Information Theory/ Data Compression
Ma 4211

Jürgen Bierbrauer

February 28, 2007

Contents

Chapter 1

Shannon entropy and the basics of information theory

1.1 Definition. *Let Ω be a finite set. For every $x \in \Omega$ let a nonnegative number $p(x)$ be given such that*

$$\sum_{x\in\Omega} p(x) = 1.$$

We see p as a mapping with domain Ω and real values. Then (Ω, p) is a (finite) **probability space** *or* **sample space,** *$p(x)$ is the* **probability** *of x.*

Define the probability of a subset $S \subseteq \Omega$ as $p(S) = \sum_{s\in S} p(s)$. Such a (finite) sample space is defined by a probability distribution, non-negative numbers p_i such that $\sum_i p_i = 1$. The entropy is a function, which measures the amount of information contained in a sample space.

1.2 Definition. *Let $p_i,\ i = 1, \dots, n$ be a probability distribution.*
The **Shannon entropy** *H is defined by*

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

Here, as always in information theory, the logarithm is taken base 2. We put $0 \cdot log(0) = 0$. Imagine a source, which at every unit of time emits one of n symbols, called letters, according to the given probability distribution. That is, the probability for letter i to be chosen is p_i. We claim that the entropy function is a measure for the average amount of information per letter generated by this source. The unit of information is the **bit,** the amount of information in the choice between two equiprobable events: $H(\frac{1}{2}, \frac{1}{2}) = 1$.

1.3 Theorem. *Let* $H_n : \mathbb{R}^n_+ \longrightarrow \mathbb{R}_+,\ n = 2, 3, \ldots$ *be a family of functions satisfying the following properties:*

- *Each* H_n *is continuous.*
- $H_2(1/2, 1/2) = 1$ *(one bit).*
- *There is a continuous monotonely increasing function* $A(x) : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ *such that* $A(n) = H_n(1/n, \ldots, 1/n)$.
- $H_n(p_1, p_2, \ldots, p_n) = H_2(p_1, 1 - p_1)+$ $+(1 - p_1) \cdot H_{n-1}(\ p_2/(1 - p_1)\ , \ldots,\ p_n/(1 - p_1))$. *Here the numbers* p_i *describe a probability distribution.*

Then $H_n(p_1, \ldots, p_n)$ *is the entropy function.*

Proof. At first we show that $A(s^m) = A(s^{m-1}) + A(s)$ for natural numbers s, m. and conclude $A(s^m) = mA(s)$.

Consider s^m equiprobable choices. Divide them into blocks of s each. The basic property states $A(s^m) = A(s^{m-1}) + A(s)$. By induction we obtain $A(s^m) = mA(s)$.

Given natural numbers s, t, n define m such that

$$s^m \leq t^n < s^{m+1}.$$

With this notation we want to show

$$\frac{m}{n} \leq \frac{A(t)}{A(s)} < \frac{m+1}{n}$$

and also

$$\frac{m}{n} \leq \frac{log(t)}{log(s)} < \frac{m+1}{n}$$

The monotonicity of A shows $mA(s) \leq nA(t) < (m+1)A(s)$. Divide by n to obtain the first inequality. Using log instead of A the same argument applies. Let n go to infinity. This shows $A(t)/A(s) = log(t)/log(s)$. The choice $s = 2$ yields the claim $A(t) = log(t)$.

We are almost there. Let us prove the formula for H in the case of rational probabilities. It is clear that because of continuity this suffices. So let $p_i = \frac{m_i}{m}$, where m_i and m are natural numbers, $\sum_{i=1}^{r} m_i = m$. It is obvious how to use the central property of H in an appropriate situation to obtain

$$A(m) = H(p_1, \dots, p_r) + \sum_{i=1}^{r} p_i A(m_i).$$

Consider at first m equally probable events. The corresponding entropy is $A(m)$. Divide them into r blocks of sizes $m_1, m_2, \dots, m_r$. The basic property yields

$$A(m) = H(p_1, \dots, p_r) + \sum_{i} p_i A(m_i).$$

Using $A(t) = log(t)$ and solving for $H(p_1, \dots, p_r)$ the claim is obtained. $\square$

This theorem justifies why we consider the entropy function from Definition 1.2 as the measure of information we were looking for.

The whole idea behind the definition of entropy is in the last property of Theorem 1.3. It says that the average amount of information contained in the choice between certain symbols according to a given probability distribution is not changed when the information is revealed in various steps.

For example, consider $H(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$. Combine the second and third event to one event. The information contained in the choice between the first and the block of the remaining events is $H(\frac{1}{2}, \frac{1}{2}) = 1$. There is a probability of $1/2$ that the result is the second or third event. The choice between events two and three has an entropy of 1 bit again. In order to obtain $H(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ we have to add 1 (the contribution of the first stage) and $\frac{1}{2} \cdot 1$, the entropy contained in the choice of the second stage multiplied by the probability $1/2$ that this situation really arises. The result $H(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}) = 1.5$ is in accordance with our formula. As another example consider

$$H(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}) = \frac{1}{2} + \frac{1}{4}2 + \frac{1}{8}3 + \frac{1}{8}3 = 1.75.$$

It is clear that this entropy should be larger than $H(\frac{1}{2}, \frac{1}{2}) = 1$. Also, it should be expected that the maximum entropy given by a sample space of n points is obtained by the uniform distribution, and hence is $H(\frac{1}{n}, \frac{1}{n}, \dots) = log(n)$. In case $n = 4$ this yields a maximum entropy of $log(4) = 2$. Our entropy in the above example is indeed $1.75 < 2$.

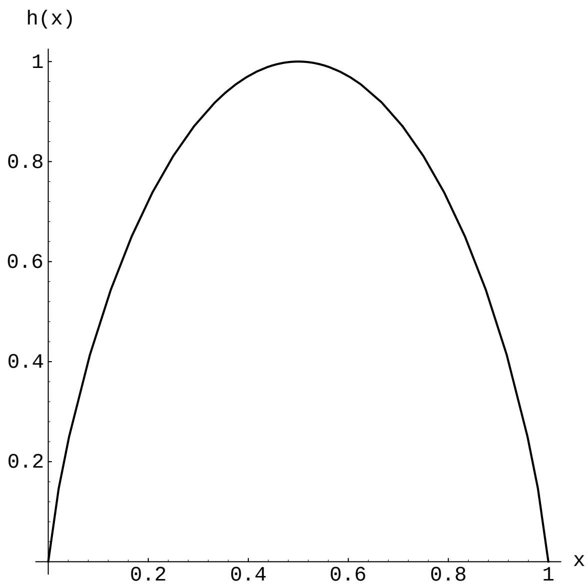

Figure 1.1: Binary entropy function

Consider the case of 4 equiprobable events. The corresponding entropy is $H(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) = log(4) = 2$. Now divide these four events into two blocks, one single event and a group of the remaining three. By the basic property the entropy is $H(\frac{1}{4},\frac{3}{4}) + \frac{1}{4}\cdot 0 + \frac{3}{4}log(3)$. We can solve for $H(\frac{1}{4},\frac{3}{4})$ and obtain $H(\frac{1}{4},\frac{3}{4}) = 2 - \frac{3}{4}log(3)$. This is in accordance with the general formula, which reads

$$H(\frac{1}{4},\frac{3}{4}) = \frac{1}{4}log(4) + \frac{3}{4}log(4/3).$$

The case of a sample space of just 2 points plays a particular role.

1.4 Definition. *The* **binary entropy function** *is defined by*

$$h(x) = -xlog(x) - (1-x)log(1-x) \ \textit{(for } 0 \leq x \leq 1).$$

Observe that $h(x) = H(x, 1-x)$ is indeed a special case of the Shannon entropy. We have $h(0) = 1,\ h(1/2) = 1$, and h is increasing on the interval $[0, 1/2]$.

Relation with binomials

Recall some basic facts on binomials. An n-set simply is a set that has n elements.

1.5 Definition. *Let $n \geq m \geq 0$ be natural numbers. Then $\displaystyle\binom{n}{m}$ denotes the number of m-subsets of an n-set.*

The numbers $\binom{n}{m}$ are known as **binomial numbers** or **binomial coefficients.** Some special values are obvious: $\binom{n}{0} = 1$ (every n-set has just 1 subset of 0 elements, the empty set), $\binom{n}{n} = 1$ (the only subset of n elements is the n-set itself) and $\binom{n}{1} = n$ (a set has as many 1-element subsets as it has elements). The next case $\binom{n}{2}$ poses a problem already. We do not have a short formula for these numbers yet. This can be done by a recursive formula.

1.6 Theorem.

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m} \quad \textit{for } m > 0.$$

Proof. Fix an n-set X and one of its elements e. The m-subsets of X (we know there are $\binom{n}{m}$ of them) can be divided into two types, those containing e and the others. There are $\binom{n-1}{m-1}$ of the first type and $\binom{n-1}{m}$ of the second. $\square$

As a special case we obtain the formula

$$\binom{n}{2} = \binom{n-1}{1} + \binom{n-1}{2}.$$

This can be used to prove recursively $\displaystyle\binom{n}{2} = n(n-1)/2$. It is a special case of a famous multiplicative formula for the binomials in terms of factorials:

1.7 Theorem.

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} = \frac{n(n-1)\cdot\cdots\cdot(n-m+1)}{m!}$$

Proof. The numerator of the expression on the right counts ordered subsets of m elements: there are n possibilities to choose the first element. Once it has been chosen there are $n-1$ possibilities to choose the second, and so on. There are $n \times (n-1) \times \cdots \times (n-m+1)$ possibilities to choose an m-tuple of different elements. Each such tuple defines an m-subset. In this way we count each m-subset many times? How many? As often as we can rearrange an m-set. It follows that we have to divide by $m!$ in order to count all m-subsets. □

There is a close relationship between the binary entropy function and binomials. As a tool we use the Stirling formula, which is frequently used in calculus and statistics.

1.8 Lemma. *The* **Stirling formula** *for factorials states that*

$$n! \sim \sqrt{2\pi n}(n/e)^n,$$

meaning that the quotient of left side and right side approaches 1 *for* $n \rightarrow \infty$.

A proof is in Feller [9]. Let now $n, m \rightarrow \infty$ such that $m/n \rightarrow p$. We want to check how the binomial coefficient $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ behaves. By the Stirling formula we have

$$lim(\frac{1}{n}log(\binom{n}{m})) = lim(log(n) - \frac{m}{n}log(m) - \frac{n-m}{n}log(n-m)) =$$

$$lim(-\frac{m}{n}log(\frac{m}{n}) - \frac{n-m}{n}log(\frac{n-m}{n})) = h(p).$$

We have seen the following:

1.9 Theorem. *Let* $n, m \rightarrow \infty$ *such that* $m/n \rightarrow p$. *Then*

$$lim(\frac{1}{n}log(\binom{n}{m})) = h(p).$$

Theorem 1.9 is an important relationship between binomials and the binary entropy function. We will use it freely in later chapters. An informal way of memorizing it is: $\binom{n}{pn} \approx 2^{nh(p)}$, for example $\binom{100}{40} \approx 2^{100 \cdot h(0.4)}$.

Basic properties of Shannon entropy

A basic tool in proving the basic properties of Shannon's entropy function is Jensen's inequality.

1.10 Definition. *A real function $f(x)$ defined on an open interval I is* **concave down** *if for every $x_0 \in I$ there is a line L through $(x_0, f(x_0))$, which is never under the graph of $f(x)$.*

A function f is concave down if when biking along its graph from left to right we lean to the right. This is certainly satisfied if $f''(x) < 0$.

Let $g(x) = f(x_0) + \lambda(x - x_0)$ be the equation of the line in Definition 1.10. By definition of concavity $f(x) \leq f(x_0) + \lambda(x - x_0)$ for all $x \in I$. Let $x_1 \leq x_2 \cdots \leq x_n$ be points in I and $p_1, \ldots, p_n$ probabilities of a probability space. Using the inequalities above yields

$$\sum_i p_i f(x_i) \leq f(x_0) + \lambda(\sum_i p_i x_i - x_0).$$

Choose in particular $x_0 = \sum_i p_i x_i$, observe that $x_0 \in I$.

1.11 Theorem (Jensen's Inequality). *Let $f(x)$ be concave down on the open real interval I, let $x_1 \leq x_2 \cdots \leq x_n$ be points in I and $p_1, \ldots, p_n$ probabilities of a probability space. Then*

$$\sum_i p_i f(x_i) \leq f(\sum_i p_i x_i).$$

Equality holds if and only if $f(x)$ is a linear function (its graph is a straight line) on the interval $[x_1, x_n]$.

Clearly the *log*-function is concave down on the interval $(0, \infty)$. Choose positive numbers x_i and a probability distribution p_i. An application of Jensen's inequality yields the famous inequality between the arithmetic and the geometric mean:

1.12 Theorem. *Let $x_i > 0,\ p_i \geq 0,\ i = 1, 2, \ldots, n,$ where $\sum_i p_i = 1$. Then:*

$$\sum_{i=1}^{n} p_i \cdot x_i \geq \prod_{i=1}^{n} x_i^{p_i}.$$

Equality holds if and only if all the x_i with $p_i \neq 0$ are equal.

Another application of Jensen's inequality to the *log*-function leads to a fundamental property of the entropy function.

1.13 Proposition (log sum inequality). *Let p_i and q_i be positive numbers, $i = 1, 2, \dots, n$ such that $\sum_i p_i = \sum_i q_i = 1$. Then the following holds:*

$$\sum_i p_i \cdot log(1/p_i) \leq \sum_i p_i \cdot log(1/q_i).$$

Equality holds if and only if $q_i = p_i$ for all i.

Proof. Apply Jensen's inequality to $x_i = q_i/p_i$. Observe
$\sum_i p_i x_i = \sum_i q_i = 1.$ □

Proposition 1.13, the famous **log sum inequality,** relates two probability distributions. Both sides of the inequality have the form of an expectation (see Definition 1.20 below). Consider the expression $\sum_i p_i \cdot log(1/q_i)$. The log sum inequality says that for p_i fixed this is minimized by choosing $q_i = p_i$. This minimum is then the entropy. Here is a consequence of this: choose $q_i = 1/n$. Then $H(p_1, \dots, p_n) \leq \sum_i p_i log(n) = log(n)$.

In words: the maximum entropy when n probabilities p_i are involved is reached when $p_1 = p_2 = \cdots = p_n = 1/n$. This maximum entropy is $log(n)$.

Joint entropy and conditional entropy

Every probability space has its entropy. Another basic notion of probability theory and statistics are random variables.

1.14 Definition. *Let Ω be a probability space. A* **random variable** *on Ω is a mapping f defined on Ω.*

As every random variable induces a probability distribution, we can also speak of the entropy of a random variable. Let X and Y be random variables with probability distributions $p_i = Pr(X = i)$ and $q_j = Pr(Y = j)$, respectively. Consider further the joint distribution of X and Y, given by the probabilities $p(i, j) = Pr(X = i \text{ and } Y = j)$. Observe that the joint distribution carries more information than the distributions of X and Y taken together. In fact, we may arrange the probabilities $p(i, j)$ in a matrix with rows indexed by i and columns indexed by j. The row sums are the p_i, the column sums are q_j. The other way around it is clear that the matrix is far from being determined by its row- and column sums.

1.15 Definition. *Let X, Y be random variables defined on the same probability space Ω. With the notation just introduced we call X and Y* **statistically independent** *if for every i in the range of X and j in the range of Y we have*

$$p(i,j) = p_i \times q_j.$$

The meaning behind this notion is that in the case of independent random variables information about the value of one of them gives no information whatsoever on the value of the other variable.

We wish to compare the entropies $H(X)$ and $H(Y)$ to the joint entropy $H(X,Y)$. In fact, there are more entropies around.

1.16 Definition. *Let X, Y be random variables defined on the same probability space Ω. The* **conditional probability** *is*

$$p(i|j) = p(i,j)/q_j.$$

This is the probability that $X = i$ if it is already known that $Y = j$.

Observe that X and Y are independent if and only if $p(i|j) = p_i$ always holds. For every fixed value j of Y the conditional probabilities $p(i|j)$ form a probability distribution. Denote the corresponding entropy by $H(X|Y = j)$. It is natural to define a conditional entropy $H(X|Y)$ as the weighted sum of these expressions, the weights being the probabilities q_j.

1.17 Definition. *The* **conditional entropy** *$H(X|Y)$ is defined by*

$$H(X|Y) = \sum_j q_j \cdot H(X|Y = j) = -\sum_{i,j} p(i,j) \cdot log(p(i|j)).$$

Here i, j run through the values of X and Y, respectively.

The conditional entropy is a new notion. It is not the entropy of a probability space. There is no probability space $X|Y$.

Here is an example to illustrate. Let the random variable X take on values $1, 2, 3, 4$ with probabilities $p_1 = p_2 = p_3 = p_4 = 1/4$. Its entropy is then $H(X) = log(4) = 2$. Let Y take on values $1, 2$ with probabilities $q_1 = q_2 = 1/2$. What can we say about the probabilities $p(i,j)$ that at the same time X takes on value i and Y takes on value j, where $1 \leq i \leq 4,\ j = 1, 2$? All we know is that those eight probabilities satisfy

$$\sum_{j=1}^{2} p(i,j) = p_i = 1/4 \text{ for all } i$$

and

$$\sum_{i=1}^{4} p(i,j) = q_j = 1/2 \text{ for } j = 1, 2.$$

In other words: if we write those 8 probabilities in a (4×2)-matrix then all row sums are 0.25 and all column sums are 0.5. Here is one of many possibilities how this matrix may look like:

$$\left(\begin{array}{c|c} 0.25 & 0 \\ \hline 0.1 & 0.15 \\ \hline 0.1 & 0.15 \\ \hline 0.05 & 0.2 \end{array}\right)$$

What would it mean for the matrix of probabilities if X and Y were independent? Then all entries would have to be $p_i \times q_j = 1/8$. Our random variables X, Y are independent if and only if the matrix of probabilities is

$$\left(\begin{array}{c|c} 0.125 & 0.125 \\ \hline 0.125 & 0.125 \\ \hline 0.125 & 0.125 \\ \hline 0.125 & 0.125 \end{array}\right)$$

Consider the conditional entropy $H(Y|X)$ for the first matrix. It is defined as a weighted sum, where the weights are the probabilities of X. In our case all those weights are $1/4$. Consider the case $X = 2$. The conditional probabilities on X are in this case $p(Y = 1|X = 2) = 0.4$ and $p(Y = 2|X = 2) = 0.6$, with entropy $H(0.4, 0.6) = h(0.4)$ where $h(x)$ is the binary entropy function of Definition 1.4. We obtain

$$H(Y|X) = \frac{1}{4}(H(1,0) + H(0.4, 0.6) + H(0.4, 0.6) + H(0.2, 0.8)) =$$

$$= \frac{1}{4}(2h(0.4) + h(0.2)).$$

Without any calculation the graph of $h(x)$ given in Figure 1.1 suggests that $h(0.4) \approx 0.96$ (very close to the maximum of 1) and $h(0.2) \approx 0.6$ (or maybe a little larger). This would give a value of $H(Y|X) \approx \frac{1}{4}(1.92 + 0.6) = 0.63$.

The main theorem

The main theorem shows that our notions cooperate in exactly the way our interpretation suggests:

1.18 Theorem. *Let X and Y be random variables defined on the same finite probability space. Then the following relations hold for the entropies:*

- $H(X,Y) \leq H(X) + H(Y)$, *with equality if and only if X and Y are independent.*
- $H(X,Y) = H(X) + H(Y|X)$.
- $H(X) \geq H(X|Y)$, *with equality if and only if X and Y are independent.*

Proof. Most interesting is the proof of the first statement. If we write out the definitions we see that we have to prove the following statement: $\sum_{i,j} p(i,j) \cdot log(1/p(i,j)) \leq \sum_{i,j} p(i,j) \cdot log(1/(p_i q_j))$. This follows from Proposition 1.13, the log sum inequality. The proposition also shows when equality occurs.

The second statement is a trivial formal manipulation, the third statement follows from the others: $H(X) + H(Y) \geq H(Y,X) = H(Y) + H(X|Y)$. □

Recall that we interpret $H(X)$ as a measure for the average amount of information per letter of a source operating with the corresponding probabilities. The first property of the main theorem says that the information contained in X and Y taken together is at most the sum of the individual informations. The second statement says that the joint information given by X and Y together can be written as the sum of the information given by X and the information given by Y when X is known. This equation is also known as the **chain rule** for entropies. The third property says that knowledge on Y cannot increase our uncertainty about X. Exactly then will this knowledge not help when X and Y are independent.

Finally, we come to a natural notion, which is not to be found in Shannon's classical papers, mutual information:

1.19 Definition. *The* **mutual information** *is*

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

Observe that this notion is symmetric in X and $Y : I(X,Y) = H(X,Y) - H(Y|X) - H(X|Y) = I(Y,X)$. Does this notion agree with our interpretation? $H(X)$ is the uncertainty about X, and $H(X|Y)$ is the remaining uncertainty about X once Y is known. We should therefore interpret the difference as the amount of information Y has disclosed about X.

The AEP and the law of large numbers

Before we go on let us review some more relevant basic notions and results from probability theory. Probability spaces and random variables are used throughout this text.

1.20 Definition. *Let $f : \Omega \longrightarrow \mathbb{R}$ be a real-valued random variable defined on the finite probability space Ω. Denote by p_i the probability of $i \in \Omega$. The* **expectation** *of f is the real number*

$$E(f) = \sum_{i \in \Omega} p_i f(i).$$

The **variance** *is defined as*

$$Var(f) = \sum_{i \in \Omega} p_i (f(i) - E(f))^2 = E((f - E(f))^2).$$

It follows from the definition that $Var(f) \geq 0$. If we multiply out the square in the definition we obtain

$$Var(f) = \sum_i p_i f(i)^2 - 2E(f) \sum_i p_i f(i) + E(f)^2 \sum_i p_i =$$

$$= E(f^2) - 2E(f)^2 + E(f)^2 = E(f^2) - E(f)^2.$$

1.21 Lemma. $Var(f) = E(f^2) - E(f)^2 \geq 0$.

Going back to the definition of the variance, let us sum not over all $i \in \Omega$, but only over those satisfying $|f(i) - E(f)| \geq \lambda$, for some $\lambda \geq 0$. We obtain $Var(f) \geq \lambda^2 \times Prob(|f(i) - E(f)| \geq \lambda)$. This proves a basic inequality:

1.22 Theorem (Tschebyscheff inequality). *Let $f : \Omega \longrightarrow \mathbb{R}$ be a real-valued random variable. Then*

$$Prob(|f(i) - E(f)| \geq \lambda) \leq \frac{Var(f)}{\lambda^2}$$

This reveals part of the meaning of the variance. If the variance is small, then large deviations from the expectation do not occur with high probability.

Let Ω be a sample space with probabilities $p_1, \dots, p_n$. We think of Ω as an experiment with n possible results. Result (letter) number i occurs with probability p_i. Equivalently we see Ω as a source which at each moment of time produces one of n possible results (letters), again with probabilities p_i. Then $H(p_1, \dots, p_n)$ is the entropy per letter of this source. Now we imagine repeating this experiment several times, maybe a large number N of times. This is described by N independent random variables, each with n possible results and the good old probabilities p_i.

1.23 Definition. *Let Ω be a sample space with probabilities $p_1, \dots, p_n$. Let $X_1, \dots, X_N$ be statistically independent random variables, each with the same distribution $p_1, \dots, p_n$. A* **signal** *σ of length N is a possible vector of results.*

Write $\sigma = (a_1, \dots, a_N)$, where each $a_i \in \{1, 2, \dots, n\}$. The probability of the signal σ is then the product of the p_{a_j}, where $j = 1, 2, \dots, N$.

As an example let $p_1 = 0.6, p_2 = 0.4$. The signals of length 2 are

$$(1,1),\ (1,2),\ (2,1),\ (2,2),$$

of probabilities

$$p_1^2 = 0.36\ p_1p_2 = 0.24 \text{ again } 0.24 \text{ and } p_2^2 = 0.16.$$

These probabilities add up to 1, as they should. For each N there is a probability distribution on the n^N signals of this length There are $2^6 = 64$ signals of length 6 in our example. One of them is $(1,1,1,2,2,1)$. It has probability $p_1^4p_2^2 = 0.020736$. It is clear what the probabilities look like in general:

1.24 Lemma. *Let σ be a signal of length N. For each $i = 1, 2, \dots, n$ let h_i be the number of coordinates where the entry of σ is i (h_i is the* **frequency** *of i in σ). Then*

$$Prob(\sigma) = p_1^{h_1} \cdot \dots \cdot p_n^{h_n} = \prod_{i=1}^{n} p_i^{h_i}.$$

In our example we had $h_1 = 4, h_2 = 2$.

The **law of large numbers** is a consequence of the Tschebyscheff inequality. It clarifies the structure of the space of signals of large length and provides another interpretation of the entropy function. Here is what it says:

For every pair $\epsilon, \delta > 0$ there is a $N_0 = N_0(\epsilon, \delta)$ such that for every $N \geq N_0$ the set of signals of length N can be partitioned in two parts:

- a set of total probability $< \epsilon$, and
- a set all of whose elements satisfy

$$|h_i/N - p_i| < \delta \text{ for all } i.$$

This means that if we pick a long signal (of length $> N_0$) at random, then we are almost sure that the relative frequencies of the letters are close to the probabilities. If σ is a signal of length N, then its probability is $p = Pr(S) = \prod_i p_i^{h_i}$. It follows $log(p) = \sum_i h_i \cdot log(p_i)$, hence $log(p)/N = \sum_i (h_i/N) \cdot log(p_i)$.
If σ is one of the **typical** signals, then this expression is close to $\sum_i p_i \cdot log(p_i) = -H(p_1, \ldots, p_n)$. This is another interpretation of the entropy function.

In information theory this property is also known as the **asymptotic equipartition property** AEP. Here is again what it says: consider the signals of large length N produced by a source of entropy $H = H(p_1, \ldots, p_n)$. If we are willing to disregard very unlikely events (signals), then we can think of the source as producing 2^{HN} signals of length N, where each such typical signal has probability very close to $1/2^{NH}$.

An illustrative example: Effective alphabet size

Let X_1 and X_2 be discrete random variables and $X = aX_1 + (1-a)X_2$ the disjoint mixture.

How is X defined? Let $p_1(1), \ldots, p_1(n)$ be the probabilities defining X_1 and $p_2(n+1), \ldots, p_2(m)$ the probabilities defining X_2. Then X is defined by

$$ap_1(1), \ldots, ap_1(n), (1-a)p_2(n+1), \ldots, (1-a)p_2(m).$$

It follows

$$H = -\sum(ap_1(i)log(ap_1(i)) - \sum_j((1-a)p_2(j))log((1-a)p_2(j)) =$$

$$= -alog(a) - (1-a)log(1-a) + aH_1 + (1-a)H_2$$

and

$$H = h(a) + aH_1 + (1-a)H_2$$

which is not surprising as we used this property as an axiom in the definition of the entropy.

See H as a function of a In order to determine the maximum we differentiate, using the derivative of the binary entropy function h :

$$dH/da = h'(a) + H_1 - H_2 = log((1-a)/a) + H_1 - H_2.$$

This vanishes when $(1-a)/a = 2^{H_2 - H_1}$. It follows that the maximum value of H is reached when

$$a = 1/(1+2^u), \text{ where } u = H_2 - H_1.$$

Use this value for a now. We claim that in fact

$$2^H = 2^{H_1} + 2^{H_2}.$$

In order to see this consider

$$2^H - 2^{H_1} = 2^{H_1}(2^{h(a)+(1-a)u} - 1).$$

We have

$$h(a) = alog(1+2^u) - (1-a)log(2^u/(1+2^u)) =$$

$$= alog(1+2^u) - (1-a)(u - log(1+2^u)) = log(1+2^u) + (a-1)u.$$

It follows $h(a) + (1-a)u = log(1+2^u)$ and therefore

$$2^H - 2^{H_1} = 2^{H_1}(2^{log(1+2^u)} - 1) = 2^{H_1}2^u = 2^{H_2}.$$

What is the interpretation of this result in the context of the AEP? As 2^{nH} is the number of typical signals produced by a source of entropy H we see 2^H as the **effective alphabet size:** the same number of typical signals would be produced by a uniform space with 2^H elements. Our calculations show that the mixture of two sources produces a source whose effective alphabet size is at most the sum of the individual effective alphabet sizes. The sum is reached by an optimal choice of the mixing probability a.

Exercises 1

1.1. *What is the relation between* $H(p_1, p_2, p_3)$ *and*
$H(\frac{p_1}{3}, \frac{2p_1}{3}, \frac{p_2}{3}, \frac{2p_2}{3}, \frac{p_3}{3}, \frac{2p_3}{3})$*?*
Give a precise formula.

1.2. *What is the relationship between* $H(p_1, p_2, p_3, p_4)$ *and*
$H(\frac{p_1}{2}, \frac{p_1}{2}, \frac{p_2}{2}, \frac{p_2}{2}, \frac{p_3}{2}, \frac{p_3}{2}, \frac{p_4}{2}, \frac{p_4}{2})$*?*
Give a precise formula.

1.3. *What is the maximum value of*

$$H(p_1, p_2, \ldots, p_{1024})$$

over all probability distributions with 1024 *probabilities?*

1.4. *Compare the following experiments:*

- **A:** *Four possible events occur with probabilities* $0.5,\ ,0.2,\ 0.2,\ 0.1$
- **B:** *Four possible events occur each with probability* $1/4$.

Which of these experiments yields a larger amount of information? Compute the entropies and compare.

1.5. *Compare the following experiments:*

- **A:** *Three possible events occur with probabilities* $0.5, 0.25, 0.25$
- **B:** *Three possible events occur each with probability* $1/3$.

Which of these experiments yields a larger amount of information? Compute the entropies and compare.

1.6. *What is larger (contains more information),* $H(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$ *or* $H(\frac{1}{3}, \frac{1}{3}, \frac{2}{9}, \frac{1}{9})$*? Compute both entropies and compare.*

1.7. *In a best of* 3 *tennis match between two evenly matched players* A, B *(each set is won by each player with probability* 0.5*) compute the conditional probability that* A *wins the match given that* A *won the first set.*

1.8. *In the situation of the preceding problem let* X *be the number of sets and* Y *the winner of the first set. Compute*

$$H(X),\ H(X|Y) \text{ and } I(X,Y).$$

1.9. *A fair coin is tossed* 4 *times. Let* X *be the random variable number of times that heads shows. Compute* $H(X)$.

1.10. *In the situation of the preceding problem let* Y *be the result of the first toss. Compute* $H(Y),\ H(X|Y)$ *and* $I(X,Y)$.

1.11. *We use two fair dice, a red and a green.* X *describes the result of throwing the red die,* Y *describes the green die and* $Z = X + Y$. *Find the following:*

1. $H(X,Y)$
2. $H(X,Z)$
3. $H(Z|X)$
4. $H(Z)$
5. $I(X,Z)$

1.12. *Let* X *take on values* $1,2$ *with probabilities* $0.5,\ 0.5$ *and* Y *take on the values* $1,2,3$ *with probabilities* $0.4,\ 0.4,\ 0.2$.
Assume X,Y *are independent.*
Write the $(2,3)$*-matrix of probabilities* $p(X = i, Y = j)$.

1.13. *An unbiased die is tossed once. The random variable* X *describes the result. If the face of the die is* $1,2,3$ *or* 4, *then a fair coin is tossed once. If the face is* 5 *or* 6, *then the coin is tossed twice. Let* Y *be the number of heads obtained.*

Compute $H(X),\ H(Y)$ *and the information* $I(X,Y)$.

1.14. *Prove the additivity of the expectation: if* f *and* g *are real-valued random variables defined on the same probability space, then* $E(f+g) = E(f) + E(g)$.

1.15. *Let $F_1, F_2, \dots, F_n$ be independent real-valued random variables each taking value 1 with probability p, value 0 with probability $q = 1 - p$. Let $F = F_1 + \cdots + F_n$. We know from the preceding exercise $E(F) = np$.*

Prove that the variance of F is $\sigma^2 = npq$.

1.16. *Flip a fair coin until head shows for the first time. Let the random variable be the number of flips required. What is the entropy of X?*

1.17. *Let X be a real-valued random variable. How is $H(X)$ related to $H(2^X)$ and to $H(cos(X))$?*

1.18. *Show that if $H(Y|X) = 0$, then Y is a function of X.*

1.19. *let X describe the possible results of the world series (it is best of seven) and Y the number of games played. Compute the entropies involved.*

1.20. *The inhabitants of a certain village are divided into two groups A and B. Half the people in group A always tell the truth, three-tenths always lie, and two-tenths always refuse to answer. In group B, three-tenths of the people are truthful, half are liars, and two-tenths always refuse to answer.*

Let p be the probability that a person selected at random belongs to group A.

Let I be the information conveyed about a person's truth-telling status by specifying his group membership.

Write a mathematical expression for I in terms of entropies. For which value of p is I maximal and what is this maximal value?

1.21. *Compute the derivative $h'(x)$ of the binary entropy function.*

1.22. *Prove the general inequality*

$$H(Z \mid X, Y) \leq H(Z \mid X)$$

and characterize the case of equality.

1.23. *Prove the general inequality*

$$H(Y, Z \mid X) \leq H(Y \mid X) + H(Z \mid X)$$

and characterize the case of equality.

1.24. *Let $A = (a_{ij})$ be a doubly stochastic (n, n)-matrix (meaning that each row and each column of A describes a probability space: $a_{ij} \geq 0$, all row and column sums equal 1.)*

Let $p_1, \dots, p_n$ be probabilities and define $q_i = \sum_j a_{ij} p_j$. Prove

$$H(q_1, \dots, q_n) \geq H(p_1, \dots, p_n)$$

Chapter 2

Source encoding

We have seen that the **entropy** function

$$H(p_1, p_2, \ldots, p_n) = -\sum_i p_i log(p_i)$$

is a measure for the average amount of information emitted by a source which at each point of time produces one of n letters according to the probability distribution p_i. This measure is per letter. The average amount of information contained in a string (signal) of r letters is then $r \cdot H(p_1, p_2, \ldots, p_n)$. We will see that entropy comes up in the context of **source coding** as well.

2.1 Definition. *An* $(n, 2)$*-source code* f *is a mapping, which associates to each letter* i *a binary* u_i-*tuple* $f(i)$*. Call* u_i *the length of* $f(i)$*. The source code is* **prefix-free** *if it never happens that for some* $i \neq j$ *the first* u_i *symbols of* $f(j)$ *form* $f(i)$*. The* **expected length** *of* f *is* $\sum_i p_i u_i$*.*

We consider the problem of constructing a prefix-free source encoding with minimal expected length. This minimum expected length $S(p_1, \ldots, p_n)$ is another natural measure, just like the entropy, for the amount of information emitted by the source. As $S(1/2, 1/2) = H(1/2, 1/2) = 1$, both measures are normalized in the same way. We would expect therefore S always to be close to H. The case of the uniform distribution $p_i = 1/n$ when n is not a power of 2 shows that we cannot expect $S = H$ in general. This would be too much to expect as H is a real measure, whereas S results from the solution of a discrete problem. We will see shortly that there is a close link.

The entropy bound

Let $u_1 \geq \cdots \geq u_n$ and consider bitstrings of length u_1. Altogether there are 2^{u_1} such strings. Codeword number i of the source code is the beginning section of $2^{u_1-u_i}$ such tuples. By the prefix condition these tuples are all distinct. It follows $\sum_i 2^{u_1-u_i} \leq 2^{u_1}$. Divide by 2^{u_1} :

2.2 Theorem (Kraft theorem). *The codeword length $u_1, \ldots, u_n$ of a prefix-free $(n, 2)$-source code satisfy*

$$\sum_{i=1}^{n} 2^{-u_i} \leq 1.$$

As an example consider the code consisting of the four codewords

000, 001, 01, 1.

This code is prefix-free (no codeword except for the last one starts with 1, the first and second word do not start with 01). The lengths of the codewords are $u_1 = u_2 = 3,\ u_3 = 2,\ u_4 = 1$. The Kraft inequality is of course satisfied, and in fact it happens to be an equality:

$$\sum_{i=1}^{4} 2^{-u_i} = 1/2 + 1/4 + 1/8 + 1/8 = 1.$$

What was the argument in the proof of the Kraft inequality? We consider the set of all 3-tuples. There are $2^3 = 8$ such triples. One of them starts with the first codeword, one starts with the second codeword, 2 start with the third codeword (010 and 011), and 4 triples start with the last codeword. We counted $1 + 1 + 2 + 4$ different triples.

The form of the inequality suggests to consider the numbers 2^{-u_i} as probabilities. So let $q_i = 2^{-u_i}$ and define α such that $\sum \alpha q_i = 1$. We have $\alpha \geq 1$. The log sum inequality states

$$H(p_1, \ldots, p_n) \leq -\sum p_i log(\alpha q_i) = -log(\alpha) - \sum p_i log(q_i).$$

However, $-log(q_i) = u_i$, so the last sum is $\sum p_i u_i$, the expected length of the encoded word.

2.3 Theorem. *The minimum expected length $S(p_1, \ldots, p_n)$ cannot be less than the entropy $H(p_1, \ldots, p_n)$.*

Shannon's source encoding

This is motivated by the analogy with the entropy. Order the probabilities such that $p_1 \geq p_2 \geq \dots$. As $H(p_1, p_2, \dots, p_n)$ is defined as the expected value of the function $log(1/p_i)$, we try to choose length u_i to be close to $log(1/p_i)$. Let u_i defined by

$$2^{-u_i} \leq p_i < 2^{-(u_i-1)}, \text{ equivalently}$$
$$2^{u_i-1} < 1/p_i \leq 2^{u_i}.$$

Let

$$Q_1 = 0,\ Q_2 = p_1,\ Q_3 = p_1 + p_2,\ \ \dots, Q_n = p_1 + p_2 + \cdots + p_{n-1}.$$

Then Q_i is the probability that one of the $i-1$ most frequent letters occurs. Define $f_S(i)$ to be the first u_i digits after the point in the binary representation of the number Q_i. Here the subscript S stands for *Shannon.* If $Q_i = 0.x_1x_2\dots$ is the binary representation of Q_i, then $f_S(i) = (x_1, x_2, \dots, x_{u_i})$. We have to show that Shannon's source encoding function f_S is prefix-free: In fact, assume $f_S(i)$ forms the first u_i digits of $f_S(j)$ for some $j > i$. We have $Q_j - Q_i \geq p_i \geq 2^{-u_i}$ by the choice of u_i. We are assuming that $Q_j = 0.y_1y_2\dots$, where $y_r = x_r, r = 1, 2, \dots, u_i$. It follows $Q_j - Q_i \leq 0.00\dots0111\dots$, where we have u_i zeroes behind the point. Thus $Q_j - Q_i \leq 2^{-u_i}$. Comparing with the inequality above we get $Q_j = Q_i + 2^{-u_i}$. Considering the binary representations we see that this is impossible.

We have seen that f_S is prefix-free. It is now an easy exercise to compare the expected length of f_S with the entropy of the source. One just has to go back to the definition of u_i.

2.4 Theorem. *Shannon's source encoding f_S is prefix-free. Its expected length satisfies*

$$H(p_1, p_2, \dots, p_n) \leq \sum_i p_i u_i < H(p_1, p_2, \dots, p_n) + 1.$$

Proof. The lower bound follows from Theorem 2.3. For the upper bound use $log(1/p_i) > (u_i - 1)$. It follows

$$H(p_1, \dots, p_n) = \sum p_i log(1/p_i) > \sum p_i u_i - \sum p_i.$$

□

The Huffman code

This is the optimal solution, giving a prefix-free $(n, 2)$-source code of minimal expected length. Starting point are the following obvious combinatorial observations: let f be an **optimal** source code and order the probabilities such that $p_1 \geq p_2 \cdots \geq p_n$. Then the following hold:

- If $p_i > p_j$, then $u_i \leq u_j$.
- $u_{n-1} = u_n$.

Moreover there exists an optimal code such that $f(n-1)$ and $f(n)$ differ only in the last bit.

The two properties above are obviously satisfied. How about the additional property? Assume for example $f(n-1) = 001001$. Obviously $f(n)$ has length 6 again. We want to choose $f(n) = 001000$. The only conceivable danger is that this may already be a codeword, say $f(n-2) = 001000$. If this is the case then we simply flip $f(n-2)$ and $f(n)$, obtaining an optimal code that satisfies the additional property.

Huffman encoding is based on the following observation: combine the two least probable letters $n-1, n$ to one letter of probability $p_{n-1} + p_n$. Assume an optimal code g has been found for this smaller alphabet. Then an optimal code f for the original problem is found by putting $f(i) = g(i)$ for $i \leq n-2$ and choosing $f(n-1), f(n)$ to be $g(n-1)$ followed by 0 and 1 respectively. This method is applied recursively. Once the alphabet is reduced to only two letters the optimal code is obvious. Then we can start working our way backwards.

An example

We consider the example in the section of Stinson's book dedicated to source encoding: $p_1 = 0.60, p_2 = 0.13, p_3 = 0.12, p_4 = 0.10, p_5 = 0.05$. Consider Shannon encoding at first It follows $u_1 = 1, u_2 = 3, u_3 = u_4 = 4, u_5 = 5$. We have $Q_1 = 0$, and we calculate $Q_2 = p_1 = 0.6 = 0.1001{*}_2, Q_3 = 0.73 = 0.1011{*}_2, Q_4 = 0.85 = 0.1101{*}_2, Q_5 = 0.95 = 0.11110 *_2$. Here we have used a subscript $_2$ to indicate binary representations. It follows

$$f_S(1) = 0,\ f_S(2) = 100,\ f_S(3) = 1011,\ f_S(4) = 1101,\ f_S(5) = 11110.$$

The expected length of f_S is 2.12.

Now consider the steps of Huffman encoding. In the first round we obtain probabilities (0.60, 0.15, 0.13, 0.12), then (0.60, 0.25, 0.15) and finally $(0.60, 0.40)$. This is where we start. The codewords are 0 and 1, of course. Going back we obtain codewords $(0, 10, 11)$, before that $(0, 11, 100, 101)$ and the original distribution is encoded as $(0, 100, 101, 110, 111)$, in other words an optimal Huffman encoding is

$$f_H(1) = 0,\ f_H(2) = 100,\ f_H(3) = 101,\ f_H(4) = 110,\ f_H(5) = 111.$$

The optimal expected length is $0.6 \times 1 + 0.4 \times 3 = 1.8$ as compared to expected length 2.12 for the Shannon code and the entropy of 1.74.

How many signals?

Consider our prefix-free code with words

$$a = 000,\ b = 001,\ c = 01,\ d = 1.$$

It has $f_1 = 1, f_2 = 1, f_3 = 2$ (one word of length 1, one of length 2 and two of length 3). It is a natural combinatorial question to ask how many signals (bitstrings) of length k can be composed out of these codewords? Denote this number by N_k and set $N_0 = 1$. Clearly $N_1 = f_1 = 1, N_2 = 2$ (the signals of length 2 being c and dd), $N_3 = 5$ (signals a, b, cd, dc, ddd), $N_4 = 9$ (signals $ad, bd, cc, cdd, da, db, dcd, ddc, dddd$). In general obviously

$$N_k = N_{k-1} + N_{k-2} + 2N_{k-3} \text{ provided } k \geq 3$$

(the **recurrence relation**). In fact, the first word of the signal is either a, b, c or d. In the first case there are N_{k-1} possibilities to complete the prefix a to a signal of length k, analogously for the other cases.

The recurrence relation shows that the N_k are uniquely determined as soon as we know N_0, N_1, N_2 (the **initial conditions**). In our case $N_0 = N_1 = 1, N_2 = 2$ and the recurrence relation yields

$$N_3 = 5,\ N_4 = 9,\ N_5 = 18,\ N_6 = 37,\ N_7 = 73,\ N_8 = 146, \ldots$$

Can we find a formula, and how fast are the N_k growing? This is a classical problem of the theory of recurrence relations. The single most famous case are the Fibonacci numbers $1, 1, 2, 3, 5, 8, 13, 21 \dots$ (recurrence relation $F_k = F_{k-1} + F_{k-2}$). Here is one way of solving this type of problem: at first forget the initial conditions and concentrate entirely on the recurrence relation. Search for constants α such that the sequence α^k satisfies the recurrence. This leads to the **characteristic equation**

$$\alpha^3 - \alpha^2 - \alpha - 2 = 0.$$

One solution is $\alpha = 2$. We can factor out:

$$\alpha^3 - \alpha^2 - \alpha - 2 = (\alpha - 2)(\alpha^2 + \alpha + 1)$$

and the solutions of the quadratic are of course third roots of unity: : $\alpha = -1/2 + i\sqrt{3}/2 = cos(2\pi/3) + isin(2\pi/3) = exp(i2\pi/3)$ and its conjugate $\overline{\alpha} = \alpha^2 = exp(-i2\pi/3)$. The roots of the characteristic equation are 2 and $\alpha, \overline{\alpha}$. This means that $2^k, \alpha^k, \overline{\alpha}^k$ are three sequences satisfying the recurrence relation. Obviously each linear combination

$$A2^k + B\alpha^k + C\overline{\alpha}^k$$

will also satisfy it. It suffices to choose A, B, C such that the initial conditions are satisfied and we have our sequence N_k. This shows immediately that N_k grows like $A2^k$. In fact, the sequences α^k and $\overline{\alpha}^k$ are periodic of period 3. The initial conditions give the system

$$\boxed{\begin{array}{ccccccc} A & + & B & + & C & = 1 \\ 2A & + & B\alpha & + & C\overline{\alpha} & = 1 \\ 4A & + & B\overline{\alpha} & + & C\alpha & = 2 \end{array}}$$

The sum of all equations shows $7A = 4$, so $A = 4/7$. The complex conjugate shows that necessarily $C = \overline{B}$. The first equation shows $Re(B) = 3/14$. This suffices to obtain an explicit formula:

$$N_k = 2^{k+2}/7 + \begin{cases} 3/7 & \text{if} \quad k \equiv 0 \bmod 3 \\ -1/7 & \text{if} \quad k \equiv 1 \bmod 3 \\ -2/7 & \text{if} \quad k \equiv 2 \bmod 3. \end{cases}$$

Exercises 2

2.1. *Compute the Huffman encoding for* 4 *symbols with probabilities*

$$p_a = 0.3,\ p_b = 0.25,\ p_c = 0.25,\ p_d = 0.2.$$

2.2. *In the same situation compute the Shannon encoding.*

2.3. *Consider a prefix code with lengths* $u_1 \geq \cdots \geq u_n$*. Prove the following: If there is equality in the Kraft inequality:*

$$\sum_{i=1}^{n} 2^{-u_i} = 1$$

then each binary u_1*-tuple has as an initial segment one of the codewords of the source code.*

2.4. *Consider the prefix-free code with words* 0, 10 *and* 11*. Find a general expression for the number* N_k *of signals of length* k*.*

Chapter 3

The entropy of the English language

Claude Shannon discusses in [20] various experimental and heuristic methods to estimate the entropy (and thus also the redundancy) of written English. This is not only interesting in itself. It also presents nice applications of the basic properties of the entropy function and it poses interesting problems. As usual Shannon's treatment of the matter was not very rigourous mathematically. We will therefore have to do some work in order to justify his results.

Approximations

The first method is to consider digrams, trigrams and so on, to estimate the corresponding probabilities and compute the entropies. Lucky [17] mentions early tables such as those by F. W. Kaeding who in 1898 counted a total of $59,298,274$ letters in German text and those put together by E. L. Thorndike in the 1920s and 1930s based on 20 million words in English text.

Let P^n denote the space of n-grams. For $n \leq 3$ there exist tables of the relative frequencies in English texts. For every n we have a probability space of size 26^n. Denote by $G_n = H(P^n)/n$ the entropy per letter of these probability spaces. For $n \leq 3$ the existing tables give the following values:

$$G_0 = log(26) = 4.7, G_1 = 4.14, G_2 = 3.56, G_3 = 3.3$$

In a 27-letter alphabet with the space as an additional letter the values are slightly different:

$$G_0 = log(27) = 4.76,\ G_1 = 4.03,\ G_2 = 3.32,\ G_3 = 3.1,\ G_4 = 2.8$$

It seems that the G_n form a descending sequence. This can be expected. The limit may then be called the **entropy of the language.** By extrapolation one could expect a value for G_8 of about 2.3. As $log(27)$ is a little less than 5 this indicates that the redundancy of the English language may be around 50% if statistical effects extending over more than 8 letters are not taken into consideration.

Shannon gives examples of randomly generated text, where the distribution of letters or words are based on statistics for English text. The most insipid example is the approximation of degree 0, where all 27 symbols (26 letters and the space) are chosen uniformly at random. Here is what Shannon obtains:

```
XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGXYD
QPAAMKBZAACIBZLHJQD
```

This is not too exciting. Here is what resulted when the symbols were chosen according to their frequency in standard English text (the first order approximation):

```
OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH FEI ALPHENHTTPA
OOBTTVA NAH BRL
```

The second-order approximation is obtained by choosing digrams (order pairs of symbols) according to their standard frequency, which is read off from tables:

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

Here is the highly interesting third-order approximation, based on the frequency of trigrams:

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

Lucky [17] gives a fourth-order approximation based on the frequencies of symbols in earlier chapters of his own book. Here each letter is determined at random according to the conditional probabilities based on the 3 preceding letters:

```
The generated job providual better trand the displayed code,
abovery upondults well the coderst in thestical it do hock
bothe merg. (Instates cons eration. Never any of puble and to theory.
Evential callegand to elast benerated in with pies as is with the
```

As Lucky puts it: *This is getting scary. By now there are many real English words, and even the pseudowords look like they ought to be real words. At a quick glance, it looks like ordinary English.*

Back to Shannon and his the first-order approximation based on word frequency:

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

Here is finally the second-order approximation, based on frequencies of pairs of words:

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

which contains everybody's favorite, the phrase AND IN FRONTAL ATTACK ON AN ENGLISH WRITER.

Using Zips's law

G. K Zipf observed in [22] that the following formula seems to be a good approximation for the relative frequency of the more frequent words:

$$p_n = k/n.$$

Here p_n is the probability of the n-th most frequent word and k is a constant which may vary from language to language. This is known as **Zipf's law.** Here are some values taken from tables:

$$p_1 = P(the) = 0.071, p_2 = P(of) = 0.034, p_{10} = P(I) = 0.01,$$

$$p_{100} = P(say) = 0.001, p_{1000} = P(quality) = 0.0001.$$

The data shows that the constant should be $k = 0.1$ A doubly-logarithmic graph of p_n as a function of n shows that this "law" is not all too far from the truth. The points of the graph are close to a line of slope -1. Shannon goes on to estimate the entropy of the English language on the assumption that Zipf's law holds. He assumes therefore $p_n = \frac{1}{10n}$ until the sum of the probabilities reaches 1. We get a total of 8727 words and an entropy of $-\sum_{n=1}^{8727} p_n log(p_n) = 11.82$ bits per word. If one uses the estimate of 4.5 for the average length of a word, this leads to an entropy of 2.62 per letter. This agrees rather well with the results of the preceding section, but again we have not taken long-range effects into consideration.

Sequences whose limit is the entropy

Recall from Definition 1.23 that an n-tuple σ of letters is a **signal** of length n, denote by $p(\sigma)$ its probability (relative frequency in standard English text). In cryptography these are also known as n**-grams.** Formally the entropy $G_n = H(P^n)/n$ of n-grams per letter as introduced in the beginning of this chapter is

$$G_n = -\frac{1}{n} \sum_{\sigma} p(\sigma) log(p(\sigma)),$$

where the sum is extended over all signals of length n. Consider also F_n, the entropy of the n-th letter when the preceding $n-1$ letters are known. This is a conditional entropy:

$$F_n = \sum_{\sigma} p(\sigma) \sum_{S} H(S|\sigma) = -\sum_{\sigma,S} p(\sigma, S) log(p(S|\sigma)).$$

Here σ represents a signal of length $n-1, S$ a letter, $p(\sigma, S)$ the probability of the signal (σ, S) of length n and $p(S|\sigma)$ the conditional probability that S is the next letter when σ is known.

It is now a simple matter to show that F_n is a monotonely decreasing sequence: with the same notation as before we have

$$F_{n+1} = \sum_{\sigma} \sum_{S'} p(\sigma, S') H(S|S', \sigma).$$

By a basic property of the entropy function (see Exercise 1.22) we have $H(S|S', \sigma) \leq H(S|\sigma)$. Consequently $F_{n+1} \leq \sum_{\sigma} p(\sigma) H(S|\sigma) = F_n$. Next we derive relations between the functions F_n and G_n.
Going back to the definition of F_n and replacing $p(S|\sigma)$ under the logarithm by $p(\sigma, S)/p(\sigma)$ we obtain

$$F_n = -\sum_{\sigma, S} p(\sigma, S) log(p(\sigma, S)) + \sum_{\sigma, S} p(\sigma, S) log(p(\sigma)).$$

Comparing with the definition of G_n we get $F_n = nG_n - (n-1)G_{n-1}$. Sum up the F_n :

$$\sum_{n=1}^{N} F_n = (G_1 - 0) + (2G_2 - G_1) + \cdots + (NG_N - (N-1)G_{N-1}) = NG_N.$$

Remember that F_n is monotonely decreasing. It follows that G_n has the same property and that the limits of both sequences coincide. The last relation also shows that $G_n \geq F_n$.

3.1 Theorem. *Let G_n denote the entropy per letter of n-grams and F_n the conditional entropy of the n-th letter when the preceding letters are known. Then $G_n \geq F_n$. Both sequences are monotonely decreasing and have the same limit, which we call the entropy of the language. Moreover we have the relations*

$$G_n = \frac{1}{n} \sum_{i=1}^{n} F_i \text{ and } F_n = nG_n - (n-1)G_{n-1}.$$

Prediction: first variant

Here is one experiment used by Shannon: a test person guesses the following letter (using here an alphabet of 27, the letters plus the period). If the guess is correct, then the person is informed of this fact and a dash is written instead of the letter. Otherwise the person is informed of the correct letter and this letter is written out. In the reduced text the dashes correspond to redundant letters. Here is an example:

t	h	e		r	o	o	m		w	a	s		n	o	t		v	e	r	y		
-	-	-	-	r	o	o	-	-	-	-	-	-	n	o	t	-	v	-	-	-	-

l	i	g	h	t		a		s	m	a	l	l		o	b	l	o	n	g	
-	i	-	-	-	-	-	-	s	m	-	-	-	-	o	b	l	-	-	-	-

r	e	a	d	i	n	g		l	a	m	p		o	n		t	h	e	
r	e	a	-	-	-	-	-	-	-	-	-	-	o	-	-	-	-	-	-	-

d	e	s	k		s	h	e	d		g	l	o	w		o	n	
d	-	-	-	-	s	h	e	d	-	g	l	o	-	-	o	-	-

p	o	l	i	s	h	e	d		w	o	o	d		b	u	t		l	e	s	s	
p	-	l	-	s	-	-	-	-	-	o	-	-	-	b	u	-	-	l	-	s	-	-

o	n		t	h	e		s	h	a	b	b	y		r	e	d		c	a	r	p	e	t
o	-	-	-	-	-	-	s	h	-	-	-	-	-	r	e	-	-	c	-	-	-	-	-

In this case 69% of the letters were guessed correctly.

Prediction:second variant

In this variant the test person keeps on guessing until the correct letter is obtained. The reduced text consists of the numbers of guesses. Here is an example given by Shannon:

T	h	e	r	e		i	s		n	o		r	e	v	e	r	s	e		o	n	
1	1	1	5	1	1	2	1	1	2	1	1	15	1	17	1	1	1	2	1	3	2	1

a		m	o	t	o	r	c	y	c	l	e		a		f	r	i	e	n	d	
2	2	7	1	1	1	1	4	1	1	1	1	1	3	2	8	6	1	3	1	1	1

o	f		m	i	n	e		f	o	u	n	d		t	h	i	s		o	u	t	
1	1	1	1	1	1	1	1	6	2	1	1	1	1	1	1	2	1	1	1	1	1	1

r	a	t	h	e	r		d	r	a	m	a	t	i	c	a	l	l	y	
4	1	1	1	1	1	1	11	5	1	1	1	1	1	1	1	1	1	1	1

t	h	e		o	t	h	e	r		d	a	y	
6	1	1	1	1	1	1	1	1	1	1	1	1	1

We imagine tables with relative frequencies in standard English text. An ideal test person would personify these tables. For such an ideal test person the reduced text carries the same information as the original text. We see that the transformation from the original to the reduced test is a source code (assuming these tables of frequencies to exist). Shannon uses this method experimentally to approximate the entropy of English when long-range effects are taken into account. We use our machinery to make sure we know what is being measured and in order to bound the error:
Let n be fixed and let σ be the portion of text known so far, where σ consists of $n-1$ letters. For each letter j let $p(j|\sigma)$ be the conditional probability that j is the next letter. The ideal test person will choose j such that $p(j|\sigma)$ is maximal. If this guess should be wrong, then j with the second to maximal $p(j|\sigma)$ is chosen and so on. Denote by $j(i)$ the letter whose conditional probability is the i-th highest, $i = 1, 2, \dots$. Define $q_i^{(n)} = \sum_\sigma p(\sigma, j(i))$. Then $q_i^{(n)}$ is the probability that the i-th guess is the first correct guess. Let $Q_s^{(n)} = \sum_{i=1}^s q_i^{(n)}$, the probability that in coordinate n one of the first s guesses is correct. It is rather clear that $Q_s^{(n)}$ is a monotonely increasing sequence as a function of n (interpretation: it cannot harm to know more). Let us formalize and prove this fact:

3.2 Definition. *Let X be a set with n elements, let $P = (p_i)$ and $Q = (q_i)$ be probability distributions defined on X. Order the probabilities such that $p_1 \geq \cdots \geq p_n$ and $q_1 \geq \cdots \geq q_n$.*
We say P **dominates** *Q if for every s we have $\sum_{i=1}^s p_i \geq \sum_{i=1}^s q_i$.*

3.3 Lemma. *The distribution $q_i^{(n+1)}$ dominates $q_i^{(n)}$.*

Proof. We have to show $Q_s^{(n+1)} \geq Q_s^{(n)}$. Consider a matrix whose rows are indexed by the signals σ of length n, whose columns are indexed by the letters S and with entries $p(\sigma, S)$. By definition $Q_s^{(n+1)}$ is obtained by choosing the s largest entries in each row and summing up over all rows. By definition $Q_s^{(n)}$ can be written in the same way, as a sum over all rows of this matrix of s entries in each row. This shows the inequality. $\square$

It follows that the sequence $Q_i^{(n)}$ converges for every i. This shows that the $q_i^{(n)}$ converge for every i. Put $q_i^\infty = lim_n q_i^{(n)}$.

Once again: we imagine frequency tables of standard English text to exist somehow. If that were the case, one would proceed as follows, for example in case $n = 15$ of Shannon's example: we are given the beginning section of the text, a signal σ of length 14 :

`There is no re`

The (imaginary) table would tell us what the conditional probabilities $p(S|\sigma)$ are for all the letters S in standard English text (whatever that may be). If for example the letter a was most likely to follow we would write $j(1) = a$ and our first guess would be a. If unsuccessful we would try the second most likely letter $j(2)$ of probability $p(j(2)|\sigma)$ and so on. The test person is supposed to approximate this procedure. We hope that his/her feeling comes close to corresponding to the (imaginary) table entries. In the case at hand the entry in the compressed text is 17, indicating that the letter v, which happened to be correct, is the 17-th most likely letter, considering all texts starting as above.

We would have to conduct this experiment with, say, 100 texts of length 15. Let a_i be the number of texts such that the compressed text has a last entry of i. Then $i \leq 27$ and $a_i/100$ is the approximation to $q_i^{(15)}$ that the experiment produces. This is what Shannon did. In order to get an idea what F_{15} may be it needs to be bounded from above and below by expressions involving the probabilities $q_i^{(15)}$.

Bounds on F_n

3.4 Theorem. $\sum_{i\geq 2} i \cdot (q_i^{(n)} - q_{i+1}^{(n)})log(i) \leq F_n \leq H(q_i^{(n)})$

Consider first the upper bound. Both expressions can be written as expectations (sums over all σ with weights $p(\sigma)$). It suffices to show the inequality for the second factor, for given σ. We have to show the following:

$$-\sum_S p(S|\sigma)log(p(S|\sigma)) \leq -\sum_i p(j(i)|\sigma)log(q_i^{(n)}).$$

This is a direct consequence of the log sum inequality Proposition 1.13. The lower bound is more interesting. It follows from a general lemma given by Shannon.

3.5 Lemma. *Let a probability space X with distribution p_i be given such that $p_1 \geq p_2 \geq \ldots, p_n$. Then*

$$H(X) = H(p_1, p_2, \ldots, p_n) \geq \sum_{i=2}^{n} i(p_i - p_{i+1})log(i).$$

Proof. We define probabilities $p(i,j), i,j = 1,2,\dots,n$. Here $p(i,j) = 0$ if $i > j$ and $p(i,j) = p_j - p_{j+1}$ otherwise. Write these numbers in a quadratic array. Here is case $n = 4$:

$p_1 - p_2$	$p_2 - p_3$	$p_3 - p_4$	p_4
0	$p_2 - p_3$	$p_3 - p_4$	p_4
0	0	$p_3 - p_4$	p_4
0	0	0	p_4

The row sums are $\sum_{j=1}^n p(i,j) = \sum_{j=i}^n (p_j - p_{j+1}) = p_i$ (here we put $p_{n+1} = 0$). The column sums are $\sum_{i=1}^n p(i,j) = j \cdot (p_j - p_{j+1})$. Interpret the $p(i,j)$ as probabilities for a random variable, let X corespond to the rows and Y to the columns. Recall the basic property $H(X) \geq H(X|Y)$ (see Theorem 1.18). As column i occurs with probability $i(p_i - p_{i+1})$ and has i nonzero (and equal) entries, we obtain

$$H(X|Y) = (p_1 - p_2) \cdot 0 + 2(p_2 - p_3)log(2) + 3(p_3 - p_4)log(3) \dots$$

□

In order to prove the lower inequality of Theorem 3.4 write
$F_n = \sum_\sigma p(\sigma) \sum_S H(S|\sigma)$, apply Theorem 3.4 to $H(S|\sigma)$ and sum up:

$$F_n \geq \sum_\sigma p(\sigma) \sum_i i{\cdot}log(i)(p(j(i)|\sigma) - p(j(i+1)|\sigma)) = \sum_i i{\cdot}log(i)(q_i^{(n)} - q_i^{(n+1)}).$$

It can also be proved that both the upper bound and lower bound on F_n as given in Theorem 3.4 are monotonely decreasing.

Conclusion

The lower bound on F_n is an application of Lemma 3.5, which is a lower bound on the entropy of a probability space. This lower bound is linear in the probabilities p_i. In fact, Lemma 3.5 states that $H(p_1, p_2, ..., p_n) \geq \sum_{i=1}^n a_i \cdot p_i$, where $a_i = i \cdot log(i) - (i-1) \cdot log(i-1)$. Equality is reached only when the probabilities have an unrealistic special form. The upper bound by contrast reaches equality in the limit. It has to be expected that the true value of the entropy is closer to the upper bound than to the lower.

Shannon carried out an experiment with one test person and 100 samples of length 15 from the book *Jefferson the Virginian* by Dumas Malone. A similar test was carried out with samples of length 100. As predicted, the values for the upper and lower bounds on F_n decreased with n (modulo sampling errors). For $n = 15$ the upper bound was a little larger than 2 bit, the lower bound about 1.2 bit. For $n = 100$ the upper bound was 1.3, the lower bound was 0.3. We expect the true value of the entropy to be between these bounds, not very close to either of them, but closer to the upper bound. Shannon's conclusion is that the entropy of standard literary English may be around 1 bit per letter, with a corresponding redundancy of roughly 75%. There seems to exist a vast literature on this subject. A slightly different approach is used by Cover-King [6], who arrive at an estimate of 1.34 bit per letter. This 1978 paper is also valuable because it quotes and comments on numerous other papers. The subject is briefly treated in the book by Cover-Thomas [7].

Chapter 4

Data compression

In most of this chapter we follow Hankerson-Harris-Johnson [11]. For the sake of illustration let us consider a source which at each unit of time emits one of the letters a,b,c,d. We want to compress the following text of length 40 :

`aadda | abacb | abcaa | aacdb | abacc | aaabb | aaccc | adaba`

Single letter Huffman coding

Assume the relative frequencies of letters are known and the text corresponds exactly to those frequencies:

$$p_a = 0.5,\ p_b = p_c = 0.2,\ p_d = 0.1$$

It is by now an easy exercise to find a Huffman code corresponding to those probabilities:

$$f_H(a) = 1,\ f_H(b) = 00,\ f_H(c) = 011,\ f_H(d) = 010.$$

The average word length is

$$0.5 \times 1 + 0.2 \times (2 + 3) + 0.1 \times 3 = 1.8.$$

This yields the following compressed (encoded) string:

```
110100101 | 100101100 | 10001111 | 1101101000 |
1001011011 | 1110000 | 11011011011 | 10101001
```

Not surprisingly, its length is 72 as compared to length 80 for the naive encoding $a \mapsto 11,\ b \mapsto 10,\ c \mapsto 01,\ d \mapsto 00$. Observe that decompression is easy as we use a prefix code.

Huffman coding of pairs

An additional gain can be expected if we base ourselves not on individual letters but on pairs. Parse the text in 20 pairs. Only 11 pairs do occur. The leader is aa (5 times), followed by ca (3 times) and ab,ac,ba (twice each). The pairs da,bb,db,cb,cc,dd occur once each. This leads to the Huffman problem for probabilities

$$5/20,\ 3/20,\ 2/20,\ 2/20,\ 2/20,\ 1/20 \text{ (six times)}$$

A solution is quickly found again:

$$f_H(aa) = 10,\ f_H(ca) = 110,\ f_H(ab) = 011,\ f_H(ac) = 010,\ f_H(ba) = 001,$$
$$f_H(da) = 0001,\ f_H(bb) = 0000,\ f_H(db) = 11111,\ f_H(cb) = 11110,$$
$$f_H(cc) = 11101,\ f_H(dd) = 11100$$

of average length

$$\frac{1}{20}(5 \times 2 + 9 \times 3 + 2 \times 4 + 4 \times 5) = 65/20 = 3.25$$

which corresponds to an average length of 1.625 per letter. The encoded sequence is

```
10111001000111110 | 0111101001011111 |
011010110100000 | 10111011100001001
```

of length 65.

As we saw in the previous chapter, we can expect that the entropy decreases when we base ourselves on signals of larger length. This leads to a more effective compression, in particular in the case of natural languages. One disadvantage is that we need to have a priori information on the frequencies. This can be avoided:

Adaptive Huffman coding

The idea is to use the frequencies of letters in the text as we scan it. In order to get started we use a naive encoding based on the assumption that all letters are equally probable: $a \mapsto 00,\ b \mapsto 01,\ c \mapsto 10,\ d \mapsto 11$. The first letter of our text is a. We determine a Huffman code based on multiplicities $1, 0, 0, 0$. A superficial problem is that the decoder has to reach the same result as the encoder, so the calculation of the Huffman code needs to be standardized. We use a total ordering $a > b > c > d$ and observe the rules

- In the initial column: letters of higher frequencies are written higher than letters of lower frequencies.
- In the case of equal frequencies the total ordering $a > b > c > d$ decides on the position in the starting column.
- From the second column on we are lazy: the ordering is changed only when the frequencies force us to do that.
- When working our way back, assigning bitstrings to the nodes, we always assign the final 0 to the lower and 1 to the higher branch.

This standardizes the procedure. The first a is encoded as 00 under the naive encoding. The first Huffman step yields

$$f_H(a) = 1,\ f_H(b) = 01,\ f_H(c) = 001,\ f_H(d) = 000.$$

The second letter of the sample text is therefore encoded as $a \mapsto 1$. The frequencies change as a now occurred twice. Clearly this does not change the Huffman code, so the third letter is encoded $d \mapsto 000$. Now that d occurred it will move into second position behind a in the starting line-up for the Huffman algorithm. The new Huffman code is

$$f_H(a) = 1,\ f_H(d) = 01,\ f_H(b) = 001,\ f_H(c) = 000.$$

It encodes the fourth letter as $d \mapsto 01$. The Huffman code remains stable and we encode $a \mapsto 1$. As this only increases the lead of letter a the code remains unchanged again and the sixth letter is encoded $a \mapsto 1$, then $b \mapsto 001, a \mapsto 1, c \mapsto 000, b \mapsto 001$. The tenth letter changes the Huffman code. We have now

$$f_H(a) = 1,\ f_H(b) = 00,\ f_H(d) = 011,\ f_H(c) = 010$$

and encode the following letters as $a \mapsto 1, b \mapsto 00$. A new Huffman code is obtained:

$$f_H(a) = 1,\ f_H(b) = 01,\ f_H(d) = 001,\ f_H(c) = 000$$

Letter thirteen maps to $c \mapsto 000$, and now the letters happen to be back to the initial ordering in the first column. For the first time a is not encoded as $1:$

$$f_H(a) = 0,\ f_H(b) = 10,\ f_H(c) = 111,\ f_H(d) = 110$$

It is clear how this continues. The compressed sequence is

```
001000011 | 10011000001 | 10000001 | 1101101000 |
1001011011 | 0110000 | 11011011011 | 001000110
```

Its length is 75.

Arithmetic coding

Again we assume we have information on the relative frequencies of letters. As all calculations are base 2 we probably wish to replace the actual frequencies by fractions with a power of 2 in the denominator Let us use frequencies

$$p_a = \frac{1}{2} = 0.1,\ p_b = p_c = \frac{3}{16} = 0.0011,\ p_d = \frac{1}{8} = 0.001.$$

The idea is the following: Start from the unit interval $[0, 1)$. Each letter is assigned a subinterval, closed to the left and open to the right, whose length is proportional to the frequency of the letter and such that those subintervals partition the unit interval. In our case those subintervals are

interval	length l	representative
$A(a) = [0,\ 0.1)$	0.1	0
$A(b) = [0.1,\ 0.1011)$	0.0011	1
$A(c) = [0.1011,\ 0.111)$	0.0011	11
$A(d) = [0.111,\ 1)$	0.001	111

Recall that we use the duadic system. For example 0.1 stands for 1/2. We subdivided the unit interval $[0, 1)$ into four subintervals whose lengths equal the probabilities of the letters. In the right column we noted a representative for each of those subintervals. For example, representative 11 in the $A(c)$ row really stands for $0.11 \in A(c)$. Here the bitstring describing the representative has been chosen as short as possible.

Next we assign intervals to pairs of letters using the same principle. The intervals $A(aa), A(ab), A(ac), A(ad)$ partition $A(a)$ and the lengths of the subintervals correspond to the additional letters: the lengths satisfy $l(A(ai)) = l(A(a) \times p_i$. In practice it is probably easiest to compute those lengths first. The left endpoint of each subinterval corresponding to pairs equals the right endpoint of its predecessor. The right endpoint is obtained by adding the length:

interval	length l	representative
$A(aa) = [0,\ 0.01)$	0.01	0
$A(ab) = [0.01,\ 0.01011)$	0.00011	01
$A(ac) = [0.01011,\ 0.0111)$	0.00011	011
$A(ad) = [0.0111,\ 0.1)$	0.0001	0111
$A(ba) = [0.1,\ 0.10011)$	0.00011	1
$A(bb) = [0.10011,\ 0.10100001)$	0.00001001	101
$A(bc) = [0.10100001,\ 0.1010101)$	0.00001001	10101
$A(bd) = [0.1010101,\ 0.1011)$	0.0000011	101011
$A(ca) = [0.1011,\ 0.11001)$	0.00011	11
$A(cb) = [0.11001,\ 0.11010001)$	0.00001001	1101
$A(cc) = [0.11010001,\ 0.1101101)$	0.00001001	11011
$A(cd) = [0.1101101,\ 0.111)$	0.0000011	110111
$A(da) = [0.111,\ 0.1111)$	0.0001	111
$A(db) = [0.1111,\ 0.1111011)$	0.0000011	1111
$A(dc) = [0.1111011,\ 0.111111)$	0.0000011	11111
$A(dd) = [0.111111,\ 0.1)$	0.000001	111111

It is clear how this continues: intervals $A(aaa), A(aab), A(aac), A(aad)$ partition $A(aa)$ and the relative lengths correspond to the letter frequencies. In other words, if we magnify the interval $A(aa)$ such that it becomes the unit interval, then the subdivision defined by $A(aaa), A(aab), A(aac), A(aad)$ corresponds exactly to the original subdivision of the unit interval. This is the general principle. Whenever we pass to the next subdivision of an interval we generate a clone of the unit interval and its original subdivision, on a smaller scale. In each interval we choose a short representative.

Here is how compression (encoding) works: imagine we wish to compress a long document (signal $x = (x_1, \dots, x_N)$) consisting of N letters. The compressed document has two parts:the number N and the representative of interval $A(x)$. For example, the message bc is encoded as $(2, 10101)$. The decompresser looks up which of the intervals corresponding to pairs has 10101 as its representative. As this is interval $A(bc)$ the decompression yields the correct result bc. Observe that the most probable signal aa of length 2 corresponds to the longest interval $A(aa)$ and to the shortest representative 0.

Lempel-Ziv compression

We know that Huffman compression is optimal if only short range effects are taken into account. If we believe that the entropy of the English language is essentially determined by sequences of 4 letters, then the best thing to do is to use Huffman compression based on signals of length 4. However, we saw in Chapter 3 that this is not justified. Shannon's experiments and arguments revealed that the entropy per letter goes down a lot of we take long range effects into account. In practical terms this means that it is not futile to search for methods which tap this source of redundancy. Huffman coding would theoretically still be best but it is of course hopeless in reality. We cannot manage the statistics of all sequences of 100 letters, say. There are 27^{100} such sequences after all.

Two successful attempts to use long-range effects for data compression have been found by Lempel and Ziv [13, 14]. Lucky [17]: *For q quarter of a century Huffman coding remained relatively unchallenged, as the ultimate form of text compression...Now only in the most recent years has a new compression algorithm, the Lempel-Ziv technique, taken over nearly completely from Huffman.*

We start with an easy example illustrating the principle of LZ78: let the text be

```
a a b b a b a b a a b a a
```

Encoding proceeds letter by letter, building up a tree where the vertices are numbered $0, 1, 2, \dots$ and the edges (each edge connects two vertices) are indexed by letters. Starting point is always the **root** of the tree, which is numbered 0. The first text letter a leads to an edge starting at the root with endpoint 1. The second letter a of the text illustrates the principle: starting from the root it is checked what is the longest section of text, starting from our second letter, which can be read following the edges. In our the a is found and that is all. The part of the compressed text we obtain so far is:

$$(0, a)\ (1, b).$$

The decoder (the uncompress algorithm) reads this as: at first take the empty string, corresponding to index 0 (the root), follow it by a. Then read the labels of edges, starting from the root, until vertex 1 is reached, followed by b. The gives aab so far and leads to a new edge of the tree, starting from vertex 1, ending at the new vertex 2, and labelled b. So far our memory, the tree consists of a branch labelled a from 0 to 1 and a branch labelled b connecting 1 and 2.

Now the fourth letter b needs to be encoded. As there is no edge labelled b originating at the root yet we now construct one. Its endpoint is 3 and the next encoding pair is $(0, b)$. The next step is the first success of compression: we want to encode $aba\ldots$ and we find a path, from the root to vertex 2, with labels ab. This means we encode the segment aba as as $(2, a)$ and construct a new vertex 4, joined to 2 by an edge labelled a. The principle should be clear by now: the next section is ba, encoded as $(3, a)$, which leads to vertex 5 in the usual manner. Finally we encode the last segment $abaa$ as $(4, a)$. By now the tree looks as in Figure 4.1.

The compressed text is

$$(0, a)\ (1, b)\ (0, b)\ (2, a)\ (3, a)\ (4, a)$$

and it should be clear how decompressing works. The original text has 13 letters, the compression has 6 letters and 6 numbers. This is of course only a toy example. It seems that LZ78 works better than Huffman on ordinary English text. Haykin [12] mentions a compression rate of about 55% for LZ78, whereas Huffman apparently compresses by about 43%.

There are many practical issues which we do not want to discuss. One concern is with the size of the memory. The tree grows with the text. What happens when it gets too large? We may flush it and start from the beginning, or we may freeze at a certain point. LZ77 is a variant which needs much less memory. We illustrate the principle with an example given in [11]. The text is

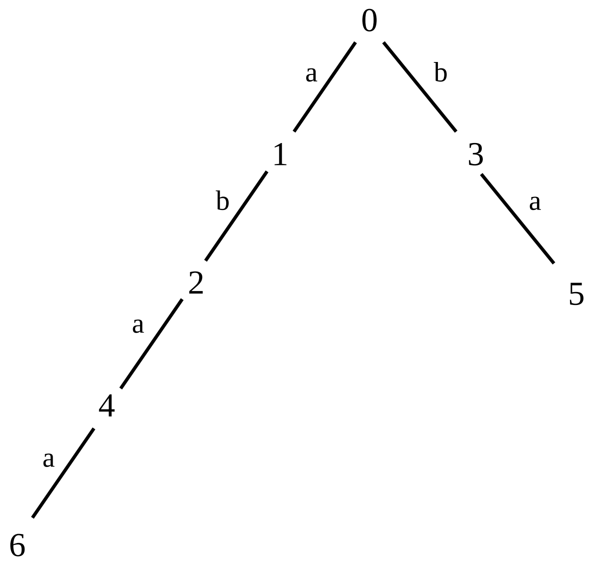

Figure 4.1: Our tree

```
She sells sea shells by the seashore
```

The idea is the following: instead of building up a large memory we use only the section of the text in a window of fixed size. Assume we have reached the point where

```
                    ells by the seashore
```

needs to be coded and the window extends all the way to the beginning of our text. At this point the encoder will search for the longest section in the window (to the left) which agrees with the text we want to encode. In the example we find a section *ells* of length 5 (the space also counts) 11 positions to the left. The next entry of the compressed text is then the pair $(11, 5)$. When decompressing this is interpreted as: go 11 to the left and read the segment of length 5 starting there.

Exercises 4

4.1. *Consider a source that produces the* 4 *letters a,b,c,d. In this and the following problems we use the following example string produced by the source:*

```
aadda | abacb | abcaa|aacdb | abacc | aaabb | aaccc | adaba
```

Compute $H(0.5,\ 0.2,\ 0.2,\ 0.1)$.

4.2. *Find the Huffman code for probabilities* $0.5,\ 0.2,\ 0.2,\ 0.1$ *and the average word length.*

4.3. *Use the Huffman code from the preceding problem to compress the example string.*

4.4. *Determine the relative frequencies of pairs in the example string, find the corresponding Huffman code based on those* 16 *probabilities and use this to compress the example string.*

4.5. *Compress the example string using adaptive Huffman coding (based on letters).*

4.6. *Consider a source producing the letters* a, b, c, d, e *with relative frequencies* $0.3,\ 0.2,\ 0.2, 0.2, 0.1$. *Find* $H(0.3, 0.2, 0.2, 0.2, 0.1)$, *the Huffman code and the average wordlength.*

4.7. *Encode the source text*

```
a b c b a c e a d d
```

produced by the source above, using adaptive Huffman encoding.

Chapter 5

Steganography and covering codes

It is the aim of steganographic techniques to obscure the existence of a message from the adversary (think of invisible ink). Imagine we wish to hide a small picture in a large picture in such a way that no third person will suspect the existence of the hidden picture. The big picture is given to us. It consists of a lot of pixels. If we modify a pixel, then only its least significant bit that is we "extract" at most one bit from each pixel. Also, the image to be hidden has already been encoded as a bitstring somehow and so we want to hide a bitstring within a bitstring.

A natural idea to use is known as **block coding.** The pixels are divided into blocks of N each (say: $N = 7$). We scan the pixels of our picture and find for example a block 0001001 (of least significant bits of pixels). Maybe we wish to extract 3 bits of the hidden picture from each block of pixel bits. For example, the next block of bits to hide may be 010. This means we wish to define a function f mapping bitstrings of length 7 to bitstrings of length 3. This function needs to be shared between sender and receiver before communication begins (sending the large picture with the hidden picture inside).

Now, this cannot work. As we scan the pixels of the picture there is no telling which 7-tuples will turn up and likewise there is no telling which triples we want as image. As f is fixed once and for all we would have to be extemely lucky. In fact the little picture would have to be hidden already, by coincidence. No, the idea was to change the least significant bits of the pixels a little bit. So, if for example it should happen that $f(0001001) = 111$ we will have to replace the 7-tuple $x = 0001001$ by x' such that $f(x') = 010$ as required. The major concern is¿ we want to minimize the number of pixels that need to be changed. In our example, here is the best choice:

$$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (x_1+x_4+x_5+x_7, x_2+x_4+x_6+x_7, x_3+x_5+x_6+x_7).$$

This function can be expressed in terms of a matrix:

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

Here we calculate mod 2, that is $1 + 1 = 0$. In computer science this is known as XORing. The entries of $f(x)$ are the dot products of x and the rows of H. It suffices to change any given 7-tuple x in at most 1 coordinate in order to obtain x' such that $f(x')$ takes on any given triple as a value. Moreover the coordinate where the entry needs to be changed is uniquely determined.

For example, $f(0011001) = 000$. Assume we wish $y = 111$ to be the image. In order to obtain this we have to change the last coordinate: $x' = 0011000$ satisfies $f(x') = y$. If we wanted 010 as an image we would have replaced x by 0111001 to obtain the desired image. To sum up: f is a mapping from binary 7-tuples x to triples y sarisfying the following:

- For any given x and y we can find x' differing from x in at most 1 coordinate such that $f(x') = y$.

For our steganographic purpose this means that at most 1 out of 7 pixels of the original picture needs to be changed in its least significant bit in order to hide the small image. Hopefully the number of changes will then be small enough to go undetected. In mathematical terms we call such a function f a **covering function** $COV(1, 7, 3)$. This leads to a general definition:

5.1 Definition. *A* **covering function** $COV(\rho, N, n)$ *is a mapping f from bitstrings x of length N to bitstrings y of length n satisfying the following property: for every x, y there exists some x', which differs from x in at most ρ coordinates and satisfies $f(x') = y$.*

The main question is now: when do $COV(\rho, N, n)$ exist and how can they be described?

We see that it is important in how many positions x needs to be changed. This is in fact one of the basic notions of information theory and coding theory, the **Hamming distance.**

5.2 Definition. *Let $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ be bitstrings in $\mathbb{F}_2^n$. The* **distance** *(or* **Hamming distance***) between x and y is*

$$d(x, y) = \textit{number of coordinates } i \textit{ where } x_i \neq y_i.$$

Here are some examples:

$$d(0000, 1111) = 4,\ d(00110, 00101) = 2,\ d(111111, 001100) = 4.$$

Things get even easier when we use the weight.

5.3 Definition. *The* **weight** *$wt(x)$ of the bitstring $x \in \mathbb{F}_2^n$ is the number of nonzero coordinates in x.*

Here are some examples:

$$wt(0000) = 0,\ wt(1111) = 4,\ wt(00110) = 2,\ wt(001101) = 3.$$

The weight of a bitstring is its distance from the all-0 bitstring.

The Hamming distance is also called the Hamming metric. The general notion of a metric is widely used in mathematics. Here is the definition:

5.4 Definition. *Let X be a set. For every pair $x \in X,\ y \in X$ let a real number $d(x, y)$ be given (the* **distance** *from x to y).*
The function d is called a **metric** *if the following are satisfied:*

- *$d(x, y) \geq 0$ for all x, y.*
- *$d(y, x) = d(x, y)$ for all x, y.*
- *$d(x, y) = 0$ if and only if $x = y$.*
- *$d(x, z) \leq d(x, y) + d(y, z)$ for all x, y, z.*

The last requirement is the most important. It is known as the **triangle inequality.** A famous metric is the Euclidean metric in Euclidean space. If for example $x = (x_1, x_2)$ and $y = (y_1, y_2)$ are two points in the plane, then their Euclidean distance is $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

5.5 Theorem. *The Hamming distance is a metric on* $\mathbb{F}_2^n$.

The properties of Definition 5.4 are obviously satisfied.

5.6 Definition. *A (binary)* **code** *of length* N *is a subset of* $\mathbb{F}_2^N$.

The term *binary* is added as codes can be defined over arbitrary alphabets. We restrict to the alphabet $\mathbb{F}_2$. These are binary codes.

The notion of a covering function is closely related to a famous kind of objects in coding theory, covering codes. In order to see this reconsider Definition 5.1 from a different angle: Fix y and its inverse image $f^{-1}(y)$, a subset of the space of all bitstrings of length N. Definition 5.1 says that $f^{-1}(y)$ has to satisfy the following: every bitstring of length N has Hamming distance $\leq \rho$ from some element of $f^{-1}(y)$. In other words: the union of the Hamming balls of radius ρ around the elements of the code $f^{-1}(y)$ cover the space of all bitstrings. If they satisfy the property in question they are known as **covering codes.**

5.7 Definition. *A code* $\mathcal{C} \subseteq \mathbb{F}_2^N$ *is a* **covering code** *of* **covering radius** ρ *if every bitstring of length* N *is at Hamming distance* $\leq \rho$ *from some element of* $\mathcal{C}$.

Another way of saying this is: the Hamming spheres of radius ρ around the codewords must cover the whole space. If we want to illustrate this with the Euclidean metric we get a picture like Figure 5.1.

An even more popular parameter in coding theory and information theory is the **minimum distance** d of a code, the smallest Hamming distance between two different codewords. If we choose the largest number e such that $2e < d$ as the radius of Hamming spheres, the picture of Figure 5.2 is obtained:

If f is a $COV(\rho, N, n)$, then for every bitstring y of length n we have that $f^{-1}(y)$ is a covering code of covering radius ρ. As we let y vary we see that the space of N-tuples is partitioned into 2^n such covering codes. Obviously this description is equivalent to Definition 5.1:

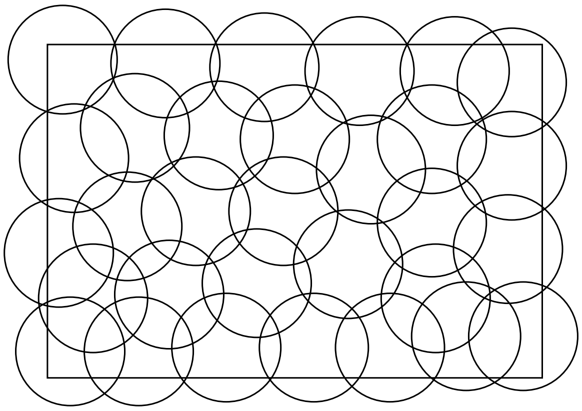

Figure 5.1: A covering code

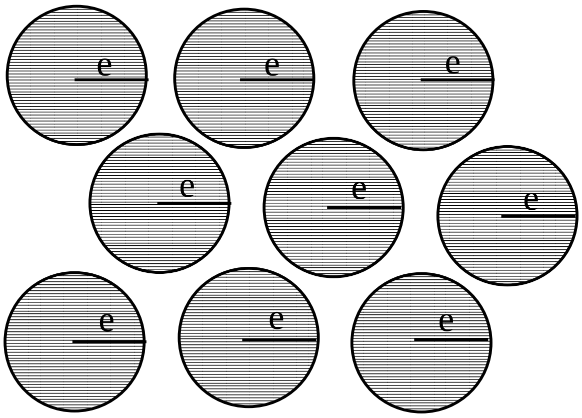

Figure 5.2: Non-overlapping Hamming spheres

5.8 Theorem. *A $COV(\rho, N, n)$ exists if and only if the space of all bitstrings of length N can be partitioned into 2^n covering codes of radius ρ.*

An advantage of Theorem 5.8 is that it makes it possible to use coding theory, a highly developed discipline. For example, the $COV(1, 7, 3)$ that we used as an illustration is based on a famous code, the **Hamming code,** which in fact is member of a large family of codes. To give just one further example, the single most famous code, the **binary Golay code,** is a $COV(3, 23, 11)$.

We want to use Shannon entropy to obtain a bound on the possible parameters of covering functions and covering codes. So let $\mathcal{C}$ be a covering code of radius ρ. What can we say about the number of its members? Clearly we are looking for a lower bound. Pick an element $x \in \mathcal{C}$ and count the bitstrings at distance i from x. The discussion of the binomials in Chapter 1 shows that this number is $\binom{N}{i}$. The number of bitstrings at distance $\leq \rho$ from x is therefore

$$\binom{N}{0} + \binom{N}{1} + \cdots + \binom{N}{\rho} = \sum_{i=0}^{\rho} \binom{N}{i}.$$

5.9 Definition. *The volume of the Hamming ball of radius ρ, equivalently the number of vectors at distance $\leq \rho$ from an element of $\mathbb{F}_2^N$, is*

$$V(\rho, N) = \sum_{i=0}^{\rho} \binom{N}{i}.$$

If we add up all those numbers for all $x \in \mathcal{C}$ we must count each bitstring at least once:

$$|\mathcal{C}| \times V(\rho, N) \geq 2^N.$$

This is a lower bound:

5.10 Theorem. *Let $\mathcal{C}$ be a covering code of bitstrings of length N with covering radius ρ. Then*

$$|\mathcal{C}| \geq 2^N / V(\rho, N).$$

In the case of covering functions this expression simplifies. At least one of the 2^n covering codes forming the partition has $\leq 2^{N-n}$ codewords. We can write this for $|\mathcal{C}|$ in Theorem 5.10 and cancel the common factor 2^N.

5.11 Theorem. *If $COV(\rho, N, n)$ exists, then*

$$V(\rho, N) \geq 2^n.$$

How about our example, the covering function $COV(1, 7, 3)$? Theorem 5.11 states that $COV(1, N, 3)$ satisfies $N + 1 \geq 8$, so the Hamming code construction is optimal. Codes which satisfy the bound of Theorem 5.11 with equality are known in coding theory as **perfect codes.** We conclude that our Hamming code is a perfect code and that our $COV(1, 7, 3)$ has extremal properties: it corresponds to a partition of the space of 7-tuples into perfect covering codes. How about $COV(1, N, n)$ for general n? Theorem 5.11 shows $N \geq 2^n - 1$. Recall how $COV(1, 7, 3)$ was constructed in terms of matrix H; the columns of H are exactly all nonzero binary triples. We can imitate this construction for all n. In case $n = 4$ the matrix is then

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

We have written 15 columns: each nonzero quadruple occurs once. The function is defined in terms of dot products with the rows of H :

$$f(x) = (x_1 + x_5 + x_6 + x_7 + x_{11} + x_{12} + x_{13} + x_{15}, ...)$$

We obtain a $COV(1, 15, 4)$.

Another situation where perfection seems possible occurs when we apply our bound in case $\rho = 3, n = 11$. The bound states

$$1 + N + \binom{N}{2} + \binom{N}{3} \geq 2048.$$

For $N = 23$ the left side happens to be $2^{11} = 2048$. This is quite a coincidence. In fact the corresponding code and the covering function $COV(3, 23, 11)$ exist. The code is the binary Golay code.

For our application we are interested in constructing covering functions $COV(\rho, N, n)$ such that the **change rate** $c = \rho/N$ is small (so the existence of a hidden picture goes undetected) and the **information rate** $R = n/N$ is large (so we can hide a relatively large picture). Clearly, both rates are bounded by 1 and the aims are in conflict. In practice one would probably fix an upper bound c on the change rate ρ/N. For example, $x = 0.1$ guarantees that not more than one tenth of all pixels need to be changed. The problem is then to construct $COV(\rho, N, n)$ such that the information rate $\rho/N \leq 0.1$ and $y = n/N$ as large as possible.

5.12 Definition. *Let* $0 \leq c, R \leq 1$. *We say the pair* (c, R) *is* **asymptotically reachable** *by covering functions if there is an infinite family of covering functions* $COV(\rho_i, N_i, n_i)$ *such that all parameters* ρ_i, N_i, n_i *go to infinity and*

$$lim_i(\rho_i/N_i) \leq c,\ lim_i(n_i/N_i) \geq R.$$

Use the bound

$$2^n \leq \sum_{j=0}^{\rho} \binom{N}{j}.$$

Now take the base 2 logarithm and divide by N on both sides. On the left side this yields R, the rate. On the right side the dominating term is $\binom{N}{\rho}$. Theorem 1.9 from our first chapter shows what the result is on the right side: $h(c)$. We have the asymptotic inequality $R \leq h(c)$.

5.13 Theorem. *Let* $0 \leq c \leq 1/2$ *and* $0 \leq R \leq 1$. *If* (c, R) *is asymptotically reachable by covering functions then*

$$R \leq h(c).$$

Here h is the good old binary entropy function, see Figure 1.1.

Reconsider our $COV(1, 7, 3)$. Its change rate is $c = 1/3$. The information rate is $R = 3/7 \approx 0.43$. The upper bound on the information rate from Theorem 5.13 is $h(1/3) = log(3) - 2/3 \approx 0.91$, still far away.

The following is proved in one of our problems (which one?):

5.14 Proposition. *If* $COV(\rho, N, n)$ *exists, then* $(\rho/N, n/N)$ *is asymptotically reachable by covering functions.*

It follows that $COV(\rho, N, n)$ for $\rho \leq N/2$ can exist only if $n/N \leq h(\rho/N)$.

The definition of covering functions and their description in terms of covering codes is in [4, 8, 3]. The linear case war rediscovered recently by Galand-Kabatiansky [10].

As a research challenge, here is what seem to be the best known bounds on N for $COV(\rho, N, n)$ in the range $\rho \leq 5,\ n \leq 25$. If the entry consists of one number, then this is the minimum N. If the entry has the form $(, N)$, then N is an upper bound and we do not know of a reasonable lower bound. The entry is in boldface if the construction uses non-linear codes.

$n\backslash\rho$	1	2	3	4	5
2	3	2			
3	7	4	3		
4	15	5	5	4	
5	31	9	6	6	5
6	63	$(12,13)$	7	7	7
7	127	$(16,19)$	11	8	8
8	255	**23**	$(13,14)$	9	9
9	511	$(,39)$	$(16,18)$	13	10
10	1023	$(,51)$	$(20,22)$	$(14,16)$	11
11		$(,\mathbf{72})$	23	$(17,19)$	15
12		$(,\mathbf{95})$	$(30,\mathbf{31})$	$(19,23)$	$(16,18)$
13		$(,159)$	$(,\mathbf{47})$	$(,25)$	$(,19)$
14		$(,215)$	$(,63)$	$(,29)$	$(,23)$
15		$(,\mathbf{274})$	$(,71)$	$(,36)$	$(,27)$
16		$(,\mathbf{383})$	$(,\mathbf{94})$	$(,\mathbf{46})$	$(,31)$
17		$(,639)$	$(,126)$	$(,62)$	$(,35)$
18		$(,863)$	$(,\mathbf{127})$	$(,\mathbf{74})$	$(,41)$
19		$(,\mathbf{1062})$	$(,\mathbf{191})$	$(,82)$	$(,\mathbf{46})$
20		$(,\mathbf{1535})$	$(,254)$	$(,90)$	$(,\mathbf{54})$
21		$(,2559)$	$(,308)$	$(,122)$	$(,\mathbf{63})$
22		$(,3455)$	$(,\mathbf{382})$	$(,\mathbf{144})$	$(,\mathbf{82})$
23		$(,\mathbf{4167})$	$(,\mathbf{510})$	$(,\mathbf{158})$	$(,94)$
24		$(,\mathbf{6143})$	$(,\mathbf{511})$	$(,190)$	$(,104)$
25		$(,\mathbf{10,239})$	$(,767)$	$(,238)$	$(,120)$

The smallest open problem is the existence of a $COV(2,12,6)$. If it existed there would have to exist a covering code of length 12 and radius 2 with $M \leq 64$ codewords. The current lower bound on M is 62, the smallest number of codewords for which such a covering code is known to exist is $M = 78$.

Exercises 5

5.1. *The code consisting of the all-*0 *string and the all-*1 *string is known as the* **repetition code.** *Determine the covering radius of the repetition code of length* N *for* $N = 3, 4, 5, 6, 7$ *and for general* N.

5.2. *Construct explicitly* $COV(2, 4, 3)$.

5.3. *Explain why* $COV(m, 2m, 2m - 1)$ *exist for all* $m \geq 1$.

5.4. *Explain why in Theorem 5.13 values* $c > 0.5$ *need not be considered.*

5.5. *Prove the following: if* $COV(\rho, N, n)$ *exists, then* $COV(2\rho, 2N, 2n)$ *exists.*

5.6. *Prove the following generalization of the previous problem: if* $COV(\rho, N, n)$ *exists, then* $COV(i \cdot \rho, i \cdot N, i \cdot n)$ *exists for all natural numbers* i.

Chapter 6

Communication channels

The problem is secure communication over noisy channels. The language adopted is rather suggestive: there is an information source, the resulting information is encoded, the encoded information is sent via the channel, the received information is then decoded and hopefully we recover the original information. In principle this may sound frightening as it is clear what all those metaphors mean in practice. Fortunately we understand already most of the above.

A source of information simply is a probability space, see Definition 1.1. Data compression as in Chapter 4 is a special case of encoding. In Chapter 4 the intention was to remove redundancy as far as possible. In the present chapter we will focus on a different (and more complex) purpose. There remains the question: what is a channel?

Transmitting pictures from space

Assume we wish to transmit a photograph from outer space, like one of the pictures of Saturn taken by the Voyager spacecrafts in the early 1980s (*Viger* for Star Trek buffs). The picture is divided into 800×800 pixels, each pixel is assigned one of $256 = 2^8$ degrees of brightness. The brightness of a pixel is thus represented by a bitstring of length 8 and the total black and white picture consists of $800 \times 800 \times 8$ bits. As the picture really is in color, the same photo is transmitted three times, each time through a different color filter. The full color picture will thus be represented by a bitstring of length $3 \times 800 \times 800 \times 8 = 15,360,000$. This is our message. The channel is determined by the properties of space between the spacecraft and the receiver on Earth, above all by the Earth's atmosphere. A certain number of bits will be destroyed. Here we only consider errors of the type that 0 is transmitted and 1 is received or vice versa.

Clearly physical channels like the Earth's atmosphere are far too complex to be handled mathematically. We will always work with idealized models instead. The most general model we want to consider is the following:

6.1 Definition. *A* **discrete memoryless channel** *consists of the following:*

- *two finite sets, the input alphabet A and the output alphabet B.*
- *For every $a \in A$ a probability distribution $p(b \mid a)$ on the $b \in B$.*

It should be clear what the intention is: the elements of A are those which the channel is able to read and transmit, the elements of B are those which come out of the channel on the receiver's end. If we feed the channel symbol a, then $p(b|a)$ is the probability that b will be received. In many cases we will have $A = B$. In general we can describe such a channel as a matrix M with $|A| = a$ rows and $|B| = b$ columns such that the entries in each row sum to 1. In mathematics these are known as **stochastic matrices.** The term *discrete* means that we work with finite sets, *memoryless* describes the fact that the transition probabilities $p(b \mid a)$ do not depend on prehistory. There are in fact more general models which can be used as well.

About the conceptually easiest model for a channel is obtained when input and output alphabet both are $\{0, 1\}$ and there is a fixed probability p for both symbols to be transmitted in error. This is the BSC.

The corresponding stochastic matrix is

$$M = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$

We can of course assume $p < 1/2$. If $p = 1/2$ then the channel output is unrelated to the input, so no information is transmitted. If $p > 1/2$ we would switch the roles of 1 and 0 in the output, say. How much information can be transmitted via this channel, and what does that mean? Naturally we expect the information that can be sent over the channel to decrease as the error probability p increases. If $p = 0$, then there are no errors and we

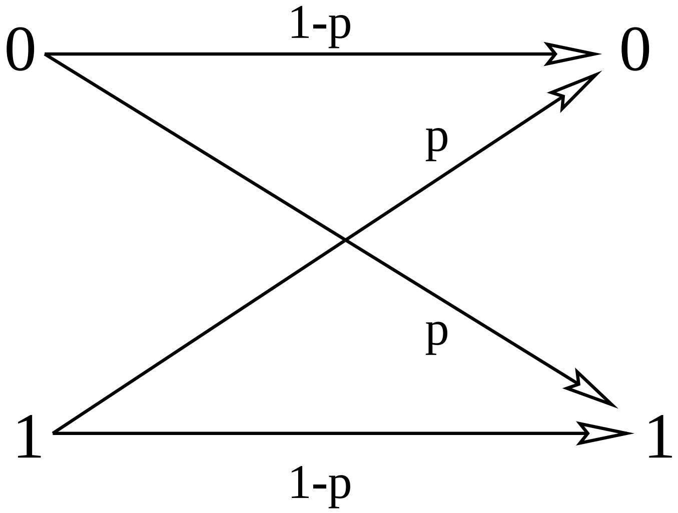

Figure 6.1: The BSC

can transmit 1 output bit per input bit. It turns out that these questions are closely related to a basic notion of information theory, the capacity of a channel.

6.2 Definition. *Let a discrete memoryless channel as in Definition 6.1 be given. For every probability distribution on the input alphabet A the channel defines a distribution on the output alphabet B. Let X be the random variable defined by A and Y the random variable defined by B. The channel capacity is defined as the maximum of $I(X,Y)$ over all probability distributions on A.*

The channel capacity is not always easy to calculate. However, this is not the principal difficulty. In fact the simplest of all channels, the BSC, whose capacity is easy to compute, is widely used.

Consider the BSC. Choosing the input distribution means that we consider the input alphabet, in our case $A = \mathbb{F}_2$, as a random variable. Let x be the probability of 0. We have $H(X) = h(x)$. What is the induced probability distribution on the output alphabet $B = \mathbb{F}_2$? The probability that $Y = 0$ is $x(1-p) + (1-x)p$. It follows $H(Y) = h(x(1-p) + (1-x)p)$. How does this compare to $H(X) = h(x)$? We have that $x(1-p) + (1-x)p$ is a **convex combination** of x and $1-x$. Recall that in general a convex combination of a and b as an expression $f(t) = (1-t)a + tb$, where $0 \leq t \leq 1$.

Typically one thinks of a particle which at time $t = 0$ is at point a, at time $t = 1$ at $f(1) = b$. When t increases the particle moves one the line from a to b. We use p as the time parameter (although it does not change) and see that $y = x(1-p) + (1-x)p$ is between x and $1-x$. Figure 1.1 shows that $h(y) \geq h(x)$

6.3 Proposition. *Consider the BSC with error probability p. If the input entropy is $H(X) = h(x)$, then the output entropy is*

$$H(Y) = h(x(1-p) + (1-x)p) \geq H(X).$$

In order to determine the channel capacity we need to maximize the mutual information

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

Which of these expressions is easier to handle? In fact, by definition of the conditional entropy,

$$H(Y|X) = xh(p) + (1-x)h(p) = h(p)$$

independent of x. This shows that the second of the two expressions above is easiest to use. All we need to do is maximize $H(Y)$. As the output alphabet has only 2 elements we have $H(Y) \leq 1$. So we have to choose x such that $x(1-p) + (1-x)p = 0.5$ Clearly the choice $x = 0.5$ does this.

6.4 Theorem. *The BSC with error probability p has capacity $1 - h(p)$. It is reached when the input letters are equally probable.*

We can also understand the idea behind the definition of the channel capacity. Concentrate on the BSC. The AEP (see the end of Chapter 1 says that, for large n, the typical signals of length n emitted by a source of entropy $H(X)$ can be thought of as a uniform space of size $2^{nH(X)}$. We saw that in the case of the BSC all conditional entropies $H(Y|a)$ are $= h(p)$. Each typical input signal determines $2^{nh(p)}$ typical output signals. All in all there are $2^{nH(Y)}$ typical output signals. Our aim is to choose the maximal possible number M of input signals such that their possible output signals do not overlap. This means that the receiver is able to distinguish those M signals: if the receiver is informed beforehand which M signals are used, then he will be able to determine which one was sent by examining the output. In order for this to be possible we must have

$$M2^{nh(p)}(= M2^{nH(Y|X)}) \leq 2^{nH(Y)}.$$

This means that we wish to maximize $H(Y) - H(Y|X) = I(X,Y)$ and M is then bounded by $2^{nI(X,Y)}$. This gives us another interpretation of the mutual information: if M is the maximal number of distinguishable signals of length n, for large n, then $log(M)/n = I(X,Y)$. This is maximized by the channel capacity.

Another term for *subset of the set of signals of length* n is *binary code of length* n. We see that codes come into the picture again (compare Definition 5.6). In the present context we are interested in two properties of a code, its error-correction capability and its asymptotic size.

6.5 Definition. *Let* $\mathcal{C} \subseteq \mathbb{F}_2^n$ *be a code,* $M = |\mathcal{C}|$. *Its* **asymptotic size** *is* $log(M)/n$.

Our study of the capacity of the BSC led to the following statement:

6.6 Theorem. *Let* $R < 1-h(p)$. *Then we can find a family of codes* $\mathcal{C}_n \subset \mathbb{F}_2^n$ *with* $|\mathcal{C}_n| = M_n$ *such that for* $n \longrightarrow \infty$ *the following hold:*

- $lim(log(M_n)/n) \leq R$.
- *The probability of a decoding error gets arbitrarily small.*

Theorem 6.6 is Shannon's famous **channel coding theorem.** We have not really proved it. In fact what the argument essentially proves is the converse: no reliable transmission is possible beyond the channel capacity. We will prove the channel coding theorem in the next section. As for now, let us interpret it.

Assume a suitable code $\mathcal{C}$ of length n has been found and one of the codewords sent. Denote this codeword by x. Let y be the received bitstring. Write $y = x + err$, where addition is componentwise and mod 2. We see err as the error signal: if err has entry 1 in a coordinate, then this coordinate of x is transmitted in error. If err has entry 0 there, then this coordinate transmits faithfully. By the AEP, the number of typical noise signals is about $2^{nh(p)}$. As $p < 1/2$, noise signals of small weight (a small number of errors) are more probable that those with a large number of errors. The typical error signals consist therefore of those with smallest weight. There are $\sum_{j=0}^{i}\binom{n}{j}$ such signals (bitstrings) of weight $\leq i$. Because of the relationship between binomials and the entropy (Theorem 1.9) we see that the typical error signals are those of weight $\leq i \approx np$. In the language of the Hamming metric this translates as follows: the code $\mathcal{C}$ must have the property that the Hamming spheres around the codewords (these are the most likely received signals when the corresponding codeword was sent) do not overlap. This leads directly to the picture given in Figure 5.2: a good code, admitting reliable communication, is obtained if and only if any two codewords have a large Hamming distance from one another.

6.7 Definition. *An (n, M, d)-code of length n and minimum distance d is a code $\mathcal{C} \subset \mathbb{F}_2^n$ such that $|\mathcal{C}| = M$ and any two different elements of $\mathcal{C}$ (codewords) have Hamming distance of at least d.*

Theorem 6.6 states that good codes of large length exist. It gives no clue how to find those codes. The explicit construction of good codes is the objective of a mathematical theory of its own, **coding theory.**

Informal proof of the channel coding theorem

The argument of this section comes close to a proof. It is essentially the argument used by Shannon and is also used in Ash [2] and Bruen-Forcinto [5]. All proofs of the channel coding theorem use an averaging process over all possible codes of the required size. This seems to be unavoidable. Here is the argument:

Let $\kappa = 1 - h(p)$ be the capacity and $R < \kappa$. Let the code C of size about 2^{nR} be chosen at random. Assume the codeword x was sent and the typical sequence y received. A decoding error can happen only if there is some $x' \neq x$ in the code for which the same y is one of the typical received signals. There are about 2^{nR} such codewords x'. As the code was chosen random the probability that a given x' has y in its output fan is about $2^{nH(X|Y)}/2^{nH(X)} = 2^{-nI(X,Y)} = 2^{-n\kappa}$. The probability of a decoding error is therefore bounded by $(2^{R-\kappa})^n$. As $R < \kappa$ this goes to 0 for large n.

Proof of the channel coding theorem

We take this from [3]. Recall some of the basic notions from probability theory, random variables, expectation, variance (Definition 1.20), and the Tschebyscheff inequality (Theorem 1.22).

Choose a length n and a code $\mathcal{C} \in \mathbb{F}_2^n$ with M elements. Assume that all codewords occur with the same probability $1/M$. Assume $x \in \mathcal{C}$ is sent and $y \in \mathbb{F}_2^n$ is received. No error will occur provided x is the unique codeword closest to y. For each codeword $x \in \mathcal{C}$ let $P(x)$ be the probability, that when x is sent the received word is not decoded correctly. The error probability of code $\mathcal{C}$ is defined as the average value $P(\mathcal{C}) = \frac{1}{M}\sum_x P(x)$. Finally

$$P^*(M,n,p) = min\{P(\mathcal{C})\},$$

where $\mathcal{C}$ varies over all codes $\mathcal{C} \subseteq \mathbb{F}_2^n,\ |\mathcal{C}| = M$.

With this terminology the channel coding theorem states:

6.8 Theorem. *Let $R < 1 - h(p)$ and M_n such that $\dfrac{log(M_n)}{n} \leq R$. Then*

$$P^*(M_n,n,p) \longrightarrow 0 \textit{ for } n \to \infty.$$

Proof. Denote by $P(x|y)$ the probability, that x was sent when y is received. Clearly

$$P(x|y) = p^d q^{n-d} = P(y|x), \text{ where } d = d(x,y),\ q = 1-p.$$

Consider the random variable $F(y)$, the number of errors. We have $F = \sum_{i=1}^n F_i$, where $F_i = \begin{cases} 1 & \text{if symbol } y_i \text{ is incorrect,} \\ 0 & \text{if symbol } y_i \text{ is correct.} \end{cases}$
By definition $E(F_i) = p$. Because of the additivity of the expectation (proved in an exercise of Chapter 1) $E(F) = np$. It follows from another exercise of the same chapter that $\sigma^2 = Var(F) = npq$. For some small fixed $\epsilon > 0$ choose $b = \sqrt{\dfrac{\sigma^2}{\epsilon/2}}$. Theorem 1.22 proves that $Prob(|F - np| > b) \leq \epsilon/2$. We can choose n large enough such that
$\rho = \lfloor np + b \rfloor < n/2$.

Introduce the function

$$f(u,v) = \begin{cases} 0 & \text{if } d(u,v) > \rho \\ 1 & \text{if } d(u,v) \leq \rho \end{cases}$$

(the points at distance $> \rho$ are taken care of by Tschebyscheff already). Each $x \in \mathcal{C}$ defines a function g_x, where

$$g_x(y) = 1 - f(y,x) + \sum_{x' \neq x} f(y,x').$$

The main property of g_x is the following:

$$g_x(y) = \begin{cases} 0 & \text{if } x \text{ is the only codeword at distance } \leq \rho \text{ from } y \\ \geq 1 & \text{otherwise} \end{cases}$$

This property shows that the probability $P(x)$ that x is not decoded correctly is bounded by $\sum_y P(y|x)g_x(y)$. It follows

$$P(x) \leq \sum_y P(y|x)(1 - f(y,x)) + \sum_y \sum_{x' \neq x} P(y|x)f(y,x').$$

The first summand is $Prob(d(y,x) > \rho) \leq \epsilon/2$. We have

$$P(\mathcal{C}) \leq \frac{\epsilon}{2} + \frac{1}{M} \sum_x \sum_y \sum_{x' \neq x} P(y|x)f(y,x').$$

In these sums x, x' vary in the code, y in $\mathbb{F}_2^n$. So far we worked with a fixed code $\mathcal{C}$. Consider now the average of this expression when $\mathcal{C}$ varies over all such codes. We have $\sum_{x \in \mathbb{F}_2^n} P(y|x)/2^n = 1/2^n$ and $\sum_{x \in \mathbb{F}_2^n} f(y,x)/2^n = V(\rho,n)/2^n$. Here $V(\rho,n)$ is the volume of the ball of radius ρ, see Definition 5.9. It follows

$$P^*(M,n,p) \leq \frac{\epsilon}{2} + (M-1)V(\rho,n)/2^n.$$

In order to bound the second term take logarithms, divide by n, take the limit for $n \to \infty$. The first term $M - 1$ yields a limit $\leq R$. As $lim(\rho/n) = p$ it follows from Theorem 1.9 that the second term $V(\rho,n)$ yields a limit of $\leq h(p)$. It follows that the limit of the entire second term above under this process is $\leq R + h(p) - 1$, which is negative. This in turn implies that the second term can be made arbitrarily small, for example $< \epsilon/2$. $\square$

Another channel

Consider the channel with matrix

$$M = \begin{pmatrix} (1-p)/2 & (1-p)/2 & p/2 & p/2 \\ p/2 & p/2 & (1-p)/2 & (1-p)/2 \end{pmatrix}$$

It has two input symbols $0, 1$ and four output symbols $0, 1, 2, 3$. Because of the symmetry it will be best to use the expression $I(X, Y) = H(Y) - H(Y|X)$. The reason is that

$$H(Y|X = 0) = H(Y|X = 1) = H((1-p)/2, (1-p)/2, p/2, p/2).$$

The definition of the conditional entropy shows that this is also the value of $H(Y|X)$. The basic property of the entropy function shows how this simplifies:

$$H(Y|X) = H((1-p)/2, (1-p)/2, p/2, p/2) = h(p) + x\cdot 1 + (1-x)\cdot 1 = 1 + h(x).$$

This does not depend on x. So all we need to do is maximize $H(Y)$. The probability $P(Y = 0) = x(1-p)/2 + (1-x)p/2 = q$. This is also $P(Y = 1)$. Likewise $P(Y = 2) = P(Y = 3) = q'$, where $q + q' = 1/2$. It follows

$$H(Y) = H(q, q, q', q') = h(2q) + 2q \cdot 1 + 2q' \cdot 1 = 1 + h(2q).$$

It follows $I(X, Y) = h(2q) - h(p)$. We need to maximize $h(2q)$. In fact $2q = x(1-p) + (1-x)p$. Choosing $x = 1/2$ yields $2q = 1/2$. This shows that the capacity of the channel is $1 - h(p)$, the same as the capacity of the BSC.

Symmetric channels

The channels considered thus far are examples of symmetric channels.

6.9 Definition. *A memoryless channel (stochastic matrix M) is* **symmetric** *if the following are satisfied:*

- *All rows of M are obtained from one another by permutation.*
- *All columns of M are obtained from one another by permutation.*

Examples are the BSC and the channel considered above. As the row sums are $= 1$ and there are b columns it follows that the column sums are a/b. The examples indicate how to proceed in general in order to compute the capacity. As all rows are permutations of ano another we have that $H(Y \mid x_i)$ is constant (independent of i), the row entropy. The information $I(X,Y)$ will therefore be maximized if we can manage that $p(y_j) = 1/b$ for all j. The examples show that this may be reached when we choose the uniform distribution $p(x_i) = 1/a$. Let us do that. Then

$$p(y_j) = \sum_i \frac{1}{a} p(y_j \mid x_i) = 1/b.$$

6.10 Theorem. *The capacity of a symmetric channel with a input and b output symbols is $log(b)$ minus the entropy of a row. It is reached by the uniform distribution on the input alphabet.*

In the example of the preceding section we obtain capacity $log(4) - H(p/2, p/2, (1-p)/2, (1-p)/2) = 2 - (1 + h(p)) = 1 - h(p)$.

The erasure channel

In the binary erasure channel the probability of correct transmission is $1 - p$ just as in the BSC, but in addition the receiver knows when problems occurred. Formally the output alphabet has three elements, $0, E, 1$, where E stands for **erasure.**

The corresponding stochastic matrix is

$$M = \begin{pmatrix} 1-p & p & 0 \\ 0 & p & 1-p \end{pmatrix}$$

It is not symmetric. Clearly its capacity cannot be smaller than that of the BSC and maybe we should expect it to be larger. Can we compute it?

This time it is easier to handle the mutual information in the form $I(X,Y) = H(X) - H(X|Y)$. The reason is that $H(X|Y=0) = H(X|Y=1) = 0$. The only surviving term in the definition of $H(X|Y)$ is therefore $P(Y=E)H(X|Y=E)$. Clearly $H(X|Y=E) = h(x)$ and $P(E) = p$. This shows $H(X|Y) = ph(x)$ and therefore

$$I(X,Y) = (1-p)h(x).$$

Clearly this is maximized by $x = 1/2$.

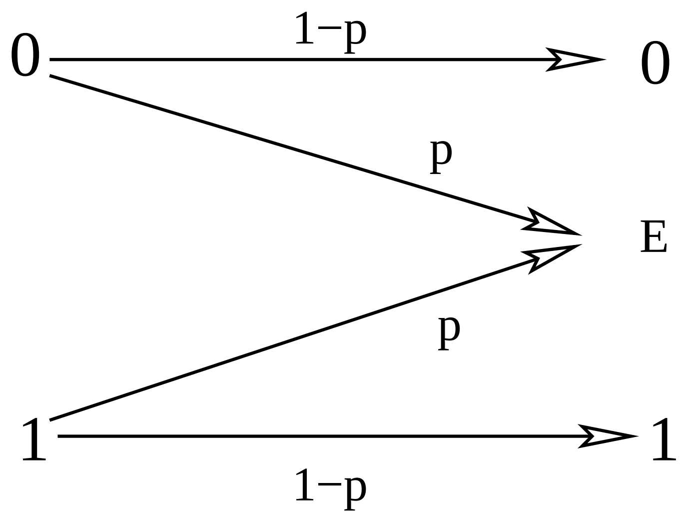

Figure 6.2: The erasure channel

6.11 Theorem. *The (binary) erasure channel has capacity* $1 - p$*. This is achieved for* $P(X = 0) = P(X = 1) = 1/2$.

Comparison with the BSC shows that we expect $1 - p \geq 1 - h(p)$, equivalently $h(p) \geq p$ for $p \leq 0.5$. This is indeed the case.

Zero-error capacity

The following non-propabilistic version of the notion of channel capacity was first considered by Shannon [21]: Input and output alphabet of the channel are the same: $\mathcal{A} = \mathcal{B}$. For each pair of input symbols it is known if they can be confused by the channel or not. The aim is transmission without errors. The structure of the channel is therefore described if we know which pairs can be confused. This gives $\mathcal{A}$ the structure of a graph.

6.12 Definition. *Let* $\mathcal{A}$ *be a finite set. A* **graph** *on* $\mathcal{A}$ *is a collection of pairs from* $\mathcal{A}$*. Those pairs are the* **edges,** *the elements of the ground set* $\mathcal{A}$ *are the* **vertices** *of the graph.*

The pentagon C_5 is a graph on 5 vertices which has 5 edges.

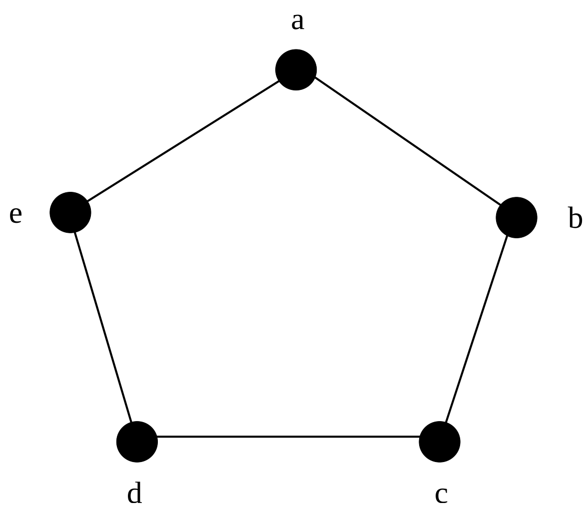

Figure 6.3: The Pentagon C_5

If we send only a or c, then this is OK. As a and c do not form an edge they cannot be confused by the channel. In general we can transmit without errors if we use only symbols (vertices) no two of which form an edge. Such sets of vertices are called independent.

6.13 Definition. *A set of vertices in a graph is* **independent** *if no two form an edge. The* **independence number** $\nu(G)$ *of a graph* G *is the maximal size of an independent set of vertices.*

The independence number of the pentagon C_5 is of course $\nu(C_5) = 2$. It should be clear how circles C_n on n vertices are defined in general. These are graphs with n vertices and n edges. It is clear that $\nu(C_n) = n/2$ if n is even and $\nu(C_n) = (n-1)/2$ if n is odd.

Now use block coding. If the length if chosen to be N, then we transmit signals of length N. As we want error-free communication the codewords have to be chosen such that no two can be confused. When can two signals be confused by the channel? If they can be confused in each coordinate. This defines a graph structure on the N-tuples.

6.14 Definition. *Let* G *be a graph on* n *vertices. The set of signals of length* N *is denoted as* G^N. *Define a graph on* G^N, *the* **product graph,** *as follows: two different signals form an edge if in each coordinate the entries are either the same or form an edge of* G.

In the case of C_5^2 we have that ab and ac form an edge (identical in the first coordinate, edge in the second), whereas ab and be do not form an edge (no edge in the second coordinate). If we aim at error-free transmission based on signals of length N, then we have to use codes which in the product graph form independent sets of vertices. The size of the largest such code is $\nu(G^N)$, the independence number of the product graph. The asymptotic size of this code (see Definition 6.5) is obtained by taking the log and dividing by N. This is what we want to optimize in order to obtain the capacity:

6.15 Definition. *Let G be a graph. Its* **zero-error capacity** *(also known simply as* **Shannon capacity***) is*

$$\Theta(G) = lim_{N \longrightarrow \infty} log(\nu(G^N))/N.$$

This looks complicated and in general the determination of the Shannon capacity is not an easy problem. Just as in the case of covering functions it can be shown that the limit really exists and that each concrete value $\nu(G^N)$ for some N gives a lower bound $log(\nu(G^N))/N$ on the capacity.

It is clear that for example $\nu(G^N) \geq (\nu(G))^N$: simply start from an independent set I of G and use as length N code all tuples whose entries are in I. The asymptotic size is $log(\nu(G))^N)/N = log(\nu(G))$. Can we improve on this in the case of the pentagon? Yes. The following set (code) is independent in C_5^2 :

$$\{aa,\ bc,\ ce,\ db,\ ed\}.$$

This shows that the capacity of the pentagon is at least $log(5)/2$.

Sometimes it is possible to determine the Shannon capacity in an elementary way.

6.16 Definition. *A* **clique** *in a graph G is a subset of vertices such that any two of its vertices form an edge.*

The concept of a clique is the opposite, the dual of independent sets. A clique and an independent set can have at most one vertex in common. What we are interested in is the minimum number of cliques which are necessary to partition the set of vertices. This minimum number can never be smaller than the independence number $\nu(G)$ (if the vertices can be partitioned into 4 cliques, say, then there can be no independent set of more than 4 vertices).

Consider now graphs for which we have equality: graphs with independence number $k = \nu(G)$ which can be partitioned into k cliques. Let us change the graph slightly: remove all edges between different cliques in the partition. The resulting graph's Shannon capacity cannot be smaller than that of G itself. If we replace each of the k cliques by just one vertex (contraction), then the capacity is not changed. This new graph has k vertices and no edges. Its capacity is $log(k)$. We have seen that G has capacity $\leq log(k)$. We knew from the start that its capacity was $\geq log(k)$.

6.17 Theorem. *Let G be a graph with independence number $\nu(G)$, whose vertices can be partitioned into $\nu(G)$ cliques. Then the Shannon capacity of G is $log(\nu(G))$.*

Theorem 6.17 is in Shannon's original paper [21]. These concepts can be expressed using standard graphtheoretic terminology.

6.18 Definition. *The* **complement** $\overline{G}$ *of a graph has the same set of vertices and a pair of vertices form an edge in $\overline{G}$ if and only if they do not form an edge in G.*
The **chromatic number** $\chi(G)$ *is the smallest number k such that the vertices of G can be coloured with k colours in such a way that neighboring vertices always are assigned different colours.*

Expressed in these terms the hypothesis of Theorem 6.17 says $\nu(G) = \chi(\overline{G})$, and the conclusion is $\Theta(G) = log(\chi(G))$.

6.19 Corollary. *We always have $\nu(G) \leq \chi(\overline{G})$. In the case of equality we have $\Theta(G) = log(\nu(G))$.*

With one exception all graphs on ≤ 5 vertices satisfy the requirements of Corollary 6.19. Also, the circles of even length satisfy those requirements. It follows that C_{2m} has Shannon capacity $log(m)$. The exception among the small graphs is, guess what? Yes, the pentagon C_5. In fact we know already that its Shannon capacity is $\geq log(5)/2 > log(\nu(C_5)) = 1$. However, it is not easy to determine its capacity precisely. This was done only in 1979 by L. Lovász [16] who showed equality: the Shannon capacity of C_5 is $log(5)/2$. A relatively short proof is in Aigner-Ziegler's *Proofs from the book* [1]. For all circles of larger odd length the problem remains open.

The Shannon capacity of C_5

A basic idea of Lovász was to represent graphs by unit vectors in Euclidean spaces:

6.20 Definition. *Let G be a graph on n vertices. An* **orthonormal representation** *of G is a mapping from the vertices of G to unit vectors in $\mathbb{R}^s$ for some dimension s such that the vectors are perpendicular whenever the corresponding vertices are independent.*

Let $v^{(1)}, \ldots, v^{(n)}$ be such an orthonormal representation T. Let $u_T = \frac{1}{n}(v^{(1)} + \cdots + v^{(1)})$. Call T **symmetric** if $u_T \cdot v^{(i)} = \sigma_T$ is the same for all i. In that case $\sigma_T = u_T \cdot v^{(i)} = |u_T|^2$

Now consider probability distributions (random variables) on the $v^{(i)}$. If $x = (x_1, \ldots, x_n)$ is such a probability distribution we assign it the value $\mu(x) = |\sum_i x_i v^{(i)}|^2$. Let μ_T be the infimum of $\mu(x)$ over all probability distributions. Choosing the uniform distribution shows $\mu_T \leq \sigma_T$. Choosing the uniform distribution on an independent set of maximum size ν shows $\mu_T \leq 1/\nu$.

Let x be an arbitrary probability distribution and apply the Cauchy-Schwartz inequality to $\sum x_i v^{(i)}$ and u_T. We have $(\sum x_i v^{(i)}) \cdot u_T = \sum x_i \sigma_T = \sigma_T$. The inequality becomes $\sigma_T^2 \leq \mu(x)\sigma_T$, so $\mu_T \geq \sigma_T$. We have equality:

$$\sigma_T = \mu_T \leq 1/\nu.$$

Now consider G^N. The symmetric orthonormal representation T of G leads to a symmetric orthonormal representation T^N of G^N, in dimension sN. In fact, identify the vertices of G with their images under T. Then T^N maps $(v_1, \ldots, v_N) \mapsto (v_1, \ldots, v_N) \in \mathbb{R}^{sN}$, and clearly this is orthonormal, symmetric, with $\sigma_{T^N} = \sigma_T^N$. By what we saw above $\nu(G^N) \leq 1/\sigma_T^N$ and therefore

$$\Theta(G) \leq -log(\sigma_T).$$

Back to C_5. An orthonormal representation in $\mathbb{R}^3$ is given by an umbrella: start from 5 copies of a unit vector originating at the origin, with tip at $(0, 0, 1)$. Then open the umbrella until alternate ribs are perpendicular. This is an orthonormal representation, and it is symmetric. By classical geometry $\sigma_T = 1/\sqrt{5}$. It follows $\Theta(C_5) \leq log(5)/2$. We knew already that this is a lower bound.

6.21 Theorem. $\Theta(C_5) = log(5)/2$.

The Shannon capacity of cycles of odd length ≥ 7 is unknown. According to [1] all we know in case of length 7 is $3.1241 \leq \Theta(C_7) \leq 3.3177$.

Bibliography

[1] M. Aigner and G. M. Ziegler: *Proofs from the book,* Springer 2001, ^{1}1998.

[2] R. B. Ash: *Information Theory,* Dover, New York 1990

[3] J. Bierbrauer: *Introduction to Coding Theory,* Chapman and Hall, CRC Press 2005.

[4] J. Bierbrauer: *Crandall's problem,* 1997.

[5] A. A. Bruen and M. A. Forcinito: *Cryptography, information theory, and error-correction,* Wiley 2005.

[6] T. M. Cover and R. King: *A convergent gambling estimate of the entropy of English,* *IEEE Transactions on Information Theory* **24** (1978), 413-421.

[7] T. M. Cover and J. A. Thomas: *Elements of Information Theory,* Wiley 1991.

[8] Ron Crandall: *Some notes on steganography.* Posted on steganography mailing list http://os.inf.tu-dresden.de/ westfeld/crandall.pdf (1998).

[9] W. Feller: *An Introduction to Probability Theory and Its Applications I,* Wiley 1950.

[10] F. Galand and G. Kabatiansky: *Information hiding by coverings, Proceedings of the IEEE Information Theory Workshop 2004,* 151-154.

[11] D. Hankerson, G. A. Harris, P. D. Johnson, Jr: *Information Theory and Data Compression,* CRC Press 1998.

[12] S. Haykin: *Communications Systems,* Wiley 2001.

[13] J. Ziv and A. Lempel: *A universal algorithm for sequential data compression, IEEE Transactions on Information Theory* **23** (1977), 337-343.

[14] J. Ziv and A. Lempel: *Compression of individual sequences via variable-rate coding,*
IEEE Transactions on Information Theory **24** (1978), 530-536. .

[15] R. Lidl and H. Niederreiter: *Introduction to finite fields and their applications,* Cambridge University Press [1]*1986, revised edition 1994.*

[16] L. Lovász: *On the Shannon Capacity of a Graph, IEEE Transactions on Information Theory* **25** (1979), 1-7.

[17] R. W. Lucky: *Silicon dreams - information, man and machine,* St. Martin's Press 1989.

[18] C. E. Shannon: *A mathematical theory of communication,*
Bell System Technical Journal **27** (1948), 379-423, 623-656.

[19] C. E. Shannon: *Communication Theory of Secrecy Systems,*
Bell System Technical Journal **28** (1949), 656-715.

[20] C. Shannon: *Prediction and entropy of printed English,*
Bell System Technical Journal January 1951, pp. 50-64.

[21] C. E. Shannon: *The zero-error capacity of a noisy channel, IRE Transactions on Information Theory* **2** (1956), 8-19.

[22] G. K. Zipf: *Human behaviour and the principle of least effort*,
Addison-Wesley 1949.