This first lecture is an informal introduction to the aims and ideas of coding theory.


The main goal of coding theory is to develop techniques which permit the detection of errors in the transmission of information and, if necessary, allow the original message to be reconstructed. This young subject is both highly applicable in practice and algebraically elegant in theory.

## 1 The Initial Problem

The following diagram represents a typical communication system symbolically. Possible examples include a telephone line, a satellite communication link, a face to face conversation between individuals, or viewing a picture over the internet.

An encoded message is transmitted through the channel, but is subject to some distortion or noise. There are two basic questions:

- Can we check whether any errors have occurred?
- Can we correct any errors that have occurred?

Clearly, these questions are not reasonable unless we have some estimate of the level of noise, and unless the noise is fairly low we have little chance of being able to answer yes. Later on we shall consider specific techniques for encoding and decoding; we do not do this now.

**Example 1.** Message YES/NO. Code: YES = 00000, NO = 11111.

Say the received vector is 01001. Our conclusion:

- There is certainly an error in the received vector.
- Provided that we know that the noise is not too bad, then the most likely correct message is YES—because this would involve the fewest number of errors in the digits.
In the sequel such codes are called repetition codes; in general they are very inefficient, i.e. we can do much better for the number of digits sent.

**Notation:** Throughout this course, \( p \) is a prime number and \( q \) is a power of \( p \). \( \mathbb{F}_p \) and \( \mathbb{F}_q \) are the finite fields with \( p \) and \( q \) elements respectively, and \( \mathbb{Z} \) is the integers. We shall mainly be concerned with \( \mathbb{F}_p \), so recall that

\[
\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z} = \mathbb{Z} \mod p,
\]

the integers modulo \( p \).

For our purposes, a \( p \)-ary code \( C \) of length \( n \) is a subset of

\[
\mathbb{F}_p^{(n)} = \mathbb{F}_p \times \cdots \times \mathbb{F}_p,
\]

the set of ordered \( n \)-tuples of integers modulo \( p \). \( \mathbb{F}_p \) is then called the alphabet of the code \( C \); we shall often write simply \( F \), where the \( p \) is understood.

**Example 2.** The repetition code of Example 1 is a binary (2-ary) code of length 5.

Most books allow alphabets other than finite fields, e.g. \( \mathbb{Z}/26\mathbb{Z} \); we shall not require them, so we shall stick to the more restrictive definition in the interests of simplicity and algebraic elegance.

In the light of our definition, it is worth emphasising that a \( p \)-ary codeword of length \( n \) is simply a vector \( \mathbf{x} = (x_1, \ldots, x_n) \) with the \( x_i \in \mathbb{F}_p \) (that is, a member of the \( n \)-dimensional vector space over \( \mathbb{F}_p \)) and with the property that \( \mathbf{x} \) is a member of the code; that is,

\[
\mathbf{x} \in C \subseteq \mathbb{F}_p^{(n)}.
\]

Next we consider a more instructive example.

**Example 3.** We wish to pass on a direction command (NWES). We consider various attempts at a good code.

**Attempt 1.** The shortest code would be something like:

\[
C_1 = \left\{ \begin{array}{ccc}
0 & 0 & N \\
0 & 1 & W \\
1 & 0 & E \\
1 & 1 & S
\end{array} \right\}
\]

So \( C_1 = \mathbb{F}_2^{(2)} \).

This may be easy, but there are two problems:

- We cannot tell whether the received vector has any errors, since every possible vector is a codeword.
- Even if we somehow knew that there was an error, we would still have no hope of finding the original codeword.

Here the main problem is that all the words are codewords.
Attempt 2.

\[
C_2 = \begin{cases}
0 & 0 & 0 & N \\
0 & 1 & 1 & W \\
1 & 0 & 1 & E \\
1 & 1 & 0 & S
\end{cases}
\]

This \( C_2 \) is obtained from \( C_1 \) by adding a parity check; the third digit is the sum in \( \mathbb{F}_2 \) of the other two digits, so that the sum of all three digits is always zero (even). This code does rather better:

- If there is one error, it will be detected by the odd sum of the digits.
- However, there is no way to recover the word in this event; if the received vector is \((0,1,0)\), say, and we know that there has been exactly one error, the word sent is equally likely to have been \( N \) as \( W \), since they both differ in one place from the word received.

Attempt 3.

\[
C_3 = \begin{cases}
0 & 0 & 0 & 0 & 0 & N \\
0 & 1 & 1 & 0 & 1 & W \\
1 & 0 & 1 & 1 & 0 & E \\
1 & 1 & 0 & 1 & 1 & S
\end{cases}
\]

Here we repeat \( C_1 \) after the parity check.

- Clearly this retains the ability to spot exactly one error, since then either the sum of the first three digits is odd or the sum of the last three digits is odd.
- It can be checked, although somewhat laboriously at this stage, that given a vector which has suffered at most one error, we can correct it and recover the original codeword. For example,

<table>
<thead>
<tr>
<th>received vector</th>
<th>corrected vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bar{v} = 00100 )</td>
<td>00000 = ( N )</td>
</tr>
<tr>
<td>( \bar{v} = 11111 )</td>
<td>11011 = ( S )</td>
</tr>
</tbody>
</table>

This is because there is at most one codeword differing in zero or one places from any possible received word.

This concludes our informal introduction; from now on we shall develop a more structured approach to coding.