Lecture 3

Evidence for

\[ h(x) = \log_2 \frac{1}{P(x)} \]

Source coding theorem
How to compress a redundant file?

e.g., $N = 1000$ tosses of a bent coin with $p_1 = 0.1$
How to measure information content?

Claims: 1. The Shannon information content of an outcome

\[ h(x=a_i) = \log_2 \frac{1}{P(x=a_i)} \]

is a sensible measure of information content.

2. The entropy

\[ H(X) = \sum_x P(x) \log_2 \frac{1}{P(x)} \]

is a sensible measure of expected information content.

(sketch h(p))
Point out additivity of h.
Testing 'Shannon information content' claims

Weighing problem

Shannon says

Most 'informative' experiment is the one with maximum entropy
How to measure information content?

Claims:  1. The *Shannon information content* of an outcome
        \[ h(x=a_i) = \log_2 \frac{1}{P(x=a_i)} \]
        is a sensible measure of information content.

2. The *entropy*
   \[ H(X) = \sum_x P(x) \log_2 \frac{1}{P(x)} \]
   is a sensible measure of expected information content.

3. Source coding theorem –
   *N* outcomes from a source *X* can be compressed into roughly *NH*(X) bits.
The Bent Coin Lottery

A coin with \( p_1 = 0.1 \) will be tossed \( N = 1000 \) times.
The outcome is \( x = x_1 x_2 \ldots x_N \).
e.g., \( x = 000001001000100\ldots00010 \)
You can buy any of the \( 2^N \) possible tickets for \( £1 \) each,
before the coin-tossing.
If you own ticket \( x \), you win \( £1,000,000,000 \).

Q To have a 99% chance of winning, at lowest possible cost,
which tickets would you buy?
- And how many tickets is that?
Express your answer in the form \( 2^{(\ldots)} \).

Lottery tickets available

\[
\begin{align*}
0000000000 & \ldots 00000 \\
0000000000 & \ldots 00001 \\
0000000000 & \ldots 00010 \\
0000000000 & \ldots 00011 \\
0000000000 & \ldots 00100 \\
0000000000 & \ldots 00101 \\
0000000000 & \ldots 00110 \\
0000000000 & \ldots 00111 \\
00010000001 & \ldots 01000 \\
1111111111 & \ldots 11101 \\
1111111111 & \ldots 11110 \\
1111111111 & \ldots 11111
\end{align*}
\]
Testing 'Shannon information content' claims

Weighing problem

Sixty-three

Submarine
\[ x \in \{0, 1, 2, 3, 4, 5, 6, \ldots, 63\} \]
The thrilling game of "Sixty-three"
The result of "Sixty-three" is:

- $x \geq 32 \rightarrow 1$ Yes
- $x \mod 32 \geq 16 \rightarrow 0$
- $x \mod 16 \geq 8 \rightarrow 0$
- $x \mod 8 \geq 4 \rightarrow 1$
- $x \mod 4 \geq 2 \rightarrow 1$
- $x \mod 2 \geq 1 \rightarrow 1$
1010110 is 42 in binary.
Q1: \( C_1 \)
Q2: \( C_2 \)
Q3: \( C_3 \)
Q4: \( C_4 \)
Q5: \( C_5 \)
Q6: \( C_6 \)

\[ x \mod 32 \geq 16 ? \]
\[ x \mod 16 \geq 8 ? \]
\[ x \mod 8 \geq 4 ? \]
\[ x \mod 4 \geq 2 ? \]
\[ x \mod 2 \geq 1 ? \]
\[ h(c_i) = \log_2 \frac{1}{P(c_i)} = \log_2 2 = 1 \text{ bit} \]
\[ h(c_i) = \log_2 \frac{1}{P(c_i)} = \log_2 2 = 1 \text{ bit} \]

Total Shannon info content gained = 6

The string \( c_1, \ldots, c_6 \)
\[
\frac{1}{p(c_i)} = \log_2 2 = 1 \text{ bit}
\]

The content gained = 6 bits

\[C(x)\] is an encoding of \(x\)

\[C(42) = 101010\]
We were able to give unique binary names to every one of 64 outcomes if each name was 6 bits long.

\[ c(42) = 101010 \]

\[ c(20) = 010100 \]

**Generalize** –

Q If there are $S$ possible outcomes, how many bits long must each name be, if each outcome has a unique name?
Generalization

An outcome from a set of size $S$

Can be communicated in
from a set of size $S$
### Submarine

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Submarine

Unhit subs: 1  Unhit squares: 63

Probabilities:

P(y): 0/1  P(n): 1/1  Entropy: 0
0.0  1.0

Information learnt: 0.022720

Latest outcome: n

P(n): 63/64 = 0.9843  h(n): 0.0227  hTOT: 0.02272

```
disp('BENT-COIN LOTTERY OUTCOMES');
T=1000;
for t=1:T
    ans=input('');
    lottery
endfor
```
\[ P(r_1 = n) = \frac{63}{64} \]

\[ h(r_1 = n) = \log_2 \frac{64}{63} \]

\[ = 0.0227 \]
Submarine

Unhit subs: 1  Unhit squares: 62

Probabilities:
0/1  P(y):
0.0  1/1  P(n):
1.0  Entropy:

Information learnt: 0.045803
Latest outcome: n

P(n): 63/64 = 0.9843  h(n): 0.0227  hTOT: 0.02272
P(n): 62/63 = 0.9841  h(n): 0.0230  hTOT: 0.045800

disp('BENT COIN LOTTERY OUTCOMES');
T=1000;
for t=1:T
    ans=input('-');
    lottery
endfor

20:lewis:/home/mackay/itp/octave/lottery>
20:lewis:/home/mackay/itp/octave/lottery>
20:lewis:/home/mackay/itp/octave/lottery>
\[ P(r_z = n) = \frac{62}{63} \]

\[ h = \log_2 \frac{63}{62} \]

\[ = 0.023 \]
\[ h_{\text{TOT}} = \log_2 \frac{64}{63} + \log_2 \frac{6^3}{62} \]
### Probabilities

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(n) )</td>
<td>0.9756</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9750</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9743</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9736</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9729</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9722</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9714</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9705</td>
</tr>
<tr>
<td>( P(n) )</td>
<td>0.9696</td>
</tr>
</tbody>
</table>

### Information learnt

-\( h(n) = 0.0356 \)
-\( h_{TOT} = 0.67807 \)

### Latest outcome

-\( n \)

### Entropy

-\( h(n) = 0.0356 \)
-\( h_{TOT} = 0.67807 \)

### Code Snippet

```matlab
disp( 'BENT-COIN-LOTTERY OUTCOMES' );
T=1000;
for t=1:T
    ans=input(' - ');
    lottery
endfor
```
Submarine

Unhit subs: 0  Unhit squares: 19

Probabilities:
\[
\begin{align*}
P(y) & : 1/1 \\
P(n) & : 19/20 \\
\end{align*}
\]

Entropy: 0

Information learnt: 6.000000

Latest outcome: y

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Probability</th>
<th>Error</th>
<th>Total Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>27/28 = 0.9642</td>
<td>0.0524</td>
<td>1.24511</td>
</tr>
<tr>
<td>N</td>
<td>26/27 = 0.9629</td>
<td>0.0544</td>
<td>1.29956</td>
</tr>
<tr>
<td>N</td>
<td>25/26 = 0.9615</td>
<td>0.0565</td>
<td>1.35614</td>
</tr>
<tr>
<td>N</td>
<td>24/25 = 0.96</td>
<td>0.0588</td>
<td>1.41503</td>
</tr>
<tr>
<td>N</td>
<td>23/24 = 0.9503</td>
<td>0.0614</td>
<td>1.47643</td>
</tr>
<tr>
<td>N</td>
<td>22/23 = 0.9565</td>
<td>0.0641</td>
<td>1.54056</td>
</tr>
<tr>
<td>N</td>
<td>21/22 = 0.9545</td>
<td>0.0671</td>
<td>1.60769</td>
</tr>
<tr>
<td>N</td>
<td>20/21 = 0.9523</td>
<td>0.0703</td>
<td>1.67807</td>
</tr>
<tr>
<td>Y</td>
<td>1/20 = 0.05</td>
<td>4.3219</td>
<td>6.00000</td>
</tr>
</tbody>
</table>

```
disp('BENT COIN LOTTERY OUTCOMES');
T=1000;
for t=1:T
  ans=input('');
  lottery
endfor
```
\[-\log \frac{33}{32} + \log \frac{32}{31} + \cdots + \log \frac{21}{20} + \log \frac{20}{19}\]
\[ h_{\text{TOT}} = \log_2 \frac{64}{63} + \log_2 \frac{63}{62} + \ldots = 6 \text{ bits} \]
The Bent Coin Lottery

A coin with $p_1 = 0.1$ will be tossed $N = 1000$ times. The outcome is $x = x_1 x_2 \ldots x_N$.

E.g., $x = 000001001000100\ldots00010$

You can buy any of the $2^N$ possible tickets for £1 each, before the coin-tossing.

If you own ticket $x$, you win £1,000,000,000.

Q If you are forced to buy one ticket, which would you buy?

Q To have a 99% chance of winning, at lowest possible cost, which tickets would you buy?

- And how many tickets is that?

Express your answer in the form $2^{(\cdot)}$. 

Lottery tickets available

\[
\begin{array}{c}
0000000000\ldots00000 \\
0000000000\ldots00001 \\
0000000000\ldots00010 \\
0000000000\ldots00011 \\
0000000000\ldots00100 \\
0000000000\ldots00101 \\
0000000000\ldots00110 \\
0000000000\ldots00111 \\
00100000001\ldots01000 \\
\vdots \\
1111111111\ldots11101 \\
1111111111\ldots11110 \\
1111111111\ldots11111 \\
\end{array}
\]
If you buy one ticket

\[
\begin{array}{c}
N = 1000 \\
f = 0.4
\end{array}
\]

Which would you buy?
the all-zeros ticket
one with 900 0's & 100 1's
N = 20  f = 0.1

A
000000...00010

B
\frac{110000000000...0}{2} = 18

C
00100000000000000000...01
To have a 99% chance

\[
\begin{bmatrix}
N = 1000 \\
\beta = 0.1
\end{bmatrix}
\]

which would you buy?
oo...oo, all the \( 1^{st} \) tickets, all the \( 2^{nd} \) tickets
The number of tickets, $R_{\text{max}}$, is approximately 120. Is this number correct?

$2^{100}$ is also around 120. Is this number correct as well?
Variance = N \times f (1-f) \\
= 1000 \times 0.1 \times 0.9 \\
\approx 100 \\
\approx 90 \\
6 \approx 10 \quad 26 =
2.5% → 16 → 26 → 97.5%
\# tickets = 1 + 1000 + \left(\frac{1000}{2}\right) + \ldots + 100
\[ N = 1 \]

\[ (+ (1000)) + (1000) + (1000) + (1000) + (1000) \]
# tickets = 1 + 1000 + \binom{1000}{2} + \cdots + \binom{1000}{100} + \binom{1000}{100}

\log \frac{N!}{(N-r)! r!} \approx \text{NH}(\text{NH}(r)) \approx 2 \text{NH(0.8)}
\[ N \left( \sqrt{f_N + 23} \right) \]
\[ \text{NH}_2(f)(+\sqrt{N}) \]
Compressor

Uncompressor

99% chance of working

Can compress the outcome into $NH(0.9, 0.1)^*$ bits.
3. Source coding theorem –

$N$ outcomes from a source $X$ can be compressed into roughly $N\, H(X)$ bits.

Proved by counting the typical set

Read chapter 4 to see the full proof and the general definition of typicality
How to compress a redundant file, practically?
Project:

Invent a compressor and uncompressor for a source file of $N = 10,000$ bits, each having probability $f = 0.01$ of being a $1$.
Implement them and/or estimate how well your method works.

Also: exercises 5.22, 5.26, 5.27

Reading: Chapters 1-6