Probability for Computer Science

Spring 2021

Lecture 22

Prof. Claire Monteleoni

Boulder

Today

Introduction to Information Theory

- Compression
- Information Rate
- Huffman Coding

Exercise (or Pre-class assignment)

- Complete the reading listed on Schedule.
  - Study examples, especially McKay Book, examples 5.4 - 5.8.
- Compute the entropy of the following distributions.
  - Once you have set up the equation, you can use a calculator to solve (e.g. logarithms, multiplication), approx. to 3 decimal places.

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

Symbol	Probability
A	0.30
B	0.30
C	0.13
D	0.12
E	0.10
F	0.05

Intro to compression and coding

- We need to transmit messages, sequences of symbols, e.g. $A_1A_3A_1A_2A_4A_3$.
  - e.g. text, music, any digitized data.
  - Can also view A_i as the event that the transmitted symbol is A_i.

- Goal: represent message using as few bits as possible.

- Compressed messages must be uniquely decodable.

Intro to compression and coding

- Simplest binary code is fixed-length: each codeword has k bits.
  - Recall: Can encode 2^k events using k bits.

- Variable-length code: use shorter bit strings (codewords) for more probable events.
  - Recall: More probable events contain less information.

- The compression limit is determined by the entropy.

Fixed-length codes

- Consider the alphabet of capital letters and the space character.
- We need to encode 27 characters.
- How many bits do we need for a fixed length code?

  5 bits. (4 bits could only uniquely encode 16 characters).

- For Example: A = 00000, B = 00001, C = 00010, D = 00011, etc.

This is the idea behind ASCII.

- Fixed length codes (of k bits) are uniquely decodable
- Simply segment into bit-strings of length k, and there's a unique symbol corresponding to each bit string.

Towards a variable length code

- Given a fixed length code:

A = 00000, B = 00001, C = 00010, D = 00011, etc.

- What if we want to use fewer bits? Perhaps drop the leading 0's:
- E.g. A = 0, B = 1, C = 10, D = 11, etc.

- What's the problem with this encoding?

- How do we decode 101? Or 111?

Prefix codes

A symbol code is called a prefix code if no codeword is a prefix of any other codeword.

This makes messages (strings of symbols) uniquely decodable.

Which of the following codes are prefix codes for A, B, C?

- A = 00, B = 101, C = 111

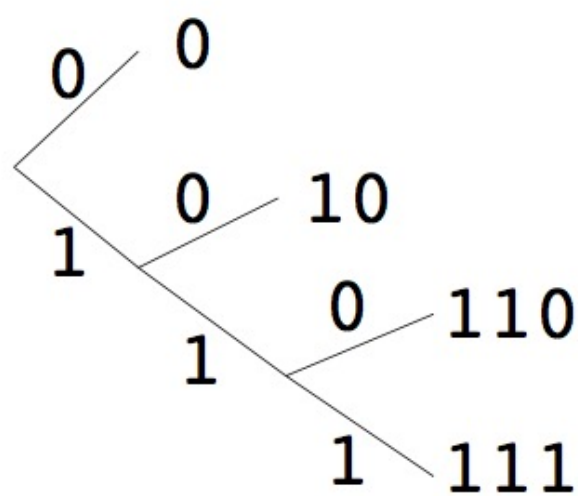
- A = 0, B = 01, C = 001

Prefix codes

Any prefix code can be represented as a binary tree.

- Each leaf is a codeword for one of the symbols, computed as the bit-string along the path from root.

Any code that can be represented as a binary tree is a prefix code, e.g.



Now: Exercise 1

Credit: D. MacKay 2003

Prefix codes

A symbol code is called a prefix code if no codeword is a prefix of any other codeword.

Any fixed-length code is a prefix code.

- Each codeword is exactly k bits, and unique, so none is a prefix of another

So why not always use a fixed-length code?

May want to reduce the number of bits transmitted on average.

We would like to design a prefix code to minimize the expected length of an encoded message.

Information Rate

If we know the frequency of each symbol A_i, we can view it as a probability, $P(A_i)$.

The information rate of a code is the average number of bits per symbol:

$$R(A_1, \ldots, A_n) = \sum_{i=1}^{n} P(A_i) L(A_i)$$

where $L(A_i)$ is the number of bits in the codeword for A_i.

Bound on the Information Rate

The best achievable (minimum) information rate, for any code in which each symbol is uniquely encoded, is the entropy:

$$R(A_1, \ldots, A_n) \geq H(A_1, \ldots, A_n)$$

$$\sum_{i=1}^{n} P(A_i) L(A_i) \geq \sum_{i=1}^{n} P(A_i) I(A_i)$$

$$= \sum_{i=1}^{n} P(A_i) \log_2 \left(\frac{1}{P(A_i)} \right)$$

Now: Exercises 2 & 3

Optimal codes

Fixed-length codes are optimal when all symbols occur with equal probability.

When the symbols have different probabilities, the optimal code will be a variable-length code.

- but not *any* variable length code; some achieve worse information rates than others.

Huffman Code: Information Rate bound

The information rate of the Huffman code is upper bounded as follows:

$$R(A_1, \ldots, A_n) \leq H(A_1, \ldots, A_n) + 1$$

This is optimal for prefix codes.

And remember, for any code which uniquely encodes each symbol,

$$H(A_1, \ldots, A_n) \leq R(A_1, \ldots, A_n)$$

Huffman algorithm

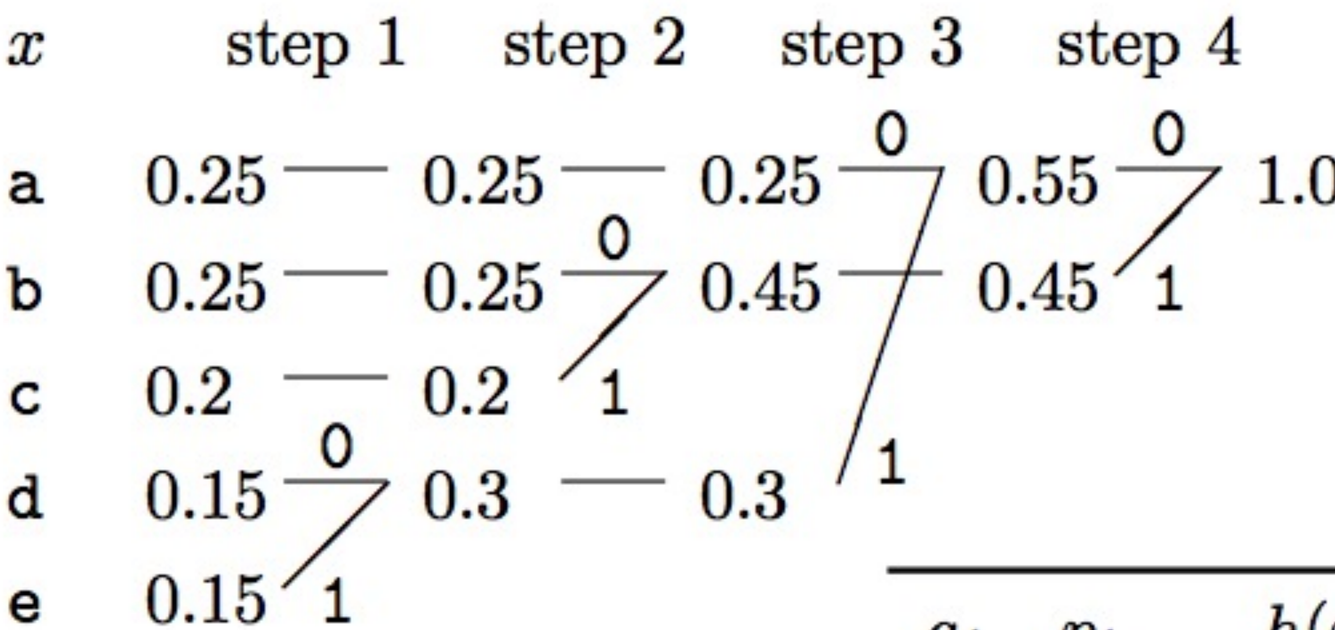
1. Take the two least probable symbols in the alphabet. These two symbols will be given the longest codewords, which will have equal length, and differ only in the last digit.

2. Combine these two symbols into a single symbol, and repeat.

At the end, codewords are read from root to leaf.

Credit: D. MacKay 2003

Example 1



a_i	p_i	$h(p_i)$	l_i	$c(a_i)$
a	0.25	2.0	2	00
b	0.25	2.0	2	10
c	0.2	2.3	2	11
d	0.15	2.7	3	0[illegible]
e	0.15	2.7	3	[illegible]

Now: Exercise 4

Credit: D. MacKay 2003

Post-class

Post-class exercises:

- Propose a symbol distribution and then design a Huffman code for it. Compute its information rate.
- Optional: Compare with info. rate of code constructed top-down (Starting from all events, recurse, making equiprobable splits).