25.3 Linear Codes

What are Linear Codes?

We will now look at a special case of block codes called linear codes. Restricting our attention to linear codes is an important part of coding theory, as the conditions which define a linear code also allow for a more plentiful and efficient roster of decoding algorithms. Linear Codes will require us to apply our knowledge of matrices and linear algebra to the results of block codes and finite fields from the previous section. How can we possibly apply linear algebra to block codes and finite fields? Think about chapter 24. We learned that Gauss Jordan Elimination works over matrices who's elements are also elements of some finite field. We specifically looked at elements of $\mathbb{F}_2 = \{0, 1\}$, but this result may be extended to all finite fields.

We will now work up to defining a linear code. Let F be a finite field with q elements. For example, F=$\mathbb{F}_2$, which means $q = 2$ here and note that for the sake of this chapter we will refer to this specific field as the binary alphabet. We will also henceforth try to refer to our k-tuples as **messages** and it may often be useful to think of them as vectors. Suppose the messages we would like to transmit are a set of k-tuples with elements from F.

Definition 25.3.1

Message Space Consider the set of all k-tuples that can be made from the field F. We will denote this set of all k-tuples over F by $V_k(F)$. This set is typically referred to as a **vector space** in linear algebra, but for the sake of our topic, we will refer to it as the **message space**.

Let's consider the set $V_k(F)$ or what we will now refer to as our **Message Space**. How many k-tuples or "messages" are in this set? Think about our counting techniques from earlier in the course and the next result should follow quite naturally.

Lemma 25.3.2

Cardinality of Message Space There are q^k messages in the message space.

As we know, Coding Theory is all about message expansion and decoding. Let's look at how we can expand these k-tuples into n-tuples for some $n \geq k$. Let's take another vector space $V_n(F)$ for an $n \geq k$ and set up a one-to-one correspondence between our q^k k-tuples in $V_k(F)$ and q^k n-tuples in $V_n(F)$. How are we going to do that? Think back to our definition of a subgroup and then consider a similar definition presented below, which may serve as a reminder from linear algebra:

Definition 25.3.3

Subspace A subspace of $V_n(F)$ is a subset $S \subset V_n(F)$ satisfying:

Non-Empty The zero vector is in S.

Closed Under Addition If $\vec{u}$ and $\vec{v}$ are in S, then $\vec{u} + \vec{v}$ is also in S.

Closed Under Scalar Multiplication If c is in F and $\vec{v}$ is in S, then $c\vec{v}$ is in S.

We are going to define this correspondence by making sure that our q^k n-tuples form a subspace of $V_n(F)$. We will call this subspace S a **k-dimensional subspace in** $V_n(F)$. Therefore, for our message space M, we will have a one-to-one correspondence $f : M \to S$ that will expand our message.

Example 25.3.4

Expanding Messages Consider the message space:

$$M = V_2(\mathbb{F}_2) = \{(0\,0), (1\,0), (0\,1), (1\,1)\}$$

Note: $k = 2$ and all calculations will be done over $F = \mathbb{F}_2$

Let's expand the message for $n = 4$:
We need a $k = 2$ dimensional subspace of $V_4(\mathbb{F}_2)$
Let's do that by taking the basis $B = \{(1\,1\,0\,0), (1\,0\,1\,0)\}$.

$$(0\,0)\begin{pmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\end{pmatrix} = (0\,0\,0\,0)$$

$$(1\,0)\begin{pmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\end{pmatrix} = (1\,1\,0\,0)$$

$$(0\,1)\begin{pmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\end{pmatrix} = (0\,1\,1\,0)$$

$$(1\,1)\begin{pmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\end{pmatrix} = (1\,0\,1\,0)$$

And hence we have our 2 dimensional subspace of $V_4(\mathbb{F}_2)$:
$S = \{(0\,0\,0\,0), (1\,1\,0\,0), (0\,1\,1\,0), (1\,0\,1\,0)\}$
And our mapping $f : M \to S$ that will expand our k-tuples into n-tuples!

$$(0\,0) \to (0\,0\,0\,0)$$

$$(1\,0) \to (1\,1\,0\,0)$$

$$(0\,1) \to (0\,1\,1\,0)$$

$$(1\,1) \to (1\,0\,1\,0)$$

Notice that in the example, we could have chosen any k linearly independent non-zero vectors in $V_n(F)$ as our basis and gotten a k dimensional subspace as S. Any of these subspaces would have worked and there exist many. In fact, we can count them:

Counting Subspaces

Since we have q^n vectors, but we may not select the zero vector, we can select the first vector in: $q^n - 1$ ways

There are exactly q vectors linearly independent from the first so we can select the second vector in: $q^n - q$ ways

We need to do this k times, hence we can select k linearly independent vectors in: $(q^n - 1)(q^n - q)(q^n - q^2)...(q^n - q^{k-1})$ ways.

But now we need the number of k dimensional subspaces. The above result over-counts them since some of these sets of k linearly independent vectors will generate the same subspace. In fact, exactly $(q^k - 1)(q^k - q)...(q^k - q^{k-1})$ of them will.

Hence, there are exactly

$$\frac{(q^n - 1)(q^n - q)(q^n - q^2)...(q^n - q^{k-1})}{(q^k - 1)(q^k - q)...(q^k - q^{k-1})}$$

k dimensional subspaces of $V_n(F)$ that we can choose from.

We now have more than enough information to formally define a linear code.

Definition 25.3.5

Linear Code An (n, k)-code over F is a k-dimensional subspace of $V_n(F)$.

Notice the use of the round brackets, as they are only used to represent linear codes. (n, k)-codes are sometimes called (n, k, d)-codes, where d is the distance of the code.

Definition 25.3.6

Hamming Weight The Hamming weight of a vector $\vec{v}$ is the number of non-zero coordinates in $\vec{v}$. It is denoted $w(\vec{v})$

Hamming Weight of an (n, k)-code Let C be an (n, k)-code, then the hamming weight of C is:

$$w(C) = min\{w(\vec{x}) \mid x \in C, x \neq 0\}$$

In other words, it is the weight of the "lightest" non-zero codeword in C.

Theorem 25.3.7

Distance of an (n,k)-code. Let d be the distance of an (n,k)-code C. Then,

$$d = W(C)$$

Generator Matrix and Parity Check Matrix

Let's think about **Example 25.3.4** again. We chose two linearly independent vectors in $V_n(F)$ to form our basis B and put them into a matrix for constructing our subspace S. We will henceforth refer to such a matrix as the **Generator Matrix** for our code C.

Definition 25.3.8

Generator Matrix A generator matrix G for an (n,k)-code C is a $k \times n$ matrix whose rows are a vector space basis for C

Notice that we are now referring to our subspace S as our code C. It is also helpful to notice that the non-zero vectors or **codewords** in C are all linear combinations, calculated over the field F, of the rows of G. Hence, we say that C is the code **generated** by G.

Let's again refer back to the discussion in chapter 24 on elementary row operations. We know that such operations are possible to do over a finite field F, which means we may apply these row operations to our generator matrix G to receive another matrix which also generates the code C. This means that a generator matrix for a code C is not unique.

It can be useful to have our generator matrix in the form $G = [I_k \ A]$, where I_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$ matrix. To get the generator matrix in this form, we can do elementary row operations over F. Sometimes the result we get from the row operations still does not give us a matrix in our desired form. In this case, we may archive the desired form by permuting some column vectors. Suppose the generator matrix G for a code C has undergone column permutations. Call this resulting matrix G'. G' generates a code C' that is called an **equivalent code** to C.

Example 25.3.9

Generator Matrix Let $G_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$

This row reduces using elementary row operations over $\mathbb{F}_2$ to the matrix:

Example 25.3.9 (cont.)

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Note this matrix is now in reduced row echelon form by our row reduction $R_1 = R_1 + R_2 + R_3 + R_4$

G_1 and G_2 will both generate the same code C.

Say we want our generator matrix in the form $G = [I_k \ A]$. We must permute or "swap" columns 4 and 5 of G_2. Call this G'.

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

Let C' be the code generated by G'. C' is an **equivalent code** to C. Note that the codewords in C' will just be the codewords in C with the elements in positions 4 and 5 permuted. Note that C' and C will have the same distance d.

Having a generator matrix of this form provides us with a convenient structure for our codewords. Say we had a vector or **message** in $V_4(F)$ that we would like to encode. We will use the generator matrix G' from **Example 25.3.9** to do so.

Let $\vec{m} = (1\,1\,0\,0)$

Now, encode: $(1\,1\,0\,0)G' = (1\,1\,0\,0\ \ 0\,0\,1\,1) = \vec{c}$

Notice that the first $k = 4$ elements of our codeword $\vec{c}$ are precisely those of our message $\vec{m}$. That is thanks to the $[I_k \ A]$ structure of our generator matrix. We call these k elements the **information bits**.

We refer to the last four bits as **check bits** since they provide us with the redundancy required for checking errors.

It is therefore useful to have our generator matrix in this form. In fact, we will always be able to put it in this form:

Theorem 25.3.10

Standard form of a Generator Matrix For an (n, k)-code C over a field F, there exists either a generator matrix G for C or G' for an equivalent code G' such that $G = [I_k \ A]$.

We have seen that the generator matrix is an essential part of encoding messages in $V_k(F)$ into codewords in $V_n(F)$, but what about decoding? This will require us to define another matrix known as the **parity check matrix**. The following definition should be a reminder from linear algebra:

Definition 25.3.11

Inner Product Let $\vec{x} = (x_1\ x_2 ...\ x_n)$ and $\vec{y} = (y_1\ y_2 ...\ y_n)$ be vectors or **codewords** in $V_n(F)$. The inner product of $\vec{x}$ and $\vec{y}$ is:

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

Definition 25.3.12

Orthogonal Complement of C If C is an (n, k)-code over F, we may define $C^{\perp}$ or **"C perp"** or **the orthogonal complement of C** as

$$C^{\perp} = \{\vec{x} \in V_n(F)\ :\ \vec{x} \cdot \vec{y} = 0, \forall y \in C\}$$

In other words, $C^{\perp}$ is the set of all vectors or **codewords** in $V_n(F)$ that are **orthogonal** to the vectors or **codewords** in C.

We often refer to $C^{\perp}$ as the **dual code** of C. Now, let's note an important fact about the dual code:

Theorem 25.3.13

Theorem 25.3.13 Let C be an (n, k)-code over a field F, then $C^{\perp}$ is an $(n, n-k)$-code over F.

This means that the set $C^{\perp}$ is a linear code. That means it must have a generator matrix. This will be a very essential definition for the rest of this section and in Coding Theory.

Definition 25.3.14

Parity Check Matrix Let C be an (n, k)-code over F. If H is a generator matrix for $C^{\perp}$, then H is called a **parity check** matrix for C.

This definition means that if G is a generator matrix for C and H is a generator matrix for $C^{\perp}$, then H is a parity check matrix for C and G is a parity check matrix for $C^{\perp}$. We are now able to refer to a code by either it's generator matrix or it's parity check matrix. You may be wondering if there is a straightforward way to calculate the parity check matrix by knowing only the generator matrix. There is!

Theorem 25.3.15

From Generator Matrix to Parity Check Matrix If $G = [I_k \ A]$ is a generator matrix for a code C, then $H = [-A \ I_{n-k}]$ is a generator matrix for $C^{\perp}$ (or a parity check matrix for C).

Single Error Decoding for Linear Codes

We will now look at our first decoding algorithm for linear codes. This algorithm relies on our knowledge of the parity check matrix. Note that if $\vec{c} \in C$, in other words, $\vec{c}$ is a codeword of C, then $H\vec{c}^T = \vec{0}$. Our algorithm is going to exploit this fact. First we are in need of one more definition. Note that when a codeword $\vec{c}$ is transmitted over a noisy channel, errors may be introduced to the elements of $\vec{c}$. We may represent this error as a vector $\vec{e}$ and our received word $\vec{r}$ will be our original codeword, plus the error.

Definition 25.3.16

Error For a received word $\vec{r}$, we may represent it as $\vec{r} = \vec{c} + \vec{e}$.

For example, say we sent $\vec{c} = (0\ 1\ 0\ 1\ 1\ 1)$ over a noisy channel and received:
$\vec{r} = (1\ 0\ 0\ 1\ 1\ 1)$
Then our error is $\vec{e} = (1\ 1\ 0\ 0\ 0\ 0)$

In the case above, two errors were introduced to the codeword by the channel. Our algorithm is only going to decode received words for which one error has been introduced, hence the name **single error decoding**.

Algorithm 25.3.17

Decoding Single-error Linear Codes Let $\vec{r}$ be the received word and let H be the parity check matrix for C.

1. Compute $H\vec{r}^T$
2. If $H\vec{r}^T$ is $\vec{0}$, then accept $\vec{r}$ as the transmitted codeword
3. If $H\vec{r}^T = s^T$ is not $\vec{0}$ then compare s^T with the columns of H:

   If there exists some column of H, call it h_i where i is the position of the column in H, such that $s^T = \alpha h$, then e is the vector with α in position i and zeros elsewhere.

   Correct to $\vec{c} = \vec{r} - \vec{e}$
4. Otherwise, more than one error has been introduced by the channel

Example 25.3.18

Decoding Single-error Linear Codes Let F$=\ \mathbb{Z}_5$. Suppose a codeword $\vec{c}$ was transmitted over a noisy channel and $\vec{r} = (4\ 4\ 3\ 3\ 0\ 1\ 0)$ was received. Suppose we also know the parity check matrix of our code C:

$$H = \begin{pmatrix} 1 & 0 & 0 & 2 & 4 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 3 & 1 & 4 & 1 & 2 \end{pmatrix}$$

1. Compute $H\vec{r}^T = \begin{pmatrix} 1 & 0 & 0 & 2 & 4 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 3 & 1 & 4 & 1 & 2 \end{pmatrix} \begin{pmatrix} 4 \\ 4 \\ 3 \\ 3 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} = s^T$

2. $s^T = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix}$ is not zero so proceed to step 3.

3. Compare $s^T = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix}$ to the columns of H. Notice that for the column $h_4 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$ of H:

$$s^T = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} = 3 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} = 3h_4$$

Note that all calculations are being done in $\mathbb{Z}_5$

So, our error vector is $\vec{e} = (0\ 0\ 0\ 3\ 0\ 0\ 0)$
Therefore, our decoded codeword is: $\vec{c} = \vec{r} - \vec{e} = (4\ 4\ 3\ 3\ 0\ 1\ 0) - (0\ 0\ 0\ 3\ 0\ 0\ 0) = (4\ 4\ 3\ 0\ 0\ 1\ 0)$