Contents

9 error detecting and correcting codes **1**
9.1 kinds of codes 1
9.2 finite fields 2
9.3 isbn - an error detecting code 5
9.3.1 isbn 10 5
9.3.2 isbn 13 7
9.4 an error correcting code 9
9.4.1 generator matrix 10
9.4.2 perfect codes 12
9.5 another set of examples 13
9.6 summary 16
9.7 exercises 16

This is page 1
Printer: Opaque this

9

error detecting and correcting codes

A main idea of this sequence of notes is that accounting is a legitimate field of academic study independent of its vocational implications and its obvious importance in the economic environment. For example, the study of accounting promotes careful and disciplined thinking important to any intellectual pursuit. Also, studying how accounting works illuminates other fields, both academic and applied. In this chapter we begin to connect the study of accounting with the study of codes. Coding theory, while theoretically rich, is also important in its applications, especially with the pervasiveness of computerized information transfer and storage. Information integrity is inherently of interest to accountants, so exploring coding can be justified for both academic and applied reasons.

9.1 kinds of codes

We will study three kinds of codes: error detecting, error correcting, and secret codes. The first two increase the reliability of message transmission. The third, secret codes, are designed to ensure that messages are available only to authorized users, and can't be read by the bad guys. The sequence of events for all three types of codes is as follows.

1. determine the message to transmit. For our purposes, we will treat messages as vectors.
2. The message is encoded. The result is termed a codeword, or cyphertext, also a vector.

3. The codeword is transmitted through a channel. The channel might inject noise into the codeword, or it might be vulnerable to eavesdropping.

4. The received vector is decoded. If the received vector is no longer the transmitted codeword, the decoding process may detect, or even correct, the error.

Setting aside secret codes until chapter 10, this chapter will concentrate on error detecting and correcting codes. Applications of these types of codes are numerous. In a business environment codes are used to design account numbers, inventory part numbers, and all sorts of identification numbers. Codes enable efficient transmission of television pictures as well as pictures sent from the farthest parts of the solar system. The cause of noise in a codeword may be as mundane as a typing mistake, or as cosmic as interference from sunspots.

We will study a popular class of error detecting and correcting codes called linear codes. Linear codes employ the same techniques we have used for understanding the linear transformations in accounting. Both encoding and decoding are accomplished by matrix multiplication. Furthermore, decoding is a direct application of the concept of a nullspace.

Consider decoding. Every linear code can be specified by a matrix called a parity check matrix, denoted H. Decoding is accomplished by multiplying the received vector by H. If the received vector is in the nullspace of H, then the received vector is a legal codeword. Notice the connection with accounting. H is "like" the accounting transformation matrix A. The legal codewords are "like" the set of looping transactions which leave the account balances unchanged. Similar to the accounting applications, the received vector, denoted y, is in the nullspace of H if H times y is a vector of zeros.

Definition 9.1 *The matrix product Hy is termed the syndrome.*

If the syndrome contains a non-zero element, an error has been detected. If we are clever in our analysis of the syndrome, we may be able to infer the position and amount of the error, thereby allowing error correction.

Before moving to examples of codes, we need to acquire another tool; the notion of a finite field.

9.2 finite fields

We will restrict ourselves to a finite set of messages which will, in turn, imply a finite number of errors. This allows for efficient error detection and correction. The mechanism to accomplish this is modular arithmetic.

The main idea is pretty simple, and the notation for it (devised by Gauss) is straightforward.

Definition 9.2 *Modular arithmetic reduces the set of integers under consideration to a finite number.*

$$a = b + cm \Longleftrightarrow a \equiv b(\text{mod } m) \text{ where } a,\ b,\ c, \text{ and } m \text{ are integers.}$$

The second equivalence is read "a is congruent to b modulo (or simply 'mod') m."

Any integer has a corresponding element in a finite set. Simply divide by m and report the remainder. There will only be m integers in the reduced set.

A convenient example is "binary." The arithmetic is done modulo 2; all integers are equivalent (congruent) to either 0 or 1. For example,

$$\begin{aligned} 2 &\equiv 4 \equiv 6 \equiv 0(\text{mod } 2) \\ 1 &\equiv 3 \equiv -1 \equiv 1(\text{mod } 2) \end{aligned}$$

For the reduced (finite) set of integers to qualify as a field, we must be able to add and mutiply in pretty much the normal way. We can here, except the answer is always reduced to either 0 or 1.

$$1 + 1 \equiv 0\,(\text{mod } 2)$$

Matrix multiplication also works in modular arithmetic.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} (\text{mod } 2)$$

In preparation for the first error detecting example consider arithmetic modulo 11, in which every number is divided by 11, and the remainder reported.

$$\begin{aligned} 7 + 6 &= 13 \equiv 2\,(\text{mod } 11) \\ 7 \times 6 &= 42 \equiv -2 \equiv 9\,(\text{mod } 11) \end{aligned}$$

Here's the entire multiplication table modulo 11.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

There are two important things to notice about the multiplication table.

1. There are no zeros. When two non-zero numbers are multiplied, the result is non-zero. This property does not hold for all moduli. For example,
$$6 \times 6 \equiv 0 \,(\mathrm{mod}\ \ 9)$$
2. There are no "repeats." Each row (and column) consists of all the possible 10 numbers.

The two properties have important applications in coding, and they follow directly from the concept of a prime number and what is known as the fundamental theorem of arithmetic.

Definition 9.3 *A prime number is divisible (evenly, that is, leaving no remainder) only by 1 and the number itself.*

Theorem 1 *Any integer can be factored into a product of prime numbers. Furthermore, the factorization is unique.*

Consider property 1 for some prime modulus, p. A zero entry in the multiplication table means

$$rs \equiv 0 \,(\mathrm{mod}\ \ \mathrm{p}) \Longleftrightarrow rs = pn$$

where r and s are integers less than p and n is some integer. The equality on the right implies two different factorizations of rs, violating unique factorization. Hence, the fundamental theorem implies property 1 is true for all p.

Property 2 follows from similar logic. "Repeats" imply the first two equations below which, in turn, imply the third.

$$\begin{aligned} rs &= pn + k \\ rt &= pm + k \\ r\,(s - t) &= p\,(n - m) \end{aligned}$$

And the third equation is not allowed by the fundamental theorem.

With these preliminaries we are ready for an example code.

9.3 isbn - an error detecting code

ISBN stands for "international standard book number." Virtually every book published in the world is assigned an ISBN. The numbers are assigned in a way so that some typographical errors which might occur in typing or transcribing an order can be detected. The probability that the wrong book is delivered is thereby reduced.

In 2007 the design of the ISBN was altered slightly. A visible manifestation is that the length of the number (codeword) increased from 10 to 13. We'll start with ISBN 10 as it supplies a nicer illustration of the two properties in the previous section, and catch up with ISBN 13 later.

9.3.1 isbn 10

The ISBN is a linear code in the sense that it is completely specified by its parity check matrix. For ISBN 10 the parity check matrix has one row and 10 columns.

$$H = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

All ISBN codewords reside in the nullspace of H where arithmetic is conducted modulo 11. That is, for an ISBN y vector,

$$Hy \equiv 0 \pmod{11}$$

If y does not satisfy the above equation, an error has been made.

Example 9.1 *The ISBN for The Norton History of Mathematics (published prior to 2007) is 0-393-04650-8.*

The ISBN is divided into 4 parts, possibly of various lengths across countries and companies. The first number is the official language of the country in which the book is published: zero is English. The second set of numbers is the number assigned to the publisher: W. W. Norton Publishing is 393. The next set is an internal inventory number chosen by the publisher. The last number is a check digit which ensures the ISBN resides in the nullspace of H.

Do the arithmetic.

$$
\begin{aligned}
Hy &= \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \\ 9 \\ 3 \\ 0 \\ 4 \\ 6 \\ 5 \\ 0 \\ 8 \end{bmatrix} \\
&= 231 = 11 \times 21 \equiv 0 \pmod{11}
\end{aligned}
$$

The ISBN code is designed to detect any single error and any transposition error, not necessarily of adjacent digits. Check to see that any such error for Norton results in a non-zero syndrome. More generally, the two error types are always caught because of the two noted properties of the multiplication table.

Consider a single error. Let the received ISBN vector be the sum of the correct ISBN and an error vector with a non-zero entry in position i.

$$
y = y_{ISBN} + e = y_{ISBN} + \begin{bmatrix} 0 \\ . \\ . \\ . \\ e_i \\ . \\ . \\ . \\ 0 \end{bmatrix}
$$

The syndrome is calculated.

$$
Hy = H\left(y_{ISBN} + e\right) = Hy_{ISBN} + He = 0 + He
$$

He is e_i times the ith element of H. As both numbers are non-zero, the syndrome is non-zero by property 1, and the error is detected.

Consider a transposition error, not necessarily of adjacent digits. Suppose in the received y that elements y_j and y_k are transposed. The received y vector appears as follows.

$$
\begin{aligned}
y &= y_{ISBN} + \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ y_k - y_j \\ \cdot \\ \cdot \\ \cdot \\ y_j - y_k \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \\
&= y_{ISBN} + \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ y_k - y_j \\ \cdot \\ \cdot \\ \cdot \\ -(y_k - y_j) \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}
\end{aligned}
$$

For the syndrome He to be zero, two distinct elements of H, H_j and H_k, must yield the same answer when multiplied by the same number:

$$y_k - y_j$$

But that is impossible by property 2, so the transposition error is detected.

One more note before moving on to ISBN 13. Sometimes to ensure the ISBN resides in the nullspace of H, the check digit, the 10th element of y_{ISBN}, must be the number 10. Rather than writing the two digit number, the ISBN assigns the Roman numeral X. From watching the super bowl, we know X stands for 10.

9.3.2 isbn 13

ISBN 13 is similar to the universal product code used for all kinds of inventory items. It has a different looking parity check matrix.

$$H = \begin{bmatrix} 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 \end{bmatrix}$$

Furthermore, the arithmetic is done modulo 10, the multiplication table for which follows.

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	0	2	4	6	8
3	3	6	9	2	5	8	1	4	7
4	4	8	2	6	0	4	8	2	6
5	5	0	5	0	5	0	5	0	5
6	6	2	8	4	0	6	2	8	4
7	7	4	1	8	5	2	9	6	3
8	8	6	4	2	0	8	6	4	2
9	9	8	7	6	5	4	3	2	1

It is noticed right away that the two properties of no zeros and no repeats do not hold, in general. They do hold, however, for the rows and columns associated with 1 and 3, among others. And, since those are the only numbers used in H, a single error remains detectable by the syndrome. Further, some transposition errors are detectable, as well. For example, transposition of adjacent digits will yield a non-zero syndrome, as the same number multiplied by a 1 and a 3 must give a different answer. However, transposition of digits removed by two places will not be caught; the syndrome will still be calculated as zero. Presumably, the designers of the code are less worried about this particular error type occuring often.

Example 9.2 *The ISBN for Managerial Uses of Accounting Information by Joel Demski (published post 2007) is 978-0-387-77450-3.*

Do the arithmetic.

$$\begin{aligned} Hy &= \begin{bmatrix} 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 9 \\ 7 \\ 8 \\ 0 \\ 3 \\ 8 \\ 7 \\ 7 \\ 7 \\ 4 \\ 5 \\ 0 \\ 3 \end{bmatrix} \\ &= 120 \equiv 0 \pmod{10} \end{aligned}$$

The universal product code (UPC) is a ubiquitous variation on ISBN 13. The UPC is particularly visible at supermarkets where the checkout scanners read the bar codes on the inventory items. Typically, a UPC code has 12 digits, and the parity check matrix is

$$H = \begin{bmatrix} 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 \end{bmatrix} \pmod{10}$$

9.4 an error correcting code

So far we have been able to discern when an error exists in the codeword. We have not, however, been able to fix the error, at least not with the information in the syndrome alone. It is possible, by expanding the parity check matrix, to not only detect, but also correct errors in the codeword.

Example 9.3 *Define the parity check matrix as follows with modulus 2.*

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

Now the syndrome, Hy, has 3 elements, and the extra information can be used for error correction. The logic is not complicated. An error means the element y_i is a one instead of a zero, or vice-versa. For no errors, the syndrome is all zeros, that is, in the nullspace of H. If y_i is one instead of zero, then by the rules of matrix multiplication, the syndrome will be the ith column of H. And, since $-1 \equiv 1 \pmod 2$, if y_i is a zero instead of a one, the syndrome will likewise be the ith column of H. Check with an example.

Example 9.4 *Suppose the received codeword is* $y = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T$. *Syndrome decoding yields*

$$Hy = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \pmod{2}$$

As the syndrome is the 4th column of H, it indicates y_4 should be corrected to a 1 from 0. The corrected y

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

And calculation of the syndrome verifies the new y is now a legal codeword.

$$Hy = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \pmod 2$$

It would be convenient to generate legal codewords directly, rather than pick a random vector and correct it using syndrome decoding as above. There must be a simpler way, and, indeed, there is: use of the generator matrix.

9.4.1 generator matrix

In the example under consideration, the original vector (inventory number, employee id, etc.) is 4 elements long. Call it x. The matrix which multiplies the original message, x, in order to generate the codeword is called the generator matrix, G.

$$Gx = y$$

In this case, the generator matrix adds three redundant elements to x, enabling syndrome decoding to correct any single error.

It turns out to be fairly easy to construct G given H. Write H in the following block matrix format.

$$\begin{aligned} H &= \begin{bmatrix} B & I_3 \end{bmatrix}, \text{ where} \\ B &= \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \text{ and} \\ I_3 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

For G to be a legitimate generator matrix, the matrix product Gx must yield the zero vector when multiplied by H.

$$HGx = Hy = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ for all possible } x\text{'s}$$

This implies G is 7×4 such that

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

Here's the one that does the trick.

$$\begin{aligned} G &= \begin{bmatrix} I_4 \\ -B \end{bmatrix} \\ HG &= \begin{bmatrix} B & I_3 \end{bmatrix} \begin{bmatrix} I_4 \\ -B \end{bmatrix} = BI_4 - I_3 B \\ &= B - B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

Substituting for B, and recalling that -1 $\equiv 1 \pmod 2$, the generator matrix is as follows.

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

Example 9.5 *Let the original message be* $x = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T$.

Calculate the codeword by multiplying by the generator matrix, G.

$$Gx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \pmod 2 = y$$

Notice that G keeps the original message, x, intact and adds three redundant elements. Decoding verifies that y is a legal codeword (no noise has been injected).

$$Hy = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \pmod 2$$

9.4.2 perfect codes

The example code has another property which is convenient, on occasion.[1]

Definition 9.4 *A perfect single error correcting code is one in which the number of possible received vectors is equal to the number of legal codewords plus the number of vectors which can be changed into a legal codeword with exactly one correction.*

For a perfect code, in other words, there are no wasted vectors. Every vector is either a legal codeword, or just one element removed from a legal codeword.

To demonstrate the example code satisfies the definition for perfect requires two steps. First, it is verified there is no vector which can simultaneously be corrected to two legal codewords. That is, no vector is one change from two different legal codewords. But that can't happen. Inspection of the parity check matrix verifies there is no ambiguity about which element should be corrected: the columns of H are distinct. Whichever column is equal to the syndrome specifies the element of the received vector to correct.

Now it is a matter of counting the number of vectors in each of the categories.

- the total number of possible received vectors: $2^7 = 128$.
- the total number of legal codewords: $2^4 = 16$.
- the total number of ways a received vector can be one off from a legal codeword: $16\,(7) = 112$.

The last calculation is the number of legal codewords times the number of positions available for an error to occur. Perfectness is verified by the sum: $112 + 16 = 128$.

[1] It is particularly easy, for example, to construct examination questions.

9.5 another set of examples

The example code of the previous section, while illustrating some nice properties, is not a very large code. That is, if the codewords is meant to characterize diffeerent inventory items, for example, the size of the inventory is limited to 16 units. It is relatively straightforward, however, to increase the size of the code by increasing the modulus. As usual, the way to specify the code is to write down the parity check matrix. For the next example, arithmetic is done modulo 5.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 \end{bmatrix} \pmod 5$$

With modulo 5 there are 5 possible values for each element. The syndrome, then, must supply information, not only about the position of an error, but the amount of the error, as well. When there is a zero in the syndrome, the error is easily positioned as in element 5 or 6.

Example 9.6 *Let the received vector be* $y = \begin{bmatrix} 2 & 1 & 2 & 1 & 3 & 1 \end{bmatrix}^T$. *Perform syndrome decoding.*

$$\begin{aligned} Hy &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 15 \end{bmatrix} \\ &\equiv \begin{bmatrix} 4 \\ 0 \end{bmatrix} \pmod 5 \end{aligned}$$

The syndrome can be fixed by subtracting 4 times the 5th column of H. And that can be accomplished by subtracting 4 from the 5th element of y. Since $-4 \equiv 1 \pmod 5$, the correction is to add 1 to the 5th element.

$$\begin{aligned} \text{corrected } y_c &= \begin{bmatrix} 2 & 1 & 2 & 1 & 4 & 1 \end{bmatrix}^T \\ Hy_c &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 15 \end{bmatrix} \\ &\equiv \begin{bmatrix} 0 \\ 0 \end{bmatrix} \pmod 5 \end{aligned}$$

And the codeword is corrected.

When there is no zero in the syndrome, the error resides in one of the first 4 elements.

Example 9.7 *Let the received vector be* $y = \begin{bmatrix} 3 & 2 & 4 & 1 & 2 & 3 \end{bmatrix}^T$. *Perform syndrome decoding.*

$$\begin{aligned} Hy &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 4 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 12 \\ 26 \end{bmatrix} \\ &\equiv \begin{bmatrix} 2 \\ 1 \end{bmatrix} \pmod 5 \end{aligned}$$

As the first row of H is all 1's, the first element of the syndrome is the amount of the error. The syndrome can be fixed by finding the column of H, when multiplied by 2, yields the syndrome. Because of the "no zeros - no repeats" properties of a prime modulus, only one column of H will satisfy the condition. There is, then, no ambiguity about the position and amount of the correction.

Searching the columns reveals the syndrome is 2 times the 3rd column of H.

$$2 \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \end{bmatrix} \equiv \begin{bmatrix} 2 \\ 1 \end{bmatrix} \pmod 5$$

The correction is to subtract 2 from the 3rd element of y.

$$\begin{aligned} \text{corrected } y_c &= \begin{bmatrix} 3 & 2 & 2 & 1 & 2 & 3 \end{bmatrix}^T \\ Hy_c &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \end{bmatrix} \\ &\equiv \begin{bmatrix} 0 \\ 0 \end{bmatrix} \pmod 5 \end{aligned}$$

And the correction results in the appropriate syndrome.

A generator matrix can be constructed using the methods of the previous section.

$$G = \begin{bmatrix} I_4 \\ -B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 4 & 4 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

Example 9.8 *Consistent with the prior example, let the original message* $x = \begin{bmatrix} 3 & 2 & 2 & 1 \end{bmatrix}$. *Calculate the redundant digits.*

$$\begin{aligned} Gx &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 4 & 4 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \\ 32 \\ 23 \end{bmatrix} \\ &\equiv \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \\ 2 \\ 3 \end{bmatrix} \pmod 5 \end{aligned}$$

And the appropriate redundant digits are added, as consistent with the syndrome analysis of the problem.

The final thing to do with this example is to verify the perfectness of the code. We already resolved there is no ambiguity in the correction, so just count the vectors in the categories.

- the total number of possible received vectors: $5^6 = 15,625$.
- the total number of legal codewords: $5^4 = 625$.
- the total number of ways a received vector can be one off from a legal codeword: $625\,(6)\,(4) = 15,000$.

The last calculation is the number of legal codewords times the number of positions available for an error to occur times the number of possible error amounts. Perfectness is verified by the sum: $15,000 + 625 = 15,625$.

One more example demonstrates the code can become as large as desired.

Example 9.9 *Perform arithmetic modulo 11, and let the parity check matrix be*

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 0 & 1 \end{bmatrix} \pmod{11}$$

The code has $11^{10} = 25,937,424,601$ legal codewords. This coding system could handle, for example, 12 digit phone numbers. If an individual dialed the number with one mistake, the system can correct the error, and the call can still go through to the intended party. There are no wasted phone numbers, as the code is a perfect one. The number of "one off" phone numbers is $11^{10}(12)(10) = 11^{10}(120)$, the number of legal phone numbers times the number of positions available for an error times the number of possible error amounts. The total of 12 digit phone numbers is $11^{12} = 11^{10}(1+120) = 11^{10}\left(11^2\right)$.

9.6 summary

This chapter is the first to deal with the issue of how to preserve data integrity. The error detecting and correcting codes presented herein rely on two primary academic tools. One is orthogonality and the concept of the nullspace, a tool used extensively in prior chapters. The other tool is number theory which will prove quite useful in subsequent chapters. In this chapter we got our first exposure to prime numbers and the fundamental theorem of arithmetic.

9.7 exercises

Exercise 9.1 *Here are some ISBN's - possibly in error. Check for accuracy.*

1. *Probability Theory* by E. T. Jaynes. 978-0-521-59271-0
2. *Essays in Accounting Theory in Honour of Joel S. Demski.* 0-387-30397-9 and 978-0387-30397-0
3. *Number Theory and Its History* by Oystein Ore. 0-486-65620-9

Exercise 9.2 *Fill in the missing digits in the following UPC numbers.*

$$\begin{array}{cccccccccccc} 0 & 7 & 0 & 5 & 5 & 4 & 0 & 0 & ? & 2 & 3 & 6 \\ 0 & 2 & 8 & 4 & 0 & 0 & 0 & 9 & ? & 9 & 0 & 1 \end{array}$$

Exercise 9.3 *Consider a "perfect" single error correcting code with parity check matrix*

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 & 0 & 1 \end{bmatrix} \pmod{7}$$

For the following received codewords, detect and correct a single error, if necessary.

$$\begin{array}{cccccccc} 1 & 5 & 3 & 6 & 1 & 3 & 1 & 6 \\ 3 & 2 & 2 & 0 & 0 & 3 & 1 & 2 \end{array}$$

Exercise 9.4 *For the same modulo 7 code as in the previous problem, append the appropriate redundant digits.*

$$\begin{array}{cccccc} 1 & 2 & 4 & 3 & 6 & 6 \\ 5 & 6 & 1 & 1 & 2 & 1 \end{array}$$

Exercise 9.5 *Using the same modulo 7 code in the previous exercises, verify the "perfectness" of the same code by computing the number of possible received codewords, the number of legal codewords, and the number of codewords within one change of a legal codeword.*

Exercise 9.6 *Consider Matthew 5:37. "Let your communication be Yea, yea; Nay, nay: for whatsoever is more than these cometh from evil." The communication is an implied binary code. What is the implied parity check matrix, and the generator matrix? Is the code error correcting? What about error detection?*