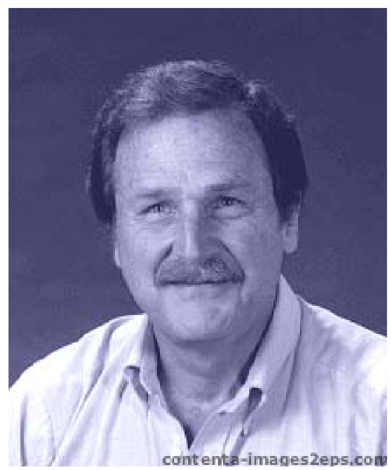
Lecture 9: Huffman Codes

- Huffman codes
- Optimality
- Kolmogorov complexity

Huffman Codes (1952)

- The optimal (shortest expected length) prefix code for a given distribution

- $H(X) \leq L < H(X) + 1$



David Huffman, 1925 - 1999

- Start from small probabilities
- Form a tree
- Assign 0 to higher branch, 1 to lower branch

- Binary alphabet $D = 2$

- Expected code length

$$L = \sum p_i l_i = (0.25 + 0.25 + 0.2) \times 2 + 3 \times 0.3 = 2.3$$

- Entropy $H(X) = \sum p_i \log(1/p_i) = 2.3$ bits

Codeword Length	Codeword	X	Probability				
2	01	1	0.25	0.3	0.45	0.55	1
2	10	2	0.25	0.25	0.3	0.45	
2	11	3	0.2	0.25	0.25		
3	000	4	0.15	0.2			
3	001	5	0.15				

- Ternary alphabet

Codeword	X	Probability		
1	1	0.25	0.5	1
2	2	0.25	0.25	
00	3	0.2	0.25	
01	4	0.15		
02	5	0.15		

$L = 1.5$

- when $D \geq 3$, there may not be sufficient number of symbols so that we can combine D at a time

- Add dummy symbols with probability 0 s.t. total number of symbols $1 + k(D-1)$ for the smallest integer k

Codeword	X	Probability			
1	1	0.25	0.25	0.5	1.0
2	2	0.25	0.25	0.25	
01	3	0.2	0.2	0.25	
02	4	0.1	0.2		
000	5	0.1	0.1		
001	6	0.1			
002	Dummy	0.0			

$L = 1.7$

Huffman coding for weighted codewords

- Solving $\min \sum w_i l_i$ instead of $\min \sum p_i l_i$

X	Codeword	Weights			
1	00	5	8	10	18
2	01	5	5	8	
3	10	4	5		
4	11	4			

20 Questions

- Determine the value of a random variable X
- Know distribution of the random variable $p_1, \ldots, p_m$
- Want to ask minimum number of questions
- Receive "yes", "no" answer

index	1	2	3	4	5
p_i	.25	.25	.2	.15	.15

- Native approach

- Start with asking the most likely outcome:
  "Is X = 1"?
  "Is X = 2"?
  ⋮

- Expected number of binary questions = 2.55

- If we can ask any question of the form "is X $\in A$"

- Huffman code

index	1	2	3	4	5
p_i	.25	.25	.2	.15	.15
Code	01	10	11	000	001

- Q1: is X $=$ 2 or 3?

- Q2: if answer "Yes": is X $=$ 2; if answer "No": if X $=$ 1 and so on.

- $E(Q) = 2.3 = H(X)$

Slice code

- What if we can only ask questions with the form "is $X > a$" or "is $X \leq a$" for some a
- Huffman code may not satisfy this requirement
- But can find a set of code words resulting in a sequence of questions like these
- Take the optimal code lengths found by Huffman codes
- Find codewords from tree

index	1	2	3	4	5
Code	00	01	10	110	111

Huffman code and Shannon code

- Shannon code $l_i = \lceil \log 1/p_i \rceil$
- Shannon code can be much worse than Huffman code (last lecture)
- Shannon code can be shorter than Huffman code:

  (1/3, 1/3, 1/4, 1/12) result in Huffman code length (2, 2, 2, 2) or (1, 2, 3, 3); but $\lceil \log 1/p_3 \rceil = 2$

- Huffman code is shorter an average

$$\sum p_i l_{i,\text{Huffman}} \leq \sum p_i l_{i,\text{Shannon}}$$

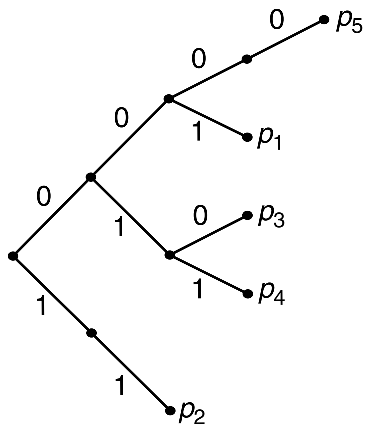
  but $l_{i,\text{Huffman}} \leq l_{i,\text{Shannon}}$ may not be true

Optimality of Huffman codes

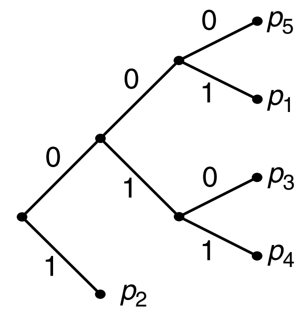
- Huffman code is not unique: investing the bits or exchanging two codewords of the same length

- Proof based on the following lemmas

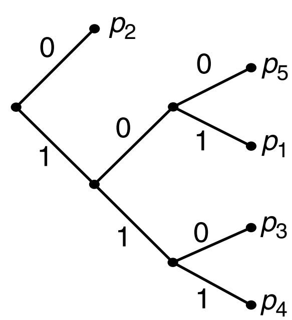
(1) if $p_j \geq p_k$, then $l_j \leq l_k$
(2) Two longest codewords are of the same length
(3) Two longest codewords differ only in the last bit


0
p_5
0
0
1
p_1
0
0
p_3
1
1
p_4
1
1
p_2

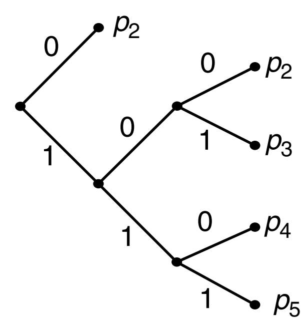
(a)


0
p_5
0
1
p_1
0
0
p_3
1
1
p_4
1
p_2

(b)


p_2
0
0
p_5
0
1
p_1
1
1
0
p_3
1
p_4

(c)


p_2
0
0
p_2
0
1
p_3
1
1
0
p_4
1
p_5

(d)

Proof idea

- Induction
- Consider we have found optimal codes for

$$C_m^*(p) = (p_1, \ldots, p_m)$$

$$C_{m-1}^*(p') = (p_1, \ldots, p_{m-2}, p_{m-1} + p_m)$$

- First, $p' \to p$:

  expand the last codewords $C^*_{m-1}(p')$ for $p_{m-1} + p_m)$ by adding 0 and 1

$$L(p) = L^*(p') + p_{m-1} + p_m$$

- Then, $p \to p'$:

  merging the codeswords for the two lowest-probability symbols

$$L(p') = L^*(p) - p_{m-1} - p_m$$

- $L(p') + L(p) = L^*(p') + L^*(p)$, since $L^*(p') \le L(p')$, $L^*(p) \le L(p)$

$$L^*(p') = L(p'), \quad L^*(p) = L(p)$$

- Huffman code has shortest average code length in that

$$L_{\text{Huffman}} \leq L$$

for any prefix code.

$$H(X) \leq L_{\text{Huffman}} < H(X) + 1$$

- Redundancy $=$ average Huffman codeword length - $H(X)$
- Redundancy of Huffman coding is at most [Gallager 78]

$$p_1 + 0.086$$

where p_1 is the probability of the most-common symbol

Kolmogorov complexity

- So far the object X has been a random variable drawn from $p(x)$
- Descriptive complexity of X is entropy, since $\lceil \log 1/p(x) \rceil$ is the number of bit required to describe x using Shannon code
- Can we extend this notion for non-random object
- Kolmogorov complexity: the length of the shortest binary computer program (algorithm) to describe the object
- Considered a way of thinking: it may take infinitely long to find such minimal program

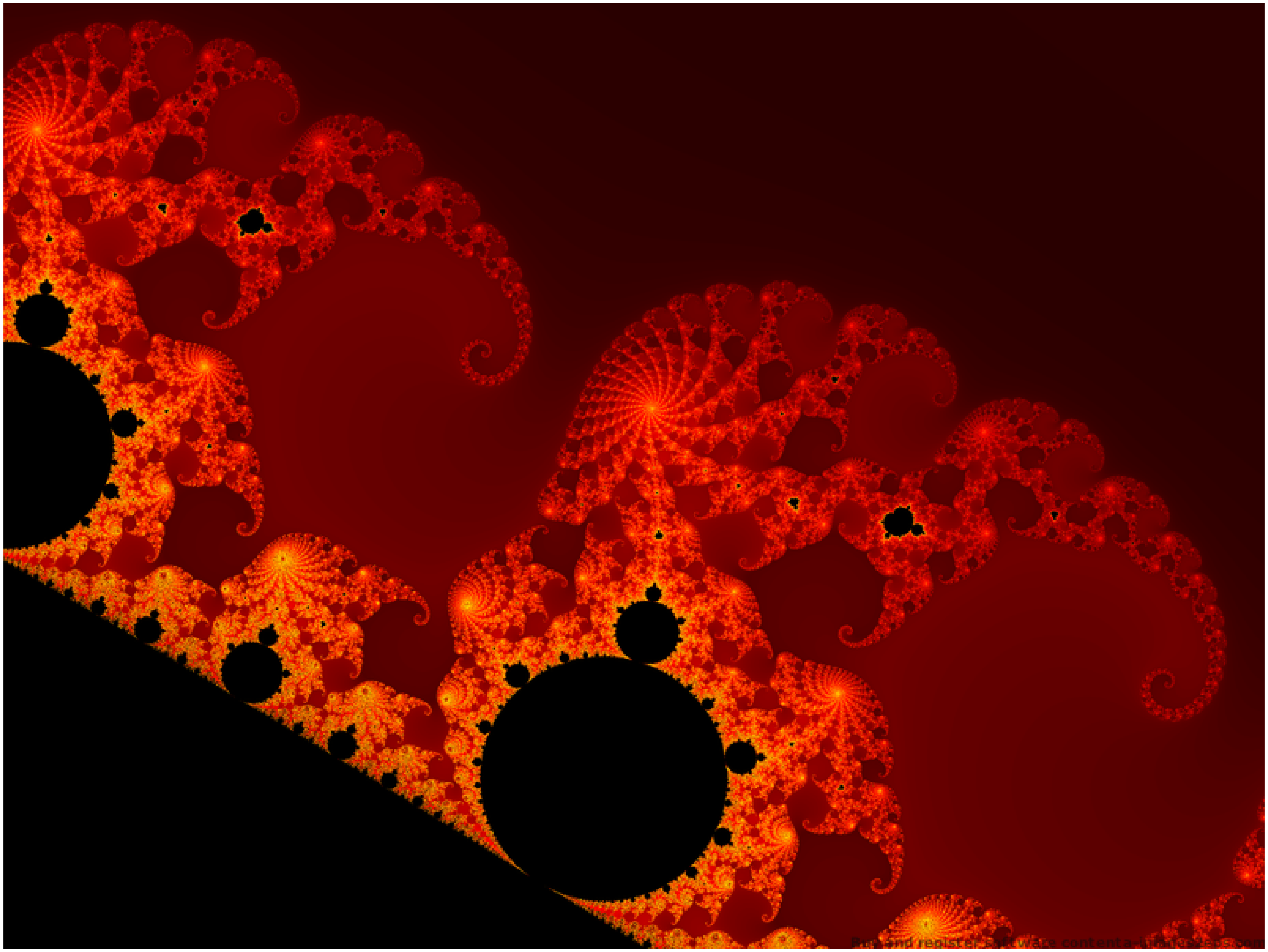
- Kolmogorov complexity of $K_U(x)$ of a string x with respect to a universal computer U is defined as

$$K_U(x) = \min_{p:U(p)=x} l(p)$$

- Example: "Print out the first 1,239,875,981,825,931 bits of the square root of e"

- Using ASCII (8 bits per character), this is 73 character

- Most number of this length has a Kolmogorov complexity of nearly 1,239,875,981,825,931 bits (say, a i.i.d. sequence of random 0, 1s)



Incompressible sequence

- An infinite string x is incompressible if

$$\lim_{n \to \infty} \frac{K(x_1, \ldots, x_n | n)}{n} = 1$$

- The proportion of 0's and 1's in any incompressible strings are almost equal, i.e., i.i.d. Bernolli (1/2) sequence

- Optimal codes form an incompressible sequence

$$C(x_1)C(x_2)\ldots C(X_n)$$

(since its complexity is nearly $nH(1/2)$)

Occam's razor

- "The shortest explanation is the best."
- Law of parsimony
- In many areas of scientific research, choose the simplest model to describe data
- Minimum description length (MDL) principle:

  $X_1, \cdots, X_n$ i.i.d. from $p(x) \in \mathcal{P}$

$$\min_{p \in \mathcal{P}} K(p) + \log \frac{1}{p(X_1, \cdots, X_n)}$$

Huffman coding and compressed sensing

- Now we are often interested in sparse representation of data y

$$\min_a \|y - \sum_i a_i d_i\|^2 + \|a\|_1$$

- Related to MDL principle
- Principle of Huffman coding has also been used in sequential compressed sensing:

SEQUENTIAL ADAPTIVE COMPRESSED SAMPLING VIA HUFFMAN CODES, Aldroubi, 2008

Summary

- Huffman code is a "greedy" algorithm that it combines two least likely symbols at each stage
- This local optimality ensures global optimality
- Minimum description length
- Kolmogorov complexity