

Entropy, Coding and Data Compression

Data vs. Information

- “yes,” “not,” “yes,” “yes,” “not” “not” ...
- In ASCII, each item is $3 \cdot 8 = 24$ bits ***of data***
- But if the only possible answers are “yes” and “not,” there is only one bit ***of information*** per item

Compression = Squeezing out the "Air"

- Suppose you want to ship pillows in boxes and are charged by the size of the box



- To use as few boxes as possible, squeeze out all the air, pack into boxes, fluff them up at the other end
- **Lossless** data compression = pillows are perfectly restored
- **Lossy** data compression = some damage to the pillows is OK (MP3 is a lossy compression standard for music)
- Loss may be OK if it is below human perceptual threshold
- Entropy is a measure of limit of **lossless** compression

Example: Telegraphy

Source English letters -> Morse Code

Sender: from Hokkaido

D → -..



-..



-.. → D

Receiver: in Tokyo

Coding Messages with Fixed Length Codes

- Example: 4 symbols, A, B, C, D
- A=00, B=01, C=10, D=11
- In general, with n symbols, codes need to be of length $\lg n$, rounded up
- For English text, 26 letters + space = 27 symbols, length = 5 since $2^4 < 27 < 2^5$
(replace all punctuation marks by space)

Uniquely decodable codes

- If any encoded string has only one possible source string producing it then we have unique decodability
- Example of uniquely decodable code is the *prefix code*

Prefix Coding (Instantaneous code)

- A **prefix code** is defined as a code in which *no* codeword is the prefix of some other code word.
- A prefix code is *uniquely decodable*.

Example

Source Symbol	Prefix Code		
	Code A Symbol Codeword	Code B Symbol Codeword	Code C Symbol Codeword
s_0	0	0	0
s_1	1	10	01
s_2	00	110	011
s_3	11	111	0111

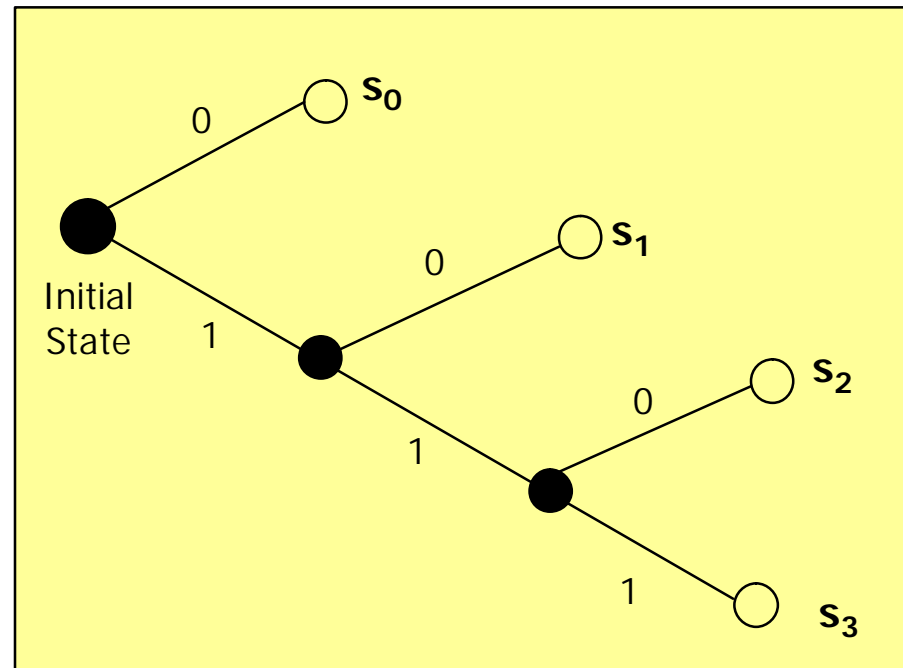
Uniquely Decodable Codes

Decoding of a Prefix Code

Example

Code B	
Source Symbol	Symbol Codeword
s_k	c_k
s_0	0
s_1	10
s_2	110
s_3	111

Decision Tree for Code B



- Example : Decode 1011111000
- Answer : $s_1s_3s_2s_0s_0$

Prefix Codes

Only one way to decode left to right when message received

Example 1

Symbol	A	B	C	D
Probability	.7	.1	.1	.1
Code	0	100	101	110

Received message:

00001000000000011000000000100
AAAA B AAAAAAA D AAAAAAA B

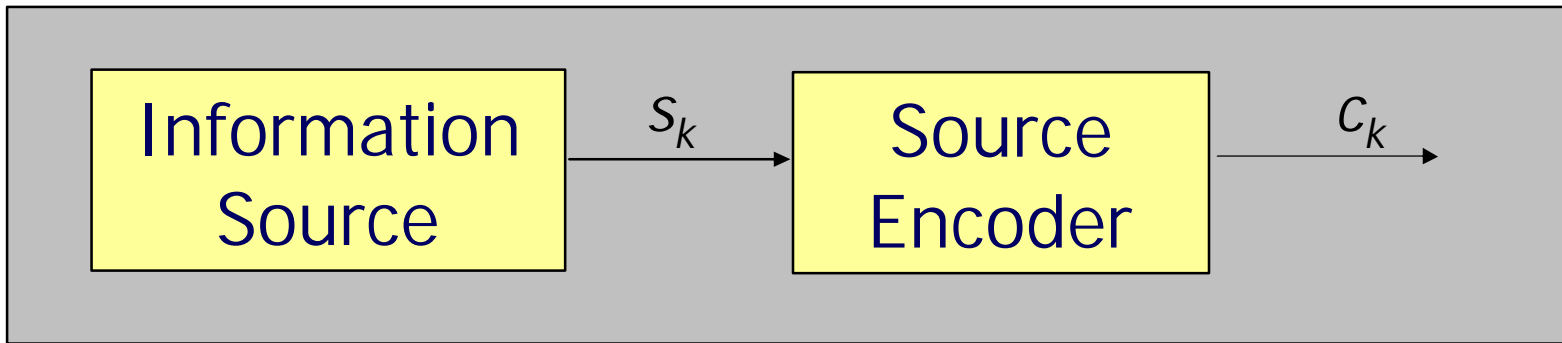
Prefix Codes

Example 2

Source Symbol s_k	Code E		
	Symbol Codeword c_k		
A	0		
B	100		
C		110	
D		11	

- IS CODE E A PREFIX CODE?
 - NO
 - WHY?
 - Code of D is a prefix to code of C

Average Code Length



- Source has K symbols
- Each symbol s_k has probability p_k
- Each symbol s_k is represented by a codeword c_k of length l_k bits
- **Average codeword length**

$$L = \sum_{k=0}^{K-1} p_k l_k$$

Shannon's First Theorem: The Source Coding Theorem

$$L \geq H(S)$$

- The outputs of an information source cannot be represented by a source code whose average length is less than the source entropy

Average Code Length

Example

Average bits per symbol:

$$L = .7 \cdot 1 + .1 \cdot 3 + .1 \cdot 3 + .1 \cdot 3 = 1.6$$

bits/symbol (down from 2)

Another prefix code that
is better

$$L = .7 \cdot 1 + .1 \cdot 2 + .1 \cdot 3 + .1 \cdot 3 = 1.5$$

A	B	C	D
.7	.1	.1	.1
0	100	101	110

A	B	C	D
.7	.1	.1	.1
0	10	110	111

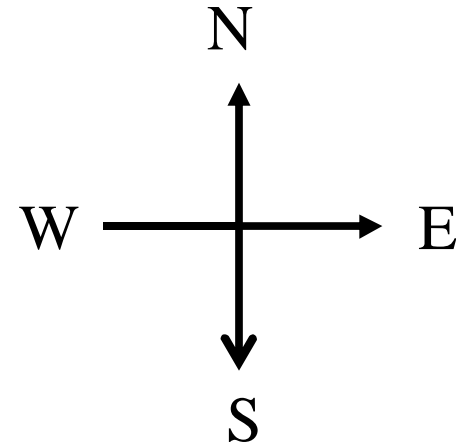
Source Entropy Examples

Robot Example

- 4-way random walk

$$\text{prob}(x = S) = \frac{1}{2}, \text{prob}(x = N) = \frac{1}{4}$$

$$\text{prob}(x = E) = \text{prob}(x = W) = \frac{1}{8}$$



$$H(X) = -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{4} \log_2 \frac{1}{4} + \frac{1}{8} \log_2 \frac{1}{8} + \frac{1}{8} \log_2 \frac{1}{8}\right) = 1.75 \text{bps}$$

Source Entropy Examples

Robot Example

Prefix Codes

<i>symbol k</i>	P_k	<i>fixed-length codeword</i>	<i>variable-length codeword</i>	
<i>S</i>	0.5	00	0	😊
<i>N</i>	0.25	01	10	
<i>E</i>	0.125	10	110	
<i>W</i>	0.125	11	111	

symbol stream : S S N W S E N N N W S S S N E S S

fixed length: 00 00 01 11 00 10 01 01 11 00 00 00 01 10 00 00

variable length: 0 0 10 111 0 110 10 10 111 0 0 0 10 110 0 0

32bits
28bits

4 bits savings achieved by VLC (redundancy eliminated)

Entropy, Compressibility, Redundancy

- Lower entropy \Leftrightarrow More redundant \Leftrightarrow More compressible
- Higher entropy \Leftrightarrow Less redundant \Leftrightarrow Less compressible

Entropy and Compression

- First-order entropy is theoretical minimum on code length when only frequencies are taken into account

- $L = .7 \cdot 1 + .1 \cdot 2 + .1 \cdot 3 + .1 \cdot 3 = 1.5$

- First-order Entropy = 1.353

A	B	C	D
.7	.1	.1	.1
0	10	110	111

- First-order Entropy of English is about 4 bits/character based on "typical" English texts

Bits

You are watching a set of independent random samples of X

You see that X has four possible values

$P(X=A) = 1/4$	$P(X=B) = 1/4$	$P(X=C) = 1/4$	$P(X=D) = 1/4$
----------------	----------------	----------------	----------------

So you might see output: BAACBADCDADDDA...

You transmit data over a binary serial link. You can encode each reading with two bits (e.g. $A = 00$, $B = 01$, $C = 10$, $D = 11$)

2 bits on average per symbol

0100001001001110110011111100...

Fewer Bits

Someone tells you that the probabilities are not equal

$P(X=A) = 1/2$	$P(X=B) = 1/4$	$P(X=C) = 1/8$	$P(X=D) = 1/8$
----------------	----------------	----------------	----------------

Is it possible...

...to invent a coding for your transmission that only uses **1.75** bits on average per symbol. How?

Fewer Bits

$$P(X=A) = 1/2 \quad P(X=B) = 1/4 \quad P(X=C) = 1/8 \quad P(X=D) = 1/8$$

It's possible...

...to invent a coding for your transmission that only uses 1.75 bits on average per symbol.

A	0
B	10
C	110
D	111

(This is just one of several ways)

Fewer Bits

Suppose there are three equally likely values...

$$P(X=A) = 1/3 \quad P(X=B) = 1/3 \quad P(X=C) = 1/3$$

Here's a naïve coding, costing 2 bits per symbol

A	00
B	01
C	10

Can you think of a coding that would need only 1.6 bits per symbol on average?

In theory, it can in fact be done with 1.58496 bits per symbol.

Kraft-McMillan Inequality

$$\sum_{k=0}^{K-1} 2^{-l_k} \leq 1$$

- If codeword lengths of a code satisfy the Kraft McMillan's inequality, then a prefix code with these codeword lengths **can be** constructed.
- For code D
 - $2^{-1} + 2^{-2} + 2^{-3} + 2^{-2} = 9/8$
 - This means that Code D **IS NOT A PREFIX CODE**

Example

Source Symbol s_k	Code D	
	Symbol Codewor d C_k	Codeword Length l_k
s_0	0	1
s_1	10	2
s_2	110	3
s_3	11	2

Use of Kraft-McMillan Inequality

- We may use it if the number of symbols are large such that we cannot simply by inspection judge whether a given code is a prefix code or not
- **WHAT Kraft-McMillan Inequality Can Do:**
 - It can determine that a given code IS NOT A PREFIX CODE
 - It can identify that a prefix code could be constructed from a set of codeword lengths
- **WHAT Kraft-McMillan Inequality Cannot Do:**
 - It cannot guarantee that a given code is indeed a prefix code

Example

Source Symbol s_k	Code E	
	Symbol Codeword c_k	Codeword Length l_k
s_0	0	1
s_1	100	3
s_2	110	3
s_3	11	2

- For code E
 - $2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1$ and hence satisfy Kraft-McMillan inequality
- IS CODE E A PREFIX CODE?
 - NO
 - WHY?
 - s_3 is a prefix to s_2

Code Efficiency ?

$$\mathbf{h} = \frac{H(S)}{L}$$

- An efficient code means ? $\rightarrow 1$

Examples

Source Symbol s_k	Symbol Probability p_k	Code I		Code II	
		Symbol Codeword c_k	Codeword Length l_k	Symbol Codeword c_k	Codeword Length l_k
s_0	1/2	00	2	0	1
s_1	1/4	01	2	10	2
s_2	1/8	10	2	110	3
s_3	1/8	11	2	111	3

- Source Entropy

- $$H(S) = 1/2 \log_2(2) + 1/4 \log_2(4) + 1/8 \log_2(8) + 1/8 \log_2(8)$$

$$= 1 \frac{3}{4} \text{ bits/symbol}$$

Code I

$$L = 2 \times \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} \right) = 2$$

$$h = \frac{7/4}{2} = 0.875$$

Code II

$$L = \left(1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 3 \times \frac{1}{8} \right) = \frac{7}{4}$$

$$h = \frac{7/4}{7/4} = 1$$

For a Prefix Code

- Shannon's First Theorem

$$H(S) \leq L < H(S) + 1$$

$$L = H(S) \quad \text{if} \quad p_k = 2^{-l_k} \quad \forall k$$



What is the Efficiency ? ?
?=1

if $p_k \neq 2^{-l_k}$ for some $k \Rightarrow ? < 1$

However, we may increase efficiency by extending the source

Increasing Efficiency by Source Extension

- By extending the source we may potentially increase efficiency
- The drawback is
 - Increased decoding complexity

$$H(S^n) \leq L_n < H(S^n) + 1$$

$$nH(S) \leq L_n < nH(S) + 1$$

$$H(S) \leq \frac{L_n}{n} < H(S) + \frac{1}{n}$$

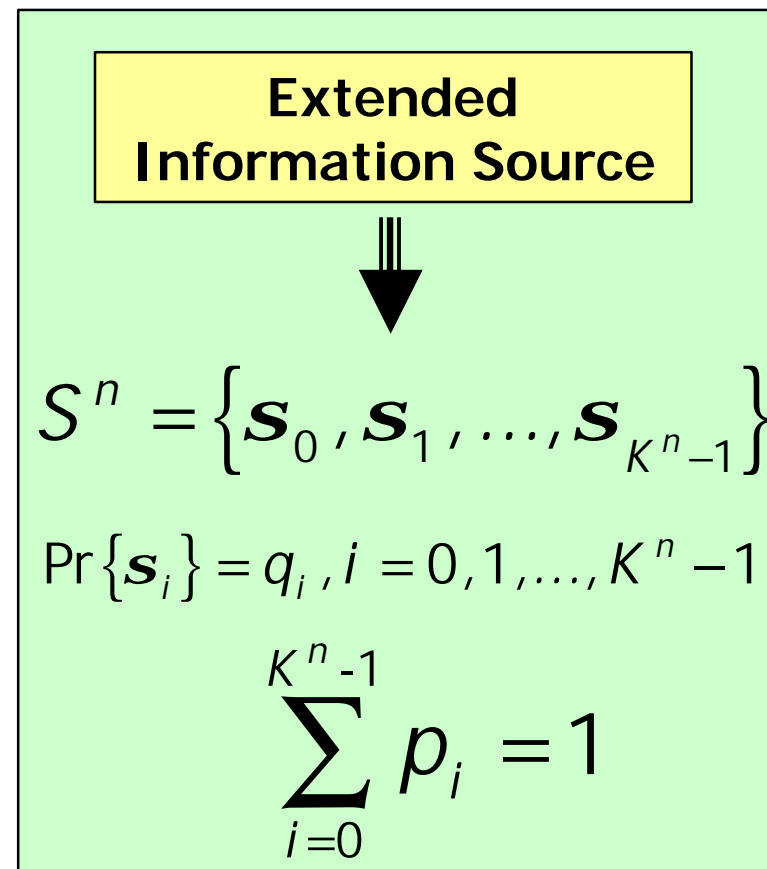
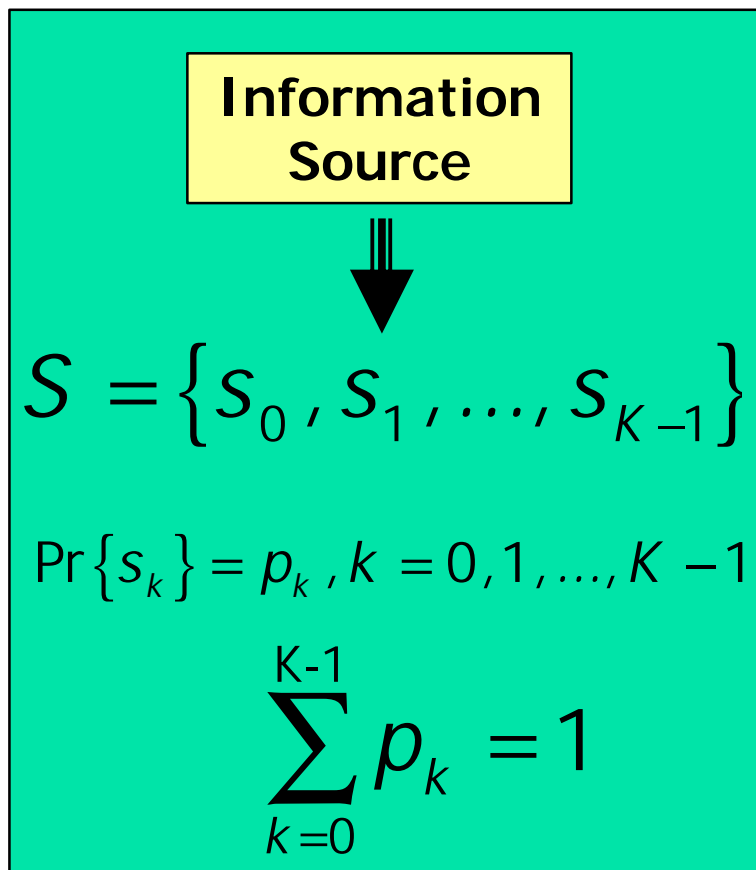
$$\mathbf{h} = \frac{H(S)}{L_n/n}$$

$$\mathbf{h} \rightarrow 1 \text{ when}$$

$$n \rightarrow \infty$$

Extension of a Discrete Memoryless Source

- Treats Blocks of n successive symbols



Example

- $S = \{s_0, s_1, s_2\}$, $p_0 = 1/4$, $p_1 = 1/4$, $p_2 = 1/2$
- $H(S) = (1/4)\log_2(4) + (1/4)\log_2(4) + (1/2)\log_2(2)$

$H(S) = 3/2$ bits

Second-Order Extended Source

Symbols of S^2	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
Sequence of Symbols from S	$s_0 s_0$	$s_0 s_1$	$s_0 s_2$	$s_1 s_0$	$s_1 s_1$	$s_1 s_2$	$s_2 s_0$	$s_2 s_1$	$s_2 s_2$
$P\{s_i\}, i=0,1,\dots,8$	1/16	1/16	1/8	1/16	1/16	1/8	1/8	1/8	1/4

By Computing: $H(S^2) = 3$ bits

Summery

- **Source Encoding**
 - **Efficient**
representation of information sources
- **Source Coding Requirements**
 - Uniquely Decodable Codes
- **Prefix Codes**
 - No codeword is a prefix to some other code word

Code Efficiency

$$h = \frac{H(S)}{L}$$

Kraft's Inequality

$$\sum_{k=0}^{K-1} 2^{-l_k} \leq 1$$

Source Coding Theorem

$$H(S) \leq L < H(S) + 1$$

Relax Time

Let's take a little rest and enjoy the following pictures

Carved watermelons

