Linear Algebra Report

Rishabh Malik

1 Linear Codes

In this chapter, we study linear error-control codes, which are special codes with rich mathematical structure. Linear codes are widely used in practice for a number of reasons. One reason is that they are easy to construct. Another reason is that encoding linear codes is very quick and easy. Decoding is also often facilitated by the linearity of a code. The theory of general linear codes requires the use of abstract algebra and linear algebra, but we will begin with some simpler binary linear codes

1.1 Binary Codes

A binary linear code C of length n is a set of binary n-tuples such that the component wise modulo 2 sum of any two codewords is contained in C.

Example 1.1.1. The set 0000, 0101, 1010, 1111 is a binary linear code, since the sum of any two codewords lies in this set. Notice that this is a (4, 4, 2) code because the codewords have length 4, there are 4 codewords, and the minimum distance between codewords is 2.

Example 1.1.2. The set 0000000, 0001111, 0010110, 0011001, 0100101, 0101010, 0110011, 0111100, 1000011, 1001100, 1010101, 1011010, 1100110, 1101001, 1110000, 1111111 is a binary linear code known as a Hamming code. Notice that this is a (7, 16, 3) code because the codewords have length 7, there are 16 codewords, and the minimum distance is 3.

2 Fields, Vector Spaces, Subspaces and General Linear Codes

In order to develop the theory for general linear codes, we need some abstract and linear algebra. Learning the definitions and getting comfortable with the examples will be very important. We begin with a discussion of the mathematical constructs known as fields. The binary alphabet, along with modulo 2 addition and multiplication, is the smallest example of a field.

2.1 Field

Let F be a nonempty set that is closed under two binary operations denoted + and *. That is, the binary operations input any elements a and b in F and output other elements c = a + b and d = a * b in F . Then (F, +, *) is a field if:

1. There exists an additive identity labeled 0 in F such that a + 0 = 0 + a = a for any element a ∈ F.

2. The addition is commutative: a + b = b + a, for all a, b ∈ F.
3. The addition is associative: (a + b) + c = a + (b + c), for all a, b, c ∈ F.
4. There exists an additive inverse for each element: For each a ∈ F , there exists an element denoted -a in F such that a + (-a) = 0.
5. There exists a multiplicative identity denoted 1 in F such that a * 1 = 1 * a = a for any element a ∈ F.
6. The multiplication is commutative: a * b = b * a, for all a, b ∈ F.
7. The multiplication is associative: a * (b * c) = (a * b) * c, for all a, b, c ∈ F.
8. There exists a multiplicative inverse for each nonzero element: For each a 6 = 0 in F , there exists an element denoted a -1 in F such that a * a -1 = a -1 * a = 1.
9. The operations distribute: $a * (b + c) = a * b + a * c$ and $(b + c) * a = b * a + c * a$, for all a, b, c ∈ F.

2.2 Vector Spaces

Let F be a field. A set V of elements called vectors is a vector space if for any u, v, w ∈ V and for any c, d ∈ F :

1. u + v ∈ V
2. cv ∈ V
3. u + v = v + u
4. u + (v + w) = (u + v) + w
5. There exists an element denoted 0 ∈ V such that u + 0 = u
6. There exists an element denoted 0 ∈ V such that u + 0 = u
7. c(u + v) = cu = cv
8. (c + d)u = cu + du
9. c(du) = (cd)u
10. 1u = u

2.3 Subspaces

Let V be a vector space over a field F and let W be a subset of V . If W is a vector space over F , then W is a vector subspace of V .

Theorem 2.3.1. If W is a subset of a vector space V over a field F , then W is a vector subspace of V iff $\alpha u + \beta v \in W$ for all α , β, $\gamma \in$ F and all u, v $\in$ W .

We can now begin to connect vector spaces to error-control codes. Earlier, we discussed the linear code C = 000, 100, 001, 101. We will now show that this code is a vector subspace of the vector space V (3, 2) of binary 3-tuples. Using the above mentioned theorem in subspaces, we must check if every linear combination of vectors in C remains in C. In fact, this is the defining condition for a linear code. We have just discovered one of the major links between coding theory and linear algebra: Linear codes are vector subspaces of vector spaces.

2.4 Linear Codes

A linear code of length n over GF (q) is a subspace of the vector space GF (q) n = V (n, q).

Theorem 2.4.1. Let C be a linear code. The linear combination of any set of codewords in C is a code word in C.

Example 2.4.1 Let's continue working with our linear code C = {000, 100, 001, 101}. We can see that this code is spanned by the linearly independent vectors [100] and [001], since all other vectors in C can be obtained as binary linear combinations of these two vectors. In other words, these two vectors form a basis for C, so the dimension of C is 2.

Theorem 2.4.2. The dimension of a linear code C of length n is its dimension as a vector subspace of $GF(2)^n = V(n, 2)$.

If C is a k-dimensional subspace of V (n, q), we say that C has dimension k and is an [n, k, d] or [n, k] linear code. In particular, if the dimension of a binary code is k, then the number of codewords is 2^k . In general, if the dimension of a q-ary code is k, then the number of codewords is q^k . This means that C can be used to communicate any of q^k distinct messages. We identify these messages with the q^k elements of V (k, q), the vector space of k-tuples over GF (q). This is consistent with the aforementioned notion of encoding messages of length k to obtain codewords of length n, where $n > k$.

2.5 Encoding Linear Codes: Generator Matrices

Linear codes are used in practice largely due to the simple encoding procedures facilitated by their linearity. A k $\times$ n generator matrix G for an [n, k] linear code C provides a compact way to describe all of the codewords in C and provides a way to encode messages. By performing the multiplication mG, a generator matrix maps a length k message string m to a length n codeword string. The encoding function m $\rightarrow$ mG maps the vector space V (k, q) onto a k-dimensional subspace (namely the code C) of the vector space of V (n, q).

Before giving a formal definition of generator matrices, let's start with some examples.

Example 2.5.1. We will now extend the work done in Example 2.4.1 on C = {000, 100, 001, 101}. We have already shown that C is a 2-dimensional vector subspace of the vector space V (3, 2). Therefore, C is a [3, 2] linear code and should have a 2 $\times$ 3 generator matrix. However, let's use our intuition to derive the size and content of the generator matrix. We want to find a generator matrix G for C that will map messages (i.e. binary bit strings) onto the four codewords of C. But how long are are the messages that we can encode? We can answer this question using three facts:

(1) we always encode messages of a fixed length k to codewords of a fixed length n; (2) a code must be able to encode arbitrary messages of length k; and (3) the encoding procedure must be "one-to-one," or in other words, no two messages can be encoded as the same codeword. Using these three facts, it is clear that the messages corresponding to the four codewords in C must be of length 2 since there are exactly 4 distinct binary messages of length 2: [00], [10], [01], and [11]. Hence, in order for the matrix multiplication to make sense, G must have 2 rows. Since the codewords are of length 3, properties of matrix multiplication tell us that G should have 3 columns. In particular, if $G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, then $[00]G = [000]$, $[10]G = [100]$, $[01]G = [001]$ and $[11]G = [101]$. Hence, this code C can encode any length 2 binary message. The generator matrix determines which length 2 message is encoded as which length 3 codeword.

We have now seen how a generator matrix can be used to map messages onto codewords. We have also seen that an [n, k] linear code C is a k-dimensional vector subspace of V (n, q) and can hence be specified by a basis of k codewords, or q-ary vectors. This leads us to the following formal definition of a generator matrix: *A $k \times n$ matrix G whose rows form a basis for an [n, k] linear code C is called a generator matrix of the code C.*

2.6 Parity Check Matrices

Let C be an [n, k] linear code. A parity check matrix for C is an (n - k) $\times$ n matrix H such that $c \in C$ if and only if $cH^T = 0$.

Example 2.6.1. If C is the code {111, 000}, then a valid parity check matrix is

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

One way to verify this is to compute the product cH T for an arbitrary codeword c = [x, y, z]:

$$[xyz] \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = [x + z, y + z]$$

So, c = [xyz] is a valid codeword iff cH T = 0, ie. iff [x + z, y + z] = [0, 0]. Algebraic manipulations over binary show that x = y = z. Thus the only valid codewords are indeed 000, 111. Notice that this is the binary repetition code of length 3.

2.7 Hamming Codes

Hamming distance: The Hamming distance d(a,b) between two vectors a and b is the number of bits they differ by.

Traditional [7,4] Hamming code encodes four data bits into seven bits by adding three parity bits. It can detect and correct single-bit errors. Lets see how generator matrix and parity Matrix are formulated and error detection and correction is done.

2.7.1 Encoding

Each of the three parity bits are calculated based on parity of the 3 of the 4 data bits. All of the parity bits are even parity.

Given: data bits d1, d2, d3, and d4 then A (7, 4) Hamming code may define parity bits p1, p2, and p3 as
p1 = d2 + d3 + d4
p2 = d1 + d3 + d4
p3 = d1 + d2 + d4, also there's a fourth equation for a parity bit that might be used in Hamming codes: p4 = d1 + d2 + d3. A valid Hamming codes may use any three parity bit definitions.

2.7.2 Error detection and correction

When the encoded data is received we need to check the parity values to verify if the data received is correct or not. Parity is validated using matrix operations. A 3×7 parity check matrix [H] may be constructed such that row 1 contains 1s in the position of the first parity bit and all of the data bits that are included in its parity calculation. Row 2 contains 1s in the position of the second parity bit and all of the data bits that are included in its parity calculation, similarly for row 3.

Multiplying the 3×7 matrix [H] by a 7×1 matrix representing the encoded data produces a 3×1 matrix called the "syndrome". There are two useful proprieties of the syndrome. If the syndrome is all zeros, the encoded data is error free. If the syndrome has a non-zero value, flipping the encoded bit that is in the position of the column in [H] that matches the syndrome will result in a valid code word.

Example:
Let data bits be column vectors

$$d_1 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, d_2 = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, d_3 = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, d_4 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Parity bits are also in form of column vectors with 1s in rows where it's 1 in the result of equations mentioned above in encoding. So, parity vectors are

$$p_1 = \begin{bmatrix}0\\1\\1\\1\end{bmatrix}, p_2 = \begin{bmatrix}1\\0\\1\\1\end{bmatrix}, p_3 = \begin{bmatrix}1\\1\\0\\1\end{bmatrix},$$

For getting the generating matrix(G), we arrange the above 7 column vectors in sequence p1, p2, p3, d1, d2, d3, d4, i.e, 3 parity vectors followed by 4 data vectors.

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

Lets take a sample data value = 1010 and apply hamming's encoding on it.

$$\begin{bmatrix}1&0&1&0\end{bmatrix}, \begin{bmatrix}0&1&1&1&0&0&0\\1&0&1&0&1&0&1\\1&1&0&0&0&1&0\\1&1&1&0&0&0&1\end{bmatrix}, = \begin{bmatrix}1&0&1&1&0&1&1\end{bmatrix}$$

[1 0 1 1 0 1 1] is the encoded data.
Now, to verify the encoded data, we apply parity check on it. For this, we require parity check matrix(H).

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

[1 1 1] contain non-zero entries. Looking back at the matrix [H], w will see that the seventh column(binary equivalent of 111,so the seventh bit is the error bit. Changing the seventh bit produces the code word 1011010.

3 Types of Linear Code

There are many types of linear block codes, such as

- Cyclic codes (e.g., Hamming codes)
- Repetition codes
- Parity codes
- Polynomial codes (e.g., BCH codes)
- Reed-Solomon codes
- Algebraic geometric codes
- Reed-Muller codes
- Perfect codes

We won't be going into the explaination of these different types of code in this report.

4 Applications

4.1 Shadow generation of medical images

For two medical images I_1 and I_2, we make 2 shadow images (S_1 and S_2). These 2 shadow images depend on each other. Having only 1 of these shadow images is of no use as for generating back original image, we require both of these. We use (7,4) hamming code for making these shadow images. Pixel level operations are performed on both the medical images simultaneously for generating shadow images.

Let pixel be of 8 bits with value in range 0-255. Let us take 2 pixels $P_1 = [i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8]$ and $P_2 = [i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8]$ from S_1 and S_2 respectively.
Algorithm: Shadow generation algorithm
Input: P_1 and P_2
Output: P_1‘ and P_2‘
Step1. Assign $[d_1d_2d_3d_4] = [i_1, 1i_1, 2i_1, 3i_1, 4]$, and $[x_1x_2x_3] = [i_2, 1i_2, 2i_2, 3]$.
Step 2. Use (7, 4) Hamming code to compute $c = d * G = [p_1p_2p_3d_1d_2d_3d_4]$.
Step 3 . For the codeword c, flip one bit at the position $(x_3x_2x_1)_2$, where x_1 is the least significant bit, to output a new codeword $c' = [p'_1p'_2p'_3d'_1d'_2d'_3d'_4]$ round P_1 was selected from I_1 , then it will be obtained from I_2 in the next round to make sure that the shadows are very much different from the original images.
In this manner, P_1 and P_2 are picked alternatively from I_1 and I_2 to have a uniform distribution. Similar to this we have algorithm for constructing the original image from these shadow images.

4.2 Construction of PRIO codes using coset coding

Random input/output (RIO) code are used to improve the performance of traditional methods of accessing flash drives which relied on sing multiple read threshold bits. RIO offers mechanism to get one logical page using one read threshold.

In the RIO code, the data of all logical pages are stored simultaneously; thus, all the data are known in advance. Parallel RIO (P-RIO) code are used to increase the efficiency by performing the encoding of each page in parallel.

P-RIO codes are constructed using coset coding. Coset coding is a technique that constructs RIO codes using linear binary codes. Hamming codes are used as the linear codes, to leverage the information on the data of all logical pages to construct the P-RIO codes. The constructed codes store more pages than RIO codes constructed via coset coding with Hamming codes.

4.2.1 Coset coding

A linear binary code of length n and dimension k is referred to as an (n, k) code. coset coding is used to construct an $[n, n-k, t]$ WOM code using (n, k) code C, where t depends on C[4]. Let H be the parity check matrix of C. For all $(\boldsymbol{d}, \boldsymbol{c}) \in \{0,1\}^{n-k} \times \mathrm{Im}\left(\mathcal{E}^{i-1}\right)$ with $i \in \mathcal{I}_t$, the encoding map $\mathcal{E}$ is as follows.

$$\mathcal{E}(\boldsymbol{d}, \boldsymbol{c}) = \boldsymbol{c} + \boldsymbol{x}$$

where

$$\boldsymbol{x} \in \left\{\boldsymbol{v} \mid \boldsymbol{v} \in \{0,1\}^n, \boldsymbol{v}H^T = \boldsymbol{d} - \boldsymbol{c}H^T, I(\boldsymbol{c}) \cap I(\boldsymbol{v}) = \emptyset\right\}$$

Further, for all $c \in \mathrm{Im}\left(\mathcal{E}^i\right)$ with $i \in \mathcal{I}_t$, the decoding map $\mathcal{D}$ is

$$\mathcal{D}(\boldsymbol{c}) = \boldsymbol{c}H^T$$

The following theorem has been proven in [4]. Theorem 1: When the linear code is the (7,4) Hamming code, coset coding can be used to construct the [7,3,3] WOM code. Let r be an integer greater than 3. Then a $\left[2^r - 1, r, 2^{r-2} + 2\right]$ WOM code can be constructed via coset coding with a $(2^r - 1, 2^r - r - 1)$ Hamming code.

4.2.2 RIO Code

In this paper, it is assumed that each cell of a flash memory represents one of q levels $(0, 1, \ldots, q-1)$. These levels are distinguished by $(q-1)$ different read thresholds. For each $i \in \mathcal{I}_{q-1}$, we denote the threshold between levels $(i-1)$ and i by i. For the state of n cells $c = (c_1, \ldots, c_n) \in \{0, \ldots, q-1\}^n$, the operation for read threshold i is denoted by $RT_i(\boldsymbol{c})$. We define $RT_i(\boldsymbol{c}) = (r_1, \ldots, r_n) \in \{0,1\}^n$ where for each $j \in \mathcal{I}_n, r_j = 1$ if $c_j \geq i$; otherwise , $r_j = 0$. For any $c_1, \ldots, c_{q-1} \in \{0,1\}^n$ such that $c_1 \leq c_2 \leq \ldots \leq c_{q-1}$, the following property holds: For each $i \in \mathcal{I}_{q-1}$,

$$RT_{q-i}\left(\sum_{j=1}^{q-1} \boldsymbol{c}_j\right) = \boldsymbol{c}_i$$

RIO code is a coding scheme that stores t pages in a $(t+1)-$ level flash memory, such that page i can be read using read threshold $(t+1-i)$ for each $i \in \mathcal{I}_t$[1]. From (6), pages $1, \ldots, t$ are encoded into $c_1, \ldots, c_t$, respectively, such that $\boldsymbol{c}_1 \leq \boldsymbol{c}_2 \leq \cdots \leq \boldsymbol{c}_t$, and the cell state is set to $\sum_{i=1}^{t} \boldsymbol{c}_i$ where c_i is a binary vector for each $i \in \mathcal{I}_t$.

Definition 2: An $[n, l, t]$ RIO code is a coding scheme that enables storage of t pages of l data bits in $n(t+1)$ -level cells, such that each page can be read using a single read threshold. For each $i \in \mathcal{I}_t$, the encoding map of page $i\mathcal{E}_i$ is defined by

$$\mathcal{E}_i : \{0,1\}^l \times \mathrm{Im}\left(\mathcal{E}_{i-1}\right) \to \{0,1\}^n$$

where $\mathrm{Im}\left(\mathcal{E}_0\right) = \{(0, \ldots, 0)\}$. For all $(\boldsymbol{d}, \boldsymbol{c}) \in \{0,1\}^l \times \mathrm{Im}\left(\mathcal{E}_{i-1}\right), \boldsymbol{c} \leq \mathcal{E}_i(\boldsymbol{d}, \boldsymbol{c})$ is satisfied. The cell state is the sum of the $\mathcal{E}_i$ results for all $i \in \mathcal{I}_t$. Further, for each $i \in \mathcal{I}_t$, the decoding map of page $i\mathcal{D}_i$ is defined by

$$\mathcal{D}_i : \mathrm{Im}\left(\mathcal{E}_i\right) \to \{0,1\}^l$$

such that $\mathcal{D}_i\left(\mathcal{E}_i(\boldsymbol{d}, \boldsymbol{c})\right) = \boldsymbol{d}$ for all $(\boldsymbol{d}, \boldsymbol{c}) \in \{0,1\}^l \times \mathrm{Im}\left(\mathcal{E}_{i-1}\right)$ If the cell state is $c \in \{0, \ldots, t\}^n$, the argument of $\mathcal{D}_i$ is $RT_{t+1-i}(\boldsymbol{c})$ for each $i \in \mathcal{I}_t$ From Definitions 1 and 2, it is apparent that the construction of an $[n, l, t]$ RIO code is equivalent to that of an $[n, l, t]$ WOM code[1] Example 2: The [3,2,2] RIO code based on the [3,2,2] WOM code is shown in Table III. As an example, let the data of pages 1 and 2 be 10 and 01, respectively. Then, from Table II, $\mathcal{E}_1(10, 000) = 010$ and $\mathcal{E}_2(01, 010) = 011$. Therefore, the cell state is 021 .

5 References

1) `http://www.math.niu.edu/~beachy/courses/523/coding_theory.pdf`
2) `https://www.win.tue.nl/~henkvt/images/CODING.pdf`