Error Correcting Codes: An Introduction Through Problems

Jim Carlson

November 19, 1999

Contents

1 Introduction

Error correcting codes make possible much of digital technology, from modems to cell phones to compact disk players. The problem is to send a message across a noisy channel with perfect fidelity, or at least with as much fidelity as possible. Below we will describe the theory of these codes. It is based on some ideas from "modern algebra," including finite number systems called rings and fields. We'll learn about these, but let's first think about digital messages. Whether they represent a newspaper article, a musical performance, a conversation, or a painting, they consists of a sequence of 0's and 1's. A message might look like this:

$$011010110010 \ldots \tag{1}$$

However, noise can change bits of information: some 0's might get changed to 1's and vice versa as the message moves from sender to receiver. Thus the above message might be received as

$$011110110000 \ldots \tag{2}$$

In this case two bits, the ones in positions 4 and 11 were changed. One way to improve the fidelity of transmission is to repeat all bits three times, then decode by "majority rules." To see how this works, we first encode the message (2) as

$$000\ 111\ 111\ 111\ 000\ 111\ 000\ 111\ 111\ 000\ 000\ 111\ 000 \ldots \tag{3}$$

Now suppose that, just as before, bit errors occur in positions 4 and 11, so that the receiver sees the sequence

$$000\ 011\ 111\ 101\ 000\ 111\ 000\ 111\ 111\ 000\ 000\ 111\ 000\ \ldots \tag{4}$$

Then the second and fourth blocks are not among the legitimate codewords 000 and 111, and so we know that an error has occured. Moreover, according to the logic of "majority rules," the second block should be decoded as a 1, as should be the fourth block. However, suppose that 111 was sent and that two bits were changed, so that 010 was received. Again, we know that an error has occured. Unfortunately, majority logic gives the wrong answer in this case.

What we have described is the simplest example of an error-correcting code. It has just two codewords, 000 and 111, out of the eight possible three-bit message sequences 000, 001, 010, etc. Even though it does not completely eliminate the possibility of error, it reduces the chance of it. Suppose, for instance, that the probability of a one-bit change is 0.01, and that errors are "statistically independent." Then the probability of a two-bit change in one three-bit block is $3 \times 0.99 \times 0.01^2 = 0.000297$. Thus the error rate is reduced by a factor of about thirty.

The error reduction is good, but there is a cost to be paid for it: information is transmitted three times more slowly than if the error correcting code is not used. Now a remarkable theorem of Claude Shannon [4] discovered in 1948 shows that it is possible to have the best of both worlds: good error correction and a fast transmission rate. More precisely, there is a code which transmits information at a rate as close to the capacity of the trasmission channel as one likes, but which makes the error rate as low as one likes. The difficulty is that Shannon's theorem does not tell us how to design such marvelous codes. The first progress toward the goal he set was made two years later by Richard Hamming [2], like Shannon a mathematician working at Bell Labs.

The Hamming codes and their more sophisticated descendants — the ones used now in compact disk recordings, cell phones, satellite communications, etc. — are based algebraic ideas that have their origin in the number-theoretic research of Gauss, around 1801. In the next section we will explain some of this mathematics, and then we explain how the Hamming code works. For more information, see the references. Two good places to start are the book by Lindsay Childs [1] and the article about the work of Richard Hamming [3].

2 Mathematics for codes

Error correcting codes are based on number systems with only finitely many elements. To describe them, we begin with familiar, infinite number systems and introduce some basic vocubulary. Consider first the integers, that is, the positive and negative whole numbers together with zero, which we write like this:

$$\mathbf{Z} = \{\ \ldots -2,\ -1,\ 0,\ 1,\ 2,\ \ldots\ \} \tag{5}$$

It is an example of a *commutative ring*: a set of things with an operation of addition and an operation of multiplication that obey the usual rules. Among these rules are the two commutative laws: $a+b = b+a$ and $ab = ba$ for all a and b. (There are also structures called noncommutative rings, for which $ab \neq ba$).

Consider next the rational numbers, that is, the set of positive and negative fractions, together with zero:

$$\mathbf{Q} = \{ \ a/b \mid a, b \in \mathbf{Z}, \ b \neq 0 \ \} \tag{6}$$

It is a ring, but also an example of a *field*. In a field the equation $ax = 1$ has a solution if $a \neq 0$. This solution (which is unique) is called the *multiplicative inverse* of a, and it is what makes division possible: $b/a = bx$. Other familiar fields are the real numbers, written $\mathbf{R}$, and represented by possibly infinite decimal expansions, and the complex numbers $\mathbf{C}$, obtained by adjoining $\sqrt{-1}$ to $\mathbf{R}$.

To build error-correcting codes we will use a kind of arithmetic different from the one we find in familiar rings and fields such as $\mathbf{Z}$ and $\mathbf{Q}$. All these new rings will have finitely many elements, which makes them easy to store or represent in a computer. The smallest field has just two elements, 0 and 1, standing for "off" and "on," respectively. It is called $\mathbf{F}_2$ — $\mathbf{F}$ for field and 2 for the number of elements it has. The addition and multiplication tables for $\mathbf{F}_2$ are as follows.

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

(7)

It turns out there are other finite fields, for example, ones with three, four, or five elements. We call them $\mathbf{F}_3$, $\mathbf{F}_4$, $\mathbf{F}_5$, etc. However, there is no field with six elements. (Sorry, it's a theorem!)

Problem 1 *Show that the law $a + a$ holds for all a in $\mathbf{F}_2$.*

Let's now choose a field of numbers to work with — it could be $\mathbf{Q}$ or it could be $\mathbf{F}_2$, for example. In order to be as nonspecific as possible, we'll call it K. Then we can think about vectors with components in K. These are objects of the form $(1, 0)$ or $(0, 1, 0)$, or $(0, 1, 0, 1)$, etc. More generally, a vector is an n-tuple $(a_1, \ a_2, \ \ldots, \ a_n)$, and we call the set of all such vectors K^n.

The good thing about vectors is that they can be added. We do this by adding their components:

$$(a_1, \ a_2, \ \ldots, \ a_n) + (b_1, \ b_2, \ \ldots, \ b_n) = (a_1 + b_1, \ a_2 + b_2, \ \ldots, \ a_n + b_n) \tag{8}$$

Let's do an example, but with $K = \mathbf{F}_2$:

$$(0, 1, 1, 0) + (1, 0, 1, 0) = (0 + 1, 1 + 0, 1 + 1, 0 + 0) = (1, 1, 0, 0) \tag{9}$$

There is also a way of multiplying vectors: we multiply corresponding components, then add up the results.

$$(a_1, \ a_2, \ \ldots, \ a_n) \cdot (b_1, \ b_2, \ \ldots, \ b_n) = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n \tag{10}$$

This is called the *dot product*. For example,

$$(1,1,1,0)\cdot(1,0,1,0) = 1+0+1+0 = 0 \tag{11}$$

Note that we've worked with $K = \mathbf{F}_2$.

If we put elements of a field into a rectangular array, we get something called a matrix. Below is a matrix with four rows and four columns, and we have shown how to multiply it by a four by one matrix, which we think of as a vector.

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{12}$$

The multiplication rule is simple: to get i-th element of the product, take the dot product of the i-th row of the matrix with the given vector. Thus the first element of the product is

$$(1,1,0,0)\cdot(1,1,1,1) = 1+1+0+0 = 0 \tag{13}$$

Problem 2 *Check that the preceding matrix calculation is correct! What would the result be if we were working over the field of rational numbers or the ring of integers?*

Problem 3 *How many vectors are there in $\mathbf{F}_2^4$? Now consider the set of vectors (x, y, z, w) in $\mathbf{F}_2^4$ satisfying the equation $x+y+z+w = 0$. We'll call the solution set C. How many elements does C have? What happens when two vectors in C are added together?*

The theory of vectors, matrices, and linear equations over $\mathbf{F}_2$ will be enough to understand the Hamming code. For the modern codes used in today's digital devices one needs more mathematics, including the theory of polyomial equations over $\mathbf{F}_2$. The really good codes that correct multiple errors and bursts of errors use the theory of polynomial equations over a finite field. As a warm-up exercise to see what kind of math is involved, let's see what happens when we multiply two polynomials over $\mathbf{F}_2$:

$$(x+1)(x+1) = (1\times 1)x^2 + (1+1)x + 1 = x^2+1 \tag{14}$$

That was easy enough! Note that we have stumbled across a little theorem: over $\mathbf{F}_2$, unlike over $\mathbf{R}$, the polynomial x^2+1 factors. Said differently, the equation $x^2+1 = 0$ has root. Which root? It is $x = 1$! We can check this, of course:

$$(1)^2 + 1 = 1 + 1 = 0 \tag{15}$$

Problem 4 *Does the polynomial x^2+x+1 factor over $\mathbf{F}_2$. Does it have any roots in $\mathbf{F}_2$? Consider the same problem for x^3+1 and x^4+1.*

The set of all polynomials with coefficients in K is called "$K[x]$" — K for the field of coefficients and x for the variable. The structure $K[x]$ is a ring: we can add, subtract, and multiply polynomials in $K[x]$ and get polynomial in $K[x]$ as a result, but we can't generally divide polynomials and get a polynomial back. However, $K[x]$ is a really good ring, like the integers: every polynomial can be uniquely factored into irreducibles, just like every integer can be uniquely factored into primes. Irreducibles are polynomials that can't be factored into polynomials of lower degree.

Problem 5 *Show that $x^2 + x + 1$ is irreducible in $\mathbf{F}_2[x]$. Factor $x^4 + 1$ into irreducibles in $\mathbf{F}_2[x]$. Finally, make up a problem of your own, and solve it.*

3 Codes over $\mathbf{F}_2$

We are now going to use what we have learned about vectors and matrices over $\mathbf{F}_2$ to design better codes: codes which correct errors well but are still efficient, To get a good measure of efficiency, let's look at our old friend, the triple repeat code. We can think of it as defined by the function

$$c(x) = (x, x, x) \tag{16}$$

The function c takes in a bit of information x and then spews out a code word $c(x)$. Thus $c(0) = (0, 0, 0)$ and $c(1) = (1, 1, 1)$. In this case the *rate* of the code is $1/3$, since one bit of information determines all three bits in the code word. Our general measure is

$$\text{rate} = \frac{\text{length of } x}{\text{length of } c(x)} \tag{17}$$

Let's see if we can make a more efficient code (one with a better rate) using a different function. Here is one try:

$$c(x, y) = (x, y, x + y) \tag{18}$$

It is easy to recognize a code word, and hence to recognize when an error has occured. The sum of the components of $c(x, y)$ is $x + y + (x + y)$. Using the law $a + a = 0$ which holds in $\mathbf{F}_2$, we see that the sum is zero. Thus if we receive $(1, 1, 1)$, we know that an error has occured.

The code we have just defined is a *parity* code: computing in the integers, the sum of the components of a codeword is even. It is not a bad code — it's rate is $2/3$, and so it transmits information more efficiently than does the triple repeat code. However, while it can detect errors, it cannot correct them. To see why, we introduce the *Hamming distance* between two vectors:

$$H(v, w) = \text{number of positions in which } v \text{ and } w \text{ differ} \tag{19}$$

Thus the Hamming distance between $(1, 1, 1)$ and $(1, 1, 0)$ is 1, while the distance between $(1, 1, 1)$ and $(1, 0, 0)$ is 2. We now ask the question: what is the codeword closest to $(1, 1, 1)$? The answer is that there are three such codewords, all at distance 1: $(0,1,1)$, $(1,0,1)$, and $(1,1,0)$. All possibilities are equally likely, so there is no way of correcting the error in $(1,1,1)$

Problem 6 *Consider the triple repeat code. What codewords are closest to* $(1,1,0)$*? Describe the set of vectors at distance 1 or less from the codeword* $(0,0,0)$*. Do the same for the set of vectors at distance 1 or less from the codeword* $(1,1,1)$*. What is the relation between these two sets? (We think of them as balls of radius one around the respective codewords.) What is the distance between the codewords* $(0,0,0)$ *and* $(1,1,1)$*?*

The last problem gives us several important hints. The first is that the way to decode a corrupted message vector is to find the codeword that is closest to it. This "minimum distance decoding" what all codes used in practice do. The second, which may not be quite so clear, is that when the minimum distance between codewords is large, more errors can be corrected. In fact, if the minimum distance is $2r+1$, then r errors can be corrected. (Ths is a theorem.) Thus, in the case of the triple repeat code, the minimum distance is three, and one error can be corrected.

Problem 7 *What is the minimum distance for the parity code* $c(x,y)=(x,y,x+y)$*?*

We now see why it is going to be hard to design codes do both things well: transmit information efficiently, and correct many errors. For the first, we want the set of codewords to be large compared to the set of all possible bit sequences $\mathbf{F}_2^n$. For the first we want the distance between codewords to be large.

Let's note that our two codes — parity and triple repeat — can be described by equations as well as by functions. The set of codewords C for the parity code is the same as the set of solutions of the equation $x+y+z=0$. he set of codewords C for the triple-repeat code is the same as the set of solutions of the system of equations $x+y=0$, $y+z=0$.

Problem 8 *Check the statements made in the last paragraph.*

We now come to the Hamming code, which can correct a single error, but which has a better rate than the triple repeat code. matrix

$$\mathbf{H}=\begin{pmatrix}1&0&1&0&1&0&1\\0&1&1&0&0&1&1\\0&0&0&1&1&1&1\end{pmatrix} \tag{20}$$

and which appears on the Hamming IEEE medal, one of the highest awards bestowed by the Institute of Electrical and Electronic Engineers. The set of codewords C is the set of solutions of the equation

$$\mathbf{H}v=0 \tag{21}$$

where

$$v = \begin{pmatrix} x \\ y \\ p \\ z \\ q \\ r \\ s \end{pmatrix} \tag{22}$$

If we write out (21) we obtain the system of linear equations

$$\begin{aligned} x + p + q + s &= 0 \\ y + p + r + s &= 0 \\ z + q + r + s &= 0 \end{aligned} \tag{23}$$

This tells us how to form a codeword from bits p, q, r, s:

$$c(p, q, r, s) = (p + q + s,\ p + r + s,\ p, q + r + s,\ q,\ r,\ s) \tag{24}$$

It also tells us that the rate of the code is 4/7 — four information bits in a 7-bit codeword. Thus the Hamming code has a better rate than the triple repeat code.

What about error detection and correction? Detection is easy. Suppose, for example, that we want transmit the message 1010. Then $c(1, 0, 1, 0) = (1, 0, 1, 1, 0, 1, 0)$. Now suppose that the first bit is changed, so that $v = (0, 0, 1, 1, 0, 1, 0)$ is received. We compute

$$\mathbf{H}v = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{25}$$

Since $\mathbf{H}v \neq 0$, the v is not a codeword, and so an error must have occured. But our result tells us even more (it turns out). The vector $\mathbf{H}v$, tipped over on its side reads 001. This is the number one written in binary. Not coincidentally, this is the position of the error: the first of the codeword was changed.

Let's check this with another artificially introduced error. Let's change bit five of the codeword to get $v = (1, 0, 1, 1, 1, 1, 0)$. We find that

$$\mathbf{H}v = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \tag{26}$$

Tipping this vector on its side we get 101, which is 5 in binary. Again, that gives the position of the error!

To prove that this is always the case, start with a codeword c and add to it the vector e_i that has a 1 in position i and zero elsewhere. This corresponds to changing the bit in position i. Call the result v, so that $v = c + e_i$. This is the received message. Now we do a bit of matrix algebra. First, use the distributive law:

$$\mathbf{H}v = \mathbf{H}c + \mathbf{H}e_i \tag{27}$$

Then use the fact that $\mathbf{H}c = 0$ to get

$$\mathbf{H}v = \mathbf{H}e_i \tag{28}$$

Now comes a little fact about matrices. The result of mutiplying a matrix by a vector with a one in position i and zero elsewhere is the i-th column of the matrix. But the i-th column of $\mathbf{H}$, tipped on its side, is i written in binary form. This proves what we set out to prove.

Problem 9 *The message 1111101 was sent using the Hamming code. What message was sent?*

Problem 10 *What is the minimum distance between vectors in the Hamming code? Prove your answer to be correct.*

Problem 11 *Prove that if a code has minimum distance $2r+1$, then r errors can be corrected.*

4 Final remarks

There is a whole family of Hamming codes, and they can all correct single errors, but no more. Codes developed later, like the Reed-Solomon codes, can correct multiple errors and bursts of errors. They use somewhat more sophisticated mathematics: finite fields $\mathbf{F}_q$ with $q > 2$ play an important role, as do the associated polynomial rings $\mathbf{F}_q[x]$. NASA uses a Reed-Solomon (RS) code with $q = 2^8$ for deep-space communication, and other RS codes are used for compact disk recording.

For more information, see the references below, especially [1], or the references at

http://www.math.utah.edu/ugrad/colloquia/1999/1/

Also, this document and related information can be found at

http://www.math.utah.edu/hschool/carlson

We end with some rather open-ended problems.

Problem 12 *Try to guess how the next Hamming code is defined.*

Problem 13 *The field $\mathbf{F}_4$ can be constructed by adjoining a new element to $\mathbf{F}_2$, just like the complex numbers can be constructed from the reals by adjoining a new element ($\sqrt{-1}$). Try find construct $\mathbf{F}_4$ by adjoining an element α such that $\beta^2 + \beta + 1 = 0$. Use this equation, the law $a + a = 0$, and the usual laws of algebra to construct addition and multiplication tables for your version of $\mathbf{F}_4$.*

Problem 14 *The triple repeat code and the Hamming code are both examples of* perfect codes. *This means that the entire message space $\mathbf{F}_2^n$ is covered by nonoverlapping balls of fixed radius around codewords. Can you find other perfect codes?*

References

[1] Lindsay Childs, A Concrete Introduction to Higher Algebra, Second Edition, Springer-Verlag 1997

[2] R.W. Hamming, *Error-detecting and error-correcting codes*, Bell System Technical Journal **29**, pp 147–160 (1950)

[3] Samuel P. Morgan, *Richard Wesley Hamming, 1915-1998*, Notices of the American Mathematical Society **45** no. 9, pp 972–977 (1998)

[4] C.E. Shannon, *A mathematical theory of communication*, Bell System Technical Journal **27**, pp 379–423, 623–656 (1948)

[5] Garding, L. and T. Tambour, Algebra for Computer Science

Jim Carlson
Department of Mathematics
University of Utah
Salt Lake City, Utah 84112

http://www.math.utah.edu/~carlson
carlson@math.utah.edu