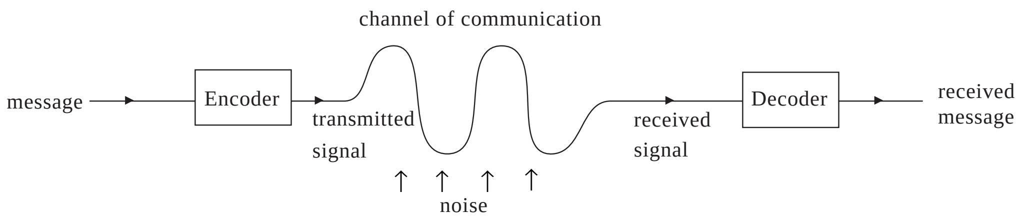
5 Binary Codes

You have already seen how check digits for bar codes (in Unit 3) and ISBN numbers (Unit 4) are used to detect errors. Here you will look at codes relevant for data transmission, for example, for transmission of pictures from Mars to the Earth, and see how such codes are designed.

5.1 Noise: Error Detection

To take an example, in TV broadcasting the message for transmission is a picture in the studio. The camera converts this into a 625-row array of packages of information, each package denoting a particular colour. This array, in the form of an electrical signal, is broadcast via antennae and the atmosphere, and is finally interpreted by the receiving set in the living room. The picture seen there differs somewhat from the original, errors having corrupted the information at various stages in the channel of communication. These errors may result in effects varying from subtle changes of colour tone to what looks like a violent snowstorm. Technically, the errors are all classified as *noise*. What form does 'noise' take in telephone calls?

A model of data transmission is shown below.



Normally, the message is encoded, the signal transmitted to the receiver, and then decoded with a received message. It is in the transmission that noise can affect the signal.

The Mariner 9 spacecraft in 1971 sent television pictures of the planet Mars across a distance of 84 million miles. Despite a very low power transmitter, the space-probe managed to send data which eventually resulted in very high quality pictures being shown on our screens. This was in part largely due to the sophisticated coding system used.

As a very simple example, consider a code which has just four *codewords*:

$$C = \{(0\,0),\ (0\,1),\ (1\,0),\ (1\,1)\}$$

Each codeword has *length* 2, and all digits are either 0 or 1. Such codes are called *Binary Codes*.

Activity 1

Could you detect an error in the transmission of any of these codewords?

One way to detect an error, would be to repeat each codeword, giving a new code

$$C_1 = \{(0000),\ (0101), (1010), (1111)\}$$

Here each pair of digits is repeated.

Example 1

Can Code C_1 detect a single error?

Solution

For example, if the codeword (0 1 0 1) was corrupted to (1 1 0 1) it is clear that an error can be detected, as (1 1 0 1) is *not* one of the codewords. So, yes, C_2 can detect a single error.

Example 2

Can a single error in a codeword be corrected?

Solution

This is not so straightforward to answer since, for example, (1 1 0 1) could have also been (1 1 1 1) with one error, as well as (0 1 0 1). So this code can detect a single error but cannot correct it.

It should also be added that the *efficiency* (or rate) of this code is given by

$$\frac{\text{number of original message bits}}{\text{length of codeword}} = \frac{2}{4} = \frac{1}{2}$$

since each codeword in the original message had only two digits (called *bits*).

Activity 2

Consider a code designed to specify one of four possible directions:

up	*down*	*left*	*right*
(0 0 0)	(1 1 0)	(0 1 1)	(1 0 1)

Can this code detect any single error made during the transmission of a codeword? Can it correct the error?

Often codes include a *parity check* so that, for example, the code C is transformed to C_2 as shown below.

C	C_2
0 0	0 0 0
0 1	0 1 1
1 0	1 0 1
1 1	1 1 0

The extra last digit in C_2 is 0 if the sum of the digits modulo 2 is zero or even, or 1 if odd. (Modulo 2 means $0+0=0$, $0+1=1$, but $1+1=0$, etc.)

Example 3

Can Code C_2 detect errors now?

Solution

Yes it can, as any single error in a codeword is no longer a codeword. For example, if 0 0 1 is received instead of 0 0 0, then there is clearly an error.

Using the previous definition, the efficiency of Code C_2 is $\frac{2}{3}$.

None of the codes considered so far can correct errors.

Activity 3

Design a code containing 4 codewords, each of length 5, which can detect and correct a single error.

5.2 Error Correction

It is clear that codes which can not only detect, but also correct errors are of far greater use than those that can only detect – but the efficiency will decrease, since extra essentially redundant information will have to be transmitted.

For example, here is a code that can be used to identify four directions:

up	*down*	*left*	*right*
(0 0 0 0 0 0)	(1 1 1 0 0 0)	(0 0 1 1 1 0)	(1 1 0 0 1 1)

The length of each codeword is 6, but since the number of message bits is essentially 2, i.e. the code could consist of

(0 0), (1 1), (0 1), (1 0)

its efficiency is $\frac{2}{6}=\frac{1}{3}$. But, as you see, it can now *correct* single errors.

Activity 4

The following codewords have been received using the code above. Assuming that only one error has been made in the transmission of each codeword, determine, if possible, the actual codeword transmitted:

(a) (1 0 0 0 0 0) (b) (1 1 0 0 0 0) (c) (0 1 0 0 1 1)

Example 4

Can the code above detect if 2 errors have been made in the transmission of a codeword?

Solution

If two errors are made, for example 1 1 0 0 0 0 is transmitted instead of 0 0 0 0 0 0, this is not identifiable. Indeed, you would assume that only *one* error had been made and that the actual codeword was 1 1 1 0 0 0.

Activity 5 Codes

Consider Code 5 given in the Appendix. Find out how many errors this code can detect and correct by considering, for example, codewords such as

(a) (1 1 0 0 0 0 0) (b) (0 1 1 1 1 1 1) (c) (1 0 0 0 1 0 0)

which are in error.

By now you should be beginning to get a feel for what is the important characteristic of a code for the determination of the number of errors that can be detected and corrected. The crucial concept is that of *distance* between codewords.

The *distance* (*d*) between any two codewords in a code is defined as the sum of the number of actual differences between the digits in the codewords; for example

$$d\left((1\,1\,1),\ (0\,1\,0)\right) = 2$$

whilst

$$d\left((0\,1\,0\,1),\ (1\,0\,1\,1)\right) = 3$$

The *Hamming distance* is defined for a BINARY CODE (in which the digits are either 0 or 1) as the *minimum* distance between any two codewords in the code and is usually denoted by δ (the letter 'delta' from the Greek alphabet).

Example 5

Determine the Hamming distance for the code with codewords

(1 1 0 0 0), (0 0 1 0 1), (1 0 1 0 1), (1 1 1 1 1)

Solution

You must first find distances between all the codewords.

$$d\left((11000),\ (00101)\right) = 4$$

$$d\left((11000),\ (10101)\right) = 3$$

$$d\left((11000),\ (11111)\right) = 4$$

$$d\left((00101),\ (10101)\right) = 1$$ $\leftarrow$ Hamming distance $\delta = 1$ (minimum of 1, 2, 3 and 4)

$$d\left((00101),\ (11111)\right) = 3$$

$$d\left((10101),\ (11111)\right) = 2$$

Why is the Hamming distance crucial for error detection and correction?

Activity 6 Hamming distance

Determine the Hamming distance for

Codes 1, 2, 3, 4 and *5*

given in the Appendix.

To try to see the connection between the Hamming distance, δ, and the number of errors that can be detected or corrected, in the next Activity you will consider Codes 1 to 5 from the Appendix.

Activity 7

Copy and complete this table.

		Errors	
Code	*Hamming distance*	*corrected*	*detected*
1	2	0	1
2	3	1	1
3	...	...	...
4	...	...	...
5	...	...	...

Also add on to the table any other codes considered so far. Can you see a pattern?

The first thing that you probably noticed about the data in the table is that the results are different depending on whether *n* is even or odd. It looks as if, for

$\delta = 2$, you can detect 1 error but correct 0 errors,

$\delta = 3$, you can detect 1 error and correct 1 error.

Example 6

How do you think the pattern continues for $\delta = 4$ and $\delta = 5$?

Solution

For $\delta = 4$, you can detect 2 errors but are only able to correct 1 error; for $\delta = 5$, you can both detect and correct 2 errors.

Activity 8

Construct a code for which $\delta = 4$. How many errors can it detect or correct? Similarly, construct a code for which $\delta = 5$ and again determine how many errors it can detect or correct. Can you suggest a generalisation of the results?

As can be seen from Activities 6 and 7, there is a distinct pattern emerging. For

δ odd, the code can correct and detect up to $\frac{1}{2}(\delta - 1)$ errors.

δ even, the code can correct up to $\frac{1}{2}(\delta - 2)$ errors, and detect up to $\frac{1}{2}\delta$ errors.

Activity 9 Validating the results

Check that the above result holds for Code 9 in the Appendix.

Exercise 1

1. *The '2 out of 5' code consists of all possible words of length 5 which have exactly two 1s; for example, (1 0 1 0 0) belongs to the code but (1 1 0 1 0) does not.*
   List all possible codewords and explain why this code is particularly useful for the transmission of numeric data. What is the Hamming distance for this code?

2. *Analyse the '3 out of 7' code, defined in a similar way to the '2 out of 5' code in question 1. Determine its Hamming distance and hence find out how many words it can detect and correct.*

3. *Determine the Hamming distance for Code 7 in the Appendix. Hence find out how many errors this code can detect and correct.*

Appendix Codes

Code 1

0 0 0 0
0 0 1 1
0 1 0 1
0 1 1 0
1 0 0 1
1 0 1 0
1 1 0 0
1 1 1 1

Code 2

0 0 0 0 0 0
0 0 1 1 1 0
0 1 0 1 0 1
0 1 1 0 1 1
1 0 0 0 1 1
1 0 1 1 0 1
1 1 0 1 1 0
1 1 1 0 0 0

Code 3

0 0 0 0
0 1 0 1
1 0 1 0
1 1 1 1

Code 4

0 0 0 0
1 1 0 0
0 0 1 1
1 1 1 1

Code 5

0 0 0 0 0 0 0
0 0 1 1 1 0 1
0 1 0 1 0 1 1
0 1 1 0 1 1 0
1 0 0 0 1 1 1
1 0 1 1 0 1 0
1 1 0 1 1 0 0
1 1 1 0 0 0 1

Code 6

0 0 0 0 0 0 0
1 1 0 1 0 0 1
0 1 0 1 0 1 0
1 0 0 0 0 1 1
1 0 0 1 1 0 0
0 1 0 0 1 0 1
1 1 0 0 1 1 0
0 0 0 1 1 1 1
1 1 1 0 0 0 0
0 0 1 1 0 0 1
1 0 1 1 0 1 0
0 1 1 0 0 1 1
0 1 1 1 1 0 0
1 0 1 0 1 0 1
0 0 1 0 1 1 0
1 1 1 1 1 1 1

Code 7

0 0 0 0 0 0 0 0
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 1 1 0 0 1 1 0
0 1 0 1 1 0 1 0
0 0 1 1 1 1 0 0
0 1 1 0 1 0 0 1
1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0
1 1 0 0 1 1 0 0
1 0 1 0 1 0 1 0
1 0 0 1 1 0 0 1
1 0 1 0 0 1 0 1
1 1 0 0 0 0 1 1
1 0 0 1 0 1 1 0

Code 8

0 0 0 0 0 0 0
0 0 1 0 1 1 1
0 1 0 0 1 1 0
0 1 1 0 0 0 1
1 0 0 0 1 0 1
1 0 1 0 0 1 0
1 1 0 0 0 1 1
1 1 1 0 1 0 0
0 0 0 1 0 1 1
0 0 1 1 1 0 0
0 1 0 1 1 0 1
0 1 1 1 0 1 0
1 0 0 1 1 1 0
1 0 1 1 0 0 1
1 1 0 1 0 0 0
1 1 1 1 1 1 1

Code 9

0 1 0 1 0 1 0
1 0 0 1 1 0 0
0 0 1 1 0 0 1
1 1 1 0 0 0 0
0 1 0 0 1 0 1
1 0 0 0 0 1 1
0 0 1 0 1 1 0