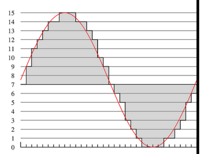
Penn Engineering

ESE



Lecture #6 – Compression

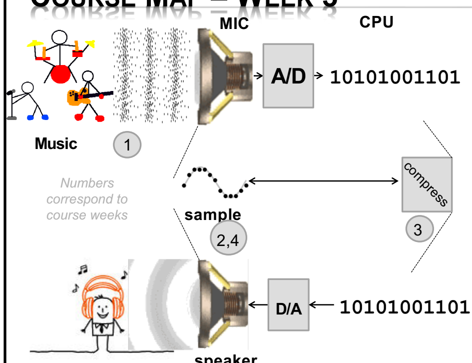
ESE 150 – Digital Audio Basics

Based on slides © 2009--2021 DeHon
Additional Material © 2014 Farmer

Lecture Topics

- Where are we on course map?
- Compression: Lossy and Lossless
- Lossless Compression
  - Probability-based lossless compression
    - Huffman Encoding
- Part 2:
  - Common case
  - Entropy
    - Shannon Limits
- Next Lab
- References

Course Map – Week 3



Preclass

Preclass

- **Tell me and I forget, teach me and I may remember, involve me and I learn**
  - -- Benjamin Franklin
- **73 symbols** (fancy, more general term for letters)
- **19 unique (ignoring case)**
  - (A, B, C, D, E, F, G, H, I, L, M, N, O, R, T, V, Y, space, comma)
  - How many bits to represent each symbol?
- **How many bits to encode quote?**

Preclass

- **Tell me and I forget, teach me and I may remember, involve me and I learn**
  - -- Benjamin Franklin
- **73 symbols**
- **19 unique (ignoring case)**
- **If symbols occurrence equally likely, how many occurrences of each symbol should we expect in quote?**
- **How many e's are there in the quote?**

Preclass

- Tell me and I forget, teach me and I may remember, involve me and I learn
  - -- Benjamin Franklin
- 73 symbols
- 19 unique (ignoring case)
- Conclude
  - Symbols do not occur equally
  - Symbol occurrence is not uniformly random

8

Preclass

- Tell me and I forget, teach me and I may remember, involve me and I learn
  - -- Benjamin Franklin
- Using uniform encoding (from question 1)
  - How many bits to encode first 24 symbols?
- How many bits using encoding given (Q5a)?

$$TotalBits = \sum_{1}^{24} bits[quote[i]]$$

9

Preclass

- Tell me and I forget, teach me and I may remember, involve me and I learn
  - -- Benjamin Franklin
- Using uniform encoding (question 1)
  - How many bits to encode all 73 symbols?
- How many bits using encoding given (Q5c)?

$$TotalBits = \sum_{1}^{73} bits[quote[i]]$$

10

Conclude

- Can encode with (on average) fewer bits than $\log_2$(unique-symbols)

11

Intro to Compression

12

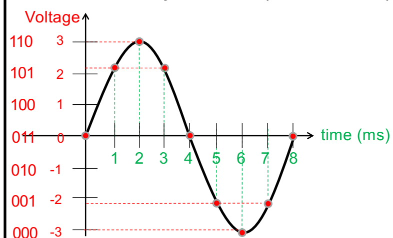
Data Compression

- What is compression?
  - Encoding information using fewer bits than the original representation
- Why do we need compression?
  - Most digital data is not sampled/quantized/represented in the most compact form
    - It takes up more space on a hard drive/memory
    - It takes longer to transmit over a network
  - Why? Because data is represented so that it is easiest to use
- Two broad categories of compression algorithms:
  - **Lossless** – when data is un-compressed, data is its original form
    - No data is lost or distorted
  - **Lossy** – when data is un-compressed, data is in **approximate** form
    - Some of the original data is lost

13

Recall ADC Process?

- Analog-to-Digital (ADC) Conversion
  - We have 7 discrete voltages, # of bits to represent 7 things?
  - 3-bits! Why? $2^{3\text{-bits}} = 8$ (1 unused state)



Samples:	Binary Encoding:
{ 0 ms, 0 Volts } →	011
{ 1 ms, 2 Volts } →	101
{ 2 ms, 3 Volts } →	110
{ 3 ms, 2 Volts } →	101
{ 4 ms, 0 Volts } →	011
{ 5 ms, -2 Volts } →	001
{ 6 ms, -3 Volts } →	000
{ 7 ms, -2 Volts } →	001
{ 8 ms, 0 Volts } →	011

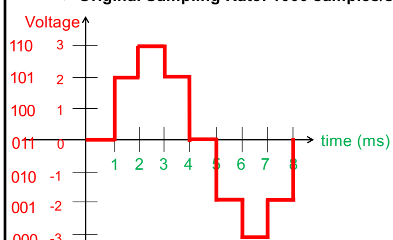
Encoding: mapping data from one form to another (not always conversion)

Example of Lossy Compression

- Sample Rate: **1000 samples/sec**, Resolution: **3-bits** per sample
- Our Sampled Signal: {0, 2.2V, 3V, 2.2V, 0, -2.2V, -3, -2.2V, 0}
- Our Quantized Signal: {0, 2V, 3, 2V, 0, -2, -3, -2, 0}
- Our 3-bit Digitized Data: {011, 101, 110, 101, 011, 001, 000, 001, 011}
  - space required to store/transmit: **27 bits**
- ADC related compression algorithm:
  - CS&Q (Coarser Sampling AND/OR Quantization)
    - Either reduce number of bits per sample AND/OR discard a sample completely
  - Example with our digitized data:
  - Our 3-bit Digitized Data: {011, 101, 110, 101, 011, 001, 000, 001, 011}
  - If we drop the sampling rate by a factor of 2, how impact number of bits needed?
  - ***Lossy*** because we **cannot** restore exact original

De-Compression of Signal:

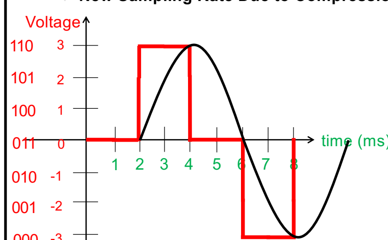
- Decompression & DAC Process
  - Original digital signal: {011, 101, 110, 101, 011, 001, 000, 001, 011}
  - Original Sampling Rate: 1000 samples/sec



Original Signal (recall ADC/DAC is approximation at best anyway!)

De-Compression of Signal:

- Decompression & DAC Process
  - Original compressed signal: {011, , 110, , 011, , 000, , 011}
  - New Sampling Rate Due to Compression: 500 samples/sec



Effect of CS&Q compression
Lowered Sampling Rate
Added "noise" to signal
Listeners might not notice!

Lossy Compression:
One can achieve high Compression ratios

Frequently used for Audio:
MP3 format uses lossy compression algorithm

Lossy: Compression removed every other sample

Two forms of classification

Compression Algorithms

Lossy → Lossless

Compression Algorithms

Fixed Group Size → Variable Group Size

Examples of **Fixed** Group Size:
Take in 2 samples: (6-bits) always spit out: (3-bits)
Take in 8-bit ASCII character (group), spit out 7-bit ASCII character (group)

Probability-Based Lossless Compression

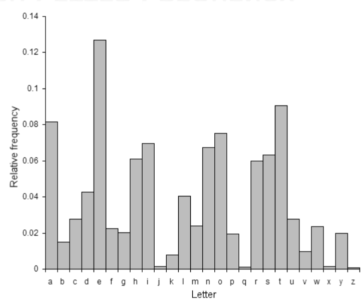
Information Content

- Does each character contain the same amount of "information"?

Statistics

- **How often does each character occur?**
  - Capital letters versus non-capitals?
  - How many e's in preclass quote?
  - How many z's?
  - How many q's?

English Letter Frequency



http://en.wikipedia.org/wiki/File:English-slf.png

Huffman Encoding

- **Developed in 1950's (D.A. Huffman)**
- **Takes advantage of frequency of stream of bits occurrence in data**
  - Can be done for ASCII (8-bits per character)
    - Characters do not occur with equal frequency.
    - How can we exploit statistics (frequency) to pick character encodings?
  - But can also be used for anything with symbols occurring frequently
    - E.g., Music (drum beats…frequently occurring data)
  - Example of **variable length** compression algorithm
    - Takes in fixed size group – spits out variable size replacement

23

How many Bits to represent all letters?

Letter	Binary Encoding	Letter	Binary Encoding
A	00000	N	01101
B	00001	O	01110
C	00010	P	01111
D	00011	Q	10000
E	00100	R	10001
F	00101	S	10010
G	00110	T	10011
H	00111	U	10100
I	01000	V	10101
J	01001	W	10110
K	01010	X	10111
L	01011	Y	11000
M	01100	Z	11001

Including upper and lower case?
…and numbers, how many bits?

ASCII Encoding (7-bit encoding)

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

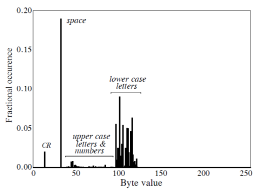
ASCII: ***American Standard Code for Information Interchange***
2^7=128 combinations

Standard encoding, developed in the 1960's

Didn't take into account international standards!

UNICODE
8-bit encoding
2^8=256 possibilities!

Huffman Encoding – The Basics

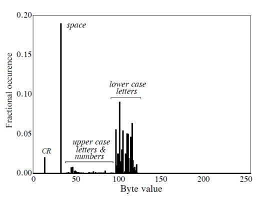


- Example: more than 96% of file consists of 31 characters
- Idea: Assign frequently used characters fewer bits
  - 31 common characters get 5b codes 00000--11110
  - Rest get 13b: 11111+original 8b code
- How many bits do we need on average per original byte?

Calculation

- Bits = #5b-characters * 5 + #13b-character * 13
- Bits=#bytes*0.96*5 + #bytes*0.04*13
- Bits/original-byte = 0.96*5+0.04*13
- Bits/original-byte = 5.32

Huffman Encoding – More Advanced



Example Encoding Table

letter	probability	Huffman code
A	.154	1
B	.110	01
C	.072	0010
D	.063	0011
E	.059	0001
F	.015	000010
G	.011	000011

- Huffman goes further: Assign MOST used characters least # of bits:
  - Most frequent: A= 1, least frequent: G=00011, etc.
  - **Example:** original data stream: C E G A D F B E A···

Huffman encoded: 0010 0001 000011 1 0011 000010 01 0001 1···

Preclass Encoding

symbol	encode	occur
(space)	00	15
A	1110	
B	100100	
C	100101	
D	10110	
E	110	11
F	011010	
G	011011	
H	011000	
I	0111	

symbol	encode	occur
L	0100	
M	1111	
N	1010	
O	10011	
R	0101	
T	10111	
V	10000	
Y	011001	
,	10001	

Preclass Encoding

symbol	encode	occur
(space)	00	15
A	1110	6
B	100100	
C	100101	
D	10110	
E	110	11
F	011010	
G	011011	
H	011000	
I	0111	4

symbol	encode	occur
L	0100	4
M	1111	6
N	1010	5
O	10011	
R	0101	4
T	10111	
V	10000	
Y	011001	
,	10001	

Preclass Encoding

symbol	encode	occur
(space)	00	15
A	1110	6
B	100100	
C	100101	
D	10110	3
E	110	11
F	011010	
G	011011	
H	011000	
I	0111	4

symbol	encode	occur
L	0100	4
M	1111	6
N	1010	5
O	10011	2
R	0101	4
T	10111	3
V	10000	2
Y	011001	
,	10001	2

PRECLASS ENCODING

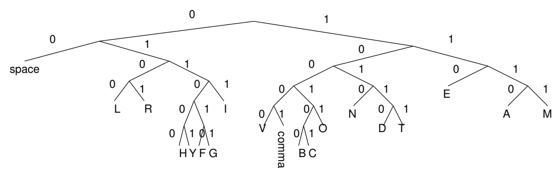
symbol	encode	occur
(space)	00	15
A	1110	6
B	100100	1
C	100101	1
D	10110	3
E	110	11
F	011010	1
G	011011	1
H	011000	1
I	0111	4

symbol	encode	occur
L	0100	4
M	1111	6
N	1010	5
O	10011	2
R	0101	4
T	10111	3
V	10000	2
Y	011001	1
,	10001	2

See how variable length encoding saved bits?
Questions?

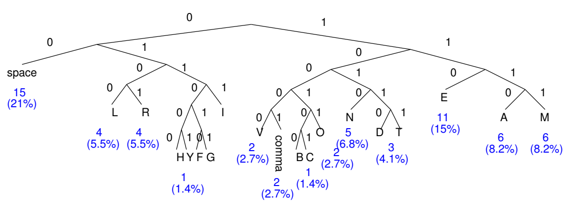
PRECLASS ENCODING

- Prefix decodable on tree



PRECLASS ENCODING

- Probability, encoding, length



INTERLUDE

INTERLUDE

- SNL – 5 minute University
  - Father Guido Sarducci
- https://www.youtube.com/watch?v=kO8x8eoU3L4
- What form of compression here?

FOR COMPUTER ENGINEERING?

- Make the common case fast
- Make the frequent case small

Part 2

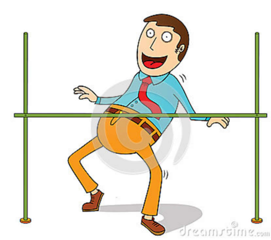
Common Case

- **Big idea in optimization engineering**
  - Make the common case inexpensive
- **Shows up throughout computer systems**
  - Computer architecture
    - Caching, instruction selection, branch prediction, …
  - Networking and communication, data storage
    - Compression, error-correction/retransmission
  - Algorithms and software optimization
  - User Interfaces
    - Where things live on menus, shortcuts, …
    - How you organize your apps on screens

Entropy

Is there a lower bound for compression?

How low can we go with compression?



What is the least # of bits required to encode information?

Claude Shannon

- **Father of Information Theory, brilliant mathematician**
- **While at AT&T Bell Labs, landmark paper in 1948**
- **Determined exactly how low we can go with compression!**

Shannon's Entropy

- **What is entropy?**
  - Chaos/Disorganization/Randomness/Uncertainty
- **Shannon's Famous Entropy Formula:**

$$H = -\sum p(x) \log_2 p(x)$$

Shannon's Entropy *(measured in bits)*

Negative Sum Of:

Probability of each outcome X $\log_2$ of (probability of each outcome)

Estimating Entropy of English Language

- 27 Characters (26 letters + space)
- If we assume all characters are equally probable:
  - $p(each\ character) = \frac{1}{27}$
- Information Entropy per character:

$$H = -\sum p(x)\log_2 p(x)$$

$$H = -27\left(\frac{1}{27}\right)\log\left(\frac{1}{27}\right) = -\log\left(\frac{1}{27}\right) = +4.75\ bits$$

Same thing we got when we said we needed $\log_2$(unique_things) bits

Shannon Entropy

- **Essentially says**
  - Should be able to encode with log(1/p) bits

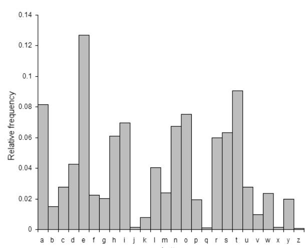
$$\text{Average Bits} = \sum_i p_i \times \text{bits}(i)$$

$$H = -\sum_i p_i \times \log_2(p_i)$$

Where did we calculate Bits earlier in lecture?

Recall:

Not all letters are equally probable in English Language



http://en.wikipedia.org/wiki/File:English-slf.png

Shannon Entropy English Letters

$$H = -\sum_i p_i \times \log_2(p_i)$$

letter	p	$-\log_2(p)$	$-p*\log_2(p)$
a	8.17%	3.61	0.30
b	1.49%	6.07	0.09
c	2.78%	5.17	0.14
d	4.25%	4.56	0.19
e	12.70%	2.98	0.38
f	2.23%	5.49	0.12
z	0.07%	10.40	0.01
sum	100.00%		4.18

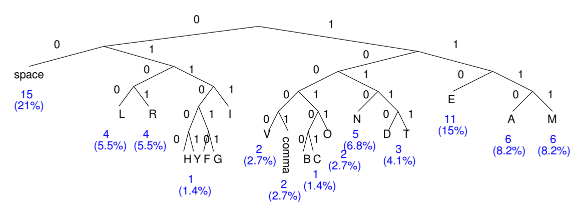
Shannon Entropy Preclass Quote

$$H = -\sum_i p_i \times \log_2(p_i)$$

Symbol	Bits	Occur	P	-log2(p)	$-p*\log_2(p)$	p*bits
(space)	2	15	0.21	2.28	0.47	0.41
A	4	6	0.08	3.60	0.30	0.33
B	6	1	0.01	6.19	0.08	0.08
C	6	1	0.01	6.19	0.08	0.08
D	5	3	0.04	4.60	0.19	0.21
E	3	11	0.15	2.73	0.41	0.45
,	5	2	0.03	5.19	0.14	0.14
				sum	3.74	3.77

Preclass Encoding

- **Probability, encoding, length**



Summing it up: Shannon & Compression

- **Shannon's Entropy represents a lower limit for lossless data compression**
  - It tells us the minimum number of bits that can be used to encode a message without loss (according to a particular model)
- **Shannon's Source Coding Theorem:**
  - A lossless data compression algorithm cannot compress messages to have (on average) more than 1 bit of Shannon's Entropy per bit of encoded message

Learn More

- **ESE 301– Probability**
  - Central to understanding probabilities
    - What cases are common and how common they are
- **ESE 674 – Information Theory**
- **Most all computer engineering courses**
  - Deal with common-case optimizations
  - CIS240, CIS471, CIS380, ESE407, ….

Big Ideas

- **Lossless Compression**
  - Exploit non-uniform statistics of data
  - Given short encoding to most common items
- **Common Case**
  - Make the common case inexpensive
- **Shannon's Entropy**
  - Gives us a formal tool to define lower bound for compressibility of data

This week in Lab

- **Implement Compression!**
  - Implement Huffman Compression
  - **Note:** longer prelab with MATLAB intro; plan accordingly
    - Budget a few hours
- **Remember**
  - Feedback
  - Lab 2 report due today

References

- **S. Smith, "The Scientists and Engineer's Guide to Digital Signal Processing," 1997.**
- **Shannon's Entropy (excellent video) http://www.youtube.com/watch?v=JnJq3Py0dyM**
  - Used heavily in the creation of entropy slides