residual entropy on the last variable when all the past is known. It turns out that those two quantities coincide for stationary processes:

Theorem 3.1 *For a stationary processes, the non-negative quantity h_k defined in 3.2 is non-increasing in k. Its limit $h_\infty := \lim_{k\to\infty} h_k$ defines the* entropy rate h_∞ *of the process exists and can be computed either way as*

$$h_\infty = \lim_{k\to\infty} \frac{1}{k} H(X_1^k) = \lim_{k\to\infty} H(X_k|X_1^{k-1}) \tag{3.3}$$

Example 3.2 For an i.i.d. process, $H(X_1^k) = k\, H(X)$, where $H(X)$ is the entropy for a single variable. As a result, $h_\infty = \lim_{k\to\infty} \frac{1}{k}\, kH(X) = H(X)$. The behavior of h_k for this and other processes is depicted in figure 3.10.

Theorem 3.2 *$h_\infty \leq \log m$, where $m := |\Omega|$ be the alphabet size. Equality holds iff the process is independent (that is $p(x_1^k) = \prod_{i=1}^k p(x_i)$) and uniform (that is $p(x_i) = 1/m$).*

PROOF

$$h_\infty = \lim_{k\to\infty} H(X_k|X_1^{k-1}) \overset{(a)}{\leq} \lim_{k\to\infty} H(X_k) \overset{(b)}{\leq} \log m$$

where equality in (a) holds under independence, and equality in (b) holds under uniformity. ■

The entropy rate h_∞ measures the conditional uncertainty associated to each single outcome of a process, knowing its whole past. For fixed m, this uncertainty is maximal when the predictibility of the outcome is minimal, that is when the process is maximally random. Theorem 3.2 says that a maximally random process must be uniform and independent, exacly as a fair dice with m sides which must be unbaised (= uniform) and without memory (= independent successive outcomes).

More generally, the following exact decomposition holds:

Theorem 3.3 *For a stationary process,*

$$h_k = \log m - K_k(p||p^{IND}) - K(p||p^{UNI})$$

where $K_k(p||p^{IND}) \geq 0$ *is the Kullback-Leibler departure from independence, namely*

$$K_k(p||p^{IND}) := \sum_{x_1^k} p(x_1^k)\ \log \frac{p(x_1^k)}{p^{IND}(x_1^k)} \qquad p^{IND}(x_1^k) := p(x_1^{k-1})\, p(x_k) \tag{3.4}$$

and $K(p||p^{UNI})$ *is the Kullback-Leibler departure from uniformity, namely*

$$K(p||p^{UNI}) := \sum_{x_k} p(x_k)\ \log \frac{p(x_k)}{1/m} \tag{3.5}$$

Note: observe $K(p||p^{\text{UNI}})$ to be independent of k by stationarity.

PROOF By construction

$$h_k = H(X_k|X_1^{k-1}) = -\sum_{x_1^{k-1}} p(x_1^{k-1}) \sum_{x_k} p(x_k|x_1^{k-1}) \log p(x_k|x_1^{k-1}) =$$

$$= -\sum_{x_1^{k-1}} p(x_1^{k-1}) \sum_{x_k} p(x_k|x_1^{k-1}) \log[\frac{p(x_1^k)}{p(x_1^{k-1})}\ \frac{p(x_k)}{p(x_k)}\ \frac{1/m}{1/m}] =$$

$$= -\sum_{x_1^k} p(x_1^k) \log \frac{p(x_1^k)}{p(x_1^{k-1})\, p(x_k)}\ - \sum_{x_k} p(x_k)\ \ln \frac{p(x_k)}{1/m} + \log m$$

Remark: using information-theoretical quantities, the proof can alternatively be presented as

$$h_k = H(X_1^k) - H(X_1^{k-1}) =$$

$$= \underbrace{H(X_1^k) - H(X_1^{k-1}) - H(X_k)}_{-K_k(p||p^{\text{IND}})} + \underbrace{H(X_k) - \log m}_{-K(p||p^{\text{UNI}})} + \log m$$

3.2 The AEP theorem

Recall that a variable Z_n is said to converge in probability towards the constant c (noted $Z_n \xrightarrow{P} c$) iff

$$\lim_{n\to\infty} P(|Z_n - c| \leq \varepsilon) = 1) \qquad \forall \varepsilon > 0 \tag{3.6}$$

Example 3.3 Let $\{Y_i\}$ represent a sequence of i.i.d. numerical variables with mean μ and finite variance σ^2. The variable $S_n := \frac{1}{n}\sum_{t=1}^n Y_t$ then converges in probability to μ. That is, for n sufficiently large, the probability to observe a deviation $S_n - \mu$ larger than any finite quantity $\varepsilon > 0$ becomes negligible.

Theorem 3.4 *AEP (= asymptotic equipartition property) theorem: for a stationary ergoodic processes, the variable*

$$Z_n := -\frac{1}{n}\log p(X_1^n) \tag{3.7}$$

converges in probability towards the entropy rate h_∞ defined in 3.3: $Z_n \xrightarrow{P} h_\infty$.

Equivalently (see theorem 3.5 (a)), and using from now on natural logarithms for simplicity, theorem 3.4 tells that

$$\forall \varepsilon > 0 \quad , \qquad \lim_{n\to\infty} P(\exp[-n(h_\infty+\varepsilon)] \;\le\; p(X_1^n) \;\le\; \exp[-n(h_\infty-\varepsilon)]) = 1$$

Definition 3.3 The (n,ε)-*typical set* $T_n(\varepsilon)$ is the set of all empirical sequences x_1^n (called *typical*) whose probability $p(x_1^n)$ is close to $\exp(-n\,h_\infty)$ in the sense

$$T_n(\varepsilon) := \{x_1^n \in \Omega^n \,|\, \exp[-n(h_\infty+\varepsilon)] \;\le\; p(x_1^n) \le\; \exp[-n(h_\infty-\varepsilon)]\} \tag{3.8}$$

The probability for data to belong to $T_n(\varepsilon)$ is

$$P(T_n(\varepsilon)) := P(X_1^n \in T_n(\varepsilon)) = \sum_{x_1^n \in \Omega^n} p(x_1^n)\, I(x_1^n \in T_n(\varepsilon)) \tag{3.9}$$

and theorem 3.4 can be rewritten as

$$\forall \varepsilon > 0 \quad , \qquad \lim_{n\to\infty} P(X_1^n \in T_n(\varepsilon)) = 1 \tag{3.10}$$

That is, for increasingly large n, it becomes increasingly certain that, drawing a n-gram X_1^n turning out to be x_1^n, one finds that $p(x_1^n)$ is very close $\exp(-n\,h_\infty)$: "most" of the observed empirical sequences x_1^n have a probability increasingly close to $\exp(-n\,h_\infty)$: "for n large, almost all events are almost equally surprising". The following theorem makes this statement precise and rigorous:

Theorem 3.5

1. $x_1^n \in T_n(\varepsilon)$ *iff* $|-\frac{1}{n}\ln p(X_1^n) - h_\infty| \le \varepsilon$.
2. $P(T_n(\varepsilon)) > 1-\varepsilon$ *for n large enough.*
3. $|T_n(\varepsilon)| \le \exp(n\,(h_\infty+\varepsilon))$.
4. $|T_n(\varepsilon)| > (1-\varepsilon)\,\exp(n\,(h_\infty-\varepsilon))$ *for n large enough.*

PROOF

1. apply definition 3.6.
2. as a consequence of equation 3.10, $P(T_n(\varepsilon))$ is arbitrarily close to 1 for n large enough.

3.

$$1 = \sum_{x_1^n \in \Omega^n} p(x_1^n) \geq \sum_{x_1^n \in T_n(\varepsilon)} p(x_1^n) \geq \sum_{x_1^n \in T_n(\varepsilon)} \exp[-n(h_\infty + \varepsilon)] = \\ = \exp[-n(h_\infty + \varepsilon)] \, |T_n(\varepsilon)|$$

4.

$$1 - \varepsilon < P(T_n(\varepsilon)) \leq \sum_{x_1^n \in T_n(\varepsilon)} \exp(-n(h_\infty - \varepsilon)) = \exp(-n(h_\infty - \varepsilon)) \, |T_n(\varepsilon)|$$

3.2.1 The concept of typical set: redundancy and compressibility

The set Ω^n of all sequences x_1^n of length n with $m = |\Omega|$ categories contains m^n distinct elements. Supppose the probabilistic process producing them to be independent and uniform; then each sequence possesses the same probability, namely $P(x_1^n) = m^{-n}$. In the other direction, suppose the dependence in the process to be so strong that a single sequence of length n can possibly be produced; by construction, this sequence has probability one, and the remaining $m^n - 1$ sequences have probability zero.

In general, that is inbetween those two extreme situations, the key quantity controlling the number of "typically observable" sequences as well as their probability is the entropy rate h_∞ (recall $h_\infty = \log m$ in the first case above, and $h_\infty = 0$ in the second case). One has that the Ω^n of all sequences x_1^n of length n splits, for n large enough, into two subsets [1]:

- the set $T_n(\varepsilon)$ of *typical sequences*, containing essentially $|T_n(\varepsilon)| \cong \exp(nh_\infty)$ sequences, each of them having probability $\exp(-nh_\infty)$. For n large, all the probability is concentrated in the typical set: $P(T_n(\varepsilon)) \cong 1$.
- the set of *non-typical sequences* $T_n^c(\varepsilon) := \Omega^n \setminus T_n(\varepsilon)$, containing $m^n - \exp(nh_\infty)$ sequences, each of them having negligible probability: $P(T_n^c(\varepsilon)) \cong 0$.

Thus, the higher the entropy rate h_∞, that is the more uncertain the future given the whole past, the more numerous the typical sequences, the only ones to be actually observable (for n large). Closely associated to the concept of entropy rate is the notion of *redundancy* :

Definition 3.4 (redundancy) The redundancy R of a stationary ergodic stochastic process on m states with entropy rate h_∞ is

$$R := 1 - \frac{h_\infty}{\log m} \tag{3.11}$$

It is understood that the same logarithmic unit (e.g. bits or nats) is used in $\log m$ and in the defintion of h_∞, which makes R independent of the choice of units. It

[1] The previous statements can be given a fully rigorous status, thanks to the *asymptotic equipartition property* (AEP) theorem; here $T_n(\varepsilon)$ denotes the set of sequences of length n whose probability stands between $\exp(-n(h_\infty + \varepsilon))$ and $\exp(-n(h_\infty - \varepsilon))$.

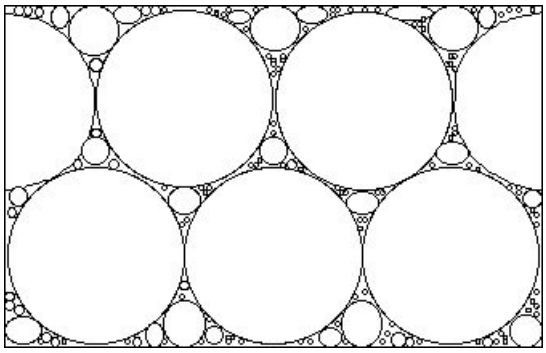


Figure 3.1: The AEP theorem: for $0 < h_\infty < \log m$ and n large, almost all sequences of Ω^n are non-typical (sand grains) and their contribution to the total volume is negligible. By contrast, the relative number of typical sequences (peebles) is negligible, but their volume accounts for the quasi-totality of the total volume.

follows from $0 \leq h_\infty \leq \log m$ (theorem 3.2) that $0 \leq R \leq 1$. By construction, $h_\infty = (1 - R) \log m$ and $\exp(n h_\infty) = m^{(1-R)n}$. Thus, among all the m^n sequences of Ω^n, a total of $|T_n(\varepsilon)| \cong m^{(1-R)n}$ of them are typical; each of those typical sequences has probability $m^{-(1-R)n}$. In particular:

- a maximally random process (i.e. independent and uniform) is characterized by $h_\infty = \log m$, or equivalently $R = 0$. The number of typical sequences is equal to m^n, the total number of sequences; that is, each sequence is typical and has probability m^{-n}.
- a minimally random process is characterized by $h_\infty = 0$, or equivalently $R = 1$. The process is eventually deterministic: given a large enough piece of past $\ldots, x_l, x_{l+1}, \ldots, x_n$, the value x_{n+1} of the next state X_{n+1} can be predicted with certainty. The number of typical sequences is equal to $m^0 = 1$: that is, there is asymptotically an unique typical sequence with probability 1, namely the only sequence produced in a deterministic process.
- inbetween those extreme cases, a generic stochastic process obeying $0 < h_\infty < \log m$ satisfies $0 < R < 1$: the process is partly redundant and partly impredictible. The proportion of typical sequences is

$$\frac{m^{(1-R)n}}{m^n} = m^{-R\,n}$$

and vanishes for $n \to \infty$.

Example 3.4 The following metaphor, due to Hillman (1996), might help making the AEP concept intuitive (see figure 3.1). A beach of total volume m^n represents the totality of the sequences. Most of the volume is made up of $\exp(n h_\infty)$ peebles, each having a volume of $\exp(-n h_\infty)$. The beach also contains about m^n sand grains, but so small that their total contribution to the volume of the beach is of order $\varepsilon << 1$.

Thus, among the m^n possible sequences, only $m^{(1-R)\,n}$ can indeed occur, all with the same probability (for n large). That is, the total average amount of information $n\,h_\infty$ carried by a sequence of length n with entropy rate h_∞ on an alphabet m can equivalently be obtained

A) by the set of sequences of effective length $n_{\text{eff}} = (1-R)n$ equiprobably distributed over m categories; the entropy rate of this new process reaches now its maximum $\log m$.

B) by the set of sequences of length n equiprobably distributed over $m^{(1-R)}$ categories, with a corresponding entropy rate of $\log m^{(1-R)} = (1-R)\,\log m$.

Modifying the original process does not modify the *total information*, which remains equal to nh_∞:

$$n_{\text{eff}}\,\log m = n\,(1-R)\,\log m = n\,h_\infty \quad \text{(A)}$$
$$n\,\log m^{(1-R)} = n\,(1-R)\,\log m = n\,h_\infty \quad \text{(B)}$$

However, the *redundancy* of the modified processes is now zero in both cases: application of 3.11 yields

$$R_A = 1 - \frac{\log m}{\log m} = 0 \quad \text{(A)} \qquad\qquad R_B = 1 - \frac{(1-R)\,\log m}{(1-R)\,\log m} = 0 \quad \text{(B)}$$

The precise, detailled ways in which the initial sequences can be *compressed* (from the point of view of their lengths (A) or their alphabet size (B)) constitutes a part of the *coding theory* (see module C2).

Example 3.5 The entropy of a "representative" text in simplified english with $m = 27$ categories (no punctuation nor distinction of cases, that is 26 letters plus the blank) has been estimated to about $h = 1.3$ bits per letter, corresponding to an redundancy of about $R = 1-1.3/\log_2 27 = 0.73$ (Shannon 1950, cited in Cover and Thomas 1991). That is, a 300 pages novel could typically be reduced to a $300(1-0.73) = 81$ pages novel on the same alphabet, or to a novel of same length with only $27^{(1-0.73)} = 2.43$ (i.e. at least 3) symbols. The aspect of a sample of the latter could be something like

```
MUUMMXUUMMMMUMXXUMXMMXMMUXMUMXMMMMXXXUUXMXMUUXMUXMUXMU
UMUXUMUUUXMMUUUMXMMMXXXXMUMXXMMXXMUMXUMUUXMUUXMMMXMUXX
UXXXUXXUUMMUXUXMUUMXUUXMUXXUXUMUUUXXXMMXXUMXUUUMMUXMXM
```

where the redundancy of the new text is now zero, meaning that the slightest modification of the latter will irreversibly alter its content. By contrast, the high redundancy ($R = 0.73$) of standard english permits to correct and recover an altered text, containing for instance misspellings.

Note: the above example presupposes that written english is produced by a stationary stochastic process, which is naturally questionnable.

Control Question 35

A stationary stochastic process produces a sequence of n consecutive symbols (n large) on an alphabet with m signs. Suppose the redundancy of the process to be $R = 0.25$. Then

1. it is possible to compress (without diminishing the total information) the sequence length from n to $0.75 \times n$? **Possible answers:** yes - no - it depends on n.

2. it is possible to compress (without diminishing the total information) the alphabet length from m to $0.75 \times m$? **Possible answers:** yes - no - it depends on m.

Answer

1. "yes": correct answer: compression scheme A) above permits to compress the sequence length (on the same alphabet) from n to $n_{\text{eff}} = (1 - 0.75)\, n = 0.75 \times n$ (maximum compression, asymtotically attainable for $n \to \infty$).

2. "it depends on m": correct answer: the alphabet size can be taken (with out diminishing the sequence length) to $m^{1-R} = m^{0.75}$. But one readily checks $m^{0.75} > 0.75 \times m$ (unfeasible compression) if $m = 2$, and $m^{0.75} < 0.75 \times m$ (feasible compression) if $m = 3, 4, \ldots$.

Example 3.6 A highly simplified meteorological description assigns each day into one of the three categories "nice" (N), "rainy" (R) or "snowy" (S). For instance, "NNRNSRN" constitutes a possible a meteorological week. The are a maximum of 3^n different sequences of n days; each of those sequence would be equiprobable (with probability 3^{-n}) if the whether did follow a perfectly random process (with a maximum entropy rate of $h_\infty = \log 3$, as given by theorem 3.2). However, real weather is certainly not a completely random process, that is its redundancy R is strictly positive:

- if $R = 0.5$ for instance, then, among all possible 3^n different sequences of n days, $3^{(1-0.5)\, n} = 1.73^n$ are typical, that is likely to occur.
- if $R = 0.75$, only $3^{(1-0.75)\, n} = 1.32^n$ sequences of n days are typical.
- if $R = 1$, only $3^{(1-1)\, n} = 3^0 = 1$ sequence is typical, that is the only sequence generated by the deterministic ($R = 1$) process.

In this example, the number of effective "full" possible types of weather for next day (as measured by the uncertainty conditional to the day before) passes from 3 to 1.73, 1.32 and even 1 as R increases from 0 to 1.

3.3 First-order Markov chains

Learning Objectives for Section 3.3

After studying this section you should

- be familiar with the concept of (first-order) *Markov chain*, its transition matrix and their iterates, and the concept of stationary distribution.
- be able to classify the states as *recurrent*, *transient*, *absorbing* and *periodic*.
- understand the theoretical meaning as well as the computation of the associated entropy.
- understand the nature of the irreversibility produced by temporal evolution.

Definition 3.5 A *first-order Markov chain*, or simply *Markov chain*, is a discrete stochastic process whose memory is limited to the last state, that is:

$$p(x_{t+1}|x_{-\infty}^{t}) = p(x_{t+1}|x_t) \qquad \forall t \in \mathbb{Z}$$

Let $\Omega := \{\omega_1, \ldots, \omega_m\}$ represent the m states of system. The Markov chain is entirely determined by the $m \times m$ *transition matrix*

$$p_{jk} := P(X_{t+1} = \omega_k | X_t = \omega_j) = p(\omega_k|\omega_j)$$

obeying the consistency conditions

$$p_{jk} \geq 0 \qquad \sum_{k=1}^{m} p_{jk} = 1$$

Example 3.7 Consider a two-states process with state space $\Omega = \{a, b\}$. When the system is in state a, it remains in the same state with probability 0.7 (and moves to state b with probability 0.3). When the system is in state b, it remains in the same state with probability 0.6 (and moves to state a with probability 0.4). The conditional probabilities are $p(a|a) = 0.7$, $p(b|a) = 0.3$, $p(b|b) = 0.6$ and $p(a|b) = 0.4$. Numbering a as 1 and b as 2, the probabilities equivalently express as $p_{11} = 0.7$, $p_{12} = 0.3$, $p_{21} = 0.4$ and $p_{22} = 0.6$, or, in matrix form,

$$P = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} \tag{3.12}$$

Observe each *row* to sum to 1.

3.3.1 Transition matrix in n steps

Consider a process governed by a Markov chain which is in state ω_j at time t. By definition, its probability to be in state k at time $t + 1$ is p_{jk}. But what about its

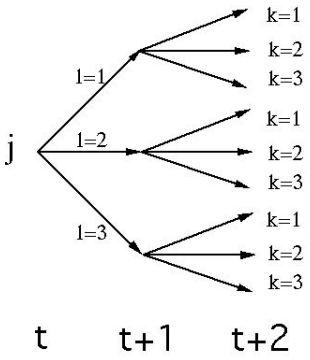


Figure 3.2: The probability to reach state k at time $t+2$ (say $k=3$) starting from state j at time t obtains by summing over all possible intermediate states l at time $t+1$.

probability to be in state k at time $t+2$?

Figure 3.2 shows that the searched for probability obtains by summing over all possible intermediate states l at time $t+1$ (here $m=3$). That is

$$p_{jk}^{(2)} := P(X_{t+2} = \omega_k | X_t = \omega_j) = \sum_{l=1}^{m} p_{jl}\, p_{lk} \tag{3.13}$$

Denoting $P := (p_{jk})$ and $P^{(2)} := (p_{jk}^{(2)})$ the $m \times m$ one-step and two-steps transition matrices respectively, equation 3.13 shows the latter to obtain from the former by straighforward matrix multiplication, that is $P^{(2)} = P^2 := P \cdot P$. The mechanism generalizes for higher-order lags and we have the result

Theorem 3.6 *The n-step transition matrix $P^{(n)}$ whose components $p_{jk}^{(n)} := P(X_{t+n} = \omega_k | X_t = \omega_j)$ give the probability to reach state k at time $t+n$ given that the system was in state j at time t obtains from the (one-step) transition matrix $P = (p_{jk})$ as*

$$P^{(n)} = \underbrace{P \cdot P \cdot P \cdots P}_{n \text{ times}} = P^n$$

Example 3.8 (example 3.7, continued:) The two- and three-steps transition matrices giving the probabilitites to reach state k from state j in $n=2$, respectively $n=3$ steps, are $P^{(2)}$ and $P^{(3)}$ with

$$P^{(2)} := P \cdot P = \begin{pmatrix} 0.61 & 0.39 \\ 0.52 & 0.48 \end{pmatrix} \qquad P^{(3)} := P \cdot P \cdot P = \begin{pmatrix} 0.583 & 0.417 \\ 0.556 & 0.444 \end{pmatrix}$$

Note the property that each row of P sums to 1 to be automatically inherited by $P^{(2)}$ and $P^{(3)}$.

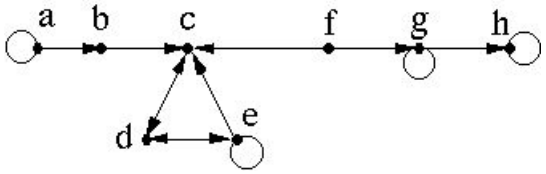


Figure 3.3: State a *reaches* itself and states b, c, d and e. State b reaches itself and states c, d and e. State c reaches itself and states d and e. State d reaches itself and states c and e, etc. The *communication* equivalence classes are $\{a\}$, $\{b\}$, $\{c, d, e\}$, $\{f\}$, $\{g\}$ and $\{h\}$.

3.3.2 Flesh and skeleton. Classification of states

The concept of communication between states defines an equivalence relation among the set of m states involved in a finite Markov chain:

Definition 3.6

- state j *reaches* state k, written $j \to k$, if there is a path $jl_1l_2 \cdots l_{n-1}k$ of length $n \geq 0$ such that $p_{jl_1} p_{l_1l_2} \cdots p_{l_{n-1}k} > 0$, i.e. if there is a $n \geq 0$ such that $p_{jk}^{(n)} > 0$. As $p_{jj}^{(0)} = 1 > 0$, each state reaches itself by construction: $j \to j$ for all j.
- states j and k commmunicate, noted $j \leftrightarrow k$, iff $j \to k$ and $k \to j$.

Thus the relation "$\to$" is reflexive ($j \to j$) and transitive ($j \to l$ and $l \to k$ imply $j \to k$). The communicability relation "$\leftrightarrow$" is in addition symmetric ($j \leftrightarrow k$ imply $k \leftrightarrow j$). That is, the relation "communicates with" is an equivalence relation, partitioning states into groups of states, each state inside a group communicating with all the other states inside the same group.

Note that the "skeleton" aspects (i.e whether a transition is possible or not) dominate the "flesh" aspects (i.e. the question of the exact probability a possible transition) in the above classification. That is, j and k communicate iff there are integers n and n' with $p_{jk}^{(n)} > 0$ and $p_{kj}^{(n')} > 0$; the question of the exact values of $p_{jk}^{(n)} > 0$ and $p_{kj}^{(n')} > 0$ is of secondary importance relatively to the fact that those two quantities are stricly positive.

Example 3.9 Consider a Markov chain whith skeleton given by figure 3.3. The arrows denote reachability *in one step*. State a reaches itself as well as states b, c, d and e. However, a can be reached from itself only. Thus the equivalence class of a (relatively to the relation "$\leftrightarrow$") contains a itself. Reasonning further, one finds the equivalence classes to be $\{a\}$, $\{b\}$, $\{c, d, e\}$, $\{f\}$, $\{g\}$ and $\{h\}$.

Definition 3.7 State j is *recurrent* (or *persistent*, or *ergodic*) if the probability that the process starting from j will eventually return to j is unity. State j is *transient* if it is not recurrent, that is if the probability of no return to j starting from j is non zero. ♦

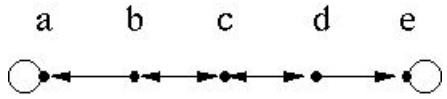


Figure 3.4: Example 3.10: the communication equivalence classes are $\{a\}$, $\{e\}$ (recurrent classes) and $\{b, c, d\}$ (transient class). States a et e are absorbing.

One can show that states belonging to the same equivalence class are either all recurrent or all transient, which justifies the following definition:

Definition 3.8 *Reccurent classes* are equivalence classes whose elements are all recurrent. *Transient classes* are equivalence classes whose elements are transient. ♦

In example 3.9, the recurrent classes are $\{c, d, e\}$ and $\{h\}$. All other classes are transient.

Example 3.10 Consider the following Markov transition matrix

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.14}$$

whose skeleton is represented in figure 3.4. There are two recurrent classes, namely $\{a\}$ and $\{e\}$, and one transient class, namely $\{b, c, d\}$. Recurrent states which are single members of their class, such as a and e, cannot be left once entered. Such states are said to be *absorbing*. A necessary and sufficient condition for a state j to be absorbing is $p_{jj} = 1$, as demonstrated in rows 1 and 5 of 3.14.

It might happen that $p_{jj}^{(n)} = 0$ for all n not divisible by d, and d is the greatest such integer. This means that if the chain is in state j at some time t, it can only return there at times of the form $t + md$ where m is an integer. Then state j is said to have *period* d. A state with period $d = 1$ is said to be *aperiodic*. If $p_{jj}^{(n)} = 0$ for all $n \geq 1$, state j has an infinite period $d = \infty$. One can show that states belonging to the same equivalence class have all the same period: for instance, in figure 3.4, states a and e are aperiodic, while b, c and d have period $d = 2$.

Example 3.11 A *tree* is a graph containing no circuit. Figure 3.5 left depicts a Markov chain on a symmetric tree: there is a single recurrent equivalence class $\{a, b, c, d, e, f\}$, all states of which have period $d = 2$. Adding a single circuit such as in figure 3.5 middle or right still conserves the single equivalence class $\{a, b, c, d, e, f\}$, but all states are now *aperiodic* ($d = 1$).

3.3.3 Stationary distribution

From now on one considers *regular Markov chains only*, that is consisting of a single aperiodic recurrent class: equivalently, each state can be attained from each other after

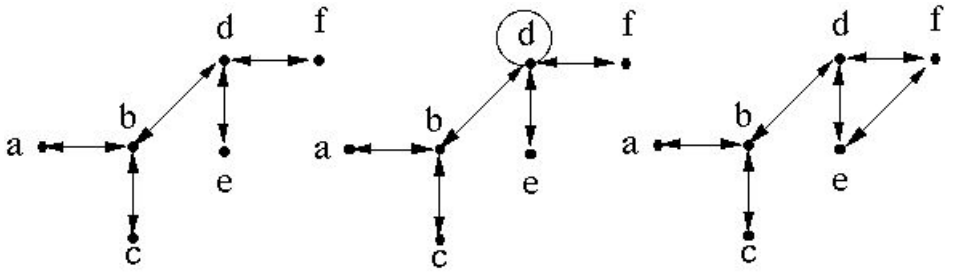


Figure 3.5: Example 3.11. Left: the underlying skeleton is a symmetric tree, and all states have period $d = 2$. The addition of a single circuit (middle or right) makes all states aperiodic ($d = 1$).

sufficient lapse of time, i.e. there exist an integer N such that $p_{jk}^{(n)} \geq 0$ for all states j, k and all times $n \geq N$.

Theorem 3.7 *Let $P = (p_{jk})$ be the $m \times m$ transition matrix of a regular Markov chain on m states. Then*

- *for $n \to \infty$, the powers P^n approach a transition matrix P^∞ of the form*

$$P^\infty = \begin{pmatrix} \pi_1 & \pi_2 & \cdots & \pi_{m-1} & \pi_m \\ \pi_1 & \pi_2 & \cdots & \pi_{m-1} & \pi_m \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \pi_1 & \pi_2 & \cdots & \pi_{m-1} & \pi_m \\ \pi_1 & \pi_2 & \cdots & \pi_{m-1} & \pi_m \end{pmatrix} \qquad \text{with } \pi_j > 0 \text{ and } \sum_{j=1}^{m} \pi_j = 1 \quad (3.15)$$

- *the distribution $\pi = (\pi_1, \pi_2, \ldots, \pi_m)'$ is the only solution of the equation*

$$\sum_{j=1}^{m} \pi_j \, p_{jk} = \pi_k \quad \forall k \qquad \text{i.e.} \quad P'\pi = \pi \quad (3.16)$$

obeying the normalization condition $\sum_{j=1}^{m} \pi_j = 1$.

The distribution π is referred to as the *stationary* or *equilibrium* distribution associated to the chain P.

PROOF *** classical proof remaining to be done *** ■

Example 3.12 Consider the following transition matrix

$$P = \begin{pmatrix} 0.823 & 0.087 & 0.045 & 0.044 \\ 0.058 & 0.908 & 0.032 & 0.001 \\ 0.030 & 0.032 & 0.937 & 0.001 \\ 0.044 & 0.002 & 0.001 & 0.952 \end{pmatrix} \quad (3.17)$$

Some of its successive powers are

$$P^{(5)} = \begin{pmatrix} 0.433 & 0.263 & 0.161 & 0.143 \\ 0.175 & 0.662 & 0.137 & 0.025 \\ 0.107 & 0.137 & 0.741 & 0.014 \\ 0.143 & 0.037 & 0.021 & 0.798 \end{pmatrix}$$

$$P^{(25)} = \begin{pmatrix} 0.204 & 0.308 & 0.286 & 0.202 \\ 0.205 & 0.349 & 0.305 & 0.140 \\ 0.191 & 0.305 & 0.390 & 0.114 \\ 0.202 & 0.211 & 0.171 & 0.416 \end{pmatrix}$$

$$P^{(\infty)} = \begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.2 & 0.3 & 0.3 & 0.2 \\ 0.2 & 0.3 & 0.3 & 0.2 \\ 0.2 & 0.3 & 0.3 & 0.2 \end{pmatrix} = \begin{pmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \\ \pi_1 & \pi_2 & \pi_3 & \pi_4 \\ \pi_1 & \pi_2 & \pi_3 & \pi_4 \\ \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{pmatrix}$$

and the corresponding equilibrium distribution is $\pi = (0.2, 0.3, 0.3, 0.2)$. One can verify $\sum_{j=1}^{m} \pi_j \, p_{jk} = \pi_k$ to hold for each k: indeed

$$\begin{aligned} 0.2 \cdot 0.823 + 0.3 \cdot 0.087 + 0.3 \cdot 0.045 + 0.2 \cdot 0.0.044 = 0.2 \qquad & (k = 1) \\ 2 \cdot 0.058 + 0.3 \cdot 0.908 + 0.3 \cdot 0.0.032 + 0.2 \cdot 0.001 = 0.3 \qquad & (k = 2) \\ 2 \cdot 0.030 + 0.3 \cdot 0.0.032 + 0.3 \cdot 0.937 + 0.2 \cdot 0.001 = 0.3 \qquad & (k = 3) \\ 2 \cdot 0.044 + 0.0.3 \cdot 0.002 + 0.3 \cdot 0.001 + 0.2 \cdot 0.952 = 0.2 \qquad & (k = 4) \end{aligned}$$

3.3.4 The entropy rate of a Markov chain

Theorem 3.8 *The entropy rate of a first-order regular Markov chain with transition matrix $P = (p_{jk})$ is*

$$h_\infty = -\sum_j \pi_j \sum_k p_{jk} \log p_{jk} \tag{3.18}$$

where π is the stationary distribution associated with the transition matrix P.

PROOF Theorem 3.1 yields

$$h_\infty = \lim_{n\to\infty} H(X_n|X_1^{n-1}) \overset{(a)}{=} \lim_{n\to\infty} H(X_n|X_{n-1})$$
$$\overset{(b)}{=} \lim_{n\to\infty} \sum_{j=1}^{m} p_j^{(n-1)} \, [-\sum_{k=1}^{m} p_{jk} \log p_{jk}] \overset{(c)}{=} -\sum_j \pi_j \sum_k p_{jk} \log p_{jk}$$

where (a) follows from definition 3.5, (b) follows from the definition $H(X_n|X_{n-1})$ involving $p_j^{(n-1)}$, the probability for the system to be in state j at time $n-1$, and (c) follows from theorem 3.7, implying $\lim_{n\to\infty} p_j^{(n-1)} = \pi_j$. ■

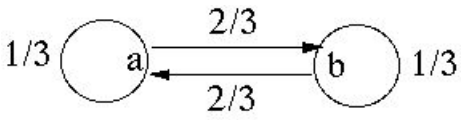


Figure 3.6: A first-order Markov chain on $\Omega = \{a, b\}$.

Example 3.13 Consider the Markov chain (of order 1) on 2 states $\Omega = \{a, b\}$, with $p(a|a) = 2/3$, $p(b|a) = 1/3$, $p(b|b) = 2/3$ and $p(a|b) = 1/3$ (figure 3.6). By symmetry, the corresponding stationary distribution is $\pi(a) = \pi(b) = 0.5$. In view of 3.18, its entropy rate is

$$\begin{aligned} h_\infty = h_2 &= -\pi(a)\,[p(a|a)\ln p(a|a) + p(b|a)\ln p(b|a)] \\ &\quad -\pi(b)\,[p(a|b)\ln p(a|b) + p(b|b)\ln p(b|b)] = \\ -\frac{1}{2}[\frac{2}{3}\ln\frac{2}{3} + \frac{1}{3}\ln\frac{1}{3}] &- \frac{1}{2}[\frac{1}{3}\ln\frac{1}{3} + \frac{2}{3}\ln\frac{2}{3}] = 0.325 \text{ nats} = 0.469 \text{ bits} \end{aligned}$$

Example 3.14 Consider the following *mobility table* $N = (n_{jk})$, cross-classifying father's occupation (rows $j = 1, \ldots, 5$) by son's first full time civilian occupation (columns $k = 1, \ldots, 5$) for 19'912 U.S. men in 1973 in five categories: a = "upper nonmanual"; b = "lower nonmanual"; c = "upper manual"; d = "lower manual"; e = "farm" (source: Hout 1986, cited in Mirkin 1996).

$$N = \left(\begin{array}{c|cccccc} & a & b & c & d & e & \text{total} \\ \hline a & 1'414 & 521 & 302 & 643 & 40 & 2'920 \\ b & 724 & 524 & 254 & 703 & 48 & 2'253 \\ c & 798 & 648 & 856 & 1'676 & 108 & 4'086 \\ d & 756 & 914 & 771 & 3'325 & 237 & 6'003 \\ e & 409 & 357 & 441 & 1'611 & 1'832 & 4'650 \\ \hline \text{total} & 4'101 & 2'964 & 2'624 & 7'958 & 2'265 & 19'912 \end{array}\right) \tag{3.19}$$

Dividing each cell n_{jk} by its row total $n_{j\bullet}$ results in a transition matrix $P = (p_{jk})$ with $p_{jk} := n_{jk}/n_{j\bullet}$, giving the conditional probabilities for an individual (whose father has occupation j) to have first full time civilian occupation k:

$$P = \left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0.48 & 0.18 & 0.10 & 0.22 & 0.01 \\ b & 0.32 & 0.23 & 0.11 & 0.31 & 0.02 \\ c & 0.20 & 0.16 & 0.21 & 0.41 & 0.03 \\ d & 0.12 & 0.15 & 0.13 & 0.55 & 0.04 \\ e & 0.09 & 0.08 & 0.09 & 0.35 & 0.39 \end{array}\right) \tag{3.20}$$

The components of the stationary solution π associated to the transition matrix 3.20 are $\pi_a = 0.26$, $\pi_b = 0.17$, $\pi_c = 0.13$, $\pi_d = 0.40$ and $\pi_e = 0.04$. That is, under the fiction of a constant transition matrix 3.19, one will observe in the long run 26% of people in category a, 17% in category b, etc. The conditional entropies $H(X_n|X_{n-1} = j) = -\sum_k p_{jk}\log p_{jk}$, measuring the uncertainty on the son's occupation X_n (knowing

the father's occupation $X_{n-1} = j$) are (in bits):

$$H(X_n|X_{n-1} = a) = 1.85 \qquad H(X_n|X_{n-1} = b) = 2.01$$
$$H(X_n|X_{n-1} = c) = 2.02 \qquad H(X_n|X_{n-1} = d) = 1.83$$
$$H(X_n|X_{n-1} = e) = 1.95$$

Thus son's occupation is most uncertain when the father is upper manual (2.02 for $X_{n-1} = c$) and least uncertain when the father is lower manual (1.83 for $X_{n-1} = d$). On average, the uncertainty is

$$\sum_{j=1}^{5} \pi_j \, H(X_n|X_{n-1} = j) = 0.26 \times 1.85 + 0.17 \times 2.01 + 0.13 \times 2.02 + \\ +0.40 \times 1.83 + 0.04 \times 1.95 = 1.90 = h_\infty$$

which is nothing but the entropy rate of the process h_∞ in virtue of 3.18: as expected and by construction, the entropy rate of a Markov process measures the mean *conditional* uncertainty on the next state knowing the previous state. By contrast, the corresponding *unconditional* uncertainty is $-\sum_j \pi_j \log \pi_j = 2.05$ bits, which is larger than $h_\infty = 1.90$ but smaller than the uniform uncertainty $\log_2 5 = 2.32$ bits.

Control Question 36

True or false?

1. the entropy rate h_∞ of a first-order Markov chain can never exceed its unconditional entropy $H(X)$ **Possible answers:** true - false

2. the entropy rate h_∞ of a first-order Markov chain can never equal its unconditional entropy $H(X)$ **Possible answers:** true - false

3. the entropy rate h_∞ of a first-order Markov chain is not defined if the chain is not regular **Possible answers:** true - false

4. the entropy rate h_∞ associated to a chain with m categories can never exceed $\log m$. **Possible answers:** true - false

Answer

1. "true", since, for first-order Markov chains, $h_\infty = H(X_2|X_1) \leq H(X_2)$.

2. "false", since first-order Markov chains include the trivial case of independent sequences (= Markov chain of order 0), whose entropy rates h_∞ equal their unconditional entropies $H(X)$.

3. "true", since in general the stationary distribution π_j is not defined if the chain is not regular.

4. "true", since $h_\infty \leq H(X) \leq \log m$

3.3.5 Irreversibility

Consider a $(m \times m)$ transition matrix $P = (p_{jk})$ defining a regular Markov process, with associated stationary distribution π. Let $f_j^{(0)} \geq 0$ (obeying $\sum_{j=1}^m f_j^{(0)} = 1$) be the initial distribution ($t = 0$) of the system across its different possible states $j = 1, \ldots, m$. At time $t = 1$, the distribution becomes $f_k^{(1)} = \sum_{j=1}^m f_j^{(0)}\, p_{jk}$. More generally, the distribution $f^{(n)}$ at time $t = n$ is given by

$$f_k^{(n)} = \sum_{j=1}^{m} f_j^{(0)}\, p_{jk}^{(n)} \tag{3.21}$$

As $\lim_{n\to\infty} p_{jk}^{(n)} = \pi_k$ from theorem 3.7, formula 3.21 shows $\lim_{n\to\infty} f_k^{(n)} = \sum_{j=1}^m f_j^{(0)}\pi_k = \pi_k$. That is, *irrespectively of the profile of the initial distribution $f^{(0)}$, the long run distribution $f^{(n)}$ converges in the long run $n \to \infty$ towards the stationary distribution π.* One speaks of *equilibrium* if $f^{(0)} = \pi$. In summary, a general non-equilibrium profile $f^{(0)} \neq \pi$ evolves towards the equilibrium profile $f^{(\infty)} = \pi$ where it remains then unchanged, since $\pi_k = \sum_{j=1}^m \pi_j\, p_{jk}$ by virtue of 3.16.

Thus the Markov dynamics is *irreversible*: any initial distribution $f^{(0)}$ always evolves towards the (unique) stationary distribution π, and never the other way round; also the dissimilarity between any two distributions fades out during evolution, as the following theorem shows:

Theorem 3.9 *Let $f^{(n)}$ and $g^{(n)}$ be any two distributions whose evolution 3.21 is governed by the same regular Markov process $P = (p_{jk})$. Then evolution makes the two distributions increasingly similar (and increasingly similar with the stationary distribution π) in the sense*

$$K(f^{(n+1)}||g^{(n+1)}) \leq K(f^{(n)}||g^{(n)})$$

where $f^{(n+1)}$ and $g^{(n+1)}$ are the corresponding next time distributions, namely

$$f_k^{(n+1)} = \sum_{j=1}^{m} f_j^{(n)}\, p_{jk} \qquad\qquad g_k^{(n+1)} = \sum_{j=1}^{m} g_j^{(n)}\, p_{jk} \tag{3.22}$$

Particular cases:

$K(f^{(n+1)}||\pi) \leq K(f^{(n)}||\pi)$ (obtained with $g^{(n)} := \pi$, which implies $g^{(n+1)} = \pi$): the relative entropy of $f^{(n)}$ with respect to π decreases with n: again, $\lim_{n\to\infty} f^{(n)} = \pi$.

$K(\pi||g^{(n+1)}) \leq K(\pi||g^{(n)})$ (obtained with $f^{(n)} := \pi$, which implies $f^{(n+1)} = \pi$): the relative entropy of π with respect to $g^{(n)}$ decreases with n.

$K(f^{(n+1)}||f^{(n)}) \leq K(f^{(n)}||f^{(n-1)})$ (obtained with $g^{(n)} := f^{(n-1)}$, which implies $g^{(n+1)} = f^{(n)}$): the dissimilarity between the actual distribution and the previous one (as measured by $K(f^{(n)}||f^{(n-1)})$) decreases with n.

Example 3.15 (example 3.13, continued:) suppose the initial distribution $f^{(0)}$

to be $f^{(0)}(a) = 0.9$ and $f^{(0)}(b) = 0.1$. Using 3.21 and theorem 3.6, the successive distributions $f^{(n)}$ at time $t = n$ and their divergence $K(f^{(n)}||\pi)$ (in nats) with respect to the stationary distribution $\pi = f^{(\infty)}$ (with $\pi(a) = \pi(b) = 0.5$) are, in order,

n	$f^{(n)}(a)$	$f^{(n)}(b)$	$K(f^{(n)}\|\|\pi)$
0	0.9	0.1	0.420
1	0.633	0.367	0.036
2	0.544	0.456	0.004
3	0.515	0.485	0.0005
...	...	...	...
∞	0.5	0.5	0

Control Question 37

Determine the unique correct answer:

1. once leaving a state, the system will return to it with probability one if the state is a) transient; b) absorbing; c) recurrent; d) aperiodic.

2. the identity matrix I is the transition matrix of a Markov chain, all states of which are a) transient; b) absorbing; c) irreversible; d) aperiodic.

3. let P be a two-by-two transition matrix. The minimal number of non-zero components of P insuring the regularity of the associated Markov chain is a) 1; b) 2; c) 3; d) 4.

4. suppose $P^5 = I$, where P is a finite Markov transition matrix and I the identity matrix. Then P is a) undetermined; b) regular; c) I; d) aperiodic.

Answer

1. c) recurrent

2. b) absorbing

3. c) 3

4. c) I

3.4 Markov chains of general order

Learning Objectives for Section 3.4

After studying this section and the following one, you should

- be able to generalize the concepts of the previous section to Markov chains of arbitrary order r
- understand the theoretical foundations of the test of the order of the chain and be able to apply it
- be familiar with the concept of over-parameterization and of its consequences in text simulation

Definition 3.9 A *Markov chain of order* $r > 0$ is a discrete stochastic process whose memory is limited to the r past states, that is:

$$p(x_n|x_1^{n-1}) = p(x_n|x_{n-r}^{n-1}) \qquad \forall n \geq r+1$$

A Markov chain of order $r = 0$, also called *Bernoulli* process, is a stochastic process without memory, that is

$$p(x_n|x_1^{n-1}) = p(x_n) \qquad \forall n \geq 1$$

A zero-order Markov chain is then plainly an independent process. All Markov processes considered here are stationary, that is their *transition probabilities* $p(x_n|x_{n-r}^{n-1})$ are time independent. The latter are generically denoted as $p(\omega|\alpha)$ where $\omega \in \Omega$ is the present state and $\alpha \in \Omega^r$ specifies the r immediately anterior states. By construction, $p(\omega|\alpha) := p(\alpha\omega)/p(\alpha) \geq 0$ with $\sum_{\omega\in\Omega} p(\omega|\alpha) = 1$. ♦

3.4.1 Stationary distribution and entropy rate

An r-th order Markov chain on Ω, defined by the transitions $p(\omega|\alpha)$ where $\omega \in \Omega$ and $\alpha \in \Omega^r$, can *formally* be considered as a *first-order Markov chain on* Ω^r with a $(m^r \times m^r)$ transition probability matrix $Q = (q_{\alpha\beta})$ (where $\alpha = \alpha_1^r \in \Omega^r$ and $\beta = \beta_1^r \in \Omega^r$) given by

$$q_{\alpha\beta} = q(\beta|\alpha) := \delta_{\beta_1\alpha_2}\, \delta_{\beta_2\alpha_3} \ldots \delta_{\beta_{r-1}\alpha_r}\, p(\beta_r|\alpha_1^r) \tag{3.23}$$

where $\delta_{ab} := 1$ if $a = b$ and $\delta_{ab} := 0$ if $a \neq b$. Equation 3.23 tells that the transition $\alpha \to \beta$ is possible iff the $r-1$ first elementary states of β correspond to the $r-1$ last elementary states of α (see figure 3.7)

Supposing in addition the chain $Q = (q_{\alpha\beta})$ to be regular (i.e. each state of Ω^r communicates with each state of Ω^r and the chain is aperiodic), there is a unique stationary distribution $\pi(\alpha) = \pi(\alpha_1^r)$ on Ω^r satisfying 3.16 on Ω^r, that is, using 3.23:

$$\sum_{\alpha_1\in\Omega} \pi(\alpha_1, \alpha_2, \ldots, \alpha_r)\, p(\alpha_{r+1}|\alpha_1, \alpha_2, \ldots, \alpha_r) = \pi(\alpha_2, \ldots, \alpha_r, \alpha_{r+1}) \tag{3.24}$$

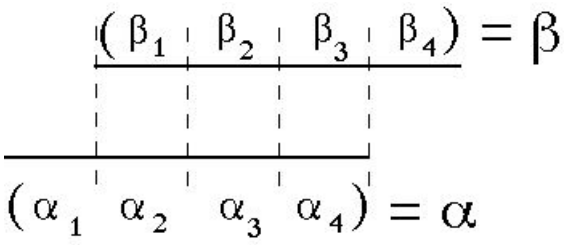


Figure 3.7: A Markov chain of order r (here $k = 4$) on Ω is specified by the set of conditional probabilities of the form $p(\beta_4|\alpha_1\alpha_2\alpha_3\alpha_4)$. The same chain can be considered as a first-order Markov chain $q_{\alpha\beta}$ on Ω^r where $\alpha = (\alpha_1\alpha_2\alpha_3\alpha_4)$ and $\beta = (\beta_1\beta_2\beta_3\beta_4)$; as expressed by 3.23, the transition matrix $q_{\alpha\beta}$ is zero unless $\beta_1 = \alpha_2$, $\beta_2 = \alpha_3$ and $\beta_3 = \alpha_4$.

Similarly, 3.18 shows the corresponding entropy rate h_∞ to be $h_\infty = -\sum_\alpha \pi(\alpha) \sum_\beta q_{\alpha\beta} \log q_{\alpha\beta}$, or, using 3.23

$$h = -\sum_{\alpha\in\Omega^r} \pi(\alpha) \sum_{\omega\in\Omega} p(\omega|\alpha)\ \log p(\omega|\alpha) \tag{3.25}$$

Recall in general the conditional entropy $h_k := H(X_k|X_1^{k-1})$ to be non-increasing in k. On the other hand, 3.25 shows the entropy rate to coincide with h_{r+1}. In conclusion:

Theorem 3.10 *For a r-th order Markov chain, $h = h_{r+1}$. That is*

$$h_1 \geq h_2 \geq h_r \geq h_{r+1} = h_{r+2} = h_{r+3} = \ldots = h_\infty \tag{3.26}$$

The behavior 3.26 of h_k is illustrated in figure 3.10 b) for $r = 1$ and in figure 3.10 c) for $r = 3$.

Particular cases:

r = 1: the entropy rate becomes

$$h_\infty = -\sum_{\alpha\in\Omega} \pi(\alpha) \sum_{\omega\in\Omega} p(\omega|\alpha)\ \log p(\omega|\alpha) = -\sum_j \pi_j \sum_k p_{jk}\ \log p_{jk}$$

which is the same expression as 3.18.

r = 0: the entropy rate becomes

$$h_\infty = -\sum_{\omega\in\Omega} p(\omega)\ \log p(\omega) = -\sum_k \pi_k\ \log \pi_k$$

which is the entropy of the corresponding stationary distribution.

Example 3.16 Consider (figure 3.8) the Markov chain of order $r = 2$ on $m = 2$ states $\Omega = \{a, b\}$, with

$$p(a|aa) = 0.3 \quad p(b|aa) = 0.7 \quad p(a|ab) = 0.6 \quad p(b|ab) = 0.4$$
$$p(a|ba) = 0.7 \quad p(b|ba) = 0.3 \quad p(a|bb) = 0.4 \quad p(b|bb) = 0.6$$

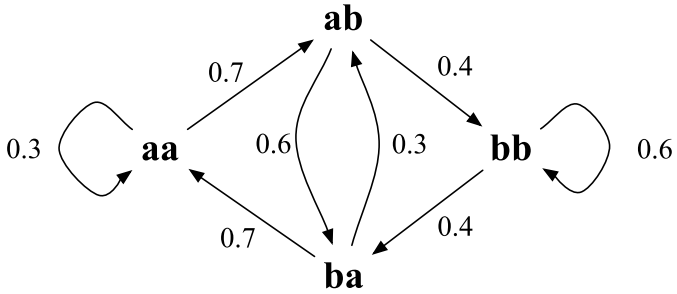


Figure 3.8: A second-order Markov chain $p(\omega|\alpha)$ on $\Omega = \{a, b\}$ (example 3.16) written as a first order chain $q_{\alpha\beta}$ on bigrams $\alpha = \alpha_1\alpha_2 \in \Omega^2$ and $\beta = \beta_1\beta_2 \in \Omega^2$. Transitions are forbiden if $\alpha_2 \neq \beta_1$: for instance, transition form $\alpha = ab$ to $\beta = aa$ is impossible.

By symmetry, the stationary distribution $\pi(\alpha_1, \alpha_2)$ obeying 3.24 turns out to be the uniform distribution on Ω^2, that is $\pi(aa) = \pi(ab) = \pi(ba) = p(bb) = \frac{1}{4}$. For instance, one verifies the following equality to hold

$$\pi(aa)\, p(a|aa) + \pi(ba)\, p(a|ba) = \frac{1}{4}\, 0.3 + \frac{1}{4}\, 0.7 = \frac{1}{4} = \pi(aa)$$

as well as other equalities involved in 3.24. The entropy rate 3.25 is

$$\begin{aligned} h_\infty = &-\pi(aa)\,[p(a|aa)\log p(a|aa) + p(b|aa)\log p(b|aa)] \\ &-\pi(ab)\,[p(a|ab)\log p(a|ab) + p(b|ab)\log p(b|ab)] \\ &-\pi(ba)\,[p(a|ba)\log p(a|ba) + p(b|ba)\log p(b|ba)] \\ &-\pi(bb)\,[p(a|bb)\log p(a|bb) + p(b|bb)\log p(b|b)] = \\ -\frac{1}{2}\,&[0.3\log 0.3 + 0.7\log 0.7 + 0.6\log 0.6 + 0.4\log 0.4] = 0.189 \text{ nats} \end{aligned}$$

3.5 Reconstruction of Markov models from data

Up to now, we have assumed the diverse models of interest (stationary, Markov of order r, Markov of of order 1, etc.) to be *given*. Very often, however, we only have at disposal an empirical realization of a process, i.e. only the data D are known, and models M must be inferred form data D. This kind of situation is paradigmatic of *inferential statistics* (see module S1). For clarity sake, empirical (respectively model) quantities will be indexed from now one by the letter D (respectively M).

3.5.1 Empirical and model distributions

A sequence of k consecutive states $x_l^{l+r-1} \in \Omega^k$ is called a *k-gram*. Given the k-gram $\beta \in \Omega^k$ and the l-gram $\gamma \in \Omega^l$, the $k+l$-gram obtained by concatenating γ to the right of β is simply denoted by $\beta\gamma$. The length of a subsequence α is simply denoted as $|\alpha|$: for instance, $k = |\beta|$ and $l = |\gamma|$.

Data D consist of x_1^n, the complete observed sequence of size n. Let $n(\beta)$ be the empirical count of k-gram $\beta \in \Omega^k$, that is the number of times subsequence β occurs

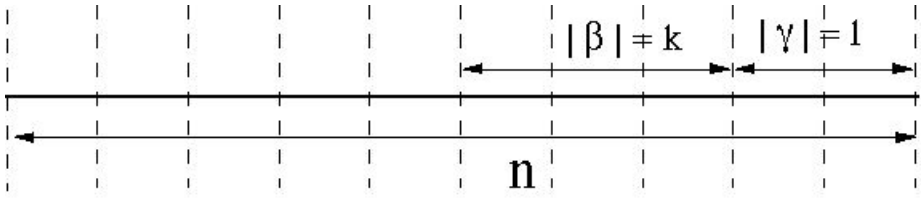


Figure 3.9: The identity $\sum_{\gamma\in\Omega^l} n(\beta\gamma) = n(\beta)$ holds iff there is no occurence of a symbol of β in the l last symbols of x_1^n.

in x_1^n (for instance, the bigram $\beta = 11$ is contained $n(\beta) = 3$ times in the sequence $x_1^7 = 0111011$). The number of all k-grams contained in a sequence of length n is (for $n \geq k$)

$$\sum_{\beta\in\Omega^k} n(\beta) = n - k + 1$$

Also, one has

$$\sum_{\gamma\in\Omega^l} n(\beta\gamma) \leq n(\beta)$$

where identity holds iff data x_1^n do not contain occurences of elements of β closer than l places from the right boundary (see figure 3.9).

The empirical distribution and empirical conditional distribution are defined as

$$f^D(\beta) := \frac{n(\beta)}{n-k+1} \quad f^D(\gamma|\beta) := \frac{n(\beta\gamma)}{\sum_{\gamma'\in\Omega^l} n(\beta\gamma')} \quad \beta \in \Omega^k \quad \gamma \in \Omega^l \tag{3.27}$$

where the denominators insure proper normalization, namely $\sum_{\beta\in\Omega^k} f^D(\beta) = 1$ and $\sum_{\gamma\in\Omega^l} f^D(\gamma|\beta) = 1$. Asymptotically, that is for n large, one has approximatively $n - k + 1 \cong n$ and $\sum_{\gamma\in\Omega^l} n(\beta\gamma) \cong n(\beta)$, and thus

$$f^D(\beta) \cong \frac{n(\beta)}{n} \qquad f^D(\gamma|\beta) \cong \frac{n(\beta\gamma)}{n(\beta)} \qquad \beta \in \Omega^k \qquad \gamma \in \Omega^l \tag{3.28}$$

To emphasize the contrast with empirical distributions, the corresponding theoretical distributions will from now on be denoted as $f^M(\beta)$ and $f^M(\gamma|\beta)$ with

$$f^M(\beta) := p(\beta) \qquad f^M(\gamma|\beta) := \frac{p(\beta\gamma)}{p(\beta)}$$

where $p(\ldots)$ is the consistent probability measure defined in section 3.

Example 3.17 The l-th *Thue-Morse sequence* D^l is a binary sequence recursively constructed as follows:

$$D^0 = 1 \qquad D^{l+1} = D^l \cdot \bar{D}^l$$

where $\cdot$ denotes concatenation and $\bar{D}$ binary inversion, replacing each symbol of D (in order) by its complement, namely $\bar{1} = 0$ and $\bar{0} = 1$. The first Thue-Morse sequences are

l	Thue-Morse sequence D^l
0	1
1	10
2	1001
3	10010110
4	1001011001101001
5	10010110011010010110100110010110
6	1001011001101001011010011001011001101001011001101001

In general, the l-th sequence D^l contains 2^l binary symbols in equal proportion (for $l \geq 1$). D^l can also be obtained by applying l times the substitution rule $1 \to 10$ and $0 \to 01$ to the initial sequence $D^0 = 1$. Also, the odd entries of D^{l+1} reproduce D^l.

Although purely deterministic, the sequences D^l can be used to define empirical distributions $f^D(\beta)$ and conditional empirical distributions $f^D(\gamma|\beta)$. For instance, for $D = D^6$, one finds

symbol	1	0	total
relative empirical frequency	$\frac{1}{2}$	$\frac{1}{2}$	1

bigram	11	10	01	00	total
relative empirical frequency	$\frac{10}{63} \cong \frac{1}{6}$	$\frac{21}{63} = \frac{1}{3}$	$\frac{21}{63} = \frac{1}{3}$	$\frac{11}{63} \cong \frac{1}{6}$	1

trigram	111	110	101	100	011	010	001	000	total
rel. emp. frequency	0	$\frac{10}{62} \cong \frac{1}{6}$	$\frac{10}{62} \cong \frac{1}{6}$	$\frac{11}{62} \cong \frac{1}{6}$	$\frac{10}{62} \cong \frac{1}{6}$	$\frac{10}{62} \cong \frac{1}{6}$	$\frac{11}{62} \cong \frac{1}{6}$	0	1

conditional symbol	1|11	0|11	total	1|10	0|10	total
relative empirical frequency	0	1	1	$\frac{10}{21} \cong \frac{1}{2}$	$\frac{11}{21} \cong \frac{1}{2}$	1

conditional symbol	1|01	0|01	total	1|00	0|00	total
relative empirical frequency	$\frac{1}{2}$	$\frac{1}{2}$	1	1	0	1

The behavior of h_k for $k \geq 1$ for the sequence D^{14} (containing 16'384 binary symbols) is depicted in figure 3.10.

The values of the empirical distributions $f^D(\beta)$ and conditional distributions $f^D(\gamma|\beta)$ (such as those as found in example 3.17) can serve to define *model* or *theoretical* distributions $f^M(\beta)$ and $f^M(\gamma|\beta)$. New stochastic sequences $\tilde{D}$ can in turn be simulated from the Markov chains with parameters $f^M(\beta)$ and $f^M(\gamma|\beta)$. By construction, the statistical properties of the resulting sequence $\tilde{D}$ will look similar to those of the "training sequence" D:

3.5.2 The formula of types for Markov chains

Consider a r-th order Markov chain defined by the conditional distribution $f^M(\omega|\alpha)$ where $\omega^{\in}\Omega$ and $\alpha \in \Omega^r$. The probability to observe data x_1^n is

$$P(x_1^n) = p(x_1^r) \prod_{i=r+1}^{n} p(x_i|x_{i-r}^{i-1}) = p(x_1^r) \prod_{\alpha\in\Omega^r} \prod_{\omega\in\Omega} f^M(\omega|\alpha)^{n(\alpha\omega)}$$

For r fixed and for $n \to \infty$, the contribution of the term $p(x_1^r)$ becomes negligible relatively to the contribution of the product. Thus, *asymptotically*, that is for n large, the "boundary effects" caused by the finitude of n, and occuring at the very beginning or the very end of the sequence x_1^n, become negligible and one can write approximatively

$$P(x_1^n) \cong \prod_{\alpha\in\Omega^r} \prod_{\omega\in\Omega} f^M(\omega|\alpha)^{n(\alpha\omega)} \tag{3.29}$$

In the same approximation (compare with 3.27) one has

$$f^D(\alpha) \cong \frac{n(\alpha)}{n} \qquad f^D(\omega|\alpha) \cong \frac{n(\alpha\omega)}{n(\alpha)} \qquad \alpha \in \Omega^r \qquad \omega \in \Omega \tag{3.30}$$

Intuitively, one expects that, for n large, the empirical distribution $f^D(\omega|\alpha)$ tends to $f^M(\omega|\alpha)$ with fluctuations around this value. The next theorem (where entropies are expressed in *nats* for convenience) shows this to be indeed the case; morevover, the fluctuation of the empirical values around the theoretical ones are controlled by the *conditional Kullback-Leibler divergence* $K_r(f^D||f^M)$ *of order* r:

Theorem 3.11 (formula of types for Markov chains) *For Markov chains of order r, the probability to observe the conditional empirical distribution $f^D(\omega|\alpha)$ (for all $\omega \in \Omega$ and $\alpha \in \Omega^r$) is, asymptotically*

$$P(f^D \mid f^M) \approxeq \exp(-n\, K_r(f^D||f^M)) \tag{3.31}$$

where

$$K_r(f^D||f^M) := \sum_{\alpha\in\Omega^r} f^D(\alpha)\, K([f^D||f^M]|\alpha) \tag{3.32}$$

$$K([f^D||f^M]|\alpha) := \sum_{\omega\in\Omega} f^D(\omega|\alpha) \ln \frac{f^D(\omega|\alpha)}{f^M(\omega|\alpha)} \tag{3.33}$$

In particular, for given counts $n(\alpha\omega)$ (and thus given $f^D(\alpha)$ and $f^D(\omega|\alpha)$), the probability 3.31 is maximum iff $\tilde{f}^M(\omega|\alpha) = f^D(\omega|\alpha)$: as expected, the *maximum-likelihood* estimate of the model M is simply given by the corresponding empirical quantity (see module S1 for a more detailed exposition for the independent case).

Remark: $K_0(f^D||f^M)$ is the ordinary (unconditional) divergence:

$$K_0(f^D||f^M) = K(f^D||f^M) = \sum_{\omega\in\Omega} f^D(\omega)\, \log \frac{f^D(\omega)}{f^M(\omega)}$$

PROOF Recall that $a_n \approxeq b_n$ means $\lim_{n\to\infty} \frac{1}{n}\log(a_n/b_n) = 0$. For instance, Stirling's approximation is $n! \approxeq n^n \exp(-n)$.

The value of $f^D(\omega|\alpha)$ is the same for all $n(\alpha)!/(\prod_{\omega\in\Omega} n(\alpha\omega)!)$ data x_1^n possessing the same counts $n(\alpha\omega)$ but differing by the occurence order. Thus

$$P(f^D \mid f^M) \cong \prod_{\alpha\in\Omega^r} n(\alpha)! \prod_{\omega\in\Omega} \frac{1}{n(\alpha\omega)!}\, f^M(\omega|\alpha)^{n(\alpha\omega)}$$

Taking the logarithm, using 3.30 and Stirling's approximation yields

$$\ln P(f^D \mid f^M) \cong \sum_{\alpha\in\Omega^r}\sum_{\omega\in\Omega} \ln\left[\frac{n(\alpha)^{n(\alpha\omega)}}{n(\alpha\omega)^{n(\alpha\omega)}}\, f^M(\omega|\alpha)^{n(\alpha\omega)}\right] =$$

$$= \sum_{\alpha\in\Omega^r}\sum_{\omega\in\Omega} n(\alpha\omega)\ln\frac{f^M(\omega|\alpha)}{f^D(\omega|\alpha)} = -n\sum_{\alpha\in\Omega^r} f^D(\alpha)\sum_{\omega\in\Omega} f^D(\omega|\alpha)\ln\frac{f^D(\omega|\alpha)}{f^M(\omega|\alpha)}$$

Example 3.18 Consider a first order Markov chain with two states a and b. When in a given state (a or b), the system remains in the same state with probability 0.9, and changes with probability 0.1. That is, $f^M(a|a) = 0.9$, $f^M(b|a) = 0.1$, $f^M(b|b) = 0.9$ and $f^M(a|b) = 0.1$. Suppose data to be of the form

$$D \equiv \underbrace{aaaa\cdots aaaa}_{n \text{ times}}$$

Then $f^D(a) = 1$, $f^D(b) = 0$, $f^D(a|a) = 1$ and $f^D(b|a) = 0$. Then

$$K([f^D||f^M]|a) = f^D(a|a)\ln\frac{f^D(a|a)}{f^M(a|a)} + f^D(b|a)\ln\frac{f^D(b|a)}{f^M(b|a)} =$$
$$= 1\ln\frac{1}{0.9} + 0\ln\frac{0}{0.1} = 0.105 \text{ nats}$$

On the other hand, neither $f^D(a|b)$ nor $f^D(b|b)$ are defined, since the system has never been observed in state b: equations 3.27 or 3.28 return the undetermined value 0/0 (assumed finite). Thus $K([f^D||f^M]|b)$ is not defined, but $K_1(f^D||f^M)$ is:

$$K_1(f^D||f^M) = f^D(a)\, K([f^D||f^M]|a) + f^D(b)\, K([f^D||f^M]|b) =$$
$$= 1 \times 2.30 + 0 \times K([f^D||f^M]|b) = 0.105 \text{ nats}$$

Thus

$$P(f^D \mid f^M) \approxeq \exp(-n\, 0.105) = (0.9)^n$$

For instance, the probability of observing the sequence $aaaaaaaaaa$ under the model ($n = 10$) is approximatively $(0.9)^{10} = 0.35$ (the formula is exact up to the initial term $P(X_1 = a)$, already neglected in 3.29); the probability of observing the sequence $aaaaaaaaaaaaaaaaaaaa$ ($n = 20$) is $(0.9)^{20} = 0.12$, etc.

Example 3.19 (example 3.18, continued) By symmetry, the stationary proba-

bility associated to the chain is $\pi(a) = \pi(b) = 0.5$, and the entropy rate is

$$h_\infty = -\pi(a)[f^M(a|a)\ln f^M(a|a) + f^M(b|a)\ln f^M(b|a)] - \pi(b)[f^M(a|b)\cdot \\ \cdot \ln f^M(a|b) + f^M(b|b)\ln f^M(b|b)] = -0.9\ln 0.9 - 0.1\ln 0.1 = 0.325 \text{ nats}$$

Thus the size of the typical sequences set grows as $|T_n(\varepsilon)| \cong \exp(0.325 \cdot n) = (1.38)^n$, instead of 2^n for a maximally random (= independent + uniform) process. Otherwise said, the dynamics of the system under investigation behaves as if only 1.38 effective choices were at disposal at each step, instead of 2 effective choices (namely a and b) for the maximally random dynamics.

3.5.3 Maximum likelihood and the curse of dimensionality

Suppose one observes a sequence $x_1^n \in \Omega^n$ of length n with $m := |\Omega|$ states, believed to be generated by a Markov chain of order r. The complete specification a the latter model necessitates to determine all the quantitites of the form $f^M(\omega|\alpha)$ for all $\omega \in \Omega$ and $\alpha \in \Omega^r$, that is a total of $m^r(m-1)$ quantities (the quantities $f^M(\omega|\alpha)$ are not completely free, but must obey the m^r constraints $\sum_{\omega\in\Omega} f^M(\omega|\alpha) = 1$ for all $\alpha \in \Omega^r$, whence the factor $m-1$).

But, even for relatively modest values of m and r, the number of free parameters $m^r(m-1)$ grows *very fast* (for instance 48 free parameters for $m = 4$ and $r = 2$, or 54 free parameters for $m = 3$ and $r = 3$). In consequence, the amount of data D required to estimate all those parameters with a reasonably small error becomes very large! This phenomenon, sometimes referred to as the *curse of dimensionality*, constitutes a major drawback, severely restricting the use of Markov chain modelling for growing r, despite the generality and flexibility of the latter.

Concretely, consider the *maximum likelihood estimation*, consisting in estimating $f^M(\omega|\alpha)$ as the value $\tilde{f}^M(\omega|\alpha)$ maximizing $P(f^D|f^M)$ for given f^D. The formula of types 3.31 then demonstrates the searched for estimate $\hat{f}^M(\omega|\alpha)$ to be simply given by the corresponding empirical distribution $\tilde{f}^M(\omega|\alpha) = f^D(\omega|\alpha)$. But a sequence $D = x_1^n$ of length n contains a maximum of $n - r$ distinct transitions $\alpha \to \omega$, and if m or r are large enough, the majority of the theoretically observed transitions will simply not occur in $D = x_1^n$, and the corresponding maximum likelihood estimates will be given the value $\hat{f}^M(\omega|\alpha) = 0$, even if $f^M(\omega|\alpha) > 0$.

This problem of *unobserved transitions* occurs each time the number of possible states m as well as the order of the chain r are large in comparison of the sample size n. Different remedies have been proposed to allevy the problem, such as the "trigram strategy" consiting in estimating $f^M(\omega_3|\omega_1\omega_2)$ (for a Markov chain of order $r = 2$) as

$$\hat{f}^M(\omega_3|\omega_1\omega_2) = \lambda_0 f^D(\omega_3) + \lambda_1 f^D(\omega_3|\omega_2) + \lambda_2 f^D(\omega_3|\omega_1\omega_2)$$

where the optimal choice of the non-negative weights λ_0, λ_1 and λ_2, obeying $\lambda_0 + \lambda_1 + \lambda_2 = 1$, is typically determined by trial and error, aiming at maximizing some overall performance index relatively to a given application.

Although sometimes satisfactory for a given practical application, such estimates lack theoretical foundation and formal justification. This situation is somewhat analogous to the problem of *unobserved species*, occuring each time the number of possible states m is so large in comparison to the sample size n that some states might have not

been observed at all in the data D. Although well identified in textual and biological data for a while, this problem has nevertheless not received a simple and universally aggreed upon solution; see however module L1 for a strategy aimed at estimating the total number of possible (= observed + unobserved) states.

3.5.4 Testing the order of a Markov chain

Denote by $f^D(\ldots)$ be the empirical distribution function determined from data $D = x_1^n \in \Omega^n$, and denote by $f^M(\ldots) \equiv p(\ldots)$ its theoretical counterpart, relatively to some model M. The corresponding empirical, respectively theoretical conditional entropies h_k introduced in section 3.1 are

$$h_k^M = H(X_1^k) - H(X_1^{k-1}) = -\sum_{\alpha\in\Omega^{k-1}} f^M(\alpha) \sum_{\omega\in\Omega} f^M(\omega|\alpha) \ln f^M(\omega|\alpha)$$

$$h_k^D := -\sum_{\alpha\in\Omega^{k-1}} f^D(\alpha) \sum_{\omega\in\Omega} f^D(\omega|\alpha) \ln f^D(\omega|\alpha)$$

Suppose the model M to be a Markov chain of order r. Then theorem 3.10 implies

$$h_1^M \geq h_2^M \geq h_r^M \geq h_{r+1}^M = h_{r+2}^M = h_{r+3}^M = \ldots = h_\infty^M = h^M$$

Euqivalently, the quantity d_k^M defined for $k \geq 1$ by

$$d_k^M := h_k^M - h_{k+1}^M = 2H(X_1^k) - H(X_1^{k-1}) - H(X_1^{k+1})$$

obey, for a Markov chain of order r, the following:

$$d_1^M \geq 0 \qquad d_2^M \geq 0 \qquad d_r^M \geq 0 \qquad d_{r+1}^M = d_{r+2}^M = \ldots d_\infty^M = 0$$

Thus the greatest k for which d_k^M is strictly positive indicates the order $r = k$ of the Markov chain M. Intuitively, d_k^M measures the uncertainty reduction when the last symbol of a sequence is predicted using a past of length k instead of $k-1$, whence $d_k^M = 0$ if $k > r$.

As with any inferential problem in statistics, the difficulty is that the conditional entropies h_k^M are theoretical quantities defined by $(m-1)m^k$ parameters, not directly observable. But, if the data $D = x_1^n$ are numerous enough, one then expects h_k^D to be close to h_k^M, for k small. Also, the ratio "number of parameters to be estimated/number of data" is small iff k is small relatively to $\frac{\log n}{\log m}$. Thus empirical estimates are good as long as $k \leq k_{\max}$, where $k_{\max}$ is of order of $\frac{\log n}{\log m}$. Figures 3.10 and 3.11 suggest that

$$k_{\max} := \frac{1}{2}\,\frac{\log n}{\log m} \tag{3.34}$$

is a satisfactory pragmatic choice.

Hence, for $k \leq k_{\max}$, large values of d_k^D constitute an evidence pointing towards a model of order k, as confirmed by the following maximum likelihood test:

The order test

Consider a Markov process on $|\Omega| = m$ states, observed n successive times, about which, for some $k \leq k_{\max}$, two hypotheses compete, namely:

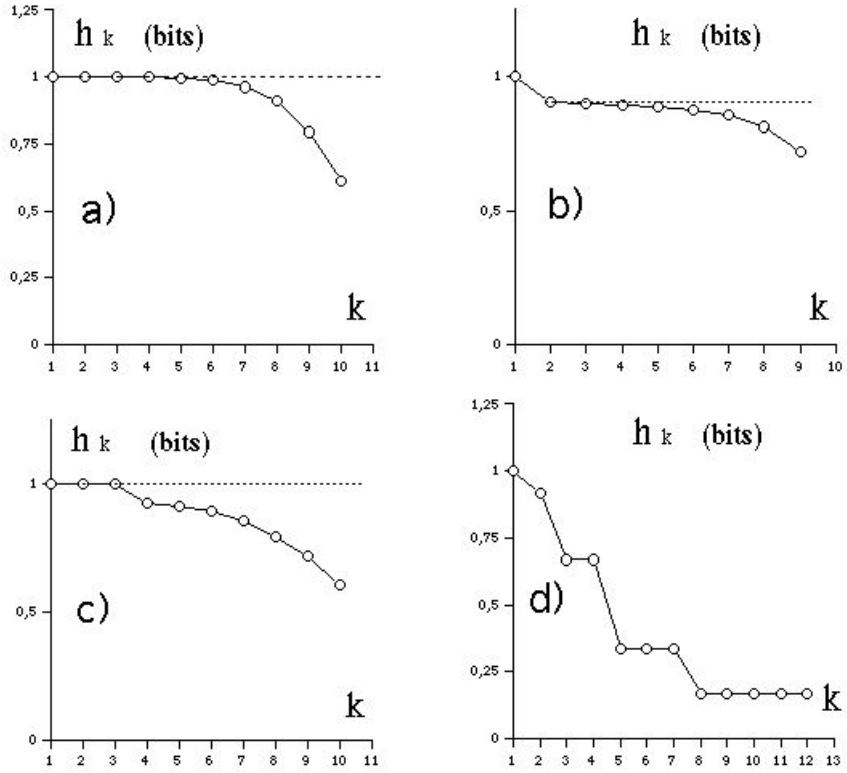


Figure 3.10: Observed values h_k^D (continuous line) and theoretical values h_k^M (dotted line) in terms of k for different models. In a), b) and c), empirical maximum likelihood estimates h_k^D coincide approximatively with theoretical values h_k^M as far as $k \leq k_{\max} = \frac{1}{2}\frac{\log n}{\log m}$. Estimates with $k > k_{\max}$ are not reliable due to the proliferation of unobserved transitions. **a)**: uniform and independent process (fair heads or tails) of length $n = 1024$ on $m = 2$ states. **b)**: Markov chain of order $r = 1$ of length $n = 1024$ on $m = 2$ states (see example 3.13). **c)**: Markov chain of order $r = 3$ of length $n = 1024$ on $m = 2$ states (see example 3.20). **d)**: the figure depicts the empirical values h_k^D obtained from the 14-th Thue-Morse sequence D^{14}, of length $n = 2^{14}$ on $m = 2$ states (see example 3.17).

H_0^k : : "the process is governed by a Markov chain of order $k-1$"

H_1^k : "the process is governed by a Markov chain of (strict) order k"

Then H_0^k is rejected at level α if

$$2n\, d_k^D = 2n\, [h_k^D - h_{k+1}^D] \geq \chi^2_{1-\alpha}[(m-1)^2 m^{k-1}] \tag{3.35}$$

where d_k^D is measured in nats.

The test can be applied in succession for $k = 1, 2, ... \leq k_{\max} := \frac{1}{2}\frac{\ln n}{\ln m}$. Potential k-th order candidate models are signalized by high values of d_k^D. For instance, if all d_k are small, an independence model can be considered (see figure 3.11). If all d_k are large, each $k+1$-th model beats the immediately inferior k-order model. Figure 3.11 shows the order r of the chain to be signalized by a peak at d_k^D (for $k \leq k_{\max}$).

Example 3.20 Let M be a binary Markov chain of order 3 specified by

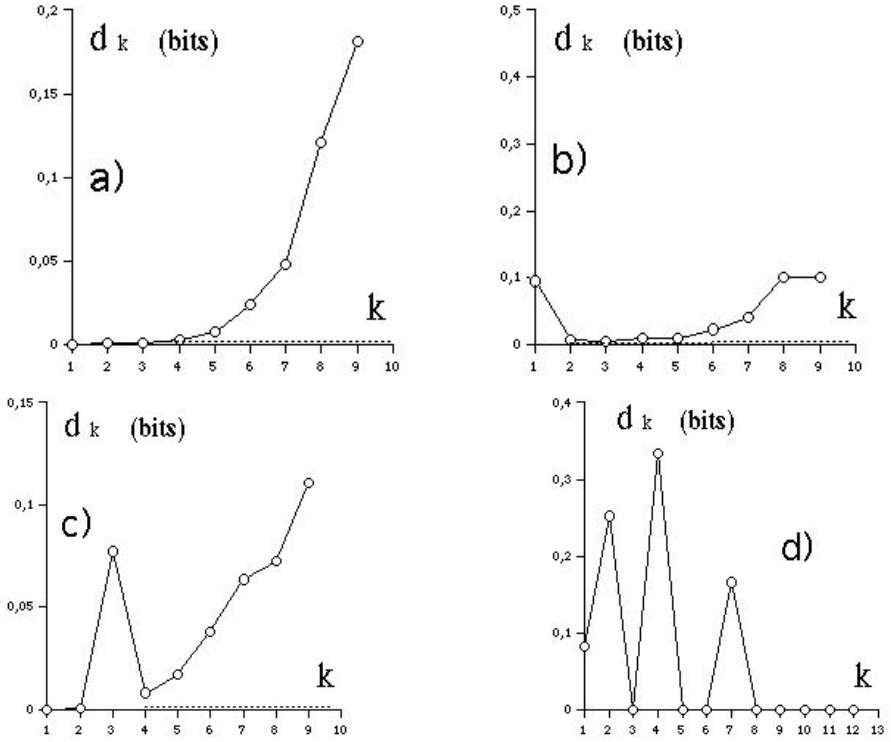


Figure 3.11: Observed (line) and expected (dots) values of $d_k = h_k - h_{k+1}$ in terms of k for the models presented in figure 3.10.

$f^M(\omega_4|\omega_1\omega_2\omega_3)$ on $\Omega = \{a, b\}$. A sequence of length $n = 1024$ is generated form this model, from which empirical distributions f^D are determined, and conditional entropies h_k^D are computed.

The values of h_k^D and d_k^D as well as the threshold $\chi^2_{1-\alpha}[\mathrm{df}]$ with $\mathrm{df} = (m-1)^2 m^{k-1}$ at the significance level $\alpha = 0.001$ are:

k	h_k^D	d_k^D	$2n\, d_k^D$	df	$\chi^2_{0.999}[\mathrm{df}]$
1	0.692	0	0	1	10.8
2	0.692	0.001	2.05	2	13.8
3	0.691	0.054	110.59	4	16.3
4	0.637	0.006	12.29	8	18.
$5 = k_{\max}$	0.631	0.0088	18.022	16	20.5

PROOF Likelihood ratio strategies, shown by Neyman and Pearson to minimize errors of both kinds (see module S1), command to take as decision variable the ratio of probabilities corresponding to both models, or equivalently their logarithmic likelihood ratio LL

$$\frac{P(f^D|\tilde{f}_k)}{P(f^D|\tilde{f}_{k-1})} \qquad \mathrm{LL} := \log \frac{P(f^D|\tilde{f}_k)}{P(f^D|\tilde{f}_{k-1})}$$

where $\tilde{f}_k$ is the best (in the ML-sense) modelling of f^D by a Markov chain of order k, and $\tilde{f}_{k-1}$ is the best model for f^D of order $k-1$. The former is more general than the

latter and thus bound yield a better fit, that is $P(f^D|\tilde{f}_k) \geq P(f^D|\tilde{f}_{k-1})$, i.e. $\text{LL} \geq 0$. On the other hand, formula 3.31 of types $P(f^D \mid f^M) \cong \exp(-n\, K_r(f^D||f^M))$ yields

$$\text{LL} = n\,[K_{k-1}(f^D||\tilde{f}_{k-1}) - K_k(f^D||\tilde{f}_k)] \geq 0$$

Finally, Kullback-Leibler conditional divergences $K_r(f^D||\tilde{f}_r)$ are, for n large, well represented by their quadratic approximations, the "computed chi-squares" $\frac{1}{2\,n}\chi^2(f^D||\tilde{f}_r)$. Remarkably, the latter have been shown to behave, under model $\tilde{f}_r$, as a chi-square distribution $\chi^2[\text{df}]$, where df is equal to the number of free parameters assigned to models of order r. Also, the difference $2n[K_{k-1}(f^D||\tilde{f}_{k-1}) - K_k(f^D||\tilde{f}_k)]$, caused by fluctuations of f^D accountable by $\tilde{f}_k$ but not by $\tilde{f}_{k-1}$, has been shown to behave as $\chi^2[\text{df}]$ with df equal to the difference in the number of free parameters between the two models.

Gluing the pieces together yields the following inferential recipy "reject model $\tilde{f}_{k-1}$ and accepts model $\tilde{f}_k$ at level α if $2n\,[K_{k-1}(f^D||\tilde{f}_{k-1}) - K_k(f^D||\tilde{f}_k)] \geq \chi^2_{1-\alpha}[\text{df}]$" where df is here $(m-1)\,m^k - (m-1)\,m^{k-1} = (m-1)^2\,m^{k-1}$.

The proof is complete if we finally show that $K_{k-1}(f^D||\tilde{f}_{k-1}) = h_k^D$ (and that $K_k(f^D||\tilde{f}_k) = h_{k+1}^D$). But it follows from 3.2 that

$$\begin{aligned} h_k^D = H(X_1^k) - H(X_1^{k-1}) &= -\sum_{\beta\in\Omega^k} f^D(\beta)\,\log f^D(\beta) + \sum_{\alpha\in\Omega^{k-1}} f^D(\alpha)\,\log f^D(\alpha) \\ &\overset{(a)}{=} -\sum_{\omega\in\Omega}\sum_{\alpha\in\Omega^{k-1}} f^D(\alpha\omega)\,\log f^D(\alpha\omega) + \sum_{\alpha\in\Omega^{k-1}} f^D(\alpha)\,\log f^D(\alpha) \\ &\overset{(b)}{=} -\sum_{\alpha\in\Omega^{k-1}} f^D(\alpha)\sum_{\omega\in\Omega} f^D(\omega|\alpha)\,\log f^D(\omega|\alpha) \end{aligned}$$

3.5.5 Simulating a Markov process

Given the n observations x_1^n, and, under the hypothesis that the underlying process is a Markov chain of order r,

- one first determines the order k of the chain (with $k \leq \frac{1}{2}\frac{\ln n}{\ln m}$) by using the test 3.35.
- one then estimates the corresponding theoretical transitions $f^M(\omega|\alpha)$ (with $\omega \in \Omega$ and $\alpha \in \Omega^r$) by the empirical ones $f^D(\omega|\alpha) := \frac{n(\alpha\omega)}{\alpha}$ (maximum likelihood estimation).

At this stage, one is in position to *simulate* the Markov process, simply by running a k-th order process with transition matrix $f^D(\omega|\alpha)$ from some initial state $\alpha \in \Omega^r$ drawn with probability $f^D(\alpha)$.

Example 3.21 Written with $m = 27$ states (the alphabet + the blank, without punctuation), the english version of the *Universal declaration of Human Rights* constitutes a text x_1^n of length $n = 8'149$, from which conditional empirical distributions $f^D(\omega|\alpha)$ can be computed. One can imagine the text to have been produced by a Markov chain of order r, defined by the set of theoretical conditional probabilities $\{f^M(\omega|\alpha)\}$ where

α is a r-gram. Those theoretical probabilities can be estimated (ML-estimation) by the corresponding empirical estimates, that is $\tilde{f}^M(\omega|\alpha) := f^D(\omega|\alpha)$, and, in virtue of 3.34 the estimate is guaranteed to reliable for $r \leq \frac{1}{2}\frac{\log n}{\log m} = \frac{1}{2}\frac{\log 8'149}{\log 27} = 1.36$, which is rather small! Simulations based on $\tilde{f}^M(\omega|\alpha) = f^D(\omega|\alpha)$ with $|\omega| = r$ yield:

$r = 0$ **(independent process)**

```
iahthire edr pynuecu d lae mrfa ssooueoilhnid nritshfssmo nise yye
noa it eosc e lrc jdnca tyopaooieoegasrors c hel niooaahettnoos rnei s
sosgnolaotd t atiet
```

The relative frequencies of all $m = 27$ symbols are correctly sampled; in particular, the proportion of blanks (16.7%) is respected and words have about the correct length. However, original transitions between symbols are obviously not correctly reproduced.

$r = 1$ **(First-order Markov chain)**

```
erionjuminek in l ar hat arequbjus st d ase scin ero tubied pmed beetl
equly shitoomandorio tathic wimof tal ats evash indimspre tel sone aw
onere pene e ed uaconcol mo atimered
```

First-order transitions are taken into account, which makes the sample more readable than the previous one (in particular, the consonants-vowels transitions are respected).

$r = 2$ **(Second-order Markov chain)**

```
mingthe rint son of the frentery and com andepent the halons hal
to coupon efornitity the rit noratinsubject will the the in priente
hareeducaresull ch infor aself and evell
```

The sample begins to look like english, with a zest of latin....

$r = 3$ **(Third-order Markov chain)**

```
law socience of social as the right or everyone held genuinely
available sament of his no one may be enties the right in the cons
as the right to equal co one soveryone
```

The text definitely looks like english, with whole words correctly reproduced. However, we are beyond the safe limit $k_{\max} = 1.36$: the simulated text betrays its origin, namely the Universal declaration of Human Right, and not another original english text of comparable size, such as a a cooking recipy or a text about mathematics. Equivalently, the model estimates $\tilde{f}^M(\omega|\alpha) = f^D(\omega|\alpha)$ with $|\alpha| = 3$ are over-parameterized.

$r = 4$ **(Fourth-order Markov chain)**

```
are endowed with other means of full equality and to law no one is the
right to choose of the detent to arbitrarily in science with pay for
through freely choice work
```

All simulated words are now genuine english words, which reflects the high redundancy of english $R \cong 0.73$ (see example 3.5). Meanwhile, the over-parameterization problem is getting worse....

$r = 9$ **(Markov chain of order 9)**

```
democratic society and is entitled without interference and to seek
receive and impartial tribunals for acts violating the fundamental
```

```
rights indispensable for his
```

The over-parameterization has reached outrageously scandalous levels by statistical standards: the set $\{f^M(\omega|\alpha)\}$ of nine-order transitions ($|\alpha| = 9$) contains $(27-1)\,27^9 = 6.1 \times 10^{13}$ parameters estimated from a text of length $n = 8.1 \times 10^3$ only! As a result, whole chunks of the original text are reproduced without alteration in the simulated text.

Control Question 38

Determine the unique correct answer:

1. the "curse of dimensionality" alludes to a problem of a) visualization in high-dimensional spaces; b) lack of regularity for Markov chains; c) unability to perform hypothesis testing; d) over-parameterization.

2. the number of arguments of the stationary distribution function associated to a Markov chain of order r is a) 1;; b) $r-1$; c) r; d) variable.

3. the observation of a transition vorbiden by a Markov model a) is a rare event; b) is possible is the sample is small enough; c) should occur in proportion less than the significance level; d) indicates the inadequacy of the model.

4. the conditional Kullback-Leibler divergence $K_r(f^D||f^M)$ of order r a) is zero iff a vorbidden transition occurs; b) is infinite iff a vorbidden transition occurs; c) is increasing in r; d) increases with the probability $P(f^D|f^M)$.

Answer

1. d) over-parameterization

2. c) r

3. d) indicates the inadequacy of the model

4. b) is infinite iff a vorbidden transition occurs

Historical Notes and Bibliography

Section 3.3.5. Irreversible behavior was first clearly attested and formalized in Thermodynamics in the middle of the XIXth century under the name of the *Second Principle of Thermodynamics*, stating that "the (physical) entropy of an isolated non-equilibrium physical system grows until it reaches equilibrium". The modern, purely information-theoretical formulation of the second principle is given (in the framework of Markov processes) by theorem 3.9. It shows in particular $K(f^{(n)}||\pi)$ to decrease to zero for $n \to \infty$. If the stationary distribution π is uniform (i.e. $\pi_j = 1/m$ for all $j = 1, \ldots, m$), then $K(f^{(n)}||\pi) = \log m - H(f^{(n)})$ where $H(f)$ is Shannon's entropy of distribution f: here theorem 3.9 confirms that $H(f^{(n)})$ is indeed increasing in n with limit $\log m$. But in the general case

where π is not uniform, the Second Principle should be more correctly stated as "the relative entropy (with respect to π) of an isolated non-equilibrium physical system grows until it reaches equilibrium".

Section 3.5. The Universal Declaration of Human Rights was adopted by UNO's General Assembly (resolution 217 A (III)) of 10 December 1948.

OutLook

- Cover,T.M. and Thomas,J.A. *Elements of Information Theory*, Wiley (1991)
- Hillman, C. *An entropy primer*, http://www.math.washington.edu/ hillman/PUB/primer.ps, (1996)
- Jelinek,F. *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, MA (1998)
- Mirkin,B. *Mathematical Classification and Clustering*, Kluwer, Dordrecht, (1996)
- Shields, P.C. *The Ergodic Theory of Discrete Sample Paths*, Graduate Studies in Mathematics, Volume 13, American Mathematical Society (1996)
- Xanthos, A. Entropizer 1.1: un outil informatique pour l'analyse séquentielle. *Proceedings of the 5th International Conference on the Statistical Analysis of Textual Data (2000).*

Chapter 4

Module C4: Coding for Noisy Transmission

by J.-C. Chappelier

Learning Objectives for Chapter 4

In this chapter, we present:

1. the basics of coding a discrete information source in order to be able to accurately transmit its messages even in the presence of noise;
2. how a noisy transmission can be formalized by the notion of a "*channel*";
3. the two fundamental notions ruling noisy transmissions: the "*channel capacity*" and the "*transmission rate*";
4. the fundamental limit for the transmission error in the case of a noisy transmission.

Introduction

When dealing with "information", one of the basic goals is to transmit it reliably. In this context, "*transmit*" both means "transmitting some information from one point to another", as we usually understand it, but also to "transmit" it through time; for instance to store it somewhere (to memorize it) and then retrieve it later on. In both cases however, transmission of information can, in real life, hardly be achieved in a fully reliable manner. There always exists a risk of distortion of the transmitted information: some noise on the line, some leak of the memory or the hard disk storing the information, etc.

What effect does noise have on the transmission of messages? Several situations could be possible:

- it is never possible to transmit any messages reliably (too much noise);
- it is possible to transmit messages with a "reasonable" error probability;
- it is possible to transmit messages with with an error probability which is as small as we can wish for (using error correcting codes).

The purpose of the present chapter is to study how coding can help transmitting information in a reliable way, even in the presence of noise during the transmission. The basic idea of such codings is to try to add enough redundancy in the coded message so that transmitting it in "reasonably" noisy conditions leaves enough information undisturbed for the receiver to be able to reconstruct the original message without distortion.

Of course, the notions of "*enough redundancy*" and "*reasonably noisy conditions*" need to be specified further; and even quantified and related. This will be done by first formalizing a bit further the notion of "noisy transmission", by introducing the notion of a "communication channel", which is addressed in section 4.1.

As we will see in section 4.3, the two fundamental notions ruling noisy transmissions are the *channel capacity* and the *rate* use for transmitting the symbols of the messages. These notions are first introduced in section 4.1.

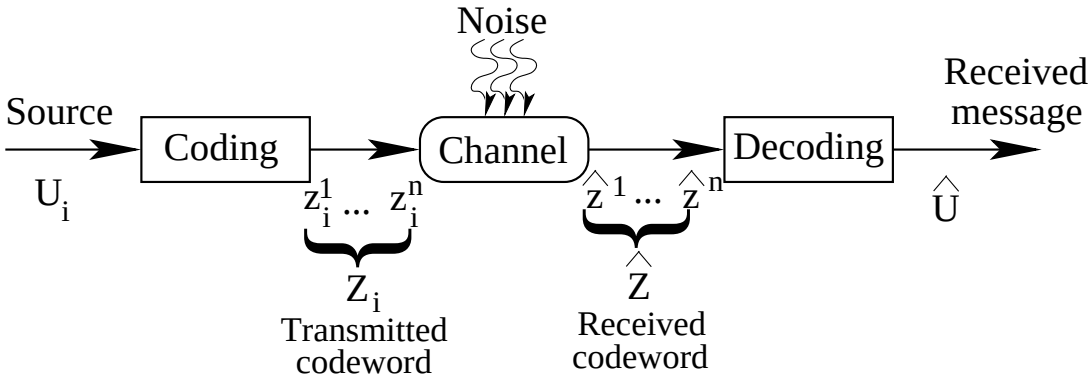


Figure 4.1: Error correcting communication over a noisy channel.

The general framework this chapter will focuses on is summarized in figure 4.1.

4.1 Communication Channels

Learning Objectives for Section 4.1

This section presents the notion of a communication channel and the main characterization of it: the capacity.
After studying it, you should be able to formalize a communication (i.e. to give the corresponding channel) and to compute its capacity, at least in the most simple cases.
You should also know what a symmetric channel is, and what this implies on its capacity
We finally also introduce the notion of *transmission rate*.

4.1.1 Communication Channels

Roughly speaking, a *communication channel** (shorter "channel") represents all that could happen to the transmitted messages between their emission and their reception.

A *message** is a sequences of symbols. A *symbol* is simply an element of a set, called an *alphabet*. In this course, only finite alphabets will be addressed.

The input sequence $X_1, X_2, X_3, \ldots$ (i.e. the message to be transmitted) is fully determined by the source alone; but the transmission determines the resulting conditional probabilities of the output sequence $Y_1, Y_2, Y_3, \ldots$ (i.e. the message received) knowing the input sequence.

In mathematical terms, the channel specifies the *conditional* probabilities of the various messages that can be received, conditionally to the messages that have been emitted; i.e. $P(Y_1 = y_1, ..., Y_n = y_n | X_1 = x_1, ..., X_n = x_n)$ for all possible n and values $y_1, x_1, ..., y_n, x_n$.

Definition 4.1 (Discrete Memoryless Channel) The *discrete memoryless channel* (*DMC**) is the simplest kind of communication channel. Formally, DMC consists of three quantities:

1. a discrete *input alphabet*, $\mathcal{V}_X$, the elements of which represent the possible emitted symbols for all input messages (the source X);
2. a discrete *output alphabet*, $\mathcal{V}_Y$, the elements of which represent the possible received symbols (output sequence); and
3. for each $x \in \mathcal{V}_X$, the *conditional probability distributions* $p_{Y|X=x}$ over $\mathcal{V}_Y$ which describe the channel behavior in the manner that, for all $n = 1, 2, 3, \ldots$:

$$P(Y_n = y_n | X_1 = x_1, \ldots, X_n = x_n, Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}) = P(Y = y_n | X = x_n), \quad (4.1)$$

These are called the *transmission probabilities** of the channel.

Equation (4.1) is the mathematical statement that corresponds to the "memoryless" nature of the DMC. What happens to the signal sent on the n-th use of the channel is independent of what happens on the previous $n-1$ uses.

Notice also that (4.1) implies that the DMC is *time-invariant*, since the probability distribution $p_{Y_n|x_n}$ does not depend on n.

When $\mathcal{V}_X$ and $\mathcal{V}_Y$ are finite, a DMC is very often specified by a diagram where:

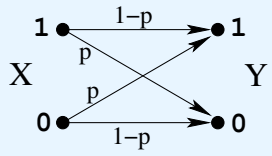
1. the nodes on the left indicate the input alphabet $\mathcal{V}_X$;
2. the nodes on the right indicate the output alphabet $\mathcal{V}_Y$; and
3. the directed branch from x_i to y_j is labeled with the conditional probability $p_{Y|X=x_i}(y_j)$ (unless this probability is 0, in which case the branch is simply omitted.)

Example 4.1 (Binary Symmetric Channel) The simplest (non trivial) case of DMC is the *binary symmetric channel* (*BSC**), for which $\mathcal{V}_X = \mathcal{V}_Y = \{0, 1\}$ ("binary") and

$$p_{Y|X=0}(1) = p_{Y|X=1}(0)$$

("symmetric"). This value $p = p_{Y|X=0}(1) = p_{Y|X=1}(0)$ is called the error rate and is the only parameter of the BSC. Indeed, $p_{Y|X=0}(0) = p_{Y|X=1}(1) = 1 - p$.

The BSC is represented by the following diagram:



Example 4.2 (Noisy Transmission over a Binary Symmetric Channel) Suppose we want to transmit the 8 following messages: 000, 001, 010, 011, 100, 101, 110 and 111.

Suppose that the channel used for transmission is noisy in such a way that it changes one symbol over ten, regardless of everything else; i.e. each symbol has a probability $p = 0.1$ to be "flipped" (0 into 1, and 1 into 0). Such a channel is a BSC with an error rate equal to $p = 0.1$,

What is the probability to transmit one of our messages correctly?

Regardless of which message is sent, this probability is

$$(1 - p)^3 = 0.9^3 = 0.719$$

(corresponding to the probability to transmit 3 times one bit without error).

Therefore, the probability to receive and erroneous message is 0.281, i.e 28%; which is quite high!

Suppose now we decide to code each symbol of the message by twice itself:

message	000	001	010	011	100	...	111
code	000000	000011	001100	001111	110000	...	111111

What is now the probability to have a message sent correctly? In the same way, this is $(1 - p)^6 = 0.531$

And the probability to receive and erroneous message is now 0.469... ...worse than previously, it seems!

However, what is the probability to receive a erroneous message which seems to be valid; i.e. what is the probability to receive a erroneous message and to not detect it as wrong?

Not detecting an erroneous message means that two corresponding symbol have both been changed. If for instance we sent 000000, but 110000 is received, there is not way to see that some errors occurred. However, if 010000 is received, clearly at least

one error occurred (and retransmission could for instance be required).

So, not detecting an error could come either from 2 changes (at the corresponding places) or 4 changes or the whole 6 symbols. What is the probability to change 2 symbols? Answer: $\binom{6}{2} p^2 (1-p)^4 = 15\, p^2 (1-p)^4$ What is the probability to change 2 corresponding symbols? Only $\binom{3}{1} p^2 (1-p)^4 = 3\, p^2 (1-p)^4$.

Similarly, the probability to change 4 corresponding symbols is $3\, p^4 (1-p)^2$, and to change the whole six symbols is p^6.

Therefore, the probability of not detecting an error is

$$3\, p^2 (1-p)^4 + 3\, p^4 (1-p)^2 + p^6 = 0.020$$

which is much smaller. This means that the probability to make a error in the reception (i.e. to trust an erroneous message without being aware of) is only 0.02.

Conclusion: some codings are better than other for the transmission of messages over a noisy channel.

Finally, we wish to clearly identify the situation when a DMC is used *without feedback**, i.e. when the probability distribution of the inputs does not depend on the output. More formally, a DMC is said to be "without feedback" when:

$$p_{X_n|x_1,\ldots,x_{n-1},y_1,\ldots,y_{n-1}} = p_{X_n|x_1,\ldots,x_{n-1}} \tag{4.2}$$

for all $n = 1, 2, 3, \ldots$. Notice that (4.2) does not imply that we choose each input digit independently of previous input digits, only that we are not using the past output digits in any way when we choose successive input digits (as we could if a feedback channel was available from the output to the input of the DMC).

Let us now give a fundamental result about DMC without feedback.

Theorem 4.1 *For a DMC without feedback, we have for all $n \in \mathbb{N}$:*

$$H(Y_1 \ldots Y_n | X_1 \ldots X_n) = \sum_{i=1}^{n} H(Y_i | X_i)$$

where $X_1 \ldots X_n$ stands for an input sequence of length n and $Y_1 \ldots Y_n$ for the corresponding output.

PROOF From the chain rule for probabilities, we have

$$p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n}(x_1,\ldots,x_n,y_1,\ldots,y_n) =$$
$$p_{X_1}(x_1)\, p_{Y_1|x_1}(y_1) \prod_{i=2}^{n} p_{X_i|x_1,\ldots,x_{i-1},y_1,\ldots,y_{i-1}}(x_i) p_{Y_i|x_1,\ldots,x_i,y_1,\ldots,y_{i-1}}(y_i)$$

Making use of (4.1) and (4.2), we get:

$$\begin{aligned} & p_{X_1,\ldots,X_n,Y_1,\ldots Y_n}(x_1,\ldots,x_n,y_1,\ldots y_n) \\ &\quad = p_{X_1}(x_1)\, p_{Y_1|x_1}(y_1) \prod_{i=2}^{n} p_{X_i|x_1,\ldots,x_{i-1}}(x_i)\, p_{Y|x_i}(y_i) \\ &\quad = \left[p_{X_1}(x_1) \prod_{i=2}^{n} p_{X_i|x_1,\ldots,x_{i-1}}(x_j) \right] \left[\prod_{i=1}^{n} p_{Y|x_i}(y_i) \right] \\ &\quad = p_{X_1,\ldots,X_n}(x_1,\ldots,x_n) \prod_{i=1}^{n} p_{Y|x_i}(y_i). \end{aligned}$$

Dividing now by $p_{X_1,\ldots,X_n}(x_1,\ldots,x_n)$, we obtain:

$$p_{Y_1,\ldots,Y_n|x_1,\ldots,x_n}(y_1,\ldots,y_n) = \prod_{i=1}^{n} p_{Y|x_i}(y_i) \tag{4.3}$$

The relationship (4.3) is so fundamental that it is sometimes (erroneously) given as the definition of the DMC. When (4.3) is used instead of (4.1), the DMC is implicitly assumed to be used without feedback.

e-pendix: Cascading Channels

4.1.2 Channel Capacity

The purpose of a channel is to transmit messages ("information") from one point (the input) to another (the output). The *channel capacity** precisely measures this ability: it is the maximum average amount of information the output of the channel can bring on the input.

Recall that a DMC if fully specified by the conditional probability distributions $p_{Y|X=x}$ (where X stands for the input of the channel and Y for the output). The input probability distribution $p_X(x)$ is not part of the channel, but only of the input source used. The *capacity* of a channel is thus defined as the maximum mutual information $I(X;Y)$ that can be obtained among all possible choice of $p_X(x)$. More formally:

Definition 4.2 (Channel Capacity) The capacity C of a Discrete Memoryless Channel is defined as

$$C = \max_{p_X} I(X;Y), \tag{4.4}$$

where X stands for the input of the channel and Y for the output. ♦

We will shortly see that this definition is indeed very useful for studying noisy transmissions over channels, but let us first give a first example.

Example 4.3 (Capacity of BSC) What is the capacity C of a BSC, defined in example 4.1?

First notice that, by definition of mutual information,

$$C = \max_{p_X}\Big(H(Y) - H(Y|X)\Big).$$

Furthermore, since in the case of a BSC, $P(Y \neq X) = p$ and $P(Y = X) = 1 - p$, we have $H(Y|X) = -p\log(p) - (1-p)\log(1-p) =: \widetilde{\mathrm{h}}(p)$, which does not depend on p_X. Therefore

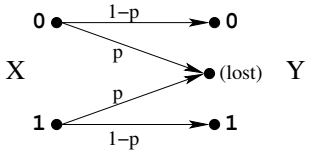
$$C = \max_{p_X}\Big(H(Y)\Big) - \widetilde{\mathrm{h}}(p).$$

Since Y is a binary random variable, we have (by theorem 1.2): $H(Y) \leq \log 2$, i.e. $H(Y) \leq 1$ bit. Can this maximum be reached for some p_X? Indeed, yes: if X is uniformly distributed, we have $p_Y(\mathtt{0}) = p \cdot p_X(\mathtt{1}) + (1-p) \cdot p_X(\mathtt{0}) = 0.5\ p + 0.5\ (1-p) = 0.5$; which means that Y is also uniformly distributed, leading to $H(Y) = 1$ bit. Therefore: $\max_X H(Y) = 1$ bit and

$$C = 1 - \widetilde{\mathrm{h}}(p) \qquad \text{(in bits)}.$$

Control Question 39

What is the capacity of the "binary erasure channel" defined by the following graph:



Is it (in the most general case):

1. $C = 1 - \widetilde{\mathrm{h}}(p) = 1 + p\log p + (1-p)\log(1-p)$,
2. $C = 1 - p$,
3. $C = 1$,
4. $C = 1 + \widetilde{\mathrm{h}}(p) = 1 - p\log p - (1-p)\log(1-p)$,
5. or $C = 1 + p$?

Answer

Let r be $r = P(X = \mathtt{0})$ (thus $P(X = \mathtt{1}) = 1 - r$). By definition, the channel capacity is:

$$C = \max_{P(X)} I(Y;X) = \max_{r} \left[H(Y) - H(Y|X)\right]$$

Let us first compute $H(Y|X)$:

$$\begin{aligned} H(Y|X) &= -\sum_x P(X=x) \sum_y P(Y|X=x) \log P(Y|X=x) \\ &= -\sum_x P(X=x) \left[p\,\log p + (1-p)\,\log(1-p) + 0\right] \\ &= -\left[p\,\log p + (1-p)\,\log(1-p)\right] \\ &= \widetilde{\mathrm{h}}(p) \end{aligned}$$

Since $H(Y|X) = \widetilde{\mathrm{h}}(p)$ is independent of r

$$C = \max_r [H(Y)] - \widetilde{\mathrm{h}}(p)$$

Let us now compute $H(Y)$. We first need the probability distribution of Y:

$$P(Y = \mathtt{0}) = r\,(1-p)$$

$$P(Y = \mathtt{1}) = (1-r)\,(1-p)$$

$$P(Y = \text{lost}) = r\,p + (1-r)\,p = p$$

Thus

$$H(Y) = -r\,(1-p)\log(r\,(1-p)) - (1-r)\,(1-p)\log((1-r)\,(1-p)) - p\log p$$

Since $P(Y = \text{lost})$ is independent of r, $H(Y)$ is maximal for $P(Y = \mathtt{0}) = P(Y = \mathtt{1})$ i.e. $r\,(1-p) = (1-r)\,(1-p)$ and thus $r = 1 - r = 0.5$.

Another way to find the maximum of $H(Y)$ is to see that

$$\begin{aligned} H(Y) &= -r\,(1-p)\log(r\,(1-p)) - (1-r)\,(1-p)\log((1-r)\,(1-p)) - p\log p \qquad (4.5)\\ &= -r\,(1-p)\log(r) - r\,(1-p)\log(1-p) - (1-r)\,(1-p)\log(1-r)\\ &\quad -(1-r)\,(1-p)\log(1-p) - p\log p\\ &= (1-p)\;\widetilde{\mathrm{h}}(r) - (r+1-r)\,(1-p)\;\log(1-p) - p\log p\\ &= (1-p)\;\widetilde{\mathrm{h}}(r) + \widetilde{\mathrm{h}}(p) \end{aligned}$$

which is maximum for $r = 0.5$.

Eventually, the maximum of $H(Y)$ is $(1-p) + \widetilde{\mathrm{h}}(p)$, and at last we find:

$$C = 1 - p + \widetilde{\mathrm{h}}(p) - \widetilde{\mathrm{h}}(p) = 1 - p.$$

Thus, the right answer is 2).

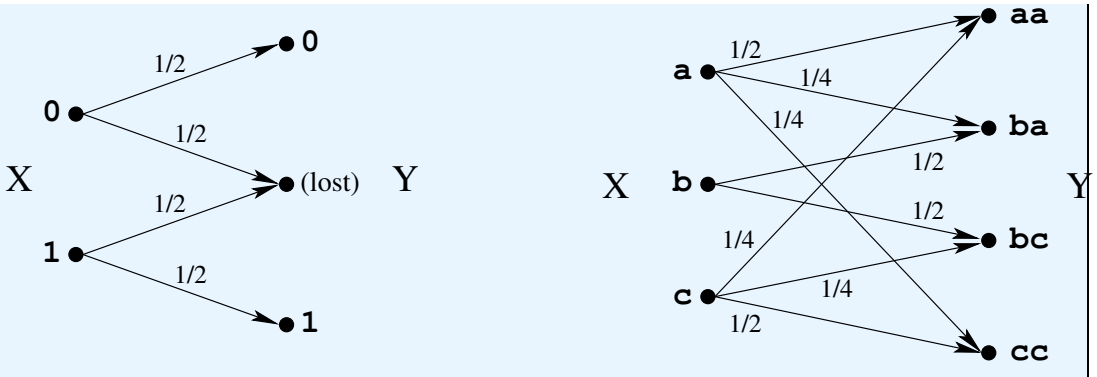
4.1.3 Input-Symmetric Channels

We here consider only DMC with a finite alphabet, i.e. a finite number, K, of input symbols, and a finite number, J, of output symbols.

Definition 4.3 Such a DMC is said to be *input-symmetric* if the error probability distributions are all the same[a] for all input symbols; i.e. the sets $\left\{p_{Y|x_i}(y) : y \in \mathcal{V}_Y\right\}$ are independent of x_i. ♦

[a] apart from a permutation

Example 4.4 (Input-Symmetric Channels)



(a) A input–symmetric DMC which is not output–symmetric.

(b) An output–symmetric DMC which is not input–symmetric.

A BSC is input-symmetric. That of Figure (a) above is also input-symmetric, but the channel of Figure (b) is not input-symmetric.

Lemma 4.1 *For a input-symmetric DMC, $H(Y|X)$ is independent of the distribution p_X, and $H(Y|X) = H(Y|x_i)$ for any given $x_i \in \mathcal{V}_X$.*

PROOF It follows from the definition of a input-symmetric channel that

$$\forall x_i \in \mathcal{V}_X \quad H(Y|X = x_i) = H_0 = -\sum_{j=1}^{J} p_j \log p_j \tag{4.6}$$

where $\{p_1, p_2, \ldots, p_J\}$ the set of probabilities $p_{Y|x_i}(y)$ (which is indepent of the input letter x_i). Therefore

$$H(Y|X) = \sum_x p_X(x) H(Y|X = x) = \sum_x p_X(x) H_0 = H_0$$

For a input symmetric DMC, finding the input probability distribution that achieves capacity (i.e. achieves the maximum of $I(X;Y)$) reduces to simply finding the input distribution that maximizes the uncertainty of the output.

Property 4.1 *For a input-symmetric DMC, we have*

$$C = \max_{P_X} [H(Y)] - H_0 \tag{4.7}$$

where $H_0 = H(Y|X = x_i)$ for any of the $x_i \in \mathcal{V}_X$.

PROOF This property directly derives from definition 4.2 and lemma 4.1. ■

4.1.4 Output-Symmetric Channels

A input-symmetric channel is one in which the probabilities leaving each input symbol are the same (appart from a permutation). We now consider channels with the prop-

erty that the probabilities reaching each output symbol are the same (appart from a permutation).

More formaly, a DMC is said to be *output-symmetric* when the sets $\{p_{Y|x}(y_i) : x \in \mathcal{V}_X\}$ are independent of y_i. Notice that in this case the sums $\sum_x p_{Y|x}(y_i)$ are independent of y_i.[1]

Example 4.5 (Output-Symmetric Channel) The BSC (see example 4.1) is output-symmetric. The channel of example 4.4, figure (b), is output-symmetric, but that of figure (a) is not.

Lemma 4.2 *For a output-symmetric DMC, the uniform input probability distribution (i.e., $p_X(x_i)$ is the same for all x_i) results in the uniform output probability distribution (i.e., $p_Y(y_i)$ is the same for all y_i).*

PROOF

$$p_Y(y_j) = \sum_{x_i} p_{Y|x_i}(y_j) p_X(x_i) = \frac{1}{|\mathcal{V}_X|} \sum_{x_i} p_{Y|x_i}(y_j)$$

But, since the DMC is output-symmetric, the sum on the right in this last equation is independent of y_j. Thus p_Y is independent of y_j; i.e. is the uniform probability distribution. ■

Property 4.2 *For an output-symmetric DMC, the input of which is X and the output of which is Y, we have:*

$$\max_{p_X} H(Y) = \log |\mathcal{V}_Y|. \tag{4.8}$$

PROOF This property comes immediately from Lemma 4.2 and theorem 1.2 on the entropy upper bound. ■

4.1.5 Symmetric Channels

Definition 4.4 (Symmetric Channel) A DMC is *symmetric* when it is both input- and output-symmetric. ♦

Theorem 4.2 *The capacity of a symmetric channel (the input of which is X and the output of which is Y) is given by:*

$$C = \log |\mathcal{V}_Y| - H_0 \tag{4.9}$$

where $H_0 = H(Y|X = x_i)$ for any of the input symbol $x_i \in \mathcal{V}_X$.

[1] Some authors called this property "week (output-)symmetry".

PROOF The above theorem is an immediate consequence of properties 4.1 and 4.2. ■

Example 4.6 The BSC (see Example 4.1) is a symmetric channel for which $H_0 = \widetilde{\mathrm{h}}(p) = -p\log p - (1-p)\log(1-p)$. Thus,

$$C_{\mathrm{BSC}} = \log 2 - \widetilde{\mathrm{h}}(p) = 1 - \widetilde{\mathrm{h}}(p) \qquad (\text{in bits})$$

4.1.6 Transmission Rate

Definition 4.5 (Transmission Rate) The transmission rate (in base b) of a code encoding a discrete source U of $|\mathcal{V}_U|$ messages with codewords of fixed length n is defined by:

$$R_b = \frac{\log_b |\mathcal{V}_U|}{n}$$

Notice that $|\mathcal{V}_U|$ is also the number of possible codewords (deterministic non-singular coding).

In practice, the base b chosen for the computation of the transmission rate is the arity of the code.

Example 4.7 The (binary) transmission rate of the code used in example 4.2 is $R = \frac{\log 8}{6} = \frac{3}{6} = \frac{1}{2}$. This sounds sensible since this code repeats each message twice, i.e. uses twice as many symbols as originally emitted.

Control Question 40

On a noisy channel, we plan to use a code consisting of tripling each symbol of the messages. For instance `a` will be transmitted as `aaa`.

What is the transmission rate R of such a code?
(If you are a bit puzzled by the base, choose for the base the arity of the source, i.e. the number of different symbols the source can emit)

Answer

Whatever the alphabet used for the messages, each symbol is repeated 3 times. Therefore whatever the message is, it will always be encoded into a coded message three times longer. Therefore, the transmission rate is $R = \frac{1}{3}$.

For people preferring the application of the definition, here it is:
the fixed length of codewords is $n = 3\,m$, where m is the length of a message. If D is the arity of the source U (i.e. the size of its alphabet), the number of possible messages of length m is $|\mathcal{V}_U| = D^m$. Thus, we have:

$$R_D = \frac{\log_D |\mathcal{V}_U|}{n} = \frac{\log_D D^m}{3\,m} = \frac{m}{3\,m} = \frac{1}{3}$$

Summary for Section 4.1

Channel: input and output alphabets, transmission probabilities ($p_{Y|X=x}$).
DMC = Discrete Memoryless Channel.

Channel Capacity: $C = \max_{p_X} I(X;Y)$

Code/Transmission Rate: $R_b = \frac{\log_b |\mathcal{V}_U|}{n}$

Capacity of Input-symmetric channels: $C = \max_{P_X} [H(Y)] - H(Y|X = x_i)$ for any of the $x_i \in \mathcal{V}_X$.

Capacity of Symmetric channels: $C = \log |\mathcal{V}_Y| - H(Y|X = x_i)$ for any of the $x_i \in \mathcal{V}_X$.

4.2 A Few Lemmas

Learning Objectives for Section 4.2

In this section, we introduce several general results that we shall subsequently apply in our study of channels, but have also many other applications.

4.2.1 Multiple Use Lemma

We previously define the capacity of a channel as the maximum amount of information the output of the channel can bring on the input. What can we say if we use the same channel several times (as it is expected to be the case in real life!)?

Lemma 4.3 *If a DMC without feedback of capacity C is used n times, we have:*

$$I(X_1 \dots X_n; Y_1 \dots Y_n) \leq n\, C$$

where $X_1 ... X_n$ stands for an input sequence of length n and $Y_1 ... Y_n$ for the corresponding output.

PROOF Using definition of mutual information and theorem 4.1, we have:

$$\begin{aligned} I(X_1 \dots X_n; Y_1 \dots Y_n) &= H(Y_1 \dots Y_n) - H(Y_1 \dots Y_n | X_1 \dots X_n) \\ &= H(Y_1 \dots Y_n) - \sum_{i=1}^{n} H(Y_i|X_i) \end{aligned} \tag{4.10}$$

Recalling that

$$H(Y_1 \dots Y_n) = H(Y_1) + \sum_{i=2}^{n} H(Y_i | Y_1 ... Y_{i-1})$$

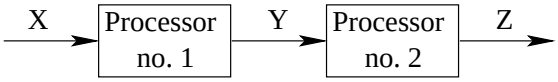


Figure 4.2: The conceptual situation for the Data Processing Lemma.

and that

$$H(Y_i|Y_1...Y_{i-1}) \leq H(Y_i)$$

we have:

$$H(Y_1 \dots Y_n) \leq \sum_{i=1}^{n} H(Y_i)$$

which, after substitution into (4.10) gives

$$\begin{aligned} I(X_1 \dots X_n; Y_1 \dots Y_n) &\leq \sum_{i=1}^{n} [H(Y_i) - H(Y_i|X_i)] \\ &= \sum_{i=1}^{n} I(X_i; Y_i) \\ &\leq n\,C \end{aligned} \tag{4.11}$$

by the definition of channel capacity. ■

4.2.2 Data Processing Lemma

We here consider the situation shown in Figure 4.2, where several "processors" are used successively. "Processors" considered here are completely arbitrary devices. They may be deterministic or even stochastic. They may even have anything at all inside. The only thing that Figure 4.2 asserts is that there is no "hidden path" by which X can affect Z, i.e., X can affect Z only indirectly through its effect on Y. In mathematical terms, this constraint can be expressed as

$$p_{Z|x,y}(z) = p_{Z|y}(z) \tag{4.12}$$

for all y such that $p_Y(y) \neq 0$, which means simply that, when y is given, then z is not further influenced by x.

More formally the sequence X, Y, Z is a *first-order Markov chain*.

The Data Processing Lemma states essentially that information cannot be increased by any sort of processing (although it can perhaps be put into a more accessible form!).

Lemma 4.4 (Data Processing Lemma) *When X, Y, Z is a first-order Markov chain, i.e. when (4.12) holds, we have:*

$$I(X; Z) \leq I(X; Y) \tag{4.13}$$

and

$$I(X; Z) \leq I(Y; Z). \tag{4.14}$$

In other words, the mutual information between any two states of a Markov chain is always less or equal to the mutual information between any two intermediate states.

PROOF (4.12) implies that

$$H(Z|XY) = H(Z|Y) \tag{4.15}$$

and hence

$$\begin{aligned} I(Y;Z) &= H(Z) - H(Z|Y) \\ &= H(Z) - H(Z|XY) \\ &\geq H(Z) - H(Z|X) = I(X;Z) \end{aligned}$$

which proves (4.14). To prove (4.13), we need only show that Z, Y, X (the reverse of the sequence X, Y, Z) is also a Markov chain, i.e. that

$$p_{X|yz}(x) = p_{X|y}(x) \tag{4.16}$$

for all y such that $p_Y(y) \neq 0$.

Indeed,

$$\begin{aligned} & p_{Z|x,y}(z) = p_{Z|y}(z) \implies \\ p_{X|y,z}(x) &= \frac{p_{Z|x,y}(z) \cdot p_{XY}(x,y)}{p_{YZ}(y,z)} \\ &= \frac{p_{Z|y}(z) \cdot p_{XY}(x,y)}{p_{YZ}(y,z)} \\ &= \frac{p_{YZ}(y,z) \cdot p_{XY}(x,y)}{p_Y(y) \cdot p_{YZ}(y,z)} \\ &= \frac{p_{XY}(x,y)}{p_Y(y)} \\ &= p_{X|y}(x) \end{aligned}$$

4.2.3 Fano's Lemma

Now, for the first time in our study of information theory, we introduce the notion of "errors". Suppose we think of the random variable $\widehat{U}$ as being an *estimate* of the random variable U. For this to make sense, $\widehat{U}$ needs to take on values from the same alphabet as U. Then an *error* is just the event that $\widehat{U} \neq U$ and the probability of error, P_e, is thus

$$P_e = P(\widehat{U} \neq U). \tag{4.17}$$

We now are ready for one of the most interesting and important results in information theory, one that relates P_e to the conditional uncertainty $H(U|\widehat{U})$.

Lemma 4.5 (Fano's Lemma) *Let U and $\widehat{U}$ be two D-ary random variables with the same values. Denoting by P_e the probability that U is different from $\widehat{U}$, we have:*

$$\widetilde{\mathrm{h}}(P_e) + P_e \log_2(D-1) \geq H(U|\widehat{U}) \tag{4.18}$$

where the uncertainty $H(U|\widehat{U})$ is in bits, and $\widetilde{\mathrm{h}}$ is is the entropy of a binary random variable: $\widetilde{\mathrm{h}}(p) = -p\log(p) - (1-p)\log(1-p)$.

PROOF We begin by defining the random variable Z as the random variable indicating a difference between U and $\widehat{U}$:

$$Z = \begin{cases} 0 & \text{when } \widehat{U} = U \\ 1 & \text{when } \widehat{U} \neq U, \end{cases}$$

Z is therefore a binary random variable with parameter P_e. So, we have:

$$H(Z) = \widetilde{\mathrm{h}}(P_e). \tag{4.19}$$

Furthermore:

$$\begin{aligned} H(UZ|\widehat{U}) &= H(U|\widehat{U}) + H(Z|U\widehat{U}) \\ &= H(U|\widehat{U}), \end{aligned}$$

since U and $\widehat{U}$ uniquely determine Z (therefore $H(Z|U\widehat{U}) = 0$. Thus

$$\begin{aligned} H(U|\widehat{U}) &= H(UZ|\widehat{U}) \\ &= H(Z|\widehat{U}) + H(U|\widehat{U}Z) \\ &\leq H(Z) + H(U|\widehat{U}Z) \end{aligned} \tag{4.20}$$

But

$$H(U|\widehat{U}, Z=0) = 0 \tag{4.21}$$

since in this case U is uniquely determined, and

$$H(U|\widehat{U}, Z=1) \leq \log_2(D-1) \tag{4.22}$$

since, for each value u of $\widehat{U}$, there are, when $Z=1$, at most $D-1$ values with non-zero probability for U:

$$\forall u \in \mathcal{V}_U \; H(U|\widehat{U}=u, Z=1) \leq \log_2(D-1)$$

Equations (4.21) and (4.22) imply

$$\begin{aligned} H(U|\widehat{U}Z) &= p_Z(0)H(U|\widehat{U}, Z=0) + p_Z(1)H(U|\widehat{U}, Z=1) \\ &= 0 + p_Z(1)H(U|\widehat{U}, Z=1) \\ &\leq p_Z(1)\log_2(D-1) = \\ &\leq P_e \log_2(D-1). \end{aligned} \tag{4.23}$$

Substituting (4.19) and (4.23) into (4.20), we obtain (4.18) as was to be shown. ■

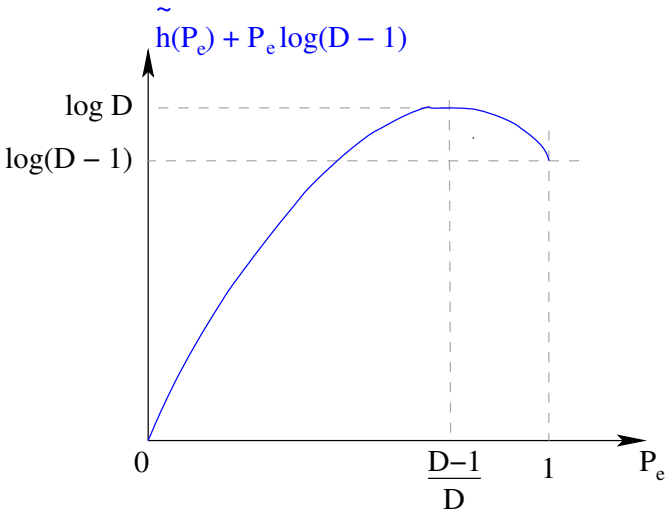


Figure 4.3: Evolution according to P_e of the bound on $H(U|\widehat{U})$ given by Fano's Lemma.

We can provide an interpretation of Fano's Lemma. Notice first that the function on the left of (4.18), sketched in figure 4.3, is concave in P_e) and is positive for all P_e such that $0 \leq P_e \leq 1$.

Thus, when a positive value of $H(U|\widehat{U})$ is given, (4.18) implicitly specifies a positive lower bound on P_e.

Example 4.8 Suppose that U and $\widehat{U}$ are binary-valued (i.e., $D = 2$) and that $H(U|\widehat{U}) = \frac{1}{2}$ bit. Then (4.18) gives

$$\widetilde{\mathrm{h}}(P_e) \geq \frac{1}{2}$$

or, equivalently,

$$.110 \leq P_e \leq .890.$$

The fact that here (4.18) also gives a non-trivial upper bound on P_e is a consequence of the fact that the given $H(U|\widehat{U})$ exceeds the value taken on by the left side of (4.18) when $P_e = 1$, i.e. $\log_2(D-1) = 0$.

This is not always the case. For instance, taking $D = 3$ and $H(U|\widehat{U}) = \frac{1}{2}$ bits, (4.18) becomes

$$\widetilde{\mathrm{h}}(P_e) + P_e \geq \frac{1}{2}$$

which leads to

$$P_e \geq .084,$$

but does not provide any useful upper bound. Only the trivial bound $P_e \leq 1$ can be asserted.

A review of the proof of Fano's Lemma reveals that equality holds in (4.18) if and only if the probability of an error given that $\widehat{U} = u$ is the same for all u and when there is an error (i.e., when $U \neq \widehat{U}$), the $n-1$ erroneous values of U are always equally likely. It follows that (4.18) gives the strongest possible lower bound on P_e, given only $H(U|\widehat{U})$

and the size, n, of the alphabet for U.

Summary for Section 4.2

Data Processing Lemma: $p_{Z|x,y}(z) = p_{Z|y}(z) \Longrightarrow I(X;Z) \leq I(X;Y)$ and $I(X;Z) \leq I(Y;Z)$

Fano's Lemma: $\widetilde{\mathrm{h}}(P_e) + P_e \log_2(|\mathcal{V}_U| - 1) \geq H(U|\widehat{U})$

4.3 The Noisy Coding Theorem

Learning Objectives for Section 4.3

This section is the core of the chapter and gives the important "*Noisy Coding Theorem*", which explains under what conditions reliable communication is possible or not.
After studying this section, you should be able to decide:

1. what is the maximum transmission speed you can use on a noisy communication channel to be able to construct reliable communication on this channel using error correcting codes;
2. what is the minimum amount of errors you are sure to make if you transmit information faster than this maximum.

4.3.1 Repetition Codes

The role of this section is to give a concrete example of very simple (naive) error-correcting codes.

As such, there is not much to be learned from this section, but the fact that naive coding is not very good and that other, more appropriate, codes should be considered.

We here consider *binary repetition codes*; i.e. codes $\mathcal{R}_k$ for which each (binary) input symbol is repeated $n = 2\ k + 1$ times ($k > 1$). For instance the code $\mathcal{R}_1$ was used in example 4.2. We consider only odd number of repetitions because decoding such codes is done by the majority. We thus avoid the non-determinism that even number of repetitions could introduce (in cases where the received codeword contains as many `0`s as `1`s).

With such codes, the probability of a wrong decision on decoding a symbol is the probability that at least $k + 1$ errors have occurred on the corresponding block.

Let us consider the case where such a code is used over a BSC. The number of errors made in this case by the channel has then a binomial distribution with parameters (n, p). Therefore, the expected number of errors at the level of the transmission of codeword symbols is $n\,p$.

For $p < 0.5$, this expected number is less than $k + 0.5$, therefore the probability that at least $k + 1$ errors have occurred on a codeword (i.e. one coding block), tends to

be negligible as k (hence n) tends to infinity. In other words, the probability that we take a wrong decision when decoding becomes negligible as the number of repetition increases (hence the length of the codewords). Thus we are able to compensate the loss due to the noise on the channel to any desired degree by choosing a large enough number of repetitions.

However, the price we pay for this is in this case quite huge in terms of efficiency. Indeed the transmission rate of such a code is $\frac{1}{n}$...
...which also tends to 0 as n grows to infinity! For large n this is likely to be an unacceptably low transmission rate.

The importance of the Noisy Coding Theorem is precisely that it ensures that good codes can be achieved for any given transmission rate below the capacity. The transmission rate can be fixed *a priori* and does not need to be ridiculously small as in the above example. It only requires to be below the channel capacity.

e-pendix: Repetition Codes over a BSC

Control Question 41

On a BSC with error probability p, we plan to use the repetition code $\mathcal{R}_1$ consisting of tripling each symbol of the messages. Decoding is done according to the majority in the received block.

What is, as a function of p, the output bit error probability P_b for such a communication system?

1. $p^2 - p^3$
2. p^2
3. $3p^2 - 2p^3$
4. $2p^2 - 3p^3$
5. p^3

Answer

The correct answer is 3.

When does an error occur in decoding? When 2 or 3 symbols of the block have been corrupted.

What is the probability of this to occur? The probability of one erroneous block with exactly two symbols are wrong is $p^2\,(1-p)$, and there are exactly $\binom{3}{2} = 3$ ways to have 2 wrong symbols among 3. Therefore the probability to get a wrong block with exactly 2 errors is $3\,p^2\,(1-p)$.

Similarly, the probability to get a wring block with exactly 3 errors is p^3.

Therefore:

$$P_b = 3\,p^2\,(1-p) + p^3 = 3\,p^2 - 2\,p^3$$

4.3.2 The Converse to the Noisy Coding Theorem for a DMC without Feedback

In this section, we show that it is impossible to transmit information reliably through a DMC at a transmission rate above the capacity of this DMC. Without loss of essential generality, we suppose that the "information" to be transmitted is the output from a *binary symmetric source* (*BSS**), which is a (memoryless) binary source with $P(0) = P(1) = \frac{1}{2}$. We here consider the case where the DMC is used without feedback (see Figure 4.1), for which we now give an important result on noisy transmissions (introduced by C. Shannon), known as the "Converse Part of the Noisy Coding Theorem".

Theorem 4.3 (Converse Part of the Noisy Coding Theorem) *If a BSS is used at rate R on a DMC without feedback of capacity C, and if $R > C$, then P_b, the bit error probability at the output, satisfies:*

$$P_b \geq \widetilde{\mathrm{h}}^{-1}\left(1 - \frac{C}{R}\right), \tag{4.24}$$

where $\widetilde{\mathrm{h}}^{-1}(x) = \min\{p : -p\log(p) - (1-p)\log(1-p) = x\}$.

Here, we have written $\widetilde{\mathrm{h}}^{-1}$ to denote the inverse binary entropy function defined by $\widetilde{\mathrm{h}}^{-1}(x) = \min\{p : -p\log(p) - (1-p)\log(1-p) = x\}$, where the minimum is selected in order to make the inverse unique (see figure 4.4).

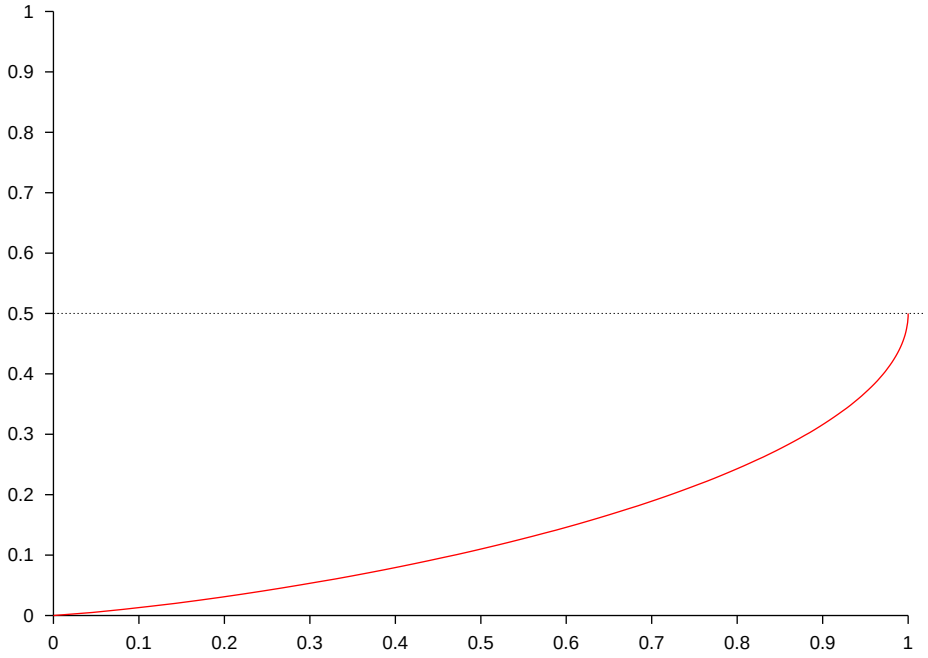


Figure 4.4: The function $\widetilde{\mathrm{h}}^{-1}$, the inverse binary entropy function defined by $\widetilde{\mathrm{h}}^{-1}(x) = \min\{p : -p\log(p) - (1-p)\log(1-p) = x\}$.

One important consequence of this fact is that, whenever $R > C$, (4.24) will specify a positive lower bound on the output bit error probability P_b that any coding system, as complex as it might be, can overcome. More clearly:. it is impossible to transmit any amount of information reliably at a rate bigger than the channel capacity.

Before going to the proof of the theorem, let us first give an example.

Example 4.9 (Converse Part of the Noisy Coding Theorem) Suppose we are using a DMC with capacity $C = \frac{1}{4}$ bit for transmitting message out of a BSS. If we transmit at a rate $R = \frac{1}{2}$, then (4.24) gives us an error of at least 11%:

$$P_b \geq \widetilde{\mathrm{h}}^{-1}\left(1 - \frac{1}{2}\right) = .11.$$

This means that at least 11% of the binary symbols will be transmitted incorrectly (even after error correction decoding).

PROOF We want to show that, when $R > C$, there is a positive lower bound on P_b that no manner of coding can overcome. As a first step in this direction, we note that the *output bit error probability* is

$$P_b = \frac{1}{m}\sum_{i=1}^{m} P(\widehat{U}_i \neq U_i). \tag{4.25}$$

where m is the length of the input message.

By the use of Fano's Lemma, we have in the binary case considered here:

$$\widetilde{\mathrm{h}}(P(\widehat{U}_i \neq U_i)) \geq H(U_i|\widehat{U}_i)$$

thus:

$$\sum_{i=1}^{m} H(U_i|\widehat{U}_i) \leq \sum_{i=1}^{m} \widetilde{\mathrm{h}}(P(\widehat{U}_i \neq U_i)) \tag{4.26}$$

To proceed further, we observe that

$$\begin{aligned} H(U_1 \ldots U_m|\widehat{U}_1 \ldots \widehat{U}_m) &= H(U_1|\widehat{U}_1) + \sum_{i=2}^{m} H(U_i|\widehat{U}_1 \ldots \widehat{U}_m U_1 \ldots U_{i-1}) \\ &\leq \sum_{i=1}^{m} H(U_i|\widehat{U}_i) \end{aligned} \tag{4.27}$$

since further conditioning can only reduce uncertainty.

Furthermore, because $\widetilde{\mathrm{h}}(x)$ is concave in x (for $0 \leq x \leq 1$):

$$\frac{1}{m}\sum_{i=1}^{m} \widetilde{\mathrm{h}}(P(\widehat{U}_i \neq U_i)) \leq \widetilde{\mathrm{h}}\left(\frac{1}{m}\sum_{i=1}^{m} P(\widehat{U}_i \neq U_i)\right) = \widetilde{\mathrm{h}}(P_b). \tag{4.28}$$

Thus, up to this point we have:

$$\widetilde{\mathrm{h}}(P_b) \geq \frac{1}{m} H(U_1...U_m|\widehat{U}_1...\widehat{U}_m).$$

Let us continue with $H(U_1...U_m|\widehat{U}_1...\widehat{U}_m)$:

$$\begin{aligned} H(U_1 \dots U_m|\widehat{U}_1 \dots \widehat{U}_m) &= H(U_1 \dots U_m) - I(U_1 \dots U_m; \widehat{U}_1 \dots \widehat{U}_m) \\ &= m - I(U_1 \dots U_m; \widehat{U}_1 \dots \widehat{U}_m) \end{aligned}$$

because in the case where the input source U is a BSS, $H(U_1 \dots U_m) = \log 2^m = m$.

What about $I(U_1 \dots U_m; \widehat{U}_1 \dots \widehat{U}_m)$?

If we consider the "Coding" and the "Channel" in Figure 4.1 respectively as "Processor #1" and "Processor #2" of Figure 4.2, using the Data Processing Lemma we have:

$$I(U_1 \dots U_m; \widehat{U}_1 \dots \widehat{U}_m) \leq I(z_1 \dots z_n; \widehat{U}_1 \dots \widehat{U}_m) \tag{4.29}$$

where z_i is the code of U_i (see figure 4.1).

Next, consider the "Channel" and the "Decoder" respectively as "Processor #1" and "Processor #2". Using the Data Processing Lemma again we conclude that:

$$I(z_1 \dots z_n; \widehat{U}_1 \dots \widehat{U}_m) \leq I(z_1 \dots z_n; \widehat{z}_1 \dots \widehat{z}_n). \tag{4.30}$$

Combining (4.29) and (4.30), we obtain:

$$I(U_1 \dots U_m; \widehat{U}_1 \dots \widehat{U}_m) \leq I(z_1 \dots z_n; \widehat{z}_1 \dots \widehat{z}_n).$$

which combined with lemma 4.3 leads to:

$$I(U_1 \dots U_m; \widehat{U}_1 \dots \widehat{U}_m) \leq n\,C \tag{4.31}$$

So we finally can conclude that

$$H(U_1...U_m|\widehat{U}_1...\widehat{U}_m) \geq m - n\,C$$

and thus

$$\widetilde{\mathrm{h}}(P_b) \geq 1 - \frac{n}{m} C$$

i.e., by definition of the transmission rate R:

$$\widetilde{\mathrm{h}}(P_b) \geq 1 - \frac{C}{R}$$

You may wonder whether (4.24) still holds when feedback is permitted from the output of the DMC to the channel encoder. The answer is a somewhat surprising yes. To arrive at (4.11), we made strong use of the fact that the DMC was used without feedback – in fact (4.11) may not hold when feedback is present. However, (4.31) can still be shown to hold so that the converse (4.24) still holds true when feedback is present. This fact is usually stated as saying that *feedback does not increase the capacity of a DMC*. This

does *not* mean, however, that feedback is of no value when transmitting information over a DMC. When $R < C$, feedback can often be used to simplify the encoder and decoder that would be required to achieve some specified output bit error probability.

Control Question 42

On a DMC without feedback of capacity C , we want to transmit message with a bit error probability P_b lower than a given value $P_{b\max} \in (0, \frac{1}{2})$.

We do not now already which kind of code will be used. However, we wish to determine the maximal transmission rate $R_{\max}$ we can use on that channel (and compatible with $P_b \leq P_{b\max}$).

What is $R_{\max}$ in terms of C and $P_{b\max}$?

1. C
2. $\widetilde{\mathrm{h}}^{-1}(P_{b\max})$
3. $C - \widetilde{\mathrm{h}}(P_{b\max})$
4. $\dfrac{C}{(1 - \widetilde{\mathrm{h}}(P_{b\max}))}$
5. $\dfrac{1}{(C - \widetilde{\mathrm{h}}^{-1}(P_{b\max}))}$
6. $\widetilde{\mathrm{h}}^{-1}(1 - C/P_{b\max})$

Answer

The correct answser is 4.

We are looking for the maximal transmission rate $R_{\max}$ we can use. Let us see if this maximal rate could be above the capacity C.

The converse part of the noisy coding theorem tells us that if we are transmitting at a rate $R = R_{\max} > C$ we have:

$$P_b \geq \widetilde{\mathrm{h}}^{-1}(1 - \frac{C}{R})$$

Since $P_{b\max} \geq P_b$, if $R = R_{\max} > C$ we also have:

$$P_{b\max} \geq \widetilde{\mathrm{h}}^{-1}(1 - \frac{C}{R})$$

And thus, h being non-decreasing on $(0, \frac{1}{2})$:

$$R \leq \frac{C}{1 - \widetilde{\mathrm{h}}(P_{b\max})}$$

Therefore, the maximal transmission rate we could use is

$$R_{\max} = \frac{C}{1 - \widetilde{\mathrm{h}}(P_{b\max})}.$$

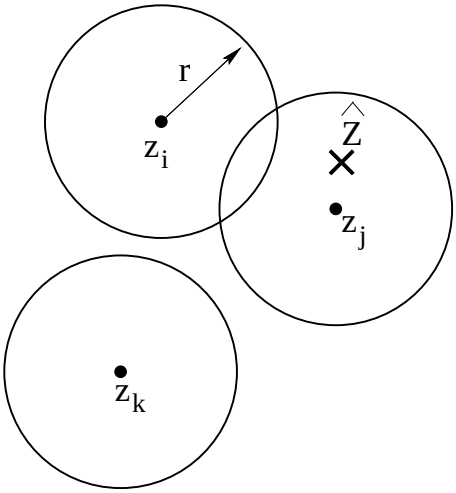


Figure 4.5: Example of decoding error in the random coding framework: z_i was send, $\widehat{z}$ was received and will here be decoded into z_j.

Indeed, if $R > R_{\max}$ then $R > C$ (since $\frac{C}{1-\widetilde{\mathrm{h}}(P_{b\max})} > C$), and therefore $P_b > \widetilde{\mathrm{h}}^{-1}(1 - \frac{C}{R}) > \widetilde{\mathrm{h}}^{-1}(1 - \frac{C}{R_{\max}}) = P_{b\max}$.

So if $R > R_{\max}$, we are sure that $P_b > P_{b\max}$.

Notice that this does not mean that if $R < R_{\max}$ we are sure to have $P_b \leq P_{b\max}$. We do not know if there exist such a code that $R = R_{\max}$ and $P_b \leq P_{b\max}$. All what we now is that if we transmit too fast ($R > R_{\max}$) we are sure to make too many errors ($P_b > P_{b\max}$).

4.3.3 The Noisy Coding Theorem for a DMC

Up to now, we have seen that it is impossible to have a transmission error below a certain level if the transmission rate R is bigger than the capacity. We now study what is going on when we wish to transmit a BSS over a DMC at a rate below its capacity.

Theorem 4.4 (Noisy Coding Theorem for a DMC) *Consider transmitting messages at rate R on a DMC without feedback of capacity C. For all $R < C$ and all $\varepsilon > 0$, there exists some (error-correcting) code whose rate is R and with an output bit error probability $P_b < \varepsilon$.*

This important theorem claims that by choosing an appropriate way of encoding information over a channel, one can reach as small error probability as wished.

PROOF At the level of this introductory course, let us, for the sake of simplicity, prove the theorem only in the case of a BSC. The proof will in this case be constructive in the sense that we actually construct a code verifying the theorem.

General proof of Shannon's theorem extends to DMC with arbitrary input and output

alphabet, and arbitrary channel capacity. The main idea of the general proof is the same, namely to code messages randomly and to decode with "nearest codeword" decision. Difficulties in the proof are caused mainly by the general form of the channel capacity when it is not a BSC. Interested readers can find complete proofs for the general situation in [1] or [6].

Let us now proceed with the proof in the case of a BSC.

The sketch of it is the following:

1. given R and ε, choose the appropriate codeword length n and number of codewords $M = 2^{Rn}$.
2. Then choose the M codewords $z_1, ..., z_M$ at random (as binary vectors of length n), without repetition. If $\mathcal{V}_U$, the number of possible binary messages to be transmitted, is bigger than M, then each input message will be split into pieces of at most n symbols which will be encoded separately (by one of the codewords z_i).
3. compute the "decoding threshold" r. This in fact corresponds to the maximum number of transmission errors the code is able to correct.
4. Use the following decoding procedure:
   - if there exists one and only one codeword z_i such that $d(\widehat{z}, z_i) \leq r$, decode as z_i.
     Here $\widehat{z}$ denotes the message received after transmission, i.e. at the output of the channel, and $d(x, y)$ is the Hamming distance between two binary strings, i.e. the number of different symbols between x and y.
   - otherwise decode as z_1.

Such a coding scheme (called "random coding") is enough to ensure $P_b < \varepsilon$ (provide the right n have been chosen).

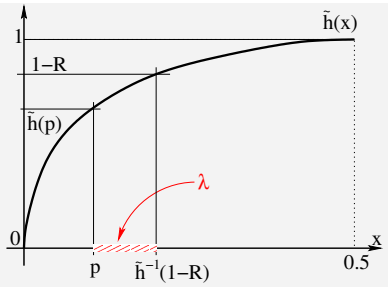
Let us now be more precise and go in the details.

First recall that the capacity of a BSC is $C = 1 - \widetilde{\mathrm{h}}(p)$, where p is the error probability of the channel and $\widetilde{\mathrm{h}}(x)$ is the binary entropy function: $\widetilde{\mathrm{h}}(x) = -x \log x - (1-x) \log(1-x)$.

Since $0 < R < C$, we have $1 > 1 - R > \widetilde{\mathrm{h}}(p)$, and therefore:

$$\exists \lambda,\ 0.5 \geq \lambda > p,\ \text{such that } R < 1 - \widetilde{\mathrm{h}}(\lambda)$$

as illustrated in this figure:



For such a λ, we have:

$$\lim_{n\to\infty}\left[\frac{p\,(1-p)}{n\,(\lambda-p)^2}+2^{n(R+\widetilde{\mathrm{h}}(\lambda)-1)}\right]=0$$

since $R+\widetilde{\mathrm{h}}(\lambda)-1<0$.

Therefore, for any given $\varepsilon>0$, there exists n_0 such that

$$\forall n\geq n_0\quad \frac{p\,(1-p)}{n\cdot(\lambda-p)^2}+2^{n\,(R+\widetilde{\mathrm{h}}(\lambda)-1)}\leq\varepsilon$$

For technical reasons that will appear clear later on, we also need to have n such that $\lfloor\lambda\,n\rfloor=\max\{q\in\mathbb{N}:q\leq\lambda\,n\}>n\,p$. This is the case provided that $n>n_1=\frac{1}{(\lambda-p)}$.

To summarize up to this point, we get to the following result:

$$\begin{aligned}&\forall p\in[0,0.5)\ \forall R,0<R<1-\widetilde{\mathrm{h}}(p),\forall\varepsilon>0,\\&\quad\Big(\exists n\in\mathbb{N}\ \exists\lambda\in(\widetilde{\mathrm{h}}^{-1}(1-R),p)\text{ such that }\frac{p\,(1-p)}{n\cdot(\lambda-p)^2}+2^{n\,(R+\widetilde{\mathrm{h}}(\lambda)-1)}\leq\varepsilon\\&\quad\text{and }\lfloor\lambda\,n\rfloor>n\,p\Big)\end{aligned}$$

This is in fact true for all $n>\max\{n_0,n_1\}$ above defined.

So we found one of "the appropriate codeword length n". Let us proceed as explain in the beginning with $M=2^{n\,R}$ codewords and $r=\lfloor\lambda\,n\rfloor=\max\{m\in\mathbb{N}:m\leq\lambda\,n\}$. In this framework, for a given codeword z_i, an error occurs when (see figure 4.5)

1– there have been more that r transmission errors: $d(\widehat{z},z_i)>r$
or
2– **a)** $d(\widehat{z},z_i)\leq r$
and
b) $\exists z\in\mathcal{C},z\neq z_i\ :\ d(\widehat{z},z)\leq r$
and
c) $i\neq 1$

where $\mathcal{C}=\{z_1,...,z_M\}$ denotes the code.

Therefore, the probability $P_{\mathrm{err}}(z_i)$ that a given codeword z_i is incorrectly transmitted (including decoding) is bounded by

$$P_{\mathrm{err}}(z_i)=P(\text{case 1})+P(\text{case 2})\leq P(d(\widehat{z},z_i)>r)+P(\exists z\in\mathcal{C}\setminus\{z_i\}\ :\ d(\widehat{z},z)\leq r)$$

Let us now find upper bounds for these two terms.

$d(\widehat{z}, z_i)$ is the number of transmission errors at the output of the BSC (i.e. before decoding). It is a binomial random variable with probability distribution $P(d(\widehat{z}, z_i) = k) = \binom{n}{k} p^k (1-p)^{n-k}$. Thus, its average and variance respectively are:

$$E[d(\widehat{z}, z_i)] = n\,p \qquad \mathrm{var}(d(\widehat{z}, z_i)) = n\,p\,(1-p)$$

Then, by Chebyshev's inequality we have (since $r > n\,p$)[2]:

$$P(|d(\widehat{z}, z_i) - n\,p| \geq r - n\,p) \leq \frac{n\,p\,(1-p)}{(r - n\,p)^2}$$

Therefore:

$$\begin{aligned} P(d(\widehat{z}, z_i) > r) &\leq P(d(\widehat{z}, z_i) \geq r) = P(d(\widehat{z}, z_i) - n\,p \geq r - n\,p) \\ &\leq P(|d(\widehat{z}, z_i) - n\,p| \geq r - n\,p) \\ &\leq \frac{n\,p\,(1-p)}{(r - n\,p)^2} \end{aligned} \tag{4.32}$$

For the second term ($P(\exists z \in \mathcal{C} \setminus \{z_i\} \,:\, d(\widehat{z}, z) \leq r)$), we have:

$$\begin{aligned} &P(\exists z \in \mathcal{C} \setminus \{z_i\} \,:\, d(\widehat{z}, z) \leq r) \\ &\quad = P(d(\widehat{z}, z_1) \leq r \text{ or } \ldots \text{ or } d(\widehat{z}, z_{i-1}) \leq r \text{ or } d(\widehat{z}, z_{i+1}) \leq r \text{ or } \ldots \text{ or } d(\widehat{z}, z_M) \leq r) \\ &\quad \leq \sum_{z \in \mathcal{C} \setminus \{z_i\}} P(d(\widehat{z}, z) \leq r) = (M-1) \cdot P(d(\widehat{z}, z) \leq r) \end{aligned}$$

since there are $M-1$ codewords z such that $z \neq z_i$.

Moreover, for a given $z \in \mathcal{C}$, the number of possible $\widehat{z}$ such that $d(\widehat{z}, z) = k$ is equal to the number of binary strings of length n that differ from z in exactly k positions. Thus, this number is $\binom{n}{k} = \frac{n\,!}{k\,!\,(n-k)\,!}$

Therefore, the total number of possible $\widehat{z}$ such that $d(\widehat{z}, z) \leq r$ is equal to

$$\sum_{k=0}^{r} \binom{n}{k}$$

and thus

$$P(d(\widehat{z}, z) \leq r) = \frac{1}{2^n} \sum_{k=0}^{r} \binom{n}{k}$$

So we get:

$$P(\exists z \in \mathcal{C} \setminus \{z_i\} \,:\, d(\widehat{z}, z) \leq r) \leq \frac{M-1}{2^n} \sum_{k=0}^{r} \binom{n}{k}$$

Moreover, as proven at the very end of this proof, we have for all r, $0 \leq r \leq \frac{n}{2}$:

$$\sum_{k=0}^{r} \binom{n}{k} \leq 2^{n\,\widetilde{\mathrm{h}}(\frac{r}{n})},$$

so (recall that $M = 2^{nR}$ and that $r \leq \frac{n}{2}$ since $\lambda < 0.5$) we finally found:

$$\begin{aligned} P(\exists z \in \mathcal{C} \setminus \{z_i\} \,:\, d(\widehat{z}, z) \leq r) &\leq \frac{2^{nR}-1}{2^n} 2^{nh(\frac{r}{n})} \\ &\leq 2^{n(R+\widetilde{\mathrm{h}}(\frac{r}{n})-1)} \end{aligned} \tag{4.33}$$

Now, regrouping equations (4.32) and (4.33) together, we find:

$$P_{\mathrm{err}}(z_i) \leq \frac{n\,p\,(1-p)}{(r-n\,p)^2} + 2^{n(R+\widetilde{\mathrm{h}}(\frac{r}{n})-1)}$$

which, by the initial choices of r and n is smaller than ε.

To conclude the proof we only have to notice that $P_b = \frac{1}{n} P_{\mathrm{err}}(z_i) \leq P_{\mathrm{err}}(z_i)$ (for all i).

The only missing step in the above proof is the proof of the following technical result:

$$\forall n \in \mathbb{N}\ \forall r \in \mathbb{N}, 0 \leq r \leq \frac{n}{2} \quad \sum_{k=0}^{r} \binom{n}{k} \leq 2^{n\ \widetilde{\mathrm{h}}(\frac{r}{n})}$$

which is now given.

$$\sum_{k=0}^{r} \binom{n}{k} = \sum_{k=0}^{n} \binom{n}{k} \delta(k-r)$$

$$\text{where} \qquad \delta(t) = \begin{cases} 1 & \text{if } t \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Thus, for all x, $0 < x \leq 1 : \delta(t) \leq x^t$, and therefore:

$$\sum_{k=0}^{r} \binom{n}{k} \leq \sum_{k=0}^{n} \binom{n}{k} x^{k-r}$$

$$\text{i.e.} \qquad \sum_{k=0}^{r} \binom{n}{k} \leq \frac{(1+x)^n}{x^r}$$

which is in particular true for $x = \frac{r}{n-r}$, with $r \leq n/2$:

$$\sum_{k=0}^{r} \binom{n}{k} \leq \frac{(1+\frac{r}{n-r})^n}{(\frac{r}{n-r})^r}$$

$$\text{i.e.} \qquad \sum_{k=0}^{r} \binom{n}{k} \leq 2^{n \cdot \log(1+\frac{r}{n-r}) - r \cdot \log(\frac{r}{n-r})}$$

$$\begin{aligned} \text{But:} \quad & n \cdot \log(1+\frac{r}{n-r}) - r \cdot \log(\frac{r}{n-r}) \\ = \; & n \cdot \left(\log(\frac{1}{1-\frac{r}{n}}) - \frac{r}{n} \log(\frac{\frac{r}{n}}{1-\frac{r}{n}}) \right) \\ = \; & n \cdot \left(-\frac{r}{n} \log \frac{r}{n} - (1-\frac{r}{n}) \log(1-\frac{r}{n}) \right) \\ = \; & n \cdot \widetilde{\mathrm{h}}(\frac{r}{n}) \end{aligned}$$

which concludes the proof. ■

Control Question 43

Consider using a BSC with an error probability p (whose capacity is therefore $C = 1 - \widetilde{\mathrm{h}}(p)$). In the following cases, tell if a code fulfilling the requirement could be constructed:

channel	p	5%						10%					
	C	0.801						0.675					
code	R	2/3		3/4		9/10		2/3		3/4		9/10	
	P_b (in %)	1	2.2	1	2.2	1	2.2	1	2.2	1	2.2	1	2.2
exists?													

Answer

If $R < C$ we are sure that there exists a code with an output bit error probability P_b as small as we wish.

On the other hand, if $R > C$ we cannot have $\widetilde{\mathrm{h}}(P_b) < 1 - \frac{C}{R}$.

Finally, if $R > C$ and $\widetilde{\mathrm{h}}(P_b) \geq 1 - \frac{C}{R}$ we cannot conclude since the situation could be possible (i.e. does not contradict the theorem) but we do not know enough about coding so as to be sure that such a code actually exists.

So here are the conclusions:

channel	p	5%					
	C	0.801					
code	R	2/3		3/4		9/10	
	P_b (in %)	1	2.2	1	2.2	1	2.2
$R < C$?		yes		yes		no	
$1 - C/R$		–		–		0.109	
$\widetilde{\mathrm{h}}(P_b)$		–		–		0.081	0.153
exists?		yes	yes	yes	yes	no	maybe

channel	p	10%					
	C	0.675					
code	R	2/3		3/4		9/10	
	P_b (in %)	1	2.2	1	2.2	1	2.2
$R < C$?		yes		no		no	
$1 - C/R$		–		0.100		0.250	
$\widetilde{\mathrm{h}}(P_b)$		–		0.081	0.153	0.081	0.153
exists?		yes	yes	no	maybe	no	no

Summary for Section 4.3

Shannon's Noisy Coding Theorem: For all $\varepsilon > 0$ and all $R < C$, C being the capacity of a DMC, there exists a code, the transmission rate of which is R and the output bit error probability P_b of which is below ε.

Conversely, all codes for which the transmission rate is above the channel capacity have a output bit error probability P_b greater than $\widetilde{\mathrm{h}}^{-1}\left(1 - \frac{C}{R}\right)$.

Summary for Chapter 4

Channel: input and output alphabets, transmission probabilities $(p_{Y|X=x})$.

DMC = Discrete Memoryless Channel.

Channel Capacity: $C = \max_{p_X} I(X;Y)$

Code/Transmission Rate: $R_b = \frac{\log_b |\mathcal{V}_U|}{n}$

Capacity of Input-symmetric channels: $C = \max\limits_{P_X} [H(Y)] - H(Y|X = x_i)$ for any of the $x_i \in \mathcal{V}_X$.

Capacity of Symmetric channels: $C = \log |\mathcal{V}_Y| - H(Y|X = x_i)$ for any of the $x_i \in \mathcal{V}_X$.

Data Processing Lemma: $p_{Z|x,y}(z) = p_{Z|y}(z) \implies I(X;Z) \leq I(X;Y)$ and $I(X;Z) \leq I(Y;Z)$

Fano's Lemma: $\widetilde{\mathrm{h}}(P_e) + P_e \log_2(|\mathcal{V}_U| - 1) \geq H(U|\widehat{U})$

Shannon's Noisy Coding Theorem: For all $\varepsilon > 0$ and all $R < C$, C being the capacity of a DMC, there exists a code, the transmission rate of which is R and the output bit error probability P_b of which is below ε.

Conversely, all codes for which the transmission rate is above the channel capacity have a output bit error probability P_b greater than $\widetilde{\mathrm{h}}^{-1}\left(1 - \frac{C}{R}\right)$.

Historical Notes and Bibliography

This theorem was the "bombshell" in Shannon's 1948 paper [10]. Prior to its publication, it was generally believed that in order to make communications more reliable it was necessary to reduce the rate of transmission (or, equivalently, to increase the "signal-to-noise-ratio", as the 1947 engineers would have said). Shannon dispelled such myths, showing that, provided that the rate of transmission is below the channel capacity, increased reliability could be purchased entirely by increased complexity in the coding system, with no change in the signal-to-noise-ratio.

The first rigorous proof of Shannon's noisy coding theorem was due to Feinstein in 1954 [4]. A simpler proof using random coding was published by Gallager in 1965 [5]. The converse part of the theorem was proved by Fano in 1952, published in his class notes.

OutLook

The Noisy Coding Theorem lacks practical considerations, for it does not give concrete ways to construct good codes efficiently. For instance, no indication of how large the codewords need to be for a given ε; i.e. how complex the encoder and decoder need to be to achieve a given reliability. This is the reason why "*coding theory*" has become a so important field: finding good error correcting codes, "good" in the sense that the error probability is low but the transmission rate is high, is indeed challenging.

Chapter 5

Module I1: Complements to Efficient Coding of Information

by J.-C. Chappelier

Learning Objectives for Chapter 5

In this chapter, we present several different complements to the basics of efficient coding, i.e. data compression.
Studying this chapter, you should learn more about

1. how to perform optimal variable-to-fixed length coding (Tunstall codes);
2. how to simply and efficiently encode the integers with a prefix-free binary code (Elias code);
3. some of the techniques used for coding sequences with inner dependencies (stationary source coding), as for instance the famous Lempel-Ziv coding.

5.1 Variable-to-Fixed Length Coding: Tunstall's Code

Learning Objectives for Section 5.1

In this section, you will learn:

- what "variable-to-fixed length" coding is about;
- what "proper message sets" are and why are they useful;
- how the first part of Shannon Noiseless Coding Theorem generalizes to variable-to-fixed length coding of proper sets;
- what "Tunstall message sets" are;
- and what are they useful for: providing optimal variable-to-fixed length coding;
- how to build such sets, i.e. Tunstall codes.

5.1.1 Introduction

The variable length codewords considered in chapter 2 are not always convenient in practice. If the codewords are, for instance, to be stored in memory, codewords the length of which is equal to the memory word length (e.g. 8, 16 or 32 bits) would certainly be preferred. However, it was precisely the variability of the codeword lengths that provided efficiency to the codes presented in chapter 2! So the question is if it is possible to get similar coding efficiency when all codewords are forced to have the same length? The answer is yes, provided that codewords are no longer assigned to fixed length blocks of source symbols but rather to variable-length sequences of source symbols, i.e. *variable-length segmentation* of the source stream must be achieved. This is called "*Variable-to-Fixed Length* coding": the D-ary codewords have all the same length n, but the length, L_V, of the messages V to which the codewords are assigned, is a random variable.

Since $n/E[L_V]$ is the average number of D-ary code digits per source symbol, the optimality criterion of such a code becomes $E[L_V]$, the average encoded message length; which should be made as large as possible.

5.1.2 Proper Sets

What properties should variable-to-fixed length codes have?

In order to be prefix-free, the codewords should correspond to the leaves of a coding tree (see property 2.4). Furthermore, in order to be able to code any sequence from the source, the coding tree must be *complete* (see definition 2.8). If indeed the code is not complete, the sequence of symbols corresponding to unused leaves can not be encoded!

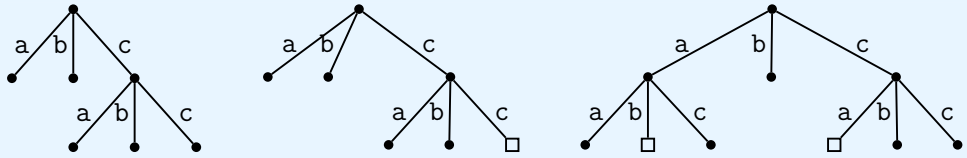
A variable-to-fixed length code is thus required to be a *proper code*; i.e. its codewords should form a *proper set*.

Definition 5.1 (Proper Set) A set of messages is a *proper set* if and only if it corresponds to the complete set of leaves of a coding tree. ♦

Example 5.1 (Proper Set) The set $\{\texttt{a}, \texttt{b}, \texttt{ca}, \texttt{cb}, \texttt{cc}\}$ is a proper set.

The sets $\{\texttt{a}, \texttt{b}, \texttt{ca}, \texttt{cb}\}$ and $\{\texttt{aa}, \texttt{ac}, \texttt{b}, \texttt{cb}, \texttt{cc}\}$ are not proper sets.

Here are the corresponding coding trees:



Control Question 44

For each of the following sets, decide whether it is a proper set or not:

1. 010, 00, 1, 011
2. 000, 010, 001, 01, 1
3. 110, 010, 011, 10, 00
4. 110, 011, 00, 010, 111, 10

Answer

1. yes.
2. no: it is not prefix free.
3. no: it is not complete.
4. yes.

Theorem 5.1 *The uncertainty $H(V)$ of a proper set V for a D-ary discrete memoryless information source, the uncertainty of which is $H(U)$, satisfies:*

$$H(V) = E[L_V] \cdot H(U),$$

where $E[L_V]$ is the average encoded message length.

PROOF The coding tree corresponding to a proper set is by definition a complete tree, and thus the entropy of each of its internal nodes is equal to the entropy of the source U.

Then, from the leaf entropy theorem (theorem 2.2), we have:

$$H(V) = \sum P_i H(U) = \left(\sum P_i\right) \cdot H(U)$$

et from the Path Length Lemma (lemma 2.1):

$$H(V) = E\left[L_V\right] \cdot H(U)$$

We now can see how Shannon Noiseless Coding Theorem (part 1) applies to proper sets of memoryless information sources:

Theorem 5.2 *For any D-ary prefix-free encoding Z of any proper message set V for a discrete memoryless information source U, the ratio of the average codeword length $E\left[L_Z\right]$, to the average encoded message length, $E\left[L_V\right]$, satisfies*

$$\frac{H(U)}{\log D} \leq \frac{E\left[L_Z\right]}{E\left[L_V\right]}$$

where $H(U)$ is the uncertainty of a single source symbol.

PROOF From Theorem 5.1:

$$H(V) = E\left[L_V\right] H(U)$$

and from the Shannon Noiseless Coding Theorem (theorem 2.4):

$$\frac{H(V)}{\log D} \leq E\left[L_Z\right],$$

thus:

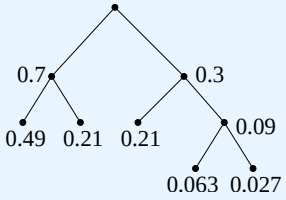
$$\frac{H(U)}{\log D} E\left[L_V\right] \leq E\left[L_Z\right].$$

5.1.3 Tunstall message sets

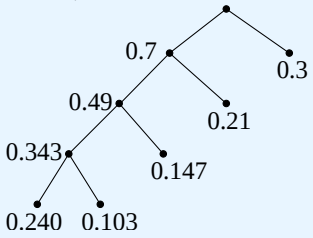
Next section considers the effective procedure for building efficient variable-to-fixed length codes, but we first require another definition; which is the topic of this section.

Definition 5.2 (Tunstall message set) A set of messages is a *Tunstall message set* if and only if it is a proper set such that, in the corresponding coding tree, every node is at least as probable as every leaf. ♦

Example 5.2 (Tunstall set)



This is not a Tunstall set since there is a leaf and an internal node such that the probability of that leaf (0.49) is bigger than the probability of that internal node (0.3).



This is a Tunstall tree since every internal node is more probable than every leaf.

Tunstall message sets are optimal variable length to block coding i.e. provide maximal average length of encoded messages as the following theorem states:

Theorem 5.3 *A proper message set maximizes the average encoded message length (over all proper message sets) if and only if it is a Tunstall message set.*

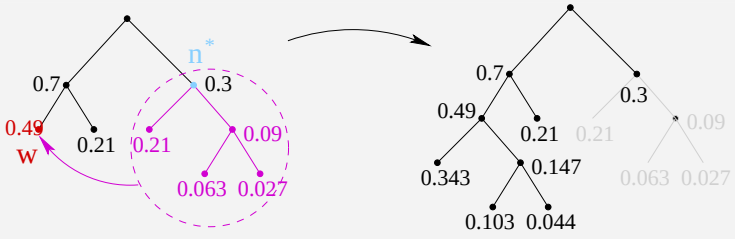
PROOF Let us first prove that is a proper set is not a Tunstall set then it cannot maximize the average encoded message length.

Let W be a proper set that is not a Tunstall set. Therefore there exist in the corresponding coding tree a leaf w and an internal node n^* such that

$$P(n^*) < P(w_i)$$

Consider then the coding tree consisting of moving the subtree below n^* to the leaf w (n^* thus becomes a leave and w an internal node).

With the former example:



The tree obtained by this operation still defines a proper set.

Furthermore, since $P(n^*) < P(w_i)$ the probability of all the nodes of the moved subtree are greater in the new coding tree than in the original tree.

Thus, the average encoded message length, which according to the Path Length Lemma is the sum of all the internal nodes probabilities, is greater for the new message set than for the former.

Thus the former message set could not perform the maximum average encoded message length.

Conversely, using a similar argument, a tree that is not maximal cannot be a Tunstall set: consider the upper node where it differs from a tree that is maximal. ■

5.1.4 Tunstall Code Construction Algorithm

Let n be the desired size of the codewords, and let D_U and D_Z respectively be the arity of the source U and the arity of the code Z. The maximum number of codewords is thus ${D_Z}^n$.

We want to build a Tunstall set of size M for U ($M \leq {D_Z}^n$), i.e. we are looking (among other things) for a complete coding tree, i.e. with no unused leaves. Thus we must have M in the form of (see lemma 2.3)

$$M = 1 + k\,(D_U - 1)$$

for some k.

To be optimal we are looking for *maximal* M (with $M \leq {D_Z}^n$). Thus, we must choose:

$$k = \left\lfloor \frac{{D_Z}^n - 1}{D_U - 1} \right\rfloor$$

This leads to the following algorithm:

Tunstall's Algorithm for Optimum Variable-to-Fixed Length Coding

1. Check whether ${D_Z}^n \geq D_U$. If not, abort: variable-to-fixed length coding is impossible.
2. Compute $k = \lfloor \frac{{D_Z}^n - 1}{D_U - 1} \rfloor$
3. Compute the encoded message set size M as $M = 1 + k\,(D_U - 1)$
4. Construct the Tunstall coding tree of size M (i.e. M leaves) by repeating k times (root included):
   - extend (with D_U branches) the most probable node

   It is easy to check that the obtained set is indeed a Tunstall set.
5. Assign a distinct D_Z-ary codeword of length n to each leaf, i.e. to each message in the Tunstall message set.

Example 5.3 $D_Z = 2, n = 3$ (i.e. codeword = 3 bits)
U ternary source ($D_U = 3$) such that $P(U = -1) = 0.3$, $P(U = 0) = 0.2$ and $P(U = 1) = 0.5$.

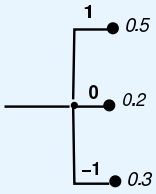
Then we have:

$$k = \lfloor (8 - 1)/2 \rfloor = 3$$

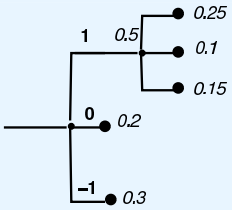
and

$$M = 1 + 3 \cdot 2 = 7$$

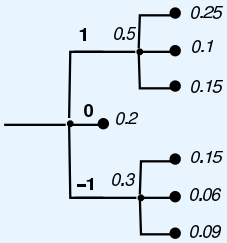
k loops:
1) extend the root



2) extend the most probable node:



3) extend the most probable node:



Finally, affect codewords:

V	1,1	1,0	1,-1	0	-1,1	-1,0	-1,-1
Z	000	001	010	011	100	101	110

Control Question 45

We are interested in an optimal 4 bits binary variable-to-fixed length code of the ternary source, the symbols probabilities of which are $P(U = \texttt{a}) = 0.6$, $P(U = \texttt{b}) = 0.3$ and $P(U = \texttt{c}) = 0.1$.

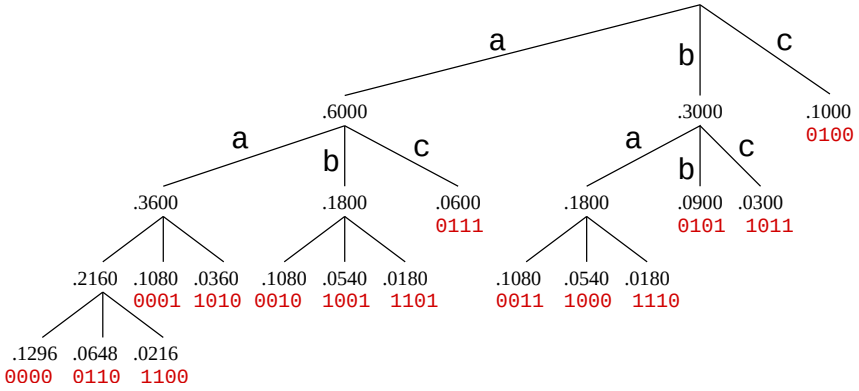
1. How many codewords has this code?
2. How many steps are required to build the tree?
3. How is the input message `acaaabaaaabbaaabbbc` segmented (i.e. split into parts to be encoded)?
4. How is `acaaabaaaabbaaabbbc` (same message) encoded, using the convention that the leaves are numbered according to increasing probabilities?

Answer

1. $M = 15$
2. $k = 7$
3. `ac,aaab,aaaa,bb,aaab,bb,c`

4. 011101100000010101100101010 0: 0111, 0110, 0000, 0101, 0110, 0101, 0100.

Here is the corresponding coding tree:



Summary for Chapter 5

variable-to-fixed length coding: encoding segment of variable length of the input information source with codewords that have all the same size

proper set: a complete set of leaves of a coding tree (no useless leaf).

Tunstall set: a proper set such that every node of the corresponding coding tree is at least as probable as any of its leaves.

optimality of Tunstall sets: A proper set maximizes the average encoded message length if and only if it is a Tunstall set.

Tunstall algorithm: $k = \lfloor \frac{D_Z{}^n - 1}{D_U - 1} \rfloor$, $M = 1 + k\ (D_U - 1)$, extend k times the most probable node.

5.2 Coding the Positive Integers

Learning Objectives for Section 5.2

In this section, we study how to simply and efficiently encode the integers with a prefix-free binary code (Elias code).

Let us now turn to another, completely different, aspect of efficient coding: how to efficiently represent integers with binary codes? In this section we describe a very clever binary prefix-free code for the positive integers that was brought by Elias.

Let us start from a usual code for integer the reader should be awared of: (most significant bit) binary representation. Here is an example of this code, we call it here Z_0:

n	1	2	3	4	5	6	7	8
$Z_0(n)$	1	10	11	100	101	110	111	1000

The length of these codewords is:

$$|Z_0(n)| = \lfloor \log_2 n \rfloor + 1$$

which is pretty close to theoretical optimum in the most general case ($\log_2 n$).

However, this code suffers a major drawback: it is far from being prefix-free. In fact, every codeword $Z_0(n)$ is the prefix of infinitely many other codewords!

The first idea brought by Elias was to add an encoding of the length $|Z_0(n)|$ in front of $Z_0(n)$ to make the code prefix-free. The encoding proposed is to add $|Z_0(n)| - 1$ zeros in front of $Z_0(n)$. Here is an example of this code, we call Z_1:

n	1	2	3	4	5	6	7	8
$Z_1(n)$	1	010	011	00100	00101	00110	00111	0001000

Z_1 is now a prefix-free code. However, it length is far from being the desired one: it is twice as long:

$$|Z_1(n)| = 2\lfloor \log_2 n \rfloor + 1$$

The clever trick used by Elias to get rid of this drawback was to change the encoding of the length made by zeros for the Z_1 encoding of this length. A codeword is thus made of the concatenation of $Z_1(|Z_0(n)|)$ and $Z_0(n)$.

For instance, 7 is encoded into 111 with Z_0, having thus a length of 3. This leads to the prefix $Z_1(3)$ =011, and therefore 7 is encoded into 011111 (=011,111). Here are more examples:

n	1	2	3	4	5	6	7	8
$Z_2'(n)$	11	01010	01011	011100	011101	011110	011111	001001000

Notice that $Z_0(n)$ is always starting with a 1, which is now no longer required to avoid ambiguity. We thus can get ride of this useless 1. This leads us to the final Elias encoding for integers, here denoted by Z_2:

n	1	2	3	4	5	6	7	8
$Z_2(n)$	1	0100	0101	01100	01101	01110	01111	00100000

This code is prefix-free. What about its length?

- for the main part:
$$|Z_0(n)| = \lfloor \log_2 n \rfloor + 1$$
- and for the prefix:
$$|Z_1(|Z_0(n)|)| = 2\lfloor \log_2(\lfloor \log_2 n \rfloor + 1) \rfloor + 1$$

Thus:

$$\begin{aligned} |Z_2(n)| &= 2\lfloor \log_2(\lfloor \log_2 n \rfloor + 1) \rfloor + 1 + \lfloor \log_2 n \rfloor + 1 - 1 \\ &= \lfloor \log_2 n \rfloor + 2\lfloor \log_2(\lfloor \log_2 n \rfloor + 1) \rfloor + 1 \end{aligned} \tag{5.1}$$

It is remarkable enough that Elias found a binary prefix-free code for integers, the length of which is quite close to the optimal $\log_2 n$, and which is furthermore easily implementable.

Control Question 46

What is the Elias encoding for 255?

Answer

$Z_2(255) =$00010001111111.

Indeed: $Z_0(255) =$11111111 and $Z_1(|Z_0(255)|) = Z_1(8) =$0001000

Summary for Chapter 5

Elias Codes are prefix-free binary codes for integers, whose length is (asymptotically) close to optimal $\log_2(n)$.

These codes result from the concatenation of a prefix, made of the first Elias code of the length of the usual binary representation of the number to be encoded, and a suffix, made of the usual binary representation without its most significant bit.

5.3 Coding of Sources with Memory

Learning Objectives for Section 5.3

After studying this session, you should know:

1. how first part of Shannon Noiseless Coding Theorem generalizes to stationary sources;
2. several methods to perform compression of stationary sources message flows:
   (a) Elias-Willems Recency encoding;
   (b) Lempel-Ziv codes.

Let us finish this chapter by touching upon the difficult subject of coding sources with memory, i.e. with internal dependencies in the symbol sequences.

We have see in chapter 2 that Huffman coding is optimal for coding a fixed number of independent and identically distributed random variables (i.e. several independent occurrences of the same source). What about the case where the symbols are dependent?

The sources considered in this section are thus stationary stochastic process (see definitions 3.1 and 3.2 in chapter 3); more precisely, sources that emit sequences $U_1, U_2, U_3, \ldots$ of D-ary random variables such that for every $t \geq 1$ and every $n \geq 1$, the

random vectors $(U_1, U_2, \ldots, U_n)$ and $(U_{t+1}, U_{t+2}, \ldots, U_{n+n})$ have the same probability distribution.

Example 5.4 Let us consider as an example of a source with internal dependencies, the "oscillatory source" U consisting of a stationary binary source such that $p_U(\mathtt{0}) = p_U(\mathtt{1}) = 0.5$ and:

$$P(U_i = \mathtt{0}|U_{i-1} = \mathtt{0}) = 0.01 \quad P(U_i = \mathtt{1}|U_{i-1} = \mathtt{0}) = 0.99$$
$$P(U_i = \mathtt{0}|U_{i-1} = \mathtt{1}) = 0.99 \quad P(U_i = \mathtt{1}|U_{i-1} = \mathtt{1}) = 0.01$$

(and no other longer term dependencies, i.e. $P(U_i|U_{i-1}...U_1) = P(U_i|U_{i-1})$, otherwise at least one of the above equations could not hold).

The entropy of a single symbol of this source is clearly $H(U) = 1$ bit. What about the entropy rate?

$$\begin{aligned} h_\infty &= \lim_{i\to\infty} H(U_i|U_{i-1}U_{i-2}...U_1) \\ &= \lim_{i\to\infty} H(U_i|U_{i-1}) \\ &\overset{\text{stationarity}}{=} H(U_2|U_1) \\ &= P(U_1 = \mathtt{0}) \cdot \widetilde{\mathrm{h}}(0.01) \\ &\quad + P(U_1 = \mathtt{1}) \cdot \widetilde{\mathrm{h}}(0.99) \\ &= 2 \cdot 0.5 \cdot \widetilde{\mathrm{h}}(0.01) \\ &= 0.081 \text{ bit} \end{aligned}$$

where $\widetilde{\mathrm{h}}(p)$ is the entropy of a binary random variable of parameter p:
$\widetilde{\mathrm{h}}(p) = -p\log(p) - (1-p)\log(1-p)$.

For such discrete stationary sources, the first part of the Shannon Noiseless Coding Theorem generalizes as:

Theorem 5.4 *The average length $E[L_Z]$ of a prefix-free D-ary code Z for segments of length k of a stationary discrete source U verifies:*

$$\frac{h_\infty(U)}{\log D} \leq \frac{E[L_Z]}{k}$$

where $h_\infty(U)$ is the entropy rate of the source U as defined in theorem 3.1.

PROOF The proof comes directly from the first part of the Shannon Noiseless Coding Theorem and the properties of entropy rates. Indeed:

$$\frac{k \cdot h_\infty(U)}{\log D} \leq \frac{H(U_1...U_k)}{\log D} \leq E[L_Z]$$

5.3.1 Huffman coding of slices

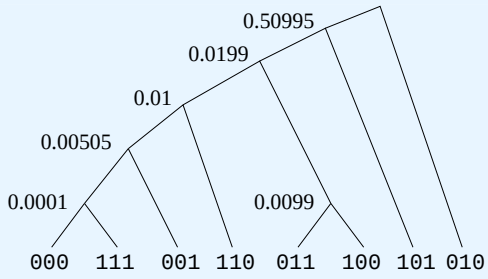
One simple way to efficiently encode stationary sources consists in segmenting the source symbol flow into sequences of fixed length k and consider this new source, hereby designed as $U^{(k)}$, for (memoryless) Huffman coding.

For instance, if the original flow is `001010011010...` and we take $k = 3$, we will consider the messages `001`, `010`, `011`, etc... separately.

The larger k, the more dependencies $U^{(k)}$ handles and the more efficient the corresponding Huffman coding will be. This solution is unfortunately requiring too much computation power to be used in practice for large k.

Example 5.5 Here is the Huffman code of the oscillator source of example 5.4 with slices of size 3.

U	$P(U)$
`000`	$0.5 \cdot 0.01 \cdot 0.01 = .00005$
`001`	$0.5 \cdot 0.01 \cdot 0.99 = .00495$
`010`	$0.5 \cdot 0.99 \cdot 0.99 = .49005$
`011`	$0.5 \cdot 0.99 \cdot 0.01 = .00495$
`100`	$0.5 \cdot 0.99 \cdot 0.01 = .00495$
`101`	$0.5 \cdot 0.99 \cdot 0.99 = .49005$
`110`	$0.5 \cdot 0.01 \cdot 0.99 = .00495$
`111`	$0.5 \cdot 0.01 \cdot 0.01 = .00005$



$E[L] = 1 + 0.50995 + 0.0199 + 0.01 + 0.0099 + 0.00505 + 0.0001 = 1.5549$

$= 0.5183$ per symbol

5.3.2 Elias-Willems Source Coding Scheme

The Elias-Willems scheme for coding source with memory is the following:

1. split the source symbol flow in blocks of size k (i.e. $U^{(k)}$ as in the former section)
2. code each block with the Elias code of its recency distance[1].

The only piece in the above scheme we are still at this point lacking is the "*recency distance*".

[1]Elias-Willems is actually using recency rank, which is smaller than recency distance but does not affect the general results presented here.

Definition 5.3 (Recency Distance) The *Recency Distance* of a symbol v at position n in a symbol sequence V is $R_n(v) = n - L_n(v)$, where $L_n(v)$ is the last index t (before n) such that $V_t = v$:

$$L_n(v) = \max\{t < n : V_t = v\}$$

$R_n(v)$ is thus the number of symbols received at time n since the last reception of v (before n).
For recency distance to be defined for every possible n (even the first ones), a convention needs to be chosen to give an initial index value to every possible symbols. ♦

Definition 5.4 (Recency Distance Series) The *recency distance series* N associated with a random process V is the random process defined by $N_n = R_n(V_n)$. ♦

Example 5.6 (Recency Distance) As first example, consider the sequence `01011010` out of a binary source. The corresponding recency distances, with the convention that `0` has default initial index -1 and symbol `1` has 0, are then:

v_n	$R_n(0)$	$R_n(1)$	N_n
0	2	1	2
1	1	2	2
0	2	1	2
1	1	2	2
1	2	1	1
0	3	1	3
1	1	2	2
0	2	1	2

And the corresponding recency distance series is thus 2,2,2,2,1,3,2,2.

For a second example, consider the source V the symbols of which are 2 bits words: `00`, `01`, ... and the convention that `00` has initial default index value -3, `01` -2, `10` -1 and `11` 0.

For the sequence `11,01,00,10,01,11,01,01,00`, the recency distance series will thus be 1,4,6,5,3,5,2,1,6.

Control Question 47

Considering a binary source with single bit symbols and the convention that `0` has initial default index -1 and `1` 0, what is the recency distance series for the corresponding source sequence: `0001101000111`?

Answer

2,1,1,4,1,3,2,2,1,1,4,1,1.

Control Question 48

Considering a binary source with single bit symbols and the convention that `0` has initial default index -1 and `1` 0, what is source sequence corresponding to the recency distance series: 1,3,1,3,1,3,2,2?

Answer

`10011010`

Here comes now a property that will be useful to see how efficient the Elias-Willems scheme is.

Property 5.1 *If the source V is stationary and* ergodic[a]*, then the symbol v appears on average every $1/p_V(v)$ times:*

$$E\left[N_i | V_i = v\right] = E\left[R_i(v)\right] = \frac{1}{p_V(v)}$$

[a] If you do not know what "ergodic" means, do not bother too much here, since almost all interesting real information sources are ergodic. If you want to know, look in any good book of probabilities, dealing with random process.

PROOF Let $k_n(v)$ be the number of occurrences of v in a sequence of length n from source V. The number of intervals between two consecutive repetitions of v is thus $k_n(v)$ and the total length of these intervals is n. The average interval on a sequence of length n is thus $n/k_n(v)$. When n grows to infinity, because the source is ergodic, $k_n(v)$ tends to infinity as $(n \cdot p_V(v))$ and thus

$$E\left[R_i(v)\right] = \lim_{n\to\infty} \frac{n}{k_n(v)} = \frac{1}{p_V(v)}$$

Let us now prove that Elias coding of recency distance performs a coding that is asymptotically efficient, i.e. asymptotically to the theoretical optimum given by theorem 5.4.

Theorem 5.5 *The expected length $E\left[|Z_2(N)|\right]$ of a Elias-Willems encoding with blocks of size k of a stationary source U verifies:*

$$h_k(U) \leq \frac{E\left[|Z_2(N)|\right]}{k} \leq h_k(U) + \frac{2}{k}\log_2(k \cdot h_k(U) + 1) + \frac{1}{k}$$

Corollary 5.1 *The expected length $E\left[|Z_2(N)|\right]$ of a Elias-Willems encoding with blocks of size k of a stationary source U verifies:*

$$\lim_{k\to\infty} \frac{E\left[|Z_2(N)|\right]}{k} = h_\infty(U)$$

PROOF What is the expected length of a Elias-Willems code?

$$E\left[|Z_2(N)|\right] = \sum_{v \in V} p_V(v) \cdot E\left[|Z_2(N_i(V_i))||V_i = v\right] = \sum_{v \in V} p_V(v) \cdot E\left[|Z_2(R_i(v))|\right]$$

But, from equation 5.2 we have:

$$E\left[|Z_2(R_i(v))|\right] \leq \log_2(R_i(v)) + 2\log_2(\log_2(R_i(v)) + 1) + 1$$

Using Jensen inequality we have:

$$E\left[|Z_2(N)|\right] \leq \sum_{v \in V} \log_2(E\left[R_i(v)\right]) + 2\log_2(\log_2(E\left[R_i(v)\right]) + 1) + 1$$

and from property 5.1

$$E\left[|Z_2(N)|\right] \leq -\sum_{v \in V} \log_2(p_V(v)) + 2\sum_{v \in V} \log_2(1 - \log_2(p_V(v))) + 1$$

i.e., using Jensen inequality once again:

$$E\left[|Z_2(N)|\right] \leq H(V) + 2\log_2(H(V) + 1) + 1$$

Notice finally that $H(V) = H(U^{(k)}) = H(U_1, ..., U_k) = k \cdot h_k(U)$, thus:

$$\frac{E\left[|Z_2(N)|\right]}{k} \leq h_k(U) + \frac{2}{k}\log_2(k \cdot h_k(U) + 1) + \frac{1}{k}$$

Control Question 49

How is the sequence `100000101100` encoded using Elias-Willems scheme with $k = 2$ and the convention that `00` has initial default index value -3, `01` -2, `10` -1 and `11` 0?

Answer

The answer is `01000110110101011010101`.

With $k = 2$, `100000101100` is split into `10,00,00,10,11,00` which corresponds to the recency distance series: 2,5,1,3,5,3

which is encoded, using Elias code for integers, into `0100`, `01101`, `1`, `0101`, `01101`, `0101`.

5.3.3 Lempel-Ziv Codings

The idea of the very popular Lempel-Ziv codings is quite similar to the former Elias-Willems coding scheme: using ideas similar to the recency distance, it also aims at being *universal*, i.e. to work well for different kind of stationary source without precisely knowing all their statistical properties.

There exist many variations of the basic Lempel-Ziv algorithm, using dictionaries, post-

processing and many other improvements. Among the most famous variants we can cite:

Name	Authors	Method
LZ77	Lempel & Ziv (1977)	1 character and 1 pair of fixed-size pointers no dictionary
LZ78	Lempel & Ziv (1978)	same as LZ77 but with a dictionary (pointers in the dictionary)
LZSS	Storer & Szymanski (1982)	1 fixed-size pointer **or** 1 character (+ 1 indicator bit) no dictionary
LZW	Welch (1984)	only fixed-size pointers alphabet included in the dictionary

These algorithms are the most largely used compression algorithms in practice (e.g. in `zip`, `compress`, `gzip`, ...). The main reasons are because these algorithms perform good compression rate in a efficient manner. These algorithms are indeed linear time complex and do not require much memory.

In this section, we focus on the core of these compression algorithms by presenting the simplest of them: LZ77.

For this code, the codewords are tuples (i, j, u). i and j are integers and u is a source symbol.

The codeword (i, j, u) represents a sequence of symbols that can be obtained from the current sequence by

- copying j symbols starting from i position back
- and adding the symbol u at the end.

If i is null, j is ignored.

Example 5.7 (LZ77 codeword) For instance, if the current decoded sequence is `10010`, the codeword $(3, 2, 1)$ represents the sequence `011`: copying 2 symbols (`01`) starting from 3 positions back (`10|010`), and adding `1` at the end.

If j is greater than i, the copying of character continues with the newly copied characters (i.e. the buffer starting at i positions back is cyclic).

For instance, if the current decoded sequence is `10010`, the codeword $(3, 5, 1)$ represents the sequence `010011`: starting from three positions back (`10|010`) copying five symbols: first the thre symbols already existing at $i = 3$ positions back (`010`), leading to `10010|010`, and going on with the next two characters, `01` from the newly added character, leading to `1001001001`. The decoding finally ends adding the `1` (last element of the codeword) at the end, leading to `10010010011`.

In summary: `10010` + $(3, 5, 1)$ = `10010010011`.

Example 5.8 (LZ77 decoding) Here is an example of how the sequence
(0,0,0) (0,0,1) (2,1,0) (3,2,1) (5,3,1) (1,6,0)
is decoded:

codeword	cyclic buffer	added sequence	complete decoded sequence
(0,0,0)	–	0	0
(0,0,1)	0	1	01
(2,1,0)	01|0101...	00	0100
(3,2,1)	0|100|100100...	101	0100101
(5,3,1)	01|00101|00101...	0011	01001010011
(1,6,0)	0100101001|1|1111...	1111110	010010100111111110

The final result is thus 010010100111111110.

The corresponding coding algorithm, using a sliding window buffer for remembering the past context, is the following:

1. look in the current context (i.e. the beginning of the sequence still to be encoded) for the shortest sequence that is not already in the buffer;
2. remove the last character u of this unseen sequence and look for the closest corresponding sequence back in the buffer;
3. emit the corresponding back position (i) and length (j), followed by the removed last character (u)
4. update the buffer (with the newly encoded sequence) and go back in 1 as long as there is some input.

Here is an example of this encoding algorithm:

Example 5.9 (LZ77 encoding) Consider the message 1111111111101111011 to be encoded. At the beginning, since the buffer is empty, the first pointer pair must be (0,0) and the corresponding character is the first character of the sequence. Thus the first codeword is $(0, 0, 1)$.

Now the buffer is updated into 1. With the convention used for $j > i$, we are now able to encode sequences of any length made only of 1. Thus the sequence considered for encoding, i.e. the shortest sequence starting after the last encoded sequence and not already in the (cyclic) buffer is now: 1111111110.

This sequence is encoded into the codeword $(1, 9, 0)$: loop 1 step back in the buffer, copy 9 characters from the buffer (with repetitions since $9 > 1$, and add the last part of the codeword, here 0.

So the part of the message encoded so far (and the buffer) is 11111111110, and the part still to be encoded is 1111011.

Back to step 1: what is the shortest sequence to be encoded that is not contained in the buffer?

At this stage, it is 1111011. Removing the last char (1) we end up with 111101 which in the current buffer correspond to $i = 5$ and $j = 6$ (using once again the cyclic aspect of the buffer: $i < j$).

The corresponding codeword is thus $(5, 6, \texttt{1})$.

To summarize: `111111111101111011` is encoded into $(0, 0, \texttt{1})(1, 9, \texttt{0})(5, 6, \texttt{1})$.

Control Question 50

How is the sequence `100000101100` encoded using LZ77 algorithm?

Answer

The answer is (0,0,1) (0,0,0) (1,4,1) (2,2,1) (3,1,0)

Control Question 51

How is the sequence (0,0,0) (1,2,1) (2,5,0) decoded (assuming LZ77 encoding)?

Answer

`0001010100`

5.3.4 gzip and bzip2

When I will have time, I will here add a few words on gzip and maybe on bzip2, compression algorithms which are very popular in GNU world.

What I can now say in the only minute I have is that gzip is using a variation of the LZ77 algorithm combined with a post-processing of pointers using Huffman coding.

Interested readers could refer to `http://www.gzip.org/algorithm.txt` for more details.

Summary for Chapter 5

Shannon Noiseless Coding Theorem for prefix-free codes of stationary source: $\frac{h_\infty(U)}{\log D} \leq \frac{E[L_Z]}{k}$.

Recency Distance $R_n(v)$ is the number of symbols received at time n till the last reception of v (before n).

Elias-Willems Codes Elias coding of recency distance.

Lempel-Ziv Algorithm LZ77: using a (cyclic) buffer remembering past sequences, encode sequences with codewords consisting of one position back in the buffer, one length and one character to be added at the end of the sequence encoded so far.

Summary for Chapter 5

variable-to-fixed length coding: encoding segment of variable length of the input information source with codewords that have all the same size

proper set: a complete set of leaves of a coding tree (no useless leaf).

Tunstall set: a proper set such that every node of the corresponding coding tree is at least as probable as any of its leaves.

optimality of Tunstall sets: A proper set maximizes the average encoded message length if and only if it is a Tunstall set.

Tunstall algorithm: $k = \lfloor \frac{D_Z{}^n - 1}{D_U - 1} \rfloor$, $M = 1 + k\ (D_U - 1)$, extend k times the most probable node.

Elias Codes are prefix-free binary codes for integers, whose length is (asymptotically) close to optimal $\log_2(n)$.

These codes result from the concatenation of a prefix, made of the first Elias code of the length of the usual binary representation of the number to be encoded, and a suffix, made of the usual binary representation without its most significant bit.

Shannon Noiseless Coding Theorem for prefix-free codes of stationary source: $\frac{h_\infty(U)}{\log D} \leq \frac{E[L_Z]}{k}$.

Recency Distance $R_n(v)$ is the number of symbols received at time n till the last reception of v (before n).

Elias-Willems Codes Elias coding of recency distance.

Lempel-Ziv Algorithm LZ77: using a (cyclic) buffer remembering past sequence, encode sequences with codewords consisting of one position back in the buffer, one length and one character to be added at the end of the sequence so far.

Historical Notes and Bibliography

In spite of its fundamental importance, Tunstall's work was never published in the open literature. Tunstall's doctoral thesis (A. Tunstall, "Synthesis of Noiseless Compression Codes", Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, 1968), which contains this work, lay unnoticed for many years before it became familiar to information theorists.

To be continued...

OutLook

Chapter 6

Module I2: Error Correcting Codes

by J.-C. Chappelier

Learning Objectives for Chapter 6

After studying this chapter, you should know more about error-correcting codes, and more precisely:

1. what is minimum distance of a code and how many errors a code of a given minimum distance can correct;
2. the basics of linear codes: how to construct such codes, how to encode and decode with linear code, ...;
3. what "Hamming Codes" are and how to use them;
4. the basics of cyclic codes;
5. and the basics of convolutional codes: encoder circuit, lattice representation, Viterbi algorithm for decoding.

Introduction

The fundamental Shannon's Noisy Coding Theorem presented in chapter 4 provides theoretical bounds on the coding of messages for transmission over a noisy channel. Unfortunately, this important theorem (nor its proof) does not give any hint on how to actually build "good" error correcting codes in practice.

This is the reason why a theory of error correcting codes has been developing for many years. This theory focuses mainly on the algebraic structure of codes. The basic idea is to give structure to the set of codewords in order to use this structure to provide hints for decoding messages in case of transmission error.

Due to its strong mathematical grounding, "algebraic coding theory" is now well established as a full scientific field on its own, with applications to many problems going beyond channel coding.

The purpose of this chapter is certainly not to provide an exhaustive view of algebraic error correction codes (a whole book would hardly do it) but rather to introduce the key ideas of the domain. The reader interested in investing this topic further is referred to the rather vast literature of this field.

The study of this chapter requires a bit of mathematical background, especially in the field of algebraic structures.

6.1 The Basics of Error Correcting Codes

Learning Objectives for Section 6.1

In this section, the following key points about error correcting codes are presented:

1. what is a block-code;
2. how distance between codewords is measured;
3. what is the weight of a codewords;
4. what are minimum distance and minimum weight of a code;
5. and how useful they are for determining how many errors a code can detect and/or correct.

6.1.1 Introduction

The context of the present chapter is noisy transmission as presented in the introduction of chapter 4.

When a codeword z_i is transmitted via a noisy channel and $\widehat{z}$ is received, the transmission error corresponds to the difference between $\widehat{z}$ and z_i: $e = \widehat{z} - z_i$.

The key idea of algebraic coding is to add algebraic structure to the set of codewords such that the transmission error can easily be expressed in term of the operations defining this algebraic structure (starting from the above "difference" operation).

For instance, if we are dealing with binary codewords (e.g. `10011`) the "natural" difference on binary words is the bit by bit difference (a.k.a. "exclusive or" for readers familiar with computer sciences), i.e. the difference in each position such that there is a `0` whenever the two corresponding bits are the same and `1` when they are not: `0 - 1` = `1` and, as usual, `0 - 0` = `0`, `1 - 1` = `0`, `1 - 0` = `1`.

Example 6.1 (Binary Difference) Here is an example of the difference of two binary words:

$$\texttt{11011} - \texttt{01101} = \texttt{10110}$$

Technically speaking, the above defined operation actually corresponds to "modulo 2"

arithmetic, i.e. the Galois Field GF(2) is considered and codewords are elements of the vector space $\mathrm{GF}(2)^n$ (where n is the length of the codewords). This framework easily extends to any D-ary codes using "modulo D" arithmetic.

Definition 6.1 (Block-Code) A D-ary *block-code** of length n is an non-empty subset of the vector space of n-tuples $\mathrm{GF}(D)^n$ (i.e. D-ary words of the same length n, considered as "row vector"). ♦

Example 6.2 (Block-Codes) The set {1101, 0110, 1110} is an example of a binary block-code.

Another example of block-code, considering ternary codes using the symbols 0 1 and 2, such that 1+2 = 0 (i.e. GF(3) arithmetic), is given by the set {120, 201, 222, 010}.

The set {011, 110, 10} is not a block-code since these words have not all the same length.

Two important notions in the field of algebraic error codes are now introduced: the Hamming distance between words and the weight of a word. How this notions relates to the problem of error correction is shown in the following section.

6.1.2 Hamming Distance and Codeword Weight

Definition 6.2 (Hamming distance) The *Hamming distance**, $d(z_i, z_j)$, between two words of the same length z and z' (i.e. two n-tuples in the most general case) is the number of symbols (i.e. positions) in which z and z' differ. ♦

Example 6.3 (Hamming distance) The Hamming distance between 101010 and 111010 is 1 since these two words differ only in the second position.

The Hamming distance between 1010 and 0101 is 4 since these two words differ in all positions.

The Hamming distance between 1010 and 111010 in not defined (and will not be considered).

The Hamming distance is indeed a metric (i.e. following three axioms of metric distance) for n-tuples (over a non empty set!). The demonstration is left as an exercise.

Definition 6.3 (Codeword Weight) The *weight** of a word is the number of *non-zero symbols* in it. ♦

Example 6.4 (Codeword Weight) The weight of 10110 is 3, whereas the weight of 00000000 is 0 and the weight of 001000 is 1.

Property 6.1 *The Hamming distance between two codewords is the weight of their difference :*

$$d(z_i, z_j) = w(z_i - z_j)$$

denoting by $d(\cdot)$ the Hamming distance and $w(\cdot)$ the weight.

PROOF z_i and z_j differ in a position if and only if $z_i - z_j$ is non-zero in that position. ■

Example 6.5 Here is an example of the equivalence between the Hamming distance of two binary words and the weight of their difference:

$$\begin{aligned} d(\texttt{10110}, \texttt{11010}) &= 2 \\ w(\texttt{10110} - \texttt{11010}) &= w(\texttt{01100}) = 2 \end{aligned}$$

Here comes now some useful properties of the codeword weights.

Property 6.2 *The weight of a codeword is always positive or null.*

Trivial: by definition.

Definition 6.4 (null codeword) The *null codeword** is the codeword made only of zeros.
It will be denoted by **0**. ♦

Property 6.3 *The weight of a codeword is 0 if an only if the codeword is the null codeword* **0**.

Property 6.4 *The weight is symmetric: for every codeword z_i, $w(z_i) = w(-z_i)$ (where $-z_i$ is the word in which each symbol is the opposite of the corresponding symbol in z_i).*

Example 6.6 (Weight symmetry) Considering ternary codes using the symbols `0` `1` and `2`, such that $\texttt{1} + \texttt{2} = \texttt{0}$ (i.e. GF(3) arithmetic), we have:

$$w(-1202102) = w(2101201) = 5 = w(1202102)$$

Notice that in the binary case, the latest property is trivial since in that case $-z_i = z_i$ for every z_i.[1]

Property 6.5 *For every codewords z_i and z_j, we have:*

$$w(z_i) + w(z_j) \geq w(z_i + z_j)$$

[1] Recall that in the binary case, $-1 = 1$.

Example 6.7 In the binary case, we have for instance:

$$w(110101) + w(010101) = 4 + 3 = 7 \geq 1 = w(100000) = w(110101 + 010101)$$

Considering ternary codes using the symbols `0` `1` and `2` as above, we have for instance:

$w(01221021)+w(21002010) = 6+4 = 10 \geq 5 = w(22220001) = w(01221021+21002010)$

PROOF These properties directly comes from property 6.2 and the fact that the Hamming distance is indeed a metric. ■

Control Question 52

1. What is the weight of `11101101`?
2. What is the weight of `0`?
3. What is the weight of `1`?
4. What is the weight of `2`?
5. What is the weight of `1221032`?
6. What is the Hamming distance between `11` and `00`?
7. What is the Hamming distance between `101` and `001`?
8. What is the Hamming distance between `1234` and `3214`?

Answer

Weights: 6; 0; 1; 1; 6.

Hamming distances: 2; 1; 2.

Why are Hamming distance and weight of fundamental importance for algebraic error correction?

In the framework defined in the previous section, where the error e that occurred in the transmission where z_i was emitted and $\widehat{z}$ received is defined as $e = \widehat{z} - z_i$, the number of errors that occurred in transmission now appears to be simply the weight of e, i.e. $w(\widehat{z} - z_i)$. Due to property 6.1 this is the Hamming distance $d(\widehat{z}, z_i)$ between the emitted codeword and the one received.

In this framework, *detecting* an error then simply means detecting non-null weights. *Correcting* an error however, implies to be able to further compute the actual difference (without knowing z_i, of course! Only $\widehat{z}$ is known at the reception).

6.1.3 Minimum Distance Decoding and Maximum Likelihood

How could a block-code decoder take its decision to decode a received word? A natural intuitive answer is to assume the smallest number of errors (i.e. $w(e)$ minimum), which,

according to what has just been said, leads to take the closest codeword (i.e. $d(\widehat{z}, z_i)$ minimum).

For instance, if the only two possible codewords are `000` and `111` and `010` is received, we certainly would like[2] to have it decoded as `000`.

Definition 6.5 (Minimum Distance Decoding) A code $\mathcal{C}$ is said to use *minimum distance decoding** whenever the decoding decision $\mathcal{D}$ consists, for any received word $\widehat{z}$, in choosing (one of) the closest codeword(s):

$$\mathcal{D}(\widehat{z}) = \underset{z \in \mathcal{C}}{\text{Argmin}}\, d(z, \widehat{z})$$

How is minimum distance decoding mathematically sound?

In the case of the Binary Symmetric Channel (see example 4.1), this intuitively sensible minimum distance decoding procedure follows from "Maximum Likelihood Decoding".

Let us see the decoding procedure from a probabilistic (Bayesian) point of view. When a word $\widehat{z}$ is received, the most likely codeword to have been emitted (knowing that this word is received) is the codeword z which maximize the probability $P(X = z|Y = \widehat{z})$ (where X is the input of the channel and Y its output).

In practice this probability is not easy to cope with[3] if the distribution $P(X = z_i)$ of the codewords at the emission (so called "*a priori* probabilities") are not known. In this context, no further assumption is made but the less biased one that all the codewords are equally likely (maximum entropy assumption). Then it comes:

$$\begin{aligned}
\underset{z \in \mathcal{C}}{\text{Argmax}}\, P(X = z|Y = \widehat{z}) &= \underset{z \in \mathcal{C}}{\text{Argmax}}\, \frac{P(Y = \widehat{z}|X = z) \cdot P(X = z)}{P(Y = \widehat{z})} && (6.1)\\
&= \underset{z \in \mathcal{C}}{\text{Argmax}}\, P(Y = \widehat{z}|X = z) \cdot P(X = z) && (6.2)\\
&= \underset{z \in \mathcal{C}}{\text{Argmax}}\, P(Y = \widehat{z}|X = z) && (6.3)\\
& && (6.4)
\end{aligned}$$

The last equality is obtained using the equally likely codewords hypothesis.

The remaining term $P(Y = \widehat{z}|X = z)$ is the so-called likelihood, and it finally happen that the decoder should decode $\widehat{z}$ by the most likely codeword, i.e. the codeword z that maximizes $P(Y = \widehat{z}|X = z)$.

So what?

Well, the last term $P(Y = \widehat{z}|X = z)$ appears to be more easy to handle than the first $P(X = z|Y = \widehat{z})$. For instance in the case of a discrete memoryless channel (DMC, see 4.1) this turns into the product of the transmission probabilities for each symbol.

In the further case of a BSC, where all symbols have the same error probability p, this probability simply becomes:

$$P(X = z|Y = \widehat{z}) = p^{d(z,\widehat{z})}(1 - p)^{n - d(z,\widehat{z})}$$

[2] knowing nothing else

[3] or even to estimate, although this is possible.

since there are exactly $d(z, \widehat{z})$ symbols which have been corrupted by the transmission and $n - d(z, \widehat{z})$ which have been transmitted correctly.

It is then easy to see that the the codeword z that maximizes $P(Y = \widehat{z}|X = z) =$ is the one that minimizes the distance $d(z, \widehat{z})$.

This proves that for a BSC, minimum distance decoding and maximum likelihood decoding are equivalent.

6.1.4 Error Detection and Correction

Is there a way to know *a priori* how many errors a given code can correct? detect?[4] The answer is yes and relies mainly on an important characteristic of a block-code: its *minimum distance*.

Definition 6.6 (Minimum Distance of a Code) The *Minimum Distance** $d_{\min}(\mathcal{C})$ of a code $\mathcal{C} = \{z_1, ..., z_i, ..., z_M\}$ is the minimum (non null) Hamming distance between any two different of its codewords:

$$d_{\min}(\mathcal{C}) = \min_{i \neq j} d(z_i, z_j).$$

The following results about error correction and detection illustrates why the minimum distance of a code is of central importance in error correcting coding theory.

Theorem 6.1 (Error-Correcting and Detection Capacity) *A block-code of length n using minimum distance decoding can, for any two integers t and s such that $0 \leq t \leq n$ and $0 \leq s \leq n - t$, correct all patterns of t or fewer errors and detect all patterns of $t + 1, ..., t + s$ errors if and only if its minimum distance is strictly bigger than $2t + s$.*

PROOF The implication is demonstrated ad absurdio:
$\mathcal{C}$ cannot correct all patterns of t or cannot detect all patterns of $t + 1, ..., t + s$ errors if and only if $d_{\min}(\mathcal{C}) \leq 2t + s$

If the code $\mathcal{C}$ cannot correct all patterns of less than (including) t errors, this means there exists at least one codeword z_i and one error pattern e of weight less than t such that the decoding $\mathcal{D}(z_i + e)$ is not z_i. Let us call z_j the codeword decoded instead of z_i in this case: $z_j = \mathcal{D}(z_i + e)$.

Using triangle inequality for metric d, we have:

$$d(z_i, z_j) \leq d(z_i, z_i + e) + d(z_i + e, z_j)$$

But $d(z_i, z_i + e) = w(e) \leq t$ and $d(z_i + e, z_j) \leq d(z_i + e, z_i)$ since the code is using minimum distance decoding. Thus

$$d(z_i, z_j) \leq t + t \leq 2t + s$$

[4] By "detecting an error" we actually mean "detecting but not correcting an error", since of course correcting an error implies having detected it!

and therefore $d_{\min}(\mathcal{C})$, which is less than or equal to $d(z_i, z_j)$, is also less than or equal to $2t + s$.

If on the other hand the code can correct all patterns of less than (including) t errors but cannot detect all patterns of $t+1, ..., t+s$, there exists at least one codeword z_i and one error pattern e of weight between $t+1$ and $t+s$ which is not detected but decoded into another codeword z_j: $\mathcal{D}(z_i + e) = z_j$. Introducing the error $e' = z_j - (z_i + e)$, we also have $\mathcal{D}(z_j + e') = z_j$, i.e. e' is an error that is corrected when applied to z_j. Since $w(e') = d(z_j + e', z_j) = d(z_i + e, z_j) \leq d(z_i + e, z_i)$ (because of minimum distance decoding), we have both $w(e') \leq t + s$ and $\mathcal{D}(z_j + e') = z_j$. Thus $w(e')$ must be less than (or equal to) t. This allows us to conclude similarly to above:

$$d(z_i, z_j) \leq d(z_i, z_i + e) + d(z_i + e, z_j) \leq (t + s) + t = 2t + s$$

and therefore $d_{\min}(\mathcal{C}) \leq 2t + s$.

Thus if $\mathcal{C}$ cannot correct all patterns of t or cannot detect all patterns of $t+1, ..., t+s$ errors then $d_{\min}(\mathcal{C}) \leq 2t + s$.

Conversely, if $d_{\min}(\mathcal{C}) \leq 2t + s$, there exists two distinct codewords, z_i and z_j, such that $d(z_i, z_j) \leq 2t + s$. This means that the weight of the vector $z = z_i - z_j$ is also less than $2t + s$.

But any vector z of weight less than (or equal to) $2t + s$ can be written as the sum of two vectors e and f such that $w(e) \leq t$ and $w(f) \leq t + s$: take the first components up to t non-zero as the components of e [or all the components if $w(z) < t$] and the remaining components (zero elsewhere) for f. For instance `011010`, can be written as `010000` + `001010` ($t = 1$ and $s = 1$).

Thus, there exists two errors e and e' (take $e' = -f$) such that $w(e) \leq t$, $w(e') \leq t+s$ and $z_i - z_j = e - e'$; i.e. $z_i + e' = z_j + e$.

This means that two distinct codewords and two error patterns will be decoded the same way (since $z_i + e' = z_j + e$, $\mathcal{D}(z_i + e') = \mathcal{D}(z_j + e)$.

This implies that (at least) either $z_i + e'$ is not corrected ($\mathcal{D}(z_i + e') \neq z_i$) or $z_j + e$ is not detected ($\mathcal{D}(z_j + e) = z_i$).

Thus not all patterns of less than (including) t error can be corrected or not all patterns of $t + 1$, ... $t + s$ errors can be detected. ■

Property 6.6 (Maximum Error-Detecting Capacity) *A block-code $\mathcal{C}$ using minimum distance decoding can be used to detect all error patterns of $d_{min}(\mathcal{C}) - 1$ or fewer errors.*

PROOF Use $t = 0$ and $s = d_{\min}(\mathcal{C}) - 1$ in the above theorem. ■

Property 6.7 (Maximum Error-Correcting Capacity) *A block-code $\mathcal{C}$ using minimum distance decoding can be used to correct all error patterns of $(d_{min}(\mathcal{C}) - 1)/2$ (Euclidean, also called integer, division) or fewer errors, but cannot be used to correct all error patterns of $1 + (d_{min}(\mathcal{C}) - 1)/2$ errors.*

PROOF Use $t = (d_{\min}(\mathcal{C}) - 1)/2$ and $s = 0$ in the above theorem.

$(d_{\min}(\mathcal{C}) - 1)/2$ is furthermore the maximum t that can be used in the above theorem, since $d_{\min}(\mathcal{C}) \leq 2 \cdot (1 + (d_{\min}(\mathcal{C}) - 1)/2)$ [Recall that / denotes the Euclidean division]. ■

Example 6.8 A block-code having a minimum distance of 8 can be used to either

- correct all error patterns of less than (including) 3 errors and detect all patterns of 4 errors ($t = 3$, $s = 1$);
- correct all error patterns of less than 2 errors and detect all patterns of 3 to 5 errors ($t = 2$, $s = 3$);
- correct all error patterns of 1 error and detect all patterns of 2 to 6 errors ($t = 1$, $s = 5$);
- detect all patterns of less than (including) 7 errors ($t = 0$, $s = 7$).

A block-code having a minimum distance of 7 can either

- correct all error patterns of less than (including) 3 errors ($t = 3$, $s = 0$);
- correct all error patterns of less than 2 errors and detect all patterns of 3 to 4 errors ($t = 2$, $s = 2$);
- correct all error patterns of 1 error and detect all patterns of 2 to 5 errors ($t = 1$, $s = 4$);
- detect all patterns of less than (including) 6 errors ($t = 0$, $s = 6$).

Example 6.9 To be able to correct all patterns of 1 error (and that's it), a block-code must have a minimum distance at least equal to 3.

Control Question 53

1. A communication engineer want to have a channel where all patterns of 3 or less errors are corrected. Can he use a block code with a minimum distance of 5?
2. How many errors can be corrected at most with a block code with a minimum distance of 6?
3. Can such a code furthermore detect errors? If yes, how many?

Answer

1. A block code with minimum distance of 5 can at most correct all patterns of $(5 - 1)/2 = 2$ or less errors.

   The answer is thus: no.

2. A block code with minimum distance of 6 can at most correct all patterns of $(6-1)/2 = 2$ or less errors.

3. In such a case $2t = 2 \cdot 2 = 4$, thus the code can furthermore detect all patterns of 3 errors ($s = 1$). Indeed: $6 > 2 \cdot 2 + 1$ ($d_{\min}(\mathcal{C}) > 2t + s$).

Summary for Chapter 6

block-code: a non-empty set of words of the same length, considered as "row vectors".

weight: (of a word) the number of non-zero symbols.

Hamming distance: the number of coordinates in which two vectors differ.

The Hamming distance between two words is the weight of their difference.

minimum distance decoding: error correction framework in which each received word is decoded into the closest (according to Hamming distance) codeword.

maximum likelihood decoding: error correction framework in which each received word $\widehat{z}$ is decoded into (one of) the most likely codeword(s) z, i.e. a codeword such that $P(Y = \widehat{z}|X = z)$ is maximal (with X the input of the noisy channel and Y its output).

minimum distance of a code: the minimum (non null) Hamming distance between any two (different) codewords.

error correcting and detecting capacity: A block-code $\mathcal{C}$ of length n using minimum distance decoding can, for any two integers t and s such that $0 \leq t \leq n$ and $0 \leq s \leq n-t$, correct all patterns of t or fewer errors and detect all patterns of $t+1, ..., t+s$ errors if and only if its minimum distance $d_{\min}(\mathcal{C})$ is strictly bigger than $2t + s$:

$$d_{\min}(\mathcal{C}) > 2t + s \iff \mathcal{C} \text{ corrects } t \text{ and detects } t + s \text{ errors.}$$

6.2 Linear Codes

Because of their properties and their simplicity, linear codes, which are studied in this section, are of major interest in error coding. One of their main advantage in practice is that they are easy to implement.

Learning Objectives for Section 6.2

In this section, the basics of linear codes are presented. What you should know about this topic is:

1. what is a (n, m) D-ary linear code is;
2. that for such codes minimum distance and minimum weight are equivalent;
3. how to encode using a generator matrix;
4. what the systematic form generator matrix of a linear code;
5. how to decode using verification matrix and syndrome table;
6. how to compute the minimum distance of a linear code;
7. what binary Hamming codes are and how to use them.

6.2.1 Definitions

Linear codes are block-codes on which an algebraic structure is added to help decoding: the vector space structure.

Definition 6.7 (Linear Code) An (n, m) D-ary *linear code** $(1 \leq m \leq n)$ is a m-dimensional subspace of the vector space $\mathrm{GF}(D)^n$ of n-tuples over $\mathrm{GF}(D)$.

♦

Looking at the definition of block-codes, the above definition could be rephrased: a linear code is a block-code which is a vector space.

Example 6.10 (Linear Code) The code $\{\mathtt{1101}, \mathtt{0110}, \mathtt{1110}\}$ given in example 6.2 is not a linear code since $\mathtt{0000}$ is not part of it (it could therefore not be a vector space).

The (binary) code $\{\mathtt{0000}, \mathtt{1101}, \mathtt{0110}, \mathtt{1011}\}$ is a linear code since any (binary) linear combination of codewords is also a codeword. It is furthermore a $(4, 2)$ binary linear code since its dimension (i.e. the dimension of the vector subspace) is 2 and its length is 4.

Notice further that the minimum distance of this code is 2 (easy to check) and therefore this code can only be used for single error detection (refer to theorem 6.1).

Control Question 54

For each of the following binary codes, say whether or not this is a linear code. If yes, give the two numbers n and m of the definition:

1. $\mathcal{C} = \{0000, 1000, 0001, 1001\}$
2. $\mathcal{C} = \{1000, 0001, 1001\}$

3. $\mathcal{C} = \{00, 01, 11\}$
4. $\mathcal{C} = \{0000, 1000, 1100, 0100, 1101, 0001, 0101, 1001\}$
5. $\mathcal{C} = \{0, 1, 10, 11\}$
6. $\mathcal{C} = \{00, 11, 10, 01\}$
7. $\mathcal{C} = \{0000, 0001, 0010, 0100\}$

Answer

1. yes: any linear combination of codewords is also a codeword (it is enough to check that for instance the last codeword is the combination of the first two non null).
   It's a $(n = 4, m = 2)$ linear codeword.
2. no: this code does not have the null codeword.
3. no (e.g. $01 + 11 = 10$ is not a codeword)
4. yes (the thirds column is always 0, the other 3 generates all 3 bits words).
   It's a $(n = 4, m = 3)$ code.
5. No! This is even not a block code since the codewords do not all have the same length.
6. yes. $n = 2$ and $m = 2$
7. no (e.g. $0001 + 0010 = 0011$ is not a codeword).

6.2.2 Some Properties of Linear Codes

Property 6.8 *Every linear code contains the null codeword* $\mathbf{0}$.

This comes directly from the fact that a linear code is a vector (sub)space.

Let us now study further these linear code. First of all, how many codewords contains a (n, m) D-ary linear code?

Property 6.9 *A* (n, m) *D-ary linear code contains* D^m *different codewords (including the null codeword).*

Notice that this property can be used to quickly determine that codes with a wrong number of codewords (not a power of D) are not linear. For linear codes, this can also be used to quickly determine m (e.g. to "guess" the size of a basis).

PROOF Since a linear code is a vector space of dimension m, every codeword is the linear combination of m basis codewords (one basis for this vector space).

For a (n, m) D-ary code, there are therefore exactly D^m different codewords: all the D^m linear combinations. ■

What is then the transmission rate of such a code?

Property 6.10 *The transmission rate of a (n, m) linear code is*

$$R = \frac{m}{n}$$

PROOF Recall that the transmission rate of a D-ary code encoding a source of M different messages with codewords of length n is $R = \frac{\log_D M}{n}$.

How many different messages could be encoded using a (n, m) linear code? As many as there are codewords, i.e. D^m.

Thus,

$$R = \frac{\log_D D^m}{n} = \frac{m}{n}$$

Let us now come to another useful property of linear code.

Theorem 6.2 (Minimum Weight and Minimum Distance Equivalence) *For every linear code $\mathcal{C}$*

$$d_{min}(\mathcal{C}) = w_{min}(\mathcal{C})$$

where $w_{min}(\mathcal{C})$is the minimum weight of the code, i.e. the smallest weight of non-zero codewords:

$$w_{min}(\mathcal{C}) = \min_{\substack{z \in \mathcal{C} \\ z \neq \mathbf{0}}} w(z)$$

This result is very important in practice since $w_{\min}(\mathcal{C})$is much easier to compute than $d_{\min}(\mathcal{C})$.

PROOF For a code $\mathcal{C} = \{z_1, ..., z_i, ...\}$, we have by definition 6.6 $d_{\min}(\mathcal{C}) = \min_{i \neq j} d(z_i, z_j)$.

Thus, using property 6.1, $d_{\min}(\mathcal{C}) = \min_{i \neq j} w(z_i - z_j)$.

But if $\mathcal{C}$ is a linear code, for every two codewords z_i and z_j, $z_i - z_j$ is also a codeword. It is furthermore the null codeword if and only if $z_i = z_j$ (i.e. $i = j$).

Thus $d_{\min}(\mathcal{C}) \geq w_{\min}(\mathcal{C})$.

Conversely, for every codeword z_i, $w(z_i) = d(z_i, \mathbf{0})$. Since $\mathbf{0}$is always part of a linear code, we get: $w_{\min}(\mathcal{C}) \geq d_{\min}(\mathcal{C})$, which concludes the proof. ■

Example 6.11 The $(11, 3)$ binary code {00000000000, 10011110000, 01000111100, 00111001111, 11011001100, 10100111111, 01111110011, 11100000011} has a minimum weight of 5 and thus a minimum distance of 5.

This code can therefore correct all error patterns with 1 or 2 errors (cf Property 6.7).

[It is left as an exercise to check that this code is indeed a linear code.]

Control Question 55

What is the minimum distance of the following codes?

1. $\mathcal{C} = \{0000, 1000, 0001, 1001\}$
2. $\mathcal{C} = \{0000, 1000, 1100, 0100, 1101, 0001, 0101, 1001\}$
3. $\mathcal{C} = \{00000000, 00001011, 00110000, 00111011, 11000000, 11001011, 11110000, 11111011\}$
4. $\mathcal{C} = \{1000, 0001, 0010, 0100\}$

Answer

1. $d_{\min}(\mathcal{C}) = w_{\min}(\mathcal{C}) = 1$
2. $d_{\min}(\mathcal{C}) = w_{\min}(\mathcal{C}) = 1$
3. $d_{\min}(\mathcal{C}) = w_{\min}(\mathcal{C}) = 2$ (e.g. third codeword)
4. $d_{\min}(\mathcal{C}) = 2$!! Although $w_{\min}(\mathcal{C}) = 1$! This is a pitfall! This code not a linear code. Thus the minimum weight theorem cannot be applied. Her the minimum distance is simply computed using its definition. It is easy to see that the distance between any two codewords is 2.

Control Question 56

How many errors can (at most) correct the following linear codes:

1.
2. $\mathcal{C} = \{0000, 1000, 1100, 0100, 1101, 0001, 0101, 1001\}$
3. $\mathcal{C} = \{000000000, 000000111, 000111000, 000111111, 111000000, 111000111, 111111000, 111111111$

Answer

1. $d_{\min}(\mathcal{C}) = w_{\min}(\mathcal{C}) = 1$ thus this code can correct $(1-1)/2 = 0$ errors!! This is not a very useful code!
2. $d_{\min}(\mathcal{C}) = w_{\min}(\mathcal{C}) = 3$ thus this code can correct $(2-1)/2 = 1$ error.

6.2.3 Encoding with Linear Codes

The relationship linking messages to be encoded to corresponding codewords must now be emphasized further: it is time to see how to efficiently use linear codes to encode messages.

If the m codewords chosen to be a basis of a (n, m) linear code (vector space)[5] are denoted by z_1, ..., z_m, then any codeword z_i can be written as

$$z_i = \sum_{k=1}^{m} u_{i,k}\ z_k$$

where $u_{i,k}$ is the component of z_i on the basis vector z_k.

In a more compact way, using linear algebra, we have:

$$z_i = (u_{i,1}, ..., u_{i,m}) \cdot G = u_i \cdot G$$

where u_i is the row vector $(u_{i,1}, ..., u_{i,m})$ and G the matrix whose rows are $z_1, ..., z_m$.

It is then very natural to choose to encode the m symbol message u_i by the codeword z_i which results from the above multiplication by G.

For this reason, the matrix G is called a *generator matrix** of the code.[6]

Definition 6.8 (Generator Matrix) A $m \times n$ matrix G is said to be a generator matrix of a (n, m) linear code $\mathcal{C}$ if and only if its m row vectors are a basis of the vector space $\mathcal{C}$.
The encoding of a message u (of size m) is then done by $z = u \cdot G$. ♦

Example 6.12 (Generator Matrix) Let us go on with the $(4, 2)$ binary linear code used in example 6.10: $\{\texttt{0000}, \texttt{1101}, \texttt{0110}, \texttt{1011}\}$.

This code, having four codewords, can encode four messages: the four binary words of two bits, u_0 =00, u_1 =10, u_2 =01, u_3 =11. Let us do this encoding using a generator matrix.

One basis for this linear code could be z_1 = 1101, z_2 = 0110, which leads to

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

u_1 is then encoded into

$$u_1 \cdot G = (10) \cdot \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = (1101) = z_1$$

and similarly u_2 into z_2, u_3 into 1011 and u_0 into 0000.

Notice that a linear code always encode the null message with the null codeword **0**. This is precisely due to the linear (i.e. vector space) aspect of the code.

[5] Recall that in most cases this choice is not unique

[6] Notice that, for a given code, this generator matrix is not unique: it depends on the basis chosen for representing the code.

Using this way of encoding with generator matrix the actual encoding of messages is very easy to implement in practice. In the binary case for instance, only a few[7] exclusive-or (XOR) gates can do the encoding.

6.2.4 Systematic Form of a Linear Codes

Among all possible generator matrices, one is of special interest (if it exists): the systematic form.

Definition 6.9 (Systematic Form) A generator matrix G of a (n, m) linear code is said to be in systematic form when it is written as

$$G = [I_m \; P] = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{1,1} & \cdots & p_{1,n-m} \\ 0 & 1 & \cdots & 0 & p_{2,1} & \cdots & p_{2,n-m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{m,1} & \cdots & p_{m,n-m} \end{bmatrix}$$

where I_m is the identity matrix of size m and P a $m \times (n-m)$ matrix, often call "*Parity Matrix*". ♦

Notice that, when it exists for a given code, the systematic generator matrix is unique.

Definition 6.10 (Systematic Linear Code) A linear code that uses a generator matrix in systematic form is called a systematic (linear) code. ♦

When a (n, m) linear code uses systematic form generator matrix, the m first symbols of the n symbols of a codeword are exactly the symbols of the encoded message:

$$z_i = (u_{i,1}\, u_{i,2} \ldots u_{i,m}\, z_{i,m+1}, \ldots z_{i,n})$$

In other words, systematic codes send first the message unencoded and then $(n-m)$ encoding symbols used for error detection/correction.

Example 6.13 Recalling example 6.12, another choice for the basis vectors could have been $z_1 = \texttt{1011}$, $z_2 = \texttt{0110}$, leading to

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

which is the systematic form generator matrix for this code.

Example 6.14 (Parity-Check Bit) For binary messages, the *Parity-Check Bit* is the bit that corresponds to the parity of the message, i.e. the (binary) sum of its bits.

For instance the parity-check bit for `01101` is $\texttt{1} + \texttt{1} + \texttt{1} = \texttt{1}$ and the parity-check bit for `00101` is $\texttt{1} + \texttt{1} = \texttt{0}$.

[7] at most $n \cdot (m-1)$ actually

Parity-Check Bit encoding consists simply in sending first the message as it is, follow by its parity-check bit. In terms of codes, this corresponds to the $(m+1, m)$ binary linear code, the generator matrix of which is

$$G = \begin{bmatrix} & 1 \\ I_m & \vdots \\ & 1 \end{bmatrix}$$

which is in systematic form.

Notice that the minimum distance of this code is 2 (using Theorem 6.2), thus this code is only able to do single error detection (refer to Theorem 6.1).

Control Question 57

For the following matrices

1. say if it could be a generator matrix.
2. if yes, say if it is in systematic form.
3. if the matrix is not in systematic form, give the systematic form matrix for the corresponding code.
4. (when it is a generator matrix) how will the message `1011` be encoded using the systematic form?

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

2. $G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

3. $G = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$

Answer

1. This is not a generator matrix since the four rows are linearly dependent (their sum is zero) they cannot be a basis of a vector space.
2. Yes this matrix is a generator matrix (for a $(7, 4)$ binary linear code). It is furthermore the generator matrix of this code since the four first columns make the identity matrix.

   The encoding of `1011` with this code is `1011001`.

3. Yes, this matrix is indeed a generator matrix. The four rows are four linearly independent vector: to have a zero in the next to last column you must have a zero coefficient on the last line, then to have a 0 at the last column you need to annihilate the second row and then it's easy to see that the first and third row are linearly independent (using for instance the second column).

   This matrix is not in systematic form.

   To find the systematic form generator matrix, we need to find a basis of this linear code whose four first coordinates form the identity matrix. There are several mathematical way to do this. Here is one rather basic:

   adding the fours rows, we get $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$ which can make the first vector.

   Then adding this vector to the first row of the original matrix, we get $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$ which is the second vector. Similarly adding the first new vector to the last row of the original matrix leads to $\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$ which is the third vector we are looking for. And for the last one, we can add the second row of the original matrix $\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$ to the new second vector $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$ to get $\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ and the systematic form generator matrix is:

$$G' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The encoding of `1011` with this matrix is `1011000`.

6.2.5 Decoding: Verification Matrix

At this level, we know how to encode with linear codes. But what about decoding? How can errors be corrected? This is the whole story after all!

Here is precisely where the linearity of linear codes will help.

Suppose that a matrix F such that, for every codeword z, $z \cdot F = \mathbf{0}$ has been found[8]. Then, if an error e occurs during the transmission of z and $\widehat{z} = z + e$ is received, we have

$$\widehat{z} \cdot F = (z + e) \cdot F = z \cdot F + e \cdot F = \mathbf{0} + e \cdot F = e \cdot F$$

This last result is very useful since $\widehat{z} \cdot F$ is independent of the emitted codeword z but depends only on the error e. The result of this transmission error appears as a linear combination of the rows of F. In order to correct/detect the error, the vectors of the vector space generated by the rows of F "simply" needs to be mapped to the corresponding correction (or detection message).

This is this key idea that is formalized and studied an bit further now.

For good mathematical reasons[9], the above equation $z \cdot F = \mathbf{0}$ is always given in the following form:

$$z \cdot H^T = \mathbf{0}$$

[8] As you will see in a while, this is not so difficult.

[9] orthogonality: $G \cdot H^T = 0$

where ^{T} is the transpose operator and $H = F^T$.

Definition 6.11 (Verification Matrix) A $(n-m) \times n$ matrix H is a *verification matrix** for a (n, m) D-ary linear code $\mathcal{C}$ if and only if

$$\forall z \in \mathrm{GF}(D)^n \; z \cdot H^T = \mathbf{0} \iff z \in \mathcal{C}$$

In other words, a verification matrix for a code $\mathcal{C}$ is a matrix, the kernel of which is $\mathcal{C}$. ♦

Notice that a given linear code might have several different verification matrices: any matrix, the rows of which are a basis of the vector space orthogonal to the linear code[10] is a verification matrix for this code.

How to find verification matrices?

It the case where the code is systematic, it is easy to find a verification matrix as the following theorem show:

Theorem 6.3 *For a systematic (n, m) linear code, the systematic form generator matrix of which is*

$$G = [I_m \; P],$$

the matrix

$$H = \left[-P^T \; I_{n-m}\right]$$

is a verification matrix.

PROOF For every message u_i, the corresponding codeword is

$$\begin{aligned} z_i &= u_i \cdot G \\ &= u_i \cdot [I_m \; P] \end{aligned}$$

i.e.

$$\begin{cases} (z_{i,1}, ..., z_{i,m}) &= u_i \\ (z_{i,m+1}, ..., z_{i,n}) &= u_i \cdot P \end{cases}$$

Thus

$$(z_{i,m+1}, ..., z_{i,n}) = (z_{i,1}, ..., z_{i,m}) \cdot P$$

i.e.

$$-(z_{i,1}, ..., z_{i,m}) \cdot P + (z_{i,m+1}, ..., z_{i,n}) = \mathbf{0}$$

or in its matrix form:

$$z_i \cdot \begin{bmatrix} -P \\ I_{n-m} \end{bmatrix} = \mathbf{0}$$

Therefore, we have found a matrix ($\left[-P^T \; I_{n-m}\right]^T$) such that its product with every

[10] recall: a linear code is a vector subspace

codeword gives the null vector.

It is easy to see that the inverse construction leads to the result that every word x such that $x \cdot \left[-P^T \; I_{n-m}\right]^T = \mathbf{0}$ verifies

$$x = (x_1, ..., x_m) \cdot G$$

and appears therefore as a codeword. ■

Notice that in the binary case (GF(2)) $-P = P$.

Example 6.15 Consider the systematic code $\mathcal{C}$, the generator matrix of which is

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} I_2 & 1 & 0 & 1 \\ & 1 & 1 & 1 \end{bmatrix}$$

Then ($n = 5$ and $m = 2$)

$$H = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}^T & I_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

is one possible verification matrix for $\mathcal{C}$.

It has just been shown how easy it is to find the verification matrix when the systematic form generator matrix is know. What about the general case, when a generator matrix not is systematic form is used?

A verification matrix H for a (n, m) D-ary linear code with generator matrix G, the rows of which are denoted by $z_1, ..., z_m$, can be constructed using the following procedure[11]:

1. For $i = m + 1$ to n, choose z_i as any $\mathrm{GF}(D)^n$ vector linearly independent of $z_1, ..., z_{i-1}$
2. Compute the inverse M^{-1} of the matrix M the rows of which are $z_1, ..., z_n$.
3. Extract H^T as the last $n - m$ columns of the matrix M^{-1}.

Example 6.16 Let us come back to example 6.12 and the $(4, 2)$ code, one generator of which was given as:

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Let's first find two vectors linearly independent of the former. Choose for instance `1000` and `0100`.

Thus $M = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ and $M^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$

[11] This is a standard procedure in linear algebra used to construct orthogonal subspaces.

Finally, we have $H^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$, i.e. $H = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$.

Control Question 58

Give one verification matrix for the linear code the systematic form encoding matrix is

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Answer

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Control Question 59

Is the word $z =$`1001101` a codeword of the code, one verification matrix of which is

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Answer

yes: $z \cdot H^T = 0$ thus z is a codeword.

6.2.6 Dual Codes

A verification matrix for a (n, m) D-ary linear code $\mathcal{C}$ is a $(n - m) \times n$ matrix H such that its kernel $\ker(H)$ is $\mathcal{C}$: $\ker(H) = \mathcal{C}$.

Furthermore, by the fundamental property of dimensions of vector space, $\dim(\ker(H))+\text{rank}(H) = n$, i.e. $\dim(\mathcal{C}) + \text{rank}(H) = n$ or $\text{rank}(H) = n - m$.

So the $n-m$ rows of H generate a $n-m$ dimension subspace of $G(D)^n$, i.e. a $(n, n-m)$ linear code. Thus H appears to be the generator matrix of a $(n, n - m)$ linear code. This code is called the *dual code* of $\mathcal{C}$ (and vice versa).

Property 6.11 *The dual code of a dual code of a code $\mathcal{C}$ is the code $\mathcal{C}$ itself.*

Property 6.12 *A generation matrix of a code is a verification matrix for its dual code, and conversely.*

6.2.7 Syndromes

Let us now repeat the important key idea of linear code.

If z is the transmitted codeword and an error e occurs, the received word is then $\widehat{z} = z + e$. If H is a verification matrix for the code used, then

$$\widehat{z} \cdot H^T = (z + e) \cdot H^T = z \cdot H^T + e \cdot H^T = \mathbf{0} + e \cdot H^T = e \cdot H^T$$

This illustrates the important fact that $\widehat{z} \cdot H^T$ depends only on the actual error pattern e and not at all on the transmitted codeword $\widehat{z}$. For this reason, this result $\widehat{z} \cdot H^T$ is of peculiar importance for decoding. This is call the *syndrome* (of $\widehat{z}$ relative to H).

Definition 6.12 (Syndrome) The *syndrome** of a word $\widehat{z}$ relative to a verification matrix H is the product $\widehat{z} \cdot H^T$. ♦

Property 6.13 *The syndrome $s = \widehat{z} \cdot H^T$ of a received word $\widehat{z}$ relative to the verification matrix H of a code $\mathcal{C}$ depends only on the transmission error $e = \widehat{z} - z_i$ and not on the transmitted codeword z_i ($z_i \in \mathcal{C}$).*

Furthermore, the error pattern e is decomposed into elementary errors e_k (i.e. made of only one error on one single symbol): $e = (e_1, ..., e_n)$, then

$$s(\widehat{z}) = \widehat{z} \cdot H^T = e \cdot H^T = \sum_k e_k \, h_k$$

where h_k is the k-th column of H: $H = [h_1, ..., h_n]$.

To find the corrector (i.e. the opposite of the error), only the correctors corresponding to single errors need to be known and then sum up.

Correcting can then simply done by mapping the columns of H to corrector (stored in a memory) and adding the one corresponding to the non-zero positions of the syndrome.

Example 6.17 (Syndrome-based Correction Table) Suppose that

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

is a verification matrix for the binary code used.

Then the following correctors can be derived from the columns of H:

Syndrome	Corrector
101	10000
111	01000
100	00100
010	00010
001	00001

This is simply obtained by listing the columns of H.

Reordering by syndromes the above table in order to actually use it in practice (where only the syndrome is know), we get:

Syndrome	Corrector
000	00000
001	00001
010	00010
011	?
100	00100
101	10000
110	?
111	01000

Notice that:

1. The null syndrome always maps to no correction, due to Definition 6.11;
2. For 011 and 110, the corrector is not unique in this example: for instance 011 = 010 + 001 leads to 00011 (00001 + 00010), but 011 = 111 + 100 leads to another correction, 01100.

   This is due to the fact the minimal distance of this code is 3 (see next section), and thus this code can only correct all the patterns with 1 error, but not all the patterns with 2 errors!

   These two syndromes actually corresponds to two transmission errors.

In practice the correspondence table between syndromes and errors is stored in a memory and the general mechanism for decoding (and correcting) a received message $\widehat{z}$ is the following:

1. Compute the syndrome $s(\widehat{z}) = \widehat{z} \cdot H^T$;
2. Get the correction $c = -e$ (i.e. the opposite of the error) from the linear combination of correctors stored in the memory;
3. Decode by $z = \widehat{z} + c$.

Example 6.18 (Decoding with a Linear Code) Let us go on with the last example (Example 6.17), the generator of which is

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Suppose that $u = \texttt{10}$ is to be transmitted. This message is encoded into $z = \texttt{10101}$ with the above $(5, 2)$ binary linear code.

Suppose that $\widehat{z} = \texttt{00101}$, i.e. that the first bit has been corrupted.

The syndrome computation gives $s = z \cdot H^T = \texttt{101}$ leading to the corrector $e = 10000$ (see correctors table given in Example 6.17).

Thus the decoded codeword is $\widehat{z} + e = \texttt{00101} + \texttt{10000} = \texttt{10101}$, which leads to the decoded message (the first two bits, since a systematic code is being used): 10, which corresponds to the original message.

Control Question 60

What was the original message if you receive `101011001` and the code verification matrix is

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} ?$$

Answer

The original message was `10001`.

Explanation:

$\widehat{z} \cdot H^T = \texttt{0101}$ which correspond to the third column of H, thus one error occurred in the third position.

Thus the emitted codeword was `100011001` (change the third bit), and since the code is systematic (see H), the original message was `10001` (take the $m = 9 - 4 = 5$ first bits).

6.2.8 Minimum Distance and Verification Matrix

The general presentation of linear code is now ended by a useful result which allows to compute the minimum distance of a code[12] directly from its verification matrix.

Theorem 6.4 (Verification Matrix and Minimum Distance) *If H is a verification matrix for an (n, m) D-ary linear code $\mathcal{C}$ (with $1 \leq m < n$), then the minimum distance $d_{min}(\mathcal{C})$ of this code is equal to the smallest number of linearly dependent columns of H.*

PROOF For every vector z, $z \cdot H^T$ is a linear combination of $w(z)$ columns of H. And by Definition 6.11: $z \in \mathcal{C}$ if and only if $z \cdot H^T = 0$.

Thus if $z \in \mathcal{C}$ then there exists $w(z)$ columns of H are linearly dependent; and conversely, if q columns of H are linearly dependent, there exists a codeword of

[12] and thus the maximum number of errors to be corrected

weight q.

Thus $w_{\min}(\mathcal{C})$ is the minimum number of columns of H that are linearly dependent, and we conclude using Theorem 6.2. ■

For a binary linear code $\mathcal{C}$ with verification matrix H, this last results implies that

- If H does not have a null column, $d_{\min}(\mathcal{C}) > 1$.
- If H does further not have twice the same column, $d_{\min}(\mathcal{C}) > 2$.

Example 6.19 A binary linear code one verification matrix of which is $H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ has a minimum distance of 3.

Indeed, H does not have a null column nor twice the same column so $d_{\min}(\mathcal{C}) > 2$. Furthermore, there is a set of there columns of H that is linearly dependent. For instance, h_1, h_3 and h_5.

Property 6.14 (Singleton's bound) *For a (n, m) linear code $\mathcal{C}$,*

$$d_{min}(\mathcal{C}) \leq n - m + 1.$$

PROOF Columns of H are vectors in $\mathrm{GF}(D)^{(n-m)}$, so any set of $n - m + 1$ of these column is linearly dependent. Therefore, using Theorem 6.4, $d_{\min}(\mathcal{C}) \leq n - m + 1$. ■

Control Question 61

What is the minimum distance of a code, the verification matrix of which is

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} ?$$

How many errors can this code correct?

Answer

$d_{\min}(\mathcal{C}) = 3$, and thus $t = 1$.

e-pendix: Linear Code

6.2.9 Binary Hamming Codes

Let us now study further what are the "good" codes that can correct one error (in the binary case).

Since we are looking for only one error, it is sufficient for the syndrome to indicate where this error holds. The idea is to have a code such that the syndrome directly

indicates the position of the error; for instance as its binary code (0...001 for the first position, 0...010 for the second, 0...011 for the third, etc.).

Recall that a single error in position k leads to a syndrome which is the k-th column of H. The above idea thus leads for the verification matrix to construct a matrix, the columns of which are the binary representation of their position (see example below).

What is not clear yet is

1. what dimensions should this verification matrix have?
2. does this construction actually lead to a verification matrix of a code?
3. can such a code actually correct all patterns of 1 error?

Regarding the first point, recall that the size of a syndrome of a (n, m) linear code is $n - m$. If the syndrome is directly encoding the error position, it could then represent $2^{n-m} - 1$ positions. So no place is lost if the total number of positions to be represented (i.e. the length n of the codeword) is $n = 2^{n-m} - 1$.

Despite the trivial case $n = 3, m = 1$, here are some possible sizes for such codes:

n	m	$r = n - m$
7	4	3
15	11	4
31	26	5
63	57	6
$\vdots$	$\vdots$	$\vdots$

and here are two examples of verification matrix (for $n = 7$ and 15):

$$H_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The second above question is easy to answer: yes this construction actually leads to a linear code since matrices constructed this way are full rank (i.e. $\text{rank}(H) = n - m$), since it is easy to construct the I_{n-m} identity matrix out of their columns (take the first, second, forth, eighth, etc.. columns). Thus the dimension of their kernel is m, leading to a (n, m) linear code.

Finally, to address the last question (can such a code actually correct all patterns of 1 error?), we have to compute its minimum distance. The verification matrices resulting from the above construction never have null column nor the same column twice so at least $d_{\min}(\mathcal{C}) \geq 3$. Moreover, the first three column (binary representations of 1, 2 and 3) are always linearly dependent. So the minimum distance of such codes is always 3. Thus such codes can correct all patterns of 1 error.

Such codes are called "(binary) *Hamming codes*".

Definition 6.13 (Hamming code) A Hamming code is a $(2^r-1, 2^r-r-1)$ binary linear code ($r \geq 2$), the verification matrix of which is

$$H_r = \left[b_r(1)^T\ b_r(2)^T\ \cdots\ b_r(n)^t\right] = \begin{bmatrix} 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

where $b_n(i)$ is the binary representation of i on n bits (e.g $b_4(5) = (0101)$). ♦

Property 6.15 *Every binary Hamming code can correct all patterns of one error.*

Example 6.20 (Hamming code) Let us take $r = 3$ and construct the $(7, 4)$ binary Hamming code.

We have:

$$H_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

To find one generator matrix, we look for 4 vectors z such that $z \cdot H^T = 0$, for instance (easy to check, and there are many others):

$$\begin{aligned} z_1 &= 1110000 \\ z_2 &= 1101001 \\ z_3 &= 1000011 \\ z_4 &= 1111111 \end{aligned}$$

leading to

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Suppose now the message $u = 1001$ has to be send. It is encoded into $z = u \cdot G = 0001111$. Let's assume further that an error occurred on the third bit, so that $\widehat{z} = 0011111$ is received. How will this be decoded?

The syndrome is $s(0011111) = \widehat{z} \cdot H^T = 011$, i.e. 3 in binary, indicating that an error occurred on the third bit. So the result of the decoding is $\widehat{z} - 0010000$ (error in the third position) which is 0001111, the codeword that was actually emitted.

Summary for Chapter 6

linear code: a block code which is a vector space (i.e. any linear combination of codewords is also a codeword).

A (n, m) D-ary linear code is a dimension m vector subspace of the dimension n vector space of D-ary words.

minimum distance: for linear code minimum distance of the code is equal to the minimal weight.

generator matrix: (of a (n, m) linear code) a $m \times n$ matrix the rows of which are a basis of the code (they are thus linearly independent).

systematic form: a $m \times n$ generator matrix of a (n, m) linear code is said to be in systematic form only if its left most $m \times m$ submatrix is the identity matrix (of size m).

encoding: the encoding with linear codes is done by matrix multiplication: the word to be encoded u is multiply by one chosen generator matrix G of the code producing the codeword $z = u \cdot G$.

If the generator matrix is in systematic form, the first m symbols of the codeword are exactly the symbols of the message. Thus only the $n - m$ last symbols actually need to be computed.

verification matrix: A $(n - m) \times n$ matrix H is a verification matrix for a (n, m) linear code $\mathcal{C}$ if and only if

$$\forall z \in \mathrm{GF}(D)^n \; z \cdot H^T = \mathbf{0} \iff z \in \mathcal{C}$$

The verification matrix is very useful for decoding.

syndrome: The result of the product of a word by the verification matrix: $s = z \cdot H^T$.

The syndrome is used to determine the error to be corrected. It indeed correspond to the linear combination of columns of H which precisely is the product of the error pattern by H^T.

binary Hamming codes:
- $(2^r - 1, 2^r - r - 1)$ linear codes that can correct all patterns of 1 error;
- the verification matrix is given in the form of the binary enumeration of the columns.

6.3 Cyclic Codes

Learning Objectives for Section 6.3

After studying this section you should know:

1. what a cyclic code actually is;
2. how to (and why!) represent codewords using polynomials;
3. how to encode and decode cyclic code using the generator polynomial.

6.3.1 Introduction

Although cyclic codes is the most largely used class of error correcting codes, only a very short introduction to this topic is here presented. Indeed, a detailed presentation of cyclic codes is far enough to be the matter of a whole course in itself, which is largely out of the scope of the present lectures. Interested readers who wish to study deeper the subject should refer to the rather large literature on the domain.

Definition 6.14 A *cyclic code** is a linear code such that for every codeword z_i with n symbols $z_i = z_{i,1}...z_{i,n}$, the word $z_{i,2}...z_{i,n}z_{i,1}$ resulting of a (left-)cyclic permutation (also called "shift") of the symbols of z_i is also a codeword.

♦

Notice that this definition implies that then any cyclic permutation of a codeword is also a codeword.

Example 6.21 (Cyclic Code) The following (binary) linear code is a cyclic code:

$$z_1 = \texttt{000},\ z_2 = \texttt{101},\ z_3 = \texttt{011},\ z_4 = \texttt{110}$$

Conversely, the following code

$$z_1 = \texttt{000},\ z_2 = \texttt{001},\ z_3 = \texttt{010},\ z_4 = \texttt{011}$$

(which is linear) is not cyclic since, for instance, the cyclic permutation `100` of z_3 is not a codeword.

Cyclic codes are an important subclass of linear codes since they have many algebraic properties that simplify the encoding and decoding implementations.

Control Question 62

For each of the following binary codes, say whether this is a cyclic code or not.

1. $\mathcal{C} = \{0000, 1000, 0001, 1001\}$
2. $\mathcal{C} = \{1000, 0001, 0100, 0010\}$
3. $\mathcal{C} = \{000, 100, 010, 001, 110, 011, 111, 101\}$
4. $\mathcal{C} = \{0000, 0001, 0010, 0011\}$
5. $\mathcal{C} = \{00, 11, 10, 01\}$

Answer

1. no.
2. no! This is not a linear code! A cyclic code must first of all be a linear code.
3. yes.

4. no.
5. yes.

6.3.2 Cyclic Codes and Polynomials

In order to algebraically take into account this new constraints on codewords (cyclic permutations of codewords are also codewords), more complete algebraic structure than vector space one (which cope with linearity) is required. The algebra of polynomials is precisely a good way to represent this new constraint.

Indeed, suppose a codeword z_i with n symbols $z_i = z_{i,1}...z_{i,n}$ is represented by the polynomial $z_i(X) = z_{i,1} \cdot X^{n-1} + z_{i,2} \cdot X^{n-2} + ... + z_{i,n-1} \cdot X + z_{i,n}$, i.e. the jth symbol $z_{i,j}$ of a codeword z_i of size n is the coefficient of X^{n-j} in the corresponding polynomial $z_i(X)$. What is then $X \cdot z_i(X)$ modulo $(X^n - 1)$? A bit a simple polynomial algebra directly shows that this indeed corresponds to the left-cyclic permutation of z_i.

PROOF Let us prove that monome multiplication corresponds to left-cyclic permutation.

$$\begin{aligned} X \cdot z_i(X) &= X \cdot \left(\sum_{j=1}^{n} z_{i,j} \cdot X^{n-j} \right) \\ &= \sum_{j=1}^{n} z_{i,j} \cdot X^{n-j+1} \\ &= \sum_{k=0}^{n-1} z_{i,k+1} \cdot X^{n-k} \\ &= z_{i,1} \cdot X^n + \sum_{k=1}^{n-1} z_{i,k+1} \cdot X^{n-k} \end{aligned}$$

Working "modulo $(X^n - 1)$" simply means that X^n corresponds to 1,[13] X^{n+1} corresponds to X, etc. Therefore $z_{i,1} \cdot X^n \mod (X^n - 1)$ equals $z_{i,1}$, and the above equation, modulo $(X^n - 1)$, leads to

$$\begin{aligned} X \cdot z_i(X) &= z_{i,1} + \sum_{k=1}^{n-1} z_{i,k+1} \cdot X^{n-k} \\ &= \sum_{k=1}^{n-1} z_{i,k+1} \cdot X^{n-k} + z_{i,1} \cdot X^{n-n} \end{aligned}$$

which indeed corresponds to the codeword $z_{i,2}...z_{i,n}z_{i,1}$, the left-shift of z_i. ■

Since cyclic codes precisely deal with cyclic permutation of their codewords, polynomials seem to be a very appropriate way to represent them. This aspect will be emphasized further after a short example.

Example 6.22 (Modulo $(X^n - 1)$ arithmetic) Here is a short example of a modulo $(X^n - 1)$ computation:

$$\begin{aligned}(X^2+1)\cdot(X+1) &= X^3+X^2+X+1\\ &= 1+X^2+X+1 \mod (X^3-1)\\ &= X^2+X+(1+1) \mod (X^3-1)\\ &= X^2+X \mod (X^3-1)\end{aligned}$$

since in binary $1+1=0$.

Example 6.23 (Polynomial Representation of Cyclic Code) Recall the binary cyclic code of the last example:

$$z_1 = 000 z_2 = 101 z_3 = 011 z_4 = 110$$

The polynomial representation of this code is:

$$\begin{aligned}z_1(X) &= 0\\ z_2(X) &= 1\cdot X^2+0\cdot X+1 = X^2+1\\ z_3(X) &= 0\cdot X^2+1\cdot X+1 = X+1\\ z_4(X) = 1\cdot X^2+1\cdot X+0 &= X^2+X\end{aligned}$$

Notice furthermore that $X \cdot z_2(X) = z_3(X) \mod (X^3-1)$, which express that z_3 is the left-shift of z_2.

Control Question 63

What is the polynomial representation of the following codewords:

1. `00000000`
2. `10001`
3. `0000001`
4. `1111`

Answer

1. 0
2. X^4+1
3. 1
4. X^3+X^2+X+1

Control Question 64

Considering the two codewords z_1 and z_2 of a cyclic code, what is $z_1 \cdot z_2$ in the following cases:

1. $z_1 = \texttt{010110}$, $z_2 = \texttt{000100}$
2. $z_1 = \texttt{1010}$, $z_2 = \texttt{0101}$
3. $z_1 = \texttt{11001}$, $z_2 = \texttt{01010}$

Answer

1. `011001`, this one is easy since z_2 corresponds to a 2-left shift.
2. 0

   There are two ways to get this result. Either direct polynomial computation $\texttt{1010} \cdot \texttt{0101} = (X^3 + X) \cdot (X^2 + 1) = X^5 + X^3 + X^3 + X = X^5 + X = X + X \mod (X^4 - 1) = 0$

   or understanding that multiplying by `0101` means adding the 2-left shift (`0100`) with the message itself ($\texttt{0101} = \texttt{0100} + \texttt{0001}$), which in this case lead to 0 since the 2-left shift is the same as the message itself.
3. `11101`

The condition defining cyclic codes can now be used to characterize further cyclic code using polynomial properties:

Property 6.16 *If $z(X)$ is the polynomial corresponding to a codeword z of a cyclic code of size n, then for any polynomial $p(X)$, $p(X) \cdot z(X) \mod (X^n - 1)$ is also a polynomial corresponding to a codeword of this code (left-shifts and linear combinations).*

PROOF
- for any k, $X^k \cdot z(X) \mod (X^n - 1)$ is also a polynomial corresponding to a codeword of this code (left-shifts)
- a cyclic code is a linear code, so any linear combination of codewords is also a codeword.

A cyclic code corresponds therefore to an ideal of the ring $\mathrm{GF}(X)$.

Theorem 6.5 *For every (n, m) cyclic code $\mathcal{C}$, there exist one polynomial $g_{\mathcal{C}}(X)$ of degree $n - m$ such that*

$$\mathcal{C} = \{g_{\mathcal{C}}(X) \cdot p : p \in GF(X), \deg(p) < m\}$$

i.e. every codeword polynomial is a multiple of $g_{\mathcal{C}}(X)$ and conversely.
In other words, the code $\mathcal{C}$ is generated by $g_{\mathcal{C}}(X)$. $g_{\mathcal{C}}(X)$ is actually called the generator *of $\mathcal{C}$.*

PROOF This simply comes from the fact that GF(X), as any polynomial ring in one variable over a field, is a principal ring: every ideal is principal; i.e. can be generated by a single element. ■

Coding a word u using a cyclic code $\mathcal{C}$ could then simply consist of sending $z(X) = u(X) \cdot g_{\mathcal{C}}(X)$. However, systematic form coding, i.e. coding in such a way that the first symbols corresponds to the message itself is often preferred.

For a (n, m) cyclic code the procedure is then the following:

1. multiply the message polynonmial $u(X)$ by X^{n-m} (i.e. in practice to $n-m$ left shifts of the message) Notice that $n-m$ is the degree of the generator.
2. divide $X^{n-m}u(X)$ by the generator $g(X)$ and get the remainder $r(X)$
3. Then encoding of $u(X)$ is then $z(X) = X^{n-m}u(X) - r(X)$ (which is a multiple of $g(X)$, the m higher symbols of which correspond to the m symbols of u).

Example 6.24 (Systematic Coding with Cyclic Code) Consider for instance the $(7, 4)$ binary cyclic code

$$z_1 = 0000000,\ z_2 = 0001011,\ z_3 = 0010110,\ z_4 = 0101100,$$

$$z_5 = 1011000,\ z_6 = 0110001,\ z_7 = 1100010,\ z_8 = 1000101,$$

$$z_9 = 1010011,\ z_{10} = 0100111,\ z_{11} = 1001110,\ z_{12} = 0011101,$$

$$z_{13} = 0111010,\ z_{14} = 1110100,\ z_{15} = 1101001,\ z_{16} = 1111111$$

This code has for generator $g(X) = z_2(X) = X^3 + X + 1$.

[It is left as an exercise that $z_2(X)$ is actually a generator of this code.]

Using this code, we want to transmit the message $u = 1101$, i.e. $u(X) = X^3 + X^2 + 1$.

Let us first divide $X^3u(X) = X^6 + X^5 + X^3$ by $g(X)$:

$$\begin{array}{l|l}
X^6+X^5+\quad X^3 & X^3+X+1 \\
\cline{2-2}
X^6+\quad X^4+X^3 & X^3+X^2+X+1 \\
\quad X^5+\quad X^3+X^2 & \\
\quad\ X^4+\quad X^2+X & \\
\quad\quad X^3+\quad X+1 & \\
\cline{1-1}
\quad\quad\quad\quad 1 &
\end{array}$$

i.e. $X^3u(X) = (X^3 + X^2 + X + 1)g(X) + 1$.

Thus the codeword is $z(X) = X^3u(X) + 1 = X^6 + X^5 + X^3 + 1$ which represents 1101001.

To summarize: the message 1101 is encoded 1101001 by the above cyclic code. Notice that, as wanted, the first 4 bits are the bits of the original message u.

Theorem 6.6 *The generator of a cyclic code of length n is a factor of $X^n - 1$.*

PROOF Consider a cyclic code of length n the generator of which is $g(X)$ of degree r.

Then $X^{(n-r)}g(X)$ is of degree n and can be written (Euclidean division by (X^n-1)):

$$X^{(n-r)}g(X) = (X^n - 1) + q(X)$$

with $q(X)$ a polynomial of degree less than n.

Since $g(X)$ is a codeword of cyclic code of length n, $q(X) = X^{(n-r)}g(X) \mod (X^n-1)$ is also a codeword of this code (Property 6.16).

Thus (Theorem 6.5), there exists $p(X)$ such that $q(X) = p(X) \cdot g(X)$.

Coming back to (62), we have:

$$X^{(n-r)}g(X) = (X^n - 1) + p(X) \cdot g(X)$$

i.e.

$$X^n - 1 = \Big(X^{(n-r)} - p(X)\Big) \cdot g(X)$$

Thus $g(X)$ is a factor of $X^n - 1$. ■

Let us now see the converse:

Theorem 6.7 *Every factor of X^n-1 of degree r is the generator of a $(n, n-r)$ cyclic code.*

PROOF Let $g(X)$ be a factor of $X^n - 1$ of degree r. The $n - r$ polynomials $g(X), Xg(X), ..., X^{(n-r-1)}g(X)$ are all of degree less than n. Furthermore, any linear combination of these $n - r$ polynomials is also a polynomial of degree less than n, i.e. $g(X), Xg(X), ..., X^{(n-r-1)}g(X)$ is a basis of a vector subspace of GF(X). Thus $g(X), Xg(X), ..., X^{(n-r-1)}g(X)$ generates a $(n, n-r)$ linear code.

Is this code cyclic? Let $z(X) = z_0 + z_1X + ... + z_{n-1}X^{(n-1)}$ be one codeword of this code. Then

$$\begin{aligned} X \cdot z(X) &= z_0X + z_1X^2 + ... + z_{n-1}X^n \\ &= z_{n-1}(X^n - 1) + z_{n-1} + z_0X + z_1X^2 + ... + z_{n-2}X^{(n-1)} \\ &= z_{n-1}(X^n - 1) + y(X) \end{aligned}$$

where $y(X)$ is a left shift of $z(Y)$. But $z(X)$ is a multiple of $g(X)$ since it is a codeword of the linear code generated by $g(X), Xg(X), ..., X^{(n-r-1)}g(X)$, and $X^n - 1$ is also a multiple of $g(X)$ by assumption. Thus $y(X) = X \cdot z(X) - z_{n-1}(X^n - 1)$ appears also to be a multiple of $g(X)$, i.e. is an element of the subspace generated by $g(X), Xg(X), ..., X^{(n-r-1)}g(X)$.

Therefore any left shift of a codeword is also a codeword, i.e. the code is cyclic. ■

Control Question 65

How are the following messages encoded in systematic form with a code, the generator

of which is $g(X) = X^6 + X^3 + 1$:

1. `000`
2. `111`
3. `101`

Answer

1. `000000000`

   The null word is always encoded into the null codeword. The only problem here is to know the size of the codewords. Since the degree of the generator is 6 and the length of the input messages is 3, the size of the codewords is $6 + 3 = 9$.

2. `111111111`

   $(X^2 + X + 1) \cdot X^6 = X^8 + X^7 + X^6$

   $X^8 + X^7 + X^6 = (X^2 + X + 1)g(X) + X^5 + X^4 + X^3 + X^2 + X + 1$

   $X^8 + X^7 + X^6 - (X^5 + X^4 + X^3 + X^2 + X + 1) = X^8 + X^7 + X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$ which is the codeword.

3. `101101101`

   $(X^2 + 1) \cdot X^6 = X^8 + X^6$

   $X^8 + X^6 = (X^2 + 1)g(X) + X^5 + X^3 + X^2 + 1$

   $X^8 + X^6 - (X^5 + X^3 + X^2 + 1) = X^8 + X^6 + X^5 + X^3 + X^2 + 1$ which is the codeword.

There is actually a fastest way to encode with this special code. Looking at the generator you might see that this code simple consists in repeating the message three times!

6.3.3 Decoding

We know how to encode messages with cyclic codes. What about decoding then?

The decoding process is similar to the framework used of linear codes in general:

1. first compute a syndrome from the received word (which depends only on the error, not on the emitted codeword, and which is null when the received word is a codeword)
2. then deduce the corrector (i.e. the opposite of the error)
3. Finally, apply the corrector to the received codeword.

The construction of the syndrome of a word $\widehat{z}(X)$ is simple: it is the remainder of the division of $\widehat{z}(X)$ by the generator $g(X)$ of the code.

Indeed, we know that every codeword $z(X)$ is a multiple of $g(X)$. Thus the remainder of $z(X) + e(X)$ (w.r.t. $g(X)$) is the same as the one of $e(X)$:

$$\left.\begin{array}{c} z(X) = \alpha(X)g(X) \\ e(X) = \beta(X)g(X) + s(X) \end{array}\right\} \implies z(X) + e(X) = [\alpha(X) + \beta(X)]g(X) + s(X)$$

with $\deg(s(X)) < \deg(g(X))$.

It is also clear from the above construction that the syndrome $s(X)$ is null if an only if $\widehat{z}(X)$ is a codeword (i.e. multiple of $g(X)$).

The correctors, corresponding to all the non null syndromes, can be obtained by division by $g(X)$. Notice that

- for a single error X^i of degree i less that $n-m$ (the degree of $g(X)$), the syndrome is simply X^i;
- for the single error $X^{(n-m)}$, the syndrome is $X^{(n-m)} - g(X)$.

Example 6.25 (Decoding with cyclic code) Let us continue with the previous example: the message 1101 has been encoded into 1101001 and is now transmitted over a noisy channel.

Suppose that the second symbol has been flipped, i.e. that we receive 1001001. What is the corresponding decoded word?

1001001 corresponds to $\widehat{z}(X) = X^6 + X^3 + 1$, the division of which by $g(X)$ gives:

$$\begin{array}{l|l} X^6+ \quad X^3+ \quad 1 & X^3+X+1 \\ \cline{2-2} X^6+X^4+X^3 & X^3+X \\ \cline{1-1} \quad X^4+ \quad X^2+X & \\ \cline{2-2} \quad X^2+X+1 & \end{array}$$

Thus the syndrome is here $X^2 + X + 1$.

The corrector/syndrome table for $g(X) = X^3 + X + 1$ is the following:

syndrome	corrector
1	1
X	X
X^2	X^2
$X+1$	X^3
X^2+X	X^4
X^2+X+1	X^5
X^2+1	X^6

[The four first rows have been obtained using the above notices. The last three by division of the error by $g(X)$]

Thus we found that the corrector has to be X^5 and the decoded word is finally $z(X) = \widehat{z}(X) + X^5 = X^6 + X^5 + X^3 + 1$, i.e. 1101001.

Since a systematic code is being used, the first 4 symbols of this codeword are the 4 bits of the original message: 1101.

Control Question 66

Consider the $(7, 4)$ linear code, the generator of which is $g(X) = X^3 + X^2 + 1$. How will the following received messages be decoded

1. `1001011`
2. `1011001`
3. `0000001`

(provided that systematic form coding was used).

Answer

1. `1001`

   Indeed, `1001011` is a codeword:

   $$\texttt{1001011} = X^6 + X^3 + X + 1 = (X^3 + X^2 + X + 1)(X^3 + X^2 + 1)$$

   Since systematic form coding was used, the four first bits are the message.

2. `1010`

   $$\texttt{1011001} = X^6 + X^4 + X^3 + 1 = (X^3 + X^2)g(X) + X^2 + 1$$

   but $X^2 + 1 = g(X) + X^3$, thus there is an error pattern of weight 1 ($e = X^3$) whose syndrome is $X^2 + 1$.

   Minimum distance decoding thus lead to $X^6 + X^4 + X^3 + 1 + X^3 = \texttt{1010001}$.

3. `0000` (from corrected codeword `0000000`).

Summary for Chapter 6

Cyclic Code: a linear code such that any (symbol) shift of any codeword is also a codeword.

Polynomial Representation: $z = z_1...z_n$ is represented by the polynomial $z(X) = z_1 \cdot X^{n-1} + z_2 \cdot X^{n-2} + ... + z_{n-1} \cdot X + z_n$, i.e. the jth symbol z_j of a codeword z of size n is the coefficient of X^{n-j} in the corresponding polynomial $z(X)$.

The polynomial multiplication by X (degree one monome) corresponds to the (1 position) left shift.

All operations are made modulo $X^n - 1$.

Generator: For every cyclic code, there exist one polynomial such that every codeword polynomial representation is a multiple of it and conversely.

Systematic Form Encoding: The encoding method such that the m first symbols of a codeword are exactly the m symbols of the encoded message.

For cyclic codes, systematic form encoding is achieved through the following steps:

1. multiply the message polynonmial $u(X)$ by X^{n-m}
2. divide $X^{n-m}\,u(X)$ by the generator $g(X)$ and get the remainder $r(X)$
3. encode $u(X)$ by $z(X) = X^{n-m}u(X) - r(X)$.

Decoding: The decoding process is similar to the framework used of linear codes in general:

1. compute the syndrome of the received word: it is the remainder of the division of this word by the generator of the code;
2. then deduce the corrector from a precomputed mapping of syndromes to correctors (the division of the corrector by the generator gives the syndrome as the remainder)
3. Finally, apply the corrector to the received codeword and decode the original words as the first m symbols of the decoded codeword (provided that systematic encoding has been used).

6.4 Convolutional Codes

Learning Objectives for Section 6.4

After studying this section, you should know:

1. what convolutional codes are;
2. how encoding of such codes is achieved;
3. what a state and state diagram are;
4. what is the lattice representation associated with a convolutional code;
5. how to use Viterbi algorithm on lattices to do minimum distance decoding;
6. how to compute the minimal distance of a convolutional code.

6.4.1 Introduction

In this section, a non-block error coding framework is considered: convolutional codes. Convolutional codes differ from block codes in that the coding mechanism keeps memory about the encoded symbols.

In one sense, convolutional codes can appear as unbounded block codes, i.e. block codes with "infinite" size blocks. However, there is an significant difference in the design of these coding/decoding techniques. Furthermore, convolutional codes have been found much superior to block-codes in many applications.

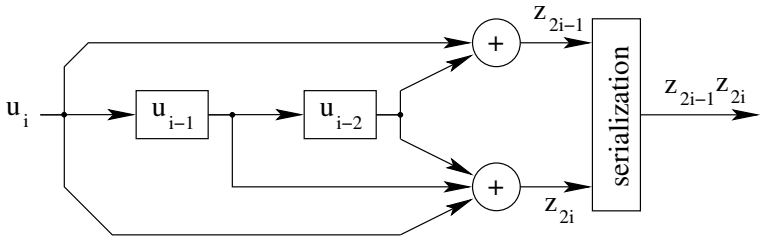


Figure 6.1: A first example of a convolutional encoder. Each message symbol u_i is encoded into two codeword symbols z_{2i-1} and z_{2i}.

6.4.2 Encoding

The starting point of a convolutional code is the encoder. Rather than beginning with precise definitions and a general analysis of convolutional codes, we prefer to start with a simple example that still contains the main features of convolutional coding.

The encoder of the example chosen for this section is depicted in Figure 6.1.

At each time step i, one message symbol u_i enters the encoder and two codewords symbols z_{2i-1} z_{2i} are emitted; i.e. $u = (u_1, ..., u_i, ...)$ is encoded into $z = (z_1, z_2, ..., z_{2i-1}, z_{2i}, ...)$. The rate of this code is thus 1/2.

The message symbols u_i and the codeword symbols z_j considered here are all binary digits. The additions shown in Figure 6.1 are binary addition (i.e. "exclusive-or").

More formally, the encode depicted in Figure 6.1 can be written as

$$z_{2i-1} = u_i + u_{i-2} \tag{6.5}$$

$$z_{2i} = u_i + u_{i-1} + u_{i-2} \tag{6.6}$$

i.e.

$$u_i \mapsto (u_{i-2} + u_i,\ u_{i-2} + u_{i-1} + u_i)$$

These equations can be viewed as a "discrete convolution" of the input sequence with the sequences $1, 0, 1, 0, 0, \ldots$ and $1, 1, 1, 0, 0, \ldots$, respectively. This explains the name "convolutional code".

However, nor the above equations, nor Figure 6.1, fully determine the codewords since the values of u_{i-2} and u_{i-1} are required. What are they at time $i = 1$, i.e. what is the initial state of the system?

The convention is that there are always null, i.e. $u_{-1} = u0 = 0$.

To ensure that this is always the case, i.e. that whenever a new message has to be encoded the initial state of the encoder is always 0 in all memories, the encoding of a former message must leave the encoder in this null state. Thus every encoding of a message must contain enough zeros at then end so as to ensure that all the memories of the system have return to 0. In the case of the encoder presented in Figure 6.1, this means that the encoding of every message will finish by encoding two more zeros.

Example 6.26 (Coding with convolutional code) Suppose we want to then the message $u = \texttt{101}$ using the encoder depicted in Figure 6.1. How does it work?

Let us trace all the components of the encoding:

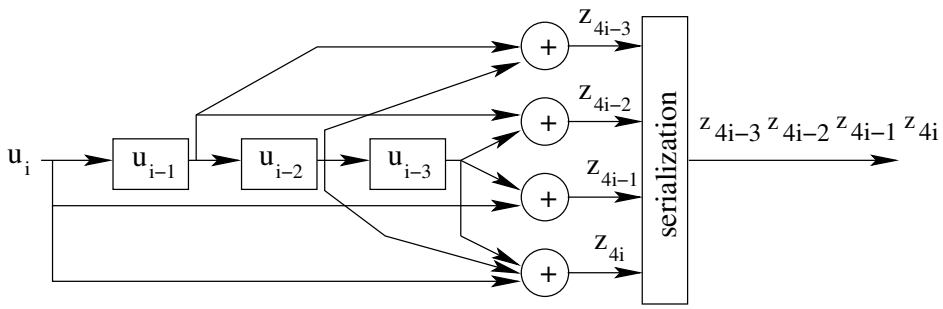
i	u_i	State $(u_{i-1}u_{i-2})$	$z_{2i-1}z_{2i}$
1	1	00	11
2	0	10	01
3	1	01	00
4	(0)	10	01
5	(0)	01	11

The corresponding codeword is thus $z =$ 1101000111.

The last two line corresponds to the two zero bit that must be introduced in the coder at the end of every message so as to put the coder back into its initial state.

Control Question 67

Consider the convolutional code, the encoder of which is described by the following diagram:



1. How many zeros must be added after each word to encode?

   (a) 1

   (b) 2

   (c) 3

   (d) 4

   (e) 5

2. How is 10110 encoded?

   (a) 00111100101010001001111001111

   (b) 10110

   (c) 00111100101010000101111001110000

   (d) 00100101100110101100010010001

   (e) 11110000111111111000000000000000

   (f) 11000011010100011010011111100000

Answer

1. (c)
2. (c): 00111100101010000101111001110000

u_i	u_{i-1}	u_{i-2}	u_{i-3}	z_{4i-3}	z_{4i-2}	z_{4i-1}	z_{4i}
1	0	0	0	0	0	1	1
0	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0
1	1	0	1	1	0	0	0
0	1	1	0	0	1	0	1
0	0	1	1	1	1	1	0
0	0	0	1	0	1	1	1
0	0	0	0	0	0	0	0

Let us now wonder what is in general the code generated by the encoder depicted in Figure 6.1?

Consider for instance a 3 bits message $u = (u_1, u_2, u_3)$. As we have seen, what has to be actually encoded is $(u_1, u_2, u_3, 0, 0)$, i.e two zero bits are added at the end of the original message in order to put the memory of the encoder back into its initial state. The size of the corresponding codeword is thus $2 \cdot 5 = 10$.

The bits of this codewords are given by equations (6.5) and (6.6), i.e. in a matrix format:

$$(z_{2i-1}, z_{2i}) = (u_{i-2}, u_{i-1}, u_i) \cdot \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \tag{6.7}$$

Thus the complete codeword z is obtained by multiplying $(u_1, u_2, u_3, 0, 0)$ by the matrix

$$G_3 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

or more simply by multiplying $u = (u_1, u_2, u_3)$ by

$$G_3 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

The encoded codeword is thus $z = u \cdot G_3$.

For a message of length m, this generalizes to $z = u \cdot G_m$ where G_m is the $m \times (2\,m+4)$ matrix made of one row two columns shifts of the small block matrix of equation (6.7).

This result is true in general, independently of the length of the encoded message. This illustrates why convolutional codes are presented as "unbounded" linear block codes.

6.4.3 General Definition

Let us now give a general definition of convolutional codes.

Definition 6.15 A (n, k, r) D-ary convolutional code is an unbounded linear code, the generator matrix of which is in the following form (infinite):

$$G = \begin{bmatrix} F_0 & F_1 & F_2 & \cdots & F_r & [0] & [0] & [0] & \cdots \\ [0] & F_0 & F_1 & \cdots & F_{r-1} & F_r & [0] & [0] & \cdots \\ [0] & [0] & F_0 & \cdots & F_{r-2} & F_{r-1} & F_r & [0] & \cdots \\ \vdots & \vdots & \ddots & \ddots & & & & \ddots & \ddots \end{bmatrix}$$

with F_i a $k \times n$ matrix, and $[0]$ the $k \times n$ null matrix; i.e. each set of k rows of G is the same as the previous set of k rows but shifted n places right.
A message u of finite length m, $u = (u_1, ..., u_m)$ is encoded by $z = \overline{u} \cdot G_{m'}$ where $\overline{u}$ is the vector of length $m' = qk$, with $q = \lceil \frac{m}{k} \rceil$, such that $\overline{u} = (u_1, ..., u_m, 0, ..., 0)$, and $G_{m'}$ is the top left submatrix of G of size $qk \times n(r+q)$. Notice that $\overline{u} = u$, i.e. $m' = m$ if m is a multiple of k (in particular when $k = 1$!). ♦

In the above definition, k actually corresponds to the number of message symbols that are going in the encoder (k input lines), n is the number of outgoing codeword symbols per input (n output lines) and r is the maximum number of memories (a.k.a. "registers") on one input line.

Example 6.27 The example encoder of Figure 6.1 builds a $(2, 1, 2)$ convolutional code: $k = 1$ input line with $r = 2$ memories, producing $n = 2$ codeword bits for each input bit.

As seen in Section 6.4.2, for an input message of length 3, its generator matrix is the 3×10 matrix

$$G_3 = \begin{bmatrix} \boxed{1\ 1}\ \boxed{0\ 1}\ \boxed{1\ 1}\ 0\ 0\ 0\ 0 \\ 0\ 0\ \boxed{1\ 1}\ \boxed{0\ 1}\ \boxed{1\ 1}\ 0\ 0 \\ 0\ 0\ 0\ 0\ \boxed{1\ 1}\ \boxed{0\ 1}\ \boxed{1\ 1} \end{bmatrix}$$

where indeed each row ("set of $k = 1$ row(s)") is a 2-left shift of the row above it.

Regarding the definition, we have for the above matrix G_3:

- $F_0 = [1\ 1]$, corresponding to the two coefficients of u_i in equations (6.5) and (6.6),
- $F_1 = [0\ 1]$, corresponding to the two coefficients of u_{i-1},
- and $F_2 = [1\ 1]$, corresponding to the two coefficients of u_{i-2}.

Notice that convolutional codes are linear: any combination of codewords is also a codeword (with the convention that smaller codewords are padded with zeros at the end such that the linear combination makes sense, i.e. all added words with the same length).

Control Question 68

1. What are (n, k, r) of the convolutional code given in the last question?

(a) $(1, 3, 4)$
(b) $(7, 4, 1)$
(c) $(3, 1, 4)$
(d) $(4, 1, 3)$
(e) $(7, 1, 4)$
(f) $(1, 3, 7)$

2. What is the generator matrix of this code?

   (a) How many F blocks are there?
   (b) What is the size of each F block: ?×? ?
   (c) Give all the F blocks.

Answer

1. 4: $(4, 1, 3)$

2. (a) $r = 3$
   (b) 1×4
   (c) $F_0 = [\,0\ 0\ 1\ 1\,]$, $F_1 = [\,1\ 1\ 0\ 0\,]$, $F_2 = [\,1\ 0\ 0\ 1\,]$, $F_3 = [\,0\ 1\ 1\ 1\,]$.

   Indeed,

$$z = u \cdot \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

   For instance:

$$G_4 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

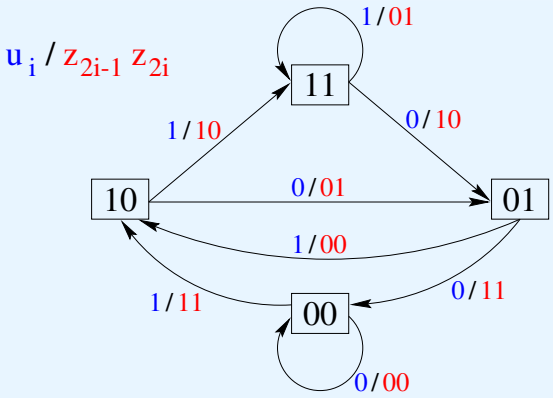
6.4.4 Lattice Representation

The "state" of a system is the set of internal parameters (memories, a.k.a. "registers"), that is required to compute the corresponding output to a given input block. For the encoder of Figure 6.1 for instance, the state at time i is the current contents of the two memories, i.e. $S_i = (u_{i-1}, u_{i-2})$.

The behavior of the encoder depends only on its state and the input: a convolutional code encoder is a **state machine**. All its behaviors are described by the state diagram.

Definition 6.16 (State Diagram) The state diagram of an encoder is a graph, the nodes of which are all possible internal states of the encoder. An arc between a node S_i and a node S_j in this graph represents the fact there exists an input that, when received in state S_i makes the encoder go in state S_j.
These arcs are usually labeled with the input symbol(s) and the corresponding output symbols. ♦

Example 6.28 (State Diagram) For instance, for the encoder of Figure 6.1, we have:



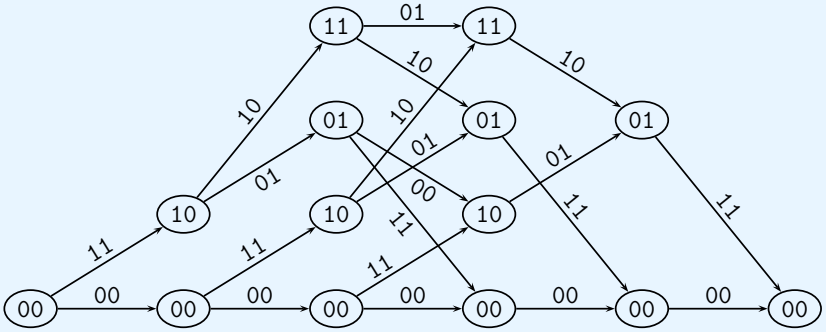
where each node represent the state of the encoder i.e. the states of the two internal memories, the blue label is the input that produces the state change, and the red label is the corresponding output symbols.

For instance, if in stage `01`, a `1` is received as input symbols, then the state becomes `10` and the two output symbols are `00`.

The set of codewords of a (n, k, r) convolutional code corresponding to all possible messages of m bits thus corresponds to the set of all paths of length $n(r + \lceil \frac{m}{k} \rceil)$ in the state diagram that start from the null state (all zero) and return to this null state.

The unfolding in time of all the paths of the same length $n(r + \lceil \frac{m}{k} \rceil)$ in the state diagram is called the *lattice** of size m of the (n, k, r) convolutional code.

Example 6.29 (Lattice) For the $(2, 1, 2)$ code considered in previous examples, the lattice of length $m = 3$, representing the encodings of all input messages of 3 bits is:

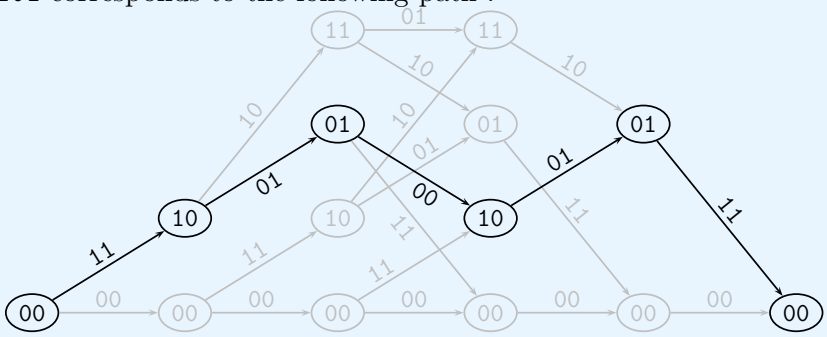


in which a the uppermost arc out of a node corresponds to input bit `1` and the lowest

arc to 0.

The first three columns of arcs thus corresponds to the encoding of the 3 message bits, and the last two columns of arcs corresponds to the ending zeros[14], having thus only bottommost arcs.

Example 6.30 (Encoding in the Lattice) For instance, the encoding of the message $u = 101$ corresponds to the following path :



i.e. to the codeword $z = 1101000111$.

Control Question 69

Consider the lattice corresponding to the encoding of a message of length 3 with the encoder of the former control question.

How many columns of states does it have?

For each column, how many states are there?

Give the arc label for the following pairs of states. If no arc is present, answer "no arc":

from 100 to 010:
from 101 to 110:
from 001 to 111:
from 111 to 011:

Answer

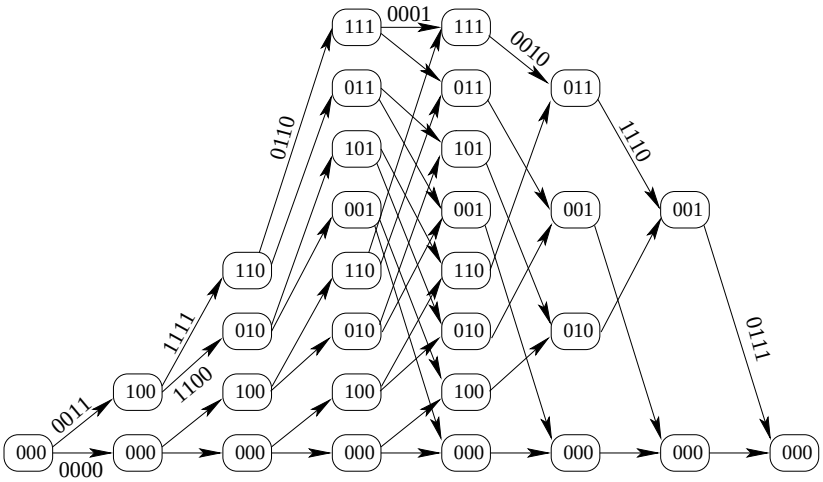
8 columns

1,2,4,8,8,4,2,1.

The answer all 8 could also be considered as correct (all column full although sub obtimal since many states are unreachable in such a lattice).

from 100 to 010: 1100
from 101 to 110: 1000
from 001 to 101: no arc
from 111 to 011: 0010

Here is an example of the lattice (not all labels have been specified):



and here are the codewords emited in each possible situation:

$S_i \backslash u_i$	0	1
000	0000	0011
001	0111	0100
010	1001	1010
011	1110	1101
100	1100	1111
101	1011	1000
110	0101	0110
111	0010	0001

6.4.5 Decoding

As we have just seen, a codeword correspond to one path form the starting node to the ending node of the lattice.

Decoding consist thus in finding the most appropriate path corresponding to the received message to decode. In the framework of minimal distance decoding, i.e. decoding the codeword with minimal number of errors, "*the most appropriate path*" means the path with the minimal Hamming distance with the message to be decoded.

Finding this "closest" path can be done using dynamic programming, i.e. the Viterbi algorithm.

This algorithm is decoding one codeword block after the other (i.e. n message bits after the others), keeping at each stage only the locally optimal solutions, i.e. for each node in the column corresponding to the current decoded block, keeping the best path that come to this node.

At the end, the best path found for the last node is the decoded message.

What is important and useful for this algorithm is that the number of possible best paths that are kept at each time step is always less than or equal to the number of states of the encoder. The algorithm is thus of linear complexity, instead of exponential complexity of the naive algorithm which consists in comparing all the paths with the message to be decoded.

Let us now give the algorithm more precisely.

Let's first introduce a bit of notation. For each state s of the encoder, let $\gamma_i(s)$ be the best (i.e. closest, i.e. with minimum number of errors) decoding of length i ending in state s

$$\gamma_i(s) = \min_{\substack{z_1,\dots,z_{2i} \\ \text{ending in } s}} d(z_1 \dots z_{2i}, \widehat{z}_1 \dots \widehat{z}_{2i})$$

The complete decoding thus corresponds to $\gamma_{|\widehat{z}|}(00)$ $\left(|\widehat{z}| = m + 2 = \frac{n}{2} \text{ ici}\right)$ where $|\widehat{z}|$ is the length of the message to be decoded.

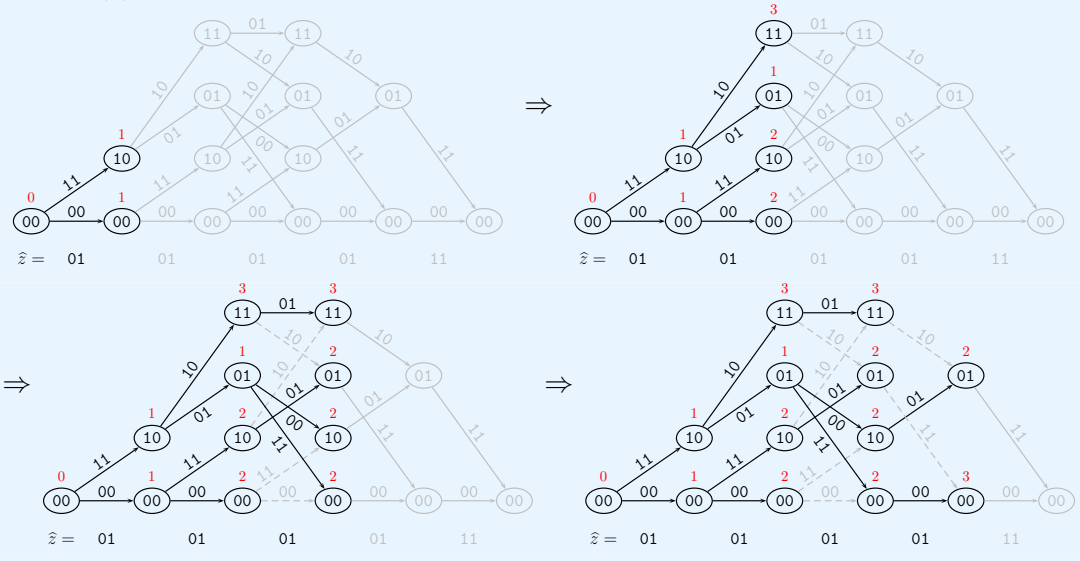
It is easy to see that for every couple of encode states (s, s'), we have:

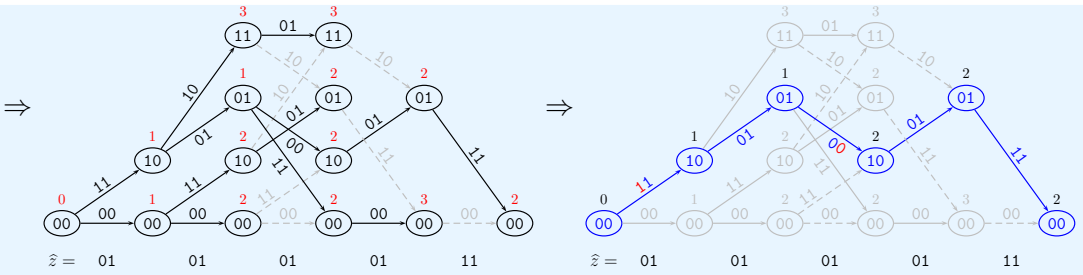
$$\gamma_i(s) = \min_{\substack{z_{2i-1}z_{2i} \\ \text{from } s' \text{ to } s}} \Big(d(z_{2i-1}z_{2i}, \widehat{z}_{2i-1}\widehat{z}_{2i}) + \gamma_{i-1}(s') \Big).$$

This leads to the following algorithm (Viterbi algorithm):

$\gamma_0(00) = 0$
for i from 1 to $|\widehat{z}|$ **do**
  for all s **do**
    $\gamma_i(s) = \min_{s' \to s} \left(d(z_{2i-1}z_{2i}, \widehat{z}_{2i-1}\widehat{z}_{2i}) + \gamma_{i-1}(s') \right)$
    mark the/one arc from s' to s that achieves the minimum
  end for
end for
reconstruct the optimal path backwards from the ending to the beginning null state

Example 6.31 Suppose that the codeword $z = 1101000111$ is sent over a noisy channel and that $\widehat{z} = 0101010111$ is received, i.e. two errors occurred. The above algorithm will then go through the following steps, where the minimal number of error $\gamma_i(s)$ at this stage i is written in red above every state s in the lattice:





In the first step, only two path can be considered:

- either a `1` was encoded, leading from state `00` to state `10`. If this occurred then `11` was emitted and thus one error occurred during the transmission (since `01` has been received).
- or a `0` was encoded, leading from state `00` to state `00`. If this occurred then `00` was emitted and one error also occurred during the transmission.

At the second step there is still only one possible path to reach each state (i.e. 4 paths now). The minimal number of errors for each of this 4 states is the minimal number of errors of the former state plus the number of error of the corresponding path (i.e. the difference between the emitted and the received symbols). For instance, would the path going from state `10` to state `11` have been used, two errors occurred then since in such a case `10` would have been emitted and `01` is received. This leads to a minimum number of errors for state `11` at step 2 $\gamma_2(\texttt{11}) = 1 + 2 = 3$.

At step three, we have two possible paths to reach each state. The algorithm keeps only one of the two, where the minimum number of errors is done. The other arc is drawn with dashed gray line.

The algorithm goes on like this up to the final stage, where the null state is reached with a minimal number of error of 2.

The very last step is the backward reconstruction of the path, which is drawn in blue on the last picture. This path corresponds to the codeword `1101000111`.

The final step of the decoding is to reconstruct the original message from the codeword, which is done by knowing which arc (`0` or `1`) has been followed (or simply by looking at the first bit of each state). In this case, we come to `10100`, and suppressing the last 2 zero which are not part of the original message, we end up with `101`.

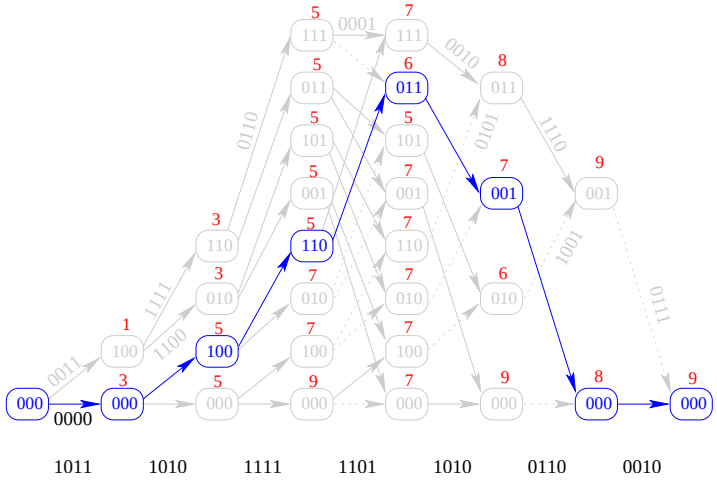
Control Question 70

For the code used in the former control question, how is `10111010111111011010011000 10` decoded?

Answer

The answer is `0110`.

Here is the corresponding treillis with minimum number of errors at each node:



The received message 1011 1010 1111 1101 1010 0110 0010 has thus 9 errors. The emitted codeword was 0000 0011 1111 0101 1110 0111 0000 which correspond to the encoded message 0110000, i.e. the original message 0110.

6.4.6 Minimum Distance

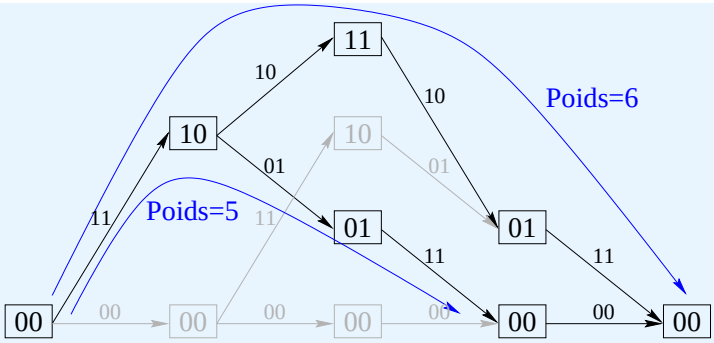
As presented in section 6.4.3, a convolutional code is a linear code. Thus, by Theorem 6.2, its minimum distance is equal to its minimal weight.

Property 6.17 (Minimal Weight of a Convolutional Code) *For a convolutional code, the minimal weight is the minimal number of non zero symbols on a path going from the null state to the null state.*

PROOF This comes directly from definition of minimal weight and the fact that a codeword corresponds to a path in the state diagram going from the null state back to the null state ■

Example 6.32 Consider the convolutional code we have been dealing with since the beginning of this section. The best way to represent paths in the state diagram is to use the lattice. For minimal weight computation the lattice of size two is enough since every arc in the lattice of size three that is not present in the lattice of size two will come back into a same state with a bigger number of non zero symbol, so cannot be part of the minimal path.

In out case, this gives us:

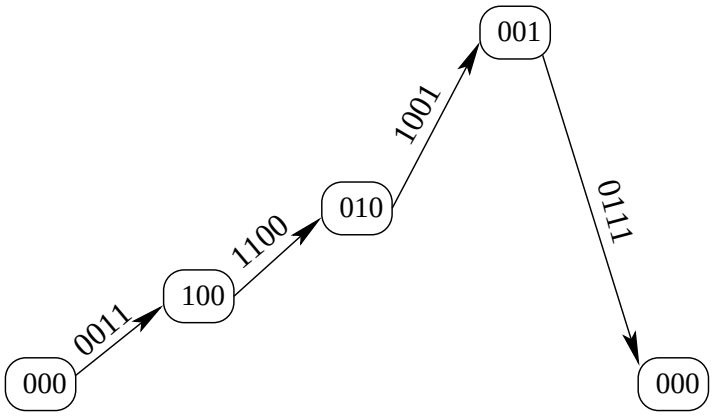


Thus $d_{\min}(\mathcal{C}) = 5$.

Control Question 71

What is the minimum distance of the encoder given in former control questions?
Answer

$d_{\min}(\mathcal{C}) = 9$. Here is the codeword with this weight:



e-pendix: Convolution Code

Summary for Chapter 6

convolutional code: A (n, k, r) D-ary convolutional code is an unbounded linear code, the generator (infinite) matrix of which is such that each set of k rows is the same as the previous set of k rows but shifted n places right.

This corresponds to the matrix description of the encoder algorithm which is often given in the form of a picture of a circuit with k input lines, n output lines and at most r memories on a input to output path.

encoding: A message u of finite length m, $u = (u_1, ..., u_m)$ is encoded by $z = \overline{u} \cdot G_{m'}$ where $q = \lceil \frac{m}{k} \rceil$, $m' = qk$, $\overline{u}$ is the vector of length m' such that $\overline{u} = (u_1, ..., u_m, 0, ..., 0)$, and $G_{m'}$ is the top left submatrix of size $qk \times n(r + q)$ of the generator matrix.

encoder (internal) state: the set of states of the memories (or registers) of the encoder

state diagram: The state diagram of an encoder is a graph, the nodes of which are all possible internal states of the encoder. An arc between a node S_i and a node S_j in this graph represents the fact there exists an input that, when received in state S_i makes the encoder go in state S_j.

These arcs are labeled with the input symbol(s) and the corresponding output symbols.

lattice representation: The time unfolded representation of all possible paths in the state diagram.

Viterbi decoding algorithm: The dynamic programming algorithm which, for a given message to be decoded, finds the "shortest" path (in terms of number of errors) in the lattice.

Summary for Chapter 6

block-code: a non-empty set of words of the same length, considered as "row vectors".

weight: (of a word) the number of non-zero symbols.

Hamming distance: the number of coordinates in which two vectors differ.

The Hamming distance between two words is the weight of their difference.

minimum distance decoding: error correction framework in which each received word is decoded into the closest (according to Hamming distance) codeword.

maximum likelihood decoding: error correction framework in which each received word $\widehat{z}$ is decoded into (one of) the most likely codeword(s) z, i.e. a codeword such that $P(Y = \widehat{z}|X = z)$ is maximal (with X the input of the noisy channel and Y its output).

minimum distance of a code: the minimum (non null) Hamming distance between any two (different) codewords.

error correcting and detecting capacity: A block-code $\mathcal{C}$ of length n using minimum distance decoding can, for any two integers t and s such that $0 \leq t \leq n$ and $0 \leq s \leq n-t$, correct all patterns of t or fewer errors and detect all patterns of $t+1, ..., t+s$ errors if and only if its minimum distance $d_{\min}(\mathcal{C})$ is strictly bigger than $2t+s$:

$$d_{\min}(\mathcal{C}) > 2t+s \iff \mathcal{C} \text{ corrects } t \text{ and detects } t+s \text{ errors.}$$

linear code: a block code which is a vector space (i.e. any linear combination of codewords is also a codeword).

A (n, m) D-ary linear code is a dimension m vector subspace of the dimension n vector space of D-ary words.

minimum distance of a linear code: for linear code minimum distance of the code is equal to the minimal weight.

generator matrix of a linear code: (of a (n, m) linear code) a $m \times n$ matrix the rows of which are a basis of the code (they are thus linearly independent).

systematic form of the generator matrix of a linear code: a $m \times n$ generator matrix of a (n, m) linear code is said to be in systematic form only if its left most $m \times m$ submatrix is the identity matrix (of size m).

encoding with a linear code: the encoding with linear codes is done by matrix multiplication: the word to be encoded u is multiply by one chosen generator matrix G of the code producing the codeword $z = u \cdot G$.

If the generator matrix is in systematic form, the first m symbols of the codeword are exactly the symbols of the message. Thus only the $n - m$ last symbols actually need to be computed.

verification matrix of a linear code: A $(n - m) \times n$ matrix H is a verification

matrix for a (n, m) linear code $\mathcal{C}$ if and only if

$$\forall z \in \mathrm{GF}(D)^n \; z \cdot H^T = \mathbf{0} \iff z \in \mathcal{C}$$

The verification matrix is very useful for decoding.

syndrome of a word with respect to a linear code: The result of the product of a word by the verification matrix: $s = z \cdot H^T$.

The syndrome is used to determine the error to be corrected. It indeed correspond to the linear combination of columns of H which precisely is the product of the error pattern by H^T.

binary Hamming codes:
- $(2^r - 1, 2^r - r - 1)$ linear codes that can correct all patterns of 1 error;
- the verification matrix is given in the form of the binary enumeration of the columns.

Cyclic code: a linear code such that any (symbol) shift of any codeword is also a codeword.

Polynomial representation of cyclic codes: $z = z_1...z_n$ is represented by the polynomial $z(X) = z_1 \cdot X^{n-1} + z_2 \cdot X^{n-2} + ... + z_{n-1} \cdot X + z_n$, i.e. the jth symbol z_j of a codeword z of size n is the coefficient of X^{n-j} in the corresponding polynomial $z(X)$.

The polynomial multiplication by X (degree one monome) corresponds to the (1 position) left shift.

All operations are made modulo $X^n - 1$.

Generator of a cyclic code: For every cyclic code, there exist one polynomial such that every codeword polynomial representation is a multiple of it and conversely.

Systematic form cyclic code encoding: The encoding method such that the m first symbols of a codeword are exactly the m symbols of the encoded message.

For cyclic codes, systematic form encoding is achieved through the following steps:

1. multiply the message polynonmial $u(X)$ by X^{n-m}
2. divide $X^{n-m}\,u(X)$ by the generator $g(X)$ and get the remainder $r(X)$
3. encode $u(X)$ by $z(X) = X^{n-m}u(X) - r(X)$.

Decoding with cyclic codes: The decoding process is similar to the framework used of linear codes in general:

1. compute the syndrome of the received word: it is the remainder of the division of this word by the generator of the code;
2. then deduce the corrector from a precomputed mapping of syndromes to correctors (the division of the corrector by the generator gives the syndrome as the remainder)
3. Finally, apply the corrector to the received codeword and decode the original words as the first m symbols of the decoded codeword (provided that

systematic encoding has been used).

Convolutional code: A (n, k, r) D-ary convolutional code is an unbounded linear code, the generator (infinite) matrix of which is such that each set of k rows is the same as the previous set of k rows but shifted n places right.

This corresponds to the matrix description of the encoder algorithm which is often given in the form of a picture of a circuit with k input lines, n output lines and at most r memories on a input to output path.

Encoding with convolutional code: A message u of finite length m, $u = (u_1, ..., u_m)$ is encoded by $z = \overline{u} \cdot G_{m'}$ where $q = \lceil \frac{m}{k} \rceil$, $m' = qk$, $\overline{u}$ is the vector of length m' such that $\overline{u} = (u_1, ..., u_m, 0, ..., 0)$, and $G_{m'}$ is the top left submatrix of size $qk \times n(r + q)$ of the generator matrix.

Convolutional code encoder (internal) state: the set of states of the memories (or registers) of the encoder

State diagram: The state diagram of an encoder is a graph, the nodes of which are all possible internal states of the encoder. An arc between a node S_i and a node S_j in this graph represents the fact there exists an input that, when received in state S_i makes the encoder go in state S_j.

These arcs are labeled with the input symbol(s) and the corresponding output symbols.

Lattice representation: The time unfolded representation of all possible paths in the state diagram.

Viterbi decoding algorithm: The dynamic programming algorithm which, for a given message to be decoded, finds the "shortest" path (in terms of number of errors) in the lattice.

Historical Notes and Bibliography

This section still needs to be improved.

The work on error-correcting codes started of course from Shannon pioneer work in 1948. The design of good and efficient codes started in the fifties with the works of Hamming, Slepian and many others. during the fifties, most of the work in this area was devoted to the development of a real theory of coding (linear codes, both block and convolutional).

Convolutional codes were first introduced in 1955 by Elias [3] as an alternative to block codes. Wozencraft proposed later an efficient sequential decoding method for such codes [14]. Then in 1967, Viterbi proposed a maximum-likelihood decoding algorithm [13] quite easy to implement which leads to several applications of convolutional codes, in particular deep-space satellite communications.

A theory to practice shift was made during the seventies, with a rapid growth of military and spatial communication applications.

OutLook

See also [2], [6], [12] and [8].

Chapter 7

Module I3: Cryptography

by J.-C. Chappelier

Learning Objectives for Chapter 7

In this chapter, the basics of cryptography are presented. After studying them, you should know

1. what perfect and practical security are,
2. how much secure modern ciphering systems are,
3. why security and authentication are theoretically incompatible,
4. what RSA and DES are and how they work,
5. what unicity distance is and how to compute it.

Introduction

Cryptography, as its Greek root ("hidden writing") suggested, is concerned with the *secrecy* of information. But in the modern sense, this scientific domain is also concerned with the *authenticity* of information.

In the "Information Age" we are now living in, cryptography can no longer be avoided and, indeed, became a standard tool for communications. As information can nowadays be extremely sensitive and have enormous economic value, its transmission over easily accessible channels, e.g. the Internet, sometimes requires *confidentiality* and *authenticity* to be ensured. The purpose of cryptography is to provide such guarantees.

This chapter introduces you to the basics of this rather modern field of computer sciences and study more formally its two goals: *secrecy* and *authenticity*. Roughly speaking, the goal of *secrecy* is to ensure that the message is *received* by authorized persons; the goal of *authenticity* is to ensure that the message as been *sent* by an authorized person.

7.1 General Framework

Learning Objectives for Section 7.1

After studying this section you should know:

1. what cryptography deals with;
2. how to formally describe the general framework cryptography focuses on;
3. and several historical (unsecure) ciphering examples.

7.1.1 Cryptography Goals

The framework of cryptography is to encode messages so as to ensure either secrecy or authenticity.

As described in chapter 2, a message M is a sequence of symbols out of an alphabet Σ. In cryptography, the encoding of message is called *encrypting* or *ciphering*. In the framework considered in this chapter, encrypting will be done using a function e and a key K, which is itself a finite sequence of symbols out of an alphabet, usually but not necessarily the same as the message alphabet Σ.

The encrypted message, or cryptogram, is thus $C = e(M, K)$. The encryption function is here assumed to be deterministic. C is thus perfectly determined once M and K are given, i.e. $H(C|M, K) = 0$.

The decrypting (or deciphering) is done using a function d and the key K, such that (unsurprisingly!) $d(e(M, K), K) = M$. We also assume that decrypting is deterministic, i.e. $H(M|C, K) = 0$.

Notice that $H(C|M, K) = 0$ and $H(M|C, K) = 0$ do not imply $H(K|M, C) = 0$; several keys could indeed be possible for a given (M, C) pair. In practice however this is hardly the case (and a bad idea), and almost always $H(K|M, C)$ is also 0.

The general framework cryptography focuses on can be summarize by the picture given in figure 7.1.

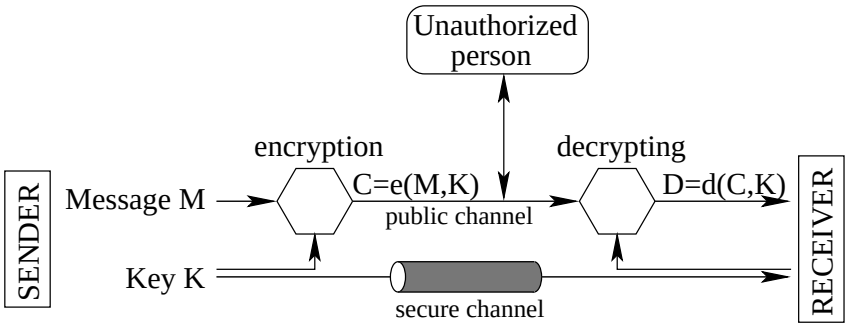


Figure 7.1: The general framework cryptography focuses on.

The goal of cryptography is to protect the message against

- wrong receipt ("*secrecy*"): it should be impossible to get the message M out of

the encrypted message $C = e(M, K)$ without knowing K;

- wrong emission ("*authentication*"): it should be impossible to substitute another message C' without knowing K.

The cryptanalysis is wondering about "cracking" security/authentication on a communication channel. "Cracking" a security systems means finding M or K knowing $C = e(M, K)$. The hypothesis usually made are:

- encrypting and decrypting algorithms are known by everybody (Kerckhoffs' hypothesis) and even statistics about the messages (but not the message itself!) could be collected;
- unauthorized persons doesn't know the key K;
- everybody can get $C = e(M, K)$ (but not M nor K).

Thus, all the secrecy is due only to the fact that the "enemies" do not know the actual value of the secret key. It is indeed risky to hope that the design of the ciphering algorithm could be safeguarded from the enemies. Nonetheless, in many applications of cryptography, notably in military and diplomatic applications, the cryptographers try to keep the ciphering algorithm as secret as possible. Kerckhoffs' hypothesis doesn't forbid this, but only warns not to count too much on the success of such safekeeping. On the other hand, Kerckhoffs would certainly have admired the designers of the Data Encryption Standard (DES) (see section 7.3.3) who published a complete description of their encryption system, and is nonetheless perhaps the most widely used cipher today.

7.1.2 Historical Examples

Before studying further the fundamentals of cryptography with the modern tools of Information Theory, let us first give three historical (but unsecure) examples of cryptosystems: substitution, transposition and Vigenère ciphers.

Substitution

The substitution cipher simply consists in replacing every symbol of the message alphabet by another symbol of this alphabet, known in advance. The key of such a system is a permutation of the alphabet Σ, which define the substitution for all the symbols.

Example 7.1 (Substitution cipher) Consider messages out of the usual alphabet made of 27 letters (including the whitespace!): $\Sigma = \{\mathrm{A}, ..., \mathrm{Z}, ' \ '\}$. A possible key k, i.e. a permutation of Σ, could be:

$$\begin{array}{lcl} \mathrm{A} & \to & \mathrm{R} \\ \mathrm{B} & \to & \mathrm{I} \\ \vdots & & \vdots \\ \mathrm{Y} & \to & \mathrm{B} \\ \mathrm{Z} & \to & \mathrm{E} \\ \text{' '} & \to & \mathrm{L} \end{array}$$

In this case $e(\text{“A BAY}'', k) = \text{“RLIRB}''$.

Transposition

In the transposition cipher, the key consists of a permutation of the $d > 1$ first integers (d is also part of the definition of the key).

The encrypting algorithm is then the following:

1. pad the message with (less than $d - 1$) whitespaces, so that the length of the message is a multiple of d;
2. split the message in blocks of length d;
3. permute the symbols in each block according to the permutation K.

Example 7.2 (Transposition Cipher) Let us take the permutation (2 4 3 1 5) as the key (thus $d = 5$).

[Note on permutation notation: (2 4 3 1 5) means that the second letter of the original message becomes the first of the encrypted message, the fourth of the original message becomes the second, etc.]

Suppose now we want to encore the message 'TRANSPOSITION CIPHER IS SIMPLE'.

The length of the message is 29, which is not a multiple of $d = 5$. One extra whitespace thus needs to be added at the end.

Then we split the message in six blocks of size 5 (whitespaces have here been marked by a dot to make them appear more clearly):

TRANS | POSIT | ION.C | IPHER | IS.SI | MPLE. |

And finally we apply the transposition to each block:

RNATS | OISPT | O.NIC | PEHIR | SS.II | PELM. |

The transmitted message is thus 'RNATSOISPTO NICPEHIRSS IIPELM' (by convention ending whitespaces could be removed).

The decoding is done exactly the same way but using the inverse permutation (which in this case is (4 1 3 2 5)).

Vigenère Cipher

The last historical example we want to present is the Vigenère cipher. In this cryptography system, the key is a sequence of symbols from the same alphabet Σ as the messages. In practice, it is very often one usual word or a sentence of a few words.

Using an order on Σ (e.g. the usual alphabetical order), this key is transformed into a sequence of integers, e.g. 'A' = 1, 'B' = 2, ..., 'Z' = 26 and ' ' = 27.

More formally, if

- n is the size of Σ,
- $i(a)$ is the position of symbol a in Σ (according to the order chosen on Σ), $1 \leq i(a) \leq n$,
- $\sigma(i)$ the i-th symbol in Σ ($1 \leq i \leq n$, otherwise consider $i \mod n$),
- the key K is made of p symbols $K = k_1...k_p$,
- and M of q symbols $M = m_1...m_q$,

then

$$\begin{aligned} C &= \sigma(i(m_1) + i(k_1))\, \sigma(i(m_2) + i(k_2)) \ldots \sigma(i(m_p) + i(k_p))\, \sigma(i(m_{p+1}) + i(k_1)) \\ &\quad \ldots \sigma(i(m_q) + i(k_{q \mod p})) \end{aligned}$$

Example 7.3 (Vigenère Cipher) Let's once again consider messages made out of the 27 English letters (including the whitespace). The key is thus a sequence of characters, for instance k ='INFORMATION'. How is the message 'VIGENERE CIPHER IS ALSO QUITE SIMPLE' encoded?

Assuming that letter 'A' corresponds to '1' and whitespace to 27, letter 'I' then corresponds to 9, and thus the first letter of the message, 'V', is encoded by 'V'+9='D', the second letter of the message 'I' is encoded by 'I'+'N'='I'+14='W', the third letter 'G' by 'G'+'F'='G'+6='M', etc.

Here is the complete encoding:

```
VIGENERE CIPHER IS ALSO QUITE SIMPLE
INFORMATIONINFORMATIONINFORMATIONINF
DWMTERSYIRWYVKFRVTTJ FXNWI FFTAX YZK
```

i.e. the encoded message is 'DWMTERSYIRWYVKFRVTTJ FXNWI FFTAX YZK'.

Summary for Chapter 7

- cryptography aims at either transmitting messages securely (only authorized persons can read it) or authenticate messages (no unauthorized persons could have send it).
- To do so, the clear messages M are encoded using a key K and a deterministic function: $C = e(M, K)$.
- Encrypted messages can be decoded deterministically using the decoding function d and the same key K, so that $d(e(M, K), K) = M$.
- $H(C|M, K) = 0$.
- $H(M|C, K) = 0$.

7.2 Perfect Secrecy

Learning Objectives for Section 7.2

After studying this section you should know:

1. what is a perfectly secret cryptosystem;
2. one example of such a cryptosystem;
3. and for imperfectly secure systems, how to estimate the maximum message size that can be securely transmitted.

After the entertaining historical examples of the last section, let us now come into the modern science of cryptography. This begins with a information theoretical definition of what a good ("perfect" is the word used by Shannon) cryptographic system is.

7.2.1 Definition and Consequences

In the framework depicted in figure 7.1, where only encrypted messages can be captured by the unauthorized persons[1], the system will be safe if the encrypted message does not bring any information on the original message, i.e. if

$$I(C; M) = 0,$$

which also means that M and C are independent random variables.

Definition 7.1 (Perfect Secrecy) A encryption system is said to be *perfect*, i.e. provide *perfect secrecy*, if and only if the mutual information of the clear message M with the encrypted messages C is null: $I(C; M) = 0$. ♦

Theorem 7.1 *In a perfect ciphering system, there must be at least as many possible keys as possible messages.*

Proof

$$I(C; M) = 0$$

implies that for every message m, $P(C|M = m) = P(C)$.

Let us now consider a possible cryptogram, i.e. an encrypted message c such that $P(C = c) \neq 0$.

Thus for every possible original message m, we have $P(C = c|M = m) \neq 0$ which means that for every m there exist a key, denoted $k(m)$, such that $c = e(k(m), m)$.

Furthermore, $m \neq m' \implies k(m) \neq k(m')$ otherwise deciphering would no longer be deterministic: we would have two different messages that with the same key give the same cryptogram c!

There are thus at least as many keys as there are possible messages m. ■

[1] This kind of attack is called *ciphertext-only attack*.

Theorem 7.2 *In a perfect cryptographic system, the uncertainty on the keys $H(K)$ is at least as big as the uncertainty on the messages $H(M)$:*

$$H(K) \geq H(M).$$

PROOF

$$\begin{aligned} H(M) &= H(M|C) \\ &\leq H(M,K|C) \\ &= H(K|C) + H(M|K,C) \\ &= H(K|C) \\ &\leq H(K) \end{aligned}$$

The consequence of these two theorems is that in a perfect system keys must be complex enough, at least more complex than the messages themselves.

7.2.2 One Example: One-Time Pad

Let us now present a well-know example of a perfect cryptographic system: the "*one-time pad*", which is actually used by diplomats.

Without any loss of generally, we here consider the binary one-time pad, i.e. messages, cryptograms and keys are binary sequences ($\Sigma = \{0, 1\}$).

In this system, the key is a random sequence of n independent bits, $K = K_1K_2...K_n$:

$$p(K_i = 0) = p(K_i = 0|K_1, ..., K_{i-1}) = 0.5$$

where n is the size of the longest message to be transmitted.

The encryption is done simply by adding the symbols of the message and the symbols of the key[2]: $C_i = M_i + K_i$.

Example 7.4 (One Time Pad) Suppose the key is $k = 11010101010010101001$ and the message to be transmitted $m = 11110000111100001111$, then the encrypted message is $c = m + k = 00100101101110100110$.

Theorem 7.3 *"One-time pad" is a perfect cipher.*

PROOF For $1 < i \leq n$:

$$\begin{aligned} &p(C_i = 0|C_1, ..., C_{i-1}, M_1, ...M_n) \\ &\quad = \quad p(M_i = 0|C_1, ..., C_{i-1}, M_1, ...M_n) \cdot p(K_i = 0) \\ &\qquad +p(M_i = 1|C_1, ..., C_{i-1}, M_1, ...M_n) \cdot p(K_i = 1) \\ &\quad = \quad 0.5\,[p(M_i = 0|C_1, ..., C_{i-1}, M_1, ...M_n) + p(M_i = 1|C_1, ..., C_{i-1}, M_1, ...M_n)] \\ &\quad = \quad 0.5 \end{aligned}$$

[2]binary addition (a.k.a. "exclusive or" for readers familiar with computer sciences) is the usual modulo 2 addition, without carry: $0 + 1 = 1$ and, as usual, $0 + 0 = 0$, $1 + 1 = 0$, $1 + 0 = 1$.

Similarly, $p(C_1|M_1, ...M_n) = 0.5$, and $p(C_i|C_1, ..., C_{i-1}) = 0.5$ for all i, $1 \leq i \leq n$.
Thus, $P(C|M) = P(C)$, i.e. $I(C; M) = 0$. ■

7.2.3 Imperfect Secrecy and Unicity Distance

We have seen that for a cryptographic system to be perfect, the key must be complex enough. In practice, at least for wide range usage, this is not very convenient.

For a practical wide range system (e.g. security on the Internet) the key must be small (at least smaller than the messages) and to be used several times, i.e. the system has to be "imperfect" from a formal point of view.

What can we thus say about "imperfect" (but more convenient) systems?

To determine when a ciphering system that did not offer perfect secrecy could in principle be broken, Shannon introduced the so-called *key equivocation function* defined for integers by

$$a(n) = H(K|C_1...C_n).$$

It seem obvious that the more encrypted text has been seen, the less uncertainty remains of the key. More formally:

$$\lim_{n\to\infty} a(n) = 0$$

The *unicity distance* u is then defined as the smallest n such that $a(n) \approx 0$.

Definition 7.2 (unicity distance) The unicity distance of a cryptosystem is the smallest n such that

$$H(K|C_1...C_n) \approx 0$$

Thus, u is the least amount of ciphertext from which unauthorized persons are able to determine the secret key almost uniquely. Roughly speaking, the unicity distance is the least amount of ciphertext from which the ciphering system can be broken.

Let us now compute the unicity distance under certain circumstances.

Theorem 7.4 *If*

- *M and C are of the same length n and from the same alphabet Σ;*
- *encrypted messages have roughly maximal uncertainty: $H(C_n) \simeq n \cdot \log|\Sigma|$ (which is something every cryptographer tries to reach);*
- *the key and messages are independent: $H(M_n, K) = H(M_n) + H(K)$ (which is also very natural and usual).*

Then the unicity distance can be approximated by

$$u \simeq \frac{H(K)}{R(M) \cdot \log|\Sigma|} \tag{7.1}$$

where $R(M)$ is the redundancy of the unencrypted messages M, as defined in section 3.2.1 of chapter 3:

$$R(M) = 1 - \frac{H_\infty(M)}{\log|\Sigma|}.$$

PROOF Assume n to be large enough so that $H(M_n) \simeq n \cdot H_\infty(M)$, which is a sensible hypothesis (consider otherwise the maximum of such an n and the value of u obtained with the given formula).

$$\begin{aligned} H(K|C_n) &= \overbrace{H(M_nKC_n) - H(M_n|KC_n)}^{H(KC_n)} - H(C_n) \\ &= H(M_nKC_n) - H(C_n) \\ &= H(M_nK) - H(C_n) \\ &= H(M_n) + H(K) - H(C_n) \\ &= n \cdot H_\infty(M) + H(K) - n \cdot \log|\Sigma| \end{aligned}$$

Unicity distance u is defined by: $H(K|C_u) = 0$, i.e.

$$u\Big(H_\infty(M) - \log|\Sigma|\Big) + H(K) = 0$$

or:

$$\begin{aligned} u &= \frac{H(K)}{\log|\Sigma| - H_\infty(M)} \\ &= \frac{H(K)}{R(M) \cdot \log|\Sigma|} \end{aligned}$$

Example 7.5 (unicity distance) Let us consider English messages (made of the 27 letters alphabet, including whitespace) encrypted with a cipher using key of 20 independent letters. $H(K) = 20 \cdot \log(27)$.

Knowing that the entropy rate of English is roughly 2 bits per letter, the redundancy of messages is $R(M) = 1 - 2/\log(27) \simeq 0.58$ and the unicity distance of such a system

is:

$$\begin{aligned} u &= \frac{H(K)}{R \log |\Sigma|} \\ &= \frac{20 \cdot \log(27)}{\log(27) - 2} \simeq 35 \end{aligned}$$

i.e. cryptograms of about 35 characters will allow to determine the key almost uniquely!

Shannon was well aware that the formula (7.1) was valid in general and "*can be used to estimate equivocation characteristics and the unicity distance for the ordinary types of ciphers*". Indeed, cryptographers routinely use this formula to estimate the unicity distance of almost all ciphers.

Notice also that u is, in principle, the required amount of ciphertext to determine the key almost uniquely. However, finding K from $C_1, C_2, \ldots, C_u$ may very well be an intractable problem in practice. This formula only says that all the information is there, but does not say a word on how much difficult it might be to "extract" it. We will come to this aspect later on in section 7.3.

7.2.4 Increasing Unicity Distance: Homophonic Coding

It can be seen from (7.1) that a good way to increase the unicity distance (i.e. to tell less about the system) is to decrease the redundancy of the messages, i.e. to increase their entropy.

It is for instance a good idea to compress the messages before encrypting them. Indeed, in the best compression cases, $H(M_n) \simeq n \log |\Sigma|$ et thus $R(M) \simeq 0$, so $u \to \infty$.

Another possibility is to use an old cryptographic trick called "*homophonic substitution*". In this process, several different "homophones" are used to represent each symbol of the original alphabet; the more homophones for the most frequent symbols so that the homophones appear almost equally likely (whereas original symbols do not).

Example 7.6 In English, the most probable symbol is the whitespace, with a probability about .1859, the next most probable symbol is 'E', which has probability about .1031. The least likely is 'Z' with a probability about .0005.

If we want to convert such an English text using almost equally likely symbols, we need at least $1/0.005 = 2000$ symbols (so as to be able to have at least one for the 'Z'). Suppose we are thus using 2000 "homophones" to represent the 27 letters. The whitespace will be represented by any of the 372 ($\approx .1859 \times 2000$) symbols we choose for it, 'E' with any of the 206 ($\approx .1031 \times 2000$) other symbols reserved for it, etc., and 1 ($\approx .0005 \times 2000$) homophone symbol is used to represent the 'Z'. The choice of a substitute for an English letter is then made by a uniform random choice from the set of homophone substitutes for that letter. The successive choices are made independently. After such a conversion, each homophone symbol in the converted text is essentially equally likely to any of the other.

The decoding can be easily achieved by replacing each of the substitutes by the corresponding letter. There is no need to know in advance which substitutes were

randomly chosen in the pre-coding process.

Control Question 72

What is the unicity distance of a cryptosystem ciphering messages of 96 characters having an entropy rate of 3 bits per character with keys, the entropy of which is 33 bits.

Answer

$$u = \frac{33}{\log 96 - 3} = 9.2$$

(if needed: $R = 1 - \frac{3}{\log 96} = 54.4\%$)

Control Question 73

What is the unicity distance of a cryptosystem encoding binary messages which are 25% redundant, with uniformly distributed keys of 16 binary symbols?

Answer

$$u = \frac{16}{0.25 \cdot 1} = 64$$

Indeed

- entropy of uniformly distributed keys of 16 bits is... 16 bits!
- $\log |\Sigma| = \log 2 = 1$

Summary for Chapter 7

Perfect Secrecy: $I(C; M) = 0$

- for a system to be perfectly secret there must be at least as many keys as messages and $H(K)$ must be greater than (or equal to) $H(M)$.

One-Time Pad: for each encryption, a random key is chosen, whose length is equal to the message length and whose symbols are independent. The key is then simply added (symbol by symbol) to the message.

- One-Time Pad is a perfect cipher.

unicity distance: the minimum number of encrypted text that must be known to determine the key almost surely: $H(K|C_1...C_u) \simeq 0$.

- under certain general assumptions, the unicity distance can be approximated by

$$u \simeq \frac{H(K)}{R(M) \cdot \log |\Sigma|}$$

where $R(M)$ is the redundancy of the unencrypted messages M.

7.3 Practical Secrecy: Algorithmic Security

Learning Objectives for Section 7.3

After studying this section you should know:

1. how practical secrecy is achieved for unperfectly secure cryptosystems;
2. what 'difficult' means for a computer (algorithmic complexity);
3. what a 'one-way function' is;
4. how does DES work.

Up to this point, no particular attention has been paid to the computational power required to actually crack the system. The analysis of secrecy developed up to here applies independently of the time and computing power available for the attacks. Security against computationally unrestrained enemies is called *unconditional security* (or "*theoretical*" security as Shannon used to call it). As we have seen in theorems 7.1 and 7.2, achieving unconditional security usually requires enormous quantities of complex secret keys much more than what could be accepted in practice for wide scope cryptography applications. Most cryptographic systems used in practice thus do not rely not on the impossibility of being broken but rather on the difficulty of such a breaking. In this framework, the goal is to ensure security against unauthorized persons who have a limited time and computing power available for their attacks. This is called *computational security* (or "*practical*" security as Shannon used to call it). The point is to change lack of information (unconditional security) for difficulty to access to the information.

But what does it actually mean to be difficult? How could we measure the difficulty to crack a code? This is the aim of "algorithmic complexity".

7.3.1 Algorithmic Complexity

It is not the purpose of this section to provide a complete course on algorithmic complexity, but we would rather present the basic concepts so that the rest of the lecture regarding computational security can be understood well enough.

Algorithmic complexity aims at defining the complexity of *decision problems.* A decision problem is simply a yes/no question on a well defined input. For instance, given an integer number n (the "input"), is this number a prime number?

If the answer to the decision problem could be found by some algorithm (on a Turing machine), we call the decision problem "algorithmic".[3]

[3] In the general algorithmic complexity theory, algorithmic decision problems are called "Turing decidable problems", but this goes a bit beyond the scope of this chapter.

For algorithmic decision problems, the (time-)complexity[4] is defined as the smallest number of time steps (on a Turing machine) of the algorithms that can answer the question.[5]

For good fundamental reasons, this complexity is not expressed exactly, but only in the way it depends on the size of the input: a problem is said to be linear, quadratic, exponential, ... this mean that its complexity grows linearly, quadratically, exponentially, ... with the size of the input.

The complexity is thus expressed in terms of "big O" notation.

Definition 7.3 (Big O notation) For two functions f and g over the real numbers, g is said to be $\mathcal{O}(f)$ if and only if

$$\exists x_0 \in \mathbb{R},\ \exists c \in \mathbb{R},\ \forall x \geq x_0 \quad |g(x)| \leq c \cdot f(x)$$

Notice that if g is $\mathcal{O}(f)$ and f is $\mathcal{O}(h)$, g is also $\mathcal{O}(h)$. For complexity measure, we are looking for the "smallest and simplest" f such that g is $\mathcal{O}(f)$ (e.g. such that f is also $\mathcal{O}(|g|)$).

Example 7.7 (Big O notation) $3 \cdot n + \log n + 4$ is $\mathcal{O}(n)$. Notice this is also $\mathcal{O}(n + n^3)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n^2)$, ... which are not pertinent when used for complexity measure.

$5 \cdot x^2 - 12 \cdot x^7 + 5 \cdot x^3$ is $\mathcal{O}(x^7)$.

$1/x$ is $\mathcal{O}(1)$.

The complexity of a linear problem is $\mathcal{O}(n)$, where n is the size of the input.

A problem whose complexity is $\mathcal{O}(2^n)$ is an exponential problem.

Definition 7.4 (P and NP) P is the set of algorithmic decision problems, the complexity of which is polynomial.
NP is the set of algorithmic decision problems, such that if a possible solution is given, it is possible to verify this solution in a polynomial time. ♦

A classical pitfall is to think that NP means 'not-P' or 'non-P'. This is wrong for several reasons:

- P and NP are not complementary: P is actually totally included in NP;
- there are problems which are neither in P nor in NP.

What does the 'N' of NP means then? It stands for "Non-deterministic". NP problems are problems which are polynomial in a non-deterministic way: pick up a possible solution at random, then you can conclude (for that candidate solution only!), in polynomial time.

Clearly P $\subset$ NP, but it is for the moment still an open question whether NP $\subset$ P or not.

[4] only time-complexity is considered in this chapter.
[5] We do not go here into the details of problems and co-problems.

With respect to this question, there is a subset of NP which is of particular interest: the NP-Complete (or 'NP-C') problems.

Definition 7.5 (NP-Complete) A problem is said to be NP-Complete if it is

- in NP
- at least as difficult as any problem in NP.

This class is of particular importance because if someone manages to prove that one single NP-C problem is actually in P, then all NP is included in P!

There are finally a last class of problems (the "difficult" ones): NP-hard problems.

Definition 7.6 A problem[a] is said to be NP-hard if it at least as difficult as any problem in NP. ♦

[a] In its most general definition, the NP-hard class also includes problems which are not decision only problems.

NP-C and NP-hard problems are often confused. The difference between NP-C and NP-hard problems is that a NP-hard problem does not required to be in NP (either because you do not bother or you do not want to spend time to prove that or, more fundamentally, because it is a so difficult problem that even testing on single solution cannot be achieved in polynomial time).

Example 7.8 (NP-Complete problems)
Satisfiability (**SAT**) :
The input is a set of n boolean (i.e. true/false) variables $x_1, ..., x_n$. Decision: could

$$(x_{i_1} \vee x_{j_1} \vee x_{k_1}) \wedge (x_{i_2} \vee x_{j_2} \vee x_{k_2}) \wedge (x_{i_3} \vee x_{j_3} \vee x_{k_3}) \wedge ....$$

be satisfied, i.e. be true for some values of the variables x_i?

Traveling Salesman Problem (**TSP**):
The input is a graph (a set of nodes and arcs) G and a distance d. Decision: is there a circuit going through all nodes of G and whose length is below d?

We now have all the required ingredients to make a code difficult to crack: get inspiration from NP-hard problems. Let us now focus more precisely on the usage of difficult problems for cryptography: one-way functions and, latter on, trap-door functions.

7.3.2 One-Way Functions

Definition 7.7 (One-Way Function) A *one-way function** is a function that is easy to compute but (computationally) difficult to invert. ♦

How could a one-way function be useful for cryptography?

The key idea is that both the encoding $e(M, K) = C$ and the decoding $d(C, K) = M$ are easy to compute but that their inversion is difficult (even if $H(K|C, M) = 0$).

However, the most obvious application of one-way functions is certainly for password-based systems.

For each authorized user of the system, his password w is stored in the encrypted form $e(w)$, where e is a one-way function. When someone wants to use the system ("log in"), he presents a password $\widetilde{w}$ and the system computes (easy way) $e(\widetilde{w})$ and checks if it correspond to the stored information $e(w)$. If so, the user is granted access; if not, he is denied. The virtue of this system is that the stored encrypted passwords do not need to be kept secret[6]. If e is truly a one-way function, an attacker who somehow gets access to these encrypted passwords can not do anything with that as it is computationally infeasible for him to find a (pass)word x such that $e(x) = e(w)$.

Notice, interestingly, that this first example application of one-way functions, actually used in practice, provides authenticity rather than security, in the sense developed earlier in this chapter.

An example of one-way function is given in the next section.

7.3.3 DES

Data Encryption Standard (DES in short) is one instance of a computationally secure cryptographic system (or at least thought to be!) that uses one-way functions. Let us present here the basic idea of DES. We here only focus on the core of the system as the actual standard contains several other practical tricks.

DES is using a NP-Complete problem very similar to SAT for this: equation systems in GF(2).

Example 7.9 Deciding if the system

$$\begin{aligned} x_1x_4 + x_2x_3x_5 &= 1 \\ x_2x_3 + x_1x_3x_4 &= 1 \\ x_1x_3 + x_1x_2x_5 &= 1 \end{aligned}$$

has a solutions or not is NP-Complete with respect to the number of variables.

The fact that the solution

$$(x_1, x_2, x_3, x_4, x_5) = (1, 0, 1, 1, 0)$$

is here easy to find should not hide the fact that for a larger number a variables, finding a solution is indeed a difficult problem.

How is it used for a cryptographic system?

Choose two integers n and m, and a non-linear function f from $\mathrm{GF}(2)^m \times \mathrm{GF}(2)^n$ to $\mathrm{GF}(2)^n$:

$$f(x_1, ..., x_m, y_1, ..., y_n) = (p_1, ..., p_n)$$

[6] although there is also no reason to make it public!

Choose also a key K of $(d-1)m$ bits, and split it into $(d-1)$ parts of m bits each: $K = (K_1, ..., , K_{d-1})$.

Suppose that be the binary message M to be send has of $2n$ bits.[7] M is split into two parts of length n: $M = (M_0, M_1)$.

The encryption is then done iteratively in $d-1$ steps ($i = 2, ..., d$):

$$M_i = M_{i-2} + f(K_{i-1}, M_{i-1})$$

Finally, the cryptogram sent is

$$C = (M_{d-1}, M_d)$$

Decrypting is simply done the other way round ($i = d, d-1, ..., 2$):

$$M_{i-2} = M_i + f(K_{i-1}, M_{i-1})$$

Example 7.10 (DES) Let us consider the following non-linear function (with $m = 3$ and $n = 3$):

$$\begin{aligned} & f(x_1, x_2, x_3, y_1, y_2, y_3) \\ = \; & (x_1x_2y_1y_2, x_2x_3y_1y_3, (x_1 + x_2)y_1y_3) \end{aligned}$$

and let's choose a key $K = \texttt{101011}$ ($d = 3$):

$$K_1 = \texttt{101}, \quad K_2 = \texttt{011}$$

How will the message `101111` be encoded?

$$M = \texttt{101111} \Rightarrow M_0 = \texttt{101}, M_1 = \texttt{111}$$

Iterations:

$$\begin{aligned} M_2 &= M_0 + f(K_1, M_1) \\ &= (1,0,1) + f((1,0,1),(1,1,1)) \\ &= (1,0,1) + (0,0,1) = (1,0,0) \\ M_3 &= M_1 + f(K_2, M_2) \\ &= (1,1,1) + f((0,1,1),(1,0,0)) \\ &= (1,1,1) + (0,0,0) = (1,1,1) \\ C &= (M_2, M_3) = (1,0,0,1,1,1) \end{aligned}$$

So finally, `100111` is send.

Security of DES

The security of DES is based on a NP-Complete problem. As such, there are at least three possible sources of insecurity:

[7] otherwise pad and split the original message to have a smaller pieces of length $2n$.

- NP = P: if indeed someday it happens that polynomial solutions could be found for NP problems, then these "difficult" problems will no longer be difficult at all! However, this is nowadays very unlikely.
- The size of the key is not big enough (recall that complexity is growing with the size, and thus, only long enough inputs lead to computation time long enough to be unreached). Actually, ever since DES was proposed, it has a lot been criticized for its short 56 bits key size.
- But the most serious critic is certainly that, because the problem is NP, any possible solution can, by definition, be tested in polynomial time, i.e. if by chance the attacker guesses the right key, it is easy for him to check that it is the right key!

The main conclusion is that the security is not always guaranteed for all cases: it might, by chance, be easily cracked on some special cases. The only security comes from the low chance for the attacker to guess the key.

Summary for Chapter 7

Algorithmic Complexity: how does the time of an algorithm grow with respect to its input size.

P and NP: A problem is said to be in P if it can be solved by an algorithm, the complexity of which is polynomial.

A problem is said to be in NP if a proposed solution of it can be check in polynomial time (with respect to the size of this solution).

Pitfall: NP does **not** mean "not P" but rather "non-deterministic P".

One-Way function: a function that is easy to compute but difficult to invert.

DES: a cryptosystem based on the difficulty to solve non-linear boolean systems.

7.4 Public-Key Cryptography

Learning Objectives for Section 7.4

After studying this section you should know:

1. what public-key cryptography means and how is it possible;
2. what a "trapdoor function" is;
3. what the "Diffie-Lamport Distribution System" is and how it works;
4. how does RSA work and what is its security.

The main problem to be addressed in large scale cryptographic systems is: how to transmit the keys in a secure manner?

The intelligent idea of Diffie and Hellman is not to transmit the key at all, but to use public keys. Each pair of users can, using a generic system for the distribution of keys, have its own key for their communication. We present in this section two different schemes for public key distribution: Diffie-Hellman scheme and RSA.

The paper Diffie and Hellman published in 1976, had created a real shock in the cryptography community at the time. That paper suggested it is possible to build computationally secure ciphering systems without any secure channel for the exchange of the keys!

Indeed it was well known that public-key systems can actually not provide any unconditional security, since $H(K) = 0$ for such systems! The breakthrough came from their clever idea that if computational security as been decided to be used (as indeed in most practical applications), then the secure exchange of secret keys is no longer required.

This rather counter intuitive idea lies on the fundamental notions of *one-way functions*, already presented, and *trapdoor function*, to be introduced in a few sections. But before going on on this topic, we need a bit more of mathematics.

7.4.1 A bit of Mathematics

Modern computational cryptography is based on finite field algebra and more precisely on the "modulo p multiplication" (where p is a prime), i.e. the multiplicative group $\mathrm{GF}^*(p)$ of Galois field $\mathrm{GF}(p)$: 1, ..., $p-1$.

Since this group has $p-1$ elements, Euler-Fermat theorem ensures that:

$$n^{p-1} = 1 \mod p$$

for every n in $\mathrm{GF}^*(p)$.

Example 7.11 ($\mathbf{GF^*(5)}$) Let us consider $\mathrm{GF}^*(5) = \{1, 2, 3, 4\}$ (i.e. $p = 5$), where, for instance, $4 \cdot 3 = 2$, $2 \cdot 4 = 3$, $2 \cdot 3 = 1$. $(4 \cdot 3 = 12 = 2 \mod 5)$

Regarding Euler-Fermat theorem: e.g. for $n = 2$ we have: $2^4 = 16 = 1 \mod 5$.

Definition 7.8 (Primitive Root) An integer n is a primitive root of a prime p if and only if its modulo order is $p-1$, i.e.:

$n^i \neq 1 \mod p,\ 0 < i < p-1$

and $n^{p-1} = 1 \mod p$

Theorem 7.5 *For every prime number p, there exists at least one primitive root in $GF^*(p)$.*

Example 7.12 Let us consider $\mathrm{GF}^*(5)$ once again.

$n = 2 :\ 2^2 = 4,\ 2^3 = 3,\ 2^4 = 1$, thus 2 is a primitive root in $\mathrm{GF}^*(5)$.

$n = 4 :\ 4^2 = 1$, i.e. 4 is <u>not</u> a primitive root.

Discrete Exponentiation

Consider $\mathrm{GF}^*(p)$ for some prime p, and let a be a primitive root. By *discrete exponentiation to the base a* in $\mathrm{GF}^*(p)$ we mean the function $\exp_a : \mathrm{GF}^*(p) \to \mathrm{GF}^*(p)$ such that $\exp_a(n) = a^n$.

Since a is primitive root, the $p-1$ possible values of $\exp_a(x)$ (as n ranges over $\mathrm{GF}^*(p)$) are all distinct. Thus, its inverse function ${\exp_a}^{-1}$ exists. This function and is called the *discrete logarithm to the base a* and is denoted by Log_a.

Example 7.13 In $\mathrm{GF}^*(5)$, we have seen that 2 is a primitive root.

$\mathrm{Log}_2(3) = 3$ (in $\mathrm{GF}^*(5)$): indeed, as seen in example 7.12, $2^3 = 3 \mod 5$.

Notice that $\mathrm{Log}_a(a) = 1$ and that $\mathrm{Log}_a(1) = p - 1$ in $\mathrm{GF}^*(p)$.

Control Question 74

Which of the following numbers are primitive roots in $\mathrm{GF}^*(11)$: 2, 4, 5, 6, 9, 10?

Answer

2: yes; 4: no ($4^5 = 1$); 5: no ($5^5 = 1$); 6: yes; 9: no ($9^5 = 1$); 10: no ($10^2 = 1$).

Control Question 75

In $\mathrm{GF}^*(11)$, compute

- $\mathrm{Log}_7\, 2$
- $\mathrm{Log}_7\, 4$
- $\mathrm{Log}_7\, 5$
- $\mathrm{Log}_7\, 6$
- $\mathrm{Log}_7\, 10$

Answer

- $\mathrm{Log}_7\, 2 = 3$. Indeed, $7^3 = 7^2 \cdot 7 = 49 \cdot 7 = 5 \cdot 7 = 35 = 2 \mod 11$.
- $\mathrm{Log}_7\, 4 = 6$: $7^6 = (7^3)^2 = 2^2 = 4 \mod 11$.
- $\mathrm{Log}_7\, 5 = 2$: $7^2 = 5 \mod 11$.
- $\mathrm{Log}_7\, 6 = 7$: $7^7 = 7^6 \cdot 7 = 4 \cdot 7 = 28 = 6 \mod 11$.
- $\mathrm{Log}_7\, 10 = 5$: $7^5 = 7^3 \cdot 7^2 = 2 \cdot 5 = 10 \mod 11$.

Conjecture 7.1 (Diffie-Hellman-Pohlig) *The discrete exponentiation is a one-way function.*

First of all, discrete exponentiation is always easy to compute, requiring at most $2\log_2 n$ multiplications in $\mathrm{GF}^*(p)$ using the square-and-multiply algorithm.

On the other hand, the fastest known algorithm known today [7] for finding discrete logarithms is in $\mathcal{O}(\exp(n^{1/3}(\log n)^{2/3}))$.

However, there is no proof that there is no algorithm for computing the general discrete logarithm in a shorter time than the above.

The Diffie, Hellman and Pohlig conjectured that discrete exponentiation in $\mathrm{GF}^*(p)$ (when the base is a primitive root) is a one-way function, provided that p is a large number such that $p-1$ also has a large prime factor.

No proof of the conjecture has been given yet. But neither has an algorithm been found for efficiently computing the discrete logarithm. In other words, the historical evidence in favor of the conjecture has been increasing, but no theoretical evidence has yet been produced.

7.4.2 The Diffie-Hellman Public Key Distribution System

In their 1976 paper, Diffie and Hellman suggested an ingenious scheme for creating a common secret key between sender and receiver in a network without the need for a secure channel to exchange secret keys; their scheme relies on the one-way aspect of discrete exponentiation. Suppose $f(x) = a^x$ is truly a one-way function and is known to all users of the network.

Each person (say, user A) randomly chooses (in secret!) a *private* (or *secret*) key x_A and then computes her public key $y_A = a^{x_A}$, which is publicly published.

When another person (say users B) wish to communicate securely with A, each fetches the public number of the other, and uses this key to the power his/her own private key for the communication; i.e user A computes ${y_B}^{x_A}$ and user B computes ${y_A}^{x_B}$.

What is "magic" is that these two number are indeed the same: ${y_B}^{x_A} = (a^{x_B})^{x_A} = a^{x_A x_B} = (a^{x_A})^{x_B} = {y_A}^{x_B}$. This number $k_{AB} = a^{x_A x_B}$, that both users A and B can compute, is their "common secret", which they can safely use as their secret key for communication using a conventional secret-key cryptosystem.

What the Diffie-Hellman scheme provides is thus a public way to distribute secret keys.

If some unauthorized person wishes to crack the key, he should be able to take discrete logarithms of either y_A or y_B (e.g. $x_A = \mathrm{Log}_a\, y_A$) and then get the desired secret key as $K_{AB} = {y_B}^{x_A}$. But if the discrete exponentiation used is truly one-way, this attack is computationally infeasible.

Up to now (2003), nobody has produced an attack on the Diffie-Hellman public key-distribution scheme that is not computationally equivalent to computing the discrete logarithm. However, it has neither been proved that all attacks on this system are computationally equivalent to computing the discrete logarithm.

Example 7.14 (Diffie-Hellman public key) In a Diffie-Hellman scheme, with

p=127 and a=67, a user A chooses as private key $x_A = 111$. He then publishes his public key $y_A = 67^{111} = 102 \mod 127$.

Another user, B, chooses $x_B = 97$; thus $y_B = 67^{97} = 92 \mod 127$.

These two users can communicate using the key $k_{AB} = 92^{111} = 102^{97} = 77 \mod 127$.

Control Question 76

In a Diffie-Hellman public key scheme, with $p = 19$ and $a = 3$ (which is indeed a primitive root in GF*(19)),

- what is the public key corresponding to the private key 5?
- what key does a person whose private key is 7 use to communicate with a person whose public key is 14?

Same question with $p = 101$ and $a = 51$.

Answer

- 15: $3^5 = 15 \mod 19$
- 3: $14^7 = 3 \mod 19$

With $p = 101$ and $a = 51$:

- 60: $51^5 = 60 \mod 101$
- 6: $14^7 = 6 \mod 101$

7.4.3 Trapdoor Functions

Trapdoor functions, the second crucial aspect introduced by Diffie and Hellman for their public key cryptography framework, is more subtle and more difficult than the first one of one-way functions.

Definition 7.9 A *trapdoor function** is actually a *family* of bijective functions f_t, indexed by a parameter t (the "trapdoor" key), such that each function is a one-way function, but when t is known, ${f_t}^{-1}$ is also easy to compute. ♦

The cryptographic utility of a trapdoor function is the following: each user randomly (and secretly) chooses a trapdoor key, let us say t, and publishes f_t (but not t itself!). Usually f_t is taken in a family of functions so that only some parameters need to be published. These parameters are called the "public key".

If someone want to communicate a message M to the person whose published trapdoor function is f_t, he simply sends $f_t(M)$, which is easy to compute since f_t is one-way. To

get the proper message, the receiver computes ${f_t}^{-1}$ which is also easy to compute for him since he possesses the trapdoor key t. This computation is however difficult for any person who does not have the trapdoor key.

An example of trapdoor function is given in the next section.

7.4.4 RSA

The first trapdoor functions was made in 1978 by R. L. Rivest, A. Shamir and L. Adleman (RSA in short). The RSA trapdoor function is based on the supposed difficulty of factorizing integers.

In this framework, both the message and the cryptogram are represented as (huge!) integers.

Each user chooses two (large) prime numbers p and q (such that $p-1$ and $q-1$ also have large prime factors) so that for all possible message M, $M < pq$ (otherwise split M in several parts so that each part is less that pq and consider as each part as M in the following).

Let $n = pq$ and $m = (p-1)(q-1)$. The user chooses then $d < m$ which is prime with m (i.e. d and m do not have any divisor in common) and computes e such that $ed = 1 \mod m$. An algorithm that computes e knowing d and m is given in appendix at the end of this chapter.

The public key (to be published) is then (e, n) and the private key (to be kept secret) is (d, p, q, m).

The encryption function (which is public) is

$$C = M^e \mod n$$

and the decryption (which is secret) is

$$D = C^d \mod n$$

RSA framework works properly if $D = M$, i.e. $M^{ed} = M \mod n$.

This is indeed the case: since $ed = 1 \mod m$, there exist some $\lambda > 0$ such that $M^{ed} = M \cdot M^{\lambda m}$.

Recall furthermore that since p and q are prime numbers $M^{p-1} = 1 \mod p$ and $M^{q-1} = 1 \mod q$, thus

$$M^{ed} = M \cdot M^{\lambda m} = M \cdot M^{\lambda(p-1)(q-1)} = M \cdot \left(M^{p-1}\right)^{\lambda(q-1)} = M \cdot 1 \mod p$$

and similarly

$$M^{ed} = M \mod q.$$

A simple result from basic arithmetic is now required:

Theorem 7.6 *Given three integers m, n, p, if n and m do not have any divisor in common, and $x = 0 \mod m$ and $x = 0 \mod n$ then $x = 0 \mod (mn)$.*

The proof is really straightforward. So is the following corollary we are interested in:

Corollary 7.1 *If p and q are two prime numbers and if $x = y \mod p$ and $x = y \mod q$ then $x = y \mod (pq)$.*

Thus we have $M^{ed} = M \mod n$.

Example 7.15 (RSA) Suppose we want to use a RSA system to communicate, and we choose $p = 47$ and $q = 59$ (This is not really secure, but for the illustration purposes! In practice p and q should have more than 150 digits.)

Then we compute $n = pq = 2773$ and $m = (p-1)(q-1) = 2668$ and choose a d that is prime with m; e.g. $d = 157$.

Finally we compute e such that $157 \cdot e = 1 \mod 2668$ using Euclid's extended greatest common divisor algorithm: $e = 17$.

e and n are published: $(17, 2773)$, but the other numbers are keep secret.

Assume now someone wants to send us the message `ITS ALL GREEK TO ME`. By a convention agreed before (and which is public: Kerckhoffs' hypothesis), she transforms it into numbers:

$$09\ 20\ 19\ 00\ 01\ 12\ 12\ 00\ 07\ 18\ 05\ 05\ 11\ 00\ ...$$

Since M must be less than n, i.e $M < 2773$, she splits the above stream into integers of at most 4 digits (indeed the maximum code will then be 2626 corresponding to `ZZ`):

$$920,\ 1900,\ 112,\ 1200,\ 718,\ 505,\ 1100,\ ...$$

Then she computes $920^{17} \mod 2773$, $1900^{17} \mod 2773$, $112^{17} \mod 2773$, ... and sends us the corresponding integers; i.e

$$948,\ 2342,\ 1084,\ 1444,\ 2663,\ 2390,\ 778,\ ...$$

This message is decrypted using our own private key:

$$948^{157} = 920,\ 2342^{157} = 1900,\ ...$$

and end the decoding by applying the convention back: $920, 1900, ... = 09, 20, 19, 00, ... = \text{ITS}\ ...$

Now you may wonder how did our correspondent compute $1900^{17} \mod 2773$, or how did we compute $2342^{157} \mod 2773$?...

This is done by the "square and multiply" method and keeping in mind that

$$a \cdot b = \alpha \cdot \beta \mod n$$

for any $a = \alpha \mod n$ and $b = \beta \mod n$

For instance

$$\begin{aligned} 2342^{157} &= (2342^{128}) \cdot (2342^{16}) \cdot (2342^{8}) \cdot (2342^{4}) \cdot 2342 \\ &= 1428 \cdot 239 \cdot 284 \cdot 900 \cdot 2342 = 1900 \mod 2773 \end{aligned}$$

since

$$\begin{array}{lcllll} 2342^2 & = & 431^2 & = & 2743 & \mod 2773 \\ 2342^4 & = & 2743^2 = 30^2 & = & 900 & \mod 2773 \\ 2342^8 & = & 900^2 & = & 284 & \mod 2773 \\ 2342^{16} & = & 284^2 & = & 239 & \mod 2773 \end{array}$$

and

$$2342^{128} = 1428 \mod 2773$$

Control Question 77

Consider a very simple RSA set up where two people have the following parameters:

	p	q	d	e
A	3	19	5	29
B	5	11	7	23

1. What is the public key of A?
2. What is the public key of B?
3. How does A send the message '43' to B?
4. How does B send the message '43' to A?

Answer

1. $(29, 57)$
2. $(23, 55)$
3. 32: $43^{23} = 32 \mod 55$
4. 25: $43^{29} = 25 \mod 57$

Security of RSA

Breaking RSA by finding $m = (p-1)(q-1)$ is computationally equivalent to factoring n. In fact, all attacks thus far proposed against RSA have turned out to be computationally equivalent to factoring n, but no proof has been forthcoming that this must be the case.

Furthermore, there is no hard evidence that factoring of integers is inherently difficult, but there is much historical evidence in support of this second conjecture. Anyone who has tried to improve upon known factoring algorithms soon convinces himself that this is a very hard problem.

In 1999, the factorization of the 155 digit (512 bit) RSA Challenge Number was completed. It required 3.7 (real life) months and 35.7 CPU-years in total. This CPU-effort was estimated to be equivalent to approximately 8000 MIPS years.[8]

[8] see `http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa155.html` for further details.

Up to now (2003), the RSA system is considered safe from attacks made by factoring, at least with keys of more than 1024 bits... ...until revolutionary advances in factoring algorithms have been made! [which is unlikely today]

Summary for Chapter 7

primitive root: a number n smaller than a prime p is said to be a primitive root in $\mathrm{GF}^*(p)$ if and only if the only power $1 < i < p$ such that $n^i = 1$ is $p-1$.

discrete logarithm: an integer n is the discrete logarithm to the base a of another integer m in $\mathrm{GF}^*(p)$ if $a^n = m \mod p$ (where p is a prime number and a a primitive root in mGF).

Diffie-Hellman Public Key Distribution Scheme: being given a prime p and a primitive root a in $\mathrm{GF}^*(p)$, each user chooses a private key x and published his public key $y = a^x \mod p$.

When two users wish to communicate, each one uses the key consisting of the public key of the other to the power his own private key:

$$k_{AB} = {y_B}^{x_A} = {y_A}^{x_B}$$

trapdoor functions: a family of one-way functions depending on a parameter, such that when this parameter is know, the inverse is no longer hard to compute.

RSA: Each user chooses two prime numbers p and q and a number $d < (p-1)(q-1)$ which has no common divisor with $(p-1)(q-1)$. The public key is then (e, pq) where e is such that $ed = 1 \mod (p-1)(q-1)$, and the private key is (d, p, q).

A message M (which is a integer less than pq) is encrypted using public keys by $C = M^e \mod pq$. The decrypting is done using private keys: $D = C^d \mod pq$.

7.5 Authentication

Learning Objectives for Section 7.5

After studying this section you should know:

1. Why authentication and security are theoretically incompatible;
2. how to ensure authentication in practice;
3. what Diffie-Hellman and RSA authentication scheme consist of.

In this last section, we now want to address the second aspect of cryptography: authentication, i.e. ways to ensure that the message as been *sent* by an authorized person. In other words we wonder here whether the received cryptogram C is a valid (legal) one or if it has been faked by an unauthorized person.

Authentication and Security

Authentication was for long not easy to distinguish clearly from security. In fact, cryptographers have discovered only recently that these two goals are indeed quite independent, and even incompatible from a pure theoretical point of view.

This result is due to the following theorem.

Theorem 7.7 *The probability P_I that a cryptogram is falsified (i.e. to find a cryptogram that is accepted although it has not been emitted by an authorized person) is bounded by:*

$$P_I \geq 2^{-I(C;K)}$$

where C is the random variable representing possible cryptograms and K the possible keys.
This bound is tight and can be reached in special cases.

PROOF Let $\phi(C, K)$ be the authentication function:

$$\phi(C,K) = \begin{cases} 1 & \text{if } \exists M \mid C = e(M,K) \\ 0 & \text{otherwise} \end{cases}$$

The probability that a cryptogram C is accepted as correct is:

$$P(\text{acc}(C)|C) = \sum_K \phi(C,K)P(K)$$

thus, the probability to have a falsification of authenticity is:

$$\begin{aligned} P_I = P(\text{acc}(C)) &= \sum_C P(C, \text{acc}(C)) \\ &= \sum_C P(C)P(\text{acc}(C)|C) \\ &= \sum_{C,K} \phi(C,K)P(C)P(K) \\ &= \sum_{C,K,P(C,K)\neq 0} P(C)P(K) \\ &= \sum_{C,K} P(C,K)\frac{P(C)P(K)}{P(C,K)} \\ &= E\left[\frac{P(C)P(K)}{P(C,K)}\right] \end{aligned}$$

i.e.

$$\log(P_I) = \log E\left[\frac{P(C)P(K)}{P(C,K)}\right]$$

using Jensen inequality we have

$$\begin{aligned}\log E\left[\frac{P(C)P(K)}{P(C,K)}\right] &\geq E\left[\log\frac{P(C)P(K)}{P(C,K)}\right]\\ &= -I(C;K)\end{aligned}$$

Thus

$$P_I \geq 2^{-I(C;K)}$$

with equality if and only if $\frac{P(C)P(K)}{P(C,K)}$ is constant for all (C,K) (such that $P(C,K)>0$), e.g. C and K are independent. ■

Thus, to guarantee authenticity, i.e. to have small infraction probability, the mutual information between cryptograms and keys must be big! However, to ensure perfect confidentiality, we have seen in the former sections that $I(C;M)=0$ is required!

Thus, from a strict information content point of view, authentication and confidentiality appear to be somehow incompatible. Indeed,

$$I(C;K) = I(C;M) + H(K) - H(M) - H(K|M,C)$$

thus $I(C;M)=0$ (and with the sensible assumption that $H(K|M,C)=0$) implies $I(C;K)=H(K)-H(M)$, which implies $P_I \geq 2^{H(M)-H(K)} > 0$; i.e. a perfect cryptosystem should have **very** complex keys ($H(K) \gg H(M)$) to also ensure authentication.

One solution although, is to have $I(C;K) \gg 0$ (to ensure authenticity) but ensure confidentiality by algorithmic security. This is one of the reasons why cryptographic systems based on algorithmic complexity have become popular for authentication. Let us see now how this can be done.

The basic idea is to use a *digital signature* to messages. Such a signature will ensure that the sender of the message is actually who is claimed to be, and conversely in case of disagreement, the sender whose message has been signed cannot deny having send it, since he is the only one to be able to produce this signature.

7.5.1 Diffie-Lamport Authentication

In the Diffie-Lamport authentication framework, each user chooses

- $2\,n$ secret keys: $k_1, ..., k_n$ and $\kappa_1, ..., \kappa_n$
- $2\,n$ sequences $s_1, ..., s_n$ et $t_1, ..., t_n$

and then produces the parameters: $\lambda_i = e(s_i, k_i)$ and $\mu_i = e(t_i, \kappa_i)$.

Then he publishes (i.e. make publicly available) $\lambda_i, \mu_i, s_i, t_i$, for $i = 1...n$.

To sign the the binary message M of length n, $M = m_1, ..., m_n$ he uses $\sigma = \sigma_1, ..., \sigma_n$, where:

$$\sigma_i = \begin{cases} k_i & \text{if } m_i = \mathtt{0} \\ \kappa_i & \text{if } m_i = \mathtt{1} \end{cases}$$

Notice that this is a huge signature since k_i and κ_i are not bits but keys, i.e. sequences of k bits, where k is the size of the key required by the encryption algorithm.

When the receiver receives the message and its signature, he can check that the message has been sent by the right person by doing:

- if $m_i = \texttt{0}$, $e(s_i, \sigma_i) = \lambda_i$
- if $m_i = \texttt{1}$, $e(t_i, \sigma_i) = \mu_i$

Such an authentication scheme presents however several drawbacks:

- lot of material needs to be communicated in advance;
- a message of n bits requires a signature of $k \cdot n$ bits!

7.5.2 Authentication with RSA

Diffie and Hellman also showed in 1976 that, provided the domain and the range of the trapdoor function f_t coincide for every t, f_t can also be used to ensure the authenticity of messages with digital signatures. Provided that legitimate messages have sufficient internal structure to be reliably distinguished from random sequences (which is again against what is required for security!), if user A wishes to sign a message M so that it is unmistakable that it came from him, user A applies to M the decryption algorithm with his private key to compute the signed message $M' = d(M, \tilde{k}_A)$. Any other user, say user B, who obtains M' can use the encryption algorithm with the public key of A k_A to compute $e(M', k_A) = M$. However, only user A knows how to write the meaningful message M as the random-looking string M'; and it is computationally difficult for any other user to find M' such that $e(M', k_A) = M$.

Of course, this scheme does not provide any secrecy. If secrecy is also desired, user A could send the signed message M' using usual encryption method.

Let's illustrate this general scheme with RSA.

User A sends M to B using the cryptogram $C = e_B(M)$, and signs it with $S(M) = e_B(d_A(M))$

B can verify authenticity by doing: $e_A(d_B(S(M)))$ which must be equal to M.

Notice that this presupposes that $d_A(M)$ is in the domain of e_B, i.e. in the case of RSA that $d_A(M) < n_B$. There are several practical tricks to do so, among which the easiest is to split M into smaller pieces so that $d_A(M)$ is actually smaller that n_B.

Example 7.16 (RSA Authentication) To continue our last example (example 7.15), where $p = 47$, $q = 59$, $d = 157$ and $e = 17$, how will we sign the message '`OK`'?

Using the same convention as before, '`OK`' corresponds to the integer 1511. The signature is then $d(1511) = 1511^{157} = 1657 \mod 2773$.

If we send this message to someone whose public key is $(725, 2881)$ the encrypted message will be $1511^{725} = 1369 \mod 2881$ and the encrypted signature $1657^{725} = 2304 \mod 2881$; i.e. we send $(1369, 2304)$.

The receiver decodes the message using her private key $d = 65$: $M = 1369^{65} = 1511 \mod 2881$ and checks the signature: $S = 2304^{65} = 1657 \mod 2881$, $e(S) = 1657^{17} = 1511 \mod 2773$. $e(S) = M$: the message is valid.

Control Question 78

Consider again the very simple RSA set up where two people have the following parameters:

	p	q	d	e
A	3	19	5	29
B	5	11	7	23

1. What does A send to B to authenticate message "43"?
2. B receives the message "22" from A, signed "41". Is it really A who send it?

Answer

1. A sends "07". The signature is "28" ($43^5 = 28 \mod 57$) and is encrypted into "07" ($28^{23} = 7 \mod 55$).
2. No, this message doesn't come from A.

   The ciphered text "22" corresponds the message "33" ($22^7 = 33 \mod 55$). The decoded signature is "46" ($41^7 = 46 \mod 55$), which after public encryption for A should be "33". But $46^{29}31 \mod 57 \neq 33$.

   The correct signature for "33" is "42".

Signature security

A cannot deny having send m : only d_A can produce S.

B (or anyone else but A) cannot change m for m': $S(m') \neq S(m)$ and computing $S(m')$ is not possible without d_A.

In practice however, there are several drawbacks to this signature scheme. For instance, the sender may deliberately publish his private key making then doubtful all digital signature attributed to him, or he can also deliberately "loose" his private key so that the messages he sent become unverifiable. To make up for the later, trusted organisms where the keys should be recorded before transactions, could play the role of "private key banks" .

7.5.3 Shared secrets

Let us finished with a quite different authentication scheme where the access to some critical resources or information must be shared by several persons. The idea in such system is that several persons together can reconstruct the complete key, but none can do it alone. This is the "*shared secret*" method.

Examples of such situation include, e.g. opening a safe with two keys, missile launch requiring 3 authorizations.

Suppose that we have n authorized persons ("key holders") and that k parts of the key are enough to "open the door", i.e. access the secret S. This means:

$$H(S|p_{i_1}, ..., p_{i_k}) = 0$$

for any subset $\{i_1, ..., i_k\}$ of $\{1, ..., n\}$, where $p_1, ..., p_n$ are the n parts of the key. However, less that k parts are not enough to get any information about this secret:

$$H(S|p_{i_1}, ..., p_{i_{k-1}}) = H(S)$$

To do so, let's choose for every secret a polynomial of order $k-1$ whose lower coefficient is S:

$$P(X) = S + a_1X + a_2X^2 + ... + a_{k-1}X^{k-1}$$

The other coefficients a_i are chosen at random, and are different from one secret to the other.

The authorized user i received the part of the secret as the value of the polynomial for the value i:

$$p_i = p(i),$$

This fulfills the above conditions: k users can reconstruct the polynomial by interpolation and thus get S, but $k-1$ (or less) cannot.

Example 7.17 Secret $S = 0105$, shared by $n = 5$ persons among which any two of them can access the secret ($k = 2$)

$$p(x) = 105 + 50x$$

Thus

$$p_1 = 155, p_2 = 205, p_3 = 255, p_4 = 305, p_5 = 355$$

Reconstruction by 2 participants (e.g. 2 and 5):

$$\begin{aligned} p(2) = S + a_1 \cdot 2 &= 205 \\ p(5) = S + a_1 \cdot 5 &= 355 \end{aligned}$$

$$\Rightarrow 3S + a_1 \cdot (10 - 10) = 1025 - 710$$

$$S = 105$$

But the reconstruction by only 1 participant is not feasible.

Secret sharing can usefully be used to create "access structures" to the secret: there are less n users and some of there receive more parts than the other.

For example, imagine that opening a bank safe requires 1 director, or 2 authorized representatives, or 1 authorized representative and 2 cashiers, or 5 cashiers.

For instance, with $k = 10$

- the bank director receives 10 parts,
- each authorized representatives 6 parts,

- and each cashiers 2 parts only.

Thus

- the director alone has the required 10 part to open the safe,
- 2 authorized representatives have 12 parts,
- 1 representatives and 2 cashiers 10 parts,
- and 5 cashiers: 10 parts.

The only problem with such solution is that for complex situations the number of parts may be large.

Summary for Chapter 7

- Authentication (to ensure that the message as been *sent* by an authorized person) and secrecy (to ensure that the message is *received* by authorized persons) are somehow theoretically incompatible, since the former requires $I(C;K)$ as large as possible and the latter $I(C;M)$ as small as possible.
- $P_I \geq 2^{-I(C;K)}$

Diffie-Lamport Authentication scheme: can be used to sign binary messages. Choose $2n$ keys and $2n$ sequences, publish the encryption of the latter by the former and sign sending one or the other keys depending on the message bits.

RSA Authentication scheme: The signature is the message to the power the private key. Send it encrypted using addressee's public key.

Shared secrets: The access to one common secret is spread among several "key holders" using a polynomial.

Summary for Chapter 7

- cryptography aims at either transmitting messages securely (only authorized persons can read it) or authenticate messages (no unauthorized persons could have send it).
- To do so, the clear messages M are encoded using a key K and a deterministic function: $C = e(M, K)$.
- Encrypted messages can be decoded deterministically using the decoding function d and the same key K, so that $d(e(M, K), K) = M$.

Perfect Secrecy: $I(C; M) = 0$

- for a system to be perfectly secret there must be at least as many keys as messages and $H(K)$ must be greater than (or equal to) $H(M)$.

One-Time Pad: for each encryption, a random key is chosen, whose length is equal to the message length and whose symbols are independent. The key is then simply added (symbol by symbol) to the message.

One-Time Pad is a perfect cipher.

unicity distance: the minimum number of encrypted text that must be known (in unperfect cryptosystems) to determine the key almost surely: $H(K|C_1...C_u) \simeq 0$.

Under certain general assumptions, the unicity distance can be approximated by

$$u \simeq \frac{H(K)}{R(M) \cdot \log |\Sigma|}$$

where $R(M)$ is the redundancy of the unencrypted messages M.

One-Way function: a function that is easy to compute but difficult to invert.

DES: a cryptosystem based on the difficulty to solve non-linear boolean systems.

discrete logarithm: an integer n is the discrete logarithm to the base a of another integer m in $\mathrm{GF}^*(p)$ if $a^n = m \mod p$ (where p is a prime number and a a primitive root in mGF).

Diffie-Hellman Public Key Distribution Scheme: being given a prime p and a primitive root a in $\mathrm{GF}^*(p)$, each user chooses a private key x and published his public key $y = a^x \mod p$.

When two users wish to communicate, each one uses the key consisting of the public key of the other to the power his own private key:

$$k_{AB} = {y_B}^{x_A} = {y_A}^{x_B}$$

trapdoor functions: a family of one-way functions depending on a parameter, such that when this parameter is know, the inverse is no longer hard to compute.

RSA: Each user chooses two prime numbers p and q and a number $d < (p-1)(q-1)$

which has no common divisor with $(p-1)(q-1)$. The public key is then (e, pq) where e is such that $ed = 1 \mod (p-1)(q-1)$, and the private key is (d, p, q).

A message M (which is a integer less than pq) is encrypted using public keys by $C = M^e \mod pq$. The decrypting is done using private keys: $D = C^d \mod pq$.

- Authentication (to ensure that the message as been *sent* by an authorized person) and secrecy (to ensure that the message is *received* by authorized persons) are somehow theoretically incompatible, since the former requires $I(C; K)$ as large as possible and the latter $I(C; M)$ as small as possible.

Diffie-Lamport Authentication scheme: can be used to sign binary messages. Choose $2n$ keys and $2n$ sequences, publish the encryption of the latter by the former and sign sending one or the other keys depending on the message bits.

RSA Authentication scheme: The signature is the message to the power the private key. Send it encrypted using addressee's public key.

Historical Notes and Bibliography

Secrecy of messages has for very long be a subject of study. It is indeed claim to date back to ancient Egypt (1900 BC) or ancient China. In Europe, although Greeks and Romans (e.g. 'Caesar cipher') already used ciphers, cryptography and cryptanalysis really started in the second half of the thirteen century and developed more seriously from the fifteen century.

Around 1560, the French diplomat Blaise de Vigenère (1523-1596) developed his cryptosystem from the work of several of his predecessors: Alberti (1404-1472), Trithème (1462-1516) and Porta (1535-1615). Vigenère cipher remained unbreakable for 300 years.

The hypothesis that the security of the cipher should reside entirely in the secret key was first made in 1883 by Auguste Kerckhoffs (1835-1903); and cryptographic history has demonstrated his wisdom. A determined enemy is generally able to obtain a complete "blueprint" of the enciphering and deciphering machines, either by clever deduction or by outright stealing or by measures in-between these extremes.

The first really scientific treatment of secrecy has only been provided by C. Shannon in 1949 [11]. Shannon's theory of secrecy is in fact a straightforward application of the information theory that he had formulated one year before. The ingenuity of the 1949 paper lies not in the methods used therein but rather in the new way of viewing and the intelligent formulation that Shannon made of the problem of secrecy.

Although Shannon gave his theory of secrecy in 1949, it was not until 1984 that Simmons gave an analogous theory of authenticity, illustrating how more difficult and subtle authentication is.

The foundations of cryptographic thinking were once again shook in 1976, when two Stanford University researchers, Whitfield Diffie and Martin E. Hellman, published their paper entitled "*New Directions in Cryptography*". Diffie and Hellman suggested that it is possible to have computationally secure cryptographic systems that required no secure channel for the exchange of secret keys. Ralph Merkle, then a graduate

student at the Berkeley University, independently formulated the basic ideas of such "public-key cryptography" and submitted a paper thereon at almost the same time as Diffie and Hellman, but his paper was published almost two years later than theirs and unfortunately lost the due credit for his discovery.

The fundamental contribution of the Diffie-Hellman paper consisted in the two crucial definitions, of a one-way function (which they borrowed from R. M. Needham's work on computer passwords) and of a trapdoor function (which was completely new), together with suggestions as to how such functions could eliminate the need for the exchange of secret keys in computationally-secure cryptographic systems.

Although Diffie and Hellman shrewdly defined trapdoor functions in their 1976 paper and clearly pointed out the cryptographic potential of such functions, the first proposal of such a function was made only two years latter, in 1978 by the M.I.T. researchers R. L. Rivest, A. Shamir and L. Adleman (thus RSA!).

In the meantime, the basis of DES (1977) came out from the IBM Lucifer cryptosystem (first published by Feistel in 1973!). However, whereas the Lucifer scheme used a key of 128 bits, the US National Bureau of Standard (now known as National Institute of Standards and Technology) who published the DES retains only 56 bits for the key. Ever since DES was first proposed, it has a lot been criticized for its short key size.

In 1998, the EFF (Electronic Frontier Foundation has built a dedicated machine, Deep Crack, in order to show to the world that DES is not (or at least no more) a secure algorithm. Deep Crack, which costed $200'000 and was built with 1536 dedicated chips, was able to recover a 56 bit key, using exhaustive search, in 4 days in average, checking 92 billions of keys each second.[9]

Later on (January 18, 1999), with the help of distributed.net, an organization specialized in collecting and managing computer's idle time, they broke a DES key in 22 hours and 15 minutes! More than 100'000 computers (from the slowest PC to the most powerful multiprocessors machines) have received and done a little part of the work; this allowed a rate of 250,000,000,000 keys being checked every second. [10]

In November 2002, AES (Advanced Encryption Standard), the successor to DES was published. AES uses another type of algorithm (Rijndael algorithm, invented by Joan Daemen and Vincent Rijmen) and supports key sizes of 128 bits, 192 bits, and 256 bits, which nowadays seems more than enough.

OutLook

D. Welsh, "Codes and Cryptography", Oxford University Press, 1988.

`http://www.ars-cryptographica.com/`

Appendum: Solving $e \cdot d = 1 \mod m$

Finding e and k such that $e \cdot d - k \cdot m = 1$ can be done using Euclid's greatest common divisor algorithm (since the greatest common divisor of d and m is precisely 1) .

[9] This part is borrowed from `http://lasecwww.epfl.ch/memo_des.shtml`

[10] for more details, see http://www.rsasecurity.com/rsalabs/challenges/des3/index.html

Let $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{t}$ be vectors of $\mathbb{Q}^2$ (i.e. couples of rational numbers).

The initialization step of the algorithm consist of $\mathbf{u} = (0, m)$, $\mathbf{v} = (1, d)$.

The stop condition is that the second component v_2 of $\mathbf{v}$ equals 1. In this case the first component is $v_1 = e$, i.e. at the end of the algorithm $\mathbf{v} = (e, 1)$.

After the initialization step, the algorithm loops until the stop condition is fulfilled:
$\mathbf{t} \leftarrow \mathbf{u} - r\,\mathbf{v}$,
$\mathbf{u} \leftarrow \mathbf{v}$,
$\mathbf{v} \leftarrow \mathbf{t}$
where $r = \frac{u_2}{v_2}$

Example 7.18 Let us find e such that $7e = 1 \mod 60$, i.e. $d = 7$ and $m = 60$:

$\mathbf{u}$	$\mathbf{v}$	r	$\mathbf{t}$
$(0, 60)$	$(1, 7)$	$\frac{60}{7} = 8$	$(-8, 4)$
$(1, 7)$	$(-8, 4)$	$\frac{7}{4} = 1$	$(9, 3)$
$(-8, 4)$	$(9, 3)$	$\frac{4}{3} = 1$	$(-17, 1)$
$(9, 3)$	$(-17, 1)$	(stop)	

thus $e = -17 \mod 60$, i.e.: $e = 43$.

Bibliography

[1] R. Ash. *Information Theory*. Wiley, 1965.

[2] R. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley, 1983.

[3] P. Elias. Coding for noisy channels. In *IRE Conv. Rec.*, volume 4, pages 37–47, 1955.

[4] A. Feinstein. A new basic theorem of information theory. *IRE Trans. Information Theory*, IT-4:2–22, 1954.

[5] R. G. Gallager. A simple derivation of the coding theorem and some applications. *IEEE Trans. Information Theory*, IT-11:3–18, 1965.

[6] R. G. Gallager. *Information Theory and Reliable Communication*. Wiley, 1968.

[7] D.M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM Journal of Computing*, 1(6):124–138, 1993.

[8] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 1983.

[9] David Salomon. *Data Compression – The complete reference*. Springer, 1998.

[10] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. Journal*, 27:379–423, 623–656, 1948.

[11] C. E. Shannon. Communication theory of secrecy systems. *Bell Sys. Tech. Journal*, 28:656–715, 1949.

[12] J. H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1982.

[13] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[14] J. M. Wozencraft and B. Reiffen. *Sequential Decoding*. MIT Press, 1961.

Glossary

BSC: Binary Symmetric Channel. 150

BSS: Binary Symmetric Source: a memoryless binary source with $P(0) = P(1) = \frac{1}{2}$. 165

DMC: Discrete Memoryless Channel, see definition page 149. 149

Hamming distance: is the number of coordinates in which two vectors differ. 199

Huffman code: an efficient coding framework for memoryless information sources. 104

Minimum Distance: the minimum (non null) Hamming distance between any two (different) codewords. 203

arity: size of the alphabet. For a n-ary tree: n, the number of child nodes of each node. 82

block-code: a non-empty set of words of the same length, considered as "row vectors". 199

channel capacity: the maximum average amount of information the output of the channel can bring on the input. 152

codeword: a non empty sequence of symbols from the coding alphabet. 82

communication channel: formalization of what happens to the the transmitted message between their emission and their reception. 149

complete code: a prefix-free code, the coding tree of which does not have unused leaf. 89

cyclic code: a linear code, every shift of the codewords of which is also a codeword.. 225

expected code length: the expected value of the length of the codewords. 95

generator matrix: One matrix the rows of which are linearly independent codeword. Such a matrix is used to encode the messages.. 211

information source: generator of sequences of symbols. 82

instantaneously decodable: a code is said to be instantaneously decodable if each codeword in any string of codewords can be decoded as soon as its end is reached. 85

lattice: The unfolding in time of all the paths in the state diagram of a convolutional code corresponding to all the possible the encodings of messages of different lengths. 240

linear code: a block-code with vector space structure. 207

message: sequence of symbols. 82, 149

minimum distance decoding: decoding framework in which each received word is decoded into the closest codeword. 202

non-singular codes: codes for which different source symbols maps to different codewords. 83

null codeword: The codeword made only of zeros. 200

one-way function: a function that is easy to compute but difficult to computationally invert. 266

stationary sources: information sources, the statistics of which do not depend on time. 83

syndrome: product of the received word by a verification matrix. 218

transmission probabilities: conditional probability distributions of the output of a channel knowing the input. 149

trapdoor function: family of one-way functions depending on a parameter, such that, when the parameter is known, the inverse is no longer hard to compute.. 273

weight: the number of non-zero symbols. 199

without feedback: a channel is said to be used without feedback when the probability distribution of the inputs does not depend on the output.. 151

Index