Error-Control Coding

Error-Control Coding and the Binomial Distribution

Hans Georg Schaathun

Session 23 January 2015
Updated 20th January 2015

1 Error-Control Coding

To reduce the error probability in communications, we use error-correcting codes, as depicted in Figure 1. The k-bit message word $\mathbf{m}$ is encoded as an n-bit codeword $\mathbf{c}$. Because of noise on the Channel, the received word $\mathbf{r}$ may or may not be equal to $\mathbf{c}$. The decoder function aims to recover the original message $\mathbf{m}$ by returning an estimate $\hat{\mathbf{m}}$. In general the probability that $\hat{\mathbf{m}} \neq \mathbf{m}$ should be much smaller than the probability that $\mathbf{r} \neq \mathbf{c}$.

On the module web page, you can find matlab implementations of encoder/decoder for the $[7, 4, 3]$ Hamming code. The encoder takes a four-bit word in, and returns a seven-bit word out. Conversly, the decoder decodes a seven-bit word into a four-bit word.

Exercise 1 *Download the encoder/decoder functions for the Hamming code and test them in Matlab.*

1. *Generate a random four-bit word* $\mathbf{m}$.
2. *Encode the word to get* $\mathbf{c}$. *What does it look like?*
3. *Decode* $\mathbf{c}$ *to get* $\hat{\mathbf{m}}$. *Is* $\hat{\mathbf{m}}$ *equal to* $\mathbf{m}$ *or not?*

Figure 1: Channel with coding.

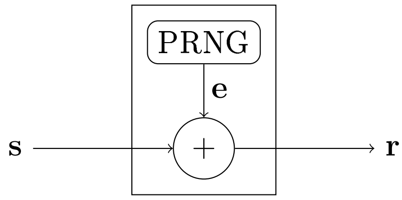

Figure 2: The binary symmetric channel. PRNG stands for Pseudo-Random Number Generator.

2 A Channel Simulator

We want to simulate the BSC, which can be viewed as a probabilistic function as shown in Figure 2. In the first session we did not actually implement the channel; all we did was to draw the random error vector $\mathbf{e}$ and count the bit errors (Hamming weight). In the presence of error-control coding, this is no longer sufficient.

Exercise 2 *Implement an m-file with a function simulating the BSC(p). It needs two input arguments, the channel parameter p and the input word $\mathbf{s}$.*

3 Testing the Hamming Code

We want to compare the error distributions in two scenarios:

1. Sending four-bit words on a BSC (Figure 2).
2. Encoding four-bit words with the Hamming to before they are sent on the BSC (Figure 1).

The figures give the models for the system to be simulated.

Exercise 3 *Write a function to make one trial (with one word) of the communication system with the Hamming code and count the number of bit errors at the receiver. I other words, we need a function which takes the channel probability as input and*

- *Generates a random message $\mathbf{m}$.*
- *Encodes the message to get a codeword $\mathbf{c}$.*
- *Generates a error word $\mathbf{e}$.*
- *Calculates the received word $\mathbf{r} = \mathbf{c} \oplus \mathbf{e}$.*
- *Decodes $\mathbf{r}$ to get $\hat{\mathbf{m}}$.*
- *Compare the $\mathbf{m}$ and $\hat{\mathbf{m}}$. The number of errors is given as $w(\mathbf{m} \oplus \hat{\mathbf{m}}$.*

Test the function.

Exercise 4 *Run the function from the previous exercise 100 times and record the error counts for each run in a vector. The number of bit errors from the decoder is a stochastic variable X.*

- *Tabulate the empirical probability distribution of X.*
- *Compare this distribution to the distribution you found in the first session, for communication without coding.*
- *Judging from your data, does X appear to be binomially distributed?*

4 Word errors

Sometimes we are more interested in word errors than in bit errors. A word error event occurs when there is at least one bit error.

Exercise 5 *Review the results from the simulation in Exercise 4. How many word errors did you have in 100 trials?*

Exercise 6 *Review the simulation results for the uncoded system in the first session. How many word errors did you get in 100 trials without ECC?*

Exercise 7 *Let X be the number of word errors after the transmission of m independent words on the BSC. What probability distribution does X have?*