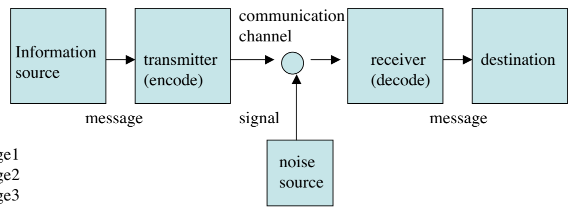
MAS.160 / MAS.510 / MAS.511 Signals, Systems and Information for Media Technology
Fall 2007

Information Theory



message1
message2
message3
…

OR

symbol1
symbol2
symbol3
…
AND
message1=symbol1, symbol2
message2=symbol3, symbol5

Information source selects a desired message from a set of possible messages
OR
selects a sequence of symbols from a set of symbols to represent a message.

Destination decides which message among set of (agreed) possible messages, the information source sent.

Transition Matrix

The first number has a 20% chance of being 1, 40% of being 2, and 40% of being 3

Starting from 1, the next number will be 1 (20%), 2(80%), 3 (0%)
Starting from 2, the next number will be 1 (40%), 2(0%), 3 (60%)
Starting from 3, the next number will be 1 (50%), 2(50%), 3 (0%)

To

$$\text{From}\quad \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0.4 & 0 & 0.6 \\ 0.5 & 0.5 & 0 \end{bmatrix} \end{array}$$

$$v = \begin{bmatrix} 0.2 & 0.4 & 0.4 \end{bmatrix}$$

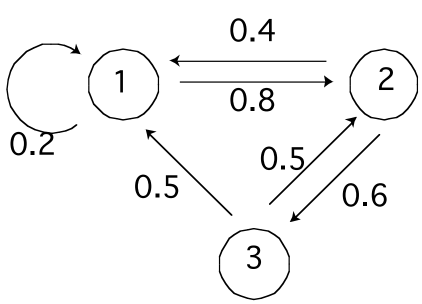
$$P = \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0.4 & 0 & 0.6 \\ 0.5 & 0.5 & 0 \end{bmatrix}$$

Discrete Markov Chain

To

$$\text{From}\quad \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0.4 & 0 & 0.6 \\ 0.5 & 0.5 & 0 \end{bmatrix} \end{array}$$

Arrows originating in State 1
Arrows originating in State 2
Arrows originating in State 3



Digram probabilities

What are the relative frequencies of the combination of symbols ij=11,12,13… (digram) after one time step?

$$p(i,j)=p(i)p_i(j)$$

i	p(i)
1	0.2
2	0.4
3	0.4

$p_i(j)$		j: 1	2	3
i	1	0.2	0.8	0
	2	0.4	0	0.6
	3	0.5	0.5	0

After one step

p(i,j)		j: 1	2	3
i	1	0.04	0.16	0
	2	0.16	0	0.24
	3	0.2	0.2	0

mymarkovonestep.m

11	4%
12	16%
13	0%
21	16%
22	0%
23	24%
31	20%
32	20%
33	0%

What about in steady state?

What is the probability distribution in the steady state?

$V_{ss}P = V_{ss}$

$v_x + v_y + v_z + \ldots = 1$

$v_{ss} = [v_x\ v_y\ v_z\ \ldots]$

n+1 equations
n unknowns

$V_{ss}P = V_{ss}$

$$\begin{bmatrix} v_x & v_y & v_z \end{bmatrix}\begin{bmatrix} 0.2 & 0.8 & 0 \\ 0.4 & 0 & 0.6 \\ 0.5 & 0.5 & 0 \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}$$

$$0.2v_x + 0.4v_y + 0.5v_z = v_x$$
$$0.8v_x + 0v_y + 0.5v_z = v_y$$
$$0v_x + 0.6v_y + 0v_z = v_z$$
$$v_x + v_y + v_z = 1$$

$$\begin{bmatrix} v_x & v_y & v_z \end{bmatrix} = \begin{bmatrix} 0.354 & 0.404 & 0.242 \end{bmatrix}$$

mymarkovss.m

1	35.4%
2	40.4%
3	24.2%

In the homework, the initial and steady state probability distributions are the same.

Steady State Digram probabilities

What are the steady state relative frequencies of the combination of symbols ij=11,12,13,… (digram)?

$$p(i,j) = p(i)p_i(j)$$

1	35.4%
2	40.4%
3	24.2%

i	p(i)
1	0.354
2	0.404
3	0.242

$p_i(j)$		j = 1	2	3
i	1	0.2	0.8	0
	2	0.4	0	0.6
	3	0.5	0.5	0

Step in steady state

p(i,j)		j = 1	2	3
i	1	0.0708	0.2832	0
	2	0.1616	0	0.2424
	3	0.121	0.121	0

mymarkov.m,
mymarkovss2.m

11	7%
12	28%
13	0%
21	16%
22	0%
23	24%
31	12%
32	12%
33	0%

H=-(0.708)log2(0.708)-(0.2832)*log2(0.2832)-0-(0.1616)log2(0.1616)- (0.2424)log2(0.2424)- (0.121)log2(0.121)
= 2.526 bits/(symbol combo)

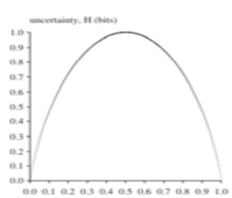
Entropy

Average information/symbol called Entropy H

$$H = -\sum_i p_i \log(p_i)$$

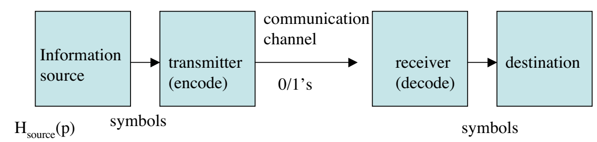
H(X,Y)=H(X)+H(Y) H also obeys additive property if events are independent.

For unfair coin, $p_H = p$, $p_T = (1-p)$



The average information per symbol is greatest when the symbols equiprobable.

Information Theory



H is the average number of bits/symbol to represent the information. If you encode using more bits/symbol, then the extra bits are redundant. Compression removes the redundancy. You can't compress more after the redundancy is gone; all you are left with is information.

The entropy gives us a lower limit on the number of bits per symbol we can achieve.

"Quinn's interpretation"
When sending straight binary code, if some letters are more common than others, 0's and 1's won't be equally probable in the (0/1) stream. Using encoding, we can try to make the (0/1) stream have equiprobable 0's and 1's.

Shannon-Fano coding splits the symbol frequency chart 50/50, then repeats for each branch. Each (0/1) stream will be answering "is the symbol you want to send in the upper branch or lower branch?" and there's a (close to) equiprobable chance it will be either.

So, the amount of information in the (0/1) stream is increased using the compression coding over the simple binary coding. Using more complicated compression coding, you can come closer to the (0/1) stream having equiprobable 0's and 1's.

Compression

Shannon Fano

ZIP implosion algorithm uses this

Split symbols so probabilities halved

Ex. “How much wood would a woodchuck chuck” 31 characters

ASCII 7bits/character, so 217 bits

Frequency chart

o	0.194
c	0.161
h	0.129
w	0.129
u	0.129
d	0.097
k	0.065
m	0.032
a	0.032
l	0.032

$$H = -\sum_i p_i \log(p_i)$$

H=-(0.194 $\log_2$0.194 + 0.161 $\log_2$0.161 +…)

H=2.706 bits/symbol, so 83.7 bits for sentence

o has $-\log_2 0.194=2.37$ bits of information

l has $-\log_2 0.032= 4.97$ bits of information

The rare letters carry more information

Compression

Shannon Fano

Split symbols so probabilities halved (or as close as possible)

Ex. “How much wood would a woodchuck chuck” 31 characters

Frequency chart

o	0.194
c	0.161
h	0.129
w	0.129
u	0.129
d	0.097
k	0.065
m	0.032
a	0.032
l	0.032

0.484 / 0.516

0.194 / 0.29

0.161 / 0.129

0.258 / 0.258

0.129 / 0.129

0.097 / 0.161

0.065 / 0.096

0.032 / 0.064

0.032 / 0.032

Compression

Shannon Fano

Split symbols so probabilities halved

Ex. “How much wood would a woodchuck chuck” 31 characters

Frequency chart

o	0.194
c	0.161
h	0.129
w	0.129
u	0.129
d	0.097
k	0.065
m	0.032
a	0.032
l	0.032

1 / 0

11 / 10

101 / 100

01 / 00

011 / 010

001 / 000

0001 / 0000

00001 / 00000

000001 / 000000

Compression

Shannon Fano

Split symbols so probabilities halved

Ex. “How much wood would a woodchuck chuck” 31 characters

“Prefix free - one code is never the start of another code”

Frequency chart

11	o	0.194
101	c	0.161
100	h	0.129
011	w	0.129
010	u	0.129
001	d	0.097
0001	k	0.065
00001	m	0.032
000001	a	0.032
000000	l	0.032

1 / 0

11 / 10

101 / 100

01 / 00

011 / 010

001 / 000

0001 / 0000

00001 / 00000

000001 / 000000

Compression

Shannon Fano — Split symbols so probabilities halved

Ex. “How much wood would a woodchuck chuck” 31 characters

Encoding chart		p	#
11	o	0.194	6
101	c	0.161	5
100	h	0.129	4
011	w	0.129	4
010	u	0.129	4
001	d	0.097	3
0001	k	0.065	2
00001	m	0.032	1
000001	a	0.032	1
000000	l	0.032	1

6(2)+5(3)+4(3)+4(3)+4(3)+
3(3)+2(4)+1(5)+1(6)+1(6)
=97 bits

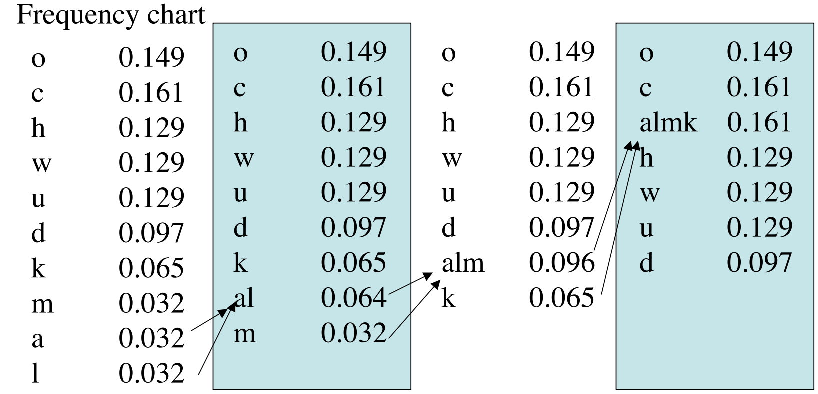
97bits/31 characters
=3.129 bits/character

H=2.706 bits/symbol

Huffman Coding

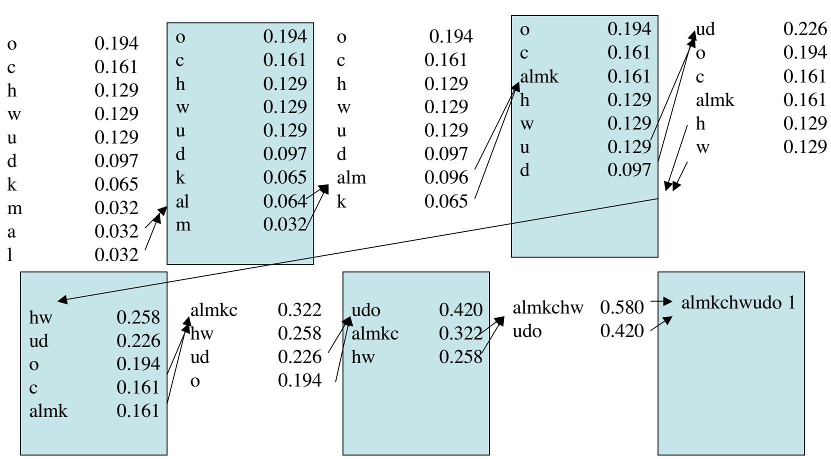
Add two lowest probabilities
group symbols
Resort
Repeat

Ex. “How much wood would a woodchuck chuck”



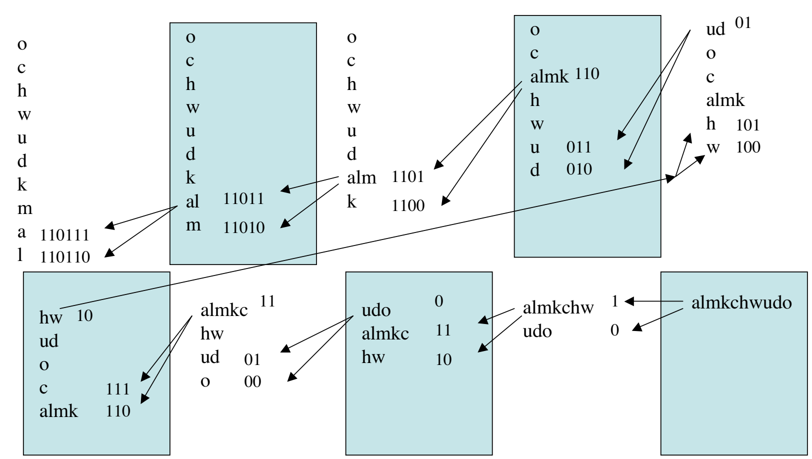
Huffman Coding

Ex. “How much wood would a woodchuck chuck”



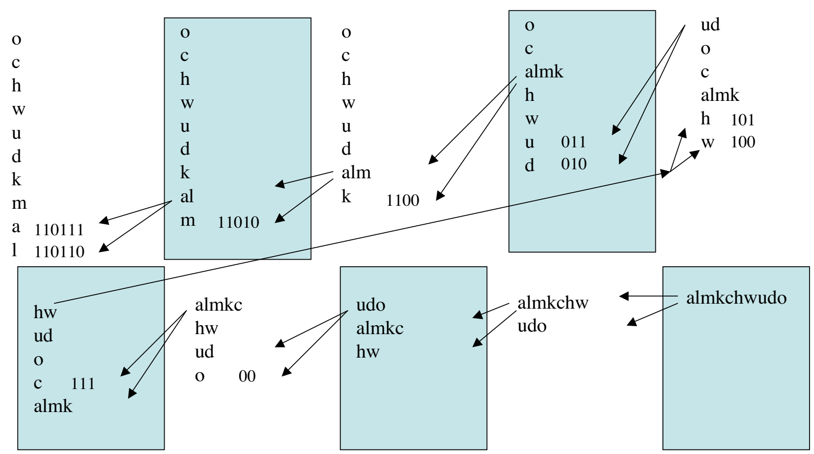
Huffman Coding

Ex. “How much wood would a woodchuck chuck”



Huffman Coding

Ex. “How much wood would a woodchuck chuck”



Huffman Coding

Ex. “How much wood would a woodchuck chuck”

Huffman

o	00	6
c	111	5
h	100	4
w	101	4
u	011	4
d	010	3
k	1100	2
m	11010	1
a	110111	1
l	110110	1

6(2)+5(3)+4(3)+4(3)+4(3)+
3(3)+2(4)+1(5)+1(6)+1(6)
=97 bits

97bits/31 characters
=3.129 bits/character

Shanon Fano

11	o
101	c
100	h
011	w
010	u
001	d
0001	k
00001	m
000001	a
000000	l

97bits/31 characters
=3.129 bits/character

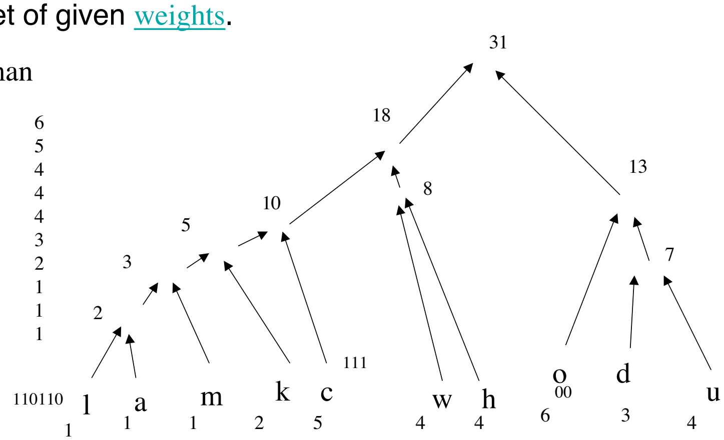
Notice o has 2 bits;
a,l have 6 bits

H=2.706 bits/symbol

Huffman's algorithm is a method for building an extended binary tree of with a minimum weighted path length from a set of given weights.

Huffman

o	00	6
c	111	5
h	100	4
w	101	4
u	011	4
d	010	3
k	1100	2
m	11010	1
a	110111	1
l	110110	1

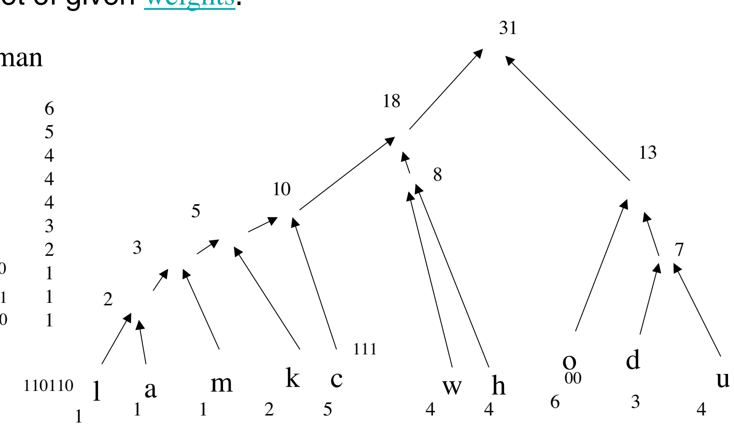


Frequencies*(edges to root)= weighted path length

Huffman's algorithm is a method for building an extended binary tree of with a minimum weighted path length from a set of given weights.

Huffman

o	00	6
c	111	5
h	100	4
w	101	4
u	011	4
d	010	3
k	1100	2
m	11010	1
a	110111	1
l	110110	1

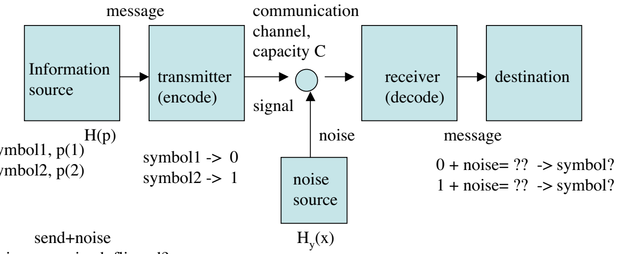


Each branch adds a bit. Minimize (#branches * frequency)
Least frequent symbol further away. More frequency, closer.

Huffman

Huffman coding is used in the final step of creating an MP3 file. The MP3 format uses frames of 1152 sample values. If the sample rate is 44.1kHz, the time that each frame represents is ~26ms. The spectrum of this 1152-sample frame is spectrally analyzed and the frequencies are grouped in 32 channels (critical bands). The masking effects within a band are analyzed based on a psycho-acoustical model. This model determines the tone-like or noise-like nature of the masking in each channel and then decides the effect of each channel on its neighboring bands. The masking information for all of the channels in the frame is recombined into a time varying signal. This signal is numerically different from the original signal but the difference is hardly noticeable aurally. The signal for the frame is then Huffman coded. A sequence of these frames makes up an MP3 file.
http://webphysics.davidson.edu/faculty/dmb/py115/huffman_coding.htm

Information Theory



symbol1, p(1)
symbol2, p(2)

symbol1 -> 0
symbol2 -> 1

0 + noise= ?? -> symbol?
1 + noise= ?? -> symbol?

sent	noise	send+noise received	flipped?
0	0	0	no
1	0	1	no
0	1	1	yes
1	1	0	yes

0 -> no flipped bit p('not flip')
1 -> flipped bit p('flip') = 1 - p('not flip')

$H_y(x)$ max when
p('not flip') = p('flip')=1/2

$H_y(x)$ min when
p('not flip') =1, p('flip')=0
OR
p('not flip') =0, p('flip')=1

equivocation: 'confusion in received signal' or 'spurious information in noise'
entropy of noise source.

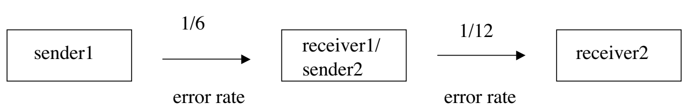
$H_y(x)$ = - p('not flip') log2(p('not flip')) - p('flip') log2(p('flip'))

R= H(x) - $H_y(x)$ Actual info transmission rate is the info source entropy minus the noise source entropy

C = max{R} over all possible information sources
C >H(x) then you can encode to get ε errors (error correction codes -> use redundancy)

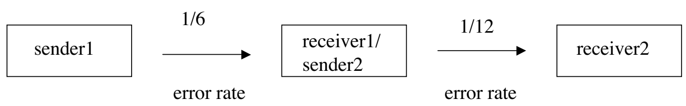
Noisy Channel

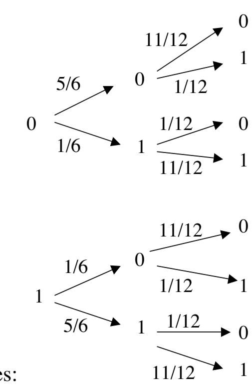
A binary communication system contains a pair of error-prone wireless channels, as shown below.



Assume that in each channel it is equally likely that a 0 will be turned into a 1 or that a 1 into a 0. Assume also that in the first channel the probability of an error in any particular bit is 1/6, and in the second channel it is 1/12.

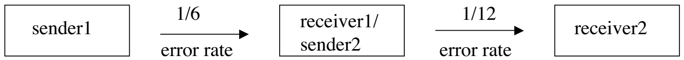
Compute the four probabilities:
- 0 sent 0 received
- 0 sent 1 received
- 1 sent 0 received
- 1 sent 1 received

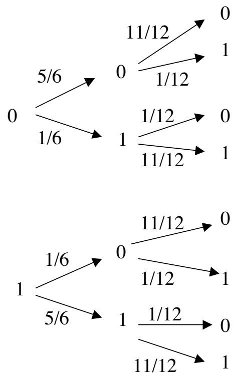




Channel symmetric
So p(0->0)= p(1->1), etc.

Compute the four probabilities:
- 0 sent 0 received: 000, 010
- 0 sent 1 received: 001, 011
- 1 sent 0 received: 100, 110
- 1 sent 1 received: 101, 111





p(000) = p1(0->0)*p2(0->0)
p(001) = p1(0->0)*p2(0->1)

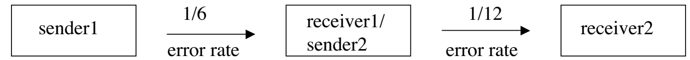
p(010) = p1(0->1)*p2(1->0)
p(011) = p1(0->1)*p2(1->1)

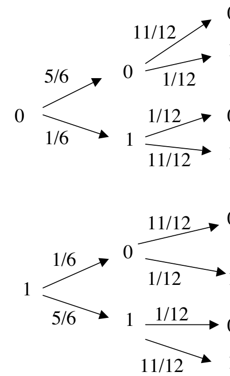
Channel symmetric
So p(0->0)= p(1->1), etc.

p(100) = p1(1->0)*p2(0->0)
p(101) = p1(1->0)*p2(0->1)

p(110) = p1(1->1)*p2(1->0)
p(111) = p1(1->1)*p2(1->1)

Compute the four probabilities:
- 0 sent 0 received: p(000)+p(010)
- 0 sent 1 received: p(001)+p(011)
- 1 sent 0 received: p(100)+p(110)
- 1 sent 1 received: p(101)+p(111)





p(000): (5/6)*(11/12)= 55/72
p(001): (5/6)*(1/12) = 5/72

p(010): (1/6)*(1/12)= 1/72
p(011): (1/6)*(11/12) = 11/72

Channel symmetric
So p(0->0)= p(1->1), etc.

p(100): (1/6)*(11/12)= 11/72
p(101): (1/6)*(1/12) = 1/72

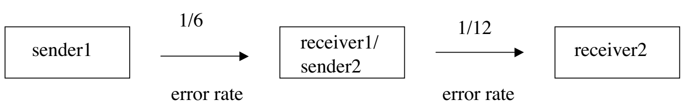
p(110): (5/6)*(1/12)= 5/72
p(111): (5/6)*(11/12) = 55/72

Compute the four probabilities:
- 0 sent 0 received: p(000)+p(010)= 55/72 +1/72= 56/72 ≈ 0.778
- 0 sent 1 received: p(001)+p(011)= 5/72 + 11/72 = 16/72≈ 0.222
- 1 sent 0 received: p(100)+p(110)= 11/72 + 5/72 = 16/72≈ 0.222
- 1 sent 1 received: p(101)+p(111)=1/72 + 55/72 = 56/72≈ 0.778

myflipsim.m

Error Correction: Repeat Code

A binary communication system contains a pair of error-prone wireless channels, as shown below.

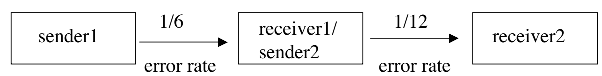


Repeat code: a 0 is transmitted as three successive 0's and a 1 as three successive 1's. At the decoder, a majority decision rule is used: if a group of three bits has more 0's than 1's (e.g. 000, 001, 010, 100), it's assumed that a 0 was meant, and if more 1's than 0's that a 1 was meant.

If the original source message has an equal likelihood of 1's and 0's, what is the probability that a decoded bit will be incorrect?

Error Correction: Repeat Code

A binary communication system contains a pair of error-prone wireless channels, as shown below.



- 0 sent 0 received: ≈ 0.778
- 0 sent 1 received: ≈ 0.222
- 1 sent 0 received: ≈ 0.222
- 1 sent 1 received: ≈ 0.778

Repeat code: a 0 is transmitted as three successive 0's and a 1 as three successive 1's. At the decoder, a majority decision rule is used. What is the probability that a decoded bit will be incorrect?

send	sender1	receiver2	decision	bits flipped	p
		000	0	0	
		001	0	1	
		010	0	1	
0	000	011	1	2	
		100	0	1	
		101	1	2	
		110	1	2	
		111	1	3	

0 bits flipped
0.778* 0.778 * 0.778 =0.471

1 bits flipped
0.222*0. 778 *0. 778 =0.134

2 bits flipped
0.222 * 0.222 *0. 778 = 0.038

3 bits flipped
0.222 * 0.222 * 0.222 = 0.011

myflipsim2.m

Error Correction: Repeat Code

A binary communication system contains a pair of error-prone wireless channels, as shown below.

$$\boxed{\text{sender1}} \xrightarrow[\text{error rate}]{1/6} \boxed{\text{receiver1/ sender2}} \xrightarrow[\text{error rate}]{1/12} \boxed{\text{receiver2}}$$

- 0 sent 0 received: ≈ 0.778
- 0 sent 1 received: ≈ 0.222
- 1 sent 0 received: ≈ 0.222
- 1 sent 1 received: ≈ 0.778

Repeat code: a 0 is transmitted as three successive 0's and a 1 as three successive 1's. At the decoder, a majority decision rule is used. What is the probability that a decoded bit will be incorrect?

send	sender1	receiver2	decision	bits flipped	p
		000	0	0	0.471
		001	0	1	0.134
		010	0	1	0.134
0	000	011	1	2	0.0308
		100	0	1	0.134
		101	1	2	0.0308
		110	1	2	0.0308
		111	1	3	0.011

decoded bit incorrect

011, 101, 110, 111

0.0308+0.0308+0.0308+0.011=0.126

12.6% chance decoded bit will be incorrect

Check using Matlab simulation

Compression

You are given a data file that has been compressed to a length of 100,000 bits, and told that it is result of running an "ideal" entropy coder on a sequence of data. You are also told that the original data are samples of a continuous waveform, quantized to two bits per sample. The probabilities of the uncompressed values are

s	p(s)	s	p(s)
00	1/2	10	1/16
01	3/8	11	1/16

What (approximately) was the length of the uncompressed file, in bits?

compressed file

$H=-(0.5 \log_2 0.5) - \ldots = h$ bits/sample

100,000 bits

uncompressed file

2 bits/sample

x bits

$$\frac{x \;\; bits}{100000 \;\; bits} = \frac{2 \;\; bits/sample}{h \;\; bits/sample}$$

Compression

The number of (two-bit) samples in the uncompressed file is half the value you computed in part a). You are told that the continuous waveform was sampled at the minimum possible rate such that the waveform could be reconstructed exactly from the samples (at least before they were quantized), and you are told that the file represents 10 seconds of data. What is the highest frequency present in the continuous signal?

uncompressed file

2 bits/sample

x bits

s	p(s)	s	p(s)
00	1/2	10	1/16
01	3/8	11	1/16

Compute number of samples

y sample = x bits / (2 bits/sample)

Compute sampling rate

sampling rate = y samples / t seconds

Use Shannon Sampling Theorem

Max frequency (Nyquist rate) = sampling rate/2

Error Correction: Hamming Code (7,4)

$(2^m-1, 2^m-m-1)$, m=3

in

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$\vec{a}$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \boxed{\begin{matrix} a_4 \\ a_5 \\ a_6 \end{matrix}} \end{bmatrix}$$

H

$$\begin{bmatrix} 0001111 \\ 0110011 \\ 1010101 \end{bmatrix}$$

columns are binary digits from 1 to 2^m-1

H tests if $\vec{a}$ is a valid code

valid codes satisfy

$$H\vec{a} = 0$$

$$\begin{bmatrix} 0001111 \\ 0110011 \\ 1010101 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = 0$$

a3+a4+a5+a6=0
a2+a3+a5+a6=0
a0+a2+a4+a6=0

solve system of equations for remaining bits a4, a5, a6 to make valid code

use modulo-2 math

Error Correction: Hamming Code (7,4)

$$\begin{array}{cc} \text{in} & \vec{a} \\ \begin{bmatrix}a_2\\a_4\\a_5\\a_6\end{bmatrix} & \begin{bmatrix} a_0 = a_2 + a_4 + a_6 \\ a_1 = a_2 + a_5 + a_6 \\ a_2 \\ a_3 = a_4 + a_5 + a_6 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \end{array}$$

easier to solve for remaining bits to make a valid code if using this layout.

valid codes satisfy

$$H\vec{a} = 0$$

$$\begin{bmatrix}0001111\\0110011\\1010101\end{bmatrix}\begin{bmatrix}a_0\\a_1\\a_2\\a_3\\a_4\\a_5\\a_6\end{bmatrix} = 0$$

a3=a4+a5+a6
a1=a2+a5+a6
a0=a2+a4+a6

$\leftarrow$

a3+a4+a5+a6=0
a1+a2+a5+a6=0
a0+a2+a4+a6=0

use
modulo-2 math

Error Correction: Hamming Code (7,4)

$$\begin{array}{cc} \text{in} & \vec{a} \\ \begin{bmatrix}1\\0\\1\\0\end{bmatrix} & \begin{bmatrix}a_0\\a_1\\1\\a_3\\0\\1\\0\end{bmatrix} \end{array}$$

valid code if

a3=a4+a5+a6
1=0+1+0
a1=a2+a5+a6
0=1+1+0
a0=a2+a4+a6
1=1+0+0

make $\vec{a}$ a valid code

$$\begin{array}{ccc} \vec{a}_{send} & & \vec{a}_{r'cvd} \\ \begin{bmatrix}1\\0\\1\\1\\0\\1\\0\end{bmatrix} & \xrightarrow{\text{noisy}} & \begin{bmatrix}1\\0\\\boxed{0}\\1\\0\\1\\0\end{bmatrix} \\ \text{sender} & & \text{receiver} \end{array}$$

Error Correction: Hamming Code (7,4)

$$\begin{array}{ccc} \vec{a}_{send} & & \vec{a}_{r'cvd} \\ \begin{bmatrix}1\\0\\1\\1\\0\\1\\0\end{bmatrix} & \xrightarrow{\text{noisy}} & \begin{bmatrix}1\\0\\\boxed{0}\\1\\0\\1\\0\end{bmatrix} \\ \text{sender} & & \text{receiver} \end{array}$$

$$H\vec{a} \neq 0$$

so there is an error

$$\begin{bmatrix}0001111\\0110011\\1010101\end{bmatrix}\begin{bmatrix}1\\0\\0\\1\\0\\1\\0\end{bmatrix} = \begin{bmatrix}2\\1\\1\end{bmatrix} = \begin{bmatrix}0\\1\\1\end{bmatrix}$$

mod2

same as 3rd column, therefore error in the 3rd position

Hamming (7,4) can correct for 1 bit errors

out

$$\begin{array}{c} a2\\a4\\a5\\a6 \end{array} = \begin{bmatrix}1\\0\\1\\0\end{bmatrix}$$

same as in. Bit flip corrected