Imperial College
London

COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Information and Coding Theory

Author:
Your Name (CID: your college-id number)

Date: December 10, 2018

1 Introduction

Coding is a rule for replacing one message by another message. The second message may or may not use the same alphabet as the first.

There are three major reasons for coding a message:

Economy: In many situations it is necessary or desirable to use an alphabet smaller than those that occur in natural languages.

Reliability: Messages may be altered by ?noise? in the process of transmission. Thus there is a need for codes that allow for Error Correction.

Security: Some messages are sent with the requirement that only the right person can understand them.

1.1 Definitions

Alphabet

An alphabet is a finite set S, we shall refer to the members of S as symbols.

Message

A message in the alphabet S is a finite sequence of members of S:

$$x_1 x_2 ... x_n \ \ (x_i \in S, 1 \leq i \leq n),$$

where n is the length of the message.

The set of all strings of length n is denoted by S^n.

The set of all strings in S is denoted by S^*:

$$S^* = S^0 \cup S^1 \cup S^2 \cup ...$$

Codeword

Let S and T be alphabets. A code c for S using T is an injective function $c : S \to T^*$. Meaning c does not assign the same codeword to two different symbols. In other words, if $c(s) = c(s')$ then $s = s'$.

For each symbol $s \in S$ the string $c(s) \in T^*$ is called the codeword for s.

The set of all codewords $C = \{c(s) \mid s \in S\}$

Concatenation

If $c : S \to T^*$ is a code, we extend c to S^* as follows. Given a string $x_1 x_2 ... x_n$ in S^*, define $c(x_1 x_2 ... x_n) = c(x_1)c(x_2)...c(x_n)$. This process is known as concatenation.

Uniquely decodable

The code $c : S \to T^*$ is uniquely decodable if the extended function $c : S^* \to T^*$ is an

injection. This means that any string in T^* corresponds to at most one message in S^*.

1.2 A Recap to Probability

A probability space $(\Omega, \boldsymbol{p})$ is a mathematical construct that models a real-world process consisting of states that occur randomly. Where the sample space, Ω, is the set of possible outcomes. And probability distribution, $\boldsymbol{p}$, is a function mapping events to probability.

If we enumerate the elements in Ω in some arbitrary way as $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$, then we can also represented $\boldsymbol{p}$ by a (row) vector in $\mathbb{R}^n$

$$\boldsymbol{p} = \begin{bmatrix} \boldsymbol{p}(\omega_1) & \boldsymbol{p}(\omega_2) & \boldsymbol{p}(\omega_3) & ... & \boldsymbol{p}(\omega_n) \end{bmatrix}$$

Given two probability spaces $(\Omega_1, \boldsymbol{p_1})$ and $(\Omega_2, \boldsymbol{p_2})$. The probability distribution $\boldsymbol{p}$ for the cartesian product $\Omega_1 \times \Omega_2$ can be represented as a tensor product

$$\boldsymbol{p} = \boldsymbol{p_1} \otimes \boldsymbol{p_2}$$

or in other words,

$$\boldsymbol{p}(\langle \omega_1, \omega_2 \rangle) = \boldsymbol{p_1}(\omega_1)\boldsymbol{p_2}(\omega_2)$$

However the reverse may not be true, not all probability distribution of $\Omega_1 \times \Omega_2$ can be represented as a tensor product.

2 Prefix-free Code

A coding function $c : S \to T^*$ replaces the original stream of symbols belonging to the alphabet S by an encoded stream of symbols belonging to the alphabet T .

2.1 Definitions

Prefix-free

We say that a code $c : S \to T^*$ is prefix-free (or simply PF) if there is no pair of codewords $q = c(s), q' = c(s')$ such that $q' = qr$, for some nonempty word $r \in T^*$.

Parameters of Code

Given a code $c : S \to T^*$, n_i is the number of $s \in S$ such that $c(s)$ is in T^i. If M is the maximum length of a codeword, we refer to the numbers $n_1, n_2, ..., n_M$ as the parameters of c.

2.2 Representing Codes by Trees

There is a useful way of representing a code by means of a tree. It works for codes in an alphabet of any size b, but for the purposes of illustration we shall focus on the binary alphabet with b = 2.

Codewords are represented by the corresponding nodes of the binary tree. A codeword q is a prefix of another codeword q' if the unique path from q' to the root of the tree passes through q.

If the code C is PF it follows that, for any codeword q, none of the descendants of q can be a codeword.

Represent the code $C = \{0, 10, 110, 111\}$ by means of a tree.

Solution The tree is shown in Figure 2.2.

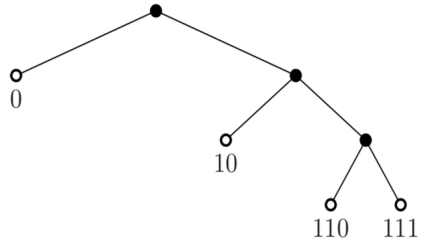


Figure 2.2 A set of codewords represented as the leaves on a tree

2.3 The Kraft-McMillan number

The Kraft-McMillan number associated with the parameters $n_1, n_2, ..., n_M$, and the base b, is defined to be

$$K = \sum_{i=1}^{M} \frac{n_i}{b^i}$$

We shall prove two important results about the existence of b-ary codes:

1. If $K \leq 1$ for the parameters $n_1, n_2, ..., n_M$, then a PF b-ary code with these parameters exists.

2. If a uniquely decodable b-ary code exists, with a given set of parameters, then its Kraft-McMillan number satisfies $K \leq 1$.

2.3.1 $K \leq 1$ implies existence of PF code

$$K = \sum_{i=1}^{M} \frac{n_i}{b^i} = \frac{n_1}{b} + \frac{n_2}{b^2} + \frac{n_3}{b^3} + ... + \frac{n_M}{b^M} \leq 1$$

Since each term is non-negative, each partial sum is also less than equal to 1.

$$\frac{n_1}{b} \leq 1, \quad \frac{n_1}{b} + \frac{n_2}{b^2} \leq 1, \quad \frac{n_1}{b} + \frac{n_2}{b^2} + \frac{n_3}{b^3} \leq 1...$$
$$n_1 \leq b, \quad n_2 \leq b(b - n_1), \quad n_3 \leq b(b^2 - n_1 b - n_2)...$$
$$n_i \leq b(b^{i-1} - n_1 b^{i-2} - ... - n_{i-1})$$

The inequalities mean it is possible to choose n_1 different codewords of length 1.

There remain $b - n_1$ words of length 1, each of which gives rise to b words of length 2, and the PF condition means we can choose n_2 codewords from these $b(b - n1)$ words.

The number of unused codewords of length $i - 1$ is

$$f_{i-1} = b^{i-1} - n_1 b^{i-2} - n_2 b^{i-3} - ... - n_{i-1}$$

Therefore $n_i \leq b f_{i-1}$, meaning it is possible to choose n_i words of length i as codewords without violating the PF condition.

2.3.2 Unique decodability implies $K \leq 1$

Let $c : S \to T^*$ be a code with parameters $n_1, n_2, ..., n_M$.

Since M is the length of the longest codeword for a single symbol, the maximum length of the code for a string of r symbols is rM.

Define a generating function for the sequence $q_r(1)$, $q_r(2)$, ... $q_r(rM)$, as follows:

$$Q_r(x) = q_r(1)x + q_r(2)x^2 + ... + q_r(rM)x^{rM}$$

Putting $x = \frac{1}{b}$ in the definition of $Q_r(x)$:

$$Q_r\left(\frac{1}{b}\right) = \frac{q_r(1)}{b} + \frac{q_r(2)}{b^2} + ... + \frac{q_r(rM)}{b^{rM}}$$

Unique decodability implies that $q_r(i)$ cannot exceed b^i, so each term in $Q_r(\frac{1}{b}) \leq 1$

There is at most rM terms, so $Q_r(\frac{1}{b}) < rM$

From $q_1(i) = n_i$:

$$K = \frac{n_1}{b} + \frac{n_2}{b^2} + \frac{n_3}{b^3} + ... + \frac{n_M}{b^M} = Q_1\left(\frac{1}{b}\right)$$

From Counting Principle (Theorem 2):

$$K^r = \left[Q_1\left(\frac{1}{b}\right)\right]^r = Q_r\left(\frac{1}{b}\right)$$

From $Q_r(\frac{1}{b}) < rM$:

$$K^r < rM$$
$$\frac{K^r}{r} < M$$

This means if a uniquely decodable b-ary code exists, $\frac{K^r}{r}$ must be less than a constant for all r

Suppose $K > 1$, for example $K = 1 + h$ for some $h > 0$. Then from binomial theorem:

$$K^r = (1+h)^r = 1 + rh + \frac{r(r-1)h^2}{2} + ...$$
$$\therefore K^r > \frac{r(r-1)h^2}{2}$$
$$\therefore \frac{K^r}{r} > \frac{(r-1)h^2}{2}$$

This contradicts with $\frac{K^r}{r}$ must be less than a constant for all r. Hence $K \leq 1$

2.3.3 Unique decodability implies existence of PF code

Existence of a UD code with given parameters implies the existence of a PF code with the same parameters.

This is obvious when we combine the two theorems:

$$\exists UD \rightarrow K \leq 1 \rightarrow \exists PF$$

2.4 Theorems

Theorem 1

If a code $c : S \rightarrow T^*$ is prefix-free, then it is uniquely decodable.

Proof

Suppose that $x_1x_2...x_m$ and $y_1y_2...y_n$ are strings in S^* such that their codes $c(x_1x_2...x_m)$ and $c(y_1y_2...y_n)$ are the same. That is $c(x_1)c(x_2)...c(x_m) = c(y_1)c(y_2)...c(y_n)$.

Since these strings are the same, their initial parts are the same. If $c(x_1) \neq c(y_1)$ then one is a prefix of the other, contradiction.

Hence $c(x_1) = c(y_1)$, and since c is an injection, $x_1 = y_1$. Repeating the same argument, it follows that $x_2 = y_2$, and so on.

Hence $x_1x_2...x_m = y_1y_2...y_n$, so c is UD

Note

A code that is UD need not be PF. For example, the code $s_1 \rightarrow 1, s_2 \rightarrow 10, s_3 \rightarrow 100$ is

not PF but UD.

Theorem 2 (Counting Principle)

Let $c : S \rightarrow T^*$be a code such that, for all $s \in S$, the length of $c(s)$ is not greater than M.

Given $r \geq 1$, let $q_r(i)$ be the number of strings of length r that are encoded by strings of length $i(1 \leq i \leq rM)$.

Let $Q_r(x)$ be the generating function $Q_r(x) = q_r(1)x + q_r(2)x^2 + ... + q_r(rM)x^{rM}$.

Then for all $r \geq 1$, $Q_r(x) = [Q_1(x)]^r$

3 Economical coding

3.1 Sources

Roughly speaking, a 'source' is a means of producing messages. A source emits a stream of symbols, denoted by

$$\xi_1\xi_2\xi_3...$$

Here each ξ_k is a variable that can take as its value any symbol belonging to a given alphabet.

3.2 Definitions

Source with Probability Distribution *p*

Let $S = \{s_1, s_2, ..., s_m\}$ be an alphabet. A probability distribution on S is a set of real numbers $p_1, p_2, ..., p_m$ such that

$$\sum_{i=0}^{m} p_i = 1 \qquad 0 \leq p_i \leq i \;\; (i = 1, 2, 3, ..., m)$$

The source $(S, \boldsymbol{p})$ emits a stream $\xi_1\xi_2\xi_3...$, where the value of each ξ_k is a member of the alphabet S and, for all k, the probability that ξ_k takes a particular value s_i is given by

$$Pr(\xi_k = s_i) = p_i$$

Memoryless Source

A source $(S, \boldsymbol{p})$ that emits a stream $\xi_1\xi_2\xi_3...$ is memoryless if the random variables ξ_k are independent. That is, for all k and l

$$Pr(\xi_k = s_i \wedge \xi_l = s_j) = Pr(\xi_k = s_i)Pr(\xi_l = s_j)$$

Average Word-length

Let $S = \{s_1, s_2, ..., s_m\}$, and a code $c : S \to T^*$. Let y_i be the length of the codeword $c(s_i)$. The average word-length for the source $(S, \boldsymbol{p})$

$$L = p_1 y_1 + p_2 y_2 + ... + p_m y_m$$

Optimal Code

Given a source $(S, \boldsymbol{p})$ and an alphabet T, a UD code $c : S \to T^*$ is optimal if there is no such code with smaller average word-length.

The requirement that the code be UD means that the Kraft-McMillan number, $K \leq 1$.

$$K = \sum_{i=1}^{M} \frac{n_i}{b^i}$$

In this notation, n_i is the number of symbols s_j such that $y_j = i$

The term $\frac{n_i}{b^i}$ in K can be represented as the sum of n_i $\frac{1}{b^i}$ terms.

It follows that if $S = \{s_1, s_2, s_3, ..., s_m\}$, then K can be written as the sum of all the terms $\frac{1}{b^{y_j}}$

$$K = \sum_{i=1}^{m} \frac{1}{b^{y_i}}$$

This leads to the **Optimisation Problem**: given b and $p_1, p_2, ..., p_m$ find positive integers $y_1, y_2, ..., y_m$ that

1) Minimize $p_1 y_1 + p_2 y_2 + ... + p_m y_m$

2) Satisfies $\frac{1}{b^{y_1}} + \frac{1}{b^{y_2}} + \frac{1}{b^{y_3}} + ... + \frac{1}{b^{y_m}} \leq 1$

Entropy

The entropy to base b of a probability distribution $\boldsymbol{p} = [p_1, p_2, ..., p_m]$ is

$$H_b(\boldsymbol{p}) = H_b(p_1, p_2, p_3, ..., p_m) = \sum_{i=1}^{m} p_i \log_b \left(\frac{1}{p_i} \right)$$

$$H_b(\boldsymbol{p}) = H_b(p_1, p_2, p_3, ..., p_m) = -\sum_{i=1}^{m} p_i \log_b (p_i)$$

In memoryless source, we often speak of the entropy of $\boldsymbol{p}$ as the entropy of the source.

We can change the base simply via

$$H_a(\boldsymbol{p}) = \log_a(b) H_b(\boldsymbol{p})$$

3.3 Entropy, uncertainty, and information

Information resolves or allows to resolve uncertainty. The larger the resolved uncertainty is the more information we obtain.

Entropy is a measure of the uncertainty about the identity of a symbol that is taken from a set according to the probability distribution $\boldsymbol{p}$.

Suppose a memoryless source emits a steam of symbols $\xi_1\xi_2\xi_3$..., where each ξ_k has the distribution $\boldsymbol{p}$. The entropy is the sum of terms $p_i \log\left(\frac{1}{p_i}\right)$.

We can interpret $\log\left(\frac{1}{p_i}\right)$ as the amount of information provided by knowing that $\xi_k = s_i$

Then the entropy is the average amount of information provided by knowing the value of ξ_k

3.3.1 Maximum Entropy

The entropy (uncertainty) of a distribution $\boldsymbol{p}$ on m symbols is at most $\log_b m$. The maximum value occurs if and only if all the symbols are equally probable.

Proof

From Theorem 1 (Comparison Theorem), take $q_i = \frac{1}{m}$:

$$\begin{aligned} H_b(\boldsymbol{p}) &\leq \sum_{i=1}^{m} p_i \log_b m \\ &= (\log_b m) \sum_{i=1}^{m} p_i \\ &= \log_b m \end{aligned}$$

with equality if and only if $q_i = p_i = \frac{1}{m}$

3.4 Shannon-Fano (SF) Rule

From Theorem 2 (Fundamental Theorem), we know that

$$L \geq H_b(\boldsymbol{p})$$

The obvious question is: how close to the lower bound $H_b(\boldsymbol{p})$ can L be?

For L to be equal to $H_b(\boldsymbol{p})$, the equality holds only when:

1) $\log_b(K) = 0$, so K must equal to 1.

2) $p_i = q_i = \frac{1}{Kb^{y_i}}$ for all i.

In other words, $b^{y_i} = \frac{1}{p_i}$ $(K = 1)$

This suggests that we can try to construct a code with average word-length close to $H_b(\boldsymbol{p})$ by choosing b^{y_i} to be as close as possible to $\frac{1}{p_i}$.

This is known as Shannon-Fano Rule, where we choose the length of codeword y_i to be the least positive integer such that $b^{y_i} \geq \frac{1}{p_i}$

Since the value $K \leq 1$, a PF code with these parameters does exist.

3.5 Huffman's Rule

For many purposes the Shannon-Fano rule produces a satisfactory code, but in general it does not give an optimal code. On the other hand, Huffman's Rule is guaranteed to give an optimal code.

3.5.1 Motivation

To understand Huffman's Rule, it is important to understand the properties of optimal PF code.

An optimal PF code $c : S \to B^*$ for a source $(S, \boldsymbol{p})$ has two properties:

1) if the codeword $c(s')$ is longer than $c(s)$ then $p_s \geq p_{s'}$

Suppose $c(s) = w$ and $c(s') = w'$. Define a new code c^* such that $c^*(s) = w'$ and $c^*(s') = w$
The average word length of c and c^* are $L(c)$ and $L(c^*)$ respectively.

$$\begin{aligned} L(c^*) - L(c) &= (p_s|w'| + p_{s'}|w|) - (p_s|w| + p_{s'}|w'|) \\ &= (p_s - p_{s'})(|w'| - |w|) \end{aligned}$$

Since c is optimal, $L(c^*) - L(c)$ must be positive. Hence if $|w'| > |w|$ then $p_s \geq p_{s'}$.

2) among the codewords of maximum length there are two of the form $x0$ and $x1$, for some $x \in B^*$.

If no two words of maximum length have the form stated, then deleting the last bit from all codewords of maximum length would produce a better code that still has the PF property.

3.5.2 Construction

H1: Given a source $(S, \boldsymbol{p})$, let s' and s'' be two symbols with the smallest probabilities. Construct a new source $(S^*, \boldsymbol{p}^*)$ by replacing s' and s'' by a single symbol s^*, with probability $\boldsymbol{p}^*_{s^*} = \boldsymbol{p}_{s'} + \boldsymbol{p}_{s''}$. All other symbols have unchanged probabilities

H2: If we are given a PF binary code h^* for $(S^*, \boldsymbol{p}^*)$, with $h^*(s^*) = w$, then a PF binary code h for $(S, \boldsymbol{p})$ is defined by the rules $h(s') = w0$, $h(s'') = w1$, and $h(u) = h^*(u)$ for all $u \neq s', s''$.

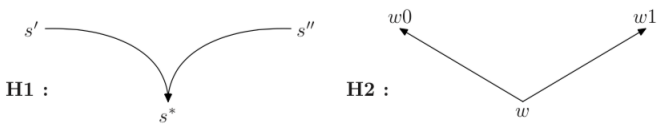


Figure 3.3 The two parts of Huffman's rule

3.5.3 Example

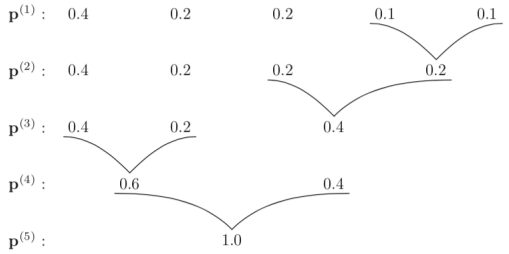


Figure 3.4 Application of rule **H1**

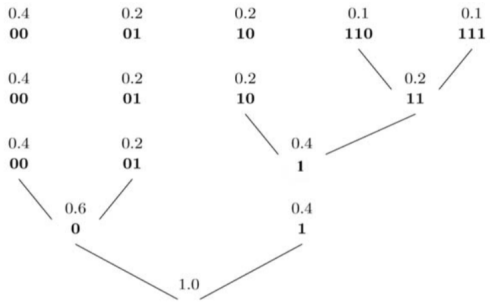


Figure 3.5 Application of rule **H2**

3.5.4 Optimality of Huffman codes

To proof optimality of Huffman codes, we first have to quantify the increase of L in the construction of **H2**.

Let h and h^* be defined as in the construction **H2**. Then the average word lengths of h and h^* satisfy

$$L(h) = L(h^*) + \boldsymbol{p}^*_{s^*}$$

Proof

Suppose that the length of $h^*(s^*)$ is β. Then in the code h the symbols s' and s'' are assigned codewords of length $\beta + 1$.

$$\begin{aligned} L(h) - L(h^*) &= (\boldsymbol{p}_{s'} + \boldsymbol{p}_{s''})(\beta + 1) + \boldsymbol{p}^*_{s^*}\beta \\ L(h) - L(h^*) &= \boldsymbol{p}^*_{s^*}(\beta + 1) + \boldsymbol{p}^*_{s^*}\beta \\ L(h) - L(h^*) &= \boldsymbol{p}^*_{s^*} \\ L(h) &= L(h^*) + \boldsymbol{p}^*_{s^*} \end{aligned}$$

Now we try to proof that If h^* is optimal for $(S^*, \boldsymbol{p}^*)$ then h is optimal for $(S, \boldsymbol{p})$

Proof

We can prove the contrapositive, if h is not optimal for $(S, \boldsymbol{p})$ then h^* is not optimal for $(S^*, \boldsymbol{p}^*)$.

Assume that h is not optimal for $(S, \boldsymbol{p})$, but c is optimal for $(S, \boldsymbol{p})$.

From (2) of **Motivation**, if c is optimal then c assigns to some t', t'' codewords $w0$, $w1$ of maximum length.

Take another disjoint pair of symbols s', s'' involved in the construction of h. And define c as

$$c(t') = w0 \quad c(t'') = w1 \quad c(s') = y \quad c(s'') = z$$

Define a new code c^* for $(S^*, \boldsymbol{p}^*)$, and define c^* as

$$c^*(t') = y \quad c^*(t'') = z \quad c^*(s^*) = w$$

Hence,

$$\begin{aligned}
L(c) - L(c^*) &= \boldsymbol{p}_{t'}|w0| + \boldsymbol{p}_{t''}|w1| + \boldsymbol{p}_{s'}|y| + \boldsymbol{p}_{s''}|z| - (\boldsymbol{p}^*_{t'}|y| + \boldsymbol{p}^*_{t''}|z| + \boldsymbol{p}^*_{s^*}|w|) \\
&\quad \because \boldsymbol{p}^*_{t'} = \boldsymbol{p}_{t'} \quad \boldsymbol{p}^*_{t''} = \boldsymbol{p}_{t''} \\
L(c) - L(c^*) &= \boldsymbol{p}_{t'}|w0| + \boldsymbol{p}_{t''}|w0| + \boldsymbol{p}_{s'}|y| + \boldsymbol{p}_{s''}|z| - [\boldsymbol{p}_{t'}|y| + \boldsymbol{p}_{t''}|z| + \boldsymbol{p}^*_{s^*}(|w0| - 1)] \\
&\quad \because \boldsymbol{p}^*_{s^*} = \boldsymbol{p}_{s'} + \boldsymbol{p}_{s''} \\
L(c) - L(c^*) &= \boldsymbol{p}_{t'}|w0| + \boldsymbol{p}_{t''}|w0| + \boldsymbol{p}_{s'}|y| + \boldsymbol{p}_{s''}|z| - \boldsymbol{p}_{t'}|y| - \boldsymbol{p}_{t''}|z| - \boldsymbol{p}_{s'}|w0| - \boldsymbol{p}_{s''}|w0| + \boldsymbol{p}^*_{s^*} \\
L(c) - L(c^*) &= (\boldsymbol{p}_{t'} - \boldsymbol{p}_{s'})(|w0| - |y|) + (\boldsymbol{p}_{t''} - \boldsymbol{p}_{s''})(|w0| - |z|) + \boldsymbol{p}^*_{s^*}
\end{aligned}$$

Since $|w0|$ is a codeword with maximum length, $|w0| \geq |y|$ and $|w0| \geq |z|$.

Since s' and s'' are symbols with the smallest probability, $\boldsymbol{p}_{t'} \geq \boldsymbol{p}_{s'}$ and $\boldsymbol{p}_{t''} \geq \boldsymbol{p}_{s''}$

Therefore we have $L(c) - L(c^*) \geq \boldsymbol{p}^*_{s^*}$.

From above, we know the average word lengths of h and h^* satisfy $L(h) = L(h^*) + \boldsymbol{p}^*_{s^*}$

And since we assumed c to be the optimum code, $L(c) < L(h)$.

$$L(c^*) \leq L(c) - \boldsymbol{p}^*_{s^*} < L(h) - \boldsymbol{p}^*_{s^*} = L(h^*)$$

And so we proved that h^* is not optimal for $(S^*, \boldsymbol{p}^*)$

3.6 Theorems

Theorem 1 (Comparison Theorem)

Given probability distributions $\boldsymbol{p} = (p_1, p_2, ..., p_m)$ and $\boldsymbol{q} = (q_1, q_2, ..., q_m)$.

$$H_b(\boldsymbol{p}) = \sum_{i=1}^{m} p_i \log_b\left(\frac{1}{p_i}\right) \leq \sum_{i=1}^{m} p_i \log_b\left(\frac{1}{q_i}\right)$$

There is equality if and only if $q_i = p_i$ for all i $(1 \leq i \leq m)$.

Proof

It is sufficient to prove the result for any one value of b, we shall take $b = e$

Using the inequality $ln(x) \leq x-1$ for all $x>0$, where there is equality iff $x=1$:

$$\ln\left(\frac{q_i}{p_i}\right) \leq \frac{q_i}{p_i} - 1$$

with equality if and only if $q_i = p_i$

$$\begin{aligned}
\sum_{i=1}^{m} p_i \ln\left(\frac{1}{p_i}\right) - \sum_{i=1}^{m} p_i \ln\left(\frac{1}{q_i}\right) &= \sum_{i=1}^{m} p_i \left[\ln\left(\frac{1}{p_i}\right) - \ln\left(\frac{1}{q_i}\right)\right] \\
&= \sum_{i=1}^{m} p_i \ln\left(\frac{q_i}{p_i}\right) \\
&\leq \sum_{i=1}^{m} p_i \left(\frac{q_i}{p_i} - 1\right) \\
&= \sum_{i=1}^{m} q_i - \sum_{i=1}^{m} p_i \\
&= 1 - 1 = 0
\end{aligned}$$

Theorem 2 (Fundamental Theorem)

The average word-length, L of any UD b-ary code for the source $(S, \boldsymbol{p})$ satisfies

$$L \geq H_b(\boldsymbol{p})$$

Proof

Let $S = \{s_1, s_2, ..., s_m\}$. Let y_i be the length of the codeword s_i. Then the Kraft-McMillan number for the code is

$$K = \sum_{i=1}^{m} \frac{1}{b^{y_i}} \leq 1$$

From Theorem 1 (Comparison Theorem), take $q_i = \frac{1}{Kb^{y_i}}$:

$$\begin{aligned}
H_b(\boldsymbol{p}) &\leq \sum_{i=1}^{m} p_i \log_b(Kb^{y_i}) \\
&= \sum_{i=1}^{m} p_i \left[\log_b(K) + y_i\right] \\
&= \log_b(K) + \sum_{i=1}^{m} p_i y_i \\
&= \log_b(K) + L
\end{aligned}$$

Since $K \leq 1$, it means that $\log_b(K) \leq 0$. Therefore $L \geq H_b(\boldsymbol{p})$

Theorem 3 (Fundamental Theorem)

There is a PF b-ary code for a source with probability distribution $\boldsymbol{p}$ that satisfies the inequality

$$L < H_b(\boldsymbol{p}) + 1$$

Proof

Let $S = \{s_1, s_2, ..., s_m\}$. Let y_i be the length of the codeword s_i. We defined y_i using SF Rule such that a PF code with these parameters exists.

Since we used SF Rule to define each y_i, y_i is the least integer such that

$$\begin{aligned} b^{y_i} &\geq \frac{1}{p_i} \\ b^{y_i - 1} &< \frac{1}{p_i} \\ y_i &< \log_b\left(\frac{1}{p_i}\right) + 1 \\ L = \sum p_i y_i &< \sum p_i \left[\log_b\left(\frac{1}{p_i}\right) + 1\right] = H_b(\boldsymbol{p}) + 1 \end{aligned}$$

4 Data Compression

From the Optimisation Problem, we found that the entropy $H_b(\boldsymbol{p})$ is the lower bound for average word length L of a source $(S, \boldsymbol{p})$. However, this lower bound is rarely attained.

Previously we only considered single symbol encoding $c : S \to T^*$ that replaces symbols in an alphabet S by strings of symbols in an alphabet T.

In practice, splitting the stream of symbols emitted by a source into blocks, and to regard the blocks themselves as symbols enables us to approach the lower bound as much as possible

Consider a source $(S, \boldsymbol{p})$ that emits two symbols W and B with $\boldsymbol{p}_W = 0.9$ and $\boldsymbol{p}_B = 0.1$.

The entropy of $(S, \boldsymbol{p})$ is

$$H_b(\boldsymbol{p}) = 0.9 \log_2\left(\frac{1}{0.9}\right) + 0.1 \log_2\left(\frac{1}{0.1}\right) \approx 0.469$$

Consider $S \to B^*$ coding, then performing Huffman's coding:

$$W \mapsto 0 \quad B \mapsto 1$$

Therefore $L = 1$, the average word length for each letter is 1

Consider $S^2 \to B^*$ coding, calculating the probability for each block then performing Huffman's coding:

$$\begin{array}{cccc} WW & WB & BW & BB \\ 0.81 & 0.09 & 0.09 & 0.01 \end{array}.$$

$$WW \mapsto 0, \quad WB \mapsto 10, \quad BW \mapsto 110, \quad BB \mapsto 111.$$

Therefore $L = 1.29$, the average word length for each letter is $\frac{1.29}{2} = 0.645$

Consider $S^3 \to B^*$ coding, calculating the probability for each block then performing Huffman's coding:

$$\begin{array}{cccccccc} WWW & WWB & WBW & WBB & BWW & BWB & BBW & BBB \\ 0.729 & 0.081 & 0.081 & 0.009 & 0.081 & 0.009 & 0.009 & 0.001 \end{array}.$$

$$WWW \mapsto 0,\ WWB \mapsto 100,\ WBW \mapsto 101,\ WBB \mapsto 11100,$$

$$BWW \mapsto 110,\ BWB \mapsto 11100,\ BBW \mapsto 11110,\ BBB \mapsto 11111,$$

Therefore $L = 1.598$, the average word length for each letter is $\frac{1.598}{3} = 0.533$

The technique above in the example is the basis of **data compression**.

4.1 Definitions

Marginal Distributions

Let $S' = \{s'_1, s'_2, ..., s'_m\}$ and $S'' = \{s''_1, s''_2, ..., s''_n\}$. Let $Y = S' \times S''$ be a product set. Let $\boldsymbol{p}$ be a probability distribution on Y and denote the probability of $s'_i s''_j$ as p_{ij}.

$$\boldsymbol{p'}_i = \sum_{j=1}^{n} \boldsymbol{p}_{ij}$$

$$\boldsymbol{p''}_j = \sum_{i=1}^{m} \boldsymbol{p}_{ij}$$

The probability distributions $\boldsymbol{p'}$ on S' and $\boldsymbol{p''}$ on S'' are known as the marginal distributions.

The distributions $\boldsymbol{p'}$ and $\boldsymbol{p''}$ are independent if $p_{ij} = \boldsymbol{p'}\boldsymbol{p''}$.

Stationary Sources

Stationarity means that the probability that a given word (string of consecutive symbols) will appear on 'page 1' of the message is the same as the probability that it will appear on 'page 99', or any other 'page'.

A source emitting a stream $\xi_1\xi_2\xi_3...$ is stationary if, for any positive integers $l_1, l_2, ..., l_r$, the probability

$$Pr(\xi_{k+l_1} = x_1, \xi_{k+l_2} = x_2, ..., \xi_{k+l_r} = x_r)$$

depends only on the string $x_1x_2...x_r$ but not k

Entropy of a stationary source

The entropy H of a stationary source with probability distributions $\boldsymbol{p^r}$ is the infimum of the numbers

$$\frac{H(\boldsymbol{p^r})}{r} \quad r = 1, 2, 3, ...$$

The entropy $H(\boldsymbol{p^r})$ represents the uncertainty of the stream, per block of r symbols. Therefore the uncertainty per symbol is $\frac{H(\boldsymbol{p^r})}{r}$

Dictionary

A dictionary D based on the alphabet S is a sequence of distinct words in S^*:

$$D = d_1, \ d_2, \ d_3, ..., \ d_n$$

We say that the index of d_i is i

4.2 Stationary Sources

Suppose we have a stream $\xi_1\xi_2\xi_3...$, where the value of each ξ_k is a member of the alphabet S. We can consider the stream as blocks that takes value from the set S^r of strings of length r.

For example by taking $r = 2$, we can consider the stream as blocks of $\xi_{2k-1}\xi_{2k}$. Then we can define a probability distribution $\boldsymbol{p^2}$ on S^2 as

$$\boldsymbol{p^2}(s_is_j) = Pr(\xi_{2k-1} = s_i, \ \xi_{2k} = s_j)$$

Although the definition suggests that for all integers l. In practice, we only consider consecutive symbols, that is, the case $l_1 = 1, \ l_2 = 2, ..., l_r = r$.

Then the definition implies that for a stationary source we have probability distributions $\boldsymbol{p^r}$, defined on S^r

$$\boldsymbol{p^r}(x_1x_2...x_r) = Pr(\xi_{k+1} = x_1, \ \xi_{k+2} = x, ..., \ \xi_{k+r} = x_r)$$

A memoryless source is a very special case of a stationary source. In that case, each $\boldsymbol{p^r}$ is simply determined by $\boldsymbol{p^1}$:

$$\boldsymbol{p^r}(x_1x_2...x_r) = \boldsymbol{p^1}(x_1)\boldsymbol{p^1}(x_2)...\boldsymbol{p^1}(x_r)$$

4.3 Algorithms for Data Compression

Huffman?s rule is not sufficient for data compression. It requires the construction of codewords for all the blocks before any coding can be done, and if the required block-length is n, the set of codewords will have size 2^n. Another weakness of Huffman?s rule is that it assumes that the characteristics of the source are known in advance.

There is a need for alternative methods of coding with the following properties:

1. the codeword for a given block can be calculated in isolation, without the need to find all the codewords

2. the codewords can be updated 'on-line' (adaptive coding)

4.3.1 LZW encoding

Take alphabet $S = \{A, B, C, D, R\}$ and LZW encode the string: $X = ABRACADABRA$

We start with the dictionary $D_0 = A, B, C, D, R$

$c(A) = 1$	$D_1 = A, B, C, D, R, AB$
$c(AB) = 12$	$D_2 = A, B, C, D, R, AB, BR$
$c(ABR) = 125$	$D_3 = A, B, C, D, R, AB, BR, RA$
$c(ABRA) = 1251$	$D_4 = A, B, C, D, R, AB, BR, RA, AC$
$c(ABRAC) = 12513$	$D_5 = A, B, C, D, R, AB, BR, RA, AC, CA$
$c(ABRACA) = 125131$	$D_6 = A, B, C, D, R, AB, BR, RA, AC, CA, AD$
$c(ABRACAD) = 1251314$	$D_7 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA$
$c(ABRACADAB) = 12513146$	$D_8 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA, ABR$
$c(ABRACADABRA) = 125131468$	$D_9 = A, B, C, D, R, AB, BR, RA, AC, CA, AD, DA, ABR$

4.3.2 LZW decoding

Consider the initial dictionary $D_0 = I, M, P, S$ and LZW decode: 214468331

$2 \mapsto M$	$D_1 = I, M, P, S, MI$
$21 \mapsto MI$	$D_2 = I, M, P, S, MI, IS$
$214 \mapsto MIS$	$D_3 = I, M, P, S, MI, IS, SS$
$2144 \mapsto MISS$	$D_4 = I, M, P, S, MI, IS, SS, SI$
$21446 \mapsto MISSIS$	$D_5 = I, M, P, S, MI, IS, SS, SI, ISS$
$214468 \mapsto MISSISSI$	$D_6 = I, M, P, S, MI, IS, SS, SI, ISS, SIP$
$2144683 \mapsto MISSISSIP$	$D_7 = I, M, P, S, MI, IS, SS, SI, ISS, SIP, PP$
$21446833 \mapsto MISSISSIPP$	$D_8 = I, M, P, S, MI, IS, SS, SI, ISS, SIP, PP, PI$
$214468331 \mapsto MISSISSIPPI$	$D_9 = I, M, P, S, MI, IS, SS, SI, ISS, SIP, PP, PI$

4.3.3 UD of LWZ code

An LZW code is uniquely decodable.

Proof

Suppose that the partial code $c_1c_2...c_{k?1}$ has been successfully decoded as $x_1x_2...x_i$. Suppose also that the dictionary $D_{k?1}$ has been constructed by adding $k?1$ strings to D_0.

We must give the rules for decoding c_k and constructing $D_k = (D_{k?1}, d_{m+k})$.

The encoding rules imply that c_k is the index of a string $s_u...s_v$ in $D_{k?1}$. Thus c_k is decoded by putting $x_{i+1} = s_u, ..., x_j = s_v$.

In order to construct the new dictionary entry d_{m+k}, the encoder must use the value of c_{k+1}.

If $c_{k+1} = r$ and d_r is in $D_{k?1}$, ,say $d_r = s_a...$, then the new dictionary entry is $d_{m+k} = s_u...s_vs_a$.

The other possibility is that c_{k+1} is exactly the new entry in D_k. Then the new dictionary entry is $d_{m+k} = s_u...s_vs_u$.

This completes the emulation of Step k, and the proof is complete.

4.4 Theorems

Theorem 1

Let $S' = \{s'_1, s'_2, ..., s'_m\}$ and $S'' = \{s''_1, s''_2, ..., s''_n\}$. Let $Y = S' \times S''$ be a product set. Let $\boldsymbol{p}$ be a probability distribution on Y. And $\boldsymbol{p}'$ on S' and $\boldsymbol{p}''$ on S'' are marginal distributions. Then the entropies of $\boldsymbol{p}$, $\boldsymbol{p}'$, $\boldsymbol{p}''$ satisfy

$$H(\boldsymbol{p}) \leq H(\boldsymbol{p}') + H(\boldsymbol{p}'')$$

And equality holds iff $\boldsymbol{p}'$ and $\boldsymbol{p}'$ are independent.

Proof

By definition

$$H(\boldsymbol{p}') + H(\boldsymbol{p}'') = \sum_i \boldsymbol{p}'_i \log\left(\frac{1}{\boldsymbol{p}'_i}\right) + \sum_j \boldsymbol{p}''_j \log\left(\frac{1}{\boldsymbol{p}''_j}\right)$$

Since $\boldsymbol{p}'_i = \sum_j p_{ij}$ and $\boldsymbol{p}''_j = \sum_i p_{ij}$

$$H(\boldsymbol{p}') + H(\boldsymbol{p}'') = \sum_i \sum_j \boldsymbol{p}_{ij} \log\left(\frac{1}{\boldsymbol{p}'_i \boldsymbol{p}''_j}\right)$$

Defining another probability distribution $\boldsymbol{q}$ on $S' \times S''$ such that $q_{ij} = \boldsymbol{p'}_i \boldsymbol{p''}_j$. Then apply Comparison Theorem

$$H(\boldsymbol{p}) \leq \sum_i \sum_j \boldsymbol{p}_{ij} \log\left(\frac{1}{q_{ij}}\right) = \sum_i \sum_j \boldsymbol{p}_{ij} \log\left(\frac{1}{\boldsymbol{p'}_i \boldsymbol{p''}_j}\right)$$

Theorem 2

If a stationary source is memoryless, its entropy H is equal to $H(\boldsymbol{p}^1)$.

Proof

If the source is memoryless then by the theorem 1, $H(\boldsymbol{p}^2) = 2H(\boldsymbol{p}^1)$.

Using induction, we can proof that $H(\boldsymbol{p}^r) = rH(\boldsymbol{p}^1)$

Therefore all the terms $\frac{H(\boldsymbol{p}^r)}{r}$ are equal to $H(\boldsymbol{p}^1)$

Theorem 3

Suppose n is a multiple of r then

$$\frac{H(\boldsymbol{p}^n)}{n} \leq \frac{H(\boldsymbol{p}^r)}{r}$$

Proof

Assume $n = 2r$.

Then consider the distribution $\boldsymbol{p}^r$ on $\xi_{k+1}\xi_{k+2}\xi_{k+3}...\xi_{k+r}$ and $\xi_{k+r+1}\xi_{k+r+2}\xi_{k+r+3}...\xi_{k+2r}$ and also the distribution $\boldsymbol{p}^{2r}$ on $\xi_{k+1}\xi_{k+2}\xi_{k+3}...\xi_{k+2r}$

Then from theorem 1

$$\frac{H(\boldsymbol{p}^{2r})}{2r} \leq \frac{H(\boldsymbol{p}^r) + H(\boldsymbol{p}^r)}{2r} = \frac{H(\boldsymbol{p}^r)}{r}$$

A more generalising argument is

$$H(\boldsymbol{p}^{qr}) \leq H(\boldsymbol{p}^{(q-1)r}) + H(\boldsymbol{p}^r)$$

Hence this can be proved by induction on q

Theorem 4

For a stationary source on S and entropy H. Given that $\epsilon > 0$ there exists $n > 0$ and a PF binary code $(S^n, \boldsymbol{p}^n)$ with L_n satisfying

$$\frac{L_n}{n} < H + \epsilon$$

Proof

Since H is the infimum of the set of $\frac{H(\boldsymbol{p}^r)}{r}$, there exists an r such that

$$\frac{H(\boldsymbol{p}^r)}{r} < H + \frac{\epsilon}{2}$$

Choose an n that is a multiple of r such that $\frac{1}{n} < \frac{\epsilon}{2}$

Using Theorem 3 in section 3, there is a PF binary code for $\boldsymbol{p}^n$ with

$$L_n < H(\boldsymbol{p}^n) + 1$$
$$\frac{L_n}{n} < \frac{H(\boldsymbol{p}^n)}{n} + \frac{1}{n}$$

From Theorem 3 and the value with pick for n

$$\frac{L_n}{n} < \frac{H(\boldsymbol{p}^r)}{r} + \frac{\epsilon}{2}$$
$$\frac{L_n}{n} < H + \epsilon$$

5 Noisy Channels

5.1 Definitions

Channel

Let I be a finite alphabet, and symbols in I are inputs to a device, referred to as a channel. The symbols that form the output of the device belong to another set J.

For each $i \in I$ and $j \in J$, we denote $Pr(j|i)$ the conditional probability that the output is j, given that the input is i.

A channel Γ with input set I and output set J is a matrix whose entries are the numbers $Pr(j|i)$:

$$\Gamma_{ij} = Pr(j|i)$$

Rows correspond to inputs and columns to outputs. If at least one of the terms Γ_{ij} with $i \neq j$ is non-zero, we say that the channel is noisy.

Conditional Entropy

The conditional entropy $H(\boldsymbol{p}|\boldsymbol{q})$ (for $\boldsymbol{q} = \boldsymbol{p}\Gamma$) is defined as

$$H(\boldsymbol{p}|\boldsymbol{q}) = H(\boldsymbol{t}) - H(\boldsymbol{q}) = H(\Gamma, \boldsymbol{p})$$

Capacity

The capacity γ of Γ is the maximum value of $f_\Gamma(\boldsymbol{p})$ taken over all probability distributions on a set of size m. That is

$$\gamma = max \;\; f_\Gamma(\boldsymbol{p}) = max \;\; (H(\boldsymbol{p}) - H(\Gamma; \boldsymbol{p}))$$

5.2 Binary Symmetric Channel (BSC)

A Binary Symmetric Channel (BSC) corresponds to a matrix of the form:

$$\Gamma_{ij} = \begin{pmatrix} 1-e & e \\ e & 1-e \end{pmatrix}$$

we refer to $e > 0$ as the bit-error probability. The probability that a symbol is transmitted correctly is $1 - e$

5.2.1 Conditional Entropy for BSC

Let Γ be a BSC with bit-error probability e, and $\boldsymbol{p}$ the source distribution $(p_0, p_1) = (p, 1-p)$ then

$$H(\Gamma; \boldsymbol{p}) = h(p) + h(e) - h(q)$$

Where $q = p(1-e) + (1-p)e$ and h is the standard entropy

$$h(x) = xlog\left(\frac{1}{x}\right) + (1-x)log\left(\frac{1}{1-x}\right)$$

Proof

Compute $t_{ij} = \boldsymbol{p}_i \Gamma_{ij}$

$$t_{00} = p(1-e) \qquad t_{01} = pe \qquad t_{10} = (1-p)e \qquad t_{11} = (1-p)(1-e)$$

This means the probability distribution $\boldsymbol{t}$ is simply the product of independent distributions $[p, 1-p]$ and $[1-e, e]$

Using Theorem 1 from Chapter 4:

$$H(\boldsymbol{t}) = h(p) + h(e)$$

Since $H(\Gamma, \boldsymbol{p}) = H(\boldsymbol{t}) - H(\boldsymbol{q})$

$$H(\Gamma, \boldsymbol{p}) = h(p) + h(e) - h(q)$$

5.2.2 Capacity of BSC

Let Γ be a BSC with bit-error probability e with $0 \leq e \leq 1$, then

$$\gamma = 1 - h(e)$$

where h is the standard entropy

Proof

By definition of capacity, we want to find the maximum of

$$H(\boldsymbol{p}) - H(\Gamma; \boldsymbol{p})$$

From above, we know that $H(\Gamma;\boldsymbol{p})$ of BSC is $h(p)+h(e)-h(q)$. Therefore, we want to find the maximum of

$$h(\boldsymbol{p})-h(p)-h(e)+h(q)=h(q)-h(e)$$

Since e is a given constant, the maximum occurs when $h(q)$ is a maximum.

From standard entropy function, we know that this maximum is 1 and occurs when $q=\frac{1}{2}$.

Therefore the capacity of BSC is $1-h(e)$

5.3 Conditional Entropy

From the viewpoint of an observer who has access to both input and output, the effect of transmitting a source through a noisy channel Γ is to increase the uncertainty.

However, we want to consider the viewpoint of the Receiver, for whom the output provides the only available information about the input.

In particular, we want to answer the question: How much information about the input is available to a Receiver who knows the output of Γ?

To answer the question, we consider the probability distribution $\boldsymbol{t}$ on $I\times J$ where

$$t_{ij}=Pr(output\ is\ j\ AND\ input\ is\ i)$$

The significant point is that, when $\boldsymbol{t}$ is given, all the other quantities can be derived from it

$$\boldsymbol{p}_i=\sum_{j\in J}t_{ij}\qquad \boldsymbol{q}_j=\sum_{i\in I}t_{ij}\qquad t_{ij}=\Gamma_{ij}p_i$$

$\boldsymbol{p}$ and $\boldsymbol{q}$ are the marginal distributions associated with $\boldsymbol{t}$.

The conditional entropy is defined to be

$$H(\boldsymbol{p}|\boldsymbol{q})=H(\boldsymbol{t})-H(\boldsymbol{q})$$

The motivation is that $H(\boldsymbol{t})$ measures the uncertainty about the input output pair (i,j), and $H(\boldsymbol{q})$ measures the uncertainty about the output j.

The difference of the two represents the uncertainty of a receiver who knows the output and is trying to determine the input.

Since $\boldsymbol{q}=\boldsymbol{p}\Gamma$, it follows that $H(\boldsymbol{p}|\boldsymbol{q})$ depends on Γ and $\boldsymbol{p}$. Therefore we can denote the the conditional entropy of $\boldsymbol{p}$ with respect to transmission through Γ as $H(\Gamma,\boldsymbol{p})$.

5.4 Capacity of a channel

Suppose we are given a channel Γ that accepts an input alphabet I of size m; in other words, the channel matrix has m rows.

Then for each probability distribution $\boldsymbol{p}$ on I we have a value of

$$f_\Gamma(\boldsymbol{p}) = H(\boldsymbol{p}) - H(\Gamma;\boldsymbol{p})$$

The capacity γ of Γ is the maximum value of $f_\Gamma(\boldsymbol{p})$ taken over all probability distributions on a set of size m. That is

$$\gamma = max\ \ f_\Gamma(\boldsymbol{p}) = max\ \ (H(\boldsymbol{p}) - H(\Gamma;\boldsymbol{p}))$$

5.4.1 Calculating capacity

The definition of channel capacity is formulated in terms of finding the maximum of

$$H(\boldsymbol{p}) - H(\Gamma;\boldsymbol{p}) \quad or \quad H(\boldsymbol{p}) - H(\boldsymbol{p}|\boldsymbol{q})$$

From definition of conditional entropy,

$$H(\boldsymbol{p}|\boldsymbol{q}) = H(\boldsymbol{t}) - H(\boldsymbol{q})$$

Also, we can define

$$H(\boldsymbol{q}|\boldsymbol{p}) = H(\boldsymbol{t}) - H(\boldsymbol{p})$$

The motivation is that $H(\boldsymbol{t})$ measures the uncertainty about the input output pair (i, j), and $H(\boldsymbol{p})$ measures the uncertainty about the input i.

The difference of the two represents the uncertainty of a sender who knows the input and is trying to determine the output.

Therefore another mathematically equivalent way to calculate capacity is

$$\gamma = max\ \ (H(\boldsymbol{q}) - H(\boldsymbol{q}|\boldsymbol{p}))$$

We shall describe an alternative way to calculate $H(\boldsymbol{q}|\boldsymbol{p})$. To do this, we denote

$$H(\boldsymbol{q}|i) = \sum_{j\in J} \Gamma_{ij} log\left(\frac{1}{\Gamma_{ij}}\right)$$

As a measure of the uncertainty about the output symbol, given that the input symbol is i.

Then we show that $H(\boldsymbol{q}|\boldsymbol{p})$ can be calculated as

$$H(\boldsymbol{q}|\boldsymbol{p}) = \sum_{i\in I} p_i H(\boldsymbol{q}|i)$$

Proof

$$\sum_{i\in I} p_i H(\boldsymbol{q}|i) + H(\boldsymbol{p}) = \sum_{i\in I} p_i H(\boldsymbol{q}|i) + \sum_{i\in I} p_i log\left(\frac{1}{p_i}\right)$$
$$\sum_{i\in I} p_i H(\boldsymbol{q}|i) + H(\boldsymbol{p}) = \sum_{i\in I} p_i \left[H(\boldsymbol{q}|i) + log\left(\frac{1}{p_i}\right)\right]$$

Using definition of $H(\boldsymbol{q}|i)$ and Theorem 1

$$\sum_{i\in I} p_i H(\boldsymbol{q}|i) + H(\boldsymbol{p}) = \sum_{i\in I} p_i \left[\sum_{j\in J} \Gamma_{ij} log\left(\frac{1}{\Gamma_{ij}}\right) + \sum_{j\in J} \Gamma_{ij} log\left(\frac{1}{p_i}\right)\right]$$
$$\sum_{i\in I} p_i H(\boldsymbol{q}|i) + H(\boldsymbol{p}) = \sum_{i\in I} p_i \left[\sum_{j\in J} \Gamma_{ij} log\left(\frac{1}{p_i\Gamma_{ij}}\right)\right]$$

Using $t_{ij} = p_i\Gamma_{ij}$

$$\sum_{i\in I} p_i H(\boldsymbol{q}|i) + H(\boldsymbol{p}) = \sum_{i\in I} p_i \left[\sum_{j\in J} \frac{t_{ij}}{p_i} log\left(\frac{1}{t_{ij}}\right)\right]$$
$$\sum_{i\in I} p_i H(\boldsymbol{q}|i) + H(\boldsymbol{p}) = \sum_{i\in I, j\in J} t_{ij} log\left(\frac{1}{t_{ij}}\right) = H(\boldsymbol{t})$$
$$\sum_{i\in I} p_i H(\boldsymbol{q}|i) = H(\boldsymbol{t}) - H(\boldsymbol{p}) = H(\boldsymbol{q}|\boldsymbol{p})$$

5.5 Theorems

Theorem 1

Each row of a channel matrix has sum equal to 1. That is for every for i

$$\sum_{j\in J} \Gamma_{ij} = 1$$

Proof

By definition

$$\sum_{j\in J} \Gamma_{ij} = \sum_{j\in J} Pr(j|i)$$

$\sum_{j\in J} Pr(j|i)$ is the total probability of all outputs, given that the input is i, is 1.

Theorem 2

We can think of the input to the channel Γ as being generated by a source $(I, \boldsymbol{p})$, the output can be regarded as a source $(J, \boldsymbol{q})$. Then

$$\boldsymbol{q} = \boldsymbol{p}\Gamma$$

Proof

For each $i \in I$ and $j \in J$, denote t_{ij} as the probability that the input to Γ is i and the output is j. It follows that

$$\boldsymbol{q}_j = \sum_{i \in I} t_{ij}$$

From the definition of conditional probability

$$\begin{aligned} t_{ij} &= Pr(output\ \ is\ \ j\ \ AND\ \ input\ \ is\ \ i) \\ t_{ij} &= Pr(output\ \ is\ \ j | input\ \ is\ \ i) \times Pr(input\ \ is\ \ i) \\ t_{ij} &= \Gamma_{ij} p_i \end{aligned}$$

Therefore,

$$\begin{aligned} \boldsymbol{q}_j &= \sum_{i \in I} \Gamma_{ij} p_i \\ \boldsymbol{q}_j &= (\boldsymbol{p}\Gamma)_j \\ \boldsymbol{q} &= \boldsymbol{p}\Gamma \end{aligned}$$

Theorem 3

Let Γ be a channel and $\boldsymbol{p}$ an input distribution for Γ. Then

$$H(\Gamma; \boldsymbol{p}) \leq H(\boldsymbol{p})$$

Equality holds if and only if $\boldsymbol{p}$ and $\boldsymbol{q} = \boldsymbol{p}\Gamma$ are independent distributions.

Proof

$\boldsymbol{p}$ and $\boldsymbol{q}$ are the marginal distribution of the joint distribution $\boldsymbol{t}$.

From Theorem 1 in Chapter 4,

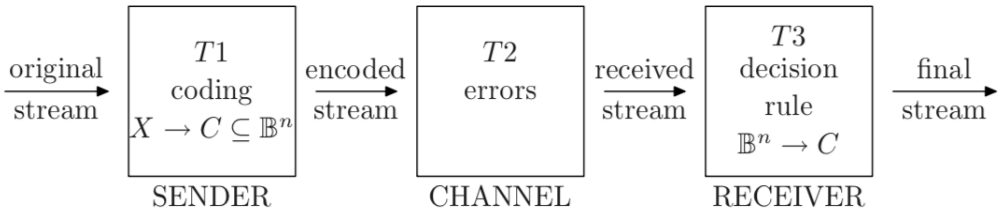
$H(\boldsymbol{t}) \leq H(\boldsymbol{p}) + H(\boldsymbol{q})$, with equality iff $\boldsymbol{p}$ and $\boldsymbol{q}$ are independent

By definition,

$$\begin{aligned} H(\Gamma; \boldsymbol{p}) &= H(\boldsymbol{t}) - H(\boldsymbol{q}) \\ H(\Gamma; \boldsymbol{p}) + H(\boldsymbol{q}) &= H(\boldsymbol{t}) \\ H(\Gamma; \boldsymbol{p}) + H(\boldsymbol{q}) &\leq H(\boldsymbol{p}) + H(\boldsymbol{q}) \\ H(\Gamma; \boldsymbol{p}) &\leq H(\boldsymbol{p}) \end{aligned}$$

6 The problem of reliable communication

In this chapter we consider the transmission of coded messages through a noisy channel.



We consider a source emitting a stream of symbols belonging to an alphabet X. We shall refer to the output from this source as the **original stream**.

To send the message using a binary symmetric channel, the Sender encodes the original stream using a binary code $c : X \to B^n$. We shall refer to the stream emitted by the Sender as the **encoded stream**.

When the encoded stream is transmitted through a BSC, errors are made. The output of the channel, which we shall refer to as the **received stream**, is not the same as the encoded stream.

Due to error, the received stream may contain any word $z \in B^n$ that is not defined as a codeword. Therefore, the Receiver must use a decision rule $\sigma : B^n \to C$ to decide the codeword C when z is received. Using a decision rule σ, the Receiver produces a **final stream**.

6.1 Definitions

Decision Rule

Let $C \subset B^n$ be a code. A decision rule for C is a function $\sigma : B^n \to C$ which assigns to each $z \in B^n$ a codeword $c \in C$.

Mistake

We say that a mistake occurs if a codeword in the final stream is not the same as the codeword in the corresponding position in the encoded stream.

Mistakes are caused by the errors introduced during the transmission through a noisy channel.

Product channel

Let Γ and Γ' be channels with input alphabets I,I' and output alphabets J,J' respectively. The product channel $\Gamma'' = \Gamma \times \Gamma'$ has input alphabet $I \times I'$ and output alphabet $J \times J'$, and its matrix is given by the rule

$$(\Gamma'')_{ii'\ jj'} = \Gamma_{ii'}\Gamma'_{jj'}$$

Meaning $Pr(output\ is\ jj' \mid input\ is\ ii') = Pr(j|i)Pr(j'|i')$

In the case $\Gamma = \Gamma'$, we denote $\Gamma \times \Gamma'$ by Γ^2.

Extended Channel

Given a channel Γ with input alphabet I and output alphabet J. The extended channel is defined on words of length n as Γ^n.

Hamming Distance

Given two words $x, y \in B^n$ such that $x = x_1x_2x_3...x_n$ and $y = y_1y_2y_3...y_n$. The Hamming distance $d(x,y)$ is the number of places where x and y differ.

In mathematical terms, the number of i $(1 \leq i \leq n)$ such that $x_i \neq y_i$

Minimum Distance - Error Correction

Let $d(c,c')$ denote the Hamming distance between codewords c and c' in a code $C \subset B^n$. The minimum distance is

$$\delta = min\ d(c,c') \quad c \neq c'$$

In other words, it computes the hamming distance between all codewords in C, and find the minimum distance

A code with minimum distance d can **correct** $\frac{d-1}{2}$ errors.

A code with minimum distance d can **detect** $d-1$ errors.

Neighbourhood

For any word $x \in B^n$ and any $r \geq 0$, the neighbourhood of x with radius r is the set

$$N_r(x) = \{y \in B^n | d(x,y) \leq r\}$$

$N_r(x)$ contains all the words that can be obtained from x by making not more than r bit-errors.

Aware that $N_r(x)$ includes x itself.

Error-correcting code

A code $C \subset B^n$ is an r-error-correcting code if $\delta \geq 2r+1$.

6.2 The Extended BSC

Suppose we are given a channel Γ with input set I and output set J, and a positive integer n. Then we define the extended channel Γ^n to be the n-fold product of copies of Γ.

The inputs to Γ^n are words of length n in I, and the outputs are words of length n in J.

If Γ is a binary symmetric channel, then we say that Γ^n is an extended BSC.

For example, let Γ is a binary symmetric channel, then we can define Γ^2 as

$$\Gamma^2 = \begin{pmatrix} 1-e & e \\ e & 1-e \end{pmatrix} \otimes \begin{pmatrix} 1-e & e \\ e & 1-e \end{pmatrix} = \begin{pmatrix} (1-e)^2 & e(1-e) & e(1-e) & e^2 \\ e(1-e) & (1-e)^2 & e^2 & e(1-e) \\ e(1-e) & e^2 & (1-e)^2 & e(1-e) \\ e^2 & e(1-e) & e(1-e) & (1-e)^2 \end{pmatrix}$$

6.2.1 Bit error probability of Extended BSC

Let $x, y \in B^n$. The entry $(\Gamma^n)_{xy}$ in the channel matrix for the extended BSC with bit-error probability e is given by

$$(\Gamma^n)_{xy} = e^d(1-e)^{n-d}$$

where $d = d(x,y)$ is the Hamming distance of x and y

Proof

Take input $x \in B^n$ and output $y \in B^n$ with Hamming distance $d(x,y) = d$.

Thus, d errors occur, the probability of this is e^d.

The remaining bits are transferred correctly, this has the probability $(1-e)^{n-d}$

Therefore $(\Gamma^n)_{xy} = Pr(y|x) = e^d(1-e)^{n-d}$

6.2.2 Equivalence of Decision Rules

For an extended BSC channel Γ^n with bit-error $e < 1$ the maximal likelihood rule is equivalent to the MD rule.

Proof

Suppose $c \in C$ with $d(c,z) = i$, then $Pr(z|c) = e^i(1-e)^{n-i}$.

Suppose $c' \in C$ with $d(c,z) = i'$, then $Pr(z|c') = e^{i'}(1-e)^{n-i'}$.

If $i < i'$ then

$$\begin{aligned} \frac{Pr(z|c)}{Pr(z|c')} &= \frac{e^i(1-e)^{n-i}}{e^{i'}(1-e)^{n-i'}} \\ &= e^{i-i'}(1-e)^{i'-i} \\ &= \left(\frac{1-e}{e}\right)^{i'-i} \end{aligned}$$

Since $e < \frac{1}{2}$ then $\frac{1-e}{e} > 1$

Since $i < i'$ then $\left(\frac{1-e}{e}\right)^{i'-i} > 1$.

Therefore, $Pr(z|c) > Pr(z|c')$

In other words, choosing c so that $d(z,c)$ is minimal is the same as maximizing $Pr(z|c)$.

6.3 Decision rules

The idea of decision rule is that when a word $z \in B^n$ is received, the Receiver must try to make a reasonable choice as to which codeword $\sigma(z) \in C$ was sent.

6.3.1 Ideal observer rule

The ideal observer rule says that, given z, the Receiver should choose $\sigma(z)$ to be a codeword c for which the probability that c was sent, given that z is received, is greatest.

In other words, we want to find a c for z such that $Pr(c|z)$ is the greatest.

The terms in the channel matrix Γ_{cz} only gives $Pr(z|c)$.

$$Pr(c|z) = \frac{\boldsymbol{p}_c Pr(z|c)}{\boldsymbol{q}_z} = \frac{\boldsymbol{p}_c \Gamma_{cz}}{\boldsymbol{q}_z}$$

Where $\boldsymbol{p}$ and $\boldsymbol{q}$ are the input and output probability distribution.

In general the characteristics of the input will not be known to the Receiver, therefore the ideal observer rule is impractical

6.3.2 Maximum likelihood rule

The maximum likelihood rule says that, given z, the Receiver should choose $\sigma(z)$ to be a codeword c that satisfies

$$Pr(z|c) \geq Pr(z|c') \quad \forall c' \in C$$

Given z, pick a largest term Γ_{cz} in column z of the channel matrix and put $\sigma(z) = c$.

If there is more than one c satisfying this condition, the receiver must make an arbitrary choice.

6.3.3 Minimum distance rule

The minimum distance rule (or MD rule) says that given $z \in B^n$, the Receiver should choose $\sigma(z)$ to be a codeword c such that $d(z,c)$ is a minimum.

6.4 Error Correction

Suppose the Sender uses a code C having minimum distance $\delta \geq 2r+1$, and the Receiver uses the MD rule.

Then, provided that not more than r bit-errors are made in transmitting any codeword, no mistakes will occur.

We call the code C a r-correcting code because it corrects r errors. For example if $\delta \geq 3$, then C corrects 1 error.

Proof

Let c be a codeword, and z the corresponding received word. Provided that not more than r bit-errors are made, then $z \in N_r(c)$.

From Theorem 1, $z \notin N_r(c')$ for all $c' \neq c$.

Therefore $d(c,z) < d(c',z)$ for all c'. And so the MD rule will correctly assign c to z.

6.5 Packing Bound

When we try to construct error-correcting codes by ensuring that δ has a given value, the number of codewords $|C|$ is constrained. This is because we want to find a set of codewords that are 'far apart'

The packing bound states that for a code $C \subset B^n$ with minimum distance $\delta \geq 2r+1$ then

$$|C|\left(1+n+\binom{n}{2}+\ldots+\binom{n}{r}\right) \leq 2^n$$

Proof

Given a codeword c, the words z such that $d(c,z)=i$ is generated by any i of the n bits in c

Thus the number of words z such that $d(c,z) \leq r$ is

$$1+n+\binom{n}{2}+\ldots+\binom{n}{r}$$

This is the size of neighbourhood $N_r(c)$ for each $c \in C$.

Since $\delta \geq 2r+1$, From Theorem 1, we know that all neighbourhoods are disjoint.

The number of codewords $|C|$ multiply by the size of one neighbourhood $N_r(c)$ is less than equal the total number of words 2^n

6.5.1 Finding Minimum bits with Packing Bound

Suppose we want to issue 100 ID numbers ($|C| = 100$), in the form of strings of n bits.

If we want a 2-error-correcting code, what is the minimum value of n

Using Packing Bound Therory

$$100\left(1 + n + \binom{n}{2}\right) \leq 2^n$$
$$50(n^2 + n + 2) \leq 2^n$$
$$n \geq 14$$

Meaning we can construct a set of 100 codewords, each of 14 length, for the code to be 2-error-correcting

6.5.2 Finding Maximum number of Codewords

Given the number of bits n for each codeword and minimum distance δ. What is the largest possible size for the number of codewords $|C|$.

The packing bound only gives an upper limit of $|C|$, however it does not guarantee that a code of that size actually exists.

For example, for $n = 10$ and $\delta = 7$, using packing bound

$$|C|\left(1 + 10 + \binom{10}{2} + \binom{10}{3}\right) \leq 2^10$$
$$|C| \leq 5$$

Although the theorem tells us $|C| \leq 5$, this only tells us the upper limit is 5, but it does not guarantee that a the code with $|C| = 5, n = 10, \delta = 7$ exists.

In fact, when $n = 5, \delta = 7$, $|C|$ can only be a maximum of 2 to satisfy the conditions

6.6 Theorems

Theorem 1

If C is a code with minimum distance $\delta \geq 2r + 1$, then for any codewords $c, c' \in C$, the neighbourhoods $N_r(c)$ and $N_r(c')$ are disjoint.

Proof

Proof by contradiction. Suppose $N_r(c)$ and $N_r(c')$ are not disjoint, so there is a common member z. Therefore

$$d(c, c') \leq d(c, z) + d(z, c') \leq r + r \leq 2r$$

Which contradicts the fact $\delta \geq 2r + 1$.

7 The noisy coding theorems

7.1 Definitions

Probability of a mistake

The probability of a mistake when the encoded stream corresponds to a source $(C, \boldsymbol{p})$ is

$$M(C, \boldsymbol{p}) = \sum_{c \in C} \boldsymbol{p}_c M_c$$

Where $M_c = \sum_{z \in F(c)} P(z|c)$ and $F(c) = \{z \in B^n | \sigma(z) \neq c\}$

Information Rate

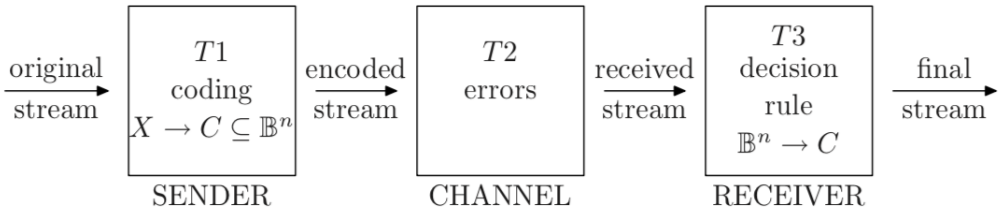
The information rate of a code $C \subset B^n$ is

$$\rho = \frac{log_2|C|}{n}$$

Where $|C|$ is the number of codewords in C

7.2 Probability of Mistake

Looking at the model of communication from the previous chapter.



We can regard the encoded stream as being emitted by a source $(C, \boldsymbol{p})$.

Given that the channel is the extended BSC, the Receiver can use the MD rule as the decision rule.

For each $c \in C$, we can define a set of words, $F(c)$, to which the MD rule σ does not assign c

$$F(c) = \{z \in B^n | \sigma(z) \neq c\}$$

According to the definition of **mistake** in Chapter 6, a mistake occurs when a codeword $c \in C$ is sent but a word $z \in F(c)$ is received.

Therefore the probability that a mistake occurs when $c \in C$ is sent is

$$M_c = \sum_{z \in F(c)} P(z|c)$$

It is the sum of probability that c is sent and z is received for all $z \in F(c)$

Hence, the probability of a mistake for the encoded stream corresponds to a source $(C, \boldsymbol{p})$ is

$$M(C, \boldsymbol{p}) = \sum_{c \in C} \boldsymbol{p}_c M_c$$

7.2.1 Estimating Probability of Mistake

If $C \subset B^n$ is an r-error-correcting code, and z is in $F(c)$, then $d(z,c) \geq r+1$. For r-error-correcting code, if $d(z,c) < r$ then z will be assigned to c and $z \notin F(c)$

From Chapter 6, we see that for an extended BSC, $Pr(z|c) = \Gamma_{cz} = e^{d(c,z)}(1-e)^{n-d(c,z)}$

Since the probability of error $0 < e < 1$ and $d(z,c) \geq r+1$.

$$e^{d(c,z)}(1-e)^{n-d(c,z)} \leq e^{r+1}$$

Since $M_c = \sum_{z \in F(c)} P(z|c)$ for a codeword $c \in C$. If we have an estimate of $|F(c)|$, the number of words not assign to c, we can estimate the upper bound of M_c by

$$M_c \leq \sum_{z \in F(c)} e^{r+1} = |F(c)|e^{r+1}$$

7.3 Coding at a given rate

In this section we want to choose the code $C \subset B^n$ so that the probability of a mistake is arbitrarily small, while information is transmitted at a given rate ρ.

The key to maintaining a given information rate ρ is to code blocks of symbols. If the Sender divides the original stream of bits into blocks of size k, there will be 2^k possible blocks, in other words $|C| = 2^k$.

The information rate ρ is given by

$$\rho = \frac{log_2|C|}{n}$$
$$\rho = \frac{k}{n}$$

Meaning for a code to be transmitted at a given rate ρ, the sender must choose parameters k and n such that

1. $|C| = 2^k$
2. $k \geq \rho n$

7.4 Transmission using the extended BSC

7.4.1 Capacity of Extended BSC

From Section 5.2.2, we know the capacity of BSC is $\gamma = 1 - h(e)$. We now try to find the capacity of extended BSC.

Suppose the encoded stream is emitted by a source $(C, \boldsymbol{p})$. Then the received stream is a source $(B^n, \boldsymbol{q})$, where $\boldsymbol{q} = \boldsymbol{p}\Gamma_n$.

From Section 5.4.1, we see that the capacity of a channel can be calculated as

$$\gamma = max(H(\boldsymbol{q}) - H(\boldsymbol{q}|\boldsymbol{p}))$$

Where

$$H(\boldsymbol{q}|\boldsymbol{p}) = \sum_{i \in I} p_i H(\boldsymbol{q}|i) \qquad H(\boldsymbol{q}|i) = \sum_{j \in J} \Gamma_{ij} log\left(\frac{1}{\Gamma_{ij}}\right)$$

For extended BSC

$$\begin{aligned} H(\boldsymbol{q}|c_1 ... c_n) &= \sum_{z_1 ... z_n} \Gamma^n_{c_1 ... c_n z_1 ... z_n} log\left(\frac{1}{\Gamma^n_{c_1 ... c_n z_1 ... z_n}}\right) \\ &= \sum_{z_1} ... \sum_{z_n} \Gamma_{c_1 z_1} ... \Gamma_{c_n z_n} log\left(\frac{1}{\Gamma_{c_1 z_1} ... \Gamma_{c_n z_n}}\right) \\ &= \sum_{i=1}^{n} \sum_{z_i} \Gamma_{c_i z_i} log\left(\frac{1}{\Gamma_{c_i z_i}}\right) \end{aligned}$$

Since the values of z_i is 0 or 1

$$\begin{aligned} H(\boldsymbol{q}|c_1 ... c_n) &= \sum_{i=1}^{n} \left[\Gamma_{c_i 0} log\left(\frac{1}{\Gamma_{c_i 0}}\right) + \Gamma_{c_i 1} log\left(\frac{1}{\Gamma_{c_i 1}}\right) \right] \\ &= \sum_{i=1}^{n} \left[e log\left(\frac{1}{e}\right) + (1-e) log\left(\frac{1}{1-e}\right) \right] \\ &= \sum_{i=1}^{n} h(e) \\ &= n \cdot h(e) \end{aligned}$$

where h is the standard entropy.

Therefore the capacity of extended BSC is

$$\gamma_{extendedBSC} = max(H(\boldsymbol{q}) - n \cdot h(e))$$

From Section 3.3.1, we know that the maximum entropy of m symbols is at most $log_b m$.

Since $\boldsymbol{q}$ is a distribution on the set B^n, $m = 2^n$. It follows that the maximum of $H(\boldsymbol{q}) \leq log_2 2^n = n$.

Therefore the capacity of extended BSC is

$$\begin{aligned} \gamma_{extendedBSC} &= n - n \cdot h(e) \\ &= n\gamma \end{aligned}$$

7.4.2 Rate and Capacity for Extended BSC

Let $C \subset B^n$ be a code with rate ρ. Let $\boldsymbol{p}^*$ be the distribution in which each codeword in C is equally probable.

Suppose the stream emitted by source $(C, \boldsymbol{p}^*)$ is transmitted through extended BSC Γ^n, where the capacity of Γ is γ. Then

$$H(\Gamma^n; \boldsymbol{p}^*) \geq n(\rho - \gamma)$$

Proof

From above we know capacity of $\Gamma^n = n\gamma$. By definition of capacity, $n\gamma$ is the maximum of $H(\boldsymbol{p}) - H(\Gamma^n; \boldsymbol{p})$ for all $\boldsymbol{p}$. Therefore, we have

$$H(\boldsymbol{p}^*) - H(\Gamma^n; \boldsymbol{p}^*) \leq n\gamma$$

In Section 3.3.1, we know that the maximum entropy occurs when all symbols are equally probable, meaning when the distribution is $\boldsymbol{p}^*$, and the maximum entropy is $log_b m$. Therefore

$$H(\boldsymbol{p}^*) = log_2 |C|$$

Rearranging the equations above

$$\begin{aligned} &H(\Gamma^n; \boldsymbol{p}^*) \geq log_2 |C| - n\gamma \\ &H(\Gamma^n; \boldsymbol{p}^*) \geq n\rho - n\gamma \qquad\qquad \rho = \frac{log_2 |C|}{n} \\ &H(\Gamma^n; \boldsymbol{p}^*) \geq n(\rho - \gamma) \end{aligned}$$

This shows that, when the required information rate exceeds the capacity of the channel, the uncertainty per bits cannot be made arbitrarily small for all distributions $\boldsymbol{p}$, no matter what we choose for C and n.

7.5 Rate and Capacity

If the probability of a mistake is required to be arbitrarily small, then the rate of information should not exceed the capacity of the channel.

Suppose an infinite sequence $n = n_1, n_2, ...$, we construct code $C \subset B^n$ such that $|C| \geq 2^{n\rho}$.

Let $\boldsymbol{p}^*$ be the equiprobable distribution on C. If $\rho > \gamma$, then probability of mistake $M(C, \boldsymbol{p}^*)$ does not tend to 0 as n tends to infinity.

Proof

Let $|C| \geq 2^k$ where $k \geq n\rho$, and probability of mistake $M = M(C, \boldsymbol{p}^*)$.

From Theorem 1 Fano's inequality

$$H(\Gamma^n, \boldsymbol{p}) \leq h(M) + M \cdot log_2(|C| - 1)$$
$$H(\Gamma^n, \boldsymbol{p}) < 1 + Mk \qquad h(M) \leq 1, \quad log_2(|C| - 1) < log_2|C| = k$$

From Section 7.4.2 we know

$$H(\Gamma^n; \boldsymbol{p}^*) \geq log_2|C| - n\gamma$$
$$H(\Gamma^n; \boldsymbol{p}^*) \geq k - n\gamma$$

Combining the inequalities

$$1 + Mk > k - n\gamma$$
$$M > 1 - \frac{n\gamma + 1}{k}$$
$$M \geq 1 - \frac{n\gamma + 1}{n\rho}$$

Therefore when $n \to \infty$, M does not tend to 0

7.5.1 Shannon's Theorem

Shannon's Theorem is logically equivalent to the converse of the above theorem.

It states that if $\rho < \gamma$, then it is possible to construct code $C \subset B^n$ such that $|C| \geq 2^{n\rho}$ and then probability of mistake $M(C, \boldsymbol{p})$ tend to 0 as n tends to infinity.

Although Shannon proved this is true, it does not provide a practical method for constructing the required code.

7.6 Theorems

Theorem 1 (Fano's Inequality)

For a code $C \subset B^n$, a source $(C, \boldsymbol{p})$ with probability of mistake $M = M(C, \boldsymbol{p})$ being transmitted through extended BSC Γ^n using MD rule. Then

$$H(\Gamma^n, \boldsymbol{p}) \leq h(M) + M \cdot log_2(|C| - 1)$$

8 Linear Codes

8.1 Definitions

Linear Codes

A (binary) linear code is a subspace C of the vector space F_2^n.

A subset C of the vector space F_2^n is a subspace if $x, y \in C \rightarrow x + y \in C$

A subspace C has dimension k, which is the size of minimum spanning set.

Every element in C can be expressed as a linear combination of the spanning set, meaning $|C| = 2^k$, where $0 \leq k \leq n$.

The information rate ρ can be represented using k and n

$$\rho = \frac{log_2|C|}{n} = \frac{k}{n}$$

Therefore we can describe a linear code with the parameters (n, k, δ).

Weight

The weight $w(x)$ of a word x is the number of non-zeros in x.

For $x, y \in F_2^n$, the weight of x is the number of 1s in x, and the weight of y is the number of 1s in y

The weight of $x + y$ can be calculated by

$$w(x + y) = w(x) + w(y) - 2(x \wedge y)$$

Syndrome

Let C be the linear code defined by a check matrix H. For any word $z \in F_2^n$ the syndrome of z is s, where $Hz^T = s^T$.

Dual

If a code C has the parameters $[n, k, d]_q$ over F_q, the dual of the code is the set

$$C^\perp = \{x \in F_q^n \mid \forall y \in C \langle x, y \rangle = 0\}$$

where $\langle x, y \rangle$ is the standard dot product.

Singleton bounds

The Singleton bound states that if C is a (n, k, d) code, then $k \leq n - d + 1$.

Maximum Distance Separable Code (MDS)

A (n, k, d) code is called Maximum Distance Separable (MDS) if $d = n - k + 1$.

8.2 Linear Codes

Now we shall use algebraic methods in order to construct codes for the purpose of error-correction.

In the case of the binary alphabet B, the symbols 0 and 1 can be added and multiplied by

$$0+0=0 \quad 0+1=1 \quad 1+0=1 \quad 1+1=0$$
$$0\times 0=0 \quad 0\times 1=0 \quad 1\times 0=0 \quad 1\times 1=1$$

The set B with these operations is called a **Field**. We use the notation F_2 for this field.

The set F_2^n of words of length n in F_2 is a vector space. The vectors can be added and multiplied by

$$(x_1x_2x_3...x_n)+(y_1y_2y_3...y_n)=(x_1+y_1 \;\; x_2+y_2 \;\; x_3+y_3...x_n+y_n)$$
$$0\times(x_1x_2x_3...x_n)=(000..00) \quad 1\times(x_1x_2x_3...x_n)=(x_1x_2x_3...x_n)$$

Vector addition is equivalent to logical XOR. Vector multiplication is equivalent to logical AND.

In the case of F_2, some special features are

$$x=-x \qquad x+y=x-y$$

8.2.1 Weights

For a linear code, the minimum distance δ is equal to the minimum weight of a nonzero codeword.

Proof

Suppose c_1 and c_2 are codewords such that $c_1 \neq c_2$.

A bit in c_1+c_2 is 1 iff the corresponding bits in c_1 and c_2 are different. Therefore $d(c_1,c_2)=w(w_1+w_2)$.

Since C is a subspace of F_2^n, $x,y\in C\rightarrow x+y\in C$. Hence each value of $d(c_1,c_2)$ is the weight of a codeword, and the minimum value is the minimum weight.

For example given the code $\{000000,100000,010000,110000\}$. The code has word length $n=6$. There are $4=2^2$ codewords so the dimension $k=2$. The weights are 0, 1, 1, 2, and so $\delta=1$, meaning this code has no error correction. We can represent this linear code as $(n=6,k=2,\delta=1)$.

8.3 Constructing Codes Using Matrix

In the previous chapter, we see that in order to reduce mistake while transferring at a given rate, the key was to split the original stream of bits into blocks of size k and

assign a codeword of length n to each of the 2^k possible blocks.

We require an encoding function $F_2^k \to F_2^n$ defined by a matrix G, known as **Generator Matrix**

For a word $x \in F_2^n$, G is an $k \times n$ matrix such that $x = yG$, where $y \in F_2^k$.

For example we have a code

$$\begin{array}{cccc} 000 \mapsto 000000 & 001 \mapsto 001011 & 010 \mapsto 010110 & 100 \mapsto 100101 \\ 011 \mapsto 011101 & 101 \mapsto 101110 & 110 \mapsto 110011 & 111 \mapsto 111000. \end{array}$$

For this code we have

$$\begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix} G = \begin{pmatrix} y_1 & y_2 & y_3 & y_1+y_2 & y_2+y_3 & y_1+y_3 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{pmatrix}$$

Therefore the matrix

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

8.4 Check Matrix of Linear Code

For an $k \times n$ matrix G over F_2 in the form

$$\begin{pmatrix} I & A \end{pmatrix}$$

where I is the Identity matrix with size k, and A is any $k \times (n-k)$ matrix.

The the code $\{x \in F_2^n \mid x = yG,\ y \in F_2^k\}$ can be defined as $\{x \in F_2^n \mid Hx^T = \mathbf{0}^T\}$.

Where H is a $(n-k) \times n$ matrix in the form $\begin{pmatrix} -A^T & I \end{pmatrix}$.

Example

Consider the code above the transformation $x = yG$ was defined by the linear equations

$$x_1 = y_1 \quad x_2 = y_2 \quad x_3 = y_3 \quad x_4 = y_1 + y_2 \quad x_5 = y_2 + y_3 \quad x_6 = y_1 + y_3$$

Recall in the case of F_2, $x = -x$, therefore we can eliminate y_1, y_2, y_3 to give

$$x_1 + x_2 + x_4 = 0 \quad x_2 + x_3 + x_5 = 0 \quad x_1 + x_3 + x_6 = 0$$

This equations can be written of matrix equation $Hx^T = \mathbf{0}^T$, where

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

In Linear Algebra this set is known as the kernel or null space of H

8.4.1 Check Matrix

H is called the check matrix for the code $\{x \in F_2^n \mid Hx^T = \mathbf{0}^T\}$.

$$H = \begin{pmatrix} A & I \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1k} & 1 & 0 & ... & 0 \\ a_{21} & a_{22} & ... & a_{2k} & 0 & 1 & ... & 0 \\ . & . & ... & . & . & . & ... & . \\ . & . & ... & . & . & . & ... & . \\ a_{m1} & a_{m2} & ... & a_{mk} & 0 & 0 & ... & 1 \end{pmatrix}$$

The matrix can be represented in linear equations

$$\begin{aligned} x_{k+1} &= a_{11}x_1 + a_{12}x_2 + ... + a_{1k}x_k \\ x_{k+2} &= a_{21}x_1 + a_{22}x_2 + ... + a_{2k}x_k \\ &. \\ &. \\ &. \\ x_n &= a_{m1}x_1 + a_{m2}x_2 + ... + a_{mk}x_k \end{aligned}$$

In a typical word $x = x_1x_2...x_n$, $x_1, x_2, ..., x_k$ are called **message bits**. $x_{k+1}, x_{k+2}, ..., x_n$ are called **check bits**.

Since there are 2^k possible values of the message bits, the dimension of the code is k.

Example

Given the check matrix

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

We know that for a word $x = x_1x_2x_3x_4$.

x_3 and x_4 are check bits.

x_1 and x_2 are the message bits.

In particular, when we represent the matrix in linear equations

$$x_3 = x_1 \qquad x_4 = x_1 + x_2$$

To find all the codewords, we find all possible values for the message bits x_1 and x_2, giving us

$$\{0000, 0101, 1011, 1110\}$$

The code has parameters $n = 4$, $k = 2$, and from the weights $\delta = 2$.

8.4.2 Minimum distance from Check Matrix

The minimum distance $d(C)$ of a code C with check matrix H is given by

$$d(C) = 1 + max\{t : \text{every } t \text{ columns of } H \text{ are linearly independent}\}$$

Or alternatively

$$d(C) = min\{t : t \text{ columns of } H \text{ that are linearly dependent}\}$$

Example

Given check matrix H

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}$$

The maximum number of linear independent columns is 2, i.e. $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Adding anymore of the columns will make them linearly dependent. Therefore

$$\begin{aligned} d(C) &= 1 + max\{t : \text{every } t \text{ columns of } H \text{ are linearly independent}\} \\ &= 1 + 2 = 3 \end{aligned}$$

Alternatively, we can look at the minimum number of linear dependent columns in H. There are three dependent columns $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 1 \end{pmatrix}$. Therefore

$$\begin{aligned} d(C) &= min\{t : t \text{ columns of } H \text{ that are linearly dependent}\} \\ &= 3 \end{aligned}$$

8.5 Constructing 1-error-correcting code

We can construct a 1-error-correcting code C by a checking matrix H provided that

1. No column in H consists entirely 0
2. No two columns in H are the same

Proof

For a 1-error-correcting code, the minimum distance δ is at least 3. Since the minimum distance distance δ is equal to the minimum weight. The proof should show that if the two conditions are satisfied, the minimum weight is at least 3.

Case minimum weight of a code is 1: Proof by contradiction

Suppose there is a codeword $c \in C$ with weight 1, say $c = 000...010...000$ with the i^{th} bit being 1, and $Hc^T = \mathbf{0}^T$.

According the matrix multiplication Hc^T equals to the i^{th} column in H.

Therefore the i^{th} column in H consists entirely 0. Validating condition 1

Case minimum weight of code is 2: Proof by contradiction

Suppose there is a codeword $c \in C$ with weight 1, say $c = 000...010...010...000$ with the i^{th} and j^{th} bit being 1, and $Hc^T = \mathbf{0}^T$.

According the matrix multiplication Hc^T equals to the sum of i^{th} and j^{th} column in H.

Therefore sum of i^{th} and j^{th} column in H is $\mathbf{0}$. According to arithmetic in the field F_2, the i^{th} and j^{th} column are the same. Validating condition 2.

Hence if the 2 conditions are met, minimum weight is 3.

8.5.1 Correcting codes

Let $C \subset F_2^n$ be a linear code defined by a check matrix H. Suppose that 1 bit error is made at the i^{th} bit during transmission and word z is received. The error bit i can be determined by the fact that Hz^T is the i^{th} column in H.

Proof

Suppose codeword c is sent with the error at i^{th} bit and word z is received.

$z = c + w$ where$w = 00...010..00$ with the i^{th} bit being 1. Then

$$Hz^T = H(c+w)^T = Hc^T + Hw^T = \mathbf{0}^T + Hw^T$$

which is the i^{th} column in H.

Therefore for each received word z, the receiver first calculate Hz^T. If $Hz^T = \mathbf{0}^T$, then z is a codeword. Otherwise, Hz^T is the i^{th} column of H so correct the i^{th} bit in z.

8.6 Decoding Problem

Given a set of codewords $C \subset B^n$ and a word $z \in B^n$ the problem is to find a codeword $c \in C$ that is nearest to z, in the sense of Hamming distance.

Section 8.5.1 suggests one decoding method.

Another approach to the decoding problem is a technique called **syndrome decoding**

8.6.1 Syndrome

For a word z, its syndrome s is defined by $Hz^T = s^T$. If z is a codeword, its syndrome will be $\mathbf{0}^T$. If z has an error at the i^{th} bit, its syndrome will be the i^{th} column in H.

Two words $y, z \in F_2^n$ will have the same syndrome iff $y = z + c$ for some $c \in C$.

Proof

$$Hz^T = Hy^T \leftrightarrow H(y^T - z^T) = \mathbf{0}^T \leftrightarrow y - z \in C \leftrightarrow z = y + c$$

8.6.2 Coset

The **coset** of z with respect to C is a set of $y \in F_2^n$ such that $y = z + c$

The number of coset for a code is 2^m, where m is the number of rows in H. Each coset has a different syndrome.

A coset leader is a word of least weight in its coset; if there are several possibilities, we choose one for definiteness.

A syndrome look-up table is an ordered list of pairs (s, f) such that, for each syndrome s, f is the coset leader for the coset that has syndrome s.

Example

Find all cosets for the given check matrix

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

This check matrix H defines the code $C = \{00000, 01011, 10110, 11101\}$.

Since H has 3 rows, there are a total of $2^3 = 8$ cosets.

To construct the cosets, we start with the coset $00000 + C$ that is C itself. We then choose any z that is not in this coset and construct the next coset $z + C$. Continue until all 8 cosets are formed.

00000+C	00000	01011	10110	11101
10000+C	10000	11011	00110	01101
01000+C	01000	00011	11110	10101
00100+C	00100	01111	10010	11001
00010+C	00010	01001	10100	11111
00001+C	00001	01010	10111	11100
00101+C	00101	01110	10011	11000
00111+C	00111	01100	10001	11010

Each coset has a different syndrome. We can define the syndrome lookup table as follows

Coset Leader	Syndrome
00000	000
10000	110
01000	011
00100	100
00010	010
00001	001
00101	101
01100	111

8.6.3 Syndrome Decoding

The syndrome decoding method is based on the assumption that the Receiver knows the check matrix H for a code C and has a copy of the look-up table.

For a received word z, calculate syndrome s using $s^T = Hz^T$.

Look up the coset leader f for syndrome s

Decode z using $\sigma(z) = z + f$

Using the method above we guarantee:

1. $\sigma(z) = z + f$ is a codeword
2. There is no codeword $c \in C$ such that $d(z,c) < d(z, z+f)$.

Proof

1. Since f and z have the same syndrome s, they are in the same coset. And $f = z + c$ for some $c \in C$.Therefore

$$z + f = z + z + c = c$$

2. $d(z, z+f) = w(z+z+f) = w(f)$ where f is the coset leader.

For any other codeword c, $d(z,c) = w(z+c)$. Since $z+c$ and f are in the same coset, and f is the coset leader, it follows that $w(z+c) \geq w(f) = d(z, z+f)$

Example

Using the code above, suppose $z = 10111$ is received.

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

The coset leader f for syndrome 001 is 00001. Therefore $\sigma(10111) = 10111 + 00001 = 10110$

8.7 Duals of Linear Codes

If a code C has the parameters $[n,k,d]_q$ over F_q, the dual of the code is the set

$$C^{\perp} = \{x \in F_q^n \mid \forall y \in C \langle x,y \rangle = 0\}$$

where $\langle x,y \rangle$ is the standard dot product.

Because $C^{\perp}$ is the space of elements perpendicular to C, the dimension of the parameters of $C^{\perp}$ is $[n, n-k, d]$.

8.7.1 Properties of Dual Codes

1. If code C has generator matrix G and check matrix H. $C^{\perp}$ has generator matrix H and check matrix G

2. The dual of an MDS code is also a MDS code.

8.7.2 Self Dual Codes

Let C be a linear code, the code C is a self dual if $C = C^{\perp}$.

1. n must be even

If the code C has the parameters $[n,k,d]_q$, the dual code has the parameters $[n,n-k,d]_q$. For a self dual code, $k = n-k$, and therefore $n = 2k$ must be even.

2. Every codeword in C must be orthogonal to itself and orthogonal to all codewords in C.

In the case where C is a binary code with parameters $[n,k,d]_2$, every codeword in C must have even weight to be orthogonal to itself.

This means the n-bits all 1 codeword is orthogonal to all codewords in C as n is even. Therefore the codeword $111...111 \in C^{\perp}$ hence is also in C.

8.8 Boundaries of a Code

We are interested in finding the maximum possible size of q-ary code of length n and minimum distance d.

In general, we do not know the exact maximum possible size, but we do know their boundaries.

8.8.1 Singleton Bound

The Singleton bound states that if C is a (n,k,d) code, then $d \leq n-k+1$.

8.8.2 Hamming Bound

The Hamming bound is also known as Packing bound, which we looked into in section 6.5. Here we look at the packing bound of q-ary code.

Recall the definition of Neighbourhood in section 6. For any word x, and radius r. The neighbourhood

$$N_r(x) = \{y \in F_q^n \mid d(x,y) \leq r\}$$

The size of a neighbourhood for q-ary code is given by

$$|N_r(x)| = 1 + (q-1)n + (q-1)^2\binom{n}{2} + ... + (q-1)^r\binom{n}{r}$$

For Hamming bound, given minimum distance d, the radius $r = \frac{d-1}{2}$

Each codeword c will take up **at least** $|N_{\frac{d-1}{2}}(c)|$ space from F_q^n. The Hamming bound is

$$q^k\left(1 + (q-1)n + (q-1)^2\binom{n}{2} + ... + (q-1)^{\frac{d-1}{2}}\binom{n}{\frac{d-1}{2}}\right) \leq q^n$$

Rearranging the equation, we can find a boundary for the rate

$$\begin{aligned} q^k &\leq \frac{q^n}{|N_{\frac{d-1}{2}}(c)|} \\ k &\leq n - log_q|N_{\frac{d-1}{2}}(c)| \\ \frac{k}{n} &\leq 1 - \frac{log_q|N_{\frac{d-1}{2}}(c)|}{n} \end{aligned}$$

8.8.3 Gilbert-Varshamov bound

The GV Bound gives the lower bound for maximum possible size of a q-ary code of length n and minimum distance at least d.

Assume a code C of code length n and minimum distance d is in maximum size.

The for all $x \in F_q^n$, there must be a codeword $c \in C$ such that $d(x,c) \leq d-1$. Otherwise, x can be added to C and still maintain minimum distance d, hence contradicts that C is in maximum size.

For GV bound, given minimum distance d, the radius $r = d-1$.

Each codeword c will take up **at most** $|N_{d-1}(c)|$ space from F_q^n. The GV bound is

$$q^k \geq \frac{q^n}{1 + (q-1)n + (q-1)^2\binom{n}{2} + ... + (q-1)^{d-1}\binom{n}{d-1}}$$

Rearranging the equation, we can find a boundary for the rate

$$k \geq n - log_q|N_{d-1}(c)|$$
$$\frac{k}{n} \geq 1 - \frac{log_q|N_{d-1}(c)|}{n}$$

9 Algebraic Coding Theory

9.1 Hamming Codes

In Section 8.5, we saw that a binary linear code with $\delta \geq 3$ can be constructed by a check matrix H, provided that

1. No column in H is $\mathbf{0}$
2. All columns in H are distinct

A binary hamming code is defined by a check matrix H with m rows and $2^m - 1$ distinct columns.

A Hamming code with $m = 3$ can be defined by the check matrix H_1, with columns arranged in numerical order

$$H_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

We can rearrange the check matrix to **systematic form**

$$H_2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Note the two codes by H_1 and H_2 are equivalent, rearrangement in columns will rearrange the bits in each codewords, but does not affect the parameters of the code.

9.1.1 Properties

In binary Hamming code, every word in F_2^n is either a codeword, or at distance 1 from exactly one codeword.

Proof

There are 2^k codewords. For each codeword $c \in C$, $N_1(c) = n + 1$, every word that is distance 1 from c and c itself.

From property of check matrix H, $\delta \geq 3$ meaning all neighbourhoods are disjoint.

$$2^k(n+1) = 2^n$$

That means the 2^k codewords in C and their neighbourhood completely covers F_2^n

9.1.2 Encoding

We can easily convert check matrix H_2 into a generator matrix G,

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

We can now encode a message m by $m \cdot G = c$ to obtain codeword c.

Alternatively from the check matrix H, we can determine the check bits are given by

$$x_5 = x_2 + x_3 + x_4 \qquad x_6 = x_1 + x_3 + x_4 \qquad x_7 = x_1 + x_2 + x_4$$

So given a message m, we can append the check bits based on the rule.

9.1.3 Decoding

We can use the syndrome decoding rule for decoding. Given a code c, if the syndrome is equal i^{th} column of H, meaning there is an error at the i^{th} bit.

In this case, using H_1 has an advantage because its i^{th} column is the binary representation of the number i.

9.2 Reed Solomon Code

We define an encoding function for Reed Solomon code $RS : F_q^k \to F_q^n$ as follows.

A message $m = m_0 m_1 m_2 ... m_{k-1}$ with $m_i \in F_q$ is mapped to a degree $k-1$ polynomial, where

$$f_m(x) = m_0 + m_1 x + m_2 x^2 + ... + m_{k-1} x^{k-1}$$

Then pick a set of F_q, $\{\alpha_1, \alpha_2, ..., \alpha_n\}$, known as **evaluation points**.

The encoding of message m is

$$RS(m) = (f_m(\alpha_1), f_m(\alpha_2), ..., f_m(\alpha_n))$$

Example

Given the field F_3, we choose a set of evaluation points $\{0, 1, 2\}$.

Given a message $m = 2, 3$. We can define the function

$$f_m(x) = 2 + 3x$$

The encoding of message m is

$$\begin{aligned} RS(m) &= (f_m(\alpha_1), f_m(\alpha_2), ..., f_m(\alpha_n)) \\ &= (f_m(0), f_m(1), ..., f_m(2)) \\ &= (2, 5, 8) \end{aligned}$$

9.2.1 Generator Matrix

The Generator Matrix for Reed Solomon coding is known as the Vandermonde matrix.

Given evaluation points $\{\alpha_1, \alpha_2, ..., \alpha_n\}$, and message length k, the generator matrix is defined as

$$G = \begin{pmatrix} 1 & 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & ... & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & ... & \alpha_n^2 \\ ... & ... & ... & ... & ... \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \alpha_3^{k-1} & ... & \alpha_n^{k-1} \end{pmatrix}$$

Example

Using the example above, the set of evaluation points are $\{0, 1, 2\}$ and message length $k = 2$. Therefore

$$G = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

9.2.2 Properties

1. **Reed Solomon code is Linear**

Adding two polynomials with degree less than k gives a polynomial with degree less than k.

Multiplying a polynomial with degree less than k with a scalar in $GF(q)$ gives a polynomial with degree less than k

Therefore Reed Solomon code is a vector space

2. **Reed Solomon code matches the Singleton bound**

Reed Solomon code is a $(n, k, n-k+1)$ code with minimum distance $d = n-k+1$.

3. **Reed Solomon code is an MDS code**

A (n, k, d) code is called Maximum Distance Separable (MDS) if $d = n-k+1$.

9.3 Reed Muller Code

The problem with Reed Solomon Code is that the alphabet size has to be as large as the desired length.

Reed Muller is similar to Reed Solomon Code, the only difference is Reed Solomon Code uses univariate polynomials (only one variable x), but Reed Muller Code uses multivariate polynomials.

A Reed Muller Code $RM(r,m)$ has m variables and has order r.

Given the Field F_q, the parameters of Reed Muller Code is

$$n = q^m \qquad k = \binom{m+r}{m} \text{ if } r < q \qquad \delta = q^m - rq^{m-1}$$

9.3.1 Generator Matrix

Example 1

Given the Field F_2, start from the simplest example $RM(1,1)$ where there is only one variable to the order 1. The generator matrix is given by

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

There can only be one variable x, and the value of x is 0 or 1.

Example 2

Given the Field F_2,construct the generator matrix for $RM(1,3)$

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

There are 3 variables with order 1, (x, y, z) each of the them can be either 0 or 1.

The second row is the value for x. The third row is the values for y. The fourth row is the values of z.

Example 3

Given the Field F_2, construct the generator matrix for $RM(2,3)$

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

There are 3 variables, with $r = 2$ it means all possible combinations such that the degree of the polynomial is 2. Therefore the terms are x, y, z, xy, xz, yz.

Row 2,3,4 are the values for variables x, y and z.

Row 5 is the values of xy, which is obtained by $x \wedge y$, multiplying row 2 and 3.

Row 6 is the values of xz, which is obtained by $x \wedge z$, multiplying row 2 and 4.

Row 7 is the values of yz, which is obtained by $y \wedge z$, multiplying row 3 and 4.

Normally we do also have to include rows for x^2, y^2 and z^2. However in this case, $x^2 = x$, $y^2 = y$ and $z^2 = z$, and therefore it is not included in G.

9.3.2 Properties

The parity check matrix of $RM(r,m)$ is the generator matrix for the code $RM(m-r-1,m)$. The generator matrix of $RM(r,m)$ is orthogonal to the generator matrix of $RM(m-r-1,m)$. In Example 3 above, the generator matrix for $RM(1,3)$ is its self dual.