stream codes (Mackay chapter 6)

problems with Huffman codes

changing ensembles

Huffman assumes iid, real world data is non-iid

the extra bit: we know Huffman gives $H(X) \leq \mathbb{E}[L_C(X)] \leq H(X) + 1$

a	0.001	00000
b	0.001	00001
c	0.990	1
d	0.001	00010
e	0.001	00011
f	0.001	0100
g	0.001	0101
h	0.001	0110
i	0.001	0111
j	0.001	0010
k	0.001	0011

If one common + many uncommon symbols, the +1 bit is very bad

$\mathbb{E}[\text{length}] = 1.034$

$H(X) = 0.114$

$\mathbb{E}[L]/H(X) = 9$

the guessing game

14 1 11 1 1 1 1 1 1 2 3 1 1 1 1 1

M A J O R I T Y O F P E O P L E -

26 5 ~6 1

F O U R I N T E N - H A T E M A T H

R

how to model data sources

- iid source, known distrⁿ – opt code is Huffman
- iid source, unknown distrⁿ – need to learn distⁿ (inference)
- non iid source, known distrⁿ – separate probabilistic model from encoding/learning/comm ... (eg. arithmetic coding)

easy to learn
⋮
hard to learn

- non iid source, unknown distrⁿ – 'agnostic' setting
  - combine modeling + task
  - universal codes, online learning, 'adversarial' methods, etc.

two approaches to stream coding (for comparison - see Mackay)

(known model, non iid)

Arithmetic Coding - needs to know model

(djvu, pp3)

$$L(D) \leq H(D) + 2 \text{ bits}$$

Dictionary Coding (LZW) ('universal', i.e. unknown model)

(gzip)

$$\lim_{D \to \infty} L(D) \approx H(D) \text{ without knowing model}$$

arithmetic coding

idea - 'represent every database as a single real number'

$$D \equiv \text{'to be or to be } \boxtimes \text{'} \xrightarrow{AC} 0.3141592652$$

decode

- $\text{If } D = X_1 X_2 \ldots X_n \boxtimes$

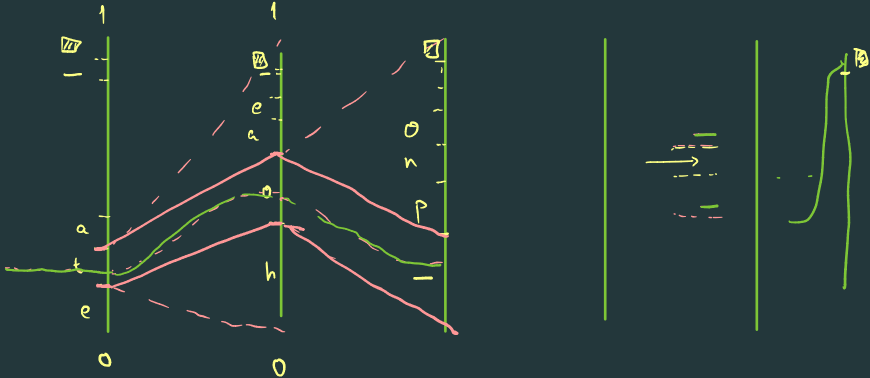
$$\in \mathcal{X} \cup \{\boxtimes\}$$

Need - $$P_t \equiv \underbrace{\mathbb{P}[X_t = x \mid X_1, X_2, \ldots, X_{t-1}]}_{\text{probabilistic model}}$$

Eg - Markovian model - $P_t \equiv \mathbb{P}[X_t = x \mid X_{t-1}]$ (bigram)

arithmetic coding

to_be_o



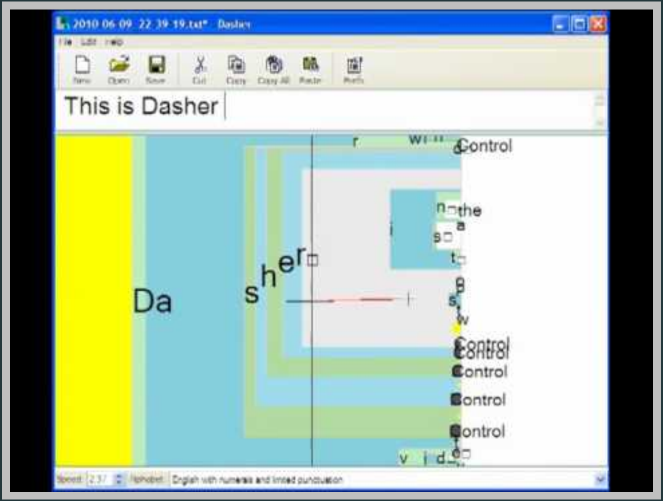
$P_1 = \mathbb{P}[X_1|\phi]$ $\quad$ $\mathbb{P}[X_2|X_1=t]$ $\quad$ $\mathbb{P}[X_3|\text{'to'}]$

$$\underbrace{0.3 \qquad 0.14 \qquad 0.159}_{0.31415\ \ldots}$$

arithmetic coding

- for decoding – decoder knows the model
  - runs the encoder 'in reverse'
  -

application of arithmetic coding beyond compression



https://www.youtube.com/watch?v=nr3s4613DX8

Lempel-Ziv-Welch coding

source data - 1011010100010

source substrings	λ	1	0	11	01	010	00	10
$s(n)$	0	1	2	3	4	5	6	7
$s(n)_{\text{binary}}$	000	001	010	011	100	101	110	111
(pointer, bit)		(, 1)	(0, 0)	(01, 1)	(10, 1)	(100, 0)	(010, 0)	(001, 0)

Eg -

$$D = \overbrace{\underbrace{1}_{01}}^{1}\ \overbrace{\underbrace{11}_{10}}^{2}\ \overbrace{\underbrace{111}_{11}}^{3}\ \overbrace{\underbrace{1111}_{100}}^{4}\ \cdots \quad k \approx k^2/2 \quad n \text{ bits}$$

posn 01 10 11 100

$(s(01)+1)$ $(s(10)+1)$ $(s(11)+1)$

$$\approx \sqrt{n} \log n \text{ bits}$$

Formal guarantee $\lim_{n\to\infty} \frac{L(D_n)}{n} \approx H(X)$

till now – Chs 1,2 (intro) + Ch 4,5,6 (source coding)

Plan

- today – information theory for dependant r.v. (Ch 8)
  - channel coding (Ch 9)
- next class – Shannon's channel coding thm (Ch 10)

In 2 classes – Intro to Bayesian stats (Ch 2 of Mackay, Ch 1 of Bishop)