MAS160: Signals, Systems & Information for Media Technology

Problem Set 4

Instructor : V. Michael Bove, Jr.

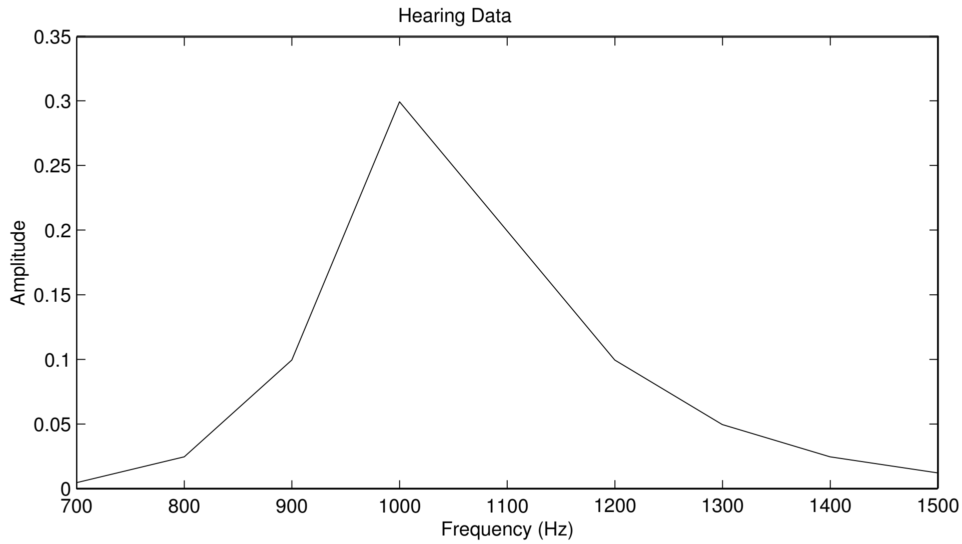
Problem 1: Simple Psychoacoustic Masking

The following MATLAB function performs a simple psychoacoustic test. It creates bandlimited noise, centered at 1000 Hz and also creates a sinusoid. It then plays the noise alone and then the noise plus the sinusoid. Try different values of f and A to see whether you can detect the sinusoid. For a particular value of f we'll call $A_{\min}(f)$ the minimum amplitude at which the frequency f sinusoid could still be heard. Plot several values on the graph of f vs. $A_{\min}$ to determine a simple masking curve.

```
function mask(f,A)
% MASK Performs a simple psychoacoustic masking test by creating
%      bandlimited noise around 1000 Hz and a single sinusoid at
%      frequency f with amplitude A. It then plays the noise
%      alone, and then the noise plus the sinusoid.
%
%       f - frequency of sinusoid (0 to 11025)
%       A - amplitude of sinusoid (0 to 1)

% Set sampling rate to 22050 Hz
fs = 22050;

% Create a bandpass filter, centered around 1000 Hz.  Since the
% sampling rate is 22050, the Nyquist frequency is 11025.
% 1000/11025 is approximately 0.09, hence the freqeuency
% values of 0.08 and 0.1 below.  For more info, do 'help butter'.
[b,a] = butter(4,[0.08 0.1]);

% Create a vector of random white noise (equal in all frequencies)
wn = rand(1,22050);

% Filter the white noise with our filter
wf = filter(b,a,wn);
% By filtering, we've reduced the power in the noise, so we normalize:
wf = wf/max(abs(wf));

% Create the sinusoid at frequency f, with amplitude A:
s = A*cos(2*pi*f/fs*[0:fs-1]);

% Play the sounds
sound(wf,22050)
pause(1)            % Pause for one second between sounds
sound(wf+s,22050)
```

SOLUTION :

Here is the plot generated from my results:



This is pretty much what you'd expect. Since the noise is centered around 1000 Hz, frequencies close to 1000 Hz will be more easily masked, and will require greater amplitudes for you to hear them. As you move further away from 1000 Hz, less amplitude is required to hear the sound.

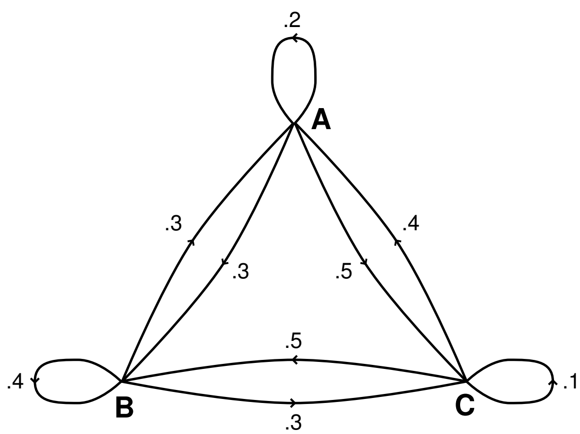
Problem 2: Markoff processes, entropy, and grading ;

A particularly lazy teaching assistant is faced with the task of assigning student grades. In assigning the first grade, he decides that the student has a 30% chance of getting an A, a 40% chance of getting a B, and a 30% chance of getting a C (he doesn't give grades other than A, B, or C). However, as he continues to grade, he is affected by the grade he has just given. If the grade he just gave was an A, he starts to feel stingy and there is less chance he will give a good grade to the next student. If he gives a C, he starts to feel guilty and will tend to give the next student a better grade. Here is how he is likely to grade given the previous grade:

If he just gave an A, the next grade will be: A (20% of the time), B (30%), C (50%).
If he just gave a B, the next grade will be: A (30%), B (40%), C(30%).
If he just gave a C, the next grade will be: A (40%), B (50%), C(10%).

(a) Draw a Markoff graph of this unusual grading process.
(b) Calculate the joint probability of all successive *pairs* of grades (i.e. AA, AB, AC, etc.)
(c) Calculate the entropy, H, of two successive grades given.

SOLUTION :



(a)

(b) $p(i)$ *is the initial probability of grade* i, $p_i(j)$ *is the probability grade* i *is followed by grade* j, *and* $p(i,j)$ *is the probability of successive grades* i *and* j.

$p_i(j)$		j: A	B	C
i	A	.2	.3	.5
	B	.3	.4	.3
	C	.4	.5	.1

i	$p(i)$
A	.3
B	.4
C	.3

$p(i,j)$		j: A	B	C
i	A	.06	.09	.15
	B	.12	.16	.12
	C	.12	.15	.03

(c) *The total entropy of two successive grades is simply the summation of the entropies of each successive grade pair:*

$$\begin{aligned}
H &= -\sum_{i,j} p(i,j)\log_2 p(i,j) \\
&= -p(A,A)\log_2 p(A,A) - p(A,B)\log_2 p(A,B) - p(A,C)\log_2 p(A,C) \\
&\quad - p(B,A)\log_2 p(B,A) - p(B,B)\log_2 p(B,B) - p(B,C)\log_2 p(B,C) \\
&\qquad - p(C,A)\log_2 p(C,A) - p(C,B)\log_2 p(C,B) - p(C,C)\log_2 p(C,C) \\
&= 3.0533
\end{aligned}$$

Problem 3: Entropy Coding

Often it is the case that a set of symbols we want to transmit are not equally likely to occur. If we know the probabilities, then it makes sense to represent the most common symbols with shorter bit strings, rather than using an equal number of binary digits for all symbols. This is the principle behind variable-length coders.

An easy-to-understand variable-length coder is the Shannon-Fano code. The way we make a Shannon-Fano code is to arrange all the symbols in decreasing order of probability, then to split them into two groups with approximately equal probability totals (as best we can, given the probabilities we have to work with), assigning 0 as an initial code digit to the entries in the first group and 1 to those in the second. Then, keeping the symbols in the same order, we recursively apply the same algorithm to the two groups till we've run out of places to divide. The pattern of ones and zeros then becomes the code for each symbol. For example, suppose we have an alphabet of six symbols:

Symbol	**% probability**	**Binary code**	**Shannon-Fano code**
A	25	000	00
B	25	001	01
C	25	010	10
D	12.5	011	110
E	6.25	100	1110
F	6.25	101	1111

Let's see how much of a savings this method gives us. If we want to send a hundred of these symbols, ordinary binary code will require us to send 100 times 3 bits, or 300 bits. In the S-F case, 75 percent of the symbols will be transmitted as 2-bit codes, 12.5 as 3-bit codes, and 12.5 as 4-bit codes, so the total is only 237.5 bits, on average. Thus the binary code requires 3 bits per symbol, while the S-F code takes 2.375.

The entropy, or "information content" expression gives us a lower limit on the number of bits per symbol we might achieve.

$$H = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$= -[0.25\log_2(0.25) + 0.25\log_2(0.25) + 0.25\log_2(0.25) + 0.125\log_2(0.125)$$
$$+0.0625\log_2(0.0625) + 0.0625\log_2(0.0625)]$$

If your calculator doesn't do base-two logs (most don't), you'll need the following high-school relation that many people forget:

$$\log_a(x) = \log_{10}(x)/\log_{10}(a),$$

so

$$\log_2(x) = \log_{10}(x)/0.30103.$$

And the entropy works out to 2.375 bits/symbol. So we've achieved the theoretical rate this time. The S-F coder doesn't always do this well, and more complex methods like the Huffman coder will work better in those cases (but are too time-consuming to assign on a problem set!).

Now it's your turn to do some coding. The below is a letter-frequency table for the English language (also available at `http://ssi.www.media.mit.edu/courses/ssi/y03/ps4.freq.txt`).

E	13.105	T	10.468	A	8.151	O	7.995
N	7.098	R	6.832	I	6.345	S	6.101
H	5.259	D	3.788	L	3.389	F	2.924
C	2.758	M	2.536	U	2.459	G	1.994
Y	1.982	P	1.982	W	1.539	B	1.440
V	0.919	K	0.420	X	0.166	J	0.132
Q	0.121	Z	0.077				

(a) Twenty-six letters require five bits of binary. What's the entropy in bits/letter of English text coded as individual letters, ignoring (for simplicity) capitalization, spaces, and punctuation?

(b) Write a Shannon-Fano code for English letters. How many bits/letter does your code require?

(c) Ignoring (as above) case, spaces, and punctuation, how many total bits does it take to send the following English message as binary? As your code? [You don't need to write out the coded message, just add up the bits.]

"There is too much signals and systems homework"

(d) Repeat (c) for the following Clackamas-Chinook sentence (forgive our lack of the necessary Native American diacritical marks!).

"nugwagimx lga dayaxbt, aga danmax wilxba diqelpxix."

SOLUTION :

(a)

$$H = -\sum_{i=1}^{26} p_i \; log_2(p_i) = 4.131 \;\; bits/symbol$$

(b) Because of the probability distribution, one can't always split the table into exactly equal halves. Depending on whether one chooses to put the "extra" probability above or below the split, there are several slightly different answers. Here's one possibility:

Letter	**% prob.**	**Shannon-Fano code**
E	*13.105*	*000*
T	*10.468*	*001*
A	*8.151*	*010*
O	*7.995*	*0110*
N	*7.098*	*0111*
R	*6.832*	*1000*
I	*6.345*	*1001*
S	*6.101*	*1010*
H	*5.259*	*1011*
D	*3.788*	*11000*
L	*3.389*	*11001*
F	*2.924*	*11010*
C	*2.758*	*11011*
M	*2.536*	*11100*
U	*2.459*	*111010*
G	*1.994*	*111011*
Y	*1.982*	*111100*
P	*1.982*	*111101*
W	*1.539*	*1111100*
B	*1.440*	*1111101*
V	*0.919*	*1111110*
K	*0.420*	*11111110*
X	*0.166*	*111111110*
J	*0.132*	*1111111110*
Q	*0.121*	*11111111110*
Z	*0.077*	*11111111111*

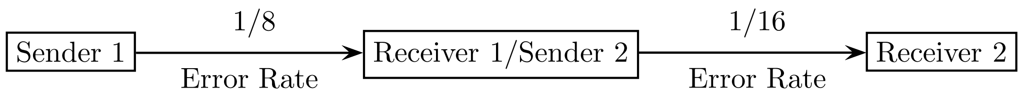
For the above code, the rate is just under 4.168 bits/symbol. Not perfect, but not too far from the theoretical limit. Your answer may differ slightly depending on how you split the table.

(c) Normal binary uses $39 \times 5 = 195$ *bits, the above code requires 166 (again, your mileage may vary slightly). The message given for coding only partially reflect the statistics of the English language, hence our coder's relatively poor performance.*

(d) *Normal binary requires 215 bits, the above code requires 233, illustrating the problem that occurs when the message doesn't match the statistics used in formulating the variable-length code!*

Problem 4: Error Correction

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 1/8, and in the second channel it is 1/16.

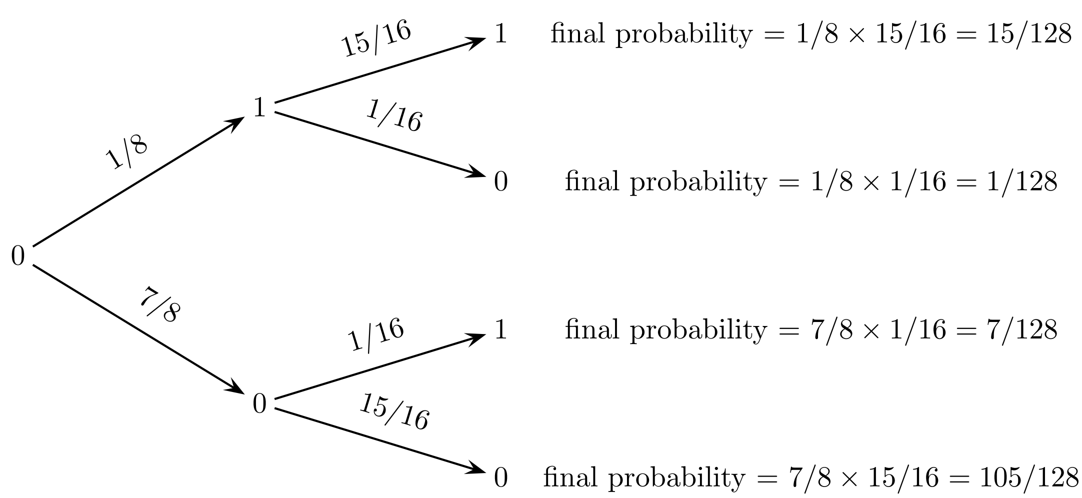
(a) For the combined pair of channels, compute the following four probabilities:

- a 0 is received when a 0 is transmitted,
- a 0 is received when a 1 is transmitted,
- a 1 is received when a 1 is transmitted,
- a 1 is received when a 0 is transmitted.

(b) Assume that a very simple encoding scheme is used: 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?

SOLUTION :

(a) Given the above information we can determine the bit flip probability by looking at the following tree



From this you can see that the total probability that a 0 becomes a 1 is $15/128 + 7/128 = 22/128 = 11/64$. *The probability that the 0 is not flipped than must be* $53/64$, *which checks out since* $106/128 = 53/64$. *In this channel 0 and 1 are treated symmetrically so*

- $p(0 \to 0) = p(1 \to 1) = 53/64 \approx 0.8281$
- $p(0 \to 1) = p(1 \to 0) = 11/64 \approx 0.1719$

(b) *If you use the encoding scheme described in the problem 000 will be sent in the place of 0. If majority determines the bit identity are receiver 2 then the following sequences will be counted as a 0: 000,001,010, and 100.*

- $p(000 \to 000) = 53/64 \times 53/64 \times 53/64 = 148877/262144$
- $p(000 \to 001) = 53/64 \times 53/64 \times 11/64 = 30899/262144$
- $p(000 \to 010) = 53/64 \times 11/64 \times 53/64 = 30899/262144$
- $p(000 \to 100) = 11/64 \times 53/64 \times 53/64 = 30899/262144$

So given the error correction, our new $p(0 \to 0) = 241574/262144 = 120787/131072 \approx 0.9215$. *And* $p(0 \to 1) = 10285/131072 \approx 0.0785$. *Notice that our error rate has gone down, but not by a factor of 3 as you might have hoped.*

Problem 5: Data 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	$p(s)$
00	1/2
01	3/8

s	$p(s)$
10	1/16
11	1/16

(a) What (approximately) was the length of the uncompressed file, in bits? (You may not need to design a coder to answer this question!)

(b) 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?

SOLUTION :

(a) *If the file is ideally compressed then every bit in the file provides a bit of* information. *If you calculate the information content of the original alphabet*

$$H = -\sum_{i=1}^{4} p_i \log_2 p_i = 1.5306$$

So the original alphabet contained only 1.5306 bits/symbol. The length of the uncompressed then must have been

$$100,000 \text{ bits } \times \frac{1 \text{ symbol}}{1.5306 \text{ bits}} \times \frac{2 \text{ bit}}{1 \text{ symbol}} \approx 130660 \text{ bits}$$

(b) *If the original file contained 65332 samples (2 bits per sample), and the samples were taken over 10 seconds then 6533.2 samples were taken per second. The maximum frequency that can be reconstructed is half this or 3266.6 Hz.*