CODING

- Two Problems and Coding Solutions
- Information
- Huffman Coding
- Linear Block Coding

Two Problems and Coding Solutions

1. *Problem*: Inefficiently coded messages squander channel capacity.
   - *Solution*: Pre-process message to improve efficiency (aka source coding).
   - *Example*: Huffman coding
2. *Problem*: Messages must be resistant to bit errors.
   - *Solution*: Add bits to message with structure that can help reveal errors in noisy recovered bits (aka channel or error-correcting coding).
   - *Example*: Linear block coding

Information

- Consider a message composed from symbols x_i occuring with probabilities $p(x_i)$ (where $\sum_i p(x_i) = 1$).
- The message sequence terms are composed independently, i.e. probability of a particular x_i at time k does not depend on message sequence terms chosen before or after time k.
- We will define information conveyed by a message symbol in bits.
- We will extend this to its mean over a symbol alphabet and call it entropy with units of bits/symbol.

Information (cont'd)

- We will discover that: *Maximum entropy (base 2 log of the number of symbols in alphabet) for sequence drawn independently from alphabet occurs when symbols occur equally probably.*
- We will define a code's efficiency as the symbol alphabet's entropy divided by the average number of bits per symbol used by the code.
- We will discover that: *Maximum efficiency is achieved by a code that conveys an average number of bits per symbol equal to the symbol alphabet's entropy.*

Information (cont'd)

Definition:
Our definition of information should satisfy the following four properties:

(i) two equally probable symbols should give the same information

$$p(x_i) = p(x_j) \Rightarrow I(x_i) = I(x_j)$$

(ii) a less likely symbol gives more information

$$p(x_i) < p(x_j) \Rightarrow I(x_i) > I(x_j)$$

(iii) if there is no other choice than a particular symbol, then it conveys no information

$$p(x_i) = 1 \Rightarrow I(x_i) = 0$$

Information (cont'd)

Definition (cont'd)

(iv) for two independent symbols, their information as a pair should equal the sum of their individual information values
$p(x_i \text{ and } x_j) = p(x_i)p(x_j)$

$$\Rightarrow I(x_i \text{ and } x_j) = I(x_i) + I(x_j)$$

There is one (and only one) possibility for a function that satisifes all four of these properties:

$$I(x_i) = \log_2\left(\frac{1}{p(x_i)}\right) = -\log_2(p(x_i))$$

which defines the information (measured in bits) conveyed by a particular message symbol.

Example: With probabilities of occurence $p(x_1) = 1/2$, $p(x_2) = 1/4$, and $p(x_3) = 1/4$,

$$I(x_1) = \log_2\left(\frac{1}{p(x_1)}\right) = \log_2(2) = 1$$

Information (cont'd)

Entropy:
Extending information measure from symbol to sequence, define the entropy (or mean information) of an alphabet with independent symbols of various probabilities

$$H(x) = \sum_{i=1}^{N} p(x_i) I(x_i)$$

$$= \sum_{i=1}^{N} p(x_i) \log\left(\frac{1}{p(x_i)}\right)$$

$$= -\sum_{i=1}^{N} p(x_i) \log(p(x_i))$$

with units of bits/symbol.

Information (cont'd)

Example:

- 4-symbol Source A:
  - Symbol probabilities: $p(x_1) = 0.5$, $p(x_2) = 0.25$, $p(x_3) = p(x_4) = 0.125$
  - Total information:

$$\sum_{i=1}^{4} I(x_i) = \sum_{i=1}^{4} \log\left(\frac{1}{p(x_i)}\right)$$

$$= 1 + 3 + 2 + 1 = 7 \text{ bits}$$

  - Entropy of the source:

$$H(x) = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3$$

$$= 1.75 \text{ bits/symbol.}$$

Information (cont'd)

Another Example:

- 4-symbol Source B:
  - source symbol probabilities: $p(x_i) = 0.25$ for all i
  - Total information: $I(x_i) = 2 + 2 + 2 + 2 = 8$
  - Entropy of the source:

$$H(x) = \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2$$

$$= 2 \text{ bits/symbol.}$$

- These examples suggest that maximum entropy achieved by sources with equally probable symbols.

Information (cont'd)

Maximizing Entropy:
To confirm this observation about equally probable independent symbols maximizing entropy examine the entropy minus the base 2 log of the number of symbols N with $\sum_{i=1}^{N} p(x_i) = 1$

$$H(x) - \log(N) = \sum_{i=1}^{N} p(x_i) \log\left(\frac{1}{p(x_i)}\right) - \log(N)$$

$$= \sum_{i=1}^{N} p(x_i) \log\left(\frac{1}{p(x_i)}\right) - \sum_{i=1}^{N} p(x_i) \log(N)$$

$$= \sum_{i=1}^{N} p(x_i) \left[\log\left(\frac{1}{p(x_i)}\right) - \log(N)\right]$$

$$= \sum_{i=1}^{N} p(x_i) \log\left(\frac{1}{Np(x_i)}\right)$$

Information (cont'd)

Maximizing Entropy (cont'd)

- Changing the base of the logarithm (using $\log(z) \equiv \log_2(z) = \log_2(e)\ln(z)$ where $\ln(z) \equiv \log_e(z)$) gives

$$H(x) - \log(N) = \log(e)\sum_{i=1}^{N} p(x_i)\ln\left(\frac{1}{Np(x_i)}\right).$$

- If all symbols are equally likely, $p(x_i) = 1/N$, then $\frac{1}{Np(x_i)} = 1$ and

$$\ln\left(\frac{1}{Np(x_i)}\right) = \ln(1) = 0$$

Hence

$$H(x) = \log(N)$$

as with $p(x_i) = 0.25$ for $i = 1, 2, 3, 4$ and $H(x) = -4(1/4)\log_2(1/4) = 2$ bits/symbol which matches the maximum of $\log_2(N)$ with $N = 4$.

Information (cont'd)

Maximizing Entropy (cont'd)

- Conversely, if the symbols are not equally likely, then the inequality $\ln(z) \leq z - 1$ (which holds for $z \geq 0$) implies that

$$H(x) - \log(N) < \log(e) \sum_{i=1}^{N} p(x_i) \left[\frac{1}{N p(x_i)} - 1 \right]$$

$$= \log(e) \left[\sum_{i=1}^{N} \frac{1}{N} - \sum_{i=1}^{N} p(x_i) \right] = \log(e)[1 - 1] = 0$$

- Rearranging yields the generic bound on the entropy

$$H(x) \leq \log(N)$$

- This analysis establishes that maximum entropy achieved by sources with equally probable symbols.

Information (cont'd)

Maximizing Efficiency

- Define efficiency as entropy divided by average number of bits per symbol used in code
- Example: $p(x_1) = 0.5$, $p(x_2) = 0.25$, $p(x_3) = p(x_4) = 0.125 \Rightarrow$ $H(x) = 0.5\log_2(0.5) + 0.25\log_2(0.25) + 2(0.125)\log_2(0.125) = 1.75$
- Code A:

$$x_1 \leftrightarrow 11,\ x_2 \leftrightarrow 10,\ x_3 \leftrightarrow 01,\ \text{and } x_4 \leftrightarrow 00$$

  average number of bits used in code $= 2 \Rightarrow$ efficiency $= 1.75/2 =$ 87.5%

Information (cont'd)

Maximizing Efficiency

- Code B:

$$x_1 \leftrightarrow 1,\ x_2 \leftrightarrow 01,\ x_3 \leftrightarrow 001,\ \text{and } x_4 \leftrightarrow 000$$

  average number of bits used in code
  $= 0.5(1) + 0.25(2) + 2(0.125)(3) = 1.75 \Rightarrow$ efficiency $= 1.75/1.75 =$ 100%, i.e. maximum possible.
- Thus, we seek a procedure that takes an independent source with unequally probable symbols and produces a maximally efficient code.

Huffman Coding

Procedure:

1. List the symbols in order of decreasing probability. These are the original "nodes".
2. Find the two nodes with the smallest probabilities, and combine them into one new node, with probability equal to the sum of the two. Connect the new nodes to the old ones with "branches" (lines).
3. Continue combining the pairs of nodes with the smallest probabilities. (If there are ties, pick any of the tied symbols).
4. Place a 0 or a 1 along each branch. The path from the rightmost node to the original symbol defines a binary list, which is the code word for that symbol.

Huffman Coding (cont'd)

Example:

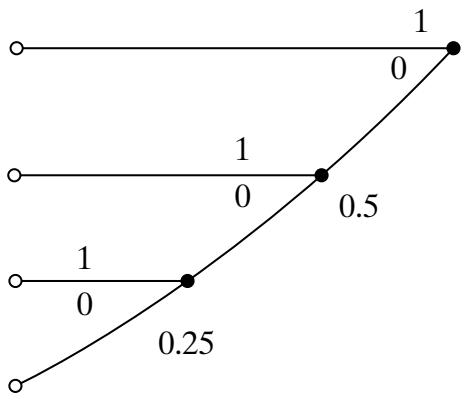
- Symbol probabilities: $p(x_1) = 0.5$, $p(x_2) = 0.25$, $p(x_3) = p(x_4) = 0.125$
- Graph construction:
  - x_3 and x_4 combined to form node with probability $= 0.25$ $(= p(x_3) + p(x_4))$
  - new node combined with x_2 to form a new node with probability 0.5
  - newest node combined with x_1 to form rightmost node
- Branch labelling: place a 1 on the top and a 0 on the bottom

Huffman Coding (cont'd)

Example (cont'd):

- Code tree:

Symbol	Probability
x_1	0.5
x_2	0.25
x_3	0.125
x_4	0.125



- Huffman code read from the right hand side:
  - x_1 corresponds to 1
  - x_2 corresponds 01
  - x_3 to 001
  - x_4 to 000

  This matches 100% efficient code B of previous example.

Huffman Coding (cont'd)

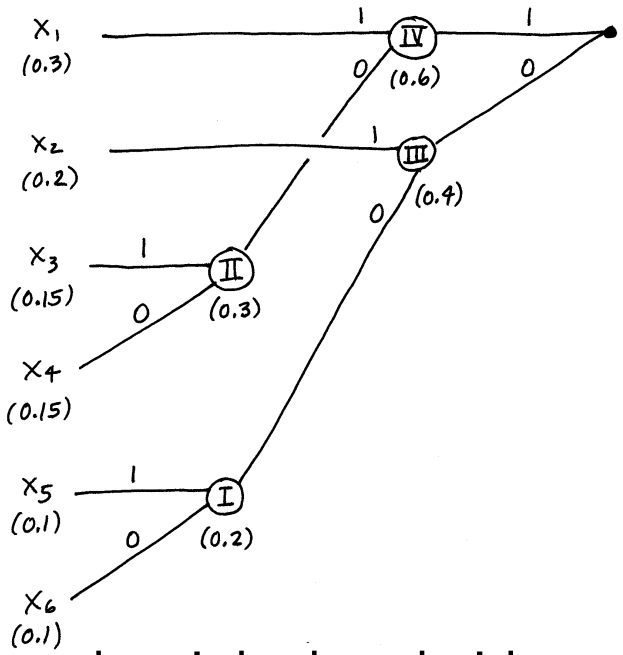
Another Example:

- Symbol probabilities: $p(x_1) = 0.3$, $p(x_2) = 0.2$, $p(x_3) = p(x_4) = 0.15$, $p(x_5) = p(x_6) = 0.1$
- Graph construction:
  - x_5 and x_6 combined to form node I with probability 0.2
  - x_3 and x_4 combined to form node II with probability 0.3
  - node I combined with x_2 to form node III with probability 0.4
  - node II combined with x_1 to form node IV with probability 0.6
  - nodes III and IV combined to form rightmost node
- Branch labelling: place a 1 on the top and a 0 on the bottom

Huffman Coding (cont'd)

Another Example (cont'd):

- Code tree:



- Huffman code read from the right hand side:
  - x_1 corresponds to 11
  - x_2 corresponds 01
  - x_3 to 101
  - x_4 to 100
  - x_5 to 001

Huffman Coding (cont'd)

Another Example (cont'd):

- Symbol ends if 2nd bit is 1; otherwise ends with 3 bits.
- Entropy:

$$H(x) = \sum_{i=1}^{6} p(x_i) \log_2 \left(\frac{1}{p(x_i)} \right)$$

$$= 0.3 \log_2 \left(\frac{1}{0.3} \right) + 0.2 \log_2 \left(\frac{1}{0.2} \right)$$

$$+2(0.15) \log_2 \left(\frac{1}{0.15} \right) + 2(0.1) \log_2 \left(\frac{1}{0.1} \right) = 2.471$$

- Average number of bits per symbol:

$$(0.3 + 0.2)(2) + (0.15 + 0.15 + 0.1 + 0.1)(3) = 2.5$$

- Efficiency (%):$100 \times \dfrac{\text{entropy}}{\text{average number bits per symbol}}$

$$= 100 \left(\frac{2.471}{2.5} \right) = 98.84\%$$

Huffman Coding (cont'd)

Observations:

- Huffman procedure (always) leads to a prefix code (with start and end immediately recognizable) because all the symbols end the same (except for the maximal length symbol x_4).
- Huffman procedure (always) leads to a code which has average length very near the optimal and efficiency near maximum.
- Logical branching can be used in decoding the variable length symbols of the Huffman code (see `codex`).

Having dealt with inefficiently coded messages that squander capacity, the first of our two problems with coding solutions, we now turn to the second problem of engendering resistance to bit errors due to noise.

Linear Block Coding

- With signal power kept fixed, an increase in number of levels in PAM signal requires that distance between signal levels shrinks.
- As the distance between signal levels shrinks, the ratio of powers in the total signal of the component due to signal $\mathcal{S}$ and that due to noise $\mathcal{P}$ must increase to maintain a maximum error rate.
- The maximum capacity in bits per second for a channel with additive white gaussian noise is (in bits/second)

$$\mathcal{C} = B \log_2 \left(1 + \frac{\mathcal{S}}{\mathcal{P}} \right)$$

  where B is the channel bandwidth.
- Increasing bandwidth or SNR increases capacity.
- We seek codes that fully utilize channel capacity.

Linear Block Coding (cont'd)

- For a source producing information at $\mathcal{R}$ bits/second, with $\mathcal{R} < \mathcal{C}$ there exists a code that can be transmitted with arbitrarily small error.
- A simple, but inefficient, coding technique offering resistance to bit errors in transmission is to repeat each bit three times. Thus, to communicate 01, we send 000111.
- The decoder at the receiver uses a "majority rules" strategy for each received triple. Thus,

$$\begin{array}{cccc} 000 \leftrightarrow 0 & 001 \leftrightarrow 0 & 010 \leftrightarrow 0 & 100 \leftrightarrow 0 \\ 101 \leftrightarrow 1 & 110 \leftrightarrow 1 & 011 \leftrightarrow 1 & 111 \leftrightarrow 1 \end{array}$$

- This table indicates that this strategy will successfully decode any triple with no more than one error.
- Linear block coding offers a more efficient error-correcting code.

Linear Block Coding (cont'd)

- The ability of English text to retain its meaning despite errors in several characters (up to $\sim 10\%$) is a testament to the error-correcting possibilities with a message possessing structural redundancy.
- The characters in English text occur with different probabilities

  (In $\mathrm{Wizard\ of\ Oz}$ the letter 'e' appears almost 10% of the time, while 'j' occurs less than 1/10 of 1% of the time.)

  and are not independent as characters influence adjacent characters.

  ('u' usually follows 'q'.)
- Linear block coding produces a much simpler structured redundancy than English language syntax.

Linear Block Coding (cont'd)

Linear block coding procedure:

1. Collect k symbols into a vector $\mathbf{x} = \{x_1, x_2, \ldots x_k\}$.
2. Transmit the length n code word $\mathbf{c} = \mathbf{x}G$.
3. At the receiver, the vector $\mathbf{y}$ is received. Calculate $\mathbf{y}H^T$.
4. If $\mathbf{y}H^T = 0$, then no errors have occurred.
5. When $\mathbf{y}H^T \neq 0$, errors have occurred. Look up $\mathbf{y}H^T$ in a table of "syndromes", which contains a list of all possible received values and the most likely symbol to have been transmitted, given the error that occurred.

Note: Arithmetic used throughout is binary, i.e. modulo 2, so 0+0=0, 0+1=1, 1+0=1, 1+1=0 and 0·0=0, 0·1=0, 1·0=0, 1·1=1.

Linear Block Coding (cont'd)

$(5, 2)$ *binary code*:

- Generator matrix

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

- Parity check matrix

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

Linear Block Coding (cont'd)

$(5, 2)$ *binary code (cont'd)*:

- This code bundles the bits into pairs, and the four corresponding code words are:

$$x_1 = 00 \leftrightarrow c_1 = x_1 G = 00000$$
$$x_2 = 01 \leftrightarrow c_2 = x_2 G = 01011$$
$$x_3 = 10 \leftrightarrow c_3 = x_3 G = 10101$$
$$x_4 = 11 \leftrightarrow c_4 = x_4 G = 11110$$

- Accompanying syndrome table

Syndrome $\mathbf{e}H^T$	Most likely error $\mathbf{e}$
000	00000
001	00001
010	00010
011	01000
100	00100
101	10000
110	11000
111	10010

Linear Block Coding (cont'd)

$(5, 2)$ *binary code (cont'd)*:

- For defined H

$$c_1 H^T = c_2 H^T = c_3 H^T = c_4 H^T$$

  so when received signal vector y is one of the code words $yH^T = 0$.
- When $yH^T \neq 0$, $y \neq c_i$ for any i.
- Define $e = y - c$ where c is the transmitted code word.
- Since $cH^T = 0$,

$$yH^T = (c + e)H^T = cH^T + eH^T = eH^T$$

- A nonzero eH^T is associated in the syndrome table with the most likely e which can be added to y to recover the most likely c.

Linear Block Coding (cont'd)

$(5, 2)$ *binary code (cont'd)*:

- For example, consider communication of symbol x_2 corresponding to 01, which is accomplished by transmitting the code word $c_2 = 01011$. An error in transmission produces $y = 11011$. $yH^T = eH^T = 101 \rightarrow$ most likely $e = 10000 \rightarrow e + y = 10000 + 11011 = 01011$, which correctly corresponds to c_2.
- As another example, again consider communication of symbol $x_2 = 01$ by transmitting the code word $c_2 = 01011$. Two errors in transmission produce $y = 00111$. $yH^T = eH^T = 111 \rightarrow$ most likely $e = 10010 \rightarrow e + y = 10010 + 00111 = 10101$, which incorrectly corresponds to c_3.

NEXT... We enter the integration layer and present a receiver design methodology and the specifications of a challenging design project.