ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Signal and Information
Processing Laboratory
Institut für Signal- und
Informationsverarbeitung

Fachpraktikum Signalverarbeitung

SV5: Error Correcting Codes

1 Introduction

Error detection and correction are an essential part of any communication system and data storage device. Some examples are magnetic drives, CDs and mobile phones. Error correcting codes are used to protect information: if, for some reasons, data bits are missing or corrupted, the use of error correcting codes allows to still recover the information.
After studying the typical structure of a code and defining some common parameters, we investigate the error detection and correction capabilities of some example codes. In this session, we focus on two families of linear codes: Hamming codes and Reed-Solomon codes.

2 Generalities on Error Correcting Codes

Consider the following scenario: We are given a sequence of information symbols $b = b_1, b_2, \ldots$ which are to be protected from errors in, for example, a communication or storage system.

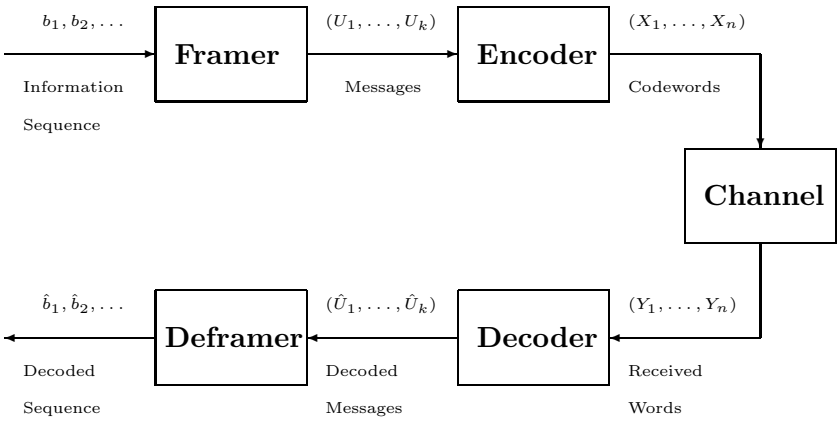

Figure 1: System Diagram

Figure 1 shows the information flow of a typical system that uses an error correcting code.

- First, at the transmitter, the *framer* splits the information sequence $b_1, b_2, \ldots$ and maps it using a suitable alphabet and outputs the *message* $U = (U_1, \ldots, U_k)$ of

length k with $U_i \in \mathcal{A}$. Example of finite alphabet $\mathcal{A}$ are $\{0,1\}$ or $\mathbb{Z}_p$ (integers modulo p).

- Then the message $U = (U_1, \ldots, U_k)$ is *encoded* using an error correcting code (n, k) into a *codeword* $(X_1, \ldots, X_n)$ of length n with $X_i \in \mathcal{A}$. Note that the same alphabet $\mathcal{A}$ is used. In order to protect the data, the encoder should add some redundancy therefore n has to be greater than k. The amount of redundancy of a code is characterized by its rate $R = k/n$. Codes are efficient if they can correct many errors while having a high rate. Figure 2 shows two examples of an encoder. The encoder maps a k-symbol message into a codeword of n symbols in the same alphabet. In Example-1, the message is drawn from the alphabet $\{0,1\}$. The encoder maps the $k = 3$ bit message into an $n = 5$ bit codeword. In Example-2, the message is drawn from the alphabet $\{A, B, C, \ldots, Z\}$. The codeword **'L D A F Z E Y'** has redundant symbols **'D F E'** in the 2nd, 4th and 6th position.

- A codeword $(X_1, \ldots, X_n)$ is then transmitted over an unreliable channel. The channel delivers $(Y_1, \ldots, Y_n)$ which is a *noisy version* of the codewords to the receiver. For the purpose of this experiment, the Y_i's belong to the same alphabet $\mathcal{A}$.

- The decoder performs the reverse process of the encoder in order to recover the original message. The decoded message is denoted $\hat{U} = (\hat{U}_1, \ldots, \hat{U}_k)$. Note that the received word may be corrupted too much such that the decoder might fail to recover the original message that is to say $\hat{U} \neq U$.

- Finally, the deframer remaps and reorganizes the decoded message to recover the estimated information sequence $\hat{b}_1, \hat{b}_2, \ldots$.

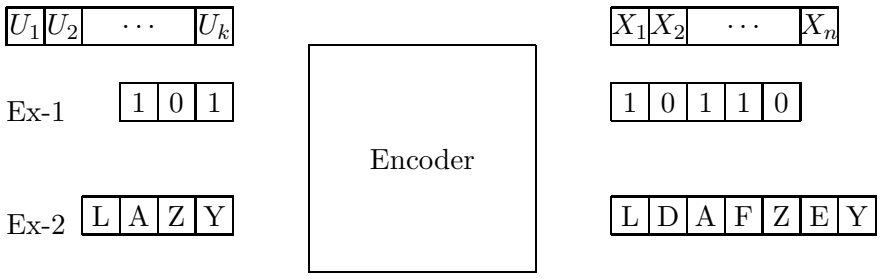

Figure 2: Encoding Overview

In this experiment, we are mainly interested in the encoder and decoder. A *code* C usually refers to the set of possible codewords at the output of an encoder. A (n, k) code C is a subset of $\mathcal{A}^n$ which contains $|\mathcal{A}|^k$ elements. A *codeword* is a specific element of the code.

3 Experiments

Copy the working files to your home directory:
`cp -irL /home/isistaff/glf/fachprak_isi/SV5 ./`
Start Matlab from a shell in the `SV5` folder by using the command `matlab &`.

3.1 A simple linear code: (7,4) Hamming Code

A very simple binary linear code is the $n = 7$, $k = 4$ Hamming code that can correct a single error in a codeword. Let us try some simple experiments with the Hamming Code. For comparison purpose, you may like to keep notes on the parameters we use.

Tip: Write down your results on a piece of paper for later comparison.

3.1.1 Know the Code

1. Run *hamming_list*. A list of all the messages and codewords of a Hamming Code is displayed. Note that the input message appears in the last four bits of the corresponding codeword. What are the 3 remaining bits used for?

2. Check that the code is linear that is to say, in the binary case, for two codewords c_1 and c_2, $b = c_1 + c_2$ is also a codeword. As we manipulate bits, the usual symbol operations are modulo 2.

3. As the code is linear, the encoder mapping can be elegantly described using the generator matrix G. Find out how and express the dimensions of G in terms of n and k.

4. A second matrix H is displayed in the output of *hamming_list*. After expressing the dimensions of H in terms of n and k, for each codeword c, compute with Matlab the product Hc^T (use the variables already defined). Compute the same quantity for some element x of $\{0,1\}^n$ that does not belong to the Hamming code. What can you observe? This matrix is called the parity check matrix. Explain why.

5. Give two mathematical characterizations of this code C using on the one hand the G matrix and on the other hand the H matrix. Note that these descriptions characterize linear codes in general and not only Hamming codes.

6. Let $p > 2$ be an integer. A Hamming code is defined by its parity check matrix H which contains all the binary tuples of size p in its columns except for the all zero one. Check this definition for H in *hamming_list*. Express k, n, and the rate R in terms of p.

3.1.2 Detecting and Correcting Errors

A code is useful if it is able to take advantage of the redundant bits to detect and correct errors. Let $y \in \{0,1\}^n$ be the received word that may not be a codeword. y may have been corrupted in some places (i.e., 1 becomes a 0 and vice versa).

7. How can you detect if $y \notin C$?

In order to correct errors, it is desirable that any pair of different codewords c_1, c_2 are as far as possible from each other. Indeed the more two codewords differ, the more errors can be introduced by the channel without confusing the two codewords. A good measure of distance between two codewords is the *Hamming distance* $d_H(c_1, c_2)$ which is defined as the number of elements that differ between c_1 and c_2.

8. Let C be a code such that the substraction between codewords is defined. Prove that:

$$d_H(c_1, c_2) = w_H(c_1 - c_2) \ ,$$

where $w_H(b)$ is the number of non-zero elements in b.

We define the *minimum Hamming distance* d_{min} as follows:

$$d_{min} := \min_{c_1, c_2 \in C : c_1 \neq c_2} d_H(c_1, c_2) \ .$$

9. For linear codes, prove that:

$$d_{min} = \min_{c \in C : c \neq 0} w_H(c) \ .$$

Then, compute the *minimum Hamming distance* d_{min} of our Hamming Code. Note that the simplification saved us a lot of time.

The (7,4) Hamming Code is listed in Appendix-A

The decoder strategy is to associate the received word y to the closest codeword c:

$$\hat{c}(y) := \arg\min_{c \in C} d_H(c, y) \ . \tag{1}$$

This is the *minimum Hamming distance decoder*.

10. **Correcting One Error** Run *hamming_ec*. Choose 1 error. A corrupted codeword is shown, which has an error in 1 position. Try to use the list in Appendix-A to decode the corrupted codeword using strategy (1). Then press any key on Matlab to see the result. It outputs a *decoded codeword* which is the closest codeword in the list to the corrupted codeword.

11. **Correcting Two Errors** Repeat this simulation by introducing 2 errors. Try to decode the corrupted codeword using strategy (1). Unfortunately both your answer and the answer from the program are wrong because the code we have chosen is not capable of correcting more than one error! To convince yourself, compute the Hamming distance between the original codeword and the corrupted word and then between the decoded codeword and the corrupted word. Check your values with the Matlab output. What do you notice from these distances?

12. Explain why this code is not capable of correcting more than one error in a codeword. Use the fact that the minimum Hamming distance d_{min} of a Hamming code is 3.

13. Can you guess the relation between d_{min} and the maximum number n_e of errors a linear code can correct?

14. What about the relation between d_{min} and the maximum number n_d of errors a linear code can detect?

3.1.3 Visualization of Errors

Now let us try the above experiments again, but on a much longer information sequence. We use an image for this purpose.

Tip: For comparison purpose, do not close the figure windows!

15. Run *hamming_rec*. Choose "Fix number of errors per codeword", number of errors in each codeword = 1, image file = *tiger_gray*. The original image (left), the image corrupted with 1 error in every codeword (center) and the corrected image (right) are displayed by Matlab. What can you observe?

16. Repeat the above experiment for the same image, but with two errors in every codeword. What can you conclude? Would have it been better to leave the received word uncorrected?

We have introduced errors in every codeword. In reality, this never happens. Errors may be occurring randomly or in bursts depending on the channel.

17. Run *hamming_rec* in random errors mode. Try different bit error probabilities (e.g 0.001, 0.1, 0.4). Compare the average error probability displayed in Matlab.

3.2 Reed-Solomon Codes

In this section, we introduce Reed-Solomon (RS) codes. These linear codes are used in CDs, DVDs, Bluray and in a plethora of other devices. They are specified over large symbol sets.

3.2.1 A simple RS code over $\mathbb{Z}_7$

Constructing a general Reed-Solomon code requires a deep understanding of algebra theory. We focus on a simple example. Let the alphabet $\mathcal{A}$ be $\mathbb{Z}_7 = \{0, 1, \ldots, 6\}$ with addition and multiplication modulo 7. For our code, we choose $k = 3$ and $n = 6$. The construction start by selecting an element α of $\mathbb{Z}_7$ such that $\alpha^n = 1$ and $\forall i = 1, \ldots, n-1$, $\alpha^i \neq 1$. This element is called primitive n^{th} root of unity. Note that such element exists only for well-chosen alphabet $\mathcal{A}$.

18. Find the minimum suitable α for $\mathbb{Z}_7$ **without using Matlab** for computation. Write down all the powers of α in a table.

Given this α, a RS encoder $(u_1, \ldots, u_k) \mapsto (x_1, \ldots, x_n)$, has the following form:

$$x_j = \sum_{i=1}^{k} u_i \alpha^{(i-1)(j-1)} \ , \ j = 1, \ldots, n \ . \tag{2}$$

19. Prove that this RS code is indeed linear (i.e. for two codewords x and y and for $\lambda \in \mathbb{Z}_7$, $\lambda x + y$ is also a codeword). Express the generator matrix G in terms of α. Does this matrix look familiar?

An important result for RS codes is that their *minimum Hamming distance* is $d_{min} = n - k + 1$, which you can prove by doing the following optional question.

20. **Optional:** We define the polynomial $u(z) = \sum_{i=1}^{k} u_i z^{(i-1)}$. After checking that $x_j = u(\alpha^{(j-1)})$, prove that $d_{min} \geq n - k + 1$. According to the Singleton Bound theorem, every (n, k) linear code must satisfy $d_{min} \leq n - k + 1$. What can you conclude on the minimum Hamming distance of a RS code?

21. Implement the encoder function *encoder_RS7*. Compute and enter the G matrix by hand for this specific case (hint: use your table that contains the powers of α). Explain why it is not a good idea to directly enter the expressions of the formula (2) in Matlab. Test your encoder by running *simu_RS7*.

22. Once your encoder works, experiment it on the picture *tiger_gray* by running *main_RS7* while choosing the number of error per codeword equal one. Then, choose the number of errors equal 2. Did you expect those results? (These two simulations may take few minutes.)

Decoding a RS code is non-trivial. You can have a look at the decoder *PIM_decoder*.

3.2.2 Experimenting with RS codes

Now we experiment with RS codes over bigger alphabets and use the built-in Matlab functions to encode and decode. Symbols are composed of M bits.

23. **RS(255, 147)** In this experiment, we will try a very large RS Code. The code we choose has a capability to correct up to 54 symbol errors! Run *rscode_ec* with $M = 8$, $N = 255$, $T = 54$, number of errors = 54 and image file = *tiger_gray*. The program will introduce errors in 54 symbols in every codeword. Since M=8, the image is split into 8 bit symbols. Then the symbols are grouped into messages of length $K = N - 2T = 147$. Observe the output on Matlab. What is so special about this code? Note that the code rate is nearly the same as the Hamming code in Sec (3.1.3) but this code has much more powerful error correction capabilities. Explain why the comparison between this RS Code and the Hamming code of Sec (3.1.3) is unfair.

24. **RS(255, 147) Burst Errors** Run the same experiment *rscode_ec* as before but choose number of errors = -1. The program will introduce errors in bursts, that is to say, all 54 consecutive symbols are corrupted. The output of the program shows a corrupted image that is nearly erased on one side (corresponding to the burst error in every codeword). The image, nevertheless, is reconstructed perfectly! Explain why.

Before proceeding to the Quiz, you can now experiment with the Matlab simulations.

3.3 Quiz on Error Correcting Codes

Solve the following problems.

25. Let your information sequence be from the English alphabet $\{A, B, \ldots, Z\}$ with the space character. If you have to design a framer which takes groups of 2 letters, and map them to messages of size $K = 4$ in an alphabet $\mathbb{Z}_p$ for some integer p, what is the smallest p you can choose? How would you design this mapper (remember that you should also be able to demap)? As an example, convert the following information sequence (or at least the first two symbols) into messages: **I THINK THEREFORE I AM**

26. Let the symbols belong to a finite alphabet $\mathcal{A}$ with $|\mathcal{A}|$ elements. Let the messages and codewords have lengths of k and n respectively.

    (a) How many different messages are possible? How many different words are possible? How many different codewords do you have to pick among those words?

    (b) How many codes can be constructed?

27. Assume we have a binary linear code $C \subset \{0,1\}^n$ but $C \neq \{0,1\}^n$. Someone claims that a possible parity check matrix of C is $H = (0, 0, \ldots, 0)$ with the justification $Hc^T = 0$ for all $c \in C$. Can you trust this person?

28. Assume we have a binary code. The encoder forms the codeword by appending a parity check bit to the message bits. A parity check bit equals 1 if the sum of the bits of the message is odd else it is 0. Find a generator matrix and a parity check matrix of this code. Compute the rate of this code and the minimum Hamming distance. Explain why this code is not powerful.

29. Assume the messages consist of exactly one bit ($k = 1$). The encoder repeats this bit n times to form the codeword. This is a repetition code. Find a generator matrix and a parity check matrix of this code. Compute its rate and its minimum Hamming distance. Explain why it is highly likely that a minimum Hamming distance decoder will be successful if n is large and the bit error probability is $p < 0.5$. Explain why this code is not efficient.

30. In certain systems, an *interleaver* is added just before transmission (i.e., an additional block between *encoder* and *channel* in Figure 1) and a *deinterleaver* is put after the receiver. If we want to transmit M codewords $X = (x^{(1)}, \ldots, x^{(M)})$ where $x^{(j)} = (x_1^{(j)}, \ldots, x_n^{(j)})$, an interleaver permutes in a predefined way the $M \times n$ symbols of X to form X' which is then transmitted. For instance, choosing $M = n$ and rearranging X as a matrix of size $n \times n$ (one row is one codeword), a simple interleaver consists in transmitting X column by column instead of row by row. In which context the interleaver gradually improves the performance of the overall system?

If you want to know more about Error Correcting Codes, you can open the Matlab help window and go through the section 'Communications System Toolbox > System Design > Error Detection and Correction'.
Alternatively, you can check on Wikipedia for some information while waiting to take the lecture *Algebra and Error Correcting Codes*. Congratulations, you have reached the end of this fachpraktikum.

Appendix A: Hamming Code

Message	Encoded Message (Codewords)	Weight
0 0 0 0	0 0 0 0 0 0 0	0
0 0 0 1	1 0 1 0 0 0 1	3
0 0 1 0	1 1 1 0 0 1 0	4
0 0 1 1	0 1 0 0 0 1 1	3
0 1 0 0	0 1 1 0 1 0 0	3
0 1 0 1	1 1 0 0 1 0 1	4
0 1 1 0	1 0 0 0 1 1 0	3
0 1 1 1	0 0 1 0 1 1 1	4
1 0 0 0	1 1 0 1 0 0 0	3
1 0 0 1	0 1 1 1 0 0 1	4
1 0 1 0	0 0 1 1 0 1 0	3
1 0 1 1	1 0 0 1 0 1 1	4
1 1 0 0	1 0 1 1 1 0 0	4
1 1 0 1	0 0 0 1 1 0 1	3
1 1 1 0	0 1 0 1 1 1 0	4
1 1 1 1	1 1 1 1 1 1 1	7