Homework #1 Solutions

1. **Convolutional code.** A simple convolutional code with rate 1/2 has encoding equations

\[ c_1^i = m_i, \quad c_2^i = m_i \oplus m_{i-1}, \]

where \( m_i \) is an information bit and \( c_1^i, c_2^i \) are the corresponding codeword bits. For example, \( m_1 m_2 m_3 = 101 \) is encoded to \( c_1^1 c_2^1 c_1^2 c_2^2 c_1^3 c_2^3 = 110111 \) (assuming that \( m_0 = 0 \)). This code can correct single bit errors that are sufficiently far apart.

a. Each information bit \( m_i \) affects three codeword bits. Use these three equations to obtain a majority-logic decoder for this convolutional code.

b. Find the minimum separation between errors that guarantees that errors can be corrected by the decoder of part (a). (Encoded bits are transmitted in the order \( c_1^i, c_2^i \).)

c. (Bonus) Describe a decoding method that is more “powerful” than the method of part (a).

**Solution** (20 points)

a. The encoding equations provide two equations that include the information bit \( m_i \):

\[ c_1^i = m_i, \quad c_2^i = m_i \oplus m_{i-1}. \]

A third equation arises from the effect of \( m_i \) on the next codeword block:

\[ c_{i+1}^2 = m_{i+1} \oplus m_i \]

Using \( c_{i-1}^1 = m_{i-1} \) and \( c_{i+1}^1 = m_{i+1} \), we can rewrite these three equations as follows:

\[ m_i = c_i^1, \quad m_i = c_{i-1}^1 \oplus c_i^2, \quad m_i = c_{i+1}^1 \oplus c_{i+1}^2 \]

Each equation provides one vote for the value of \( m_i \), leading to the decoding equation

\[ \hat{m}_i = \text{majority}( c_i^1, c_{i-1}^1 \oplus c_i^2, c_{i+1}^1 \oplus c_{i+1}^2 ). \]

b. If the information bit \( c_i^1 \) is incorrect, but the nearby received bits are correct, the last two equations will outvote the incorrect first equation. Similarly, any single error in the 6 consecutive bits

\[ \ldots c_{i-1}^1, c_{i-2}^1, c_i^1, c_{i+1}^1, c_{i+1}^2, c_{i+2}^2, \ldots \]

will be corrected because each bit appears in at most one of the three equations. Thus errors that are 6 or more bit positions apart can always be corrected, because the decoding window of 6 bits will include at most one wrong bit. On the other hand, errors in \( c_{i-1}^1 \) and \( c_{i+1}^2 \) will result in miscorrecting \( m_i \). The distance between these two incorrect bits is 5. Therefore 6 is the minimum distance between bit errors that guarantees that the errors will be corrected using the decoder of part (a).

c. (10 points) The decoder can be improved by using the previously decoded estimate \( \hat{m}_{i-1} \) instead of \( c_{i-1}^1 \) in the second equation:

\[ \hat{m}_i = \text{majority}( c_i^1, \hat{m}_{i-1} \oplus c_i^2, c_{i+1}^1 \oplus c_{i+1}^2 ). \]
This majority logic decoder with feedback operates successfully as long as there is at most one error in any two consecutive 2-bit received blocks, which is guaranteed by a distance of at least 4 between bit errors.

2. Simple product code. A (9,4) binary simple product code is shown in the left part of the figure below. Bits 1–4 are message bits, while bits 5–6 are rows checks and bits 7–9 are column checks.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

a. What is the minimum distance of this simple product code?
b. Find all codewords of minimum weight.

Now suppose that the code is modified by dropping bit 9, as shown in the right part of the figure above. The resulting expurgated code is an (8,4) code.

c. What is the minimum distance of the modified simple product code?
d. Find all codewords of minimum weight.
e. For \( k = 0, \ldots, 8 \) find the number of codewords with \( k \) ones. This sequence is called the weight distribution of the code.

**Solution (25 points)**

a. If we change any message bit then that bit’s row and column change parity, so the row and column checks must also change to maintain even parity. But now the last column and row have changed parity, so the overall parity-check bit (bit 9) must change. So every other codeword must differ in at least four bits, that is, \( d^* \geq 4 \). On the other hand, if we change all four message bits then the new block has even parity for all rows and columns, hence is a codeword that differs in exactly four bits from the original codeword. Therefore \( d^* = 4 \).

b. Since there are four message bits, there are \( 2^4 = 16 \) codewords. The 15 nonzero codewords are shown below in order of weight of the message block.

<table>
<thead>
<tr>
<th>1 0 1</th>
<th>0 1 1</th>
<th>0 0 0</th>
<th>0 0 0</th>
<th>1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 1 1</td>
<td>1 0 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
<td>1 0 1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 1 1</td>
<td>0 0 0</td>
<td>1 0 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 1 1</td>
<td>1 1 0</td>
<td>0 1 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 0 0</td>
<td>1 1 0</td>
<td>1 1 0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 0 1</td>
<td>1 1 0</td>
<td>1 1 0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 0</td>
<td>1 0 1</td>
<td>0 1 1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
<td>1 0 1</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

By inspection, we see that there are nine codewords of weight 4 and six codewords of weight 6. We can obtain this result more systematically by considering the number of 1’s in the message block.

1. A single 1 in the message results in a codeword of weight 4. There are 4 such codewords.
2. If two 1’s are located in the same row or column of the message block, the resulting code
word has weight 4. There are 4 such codewords. The other two codewords with weight 2
message blocks have weight 6.

3. All codewords with three 1’s in the message have weight 6.

4. There is only one codeword with four 1’s in the message. It has weight 4.

c. If a codeword of the original product code has weight 4 and nonzero bit 9, then the modified
codeword has weight 3. Therefore the minimum distance of the code is the difference between
the zero codeword and any of the weight 3 codewords, that is, $d^* = 3$.

d. By inspection of the codewords from part (b) we find the following codewords of weight 3.

\[
\begin{array}{ccc}
1 & 0 & 1 \\
0 & 0 & 0 \\
1 & 0 & 1 \\
\end{array} \quad 
\begin{array}{ccc}
0 & 1 & 1 \\
0 & 0 & 0 \\
0 & 1 & 1 \\
\end{array} \quad 
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
\end{array}
\]

e. The weight distribution of the original code is $1, 0, 0, 0, 9, 0, 6, 0, 0, 0$; that is, there is one zero
codeword, nine codewords of weight 4, and six codewords of weight 6. When we delete bit 9,
we obtain four codewords of weight 3 and and four codewords of weight 5, converted from
codewords of weights 4 and 6, respectively. The weight distribution of the modified code is
$1, 0, 0, 4, 5, 4, 2, 0, 0$.

Weight distributions are often represented by polynomials called weight enumerators. These
are simply finite generating functions. The weight enumerators for the original and modified
simple product codes are

$$A(x) = 1 + 9x^4 + 6x^6$$

and

$$\tilde{A}(x) = 1 + 4x^3 + 5x^4 + 4x^5 + 2x^6.$$

3. Cooked error rate. A (15,11) binary Hamming code is used on a binary symmetric channel with
raw error rate $10^{-3}$. Find the “cooked” error rate — the probability that a decoded mes-
gage bit is incorrect. Fact: when two bit errors occur, the decoder incorrectly changes a third bit in the
codeword. You may assume that the probability of $\geq 3$ errors is negligible. Bonus: take into
account the possibility of 3 errors.

Solution (10 points)
For raw bit error rate $10^{-3}$, the probability of two errors in a codeword is given by the binomial
distribution:

$$P\{2 \text{ raw errors}\} = \binom{15}{2} (10^{-3})^2 (1 - 10^{-3})^{13} = 1.0364 \times 10^{-4}.$$ 

(For now, we ignore the probability of three errors, which is $4.4957 \times 10^{-7}$.) When two errors
occur in a received codeword of length 15, the decoder miscorrects by changing one more bit to
arrive at a codeword that differs from the transmitted codeword in three bits. The probability
that any particular bit within a codeword (information bit or check bit) is incorrect is the average
number of wrong bits per codeword divided by the number of bits per codeword. Therefore the
“cooked” bit error rate is

$$\frac{3}{15} \times P\{2 \text{ raw errors}\} = 2.0729 \times 10^{-5}.$$
The cooked error rate is about two orders of magnitude smaller than the raw error rate. For raw error rate \( \epsilon < 2 \times 10^{-3} \), an accurate approximation to the cooked error rate is

\[
\frac{3}{15} \cdot \binom{15}{2} \epsilon^2 = 21\epsilon^2.
\]

Now consider three errors. If the decoder always miscorrects a fourth bit, then the contribution to the cooked error rate is

\[
\frac{4}{15} \times P\{3 \text{ raw errors}\} = 1.1989 \times 10^{-7}.
\]

In fact, when the three errors correspond to a codeword, the decoder does not change a fourth bit. Of the 455 15-tuples of weight 3, 35 are codewords. The exact contribution to the cooked error rate is

\[
\left(\frac{35}{455} \cdot \frac{3}{15} + \frac{420}{455} \cdot \frac{4}{15}\right) \times P\{3 \text{ raw errors}\} = 1.1758 \times 10^{-7}.
\]

Combining the cases for two and three raw errors and ignoring four or more errors, the cooked error rate is

\[
2.0729 \times 10^{-5} + 1.1758 \times 10^{-7} = 2.0847 \times 10^{-5}.
\]

The weight distribution for Hamming codes could be used to obtain the exact value for the cooked error rate. However, the exact answer is not useful in practice because it depends on an idealized error model (Bernoulli errors) and also requires a precise estimate of the raw error rate.

4. Rates of block codes. (Combinatorial/Computational)

a. Let \( \mathcal{C}_1 \) be the binary code of blocklength 14 consisting of all sequences in which there are at least three 0s between any two 1s. Find the rate of \( \mathcal{C}_1 \).

b. Let \( \mathcal{C}_2 \) be the code over alphabet \( \{-2, -1, 0, +1, +2\} \) of blocklength 5 consisting of all sequences such that the sum of the codeword components is 0. Find the rate of \( \mathcal{C}_2 \).

**Solution** (20 points)

The rate of a block code with alphabet size \( Q \), blocklength \( n \), and \( M \) codewords is \( \frac{1}{n} \log_Q M \).

a. First we give two computational solutions. Let \( M_n \) be the number of binary sequences of length \( n \) that have at least three 0s between any two 1s. The first four values of \( M_n \) are given in the following table together with the codewords of length \( n \).

<table>
<thead>
<tr>
<th>( n )</th>
<th>( M_n )</th>
<th>code of blocklength ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>{0, 1}</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>{00, 01, 10}</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>{000, 001, 010, 100}</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>{0000, 0001, 0010, 0100, 1000}</td>
</tr>
</tbody>
</table>

For \( n \geq 5 \) any valid sequence begins with either 0 or 1. If the first bit is 0, then the remaining \( n - 1 \) bits can be any valid sequence of length \( n - 1 \). If the first bit is 1, then the next three bits must be 0, followed by any valid sequence of length \( n - 4 \). Therefore \( \{M_n\} \) satisfies the following recurrence:

\[
M_n = \begin{cases} 
  n + 1 & \text{if } n = 1, 2, 3, 4 \\
  M_{n-1} + M_{n-4} & \text{if } n \geq 5
\end{cases}
\]
Using this recurrence, we can generate $M_n$ for $n = 5, \ldots, 14$:

\[
\begin{array}{cccccccccc}
  n & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
  M_n & 7 & 10 & 14 & 19 & 26 & 36 & 50 & 60 & 95 & 131 \\
\end{array}
\]

Therefore the rate of $C_1$ is $\frac{1}{14} \log_2 131 = \frac{1}{14} \cdot 7.033 = 0.5024$.

Next we give a combinatorial solution. Let us count the number of acceptable sequences by breaking down according to the number of 1’s in the sequence. Since each 1 must be followed by three or more 0’s (except at the end of the sequence), the maximum number of 1’s in a sequence of length $n$ is $\lceil n/4 \rceil$. For $n = 14$, this is 4.

The key insight is to treat each subsequence 1000 as a single object that consumes 4 bits. The set of codewords with $k$ 1’s can be constructed by first placing $k - 1$ objects 1000 and a single object 1 into the codeword. Then the number of 0’s that remain to be distributed between these objects is $m = 14 - 4(k - 1) - 1 = 17 - 4k$.

How many ways are there to distribute $m$ objects amongst (between and to the outside of) $k$ partitions? A famous method is called stars and bars. The trick is to imagine the partitions as bars (|) and the space between the partitions as stars (*). For example, one way to distribute 5 objects amongst 3 partitions can be visualized as follows:

\[
* | ** | * 
\]

In this case, the total number of ways to distribute 5 objects amongst the 3 partitions is

\[
\frac{8!}{3!5!} = \binom{8}{5},
\]

because there are 8! arrangements of the 8 items, but the 3! arrangements of the bars and the 5! arrangements of the stars are indistinguishable.

In general, the expression for distributing $m$ objects amongst $k$ partitions is

\[
\binom{m + k}{m} = \binom{m + k}{k}.
\]

The second expression can be understood as the number of ways of inserting $k$ partitions somewhere within a line of $m$ objects.

For our problem, if $k$ is the number of 1’s then the number of 0’s in an acceptable sequence of length $n$ is

\[
m = n - 4(k - 1) - 1 = n - 3k + 3.
\]

Let $M_{n,k}$ be the number of codewords of length $n$ having $k$ ones. Then

\[
M_{n,k} = \binom{m + k}{m} = \binom{n - 3k + 3}{n - 4k + 3} = \binom{n - 3k + 3}{k}.
\]

The general expression for the total number of codewords is

\[
M_n = \sum_{k=0}^{\lceil n/4 \rceil} M_{n,k} = \sum_{k=0}^{\lceil n/4 \rceil} \binom{n - 3k + 3}{k}.
\]
Plugging in \( n = 14 \), we get
\[
M_{14} = \sum_{k=0}^{4} \binom{17 - 3k}{k} = \binom{17}{0} + \binom{14}{1} + \binom{11}{2} + \binom{8}{3} + \binom{5}{4} = 1 + 14 + 55 + 56 + 5 = 131.
\]

b. The number of codewords of length \( n \) having a particular sum can be found using convolution. Consider the vector \( s \) with entries defined as follows:
\[
s(k) = \begin{cases} 
1 & \text{if } k \in \{-2, -1, 0, +1, +2\} \\
0 & \text{otherwise}
\end{cases}
\]
The entry \( s(k) \) gives the number of codewords of length 1 whose weight (sum of components) is equal to \( k \). Let \( r_n \) be the vector such that \( r_n(k) \) gives the number of codewords of length \( n \) with weight \( k \). Then \( r_1 = s \).

A codeword of length \( n \) with weight \( k \) can be found as a codeword of length \( n - 1 \) with weight \( k - i \), and then appending the symbol \( i \) at the end of the codeword. This procedure is comprehensive in the sense that all codewords of length \( n \) with a particular sum \( k \) can be constructed from codewords of length \( n - 1 \). The entries of \( r_n \) can be computed sequentially as follows.
\[
r_n(k) = \sum_{i=-\infty}^{\infty} s(i)r_{n-1}(k-i).
\]
The sum is actually over \( i \in \{-2, -1, 0, +1, +2\} \), since all other entries of \( s(i) = 0 \). The product \( s(i)r_{n-1}(k-i) \) thus adds one more possible codeword of weight \( k \) to the count in \( r_n \) for each codeword of weight \( k - i \) counted in \( r_{n-1} \). Using convolution,
\[
r_n = s \ast r_{n-1}.
\]
The number of codewords of length \( n \) which sums to 0 is the entry \( r_n(0) \).

Matlab can be used to compute the convolution and find the number of codewords \( M_n \) summing to 0.

\begin{verbatim}
nmax = 5; s = [1 1 1 1 1]; r = s; M(1) = r(3); for n=2:nmax r = conv(s,r); M(n) = r(2*n+1); end
\end{verbatim}

The output gives \( M = 381 \) for \( n = 5 \). The rate of \( C_2 \) is \( \frac{1}{5} \log_5 381 = 0.7385 \). Since 381 \( \geq 256 \), a subset of \( C_2 \) can be used to represent 8-bit bytes with balanced 5-ary 5-tuples.

A brute force computational approach can also be used: enumerate all \( 5^5 = 3125 \) 5-ary 5-tuples and count those of weight 0.