Information Theory

http://www.inf.ed.ac.uk/teaching/courses/it/

Week 3
Symbol codes

Iain Murray, 2012
School of Informatics, University of Edinburgh

(Binary) Symbol Codes

For strings of symbols from alphabet e.g.,
$x_i \in \mathcal{A}_X = \{A, C, G, T\}$

Binary codeword assigned to each symbol

$$CGTAGATTACAGG$$
$$\downarrow$$
$$10111110011101101100100111111$$

A	0
C	10
G	111
T	110

Codewords are concatenated without punctuation

Uniquely decodable

We'd like to make all codewords short

But some codes are not **uniquely decodable**

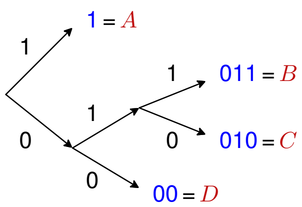
$$CGTAGATTACAGG$$
$$\downarrow$$
$$111111001110110110010111111$$
$$\downarrow$$
$$CGTAGATTACAGG$$
$$CCCCCCAACCCACCACCAACACCCCCC$$
$$CCGCAACCCATCCAACAGCCC$$
$$GGAAGATTACAGG$$
$$???$$

A	0
C	1
G	111
T	110

Instantaneous/Prefix Codes

Attach symbols to leaves of a binary tree

Codeword gives path to get to leaf



"Prefix code" because no codeword is a prefix of another

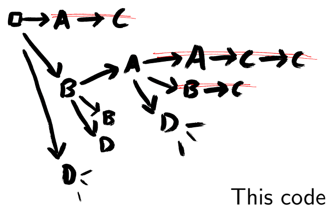
Decoding: follow tree while reading stream until hit leaf

Symbol is *instantly* identified. Return to root of tree.

Non-instantaneous Codes

The last code was **instantaneously decodable**:
We knew as soon as we'd finished receiving a symbol

$$101100000101100$$
$$\downarrow$$



A	1
B	10
C	000
D	100

This code *is* uniquely decodable,
but not instantaneous or pleasant!

Expected length/symbol, $\bar{L}$

Code lengths: $\{\ell_i\} = \{\ell_1, \ell_2, \ldots, \ell_I\}$

Average, $\bar{L} = \sum_i p_i\, \ell_i$

Compare to Entropy:

$$H(X) = \sum_i p_i \log \frac{1}{p_i}$$

If $\ell_i = \log \frac{1}{p_i}$ **or** $p_i = 2^{-\ell_i}$ **we compress to the entropy**

An optimal symbol code

An example code with:

$$\bar{L} = \sum_i p_i\, \ell_i = H(X) = \sum_i p_i \log \frac{1}{p_i}$$

x	$p(x)$	codeword
A	1/2	0
B	1/4	10
C	1/8	110
D	1/8	111

Entropy: decomposability

Flip a coin:
Heads → A
Tails → **flip again:**
Heads → B
Tails → C

$\mathcal{A}_X = \{A,\ B,\ C\}$
$\mathcal{P}_X = \{0.5,\ 0.25,\ 0.25\}$

$$H(X) = 0.5 \log \tfrac{1}{0.5} + 0.25 \log \tfrac{1}{0.25} + 0.25 \log \tfrac{1}{0.25} = 1.5 \text{ bits}$$

Or: $H(X) = H_2(0.5) + 0.5\, H_2(0.5) = 1.5$ bits

Shannon's 1948 paper §6. MacKay §2.5, p33

Why look at the decomposability of Entropy?

Mundane, but useful: it can make your algebra a lot neater. Decomposing computations on graphs is ubiquitous in computer science.

Philosophical: we expect that the expected amount of information from a source should be the same if the same basic facts are represented in different ways and/or reported in a different order.

Shannon's 1948 paper used the desired decomposability of entropy to derive what form it must take, section 6. This is similar to how we intuited the information content from simple assumptions.

Limit on code lengths

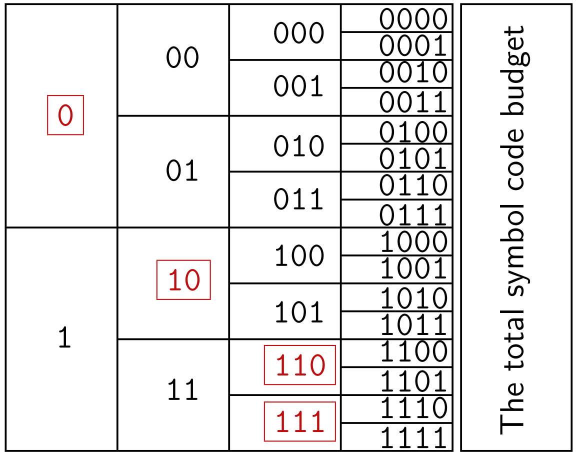
Imagine coding under an implicit distribution:

$$q_i = \frac{1}{Z} 2^{-\ell_i}, \quad Z = \sum_i 2^{-\ell_i}.$$

$$H = \sum_i q_i \log \frac{1}{q_i} = \sum_i q_i \left(\ell_i + \log Z\right) = \bar{L} + \log Z$$
$$\Rightarrow \log Z \le 0, \quad Z \le 1$$

Kraft–McMillan Inequality $\boxed{\sum_i 2^{-\ell_i} \le 1}$ (if uniquely-decodable)

Proof without invoking entropy bound: p95 of MacKay, or p116 Cover & Thomas 2nd Ed.



Kraft Inequality

If height of budget is 1, codeword has height $= 2^{-\ell_i}$

Pick codes of required lengths in order from shortest–largest

Choose heighest codeword of required length beneath previously-chosen code (There won't be a gap because of sorting)

Can always pick codewords if total height, $\sum_i 2^{-\ell_i} \le 1$

Kraft–McMillan Inequality $\boxed{\sum_i 2^{-\ell_i} \le 1}$ (instantaneous code possible)

Corollary: there's probably no point using a non-instantaneous code.
Can always make **complete code** $\sum_i 2^{-\ell_i} = 1$: slide last codeword left.

Summary of Lecture 5

Symbol codes assign each symbol in an alphabet a codeword.
(We only considered binary symbol codes, which have binary codewords.)
Messages are sent by concatenating codewords with no punctuation.

Uniquely decodable: the original message is unambiguous

Instantaneously decodable: the original symbol can always be determined as soon as the last bit of its codeword is received.

Codeword lengths must satisfy $\sum_i 2^{-\ell_i} \le 1$ for unique decodability

Instantaneous prefix codes can always be found (if $\sum_i 2^{-\ell_i} \le 1$)

Complete codes have $\sum_i 2^{-\ell_i} = 1$, as realized by prefix codes made from binary trees with a codeword at every leaf.

If (big if) symbols are drawn i.i.d. with probabilities $\{p_i\}$, *and* $\ell_i = \log \frac{1}{p_i}$, then a prefix code exists that offers optimal compression.

Next lecture: how to form the best symbol code when $\{\log \frac{1}{p_i}\}$ are not integers.

Performance of symbol codes

Simple idea: set $\ell_i = \left\lceil \log \frac{1}{p_i} \right\rceil$

These codelengths satisfy the Kraft inequality:

$$\sum_i 2^{-\ell_i} = \sum_i 2^{-\lceil \log 1/p_i \rceil} \le \sum_i p_i = 1$$

Expected length, $\bar{L}$:

$$\bar{L} = \sum_i p_i \ell_i = \sum_i p_i \lceil \log 1/p_i \rceil < \sum_i p_i \left(\log 1/p_i + 1\right)$$

$$\bar{L} < H(\mathbf{p}) + 1$$

Symbol codes can compress to within 1 bit/symbol of the entropy.

Optimal symbol codes

Encode independent symbols with known probabilities:

E.g., $\mathcal{A}_X = \{A, B, C, D, E\}$
$\mathcal{P}_X = \{0.3, 0.25, 0.2, 0.15, 0.1\}$

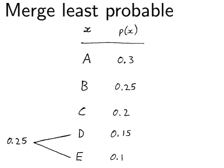
We can do better than $\ell_i = \left\lceil \log \frac{1}{p_i} \right\rceil$

The *Huffman algorithm* gives an optimal symbol code.

Proof: MacKay Exercise 5.16 (with solution).
Cover and Thomas has another version.

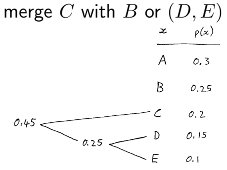
Huffman algorithm

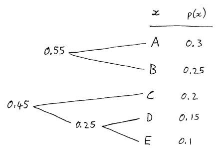
Merge least probable

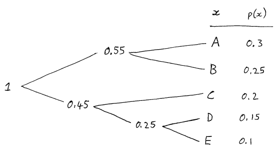


$P(D \text{ or } E) = 0.25$

Can merge C with B or (D, E)



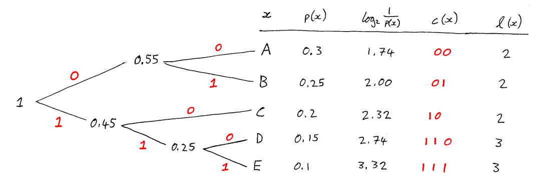




Continue merging least probable, until root represents all events $P = 1$

Huffman algorithm

Given a tree, label branches with 1s and 0s to get code



Code-lengths are close to the information content
(not just rounded up, some are shorter)

$H(X) \approx 2.23$ bits. Expected length $= 2.25$ bits.

Wow! Despite limitations we will discuss, Huffman codes can be very good. You'll find them inside many systems (e.g., bzip2, jpeg, mp3), although all these schemes do clever stuff to come up with a good symbol representation.

Huffman decoding

Huffman codes are easily and uniquely decodable because they are prefix codes

Reminder on decoding a prefix code stream:

- Start at root of tree
- Follow a branch after reading each bit of the stream
- Emit a symbol upon reaching a leaf of the tree
- Return to the root after emitting a symbol...

An input stream can only give one symbol sequence, the one that was encoded

Building prefix trees 'top-down'

Heuristic: if you're ever building a tree, consider top-down vs. bottom-up (and maybe middle-out)

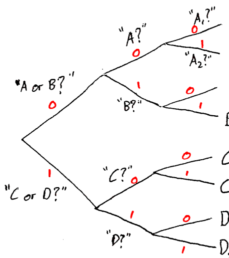
Weighing problem strategy:
Use questions with nearly uniform distribution over the answers.

How well would this work on the ensemble to the right?

x	$P(x)$
A_1	0.24
A_2	0.01
B_1	0.24
B_2	0.01
C_1	0.24
C_2	0.01
D_1	0.24
D_2	0.01

$H(X) = 2.24$ bits (just over $\log 4 = 2$). Fixed-length encoding: 3 bits

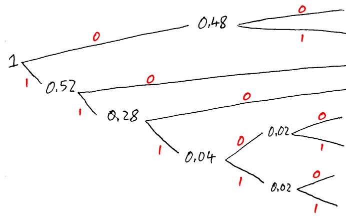
Top-down performing badly



x	$p(x)$	$c(x)$	$\ell(x)$
A_1	0.24	000	3
A_2	0.01	001	3
B_1	0.24	010	3
B_2	0.01	011	3
C_1	0.24	100	3
C_2	0.01	101	3
D_1	0.24	110	3
D_2	0.01	111	3

Probabilities for answers to first two questions is $(1/2, 1/2)$
Greedy strategy $\Rightarrow$ very uneven distribution at end

Compare to Huffman



x	$p(x)$	$c(x)$	$\ell(x)$
A_1	0.24	00	2
B_1	0.24	01	2
C_1	0.24	10	2
D_1	0.24	110	3
A_2	0.01	11100	5
B_2	0.01	11101	5
C_2	0.01	11110	5
D_2	0.01	11111	5

Expected length 2.36 bits/symbol

(Symbols reordered for display purposes only)

Relative Entropy / KL

Implicit probabilities: $q_i = 2^{-\ell_i}$
($\sum_i q_i = 1$ because Huffman codes are complete)

Extra cost for using "wrong" probability distribution:

$$\begin{aligned}\Delta L &= \sum_i p_i \ell_i - H(X)\\ &= \sum_i p_i \log 1/q_i - \sum_i p_i \log 1/p_i\\ &= \sum_i p_i \log \frac{p_i}{q_i} \;=\; D_{\mathrm{KL}}(p\,||\,q)\end{aligned}$$

$D_{\mathrm{KL}}(p\,||\,q)$ **is the *Relative Entropy* also known as the *Kullback–Leibler divergence* or *KL-divergence***

Gibbs' inequality

An important result:

$$D_{\mathrm{KL}}(p\,||\,q) \geq 0$$

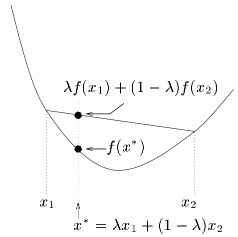
with equality only if $p = q$

"If we encode with the wrong distribution we will do worse than the fundamental limit given by the entropy"

A simple direct proof can be shown using convexity.
(Jensen's inequality)

Convexity

$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$



Strictly convex functions:
Equality only if λ is 0 or 1, or if $x_1 = x_2$
(non-strictly convex functions contain straight line segments)

Convex vs. Concave

For (strictly) concave functions reverse the inequality

A (con)cave

Photo credit: Kevin Krejci on Flickr

Summary of Lecture 6

The **Huffman Algorithm** gives optimal symbol codes:
Merging event adds to code length for children, so
Huffman always merges least probable events first

A complete code implies negative log probabilities: $q_i = 2^{-\ell_i}$.
If the symbols are generated with these probabilities, the symbol code compresses to the entropy. Otherwise the number of extra bits/symbol is given by the **Relative Entropy** or **KL-divergence**: $D_{\mathrm{KL}}(p\,||\,q) = \sum_i p_i \log \frac{p_i}{q_i}$

Gibbs' inequality says $D_{\mathrm{KL}}(p\,||\,q) \geq 0$ with equality only when the distributions are equal.

Convexity and Concavity are useful properties when proving several inequalities in Information Theory. Next time: the basis of these proofs is **Jensen's inequality**, which can be used to prove Gibbs' inequality.