Huffman Coding - Optimality and Issues

September 4, 2017

1 Huffman Encoding Algorithm

Let p be the probability distribution of the set of symbols Σ.

Algorithm Huffman(Σ, p)

1. If $|\Sigma| = 1$, then return a single vertex.
2. Let x and y be symbols with the least probabilities, breaking ties arbitrarily.
3. $\Sigma' \leftarrow \Sigma \setminus \{x, y\} \cup \{z\}$, where $z \notin \Sigma$.
4. Let $p'(x) = p(x)$ if $x \in \Sigma' \setminus \{z\}$, and $p'(z) = p(x) + p(y)$.
5. $T' \leftarrow \text{Huffman}(\Sigma', p')$.
6. Create T from T' by adding x and y as children of z.
7. Return T.

2 Optimality

We now consider the optimality of the Huffman encoding with respect to expected compression length.

Huffman code assigns each unique symbol x_i to a leaf of a tree. The depth of x_i is the length of the code that Huffman encoding assigns x_i.

Any prefix encoding which encodes each symbol separately into a string of bits, assigns x_i to the leaf of a tree. We denote the expected compressed length of a tree T as $E_p[T]$. It is sufficient to establish that over such tree encodings, Huffman coding has the least expected path length.

We have the following two lemmas which establish the optimality of Huffman encoding over all prefix encodings where each symbol is assigned a unique string of bits. [1]

Lemma 1. *Let T be a tree for some prefix encoding A and some probability distribution p over the symbols. Let x and y be two leaves, and T' be the tree obtained by swapping x and y in T. Then*

$$E_p[T'] - E_p[T] = (p(x) - p(y))[\mathit{depth}(y, T) - \mathit{depth}(x, T)].$$

[1]The proof is taken from the lecture notes of Michael Brudno.

Proof. The difference is

$$\begin{aligned}
&p(x)\mathrm{depth}(y,T)+p(y)\mathrm{depth}(x,T)-p(x)\mathrm{depth}(x,T)-p(y)\mathrm{depth}(y,T)\\
&\quad=p(x)[\mathrm{depth}(y,T)-\mathrm{depth}(x,T)]+p(y)[\mathrm{depth}(x,T)-\mathrm{depth}(y,T)]\\
&\quad=(p(x)-p(y))[\mathrm{depth}(y,T)-\mathrm{depth}(x,T)].
\end{aligned}$$

□

Lemma 2. *There is an optimal tree in which the two symbols with least probabilities are siblings.*

Proof. HW 1 □

Lemma 3. *Huffman Encoding produces an optimal tree.*

Proof. The proof is by induction on $|\Sigma|$.

If $|\Sigma| = 1$, then there is only one symbol, with depth 0, hence Huffman encoding attains the least expected compressed length.

Assume that the claim holds for all alphabets of size $\le n-1$. Let T' be the Huffman tree for Σ' and p', following the notation in the algorithm. We have the following.

$$\begin{aligned}
E_p[T] &= \left(\sum_{a\in\Sigma\setminus\{x,y\}} p(a)\mathrm{depth}(a,T)\right)+p(x)\mathrm{depth}(x,T)+p(y)\mathrm{depth}(y,T)\\
&= \left(\sum_{a\in\Sigma\setminus\{x,y\}} p(a)\mathrm{depth}(a,T)\right)+[p(x)+p(y)]\left[\mathrm{depth}(z,T')+1\right]\\
&= \left(\sum_{a\in\Sigma'} p(a)\mathrm{depth}(a,T')\right)+p(x)+p(y)\\
&= E_{p'}[T']+p(x)+p(y).
\end{aligned}$$

We now argue that if T' is optimal for p', then so is T for p. Suppose otherwise. If T is not optimal, then let Z be an optimal tree with x and y as siblings.

Let Z' be the tree obtained by removing x and y from Z. We can view the tree Z' as a tree for Σ' and probabilty distribution p'.

It follows that $E_p[Z] = E_{p'}[Z'] + p(x) + p(y)$, similar to the above calculation. Hence,

$$\begin{aligned}
E_{p'}[T'] &= E_p[T]-p(x)-p(y)\\
&> E_p[Z]-p(x)-p(y)\\
&= E_{p'}[Z'],
\end{aligned}$$

which contradicts the optimality of T'. □

3 Suboptimality of Huffman Encoding

Huffman encoding uses $\lceil -\log p(a)\rceil$ to encode $a \in \Sigma$. This introduces overheads when $-\log p(a)$ is not an integer - equivalently, $p(a)$ is not of the form $m/2^k$ for some positive integers m and k. So the expected codelength of Huffman encoding is

$$\sum_{a\in\Sigma} p(a)\lceil -\log p(a)\rceil.$$

When probabilities can be of arbitrary form, then there may be other encodings which are optimal. Such encodings include Arithmetic Encoding and Asymmetric Numeral Systems. We will discuss Asymmetric Numeral Systems and their optimality properties later in the course. Both these encoding schemes attain an expected code length (ignoring an additive $O(1)$ term) of

$$\left\lceil \sum_{a\in\Sigma} p(a) \times -\log p(a) \right\rceil,$$

which is shorter than the expected code length of Huffman encoding.

Appendix

4 Optimality - Outline of Another Approach

We argue directly that each symbol a is encoded using approximately $\lceil -\log p(a) \rceil$ bits by the Huffman encoding algorithm.

The depth of a particular symbol a is dependent solely on the number of symbols in Σ with greater probability. The maximum number of symbols with probability greater than $p(a)$ is $\lfloor (1 - p(a))/p(a) \rfloor$.

Claim. The maximum number of steps the algorithm takes to process symbols with probability greater than $p(a)$ is

$$\log \left\lfloor \frac{1-p(a)}{p(a)} \right\rfloor.$$

Then the depth of a is

$$\log \left\lfloor \frac{1-p(a)}{p(a)} \right\rfloor + 1 \quad = \quad \log \left\lfloor \frac{1}{p(a)} - 1 \right\rfloor + 1$$

Corollary 4. *The expected codelength of Huffman encoding is at most*

$$\left\lceil \sum_{a\in\Sigma} p(a) \log \left\lfloor \frac{1-p(a)}{p(a)} \right\rfloor \right\rceil + 1.$$

Proof. The expected codelength is

$$\sum_{a\in\Sigma} \left[p(a) \log \left\lfloor \frac{1-p(a)}{p(a)} \right\rfloor + p(a) \right] = \left\lceil \sum_{a\in\Sigma} p(a) \log \left\lfloor \frac{1-p(a)}{p(a)} \right\rfloor \right\rceil + 1.$$

□