

Linear Error-Correcting Codes and Syndrome Decoding

Tim D. Heilman

6/2/2000

Math 194

Charles Thomas

University of California, Santa Cruz

Abstract

No physical system of transferring data can fully eliminate the possibility that noise will interfere with the transmission. An error-correcting code permits the receiver correctly to identify the intended transmission in spite of noise by introducing additional redundancy into the data before it is transmitted. This paper is intended briefly to survey a particular class of such error-correcting codes, called *linear* codes, as well as one method of decoding such codes, called *syndrome* decoding.

When data is transmitted across a noisy channel, it can be difficult to understand what was meant. The idea of error correcting coding and decoding is to modify the data before transmission so that after the noise has interfered with the message, the receiver may discern what the intended data was without having to request a retransmission.

There are many different approaches to modify the data before transmission, each with various methods to recover the intended message from the interfered-with data. This paper first gives an example to demonstrate the process, and speaks a little about the mathematics going on in any generic error-correcting code. The middle sections of the paper describe one family of error-correcting codes, *linear* error correcting codes, how to encode with them, and some interesting mathematical properties they exhibit. These properties are then used in the final sections to describe one method of decoding received messages, *syndrome* decoding, which works in tandem with any linear error-correcting code.

The mathematics assumed understood by the reader involves only basic algebra and proof technique. Some knowledge of linear algebra, vector spaces, group, and field theory may supply the reader with a more complete mathematical understanding.

The sections into which the paper is divided are namely:

1. Example: The Binary (7, 4)-Hamming Code
2. (De-)Coding as a Function Between Sets
3. Hamming Distance
4. Linear Codes Defined
5. Advantages of Linear Codes
6. Properties of Linear Codes
7. The Generator Matrix and Linear Encoding
8. Dual Codes
9. The Parity Check Matrix
10. Cosets of C
11. Syndrome Decoding

1 Example: The Binary (7, 4)-Hamming Code

This example should hopefully arouse some interest in the subject, and references will be made to it in the rest of the paper. The significance of the (7, 4) in “(7, 4)-Hamming” will be made clear in the section defining linear codes.

Error-correcting codes function over discrete channels, and binary codes are concerned with channels where the standard unit of information is a *bit*.

Definition 1.1 A *bit* is a unit of information which can take one of two values: 0 or 1.

Suppose the sender wishes to send four bits across a channel whose noise may change 0's to 1's or 1's to 0's. To encode the 4-bit *string* (also called a *word*, a *block*, or a *vector*) with the (7, 4)-Hamming code, place the value of bit 1 into area 1 in the following Venn diagram, bit 2 into area 2, and so on.

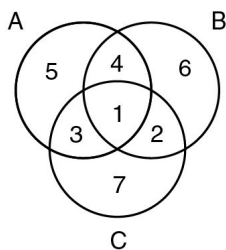


Figure 1.1 The Venn diagram for the binary (7, 4)-Hamming code.

Into the areas 5, 6, and 7 place a 0 or a 1 so that each circle A, B, and C contains an even number of 1's. Read the seven values back out of the Venn diagram into a 7-bit string where the first four values coincide with the 4-bit string to be sent, and bits 5, 6, and 7 are determined by the bits in their corresponding areas in the Venn diagram.

The resulting 7-bit string is the *codeword* which will be sent across the channel. To decode, the receiver should place bits 1 through 7 into their positions in the Venn diagram, determine which circle(s) have an odd number of 1's, *flip* the bit (from a 0 to a 1 or from a 1 to a 0) of the area which influences exactly those circles, then read out the values of areas 1 through 4. If only one error was introduced by the channel, then this coding will always correctly identify the intended 4-bit string, as will be proven later.

For a concrete example, suppose the sender wishes to send 1011. Only circle A has an odd number of 1's, so only position 5 should be a 1. The codeword for 1011 is thus 1011100. Now suppose the sender sends 1011100 but the channel flips bit 3 resulting in 1001100. The receiver places these bits back into their positions in the Venn diagram and counts the 1's in each of the circles. Circle A and circle C have an odd number of 1's, so she assumes area 3 was in error and correctly deduces the intended transmission was 1011. Notice that if an error strikes in bit 5, 6, or 7, the receiver will be able to identify the error, but it doesn't affect her deduction of the intended message.

2 (De-)Coding as a Function Between Sets

More abstractly, consider every possible message the sender might wish to send as elements of a set, and consider the code as the set of codewords which sender and receiver agree are the only strings which will be sent across the channel. In this case, the following definitions are in order:

Definition 2.1 The *alphabet* is the set of symbols from which strings can be composed. The number of symbols in the alphabet is denoted p .

Definition 2.2 The *message space* is the set of all strings that the sender has the option to encode.

Definition 2.3 The *code*, denoted C , is the set of all strings that the sender and receiver agree may be sent across the channel.

Linear error-correcting codes are *block* codes, and as such all codewords have the same length, denoted n .

The act of encoding is the exercise of a bijection from the message space to C , so these sets have the same cardinality, which is denoted M . On the other hand, the act of decoding is the exercise of a function from some superset of C onto P which, if the code is to be effective in correcting errors, is necessarily not injective (the superset is proper).

3 Hamming Distance

In the binary $(7, 4)$ -Hamming code example, we claimed that the receiver could correctly decode to the intended message every possible zero- or single-error transmission of the 7-bit string. In order to prove that claim, it will be necessary to define *Hamming distance* and investigate the process by which the string was decoded, namely, *nearest-neighbor* decoding.

Definition 3.1 The *Hamming distance* of two equal-length words x and y is the number of positions in which x and y differ, and is denoted $d(x, y)$.

Note that this definition of distance applied to the set of strings of length n forms a metric space in that, supposing that x , y , and z are arbitrary strings of length n , the following hold:

1. $d(x, y) \geq 0$;
2. $d(x, y) = 0$ if and only if $x = y$;
3. $d(x, y) = d(y, x)$; and
4. $d(x, z) \leq d(x, y) + d(y, z)$. (The triangle inequality.)

Definition 3.2 The *Hamming distance* of a block code C is $\min_{x, y \in C} d(x, y)$, and is denoted $d(C)$.

The process of decoding in the example was that of *nearest-neighbor* decoding, wherein we choose the codeword with the least Hamming distance from the received word as the intended transmission. In our example, if errors distort a codeword to a non-codeword, there is only one codeword which is Hamming

distance 1 from the non-codeword and that codeword is what we decode the received non-codeword to. Such a statement requires substantiation which is most easily explained after the concept of the parity matrix has been introduced. In the meantime, let us investigate how many errors can be corrected given a code's Hamming distance.

Definition 3.3 A code C is *e-error-correcting* means that the maximum number of errors C can correct is at least e .

Theorem 3.1 Let C be an error-correcting code with nearest-neighbor decoding. C is *e-error-correcting* if and only if $d(C) \geq 2e + 1$.

Proof Suppose $d(C) \geq 2e + 1$. Let x be sent and u received with e or fewer errors. For all codewords $y \neq x$,

$$\begin{aligned} 2e + 1 &\leq d(x, y) && \text{by the definition of } d(C) \\ &\leq d(x, u) + d(u, y) && \text{by the triangle inequality} \\ &\leq e + d(u, y) && \text{since } e \text{ or fewer errors occurred} \end{aligned}$$

Thus $d(u, y) \geq e + 1$: all codewords y other than x are farther from u than x , so nearest-neighbor decoding will correctly associate x with the received word u .

For the other direction, suppose $d(C) < 2e + 1$ and imagine, as follows, a case where e or fewer errors occur yet nearest-neighbor decoding fails to produce the correct result. Let x and y be codewords such that

$$d(x, y) = d(C) = \alpha < 2e + 1$$

We'll consider the case that x is sent, and construct a received word u . If α is even, let u be equal to x in all positions where x is equal to y , as well as in $\frac{\alpha}{2}$ more positions. In the other $\frac{\alpha}{2}$ positions, set u 's symbols equal to y 's symbols in those positions. The result is that $\frac{\alpha}{2}$ errors have occurred, yet $\frac{\alpha}{2} < e + \frac{1}{2}$, so e or fewer errors have occurred, and yet

$$d(x, u) = d(u, y) = \frac{\alpha}{2}$$

so neither neighbor x nor y is nearer to u .

On the other hand, if α is odd, let u be equal to x in all positions where x is equal to y , as well as in $\frac{\alpha-1}{2}$ more positions. In the other $\frac{\alpha+1}{2}$ positions, set u 's symbols equal to y 's symbols in those positions. The result is that $\frac{\alpha+1}{2}$ errors have occurred, yet $\frac{\alpha+1}{2} < e + 1$, so again e or fewer errors have occurred, and yet

$$d(x, u) = \frac{\alpha + 1}{2} > \frac{\alpha - 1}{2} = d(u, y)$$

indicating that y is a nearer neighbor to u than the intended codeword x . \square

This result is what motivates the commentary on Hamming distance in the following sections.

4 Linear Codes Defined

Alphabets, symbols, and strings up until this point have gone without definition. Once in the realm of linear codes, these items can be defined in ways pleasant to a mathematician's sensibilities.

Let's assume that the alphabet from which symbols are drawn is the set of integers mod multiples of p , denoted Z_p . Thus each symbol is a coset of the subgroup of the integers, multiples of p . For convenience, these cosets will be represented by the unique nonnegative integer, less than p , which they contain. For example, we will denote Z_3 as $\{0, 1, 2\}$ rather than $\{Z + 0, Z + 1, Z + 2\}$. Since we would like to define a vector space over Z_p , we will insist that p be prime, so that Z_p is a field.

Now let Z_p^n denote $\overbrace{Z_p \times Z_p \times \cdots \times Z_p}^{n \text{ times}}$. By defining vector addition and scalar multiplication in the way to which we are accustomed, only with the modification that arithmetic be performed modulo p , Z_p^n is an n -dimensional vector space over Z_p . The definition of a string as a vector from this space follows naturally, and strings will continue to be denoted $x_1x_2x_3 \dots x_n$ whereas in vector form they are better known as $(x_1, x_2, x_3, \dots, x_n)$.

Definition 4.1 A *linear code* over Z_p is a subspace of Z_p^n .

That is to say, a subset C of Z_p^n is a linear code if and only if

1. For all \vec{x}, \vec{y} in C , their sum $\vec{x} + \vec{y}$ is also in C , and
2. For all \vec{x} in C and α in Z_p , the scalar product $\alpha\vec{x}$ is in C .

The dimension of a code C will be denoted k , and linear codes with alphabet size p are referred to as p -ary $[n, k]$ codes, or sometimes if the Hamming distance is relevant, p -ary $[n, k, d]$ codes. In the binary $(7, 4)$ -Hamming code example, the codewords form a 4-dimensional subspace of Z_2^7 , as will be exposted in the later sections.

Finally, a standardization of the way we think about errors for block codes will aid in the analysis of linear codes:

Definition 4.2 If C is a block code over Z_p , \vec{x} is transmitted, and \vec{y} received, then the *error pattern* of the transmission is denoted $\vec{\epsilon}$ and defined $\vec{\epsilon} = \vec{y} - \vec{x}$.

5 Advantages of Linear Codes

Usage of linear error-correcting codes is especially slick for the following reasons, further explored in future sections:

$d(C)$ is easily calculated. The process is explained and its efficiency analyzed in the next section.

Encoding is fast with small storage requirements. Compared to a dictionary lookup from the message space to the code, the generator matrix (described in the section bearing the same name) is quick and consumes negligible disk space.

It's easy to identify exactly which errors are correctable. With most e -error-correcting codes, sometimes a received word with greater than e errors can be correctly decoded, depending on the sent word and the error pattern of the transmission. With linear codes, that dependence is only upon the error pattern, as is explained in the next section. Furthermore it is easy to identify those error patterns with greater than e nonzero positions which can still be correctly decoded, as explained in the section on syndrome decoding.

Decoding can be accomplished in minimal time and space. Syndrome decoding is one such technique.

6 Properties of Linear Codes

Firstly, all linear codes contain the zero vector, since 0 is in Z_p and so scalar multiplication of any vector in the code by 0 results in $\vec{0}$ which consequently must also be in the code.

Secondly, it will aid the determination of $d(C)$ for a linear code C , to proceed with the following

Definition 6.1 The *weight* of a vector $\vec{x} \in Z_p^n$, denoted $w(\vec{x})$, is the number of nonzero symbols in \vec{x} .

Theorem 6.1 For a linear code C , $d(C)$ is the least weight of all nonzero codewords.

Proof Suppose $d(C) = d$ and w is the least weight of the nonzero codewords of C . There exist codewords \vec{x} and \vec{y} such that $\vec{x} \neq \vec{y}$ and $d(\vec{x}, \vec{y}) = d$, so $\vec{x} - \vec{y} \neq \vec{0}$ and we have

$$d = d(\vec{x}, \vec{y}) = d(\vec{x} - \vec{y}, \vec{0}) = w(\vec{x} - \vec{y}) \geq w.$$

Yet there also exists some nonzero codeword \vec{z} with weight w , and $\vec{0}$ is in C , so we have

$$w = w(\vec{z}) = d(\vec{z}, \vec{0}) \geq d = d(C)$$

Combining these gives $w = d$. □

Now we can analyze the efficiency savings in calculating $d(C)$ for a linear code versus an arbitrary code. If a code has M codewords, with no magic, every pair of codewords would need to be compared to determine a minimum Hamming distance, for a total of $\binom{M}{2} = \frac{1}{2}M(M-1)$ comparisons. With a linear code and the method described above, only examinations of the $M-1$ nonzero codewords need be performed to determine the minimum Hamming distance.

To conclude this section is a result that excludes the sent word from consideration in determining the correctability of a received word.

Theorem 6.2 *For a linear code using nearest-neighbor decoding, whether a received word \bar{y} is uniquely correctly decodable depends only on the error pattern $\bar{\varepsilon}$, not on the transmitted codeword \bar{x} .*

Proof By contradiction, we will assume that given an error pattern $\bar{\varepsilon}$, there are two codewords \bar{x} and \bar{x}' such that $\bar{y} = \bar{x} + \bar{\varepsilon}$ is correctly decoded, but $\bar{y}' = \bar{x}' + \bar{\varepsilon}$ is not. Thus there must be some codeword \bar{x}'' such that $d(\bar{x}'', \bar{y}') \leq d(\bar{x}', \bar{y}')$. Let $\bar{\varepsilon}'$ be the error pattern associated with the transmission of that \bar{x}'' that results in \bar{y}' : $\bar{\varepsilon}' = \bar{y}' - \bar{x}''$. So defined, $d(\bar{x}'', \bar{y}') \leq d(\bar{x}', \bar{y}')$ can be rewritten $w(\bar{\varepsilon}') \leq w(\bar{\varepsilon})$. Now a bit of explanation to show the direction we're going: if we can show that in the same way that there exists an \bar{x}'' which is $\bar{\varepsilon}'$ away from \bar{y}' , there also exists a \bar{z} in the code which is $\bar{\varepsilon}'$ away from \bar{y} , then we've got the contradiction since \bar{z} would be closer to \bar{y} than \bar{x} , yet \bar{y} supposedly decoded to \bar{x} . To this end, define \bar{z} to be $\bar{y} - \bar{\varepsilon}'$:

$$\begin{aligned}\bar{z} &= \bar{y} - \bar{\varepsilon}' = \bar{x} + \bar{\varepsilon} - \bar{\varepsilon}' = \bar{x} + (\bar{y}' - \bar{x}') - (\bar{y}' - \bar{x}'') \\ &= \bar{x} - \bar{x}' + \bar{x}''\end{aligned}$$

So \bar{z} is in C , since C is linear, yet

$$\begin{aligned}d(\bar{z}, \bar{y}) &= w(\bar{\varepsilon}') && \text{by the definition of } \bar{z} \\ &\leq w(\bar{\varepsilon}) && \text{since } w(\bar{\varepsilon}') \text{ is also } d(\bar{y}', \bar{x}''), \\ & && \bar{\varepsilon} = \bar{y}' - \bar{x}', \\ & && \text{and } \bar{y}' \text{ was decoded to } \bar{x}'', \text{ not } \bar{x}' \\ &= d(\bar{x}, \bar{y}) && \text{by their definitions}\end{aligned}$$

This implies that \bar{z} is as close or closer to \bar{y} than \bar{x} , contradicting that \bar{y} was correctly decodable to \bar{x} . \square

7 The Generator Matrix and Linear Encoding

Generally, and in the case of the (7,4)-Hamming code example, the message space is taken to be all words of length k , denoted Z_p^k . Since C is an k -dimensional subspace of Z_p^n , we can find a basis for C composed of k linearly independent codewords. The matrix formed from these k row vectors is a *generator matrix*, G , for the code C , so called because its span is C . Continuing the (7,4)-Hamming code example, then, we have the generator matrix

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

and to encode 1011, the multiplication is carried out as follows:

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

which happily is the same result as when the Venn diagram is used.

In the excitement to get to the example, a few details were glossed over. Firstly, while any k linearly independent codewords from C would suffice to create a generator matrix whose span is C , the Venn diagram defined not only the code but also the encoding bijection. With the generator matrix approach, since the rows span C , it is apparent that the encoding is surjective, however it may not be crystal clear that the encoding process is injective, which follows from the linear independence of the rows.

Theorem 7.1 *Generator matrix encoding is injective.*

Proof Suppose G is composed of the k linearly independent codewords $\vec{g}_1, \vec{g}_2, \dots, \vec{g}_k$. Further suppose two messages $\vec{a} = a_1 a_2 \dots a_k$ and $\vec{b} = b_1 b_2 \dots b_k$ result in the same encoding: $\vec{a}G = \vec{b}G$. Then

$$\sum_{i=1}^k a_i \vec{g}_i = \sum_{i=1}^k b_i \vec{g}_i \text{ which implies } \sum_{i=1}^k (a_i - b_i) \vec{g}_i = \vec{0}$$

which by the linear independence of $\vec{g}_1, \vec{g}_2, \dots, \vec{g}_k$, implies that for all $1 \leq i \leq k$, the difference $a_i - b_i = 0$, so $a_i = b_i$, that is $\vec{a} = \vec{b}$. \square

However, while regardless of the generator matrix's construction the encoding will be bijective, different choices of the basis vectors result in different bijections. A particularly convenient bijection which leaves the first k symbols untouched occurs when the generator matrix is in *standard form*, that is, the first k columns form the $k \times k$ identity matrix, as in the example.

8 Dual Codes

Since we have the definition of codewords as vectors in Z_p^n , it's natural to define the inner product of two codewords $\vec{x} = x_1 x_2 \dots x_n$ and $\vec{y} = y_1 y_2 \dots y_n$ in the natural way: $\vec{x} \cdot \vec{y} = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$, again with all multiplication and addition done modulo p . Further, allow two vectors \vec{x} and \vec{y} to be called orthogonal if and only if $\vec{x} \cdot \vec{y} = 0$. With these definitions, we can look at the set of all vectors in Z_p^n which are orthogonal to all vectors in C , called the *dual code* of C and denoted C^\perp .

Perhaps surprisingly, regardless of whether C is linear, C^\perp is always linear.

Theorem 8.1 *If C is an arbitrary block code, then C^\perp is linear.*

Proof Suppose \vec{x} and \vec{y} are two vectors in C^\perp and α and β are elements of Z_p . Let \vec{z} be any codeword in C .

$$\begin{aligned}(\alpha\vec{x} + \beta\vec{y}) \cdot \vec{z} &= \alpha(\vec{x} \cdot \vec{z}) + \beta(\vec{y} \cdot \vec{z}) \\ &= \alpha(\vec{0}) + \beta(\vec{0}) \\ &= \vec{0}\end{aligned}$$

and so $\alpha\vec{x} + \beta\vec{y}$ is also in C^\perp . □

Perhaps still more surprisingly, if we may assume C is linear, then C^\perp is linear of dimension $n - k$. This result requires the additional transitional

Definition 8.1 A $k \times n$ generator matrix is in *nearly standard form* if k of its columns are the k columns of I_k , the $k \times k$ identity matrix.

We would like to place a generator matrix for a linear code C into nearly standard form without losing the property that it generates the same code C . The standard row operations (permutation of the rows, replacement of a row by one of its non-zero multiples, and replacement of a row by the sum of itself and any multiple of another row) allow us to do this. The row operations do not alter the code the matrix generates, since they always result in linearly independent vectors of C . Furthermore, the only way nearly standard form would be unattainable using the standard row operations would be if more than k columns were composed of all 0's. This would imply that k linearly independent rows have fewer than k meaningful columns — a contradiction. Thus every linear code C has a generator matrix in nearly standard form.

We're almost ready to prove the dimensionality of C^\perp , but first,

Theorem 8.2 For a linear code C with generator matrix G composed of basis codewords $\vec{g}_1, \vec{g}_2, \dots, \vec{g}_k$, a vector \vec{x} in Z_p^n is in C^\perp if and only if \vec{x} is orthogonal to every row of G .

Proof One direction is trivial: since every row of G is a codeword, if \vec{x} is in C^\perp then \vec{x} will be orthogonal to every row of G .

Suppose \vec{x} is orthogonal to \vec{g}_i for all $1 \leq i \leq k$. Let \vec{y} be any codeword. Then there exist scalars $\alpha_1, \alpha_2, \dots, \alpha_k$ such that $\vec{y} = \sum_{i=1}^k \alpha_i \vec{g}_i$. Thus

$$\vec{x} \cdot \vec{y} = \vec{x} \cdot \sum_{i=1}^k \alpha_i \vec{g}_i = \sum_{i=1}^k \alpha_i (\vec{x} \cdot \vec{g}_i) = \sum_{i=1}^k \alpha_i (0) = 0$$

and so \vec{x} is in C^\perp . □

We are now prepared for

Theorem 8.3 If C is a linear $[n, k]$ code over Z_p , then the dual code C^\perp of C is a linear $[n, n - k]$ code.

Proof We've already seen that C^\perp is linear, so all that remains to show is that it is of dimension $n - k$. Let $W = \{w_1, w_2, \dots, w_k\}$ be the set of indexes where a generator matrix G of C in nearly standard form has a column of I_k subject to the condition that the w_i^{th} column of G is the i^{th} column of I_k . For example, G might look like this:

$$G = \begin{pmatrix} 1 & 2 & \cdots & w_2 & \cdots & w_1 & \cdots & w_k & \cdots & n \\ g_{11} & g_{12} & \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & 1 & \cdots & 0 & \cdots & 0 & \cdots & g_{2n} \\ g_{31} & g_{32} & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & g_{3n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots & g_{kn} \end{pmatrix}$$

The idea is, in order for a vector \vec{x} to be in C^\perp , it must be orthogonal to every row of G . For each of the positions not indexed by W , choose \vec{x} 's value arbitrarily. Then for each of the positions w_i indexed by W , figure out what \vec{x} 's value must be to enforce orthogonality with the i^{th} row in G . In mathematical notation,

$$C^\perp = \left\{ (x_1, x_2, \dots, x_n) \in Z_p^n \mid x_{w_i} + \sum_{j \notin W} g_{ij} x_j = 0, \text{ for } i = 1, 2, \dots, k \right\}$$

Thus choosing the values for x_j for all $j \notin W$ uniquely specifies the values for x_{w_i} for all $w_i \in W$. There are p^{n-k} ways the x_j may be chosen, so $|C^\perp| = p^{n-k}$ which implies $\dim(C^\perp) = n - k$. \square

This result gives rise to a pleasant tightness in the gears of this machinery:

Theorem 8.4 For any linear code C , $(C^\perp)^\perp = C$.

Proof Suppose $C \subseteq Z_p^n$ has dimension k over Z_p . Let \vec{x} be a codeword of C . Then for all \vec{y} in C^\perp , $\vec{x} \cdot \vec{y} = 0$ so $\vec{x} \in (C^\perp)^\perp$, which since \vec{x} was arbitrary implies $C \subseteq (C^\perp)^\perp$. But C^\perp is a linear code of dimension $n - k$ over Z_p , so $(C^\perp)^\perp$ is a linear code of dimension $n - (n - k) = k$ over Z_p . Since $C \subseteq (C^\perp)^\perp$ and they are both of dimension k , $C = (C^\perp)^\perp$. \square

9 The Parity Check Matrix

Returning to the (7,4)-Hamming code example, rather than specifying the code with a generator matrix, a more straightforward definition is possible by emulating the equations the Venn diagram is used to solve. Any codeword $\vec{x} = x_1 x_2 \dots x_7$ generated with the Venn diagram must have an even number of 1's in each circle. In other words, C is fully specified as those \vec{x} 's which that satisfy the following equations:

$$\begin{array}{rcccccccc}
x_1 & & & + & x_3 & + & x_4 & + & x_5 & & & = & 0 \\
x_1 & + & x_2 & & & & + & x_4 & & & + & x_6 & = & 0 \\
x_1 & + & x_2 & + & x_3 & & & & & & & + & x_7 & = & 0
\end{array}$$

or, in matrix form, $xH^T = \vec{0}$, where

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

and H^T denotes the transpose of H .

More rigorously,

Definition 9.1 H is a *parity check matrix* for a linear code C if and only if

1. its rows are independent,
2. $C = \{\vec{x} \in Z_p^n \mid \vec{x}H^T = \vec{0}\}$. (That is, C is the null space of H .)

This name is appropriate in that for the binary case it checks that the parity of the cardinality of selected subsets of the symbols in a codeword is even.

Before investigating the intricacies of the parity check matrix, it would be good to prove a couple of theorems about the (7,4)-Hamming code which have until now been assumed without proof.

Theorem 9.1 *The (7,4)-Hamming code is a linear code.*

Proof Suppose H is the parity check matrix for the (7,4)-Hamming code C , and $\vec{x}, \vec{y} \in C$, that is, $\vec{x}H^T = \vec{0}$ and $\vec{y}H^T = \vec{0}$. Let $\alpha, \beta \in Z_p$. Then $(\alpha\vec{x} + \beta\vec{y})H^T = \alpha\vec{x}H^T + \beta\vec{y}H^T = \alpha\vec{0} + \beta\vec{0} = \vec{0}$, so $\alpha\vec{x} + \beta\vec{y} \in C$ and C is linear. \square

Theorem 9.2 *The (7,4)-Hamming code C with parity check matrix H has Hamming distance 3, and so by Theorem 3.1, is 1-error-correcting.*

Proof We will prove this by showing that all nonzero codewords in C have weight at least 3. Suppose $\vec{x} = x_1x_2 \dots x_n \in C$ has weight 1, that is, $x_i = 1$ for some i , with $x_j = 0$ for all $j \neq i$. This contradicts that $\vec{x}H^T = \vec{0}$ since no i^{th} column of H is all zeros, as it would need to be. Next suppose $\vec{x} \in C$ has weight 2, and let $x_i = x_j = 1$ with $x_l = 0$ for all l other than j and i . Denoting the s^{th} row of H by $h_{s1}h_{s2} \dots h_{sn}$, we have, since \vec{x} is orthogonal to each row in H , that for all $1 \leq s \leq n-k$, $h_{si} + h_{sj} = 0$ which under modulo 2 arithmetic means $h_{si} = h_{sj}$. This would mean that some two columns of H were identical, which no two are, and we've reached a contradiction. Finally, consider the codeword 0100011, which satisfies the parity check matrix. It has weight 3, and so we've proven $d(C) = 3$. \square

Returning to the general case, the parity check matrix is intimately related to the dual code.

Theorem 9.3 H is a parity check (p.c.) matrix for a linear code C if and only if it is a generator matrix for C^\perp .

Proof Suppose C is an $[n, k]$ linear code. In one direction, H is a p.c. matrix for C

$$\begin{aligned} \Rightarrow C &= \text{Null}(H) && \text{by the definition of a p.c. matrix} \\ \Rightarrow k &= n - \dim(\text{Im}(H)) && \text{by the rank-nullity theorem} \\ \Rightarrow \dim(\text{Im}(H)) &= n - k \\ \Rightarrow \dim(\text{Im}(H)) &= \dim(C^\perp) && \text{by Theorem 8.3} \end{aligned}$$

Yet $\text{Im}(H) \subseteq C^\perp$, so $\text{Im}(H) = C^\perp$, which combined with the linear independence of the rows of H implies that H is a generator matrix for C^\perp .

In the other direction, H is a generator matrix for C^\perp

$$\begin{aligned} \Rightarrow \text{rows of } H &\text{ are independent and } C^\perp = \text{Im}(H) \\ \Rightarrow \text{for all } \vec{x} \in C, \vec{x} \cdot (\text{Any linear combination of rows of } H) &= 0 \\ \Rightarrow C &\subseteq \text{Null}(H) \end{aligned}$$

yet

$$\begin{aligned} \dim(C) &= k = n - (n - k) \\ &= n - \dim(C^\perp) && \text{by Theorem 8.3} \\ &= n - \dim(\text{Im}(H)) && \text{since } H \text{ is a generator matrix for } C^\perp \\ &= \dim(\text{Null}(H)) && \text{by the rank-nullity theorem.} \end{aligned}$$

Combining $C \subseteq \text{Null}(H)$ with $\dim(C) = \dim(\text{Null}(H))$ gives $C = \text{Null}(H)$. We already know C 's rows are independent, and so H is a parity check matrix for C . \square

10 Cosets of C

The algebraic structure of a linear code is the foundation of the very fast syndrome decoding algorithm.

Firstly, Z_p^n satisfies the five properties of an abelian group with respect to vector addition: vectors in Z_p^n sum to other vectors in Z_p^n , vector addition is associative and commutative, $\vec{0}$ is an additive identity, and every vector has an additive inverse, namely, wherein every ordinate is subtracted from $p \pmod{p}$.

If C is a p -ary $[n, k]$ linear code, it is obviously finite and by the definition of linearity we have that C is closed under vector addition, and so is a subgroup of Z_p^n . Its cosets are $C + \vec{\varepsilon}$ for error patterns $\vec{\varepsilon} \in Z_p^n$, and Z_p^n can be expressed as the disjoint union of these cosets. How many cosets are there? Since there are p^n vectors in Z_p^n and p^k vectors in C and therefore in each coset of C , there must be $p^n/p^k = p^{n-k}$ cosets.

11 Syndrome Decoding

The following slick technique for decoding linear codes grew out of a simpler process called *Stein array decoding*, the implementation of which, by requiring the storage and examination of every coset of C , takes dramatically more time and space.

Definition 11.1 If H is the parity check matrix for a p -ary $[n, k]$ linear code C and \vec{x} is a vector in Z_p^n , then the *syndrome* of \vec{x} , denoted $\text{syn}(\vec{x})$, is $\vec{x}H^T$.

By the definition of H , $\text{syn}(\vec{x}) = \vec{0}$ if and only if $\vec{x} \in C$.

A standard interpretative result from group theory aids in the theorem that follows it.

Theorem 11.1 If \vec{x} and \vec{y} are vectors in Z_p^n , then they are in the same coset of the p -ary $[n, k]$ linear code C if and only if $\vec{x} \equiv \vec{y} \pmod{C}$.

Proof Suppose $\vec{x}, \vec{y} \in C + \vec{z}$ for some $\vec{z} \in Z_p^n$. Then there exist vectors \vec{a} and \vec{b} in C such that $\vec{x} = \vec{a} + \vec{z}$ and $\vec{y} = \vec{b} + \vec{z}$, so $\vec{x} - \vec{y} = \vec{a} - \vec{b}$ which is a vector in C since C is linear. Thus $\vec{x} \equiv \vec{y} \pmod{C}$.

On the other hand, $\vec{x} - \vec{y} \in C$ implies that there exists an $\vec{a} \in C$ such that $\vec{x} - \vec{y} = \vec{a}$, or $\vec{x} = \vec{a} + \vec{y}$, so $\vec{x} \in C + \vec{y}$. We know $\vec{y} \in C + \vec{y}$ since C is linear and so contains $\vec{0}$, and so \vec{x} and \vec{y} are in the same coset of C . \square

Theorem 11.2 If \vec{x} and \vec{y} are vectors in Z_p^n , then $\text{syn}(\vec{x}) = \text{syn}(\vec{y})$ if and only if \vec{x} and \vec{y} are in the same coset of C .

Proof A chain of double implications will suffice: $\text{syn}(\vec{x}) = \text{syn}(\vec{y}) \Leftrightarrow \vec{x}H^T = \vec{y}H^T \Leftrightarrow (\vec{x} - \vec{y})H^T = \vec{0} \Leftrightarrow \vec{x} - \vec{y} \in C \Leftrightarrow \vec{x} \equiv \vec{y} \pmod{C} \Leftrightarrow \vec{x}$ and \vec{y} are in the same coset of C . \square

We would like to implement nearest-neighbor decoding on the code by observing into what coset of the code a received word falls. Suppose \vec{x}' was sent and \vec{y} received via error pattern $\vec{\varepsilon}'$, that is, $\vec{y} = \vec{x}' + \vec{\varepsilon}'$. Obviously, $\vec{y} \equiv \vec{\varepsilon}' \pmod{C}$ and so \vec{y} and $\vec{\varepsilon}'$ lie in the same coset and have the same syndrome. Syndrome decoding assumes $\vec{\varepsilon}'$ is a least-weight vector in $C + \vec{y}$, calls it $\vec{\varepsilon}$, and decodes \vec{y} to $\vec{x} = \vec{y} - \vec{\varepsilon}$. If all went well, $\vec{\varepsilon}' = \vec{\varepsilon}$ and $\vec{x}' = \vec{x}$.

Theorem 11.3 Syndrome decoding is a method of nearest-neighbor decoding.

Proof We know $\vec{x} \in C$ since $\vec{\varepsilon} \in C + \vec{y}$ implies $\vec{y} - \vec{\varepsilon}$ will be in C . Suppose there exists some \vec{z} in C such that $d(\vec{y}, \vec{z}) < d(\vec{y}, \vec{x})$. Then setting $\vec{\varepsilon}''$ equal to $\vec{y} - \vec{z}$ results in $w(\vec{\varepsilon}'') < w(\vec{\varepsilon})$, yet $\vec{z} \in C \Rightarrow -\vec{z} \in C \Rightarrow \vec{\varepsilon}'' = -\vec{z} + \vec{y} \in C + \vec{y}$, contradicting that $\vec{\varepsilon}$ was the least-weight vector in $C + \vec{y}$. \square

Consider the case where there are multiple candidates for a least-weight vector in a coset, and we choose the vector \vec{x} with weight w to be associated

with that coset. Any other least-weight candidate \vec{y} also has weight w , yet any received word which is afflicted with the error pattern \vec{y} will be decoded as though it were hit with the error pattern \vec{x} , so the code fails to correct some cases of w errors, and so is not w -error-correcting. On the other hand, \vec{x} is a perfectly correctable error pattern, and so by virtue of the linearity of the code, we know exactly which error patterns are correctable even when their weights exceed e , where C is an e -error-correcting code.

To implement syndrome decoding of a linear code, identify each coset with some least-weight vector it contains, then calculate and store the syndromes of those vectors. When a word is received, calculate its syndrome, scan the list of error-pattern syndromes for a match, and subtract the corresponding error pattern from the received word.

To conclude the paper, we'll implement syndrome decoding for the (7, 4)-Hamming code. We're lucky in that the identification of the cosets and their least-weight members is utterly trivial.

By the result of section 10, C has $2^{7-4} = 8$ cosets, and from the Hamming distance arguments we know that any single error can be corrected, that is, every error pattern of weight 1 resides in its own coset. There are 7 possible such error patterns and the code itself forms the 8th coset, so the least weight vectors of each coset are $\vec{0}$ and all vectors of weight 1. Their syndromes are as follows:

$$0000000 \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 000; \quad 1000000 \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 111;$$

and similarly every other syndrome is simply the corresponding row in H^T :

$$\begin{aligned} \text{syn}(0000000) &= 000 \\ \text{syn}(1000000) &= 111 \\ \text{syn}(0100000) &= 011 \\ \text{syn}(0010000) &= 101 \\ \text{syn}(0001000) &= 110 \\ \text{syn}(0000100) &= 100 \\ \text{syn}(0000010) &= 010 \\ \text{syn}(0000001) &= 001 \end{aligned}$$

Suppose 1011 is to be sent, which we saw in section 7 encodes to 1011100. Lightning strikes the cable and the receiver hears 1001100. She calculates the syndrome:

$$1001100 \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 101$$

and 101 is the syndrome associated with error pattern 0010000. Our protagonist correctly deduces the intended transmission: $1001100 - 0010000 = 1011100$.

References

- [1] J. Baylis, *Error-Correcting Codes*, Chapman & Hall, London, 1998, Main Reference.
- [2] I. N. Herstein, *Topics in Algebra*, John Wiley & Sons, New York, 1975.
- [3] R. Hill, *A First Course in Coding Theory*, Clarendon Press, Oxford, 1986.
- [4] D. Lay, *Linear Algebra and Its Applications*, Addison-Wesley, Reading, Massachusetts, 1994.
- [5] J. C. A. van der Lubbe, *Information Theory*, Cambridge University Press, Cambridge, 1997.
- [6] F. J. Macwilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977, pp. 1-37.