レンジ・コーダを用いて圧縮率を改善する
LZ77符号によるファイルの圧縮とその改良

後編

広井 誠

前編ではLZSS符号の基本とその改良方法であるLZB符号を説明しました．LZB符号は長さの符号化にγ符号を，距離の符号化に可変長符号とCBT符号を適用することで，圧縮率を向上させることができました．

LHAやzipはLZSS符号とハフマン符号を組み合わせて高い圧縮率を達成しています．ここで，ハフマン符号のかわりにレンジ・コーダ（RangeCoder）を適用することを考えます．とくに，バイナリ・レンジ・コーダ（Binary RangeCoder）を用いてLZSS符号に適したモデルを構成すると，圧縮率をさらに向上させることが可能です．

後編である今回は，レンジ・コーダの一つであるバイナリ・レンジ・コーダの基本を解説し，LZSS符号に適したモデルの構成方法について詳しく説明します．

算術符号とレンジ・コーダ

ハフマン符号と算術符号は，記号の出現確率だけを利用してデータを圧縮する方法です．算術符号はハフマン符号よりも性能が良いのですが，実現方法が難しくて実行速度がハフマン符号よりも遅く，なおかつ特許の問題もあって，一般にはあまり普及していないのが現状です．

ところが，最近になってレンジ・コーダ（RangeCoder）という方法が注目されています．レンジ・コーダは原理的には算術符号と同じ方法ですが，参考文献(3)によると『（おそらく）特許フリー』とのことで，性能は算術符号に比べるとわずかに劣化しますが，実現方法はとても簡単で実行速度も高速です．もちろん，ハフマン符号よりも高性能です．LHAやzipはLZSS符号とハフマン符号を組み合わせた方式ですが，ハフマン符号の代わりにレンジ・コーダを用いることで圧縮率を改善することができます．

算術符号とその実装

● 算術符号の符号化 ―― 実数の区間を用いる

最初に算術符号の基本的な考え方を説明します．算術符号は記号列全体をひとつの符号語にする方法で，1960年代にP.Elias氏によって提案されました．

算術符号は記号列を実数0と1の間の区間を用いて表します．ここでは記号{0，1}だけを扱う符号化法を考えます．この方法を「二値算術符号」と言います．3個以上の記号を扱う方法は「多値算術符号」と呼ばれます．二値算術符号は多値算術符号よりも簡単に実装できるため，画像の圧縮などで使われることがあります．

たとえば，記号0の出現確率が0.8で1の出現確率が0.2とします．算術符号は，区間を記号の出現確率に比例した小区間に分割していくことで符号化を行います．簡単な例として，記号列“ 00100 ”を符号化してみましょう（**図1**）．

x以上y未満の区間を$[x, y)$と表すことにします．区間の初期値は$[0, 1)$です．記号を読み込んだら区間$[0, 1)$を分割します．記号が0ならば区間の0から0.8までの部分，1ならば0.8から1.0までの部分に分割します．

最初の記号は0なので区間は$[0, 0.8)$となります．次の記号は0なので，区間$[0, 0.8)$の0から0.8までの部分$[0, 0.64)$が新しい区間になります．このように，記号を読み込むたびに区間を分割していくと，記号列“ 00100 ”を表す区間は$[0.512, 0.59392)$になります．そして実際の符号語は，この区間に含まれる一つの実数を指定します．

ここで符号語を2進数で表して，区間内で小数点以下のビット数の少ない値を選ぶことにします．たとえば，0.5625を2進数で表すと次のようになります．

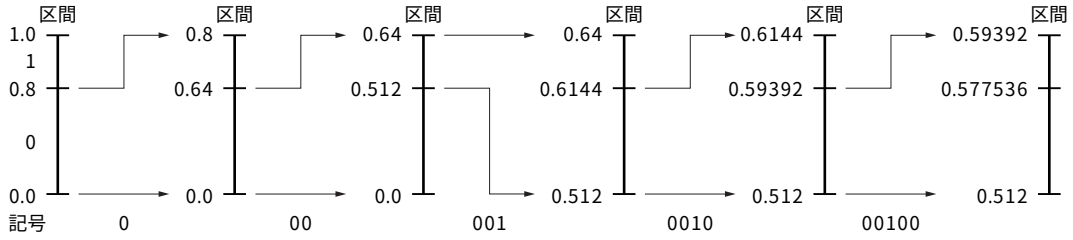

図1 符号化の過程（二値算術符号）

$$0.5625 = \frac{1}{2} + \frac{1}{16} = (0.1001)\text{（2進法）}$$

“ 1001 ”の4ビットを符号語として出力すれば，記号列“ 00100 ”の5ビットを4ビットに圧縮することができます．この例のように，二値算術符号を使うと1ビットのデータを1ビット未満に圧縮することができます．ハフマン符号の場合，最短の符号語は0または1なので，1ビット未満に圧縮することはできません．

● 算術符号の復号

次は復号を説明します．ここでは説明の都合上，符号語は区間の下限値0.512とします．0.512は［0，0.8）の間にあるので，最初の記号は0であることがわかります．次に，0を表す区間［0，0.8）を［0，1.0）になるように拡大すると，符号語は次のように変換できます．

新しい符号語 =（符号語－記号の下限値）/記号の区間幅
=（0.512 － 0）/0.8
= 0.64

新しい符号語0.64は［0，0.8）の間にあるので，次の記号は0であることがわかります．このような操作を繰り返し行うことで，**表1**のように記号列“ 00100 ”を復号することができます．

ところで，算術符号には二つの問題点があります．一つは記号列の最後を判定できないことです．**表1**の例では，途中で符号語が0になりましたが，このあとも記号0を復号することができます．つまり，符号語0.512は記号列“ 00100 ”だけではなく，001000，0010000…などにも復号することができるのです．

この問題は多値算術符号でも発生します．多値算術符号の場合，終端を表す記号を用意して，終端記号を復号したら終了する方法があります．または，記号の総数をファイルの先頭に書き込んでおく，などといった方法で解決することができます．

もう一つの問題は，入力する記号列が長くなるほど，より多くの桁数が必要になることです．また，浮動小数点演算の誤差も考慮しなければなりません．これはとても大きな問題で，解決するまでに長い時間がかかりました．1981年にC.B.Jones氏によって発表された算術符号（Jones符号）は，実数のかわりに整数で演算するようにくふうされています．算術符号に興味のある方は参考文献(1)，(2)をご覧ください．

レンジ・コーダとその実装

● レンジ・コーダの基本的な考え方 ―― 整数演算のみで符号化可能

レンジ・コーダは1998年にMichael Schindler氏が発表し，「高性能，高速，特許フリー」の方法として注目を集めるようになりました．Michael Schindler氏のレンジ・コーダは計算の途中で「桁上がり」が発生しますが，ロシアのDmitry Subbotin氏が発表した「桁上げのないレンジ・コーダ」は，その名のごとく桁上がりが発生しません．現在，レンジ・コーダはおもにこの2種類の形式が存在するようです．

表1
復号の過程（二値算術符号）

符号語	記　号	区　間
0.512	0	［0，0.8）
0.64	0	［0，0.8）
0.80	1	［0.8，1.0）
0	0	［0，0.8）
0	0	［0，0.8）

それでは，レンジ・コーダの基本的な考え方について説明します．算術符号は区間［0，1）を分割していきますが，レンジ・コーダは［0，1）を分割するのではなく，最初に大きな区間，たとえば［0，1000）を設定して，それを小さな区間に分割していくことで符号化を行います．レンジ・コーダは整数で演算するので，当然ですが記号列が長くなると区間が狭くなって分割できなくなります．そのときは区間を伸張することで対応します．

たとえば，［0，1000）を分割していくと［123，124）になります．もうこれ以上分割できないので，区間を100倍して［12300，12400）を分割することにします．このとき，区間全体の大きさは［0，1000）ではなく，それを1000倍した［0，100000）と考えるわけです．

単純に考えると，区間を表すために多倍長整数が必要になりますが，区間を引き伸ばすタイミングを定めることにより，通常の整数演算だけでレンジ・コーダをプログラムすることができます．また，区間全体の大きさも覚えておく必要はありません．レンジ・コーダは分割した区間の幅（range）と下限値だけで符号化することができます．復号の処理でも，符号化と同じタイミングでrangeを伸張していくことで，符号語を記号列に復号することができます．

● バイナリ・レンジ・コーダの符号化

レンジ・コーダは算術符号と違って，記号が多値の場合でも簡単に実装することができます．ところが，あえて記号{0，1}だけを扱うレンジ・コーダを考えることができます．これを「バイナリ・レンジ・コーダ（Binary RangeCoder）」といいます．

たとえば，大きな整数を符号化する場合，多値レンジ・コーダでは実装が難しくなります．ところが，バイナリ・レンジ・コーダを用いると，前編で説明したγ符号やδ符号に基づいたモデルを作成することができるので，大きな整数でも効率的に符号化することができます．実際，LZSS符号の長さや距離の符号化にバイナリ・レンジ・コーダを用いると，圧縮率を向上させることができます．

それでは，バイナリ・レンジ・コーダの符号化を説明します．ここでは説明の都合上，区間の幅rangeを0x1000000に設定します．実際にはrangeを0xffffffff（32ビット）に設定する場合が多いでしょう．下限値lowは0に初期化します．区間は［low，low+range）と表すことができるので，最初の区間は［0，0x1000000）になります．また，rangeの初期値が0x1000000なので，lowの値は0から0xffffffまでの範囲（24ビット）になり