ART code I: A New Method Of Optimal Coding

Artyom M. Grigoryan
Department of Electrical and
Computer Engineering
The University of Texas at San Antonio
amgrigoryan@utsa.edu

Notes for class 4663 from the presentation in the International IEEE conference ITCC-2002

. . .

A new technique for optimally encoding a given source, statistical properties of which are described by the first-order model is introduced.

The calculation of a minimum length of codewords is based on the consecutive redistribution of the self-information of symbols in accordance with their probabilities at each stage of the encoding.

The proposed method performs equally well for an arbitrary order of symbol probabilities.

While codewords are generated by a separate combinatorial procedure, the overall computational cost of the proposed method is lower than that for the Huffman code.

...

For a given alphabet $\mathcal{A}_m = \{a_1, a_2, ..., a_m\}$ of letters a_i, $i = 1, 2, \ldots, m > 1$, which have probabilities $p_i > 0$, and for which $p_1 + \ldots + p_m = 1$, the entropy rate of $\mathcal{A}_m$ is

$$\varepsilon = \sum_{i=1}^{m} \varepsilon_i = \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}, \quad \varepsilon_i = p_i \log_2 \frac{1}{p_i}$$

where ε_i denotes the self-information of a_i.

$l_i = \log \dfrac{1}{p_i}$ bits must be assigned to a_i.

Since only integer numbers of bits are taken, a difference d_i may occur between the actual number l_i and rounded integer value $[l_i]$,

$$l_i = \log_2 \frac{1}{p_i} = [l_i] - d_i\,,\ [l_i] = 1, \ldots,\ 0 \le d_i < 1.$$

...

Letters of the alphabet $\mathcal{A}_m$ are arranged in descending order of their probabilities. Starting with a letter a_1, a length

$$[l_1] = [\log \frac{1}{p_1}]$$

of the codeword $c(a_1)$ for a_1 is calculated.

The difference D_1 of the self-information of the codeword and letter a_1

$$D_1 = \widetilde{\varepsilon}_1 - \varepsilon_1 = [l_1]p_1 - \varepsilon_1$$

is distributed among the remaining letters $a_2, a_3, \ldots, a_m$ in proportion to their probabilities.

If $D_1 = 0$ (or $l_1 = [l_1]$), then a_2 is processed.

If $D_1 \neq 0$, the self-information of the rest letters are calculated as

$$\varepsilon'_{k+1} = \varepsilon_{k+1} - \frac{D_1}{\sum\limits_{n>1} p_n} \cdot p_{k+1}, \;\; k = 1 : (m-1)$$

...

The self-information ε_2 of a_2 becomes

$$\varepsilon_2' = \varepsilon_2 - \frac{D_1}{\sum\limits_{n>1} p_n} \cdot p_2$$

A new length l_2' of codeword c_2 will be calculated as

$$l_2' = \frac{\varepsilon_2'}{p_2}$$

and $[l_2']$ bits will be assigned for the codeword of letter a_2.

The remainder of the self-information of the letter a_2

$$D_2 = \tilde{\varepsilon}_2 - \varepsilon_2' = p_2[l_2'] - \varepsilon_2'$$

is distributed among the remaining $(m-2)$ letters $a_3, a_4, \ldots, a_m$ in proportion to their probabilities.

At following steps, the letters $a_3, a_4, \ldots, a_m$ are processed similarly.

$\dots$

The algorithm results in the set of the numbers of bits $[l_1], [l_2], ..., [l_m]$ which are supposed to be used for encoding the corresponding letters $a_1, a_2, ..., a_m$.

If the Kraft-McMillan condition holds

$$\frac{1}{2^{[l_1]}} + \frac{1}{2^{[l_2]}} + \cdots + \frac{1}{2^{[l_m]}} > 1$$

there exists an uniquely decodable procedure for encoding the alphabet $\mathcal{A}_m$, for which $[l_i]$ bits will be used to obtain codewords $c(a_i)$, $i = 1, \dots, m$.

Example: The alphabet $\mathcal{A}_5 = \{a_1, a_2, ..., a_5\}$ whose elements have the probabilities $p_1 = 0.4$, $p_2 = p_3 = 0.2$, and $p_4 = p_5 = 0.1$.

The entropy rate of $\mathcal{A}_5$ is $\varepsilon = 2.122$bits/letter.

Step 1: Letter is a_1, $p_1 = 0.4$, $l_1 = -\log p_1 = 1.3219$, $\varepsilon_1 = 0.52877$, and $[l_1] = 2$ bits are assigned to encode a_1.

The self information of the letter a_1 increases by the value

$$D_1 = p_1[l_1] - \varepsilon_1 = 0.4 \cdot 2 - 0.52877 = 0.27123$$

This amount of the self-information is subtracted from the self-information of the remaining letters $\varepsilon_2, \varepsilon_3, \varepsilon_4$, and ε_5, in accordance with their probabilities, i.e. respectively in proportions $D_1/3$, $D_1/3$, $D_1/6$, and $D_1/6$.

The renewed values of the self-information of letters are defined as follows:

$$\varepsilon_1^{(2)} = \widetilde{\varepsilon}_1 = p_1[l_1] = 0.4 \cdot 2 = 0.8$$
$$\varepsilon_2^{(2)} = \varepsilon_2 - \frac{D_1}{0.6} \cdot 0.2 = 0.46439 - 0.09041$$
$$\varepsilon_3^{(2)} = \varepsilon_3 - \frac{D_1}{0.6} \cdot 0.2 = 0.46439 - 0.09041$$
$$\varepsilon_4^{(2)} = \varepsilon_4 - \frac{D_1}{0.6} \cdot 0.1 = 0.33219 - 0.04521$$
$$\varepsilon_5^{(2)} = \varepsilon_5 - \frac{D_1}{0.6} \cdot 0.1 = 0.33219 - 0.04521$$

The initial and processed data at this step are shown in the following table:

Table1 : Step 1.

$\mathcal{A}$	p_i	ε_i	D_i	$\varepsilon_i^{(2)}$	l'_i	$[l'_i]$
1	.4	.52877	.27123	.8	1.3219	**2**
2	.2	.46439	$-1/3\cdot$	.37398		
3	.2	.46439	$-1/3\cdot$	.37398		
4	.1	.33219	$-1/6\cdot$	.28698		
5	.1	.33219	$-1/6\cdot$	.28698		

Step 2: Letter is a_2, $p_2 = 0.2$, and the self-information of a_2 becomes $\varepsilon_2^{(2)} = 0.37398$.
$l_2' = \varepsilon_2^{(2)}/p_2 = 1.86985$, and $[l_2'] = 2$ bits are assigned for the letter a_2.
The difference of the self-information

$$D_2 = p_2[l_2'] - \varepsilon_2^{(2)} = 0.2 \cdot 2 - 0.37398 = 0.02602$$

is subtracted from the self-inf. of letters

$$\varepsilon_3^{(3)} = \varepsilon_3^{(2)} - \frac{D_2}{0.4} \cdot 0.2 = 0.37398 - 0.01302$$
$$\varepsilon_4^{(3)} = \varepsilon_4^{(2)} - \frac{D_2}{0.4} \cdot 0.1 = 0.28698 - 0.00651$$
$$\varepsilon_5^{(3)} = \varepsilon_5^{(2)} - \frac{D_2}{0.4} \cdot 0.1 = 0.28698 - 0.00651$$

and the new data table takes the form

Table 2 : Step 2.

$\mathcal{A}$	p_i	$\varepsilon_i^{()}$	D_i	$\varepsilon_i^{(3)}$	l_i'	$[l_i']$
1	.4	.52877	.27123	.8	1.3219	2
2	.2	.37398	.02602	.4	1.8698	**2**
3	.2	.37398	$-1/2\cdot$	.36096		
4	.1	.28698	$-1/4\cdot$	.28047		
5	.1	.28698	$-1/4\cdot$	.28047		

Tables : Steps 3, 4, and 5.

$\mathcal{A}$	p_i	$\varepsilon_i^{()}$	D_i	$\varepsilon_i^{(4)}$	l'_i	$[l'_i]$
1	.4	.52877	.27123	.8	1.3219	2
2	.2	.37398	.02602	.4	1.8698	2
3	.2	.36096	.03904	.4	1.8048	**2**
4	.1	.28047	$-1/2\cdot$	.26096		
5	.1	.28047	$-1/2\cdot$	.26096		

$\mathcal{A}$	p_i	$\varepsilon_i^{()}$	D_i	$\varepsilon_i^{(5)}$	l'_i	$[l'_i]$
1	.4	.52877	.27123	.8	1.3219	2
2	.2	.37398	.02602	.4	1.8698	2
3	.2	.36096	.03904	.4	1.8048	2
4	.1	.26096	.03904	.3	2.6096	**3**
5	.1	.26096	$-1\cdot$	.22192		

$\mathcal{A}$	p_i	$\varepsilon_i^{()}$	D_i	$\widetilde{\varepsilon}_i$	l'_i	$[l'_i]$
1	.4	.52877	.27123	0.8	1.3219	2
2	.2	.37398	.02602	0.4	1.8698	2
3	.2	.36096	.03904	0.4	1.8047	2
4	.1	.26096	.03904	0.3	2.6096	3
5	.1	.22192	**.07808**	0.3	2.2192	**3**

...

The last remainder D_5 of the self-information of a_5 is equal to the redundancy of the code

$$R = \sum_{k=1}^{5} p_k[l'_k] - \varepsilon = D_5 = 0.07808 \text{ bits/letter}$$

The lengths of the codewords are $\{2, 2, 2, 3, 3\}$.

Therefore 12 bits are required for encoding the alphabet $\mathcal{A}_5$ by using the proposed method.

Indeed, the Kraft-McMillan inequality is fulfilled and the following code can be considered: $c(a_1) = 00$, $c(a_2) = 01$, $c(a_3) = 10$, $c(a_4) = 110$, $c(a_5) = 111$. Other codes are $\{11, 00, 01, 100, 101\}$, $\{10, 11, 00, 010, 011\}$, and $\{01, 10, 11, 000, 001\}$.

The variance of the length for this codes is 0.23664 and the average length for codeword is 2.20 bits/letter. That is, the encoding procedure is optimal.

...

The letter probabilities are in increasing order:

Table 6.

$\mathcal{A}$	p_i	$\varepsilon_i^{()}$	D_i	$\widetilde{\varepsilon}_i$	l_i'	$[l_i']$
5	.1	.33219	.06781	.4	3.32193	4
4	.1	.32466	.07534	.4	3.24659	4
3	.2	.43048	.16952	.6	2.15241	3
2	.2	.37398	.02602	.4	1.86988	2
1	.4	.32192	**.07808**	.4	0.80480	1

The sequence of lengths for codewords is $\{4, 4, 3, 2, 1\}$ which requires 14 bits for encoding the alphabet $\mathcal{A}_5$.

The variance of the codeword length is 0.58652 which is greater than the variance obtained in the previous example, but the average codeword length is the same, 2.20 bits/letter.

The order of letters $a_1, a_2, ..., a_5$ is not essential for the proposed procedure. For instance, by processing letters in the sequences a_1, a_2, a_4, a_3, a_5 and a_1, a_4, a_5, a_2, a_3, respectively the following data tables are obtained:

Table 7.

$\mathcal{A}$	p_i	$\varepsilon_i^{()}$	D_i	$\widetilde{\varepsilon}_i$	l_i'	$[l_i']$
1	.4	.52877	.27123	.8	1.32193	2
2	.2	.37398	.02602	.4	1.86988	2
4	.1	.28048	.01952	.3	2.80482	3
3	.2	.34795	.05205	.4	1.73976	2
5	.1	.22192	.07808	.3	2.21928	3

Table 8.

$\mathcal{A}$	p_i	$\varepsilon_i^{()}$	D_i	$\widetilde{\varepsilon}_i$	l_i'	$[l_i']$
1	.4	.52877	.27123	.8	1.32193	2
4	.1	.28699	.01301	.3	2.86988	3
5	.1	.28439	.01561	.3	2.84386	3
2	.2	.36096	.03904	.4	1.80482	2
3	.2	.32192	.07808	.4	1.60964	2

The lengths for codewords are $\{2, 2, 2, 3, 3\}$.

Let us consider another example, when letters are taken in the order a_2, a_1, a_4, a_3, a_5.

Table 9.

$\mathcal{A}$	p_i	ε_i	$\varepsilon_i^{()}$	D_i	$\widetilde{\varepsilon}_i$	l'_i	$[l'_i]$
2	.2	.46439	.46439	.13561	.6	2.32193	3
1	.4	.52877	.46096	.33904	.8	1.15241	2
4	.1	.33219	.23048	.06952	.3	2.30482	3
3	.2	.46439	.21462	.18538	.4	1.07309	2
5	.1	.33219	.02192	.07808	.3	0.21928	1

The sequence of lengths 2, 3, 2, 3, 1 for codewords $c(a_1), ..., c(a_5)$, for which no decodable code exists. The Kraft-McMillan inequality does not hold

$$\sum_{i=1}^{5} \frac{1}{2^{[l'_i]}} = \frac{1}{2^3} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^2} + \frac{1}{2} = \frac{5}{4} > 1.$$

Since, $p_2 < p_1$, $\varepsilon_2 < \varepsilon_1$, and $l_2' = 2.32193$, we have to assign only two bits for the code-word $c(a_2)$ and add the remainder of its self-information, $-D_2 = \varepsilon_2 - 2 \cdot 0.2 = 0.06439$, to the letters a_1, a_4, a_3, a_5 in accordance with their probabilities. In other words, we consider that $l_2 = [l_2] + m_2$, where $0 \leq m_2 < 1$.

Table 10.

$\mathcal{A}$	p_i	ε_i	$\varepsilon_i^{()}$	D_i	$\widetilde{\varepsilon}_i$	l_i'	$[l_i']$
2	.2	.46439	.46439	−.06439	.4	2.32193	2
1	.4	.52877	.56097	.23904	.8	1.15241	2
4	.1	.33219	.28048	.01952	.3	2.84383	3
3	.2	.46439	.34795	.05205	.4	1.07309	2
5	.1	.33219	.22192	.07808	.3	2.21920	3

The codeable sequence of lengths is $2, 2, 2, 3, 3$.

... **Conclusions**

A new approach was presented for computing the optimal lengths for codewords of a given source which has the first-order model.

The main idea of this approach is based on transferring and redistributing the self-informations of encoded symbols to the remaining symbols to be encoded, at each stage of the calculation.

The algorithm is simple in comparison with the Huffman code and provides the optimal encoding of the source irrespective of the ordering of symbol probabilities. Due to the simplicity of the proposed method, it can be used in real-time applications, as well as in applications that demand fixed transmission rates.