5XSE0: Information theory

Tj.J. Tjalkens

January 25, 2016

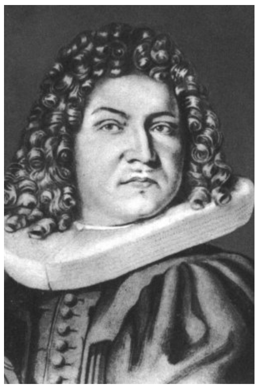
Contents

Part I

Lossless source coding

1 Introduction

1.1 Bernoulli



James Bernoulli (also known as Jacob I) was born in Basel, Switzerland on December 27, 1654 and lived until August 16, 1705. He is one of the eight prominent mathematicians in the Bernoulli family. Following his father's wish, James studies theology and entered the ministry. But contrary to the desires of his parents, he also studied mathematics and astronomy.
He became familiar with calculus through a correspondence with Gottfried Leibniz, then collaborated with his brother Johann on various applications, notably publishing papers on transcendental curves (1696) and isoperimetry (1700, 1701). In 1690, he developed the technique for solving separable differential equations.
In 1682 he founded a school for mathematics and the sciences in Basel. He was appointed professor of mathematics at the University of Basel in 1687, remaining in this position for the rest of his life.

James is best known for the work Ars Conjectandi (The Art of Conjecture), published eight years after his death in 1713 by his nephew Nicholas. This work includes the application of probability theory to games of chance and his introduction of the theorem known as the law of large numbers. The terms *Bernoulli trial* and *Bernoulli numbers* result from this work. Bernoulli crater, on the Moon, is also named after him jointly with his brother Johann.

From Wikipedia.

1.2 Problem statement

Problem
Consider binary sequences, generated by tossing an unfair coin. What is the most compact, or shortest, description of such a sequence that can, almost always, be decoded correctly.

Solution
We shall create unique short binary sequences for the most likely sequences.

2 Definitions

2.1 Repeated experiments

Consider the tossing of an *unfair* coin. The outcome of this *random experiment* is **head** or **tail**. The probability that **head** comes up is denoted by p, with $p \in [0, 1]$. We denote this experiment, or *random variable*, by U and the set of outcome values by $\mathcal{U}$. We use u to denote the outcome of the experiment U. Here $\mathcal{U} = \{\textbf{head}, \textbf{tail}\}$ and we write the probability that the experiment U produces the outcome u as $\Pr\{U = u\}$. However we often use a shorter function description and write

$$\Pr\{U = u\} = P_U(u) = P(u).$$

The last notation is used when we can infer from the context what random variable is meant. So we can write

$$\Pr\{U = \textbf{head}\} = P_U(\textbf{head}) = P(\textbf{head}) = p.$$

If we repeat this experimentwe actually perform an experiment with outcomes that are pairs, or vectors, of coin toss outcomes. We write U_1 and U_2 for these experiments and the combined experiment is written as (U_1, U_2). The set of outcomes is the *Carthesian product* $\mathcal{U} \times \mathcal{U}$. We can consider the probabilities of this experiment

$$P_{(U_1,U_2)}((u_1, u_2)) \quad \text{or the shorthand} \quad P_{U_1,U_2}(u_1, u_2) \quad \text{or even} \quad P(u_1, u_2).$$

It is reasonable to assume that the two random variables are independent and each one is governed by the same probabilities. $P_{U_1}(\textbf{head}) = P_{U_2}(\textbf{head}) = p$.

If the two random variables are independent we can write

$$P_{U_1,U_2}(u_1, u_2) = P_{U_1}(u_1)P_{U_2}(u_2).$$

In general we can write the probability of a sequence of independent random variables U^L as follows.

$$P_{U^L}(u^L) = \prod_{i=1}^{L} P_{U_i}(u_i).$$

2.2 Bernoulli trials

We wish to determine the probability of having k **heads** in n repeated coin tosses.

Let u^n be an outcome with k **heads** and $n - k$ **tails**. Then

$$P_{U^n}(u^n) = p^k(1-p)^{n-k},$$

is independent of the actual order of the outcomes.

From combinatorics we know that there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

different sequences with k **heads**.

So

$$\Pr\{k \textbf{ heads} \text{ in } n \text{ experiments}\} = \binom{n}{k} p^k(1-p)^{n-k}.$$

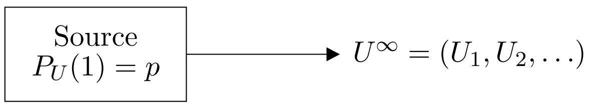


Figure 1: An information source

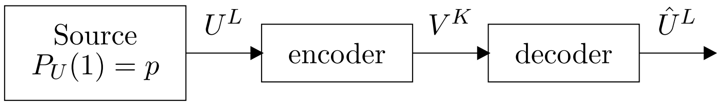


Figure 2: Fixed-to-fixed length source coding

2.3 The binary source

The *binary memoryless source* is like the coin toss experiment repeated indefinitely. The set of outcomes $\mathcal{U}$ is usually written as

$$\mathcal{U} = \{0, 1\}.$$

The description of the source is complete when we define the probability of a one, $P_U(1) = p$, and implicitely also the probability of a zero because $P_U(0) = 1 - P_U(1) = 1 - p$.

Although the source produces an infinite length sequence of zeros and ones we usually chop these up into sequences of length L.

2.4 Source code

Source code

A fixed-to-fixed length source code maps source sequences U^L of length L onto binary code words V^K of length K. The efficiency of this code is measured by the *source rate* R_s.

$$R_s \triangleq \frac{K}{L}$$

has the interpretation of code symbols per source symbol. A source rate of 0.5 means that the code uses one code symbol per two source symbols.

The *encoder* maps the source sequence u^L to one of the 2^K code words.

The *decoder* tries to do the inverse. It decodes every code word v^K to a source sequence $\hat{u}^L$.

An *error* occurs when the source generates a sequence u^L for which after encoding and decoding the received sequence $\hat{u}^L$ is not equal to the encoded word u^L.

Compression is achieved when $R_s < 1$. This implies that $K < L$ and thus there are fewer code words than source sequences. So errors are *unavoidable*. We define the error probability P_e as

$$P_e \triangleq \Pr\{\hat{U}^L \neq U^L\}.$$

3 Analysis

3.1 The encoding set

The *encoding set* $\mathcal{A}_L$ contains all source sequences that can be decoded correctly. This set cannot contain more than 2^K sequences because there are precisely that many code words.

We write $\mathcal{A}_L^c$ for the set of source sequences that are *not* in the encoding set. $\mathcal{A}_L^c$ is the *complement* of $\mathcal{A}_L$.

The error probability is now given as

$$P_e = \sum_{u^L \in \mathcal{A}_L^c} P_{U^L}(u^L).$$

3.2 Hamming sphere

In order to minimize the error probability for a given sequence length L and code rate R_s, it is obvious that the encoding set $\mathcal{A}_L$ must contain the most likely sequences.

Assume that the source produces zeros more likely than ones, so $P_U(0) \geq P_U(1)$ or equivalently $p \leq \frac{1}{2}$. Then we know that the sequence of L zeros is the most likely sequence.

Recall $\Pr\{k$ **heads** in n experiments$\}$. The more zeros a sequence contains, the more likely it is.

So we put all sequences that have no more than t ones in the encoding set $\mathcal{A}_L$. For a given t, we must make the code wordlength K large enough to accomodate all sequences in $\mathcal{A}_L$. So, if we select K as

$$K = \left\lceil \log_2 \left(\sum_{i=0}^{t} \binom{L}{i} \right) \right\rceil,$$

then the set $\mathcal{A}_L$ can contain all source sequences with at most t ones.

A bound on the binomial coefficient

First we define a function $h(\delta)$ that is valid for any real value δ with $0 \leq \delta \leq 1$.

$$h(\delta) \overset{\Delta}{=} -\delta \log_2 \delta - (1-\delta)\log_2(1-\delta).$$

Now $\delta \log_2 \delta$ does not exist for $\delta = 0$ but we define

$$0 \log_2 0 \overset{\Delta}{=} 0.$$

This is in correspondence with the standard limit

$$\lim_{x \to 0} x \log x = 0.$$

The function $h(x)$ is called the *binary entropy* function.

Theorem 1. *Let $\delta \leq \frac{1}{2}$ be such that δn is an integer for a given integer n. Then*

$$\sum_{i=0}^{\delta n} \binom{n}{i} \leq 2^{nh(\delta)}.$$

The proof is left as an excercise.

Bounding the code rate

Turn back to the encoding set $\mathcal{A}_L$. From the previous lemma it is obvious that

$$\begin{aligned} K &= \left\lceil \log_2 \left(\sum_{i=0}^{t} \binom{L}{i} \right) \right\rceil \\ &\leq \left\lceil \log_2 2^{Lh(\frac{t}{L})} \right\rceil \\ &= Lh(\frac{t}{L}) + 1. \end{aligned}$$

Thus the code rate R_s can be upper bounded as

$$R_s < h(\frac{t}{L}) + \frac{1}{L}.$$

A law of large numbers

Before we consider error probabilities we shall look at the probability distribution of the Bernoulli trial.

In Figure 3 we depict the probability of the fraction of ones generated by a memoryless binary source with $\Pr\{U = 1\} = p = 0.2$. As described before this probability is given as

$$\Pr\{k \text{ ones in } u^n\} = \binom{n}{k} p^k (1-p)^{n-k}.$$

We rescale these probabilities to fractions so that we can plot the distributions for various n in one graph.

$$P(\frac{k}{n}) = n \cdot \Pr\{k \text{ ones in } u^n\}.$$

The multiplication by n is needed to ensure that the function $P()$ still integrates to 1.

We observe that as n increases the distribution concentrates more and more on a short interval around 0.2. This is known as Bernoulli's "Law of large numbers".

This tells us that we must include the sequences with $(p+\delta)L$ ones in the set $\mathcal{U}$. The larger we pick δ, the smaller the error probability becomes, but the size of $\mathcal{U}$ increases.

The following theorem allows us to upper bound the error probability efficiently.

Theorem 2. *Let $0 < p < \delta \leq 1$ and let δn be an integer. Then*

$$\sum_{i=\delta n}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq 2^{-nd(\delta \| p)},$$

where the binary divergence $d(\delta \| p)$ is given as

$$d(\delta \| p) \triangleq \delta \log_2 \frac{\delta}{p} + (1-\delta) \log_2 \frac{1-\delta}{1-p}.$$

Before we give the proof of this theorem, we first consider the behaviour of the divergence. For $d(\delta \| p)$ the following two properties hold.

- $d(\delta \| p) \geq 0$ for all δ and p in the interval $[0,1]$.

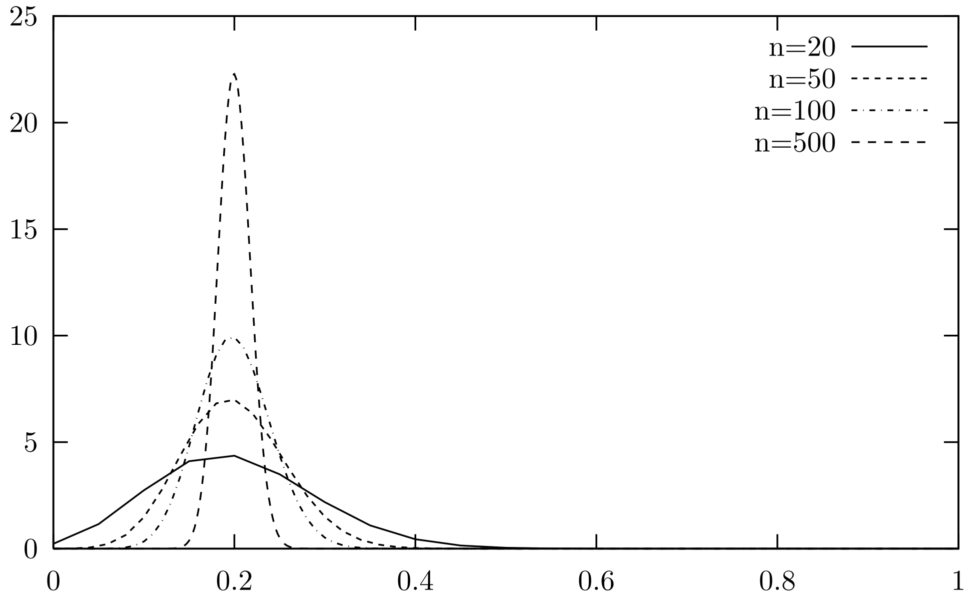


Figure 3: Bernoulli trials with $n = 20, 50, 100, 500$ samples, drawn with $p = 0.2$
x-axis: k/n;
y-axis: $P(\frac{k}{n}) = n \cdot \Pr\{k \text{ ones in } u^n\}$.

- $d(\delta \| p) = 0$ if and only if $\delta = p$.

Now we can consider the proof of the theorem.

Proof. Observe that $\delta(1-p) > (1-\delta)p$. We write

$$
\begin{aligned}
1 = (\delta + (1-\delta))^n &= \sum_{i=0}^{n} \binom{n}{i} \delta^i (1-\delta)^{n-i} \geq \sum_{i=\delta n}^{n} \binom{n}{i} \delta^i (1-\delta)^{n-i} \\
&= \sum_{i=\delta n}^{n} \binom{n}{i} \left(\frac{\delta}{p}\right)^i \left(\frac{1-\delta}{1-p}\right)^{n-i} p^i (1-p)^{n-i} \\
&= \sum_{i=\delta n}^{n} \binom{n}{i} \left(\frac{\delta(1-p)}{(1-\delta)p}\right)^i \left(\frac{1-\delta}{1-p}\right)^{n} p^i (1-p)^{n-i} \\
&\geq \sum_{i=\delta n}^{n} \binom{n}{i} \left(\frac{\delta(1-p)}{(1-\delta)p}\right)^{\delta n} \left(\frac{1-\delta}{1-p}\right)^{n} p^i (1-p)^{n-i} \\
&= \left(\frac{\delta}{p}\right)^{\delta n} \left(\frac{1-\delta}{1-p}\right)^{n-\delta n} \sum_{i=\delta n}^{n} \binom{n}{i} p^i (1-p)^{n-i} \\
&= 2^{nd(\delta\|p)} \sum_{i=\delta n}^{n} \binom{n}{i} p^i (1-p)^{n-i}
\end{aligned}
$$

□

Bounding the error probability

The previous theorem immediately gives us that if we select an encoding set $\mathcal{A}_L$ that contains all source sequences with t or less ones, where $\frac{t}{L} > p$ and p is the probability that the source generates a "1", then

$$\begin{aligned}
P_e &= \sum_{u^L \in \mathcal{A}_L^{\mathsf{c}}} P_{U^L}(u^L) \\
&\le \sum_{i=t+1}^{L} \binom{L}{i} p^i (1-p)^{L-i} \\
&\le \sum_{i=t}^{L} \binom{L}{i} p^i (1-p)^{L-i} \\
&\le 2^{-Ld(\frac{t}{L}\|p)},
\end{aligned}$$

The first inequality comes from the fact that $\mathcal{U}$ may contain some sequences with $t+1$ ones to fill it up to 2^K sequences.

Because $\delta > p$ we know that $d(\delta\|p)$ is strictly greater than zero, so the error probability P_e goes to zero exponentially fast in n.

The relation between rate and error probability

We now combine the analysis of the code rate and the error probabiltiy into the following theorem.

Theorem 3. *Let $\{U_i\}$ be a binary memoryless source with $P(1) = p$. For $\epsilon \to 0$ and $L \to \infty$ sufficiently large holds*

$$\begin{aligned}
R_s &\to h(p), \\
P_e &\to 0
\end{aligned}$$

This theorem tells us that we can *compress* the binary data from source $\{U_i\}$ to, asymptotically, $h(p)$ binary code symbols per source symbol, such that the error probability vanishes with growing L.

Proof. Consider an encoding set $\mathcal{A}_L$ that contains all sequences with at most $t = L(p+\epsilon)$ ones. Choose L larger than $\frac{1}{\epsilon}$. Then

$$R_s < h(p+\epsilon) + \epsilon.$$

Now, because $h(x)$ is a continuous function in x, we have

$$\lim_{\epsilon \to 0} h(p+\epsilon) = h(p).$$

For the error probability we have the bound

$$P_e < 2^{-Ld(p+\epsilon\|p)}.$$

Because $\epsilon > 0$ we also have $d(p+\epsilon\|p) > 0)$, so for any p and any fixed value of ϵ the error probability approaches 0 exponentially fast with growing L.

We combine these two bounds for $\epsilon \to \infty$ and $L \to \infty$, and find

$$\lim_{\epsilon \to 0} \lim_{L \to \infty} R_s = h(p),$$
$$\lim_{\epsilon \to 0} \lim_{L \to \infty} P_e = 0.$$

□

Part II

Constrained sequences

4 Introduction

4.1 Problem statement

In the previous lecture we have seen that the data from a binary memoryless source with probability p can be compressed to $h(p)$ binary digits per source symbol with vanishing probability of error.

In order to achieve this result we must assign to all words u^L in a given, well-selected set, a unique code word. The binary logarithm of the size of this set gives us the required code wordlength, which is approximately $Lh(p)$. So the number of codewords that we need is exponential in L, namely about $2^{Lh(p)}$.

This requires an enormous amount of memory.

In this lecture we shall discuss a method, *enumerative coding* that addresses this complexity issue.

4.2 Dictionary and Index

Source model
We consider binary sequences of length L that are generated by a binary memoryless source with probability parameter p, i.e. $\Pr\{u_i = 1\} = p$ independently for all letters of the sequence u^L. We assume that $p \leq \frac{1}{2}$.

Dictionary
A dictionary is a set of binary sequences of length L. We consider the dictionary $\mathcal{A}_{L,d}$ that contains all binary sequences of length L having at most d ones.

$$\mathcal{A}_{L,d} \triangleq \left\{ u^L \in \{0,1\}^L : \sum_{i=1}^{L} u_i \leq d \right\}.$$

Index and code word
To every sequence in the dictionary $\mathcal{A}$ (which we shall also call a *word*), we assign a unique integer ranging from zero up-to $|\mathcal{A}| - 1$ inclusive*.
The code word that corresponds to an index is the binary representation of the index in $\lceil \log_2 |\mathcal{A}| \rceil$ binary digits.

4.3 Lexicographical order

The first word in the dictionary is given the index zero. The next word gets an index one and so on.

An example dictionary

$\mathcal{A}_{4,2}$

index	word	index	word	index	word
0:	0000	4:	0100	8:	1001
1:	0001	5:	0101	9:	1010
2:	0010	6:	0110	10:	1100
3:	0011	7:	1000		

Note that the index of a word in the dictionary is equal to the number of words that precede it in the dictionary.

Now we consider the way words are ordered in a dictionary.

First we must define an order of the letters of the alphabet $\mathcal{U}$. Any order will do, but we usually take $0 < 1$ for the binary alphabet.

Let u^L and v^L be two different words from the dictionary $\mathcal{A}$. We say that u^L *precedes* v^L if at the first position where u^L and v^L differ, the letter from u^L comes before the corresponding letter in v^L.

Definition 4 (Lexicographical ordering)**.** The lexicographical ordering $<$ of words from a dictionary $\mathcal{A}$ is defined as follows. For any two different words $u^L \in \mathcal{A}$ and $v^L \in \mathcal{A}$

$$u^L < v^L \text{ iff } \exists i : 1 \leq i \leq L : \forall j : 1 \leq j < i, u_j = v_j \text{ and } u_i < v_i.$$

4.4 Lexicographical index

In order to determine the lexicographical index of a word u^L we must count the number of words that precede it in the dictionary, or

$$\mathbb{i}(u^L) \triangleq \left| \left\{ v^L \in \mathcal{A} : v^L < u^L \right\} \right| .$$

It is rather time consuming to determine the index by straightforward counting from the start of the dictionary. We shall take another approach, as described by Cover and Schalkwijk.

Suppose we know for every *sequence* of letters over the alphabet, how many words exist in the dictionary that start with this sequence. We denote the number of words in the dictionary $\mathcal{A}$ that start with the sequence u^i by the function $\mathbb{n}_{\mathcal{A}}(u^i)$.

*The notation $|\mathcal{A}|$ gives the number of elements in the set $\mathcal{A}$.

Definition 5 (Word-set size)**.** The number of words in a dictionary $\mathcal{A}$ that start with the sequence u^i is denoted by $\mathbb{n}_{\mathcal{A}}(u^i)$ and is defined as

$$\mathbb{n}_{\mathcal{A}}(u^i) \triangleq \left|\left\{v^L \in \mathcal{A} : v^i = u^i\right\}\right|.$$

Suppose that the word u^L is in the dictionary, then we can determine its index by summing up, for all $i = 1, 2, \ldots, L$, the number of words that start with u^{i-1} followed by a letter v that is lexicographically less than u_i.

$$\mathbb{i}(u^L) = \sum_{i=1}^{L} \sum_{v<u_i} \mathbb{n}_{\mathcal{A}}(u^{i-1}v). \tag{1}$$

Suppose that the dictionary is given as follows.

$$\mathcal{A}_{4,2} = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 1000, 1001, 1010, 1100\}.$$

The following table lists a sequence of symbols and the words in the dictionary that start with this sequence, and the word-set size.

We shall need the notion of the *empty sequence*. This is the unique sequence of length zero. We denote this sequence by the symbol λ.

sequence	list of words	word-set size
λ	000, 0001, ..., 1100	$\mathbb{n}_{\mathcal{A}}(\lambda) = 11$
0	0000, 0001, 0010, 0011, 0100, 0101, 0110	$\mathbb{n}_{\mathcal{A}}(0) = 7$
00	0000, 0001, 0010, 0011	$\mathbb{n}_{\mathcal{A}}(00) = 4$
01	0100, 0101, 0110	$\mathbb{n}_{\mathcal{A}}(01) = 3$

So, we can compute $\mathbb{i}(0110)$ as

$$\begin{aligned}\mathbb{i}(0110) &= \sum_{v<u_1} \mathbb{n}_{\mathcal{A}}(v) + \sum_{v<u_2} \mathbb{n}_{\mathcal{A}}(u_1 v) + \sum_{v<u_3} \mathbb{n}_{\mathcal{A}}(u_1 u_2 v) + \sum_{v<u_4} \mathbb{n}_{\mathcal{A}}(u_1 u_2 u_3 v)\\ &= \sum_{v<0} \mathbb{n}_{\mathcal{A}}(v) + \sum_{v<1} \mathbb{n}_{\mathcal{A}}(0v) + \sum_{v<1} \mathbb{n}_{\mathcal{A}}(01v) + \sum_{v<0} \mathbb{n}_{\mathcal{A}}(011v)\\ &= 0 + \mathbb{n}_{\mathcal{A}}(00) + \mathbb{n}_{\mathcal{A}}(010) + 0 = 0 + 4 + 2 + 0 = 6.\end{aligned}$$

5 Enumerative coding

5.1 Hamming Sphere Sequences

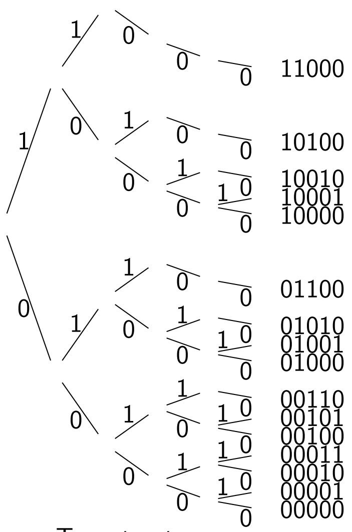
Example
The dictionary $\mathcal{A}_{5,2}$ contains 16 sequences as is calculated next.

$$|\mathcal{A}_{5,2}| = \sum_{i=0}^{2} \binom{5}{i} = \binom{5}{0} + \binom{5}{1} + \binom{5}{2} = 1 + 5 + 10 = 16.$$

$$\mathcal{A}_{5,2} = \{00000, 00001, 00010, 00011, 00100, 00101,\\ 00110, 01000, 01001, 01010, 01100, 10000,\\ 10001, 10010, 10100, 11000\}.$$

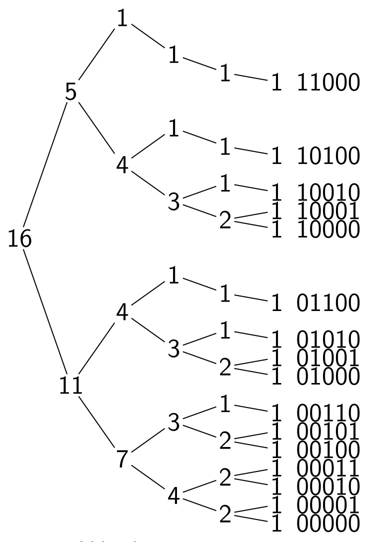
We can draw the corresponding tree with the word-set sizes $\mathbb{n}_{\mathcal{A}_{5,2}}(u^5)$.

The following slide displays the tree where the branches are labeled with the corresponding source symbol.



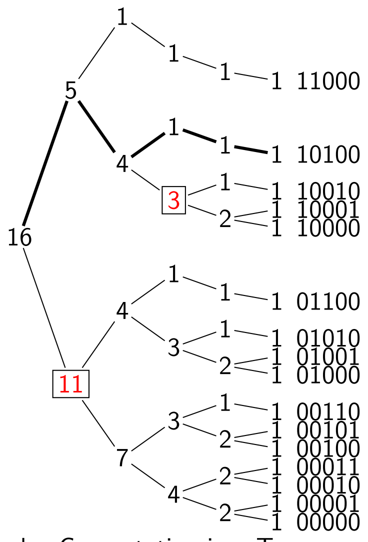
Tree structure

The next slide shows the word-set sizes.



Word-set size

The index of a sequence can be computed with the aid of formula 1. The next slide depicts this process

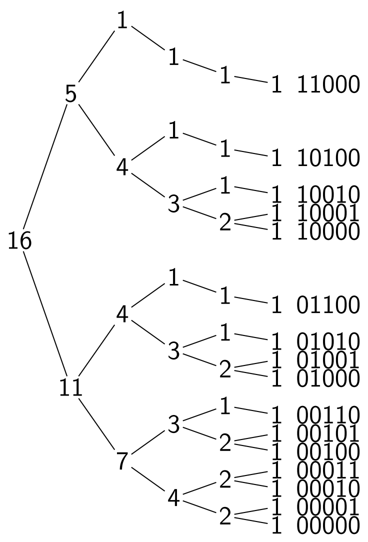


$u^5 = 1\,0\,1\,0\,0$

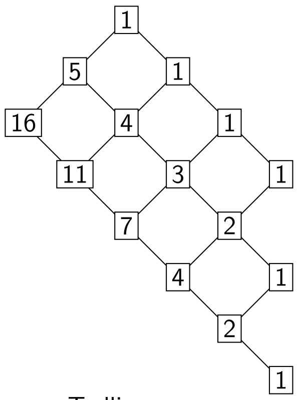
$\mathring{\imath}(10100) = 14$

Index Computation in a Tree

The word-set size $\mathbb{n}_{\mathcal{A}}(u^i)$ actually depend on the composition, i.e. the number of zeros and ones, of the sequence u^i. So all of 011, 101, 110 have the same word-set size. Thus the tree collapses into a trelles as the next slide shows.

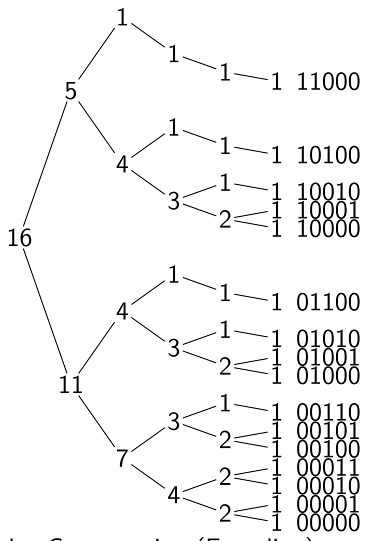


Trellis Word-set Size



Trellis structure

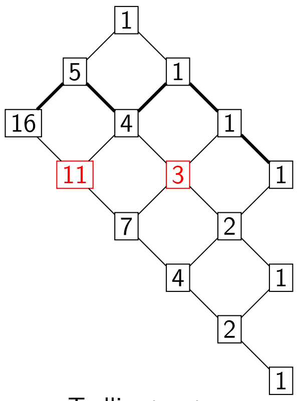
Computing the index in the trellis is similar to the computation in the tree. The next slide shows that indeed the index is the same in both cases.



Index Computation (Encoding)

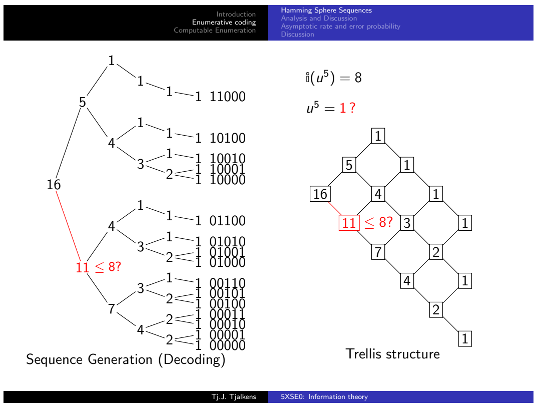
$u^5 = 1\,0\,1\,0\,0$

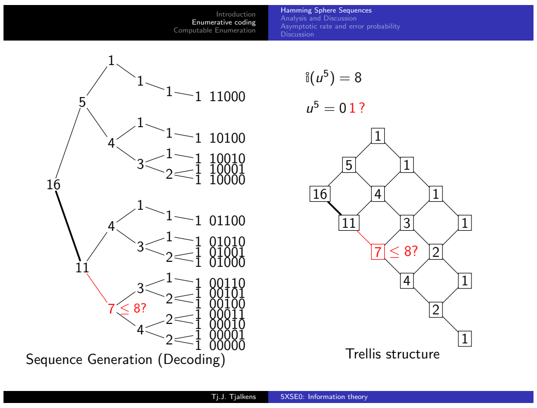
$\mathbb{i}(10100) = 14$

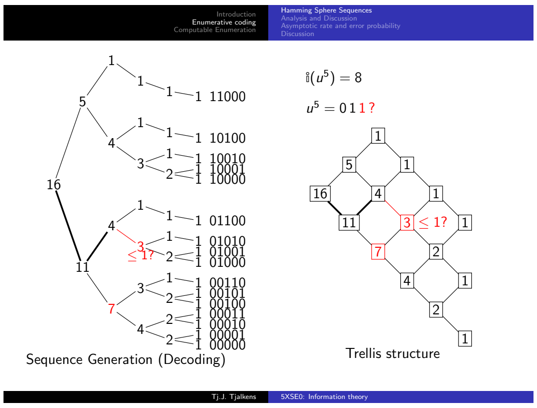


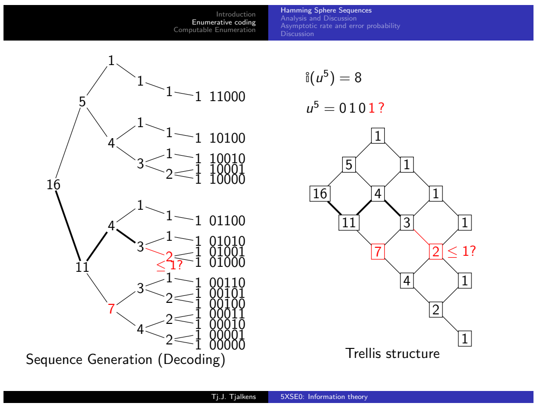
Trellis structure

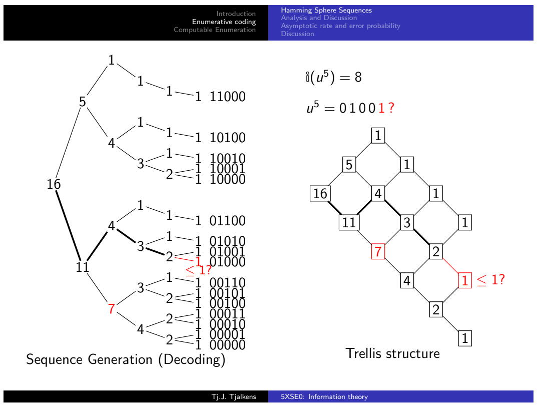
Decoding, or the determination of the sequence that corresponds to a certain index, can also be performed both in the tree and the trellis. The next sequence of six slides depicts this process.

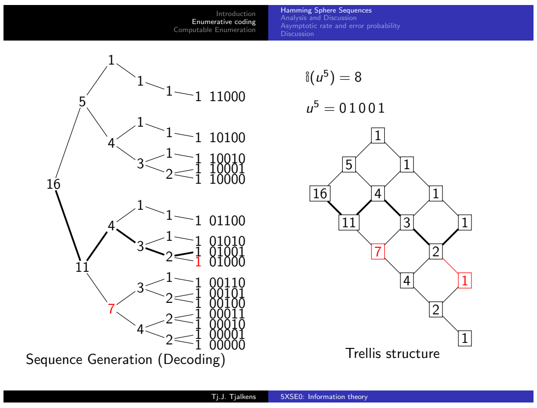












5.2 Analysis and Discussion

From the lecture on source coding we know the size (Theorem 1) and error probability (Theorem 2) of the dictionary $\mathcal{A}_{L,d}$, namely:

$$|\mathcal{A}_{L,d}| = \sum_{i=0}^{d} \binom{L}{i} \leq 2^{Lh(d/L)},$$

$$R_s = \frac{\lceil \log_2 |\mathcal{A}_{L,d}| \rceil}{L} \leq h(\frac{d}{L}) + \frac{1}{L}.$$

and if we require that $d > Lp$ we have a bound on the error probability

$$P_e = \sum_{i=d+1}^{L} \binom{L}{i} p^i (1-p)^{L-i} \leq 2^{-Ld(\frac{d+1}{L}\|p)} \leq 2^{-Ld(\frac{d}{L}\|p)}.$$

5.3 Asymptotic rate and error probability

Now select d such that for some positive ϵ

$$d = (p+\epsilon)L.$$

Just as in the previous lecture we can now conclude that

$$\lim_{\epsilon\to 0}\lim_{L\to\infty} R_s = h(p),$$
$$\lim_{\epsilon\to 0}\lim_{L\to\infty} P_e = 0.$$

5.4 Discussion

The advantage of the enumerative method described in this lecture is the reduced complexity of encoding and decoding.

In stead of storing all possible source sequences and the corresponding code words we must store a table of integers. The number of storage cells is quadratic in L and the number of bits needed to store an integer is

$$\log_2 |\mathcal{A}_{L,d}| \leq \log_2(2^{Lh(d/L)}) \leq L \text{ bits}.$$

So in total we need some $a \cdot L^3$ bits to store the table, where a is some positive constant.

In order to encode or decode the word u^L we need at most L table accesses and integer additions.

6 Computable Enumeration

6.1 Pascals Triangle

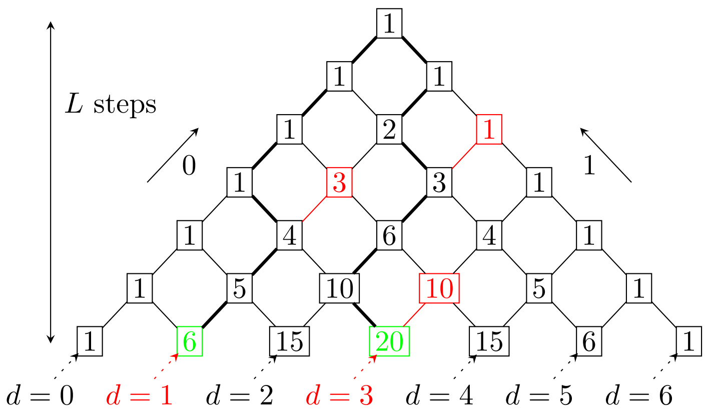
We introduce another method of enumerative coding for memoryless sources. This method assigns a code word to every source sequence u^L. These code words are *not* of fixed length and we shall consider the expected code wordlength. Also, because every source sequence has a decodable code word, this method has *zero error*.

We assume that the length L of the source sequence u^L is fixed and known to both encoder and decoder. The method, described by Schalkwijk, first determines the number of ones in the word u^L. Denote this number by d. Then it determines the lexicographical index of the word u^L in the set $\mathcal{A}_{L,d+}$ that contains all binary sequences of length L with precisely d ones. The code word for u^L consists of two parts.

1. First the number of ones, $d \in \{0, 1, \dots, L\}$, is described in $\lceil \log_2(L+1) \rceil$ bits.
2. Then the index is described in $\lceil \log_2 |\mathcal{A}_{L,d+}| \rceil$ bits.

The length of the first part is the same for every sequence u^L while the length of the second part depends on d, the number of ones in u^L.

6.2 Encoding (and decoding)



$\mathbb{i}(001000) = 3$ in $\lceil \log_2 6 = 3 \rceil$ bit [011]

$\mathbb{i}(100101) = 11$ in $\lceil \log_2 20 = 5 \rceil$ bit [01011]

6.3 Expected code rate

The length of the code word for the sequence u^L is written as $K(u^L)$. Let $d = n_1(u^L)$ equal the number of ones in the sequence u^L. It is obvious that $\mathcal{A}_{L,d+}$ contains $\binom{L}{d}$ words, so we find the following expression for the code wordlength.

$$K(u^L) = \lceil \log_2(L+1) \rceil + \left\lceil \log_2 \binom{L}{n_1(u^L)} \right\rceil .$$

The sequence is generated by a binary memoryless source with probability p, so $\Pr\{u = 1\} = p$. So, the probability of u^L is given as

$$\Pr\{U^L = u^L\} = p^{n_1(u^L)}(1-p)^{L-n_1(u^L)}.$$

The *expected code wordlength* is now

$$\begin{aligned}\bar{K} &= \sum_{d=0}^{L} \binom{L}{d} p^d (1-p)^{L-d} \left(\lceil \log_2(L+1) \rceil + \left\lceil \log_2 \binom{L}{d} \right\rceil \right) \\ &= \lceil \log_2(L+1) \rceil + \sum_{d=0}^{L} \binom{L}{d} p^d (1-p)^{L-d} \left\lceil \log_2 \binom{L}{d} \right\rceil\end{aligned}$$

From Theorem 1 of the previous lecture we know that $\binom{L}{d} \leq 2^{Lh(\frac{d}{L})}$. So we write

$$\bar{K} = \lceil \log_2(L+1) \rceil + F(p)$$

where we introduce the function F as

$$\begin{aligned} F(p) &\triangleq \sum_{d=0}^{L} \binom{L}{d} p^d (1-p)^{L-d} \left\lceil \log_2 \binom{L}{d} \right\rceil \\ &\leq \sum_{d=0}^{L} \binom{L}{d} p^d (1-p)^{L-d} (Lh(\frac{d}{L}) + 1) \end{aligned}$$

We use the fact that $h(x)$ is a *concave* function; this will be discussed in a lecture later-on. Because of this we know that the expected value of the binary entropy of a random variable X is not more than the binary entropy of the expected value of the random variable, or

$$\overline{h(X)} \leq h(\bar{X}) \quad \Longleftrightarrow \sum_{x \in \mathcal{X}} P(x) h(x) \leq h\left(\sum_{x \in \mathcal{X}} P(x) x \right)$$

Then we continue with the function F.

$$\begin{aligned} F(p) &\leq 1 + L \sum_{d=0}^{L} \binom{L}{d} p^d (1-p)^{L-d} h(\frac{d}{L}) \\ &\leq 1 + Lh \left(\sum_{d=0}^{L} \binom{L}{d} p^d (1-p)^{L-d} \frac{d}{L} \right) \end{aligned}$$

Now we use the well known fact that the expected value of a binomially distributed random variable is Lp, so

$$F(p) \leq 1 + Lh(p).$$

This results in the following upper bound on the expected code wordlength $\bar{K}$.

$$\bar{K} \leq \log_2(L+1) + Lh(p) + 2.$$

This finally results in the *expected compression rate* R_s bound

$$R_s \leq h(p) + \frac{\log_2(L+1)}{L} + \frac{2}{L}.$$

Thus as $L \to \infty$ we observe that

$$R_s \overset{L\to\infty}{\to} h(p).$$

6.4 Universal code

Note that the previous method does not need to know the actual source probability p and still achieves (asymptotically) the expected compression rate $R_s = h(p)$. (The actual rate is a random variable but with high probability the average rate will be close to the expected rate). So, this code works for any binary memoryless source. Also the decoder can *always* decode correctly.

This is a simple example of an *universal code.*

Part III
Entropy

7 Introduction

7.1 Robert Gallager

Robert G. Gallager. Born May 29, 1931 in Philadelphia, PA, USA. Robert G. Gallager is an American electrical engineer known for his work on information theory and communications networks. He was a member of the technical staff at the Bell Telephone Laboratories in 1953-1954. He has been a faculty member at MIT since 1960, and became Professor Emeritus in 2001.

Gallager's 1960 Sc.D. thesis, Low Density Parity Check Codes, was published by the M.I.T. Press as a monograph in 1963. This paper won an IEEE Information Theory Society Golden-Jubilee Paper Award in 1998 and its subject matter is a very active area of research today.

Gallager's January 1965 paper in the IEEE Transactions on Information Theory, "A Simple Derivation of the Coding Theorem and some Applications, won the 1966 IEEE W.R.G. Baker Prize "for the most outstanding paper, reporting original work, in the Transactions, Journals and Magazines of the IEEE Societies, or in the Proceedings of the IEEE' and also won another IEEE IT Society Golden-Jubilee Paper Award in 1998. His book, Information Theory and Reliable Communication, Wiley 1968, placed Information Theory on a sound mathematical foundation and is still considered by many as the standard textbook on information theory.

Over the years, Gallager has taught and mentored many graduate students, many of whom are now themselves leading researchers in their fields. He received the M.I.T. Graduate Student Council Teaching Award for 1993.

From Wikipedia.

7.2 Problem statement

Problem
We have studied the compaction and compression of binary data. Now it is time to generalize this to arbitrary discrete alphabets, i.e. alphabets that contain a finite number of elements.

8 Data compaction

8.1 Definition of entropy

Assume that U is a discrete random variable that takes values in a finite alphabet $\mathcal{U}$. Let $|\mathcal{U}|$ denote the cardinality of the alphabet $\mathcal{U}$ and suppose that $p(u) := \Pr\{U = u\}$ for $u \in \mathcal{U}$.

Definition 6. The *entropy* of U is defined by

$$H(U) \triangleq \sum_{u \in \mathcal{U}} p(u) \log_2 \frac{1}{p(u)}.$$

Because the base of the logarithm is 2, we say that the entropy is being measured in *bits* (from <u>bi</u>nary digi<u>ts</u>).

Also, if $p(u) = 0$ then the term $p(u) \log_2 \frac{1}{p(u)}$ is indeterminate, but as discussed before, we define it to be 0. This makes $H(U)$ continuous in the probability vector, or distribution, $(p(u) : u \in \mathcal{U})$.

8.2 Properties of entropy

If the random variable U takes values in the alphabet $\mathcal{U}$ then

Theorem 7. $0 \le H(U) \le \log_2 |\mathcal{U}|$. *Furthermore* $H(U) = 0$ *if and only if* $p(u) = 1$ *for some* $u \in \mathcal{U}$, *and* $H(U) = \log_2 |\mathcal{U}|$ *if and only if* $p(u) = \frac{1}{|\mathcal{U}|}$ *for all* $u \in \mathcal{U}$.

Proof. Since each $p(u)$ is ≤ 1 it follows that each term $p(u) \log_2 \frac{1}{p(u)}$ is ≥ 0. So $H(U) \ge 0$. Furthermore $H(U) = 0$ implies that all terms $p(u) \log_2 \frac{1}{p(u)}$ are 0. A term is equal to 0 only if $p(u) = 0$ or $p(u) = 1$. Therefore $H(U) = 0$ only if one $p(u) = 1$ and all the rest are 0.

Assume that $p(u) > 0$ for all $u \in \mathcal{U}$. Using the inequality $\ln t \le t - 1$ we then obtain :

$$\begin{aligned}
H(U) - \log_2 |\mathcal{U}| &= \sum_{u \in \mathcal{U}} p(u) \log_2 \frac{1}{p(u)} - \sum_{u \in \mathcal{U}} p(u) \log_2 |\mathcal{U}| \\
&= \frac{1}{\ln 2} \sum_{u \in \mathcal{U}} p(u) \ln \frac{1}{|\mathcal{U}| p(u)} \\
&\le \frac{1}{\ln 2} \sum_{u \in \mathcal{U}} p(u) \left[\frac{1}{|\mathcal{U}| p(u)} - 1 \right] \\
&= \frac{1}{\ln 2} \left[\sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{U}|} - \sum_{u \in \mathcal{U}} p(u) \right] = 0.
\end{aligned}$$

Note that the log-inequality holds with equality only if $t = 1$ and thus $H(U) = \log_2 |\mathcal{U}|$ holds with equality only if $p(u) = \frac{1}{|\mathcal{U}|}$ for all $u \in \mathcal{U}$. □

Another important property of the entropy $H(U)$ is that it is a *concave* function of the symbol probabilities $p(U)$, as we already know.

8.3 Entropy as an information measure

Information should be regarded as the entity that can resolve uncertainty. To see what this means consider a random experiment that generates a random variable U. Before starting this experiment we are uncertain about its actual result. The generated output u contains the information that takes away this initial uncertainty. We would like the entropy $H(U)$ to be a measure for the initial uncertainty or equivalently for the generated information.

Entropy has some properties that a useful measure of information must have:

- If $p(u) = 1$ for some $u \in \mathcal{U}$ then we know in advance what the output of the experiment will be. There is no initial uncertainty, in other words the experiment generates no information. Indeed in this case the entropy $H(U)$ is equal to 0.
- If $p(u) = \frac{1}{|\mathcal{U}|}$, for all $u \in \mathcal{U}$, i.e. when all outputs are equally likely, the entropy $H(U) = \log_2 |\mathcal{U}|$. It was pointed out already by Hartley in 1928 that for equiprobable outcomes $\log_2 |\mathcal{U}|$ would be the most natural choice. The reason for this is simple. Suppose we have two random experiments that generate a respectively b equiprobable outcomes. Then the information generated by the two experiments should be equal to the information generated by the first experiment plus the information generated by the second. The logarithmic function satisfies this requirement since the combined experiment has $a \cdot b$ equiprobable outcomes and $\log_2 a \cdot b = \log_2 a + \log_2 b$.
- As we have seen, entropy is always ≥ 0. Indeed uncertainty can be absent, i.e. $H(U) = 0$, but it is rather difficult to think of negative uncertainty.

8.4 Discrete memoryless source

So far we have looked only at a single random variable. A *discrete information source* generates a sequence of random variables, which take values in the finite (source-) alphabet $\mathcal{U}$. We denote such a sequence by $U^L = (U_1, U_2, \cdots, U_L)$.

Here we only consider sources that are *memoryless*, in other words we assume that the random variables $U_1, U_2, \cdots, U_L$ are independent and distributed just like a (generic) random variable U with probability distribution $\{p(u) : u \in \mathcal{U}\}$. Consequently

$$\begin{aligned} p(u^L) &\stackrel{\Delta}{=} \Pr\{U_1 = u_1, U_2 = u_2, \cdots U_L = u_L\} \\ &= p(u_1) \cdot p(u_2) \cdot \cdots \cdot p(u_L), \end{aligned}$$

for all $u^L \in \mathcal{U}^L$. The random variables $U_1, U_2, \cdots, U_L$ can be regarded as the outcomes of L independent trials of the same random experiment. The component entropies $H(U_n), n = 1, 2, \cdots, L$, are all equal to the entropy of the generic random variable U. Therefore we say that

Definition 8. The *entropy* H of a discrete memoryless source is equal to the entropy H(U) of its generic random variable U.

8.5 Fixed-to-fixed length code

In order to describe fixed-to-fixed length source codes we shall consider the communication situation depicted in Fig. 4. A discrete memoryless source generates an L-tuple $(u_1, u_2, \ldots, u_L)$.

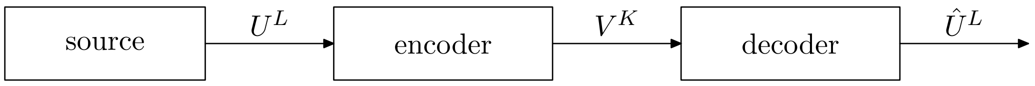


Figure 4: The source coding situation

This L-tuple (*source sequence*) is used as the input to the *encoder.* The encoder assigns to every source sequence another binary sequence , K-tuple, $(v_1, v_2, \ldots, v_K)$. We say that the source sequence u^L is *encoded* into the binary *codeword* v^K. The codeword is then fed into the *decoder* which produces an L-tuple $(\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_L)$ which is supposed to be equal to the source sequence u^L most of the time.

The *ratio* between K and L, i.e. the number of binary digits that is needed to describe a single source symbol is denoted by R_s and is called the *rate.* The *error probability* $P_{e,s}$ is defined as $P_{e,s} \stackrel{\Delta}{=} \Pr\{\hat{U}^L \neq U^L\}$. We say that *reliable compaction* at rate R_s exists if codes exist such that the error probability $P_{e,s}$ can be arbitrarily small.

Without proof we state the following theorem.

Theorem 9. *For any discrete memoryless source with entropy $H(U)$, reliable compaction is possible as long as*

$$R_s > H(U).$$

8.6 The binary source revisited

So, let us apply the previous theorem to a binary memoryless source.

The source alphabet $\mathcal{U} = \{0, 1\}$ and the letter probabilities can be expressed in a single parameter $p \in [0, 1]$, namely $p(0) = 1 - p$ and $p(1) = p$.

The entropy $H(U)$ of this source is

$$H(U) = -\sum_{u \in \{0,1\}} p(u) \log_2 p(u) = -p \log_2 p - (1-p) \log_2 (1-p) = h(p).$$

9 Entropies

9.1 Joint and conditional entropy

Assume that X and Y are discrete random variables that take values in the finite alphabet $\mathcal{X}$ and $\mathcal{Y}$ respectively. Suppose for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ that

$$\begin{aligned} p(x,y) &\stackrel{\Delta}{=} \Pr\{X = x, Y = y\}, \\ p(x) &\stackrel{\Delta}{=} \Pr\{X = x\} = \sum_{y \in \mathcal{Y}} \Pr\{X = x, Y = y\}, \\ p(y) &\stackrel{\Delta}{=} \Pr\{Y = y\} = \sum_{x \in \mathcal{X}} \Pr\{X = x, Y = y\}. \end{aligned}$$

Later we will also use

$$p(x|y) \triangleq \Pr\{X = x|Y = y\}, \text{ and}$$
$$p(y|x) \triangleq \Pr\{Y = y|X = x\}.$$

Definition 10. The *joint entropy* $H(X, Y)$ of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ is defined as

$$H(X, Y) \triangleq -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log_2 p(x, y).$$

If we consider the pair (X, Y) to be a (vector valued) random variable $Z = (X, Y)$ then this definition is a direct consequence of the entropy formula.

Definition 11. The *conditional entropy* of X, given Y, is defined by

$$H(X|Y) \triangleq \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y)\log_2 \frac{p(y)}{p(x, y)}.$$

Just like in the entropy definition we assume that $0\log 0 = 0$. Note that $p(x) = 0$ or $p(y) = 0$ implies that $p(x, y) = 0$. Therefore terms for which $p(x, y) = 0$ do not contribute to the conditional entropy.

9.2 Properties of joint entropy and conditional entropy

Assume that X and Y are discrete random variables that take values in $\mathcal{X}$ and $\mathcal{Y}$ respectively.

Theorem 12 (chain rule)**.**

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

Proof. In the following derivation we use the chain rule for distributions $p(x, y) = p(x)p(y|x)$.

$$\begin{aligned}
H(X, Y) &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log_2 p(x, y)\\
&= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log_2 p(x)\frac{p(x, y)}{p(x)}\\
&= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x)p(y|x)\log_2 p(x) + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log_2 \frac{p(x)}{p(x, y)}\\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2 p(x) + H(Y|X)\\
&= H(X) + H(Y|X)
\end{aligned}$$

□

Theorem 13. *The conditional entropy $H(X|Y) \geq 0$. $H(X|Y) \leq H(X)$ with equality only if X and Y are independent.*

Proof. The conditional entropy $H(X|Y)$ is nonnegative since always $p(y) \geq p(x,y)$, so the terms are all non-negative.

Using $p(x) = \sum_{y\in\mathcal{Y}} p(x,y)$ we find that :

$$\begin{aligned}
H(X|Y) - H(X) &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2\frac{p(y)}{p(x,y)} - \sum_{x\in\mathcal{X}} p(x)\log_2\frac{1}{p(x)} \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2\frac{p(y)}{p(x,y)} - \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2\frac{1}{p(x)} \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2\frac{p(x)p(y)}{p(x,y)}.
\end{aligned}$$

Let $\mathcal{S}$ be the set of pairs (x,y) for which $p(x,y) > 0$. Then we continue

$$\begin{aligned}
H(X|Y) - H(X) &= \frac{1}{\ln 2}\sum_{(x,y)\in\mathcal{S}} p(x,y)\ln\frac{p(x)p(y)}{p(x,y)} \\
&\leq \frac{1}{\ln 2}\sum_{(x,y)\in\mathcal{S}} p(x,y)\left[\frac{p(x)p(y)}{p(x,y)} - 1\right] \\
&= \frac{1}{\ln 2}\left[\sum_{(x,y)\in\mathcal{S}} p(x)p(y) - \sum_{(x,y)\in\mathcal{S}} p(x,y)\right] \\
&\leq \frac{1}{\ln 2}\left[\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x)p(y) - \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\right] = 0.
\end{aligned}$$

The first inequality holds with equality only if $p(x,y) = p(x)p(y)$ for all (x,y) for which $p(x,y) > 0$. The second inequality holds with equality only if $p(x)p(y) = 0$ for (x,y) for which $p(x,y) = 0$. Therefore $H(X) = H(X|Y)$ only if $p(x,y) = p(x)p(y)$ for all $x \in \mathcal{X}$ and all $y \in \mathcal{Y}$. □

9.3 Interpretation of conditional entropy

The conditional entropy $H(X|Y)$ can be interpreted as the (average) amount of information that X contains, after Y has been revealed. To see this note that

$$H(X|Y=y) = \sum_{x\in\mathcal{X}} p(x|y)\log_2\frac{1}{p(x|y)}, \text{ for } y \text{ such that } p(y) > 0,$$

then we can rewrite

$$\begin{aligned}
H(X|Y) &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2\frac{p(y)}{p(x,y)} \\
&= \sum_{y\in\mathcal{Y}} p(y)\sum_{x\in\mathcal{X}} p(x|y)\log_2\frac{1}{p(x|y)} = \sum_{y\in\mathcal{Y}} p(y)\cdot H(X|Y=y).
\end{aligned}$$

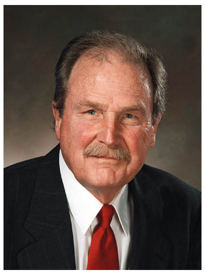
Now $H(X|Y)$ is just the average of all entropies $H(X|Y=y)$ of X for fixed y, over all $y \in \mathcal{Y}$.

Part IV

Huffman codes

10 Introduction

10.1 David A. Huffman



David Huffman earned his B.S. in electrical engineering from Ohio State University at the age of 18 in 1944. He subsequently earned his M.S. degree from Ohio State in 1949 and his Ph.D. from MIT in 1953, also in electrical engineering.

Huffman joined the faculty at MIT in 1953. In 1967, he went to University of California, Santa Cruz as the founding faculty member of the Computer Science Department. He played a major role in the development of the department's academic programs and the hiring of its faculty, and served as chair from 1970 to 1973. He retired in 1994, but remained active as an emeritus professor, teaching information theory and signal analysis courses. Huffman made important contributions in many other areas, including information theory and coding, signal designs for radar and communications applications, and design procedures for asynchronous logical circuits. As an outgrowth of his work on the mathematical properties of ”zero curvature” surfaces, Huffman developed his own techniques for folding paper into unusual sculptured shapes (which gave rise to the field of computational origami).

Huffman's accomplishments earned him numerous awards and honors. Most recently, he received the 1999 Richard Hamming Medal from the Institute of Electrical and Electronics Engineers (IEEE) in recognition of his exceptional contributions to information sciences. He also received the Louis E. Levy Medal of the Franklin Institute for his doctoral thesis on sequential switching circuits, a Distinguished Alumnus Award from Ohio State University, and the W. Wallace McDowell Award. He was a charter recipient of the Computer Pioneer Award from the IEEE Computer Society, and he received a Golden Jubilee Award for Technological Innovation from the IEEE Information Theory Society in 1998.

David Huffman died in 1999 after a 10-month battle with cancer.

Huffman never tried to patent an invention from his work. Instead, he concentrated his efforts on education. In Huffman's own words, “My products are my students.” (from Wikipedia).

10.2 Problem statement

Previously we considered source coding with arbitrarily small but *positive* probability of error. If we look carefully at the encoders and decoders used there we see that the encoder itself is the source of the errors, because it maps several source sequences onto the same codeword.

We should assign a unique codeword to every possible source sequence U^L of *fixed* length L. We will see that we must add extra conditions to the code words.

11 Data compaction codes

11.1 Fixed-to-variable length source coding

Let us now consider Morse's idea where we assign codewords of different lengths to the different source sequences. In the Morse code the most likely letters of the alphabet are represented by short codewords (dots and dashes) and the more unlikely letters received longer codewords. In this way the expected codeword length can be smaller than the length of a fixed-to-fixed length codeword.

We shall consider memoryless sources and the encoding of single letters U, i.e. "sequences" of length 1. The source is fully described by the alphabet $\mathcal{U}$ and the source probabilities $\Pr\{U = u\} = p(u)$. The codewords are selected from a D-ary code alphabet $\mathcal{V}$. We shall use the alphabet $\mathcal{V} = \{0, 1, \cdots, D-1\}$.

The *code* for this source is described by the set of codewords and the mapping from the source symbols to the codewords. We shall write $c(u)$ for the codeword of a source symbol u. So, the code C can be given as $\{c(1), c(2), \cdots, c(M)\}$, if the source alphabet $\mathcal{U}$ is given as $\{1, 2, \cdots, M\}$.

The length of the codeword for symbol u, i.e. the length of the codeword $c(u)$, is denoted by $n(u)$.

The expected codeword length $\bar{n}$ can now be stated as

$$\bar{n} = \sum_{u \in \mathcal{U}} p(u) n(u).$$

Example: *Let the source be given by* $\mathcal{U} = \{1, 2, 3, 4\}$ *and* $p(1) = 1/2$, $p(2) = 1/4$, $p(3) = 1/8$, *and* $p(4) = 1/8$. *The code alphabet is binary,* $D = 2$.
The entropy $H(U)$ *of this source is* 1.75 *bits.*
The code is given by the following four codewords:

$$c(1){=}0,\quad c(2){=}10,\quad c(3){=}110,\quad c(4){=}111.$$

The expected codeword length $\bar{n}$ *is 1.75 codesymbol per source symbol. And because the code alphabet is binary we see that indeed we achieve the source entropy (on the average).*

Example: *Now consider a ternary source,* $\mathcal{U} = \{1, 2, 3\}$ *with* $p(1) = p(2) = p(3) = 1/3$. *The entropy is* $H(U) = \log_2 3 = 1.58$ *bits.*
The code is given by the following codewords:

$$c(1){=}0,\quad c(2){=}10,\quad c(3){=}11.$$

The expected codeword length $\bar{n}$ *is 1.66 codesymbol per source symbol. So the code does not achieve the source entropy even though it seems (and actually is) the best possible binary code for this source!*

Not every set of sequences over the code alphabet (of varying lengths) can be considered a set of codewords. Not only do we want every codeword to be unique, but also we must be able to send a series of codewords.

Example: *Consider again the code from the first example. Assume that the source produced the sequence* $u_1u_2u_3\ldots = 214\ldots$. *The codewords that are output by the encoder are* 10, 0, *and* 111 *respectively. However, the decoder will see the sequence* $100111\ldots$ *and has to be able to reproduce* $u_1u_2u_3\ldots$.

Assuming that the sequence starts with a codeword the decoder knows, after observing the first symbol ("1") that the first source symbol cannot be $u_1 = 1$ *because then the code sequence had to start with a "0". The next symbol ("0") shows the decoder that* $c(2)$ *is a possible codeword. Also* $c(3)$ *and* $c(4)$ *start with two ones so* $u_1 = 2$ *is decoded. The next symbol is a "0" and thus only* $c(1)$ *is a possible candidate codeword. So,* $u_2 = 1$. *The next symbol is a "1", excluding* $c(1)$. *Then follows another "1", so* $c(2)$ *becomes impossible. Finally the third "1" identifies* $c(4)$ *uniquely and we set* $u_3 = 4$.

The code in the previous example is an example of a so called *prefix code.*

Definition 14. A code is called a *prefix code* if no codeword of the code is a prefix of another codeword in the code.

A sequence x^i is a prefix of a sequence y^j if $i \leq j$ and the initial part of y^j is x^i, i.e. if $y^i = x^i$.

As shown in the example above we see that any sequence of codewords from a prefix code can be decoded into the separate codewords. We do this by looking at the initial part of the codeword sequence until we recognize a codeword. Because no other codeword can start with this sequence we know that this is the actual codeword and we take it off. From the remaining sequence we can split off the next codeword in the same way and so on.

11.2 The Kraft inequality

We can represent the set of codewords from a prefix code in a *D-ary tree.* In a D-ary tree every (internal) node has D successors. The branches are labeled by the D symbols from $\mathcal{V}$ and then we can assign to every node and leaf of the tree a sequence of code symbols by listing the sequence of symbols along the (unique) path from the root of the tree to that node or leaf. For every tree the set of sequences of all leaves form a prefix code and, vice versa, every prefix code can be represented by the leaves of a tree.

Example: *Consider the code,* $\{0, 10, 110, 111\}$. *Figure 5 describes the tree that corresponds to this set.*

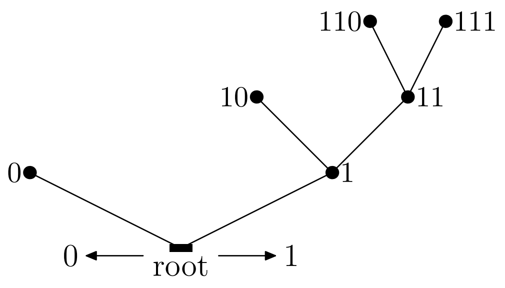


Figure 5: A binary tree and the corresponding sequences.

That the set of sequences, defined by the leaves of a tree, form a prefix code is easy to see. If a sequence x^i is a prefix of a sequence y^j, $j \geq i$, then the node corresponding to x^i must lie on the path from the root to y^j. So, because only leaves are codewords x^i and y^j cannot both be codewords. See figure 6. Conversely it is also clear that a prefix code define a D-ary tree where the codewords are leaves. Namely, start with a complete tree whose depth is not less than the maximal codeword length and cut off all branches from the nodes defined by the codewords. If some parts of the tree are not used they can be cut back as far as possible. See figure 6 with the binary prefix code $\{00, 010, 011, 10\}$.

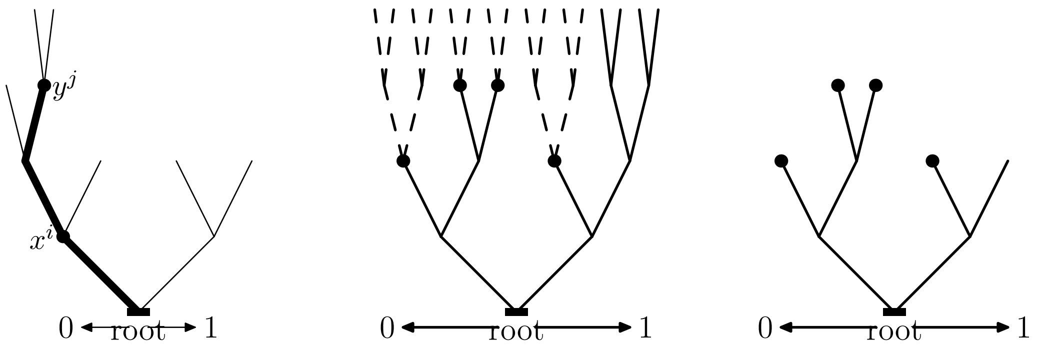


Figure 6: The Kraft inequality.

The following condition is necessary and sufficient for the existence of prefix codes whose codewords have lengths $n(u)$, for all $u \in \mathcal{U}$.

Theorem 15 (Kraft inequality)**.** *If the code C with codewords of lengths $n(u)$ for all $u \in \mathcal{U}$ is a prefix code then it must satisfy*

$$\sum_{u \in \mathcal{U}} D^{-n(u)} \leq 1.$$

Also, if a set of codeword lengths $n(u)$ satisfies the inequality given above then there exists a prefix code with codewords of those lengths.

Example: *Again consider the code $\{0, 10, 110, 111\}$. The lengths are $n(1) = 1$, $n(2) = 2$, $n(3) = 3$, and $n(4) = 3$. We compute, with $D = 2$,*

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1.$$

So this prefix code satisfies the Kraft inequality.

Now take the following code.

$c(1)$=0,
$c(2)$=01,
$c(3)$=110,
$c(4)$=111.

This code has the same lengths as the previous one and so they also satisfy the Kraft inequality. However, this code is not a prefix code because $c(1)$ is a prefix of $c(2)$. Moreover, the code is not decodable. Assume that the codeword sequence is 001110*. This can be decoded as* $0, 0, 111, 0 \Rightarrow 1141$ *but also as* $0, 01, 110 \Rightarrow 123$.

Proof of the first part of theorem 15. Given a prefix code C over a D-ary alphabet. Let $n_{\max}$ be the length of the largest codeword. Consider the complete D-ary tree of depth $n_{\max}$ whose leaves represent C. See figure 7. With this figure we see that the codeword $c(u)$ with length $n(u)$ has $D^{n_{\max}-n(u)}$ leaves as its successors at level $n_{\max}$. Because the code is a prefix

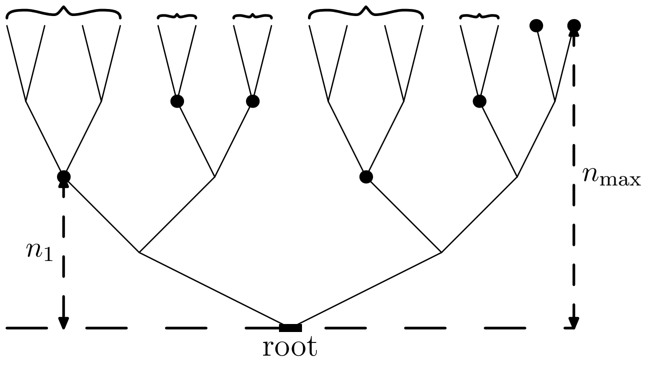


Figure 7: The complete tree.

code it is impossible that two different codewords have successors at level $n_{\max}$ in common. And because the number of leaves at level $n_{\max}$ is equal to $D^{n_{\max}}$ we find

$$\sum_{u \in \mathcal{U}} D^{n_{\max} - n(u)} \leq D^{n_{\max}},$$

or

$$\sum_{u \in \mathcal{U}} D^{-n(u)} \leq 1.$$

□

Proof of the second part of theorem 15. Assume that we are given a set of codeword lengths $n(u)$ that satisfies Kraft's inequality. Consider a complete D-ary tree of depth $n_{\max}$, where $n_{\max}$ is again the maximum of all codeword lengths. Take the first length $n(u_1)$ and label the leftmost node at depth $n(u_1)$ in the tree as codeword $c(u_1)$. Remove all successors of this node from the tree. The label the leftmost remaining node at depth $n(u_2)$ in the tree as codeword $c(u_2)$, etc. Continue until all codewords are created.

Because we remove at each step the $D^{n_{\max} - n(u)}$ leftmost leaves and Kraft's inequality is satisfied, we can indeed create all codewords in this way. □

11.3 Bounds on the code rate

The (expected) code rate R_s is defined as the expected number of code symbols per source symbol, so $R_s = \bar{n}$ the expected codeword length. We ask ourselves what is the best possible code rate?

Because our codeword lengths must satisfy Kraft's inequality we have the following minimization problem: minimize the average code wordlength under the condition that the integers $n(u)$ satisfy Kraft's inequality.

If we allow arbitrary real $n(u)$ then the solution is

$$\bar{n}_{\text{opt}} = H(U),$$

where the entropy is expressed in base D and the optimal "codeword lengths" are given by

$$n_{\text{opt}}(u) = -\log_D p(u).$$

This can easily be verified using e.g. the Lagrange multiplier method.

So, if we could use non-integer codeword lengths then $-\log_D p(u)$ would be the best possible codeword length. For that reason we call $-\log_D p(u)$ the *ideal codeword length.* Also, the entropy $H(U)$ describes the expected amount of information that a source symbol contains. Note that $H(U)$ is the expectation over $p(u)$ of $-\log_D p(u)$. For that reason we also name $-\log_D p(u)$ the *self information* of the symbol u.

In what follows we shall make use of the following concept.

Definition 16. The *relative entropy* or *divergence* between two probability vectors $p(u)$ and $q(u)$ over an alphabet $\mathcal{U}$ is defined as

$$D(p\|q) \triangleq \sum_{u\in\mathcal{U}} p(u)\log\frac{p(u)}{q(u)}.$$

The basis of the logarithm determines the unit just as with the entropy.

We again use the conventions, as with the entropy,

$$\begin{aligned} 0\log\frac{0}{q} &= 0; \quad \text{for all } q,\\ p\log\frac{p}{0} &= \infty; \quad \text{for all } p. \end{aligned}$$

Some properties of the divergence are

- $D(p\|q) \neq D(q\|p)$ in general. This is clear from the definition of $D(p\|q)$.
- $D(p\|q) \geq 0$ with equality only if $p(u) = q(u)$ for all $u \in \mathcal{U}$.

Proof. We shall use the log-inequality again.

$$\begin{aligned} \sum_{u\in\mathcal{U}} p(u)\log_2\frac{q(u)}{p(u)} &= -D(p\|q)\\ &\leq \log_2 e \sum_{u\in\mathcal{U}} p(u)\left(\frac{q(u)}{p(u)} - 1\right)\\ &= \log_2 e\left(\sum_{u\in\mathcal{U}} q(u) - \sum_{u\in\mathcal{U}} p(u)\right)\\ &= 0 \end{aligned}$$

and so

$$D(p\|q) \geq 0.$$

□

Now we return to our prefix code C. The codeword lengths are $n(u)$ and let the "Kraft sum" be

$$\sum_{u\in\mathcal{U}} D^{-n(u)} = \alpha,$$

for some α, with $0 < \alpha \leq 1$. Note that $q(u) = D^{-n(u)}/\alpha$ can be seen as a probability vector, namely $q(u) \geq 0$ for all $u \in \mathcal{U}$ and $\sum_{u\in\mathcal{U}} q(u) = 1$.

Consider the difference between the expected codeword length and the source entropy.

$$\bar{n} - H(U) = \sum_{u\in\mathcal{U}} p(u)n(u) + \sum_{u\in\mathcal{U}} p(u)\log_D p(u)$$

and using $n(u) = -\log_D(D^{-n(u)})$

$$\begin{aligned}
\bar{n} - H(U) &= -\sum_{u\in\mathcal{U}} p(u)\log_D(D^{-n(u)}) + \sum_{u\in\mathcal{U}} p(u)\log_D p(u) \\
&= \sum_{u\in\mathcal{U}} p(u)\log_D \frac{p(u)}{D^{-n(u)}} \\
&= \sum_{u\in\mathcal{U}} p(u)\log_D \frac{p(u)}{\alpha q(u)} \\
&= D(p\|q) - \log_D \alpha
\end{aligned}$$

and with $\alpha \leq 1$ and $D(p\|q) \geq 0$ we find

$$\bar{n} - H(U) \geq 0.$$

Now assuming that our code is such that the Kraft inequality is satisfied with equality, which can always be done for binary codes without increasing the expected codeword length, then we see that the divergence measures the *excess codeword length* also known as the *redundancy* $\bar{r} \triangleq \bar{n} - H(U)$.

Can we also find an upperbound on $\bar{n}$ for "good" codes? Consider the fact that the ideal codeword length is equal to $-\log_D p(u)$. Define a code C by setting the codeword lengths equal to

$$n(u) = \lceil -\log_D p(u) \rceil .$$

Here $\lceil x \rceil$ is the smallest integer not less than x.

These codeword lengths satisfy the Kraft inequality as can easily be verified using $p(u) = D^{\log_D p(u)} \geq D^{-\lceil -\log_D p(u)\rceil}$. Using the procedure as described in Theorem 15 the codewords can be found with the correct lengths $n(u)$.

The expected codeword length is now bounded as

$$\begin{aligned}
\bar{n} &= \sum_{u\in\mathcal{U}} p(u)n(u) \\
&= \sum_{u\in\mathcal{U}} p(u)\lceil -\log_D p(u)\rceil \\
&< \sum_{u\in\mathcal{U}} p(u)(-\log_D p(u) + 1) \\
&= H(U) + 1
\end{aligned}$$

This might not be the best possible code, so $H(U) + 1$ will also be an upperbound to the expected codeword length of the optimal code. Summarizing we find for the optimal code

$$H(U) \leq \bar{n} < H(U) + 1.$$

The $+1$ is the result of the rounding off to integers of the ideal codeword lengths. It seems reasonable to spread this cost over more symbols, so consider now the coding of a source sequence of k symbols at once. So, we design a code for a source with probabilities $p(u^k)$. This results in an expected codeword length bounded by

$$H(U^k) \leq \bar{n} < H(U^k) + 1.$$

Now the expected coderate R_s is given as

$$R_s = \frac{\bar{n}}{k},$$

and we find

$$\frac{H(U^k)}{k} \leq R_s < \frac{H(U^k)}{k} + \frac{1}{k}.$$

For a memoryless source we know that $\frac{H(U^k)}{k} = H(U)$.

A code like this is known as a *fixed-to-variable length* code.

This result will be stated in the following *variable length source coding theorem.*

Theorem 17. *For a stationary source U there exists fixed-to-variable length codes such that as k, the block or source sequence length, goes to infinity then the coderate R_s approaches the source entropy $H(U)$ or the source entropy rate $H_\infty(U)$.*

Conversely, it is not possible to have error-free fixed-to-variable lengths compression with a rate less than the source entropy (rate).

The proof of this theorem is already given above.

12 Huffman codes

12.1 The optimal code

There are a finite number of binary prefix codes for a given alphabet size, so there must be an optimal prefix code, i.e. with minimal expected codeword length.

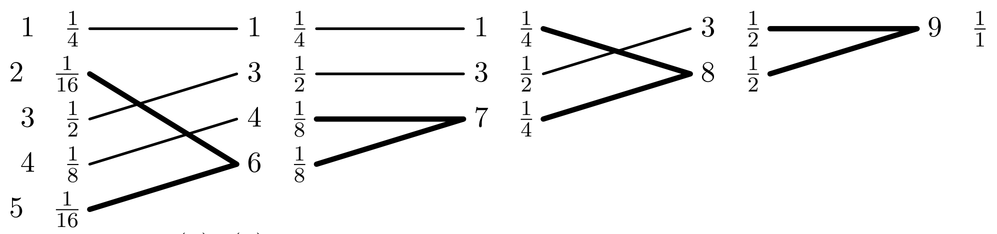
The Huffman procedure produces these optimal codes. First we shall describe the procedure and then we show the optimality of the resulting codes.

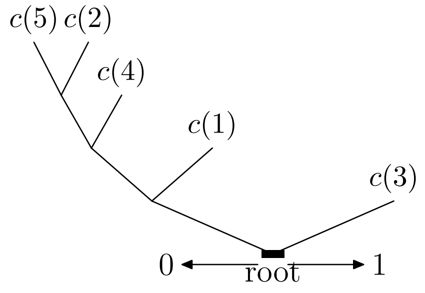
12.2 The binary Huffman algorithm

We shall explain the procedure using an example.

Example: *Consider a memoryless source U with alphabet $\mathcal{U} = \{1, 2, 3, 4, 5\}$ and probabilities $p(1) = 1/4$, $p(2) = 1/16$, $p(3) = 1/2$, $p(4) = 1/8$, and $p(5) = 1/16$. The binary Huffman procedure constructs a binary tree starting at the leaves. At the end of the procedure, the resulting tree describes the code.*

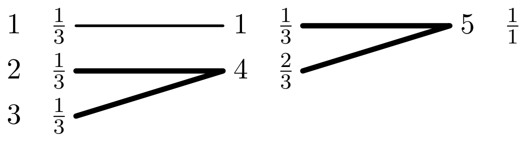
We first combine the two least probable symbols, i.e. "2" and "5", into one new symbol, say "6" with probability $p(6) = p(2) + p(5) = 1/8$. The codeword we shall assign to "2" resp. "5" will be the codeword assigned to the dummy symbol "6" followed by a zero resp. a one. In the figure below we indicate this by two bold branches connecting "2" and "5" to "6". We repeat tthe combining of the two least probable symbols over the new alphabet $\{1, 3, 4, 6\}$ and so combine "4" and "6" into the symbol "7" with $p(7) = p(4) + p(6) = 1/4$. And so on, until only one symbol is left.

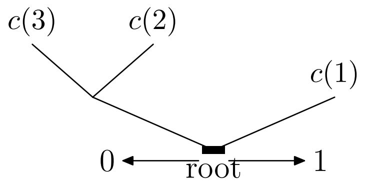




$$\begin{aligned} c(1) &= 01 \\ c(2) &= 0001 \\ c(3) &= 1 \\ c(4) &= 001 \\ c(5) &= 0000 \end{aligned}$$

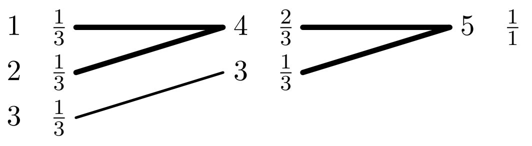
Example: Consider a memoryless source U with alphabet $\mathcal{U} = \{1, 2, 3, \}$ and probabilities $p(1) = p(2) = p(3) = 1/3$. The binary Huffman procedure is displayed below.

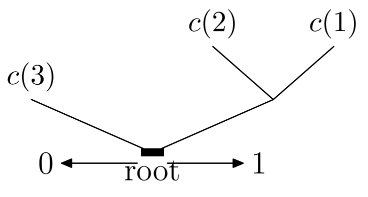




$$\begin{aligned} c(1) &= 1 \\ c(2) &= 01 \\ c(3) &= 00 \end{aligned}$$

Note that we had a choice in the first step. In stead of combining "2" and "3" we could also have combined "1" and "2" and that resulted in another code, see below. Note however, that these codes are equal in terms of coderate. For some sources we indeed can have many equivalent Huffman codes.





$$\begin{aligned} c(1) &= 11 \\ c(2) &= 10 \\ c(3) &= 0 \end{aligned}$$

12.3 Optimality of binary Huffman codes

Theorem 18. *There exists optimal binary codes for which the codewords, associated with the two least probable source symbols, are equally long and differ only in the last binary digit.*

Proof. Assume that the symbols are ordered such that $p(u_1) \geq p(u_2) \geq \ldots \geq p(u_{M-1}) \geq p(u_M)$, where $M = |\mathcal{U}|$ is the size of the source alphabet. It is clear that $n(u_1) \leq n(u_2) \leq \ldots \leq n(u_{M-1}) \leq n(u_M)$. This follows from the fact that if $p(u_i) > p(u_{i+1})$ and $n(u_i) > n(u_{i+1})$ then the code is not optimal, exchanging codeword $c(u_i)$ and $c(u_{i+1})$ improves the code. Thus the least probable symbols, u_{M-1} and u_M, have the largest codeword lengths $n(u_{M-1})$ and $n(u_M)$.

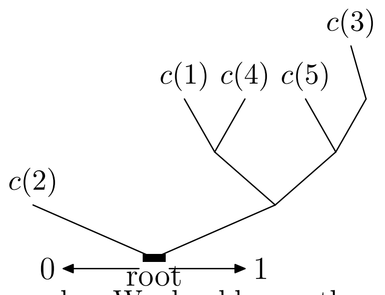
First we can assume that the code, say C, is prefix-free. This implies that $c(u_i)$ and $c(u_M)$ differ at least in a symbol position $j \leq n(u_i) \leq n(u_M)$. In particular this holds for $c(u_{M-1})$ and $c(u_M)$ and thus we can conclude that if $n(u_M) > n(u_{M-1})$ then we can shorten $c(u_M)$ to $n(u_M) = n(u_{M-1})$ thereby improving the code.

Now consider the last and longest codeword $c(u_M)$. We know that there must exist another codeword of the same length, say $c(u_k)$ that only differs from $c(u_M)$ in the last symbol. If there would be no such codeword then $c(u_M)$ could be shortened, which is impossible because it is the longest word and $c(u_{M-1})$ has the same length. If this codeword is not $c(u_{M-1})$, we can swap $c(u_k)$ and $c(u_{M-1})$. $\square$

This theorem tells us that we can find an optimal code by minimizing $\bar{n}$ over all codes where the two least probable symbols are assigned two longest codewords that differ only in the last symbol.

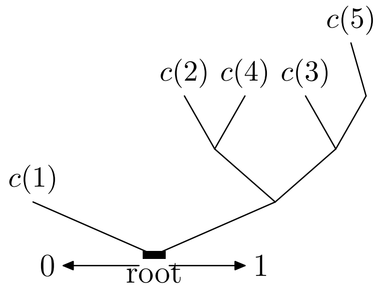
Example: Consider the following memoryless source U defined by $\mathcal{U} = \{1,2,3,4,5\}$ and $p(1) = 7/16$; $p(2) = 3/16$; $p(3) = 5/32$; $p(4) = 1/8$; $p(5) = 3/32$. The original code C is described by the codewords

$$\begin{aligned} c(1) &= 100 & n(1) &= 3 \\ c(2) &= 0 & n(2) &= 1 \\ c(3) &= 1110 & n(3) &= 4 \\ c(4) &= 101 & n(4) &= 3 \\ c(5) &= 110 & n(5) &= 3 \end{aligned} \qquad \bar{n} = 2.7813$$



Note that the codeword lengths are not in the proper order. We should swap the codewords $c(1)$ and $c(2)$ and $c(3)$ and $c(5)$ to obtain

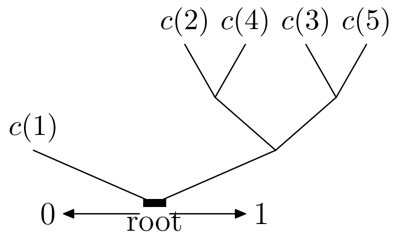
$$\begin{aligned} c(1) &= 0 & n(1) &= 1 \\ c(2) &= 100 & n(2) &= 3 \\ c(3) &= 110 & n(3) &= 3 \\ c(4) &= 101 & n(4) &= 3 \\ c(5) &= 1110 & n(5) &= 4 \end{aligned} \qquad \bar{n} = 2.2188$$



We have improved $\bar{n}$ in the process.

The last two codewords are not of equal length so we must be able to shorten $c(5)$.

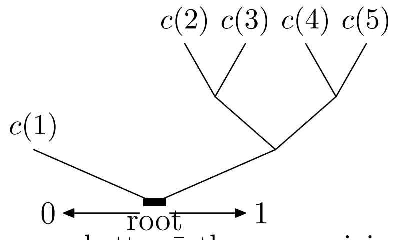
$$\begin{aligned} c(1) &= 0 & n(1) &= 1 \\ c(2) &= 100 & n(2) &= 3 \\ c(3) &= 110 & n(3) &= 3 \\ c(4) &= 101 & n(4) &= 3 \\ c(5) &= 111 & n(5) &= 3 \end{aligned} \qquad \bar{n} = 2.125$$



We have improved $\bar{n}$ again.

Because $c(5)$ cannot be shortened any further (it must be the longest word and after the previous step $n(5) = n(4)$) there must be a codeword differing only with $c(5)$ in the last digit. This is $c(3)$ so we swap $c(3)$ and $c(4)$.

$$\begin{array}{lcllcl} c(1) & = & 0 & n(1) & = & 1 \\ c(2) & = & 100 & n(2) & = & 3 \\ c(3) & = & 101 & n(3) & = & 3 \\ c(4) & = & 110 & n(4) & = & 3 \\ c(5) & = & 111 & n(5) & = & 3 \end{array} \qquad \bar{n} = 2.125$$



Now our code satisfies theorem 18 and it has an even better $\bar{n}$ than our original code.

The next theorem shows the optimality of Huffmans procedure by considering the reduced source.

Definition 19. A *Reduced Source* U' is derived from a discrete and memoryless source U by defining the reduced alphabet $\mathcal{U}'$ as equal to the alphabet $\mathcal{U}$ minus the two least probable symbols u_{M-1} and u_M and adding a new symbol u'_{M-1}. The probabilities $p'(u)$ for all symbols $u \in \mathcal{U}'$ except the new symbol are the same as for U and $p'(u'_{M-1}) = p(u_{M-1}) + p(u_M)$.

Theorem 20. *Let C' be an optimal code for the reduced source U'. Let the code C be derived from C' by creating two new codewords and removing an old one. We extend the codeword for u'_{M-1} by a zero and also by a one and assign these words to u_{M-1} and u_M.*

The code C thus formed, is optimal for the source U.

Proof. Note that the code C belongs to the class of codes that is described in theorem 18. Also note that the expected codeword length $\bar{n}$ for C can be expressed in $\bar{n}'$ as follows.

$$\bar{n} = \bar{n}' + p(u_{M-1}) + p(u_M).$$

This expected length depends only on the expected length of the reduced source and by using an optimal code $\bar{n}$ is also minimal, so C is optimal. □

12.4 Non-binary Huffman codes

The binary codes we considered up to now can be described by *complete binary trees.* In a complete binary tree, every node is either a leaf, and has no successors, or it is an internal node and has exactly *two* successors. For every positive integer M there exists complete binary trees with M leaves corresponding to the binary codes we need.

If the code alphabet is non-binary, say D-ary with $D > 2$, we consider *complete D-ary trees* in an analogous way, where every internal node now has D successors. However, it is *not* true that we can find complete D-ary trees with M leaves for arbitrary integer M. So we must ask the question how we can produce a D-ary code with M codewords. The answer is simply to consider a slightly larger complete tree and not use some of the codewords.

If the code alphabet size is D, we can find complete trees with M leaves whenever we have, for any non-negative integer α that

$$M = 1 + \alpha(D - 1).$$

This can easily bee seen when we realize that every complete tree can be build up by replacing a leaf by an internal node with D leaves as its successors.

Let us assume that for a given D, M satisfies the equation above. It is easy to see that Theorem 18 holds with the small change that now the D least probable source symbols have equal length codewords that differ only in the last symbol. Also Theorem 20 holds when we define the reduced source as the source where the D least probable symbols are combined into one new symbol.

The result is that we produce optimal, Huffman, codes by combining, in each step, the D least probable symbols of the (reduced) source.

What will happen when the equation on tree leaves does not hold? We simply add some dummy symbols, with probability equal to zero, such that our "new" source has M' symbols that satisfy the equation. The Huffman code for this source is of course also optimal four our original source and we just ignore the codewords assigned to the dummy symbols.

Part V

Stationary sources

13 Introduction

13.1 Andrey A. Markov

Andrey (Andrei) Andreyevich Markov was a Russian mathematician. He is best known for his work on theory of stochastic processes. His research later became known as Markov chains.

In 1866 Andrey Andreevichs school life began with his entrance into Saint Petersburgs fifth grammar school. Already during his school time Andrey was intensely engaged in higher mathematics. As a 17 year-old grammar school student he informed Bunyakovsky, Korkin and Yegor Zolotarev about an apparently new method to solve linear ordinary differential equations and was invited to the so-called Korkin Saturdays, where Korkin's students regularly met. In 1874 he finished the school and began his studies at the physico-mathematical faculty of St Petersburg University.

In 1877 he was awarded the gold medal for his outstanding solution of the problem "About Integration of Differential Equations by Continuous Fractions with an Application to the Equation $(1+x^2)\frac{dy}{dx} = n(1+y^2)$". In the following year he passed the candidate examinations and remained at the university to prepare the lecturers job.

In April, 1880 Andrey Markov defended his master thesis "About Binary Quadratic Forms with Positive Determinant", which was encouraged by Aleksandr Korkin and Yegor Zolotarev.

Five years later, in January 1885, there followed his doctoral thesis "About Some Applications of Algebraic Continuous Fractions".

One year after the defense of the doctoral thesis, he was appointed extraordinary professor (1886) and in the same year he was elected adjunct to the Academy of Sciences. His promotion to an ordinary professor of St Petersburg University followed in autumn 1894.

Finally in 1896, he was elected ordinary member of the Academy of Sciences as the successor of Chebyshev. In 1905 he was appointed merited professor and got the right to retire which he immediately used. Till 1910, however, he continued to lecture calculus of differences. (from Wikipedia).

13.2 Problem statement

For discrete memoryless sources we know that the entropy of a vector is equal to L times the single letter entropy. This result depends on the independence assumption that is valid for memoryless sources.

There exist a more general class of sources where the successive source outputs are not independent. In that case we know that the entropy of the vector U^L must be smaller than L times the single letter entropy. Now it seems useful to study the behavior of the entropy for vectors of increasing length.

13.3 Discrete stationary sources

First we shall discuss briefly the class of sources known as the *stationary sources.* Stationary sources produce infinite length output strings $\cdots, U_{-2}, U_{-1}, U_0, U_1, U_2, \cdots$ and assign probabilities to arbitrary source output vectors, with the following two restrictions.

1. The probabilities must be *consistent.* This means that if $J = \{i_1, i_2, \cdots, i_n\}$ is a set of n indices, for arbitrary n, and j is an index that is not in J, then for all possible symbol values u_k, for $k = j$ or $k \in J$, the following holds.

$$\begin{aligned}\Pr\{U_{i_1} = u_{i_1}, &U_{i_2} = u_{i_2}, \cdots, U_{i_n} = u_{i_n}\} = \\ &\sum_{u_j \in \mathcal{U}} \Pr\{U_{i_1} = u_{i_1}, U_{i_2} = u_{i_2}, \cdots, U_{i_n} = u_{i_n}, U_j = u_j\}.\end{aligned}$$

Example: *An example will clarify this condition.*
Suppose $\mathcal{U} = \{0, 1, 2\}$ *and the source assigns* $\Pr\{U_1 = 0, U_2 = 0\} = 1/2$*,* $\Pr\{U_1 = 0, U_2 = 1\} = 1/8$*, and* $\Pr\{U_1 = 0, U_2 = 2\} = 1/8$*. Then by summing over* $U_2 \in \mathcal{U}$ *we find* $\Pr\{U_1 = 0\} = 3/4$*. If the source does not assign this probability to* $\Pr\{U_1 = 0\}$ *it is not consistent.*

2. The probabilities only depend on their relative positions, not on the absolute positions. So they are *shift invariant.* If again J is a set of n indices, then

$$\begin{aligned}\Pr\{U_{i_1} = u_{i_1}, &U_{i_2} = u_{i_2}, \cdots, U_{i_n} = u_{i_n}\} = \\ &\Pr\{U_{i_1+1} = u_{i_1}, U_{i_2+1} = u_{i_2}, \cdots, U_{i_n+1} = u_{i_n}\}.\end{aligned}$$

13.4 The entropy rate

The entropy $H(U^L)$ usually is an unbounded non-decreasing function of L, because it is reasonable to assume that for many sources most source outputs contribute to the entropy, i.e. that for an infinite number of source outputs U_i the conditional entropy $H(U_i|U^{i-1})$ is strictly positive. For this reason it is more useful to study the *averaged* per letter entropy defined by

$$H_L(U) \triangleq \frac{1}{L} H(U^L).$$

We shall now consider the behavior of $H_L(U)$ as L grows. First note that for a memoryless source with entropy $H(U)$ we have

$$H_L(U) = \frac{1}{L} H(U^L) = \frac{1}{L} L H(U) = H(U).$$

So, in this case the *averaged* per letter entropy is indeed the actual per letter entropy.

As it will turn out the conditional entropy $H(U_L|U^{L-1})$ also known as the *innovation entropy*, plays the same role as $H_L(U)$. The following theorem will describe the behavior of these two entropies.

Theorem 21. *For a stationary source* $\{U_i\}_{i=-\infty}^{\infty}$ *with* $H(U_1) < \infty$ *holds*

1. $H(U_L|U^{L-1})$ *is non-increasing in* L*.*

2. *$H_L(U) \geq H(U_L|U^{L-1})$ for all $L = 1, 2, 3, \cdots$.*

3. *$H_L(U)$ is non-increasing in L.*

4. $\lim_{L\to\infty} H_L(U) = \lim_{L\to\infty} H(U_L|U^{L-1})$.

Proof.

1 Because conditioning cannot increase the entropy we have

$$H(U_L|U_1, U_2, \cdots, U_{L-1}) \leq H(U_L|U_2, \cdots, U_{L-1}).$$

Due to the stationarity we have

$$H(U_L|U_2, \cdots, U_{L-1}) = H(U_{L-1}|U_1, \cdots, U_{L-2}).$$

The non-increasing property now follows from combining these two (in-)equalities.

2 With the chain rule we can write $H_L(U)$ as

$$H_L(U) = \frac{1}{L}\left(H(U_1) + H(U_2|U_1) + \cdots + H(U_L|U^{L-1})\right).$$

From property 1 we know that $H(U_L|U^{L-1})$ is the smallest of the L terms, so $H_L(U) \geq H(U_L|U^{L-1})$.

3 Again with the chain rule and property 2 we write

$$\begin{aligned} H_L(U) &= \frac{1}{L}H(U^{L-1}) + \frac{1}{L}H(U_L|U^{L-1}) \\ &\leq \frac{L-1}{L}H_{L-1}(U) + \frac{1}{L}H_L(U). \end{aligned}$$

By rearranging this inequality we find that $H_L(U)$ is a non-increasing function of L.

4 From property 1 and the fact that entropies cannot be negative, it follows that $\lim_{L\to\infty} H(U_L|U^{L-1})$ exists. Likewise using property 3 we see that $\lim_{L\to\infty} H_L(U)$ exists too. With property 2 we know that

$$\lim_{L\to\infty} H_L(U) \geq \lim_{L\to\infty} H(U_L|U^{L-1}).$$

Now consider the following derivation, where L and j are arbitrary positive integers.

$$\begin{aligned} H_{L+j}(U) &= \frac{1}{L+j}H(U^{L-1}) + \frac{1}{L+j}\left\{H(U_L|U^{L-1}) + \right. \\ &\qquad \left. H(U_{L+1}|U^L) + \cdots + H(U_{L+j}|U^{L+j-1})\right\} \\ &\leq \frac{1}{L+j}H(U^{L-1}) + \frac{j+1}{L+j}H(U_L|U^{L-1}). \end{aligned}$$

Let L be fixed and $j \to \infty$ then we find

$$\lim_{j\to\infty} H_{L+j}(U) \leq \lim_{j\to\infty} \left\{ \frac{1}{L+j} H(U^{L-1}) + \frac{j+1}{L+j} H(U_L|U^{L-1}) \right\} = H(U_L|U^{L-1}).$$

And thus we find

$$\lim_{j\to\infty} H_j(U) \leq H(U_L|U^{L-1}).$$

We must conclude that

$$\lim_{L\to\infty} H_L(U) = \lim_{L\to\infty} H(U_L|U^{L-1}).$$

□

Definition 22. The *entropy rate* $H_\infty(U)$ is defined by

$$H_\infty(U) \stackrel{\Delta}{=} \lim_{L\to\infty} H_L(U).$$

It is possible to prove a source coding theorem that shows that the entropy rate plays the same role for stationary sources as $H(U)$ does for memoryless sources, i.e. it describes the smallest possible code rate for data compression.

14 Markov sources

14.1 Introduction

Markov sources are a special subclass of stationary sources. Many "real world" sources can successfully be approximated by Markov sources and still these Markov "models" are relatively simple.

We shall introduce some Markov sources. Let $\{U_i\}_{i=-\infty}^{\infty}$ be a Markov source. It is described by a finite set of *states*, $\mathcal{S} = \{1, 2, \cdots, J\}$, and a source letter alphabet, $\mathcal{U} = \{a_1, a_2, \cdots, a_K\}$. Furthermore for every state $s \in \mathcal{S}$ we have a probability vector for the source letters, $(P(u|s) \stackrel{\Delta}{=} \Pr\{U = u|S = s\} : u \in \mathcal{U})$. Also, there is a probability vector that describes the next state $S' = s'$ given the current state $S = s$ and the next output $U = u$. We denote this by $(T(s'|s, u) \stackrel{\Delta}{=} \Pr\{S' = s'|S = s, U = u\} : s' \in \mathcal{S})$. If at time instant i the source is in a state s then the next source output $U_{i+1} = u$ is produced and the source enters state $S_{i+1} = s'$ with a probability $Q(u, s'|s) \stackrel{\Delta}{=} P(u|s)T(s'|s, u)$. So we see here an important property of Markov sources:

> the behavior of the source, i.e. the probabilities of selecting an output and a next state, given the current state *does not* depend on past states or outputs.

$$\Pr\{U_{i+1} = u, S_{i+1} = s'|S_i = s, U^{i-1} = u^{i-1}, S^{i-1} = s^{i-1}\} = Q(u, s'|s).$$

Example: *Consider the following binary 2-state Markov source.* $\mathcal{U} = \{0, 1\}$*,* $\mathcal{S} = \{1, 2\}$*, and*

u	s'	s	$Q(u, s'\|s)$		u	s'	s	$Q(u, s'\|s)$
0	1	1	$\frac{1}{2}$		0	1	2	$\frac{1}{4}$
1	1	1	0		1	1	2	0
0	2	1	0		0	2	2	0
1	2	1	$\frac{1}{2}$		1	2	2	$\frac{3}{4}$

Such a source can be depicted as in figure 8.

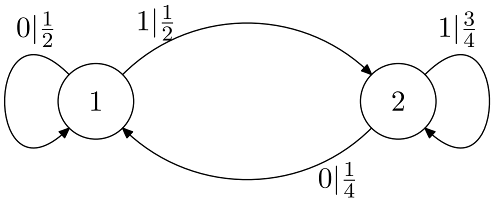


Figure 8: A binary 2-state Markov source.

In this figure the states are depicted by circles with their label in the center. Arrows from one state (s) *to another (or the same) state* (s') *depict non zero* $Q(u, s'|s)$ *values. The arrows are labeled by the source output* u *and the probability* $Q(u, s'|s)$ *as:*

$$u|Q(u, s'|s).$$

For this source we can compute the probability of the sequence $U_1, U_2, U_3, U_4 = 1, 1, 0, 0$ *given that* $S_0 = 1$ *as follows. From* $S_0 = 1$ *and* $U_1 = 1$ *we know* $S_1 = 2$*. Then we have* $(S_1 = 2, U_2 = 1) \to S_2 = 2$ *and* $(S_2 = 2, U_3 = 0) \to S_3 = 1$*. So we compute*

$$\begin{aligned}\Pr\{U_1, &U_2, U_3, U_4 = 1, 1, 0, 0|S_0 = 1\} = \Pr\{U_1 = 1|S_0 = 1\}\times \\ &\Pr\{U_2 = 1|S_1 = 2\} \Pr\{U_3 = 0|S_2 = 2\} \Pr\{U_4 = 0|S_3 = 1\} = \\ &\frac{1}{2}\frac{3}{4}\frac{1}{4}\frac{1}{2} = \frac{3}{64}.\end{aligned}$$

A Markov source exhibits a stationary behavior under certain conditions. We shall consider only those sources for which the probabilities $Q(u, s'|s)$ are time invariant. Also the source must be such that from every state $s \in \mathcal{S}$ we must be able to reach, in a finite number of steps (time instances), any other state $s' \in \mathcal{S}$ with non-zero probability. In particular, this means that for every state $s \in \mathcal{S}$ it must be possible to return to s in a finite number of steps. The *recurrence time* of a state s is the number of steps needed to return to s when started in s. This recurrence time is a random variable. We define the *period* of a set of states as the largest integer m such that all possible recurrence times of all states of the source are a multiple of m. Clearly $m = 1$ in the source of figure 8. If $m = 1$ we say that the source is *stationary and ergodic* and if $m > 1$ then it is *periodic* with period m.

14.2 Stationary and ergodic sources

We shall mainly consider *stationary and ergodic* sources. We define the *state transition probability matrix* $[W_{ji}]$ as

$$\begin{aligned}\forall i \in \mathcal{S}, j \in \mathcal{S} : W_{ji} &\triangleq \Pr\{S_{t+1} = i | S_l = j\}, \\ &= \sum_{u \in \mathcal{U}} Q(u, i|j)\end{aligned}$$

Because $Q(u, s'|s)$ is time invariant, t can be an arbitrary time instant.

Let $\underline{q}(t) \triangleq (q_1(t), q_2(t), \cdots, q_J(t))$ be the *state distribution* at time t, i.e. $q_s(t) \triangleq \Pr\{S_t = s\}$. It is obvious that the state distribution at time $t+1$ depends on $\underline{q}(t)$ and $[W_{ji}]$ as

$$\underline{q}(t+1) = \underline{q}(t)[W_{ji}].$$

Without proof we state the following important result for *stationary and ergodic* sources that says that there exists a unique limiting state distribution $\underline{q}(\infty) = (q_1(\infty),\ q_2(\infty), \cdots,\ q_J(\infty))$ independent from $\underline{q}(1)$ such that

$$\lim_{t \to \infty} \underline{q}(t) = \underline{q}(\infty).$$

Moreover, this state distribution $\underline{q}(\infty)$ is also the solution of

$$\underline{q}(\infty)[W_{ji}] = \underline{q}(\infty).$$

Because of this $\underline{q}(\infty)$ is also known as the *stationary state distribution*, also simply denoted by $\underline{q} = (q_1, q_2, \cdots, q_J)$.

Technically the source was started infinitely far in the past, "$t = -\infty$". This means that $\underline{q}(1) = \underline{q}(\infty)$. Because we usually only care for the subsequence $U_1, U_2, \cdots$ this means that $\underline{q}(t) = \underline{q}(\infty)$ for all relevant t, $t = 1, 2, \cdots$. This implies that the source is really stationary. This is also the case, for our purposes, when the source is started at time $t = 1$ with state distribution $\underline{q}(1) = \underline{q}(\infty)$.

An important distinction between different stationary and ergodic Markov sources can be made, depending on the next state probabilities $T(s'|s, u)$. If $T(s'|s, u)$ only contain zero and one entries then the next state s' is completely determined by the current state s and the next output symbol u. In this case $T(\cdot|\cdot, \cdot)$ is actually a *next state function* and given an initial state S_0 and an output sequence $U_1, U_2, \cdots, U_L$ completely determine the state sequence $S_1, S_2, \cdots, S_L$. We shall use the following definitions to distinguish these cases.

14.3 Unifilar sources

Definition 23. A stationary and ergodic Markov source is called a *unifilar* Markov source if its next state probabilities $T(s'|s, u)$ only contain zero's and ones.

A stationary and ergodic Markov source whose next state probabilities $T(s'|s, u)$ do not contain only zero's and ones is called a *non unifilar* Markov source.

14.4 The entropy rate of a unifilar source

In general the entropy rate $H_\infty(U)$ for a stationary and ergodic Markov source can only be determined by the limit expression. However in the case of unifilar Markov sources the entropy rate can be computed directly as the next theorem will state.

Theorem 24. *If $\{U_i\}_{i=-\infty}^{\infty}$ is an unifilar Markov source then its entropy rate $H_\infty(U)$ is given by*

$$H_\infty(U) = H(U|S) = \sum_{s \in \mathcal{S}} q_s H(U|S = s).$$

Proof. We consider the entropy $H(U^L, S_0)$ and rewrite it using the chain rule in two different ways.

$$\begin{aligned} H(U^L, S_0) &= H(U^L) + H(S_0|U^L), \\ &= H(S_0) + H(U^L|S_0). \end{aligned}$$

So we can write

$$H(U^L) = H(U^L|S_0) + H(S_0) - H(S_0|U^L).$$

Because the Markov source has a finite number of states $|\mathcal{S}| = J$, and conditioning cannot increase the entropy, we know that

$$0 \le H(S_0|U^L) \le H(S_0) \le \log_2 J.$$

Also using the chain rule and the fact that the current state and current output uniquely determine the next state (unifilar Markov source) we find

$$\begin{aligned} H(U^L|S_0) &= H(U_1|S_0) + H(U_2|S_0, U_1) + \cdots + H(U_L|S_0, U^{L-1}) \\ &= H(U_1|S_0) + H(U_2|S_1) + \cdots + H(U_L|S_{L-1}) \\ &= LH(U|S) \end{aligned}$$

The last line follows from the fact that the source is stationary, so $H(U_1|S_0) = H(U_2|S_1)$ etc. It is also easy to see that $H(U_2|S_0, U_1) = H(U_2|S_1)$, namely

$$H(U_2, S_1|S_0, U_1) = H(U_2|S_0, U_1) + H(S_1|S_0, U_1, U_2).$$

But S_1 is a deterministic function of S_0 and U_1 so $H(S_1|S_0, U_1, U_2) = 0$.

$$H(U_2, S_1|S_0, U_1) = H(U_2|S_0, U_1).$$

With the chain rule we also find

$$\begin{aligned} H(U_2, S_1|S_0, U_1) &= H(S_1|S_0, U_1) + H(U_2|S_0, U_1, S_1) \\ &= H(U_2|S_1). \end{aligned}$$

Again $H(S_1|S_0, U_1) = 0$ because the source is unifilar and $H(U_2|S_0, U_1, S_1) = H(U_2|S_1)$ because the source is a Markov source.

Now, finally, we can write

$$H(U^L|S_0) \leq H(U^L) \leq H(U^L|S_0) + \log_2 J,$$

and so

$$\begin{aligned}\lim_{L\to\infty} H_L(U) &= \lim_{L\to\infty} \frac{1}{L} H(U^L|S_0), \\ &= \lim_{L\to\infty} \frac{1}{L} L H(U|S) = H(U|S).\end{aligned}$$

$\square$

Part VI

Channel capacity

15 Introduction

15.1 Problem statement

We consider the transmission of binary symbols over a binary input-binary output channel that can transfer one symbol per unit of time. Usually the symbols are received correctly but the channel makes occasional errors where we receive the opposite symbol.

Is reliable communication possible over such a channel and if so, how much information can be transmitted reliable per channel use?

Answer
Yes, reliable communication is possible. Every channel has a certain *capacity.* If the amount of information is less than the capacity (per transmission) then an arbitrarily small probability of error can be achieved. When the amount of information transmitted is above capacity then many errors are unavoidable.

This last statement will be dealt with in another lecture.

15.2 The binary symmetric channel

The *binary symmetric channel* (BSC) is a simple model for a communication channel. We assume that the channel can transfer one *binary* symbol per use. The probability that this symbol is received correctly is the same for both values of the symbol. This is indicated by the term *symmetric.*

The formal definition of a BSC considers the *input* as the random variable X and the *output* as variable Y. X and Y take values in their respective sets or *alphabets* $\mathcal{X}$ resp. $\mathcal{Y}$. Both $\mathcal{X}$ and $\mathcal{Y}$ are binary alphabets, i.e. $\mathcal{X} = \mathcal{Y} = \{0, 1\}$. The probability that the channel makes an error in the transmission is given by p. We usually assume that $p \leq \frac{1}{2}$.

Alternatively we describe the channel by its *channel transition probabilities* $P_{Y|X}(y|x)$. We have

$$\begin{aligned} P_{Y|X}(0|0) &= P_{Y|X}(1|1) = 1 - p, \\ P_{Y|X}(1|0) &= P_{Y|X}(0|1) = p. \end{aligned}$$

We usually depict this channel as in Figure 9. Here the input variable X is written to the left-hand side of the graph and the vertices on the left-hand side are labeled with the values that the input variable can take.

Likewise on the right-hand side, the output variable Y is written and the vertices are labeled with the output values. The edges connect an input vertice x and an output vertice y and it is labeled with the conditional output probability $P_{Y|X}(y|x)$.

15.3 Multiple transmissions

The channel is *memoryless.* Successive transmissions are independent. Let x^N be the inputs to the channel and y^N the corresponding channel outputs. Because the channel is memoryless

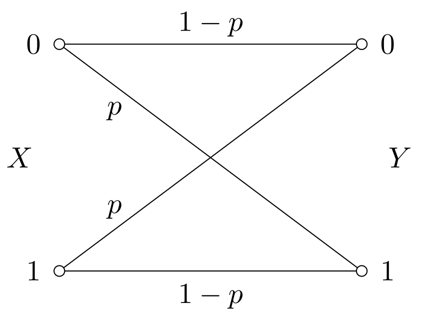


Figure 9: The binary symmetric channel

we find

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

Hamming distance

Let x^N and y^N be two binary words. The *Hamming distance* between these words is defined as the number of positions where x^N and y^N differ.

Definition 25 (Hamming distance)**.** The Hamming distance between two words x^N and y^N is denoted by $\mathrm{d_H}(x^N, y^N)$. It is defined as

$$\mathrm{d_H}(x^N, y^N) \triangleq |\{i : 1 \leq i \leq N, x_i \neq y_i\}|\,.$$

Channel output probabilities

Because the channel is memoryless we can write for the conditional output probability $P(y^N|x^N)$

$$\begin{aligned} P(y^N|x^N) &= p^{\mathrm{d_H}(x^N,y^N)}(1-p)^{N-\mathrm{d_H}(x^N,y^N)} \\ &= \left(\frac{p}{1-p}\right)^{\mathrm{d_H}(x^N,y^N)}(1-p)^N. \end{aligned}$$

Because $p \leq \frac{1}{2}$ we know that $p/(1-p) \leq 1$ and thus the probability $P(y^N|x^N)$ decreases (strictly if $p < \frac{1}{2}$) as $\mathrm{d_H}(x^N, y^N)$ increases.

16 Coding for the BSC

16.1 Redundancy

Suppose that we can transmit any of the 2^N possible input words x^N. Then, if the channel error probability p is larger than 0, the most likely input word given a received word y^N is the one that maximizes $P(y^N|x^N)$. Obviously, this is the word $x^N = y^N$. Thus we are unable to detect the errors that the channel might have made.

If, however, we do not allow all possible input words, then we might be lucky and the most likely input word could still be the correct word, even though the channel made some errors.

Example 26. Assume that there are only two valid input words, $x^N(1) = 0^N$ and $x^N(2) = 1^N$. So, obviously, $\mathrm{d_H}(x^N(1), x^N(2)) = N$.

Assume that we transmit $x^N(1)$ and the channel makes e errors, with $0 \leq e < \lfloor N/2 \rfloor^{\dagger}$. Let y^N be the received word. Then we have

$$\mathrm{d_H}(x^N(1), y^N) = e < \frac{N}{2},$$
$$\mathrm{d_H}(x^N(2), y^N) = N - e > \frac{N}{2},$$

and so we find

$$P(y^N | x^N(1)) > P(y^N | x^N(2)).$$

So, we can recognize the correct code word by finding the word with the largest probability $P(y^N|x^N)$.

16.2 Channel codes

We know that we cannot use all possible input words and that we should try to select words that are reasonably different from each other. Such a set of input words is called a *code* or *channel code.* We usually write C for the set of code words. So if the code contains M words we can write this as

$$C = \left\{x^N(1), x^N(2), \ldots, x^N(M)\right\},$$

where $x^N(m) \in \mathcal{X}^N = \{0,1\}^N$, and m ranges from 1 to M (inclusive).

Definition 27 (Channel code)**.** An (M, N) channel code is a set of binary channel input words of length N that are assigned to M source messages. We can see the source messages as just a set of M different objects, or as the complete set of $2^K = M$ binary sequences of length K, that could be the output of a source code.

16.3 Code rate

For the definition of the *channel code rate* it is most convenient to consider the messages to be binary sequences of length K. The code rate expresses the number of message symbols that are transmitted per channel use, so

$$R_c \triangleq \frac{K}{N}.$$

Because $M = 2^K$ we can also write

$$R_c = \frac{\log_2 M}{N}.$$

This might be more useful when we deal with an arbitrary number of messages that need not be a power of 2.

We assume that a message V^K is selected randomly, with equal probability for every possible value, or

$$P(V^K = v^K) = 2^{-K}.$$

† $\lfloor x \rfloor$ is the largest integer not larger than x.

16.4 The optimal decoder

The optimal decoder follows the *Maximum likelihood* rule.

Suppose the decoder receives the channel output word y^N. We select that code word $x^N \in C$ that maximizes $P(y^N|x^N)$. The corresponding message will be the decoded message.

Note that because the probability $P(y^N|x^N)$ only depends on the Hamming distance $\mathrm{d_H}(x^N, y^N)$ this rule is equivalent to selecting the code word that minimizes this Hamming distance.

16.5 The threshold decoder

A simplified rule that is easier to analyze and sufficient to show the required behaviour is the *threshold decoder*.

Select a threshold δ. Later we determine how to select its value. Define a Hamming sphere, *centered* around a code word $x^N(m)$ with *radius* δN as the set of all binary words of length N that have a Hamming distance to the sphere center $x^N(m)$ of at most the radius.

If we denote such a sphere by $\mathcal{B}_\delta(m)$, where $m \in \{1, 2, \ldots, M\}$, then we have

$$\mathcal{B}_\delta(m) \triangleq \left\{y^N : \mathrm{d_H}(x^N(m), y^N) \leq \delta N\right\}.$$

Threshold decoder rule
Let y^N be the received channel output word, let C be the set of M code words and let δ be given. Decode y^N into the m^{th} code word if the following conditions are all satisfied. If one or more conditions are not satisfied, the decoder declares an error.

1. $y^N \in \mathcal{B}_\delta(m)$.
2. For all other code words $m' \neq m$ we have: $y^N \notin \mathcal{B}_\delta(m')$.

16.6 Error probability and reliable communication

We shall use the threshold decoder. Suppose that a message m must be transmitted. The encoder selects the code word $x^N(m)$ from the code C and transmits it over the BSC. The decoder receives y^N.

Consider the following *events.*

$\mathcal{E}^1$: $y^N \notin \mathcal{B}_\delta(m)$.

$\mathcal{E}^2_{m'}$: $y^N \in \mathcal{B}_\delta(m')$. The events are defined for all $m' \neq m$.

It is clear that when all the events, $\mathcal{E}^1$, $\mathcal{E}^2_1$, $\mathcal{E}^2_2$, …, $\mathcal{E}^2_M$ do not happen, then the threshold decoder decides correctly for message m.

Consider a code C and suppose for simplicity that message $m = 1$ is selected and so the code word $x^N(1)$ is transmitted. The probability that the decoder decodes an incorrect code word or declares an error, given that code word $x^N(1)$ is transmitted is given by

$$P_{e,\text{channel}}(x^N(1)) = \Pr\{\mathcal{E}^1 \bigcup \mathcal{E}^2_2 \bigcup \mathcal{E}^2_3 \bigcup \cdots \bigcup \mathcal{E}^2_M | x^N(1)\}.$$

We can use the *union bound* to simplify this expression.

Theorem 28 (Union bound)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two events. Then*

$$\Pr\{\mathcal{A}\bigcup\mathcal{B}\} \le \Pr\{\mathcal{A}\} + \Pr\{\mathcal{B}\}.$$

Example 29. Consider a fair dice roll and denote the random outcome by R. Let event $\mathcal{A}$ indicate the fact that the outcome R is even. Let event $\mathcal{B}$ indicate the fact that $R \le 3$.

It is clear that $\Pr\{\mathcal{A}\} = \Pr\{\mathcal{B}\} = \frac{1}{2}$. However event $\mathcal{A}\bigcup\mathcal{B}$ is equal to the event that $R \in \{1,2,3,4,6\}$ so $\Pr\{\mathcal{A}\bigcup\mathcal{B}\} = \frac{5}{6}$. Clearly for this example the union bound holds.

Applying the union bound on $P_{e,\text{channel}}(x^N(1))$ we find

$$P_{e,\text{channel}}(x^N(1)) \le \Pr\{\mathcal{E}^1|x^N(1)\} + \sum_{m=2}^{M} \Pr\{\mathcal{E}_m^2|x^N(1)\}.$$

The expected error probability for a code C is given by

$$\bar{P}_{e,\text{channel}}(C) = \frac{1}{M}\sum_{m=1}^{M} P_{e,\text{channel}}(x^N(m)).$$

Unfortunately, even this bound is very complex to compute for a given code C. Moreover, we do not know the best possible code and analyzing all possible codes is much too complex to do.

17 Random coding

17.1 A random code

In stead of analyzing the expected error probability of a code C, we will bound the error probability *averaged over all possible codes.*

Let $P(C)$ be the uniform probability distribution over all possible codes and let $\mathcal{C}$ be the set of all possible (M,N) codes. Every code word x^N can take 2^N possible values and there are M code words so

$$|\mathcal{C}| = 2^{MN},$$
$$P(C) = \frac{1}{2^{MN}}.$$

All these codes have a code rate

$$R_c = \frac{\log_2 M}{N}.$$

17.2 Two types of errors

The averaged error probability is given as

$$\begin{aligned}
\bar{P}_{e,\text{channel}} &\triangleq \sum_{C\in\mathcal{C}} 2^{-MN}\bar{P}_{e,\text{channel}}(C),\\
&= \sum_{C\in\mathcal{C}} 2^{-MN}\frac{1}{M}\sum_{m=1}^{M} P_{e,\text{channel}}(x^N(m)),\\
&\le \frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\left(\Pr\{\mathcal{E}^1|x^N(m)\} + \sum_{m'\neq m}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\right).
\end{aligned}$$

17.3 Errors of the first kind

First we consider for a given code C and message m

$$\Pr\{\mathcal{E}^1|x^N(m)\} = \Pr\{Y^N \notin \mathcal{B}_\delta(m)\}.$$

Y^N is the channel output when transmitting $x^N(m)$ so

$$\Pr\{Y^N = y^N|X^N = x^N(m)\} = p^{\mathrm{d_H}(x^N(m),y^N)}(1-p)^{N-\mathrm{d_H}(x^N(m),y^N)},$$

and

$$\begin{aligned}
\Pr\{Y^N \notin \mathcal{B}_\delta(m')\} &= \sum_{y^N\notin\mathcal{B}_\delta(m)} \Pr\{Y^N = y^N|X^N = x^N(m)\}\\
&= \sum_{d=\delta N+1}^{N}\binom{N}{d}p^d(1-p)^{N-d}
\end{aligned}$$

Note that this equality holds for any code C and any message m and is independent of the code words and the message.

If $\delta > p$ then Theorem 2 of the first lecture applies and we find

$$\frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\Pr\{\mathcal{E}^1|x^N(m)\} \le 2^{-Nd(\delta\|p)}.$$

17.4 Errors of the second kind

Given a random selection of codes we determine the probability of a single code word and of a pair of code words.

$$\Pr\{X^N(m) = x^N\} = \sum_{\substack{C\in\mathcal{C}\\ x^N(m)=x^N}} 2^{-MN} = \frac{1}{2^N},$$

because there are $2^{(M-1)N}$ codes with a given fixed code word $x^N(m)$.

$$\Pr\{X^N(m) = x^N, X^N(m') = x^{N\prime}\} = \sum_{\substack{C\in\mathcal{C}\\ x^N(m)=x^N\\ x^N(m')=x^{N\prime}}} 2^{-MN} = \frac{1}{2^{2N}}.$$

All code words are selected at random and independent.

Now we consider the second part of the average error bound.

$$\begin{aligned}\frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\sum_{m'\neq m}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\\ = \frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{m'\neq m}\sum_{C\in\mathcal{C}}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\end{aligned}$$

We realize that we summing over all codes is equal to fixing two codeword positions, m and m', summing over all possible values at those positions, and summing over the rest of the code or,

$$\sum_{C\in\mathcal{C}} = \sum_{x^N}\sum_{x^{N\prime}}\sum_{\substack{C\in\mathcal{C}\\ x^N(m)=x^N\\ x^N(m')=x^{N\prime}}}$$

We use this to rewrite the previous equation

$$\begin{aligned}\frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\sum_{m'\neq m}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\\ = \frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{m'\neq m}\sum_{x^N}\sum_{x^{N\prime}}\sum_{\substack{C\in\mathcal{C}\\ x^N(m)=x^N\\ x^N(m')=x^{N\prime}}}\Pr\{Y^N\in\mathcal{B}_\delta(x^{N\prime})|x^N\}\end{aligned}$$

In the above we abused the notation of a Hamming sphere by explicitely writing the sphere center $x^{N\prime}$.

$$
\begin{aligned}
&\frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\sum_{m'\neq m}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\\
&\qquad= \frac{1}{M}\sum_{m=1}^{M}\sum_{m'\neq m}\sum_{x^N}\sum_{x^{N\prime}}\Pr\{Y^N\in\mathcal{B}_\delta(x^{N\prime})|x^N\}\overbrace{\sum_{\substack{C\in\mathcal{C}\\ x^N(m)=x^N\\ x^N(m')=x^{N\prime}}} 2^{-MN}}^{=2^{-2N}}\\
&\qquad= \sum_{x^N}\sum_{x^{N\prime}}\Pr\{Y^N\in\mathcal{B}_\delta(x^{N\prime})|x^N\}\frac{1}{M}\sum_{m=1}^{M}\sum_{m'\neq m}2^{-2N}\\
&\qquad\le \sum_{x^N}\sum_{x^{N\prime}}\Pr\{Y^N\in\mathcal{B}_\delta(x^{N\prime})|x^N\}\frac{M}{2^{2N}}\\
&\qquad= \frac{M}{2^{2N}}\sum_{x^N}\sum_{x^{N\prime}}\sum_{\substack{y^N\\ \mathrm{d_H}(x^{N\prime},y^N)\le\delta N}}P(y^N|x^N)
\end{aligned}
$$

$$
\begin{aligned}
&\frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\sum_{m'\neq m}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\\
&\qquad\le \frac{M}{2^{2N}}\sum_{x^{N\prime}}\sum_{\substack{y^N\\ \mathrm{d_H}(x^{N\prime},y^N)\le\delta N}}\sum_{x^N}P(y^N|x^N)\\
&\qquad= \frac{M}{2^{2N}}\sum_{x^{N\prime}}\sum_{\substack{y^N\\ \mathrm{d_H}(x^{N\prime},y^N)\le\delta N}}\underbrace{\sum_{d=0}^{N}\binom{N}{d}p^d(1-p)^{N-d}}_{=1}\\
&\qquad= \frac{M}{2^{2N}}\sum_{x^{N\prime}}\sum_{e=0}^{\delta N}\binom{N}{e} = \frac{M}{2^N}\sum_{e=0}^{\delta N}\binom{N}{e}\le M2^{-N(1-h(\delta)}
\end{aligned}
$$

And with $M = 2^{NR_c}$ we then obtain

$$
\frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\sum_{m'\neq m}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\le 2^{-N(1-h(\delta)-R_c)}.
$$

17.5 A good code

Now we combine the previous results.

$$
\begin{aligned}
\bar{P}_{e,\text{channel}} &\le \frac{2^{-MN}}{M}\sum_{m=1}^{M}\sum_{C\in\mathcal{C}}\left(\Pr\{\mathcal{E}^1|x^N(m)\}+\sum_{m'\neq m}\Pr\{\mathcal{E}^2_{m'}|x^N(m)\}\right),\\
&\le 2^{-Nd(\delta\|p)}+2^{-N(1-h(\delta)-R_c)}.
\end{aligned}
$$

As long as $\delta > p$ we know that $d(\delta \| p) > 0$, so the first term converges to zero exponentially fast in N. Also, when $R_c < 1 - h(\delta)$ also the second term converges to zero. Thus we can take the limit $\delta \downarrow p$ and obtain

$$\lim_{\delta \downarrow p} \bar{P}_{e,\text{channel}} = 0,$$
$$\lim_{\delta \downarrow p} R_c = 1 - h(p).$$

Theorem 30. *Random coding (average error) Let $\epsilon > 0$. For the BSC with error probability p, there exists for all rates $R_c < 1 - h(p)$ and all N large enough codes C with expected error probability $\bar{P}_{e,channel}(C) < \epsilon$.*

Proof. All we need is to realize that if a random variable Z has an expected value $\bar{z}$, then there must be a specific value $z \le \bar{z}$ with positive probability $p(z) > 0$. Thus if the expected error probability over the ensemble of all codes converges to zero, there must exist at least one sequence of codes (for increasing code wordlengths) whose error probabilities also converge to zero. □

17.6 Maximum error bound

We can also consider the maximum error probability of a code, defined by

$$P^{\max}_{e,\text{channel}}(C) \overset{\Delta}{=} \max_{m \in \{1,2,\ldots,M\}} P_{e,\text{channel}}(x^N(m)).$$

Theorem 31. *Random coding (maximum error) Let $\epsilon > 0$. For the BSC with error probability p, there exists for all rates $R_c < 1 - h(p)$ and all N large enough codes C with maximum error probability $P^{\max}_{e,channel}(C) < \epsilon$.*

Proof. We know from the previous theorem that there exists codes C for which

$$\bar{P}_{e,\text{channel}}(C) < \frac{\epsilon}{2}.$$

Pick the best half of the code words from this code C to make the code C^+. It must be true that

$$P^{\max}_{e,\text{channel}}(C^+) \overset{\Delta}{=} \max_{m \in \{1,2,\ldots,M/2\}} P_{e,\text{channel}}(x^N(m)) < \epsilon.$$

All code words in C that are not in C^+ have an error probability of at least $P^{\max}_{e,\text{channel}}(C^+)$. So, if $P^{\max}_{e,\text{channel}}(C^+) \ge \epsilon$, then $\bar{P}_{e,\text{channel}}(C)$ must be at least $\frac{\epsilon}{2}$ as can easily be verified.

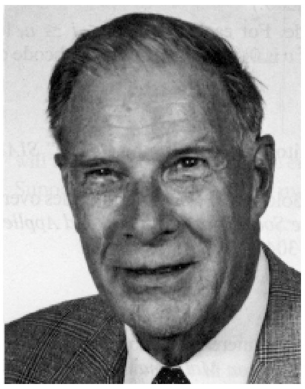
Now the rate of the code C^+ is $\frac{1}{N}$ code symbol per source symbol smaller than the rate of code C. As N becomes larger this difference approaches zero. □

Part VII

Linear block codes

18 Introduction

18.1 Richard Hamming



Richard Wesley Hamming was a mathematician whose work had many implications for computer science and telecommunications. His contributions include the Hamming code, the Hamming window, Hamming numbers, Sphere-packing (or Hamming bound) and the Hamming distance.
He was a professor at the University of Louisville during World War II, and left to work on the Manhattan Project in 1945. The objective of the program was to discover if the detonation of an atomic bomb would ignite the atmosphere. The result was that this would not occur, and so the United States used the bomb in a test in New Mexico and then twice against Japan.
Later, between 1946-1976 he worked at the Bell Telephone Laboratories, where he collaborated with Claude E. Shannon. He was a founder and president of the Association for Computing Machinery.

From Wikipedia.

18.2 Problem statement

Problem
An (M, N) block code selected randomly has little structure. An encoder and decoder for such a code must store all code words. For a given rate R this requires

$$M = 2^{RN} \text{ words of } N \text{ binary symbols.}$$

Thus, as a function of the wordlength N, we need a more than exponential number of storage bits. The coding theorem tells us that N has to be large and we quickly run out of (computer) memory.

Solution
We must add structure to the code words, so that they can be generated from a limited set of parameters. *Parity check codes,* more often called *linear codes,* have a simpler structure.

18.3 Binary arithmetic

The following discussion of linear codes requires the ability to do arithmetic on binary vectors. We shall discuss the following *binary* arithmetic operations.

Scalar addition

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	0

Scalar multiplication

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

Vector addition
Let x^N and y^N be two binary vectors, so for all $i \in \{1, 2, \ldots, N\}$ holds $x_i \in \{0, 1\}$ and $y_i \in \{0, 1\}$.
The vector addition $z^N = x^N + y^N$ is given by

$$z_i = x_i + y_i, \text{ for all } i \in \{1, 2, \ldots, N\}.$$

So, vector addition is done *componentwise.*

Scalar-vector product
Multiplying a vector x^N by a binary scalar $v \in \{0, 1\}$ gives the expected result. We write $v \cdot x^N$ for this product and find

$$0 \cdot x^N = 0^N.$$
$$1 \cdot x^N = x^N$$

Vector inproduct

We need some additional notation for the inproduct. The vectors we discuss are *row vectors*, e.g. $x^N = x_1, x_2, \ldots, x_N$. The *transpose* operator $^\tau$ changes a row vector into a column vector and vice-versa.

$$x^{N^\tau} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

The binary vector inproduct is defined in a similar way as the real valued vector inproduct. Let x^N and y^N be two binary row vectors. We write $x^N \cdot y^{N^\tau}$ for the vector inproduct and define

$$x^N \cdot y^{N^\tau} \triangleq \sum_{i=1}^{N} x_i y_i.$$

Here the summation is a binary addition.

19 Linear codes

19.1 Parity check code

The code words of the basic parity-check code are all words of length N with an even number of ones. So, x^N is a code word if and only if

$$1^N \cdot x^{N^\tau} = 0.$$

This is equivalent to the condition

$$\sum_{i=1}^{N} x_i = 0.$$

There are 2^{N-1} words of even parity, so we can transmit 2^{N-1} different messages. This can be represented by a binary message vector of length $N-1$. We write v^{N-1} for this message vector.

To every message v^{N-1} we assign a code word x^N by setting the first $N-1$ symbols of x^N equal to v^{N-1} and adding x_N such that the code word has even parity.

$$\begin{aligned} x_1 &= v_1 \\ x_2 &= \quad v_2 \\ \vdots &= \\ x_{N-1} &= \qquad v_{N-1} \\ x_N &= v_1 + v_2 + \ldots + v_{N-1} \end{aligned}$$

Matrix notation

In *matrix notation* this becomes

$$x^N = v^{N-1} \cdot G.$$

G is a $(N-1)\times N$ binary matrix given as

$$G=\begin{bmatrix}1&0&\cdots&0&1\\0&1&\cdots&0&1\\\vdots&\vdots&\cdots&\vdots&\vdots\\0&0&\cdots&0&1\\0&0&\cdots&1&1\end{bmatrix}=\begin{bmatrix}I_{N-1}; & \begin{matrix}1\\1\\\vdots\\1\end{matrix}\end{bmatrix}.$$

Here I_{N-1} is the $(N-1)\times(N-1)$ *identity matrix.*

19.2 Linear code

We can easily generalize the basic parity-check code to a linear code.

Let G be a $K\times N$ matrix of the form

$$G=\left[I_K;P\right],$$

where P is a $K\times(N-K)$ binary matrix and $K\le N$.
A matrix G of this particular type produces a so called *systematic code* and G itself is called the (systematic) *generator* matrix.

We use a K-dimensional message vector v^K and compute the corresponding code word as

$$x^N=v^K\cdot G.$$

The first K symbols of x^N are equal to the message symbols and the last $N-K$ symbols are called *parity symbols.*

Later we shall see that this results in $N-K$ parity equations.

The parity-check matrix

Consider again the code word equation $x^N=v^K\cdot G=\left[I_K;P\right]$.

$$\begin{array}{rcccccccc}x_1&=&v_1&&&&&&\\x_2&=&&&v_2&&&&\\\vdots&&&\vdots&&&&&\\x_K&=&&&&&\cdots&&v_K\\x_{K+1}&=&P_{1,1}v_1&+&P_{2,1}v_2&+&\cdots&+&P_{K,1}v_K\\\vdots&&&+&\vdots&&&&\\x_N&=&P_{1,N-K}v_1&+&P_{2,N-K}v_2&+&\cdots&+&P_{K,N-K}v_K\end{array}$$

We plug in the first K equations in the last $N-K$ ones.

$$\begin{array}{rcccccccc}x_{K+1}&=&P_{1,1}x_1&+&P_{2,1}x_2&+&\cdots&+&P_{K,1}x_K\\\vdots&&&+&\vdots&&&&\\x_N&=&P_{1,N-K}x_1&+&P_{2,N-K}x_2&+&\cdots&+&P_{K,N-K}x_K.\end{array}$$

Rewriting and realizing that in binary arithmetic addition and subtraction are the same we have

$$\begin{array}{cccccccccc}P_{1,1}x_1&+&P_{2,1}x_2&+&\cdots&+&P_{K,1}x_K&+&x_{K+1}&=&0\\\vdots&&&+&\vdots&&&&&&\\P_{1,N-K}x_1&+&P_{2,N-K}x_2&+&\cdots&+&P_{K,N-K}x_K&+&x_N&=&0\end{array}$$

We define the $(N-K)\times N$ matrix H as

$$\begin{bmatrix} P_{1,1} & P_{2,1} & \cdots & P_{K,1} & 1 & 0 & \cdots & 0 \\ P_{1,2} & P_{2,2} & \cdots & P_{K,2} & 0 & 1 & \cdots & 0 \\ \vdots & & \vdots & & & & \vdots & \\ P_{1,N-K} & P_{2,N-K} & \cdots & P_{K,N-K} & 0 & 0 & \cdots & 1, \end{bmatrix}$$

or $H = \left[P^{\tau}; I_{N-K}\right]$ where P^{τ} is the *transpose* of the matrix P.

Now we can conclude that every code word x^N satisfies the following condition

$$H \cdot x^{N\tau} = \left(0^{N-K}\right)^{\tau}.$$

19.3 A linear sub-space

Note that by construction, G has full row rank and thus defines a *linear sub-space* of the binary space $\{0,1\}^N$.
By full row rank we mean that the rows of G form a set of linearly independent vectors.

Let the $K\times N$ matrix G be a systematic generator matrix for the code C. So C is given by

$$C \triangleq \left\{x^N \in \{0,1\}^N : v^K \in \{0,1\}^K, x^N = v^K \cdot G\right\}.$$

The code satisfies the conditions for linear mappings.

Scaling: For all v^K and arbitrary $u \in \{0,1\}$: $u\cdot(v^K\cdot G) = (u\cdot v^K)\cdot G$.
For the proof we only need the fact that 0^N is a code word and is the result of the message 0^K, or

$$0^N = 0^K \cdot G.$$

Addition: The sum of any two code words is also a code word.
Let v_1^K and v_2^K be two messages and $x_1^N = v_1^K \cdot G$ and $x_2^N = v_2^K\cdot G$ the resulting code words. Let $v_3^K = v_1^K + v_2^K$. It is easy to see that

$$x_1^N + x_2^N = v_1^K\cdot G + v_2^K \cdot G = (v_1^K + v_2^K)\cdot G = v_3^K \cdot G,$$

is also a code word.

Equivalent codes

Any $K\times N$ matrix G that has full row rank generates a linear code.
We can transform G in two different ways that still result in (equivalent) codes.

Row addition: A row from G can be added to another row of G. The set of code words is not changed, only the assignment of a code word x^N to a message v^K changes.

Column reordering: This corresponds to a coordinate transformation that leaves the distance properties unchanged and therefor the essential properties of the code are unchanged.

19.4 Examples

Systematic code

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

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

v^3	x^5	$\left(H \cdot x^{5\intercal}\right)^{\intercal}$
000	00000	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
001	00111	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
010	01001	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
011	01110	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
100	10001	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
101	10110	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
110	11000	$\begin{bmatrix} 0 & 0 \end{bmatrix}$
111	11111	$\begin{bmatrix} 0 & 0 \end{bmatrix}$

Transformation: Adding rows $[1+2 \to 1,\ 2+3 \to 2]$

$$G_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

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

$$G_2 \cdot H^{\intercal} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

v^3	$v^3 \cdot G_2$	$v^3 \cdot G$
000	00000	00000
001	00111	00111
010	01110	01001
011	01001	01110
100	11000	10001
101	11111	10110
110	10110	11000
111	10001	11111

Transformation: Rearranging columns $[(12345) \to (25314)]$

$$G_3 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$H_3 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$G_3 \cdot H_3{}^{\intercal} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

v^3	$v^3 \cdot G_3$	$v^3 \cdot G$
000	00000	00000
001	10110	00111
010	00011	01001
011	10101	01110
100	01010	10001
101	11100	10110
110	01001	11000
111	11111	11111

20 Performance

20.1 Capacity

How well does a linear code perform?

We realize that a linear code is a special type of block code. So the linear code generated by a $K \times N$ generator matrix G has rate R_c.

$$R_c = \frac{\log_2 M}{N} = \frac{\log_2 2^K}{N} = \frac{K}{N}.$$

The first question we must answer is: what is the fundamental limit of reliable communication?

20.2 Random linear code

Consider a random linear generator $\mathbf{G}$ of size $K \times N$. The KN entries in the matrix are filled with random binary symbols so that the probability of a particular matrix G is constant or

$$\Pr\{\mathbf{G} = G\} = 2^{-KN}.$$

We shall need to consider the rows of the matrix G and write the j^{th} row as g_j^N, which is a row vector of length N. Note that we have

$$x^N = v^K \cdot G = \sum_{i=1}^{K} v_i \cdot g_i^N.$$

Consider two different messages v_1^K and v_2^K and their corresponding code words $x_1^N = v_1^K \cdot G$ and $x_2^N = v_2^K \cdot G$. Assume that v_1^K and v_2^K differ in the j^{th} position. We shall consider the sum $x_1^N + x_2^N$. First define ξ_j^N as

$$\xi_j^N \triangleq \sum_{\substack{i=1 \\ i \neq j}}^{K} (v_{1i} + v_{2i}) \cdot g_i^N,$$
$$x_1^N + x_2^N = \xi_j + g_j^N$$

For any fixed values of x_1^N, g_1^N, $g_2^N, \dots, g_{j-1}^N$, $g_{j+1}^N, \dots$, and g_K^N, x_2^N takes on a particular value for precisely one g_j^N. This implies that, with a random selection of G

$$\Pr\{x_2^N | x_1^N\} = 2^{-N}.$$

So, independent of $\Pr\{x_1^N\}$ we get

$$\Pr\{x_2^N\} = \sum_{x_1^N} \Pr\{x_1^N\} \Pr\{x_2^N | x_1^N\} = 2^{-N} = \Pr\{x_2^N | x_1^N\}.$$

Thus x_1^N and x_2^N are independent and this holds for any pair of code words corresponding to different messages.

The result of the random coding argument apply to random linear codes as well. We can still use the threshold decoder and the error of the first type is still upperbounded as

$$\Pr\{\mathcal{E}^1 | x^N(m)\} \leq 2^{-Nd(\delta \| p)}.$$

For the error of the second type we must realize that in the proof we only used the pair-wise independence of the code words, so

$$\sum_{m' \neq m} \Pr\{\mathcal{E}_{m'}^2 | x^N(m)\} \leq 2^{-N(1-h(\delta)-R_c)}.$$

21 Error correction

21.1 Hamming distance

Recall the Hamming distance as given before.

Definition
The Hamming distance $d_H(x^N, y^N)$ between two binary vectors x^N and y^N is equal to the number of positions where the vectors differ.

$$d_H(x^N, y^N) \triangleq |\{i \in \{1, 2, \ldots, N\} : x_i \neq y_i\}| \, .$$

The Hamming distance satisfies the following condition.

Theorem 32 (triangle inequality)**.** *If x_1^N, x_2^N, and x_3^N are three binary sequences of length N. Then we must have*

$$d_H(x_1^N, x_2^N) \leq d_H(x_1^N, x_3^N) + d_H(x_3^N, x_2^N).$$

Proof
Consider three sequences x_1^N, x_2^N, and x_3^N and write in a table how often any of the eight combinations $(x_{1,n}, x_{2,n}, x_{3,n})$ occur. In the following table we denote the number of triples $(x_{1,n}, x_{2,n}, x_{3,n}) = (0, 0, 0)$ by a_1, the number of triples $(x_{1,n}, x_{2,n}, x_{3,n}) = (0, 0, 1)$ by a_2, etc.

x_1^N	0	0	0	0	1	1	1	1
x_2^N	0	0	1	1	0	0	1	1
x_3^N	0	1	0	1	0	1	0	1
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8

Now it is easy to see that

$$\begin{aligned} d_H(x_1^N, x_2^N) &= a_3 + a_4 + a_5 + a_6, \\ d_H(x_1^N, x_3^N) &= a_2 + a_4 + a_5 + a_7, \\ d_H(x_2^N, x_3^N) &= a_2 + a_3 + a_6 + a_7. \end{aligned}$$

And thus we have

$$d_H(x_1^N, x_3^N) + d_H(x_2^N, x_3^N) = d_H(x_1^N, x_2^N) + 2a_2 + 2a_7,$$

and so $d_H(x_1^N, x_3^N) + d_H(x_2^N, x_3^N) \geq d_H(x_1^N, x_2^N)$.

21.2 t-error correcting code

A code is said to be a *t-error correcting code* if any pattern of zero up to (and including) t symbol errors can be corrected by the decoder, i.e. it is guaranteed that the decoder will find the correct codeword if not more than t symbol errors are made in the transmission of the codeword.

It will be clear that the more errors the channel makes, the larger the Hamming distance between the codeword x^N and the channel output word y^N becomes.

21.3 Minimum Hamming distance

An important parameter of a binary code C is its *minimum Hamming distance* defined as the minimal Hamming distance between two different codewords of the code.

The minimum Hamming distance $d_{H,\min}$ gives a guarantee that up to a certain number of symbol errors per codeword the codeword can be reconstructed correctly.

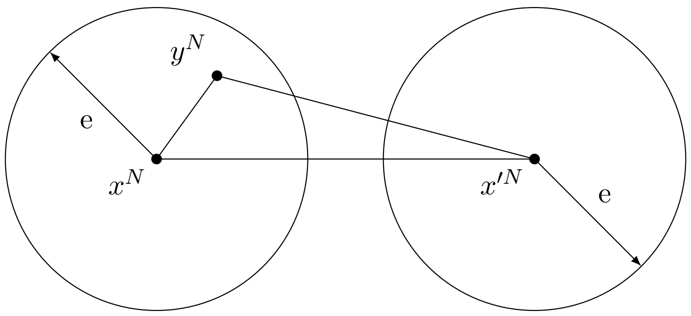
Assume that the channel has made e symbol errors and the code has a minimum Hamming distance $d_{H,\min} \geq 2e+1$. The transmitted codeword is x^N and the received channel word is y^N. Consider an arbitrary other codeword $x^{N\prime}$. From the fact that the channel made e errors we have $d_H(x^N, y^N) = e$. And from the minimum Hamming distance $d_{H,\min} \geq 2e+1$ we know that $d_H(x^N, x^{N\prime}) \geq 2e+1$. With the triangle inequality we now find, see the next figure

$$d_H(x^N, x^{N\prime}) \leq d_H(x^N, y^N) + d_H(y^N, x^{N\prime})$$

And using the results given above we find

$$d_H(y^N, x^{N\prime}) \geq d_H(x^N, x^{N\prime}) - d_H(x^N, y^N) \geq 2e+1-e = e+1$$

So the channel output word y^N is still closer to the correct codeword x^N than to any other codeword $x^{N\prime}$ and thus can be decoded correctly by a *minimum distance* decoder.



Error correction and Minimum distance.

If a binary code has a minimum Hamming distance $d_{H,\min}$ then it can correct at most

$$t = \lfloor \frac{d_{H,\min}-1}{2} \rfloor$$

symbol errors per codeword. We can find an upperbound to the error probability

$$P_e \leq 1 - \sum_{e=0}^{t} \binom{N}{e} p^e (1-p)^{N-e}.$$

21.4 Hamming weight

The Hamming weight is defined as the number of non-zero symbols in a vector. Thus it equals the Hamming distance between that vector and the all-zero word.

$$w_H(x^N) \triangleq d_H(x^N, 0^N).$$

Let x^N and y^N be two *different* code words from the *linear* code C. The binary sum $x^N + y^N$ is also a code word. It is easy to see that

$$d_H(x^N, y^N) = w_H(x^N + y^N).$$

Namely, adding a vector to both x^N and y^N does not change the Hamming distance, so

$$\begin{aligned} d_H(x^N, y^N) &= d_H(x^N + y^N, y^N + y^N) \\ &= d_H(x^N + y^N, 0^N) \\ &= w_H(x^N + y^N). \end{aligned}$$

Minimum Hamming weight

The minimum Hamming weight of a linear code is defined as the smallest non-zero weight. The fact that, for a linear code, the Hamming distance of two code words equals the Hamming weight of another code word results in the following statement.

> *For a linear code the minimum Hamming distance equals the minimum Hamming weight.*

The *advantage* of this is that in order to find the minimum Hamming distance of a linear code, we only have to consider $2^K - 1$ code words in stead of $2^K(2^K - 1)$ pairs of code words.

21.5 Syndrome decoding

The parity-check matrix is very useful for decoding, because $H \cdot x^{N\tau} = 0$ for all code words x^N.

Let e^N be the *error vector*, i.e. a binary vector containing ones in the positions where the BSC made an error. If x^N is the transmitted codeword then the decoder received the channel output word $y^N = x^N + e^N$.

The decoder computes the *syndrome* $s^{N-K} = H \cdot y^{N\tau}$. If this equals the all-zero vector then y^N is a codeword and the decoder assumes that it is the correct codeword. Otherwise s^{N-K} will take one out of $2^{N-K} - 1$ non-zero values.

We can write

$$s^{N-K} = H \cdot y^{N\tau} = H \cdot \left(x^N + e^N\right)^\tau = H \cdot x^{N\tau} + H \cdot e^{N\tau} = H \cdot e^{N\tau}$$

So, s^{N-K} only depends on the error vector. There are 2^N error vectors and only 2^{N-K} syndromes. So, several error vectors will result in the same syndrome. Actually precisely 2^K error vectors will result in one syndrome. The decoder must select one of these 2^K error vectors given the computed syndrome and it will select an error vector of minimal weight e^N among all possible candidates. The decoded codeword is then $\hat{x}^N = y^N + e^N$.

21.6 A one error correcting code (Hamming code)

Suppose that all columns of the parity-check matrix H are different and non-zero. Then the syndromes of all error vectors of weight 1, i.e. a single error, and the syndrome of the all-zero error vector, i.e. no error, are different. Thus the decoder can detect these situations.

We can define the *Hamming code* most easily by its parity-check matrix that contains all $2^m - 1$ non-zero vectors of a given length m. So the codeword length is $2^m - 1$ and the generator matrix contains $2^m - 1 - m$ rows. So the Hamming code has a code rate

$$R = \frac{2^m - m - 1}{2^m - 1}.$$

If the channel made one error in the i^{th} transmission only then the error vector $e^N = (0, 0, \cdots, 0, 1, 0, \cdots, 0)$ and

$$H \cdot y^{N^\tau} = H \cdot e^{N^\tau} = h_i^m,$$

where h_i^m is the i^{th} column of H. Because all columns of H are different, the decoder immediately knows in which position the error is located.

Because a Hamming code can correct one error its minimum Hamming distance must be at least three. It turns out that $d_{H,\min} = 3$ for Hamming codes.

22 Bounds on error correcting codes

22.1 Hamming bound

Let C be a block code with codeword length N and containing M codewords. The code does not have to be linear. C is a t-error correcting code, so its minimum distance $d_{H,\min} = 2t + 1$.

If x^N is a codeword then all channel output words y^N having a distance $d_H(x^N, y^N) \le t$ are closer to x^N than to any other codeword. We define the *decoding region*, or *decoding sphere* to be exactly this set of channel output words.

$$\mathcal{B}_t(x^N) \triangleq \left\{ y^N : d_H(x^N, y^N) \le t \right\}.$$

So the decoding regions for the different codewords are non-overlapping and together cannot contain more than 2^N words. So we get the following bound.

Hamming bound

An (M, N) code with minimum Hamming distance $d_{H,\min}$ that can correct at most $t \triangleq \left\lfloor \frac{d_{H,\min} - 1}{2} \right\rfloor$ errors on the BSC, satiesfies the following bound.

$$M \le \left\lfloor \frac{2^N}{\sum_{e=0}^{t} \binom{N}{e}} \right\rfloor,$$

where the sum gives the number of channel output words in a decoding set and is independent of x^N.

Proof. The *spheres* of radius t do not overlap. Any such sphere contains

$$|\mathcal{B}_t(x^N)| = \sum_{e=0}^{t} \binom{N}{e}$$

words. The complete space of all words contains 2^N words and therefor $M \cdot |\mathcal{B}_t(x^N)|$ cannot contain more than 2^N words. □

22.2 Gilbert bound

Gilbert bound

In this section we shall describe Gilberts code construction.

The Gilbert code construction does not really result in useful codes. It is used to derive a lowerbound on the performance, the coderate, of error correcting codes. The reason that the construction is not practical is that the resulting codes do not have any nice structure that allows us to make efficient encoders and decoders.

Assume that we want to design a code C of blocklength (codeword length) N and with a minimum distance $d_{H,\min}$. If we arbitrarily select a word x_1^N as a codeword then all words y^N from $\{y^N : d_H(x_1^N, y^N) < d_{H,\min}\}$ cannot also be a codeword. So we remove those words from consideration.

If then we still have some more words left over, we can select another codeword x_2^N and then have to remove some more words that are too close to x_2^N. Etc., until no more words remain.

How many words do we remove because we select a certain codeword x_i^N? That depends, but we can give an upperbound to this number, namely we shall never remove more words than all of the words in $\{y^N : d_H(x_i^N, y^N) \leq d_{H,\min} - 1\}$. There are

$$\sum_{i=0}^{d_{H,\min}-1} \binom{N}{i}$$

words in this set. The worst that can happen is that we again and again remove the maximum number of words. We can continue selecting another codeword until none remain so we can certainly find

$$M = \left\lceil \frac{2^N}{\sum_{i=0}^{d_{H,\min}-1} \binom{N}{i}} \right\rceil$$

codewords.

We state this result in the following theorem.

Gilbert bound.
A code with codeword length N and minimum distance $d_{H,\min}$ can contain at least M codewords, where M is bounded as

$$M \geq \left\lceil \frac{2^N}{\sum_{i=0}^{d_{H,\min}-1} \binom{N}{i}} \right\rceil$$

22.3 Asymptotic bounds

If we look at Hamming codes we see that they satisfy the Hamming bound. Namely for a given m the code has a codeword length of $2^m - 1$ and contains 2^{2^m-m-1} codewords and $d_{H,\min} = 3$. If we look at Hamming's bound we find

$$\sum_{e=0}^{1} \binom{2^m - 1}{e} = 1 + 2^m - 1 = 2^m.$$

Thus the bound tells us that the maximal number of codewords is

$$M \le \left\lfloor \frac{2^{2^m-1}}{2^m} \right\rfloor = 2^{2^m-m-1}.$$

And the Hamming code has this number of codewords! A code that satisfies the Hamming bound with equality is called a *perfect code*. Not many perfect binary codes exist.

Because the Hamming code is perfect all channel output words are either a codeword or have distance 1 to a codeword.

It is interesting to consider the Hamming bound when the codeword length becomes large. In the following we shall use the *Van Lint* bound on the binomial coefficient $\binom{N}{t}$.

Using the Hamming bound we have

$$\begin{aligned} M\binom{N}{t} &\le M\sum_{i=0}^{t}\binom{N}{i} \le 2^N \\ M &\le 2^N / \binom{N}{t} \\ \log_2 M &\le N - \log\binom{N}{t}. \end{aligned}$$

Now as $N \to \infty$ we use the bound on the binomial coefficient and get

$$R_c = \frac{\log_2 M}{N} \le 1 - h(t/N).$$

For the Gilbert bound we can use a similar argument and obtain

$$R_c = \frac{\log_2 M}{N} \ge 1 - h(\frac{2t+1}{N}).$$

22.4 Asymptotics

Consider a BSC with cross-over probability p, $0 < p \le 1/2$. The (Shannon) capacity of this channel is $C = 1 - h(p)$ and reliable codes with rates $R < 1 - h(p)$ exist.

Consider a t-error correcting code of blocklength N then if we choose $t = (p+\epsilon)N$ for some small $\epsilon > 0$ then we know that if N is large enough that it becomes arbitrarily unlikely that the channel will make more than t errors in a codeword. So our code must have a minimum distance $d_{H,\min} = 2(p + \epsilon)N + 1$.

So we find (with $t/N = p + \epsilon$)

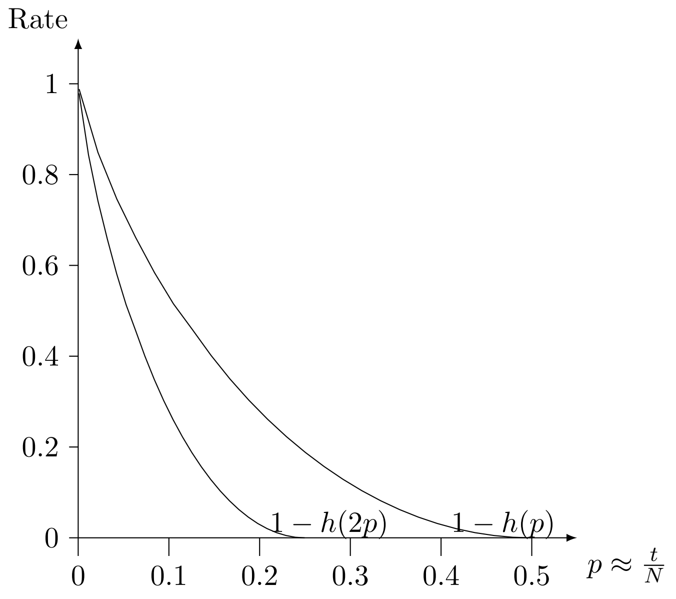
$$R < 1 - h(p + \epsilon) \approx 1 - h(p). \text{ (Hamming.)}$$

The asymptotic Hamming bound coincides with the channel capacity!

We have $(d_{H,\min} - 1)/N = 2(p + \epsilon)$ and this results in

$$R \geq 1 - h(2(p + \epsilon)) \approx 1 - h(2p), \ \text{(Gilbert)}$$

where we must restrict p to $0 < p \leq 1/4$.
See the graph in the following figure where these bounds are depicted.



The asymptotic Gilbert and Hamming bounds.

The asymptotic Gilbert bound gives us a much lower rate than Shannon promises. But even so, up to today no binary codes (code constructions) have been found that achieve the asymptotic Gilbert bound, except for the point $R = 1$, $t/N = 0$ which is achieved by the Hamming code.

Part VIII

Convexity and optimization

23 Introduction

23.1 Johan Jensen

Johan Ludwig William Valdemar Jensen. Born: 8 May 1859 in Nakskov, Denmark, Died: 5 March 1925 in Copenhagen, Denmark.

In 1876 Johan entered the College of Technology and there he studied a range of science subjects including physics, chemistry and mathematics. It was, however, mathematics which began to dominate his life. He published his first papers while a student at the College of Technology.

Jensen was essentially self taught in research level mathematics and he worked for a telephone company. His position was with the Copenhagen Division of the International Bell Telephone Company.

In 1882, the company became the Copenhagen Telephone Company. Jensen continued to work for the company until 1924 becoming head of the technical department in 1890. For his whole working life Jensen was an amateur mathematician only doing mathematics in his spare time. However, he reached a very high level of expertise as a mathematician as he did as a telephone engineer.

Jensen contributed to the Riemann Hypothesis, proving a theorem which he sent to Mittag-Leffler who published it in 1899. The theorem is important, but does not lead to a solution of the Riemann Hypothesis as Jensen had hoped. It expresses:

... the mean value of the logarithm of the absolute value of a holomorphic function on a circle by means of the distances of the zeros from the centre and the value at the centre.

He also studied infinite series, the gamma function and inequalities for convex functions. In a paper which he published in Acta mathematica in 1906, Jensen proved an inequality for convex functions which had a whole host of classical inequalities as special cases.

Article by: J J O'Connor and E F Robertson

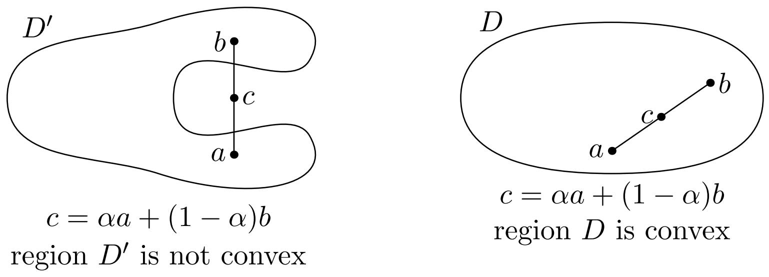

D'
b
c
a
$c = \alpha a + (1-\alpha)b$
region D' is not convex
D
b
c
a
$c = \alpha a + (1-\alpha)b$
region D is convex

Figure 10: The convexity of a region

23.2 Problem statement

Problem
Many problems we will deal with in the coming lectures are about optimization. We often must find the maximal or minimal value of information theoretic quantities (functions) such as $h(p)$ and $d(p\|q)$ (or their more extended versions).

Solution
Fortunately many of these functions are convex or concave and therefor they possess a well-defined maximum (minimum) on nice regions of their argument spaces. We will discuss these properties in this lecture.

23.3 Convex region

A *convex region* is a set of points in Euclidian space that satisfies the condition than the line connecting any two points in the region is completely inside the region, see Fig. 10.

23.4 The probability simplex

Consider the set of all vectors $p^N = (p_1, p_2, \ldots, p_N)$ such that the components p_i are non-negative and the vector components sum up to precisely 1.

Such a vector can be seen as a *probability vector* $p^N = (p(u_1), p(u_2), \ldots, p(u_N))$ or a *probability distribution*. The set of all probability vectors over an alphabet $\mathcal{U}$ is denoted by

$$D_{|\mathcal{U}|} \triangleq \left\{ p^N : \sum_{u \in \mathcal{U}}^{N} p(u) = 1, \text{for all } u \in \mathcal{U}, p(u) \geq 0 \right\}.$$

That this is a convex set is easy to check. Let p^N and q^N be two vectors in $D_{|\mathcal{U}|}$. Then, let $r^N = \alpha p^N + (1-\alpha) q^N$, with $0 \leq \alpha \leq 1$, be the points on the line between p^N and q^N.

Namely

$$r(u) = \alpha p(u) + (1-\alpha)q(u).$$

and because $p(u) \geq 0$ and $q(u) \geq 0$ and $0 \leq \alpha \leq 1$ we know that

$$r(u) \geq 0.$$

Furthermore

$$\begin{aligned}\sum_{u\in\mathcal{U}} r(u) &= \sum_{u\in\mathcal{U}} \alpha p(u) + (1-\alpha)q(u) \\ &= \alpha \sum_{u\in\mathcal{U}} p(u) + (1-\alpha)\sum_{u\in\mathcal{U}} q(u) \\ &= \alpha + (1-\alpha) = 1.\end{aligned}$$

Thus $r^N \in D_{|\mathcal{U}|}$.

23.5 Convex functions

In mathematics, a real-valued function f defined on a convex subset C of some Euclidian space is called convex, if for any two points x and y in its domain C and any $\alpha \in [0,1]$, we have

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y).$$

In other words, a function is convex if and only if its *epigraph* (the set of points lying on or above the graph) is a convex set.

A function is called strictly convex if

$$f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y),$$

for any $\alpha \in (0,1)$ and $x \neq y$.

A function f is said to be *concave* if $-f$ is convex.

23.6 Properties

- A continuous function on an interval C is convex if and only if

  $$f\left(\frac{x+y}{2}\right) \leq \frac{f(x)+f(y)}{2}.$$

  for all x and y in C.

- A differentiable function $f(x)$ is convex if and only if its derivative $f'(x)$ is monotonically increasing.

- If the function $f(x)$ is convex, $f(x)$ differentiable at point c, and $f'(c) = 0$, then c is a global minimum of $f(x)$. Note that this theorem does not follow from the previous one because it does not require the existence of $f'(x)$ except at the point c.

- A continuously differentiable function of one variable is convex on an interval if and only if the function lies above all of its tangents: $f(y) \geq f(x) + f'(x)(y-x)$ for all x and y in the interval.
- A twice differentiable function of one variable is convex on an interval if and only if its second derivative is non-negative there; this gives a practical test for convexity. If its second derivative is positive then it is strictly convex.
- Any local minimum of a convex function is also a global minimum. A strictly convex function will have at most one global minimum.

Examples of convex/concave functions

- The second derivative of x^2 is 2; it follows that x^2 is a convex function of x.
- The absolute value function $|x|$ is convex, even though it does not have a derivative at $x = 0$.
- The function x^3 has second derivative $6x$; thus it is convex for $x \geq 0$ and concave for $x \leq 0$.

23.7 Information theoretic examples

For the following results we make use of this lemma.

Lemma 33 (log inequality)**.** *For all real values x holds*

$$\ln x \leq x - 1.$$

The proof of this lemma is left as an excercise.

The binary entropy function is a concave function.

Proof. Let $x, y \in [0,1]$. We wish to show that $h(\frac{x+y}{2}) \geq \frac{h(x)+h(y)}{2}$. We shall write $r = \frac{x+y}{2}$ in the proof.

$$\begin{aligned}
&\frac{h(x)+h(y)}{2} - h(r) = \\
&= \frac{\log_2 e}{2}\,(-x\ln x - (1-x)\ln(1-x) - y\ln y - (1-y)\ln(1-y) + \\
&\quad +(x+y)\ln r + (1-x+1-y)\ln(1-r)) \\
&= \frac{\log_2 e}{2}\left(x\ln\frac{r}{x} + (1-x)\ln\frac{1-r}{1-x} + y\ln\frac{r}{y} + (1-y)\ln\frac{1-r}{1-y}\right) \\
&\leq \frac{\log_2 e}{2}\,(r - x + (1-r) - (1-x) + r - y + (1-r) - (1-y)) = 0.
\end{aligned}$$

□

The binary divergence $d(p\|q)$ is convex in p.

Proof. We know that

$$d(p\|q) = -p\log_2 q - (1-p)\log_2(1-q) - h(p).$$

So, we rewrite

$$\begin{aligned}\frac{d(p\|r)+d(q\|r)}{2} &= -\frac{p+q}{2}\log_2 r - (1-\frac{p+q}{2})\log_2(1-r) - \frac{h(p)+h(q)}{2}\\ &\geq -\frac{p+q}{2}\log_2 r - (1-\frac{p+q}{2})\log_2(1-r) - h(\frac{p+q}{2})\\ &= d(\frac{p+q}{2}\|r).\end{aligned}$$

□

Generalized entropy

We can generalize to a larger than binary alphabet. Let $\mathcal{U}$ be an arbitrary, finite alphabet, say

$$\mathcal{U} = \{1, 2, \ldots, M\},$$

for some positive integer $M \geq 2$.

We define, what later will be known as the "entropy", $H(P)$, where P is a probability vector over $\mathcal{U}$, so $P \in D_{|\mathcal{U}|}$.

$$H(P) \triangleq -\sum_{u\in\mathcal{U}} P(u)\log_2 P(u).$$

Now let $Q \in D_{|\mathcal{U}|}$ be another probability vector over $\mathcal{U}$. We write $R = \frac{P+Q}{2}$ and $R \in D_{|\mathcal{U}|}$ is another probability vector.

We shall show that $H(P)$ is a concave function, or

$$\frac{1}{2}H(P) + \frac{1}{2}H(Q) \leq H(\frac{P+Q}{2}) = H(R).$$

Consider

$$\begin{aligned}\frac{1}{2}H(P) + \frac{1}{2}H(Q) - H(R) &= \frac{\log_2 e}{2}\sum_{u\in\mathcal{U}} P(u)\ln\frac{R(u)}{P(u)} + \frac{\log_2 e}{2}\sum_{u\in\mathcal{U}} Q(u)\ln\frac{R(u)}{Q(u)}\\ &\leq \frac{\log_2 e}{2}\sum_{u\in\mathcal{U}}(R(u)-P(u)) + \frac{\log_2 e}{2}\sum_{u\in\mathcal{U}}(R(u)-Q(u)) = 0.\end{aligned}$$

24 Optimization

24.1 Jensen inequality

For a real convex function f, n numbers x_i in its domain, and n positive weights a_i that sum up to 1, Jensen's inequality can be stated as:

$$f\left(\sum_{i=1}^{n} a_i x_i\right) \le \sum_{i=1}^{n} a_i f(x_i).$$

For instance, the $\ln(x)$ function is concave, so substituting $f(x) = -\ln(x)$ in the previous formula, taking all weights equal to $\frac{1}{n}$ establishes the (logarithm of) the familiar arithmetic mean-geometric mean inequality:

$$\frac{x_1 + x_2 + \cdots + x_n}{n} \ge \sqrt[n]{x_1 x_2 \cdots x_n}.$$

In probability-theory notation (real space).

The same result can be stated in a probability theory setting. Let X be a real-valued random variable and f a measurable convex function. Then:

$$f\left(\mathbb{E}\{X\}\right) \le \mathbb{E}\{f(X)\}.$$

24.2 Proof (using the finite form)

If λ_1 and λ_2 are two arbitrary positive real numbers such that $\lambda_1 + \lambda_2 = 1$, then convexity of f implies

$$f(\lambda_1 x_1 + \lambda_2 x_2) \le \lambda_1 \, f(x_1) + \lambda_2 \, f(x_2) \text{ for any } x_1,\, x_2.$$

This can be easily generalized: if $\lambda_1,\, \lambda_2, \ldots, \lambda_n$ are n positive real numbers such that $\lambda_1 + \lambda_2 + \cdots + \lambda_n = 1$, then

$$f(\lambda_1 x_1 + \lambda_2 x_2 + \cdots \lambda_n x_n) \le \lambda_1 \, f(x_1) + \lambda_2 \, f(x_2) + \cdots + \lambda_n \, f(x_n),$$

for any $x_1,\, x_2, \ldots,\, x_n$.

This finite form of the Jensen's inequality can be proved by induction: by convexity hypotheses, the statement is true for $n = 2$. Suppose it is true also for some n, one needs to prove it for $n + 1$. At least one of the λ_i is strictly positive, say λ_1; therefore by convexity inequality:

$$\begin{aligned} f\left(\sum_{i=1}^{n+1} \lambda_i x_i\right) &= f\left(\lambda_1 x_1 + (1-\lambda_1) \sum_{i=2}^{n+1} \frac{\lambda_i}{1-\lambda_1} x_i\right) \\ &\le \lambda_1 \, f(x_1) + (1-\lambda_1) f\left(\sum_{i=2}^{n+1} \left(\frac{\lambda_i}{1-\lambda_1} x_i\right)\right). \end{aligned}$$

Since $\sum_{i=2}^{n+1} \frac{\lambda_i}{1-\lambda_1} = 1$, one can apply the induction hypotheses to the last term in the previous formula to obtain the result, namely the finite form of the Jensen's inequality.

24.3 Examples of Jensen inequality

Form involving a probability vector
If Ω is some finite set $\{x_1, x_2, \ldots, x_n\}$, f a convex function, and $g(x)$ any real valued function on Ω, then

$$f\left(\sum_{i=1}^{n} g(x_i)\lambda_i\right) \le \sum_{i=1}^{n} f(g(x_i))\lambda_i,$$

provided that $\lambda_1 + \lambda_2 + \cdots + \lambda_n = 1$, $\lambda_i \ge 0$.

Statistical physics
Jensen's inequality is of particular importance in statistical physics when the convex function is an exponential, giving:

$$e^{\langle X \rangle} \le \left\langle e^X \right\rangle,$$

where angle brackets denote expected values with respect to some probability distribution in the random variable X.

The proof in this case is very simple. The desired inequality follows directly, by writing

$$\left\langle e^X \right\rangle = e^{\langle X \rangle} \left\langle e^{X - \langle X \rangle} \right\rangle$$

and then applying the inequality

$$e^X \ge 1 + X$$

to the final exponential.

Information theory
If $p(x)$ is the true probability vector for x, and $q(x)$ is another probability vector, then applying Jensen's inequality for the random variable $Y(x) = q(x)/p(x)$ and the function $f(y) = -\log(y)$ gives

$$\begin{aligned}
&\mathbb{E}\{f(Y)\} \ge f(\mathbb{E}\{Y\}) \\
&\qquad \Rightarrow \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \ge -\log \sum_{x \in \mathcal{X}} p(x) \frac{q(x)}{p(x)} \\
&\qquad \Rightarrow \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \ge 0 \\
&\qquad \Rightarrow -\sum_{x \in \mathcal{X}} p(x) \log q(x) \ge -\sum_{x \in \mathcal{X}} p(x) \log p(x),
\end{aligned}$$

a result called Gibbs' inequality (discrete version).

Note that, if $\mathcal{X} = \{0, 1\}$ and $p(1) = p$, $q(1) = q$, then

$$\sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} = d(p \| q).$$

24.4 Kuhn-Tucker theorem

Let f be a concave function over the probability simplex $D_{|\mathcal{U}|}$. We assume that $\mathcal{U}$ contains k letters.

We wish to find the maximum of $f(p)$ over $p = (p_1, p_2, \ldots, p_k) \in D_{|\mathcal{U}|}$.

First we try to solve

$$\frac{\partial f(p)}{\partial p_i} = 0.$$

Unfortunately, this might not have a solution or the solution p^* might not lie in the region $D_{|\mathcal{U}|}$.

Fortunately, if p^* is ok, then it realises the maximum of $f(p)$, namely,

assume the contrary and p^+ exists such that $f(p^+) > f(p^*)$. Then the line connecting $(p^*, f(p^*))$ to $(p^+, f(p^+))$ increases and because of the concavity $f(p)$ cannot lie below this line so must also increase from p^* in the direction of p^+ and therefor cannot be a stationary point.

Suppose that the maximum occurs at the border of $D_{|\mathcal{U}|}$, where some $p_i^* = 0$. Then the derivative $\frac{\partial f(p^*)}{\partial p_i}$ can be negative. So a better condition could be

$$\begin{aligned} \frac{\partial f(p)}{\partial p_i} &= 0; \text{ if } p_i > 0, \\ \frac{\partial f(p)}{\partial p_i} &\leq 0; \text{ if } p_i = 0. \end{aligned}$$

Now we must use a Lagrangue method to incorporate the $\sum_i p_i = 1$ condition. So we will maximize $f(p) + \lambda(1 - \sum_i p_i)$. We find

$$\begin{aligned} \frac{\partial f(p)}{\partial p_i} &= \lambda; \text{ if } p_i > 0, \quad (*) \\ \frac{\partial f(p)}{\partial p_i} &\leq \lambda; \text{ if } p_i = 0, \quad (**) \end{aligned}$$

as the candidate conditions. (Kuhn-Tucker conditions).

We assume that the partial derivatives are continuous and then (*) and (**) are *necessary and sufficient* conditions for p that maximizes f over $D_{|\mathcal{U}|}$.

Proof. (Sufficiency). Say p^* and λ solve (*) and (**). We show that $f(p^*)$ is the global maximum.

Take any $p^+ \in D_{|\mathcal{U}|}$. We have, for any $0 < \theta < 1$ that

$$\begin{aligned} \theta f(p^+) + (1-\theta) f(p^*) &\leq f(\theta p^+ + (1-\theta)p^*) \\ f(p^+) - f(p^*) &\leq \frac{f(\theta p^+ + (1-\theta)p^*) - f(p^*)}{\theta} \end{aligned}$$

Now let $\theta \to 0$

$$\begin{aligned} f(p^+) - f(p^*) &\leq \sum_{i=1}^{k} \frac{\partial f(p^*)}{\partial p_i}(p_i^+ - p_i^*) \\ &\leq \sum_{i=1}^{k} \lambda (p_i^+ - p_i^*) \\ &= 0. \end{aligned}$$

The last inequality follows from (*) if $p_i^* > 0$ so $\frac{\partial f(p^*)}{\partial p_i}(p_i^+ - p_i^*) = \lambda(p_i^+ - p_i^*)$ and from (**) and the fact that $p_i^+ \geq 0$ so $\frac{\partial f(p^*)}{\partial p_i}(p_i^+ - p_i^*) \leq \lambda(p_i^+ - p_i^*)$. □

Proof. (Necessity). Let p^* maximize f over $D_{|\mathcal{U}|}$ and assume that the partial derivatives are continuous in p^*.

For any p^+ and any $0 < \theta < 1$, we have

$$f(\theta p^+ + (1-\theta)p^*) - f(p^*) \leq 0$$

Divide by θ and let $\theta \to 0$

$$\sum_{i=1}^{k} \frac{\partial f(p^*)}{\partial p_i}(p_i^+ - p_i^*) \leq 0.$$

Assume that $p_1^* > 0$ then let $p^+ = p^* + e$, where the vector e has all zero entries except $e_1 = -\epsilon$ and $e_j = \epsilon$ and $0 < \epsilon < p_1^*$. So $p^+ \in D_{|\mathcal{U}|}$.

From

$$\sum_{i=1}^{k} \frac{\partial f(p^*)}{\partial p_i}(p_i^+ - p_i^*) = \epsilon\left(\frac{\partial f(p^*)}{\partial p_j} - \frac{\partial f(p^*)}{\partial p_1}\right) \leq 0,$$

we conclude that

$$\frac{\partial f(p^*)}{\partial p_j} \leq \frac{\partial f(p^*)}{\partial p_1}.$$

If, however $p_j^* > 0$ then we may reverse the roles and set ϵ to a negative value, such that $p_j^* < \epsilon < 0$ and we find $\frac{\partial f(p^*)}{\partial p_1} \leq \frac{\partial f(p^*)}{\partial p_j}$. We set $\lambda = \frac{\partial f(p^*)}{\partial p_1}$ and conclude

$$\frac{\partial f(p^*)}{\partial p_j} = \lambda; \text{ if } p_j^* > 0,$$
$$\frac{\partial f(p^*)}{\partial p_j} \leq \lambda; \text{ if } p_j^* = 0.$$

□

24.5 An application of the Kuhn-Tucker theorem

Consider the function f on $D_{|\mathcal{U}|}$, where $\mathcal{U}$ contains $k = 3$ letters.

$$f(p_1, p_2, p_3) \triangleq \ln(p_1) + \ln(p_2).$$

$$\frac{\partial f(p)}{\partial p_1} = \frac{1}{p_1} \tag{2}$$
$$\frac{\partial f(p)}{\partial p_2} = \frac{1}{p_2} \tag{3}$$
$$\frac{\partial f(p)}{\partial p_3} = 0 \tag{4}$$

From $0 \leq p_1 \leq 1$ we know with (2) that $\lambda \geq 1$ in the Kuhn-Tucker conditions. Then with (4) we must conclude that $p_3 = 0$ so $p_1 + p_2 = 1$.

Now neither $\frac{\partial f(p^*)}{\partial p_1} > \frac{\partial f(p^*)}{\partial p_2}$ nor $\frac{\partial f(p^*)}{\partial p_2} > \frac{\partial f(p^*)}{\partial p_1}$ can happen because that would imply that resp $p_2 = 0$ or $p_1 = 0$ and then the corresponding partial derivative would be infinite while the larger one would be zero. So we must conclude that $\frac{\partial f(p^*)}{\partial p_1} = \frac{\partial f(p^*)}{\partial p_2} = \lambda = 2$ and $p^* = (0.5, 0.5, 0)$ and $f(p^*) = -2\ln(2)$.

Part IX
Capacity revisited

25 Introduction

25.1 Problem statement

Problem
We studied the binary symmetric channel and the reliable transmission of binary information over this channel.

Now it is time to generalize this to arbitrary discrete alphabets, i.e. alphabets that contain a finite number of elements.

26 Channels

26.1 Mutual information

Assume that X and Y are discrete random variables that take values in the finite alphabet $\mathcal{X}$ and $\mathcal{Y}$ respectively.

Definition 34. We define the *mutual information* between X and Y by

$$I(X;Y) \triangleq \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2 \frac{p(x,y)}{p(x)p(y)}.$$

Again we assume that $0\log_2 0 = 0$, so that terms for which $p(x,y) = 0$ do not contribute.

26.2 Properties of mutual information

Assume that X and Y are discrete random variables that take values in $\mathcal{X}$ and $\mathcal{Y}$ respectively.

Theorem 35. *The mutual information $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. Furthermore $I(X;Y) \geq 0$ with equality only if X and Y are independent.*

Proof.

$$\begin{aligned}
I(X;Y) &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2 \frac{p(x,y)}{p(x)p(y)} \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2 \frac{1}{p(x)} + \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log_2 \frac{p(x,y)}{p(y)} \\
&= H(X) - H(X|Y).
\end{aligned}$$

In the same way one can show that $I(X;Y) = H(Y) - H(Y|X)$.

The fact that $I(X;Y) \geq 0$ with equality only if X and Y are independent follows from theorem: $H(X) \geq H(X|Y)$ with equality only if $p(x,y) = p(x)p(y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. □

Corollary 36. $I(X;Y) \leq \min\{H(X), H(Y)\}$.

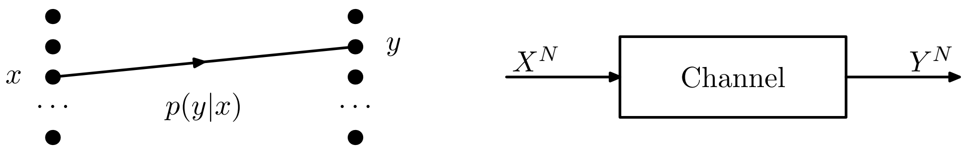


Figure 11: A discrete memoryless channel.

26.3 Interpretation of mutual information

If Y was not revealed, then our uncertainty about X would have been $H(X)$. Since knowledge of Y decreases our uncertainty of X we must conclude that Y contains information about X, or that there is an information flow from X to Y. The amount of information that Y contains about X (or that flows from X to Y) is equal to the decrease in uncertainty i.e. $H(X) - H(X|Y) = I(X;Y)$. Therefore the mutual information $I(X;Y)$ is a measure for the amount of information that flows from X to Y. Note that Y can not give us more information about X than X contains. This follows from the fact that $I(X;Y) \leq H(X)$.

Since the definition of mutual information is symmetrical in X and Y it is also true that $I(X;Y) = H(Y) - H(Y|X)$. Therefore the amount of information that Y gives about X is equal to the information that X contains about Y.

26.4 Discrete memoryless channel

Before we can define (channel) capacity we must discuss what a *discrete memoryless channel* is.

A discrete memoryless channel (see figure 11) is identified by a finite input alphabet $\mathcal{X}$, a finite output alphabet $\mathcal{Y}$ and a transition probability matrix $\{p(y|x), x \in \mathcal{X}, y \in \mathcal{Y}\}$. The number $p(y|x)$ represents the probability that y will be the output of the channel, given that x is the input.

Usually a sequence $x^N = (x_1, x_2, \cdots, x_N)$ of input symbols serves as the input for the channel. The output is a sequence $y^N = (y_1, y_2, \cdots, y_N)$ of output symbols. The probability $\Pr\{Y^N = y^N | X^N = x^N\}$ that the output sequence y^N occurs when x^N is the input sequence is denoted by $p(y^N|x^N)$. For a discrete memoryless channel

$$p(y^N|x^N) = p(y_1|x_1) \cdot p(y_2|x_2) \cdot \cdots \cdot p(y_N|x_N),$$

for all $x^N \in \mathcal{X}^N$ and $y^N \in \mathcal{Y}^N$, hence a current output depends only on the current input and not on previous inputs and (or) outputs.

What is the information flow over such a channel? Suppose we use the channel only once. We can then offer an input X to the channel which has distribution $\{p(x) : x \in \mathcal{X}\}$. Now the joint distribution $p(x,y)$ is fully determined by this input distribution $p(x)$ and the transition probability matrix $p(y|x)$ as follows :

$$p(x,y) = p(x) \cdot p(y|x) \text{ for all } x \in \mathcal{X} \text{ and } y \in \mathcal{Y}.$$

Now we can rewrite the mutual information in terms of the *channel input probabilities* $\Pr\{X = x\} = p(x)$ and the *channel transition probabilities* $\Pr\{Y = y|X = x\} = p(y|x)$ as

$$I(X;Y) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x)p(y|x) \log_2 \frac{p(y|x)}{\sum_{x'\in\mathcal{X}} p(x')p(y|x')}.$$

The information that is conveyed from X to Y is, as we have seen before, equal to $I(X;Y)$. Maximum information flow is obtained by choosing an input distribution $p(x)$ that maximizes $I(X;Y)$.

Definition 37. The *capacity* C of a discrete memoryless channel is defined as the maximum of the expected mutual information $I(X;Y)$ over all input probabilities $p(x) = \Pr\{X = x\}$, i.e.

$$C \triangleq \max_{(p(x):x\in\mathcal{X})} I(X;Y).$$

In the next lecture we shall see that $I(X;Y)$ is a concave function of the channel input probability vector p_X, so $I(X;Y)$ has a unique maximum and C is (relatively) easy to determine.

26.5 The BSC revisited

The mutual information of a BSC with cross-over probability α is

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

Now for the BSC we have

$$\begin{aligned} H(Y|X) &= \sum_{x\in\{0,1\}} p(x)H(Y|X=x) \\ H(Y|X=x) &= -\sum_{y\in\{0,1\}} p(y|x)\log_2 p(y|x) \\ &= h(\alpha). \end{aligned}$$

So,

$$C = \max_{p(x)} H(Y) - h(\alpha).$$

If we select

$$\Pr\{X=0\} = \Pr\{X=1\} = \frac{1}{2}$$

then the output probabilities are also

$$\Pr\{Y=0\} = \Pr\{Y=1\} = \frac{1}{2}$$

independent of α.

We know that in that case $H(Y) = 1$ is maximal, so we find the capacity of the BSC

$$C = 1 - h(\alpha).$$

Part X
Calculating entropies and capacity

27 Introduction

27.1 Problem statement

Problem
We have introduced the notions and basic formula's of entropy, mutual information, and capacity.

In this lecture we shall discuss the rules and methods to calculate these quantities efficiently. In the process we also derive a few important results on information transfer.

27.2 Recapitulation of results

$$\begin{aligned}
H(U) &\triangleq \sum_{u\in\mathcal{U}} p(u)\log_2\frac{1}{p(u)}\\
&\quad 0 \le H(U) \le \log_2|\mathcal{U}|\\
I(X;Y) &\triangleq \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2\frac{p(x,y)}{p(x)p(y)}\\
&= H(X)-H(X|Y)\\
&= H(Y)-H(Y|X)\\
&\quad 0 \le I(X;Y) \le \min\{H(X),H(Y)\}\\
H(U,V) &= H(U)+H(V|U)\\
&\quad 0 \le H(V|U) \le H(V)\\
C &\triangleq \max_{(p(x):x\in\mathcal{X})} I(X;Y) \qquad \text{for a fixed channel } p(y|x).
\end{aligned}$$

28 Further results

28.1 Conditional mutual information

The mutual information $I(X;Y|Z)$ describes the amount of information about X given by the observation of Y under the condition that a third random variable Z was already known, see Fig. 12.

So we have
$$I(X;Y|Z) \triangleq H(X|Z) - H(X|Y,Z).$$
It is easy to see that also holds
$$I(X;Y|Z) = H(Y|Z) - H(Y|X,Z).$$

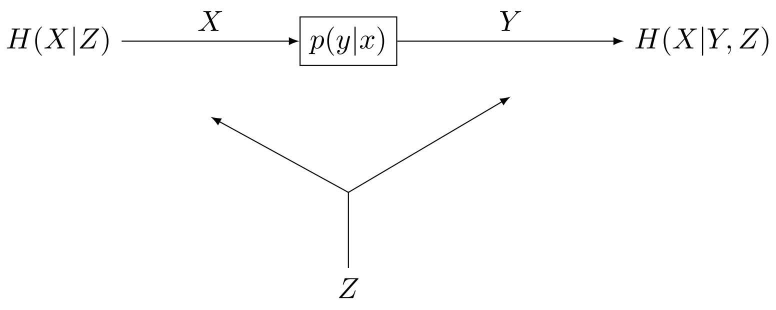


Figure 12: Conditional mutual information

This follows from

$$\begin{aligned} H(X,Y|Z) &= H(X|Z) + H(Y|X,Z) \\ &= H(Y|Z) + H(X|Y,Z) \end{aligned}$$

and so

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

28.2 Chain rule for mutual information

Lemma 38 (chain rule)**.**

$$I(X,Y;Z) = I(X;Z) + I(Y;Z|X).$$

Proof.

$$\begin{aligned} I(X,Y;Z) &= H(X,Y) - H(X,Y|Z) \\ &= \{H(X) + H(Y|X)\} - \{H(X|Z) + H(Y|X,Z)\} \\ &= \{H(X) - H(X|Z)\} + \{H(Y|X) - H(Y|X,Z)\} \\ &= I(X;Z) + I(Y;Z|X). \end{aligned}$$

□

Likewise holds

$$I(X,Y;Z) = I(Y;Z) + I(X;Z|Y).$$

28.3 Markov chain

Definition 39 (Markov chain)**.** Random variables X, Y, and Z form a Markov chain in that order if the joint probability satisfies

$$p(X,Y,Z) = p(X)p(Y|X)p(Z|Y).$$

This relation is denoted by $X \leftrightarrow Y \leftrightarrow Z$.

Markov chains have some nice properties. Let $X \leftarrow\!\ominus\!\rightarrow Y \leftarrow\!\ominus\!\rightarrow Z$. Then $p(Z|X,Y) = p(Z|Y)$. So given the middle variable (Y: *present*) the two opposite sides (X: *past* and Z: *future*) are independent, so also $p(X|Y,Z) = p(X|Y)$. This implies that $I(X;Z|Y) = 0$.

If $X \leftarrow\!\ominus\!\rightarrow Y \leftarrow\!\ominus\!\rightarrow Z$ then it is clear that $Z \leftarrow\!\ominus\!\rightarrow Y \leftarrow\!\ominus\!\rightarrow X$. The proof of this fact is left as an exercise.

28.4 The data processing theorem

The *data processing theorem* states that the information transfer, in an information theoretic sense, cannot be increased by preprocessing and/or postprocessing.

Theorem 40 (data processing theorem)**.** *For random variables X, Y, and Z that form a Markov chain $X \leftarrow\!\ominus\!\rightarrow Y \leftarrow\!\ominus\!\rightarrow Z$ holds*

$$I(X;Z) \leq I(X;Y).$$

Proof. We have

$$\begin{aligned} I(X;Y,Z) &= I(X;Y) + I(X;Z|Y) \\ &= I(X;Z) + I(X;Y|Z), \end{aligned}$$

Now $X \leftarrow\!\ominus\!\rightarrow Y \leftarrow\!\ominus\!\rightarrow Z$ and thus $I(X;Z|Y) = 0$

$$I(X;Y) = I(X;Z) + I(X;Y|Z)$$

and with $I(X;Y|Z) \geq 0$ we have

$$I(X;Y) \geq I(X;Z).$$

□

If $X \leftarrow\!\ominus\!\rightarrow Y \leftarrow\!\ominus\!\rightarrow Z$ and $I(X;Y) = I(X;Z)$ then $I(X;Y|Z)$ must be zero, and vice-versa. This can be seen from the previous proof and it implies that then also $X \leftarrow\!\ominus\!\rightarrow Z \leftarrow\!\ominus\!\rightarrow Y$.

29 Computing Channel capacity

29.1 Introduction

Now that we have seen the importance of the channel capacity we turn to the problem of computing the capacity of a given channel.

We shall repeat the capacity formula,

$$C \triangleq \max_{(p(x):x\in\mathcal{X})} I(X;Y).$$

Note that the expected mutual information $I(X;Y)$ depends both on the input distribution and on the channel transition probabilities. When computing the channel capacity for a given channel the transition probabilities are kept constant and we maximize over the input distribution. In this case we may consider the mutual information $I(X;Y)$ to be a function of the input distribution $p(X)$, so we can write $I(p(X)) \triangleq I(X;Y)$ to stress the dependence on $p(X)$, and $p(Y|X)$ is fixed.

Our problem of computing the capacity is simplified by the fact that $I(p(X))$ is a concave function, as will be shown in next weeks lecture.

We only state the theorem here.

Theorem 41. *Let $p(X)$ be the input distribution of a channel with transition probabilities $p(Y|X)$. Then the expected mutual information $I(X;Y)$ is a concave function of the input probabilities, $I(p(X))$.*

29.2 Symmetry in channels

Consider the binary symmetric channel with cross-over probability p, BSC, as depicted in figure 13. Let the input probabilities be $p(0) = 1 - p(1) = \alpha$, for an $\alpha \in [0,1]$. The mutual

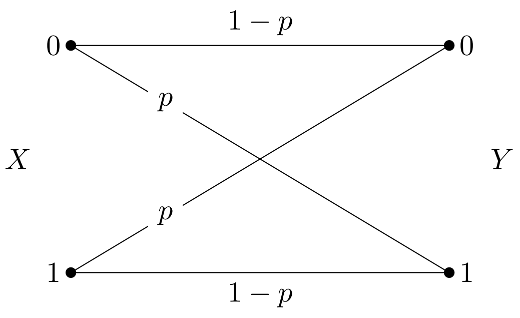


Figure 13: The binary symmetric channel.

information $I(X;Y)$ can be seen as a function of α, $I(\alpha)$. This mutual information can be shown to be

$$\begin{aligned} I(\alpha) = &-(1-\alpha-p+2\alpha p)\log_2(1-\alpha-p+2\alpha p) \\ &-(\alpha+p-2\alpha p)\log_2(\alpha+p-2\alpha p) - h(p). \end{aligned}$$

And the partial derivative is

$$\frac{\partial I}{\partial \alpha}(\alpha) = (1-2p)\log_2 \frac{\alpha+p-2\alpha p}{1-\alpha-p+2\alpha p}.$$

Solving α from $\frac{\partial I}{\partial \alpha}(\alpha) = 0$ gives us $\alpha = 1/2$ if $p \neq 1/2$. This is indeed the capacity achieving input distribution.

A disadvantage of this technique is its complexity, especially for larger channel input and output alphabets. We also have to consider the possibility that the maximum is achieved at a border of the convex probability region and that the partial derivative with respect to that parameter is non-zero. We shall use the Kuhn-Tucker theorem to find the optimum but first we shall try to simplify the problem.

Consider again the BSC. It doesn't matter how we draw the graph. In figure 14 we show that we can rotate the graph and the channel hasn't changed. However, note that *graph 2* is the mirror image of *graph 1* but also it is actually the same channel, only the inputs and outputs are relabeled.

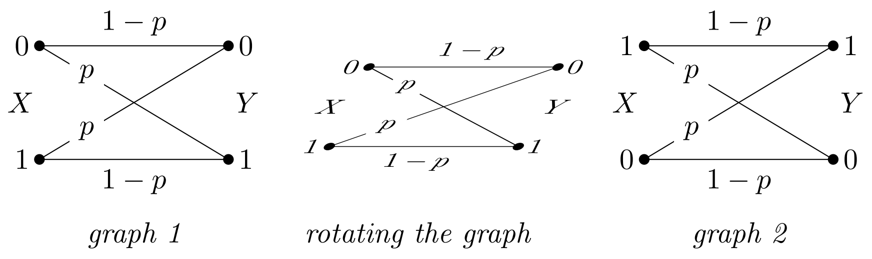


Figure 14: Symmetry in the binary symmetric channel.

So, let us assume that the probabilities $p(0) = a$, $p(1) = 1 - a$ achieve capacity, i.e.

$$I(a) = \max_{\alpha \in [0,1]} I(\alpha).$$

But due to the mirror symmetry we know that $p(0) = 1 - a$, $p(1) = a$ achieves the same mutual information. So $I(a) = I(1 - a)$. The concavity of the mutual information gives us that if

$$b \triangleq \gamma a + (1 - \gamma)(1 - a),$$

for some $\gamma \in [0, 1]$, then

$$I(b) \geq \gamma I(a) + (1 - \gamma) I(1 - a) = I(a).$$

Since $I(a)$ is maximal we must conclude that $I(b) = I(a)$ is also maximal. Now choose $\gamma = 1/2$ then $b = 1/2$ and we can conclude that $p(0) = 1/2$, $p(1) = 1/2$ is a capacity achieving input distribution.

This is the power of symmetry in channels. It reduces the number of parameters in the optimization. In this example it solves the optimization completely, but in general this might not be the case as the next channel will show.

Consider the channel as depicted in figure 15. Suppose that the input distribution $p(1) =$

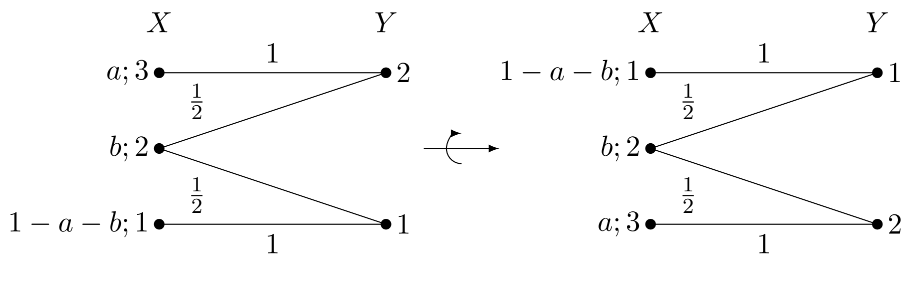


Figure 15: A partially symmetric channel.

$1-a-b$, $p(2) = b$, and $p(3) = a$ is a capacity achieving distribution, then due to the symmetry, so is $p(1) = a$, $p(2) = b$, and $p(3) = 1 - a - b$. And this again implies that the distribution $p(1) = \frac{1-a-b+a}{2} = \frac{1-b}{2}$, $p(2) = \frac{b+b}{2} = b$, and $p(3) = \frac{a+1-a-b}{2} = \frac{1-b}{2}$ achieves capacity. We have reduced the number of parameters from two to one !

Yet another example is given in figure 16, where $\gamma = 1 - \alpha - \beta$. If $p_1(1) = \alpha$, $p_1(2) = \beta$, and $p_1(3) = \gamma$ is a capacity achieving input distribution, then so is $p_2(1) = \beta$, $p_2(2) = \gamma$, and $p_2(3) = \alpha$ and likewise $p_3(1) = \gamma$, $p_3(2) = \alpha$, and $p_3(3) = \beta$. So the convex combination

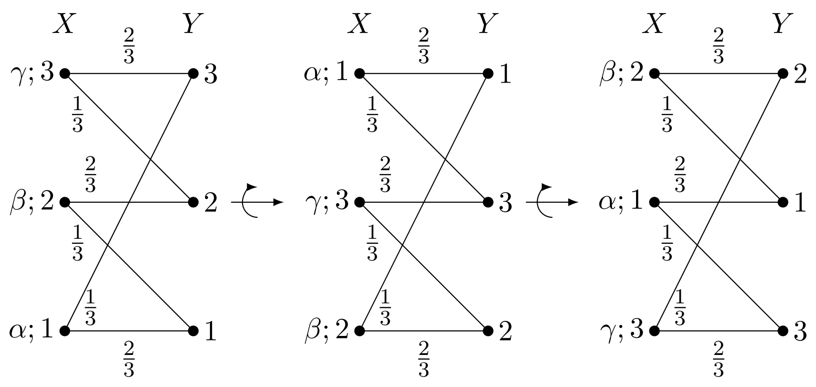


Figure 16: Another symmetric channel.

$p = 1/3(p_1+p_2+p_3) : p(1) = \frac{\alpha+\beta+\gamma}{3}, p(2) = \frac{\beta+\gamma+\alpha}{3}, p(3) = \frac{\gamma+\alpha+\beta}{3}$ or $p(1) = p(2) = p(3) = 1/3$ also achieves capacity!

We have not formally introduced the symmetry; the examples should be clear enough.

29.3 Kuhn-Tucker conditions

We have seen that the expected mutual information has a global maximum as a function of the input distribution. However, this optimum might still be difficult to compute. The considerations of symmetry in channels can help in some cases, but often we will have to rely on the more general method of Kuhn and Tucker.

We repeat the theorem.

Theorem 42 (Kuhn-Tucker). *Let $D_{|\mathcal{X}|}$ be the convex region of all $|\mathcal{X}|$ dimensional probability vectors or*

$$D_{|\mathcal{X}|} = \left\{ \underline{p} \mid \sum_{x\in\mathcal{X}} p(x) = 1, p(x) \geq 0 \textit{ for all } x \in \mathcal{X} \right\}.$$

$f : D_{|\mathcal{X}|} \mapsto \mathbb{R}$ is a concave function whose partial derivatives $\frac{\partial f}{\partial p(x)}(\underline{p})$, for all $x \in \mathcal{X}$, are continuous in $D_{|\mathcal{X}|}$. Then $\underline{p}$ is a global maximum of f if and only if there exists a constant λ such that

$$\frac{\partial f}{\partial p(x)}(\underline{p}) = \lambda \textit{ if } p(x) > 0,$$

and

$$\frac{\partial f}{\partial p(x)}(\underline{p}) \leq \lambda \textit{ if } p(x) = 0.$$

The expected mutual information, as a function of the input distribution, satisfies the requirements for theorem 42, i.e. it is concave and has continuous partial derivatives.

The partial derivative of $I(X;Y)$ with respect to the input probability $p(x)$ is

$$\begin{aligned}
\frac{\partial I(X;Y)}{\partial p(x)} &= \frac{\partial}{\partial p(x)} \sum_{x' \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x')p(y|x') \log_2 \frac{p(y|x')}{\sum_{x'' \in \mathcal{X}} p(x'')p(y|x'')}, \\
&= \sum_{y \in \mathcal{Y}} p(y|x) \log_2 \frac{p(y|x)}{\sum_{x' \in \mathcal{X}} p(x')p(y|x')} - \\
&\qquad \sum_{x' \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x')p(y|x') \frac{p(y|x)}{\sum_{x'' \in \mathcal{X}} p(x'')p(y|x'')} \log_2 e, \\
&= \sum_{y \in \mathcal{Y}} p(y|x) \log_2 \frac{p(y|x)}{\sum_{x' \in \mathcal{X}} p(x')p(y|x')} - \log_2 e.
\end{aligned}$$

We shall use a special notation for the righthandside of the last expression, so for all $x \in \mathcal{X}$

$$I(X = x; Y) \triangleq \sum_{y \in \mathcal{Y}} p(y|x) \log_2 \frac{p(y|x)}{\sum_{x' \in \mathcal{X}} p(x')p(y|x')}.$$

$I(X = x; Y)$ is sometimes called the *Kuhn-Tucker mutual information.*

Now we are ready to state the condition for a capacity achieving input distribution.

Theorem 43. *An input probability distribution $\underline{p} = (p(x) : x \in \mathcal{X})$ is a capacity achieving input distribution for a channel with channel transition probabilities $p(Y|X)$ if and only if*

$$\begin{aligned}
I(X = x; Y) &= C; && \text{for all } x \text{ with } p(x) > 0, \\
I(X = x; Y) &\leq C; && \text{for all } x \text{ with } p(x) = 0.
\end{aligned}$$

Proof. From the previous two theorems we know that a capacity achieving input distribution satisfies, for some λ,

$$\begin{aligned}
I(X = x; Y) &= \lambda + \log_2 e; && \text{for all } x \text{ such that } p(x) > 0, \\
I(X = x; Y) &\leq \lambda + \log_2 e; && \text{for all } x \text{ such that } p(x) = 0.
\end{aligned}$$

We can write the mutual information as

$$\begin{aligned}
I(X;Y) &= \sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log_2 \frac{p(y|x)}{p(y)} \\
&= \sum_{x \in \mathcal{X}} p(x) I(X = x; Y).
\end{aligned}$$

If $p(X)$ is capacity achieving then $I(X;Y) = C$ and for all x with $p(x) > 0$ we have $I(X = x; Y) = \lambda + \log_2 e$, so

$$\sum_{x \in \mathcal{X}} p(x) I(X = x; Y) = \lambda + \log_2 e = I(X;Y) = C$$

and this proves the theorem. □

Example

Consider the channel as depicted in figure 17.

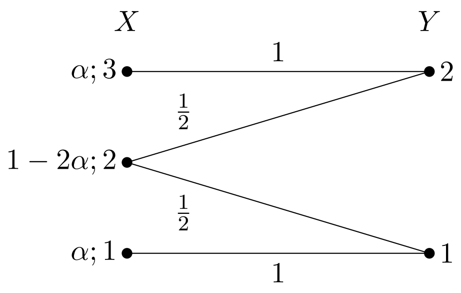


Figure 17: First Kuhn-Tucker example.

From the symmetry in the channel we know that a capacity achieving input distribution exists for which $p(1) = p(3)$. So we set $p(1) = p(3) = \alpha$ and $p(2) = 1 - 2\alpha$. Then the output probabilities can be determined and we find

$$\Pr\{Y = 1\} = \Pr\{Y = 2\} = \frac{1}{2}.$$

We compute the Kuhn-Tucker mutual informations, but due to the symmetry we always have $I(X = 1; Y) = I(X = 3; Y)$, so

$$\begin{aligned} I(X = 1; Y) &= 1 \cdot \log_2 \frac{1}{\Pr\{Y = 1\}} = 1, \\ I(X = 2; Y) &= \frac{1}{2} \cdot \log_2 \frac{1/2}{\Pr\{Y = 1\}} + \frac{1}{2} \cdot \log_2 \frac{1/2}{\Pr\{Y = 2\}} = 0. \end{aligned}$$

So, we see that for all $\alpha \in [0, 1]$

$$I(X = 1; Y) = I(X = 3; Y) > I(X = 2; Y).$$

Thus $p(2)$ must be zero, so $\alpha = 1/2$ and the capacity is $C = 1$ bit per transmission.

Example
Let the channel be given as in figure 18.

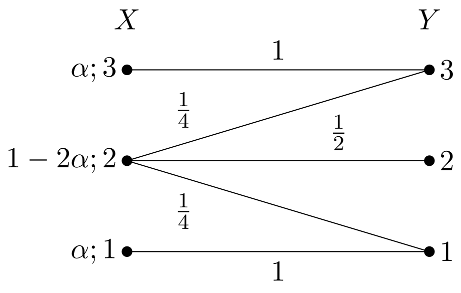


Figure 18: Second Kuhn-Tucker example.

From the symmetry in the channel we know that a capacity achieving input distribution exists for which $p(1) = p(3)$. So we set $p(1) = p(3) = \alpha$ and $p(2) = 1 - 2\alpha$. Then the output probabilities can be expressed in α and we find

$$\begin{aligned}
\Pr\{Y=1\} = \Pr\{Y=3\} &= \frac{1}{4} + \frac{1}{2}\alpha, \\
\Pr\{Y=2\} &= 1 - 2\Pr\{Y=1\}.
\end{aligned}$$

We compute the Kuhn-Tucker mutual informations, but due to the symmetry we always have $I(X=1;Y) = I(X=3;Y)$, so

$$\begin{aligned}
I(X=1;Y) &= 1 \cdot \log_2 \frac{1}{\Pr\{Y=1\}}, \\
I(X=2;Y) &= \frac{1}{4} \cdot \log_2 \frac{1/4}{\Pr\{Y=1\}} + \frac{1}{2} \cdot \log_2 \frac{1/2}{\Pr\{Y=2\}} + \\
&\quad \frac{1}{4} \cdot \log_2 \frac{1/4}{\Pr\{Y=3\}}.
\end{aligned}$$

We assume equality and try to solve

$$\begin{aligned}
I(X=1;Y) &= I(X=2;Y) \\
\log_2 \frac{1}{\Pr\{Y=1\}} &= \frac{1}{2} \cdot \log_2 \frac{1/4}{\Pr\{Y=1\}} + \frac{1}{2} \cdot \log_2 \frac{1/2}{\Pr\{Y=2\}} \\
\frac{1}{2} \log_2 \frac{\Pr\{Y=2\}}{\Pr\{Y=1\}} &= -\frac{3}{2} \\
\Pr\{Y=2\} &= 2^{-3} \Pr\{Y=1\}.
\end{aligned}$$

Using $\Pr\{Y=2\} = 1 - 2\Pr\{Y=1\}$ we finally have $\Pr\{Y=1\} = \Pr\{Y=3\} = 8/17$, $\Pr\{Y=2\} = 1/17$, and $C = \log_2(17/8) = 1.0875$ bit per transmission.

Part XI

Two converses (discrete sequences)

30 Introduction

30.1 Robert M. Fano

Robert Mario Fano (born 1917) is professor emeritus of Electrical Engineering and Computer Science at Massachusetts Institute of Technology. Fano is known principally for his work on information theory, inventing (with Claude Shannon) Shannon-Fano coding. In the early 1960s, he was involved in the development of time-sharing computers, and served as director of MIT's Project MAC from its founding in 1963 until 1968. Project MAC would become famous for ground-breaking research in operating systems, artificial intelligence, and the theory of computation.

Fano was born in Torino, Italy, where he lived until 1939, when he emigrated to the United States. Fano is a member of the U.S. National Academy of Sciences and the National Academy of Engineering. (from Wikipedia).

30.2 Problem statement

We have shown that reliable transmission over a channel with capacity C is possible for rates $R < C$. We have not considered transmission at rates larger than C yet, but the name "capacity" more or less implies that reliable transmission is not possible in that case. Shannon [1948] addressed this problem and states that if the equivocation $H(M|\widehat{M})$ is larger than zero the frequency of errors cannot be made arbitrarily small. A stronger statement was made by Wolfowitz [1957] who showed that for rates above channel capacity for $N \to \infty$ the error probability $P_e \to 1$. Here we will study an alternative approach based on Fano's inequality [1952] and show that the error probability cannot be made arbitrarily small for $R > C$.

We will also look at the optimality of our codes for lossless source coding.

31 Fano's inequality

31.1 Statement

Theorem 44 (Fano, 1952). *Let U and its estimate $\widehat{U}$ both assume values in the alphabet $\mathcal{U}$ that consists of $|\mathcal{U}|$ elements. When their joint distribution is $\{P(u,\widehat{u}), u \in \mathcal{U}, \widehat{u} \in \mathcal{U}\}$ the error probability P_e can be expressed as*

$$P_e \stackrel{\Delta}{=} \Pr\{\widehat{U} \neq U\} = \sum_u \sum_{\widehat{u} \neq u} P(u,\widehat{u}). \tag{5}$$

Then the following inequality holds:

$$P_e \log_2(|\mathcal{U}| - 1) + h(P_e) \geq H(U|\widehat{U}). \tag{6}$$

31.2 Interpretation

If $P_e \downarrow 0$ then also $H(U|\widehat{U}) \downarrow 0$ fast enough. See Fig. 19 for some examples.

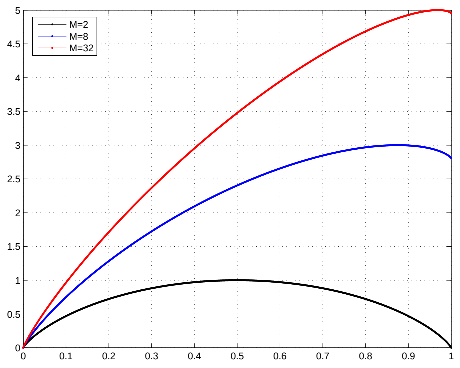


Figure 19: Fano's inequality for three alphabetsizes. Horizontally P_e, vertically $H(U|\widehat{U})$. Only pairs $(P_e, H(U|\widehat{U}))$ below the curves are possible.

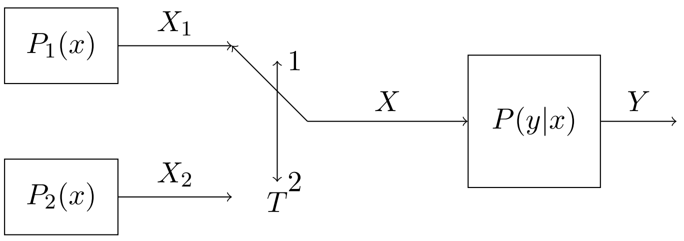


Figure 20: The concavity of $I(X;Y)$ in $P(x)$

31.3 Proof of Fano's Inequality

Let E be a random variable that indicates an error. Its value is 1 if $\widehat{U} \neq U$ and 0 if $\widehat{U} = u$. Then

$$\begin{aligned} H(U|\widehat{U}) \leq H(U,E|\widehat{U}) &= H(E|\widehat{U}) + H(U|\widehat{U},E) \\ &\leq H(E) + H(U|\widehat{U},E) \\ &= H(E) + \sum_e P(e) \sum_{u,\widehat{u}} P(u,\widehat{u}|e) \log_2 \frac{1}{P(u|\widehat{u},e)} \\ &= H(E) + \sum_e P(e) H(U|\widehat{U}, E=e) \\ &= h(P_e) + P_e H(U|\widehat{U}, e=1) \\ &\leq h(P_e) + P_e \log_2(|\mathcal{U}|-1). \end{aligned} \tag{7}$$

The last equality holds since $H(U|\widehat{U}, e=0) = 0$.

32 Convexity

32.1 $I(X;Y)$ is concave in $P(x)$

Consider the situation of Fig. 20.

Fig. 20, defines the probability $P(T,X,Y) = P(T)P(X|T)P(Y|X)$, so $T \leftrightarrow X \leftrightarrow Y$ is a

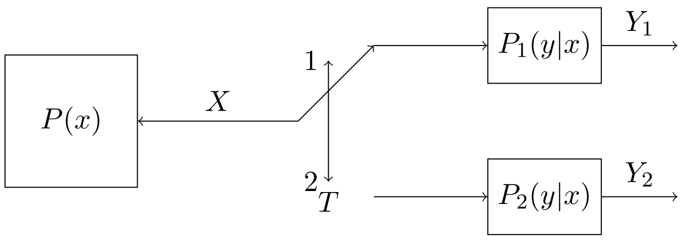


Figure 21: The convexity of $I(X;Y)$ in $P(y|x)$

Markov chain.

$$\begin{aligned} I(T,X;Y) &= I(T;Y) + I(X;Y|T) \\ &= I(X;Y) + I(T;Y|X). \end{aligned} \tag{8}$$

Since $I(T;Y|X) = 0$ we obtain that $I(X;Y) \geq I(X;Y|T)$. Note that

$$P(x) = \Pr\{T=1\}P_1(x) + \Pr\{T=2\}P_2(x). \tag{9}$$

Therefore, for each fixed $P(y|x)$, we can write

$$\begin{aligned} I(X;Y) &= \sum_{x,y} P(x)P(y|x)\log\frac{P(y|x)}{\sum_{x'} P(x')P(y|x')} \\ &= f_{\mathcal{I}}(P(x)) \\ &= f_{\mathcal{I}}(\Pr\{T=1\}P_1(x) + \Pr\{T=2\}P_2(x)). \end{aligned} \tag{10}$$

Moreover

$$\begin{aligned} I(X;Y|T) &= \Pr\{T=1\}\sum_{x,y} P_1(x)P(y|x)\log\frac{P(y|x)}{\sum_{x'} P_1(x')P(y|x')} \\ &\quad + \Pr\{T=2\}\sum_{x,y} P_2(x)P(y|x)\log\frac{P(y|x)}{\sum_{x'} P_2(x')P(y|x')} \\ &= \Pr\{T=1\}f_{\mathcal{I}}(P_1(x)) + \Pr\{T=2\}f_{\mathcal{I}}(P_2(x)). \end{aligned} \tag{11}$$

With $\Pr\{T=1\} = \alpha$ and $\Pr\{T=2\} = 1-\alpha$ we obtain therefore

$$f_{\mathcal{I}}(\alpha P_1(x) + (1-\alpha)P_2(x)) \geq \alpha f_{\mathcal{I}}(P_1(x)) + (1-\alpha)f_{\mathcal{I}}(P_2(x)), \tag{12}$$

hence $f_{\mathcal{I}}(P(x))$ is concave in $P(x)$.

32.2 $I(X;Y)$ is convex in $P(y|x)$

Consider the situation of Fig. 21.

Fig. 21, defines the probability $P(X,T,Y) = P(X)P(T)P(Y|X,T)$.

$$\begin{aligned} I(X;Y,T) &= I(X;T) + I(X;Y|T) \\ &= I(X;Y) + I(X;T|Y). \end{aligned} \tag{13}$$

Since $I(X;T)=0$ we obtain that $I(X;Y) \leq I(X;Y|T)$.

First note that

$$P(y|x) = \Pr\{T=1\}P_1(y|x) + \Pr\{T=2\}P_2(y|x). \tag{14}$$

Therefore, for each fixed $P(x)$, we can write

$$\begin{aligned} I(X;Y) &= \sum_{x,y} P(x)P(y|x) \log \frac{P(y|x)}{\sum_{x'} P(x')P(y|x')} \\ &= g_{\mathcal{I}}(P(y|x)) \\ &= g_{\mathcal{I}}(\Pr\{T=1\}P_1(y|x) + \Pr\{T=2\}P_2(y|x)). \end{aligned} \tag{15}$$

Moreover

$$\begin{aligned} I(X;Y|T) &= \Pr\{T=1\} \sum_{x,y} P(x)P_1(y|x) \log \frac{P_1(y|x)}{\sum_{x'} P(x')P_1(y|x')} \\ &\quad + \Pr\{T=2\} \sum_{x,y} P(x)P_2(y|x) \log \frac{P_2(y|x)}{\sum_{x'} P(x')P_2(y|x')} \\ &= \Pr\{T=1\} g_{\mathcal{I}}(P_1(y|x)) + \Pr\{T=2\} g_{\mathcal{I}}(P_2(y|x)). \end{aligned} \tag{16}$$

With $\Pr\{T=1\}=\alpha$ and $\Pr\{T=2\}=1-\alpha$ we obtain therefore

$$g_{\mathcal{I}}(\alpha P_1(y|x) + (1-\alpha)P_2(y|x)) \leq \alpha g_{\mathcal{I}}(P_1(y|x)) + (1-\alpha) g_{\mathcal{I}}(P_2(y|x)), \tag{17}$$

hence $g_{\mathcal{I}}(P(y|x))$ is convex in $P(y|x)$.

33 Limit on information transfer

33.1 Capacity limits the information transfer

In an information theoretic sense the amount of information transferred over a channel by a single transmission is given by $I(X;Y)$. This is upper bounded by the channel capacity C. The first partial answer to the question what happens when $R_c > C$ is given by the theorem below.

Theorem 45. *For any discrete and memoryless channel, defined by the input alphabet $\mathcal{X}$, the output alphabet $\mathcal{Y}$ and the channel transition probability matrix $p(y|x)$, we have*

$$I(X^N;Y^N) \leq \sum_{n=1,N} I(X_n;Y_n). \tag{18}$$

Because each term $I(X_n;Y_n)$ in (18) is upper bounded by C we immediately have

$$I(X^N;Y^N) \leq NC. \tag{19}$$

Thus we see that in an information theoretic sense, the information transfer over a discrete, memoryless channel is limited by the channel capacity even when we do multiple transmissions. So, we expect that reliable transmission will be impossible at rates larger than capacity.

Proof.

$$I(X^N;Y^N) = H(Y^N) - H(Y^N|X^N),$$

and with

$$\begin{aligned} H(Y^N) &= H(Y_1) + H(Y_2|Y_1) + \cdots + H(Y_N|Y_1 \cdots Y_{N-1}) \\ &\leq \sum_{n=1,N} H(Y_n),. \end{aligned}$$

Using the fact that the channel is memoryless, we find

$$\begin{aligned} H(Y^N|X^N) &= H(Y_1|X^N) + H(Y_2|X^N, Y_1) + \cdots + H(Y_N|X^N, Y^{N-1}) \\ &= \sum_{n=1,N} H(Y_n|X_n), \end{aligned}$$

and thus

$$I(X^N;Y^N) \leq \sum_{n=1,N} [H(Y_n) - H(Y_n|X_n)] = \sum_{n=1,N} I(X_n;Y_n).$$

□

34 Two converses

34.1 Lossless Source Coding

The source entropy is $H(U)$. Consider a code with $S = 2^{NR}$ codewords. The codeword with index M corresponds to estimated sequence $\widehat{U}^N$ of length N. The error probability $P_e = \Pr\{\widehat{U}^N \neq U^N\}$. Now:

$$\begin{aligned} \log_2 S &\geq H(M) \\ &\geq I(U^N;M)(\text{because } I(U^N;M) = H(M) - H(M|U^N)) \\ &\geq I(U^N;\widehat{U}^N) \\ &= H(U^N) - H(U^N|\widehat{U}^N) \\ &\geq H(U^N) - P_e \log_2(|\mathcal{U}|^N - 1) - h(P_e) \\ &\geq NH(U) - P_e N \log_2 |\mathcal{U}| - 1. \end{aligned} \tag{20}$$

Therefore

$$R = \frac{\log_2 S}{N} \geq H(U) - P_e \log_2 |\mathcal{U}| - \frac{1}{N}. \tag{21}$$

For error probability $P_e \downarrow 0$ we now obtain that $R \geq H(U)$.

34.2 Channel Coding

Code $\{x^N(m), m = 1, 2, \cdots, S\}$ consists of $S = 2^{NR}$ codewords that are indexed by M which is uniform. Then:

$$\begin{aligned}
\log_2 S &= H(M) \\
&= I(M; Y^N) + H(M|Y^N) \\
&\leq I(X^N; Y^N) + H(M|\widehat{M}) \\
&\leq \sum_{n=1,N} I(X_n; Y_n) + H(M|\widehat{M}) \\
&\leq NI(X;Y) + P_e \log_2(S-1) + h(P_e) \\
&\leq NI(X;Y) + P_e \log_2 S + 1, \qquad (22)
\end{aligned}$$

where for $x \in \mathcal{X}$, and $y \in \mathcal{Y}$

$$\Pr\{X = x, Y = y\} = \frac{1}{N} \sum_{n=1,N} \Pr\{X_n = x\} P^*(y|x), \qquad (23)$$

hence this probability distribution depends on the code that is used. Therefore

$$NR(1 - P_e) = (1 - P_e) \log_2 S \leq NI(X;Y) + 1. \qquad (24)$$

or

$$R \leq \frac{I(X;Y) + 1/N}{1 - P_e}. \qquad (25)$$

Observing that the inequality has to hold for any code we may conclude for error probability $P_e \downarrow 0$ en $N \to \infty$ that $R \leq \max_{P(x)} I(X;Y)$.

Part XII
Cryptography

35 Introduction

35.1 Nomenclature/Assumptions

- *Cryptology* is the whole area of secure communication.
  - *Cryptography* The design of systems for secure communication.
  - *Cryptanalysis* The breaking of a secure system.
- *Cipher* is the complete cryptographic system.
- *Plaintext* is the original message.
- *Cryptogram* or *ciphertext* is the encoded (encrypted) message.
- *Key* is the *secret* key that controls the encoding and decoding process.
- *Kerckhoff's assumption* The enemy cryptanalyst possesses the complete cryptogram and has full knowledge of the cryptographic system, except for the secret key.
- *Ciphertext only attack* is an attempt by the cryptanalyst to obtain the plaintext or the key when Kerckhoff's assumption is satisfied.
- *known-plaintext attack* when also (a part of) the plaintext is known by the cryptanalyst.
- *chosen-plaintext attack* when the cryptanalyst can select a text that will be encrypted with the secret key and the resulting ciphertext can be analyzed.
- *chosen-ciphertext attack* when the cryptanalyst can select a cryptogram and have it decoded using the secret key and the resulting plaintext (usually meaningless) can be analyzed.

We shall consider only the ciphertext-only attack.
In principle the cryptanalyst will try to determine the a posteriori probabilities of the plaintext (or key) given the cryptogram and choose that plaintext (key) with the largest probability.
In practice this approach is impossible due to its enormous complexity. However, it is a good method for the analysis of the strength of a cipher.

36 The (pre-)history of cryptology

36.1 Before Shannon

Before Shannon published his paper "Communication Theory of Secrecy Systems" in 1949 cryptology was more an *art* than a *science*.

We shall give some examples of ciphers that were once used or proposed.

These ciphers can be subdivided into two groups, the *substitution* ciphers and the *transposition* ciphers. In substitution ciphers the different letters from the alphabet of the plaintext are replaced by other letters from the same or even another alphabet. A transposition cipher reorders the positions of the letters in the plaintext.

36.2 Simple substitution

The letters from the message alphabet (plaintext alphabet) are each replaced by a fixed substitute, usually chosen from the same alphabet. If the message and the substitution alphabet are the same then the secret key can be described by a permutation on the alphabet.

In the following examples we shall consider the 26 letter alphabet $\{A, B, \ldots, Z\}$.

Caesar Here the permutation is a fixed cyclic shift of the alphabet. About 2000 years ago Julius Caesar used the fixed shift 3, so the letter A was replaced by D, and Z by C.

The plaintext `BEVEILIGING` now becomes, (with shift 3), `EHYHLOLJLQJ`.

There are 26 different keys.

Arbitrary permutation In this case all permutations are possible. An example is the permutation $(B, D, X, V, J, K, G, I, W, P, A, M, Q, L, S, C, Y, F, U, E, O, H, Z, T, R, N)$.

The plaintext `BEVEILIGING` now becomes `DJHJWMWGWLG`.

There are $26! = 4.033 \cdot 10^{26}$ possible keys.

Vigenère In the Vigenère cipher the letters A to Z are mapped onto the integers 0 to 25. The key of length d letters is written repeatedly below the plaintext. The ciphertext letters are obtained from the modulo 26 sums.

Assume that the keytext is `INFO`, i.e. the integers $(8, 13, 5, 14)$. The plaintext `BEVEILIGING` now becomes `JRASQYNUQAL` by

B	E	V	E	I	L	I	G	I	N	G
1	4	21	4	8	11	8	6	8	13	6
I	N	F	O	I	N	F	O	I	N	F
8	13	5	14	8	13	5	14	8	13	5
9	17	0	18	16	24	13	20	16	0	11
J	R	A	S	Q	Y	N	U	Q	A	L

There are $26^d (26^4 = 456976)$ possible keys.

Vernam The Vernam cipher is often called the *one time pad.* It is a Vigenère cipher with a non repeating completely random key. This system was used in the *hot line* between the presidents of the U.S.A. and the U.S.S.R. From its invention one has always assumed that this system is unbreakable.

When the key text is not completely random but a meaningful text, i.e. a page from a book, then the cipher is called a *running key cipher.*

N-gram substitution It is also possible to define a permutation on sequences of two or more letters, thus extending the size of the alphabet and the number of possible keys.

36.3 Transposition cipher

A transposition cipher (with fixed period d) splits the plaintext into groups of d letters and applies a fixed permutation to the letter positions in each group. Suppose $d = 5$ and the permutation is $(2, 3, 1, 5, 4)$.

The plaintext `BEVEILIGING` now becomes `EVBIEIEIGLNIG`.

The repeated application of a transposition on a text is called a *composed transposition.* If the s transpositions have respective periods d_1, d_2, …, d_s, the composed transposition is a transposition with period D, where d is the least multiple of d_1, d_2, …, d_s.

36.4 Composed ciphers

As described before a second cipher can be applied after a first whenever the cipher alphabet of the first cipher matches the message alphabet of the second cipher. These composed ciphers are also known as *product ciphers.*

Mixed alphabet Vigenère This is a simple substitution cipher followed by a Vigenère. Let x_i be the i^{th} letter of the plaintext and $f(x_i)$ the substituted letter according to the substitution. Furthermore let z_i be the integer that corresponds in the Vigenère key with the i^{e} letter and let y_i be the letter of the resulting cryptogram. Then we have

$$y_i = f(x_i) + z_i \pmod{26}$$

Matrix system This is a n-gram substitution. We use calculations modulo 26 with the letter representation $A = 0$, …, $Z = 25$. The square $n \times n$ matrix $[a_{ij}]$, $1 \le i \le n$ and $1 \le j \le n$, has an inverse in the ring. We find

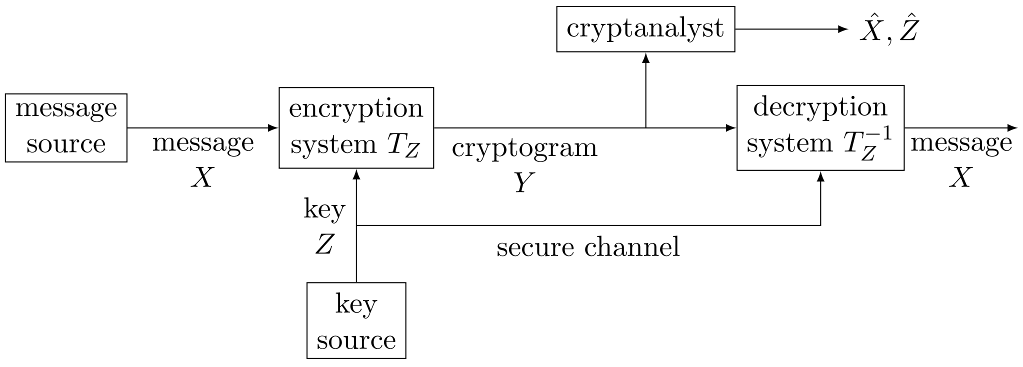
$$y_i = \sum_{j=1,n} a_{ij} x_j, \quad i = 1, \cdots, n.$$

In this matrix we can describe a transposition cipher as follows. Each row a_{ij} ($j = 1, \cdots, n$) contains a single 1 and for the rest zeros.

When the matrix is diagonal, so only $a_{ii} \neq 0$, then we have a letter substitution cipher.

37 The cryptosystem

37.1 System diagram



Message source: X.	Key source: Z.
Cryptogram: $Y = T_Z(X)$.	Decryption: $X = T_Z^{-1}(Y)$.
Cryptanalyst: $\hat{X}, \hat{Z}$.	

37.2 Stochastic description

We assume that the number of keys Z is finite. Key z will be selected for use with a probability $\Pr\{Z = z\}$. Key Z determines an encryption and decryption function T_Z resp. T_Z^{-1}. To stress the fact that the same key is used for encryption and decryption we name these systems *one-key cryptosystems* or *symmetric cryptosystems.*

We also assume that the number of possible messages of a given length is finite, possibly because the message alphabet is finite. So, just as in Information Theory, the source is modeled as a stochastic process, so there exist $\Pr\{X = x\}$ for all messages x.

37.3 Quality criteria for ciphers

1. *Amount of security.* The best we can hope for is that the cryptanalyst after receiving the cryptogram y has no more information than before. In terms of probabilities we ask for $\Pr\{X = x|Y = y\} = \Pr\{X = x\}$. A less severe condition is that there is no unique solution, i.e. there are more messages for which the a posteriori probability $\Pr\{X = x|Y = y\}$ is "large". Furthermore there can be systems that have a unique solution, i.e. $\Pr\{X = x|Y = y\}$ is much larger for the correct message x than for any other message, but the amount of work to be done in order to find this solution is prohibitive.

2. *Key size.* The secret key has to be transmitted over a secure channel and stored at both encoder and decoder. Thus the key size should be as small as possible.

3. *Complexity of encrypting and decrypting.* The complexity should be small as to improve speed and reliability of the secure transmission.

4. *Error propagation.* It is undesirable that errors in the transmission and decryption lead to long sequences of erroneous message symbols.

5. *Message expansion.* Some cryptosystems create ciphertexts that are much longer than the corresponding plaintexts. This is often done in an attempt to hide source statistics. The expansion however is inefficient and undesirable.

37.4 Theoretic secrecy

We want to know how secure a cryptosystem is against a ciphertext-only attack, when the cryptanalyst can spend an arbitrary amount of time and computing power. So, we want to know if a given cryptogram has a unique solution.

We consider the stochastic system consisting of the random key selection $\Pr\{Z = z\}$ from a finite number of keys (key source) and a message selection with probabilities $\Pr\{X = x\}$ from a finite set of messages (message source).

The cryptanalyst intercepts a cryptogram Y and determines the a posteriori probabilities $\Pr\{X = x|Y = y\}$.

37.5 Perfect secrecy

A cipher is *perfect* when for every cryptogram y the a posteriori probability is equal to the a priori probability of the plaintext x. The cryptanalyst has then not received any information about the plaintext. This condition implies that the plaintext and the cryptogram have to be *statistically independent*.

We shall now determine the "uncertainty" over the plaintexts X. For this we use the entropy. We know that if

$$H(X|Y) = 0,$$

then for every cryptogram y the plaintext x is determined uniquely.

The cryptographic function T_Z (encryption) computes the cryptogram Y from the key Z and the plaintext X, likewise the function T_Z^{-1} computes the plaintext from the key and the cryptogram. So we know that

$$\begin{aligned} H(Y|X,Z) &= 0, \\ H(X|Y,Z) &= 0. \end{aligned}$$

The condition for perfect secrecy, i.e. statistical independence for X and Y can be written as

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

Now using the simple properties of entropies as given above we shall derive a condition for the key entropy $H(Z)$ for perfect secrecy.

For every cipher holds

$$\begin{aligned} H(X,Z|Y) &= H(X|Y) + H(Z|X,Y) \\ &\geq H(X|Y). \end{aligned}$$

Thus follows

$$\begin{aligned} H(X|Y) &\leq H(Z|Y) + H(X|Y,Z) \\ &= H(Z|Y) \\ &\leq H(Z). \end{aligned}$$

In the last step we used the fact that conditioning cannot increase the entropy.

For a system with perfect secrecy we we now conclude that

$$H(X) = H(X|Y) \leq H(Z).$$

So for perfect secrecy the key uncertainty must be at least as large as the message uncertainty.

If the key is K symbols long and the key alphabet contains L_Z symbols then we have

$$H(Z) \leq K \log L_Z,$$

with equality only if the key selection is completely random.

For the message source we also know that

$$H(X) \leq N \log L_X,$$

where L_X is the source alphabet size and N the length of the plaintext.

Now if $L_X = L_Z$, as in the one-time pad, and the plaintext is completely random then

$$K \geq N.$$

So the key has to be at least as large as the plaintext, cf. the one time pad.

38 Unicity distance

38.1 Small key sets

The 2nd quality criterion: we want small key sizes so no perfect secrecy is possible. So we ask the question: for which n does the cryptanalyst know the key from the first n letters of the cryptogram, with large probability?

It is useful to study the *key equivocation function*

$$f(n) = H(Z|Y_1, Y_2, \cdots, Y_n).$$

$f(n)$ measures the uncertainty the cryptanalyst has over the key Z after he has intercepted the first n symbols of the cryptogram.

The smallest value of n for which $f(n) \approx 0$ is called the *unicity distance.* Solution of the cryptogram becomes unique, so the key Z becomes known.

The definition of the unicity distance is not precise but still useful.

38.2 Key equivocation and redundancy

Theorem: $f(n)$ is non-increasing in n.

The proof follows simply from the fact that conditioning cannot increase the entropy.

We use the following notation.

$\mathcal{X}$: The set of all messages of length N.

$\mathcal{Y}$: The set of all possible cryptograms.

$\mathcal{Z}$: The set of keys.

We write $|\mathcal{X}|$ for the number of messages, and likewise $|\mathcal{Y}|$ and $|\mathcal{Z}|$. If the message (letter-)alphabet contains L_X letters and every letter combination can be a message then $|\mathcal{X}| = (L_X)^N$.

We define the *per letter redundancy* of the messages as

$$D = \frac{\log|\mathcal{X}| - H(X)}{N}.$$

We now approximate the decrease in key uncertainty, $H(Z) - H(Z|Y)$, after the observation of the cryptogram for N message letters.

Because the plaintext and the key are independent we have

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

and also

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

This follows from

$$\begin{aligned} H(X,Y,Z) &= H(X,Z) + H(Y|X,Z) = H(X,Z) \\ &= H(Y,Z) + H(X|Y,Z) = H(Y,Z) \\ &= H(Y) + H(Z|Y). \end{aligned}$$

When we combine these equalities we obtain

$$H(Z|Y) = H(Z) - (H(Y) - H(X)).$$

An assumption that holds for many ciphers is that $|\mathcal{Y}| = |\mathcal{X}|$. Then we have that $H(Y) \leq \log|\mathcal{X}|$ and so follows

$$\begin{aligned} H(Z|Y) &\leq H(Z) - (\log|\mathcal{X}| - H(X)) \\ H(Z) - H(Z|Y) &\geq N \cdot D. \end{aligned}$$

Equality holds when $H(Y) = \log|\mathcal{X}|$. For a given cipher it is difficult to determine $H(Y)$. Just as for the case of channel coding Shannon attacked this difficulty by analyzing the ensemble of ciphers, i.e. the so called *random cipher*.

38.3 Random cipher

For every message source X we know that for sufficiently large message length N the set $\mathcal{X}$ will contain a subset $\mathcal{A}$ of size $A = |\mathcal{A}| \approx 2^{H(X)}$ containing approximately equally likely messages that are almost always chosen by the source, (AEP, convergence of large numbers).

$$\begin{aligned} &\Pr\{X = x\} \approx \frac{1}{A}, \quad \text{for } x \in \mathcal{A}, \\ &\Pr\{X \in \mathcal{A}\} \approx 1. \end{aligned}$$

Suppose now that the $|\mathcal{Z}|$ keys are chosen randomly and that to every plaintext,key pair a cryptogram is assigned randomly from the $|\mathcal{Y}| = |\mathcal{X}|$ possible cryptograms.

$$\begin{aligned} \Pr\{Z = z\} &= \frac{1}{|\mathcal{Z}|}, \\ \Pr\{Y = y|X = x, Z = z\} &= \frac{1}{|\mathcal{X}|} \end{aligned}$$

Then we have

$$\begin{aligned} \Pr\{Y = y\} &= \sum_{\substack{x\in\mathcal{X} \\ z\in\mathcal{Z}}} \Pr\{X = x\}\Pr\{Z = z\} \cdot \Pr\{Y = y|X = x, Z = z\} \\ &\approx \sum_{\substack{x\in\mathcal{A} \\ z\in\mathcal{Z}}} \frac{1}{A}\frac{1}{|\mathcal{Z}|}\frac{1}{|\mathcal{X}|} \approx \frac{1}{|\mathcal{X}|} \end{aligned}$$

And so $H(Y) \approx \log|\mathcal{X}|$.

If $A \cdot |\mathcal{Z}| \gg |\mathcal{X}|$ than it will be true for a "typical" random cipher that $H(Y) \approx \log|\mathcal{X}|$ and so we have for this cipher that

$$H(Z|Y) \approx H(Z) - N \cdot D.$$

Note: If $A \cdot |\mathcal{Z}| < |\mathcal{X}|$ than it is not possible that a cipher can produce all $|\mathcal{X}|$ cryptograms and thus $H(Y)$ will be essentially smaller than $\log|\mathcal{X}|$.

39 Discussions

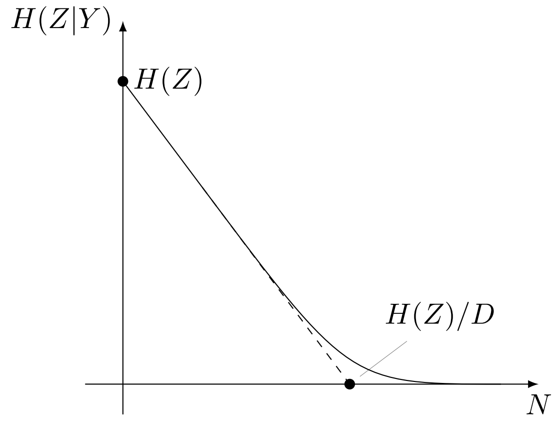
39.1 An approximation of N_0

Let $H(X) = H(X)/N \leq 1$ be the per letter entropy of a binary source, than we see that the requirement $A \cdot |\mathcal{Z}| \gg |\mathcal{X}|$ is equal to

$$|\mathcal{Z}| \gg |\mathcal{X}|/A = 2^N/2^{NH(X)} = 2^{N(1-H(X))}.$$

Thus $A \cdot |\mathcal{Z}| \gg |\mathcal{X}|$ will hold only up to a certain length N.

An example of the decrease in key equivocation.



We shall approximate the *unicity distance* by $N_0 = \frac{H(Z)}{D}$.

Although this result holds for a random cipher it appears that the approximation for the complex real world message sources and ciphers fits well with the observed (measured) strength of these ciphers. So the unicity distance is an often used indicator of quality of a cipher.

For languages as English and Dutch it seems that the redundancy D is about 3/4. (For a 26 letter alphabet and an entropy of about 1.2 bit per letter text can be reduced to one quarter of the original length.) When the messages, the cryptograms and the keys all use the same alphabet, $L_X = L_Y = L_Z$, and the key is K symbols long and both the plaintext and the cryptogram are N symbols long, than after about $N \approx \frac{4}{3}K$ symbols the key can be found (in principle).

For a permutation cipher with 26! keys and a 26 symbols alphabet the key can be found after about $\frac{4}{3}\log_{26} 26! \approx 25$ letters.

Special case $D = 0$. The unicity distance is $N_0 = \infty$. This is correct because the message source produces completely random messages. So a simple cipher, with $|\mathcal{Z}| \ll |\mathcal{X}|$, gives sufficient protection. Namely, there are always $|\mathcal{Z}|$ possible decryptions, and if $|\mathcal{Z}|$ is small but sufficiently large this gives enough protection, even though the cipher is not perfect. (There is information over the plaintext in the cryptogram, but not enough to break the cipher).

A 56 bit key for example gives $2^{56} \approx 7 \cdot 10^{16}$ possible decryptions and the probability of correct decoding for the cryptanalyst is $(\pm 10^{-16})$.

39.2 Improving N_0

- *Compression:* Data compression reduces the redundancy in the plaintext because it reduces the plaintext length. This improves the unicity distance accordingly.
- *Confusion:* Adding properly chosen symbols to the plaintext, e.g. the letter 'X' that does not change the meaning of the plaintext, increase the entropy of the text. This also reduces the redundancy.

Decreasing the redundancy increases the unicity distance. Between the two options data compression is to be preferred because it also reduces the transmission time. However, the complexity of a data compression scheme is often too high, see Quality criterion 3. Also, the actual performance of the compression depends on knowledge of the source statistics, c.f. Huffman codes, and this might not be known when the cipher is designed. Universal compression is often too complex. (And the people using ciphers don't trust anything not even a data compression algorithm.)

Confusion expands the plaintext and ciphertext and this is not acceptable according to quality criterion 5.

40 Practical security

40.1 Computational security

Practical ciphers use keys that are *much* shorter than the plaintexts. As said before data compression is not really accepted in practice to create a secure system out of simple ciphers.

Shannon postulated that every cipher has a certain *work characteristic* $W(N)$. This describes the computational complexity of the methods needed to determine the key given N symbols of the cryptogram. $W(N)$ and especially $W(\infty)$ unfortunately cannot be determined for useful ciphers. Shannon gave two principles that will be useful in the design of practical ciphers, namely *diffusion* and *confusion.*

40.2 Diffusion and Confusion

Diffusion: With diffusion one names the property that a single letter of the plaintext influences many letters in the cryptogram. The aim is to hide the probabilistic structure of the message source. It is also preferable that a key symbols influence is spread out over many cryptogram letters in order to make it more difficult to break the cryptogram part by part.

Confusion: This means an encryption that complicates the simple statistical relation between cryptogram and plaintext. An example is the addition of dummy letters as described above.

One often uses a product-cipher to implement these principles. So a series of successive simple ciphers, each contributing to diffusion and/or confusion, are used. These are often combinations of transposition and substitution ciphers. A transposition cipher confuses higher order statistics in the plaintext, while the single letter statistics remain unchanged. A substitution cipher also does not change the single letter statistics. If however the substitution is applied to a large block of letters then these blocks don't occur very often and the statistics are harder to determine.

A good example of a cipher designed with these principles is the *Data Encryption Standard* cipher. In this cipher simple but well designed transpositions and substitutions on blocks of four bits are used. These bits however are mixed in sixteen rounds where the actual transpositions and substitutions are determined by different combinations of 48 bits of the 56 bit key.

Even though DES has a short (and for current standards too short) key it was a well designed and secure cipher for its purpose.

41 Summary

- Perfect secrecy exists but demands keys that are at least as long as the plaintext that must be protected.
- The unicity distance is a useful estimation of the length of the plaintext that will be protected by a key of a given length.
- Redundancy reduction improves the quality of a cipher.
- The principles of diffusion and confusion are useful in the design of practical ciphers where $|\mathcal{Z}| \ll |\mathcal{X}|$.