ハフマン符号
Huffman Coding

「符号化技術」実験

Dec 2018

1　前書き

ハフマン符号とは、1952 年にデビッド・ハフマンによって開発された符号で、文字列をはじめとするデータの可逆圧縮などに使用される。

ほかのエントロピー符号と同様、よく出現する文字には短いビット列を、あまり出現しない文字には長いビット列を割り当てることで、メッセージ全体の符号化に使われるデータ量を削減することを狙っている。

コンパクト符号やエントロピー符号の一つ。JPEG や ZIP (Deflate) などの圧縮フォーマットで使用されている。

シャノン符号化が最適ではない場合が存在する不完全な符号であったのに対し、ハフマン符号は（整数の符号語長という制約のもとでは、）常に最適な符号を構成できる。擬似的に実数の符号語長を割り振る算術符号と比較すれば、データ圧縮効率は劣る。ただし、算術符号やその他の高効率の符号化法と異なり、特許の問題が無い。

2　種類

符号化の原理上、木を構成する前に各記号の出現頻度をあらかじめ知っておく必要がある。

ファイルなど固定長のデータに対し、1 度全部読み込んで、データのすべての記号を調べて符号木を構築しておき、もう 1 度頭から読み直して符号化を行う方法が、静的ハフマン符号 (Static Huffman coding) である。

Deflate では、データの全部ではなく、ブロック毎に符号を作っている。これを Deflate ではダイナミックハフマン符号（Dynamic Huffman coding）と呼んでいる。

一方、最初の状態では頻度情報を持たず、記号を 1 個読み込むごとに符号木を作り直す方法は適応型ハフマン符号 (Adaptive Huffman coding) である。これを指して日本では動的ハフマン符号と呼ぶことが多い。

3　符号化の原理

例として SHINSHUSUSHI という 12 文字からなるメッセージを考える。

　このメッセージ中には 5 種類の文字が使われているので、もしそれぞれの文字を固定長のビット列で表すとすれば、1 文字につき最低 3 ビット必要となる。文字とビット列の対応表として、

Table 1: 記号とビット列（コード）

H	000
I	001
N	010
S	100
U	101

　を使い、それぞれの文字をビット列に置き換えたとすると、メッセージ全体は次のビット列で表される。

Table 2: メッセージとビット列

S	H	I	N	S	H	U	S	U	S	H	I
100	000	001	010	100	000	101	100	101	100	000	001

　12 文字 × 3 ビットで全体のビット数は 36 ビットとなる。もし、よく出現する文字には短いビット列を、あまり出現しない文字には長いビット列を割り当てることができれば、メッセージ全体の符号化に必要なビット数を抑えることが期待できる。そこで、文字とビット列の対応表として、Table 3 を使ったとすると、メッセージ全体は Table 4 となり、全体のビット数は 28 ビットとなる。固定長の方式に比べると 78% ほどのデータ量に抑えられている。

4　ハフマン符号の構成

文字とビット列の対応表を作るには、まずデータに出現する文字の出現回数を求め、それをもとにハフマン木と呼ばれるバイナリツリーを構成するという手順を踏む。

　ハフマン木の構成の仕方は、次のようなアルゴリズムになる。

Table 3: 記号とビット列（ハフマンコード）

H	10
I	110
N	1111
S	0
U	1110

Table 4: メッセージとビット列（ハフマンコードを用いた）

S	H	I	N	S	H	U	S	U	S	H	I
0	10	110	1111	0	10	1110	0	1110	0	10	110

- まず、葉を含むすべての節点のうち、親を持たないものを集める。
- その中から、最小の値を持つものと 2 番目に小さい値を持つものを取り出す。
- それらを子供に持つ新しい節点を作る。このとき、新しい節点の値は、両方の子供の値の和とする。

以上を根節点に到達するまで繰り返すと、木が完成する。次に、根から順に左右に 0 と 1 の値を割り振っていく（左右のどちらに 0 と 1 を与えるかは任意）。すると、それぞれの葉（記号）に対して、一意にビット列が与えられる。この記号とビット列の関係をもとに、もとのデータの記号をビット列に変換していくことで符号化が行われる。

例
入力 SHINSHUSUSHI に対して上記のアルゴリズムを適用すると、

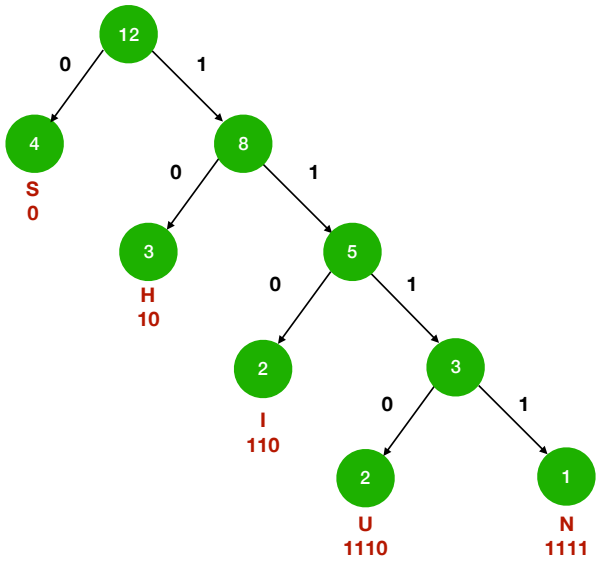

Figure 1: ハフマン木

出現頻度と割り当てられた符号

Table 5: 出現頻度と割り当てられた符号

文字	個数	符号
S	4	0
H	3	10
I	2	110
U	2	1110
N	1	1111

が得られ、個数の多い文字ほど短い符号が割り当てられていることがわかる。

5　接頭符号

ハフマン符号は、接頭符号である。つまり、ある文字に対応するビット列が、他の文字に対応するビット列の接頭辞になることはない。この特徴のおかげで、デコーダはビット列を先頭から一度読むだけで元のメッセージを一意にデコードできる。

6　利用例

算術符号やその他の高効率の符号化法と異なり、特許の問題が無い。そのため、JPEG や ZIP などの圧縮フォーマットで使用されている。