AN INTRODUCTION TO LINEAR ALGEBRA IN ERROR CORRECTING CODES

OLIVIA XU

1. Introduction

This paper serves as a gentle introduction to error-correcting codes. We begin with fundamental definitions then explain the core problem that we aim to solve with error-correcting codes. Next, we introduce an intuitive solution to this problem that provides the groundwork for the rest of the paper. After defining linear codes, we study and analyze the efficiency of a specific family of codes: Hamming codes. Finally, we conclude by discussing real world applications and other types of linear codes.

First, we introduce some definitions:

Definition 1.1. An **alphabet** A of size s is a set of s characters, or code symbols. A^n denotes the set of all strings of length n whose characters are in A.

Definition 1.2. A **code** C of length n is some subset of A^n.

Definition 1.3. A **binary code** is a code over the alphabet $\{0, 1\}$ whose characters are referred to as bits.

In this paper, we specifically examine binary codes and work in the field $\mathbb{F}_2$. We will assume familiarity with field operations.

2. The Problem

In information theory, codes are transmitted over a communication channel that in the real world may take the form of a wire or a radio channel. Sending codes over channels presents the risk of error or corruption. For example, a 0 may become a 1, or vice versa. This paper will explore codes with at most one error per string.

We construct an information theoretic framework as follows:

(1) A source wants to convey a message to a receiver.
(2) The source encodes the message to reduce redundancy for ease and efficiency of communication.
(3) The channel encodes the message to introduce redundancy to ensure that decoding is always possible after transmission across the channel.
(4) The channel decodes the message.
(5) The receiver decodes the message.

Date: April 11, 2023.

So, what really is an error-correcting code? An **error-correcting code** is a function that transforms a message of, say, length k into a code of length n such that we can always recover the original message despite errors in the encoded message.

3. An Intuitive Solution

Suppose we wish to send a message $\mathbf{x}$ of length k. Then, we transmit the code $G\mathbf{x}$. G is called the **generator matrix** of the code and essentially encodes the source's message.

A simple solution for a valid error-correcting code is transmitting each bit three times. That is, for each bit $\mathbf{x}$, we employ the generator

$$G = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

to encode the message. For example, if $\mathbf{x} = (1)$, then we transmit

$$G\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

and if $\mathbf{x} = (0)$, then we transmit

$$G\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

In the best case scenario in which no corruption occurs, the received message is equivalent to $G\mathbf{x}$. If an error occurs, however, then one bit is flipped, so we can deduce the correct bit by locating the bit that appears twice. Thus, this repetition code allows us to both detect and correct up to one error.

One way to formalize this is to use a **parity checker**. Suppose the receiver obtains the message

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}.$$

Then, when no errors occur, it must be the case that $y_1 + y_2 = 0$ and $y_1 + y_3 = 0$. So, we create the **parity checker matrix**

$$P = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Now, there are four possible outputs upon computing $P\mathbf{y}$. Analyzing the output allows us to not only detect but also correct a potential error.

(1) $P\mathbf{y} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. In this case, $y_1 = y_2$ and $y_1 = y_3$ so no error occurred. Therefore, we can conclude that $\mathbf{x} = \mathbf{y}$.

(2) $P\mathbf{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. In this case, $y_1 = y_2$ but y_3 is different. Thus, y_3 was flipped during transmission, so the original message is in y_1.

(3) $P\mathbf{y} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. In this case, $y1 = y3$ but y_2 is different. So, like above, we conclude that bit y_2 was flipped, and the original message is in y_1.

(4) $P\mathbf{y} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. In this case, $y1$ differs from both y_2 and y_3. Hence, we conclude $y_2 = y_3$ and the error lies in y_1, so the original message is in y_2.

Next, we generalize the generator matrix and parity checker matrix through examining linear codes.

4. Linear Codes

In the first example of an error-correcting code, we dealt with vectors that represent linear transformations of the message. So, we define linear codes as follows:

Definition 4.1. Suppose a source wishes to convey messages of length k through a channel, which transforms the messages into codewords of length n. An error-correcting code is **linear** if there exists an $n \times k$ generator matrix G such that every message $\mathbf{x}$ is encoded as $G\mathbf{x}$.

What allows this code to correct errors? We again introduce the parity checker matrix P.

Definition 4.2. For an error-correcting code, the **parity checker matrix** P exists such that $P\mathbf{y} = \mathbf{0}$ if and only if $\mathbf{y}$ has no errors.

Let's look at an example. Suppose a source wishes to communicate a message using the code with generator matrix

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

The codewords corresponding to each source message are as follows:

Message	Codeword
000	000000
001	001011
010	010101
011	011110
100	100110
101	101101
110	110011
111	111000

Let's consider parity checker matrix

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Upon obtaining the transmitted codeword $\mathbf{y}$, the receiver can compute $P\mathbf{y}$ to locate the potential error. For example, suppose source originally sends the message 010, but the receiver obtains the corrupted codeword $\mathbf{y} = 010100$ (which contains an error in the sixth bit). Then,

$$P\mathbf{y} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

so there is indeed an error. On the other hand, if the receiver had obtained the (correct) codeword $\mathbf{y} = 010101$, then

$$P\mathbf{y} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Observe that all valid codewords belong to the nullspace of P. This brings us to the question of efficiency. How many codewords are in this nullspace? If the nullspace comprises few codewords, then the source can send few messages. However, if the nullspace comprises many codewords, then accurately decoding the received message will prove difficult, especially as the number of errors increases. To quantify this, we introduce the notion of **Hamming distance**.

Definition 4.3. The **Hamming distance** of two vectors $\mathbf{v_1}$ and $\mathbf{v_2}$, denoted by $d(\mathbf{v_1}, \mathbf{v_2})$, is the number of bits in which they differ.

Definition 4.4. The **minimum distance** of a set of codewords S is $\min(d(\mathbf{v_1}, \mathbf{v_2}))$ over all pairs of distinct $\mathbf{v_1}, \mathbf{v_2} \in S$.

Suppose we have an error-correcting code with minimum distance d that encodes messages of length k into codewords of length n. We aim to optimize this code by specifically optimizing one parameter given the others. Why is this important? Maximizing d allows for more room for error correction as the error-correcting code can correct a larger number of errors. Minimizing n allows for faster and cheaper transmission of the message.

5. Hamming Codes

Now, we analyze Hamming codes, a family of linear codes, first introduced by Richard W. Hamming in 1950 as a means of correcting errors in punched card codes. Hamming codes are able to detect and correct one error. Therefore, for long messages, Hamming codes are not efficient as long strings of bits are prone to more errors. However, as we will see, Hamming codes are quite efficient.

First, we define that it means for a code to be efficient so we can better analyze these codes.

Definition 5.1. Given a code that transmits messages of length k as codewords of length n, the efficiency of the code is measured by $\frac{k}{n}$

We seek to maximize efficiency in a code (hence decreasing redundancy). For example, our simple repetition code has a relatively low efficiency of $\frac{1}{3}$, whereas our second example code has a higher efficiency of $\frac{3}{6} = \frac{1}{2}$.

Now, let's take a look at the fundamental and widely popular Hamming $(7, 4)$ that adds three redundancy bits to each message of length 4. The Hamming $(7, 4)$ has generator matrix

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

and parity checker matrix

$$P = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

To show the Hamming $(7, 4)$ is indeed an error-correcting code, consider a message

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}.$$

Then, the source transmits the message

$$G\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_1 + x_2 + x_3 \\ x_1 + x_2 + x_4 \\ x_1 + x_3 + x_4 \end{pmatrix}.$$

We calculate that $HG\mathbf{x} = \mathbf{0}$, and observe that this is true if and only if $G\mathbf{x} = \mathbf{0}$. Therefore, this code is a valid error-correcting code.

The Hamming $(7, 4)$ has efficiency $\frac{4}{7} > \frac{1}{2}$. The general Hamming code, given a message of length $2^m - 1 - m$, adds m bits of redundancy to transmit a codeword of $2^m - 1$. As m increases, the efficiency $\frac{2^m-1-m}{2^m-1}$ tends to 1, which is much higher than our previous codewords. Today, Hamming codes are widely used for computer modems and memory for their efficiency.