CS 112: Modeling Uncertainty in Information Systems

Prof. Jenn Wortman Vaughan

April 30, 2012

Lecture 9

Reminders & Announcements

- The course midterm is this Wednesday in class!
- The exam will cover all of Chapters 1 and 2, plus 4.2
- One double-sided sheet of hand-written notes is allowed and will be collected with your exam
- **You may not use any other notes, books, calculators, cell phones, laptops, etc.**
- Best way to prep is to practice problems from the book

Reminders & Announcements

- Since Jacob is traveling, he will not have office hours tomorrow
- Prof. Vaughan will move her weekly office hours to Jacob's usual slot, tomorrow (Tuesday) 11am–1pm
- Jacob will also be available over email and on Piazza to answer questions

Reminders & Announcements

- Homework 1 was returned on Friday
- Four problems were graded for credit, max score is 55
- Mean was 40.1 (73%), median was 42 (76%)

Reminders & Announcements

- Homework 1 was returned on Friday
- Four problems were graded for credit, max score is 55
- Mean was 40.1 (73%), median was 42 (76%)

- If you believe that a grading error was made, you may submit a regrade request within 7 days (so by Friday) by:
  - Dropping your assignment and your written regrade request in the class locker in the TA room
  - Sending an email to Jacob and the graders to let them know to look for your request and which problem it's for

Today

- Symbol codes for encoding text
- Data compression using Huffman coding
- Relationship to entropy

Encoding Data

- Suppose we would like to encode strings of characters in binary. If our strings contain capital letters A, B, C, …, Z and spaces only, what encoding might be choose?

Encoding Data

- Suppose we would like to encode strings of characters in binary. If our strings contain capital letters A, B, C, …, Z and spaces only, what encoding might be choose?

- 26 letters + spaces = 27 symbols to encode, so need 5 bits

Encoding Data

- Suppose we would like to encode strings of characters in binary. If our strings contain capital letters A, B, C, …, Z and spaces only, what encoding might be choose?
- 26 letters + spaces = 27 symbols to encode, so need 5 bits

A = 00000		X = 10111
B = 00001	…	Y = 11000
C = 00010		Z = 11001
D = 00011		space = 11010

Encoding Data

- Suppose we would like to encode strings of characters in binary. If our strings contain capital letters A, B, C, …, Z and spaces only, what encoding might be choose?

- 26 letters + spaces = 27 symbols to encode, so need 5 bits

A = 00000		X = 10111
B = 00001	…	Y = 11000
C = 00010		Z = 11001
D = 00011		space = 11010

- This is essentially how ASCII works

What if we wanted to use fewer bits?

What if we wanted to use fewer bits?

- Could drop leading zeroes…

A = 0			X = 10111
B = 1		…	Y = 11000
C = 10			Z = 11001
D = 11			space = 11010

What if we wanted to use fewer bits?

- Could drop leading zeroes…

A = 0			X = 10111
B = 1		…	Y = 11000
C = 10			Z = 11001
D = 11			space = 11010

- What is the problem with this encoding?

Prefix Codes

- A symbol code is called a prefix code if no codeword is a prefix of any other codeword
  - Prefix code: A = 00 B = 101 C = 111
  - Not a prefix code: A = 0 B = 01 C = 001

Prefix Codes

- A symbol code is called a prefix code if no codeword is a prefix of any other codeword
  - Prefix code: A = 00 B = 101 C = 111
  - Not a prefix code: A = 0 B = 01 C = 001
- Any fixed length code is a prefix code

Prefix Codes

- A symbol code is called a prefix code if no codeword is a prefix of any other codeword
  - Prefix code: A = 00 B = 101 C = 111
  - Not a prefix code: A = 0 B = 01 C = 001
- Any fixed length code is a prefix code
- Any prefix code can be represented as a binary tree, and any code that can be represented this way is a prefix code

Designing Optimal Prefix Codes

- What if we would like to design a prefix code to minimize the expected length of an encoded message?

Designing Optimal Prefix Codes

- What if we would like to design a prefix code to minimize the expected length of an encoded message?

- We can start by writing down the frequency of each character, which we can view as a probability

Designing Optimal Prefix Codes

- What if we would like to design a prefix code to minimize the expected length of an encoded message?

- We can start by writing down the frequency of each character, which we can view as a probability

$$P(A) = 1/2,\ P(B) = 1/4,\ P(C) = 1/8,\ P(D) = 1/8$$

Designing Optimal Prefix Codes

- What if we would like to design a prefix code to minimize the expected length of an encoded message?

- We can start by writing down the frequency of each character, which we can view as a probability

$$P(A) = 1/2,\ P(B) = 1/4,\ P(C) = 1/8,\ P(D) = 1/8$$

- Where have we seen this problem before?

Designing Optimal Prefix Codes

- What if we would like to design a prefix code to minimize the expected length of an encoded message?

- We can start by writing down the frequency of each character, which we can view as a probability

  P(A) = 1/2, P(B) = 1/4, P(C) = 1/8, P(D) = 1/8

- Where have we seen this problem before?

- What properties did the optimal solution satisfy?

Top-Down Approach

- We could try to build a tree from the root node down, making near-equiprobable splits…

Top-Down Approach

- We could try to build a tree from the root node down, making near-equiprobable splits…

$P(A) = 0.01$	$P(E) = 0.47$
$P(B) = 0.24$	$P(F) = 0.01$
$P(C) = 0.05$	$P(G) = 0.02$
$P(D) = 0.20$	

Top-Down Approach

- We could try to build a tree from the root node down, making near-equiprobable splits…

$P(A) = 0.01$	$P(E) = 0.47$
$P(B) = 0.24$	$P(F) = 0.01$
$P(C) = 0.05$	$P(G) = 0.02$
$P(D) = 0.20$	

- What is the average (expected) number of bits needed to encode a character using this scheme?

Bottom-Up Approach

- Alternatively, we could try to cleverly build a tree from the leaves up…

Bottom-Up Approach

- Alternatively, we could try to cleverly build a tree from the leaves up…

$P(A) = 0.01$	$P(E) = 0.47$
$P(B) = 0.24$	$P(F) = 0.01$
$P(C) = 0.05$	$P(G) = 0.02$
$P(D) = 0.20$	

Bottom-Up Approach

- Alternatively, we could try to cleverly build a tree from the leaves up…

$P(A) = 0.01$	$P(E) = 0.47$
$P(B) = 0.24$	$P(F) = 0.01$
$P(C) = 0.05$	$P(G) = 0.02$
$P(D) = 0.20$	

- What is the average (expected) number of bits needed to encode a character using this scheme?

Bottom-Up Approach

- This bottom up approach is known as Huffman coding

Bottom-Up Approach

- This bottom up approach is known as Huffman coding
- Huffman coding produces a provably optimal symbol code in terms of expected number of bits per character, R

Bottom-Up Approach

- This bottom up approach is known as Huffman coding
- Huffman coding produces a provably optimal symbol code in terms of expected number of bits per character, R
- The expected number of bits per character achieved by Huffman coding can be bounded in terms of the entropy

$$
\begin{aligned}
H(A_1, A_2, \ldots, A_n) &\le R(A_1, A_2, \ldots, A_n) \\
&\le H(A_1, A_2, \ldots, A_n) + 1
\end{aligned}
$$

Bound on Information Rate

- Recall that the entropy is the average information content of a set of events $A_1,\ldots,A_n$ that partition Ω

$$H(A_1,\ldots,A_n) = \sum_{i=1}^{n} P(A_i) I(A_i) = \sum_{i=1}^{n} P(A_i) \log_2\left(\frac{1}{P(A_i)}\right)$$

Bound on Information Rate

- Recall that the entropy is the average information content of a set of events $A_1,\ldots,A_n$ that partition Ω

$$H(A_1,\ldots,A_n) = \sum_{i=1}^{n} P(A_i) I(A_i) = \sum_{i=1}^{n} P(A_i) \log_2\left(\frac{1}{P(A_i)}\right)$$

- In the previous example, the entropy is about 1.93 (verify this on your own!) so 1.97 is pretty close

Examples of Huffman Coding

- Consider a uniform character frequency distribution

$P(A) = 0.25$	$P(C) = 0.25$
$P(B) = 0.25$	$P(D) = 0.25$

- How could we generate a Huffman code?

Examples of Huffman Coding

- Consider a uniform character frequency distribution

$P(A) = 0.25$	$P(C) = 0.25$
$P(B) = 0.25$	$P(D) = 0.25$

- How could we generate a Huffman code?
- What is the entropy?

Examples of Huffman Coding

- Consider a uniform character frequency distribution

  $P(A) = 0.25$ $\qquad$ $P(C) = 0.25$

  $P(B) = 0.25$ $\qquad$ $P(D) = 0.25$

- How could we generate a Huffman code?
- What is the entropy?
- How does this compare to the expected number of bits per character?

Examples of Huffman Coding

- Consider a uniform character frequency distribution

$P(A) = 0.25$	$P(C) = 0.25$
$P(B) = 0.25$	$P(D) = 0.25$

- How could we generate a Huffman code?
- What is the entropy?
- How does this compare to the expected number of bits per character? (Nice powers of 2 again…)

Exercise

- Consider the following character frequency distribution

$P(A) = 0.45$	$P(D) = 0.05$
$P(B) = 0.15$	$P(E) = 0.10$
$P(C) = 0.20$	$P(F) = 0.05$

- Derive the following symbol codes and calculate the expected bits per character of each
  a) An optimal fixed length code
  b) A prefix code derived using the top-down approach
  c) A prefix code derived using Huffman coding