Entropy and Huffman Coding

Martin Leslie

Department of Mathematics
University of Arizona

September 28, 2011

Twenty questions

- Imagine a 'spinner' that produces various symbols with probabilities as follows:

x	A	B	C	D	E	F
$p(x)$	$1/2$	$1/4$	$1/8$	$1/16$	$1/32$	$1/32$

- Can you devise a set of yes–no questions that minimise the maximum number of questions to find out which result occurred?
- What about minimising the average number of yes–no questions?

Data compression

- Our answers can also be used as methods to store the results of each spin. 'Uncompressed' storage needs 3 bits per letter, our optimal solution is 63/32 bits per letter.

Entropy

- Let X be a discrete random variable taking values in a finite alphabet $\mathcal{X}$ with probability mass function $p(x)$.
- The *self-information* (or *surprisal*) of $x \in \mathcal{X}$ is $\log \frac{1}{p(x)}$ where $\log$ means $\log_2$.
- The *entropy* of X is

$$\begin{aligned} H(X) &= \text{average self-information} \\ &= E_p\left[\log \frac{1}{p(X)}\right] \\ &= \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)} \\ &= -\sum_{x \in \mathcal{X}} p(x) \log p(x) \end{aligned}$$

How to think about entropy

- Entropy is the average number of bits of information you gain about the value of X.
- Equivalently, it is the average uncertainty you have about each value of X before you receive it.
- The entropy of a fair coin flip is one bit. The entropy of a biased coin flip is less. For example $H(0.1, 0.9) = 0.47$.

Some examples

- The original spinner example has $H \approx 1.97$:

p_i	$1/2$	$1/4$	$1/8$	$1/16$	$1/32$	$1/32$
$-\log p_i$	1	2	3	4	5	5

- If we have probabilities as below we get $H \approx 2.23$:

p_i	0.35	0.17	0.17	0.16	0.15
$-\log p_i$	1.51	2.56	2.56	2.64	2.74

Axioms for entropy

- Let $\Delta_m = \left\{ (p_1, p_2, \ldots, p_m) : p_i \in (0,1), \sum_{i=1}^{m} p_i = 1 \right\}$.
- Then entropy is the unique sequence of functions

$$H_m \colon \Delta_m \to \mathbb{R}_{\geq 0}$$

for $m = 2, 3, \ldots$ such that

1. H_m is symmetric in its inputs.
2. $H_2\left(\frac{1}{2}, \frac{1}{2}\right) = 1$.
3. $H_2(p, 1-p)$ is continuous in p.
4. $H_m(p_1, p_2, \ldots, p_m) =$
   $H_{m-1}(p_1 + p_2, p_3, \ldots, p_m) + (p_1 + p_2)\ H_2\left(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2}\right)$.

Entropy and physics

- Gibbs entropy of system with microstates of probability p_i is

$$H = -k_B \sum_i p_i \log p_i.$$

- This entropy (without the constant k_B) is basically the average length of description of a microstate given the macrostate.
- In equilibrium all microstates are equally likely. This maximises the entropy, so as a system goes through different macrostates towards equilibrium it increases in entropy (the Second Law of Thermodynamics!).

An Inequality

- If ϕ is a convex function then

$$\phi(px_1 + (1-p)x_2) \leq p\phi(x_1) + (1-p)\phi(x_2).$$

By induction this can be extended to

$$\phi\left(E_p[X]\right) \leq E_p[\phi(X)].$$

(Jensen's inequality!).

Gibbs' inequality

- Gibbs' inequality says that if you use the 'wrong' probabilities q_i instead of p_i entropy will only increase:

$$\sum p_i \log \frac{1}{q_i} \geq \sum p_i \log \frac{1}{p_i}.$$

- To prove this:

$$\begin{aligned}
\sum p_i \log \frac{p_i}{q_i} &= E_p \left[-\log \frac{q_i}{p_i} \right] \\
&\geq -\log E_p \left[\frac{q_i}{p_i} \right] \\
&= -\log \sum q_i \\
&= 0
\end{aligned}$$

Lossless data compression

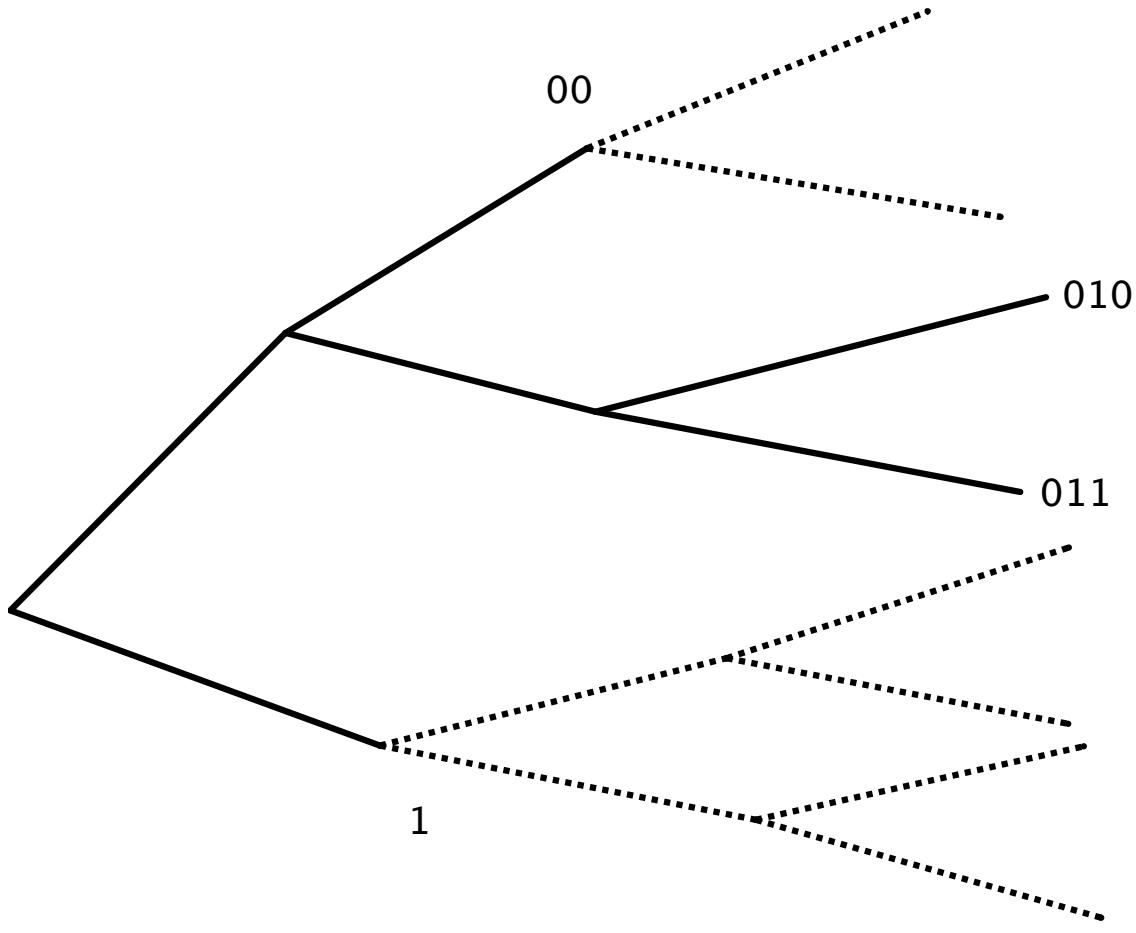
- We consider source codes (i.e. lossless data compression) which take one input symbol at a time and give out a string of bits.
- A *source code* $C\colon \mathcal{X} \to \{0,1\}^+$ is a *prefix code* if no *codeword* $C(x)$ is a prefix of any other codeword.
- A prefix code can be decoded 'instantaneously' and uniquely.
- We would like to minimise $\overline{L} = \sum p_i l_i$ where l_i are codeword lengths.

Kraft inequality

- A prefix code with codeword lengths l_i exists if and only if

$$\sum 2^{-l_i} \leq 1.$$

- To prove this consider a complete binary tree of depth l_m:



Kraft inequality Proof

- A codeword of length l_i has a 'shadow' of $2^{l_m - l_i}$ leaves in the tree.
- So $\sum_i 2^{l_m - l_i} \leq 2^{l_m}$.

H is lower bound for $\overline{L}$

- Let $z = \sum_j 2^{-l_j}$ and $q_i = 2^{-l_i}/z$.
- Rearranging, $l_i = \log \frac{1}{q_i} - \log z$.
- If we have a prefix code for X, then $H(X) \leq \overline{L}$. To see this:

$$\begin{aligned} \overline{L} = \sum p_i l_i &= \sum p_i \log \frac{1}{q_i} - \log z \\ &\geq \sum p_i \log \frac{1}{p_i} - \log z \\ &\geq H(X) \end{aligned}$$

- So entropy gives a lower bound for how much a source can be compressed! (Assuming the source has iid outputs and we compress one symbol at a time).

How might we compress something?

- We want to find integers l_i with $\sum_i 2^{-l_i} \leq 1$ that minimize $\sum p_i l_i$.
- Shannon coding takes $l_i = \lceil -\log p_i \rceil$.
- We have

$$\begin{aligned} \sum 2^{-l_i} &\leq \sum 2^{\log p_i} \\ &= 1 \end{aligned}$$

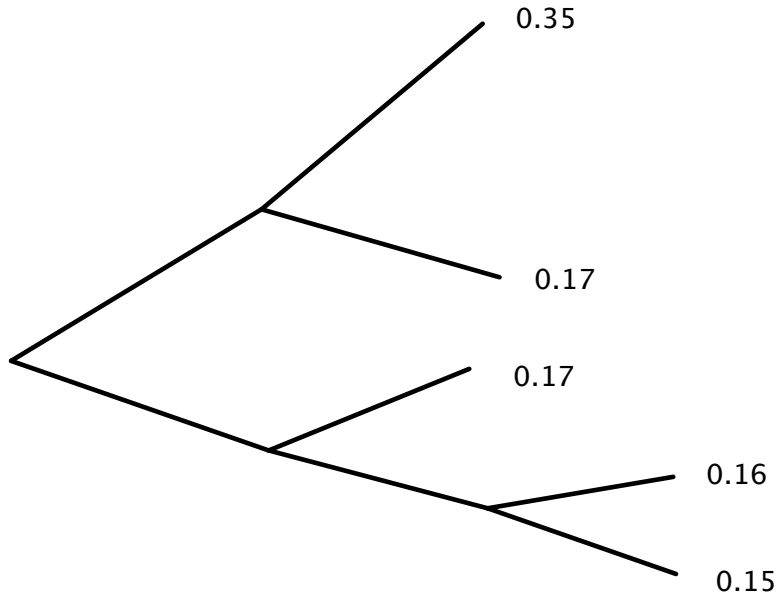
  so there exists a prefix code with these lengths.
- Also,

$$\begin{aligned} \sum p_i l_i &\leq \sum p_i(-\log p_i + 1) \\ &= -\sum p_i \log p_i + \sum p_i \\ &= H(X) + 1 \end{aligned}$$

- So with Shannon coding $H(X) \leq \overline{L} \leq H(X) + 1$.

Can you think of another way?

- Fano coding orders the probabilities and then recursively cuts them in half in the most even way possible.



p_i	0.35	0.17	0.17	0.16	0.15	
$-\log p_i$	1.51	2.56	2.56	2.64	2.74	$H = 2.23$
l_S	2	3	3	3	3	$L_S = 2.65$
l_F	2	2	2	3	3	$L_F = 2.31$

Optimal prefix codes

- Order the elements of our alphabet so that $p_1 \geq p_2 \geq \ldots \geq p_m$. Then there exists an optimal prefix code with lengths l_i such that:
  1. $p_j > p_k$ implies $l_j \leq l_k$;
  2. the two longest codewords have the same length; and
  3. Two of the longest codewords correspond to two of the least likely symbols and differ only in their last bit.
- From here we proceed by induction to construct Huffman codes
- For probabilities $p_1 \geq p_2 \geq \ldots \geq p_m$, by induction we have an optimal code on $m-1$ symbols for probabilities $p_1, p_2, \ldots, p_{m-2}, p_{m-1} + p_m$. Say this code has lengths $l_1, l_2, \ldots, l_{m-2}, \ell$.
- Construct a code for m symbols by extending this code to lengths $l_1, l_2, \ldots, l_{m-2}, l_{m-1} = \ell + 1, l_m = \ell + 1$.

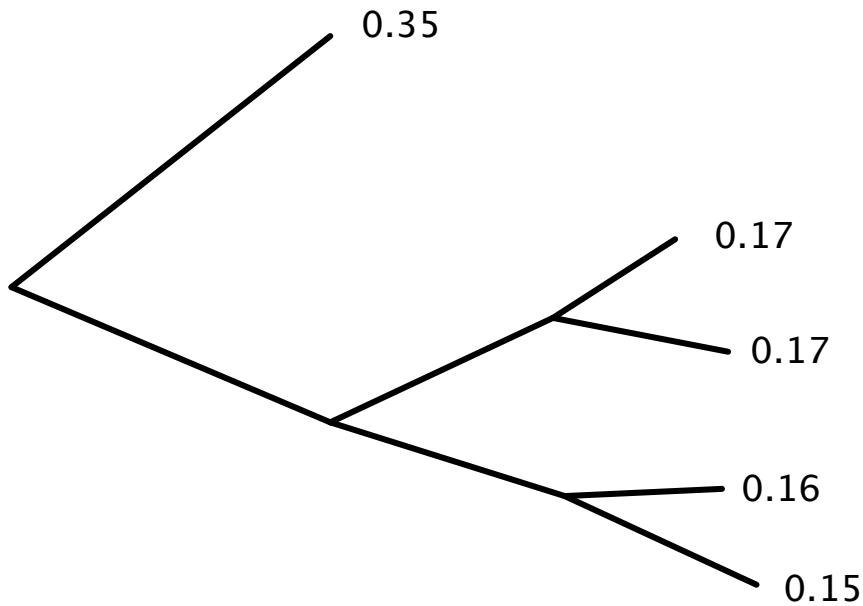
Huffman code optimality

- If the lengths $l_1, \ldots, l_m$ are not optimal then there exists a code with lengths $l'_1, \ldots, l'_m$ that satisfies 1-3 from previous slide and has $\sum_{i=1}^{m} p_i l'_i < \sum_{i=1}^{m} p_i l_i$.
- Condense down the code with lengths l'_i on m symbols to a code on $m-1$ symbols with lengths $l'_1, \ldots, l'_{m-2}, l'_{m-1} - 1$.
- This code has average length

$$\begin{aligned}
&\sum_{i=1}^{m-2} p_i l'_i + (p_{m-1} + p_m)(l'_{m-1} - 1) \\
=&\sum_{i=1}^{m} p_i l'_i - (p_{m-1} + p_m) \\
<&\sum_{i=1}^{m} p_i l_i - (p_{m-1} + p_m) \\
=&\sum_{i=1}^{m-2} p_i l_i + (p_{m-1} + p_m)(l_{m-1} - 1).
\end{aligned}$$

Huffman coding

- Huffman code builds tree from the bottom.



p_i	0.35	0.17	0.17	0.16	0.15	
$-\log p_i$	1.51	2.56	2.56	2.64	2.74	$H = 2.23$
l_S	2	3	3	3	3	$L_S = 2.65$
l_F	2	2	2	3	3	$L_F = 2.31$
l_H	1	3	3	3	3	$L_H = 2.30$