Intro to Coding Theory

In the beginning ...

Coding theory originated with the advent of computers. Early computers were huge mechanical monsters whose reliability was low compared to the computers of today. Based, as they were, on banks of mechanical relays, if a single relay failed to close the entire calculation was in error. The engineers of the day devised ways to detect faulty relays so that they could be replaced. While R.W. Hamming was working for Bell Labs, out of frustration with the behemoth he was working with, the thought occurred that if the machine was capable of knowing it was in error, wasn't it also possible for the machine to correct that error. Setting to work on this problem Hamming devised a way of encoding information so that if an error was detected it could also be corrected. Based in part on this work, Claude Shannon developed the theoretical framework for the science of coding theory.

The Coding Idea

What we have called Coding Theory, should more properly be called the Theory of Error-Correcting Codes, since there is another aspect of Coding Theory which is older and deals with the creation and decoding of secret messages. This field is called Cryptography and we will not be interested in it. Rather, the problem that we wish to address deals with the difficulties inherent with the transmission of messages. More particularly, suppose that we wished to transmit a message and knew that in the process of transmission there would be some altering of the message, due to weak signals, sporadic electrical bursts and other naturally occurring noise that creeps into the transmission medium. The problem is to insure that the intended message is obtainable from whatever is actually received.

The Repeat Code

One simple approach to this problem is what is called a repeat code. For instance, if we wanted to send the message BAD NEWS, we could repeat each letter a certain number of times and send, say,

BBBBBAAAAADDDDD NNNNNEEEEEWWWWWSSSSS.

Even if a number of these letters got garbled in transmission, the intended message could be recovered from a received message that might look like

BBBEBFAAAADGDDD . MNNNTEEEEEWWWSWRRSSS,

by a process called ***majority decoding***, which in this case would mean that for each block of 5 letters the intended letter is the one which appears most frequently in the block.

Probability

The problem with this approach is economical, the repeat code is not very efficient. The increased length of the transmitted code, and thus the increased time and energy required to transmit it, is necessary in order to be able to decode the message properly, but how efficiently a coding procedure uses this increase depends upon the coding scheme. Suppose, in our example, that the probability that a letter is garbled in transmission is $p = 0.05$ and so $q = 1 - p = 0.95$ is the probability that a letter is correctly received. Without any coding, the probability of our 8 letter (spaces included) message being correctly received is

$$q^8 = (.95)^8 = 0.66.$$

(In this calculation we are assuming that the error in transmitting a single symbol is independent of which position the symbol is in. This is a common simplifying assumption ... which may not be appropriate in real world situations.)

Probability

Using the repeat code, the probability of correctly decoding a given letter from a block of 5 symbols is

$$q^5 + 5q^4p + 10q^3p^2$$

since there are three ways to decode correctly – 1) all the symbols are correct, 2) one symbol is incorrect (5 ways this can happen) or 3) two symbols are incorrect (10 ways this can happen) [notice that these are just terms in the expansion of $(q+p)^5$]. So we obtain

$$(.95)^5 + 5(.95)^4(.05) + 10(.95)^3(.05)^2 = 0.9988$$

and thus the probability of getting the correct eight letter message after decoding is $(0.9988)^8 = 0.990$, clearly a great increase over the non-coded message (= 0.66), but this 1% probability of getting the wrong message might not be acceptable for certain applications.

Terminology

To increase the probability of decoding the correct message with this type of code we would have to increase the number of repeats - a fix which may not be desirable or even possible in certain situations. However, as we shall see, other coding schemes could increase the probability to 0.9999 without increasing the length of the coded message.

Before leaving the repeat codes to look at other coding schemes, let us introduce some terminology. Each block of repeated symbols is called a ***code word***, i.e., a code word is what is transmitted in place of one piece of information in the original message. The set of all code words is called a ***code***. If all the code words in a code have the same length, then the code is called a ***block code***. The repeat codes are block codes.

Detection and Correction

One feature that a useful code must have is the ability to detect errors. The repeat code with code words having length 5 can always detect from 1 to 4 errors made in the transmission of a code word, since any 5 letter word composed of more than one letter is not a code word. However, it is possible for 5 errors to go undetected (how?). We would say that this code is ***4-error detecting***. Another feature is the ability to correct errors, i.e., being able to decode the correct information from the error riddled received words. The repeat code we are dealing with can always correct 1 or 2 errors, but may decode a word with 3 or more errors incorrectly (how?), so it is a ***2-error correcting code***.

A Single Error Correcting Code

Before we look at the general theory of error correcting codes, let's consider another simple example. This is the simplest code that Hamming devised in his 1947 paper, *Self-Correcting Codes – Case 20878, Memorandum 1130-RWH-MFW*, Bell Telephone Laboratories. [The first paper on error correcting codes]

The information that is to be encoded is a collection of binary strings of length t^2. The code words will be binary strings of length $(t+1)^2$, so $2t+1$ check digits will be added to the information data. To compute the check digits, arrange the data into a $t \times t$ array, add a mod 2 check sum digit to each row and column (including one for the row and column of check sums). The code word is the string obtained by writing out the rows of the extended array.

A Single Error Correcting Code

As a small example, consider the t = 3 case. We would code up the data 011010111 by

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

Calculate the check digits,

and produce the codeword 0110010111111100.

When a word is received, it is arranged in a 4×4 array, the row and column check sums are recalculated and compared with the entries in the last column and last row. Differences indicate the row and column of any single error, which can then be corrected!

A Single Error Correcting Code

Suppose that the codeword 0110010111111100 is sent, but the word 0110011111111100 is received. We decode this:

0 1 1 0 0
0 1 1 1 0 ←
1 1 1 1 1
1 1 0 0 0
1 1 1 0
↑

By recalculating the check sums, we locate the row and column of the error. Clearly, any single error in a data digit will change only its row and column checks and so will be located. If an error is spotted only in a row but not a column (or vice versa) the error has occurred in a check digit and so, can again be corrected.

Distance

Suppose that you knew that an English word was transmitted and you had received the word SHIP. If you suspected that some errors had occurred in transmission, it would be impossible to determine what word was really transmitted - it could have been SKIP, SHOP, STOP, THIS, actually any four letter word. The problem here is that English words are in a sense "too close" to each other. What gives a code its error correcting ability is the fact that the code words are "far apart". We shall make this distance idea more precise.

Assumptions

First of all we shall restrict our horizons and only consider block codes, so all codewords will have the same length. There are other types of codes, with variable length codewords, which are used in practice, but their underlying theory is quite different from that of the block codes.

Our second assumption is that the symbols used in our codewords will come from a *finite* alphabet Σ. Typically, Σ will just consist of the integers $\{0,1,...,k-1\}$ when we want our alphabet to have size k, but there will be other alphabets used in our work. Note that unless otherwise specified, these numbers are only being used as symbols – they have no arithmetic properties.

Settings

A ***code*** with codewords of length n, made up from an alphabet Σ of size k, is then just a subset of $\Sigma^n = \Sigma \times \Sigma \times \ldots \times \Sigma$, that is the set of n-tuples with entries from Σ. Since the actual alphabet is important (only its size) we will denote this "space" by

$$\mathbf{V(n,k) := \Sigma^n}$$

The elements of V(n,k) are called ***words***.

In those situations where we wish to use algebraic properties of the alphabet, we modify the notation by replacing the parameter k by the name of the algebraic structure we are using. Thus,

$$V(n, \mathbb{Z}_4)$$

indicates that the n-tuples are made up from the elements of $\mathbb{Z}_4$ and that we can add n-tuples componentwise using the operations of $\mathbb{Z}_4$ (namely, adding mod 4). [Technically, this space is known as a $\mathbb{Z}_4$ -***module*** since the alphabet is a ring.]

Settings

The most important setting occurs when the alphabet is a finite field. To indicate this setting we will use the notation

V[n,q]

implying that the alphabet is the finite field with q elements (as we shall see later, q must be a prime or power of a prime). In this case, V[n,q] is a *vector space* (with scalars from the finite field).

Many of the codes we will encounter, especially those that have been useful in computer science, have the vector space setting V[n,2]. These are often called ***binary codes*** since the alphabet is the binary field consisting of only two elements. Codes in V[n,3] are called ***ternary codes***, and, in general, codes in V[n,q] are called ***q-ary codes***.

Hamming Distance

The ***Hamming distance*** between two words in V(n,k) is the number of places in which they differ.

So, for example, the words (0,0,1,1,1,0) and (1,0,1,1,0,0) would have a Hamming distance of 2, since they differ only in the 1st and 5th positions. In V(4,4), the words (0,1,2,3) and (1,1,2,2) also have distance 2.

This Hamming distance is a ***metric*** on V(n,k), i.e., if d(x,y) denotes the Hamming distance between words x and y, then d satisfies:

1) $d(x,x) = 0$
2) $d(x,y) = d(y,x)$, and
3) $d(x,y) + d(y,z) \geq d(x,z)$. (triangle inequality)

Hamming Distance

The first two of these properties are obvious, but the triangle inequality requires a little argument (this is a homework problem).

Since we will only deal with the Hamming distance (there are other metrics used in Coding Theory), we will generally omit the Hamming modifier and talk about the *distance* between words.

Minimum Distance

The ***minimum distance*** of a code C is the smallest distance between any pair of distinct codewords in C. It is the minimum distance of a code that measures a code's error correcting capabilities. If the minimum distance of a code C is 2e + 1, then C is a 2e-error detecting code since 2e or fewer errors in a codeword will not get to another codeword and is an e-error correcting code, since if e or fewer errors are made in a codeword, the resulting word is closer to the original codeword than it is to any other codeword and so can be correctly decoded (***maximum-likelihood decoding***).

In the 5-repeat code of V(5,4) (codewords: 00000, 11111, 22222, and 33333) the minimum distance is 5. The code detects 4 or fewer errors and corrects 2 or fewer errors as we have seen.

Weight of a Word

We always assume that 0 is one of the symbols in our alphabet.

The ***weight*** of a word is the number of non-zero components in the word. Alternatively, the weight is the distance of the word from the zero word.

In V(6,6) the word (0,1,3,0,1,5) has weight 4.

When we are working with an alphabet in which one can add and subtract then there is a relationship between distance and weight,

$$\mathbf{d(x,y) = wt\ (x - y)},$$

since whenever a component of x and y differ, the corresponding component of x – y will not be 0.

(n, M, d) – Codes

Let C be a code in V(n,k). If C has M codewords and minimum distance d, we sometimes refer to it as an ***(n,M,d)-code***.

For fixed n, the parameters M and d work against one another - the bigger M, the smaller d and vice versa. This is unfortunate since for practical reasons we desire a large number of codewords with high error correcting capability (large M and large d). The search for “good” codes always involves some compromise between these parameters.

Covering Radius

Since V(n,k) has a metric defined on it, it makes sense to talk about spheres centered at a word with a given radius. Thus,

$$S_r(x) = \{y \in V(n,k) \mid d(x,y) \leq r\}$$

is the ***sphere*** of radius r centered at x.

The ***covering radius*** of a code C is the smallest radius s so that

$$V(n,k) \subseteq \bigcup_{x \in C} S_s(x)$$

i.e., every word of the space is contained in some (at least one) sphere of radius s centered at a codeword.

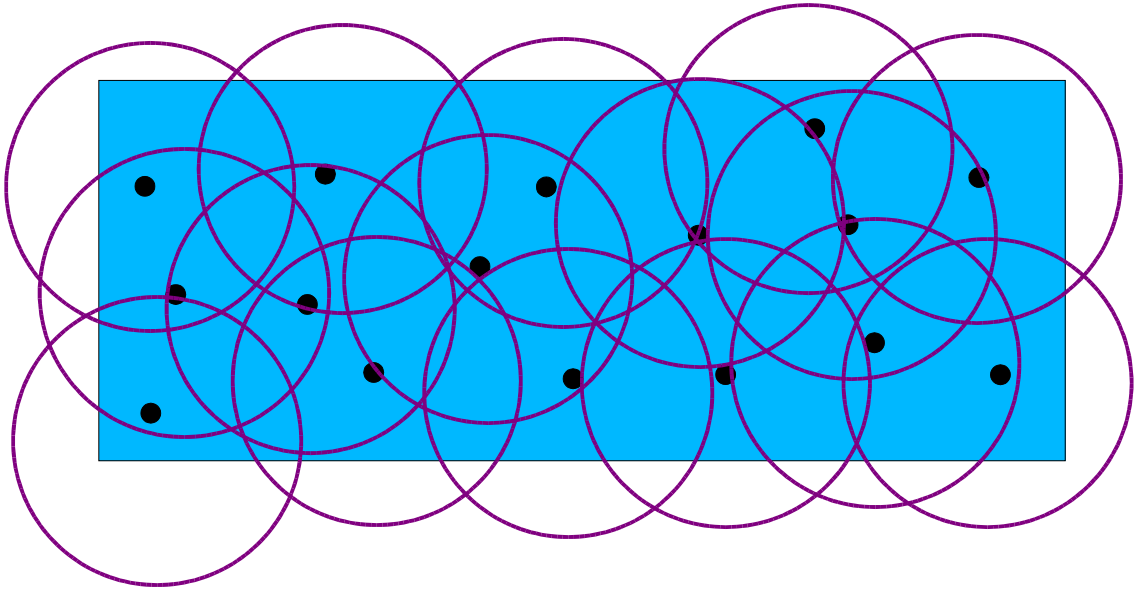
Packing Radius

The ***packing radius*** of a code C is the largest radius t so that the spheres of radius t centered at the code words are disjoint.

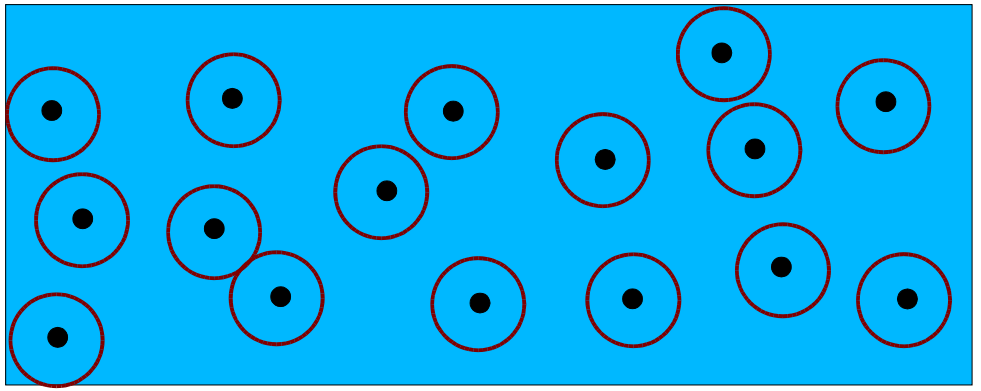
$$S_t(x) \cap S_t(y) = \emptyset \quad \forall\, x \neq y \in C$$

Clearly, $t \leq s$. When $t = s$, we say that C is a ***perfect code***. While perfect codes are very efficient codes, they are very rare – most codes are not perfect.

Covering and Packing



Covering Radius



Packing Radius

Example

Consider the 5-repeat code in V(5,3). There are just 3 codewords, 00000, 11111, and 22222. A word such as 01221 is at distance 4 from 00000, and distance 3 from both 11111 and 22222. The distance of a word x from a code word is just 5 - (# symbols in common with the codeword). Since there are just 3 symbols and 5 positions, every word must have at least one repeated symbol, and so distance at most 3 from some codeword. Spheres of radius 3 around the codewords will therefore contain all of V(5,3). This means that $s \leq 3$. The example of 01221 shows that if $s = 2$ this word would not be contained in any sphere, thus the covering radius $s = 3$. This same example shows that spheres of radius 3 are not disjoint, so $t < 3$. Two spheres of radius 2 must be disjoint, since a word in both would have 3 symbols in common with both codewords $\rightarrow\leftarrow$ So, the packing radius $t = 2$.

Sphere Packing Bound

We can count the number of words in a sphere of radius e in V(n,q) and obtain:

$$|S_e(x)| = \sum_{i=0}^{e} \binom{n}{i} (q-1)^i.$$

To count the number of words at distance i from the word x, we first select which i positions will be different and then in each of these positions we select an element of the alphabet different from the one in that position in x (there are q-1 choices per position).

If C is an (n,M,d)-code in V(n,q) and the spheres of radius e centered at the codewords are disjoint, we obtain the sphere packing bound – since V(n,q) contains q^n words:

$$M \sum_{i=0}^{e} \binom{n}{i} (q-1)^i \leq q^n$$

Sphere Packing Bound

The minimum distance d of a perfect code must be odd. If it were even, there would be words at an equal distance from two code words and spheres around those codewords could not be disjoint if they had to contain these words. So, d = 2e + 1 and it is easy to see that for a perfect code t = s = e. Furthermore,

Proposition 3 : *An (n,M,d)-code in V(n,q) is perfect if and only if d = 2e + 1 and*

$$M \sum_{i=0}^{e} \binom{n}{i} (q-1)^i = q^n .$$

Pf: If C is perfect the spheres of radius e centered at codewords are disjoint (e = t) and all words are contained in them (e = s). On the other hand, if d = 2e + 1, we have e ≤ t and the counting equality shows that s ≤ e. Thus, s ≤ e ≤ t, which implies s = e = t. ❑

Equivalent Codes

The capabilities of a code are determined by the number of and the distances between the codewords. Thus, two codes should be considered equivalent if these statistics are the same.

There are two operations on the words of V(n,k) which, while changing the words, do preserve the distances between any set of words. The first is an arbitrary permutation of the coordinate positions applied to all words of V(n,k). This doesn't change distances since the distance is determined only by the number of positions in which the entries differ and not on where these occur. The second operation applies independently to any coordinate position. Since it is only whether or not the symbols in this position differ, and not what the symbols are that matter, any arbitrary permutation of the alphabet can be applied in this coordinate position.

Equivalent Codes

Formally, we say that two codes in V(n,k) are ***equivalent*** if one can be obtained from the other by a series of the two operations:

1) arbitrary permutations of the coordinate positions of all words,
2) arbitrary permutations of the symbols independently applied to the symbols in any set of coordinate positions.

The following codes (the 3 words in a column) are equivalent in V(4,4):

0123	0213	0000
1120	1210	1002
0333	0333	0120

Note that in all cases the distance between the first two codewords is 2, as is the distance between the first and third, and 4 between the second and third.

Equivalent Codes

Note that as a consequence of the notion of equivalence, given a code C in V(n,k), there will always exist an equivalent code to C which contains the zero word (in fact, an equivalent code to C which contains any word you may like).

Linear Codes

In the V[n,q] setting, an important class of codes are the ***linear codes***, these codes are the ones whose code words form a sub-vector space of V[n,q]. If the subspace of V[n,q] is k dimensional then we talk about the subspace as an **[n,k]-*code***. (Note that the square brackets indicate a linear code).

In the V[n,q] setting, the terms "word" and "vector" are interchangeable.

Linear codes, because of their algebraic properties, are the most studied codes from a mathematical point of view. Our text (and many others) is devoted almost exclusively to linear codes.

Linear Codes

There are several consequences of a code being linear.

1) The sum or difference of two codewords is another codeword.
2) The zero vector is always a codeword.
3) The number of codewords in an [n,k]-code C of V[n,q] is q^k.

There are k vectors in a basis of C. Every codeword is expressible as a unique linear combination of basis vectors. Thus, to count the number of codewords, we just have to count the number of linear combinations. There are q choices for a scalar multiple of each basis vector and therefore q^k linear combinations in total.

Since the number of codewords of a linear code is determined by the dimension of the subspace, the (n, M, d) notation for general codes is generally replaced by **[n, k, d]** for linear codes.

Linear Codes

In general, finding the minimum distance of a code requires comparing every pair of distinct elements. For a linear code however this is not necessary.

Proposition 4: *In a linear code the minimum distance is equal to the minimal weight among all non-zero code words.*

Proof: Let x and y be code words in the code C, then x - y is in C since C is linear. We then have d(x,y) = d(x-y,0) which is the weight of x-y. ❑

(Notice that this proposition is actually valid in a larger class of codes ... one only requires that the alphabet permits algebraic manipulation and that the code is "closed" under subtraction.)

Generator Matrix

We shall now look at two ways of describing a linear code C.

The first is given by a ***generator matrix*** G which has as its rows a set of basis vectors of the linear subspace C. If C is an [n,k]-code, then G will be a $k \times n$ matrix.

The code C is the set of all linear combinations of the rows of G, or as we usually call it, the row space of G.

Given the matrix G, the code C is obtained by multiplying G on the left by all possible $1 \times k$ row vectors (this gives all possible linear combinations):

$$C = \{xG \mid x \in V[k,q] \}.$$

Equivalence of Linear Codes

The general concept of equivalence of codes does not necessarily preserve the property of a code being linear. That is, linear codes may be equivalent to non-linear codes. In order to preserve the linear property we must limit the types of operations allowed when considering equivalence.

Two linear codes are ***equivalent*** if one can be obtained from the other by a series of operations of the following two types:

1) an arbitrary permutation of the coordinate positions, and
2) in any coordinate position, multiplication by any non-zero scalar.

(Note that this second operation preserves the 0 entries.)

Generator Matrix

Due to this definition of equivalence, elementary row and column operations on the generator matrix G of a linear code produce a matrix for an equivalent code.

Since G has rank k, by elementary row operations we can transform G to a matrix with a special form. Thus, we see that every linear code has an equivalent linear code with a generator matrix of the form $G = [I_k \; P]$, where I_k is the $k \times k$ identity matrix and P is a $k \times n\text{-}k$ matrix. We call this the ***standard form of G***.

Example

Let C be the [7,4]-code of V[7,2] generated by the rows of G (in standard form):

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

We get the 16 code words by multiplying G on the left by the 16 different binary row vectors of length 4.

So for instance we get code words:

(1,1,0,0) G = (1,1,0,0,1,0,1)
(1,0,1,1) G = (1,0,1,1,1,0,0)
(0,0,0,0) G = (0,0,0,0,0,0,0).

Example

The list of all the codewords is:

0 0 0 0 0 0 0 0	1 1 0 1 0 0 0	0 1 1 0 1 0 0	0 0 1 1 0 1 0
0 0 0 1 1 0 1	1 0 0 0 1 1 0	0 1 0 0 0 1 1	1 0 1 0 0 0 1
1 1 1 1 1 1 1	0 0 1 0 1 1 1	1 0 0 1 0 1 1	1 1 0 0 1 0 1
1 1 1 0 0 1 0	0 1 1 1 0 0 1	1 0 1 1 1 0 0	0 1 0 1 1 1 0

Notice that there are 7 codewords of weight 3, 7 of weight 4, 1 of weight 7 and 1 of weight 0. Since this is a linear code, the minimum distance of this code is 3 and so it is a 1-error correcting code.

This [7,4,3] code is called the **[7,4] – *Hamming Code***. It is one of a series of codes due to Hamming and Golay.

Parity Check Matrix

We now come to the second description of a linear code C.

The orthogonal complement of C, i.e. the set of all vectors which are orthogonal to every vector in C [orthogonal = standard dot product is 0], is a subspace and thus another linear code called the ***dual code*** of C, and denoted by $C^{\perp}$. If C is an [n,k]-code then $C^{\perp}$ is an [n, n-k] code.

A generator matrix for $C^{\perp}$ is called a ***parity check*** matrix for C. If C is an [n,k]-code then a parity check matrix for C will be an $n-k \times n$ matrix. If H is a parity check matrix for C, we can recover the vectors of C from H because they must be orthogonal to every row of H (basis vectors of $C^{\perp}$).

Parity Check Matrix

Thus the code C is given by a parity check matrix H as follows:

$$C = \{ x \in V[n,q] \mid Hx^T = \mathbf{0} \}$$

since the entries of this product are just the dot products of x with the rows of H.

Example

A parity check matrix for the [7,4]-Hamming code is given by:

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

Recall from the earlier example that 0001101 is a codeword and notice that

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

Parity Check Matrices

Theorem 1 : *Let H be a parity-check matrix for an* [n,k]-*code C in V[n,F]. Then every set of s-1 columns of H are linearly independent if and only if C has minimum distance at least s.*

Proof: First assume that every set of s-1 columns of H are linearly independent over F. Let $c = (c_1\ c_2\ \ldots\ c_n)$ be a non-zero codeword and let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n$ be the columns of H. Then since H is the parity check matrix, $Hc^T = \mathbf{0}$. This matrix-vector product may be written in the form

$$H c^T = \sum_{i=1}^{n} c_i h_i = 0 .$$

The weight of c, wt(c) is the number of non-zero components of c.

Parity Check Matrices

If wt(c) ≤ s - 1, then we have a nontrivial linear combination of less than s columns of H which sums to **0**. This is not possible by the hypothesis that every set of s - 1 or fewer columns of H are linearly independent. Therefore, wt(c) ≥ s, and since c is an arbitrary non-zero codeword of the linear code C it follows that the minimum non-zero weight of a codeword is ≥ s. So, since C is linear (Prop. 4), the minimum distance of C is ≥ s.

To prove the converse, assume that C has minimum distance at least s. Suppose that some set of t < s columns of H are linearly dependent. Without loss of generality, we may assume that these columns are $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_t$.

Parity Check Matrices

Then there exist scalars λ_i in F, not all zero, such that

$$\sum_{i=1}^{t} \lambda_i h_i = 0 .$$

Construct a vector c having λ_i in position i, $1 \le i \le t$, and 0's elsewhere. By construction, this c is a non-zero vector in C since $Hc^T = \mathbf{0}$. But wt(c) = t < s. This is a contradiction since by hypothesis, every non-zero codeword in C has weight at least s. We conclude that no s-1 columns of H are linearly dependent. ■

Parity Check Matrices

It follows from the theorem that a linear code C with parity-check matrix H has minimum distance (exactly) d if and only if *every* set of d-1 columns of H are linearly independent, and *some* set of d columns are linearly dependent. Hence this theorem could be used to determine the minimum distance of a linear code, given a parity-check matrix.

Parity Check Matrices

It is also possible to use this theorem to *construct* single-error correcting codes (i.e., those with a minimum distance of 3). To construct such a code, we need only construct a matrix H such that no 2 or fewer columns are linearly dependent. The only way a single column can be linearly dependent is if it is the zero column. Suppose two non-zero columns $\mathbf{h}_i$ and $\mathbf{h}_j$ are linearly dependent. Then there exist non-zero scalars $a, b \in F$ such that

$$a\,\mathbf{h}_i + b\mathbf{h}_j = 0.$$

This implies that

$$\mathbf{h}_i = -a^{-1}b\,\mathbf{h}_j,$$

meaning that $\mathbf{h}_i$ and $\mathbf{h}_j$ are scalar multiples of each other. Thus, if we construct H so that H contains no zero columns and no two columns of H are scalar multiples, then H will be the parity-check matrix for a linear code having distance at least 3.

Example

Over the field F = GF(3) = $\mathbb{Z}_3$ (integers mod 3), consider the matrix

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

The columns of H are non-zero and no column is a scalar multiple of any other column.

Hence, H is the parity-check matrix for a [5,2]-code in V[5,3] with minimum distance at least 3.

Converting Representations

When working with linear codes it is often desirable to be able to convert from the generator matrix to the parity-check matrix and vice-versa. This is easily done.

Theorem 2: *If* $G = [I_k \; A]$ *is the generator matrix (in standard form) for the* [n,k]-*code C, then* $H = [-A^T \; I_{n-k}]$ *is the parity check matrix for C.*

Proof: We must show that H is a generator matrix for $C^{\perp}$. Now $GH^T = I_k (-A) + A I_{n-k} = 0$, implying that the rows of H are orthogonal to the rows of G, thus *span*(H) = {row space of H} is contained in $C^{\perp}$.

Converting Representations

Consider any $\mathbf{x} \in C^{\perp}$ where $\mathbf{x} = (x_1\ x_2\ ...x_n)$ and let

$$\boldsymbol{y} = \boldsymbol{x} - \sum_{i=1}^{n-k} x_{i+k}\,\boldsymbol{r_i}$$

where $\mathbf{r_i}$ is the ith row of H. Since $\mathbf{x} \in C^{\perp}$ and we have just proven that $\mathbf{r_i} \in C^{\perp}$, $1 \le i \le k$, it follows that $\mathbf{y} \in C^{\perp}$. We now examine the structure of **y**. By construction, components k + 1 through n are 0, so $\mathbf{y} = (y_1\ y_2\ ...\ y_k\ 0\ 0\ ...\ 0)$. But since $\mathbf{y} \in C^{\perp}$, $G\mathbf{y}^T = 0$, which implies that $y_i = 0$, $1 \le i \le k$. Therefore, $\mathbf{y} = \mathbf{0}$ and

$$\boldsymbol{x} = \sum_{i=1}^{n-k} x_{i+k}\,\boldsymbol{r_i}.$$

Hence, $\mathbf{x} \in$ *span*(H) and we have $C^{\perp} \subseteq$ *span*(H). Thus, *span*(H) = $C^{\perp}$ and so, H is a generator matrix of $C^{\perp}$. ■

Example (Cont.)

To look at the code we have previously constructed, it would be convenient to have the generator matrix. Since H is the generator matrix for $C^{\perp}$, if we apply the last theorem we can get the parity-check matrix for $C^{\perp}$ which is the generator matrix for C. To this end we perform row operations on H to put it into standard form H'.

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow H' = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} = (I_3 A), \, so \, A = \begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 1 & 0 \end{pmatrix}.$$

$$G = (-A^T I_2) = \begin{pmatrix} 2 & 0 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Example (Cont.)

We can now take all linear combinations (over GF(3)) of the rows to write out the 9 codewords of C. With their weights they are

Codeword	Weight
00000	0
20210	3
11001	3
10120	3
22002	3
01211	4
21121	5
12212	5
02122	4

$$G = \begin{pmatrix} 2 & 0 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

And we see that we have indeed generated a code of minimum distance 3.

Hamming Codes

A ***Hamming Code*** of order r over GF(q) is an [n,k]-code where $n = (q^r-1)/(q-1)$ and $k = n - r$, with parity check matrix H_r an $r \times n$ matrix such that the columns of H_r are non-zero and no two columns are scalar multiples of each other.

Note that $q^r - 1$ is the number of non-zero r-vectors over GF(q) and that $q - 1$ is the number of non-zero scalars, thus n is the maximum number of non-zero r-vectors no two of which are scalar multiples of each other. It follows immediately from Theorem 1 that the Hamming codes all have minimum distance exactly 3 and so are 1-error correcting codes.

Hamming Codes are Perfect

Since the number of codewords in a Hamming code is q^k, a direct computation shows that sphere packing bound is met, so:

Theorem 3 : *The Hamming codes of order r over GF(q) are perfect codes.*

Proof: With $M = q^k$, and $d = 3 = 2e + 1$, ie. $e = 1$ we have:

$$M \sum_{i=0}^{e} \binom{n}{i} (q-1)^i = q^k (1 + n(q-1)) = q^k \left(1 + \frac{q^r - 1}{q-1}(q-1)\right) = q^{k+r} = q^n.$$

Example

The Hamming code of order r = 3 over GF(2) is given by the parity-check matrix

$$H_3 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

This is the [7,4]-code with distance 3. Re-ordering the columns of H_3 would define an equivalent Hamming code.

Example

The [13,10]-Hamming code of order 3 over GF(3) is given by the parity-check matrix

$$H_3 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 2 & 0 & 1 & 2 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 2 & 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 2 & 1 & 1 & 2 \end{pmatrix}.$$

Decoding

The usefulness of an error-correcting code would be greatly diminished if the decoding procedure was very time consuming. While the concept of decoding, i.e., finding the nearest codeword to the received vector, is simple enough, the algorithms for carrying this out can vary tremendously in terms of time and memory requirements.

Usually, the best (i.e., fastest) decoding algorithms are those designed for specific types of codes. We shall examine some algorithms which deal with the general class of linear codes and so, will not necessarily be the best for any particular code in this class.

Syndromes

When a codeword **c** is transmitted and vector **r** is received, the difference between the two is called the *error vector* **e**,
i.e. $\mathbf{r} = \mathbf{c} + \mathbf{e}$.

If H is a parity-check matrix for the linear code C, then

$$H\mathbf{r}^T = H(\mathbf{c} + \mathbf{e})^T = H\mathbf{c}^T + H\mathbf{e}^T = H\mathbf{e}^T$$

since $H\mathbf{c}^T = \mathbf{0}$ for any codeword.

$H\mathbf{r}^T$ is called the ***syndrome*** of **r**.

Syndromes

If wt(**e**) ≤ 1 then the syndrome of $\mathbf{r} = H\mathbf{e}^T$ is just a scalar multiple of a column of H. This observation leads to a simple decoding algorithm for 1-error correcting linear codes.

First, calculate the syndrome of **r**. If the syndrome is **0**, no error has occurred and **r** is a codeword.

Otherwise, find the column of H which is a scalar multiple of the syndrome.

If no such column exists then more than one error has occurred and the code can not correct it.

If however, the syndrome is α times column j, say, then add the vector with $-\alpha$ in position j and 0's elsewhere to **r**. This corrects the error.

Example

Let $H = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$ be the parity check matrix of a code C in V[5,3]. Now **x** = (1 0 1 2 0) is a code word since,

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

If **r** = (1 0 1 1 0) is received when **x** is transmitted, then the syndrome of **r** is:

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 2 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

Correction: **r** + (0 0 0 -2 0) = (1 0 1 -1 0) = (1 0 1 2 0).

Hamming Decoding

This method can be improved for binary Hamming codes. The very simple decoding algorithm that results is called ***Hamming Decoding***.

Rearranging the columns of the parity check matrix of a linear code gives the parity check matrix of an equivalent code. In the binary Hamming code of order r, the columns are all the non-zero binary vectors of length r. Each such column represents the binary form of an integer between 1 and $n = 2^r - 1$. We can arrange the columns of the parity check matrix so that the column in position i represents the integer i. Let H be the parity check matrix formed in this way.

Hamming Decoding

For example, the parity check matrix for the [7,4]-Hamming code would be written as:

$$H_3' = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Here, the column $(x,y,z)^T$ represents the number

$$x(2^0) + y(2^1) + z(2^2).$$

If v is the received vector, calculate the syndrome Hv^T. If at most one error has been made, the result is a vector of length r, either the zero vector or a column of H. When one error has been made the number represented by this column is the position in v which is in error – and since this is a binary code, we can correct it.

Hamming Decoding

0101010 is a codeword in the [7,4]-Hamming code. Suppose that we received the vector v = 0101110. We calculate:

$$H_3' v^T = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

The result represents the number 5, which is the position of the error.

Code Cosets

This procedure does not work for codes which are more than single error correcting. To deal with the more general situation, we introduce an equivalence relation on the vectors of V[n,q] with respect to the code C.

We say that vectors **x** and **y** are ***equivalent with respect to C*** if and only if, $\mathbf{x} - \mathbf{y} \in C$. This is an equivalence relation since,

1) $\mathbf{x} - \mathbf{x} = \mathbf{0} \in C$ for all **x**,

2) if $\mathbf{x} - \mathbf{y} \in C$, then $\mathbf{y} - \mathbf{x} = -(\mathbf{x}-\mathbf{y}) \in C$ since it is a scalar multiple of an element of C, and

3) if $\mathbf{x}-\mathbf{y} \in C$ and $\mathbf{y}-\mathbf{z} \in C$ then $\mathbf{x}-\mathbf{z} = (\mathbf{x}-\mathbf{y}) + (\mathbf{y}-\mathbf{z}) \in C$.

and the equivalence classes (which form a partition of V[n,q]) are called the ***cosets*** of C.

Code Cosets

An alternative way of viewing the cosets of C is to note that a coset consists of all the vectors that have the same syndrome. Indeed, if $\mathbf{x} - \mathbf{y} \in$ C, then $\mathbf{x} - \mathbf{y} = \mathbf{c}$ for some $\mathbf{c} \in$ C. Thus, $\mathbf{x} = \mathbf{c} + \mathbf{y}$ and

$$H\mathbf{x}^T = H(\mathbf{c} + \mathbf{y})^T = H\mathbf{c}^T + H\mathbf{y}^T = H\mathbf{y}^T.$$

Furthermore, if two vectors have the same syndrome then they are equivalent. If $H\mathbf{x}^T = H\mathbf{y}^T$,

$$\mathbf{0} = H\mathbf{x}^T - H\mathbf{y}^T = H(\mathbf{x} - \mathbf{y})^T,$$

and so, $\mathbf{x} - \mathbf{y}$ is $\in$ C.

Example

The table below lists the cosets of the [5,2]-code of V[5,2] whose generator matrix is given by,

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

The cosets are the rows in the table, with the first row being the code C (which is also a coset).

00000	10101	01110	11011
00001	10100	01111	11010
00010	10111	01100	11001
00100	10001	01010	11111
01000	11101	00110	10011
10000	00101	11110	01011
11000	01101	10110	00011
10010	00111	11100	01001

When the vectors of V[n,q] are arranged by cosets in this way it is called a ***standard array*** for the code

Example

The parity-check matrix for this code is

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

and the syndromes for each row are given below

00000	10101	01110	11011	**000**
00001	10100	01111	11010	**001**
00010	10111	01100	11001	**010**
00100	10001	01010	11111	**100**
01000	11101	00110	10011	**110**
10000	00101	11110	01011	**101**
11000	01101	10110	00011	**011**
10010	00111	11100	01001	**111**

Syndrome Decoding

In each coset we can select a vector of minimum weight (when there is a choice we can make the selection arbitrarily), this vector is called a ***coset leader***. In the previous example, the coset leaders were written in the first column of the standard array. We can use the coset leaders in a decoding algorithm.

When a vector **r** is received, we find the coset that it is contained in (via the syndrome calculation) and then subtract the coset leader for that coset to obtain a codeword. This procedure is called ***syndrome decoding***. That syndrome decoding gives the correct error correction is the content of the next theorem.

Syndrome Decoding

Theorem 4 : *Let C be an* [n,k]-*code in* V[n,q] *with codewords* c_j, $0 \le j \le q^k - 1$. *Let* l_i, $0 \le i \le q^{n-k} - 1$ *be a set of coset leaders for the cosets of this code. If* $r = l_i + c_h$ *then*

$$d(r,c_h) \le d(r,c_j) \text{ for all } j,\ 0 \le j \le q^k - 1.$$

Proof: Expressing these distances in terms of weights we have,

$$d(r,c_h) = d(l_i + c_h, c_h) = wt(l_i + c_h - c_h) = wt(l_i) \text{ and}$$

$$d(r,c_j) = d(l_i + c_h, c_j) = wt(l_i + c_h - c_j).$$

But $l_i + c_h - c_j$ is in the same coset as l_i, and the coset leader has minimal weight. ■

Syndrome Decoding

Note that if the coset leader is unique then c_h is the closest codeword (if it is not unique, then c_h is still as close as any other codeword). We have the following result about uniqueness.

Theorem 5 : *Let C be a linear code with minimum distance d. If* ***x*** *is any vector such that*

$$wt(x) \leqslant \left[\frac{d-1}{2}\right],$$

then ***x*** *is a* ***unique*** *element of minimum weight in its coset of C and hence is always a coset leader.*

Syndrome Decoding

Proof: Suppose that there exists another vector **y** with the same weight as **x** in **x**'s coset.

Since x and y are in the same coset, **x - y** $\in$ C, but

$$wt(x-y) \leqslant wt(x)+wt(y) \leqslant \left[\frac{d-1}{2}\right]+\left[\frac{d-1}{2}\right] \leqslant d-1.$$

This contradicts the minimum distance of the code, unless **x - y** is the zero vector, i.e. **x = y**. ■

Example

In the previous example, the [5,2]-code had minimum distance 3 (the minimum non-zero weight of a codeword) and so the above theorem states that the zero vector and all vectors of weight 1 will be unique coset leaders as can easily be verified from the standard array.

00000	10101	01110	11011
00001	10100	01111	11010
00010	10111	01100	11001
00100	10001	01010	11111
01000	11101	00110	10011
10000	00101	11110	01011
11000	01101	10110	00011
10010	00111	11100	01001