1 Intro to Information Theory

(Be glad this is not the real info theory course :)

What is information?

Need to distinguish between common ideas

Listen to a modem: to you its noise, to the downstream modem its info

1.1 Basic Model

Receiver wnats to know what sender sent

2 different processes here.

1. sender generates "information"

2. the block in figure transform it

Assume that sender can only send discrete messages $m_n, n = 1, 2, ...$

Basic idea:less probable messages inherently carry more "information"

$X_i \in a_j$ source letters $Prob(a_j) = p_j$

If X_i indept. then source is memoryless.

1.2 Thought experiment

1. 32 events all equally likely, all coded by a sequence of binary digits

   How many digits do you need to code a long sequence of events (on average)?

2. 32 events but on 2 can happen(and these are equally likely)

   How many digits on average to code long sequence of events.

3. Messages $m_n \quad n = 0, 1, 2, ...$ each with prob $p(m_n) = (1 - \alpha)\alpha^n$. How many bits on average here?

   $c(m_0) = 1 \qquad \rightarrow l(c(m_0)) = 1$

   $c(m_1) = 01 \qquad = 2$

   $c(m_2) = 001 \qquad = 3$

etc

$$E(l) = \sum_{n=0}^{\infty} (1-\alpha)\alpha^n (n+1) = \frac{\alpha}{1-\alpha} + 1$$

say $\alpha = 1/2$

$$E(l) = 2$$

Infinite set of nonzero prob messages, but average code length=2

1.3 Entropy

Think of it as a measure of disorder

1. If you have no idea what should happen it should be maximal
2. If you know exactly what's going to happen it should be zero.
3. If events are indept., then the entropy should be sum of individual

$$H(X) = -\sum_{i=1}^{N} p_i \log p_i \qquad \text{or} \qquad \overbrace{\sum_i p_i \log p_i}^{\text{countably infinite}}$$

I $H(X) =$ s bits (use $\log_2$)

II $H(X) = 1$

III

$$\begin{aligned} H(X) &= -\sum_{n=0}^{\infty} (1-\alpha)\alpha^n \log(1-\alpha)\alpha^n \\ &= -(\log(1-\alpha) + \log\alpha)(1-\alpha)\sum_{n=0}^{\infty} \alpha^n n \\ &= \frac{\alpha}{1-\alpha}(\log(1-\alpha) + \log\alpha) \end{aligned}$$

$\alpha = 1/2 \qquad \to H(X) \equiv 2$

Idea

Entropy might have something to do with how compactly you can code the output of a source **Def**.

Self information is $I(p_j) = -\log p_j$ of source digit a_j

Def.

Entropy $H(I) = E(I(p_j)) = -\sum_j p_j \log p_j$

Def.

Joint entropy

$$H(X,Y) = -\sum p(x,y)\log p(x,y)$$

In general

$$H(X) = \sum_{\underline{x}} p(x_1, x_2, ..., x_n)\log p(x_1, ...x_n)$$

Show using $\log x \leq x - 1$ that $H(X) \leq \log N$

$$H(X) - \log N \leq \sum p_r \log \frac{1}{p_r N}$$

etc.

Def.

Conditional entropy

$$H(X|Y) = \sum_{x,y} p(x,y)\log p(x|y)$$

and from decomposition of $p(x,y)$

$$H(X|Y) = \sum p(x)p(y|x)\log p(x|y)$$

In general

$$H(X_n|X_{n-1}...X_1) = \sum_{\underline{x}} p(\underline{x})\log p(x_n|x_{n-1}...x_1)$$

which implites that

$$H(\underline{X}) = H(X_1) + H(X_2|X_1)... + H(X_n|X_{n-1}...X_1)$$

Usually seen as

$$H(XY) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

Notice that if X indept. Y then $H(XY) = H(X) + H(Y)$ why?

1.4 Entropy Rate

Now lets say those X_i's are correlated

$$H(X) = \lim_{n\to\infty} H(X_n|X_{n-1}...X_1)$$

i.e. how much is new under the sum. What does history tell you about the present

For stationary processes we also have

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(\underline{X}^n)$$

Aside Memory source $\Rightarrow H(x) = H(X)$

1.5 Source Coding Thm

Let $R = \sum_x l(x)p(x) = E(l)$

A source with entropy rate H can be coded with arbitrarily small probability of coding failure at any rate R(bits/source output) if R¿H. However, if $R < H$, the failure probability is bounded away from zero (no matter how hard you try)

Aside

This makes the code for (III) from before OPTIMAL! Can't do any better

Why you ask?

Look at long sequence of source letters and only code the "typical ones"

If $\underline{X}$ is typed then by LLN

$$p(\underline{X} \neq \underline{x}) \approx \prod_{j=1}^{J} p_j^{np_j} \qquad \text{(J source letters)}$$

because

typical sequences will have (on average)

$\approx mp_j$ occurrences of letter j

Well

$$p_j^{np_j} = 2^{\log p_j^{np_j}} = 2^{np_j \log p_j}$$

So

$$P(\underline{X} = \underline{x}) \approx \prod_{j=1}^{J} 2^{np_j \log p_j} = 2^{n \sum_j p_j \log p_j} = 2^{-nH(X)}$$

HAND WAVE $\Rightarrow$ prob sequence is not typical for large n is negligible

Therefore, number of typical sequences $\approx 2^{nH(X)}$

So code each of the typical with $nH(X)$ bits AND ignore the others!

$$\frac{\text{code digits}}{\text{source digit}} = H(X)$$

1.6 Data Compression

Take source letter sequences and code so that you use as few bits on average as you need to, i.e. Video/Voice Audio compression $\rightarrow$ more playing time on CD because you can pack more signals into it.

Data compression $\Rightarrow$ compress (unix) zip(PC) etc.

HOW!!!??? <u>F</u>ORMALITIES

need $a_i \rightarrow c_i$ codeword $l(c_i) = l_i$, where $l(c_i)$ is length So, $x_1x_2x_3...x_n \Rightarrow c(x_1)c(x_2)...c(x_n)$

$\Rightarrow$ average length per source letter $\frac{1}{n}\sum_{i=1}^{n} l(c_(x_i))$

This means

$$E(L) = \lim_{n\rightarrow\infty} \frac{1}{n} \sum (c(x_i)) = \sum_x p(x) l(c(x))$$

Our job find mapping $a_j \rightarrow c_j$ which minimizes $E(L)$ (also called $\overline{R}$

Can be shown that $\sum D^{-lx} \leq 1$ for any code, where D is code alphabet size. This can be used to show that the optimum code satisfies

$$H(X) \leq E(L) \leq H(X) + 1$$

1.7 HUFFMAN CODES=FUN

Finite number of letters in the alphabet $a_1 a_2 .... a_J$

$$Prob(a_j) = p_j$$

Aside

(Prefix code) - courtesy

1	100000
10	have to wait until end
00	

1.7.1 Algorithm

1. order the probabilities
2. combine the 2 lowest to generate a secondary distribution
3. prepare a 0 or 1 to the two corresponding codewords
4. goto 1

HUFFMAN CODES=FUN

How can you construct a code for a given source distribution such that L is minimized?

Huffman algorithm (D-ary)

1. order the probabilities
2. combine the D lowest probs to generate a secondary distribution
3. Prepared 0,1,...,D-1 to the two corresponding codewords as combined
4. goto 1

1.7.2 Examples

Ex.1

$a - 10$

$b - 01$

$c - 00$

$d - 111$

$e - 110$

$L = 2/4 + 2/4 + 2/5 + 3(.15) + 3(.15) = 2.3$

Ex.2

$a - 1$

$b - 0$

$c - 22$

$d - 21$

$e - 20$

$L = 1/4 + 1/4 + 2/5 + 2(1.5) + 2(1.5) = 1.5$

Ex.3

Code tree is incomplete!

We need to combine 3 at last step. So add DUMMIES

Grouping preference is

$x \to x - D + 1 \to x - 2D + 2 \to x - k(D - 1) \to 1$

so $x - 1 = k(D - 1) \to (x - 1)$ is a multiple of $(D - 1)$ at last stage

Previous case $x = 6, x - 1 = 5$ not a multiple of $D - 1 = 2$, so add 1 dummy.

$L = .25 + .25 + 2/5 + 2/10 + 3/10 + 3/10 + 0 = 1.7$

Tree complete!

Extension

Code blocks of k source letters

Ex. $a_1a_2a_3$ 3 source letters K=3

$$\left.\begin{array}{l} Prob(a_1a_1a_1) = p_1^3 \\ Prob(a_1a_1a_2) = p_1^2p_2 \\ \text{etc.} \\ \text{9 possibilities} \end{array}\right\} \text{New source distrib}$$

Clearly,

$$H(X^3) \leq \overline{R_3} \leq H(X^3) + 1$$

If n - letter blocks

$$H(X^n) \leq \overline{R_n} \leq H(X^n) + 1$$
$$\Rightarrow \frac{H(X^n)}{n} \leq \overline{R} = \frac{\overline{R}}{n} \leq \frac{H(X^n)}{n} + \frac{1}{n}$$

So for sources with memory, $\overline{R}$ is bounded by $H(X)$ the entropy rate.

1.8 Lempel-Ziv

Huffman is OK but

What if you don't have $p(a_j)$?

What if you need to code more than a few letters at a time?

Basic idea: Work from left $\rightarrow R$ in two passes

1. Identify strings which have not come before to build up a codebook of phrases
2. number the phrases and code the main string using those phrases

Ex. (from book)

Could do 2 things:

1. Dumb: assume decoder knows the phrases ahead of time and simply send phrases numbers dumb because

- the codebook is as big as the sequence
- each string has a different codebook

2. Smart: Code as (phase II, new bit)
   in example, (define "start phrase" as 0000) start phrase has zero length
   (0000,0)(0000,1)(0001,0)(0011,1)(0010,0)(0011,0)....

Assuming the decoder know the maximum number of phrases (or as in book, source and destination agree on what to throw out when)
then the destination can reconstruct the code

First time it sees 0000 it knows that the first bit in the sequence follows $\rightarrow$ which means it now knows the first phrase
Held sequence must be next phrase and so on.
Why does this work?
If ssequence has structure, phrases get longer and longer, because what comes next has probably been seen before

END