1 Framework

We are given a set $\mathcal{M}$ of messages indexed by binary sequences of length $k$. These sequences are regarded as members of the vector space $\mathbb{F}_2^k$ of vectors of length $k$ over the binary field $\mathbb{F}_2$ (consisting of the digits 0, 1, with arithmetic done mod 2). We shall often abbreviate a vector like $(x_1, \ldots, x_k)$ as the sequence $x_1 x_2 \ldots x_k$. We wish to communicate the messages through a noisy channel in such a way that the recipient can determine the message sent fairly certainly. The digits 0 and 1 will specify the signals used (generally electronic). The channel is assumed to have the following properties, and it is known as a binary symmetric channel or BSC:

- The channel is memoryless: signals previously sent through it have no effect on later signals—there is no “echo”.

- The channel is symmetric: the probabilities for a digit received when a given digit is sent are given in terms of one probability $p < 1/2$ by the following diagram.

![Diagram](attachment:binary_symmetric_channel.png)

The left side shows the digit sent, the right the one received, and the probabilities associated with each transition label the lines. A signal does not get erased, but it is always received and interpreted as a 0 or 1. (The diagram is from the Wikipedia article on the BSC.)

- And, of course, the channel is binary! The set $\{0, 1\}$ is the alphabet for the codes. We’ll see another alphabet later.
Messages are encoded by adjoining extra digits chosen so that if few enough errors occur in transmission, the intended message can be inferred from the received sequence of digits. These added digits are called check digits, those of the original message being the message digits. The resulting expanded sequences are called codewords, and they make up the code. We assume that all the codewords have the same length, which is traditionally denoted $n$. (Sometimes an encoding scheme is used in which the codeword does not explicitly have the message intended among its digits. Then one must have a codebook showing which message corresponds to which codeword. We’ll see an example of this, too.)

2 The Hamming code

The Hamming code $H$ was one of the very first systematic codes presented. It is included in the papers of Claude Shannon [7] that first described a mathematical theory of information and communication. There Shannon showed that reliable communication was possible, but the needed codes were guaranteed to exist only by a probabilistic argument. There were no constructions, other than this one that Richard Hamming came up with.

The message set $M$ is $F_2^4$, the sixteen binary sequences of length 4. The goal achieved by the Hamming code is that it will identify the message sent if at most one digit is incorrectly received. One says it is a single-error-correcting code. A simple way to achieve that goal is simply to repeat each digit three times. Thus 0111 would be encoded as 000111111111.

But Hamming’s code $H$ achieves the same result with codewords of length 7 instead of 12! It uses elementary linear algebra over $F_2$. Here’s how:

- The code is presented as the row space of a generator matrix. This is a typical ingredient of what are called linear codes, codes that are subspaces of the ambient space $F_2^n$. The generator matrix for $H$ is

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

A message $m \in M$ is encoded as $mG$. Thus 0111 is encoded as 0111001. Notice that the presence of the $4 \times 4$ identity matrix at the beginning of $G$ guarantees that $m$ shows up as the first four digits in its codeword.

- An equivalent way to present $H$ is by a check matrix, which represents a list of equations that must be satisfied by the codewords. That is, $H$ is to be the solution
space of the equations. For $\mathcal{H}$ this matrix can be taken to be

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

The condition that $c \in \mathcal{H}$ is $cH^T = 0$, the superscript $T$ denoting transpose. Alternatively, the condition is that $c \cdot h = 0$ for all rows $h$ of $H$, where this $c \cdot h$ is the usual dot product, the sum of products of corresponding entries in $c$ and $h$. In fact, the row space of $H$ is the orthogonal subspace $\mathcal{H}^\perp$ of $\mathcal{H}$ under the dot product. As usual, $\mathcal{H}$ itself will be $\mathcal{J}^\perp$, where $\mathcal{J}$ is the row space of $H$.

- **Decoding:** Suppose the codeword $c \in \mathcal{H}$ is sent and the word $r$ (the received word) is received. Then $r = c + e$, where $e$ is the error word. If a digit $c_i$ in $c$ is incorrectly received as $r_i$, then $r_i = c_i + 1$, and $e_i = 1$, showing the “amount” of the error. For example, if $c = 1111111$ and $r = 1110111$, then $e = 0001000$, and $e$ represents a single error in position 4.

To see whether $r \in \mathcal{H}$, we would compute $rH^T$. Now $rH^T = cH^T + eH^T = eH^T$, since $cH^T = 0$. So although we can’t yet determine $c$, we can determine the syndrome $s = eH^T$ of the error (the syndrome of the “disease” $e$). In the example, $s = 110$, the fourth column of $H$, transposed. This points to the key idea: if $e$ has weight at most 1, the weight being the number of nonzero entries, then $s$ will be either 000 or the transposed column of $H$ at the location of the single nonzero entry of $e$. In other words, we can correctly decode using $\mathcal{H}$ if at most one error has occurred, just as we promised.

The Hamming code has an extra property: any syndrome is either 000 or some nonzero sequence of length 3. But all possible such sequences appear as columns of $H$. In other worlds, no matter what the error word $e$ is, it will look as though no or one error has occurred. For example, if $e = 1110111$, then $s = eH^T = 110$ also, and it looks as though that same single error at position 4 has occurred. So you would make a decoding error if you assumed there was at most one error.

But if $r$ is not a codeword, the most likely explanation of what went wrong is that it really was just one error. That is, we have decoded by using maximal likelihood decoding: assume that the most likely error event has occurred in producing the received word. If there is more than one possibility for the most likely codeword, we have a decoding failure.

### 3 Geometric view

A code of length $n$ (linear or not) is a subset of the ambient space $\mathbb{F}_2^n$. Define the distance (sometimes called the Hamming distance) $d(x, y)$ between two members $x$ and $y$ of $\mathbb{F}_2^n$ to be the number of positions in which $x$ and $y$ differ. Thus $d(x, y) = wt(x - y)$, the weight of
the difference $x - y$. (In this binary case, $x - y = x + y$, but for more general alphabets, it is $x - y$ that is needed.) Maximum likelihood decoding can now be expressed by saying that a received word is decoded as the nearest codeword to it—provided such a codeword exist. If there are ties, one has a decoding failure again.

In this set-up, the relevant parameter for a code $C$ is its **minimum distance**: the smallest distance seen between different codewords (we usually assume a code has more than one word!). Suppose that distance is $d$, and that the number of errors, in a transmission of $c \in C$ and $r$ received, is $e$ with $2e + 1 \leq d$. Then for any other codeword $c'$, one has $d \leq d(c, c') \leq d(c, r) + d(r, c')$ by the triangle inequality (which is still valid in this discrete setting), so that $d(r, c') \geq d - e \geq e + 1$. On the other hand, if $d \leq 2e$, then if $d(c, c') = d$, one could find a word $r$ with $d(c, r) \leq e$ and $d(c', r) \leq d(c, r)$ (why?), showing that there is a situation with fewer than $e$ errors and incorrect decoding. Incidentally, when a code is linear, the minimum distance is the same as the minimum weight among nonzero codewords, since $d(c, c') = \text{wt}(c - c')$ and $c - c'$ is a codeword.

We conclude that if the minimum distance of $C$ is $d$, then $C$ is an **$e$-error-correcting code**, for $e = \left\lfloor \frac{d-1}{2} \right\rfloor$. Now picture the **spheres** (or better, perhaps, the “balls”) of radius $e$ around the codewords of an $e$-error-correcting code $C$. Then these spheres must be disjoint, since a word within distance $e$ of one codeword must be at distance greater than $e$ from any other codeword. For example, how large could a 1-error-correcting code $C$ of length $n$ be? The words in the union of the spheres of radius 1 around the codewords appear just once. Each sphere contains $1 + n$ words, 1 for the center and $n$ at distance 1 from it, each such word being produced by changing one digit of the center word. So the number of words in the union of these spheres is $|C| (1 + n)$, where $|C|$ denotes the size of $C$. Thus we obtain an example of the **sphere-packing bound**:

$$|C| (1 + n) \leq |\mathbb{F}_2^n| = 2^n.$$

For the Hamming code $H$, we have $|H| = 16$, and $n = 7$. So we get equality in the sphere-packing bound. One says that $H$ is **perfect**.

What about 2-error-correcting codes? A sphere of radius 2 contains one center, $n$ words at distance 1 from it, and $\binom{n}{2}$ words at distance 2, each one obtained by changing two digits of the center. The sphere-packing bound now reads

$$|C| \left\{ 1 + n + \binom{n}{2} \right\} \leq 2^n,$$

that is,

$$|C| \left\{ \frac{n^2 + n + 2}{2} \right\} \leq 2^n.$$

If $C$ is to be perfect, we need both factors on the left to be powers of 2: $\frac{n^2 + n + 2}{2} = 2^j$, or $n^2 + n + 2 = 2^{j+1}$. If you solve the quadratic for $n$, you get

$$n = \frac{\sqrt{2^{j+3} - 7} - 1}{2},$$
so that $2^{j+3} - 7$ must be a square. This **exponential Diophantine equation** was considered by Ramanujan in 1913, and he conjectured that the only values possible for the exponent $j + 3$ were 3, 4, 5, 7, and 15. This was proved correct by Nagell in 1948. (For a short proof using algebraic number theory, see [6]. That paper lists a number of other references.) The parameters for $C$ corresponding to the possible choices of $j$ are then

\[
\begin{align*}
  j &= 0 \ 1 \ 2 \ 4 \ 12 \\
  n &= 0 \ 1 \ 2 \ 5 \ 90 \\
  \log_2 |C| &= 0 \ 0 \ 0 \ 1 \ 78
\end{align*}
\]

As far as codes go, only $j = 4$ and 12 make sense. There is a code for $j = 4$, namely any set of two words $x$ and $x + 11111$. But one can prove that there is no code for $j = 12$, linear or not (the linear possibility was ruled out by another coding pioneer, Marcel Golay, in 1949).

So far, the construction of codes seems rather ad hoc. Is there some systematic approach?

## 4 Cyclic codes

The answer is “yes”, and these codes about to be described have been the work-horse codes over the years. They are easily constructed, easily implemented, and their minimum distances are easy to bound from below. The only downside, perhaps, is that they are a bit tricky to decode. But a number of algorithms for that are available (see [?], Section 5.4).

The Hamming code $H$ provides a good example. Recall its check matrix:

\[
H = \begin{bmatrix}
1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}.
\]

and observe the cycling of the rows. Any further cycling (always moving the last digit to the front) gives a word in the row space of $H$. For example,

\[
1001011 = 1011100 + 0010111.
\]

Thus all seven **cyclic shifts** of the first row (that row itself counted as one) still have $H$ in their solution space. Moreover, $H$ also has the property that the cyclic shift of any of its words is still in $H$. For example, the original generator matrix can be replaced by one showing this cyclic property:

\[
G' = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}.
\]

One says that $H$ is a **cyclic code** (such codes are always taken to be linear).
There is a useful way to describe words in this context. Number the digit positions by $0, 1, \ldots, n - 1$ instead, and represent the word $c_1 c_2 \ldots c_{n-1}$ by the polynomial

$$c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}.$$  

Then the cyclic shift of this word is produced by multiplying by $x$, with the proviso that $x^n = 1$. Technically, we are identifying the ambient space $\mathbb{F}_2^n$ with the quotient ring $\mathbb{F}_2[X]/(X^n - 1)$, $x$ being the coset $X + (X^n - 1)$. I'll use $X$ whenever we need to talk about genuine polynomials, rather than polynomials in $x$.

Suppose that $C$ is a cyclic code of length $n$. Then $C$ has a generator polynomial, the (monic) polynomial $g(X)$ of least degree for which $g(x) \in C$. This generator polynomial has these properties:

- $g(X)$ is a divisor of $X^n - 1$.
- $\dim C = \deg g(X)$.
- The codewords of $C$ are the products $f(x)g(x)$, where $f(X)$ is any polynomial of degree at most $k - 1$.

This last item means that we can represent the messages by polynomials $f(x)$ with $\deg f(X) \leq k - 1$ and encode $f(x)$ as $f(x)g(x)$. Notice that the message digits generally won’t be singled out as certain digits of the resulting codeword. For the Hamming code $H$, we have $g(X) = X^3 + X + 1$, and indeed $X^7 - 1 = (X^3 + X + 1)(X^4 + X^3 + X + 1)$. The reverse polynomial $X^4 + X^3 + X + 1$ is the generator of the row space of the check matrix $H$, that is, of the orthogonal code $H^\perp$.

So the recipe for producing cyclic codes of given length $n$ is to find all the factors of $X^n - 1$, and use them as generator polynomials. That search is made easier by the fact that $\mathbb{F}_2[X]$ is a unique factorization domain. Thus we need only to find the irreducible factors of $X^n - 1$ and take products of them. When $n$ is odd, there are no repeated irreducible factors (unlike something like $X^4 - 1 = (X - 1)^4$, for example). There are tables, algorithms, and programs such as Maple and Mathematica for doing this factoring; see, for example, the comprehensive reference [5].

For instance, let $n = 21$. Here are the irreducible factors of $X^{21} - 1$:

$$

g_0(X) = X - 1 \\
g_1(X) = X^6 + X^4 + X^2 + X + 1 \\
g_3(X) = X^3 + X^2 + 1 \\
g_5(X) = X^6 + X^5 + X^4 + X^2 + 1 \\
g_7(X) = X^2 + X + 1 \\
g_9(X) = X^3 + X + 1.
$$

What’s the meaning of the indexing? In a large enough extension field of $\mathbb{F}_2$, the polynomial $X^{21} - 1$ factors into linear factors (the field involved is the splitting field of $X^{21} - 1$).
Moreover, each factor has the form \( X - z^i \), where \( z \) is a **primitive 21-st root of unity**. That is, \( z^{21} = 1 \), but no lower positive power of \( z \) is 1. It turns out that when an irreducible factor \( g(X) \) of \( X^{21} - 1 \) in \( \mathbb{F}_2[X] \) is factored in the splitting field, its linear factors have the roots \( z^i, z^{2j}, z^{4j}, \ldots \). The indices on the \( g_j \) show the lowest power of \( z \) that is a root. Here’s a table:

<table>
<thead>
<tr>
<th>name</th>
<th>degree</th>
<th>roots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g_0(X) )</td>
<td>1</td>
<td>( z^0 = 1 )</td>
</tr>
<tr>
<td>( g_1(X) )</td>
<td>6</td>
<td>( z, z^2, z^4, z^8, z^{16}, z^{32} = z^{11} (z^{22} = z) )</td>
</tr>
<tr>
<td>( g_3(X) )</td>
<td>3</td>
<td>( z^3, z^6, z^{12} (z^{24} = z^3) )</td>
</tr>
<tr>
<td>( g_5(X) )</td>
<td>6</td>
<td>( z^5, z^{10}, z^{20}, z^{40} = z^{19}, z^{38} = z^{17}, z^{34} = z^{13} (z^{26} = z^5) )</td>
</tr>
<tr>
<td>( g_7(X) )</td>
<td>2</td>
<td>( z^7, z^{14} (z^{28} = z^7) )</td>
</tr>
<tr>
<td>( g_9(X) )</td>
<td>3</td>
<td>( z^9, z^{18}, z^{36} = z^{15} (z^{30} = z^9) )</td>
</tr>
</tbody>
</table>

(For computation, one can use the fact that the splitting field of \( X^{21} - 1 \) is \( \mathbb{F}_2(z) \) and work out the other \( g_i \) from \( g_1 \).) Now comes the importance of knowing the roots of the \( g_i \):

**The BCH bound** (Bose, Ray-Chaudhuri, Hocquenghem): _Suppose that among the roots of the generator polynomial of a cyclic (binary) code \( C \) of odd length \( n \) there is a string \( z^a, z^{a+1}, \ldots, z^{a+\delta-2} \) of \( \delta - 1 \) consecutive powers of the primitive \( n \)-th root of unity, \( z \). Then the minimum distance of \( C \) is at least \( \delta \)._

For example, suppose we want a 2-error-correcting cyclic code of length 21. That requires minimum distance \( d \geq 5 \). Then a sequence of four consecutive powers of \( z \) as roots of the generator polynomial will guarantee that. In this case, we can use \( g(X) = g_1(X)g_3(X) \), with string \( z, z^2, z^3, z^4 \). Here \( \deg g(X) = 9 \), making \( \dim C = 21 - 9 = 12 \). Now

\[
g(X) = X^8 + X^7 + X^6 + X^4 + X^3 + X + 1,
\]

and

\[
(x^2 + x + 1)g(x) = x^{10} + x^8 + x^4 + x^3 + 1,
\]

a word in \( C \) of weight 5. So, in fact, \( d = 5 \). Notice that for any odd \( n \), the cyclic code with generator polynomial \( g_1(X) \), with roots \( z, z^2, \ldots \), always has that initial string with \( \delta = 3 \) and therefore minimum distance at least 3. That code will thus be single-error-correcting for sure.

For the code of length 21 with generator polynomial \( g_1(X)g_3(X)g_7(X)g_9(X) \), the powers of \( z \) that are *not* roots have exponents

\[
0, 5, 10, 13, 17, 19, 20,
\]

making it easy to spot that the longest string of consecutive roots has length 4. Thus \( d \geq 5 \). But it turns out that \( d = 8 \)! One research question is, can you improve the BCH bound? Yes: see [?], p. 154] for a method of doing so, due to van Lint and Wilson.
5 The $[11, 6]$ ternary Golay code

Many other alphabets have been used for codes. Generally they are finite fields, but rings like $\mathbb{Z}_4$, the integers modulo 4, have led to a number of interesting codes. A common alphabet is $\mathbb{F}_3$, the ternary field with members 0, 1, 2 and arithmetic modulo 3 (so $-1 = 2$). Many of the preceding ideas carry over, especially the construction of cyclic codes and the BCH bound. We’ll set up a famous ternary cyclic code discovered by Marcel Golay, of length $n = 11$: the $[11, 6]$ ternary Golay code.

First we need the irreducible factors of $X^{11} - 1$ and their roots in terms of a primitive 11-th root of unity, $z$. The powers of $z$ that are roots of a factor now have the form $z^j, z^{2j}, z^{3j}, \ldots$. Here’s the result:

<table>
<thead>
<tr>
<th>name and expression</th>
<th>degree</th>
<th>roots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_0(X) = X - 1$</td>
<td>1</td>
<td>$z^0 = 1$</td>
</tr>
<tr>
<td>$g_1(X) = X^5 + 2X^3 + X^2 + 2X + 2$</td>
<td>5</td>
<td>$z, z^3, z^9, z^{27} = z^5, z^{15} = z^4 (z^{12} = z)$</td>
</tr>
<tr>
<td>$g_2(X) = X^5 + X^4 + 2X^3 + X^2 + 2$</td>
<td>5</td>
<td>$z^2, z^6, z^{18} = z^7, z^{21} = z^{10}, z^{30} = z^8 (z^{24} = z^2)$</td>
</tr>
</tbody>
</table>

Let $G$ be the code with generator polynomial $g_1(X)$. The BCH bound with the sequence $z^3, z^4, z^5$ shows that $d \geq 4$. However, $d$ is actually 5 (the weight of $g_1(x)$).

The sphere-packing bound applied to $G$ reads

$$|G| \left\{ 1 + 2 \times 11 + 2^2 \times \binom{11}{2} \right\} \leq 3^{11},$$

since for the ternary case, moving a distance 1 from the center involves not only a choice of digit to be changed, but also a choice of how much to change the digit—either by 1 or by $-1$. Similarly, moving a distance of 2 will involve two changes, one for each position. Putting in the numbers, we get

$$3^6 \times (1 + 22 + 220) = 3^6 \times 243 = 3^6 \times 3^5 = 3^{11}$$

on the left side and we have equality in the sphere-packing bound—the code is perfect!

How many words of weight 5 are there in $G$? Each word of weight 3 in $\mathbb{F}_3^{11}$ is at distance 2 from exactly one word in $G$, and that word can only be of weight 5 and have the given weight 3 word “embedded” in it. Schematically, the picture must be

$$\begin{array}{ccccccc}
  x_1 & x_2 & x_3 & 0 & 0 & \ldots & 0 \\
x_1 & x_2 & x_3 & x_4 & x_5 & \ldots & 0 \\
\end{array}$$

On the other hand, each word in $G$ of weight 5 is at distance 2 from $\binom{5}{1} = 10$ (embedded) words in $\mathbb{F}_3^{11}$ of weight 3. So “double counting”, that is, counting the number of pairs $(t, c)$, where $t$ is a word of weight 3 embedded in the word $c$ of weight 5 in $G$, gives

$$wt(c) \geq 5.$$
• counting by \( t \): \( \binom{11}{3} \times 8 \times 1 \);

• counting by \( c \): \( A_5 \times 10 \), where \( A_5 \) is the number of words of weight 5 in \( G \).

So \( \binom{11}{3} \times 8 \times 1 = 10A_5 \), and \( A_5 = (165 \times 8)/10 = 132 \). These words occur in \( \pm \) pairs. Thus there are 66 supports for such words, that is, subsets of the 11 digit positions where the nonzero digits of a word of weight 5 is located. (Two words of weight 5 cannot have the same support without being scalar multiples of each other, since otherwise some combination would be a nonzero word of weight less than 5 in \( G \).) These supports are more often called blocks. Two different blocks cannot meet in more than 3 places (why?). So the blocks, all together, contain \( 66 \times \binom{5}{4} = 330 \) quadruples of digit locations. But, \( \binom{14}{4} = 330 \). That means that each quadruple appears exactly once in a block. What we have is called a Steiner 4-design. These are very hard to come by, although there is an infinite number of them.

Here’s a bibliography of selected books and the items cited in the text.

**References**

A very readable elementary introduction with all kinds of special topics.

This is a comprehensive textbook that covers many items of current research interest.

Another good text, also with several special topics.

State-of-the-art coverage of all major coding-theory topics (2200+ pages!).

This just came out and it is very thorough. It has several contributions by a former post-doc of mine, Gary McGuire, who recently proved that sudoku puzzles must have at least 17 clues!


This contains reprints of the fundamental papers by Shannon, along with a popular exposition by Weaver. Shannon’s papers are also available online.