(Binary) Symbol Codes

For strings of symbols from alphabet e.g.,
\[ x_i \in \mathcal{A}_X = \{A, C, G, T\} \]

Binary codeword assigned to each symbol

\[
\begin{array}{c|c}
\text{CGTAGATTACAGG} & \text{10111110011101101100100111111} \\
\hline
A & 0 \\
C & 10 \\
G & 111 \\
T & 110 \\
\end{array}
\]

Codewords are concatenated without punctuation

Uniquely decodable

We’d like to make all codewords short
But some codes are not uniquely decodable

\[
\begin{array}{c}
\text{CGTAGATTACAGG} \\
\downarrow \\
1111110011011011001011111 \\
\downarrow \\
\text{CGTAGATTACAGG} \\
\end{array}
\]

Instantaneous/Prefix Codes

Attach symbols to leaves of a binary tree
Codeword gives path to get to leaf

\[
\begin{array}{c}
1 \rightarrow A \\
1 \rightarrow B \\
1 \rightarrow 010 = C \\
0 \rightarrow 00 = D \\
\end{array}
\]

“Prefix code” because no codeword is a prefix of another

Decoding: follow tree while reading stream until hit leaf
Symbol is instantly identified. Return to root of tree.
Non-instantaneous Codes

The last code was instantaneously decodable:
We knew as soon as we’d finished receiving a symbol

\[
101100000101100
\]

\[
\begin{array}{c|c}
  A & 1 \\
  B & 10 \\
  C & 000 \\
  D & 100 \\
\end{array}
\]

This code is uniquely decodable, but not instantaneous or pleasant!

Expected length/symbol, \( \bar{L} \)

**Code lengths:** \( \{\ell_i\} = \{\ell_1, \ldots, \ell_I\} \)

\[
\text{Average, } \bar{L} = \sum_i p_i \ell_i
\]

Compare to Entropy:

\[
H(X) = \sum_i p_i \log \frac{1}{p_i}
\]

If \( \ell_i = \log \frac{1}{p_i} \) or \( p_i = 2^{-\ell_i} \) we compress to the entropy

\begin{align*}
\text{Limit on code lengths} & \\
\text{Imagine coding under an implicit distribution:} & \\
q_i = \frac{1}{Z} 2^{-\ell_i}, & \quad Z = \sum_i 2^{-\ell_i}.
\end{align*}

\[
H = \sum_i q_i \log \frac{1}{q_i} = \sum_i q_i (\ell_i + \log Z) = \bar{L} + \log Z
\]

\[
\Rightarrow \log Z \leq 0, \quad Z \leq 1
\]

Kraft–McMillan Inequality

\[
\sum_i 2^{-\ell_i} \leq 1 \quad \text{(if uniquely-decodable)}
\]

Proof without invoking entropy bound: p95 of MacKay, or p116 Cover & Thomas 2nd Ed.
**Kraft Inequality**

If height of budget is 1, codeword has height $= 2^{-\ell_i}$

Pick codes of required lengths in order from shortest–largest

Choose highest codeword of required length beneath previously-chosen code (There won’t be a gap because of sorting)

Can always pick codewords if total height, $\sum_i 2^{-\ell_i} \leq 1$

**Kraft–McMillan Inequality**

$\sum_i 2^{-\ell_i} \leq 1$ (instantaneous code possible)

Corollary: there’s probably no point using a non-instantaneous code.

Can always make complete code $\sum_i 2^{-\ell_i} = 1$: slide last codeword left.

---

**Performance of symbol codes**

**Simple idea:** set $\ell_i = \lceil \log \frac{1}{p_i} \rceil$

These codelengths satisfy the Kraft inequality:

$$\sum_i 2^{-\ell_i} = \sum_i 2^{-\lceil \log \frac{1}{p_i} \rceil} \leq \sum_i p_i = 1$$

Expected length, $\bar{L}$:

$$\bar{L} = \sum_i p_i \ell_i = \sum_i p_i \lceil \log \frac{1}{p_i} \rceil < \sum_i p_i (\log \frac{1}{p_i} + 1)$$

$$\bar{L} < H(p) + 1$$

Symbol codes can compress to within 1 bit/symbol of the entropy.

---

**Summary of Lecture 5**

**Symbol codes** assign each symbol in an alphabet a codeword.
(We only considered binary symbol codes, which have binary codewords.)

Messages are sent by concatenating codewords with no punctuation.

**Uniquely decodable:** the original message is unambiguous

**Instantaneously decodable:** the original symbol can always be determined as soon as the last bit of its codeword is received.

**Codeword lengths** must satisfy $\sum_i 2^{-\ell_i} \leq 1$ for unique decodability

**Instantaneous prefix codes** can always be found (if $\sum_i 2^{-\ell_i} \leq 1$)

**Complete codes** can always be found from binary trees with a codeword at every leaf.

If (big if) symbols are drawn i.i.d. with probabilities $\{p_i\}$, and $\ell_i = \log \frac{1}{p_i}$, then a prefix code exists that offers optimal compression.

**Next lecture:** how to form the best symbol code when $\{\log \frac{1}{p_i}\}$ are not integers.
Optimal symbol codes

Encode independent symbols with known probabilities:

E.g., \( A_X = \{A, B, C, D, E\} \)
\( P_X = \{0.3, 0.25, 0.2, 0.15, 0.1\} \)

We can do better than \( \ell_i = \lceil \log \frac{1}{p_i} \rceil \)

The Huffman algorithm gives an optimal symbol code.

Proof: MacKay Exercise 5.16 (with solution). Cover and Thomas has another version.

Huffman algorithm

Merge least probable

\[ \begin{array}{c|c}
\text{x} & p(x) \\
\hline
A & 0.3 \\
B & 0.25 \\
C & 0.2 \\
D & 0.15 \\
E & 0.1 \\
\end{array} \]

Can merge C with B or (D, E)

\[ \begin{array}{c|c}
\text{x} & p(x) \\
\hline
A & 0.3 \\
B & 0.25 \\
C & 0.2 \\
D & 0.15 \\
E & 0.1 \\
\end{array} \]

\( P(D \text{ or } E) = 0.25 \)

Continue merging least probable, until root represents all events \( P = 1 \)

Huffman algorithm

Given a tree, label branches with 1s and 0s to get code

Huffman decoding

Huffman codes are easily and uniquely decodable because they are prefix codes

Reminder on decoding a prefix code stream:

- Start at root of tree
- Follow a branch after reading each bit of the stream
- Emit a symbol upon reaching a leaf of the tree
- Return to the root after emitting a symbol...

An input stream can only give one symbol sequence, the one that was encoded.

Wow! Despite limitations we will discuss, Huffman codes can be very good. You’ll find them inside many systems (e.g., bzip2, jpeg, mp3), although all these schemes do clever stuff to come up with a good symbol representation.
Building prefix trees ‘top-down’

<table>
<thead>
<tr>
<th>x</th>
<th>P(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0.24</td>
</tr>
<tr>
<td>A2</td>
<td>0.01</td>
</tr>
<tr>
<td>B1</td>
<td>0.24</td>
</tr>
<tr>
<td>B2</td>
<td>0.01</td>
</tr>
<tr>
<td>C1</td>
<td>0.24</td>
</tr>
<tr>
<td>C2</td>
<td>0.01</td>
</tr>
<tr>
<td>D1</td>
<td>0.24</td>
</tr>
<tr>
<td>D2</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Heuristic: if you’re ever building a tree, consider top-down vs. bottom-up (and maybe middle-out)

Weighing problem strategy:
Use questions with nearly uniform distribution over the answers.

How well would this work on the ensemble to the right?

\[ H(X) = 2.24 \text{ bits (just over } \log 4 = 2). \] Fixed-length encoding: 3 bits

Top-down performing badly

Probabilities for answers to first two questions is \((1/2, 1/2)\)

Greedy strategy ⇒ very uneven distribution at end

Compare to Huffman

Expected length 2.36 bits/symbol

(Symbols reordered for display purposes only)

Relative Entropy / KL

Implicit probabilities: \(q_i = 2^{-\ell_i}\)

Extra cost for using “wrong” probability distribution:

\[
\Delta L = \sum_i p_i \ell_i - H(X) \\
= \sum_i p_i \log \frac{1}{q_i} - \sum_i p_i \log \frac{1}{p_i} \\
= \sum_i p_i \log \frac{p_i}{q_i} = D_{KL}(p || q)
\]

\(D_{KL}(p || q)\) is the Relative Entropy also known as the Kullback-Leibler divergence or KL-divergence
**Gibbs’ inequality**

An important result:

\[ D_{KL}(p \| q) \geq 0 \]

with equality only if \( p = q \)

“If we encode with the wrong distribution we will do worse than the fundamental limit given by the entropy”

A simple direct proof can be shown using convexity.

(Jensen’s inequality)

---

**Convexity**

\[ f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \]

Strictly convex functions:

Equality only if \( \lambda \) is 0 or 1, or if \( x_1 = x_2 \)

(non-strictly convex functions contain straight line segments)

---

**Jensen’s inequality**

*For convex functions:*

\[ \mathbb{E}[f(x)] \geq f(\mathbb{E}[x]) \]

Centre of gravity at \((\mathbb{E}[x], \mathbb{E}[f(x)])\), which is above \((\mathbb{E}[x], f(\mathbb{E}[x]))\)

Strictly convex functions:

Equality only if \( P(x) \) puts all mass on one value

---

**Remembering Jensen’s**

Which way around is the inequality?

\( f(x) = x^2 \) is a convex function

\[ \text{var}[X] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 \geq 0 \]

So we know Jensen’s must be: \[ \mathbb{E}[f(x)] \geq f(\mathbb{E}[x]) \]

(Or sketch a little picture in the margin)
Convex vs. Concave

For (strictly) concave functions reverse the inequalities

For concave functions: \( E[f(x)] \leq f(E[x]) \)

Jensen’s: Entropy & Perplexity

Set \( u(x) = \frac{1}{p(x)}, \quad p(u(x)) = p(x) \)

\[
\mathbb{E}[u] = \mathbb{E}\left[ \frac{1}{p(x)} \right] = |A|
\]

(Tutorial 1 question)

\( H(X) = \mathbb{E}[\log u(x)] \leq \log \mathbb{E}[u] \)

\( H(X) \leq \log |A| \)

Equality, maximum Entropy, for constant \( u \) \( \Rightarrow \) uniform \( p(x) \)

\( 2^{-H(X)} = \text{“Perplexity”} = \text{“Effective number of choices”} \)

Maximum effective number of choices is \( |A| \)

Summary of Lecture 6

The Huffman Algorithm gives optimal symbol codes:
Merging event adds to code length for children, so
Huffman always merges least probable events first

A complete code implies negative log probabilities: \( q_i = 2^{-\ell_i} \).
If the symbols are generated with these probabilities, the symbol code compresses to the entropy. Otherwise the number of extra bits/symbol is given by the Relative Entropy or KL-divergence:

\[
D_{\text{KL}}(p \parallel q) = \sum_i p_i \log \frac{p_i}{q_i}
\]

Gibbs’ inequality says \( D_{\text{KL}}(p \parallel q) \geq 0 \) with equality only when the distributions are equal.

Jensen’s inequality is a useful means to prove several inequalities in Information Theory including (it will turn out) Gibbs’ inequality.