Hamming codes: review

The \((7, 4)\) binary Hamming code consists of \(2^4 = 16\) 7-bit codewords that satisfy three parity-check equations.

\[
\begin{align*}
c_1 \oplus c_3 \oplus c_5 \oplus c_7 &= 0 \\
c_2 \oplus c_3 \oplus c_6 \oplus c_7 &= 0 \\
c_4 \oplus c_5 \oplus c_6 \oplus c_7 &= 0
\end{align*}
\]

We can characterize the code using the parity-check matrix \(H\):

\[
c H^T = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}^T = 0
\]

Check bits \(c_1, c_2, c_4\) can be computed from \(c_3, c_5, c_6, c_7\).

\[
\begin{align*}
c_1 &= c_3 \oplus c_5 \oplus c_7 \\
c_2 &= c_3 \oplus c_6 \oplus c_7 \\
c_4 &= c_5 \oplus c_6 \oplus c_7
\end{align*}
\]

\[
\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}
\]
Hamming codes: error detection and correction

Each codeword bit affects at least one equation. Therefore every single-bit error can be detected.

Each bit is checked by a unique set of equations. Therefore the error location can be determined by which parity-check equations fail.

**Definition:** the syndrome \( s = [s_0 \ s_1 \ s_2] \) of received vector \( r = [r_1 \ r_2 \ldots r_7] \) is the binary vector that tells which parity-check equations are not satisfied.

\[
\begin{align*}
s_0 &= r_1 \oplus r_3 \oplus r_5 \oplus r_7 \\
s_1 &= r_2 \oplus r_3 \oplus r_6 \oplus r_7 \quad \iff \quad [s_0 \ s_1 \ s_2] = [r_1 \ldots r_7] H^T \\
s_2 &= r_4 \oplus r_5 \oplus r_6 \oplus r_7
\end{align*}
\]

When \( s = 0 \), the decoder assumes that no error has occurred. This is the most likely conclusion under reasonable assumptions.

Each nonzero value of \( s \) corresponds to an error in exactly one of \( 2^3 - 1 = 7 \) bit positions. The syndrome identifies the location of a single error.

For this parity-check matrix \( H \), the syndrome \( s = [s_0 \ s_1 \ s_2] \) is the binary representation of the assumed error location (most significant bit is \( s_2 \)).

EE 387, September 28, 2015
Hamming codes: minimum distance

Hamming codes can correct single errors. Thus $d^* \geq 2t + 1 = 2 \cdot 1 + 1 = 3$.

When used for error detection only, Hamming codes detect double errors.

Fact: minimum distance is exactly 3. Therefore Hamming codes can either correct single errors or detect double errors (but not both simultaneously).

A Hamming code with $m$ parity-check bits has $2^m - 1$ nonzero syndromes, hence blocklength $n = 2^m - 1$. The rate quickly approaches 1 for large $n$.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>$k$</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0.3333</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>4</td>
<td>0.5714</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>11</td>
<td>0.7333</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>26</td>
<td>0.8387</td>
</tr>
<tr>
<td>6</td>
<td>63</td>
<td>57</td>
<td>0.9047</td>
</tr>
<tr>
<td>8</td>
<td>255</td>
<td>247</td>
<td>0.9686</td>
</tr>
<tr>
<td>15</td>
<td>32767</td>
<td>32752</td>
<td>0.9995</td>
</tr>
<tr>
<td>32</td>
<td>4294967295</td>
<td>4294967263</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Extended (expanded, expurgated) Hamming codes

Two easy ways to “extend” a Hamming code:

- **Add overall parity-check bit**: \( c_0 = c_1 \oplus \cdots \oplus c_7 \iff c_0 \oplus \cdots \oplus c_7 = 0 \).

  \[
  H_1 = \begin{array}{cccccccc}
    0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
    0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
    0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
    1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  \end{array}
  \]

  This expanded code has blocklength 8 but same number of codewords. Code parameters: \((8, 4, 4)\), rate 1/2.

- **Add overall parity-check equation**: \( c_1 \oplus c_2 \oplus \cdots \oplus c_6 \oplus c_7 = 0 \).

  \[
  H_2 = \begin{array}{cccccccc}
    1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
    0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
    0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
    1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  \end{array}
  \]

  This expurgated code consists of Hamming codewords with even parity. Code parameters: \((7, 3, 4)\), rate 3/7.
Extended Hamming codes: minimum distance

Both expanded and expurgated Hamming codes are constructed by adding redundancy to code with minimum distance 3.

- The minimum distance of extended codes is no smaller, hence $\geq 3$.
- All codewords have even parity, so distance between codewords is even.

Therefore the minimum distance is an even number and so is $\geq 4$.

- Hamming codes contain codewords of weight 3.
- The additional parity-check bit increases distance by at most 1.

Therefore the minimum distance of extended Hamming codes is $d^* = 4$.

These codes can correct single errors and simultaneously detect double errors.

Double error is indicated by nonzero syndrome but even overall parity.
General product codes

Let $C_1$ be an $(n_1, k_1)$ block code and let $C_2$ be an $(n_2, k_2)$ block code. The product code $C_1 \otimes C_2$ is an $(n_1 n_2, k_1 k_2)$ code.

Encoder (systematic) for product code:

- First arrange $k_1 k_2$ information symbols in a $k_2 \times k_1$ array.
- Then encode first $k_2$ rows using code $C_1$.
- Finally encode all $n_1$ columns using code $C_2$.

Fact: the minimum distance of $C_1 \otimes C_2$ is $d^* = d_1^* \cdot d_2^*$.

By definition, every column is a codeword of $C_2$. But if $C_1$ and $C_2$ are linear codes, then all rows are codewords of $C_1$. This definition assumes systematic encoders for $C_1$ and $C_2$. 
General product code example

Consider the product of two $(8, 4, 4)$ expanded Hamming codes.

![Diagram of product code parameters]

Product code parameters: $(n, k, d^*) = (64, 16, 16)$. Rate: $1/4$

Error correcting ability: $t = \left\lfloor \frac{(16 - 1)}{2} \right\rfloor = 7$

Product codes can be decoded up to the guaranteed error correcting ability. The decoding procedure requires a column decoder that can correct both errors and erasures. (Blahut chapter 12.)

We will find more efficient codes; e.g., the $(64, 25, 16)$ expanded BCH code needs only 39 check bits for same minimum distance.
Nonbinary single error correcting code

The single check equation

\[ c_1 + c_2 + \cdots + c_n = 0 \]

allows detection of a single symbol error in a received \( n \)-tuple.

Furthermore, the syndrome \( s \) defined by

\[ s = r_1 + r_2 + \cdots + r_n \]

indicates the *magnitude* of the error. If the error is in location \( i \) and the incorrect symbol is \( r_i = c_i + e_i \), then

\[ s = r_1 + r_2 + \cdots + r_n = c_1 + \cdots + (c_i + e_i) + \cdots + c_n = e_i . \]

The syndrome tells exactly what should be subtracted from the incorrect symbol in order to obtain a codeword.

What is not known is where the error is— which symbol is wrong.
More equations needed

A second equation is needed to identify the error location. The effect of an error magnitude on the syndrome should be different for each location.

A reasonable choice for this second equation:

$$1 \cdot c_1 + 2 \cdot c_2 + \cdots + n \cdot c_n = 0.$$ 

Now every valid codeword satisfies two equations:

$$1 \cdot c_1 + 1 \cdot c_2 + \cdots + 1 \cdot c_n = 0$$

$$1 \cdot c_1 + 2 \cdot c_2 + \cdots + n \cdot c_n = 0$$

We can derive encoding equations to express $c_1, c_2$ in terms of $c_3, \ldots, c_n$.

*Example:* Let symbols be 4-bit values with addition modulo 16. For $n = 15$, 

$$H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & 15 \end{bmatrix}$$

is parity-check matrix for a code that can correct single symbols errors. Almost.
Decoding procedure

Suppose there is a single error of magnitude $e_i \neq 0$ in location $i$. The syndrome $s = [s_0 \ s_1]$ can be expressed in terms of unknowns $i$ and $e_i$:

\[
\begin{align*}
  s_0 &= \sum_{j=1}^{n} r_j = e_i + \sum_{j=1}^{n} c_j = e_i \\
  s_1 &= \sum_{j=1}^{n} j r_j = i e_i + \sum_{j=1}^{n} j c_j = i e_i
\end{align*}
\]

We can determine $e_i$ and $i$ from the syndrome equations:

\[
\begin{align*}
  e_i &= s_0 \\
  i &= \frac{i e_i}{e_i} = \frac{s_1}{s_0}
\end{align*}
\]

Sadly, division is not always defined for modulo 16 arithmetic. E.g., suppose $s_0 = 4, s_1 = 8$. Then $s_1 = is_0 \mod 16$ has four solutions:

\[2, 6, 10, 14.\]

We cannot be certain where the single error is located.
Finite fields

This problem with division is solved by using a “better” multiplication. We will define GF(16), the field of 16 elements.

In GF(16), multiplication has an inverse operation of division, and most of the other familiar properties of arithmetic are valid.

Another approach: mod 17 arithmetic with channel alphabet \{0, 1, \ldots, 16\}.

The “parity-check” matrix for a 1EC code over GF(17) is

\[ H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & 16 \end{bmatrix}. \]

The error pattern and location can be computed using the above equations:

- error pattern: \( e_i = s_0 \)
- error location: \( i = \frac{s_1}{s_0} \)

Using either GF(16) or modulo 17 arithmetic, these equations can be solved when \( s_0 \neq 0 \).
Reed-Solomon codes

The codes over GF(16) and GF(17) are examples of Reed-Solomon codes. Reed-Solomon codes use symbols from finite field GF($Q$) and have $n = Q - 1$.

Each row of $H$ consists of consecutive powers of elements of GF($Q$).

When the elements are chosen carefully, each additional check equation increases the minimum distance by 1.

For example, the following parity-check matrix corresponds to 4 equations:

$$H = \begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & 2 & 3 & \cdots & 16 \\
1 & 4 & 9 & \cdots & 256 \\
1 & 8 & 27 & \cdots & 4096
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & 2 & 3 & \cdots & 16 \\
1 & 4 & 9 & \cdots & 1 \\
1 & 8 & 10 & \cdots & 16
\end{bmatrix}$$

This PC matrix defines a code over GF(17) with minimum distance 5. It can correct two symbol errors in a codeword of length 16.

Decoding procedures for Reed-Solomon codes are chief goal of this course.