## *Generator Matrices*

We can arrange a set of basis vectors for a linear code in a *generator matrix*, each row of which is a basis vector.

A generator matrix for an $[N, K]$ code will have $N$ rows and $N$ columns.

Here's a generator matrix for the $[5, 2]$ code looked at earlier:

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

Note: Almost all codes have more than one generator matrix.

## *Encoding Blocks Using a Generator Matrix*

We can use a generator matrix for an $[N, K]$ code to encode a block of $K$ message bits as a block of $K$ bits to send through the channel.

We regard the $K$ message bits as a row vector, $\mathbf{s}$, and multiply by the generator matrix, $G$, to produce the channel input, $\mathbf{t}$:

$$\mathbf{t} = \mathbf{s}G$$

If the rows of $G$ are linearly independent, each distinct $\mathbf{s}$ will produce a different $\mathbf{t}$, and every $\mathbf{t}$ that is a codeword will be produced by some $\mathbf{s}$.

**Example:** Encoding the message block $(1, 1)$ using the generator matrix for the $[5, 2]$ code given earlier:

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

## *Parity-Check Matrices*

Suppose we have specified an $[N, K]$ code by a set of $M = N - K$ equations satisfied by any codeword, $\mathbf{v}$:

$$\begin{aligned} c_{1,1} v_1 + c_{1,2} v_2 + \cdots + c_{1,N} v_N &= 0 \\ c_{2,1} v_1 + c_{2,2} v_2 + \cdots + c_{2,N} v_N &= 0 \\ &\vdots \\ c_{M,1} v_1 + c_{M,2} v_2 + \cdots + c_{M,N} v_N &= 0 \end{aligned}$$

We can arrange the coefficients in these equations in a *parity-check matrix*, as follows:

$$\begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N} \\ & & \vdots & \\ c_{M,1} & c_{M,2} & \cdots & c_{M,N} \end{bmatrix}$$

If $\mathcal{C}$ has parity-check matrix $H$, we can check whether $\mathbf{v}$ is in $\mathcal{C}$ by seeing whether $\mathbf{v}H^T = \mathbf{0}$.

Note: Almost all codes have more than one parity-check matrix.

## *Example: The* $[5, 2]$ *Code*

Here is one parity-check matrix for the $[5, 2]$ code used earlier:

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

We see that 11001 is a codeword as follows:

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

But 10011 isn't a codeword, since

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

## *Examples: Repetition Codes and Single Parity-Check Codes*

An $[N, 1]$ repetition code has the following generator matrix (for $N = 4$):

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Here is a parity-check matrix for this code:

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

One generator matrix for the $[N, N-1]$ single parity-check code is the following:

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

Here is the parity-check matrix for this code:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

## *Manipulating the Parity-Check Matrix*

There are usually many parity-check matrices for a given code. We can get one such matrix from another using the following "elementary row operations":

- Swapping two rows.
- Multipling a row by a non-zero constant (not useful for $Z_2$).
- Adding a row to a different row.

These operations don't alter the solutions to the equations the parity-check matrix represents.

**Ex:** This parity-check matrix for the [5, 2] code:

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

can be transformed into this alternative:

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

## *Manipulating the Generator Matrix*

We can apply the same elementary row operations to a generator matrix for a code, in order to produce another generator matrix, since these operations just convert one set of basis vectors to another.

**Example:** Here is a generator matrix for the [5, 2] code we have been looking at:

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

Here is another generator matrix, found by adding the first row to the second:

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

**Note:** These manipulations leave the set of codewords unchanged, but they *don't* leave the way we encode messages by computing $\mathbf{t} = \mathbf{s}G$ unchanged!

## *Equivalent Codes*

Two codes are said to be *equivalent* if the codewords of one are just the codewords of the other with the order of symbols permuted.

Permuting the order of the columns of a generator matrix will produce a generator matrix for an equivalent code, and similarly for a parity-check matrix.

**Example:** Here is a generator matrix for the [5, 2] code we have been looking at:

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

We can get an equivalent code using the following generator matrix obtained by moving the last column to the middle:

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

## *Generator and Parity-Check Matrices In Systematic Form*

Using elementary row operations and column permutations, we can convert any generator matrix to a generator matrix for an equivalent code that is is *systematic form*, in which the left end of the matrix is the identity matrix.

Similarly, we can convert to the systematic form for a parity-check matrix, which has an identity matrix in the right end.

For the [5,2] code, only permutations are needed. The generator matrix can be permuted by swapping columns 1 and 3:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

When we use a systematic generator matrix to encode a block $\mathbf{s}$ as $\mathbf{t} = \mathbf{s}G$, the first $K$ bits will be the same as those in $\mathbf{s}$. The remaining $N-K$ bits can be seen as "check bits".

## *Relationship of Generator and Parity-Check Matrices*

If $G$ and $H$ are generator and parity-check matrices for $\mathcal{C}$, then for every $\mathbf{s}$, we must have $(\mathbf{s}G)H^T = \mathbf{0}$ — since we should only generate valid codewords. It follows that

$$GH^T = \mathbf{0}$$

Furthermore, any $H$ with $N-K$ independent rows that satisfies this is a valid parity-check matrix for $\mathcal{C}$.

Suppose $G$ is in systematic form, so

$$G = [\, I_K \mid P \,]$$

for some $P$. Then we can find a parity-check matrix for $\mathcal{C}$ in systematic form as follows:

$$H = [\, -P^T \mid I_{N-K} \,]$$

since $GH^T = -I_K P + P I_{N-K} = \mathbf{0}$. (Note that $-P^T = P^T$ in $Z_2$.)

## *More on Hamming Distance*

Recall that the Hamming distance, $d(\mathbf{u},\mathbf{v})$, of two codewords $\mathbf{u}$ and $\mathbf{v}$ is the number of positions where $\mathbf{u}$ and $\mathbf{v}$ have different symbols.

This is a proper distance, which satisfies the *triangle inequality*:

$$d(\mathbf{u},\mathbf{w}) \;\leq\; d(\mathbf{u},\mathbf{v}) + d(\mathbf{v},\mathbf{w})$$

Here's a picture showing why:

```
u :  0 1 1 0 0 1 1 0 1 1 1 0
                - - - - - -
v :  0 1 1 0 0 1 0 1 0 0 0 1
          - - -         - -
w :  0 1 1 1 1 0 0 1 0 0 1 0
```

Here, $d(\mathbf{u},\mathbf{v}) = 6$, $d(\mathbf{u},\mathbf{v} = 5)$, and $d(\mathbf{u},\mathbf{w}) = 7$.

## *Minimum Distance and Decoding*

A code's *minimum distance* is the minimum of $d(\mathbf{u},\mathbf{v})$ over all distinct codewords $\mathbf{u}$ and $\mathbf{v}$.

If the minimum distance is at least $2t+1$, a nearest neighbor decoder will always decode correctly when there are $t$ or fewer errors.

Here's why: Suppose the code has distance $d \geq 2t+1$. If $\mathbf{u}$ is sent and $\mathbf{v}$ is received, having no more than $t$ errors, then

- $d(\mathbf{u},\mathbf{v}) \leq t$.
- $d(\mathbf{u},\mathbf{u}') \geq d$ for any codeword $\mathbf{u}' \neq \mathbf{u}$.

From the triangle inequality:

$$d(\mathbf{u},\mathbf{u}') \leq d(\mathbf{u},\mathbf{v}) + d(\mathbf{v},\mathbf{u}')$$

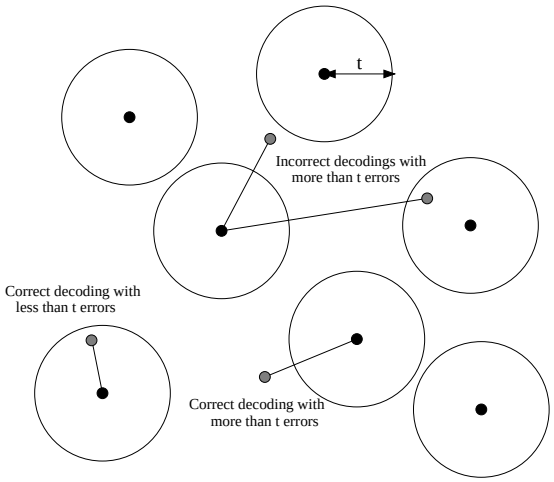
It follows that

$$d(\mathbf{v},\mathbf{u}') \geq d(\mathbf{u},\mathbf{u}') - d(\mathbf{u},\mathbf{v}) \geq d - t \geq (2t+1) - t \geq t+1$$

The decoder will therefore decode correctly to $\mathbf{u}$, at distance $t$, rather than to some other $\mathbf{u}'$.

## *A Picture of Distance and Decoding*

Here's a picture of codewords (black dots) for a code with minimum distance $2t+1$, showing how some transmissions are decoded:




## *Minimum Distance for Linear Codes*

To find the minimum distance for a code with $2^K$ codewords, we will in general have to look at all $2^K(2^K-1)/2$ pairs of codewords.

But there's a short-cut for linear codes...

Suppose two distinct codewords $\mathbf{u}$ and $\mathbf{v}$ are a distance $d$ apart. Then the codeword $\mathbf{u}-\mathbf{v}$ will have $d$ non-zero elements. The number of non-zero elements in a codeword is called its *weight*.

Conversely, if a non-zero codeword $\mathbf{u}$ has weight $d$, then the minimum distance for the code is at least $d$, since $\mathbf{0}$ is a codeword, and $d(\mathbf{u},\mathbf{0})$ is equal to the weight of $\mathbf{u}$.

So the minimum distance of a linear code is equal to the minimum weight of the $2^K-1$ non-zero codewords. (This is useful for small codes, but when $K$ is large, finding the minimum distance is difficult in general.)

## *Examples of Minimum Distance and Error Correction for Linear Codes*

Recall the [5,2] code with the following codewords:

00000 00111 11001 11110

The three non-zero codewords have weights of 3, 3, and 4. This code therefore has minimum distance 3, and can correct any single error.

The single-parity-check code with $N=4$ has the following codewords:

0000 0011 0101 0110
1001 1010 1100 1111

The smallest weight of a non-zero codeword above is 2, so this is the minimum distance of this code. This is too small to guarantee correction of even one error. (Though the presence of a single error can be detected.)