Huffman Coding

How can we represent a string of characters with a minimum number of bits?

Answered by David Huffman in 1952 when he was a graduate student at MIT.

- Each character represented by a bit string code.
- Codes are of various lengths.
- Codes are prefix codes (prefix-free). No code is prefix of another.

Prefix codes

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>01</td>
<td>101</td>
</tr>
</tbody>
</table>

Prefix code

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01</td>
<td>001</td>
</tr>
</tbody>
</table>
How would you represent a string of characters G, C, T, and A?
Assign shortest codes to most frequent characters. Let's say frequencies are

\[
\begin{align*}
G & : 0.4 \\
C & : 0.3 \\
T & : 0.2 \\
A & : 0.1
\end{align*}
\]

One prefix code is

\[
\begin{align*}
G & : 0 \\
C & : 10 \\
T & : 110 \\
A & : 111
\end{align*}
\]

The string

\[CGTTA\text{ }CC\text{ }ATAAA\]

is

\[10\text{ }0\text{ }110\text{ }110\text{ }111\text{ }10\text{ }10\text{ }111\text{ }110\text{ }111\text{ }111\]

Can you decode this? This binary tree helps:

![Binary Tree Diagram]
How many bits are needed for 1,000 G, C, T, A's?

\[
\begin{array}{cccc}
\text{G} & \text{C} & \text{T} & \text{A} \\
0.4 & 0.3 & 0.2 & 0.1 \\
\end{array}
\]

\[
\begin{align*}
\text{G: } & 0.4 \times 1000 = 400 \quad \text{Code} \\
\text{C: } & 0.3 \times 1000 = 300 \\
\text{T: } & 0.2 \times 1000 = 200 \\
\text{A: } & 0.1 \times 1000 = 100 \\
\end{align*}
\]

Number of bits = 400.1 + 300.2 + 200.3 + 100.3

\[= 400 + 600 + 600 + 300 \]

\[= 1,900 \text{ bits} \]

What if we used a fixed-length code?

Four characters only needs 2-bit each

\[
\begin{array}{cccc}
\text{G} & \text{C} & \text{T} & \text{A} \\
00 & 01 & 10 & 11 \\
\end{array}
\]

1,000 characters = 2,000 bits
What if frequencies were

\[
\begin{array}{cccc}
G & C & T & A \\
0.8 & 0.1 & 0.05 & 0.05 \\
\end{array}
\]

0 10 110 111

1,000 characters = 800.1 + 100.2 + 50.3 + 50.3
= 800 + 200 + 150 + 150
= 1,300

fixed length still 2,000

Savings increases with number of characters.
How to construct a Huffman code?

Start with a leaf for each character with its frequency.

- G 0.4
- C 0.3
- T 0.2
- A 0.1

Merge two nodes with lowest frequency. Sum their frequencies.

- G 0.4
- C 0.3
- 0.3
  - T 0.2
  - A 0.1

Repeat
Repeat again

and assign 0 to left subtrees and 1 to right ones.
Pseudo code from page 431.

Assume Q is a min-priority queue of objects containing a frequency and, if a leaf, the character. In the queue, the order is determined by the frequency.

Let C be the initial objects containing each character and its frequency.

Each object has attributes .freq, .left, .right.

\[ n = |C| \]
\[ Q = C \]

for \( i = 1 \) to \( n-1 \)

\[ z = \text{new object} \]
\[ z.$left = x = \text{pop}(Q) \]
\[ z.$right = y = \text{pop}(Q) \]
\[ z.$freq = x.$freq + y.$freq \]
\[ \text{push}(Q, z) \]

return \text{pop}(Q) \# \text{top node}
To encode a string of characters, a dictionary of Huffman codes, indexed (keyed) by characters, would be useful.

How can you construct this dictionary from the tree?
Now, how would you use this dictionary to encode a text string?

'GCCAT GC' ⇒ 01010111110010

And how would you decode this?

01010111110010 ⇒ 'GCCAT GC'

---

Diagram of a binary tree with probabilities for characters:
- G: 0.4
- C: 0.3
- T: 0.2
- A: 0.1
- Root: 1.0