EE 121: Coding for Digital Communication & Beyond	**Fall 2013**

Lecture 8 — September 26

Lecturer: Prof. Anant Sahai *Scribe: Tim Brown*

This lecture covers:

- Linear Codes
- Syndrome Decoding

8.1 Introduction & Quick Review

In the previous lecture, we explored the concept of affine codes and their logical next step, linear codes. We showed that we could generate codewords that were independent of each other by using a generator matrix (referred to as G) that was filled randomly with 1's and 0's with a Bernoulli(0.5) distribution. Multiplying this generator matrix by our message results in our codeword. See figure 8.1 for the layout and dimensions of our equation, where d is the message we want to send and X is the codeword that is generated. We went on to show that this encoding system has a probability of error that approaches zero as the n goes to infinity and the rate is held at a constant $\frac{L}{N}$.

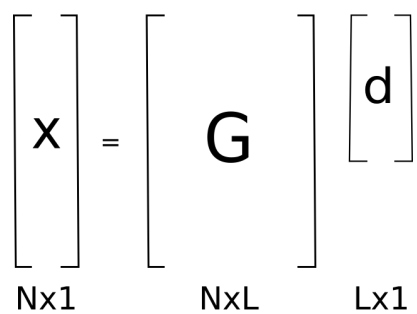

Figure 8.1. The layout and dimensions of our vectors and matrix when dealing with linear codes.

In this lecture, we continued the discussion of linear codes. We decided to take a closer look at what is really going on in the generator matrix.

8.2 Becoming More Efficienct

8.2.1 Past Schemes

Let's take a look back at Nearest Neighbor scheme for decoding. For the sake of discussion, let's define the length of our original message to be $L = 15$ and the length of the codeword we send and receive to be $N = 20$. When decoding, we take our output vector, $\vec{y}$, and compare it to $2^L = 32768$ codewords and choose the closest one. That is a lot of comparisons to make each time you want to decode what you are receiving. Maybe we should try and come up with a scheme that has better runtime performance. Since any given $\vec{y}$ will decode to the same thing each time, let's make a lookup table so we can decode in constant time. While this has a faster runtime, the space complexity is very bad. This lookup table will require $2^N = 1048576$ entries. I feel like we can do better than this and you should too because it's still early in the semester and we still have a lot to learn.

8.2.2 A Closer Look at the Generator Matrix

Let's attempt to get our random generator matrix into a form that seems more systematic. By performing column wise operations, we can get our matrix into a reduced form where the top L rows form an L x L identity matrix (see figure 8.2). You may be thinking to yourself, "OMG! what have we done!" but do not worry. Think back to your training in linear algebra and you will see that we have not changed the pages of our codebook, but rather the order of these pages. This is because linear column operations do not alter the column space of our matrix, and the column space of our matrix is really what defines our matrix. All we have changed is that original messages now map to different codewords.

Now that you are no longer overcome with fear, you will see that this new form of our generator matrix is indeed much more systematic looking, which is fitting because this is known as the "Systematic Form" of the generator matrix. The key thing to realize here is that the top portion of a matrix in systematic form will always be the identity. The last $N - L$ rows of the Systematic Form are what distinguishes one generator matrix from another. Now we only have to store $L(N-L) = 75$ digits. Looks like we are on our way to greater space efficiency. Let's take a moment to look back at the codewords we generated. When we look back at it, we should note that the first L bits of our output will be the same as our original message when we use the systematic matrix. These are known as the systematic bits and the last $N - L$ bits are known as the parity bits. The parity bits are mod 2 sums of a random subset

of data bits.

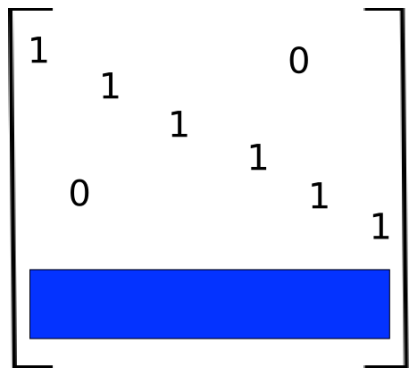

Figure 8.2. Systematic Form of the Generator Matrix. The first L rows of the Systematic Form of the Generator matrix now make up an identity matrix. The remaining N-L rows (represented by the blue rectangle) are still composed of random bit strings and are the only digits that we need to store since the upper portion of the Systematic form is always the identity.

8.3 Syndrome Decoding

8.3.1 Further Manipulation of the Generator Matrix

We've already seen that we can rewrite our generator matrix in systematic form but let's see what other tricks we can do with it. Lets take the portion of the systematic matrix below the identity and call that G'. Let's use this to make a new matrix with $N - L$ rows and N columns by letting the left L columns of this matrix be G' and letting the remaining $N - L$ columns be an identity matrix. Let's call this matrix H (see figure 8.3 for a visual).

This matrix has dimensions that allow us to left multiply our codeword, $\vec{x}$, by it. Let's do this multiplication and see what happens (note that [0] represents a matrix filled with zeros):

$$H\vec{x} = HG\vec{d}$$

$$\mathrm{HG} = \left[\begin{array}{c|c} \mathrm{G'} & \mathrm{I}_{\mathrm{N-L}} \end{array}\right] \left[\begin{array}{c} \mathrm{I}_{\mathrm{L}} \\ \hline \mathrm{G'} \end{array}\right] = \left[\mathrm{G'+G'}\right] = [0]$$

$$HG\vec{d} = [G' + G']\vec{d} = [0]\vec{d} = \vec{0}$$
$$H\vec{x} = \vec{0}$$

After doing the linear algebra in our world of mod 2, we see that the null space of H contains the whole code space. H is referred to as the Parity Check Matrix since its null space defines the code. If we multiply a received codeword by this matrix and we get back something other than a zero, then the codeword was not made by our generator matrix meaning that something or someone corrupted our transmission. In the context of this class, noise is typically the perpetrator. To demonstrate this, let $\vec{y}$ be our received symbol. Let's rewrite $\vec{y}$ as $\vec{x} + \vec{b}$ where $\vec{x}$ is a vector that is a result of a linear combination of the columns of our Generator matrix and $\vec{b}$ fills in the part of $\vec{y}$ that could not be represented by the linear combination of G. Then left multiplying by the syndrome matrix gives us the following:

$$H\vec{y} = H(\vec{x} + \vec{b}) = H\vec{x} + H\vec{b} = H\vec{b} \neq 0$$

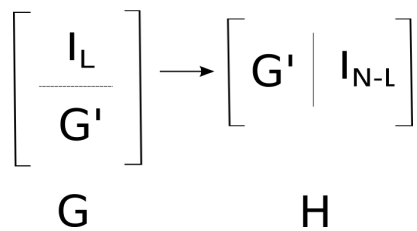

Figure 8.3. This figure shows how the Parity Check Matrix, H, is formed from our original Generator matrix, G.

8.3.2 Syndrome Decoding

Before Syndrome was a famous villain from The Incredibles (see appendix A), syndrome decoding was a scheme that exploits the properties of the parity check matrix H that we recently discovered. Let's take this matrix and multiply it by our received symbol that we will call $\vec{Y}$. This product will be our syndrome which we will represent as $\vec{S}$. The syndrome will have $N - L = 5$ elements so there are only $2^{N-L} = 32$ possible syndromes. Take note that this space is much smaller than the lookup table that was previously used with random coding (1048576 vs 32 entries in our lookup table). You may be asking yourself why we even came up with this things called

syndromes so let's look at what they actually boil down to:

$$\vec{S} = H\vec{Y} = H(G\vec{d} + \vec{N})$$
$$HG\vec{d} + H\vec{N} = H\vec{N}$$

If we put each of the syndromes in a lookup table with the key being the syndrome, we can then make the value the most likely noise. Assuming that the probability of a bit flip is less than one half then the noise with the least weight, or number of ones in the string, is the most likely noise given that the noise satisfies the property: $H\vec{n} = \vec{S}$. There are 2^L possible noise vectors that satisfy this equation. Once we have the most likely noise, we can simply add that to our received code to get back the codeword we originally tried to send.

8.3.3 Equivalence to Maximum Likelihood

This seems a bit suspicious though so let's show that this technique does indeed lead us to the most likely codeword.
First let's show that our received message plus some other noise, $\vec{y}+\vec{n}$, will result in a codeword. To do this, let's look at the following:

$$H(\vec{y} + \vec{n}) = H\vec{y} + H\vec{n} = \vec{S} + \vec{S} = 0$$

Earlier we showed that the nullspace of our Parity Check Matrix was our codebook, therefore the above equation shows that adding a noise vector to our received signal will get us a valid codeword.

Next notice that no two noise vectors will lead to the same codeword since the noise is added in our finite field: $\vec{y} + \vec{n} \neq \vec{y} + \vec{n^*}$. Therefore each noise results in a unique codeword. Taking the noise with the least noise is the same as taking the most probably noise from the set of valid noise vectors so syndrome decoding results in the most likely codeword sent.

8.4 Appendix A

The other Syndrome. A villain from Pixar's <u>The Incredibles</u> and a misunderstood engineer.


SYNDROME
Disney PRESENTS A PIXAR FILM
THE INCREDIBLES
IN THEATERS 11.5.04
© Disney Enterprises, Inc./Pixar Animation Studios. All Rights Reserved.