Encoding and Decoding with the Hamming Code

The one shortfall of our previous presentation of the Hamming code is the apparent need to look up codewords in a long list. We will soon see that the encoding and decoding can be done much more efficiently than this, using some simple linear algebra. This is significant since if error-correcting codes are to be useful, they should not only allow for error correction in principle, as well as having codewords as short as possible; they should also have simple encoding and decoding algorithms. This means that it should be possible for simple electronic circuits, implemented on silicon chips perhaps, to perform the encoding and decoding easily and in 'real time'.

Matrix Multiplication

The linear algebra we need to understand the encoding and decoding processes involves matrices. An *m*×*n matrix* is simply an array of numbers, having *m* rows and *n* columns, usually enclosed in brackets or parentheses; thus for example

$$A = \begin{bmatrix} 3 & 4 & -2 \\ 2 & -1 & 0 \end{bmatrix}$$

is a 2×3 matrix. It has six *entries* 3, 4, …, 0 which are located by row and column number; for example the (1,3)-entry of A is −2.

How do we multiply two matrices? Consider a 3×2 matrix

$$B = \begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 5 & 0 \end{bmatrix}.$$

The product of these two matrices is the 2×2 matrix

$$AB = \begin{bmatrix} 3 & 4 & -2 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 5 & 0 \end{bmatrix} = \begin{bmatrix} -11 & 18 \\ 3 & 1 \end{bmatrix}.$$

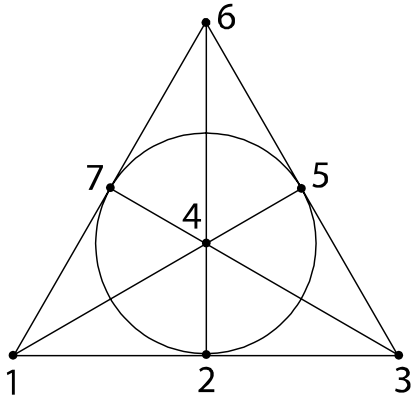
Note that the (i,j)-entry of AB is the dot product of the ith row of A with the jth row of B; for example the (2,1)-entry of AB is $2\times2 + (-1)\times(-1) + 0\times5 = 3$. Note however that the product BA is different from AB:

$$BA = \begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 5 & 0 \end{bmatrix} \begin{bmatrix} 3 & 4 & -2 \\ 2 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 7 & 2 & -2 \\ 3 & -7 & 2 \\ 15 & 20 & -10 \end{bmatrix}.$$

The product of an $m \times n$ matrix with an $n \times p$ matrix, will always give an $m \times p$ matrix. Each entry will be found by taking the dot product of two vectors of length n. The product of two matrices is not defined unless the number of columns in the first matrix, equals the number of rows in the second matrix. Although matrix multiplication is not commutative in general (we have seen an example where $AB \neq BA$), it is always associative: $(AB)C = A(BC)$ whenever the matrix products are defined (i.e. the number of columns of A equals the number of rows of B, and the number of columns of B equals the number of rows of C).

The (Revised) Projective Plane and Hamming Code

Here is a revised description of the projective plane of order two. We have given up the 'cyclic shift' description of lines (the evident pattern of lines {1,2,4}, {2,3,5}, {3,4,6}, etc. in the previous version). The present revised version allows for a simplified decoding algorithm using syndromes, as we describe below.



The Projective Plane of Order Two

Once again the Hamming code is constructed from the projective plane; for example the line {3,5,6} gives rise to the codewords 0010110 and 1101001. The revised Hamming code is listed below:

Table B: The Revised Hamming Code

Msg. No.	Message Text	Revised Hamming Codeword
0	**0000**	**0000000**
1	**0001**	**0001111**
2	**0010**	**0010110**
3	**0011**	**0011001**
4	**0100**	**0100101**
5	**0101**	**0101010**
6	**0110**	**0110011**
7	**0111**	**0111100**

Msg. No.	Message Text	Revised Hamming Codeword
8	**1000**	**1000011**
9	**1001**	**1001100**
10	**1010**	**1010101**
11	**1011**	**1011010**
12	**1100**	**1100110**
13	**1101**	**1101001**
14	**1110**	**1110000**
15	**1111**	**1111111**

Hamming Encoding and Decoding using Matrices

Encoding and decoding with the Hamming code is accomplished using matrix multiplication *modulo 2*: here the only constants are 0 and 1, using the addition and multiplication tables supplied here:

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

For encoding we use the 4×7 matrix

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

A message word x of length 4 may be considered as a vector of length 4, or equivalently, a 1×4 matrix. The codeword corresponding to x is then simply xG, which is a 1×7 matrix, or simply a bitstring of length 7. For example the message $x = 1101$ is encoded as

$$xG = [1 \quad 1 \quad 0 \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = [1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1].$$

Note that this gives the same answer as our table, namely 1101001, for the codeword corresponding to the message word 1101. The point is that matrix multiplication is easier to implement in an electronic circuit, and requires less real time to implement, than lookup in a list such as Table B. Moreover this gives us insight into the structure of the Hamming code, using the tools of linear algebra.

How can we efficiently decode? If a word y of length 7 is received, we anticipate first checking to see if y is a codeword; if so, the original message is recovered as the first 4 bits of y. But how do we check to see if y is in the code without performing a cost-intensive search through Table B? Our answer uses the 3×7 check matrix

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

Consider the Hamming codeword 1101001, which we denote by y, thus:

$$y = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Note that we have written *y* as a column vector (i.e. as a 7×1 matrix) rather than as a row vector (i.e. a 1×7 matrix). Now the matrix product

$$Hy = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

gives the zero vector, which is our evidence that *y* is a valid codeword, and so we take its first four characters 1101 to recover the original message word.

What if *y* had suffered from a single bit error during transmission? Suppose that its third bit had been altered, so that instead of *y*, we receive the word

$$y' = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

The bit error would be detected by computing the matrix product

$$Hy' = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}.$$

Since the result is not the zero vector, this alerts us to the fact that *y′* is not a valid codeword. This alerts us to the presence of a bit error, and we assume that only one bit error occurred during transmission. But how can we tell which of the seven bits of *y′* is in error? Simply: the vector above is the word 011, which is the binary representation of the number 3; this tells us that the third bit of *y′* is erroneous. Switching it recovers the valid codeword 1101001, then taking the first four bits recovers the codeword 1101.

The vector *Hy* is called the *syndrome* (or *error syndrome*) of the vector *y*. If the syndrome is zero, then y is a codeword; otherwise the syndrome represents one of the integers 1,2,…,7 in binary, and this tells us which of the seven bits of *y* to switch to recover a valid Hamming codeword from *y*.