ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/232619216

Reducing the Length of Shannon-Fano-Elias Codes and Shannon-Fano Codes

Article · October 2006

DOI: 10.1109/MILCOM.2006.302404

CITATION
1

READS
158

2 authors, including:

Rajendra Katti
North Dakota State University
70 PUBLICATIONS **515** CITATIONS

SEE PROFILE

All content following this page was uploaded by Rajendra Katti on 30 January 2014.

The user has requested enhancement of the downloaded file.

Reducing the Length of Shannon-Fano Codes and Shannon-Fano-Elias Codes

Xiaoyu Ruan, *Student Member, IEEE,* and Rajendra S. Katti, *Member, IEEE*

***Abstract*— Shannon-Fano codes and Shannon-Fano-Elias codes are widely used in communications. We introduce simple methods for reducing the length of Shannon-Fano codes and Shannon-Fano-Elias codes. The main idea is to utilize some proper probability mass function (PMF), instead of the actual PMF, to perform the encoding.**

***Index Terms*— Data compression, expected code length, probability mass function, Shannon-Fano coding, Shannon-Fano-Elias coding.**

I. Shannon-Fano Coding

Take the source symbol set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, assume for the probability mass function (PMF) we have $p(x_i) > 0$ for all symbols in $\mathcal{X}$. The cumulative distribution function $F(p(x_i))$ is defined as

$$F(p(x_i)) = \sum_{k=1}^{i-1} p(x_k) \tag{1}$$

The Shannon-Fano coding [1] requires that the probabilities be listed non-increasingly. In general $F(p(x_i))$ is a real number only expressible by an infinite number of bits. We round off the binary expansion of $F(p(x_i))$ to $l(x_i) = \lceil \log \frac{1}{p(x_i)} \rceil$ bits, denoted by $\lfloor F(p(x_i)) \rfloor_{l(x_i)}$. It can be shown that the set of lengths $l(x_i)$ satisfies the Kraft inequality [2] and hence can be used to construct a uniquely decodable code. Shannon-Fano coding uses $\lfloor F(p(x_i)) \rfloor_{l(x_i)}$ as the codeword for symbol x_i. The resulting codewords are prefix-free.

For example, if $p(x_1) = 0.23$, $p(x_2) = 0.22$, $p(x_3) = 0.21$, $p(x_4) = 0.20$, $p(x_5) = 0.10$, $p(x_6) = 0.02$, and $p(x_7) = 0.02$, then the codewords are 000 for x_1, 001 for x_2, 011 for x_3, 101 for x_4, 1101 for x_5, 111101 for x_6, and 111110 for x_7, respectively.

II. Reducing the Length of Shannon-Fano Codes

In Shannon-Fano coding we do not have to use the actual PMF $p(x_i)$. In this manuscript, the PMF used to encode the source is denoted by $q(x_i)$. In order to reduce the length of a Shannon-Fano encoded sequence, we use $q(x_i)$ instead of $p(x_i)$ for encoding.

First we define a value to evaluate how good a PMF $q(x_i)$ is. The *expected length* L is defined as

$$L = \sum_{i=1}^{n} p(x_i) l(x_i) \tag{2}$$

This work has been supported in part by the National Science Foundation under Grant CCR-0429523.

The authors are with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58105-5285, USA. (e-mail: xiaoyu.ruan@ndsu.edu; rajendra.katti@ndsu.edu)

where each $l(x_i)$ for $1 \le i \le n$ is a positive integer, representing the codeword length for symbol x_i when $q(x_i)$ is used to encode the source. According to the construction rule of Shannon-Fano codes, $l(x_i) = \lceil \log \frac{1}{q(x_i)} \rceil$.

Minimizing L involves the following optimization problem:

$$\min \qquad L = \sum_{i=1}^{n} p(x_i) l(x_i) \tag{3}$$

$$\text{subject to} \qquad \sum_{i=1}^{n} 2^{-l(x_i)} \le 1. \tag{4}$$

The constraint is from Kraft inequality [2]. This is a standard optimization problem. Using the Lagrange multiplier we get

$$l(x_i) = \log \frac{1}{p(x_i)}. \tag{5}$$

If each $p(x_i)$ for $1 \le i \le n$ has a form of 2^{-k_i}, then $l(x_i) = \lceil \log \frac{1}{p(x_i)} \rceil = \log \frac{1}{p(x_i)} = k_i$. In this case we use the actual PMF for encoding.

In most cases, not every $p(x_i)$ for $1 \le i \le n$ has a form of 2^{-k_i}. i.e., some values of $\log \frac{1}{p(x_i)}$ are not integers. Shannon-Fano coding rounds $\log \frac{1}{p(x_i)}$ up to $\lceil \log \frac{1}{p(x_i)} \rceil$. This rounding up results in an increase in L. The amount of increase is $\sum_{i=1}^{n} p(x_i)(\lceil \log \frac{1}{p(x_i)} \rceil - \log \frac{1}{p(x_i)})$. To reduce L, rounding up should be avoided.

Notice that symbol x_i with probability $p(x_i)$ satisfying $2^{-k_i} \le p(x_i) < 2^{-(k_i-1)}$ where $k_i \ge 1$ has codeword length k_i. The main strategy of our method is to use probability $q(x_i) = 2^{-m_i}$ where $m_i \ge 1$ for x_i. If $m_i > k_i$, then the codeword length for x_i is increased by $m_i - k_i$ and L is increased by $p(x_i)(m_i - k_i)$. If $m_i = k_i$, then the codeword length for x_i is not changed and L is not changed either. If $m_i < k_i$, then the codeword length for x_i is decreased by $k_i - m_i$ and L is decreased by $p(x_i)(k_i - m_i)$.

Now the problem becomes to

$$\max \quad \Delta L = \sum_{i=1}^{n} p(x_i)(k_i - m_i) \tag{6}$$

$$\text{subject to} \qquad \sum_{i=1}^{n} 2^{-m_i} \le 1. \tag{7}$$

We discuss three approaches to solve this problem.

1) Computer Search: Theoretically, given a PMF $p(x_i)$, one can always figure out an optimal set of m_i $(1 \le i \le n)$ using exhaustive computer search. However, the workload of computation dramatically increases as n becomes large.

TABLE I
EXAMPLE FOR ALGORITHM 1

i	$p(x_i)$	Codeword	$q(x_i)$	Codeword	ω
0					0.40625
1	0.23	000	0.5	0	0.03125
2	0.22	001	0.125	100	0.03125
3	0.21	011	0.125	101	0.03125
4	0.20	101	0.125	110	0.03125
5	0.10	1101	0.0625	1110	0.03125
6	0.02	111101	0.03125	11110	0.015625
7	0.02	111110	0.03125	11111	0

2) Suboptimal Algorithm: To avoid solving this maximization problem directly, we give a "suboptimal" solution in Algorithm 1.

Algorithm 1, ω is a real number and s an integer.

Algorithm 1 Computing suboptimal PMF $q(x_i)$ for Shannon-Fano codes

Input: PMF $p(x_i)$ where $2^{-k_i} \leq p(x_i) < 2^{-(k_i-1)}$ for $1 \leq i \leq n$ and $p(x_1) \geq p(x_2) \geq \ldots \geq p(x_n)$.
Output: PMF $q(x_i)$.
  $\omega \leftarrow \sum_{i=1}^{n}(p(x_i) - 2^{-k_i})$
  $i \leftarrow 0$
  while $i < n$ **do**
    $i \leftarrow i + 1$
    $r \leftarrow \max\{s : 2^{-(k_i-s)} - 2^{-k_i} \leq \omega\}$
    $q(x_i) \leftarrow 2^{-(k_i-r)}$
    $\omega \leftarrow \omega - (2^{-(k_i-r)} - 2^{-k_i})$
  end while

To illustrate Algorithm 1, an example is shown in Table I. PMF $p(x_i)$ is the same as in Section I. If $p(x_i)$ is used to encode the source, then $L = 0.23\times 3+0.22\times 3+0.21\times 3+0.20\times 3+0.10\times 4+0.02\times 6+0.02\times 6 = 3.22$. Using $q(x_i)$ to encode the source we have $L = 0.23\times 1+0.22\times 3+0.21\times 3+0.20\times 3+0.10\times 4+0.02\times 5+0.02\times 5 = 2.72 < 3.22$. Notice that the entropy for this source is $H = 2.35$. The lower bound of L remains H and the upper bound remains $H+1$. Our method reduces L but does not change the bounds of L.

3) Empirical Solutions: Consider solving

$$\sum_{i=1}^{n} q(x_i) = 1 \tag{8}$$

where $q(x_i) = 2^{-m_i}$. Assume $2^k < n \leq 2^{k+1}$, we give two "extreme" empirical solution sets and discuss their behavior.

Set I:

$$q(x_i) = \begin{cases} 2^{-k} & \text{if } 1 \leq i \leq 2^{k+1} - n \\ 2^{-(k+1)} & \text{if } 2^{k+1} - n + 1 \leq i \leq n \end{cases} \tag{9}$$

Set II:

$$q(x_i) = \begin{cases} 2^{-i} & \text{if } 1 \leq i \leq n-1 \\ 2^{1-i} & \text{if } i = n \end{cases} \tag{10}$$

Notice that if $n = 2^{k+1}$ then Set I gives $q(x_i) = n^{-1}$ for $1 \leq i \leq n$.

As an example, consider $n = 7$. Set I gives $q(x_1) = 0.25$ and $q(x_2) = q(x_3) = q(x_4) = q(x_5) = q(x_6) = q(x_7) = 0.125$. Set II gives $q(x_1) = 0.5$, $q(x_2) = 0.25$, $q(x_3) = 0.125$, $q(x_4) = 0.0625$, $q(x_5) = 0.03125$, and $q(x_6) = q(x_7) = 0.015625$. In both cases the values of $q(x_i)$ are arranged in non-increasing order just like $p(x_i)$. Therefore large $q(x_i)$ (i.e., small m_i) are always assigned to x_i with large $p(x_i)$. This ensures that L is reduced as much as possible.

We now analyze $p(x_i)$. In general, if $p(x_i)$ is almost uniformly distributed, i.e., most values of $p(x_i)$ are close to n^{-1}, then we shall use Set I to encode the source. On the other hand, if $p(x_i)$ is widely dispersed, i.e., some values of $p(x_i)$ are large and some are small, then we shall use Set II to encode the source.

Consider an example. Let PMF $p(x_i)$ be the same as in Section I. If $p(x_i)$ is used to encode the source, then $L = 3.22$. If we use Set I of $q(x_i)$ then $L = 0.23\times 2+0.22\times 3+0.21\times 3+0.20\times 3+0.10\times 3+0.02\times 3+0.02\times 3 = 2.77$. If we use Set II of $q(x_i)$ then $L = 0.23\times 1+0.22\times 2+0.21\times 3+0.20\times 4+0.10\times 5+0.02\times 6+0.02\times 6 = 2.84$. Thus we see that $p(x_i)$ is more uniformly distributed than widely dispersed, and using either Set I or Set II can reduce L. However, neither of them receives an optimal L. If we use $q(x_1) = q(x_2) = 0.25$, $q(x_3) = q(x_4) = q(x_5) = 0.125$, and $q(x_6) = q(x_7) = 0.0625$, then $L = 0.23\times 2+0.22\times 2+0.21\times 3+0.20\times 3+0.10\times 3+0.02\times 4+0.02\times 4 = 2.59$, which is the optimal L.

Notice that the values of $q(x_i)$ of Set I and Set II depend only on n but do not depend on $p(x_i)$. Therefore, compared to Algorithm 1, the empirical method does not require any computation but relies on a correct judgement on $p(x_i)$. It should be emphasized that a wrong judgement on $p(x_i)$ may lead to an increase on L. Suppose $p(x_1) = 0.80$, $p(x_2) = 0.10$, and $p(x_3) = p(x_4) = 0.05$. Using $q(x_i)$ fop encoding, we get $L = 0.80\times 1+0.10\times 4+0.05\times 5+0.05\times 5 = 1.7$. It is obvious that Set II should be employed because the values of $p(x_i)$ are widely dispersed. In case Set I is used by mistake then $L = 0.80\times 2+0.10\times 2+0.05\times 2+0.05\times 2 = 2 > 1.7$.

III. SHANNON-FANO-ELIAS CODING

Take the source symbol set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, assume for the probability mass function (PMF) we have $p(x_i) > 0$ for all symbols in $\mathcal{X}$. The modified cumulative distribution function is defined as

$$\overline{F}(p(x_i)) = \sum_{k=1}^{i-1} p(x_k) + \frac{1}{2}\, p(x_i) \tag{11}$$

$\overline{F}$ represents the sum of probabilities of all symbols placed "before" x_i plus half of the probability of x_i.

Since the random variable is discrete, the cumulative function consists of steps of size $p(x_i)$. Therefore we can determine x_i if we know $\overline{F}(p(x_i))$. In general $\overline{F}(p(x_i))$ is a real number only expressible by an infinite number of bits. We round off $\overline{F}(p(x_i))$ to $l(x_i) = \lceil \log \frac{1}{p(x_i)} \rceil + 1$ bits, denoted by $\lfloor \overline{F}(p(x_i)) \rfloor_{l(x_i)}$. It can be shown that the set of lengths $l(x_i)$ satisfies the Kraft inequality [2] and hence can be used to construct a uniquely decodable code. $\lfloor \overline{F}(p(x_i)) \rfloor_{l(x_i)}$ is within

TABLE II
SHANNON-FANO-ELIAS CODING FOR ENGLISH LETTERS ACCORDING TO THE ALPHABET ORDER

x_i	Letter	$p(x_i)$	Codeword
x_1	A	0.081	00001
x_2	B	0.015	00010110
x_3	C	0.028	0001110
x_4	D	0.043	001001
x_5	E	0.127	0011
x_6	F	0.022	0100111
x_7	G	0.020	0101001
x_8	H	0.061	010111
x_9	I	0.070	01101
x_{10}	J	0.002	0111011111
x_{11}	K	0.008	01111001
x_{12}	L	0.040	011111
x_{13}	M	0.024	1000011
x_{14}	N	0.067	10010
x_{15}	O	0.075	10100
x_{16}	P	0.019	1011000
x_{17}	Q	0.001	10110011110
x_{18}	R	0.060	101110
x_{19}	S	0.063	11001
x_{20}	T	0.091	11011
x_{21}	U	0.028	1110111
x_{22}	V	0.010	11110011
x_{23}	W	0.023	1111011
x_{24}	X	0.001	11111010011
x_{25}	Y	0.020	1111110
x_{26}	Z	0.001	11111111110

the step corresponding to x_i. Thus it is good enough to use only the first $l(x_i)$ bits of $\overline{F}(p(x_i))$ to describe x_i. Shannon-Fano-Elias coding [1] uses $\lfloor \overline{F}(p(x_i)) \rfloor_{l(x_i)}$ as the codeword for x_i. The resulting codewords are prefix-free.

Consider an example. Probabilities of the 26 English letters occurring in literature are shown in the third column of Table II. The corresponding codewords for the letters ordered from A to Z are listed in the fourth column of Table II.

IV. REDUCING THE LENGTH OF SHANNON-FANO-ELIAS CODES

One main feature of Shannon-Fano-Elias coding is that, unlike Shannon-Fano coding, the probabilities of the source symbols do not have to be ordered non-increasingly. During the encoding operation it is impossible to predict how many symbols are left and what their probabilities are.

Like Shannon-Fano coding, Shannon-Fano-Elias coding does not require us to use the actual PMF. We still denote the actual PMF by $p(x_i)$ and the PMF used to encode the source by $q(x_i)$ where $1 \le i \le n$.

Recall that the problem for reducing L is to

$$\max \quad \Delta L = \sum_{i=1}^{n} p(x_i)(k_i - m_i) \tag{12}$$

$$\text{subject to} \qquad \sum_{i=1}^{n} 2^{-m_i} \le 1. \tag{13}$$

We have seen that by rounding up the probability $p(x_i)$ where $2^{-k_i} \le p(x_i) < 2^{-(k_i-1)}$ and $k_i \ge 1$ to $q(x_i) = 2^{-m_i}$, the codeword for x_i can be shortened by $k_i - m_i$ bits. This is the foundation of the method.

In Algorithm 2, ω is a real number and s an integer.

TABLE III
EXAMPLE FOR ALGORITHM 2

i	$p(x_i)$	Codeword	$q(x_i)$	Codeword	ω
1	0.02	0000001	0.015625	0000001	0.004375
2	0.23	0010	0.125	0001	0.109375
3	0.21	0101	0.125	0011	0.194375
4	0.10	10000	0.25	011	0.044375
5	0.22	1010	0.25	101	0.014375
6	0.02	1100101	0.03125	110010	0.003125
7	0.20	1110	0.125	1101	0.078125

Algorithm 2 Computing suboptimal PMF $q(x_i)$ for Shannon-Fano-Elias codes

Input: PMF $p(x_i)$ where $2^{-k_i} \le p(x_i) < 2^{-(k_i-1)}$ for $1 \le i \le n$.
Output: PMF $q(x_i)$.
  $\omega \leftarrow 0$
  $i \leftarrow 0$
  while $i < n$ **do**
    $i \leftarrow i + 1$
    $r \leftarrow \max\{s : 2^{-(k_i-s)} - p(x_i) \le \omega\}$
    $q(x_i) \leftarrow 2^{-(k_i-r)}$
    $\omega \leftarrow \omega - (2^{-(k_i-r)} - p(x_i))$
  end while

Using Algorithm 2, a portion of ω is produced at the end of the encoding operation. It can be thought of as the probability of a dummy symbol x_{n+1} which is not used. Therefore ω does not affect the properties of Shannon-Fano-Elias codes.

To illustrate Algorithm 2, an example is shown in Table III. If $p(x_i)$ is used to encode the source, then $L = 0.02 \times 7 + 0.23 \times 4 + 0.21 \times 4 + 0.10 \times 5 + 0.22 \times 4 + 0.02 \times 7 + 0.20 \times 4 = 4.22$. Using $q(x_i)$ to encode the source we have $L = 0.02 \times 7 + 0.23 \times 4 + 0.21 \times 4 + 0.10 \times 3 + 0.22 \times 3 + 0.02 \times 6 + 0.20 \times 4 = 3.78 < 4.22$. Notice that the entropy for this source is $H = 2.35$. The lower bound of L remains $H+1$ and the upper bound remains $H + 2$. Our method reduces L but does not change the bounds of L.

V. CONCLUSION

We have proposed simple methods for reducing the length of Shannon-Fano codes and Shannon-Fano-Elias codes. The idea is to use an alternative PMF instead of the actual one in the encoding operation. The methods do not require much computation and hence are useful in improving the efficiency of Shannon-Fano codes and Shannon-Fano-Elias codes.

REFERENCES

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.
[2] B. McMillan, Two inequalities implied by unique decipherability. *IEEE Trans. Inform. Theory*, IT-2: 115-116, 1956.

View publication stats