swissQuant
Mastering Complexity

Mathematical Challenge April 2013

Error-correcting codes

References:

- Selected Unsolved Problems in Coding Theory, David Joyner/Jon-Lark Kim, 2011
- Error Correcting Codes, Hackers Delight, http://hackersdelight.org/ecc.pdf

Description:

In today's digital environment, proper transmission of data is crucial, however most of the transmissions are sent through noisy channels. For example, consider a CD player reading from a scratched music CD, or a wireless cell phone capturing a weak signal from a relay tower which is too far away. These situations give rise to problems in communication which must be solved if one is to transmit information reliably. For the following discussion, let us assume that all the messages are sent through a memoryless binary symmetric channel.

In a memoryless binary symmetric channel, all messages consist of 0s and 1s. Due to noise, the (symmetric) probability that a 0 or a 1 is correctly received is 1-p and the probability that a 0 or a 1 is incorrectly received is p. The channel is also assumed to be memoryless i.e. the probability of an error in transmitting a bit does not depend on any previous transmission.

Let us now assume we have to transmit the decimals 0-15 over such a channel. The binary representation would be:

Decimal	Binary Data
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

If we now would just transmit the binary data, the receiving instance can't decide if these 4 bits are actually the "true" data bits of the sender or if one or more bits are changed by noise. For example, if 0000 was sent and the last bit flipped, the sender would translate the data bits to decimal1 instead of decimal 0.

The first improvement to plain transmission was already done by Jewish scribes beginning before Christ. They noted the sum of words on a line and page. The page was thrown out if a single mistake was found.

The same approach can be used for binary transmissions by adding a parity bit at the end of the message. The parity bit is set to 1 if the number of 1s in the string is even or set to 0 if the number of 1s is odd. If the sender wants to send 0101, he has to send 01011. This allows the receiver to detect an odd amount of changes in the string by summing up the 1s in the received string and comparing it to the last bit, but it doesn't allow for error correction. (Actually, the parity bit is called a 1-error detecting code).

Decimal	Binary Data	Repetition
0	0000	000000000000
1	0001	000000000111
2	0010	000000111000
3	0011	000000111111
4	0100	000111000000
5	0101	000111000111
6	0110	000111111000
7	0111	000111111111
8	1000	111000000000
9	1001	111000000111
10	1010	111000111000
11	1011	111000111111
12	1100	111111000000
13	1101	111111000111
14	1110	111111111000
15	1111	111111111111

We've seen that the parity bit is a 1-error detecting code. What if we want a 1-error correcting code?

In the table on the left, we see the simplest 1-error correcting code called "Repetition". The sender simply repeats every bit 3 times. In case of one flipped bit, it is clear what the original data bit was (i.e. if we receive 011 as the first 3 bits, we can correct the code to 111).

Obviously, this is only true in the case of one flipped bit. If two bits (in the same block) are flipped, the receiver can at least detect the error but would correct it wrong (i.e. if we receive 001 we know that there was an error during transmission, but we correct it wrong to 000 even if the original probably was 111).

Additionally, this error correction is tripling the amount of transmitted data and is therefore highly suboptimal.

In the late 1940's Richard Hamming (a mathematician at the Bell labs) thought that computers should be able to correct bit errors and discovered an infinite family of 1-error correcting codes. Hamming codes are special in that they are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimal distance 3.

In mathematical terms, Hamming codes are a class of binary linear codes. For each integer $r \geq 2$ there is a code with *block length* $n=2^r-1$ and message length $k=2^r-r-1$. Hence the *rate* of Hamming codes is $R=k/n=1-r/(2^r-1)$, which is the highest possible for codes with minimal distance 3 and block length 2^r-1. The Repetition example above has 4 data bits (**k=4**), 12 bits block length (**n=12**) and a minimal distance of 3 (i.e. the minimal number of bit changes needed to go from any code word to any other code word is 3). The rate is therefore 0.33. The following Hamming code (the so called Hamming [7,4,3] code for **n=7**, **k=4** and minimum distance **d=3**) has a rate of 0.57 which is reducing the amount of sent data (compared to the repetition example) by 58% while providing the same error correction capabilities (1-error correcting code).

The Hamming [7,4,3] code can be described with an “encoding” map from $\mathbf{F}^4$ to **C** given by $\phi(\mathbf{x}) = \mathbf{y}$, where **C** is the set of all code words for 4 data bits **x** generated by the function $\phi(\mathbf{x})$:

$$y = \begin{pmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} \bmod(2) = \phi(x)$$

Additionally, the received vector of bits **v** can be tested and corrected by the matrix **H** with **mod(H*v,2)**.

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

If the check calculates (1,1,1), all parity bits are wrong and therefore position 1 is erroneous and can be corrected. For (0,1,1) its position 2, for (1,0,1) it is position 3, for (1,1,0) it is position 4 and if (1,0,0) or (0,1,0) or (0,0,1) position 5/6/7 are wrong.

Question 1 (medium): Provide the “encoding” map and the **H** matrix for a Hamming [12,8,3] code.

Question 2 (hard): For **n=7**, **k=4** and minimum distance **d=3** provide the best (rate) 2-error correcting code.

Question 3 (very hard): Up to this point we have asked, “Given a number of data bits **k** and a desired minimum distance **d**, how many parity bits are required?”. We can turn this around and ask “For a given code length **n** and a minimum distance **d**, how many code words are possible?”. For minimum distance 1, this is quite easy (pure permutation). Let **A(n, d)** denote the largest possible code size for a binary code with length **n** and minimum distance **d**:

$A(n,1) = 2^n$

But, can you provide a formula for **A(n,3)**?