ART code II: Lossless Encoding Based on Redistribution of Statistics

Artyom M. Grigoryan
amgrigoryan@utsa.edu

The calculation of codeword lengths is based on construction of a new source with statistics that is determined by the consecutive redistribution of the probabilities of symbols in accordance with their original probabilities at each stage of the encoding.

Notes for class 4663 from the presentation in the International IEEE conference ITCC-2003

1. We construct an optimal coding set for another source

$$\mathcal{A}_m = \{a_1, a_2, ..., a_m\} \to \mathcal{B}_m = \{b_1, b_2, ..., b_m\}$$

with statistics different from $\mathcal{A}_m$.

2. The set of codeword lengths for b_i is used for encoding the letters a_i, moreover, we may consider that

$$c(a_i) = c(b_i), \quad i = 1 : m.$$

1. To start, we sort the letters of the alphabet $\mathcal{A}_m$ in descending order of their probabilities and consider $p_1 \geq p_2 \geq ... \geq p_m$. For a new alphabet $\mathcal{B}_m$, we first assign the probabilities $p'_i = p_i$ to the letters b_i.

2. On each stage of the algorithm, the probabilities p'_i of the letters b_i will be renewed, and we will use the notation $p'_{i;new}$ when it will be necessary.

3. After encoding each following letter b_i by a codeword $c(b_i)$ of length $\bar{l}_i$ bits, we consider that the letter b_i has a probability equal to

$$\bar{p}_i = \frac{1}{2^{\bar{l}_i}}.$$

4. If the difference of probabilities is not zero

$$D_i = p'_i - \bar{p}_i \neq 0$$

the difference is transferred to the rest $(m - i)$ letters $b_{i+1}, b_{i+2}, ..., b_m$ in accordance with probabilities of letters $a_{i+1}, a_{i+2}, ..., a_m$.

Probabilities of these letters are considered to be equal to

$$p'_{i+k;new} = p'_{i+k} + \frac{D_i}{\sum\limits_{k>i} p_k} \cdot p_{i+k},\ k = 1 : (m-i).$$

It is obvious that $p'_{i+1} \geq p'_{i+k}$, $k = 2 : (m-i)$.

The value p'_{i+1} will be considered next. Since the probability of symbol b_{i+1} has been recounted, the new length for this symbol is calculated as

$$l'_{i+1} = \log_2 \frac{1}{p'_{i+1}} = [l'_{i+1}] - m_{i+1}, 0 \leq m_{i+1} < 1$$

and $\bar{l}_{i+1} = [l'_{i+1}]$ bits are assigned for encoding the letter b_{i+1} and its probability will be therefore considered equal

$$\bar{p}_{i+1} = \frac{1}{2^{\bar{l}_{i+1}}}. \tag{1}$$

In a similar way, the remainder $D_{i+1} = p'_{i+1} - \bar{p}_{i+1}$ of the probability is distributed between the rest $(m - i - 1)$ letters. As a result, we obtain the following optimal source coding set with the entropy ratio

$$\bar{\varepsilon} = \sum_{i=1}^{m} \left\{ \bar{\varepsilon}_i = \bar{p}_i \bar{l}_i \right\} = \sum_{i=1}^{m} \bar{p}_i \log_2 \frac{1}{\bar{p}_i}$$

and $\bar{p}_1 + ... + \bar{p}_m = 1$.

Example 1: The letters of the alphabet

$$\mathcal{A}_5 = \{a_5, a_4, a_3, a_2, a_1\}.$$

are taken in ascending order of their probabilities $P = \{0.1, 0.1, 0.2, 0.2, 0.4\}$. (We can change the order and begin from the symbol a_1.)

Step 1: Letter is a_5, $p_5 = 0.1$

$$l'_5 = -\log_2 p_5 = 3.32193 \rightarrow [l'_5] = 4\text{bits}$$

Thus, we consider the probability of the letter a_5 is considered to be

$$\bar{p}_5 = \frac{1}{2^{\bar{l}_5}} = \frac{1}{16} = 0.0625.$$

The difference of probabilities

$$D_5 = p_5 - \bar{p}_5 = 0.1 - 0.0625 = 0.0375 \quad (2)$$

will be redistributed to the rest for symbols proportional to their probabilities. Then we continue encoding the next symbol a_4 and so on.

$$p'_5 = \bar{p}_5 = 0.0625$$
$$p'_4 = 0.1 + (D_5/0.9) \cdot 0.1 = 0.1 + 0.0042 = 0.1042$$
$$p'_3 = 0.2 + (D_5/0.9) \cdot 0.2 = 0.2 + 0.0083 = 0.2083$$
$$p'_2 = 0.2 + (D_5/0.9) \cdot 0.2 = 0.2 + 0.0083 = 0.2083$$
$$p'_1 = 0.4 + (D_5/0.9) \cdot 0.4 = 0.4 + 0.0167 = 0.4167$$

Stage 1

$\mathcal{A}_5$	p_i	D_i	p'_i	$\bar{\varepsilon}_i$	l_i	$\bar{l}_i$
a_5	0.1	0.0375	$0.0625 = \bar{p}_5$	0.4	3.3	4
a_4	0.1	$+1/9\cdot$	0.1042			
a_3	0.2	$+2/9\cdot$	0.2083			
a_2	0.2	$+2/9\cdot$	0.2083			
a_1	0.4	$+4/9\cdot$	0.4167			

Stage 2.

$\mathcal{A}_5$	p_i	p'_i	D_i	$p'_{k;new}$	$\bar{\varepsilon}_i$	l'_i	$\bar{l}_i$
a_5	0.1		0.03750	0.0625	0.4	3.3	4
a_4	0.1	0.1042	0.04167	0.0625	0.4	3.3	4
a_3	0.2	0.2083	$+1/4\cdot$	0.21875			
a_2	0.2	0.2083	$+1/4\cdot$	0.21875			
a_1	0.4	0.4167	$+1/2\cdot$	0.43750			

Stage 3.

$\mathcal{A}_5$	p_i	p'_i	D_i	$p'_{k;new}$	$\widetilde{\varepsilon}_i$	l'_i	$\bar{l}_i$
a_5	0.1		0.03750	0.0625	0.4	3.3	4
a_4	0.1		0.04167	0.0625	0.4	3.3	4
a_3	0.2	0.21875	0.09375	0.1250	0.6	2.2	3
a_2	0.2	0.21875	$+1/3\cdot$	0.25			
a_1	0.4	0.43750	$+2/3\cdot$	0.5			

Stage 4.

$\mathcal{A}_5$	ε_i	p_i	D_i	$\bar{p}_i$	$\widetilde{\varepsilon}_i$	l'_i	$\bar{l}_i$
a_5	0.33219	0.1	0.03750	0.0625	0.4	3.3	4
a_4	0.33219	0.1	0.04167	0.0625	0.4	3.3	4
a_3	0.46439	0.2	0.09375	0.125	0.6	2.2	3
a_2	0.46439	0.2	0	0.25	0.4	2	2
a_1	0.52877	0.4	0	0.5	0.4	1	1

The corresponding source coding set is

$$\mathbf{A}' = \{\mathcal{A}_5, (0.1, \varepsilon_5, 4), (0.1, \varepsilon_4, 4), (0.2, \varepsilon_3, 3), (0.2, \varepsilon_2, 2), (0.4, \varepsilon_1, 1), \tilde{l}\}.$$

The rate of the coding is $\tilde{l} = 2.20$bit/symbol.

The variance is $\sigma(\tilde{l}) = 0.58652$, and the redundancy is calculated as

$$R = \tilde{l} - \varepsilon = \sum_{i=1}^{5} p_i \left(\bar{l}_i + \log p_i\right) = 0.07807.$$

Shannon-Fano algorithm for coding b_i:

Table 7.

$\mathcal{B}_5$	$\bar{p}_i$	step1	step2	step3	step4	$c(b_i)$
b_5	1/16				II	1111
b_4	1/16			II	I	1110
b_3	1/8		II	I		110
b_2	1/4	II	I			10
b_1	1/2	I				0

The obtained codewords will be used for $\mathcal{A}_5$.

Thus, we consider $c(a_5) = 1111$, $c(a_4) = 1110$, $c(a_3) = 110$, $c(a_2) = c(b_2) = 10$, and $c(a_1) = 0$.

Example 2: We consider an alphabet that consists of three letters,

$$\mathcal{A}_3 = \{a_1, a_2, a_3\}$$

with the probability data

$$\{p_1, p_2, p_3\} = \{0.8, 0.18, 0.02\}.$$

The entropy of $\mathcal{A}_3$ is $\varepsilon = 0.81573$bit/symbol.

The following table shows the above algorithm for encoding $\mathcal{A}_3$:

Table 8.

$\mathcal{A}_3$	ε_i	p_i	D_i	$\bar{p}_i$	l'_i	$\bar{l}_i$
a_1	0.25754	0.80	0.30	0.50	0.3	1
a_2	0.44531	0.18	0.20	0.25	1.2	2
a_3	0.11288	0.02	0	0.25	2.0	2

The rate of the coding $\tilde{l} = 0.8 \cdot 1 + 0.18 \cdot 2 + 0.02 \cdot 2 = 1.20 \, bit/symbol$, and the redundancy for the coding is $R = 0.38427$bit/symbol.

Example 3: Consider the alphabet $\mathcal{A}_9$ of nine letters, which is the extended alphabet composed by all groups of two letters of the alphabet $\mathcal{A}_3$ considered above.

The probability data and the outputs of the proposed algorithm for encoding $\mathcal{A}_9$:

Table 9.

$\mathcal{A}_9$	p_i	ε_i	D_i	$\bar{p}_i$	l'_i	$\bar{l}_i$
a_1	0.6400	0.41207	0.14000	0.5	0.6	1
a_2	0.1440	0.31143	0.07500	0.125	2.3	3
a_3	0.1440	0.23105	0	0.250	2.0	2
a_4	0.0324	0.09608	0.25000	0.03125	4.2	5
a_5	0.0160	0.06329	0.00663	0.03125	4.7	5
a_6	0.0160	0.06280	0.01112	0.03125	4.6	5
a_7	0.0036	0.02131	0.00699	0.00781	6.1	7
a_8	0.0036	0.02105	0.00547	0.01563	5.6	6
a_9	0.0004	0.00305	0.00391	0.00391	7.0	7

$$(\tilde{l} = 1.73160) - (\varepsilon = 1.63145) = 0.10015$$

and variance $\sigma(\tilde{l}) = 1.15467$.

It should be noted for comparison that the Huffman code results in codewords with the sequence of length

$$\mathcal{L}_H = \{1, 3, 2, 4, 5, 6, 7, 8, 8\}.$$

The Huffman code yields the better coding than the proposed algorithm, in terms of the minimal average codeword length, which is

$$\tilde{l}_H = 1.72280\text{bit/symbol} < \tilde{l} = 1.73160$$

but the variance of the Huffman code $\sigma(\tilde{l}_H) = 1.32919$ is greater than variance $\sigma(\tilde{l}) = 1.15467$ of the proposed code.

By starting with symbol a_9, we can reduce the minimal average codeword length as the Huffman code does.

The both codes are optimal, and for them the following equality holds

$$\sum_{i=1}^{9} 2^{-\bar{l}_i} = \Sigma\left\{2^{-L_i};\ L_i \in \mathcal{L}_H,\ i = 1:9\right\} = 1.$$

Conclusion

The proposed manageable algorithm is a consequent procedure for encoding the symbols from a alphabet which probability data is known.

The encoding is reduced to encoding another alphabet of letters whose probabilities are redistributed between letters which already have been encoded and rest part of letters to be encoded.

The procedure is very simple in the standpoint of calculation and reduces the problem of encoding to the problem of finding the optimal lengths for codewords.

The example of the MATLAB®-based program for computing codewords is given below.

```
% gen_code.m file of program for MATLAB 5
% To generate codewords of given lengths, L,
% that are sorted in the ascending order
%
% (C) Copyright 2003, Serkan Dursun, EE UTSA
%
function C=gen_code(L)
L=sort(L);
N=length(L);
C=cell(N,1);   % space for codewords
same =  sum(L == L(1)) == length(L);
w(1)=0;
for j=2:N
    w(j)=2^L(j)*sum(2.^-L(1:j-1));
end;
if same
    C = dec2bin(w);
else
    C{1}=dec2bin(w(1),L(1));
    for j=2:N
        if ceil(log2(w(j))) == L(j)
           C{j} = dec2bin(w(j),L(j));
        else
           C{j} = strcat(dec2bin(w(j),2),...
               dec2bin(0,L(j)-...
               length(dec2bin(w(j),2))));
        end;
    end;
end;
%  Example:
%  L=[1 2 3 4 4];
%  C=gen_code(L)'
%  '0'    '10'    '110'    '1110'    '1111'
```