Efficient Compression

Given a string of characters $c \in C$, a variable length codes assigns to each character a code (string of 0's and 1's), different characters have different length code.

Let $d(c)$ be the length of the code of character c.

Assume that the frequency of character c in the string is $f(c)$.

The **cost** of the code is

$$B(C) = \sum_{c \in C} f(c) \cdot d(c).$$

Prefix Codes

In a **Prefix code** no codeword is a prefix of another code word.

Easy encoding and decoding.

Represented as a binary tree.

In an optimal code each non-leaf node has two children.

Huffman Code

A simple greedy algorithm that generates an optimal prefix code.

Theorem 1. *The algorithm Huffman encode n characters in $O(n \log n)$ time.*

Proof. The queue is maintained as a heap.

The queue is built in $O(n)$ time

There are $n - 1$ iterations of the loop, each takes $O(\log n)$ times. $\square$

Optimality

Theorem 2. *Let x and y be the two characters in C with the lowest frequencies. There is an optimal prefix code for C for which the codewords of x and y have the same length and differ only in the last bit.*

Proof. Given a tree T of optimal prefix code of C we generate a new tree T'' by moving x and y to be siblings with maximum depth.

Let b and c be two characters that are encoded by two sibling leaves of maximum depth.

Assume $f(b) \leq f(c)$ and $f(x) \leq f(y)$, which implies $f(x) < f(b)$ and $f(y) < f(c)$.

Exchange x and b to generate a tree T' and then y and c to generate tree T''.

$$
\begin{gathered}
B(T) - B(T') = \sum_{c \in C} f(c) d_T(c) - \sum_{c \in C} f(c) d_{T'}(c) \\
= f(x) d_T(x) + f(b) d_T(b) - f(x) d_{T'}(x) - f(b) d_{T'}(b) \\
= f(x) d_T(x) + f(b) d_T(b) - f(x) d_T(b) - f(b) d_T(x) \\
= (f(b) - f(x))(d_T(b) - d_T(x)) \geq 0
\end{gathered}
$$

Similar for the move from T' to T''. $\square$

Theorem 3. *Let T be a tree representing an optimal prefix code for C. Consider two characters x and y that appears as siblings in the tree. Let z be their parent in the tree. Consider z a character with frequency $f(z) = f(x)+f(y)$, the tree $T' = T - \{x, y\}$ represents an optimal prefix code for the alphabet $C' = C - \{x, y\} \cup \{z\}$.*

Proof.

For $c \in C - \{x, y\}$, $d_T(c) = d_{T'}(c)$.

$d_T(x) = d_T(y) = d_{T'}(z) + 1$

$B(T) = B(T') + f(x) + f(y)$

If T' is not optimal, there is a tree T'' such that $B(T'') < B(T')$. Replacing z with x and y in T'' will give a code with cost

$$B(T'') + f(x) + f(y) < B(T)$$

which contradicts the fact that T was optimal. $\square$

Theorem 4. *The procedure Huffman generates an optimal prefix code.*

Proof. We can always merge the two lowest frequency characters and continue with the remaining set of characters. $\square$

Information Theory

Assume that the string is generated by a **memoryless source**: regardless of the past, the next character in the string is c with probability $f(c)$.

[Same results hold for ergodic stationary processes]

Theorem 5. *The optimal compression ratio of a memoryless source is give by the entropy of the source*

$$E = -\sum_{c \in C} f(c) \log f(c).$$

Theorem 6. *The Huffman code is asymptotically optimal.*

Not a real proof...

Let $C = \{c_1, ..., c_\ell\}$.

Assume that all probabilities are of the form $f(c) = \frac{1}{2^s}$.

Note that $\sum_{c \in C} f(c) = 1$.

Assume that $1/2^r$ is the smallest $f(c)$, there must be at least two characters with that probability, the algorithm pairs all of them to nodes with probability $1/2^{r-1}$.

The Huffman algorithm generates a tree such that the probability of visiting a node of depth i is $1/2^i$.

If $f(c) = \frac{1}{2^i}$ then $d(c) = i$.

$$B(C) = \sum_{c \in C} f(c) \cdot d(c) = -\sum_{c \in C} f(c) \log f(c) = E$$