Variable-length codes for sources with equiprobable symbols

Boris Assanovich

Ulrich Güenther

CDMTCS-103
May 1999

Variable-length codes for sources with equiprobable symbols

Boris Assanovich, Ulrich Güenther
bas@grsu.grodno.by, ulrich@ihug.co.nz

May 16, 1999

Keywords: Variable-length codes, equal probability letters, average code length, digital channels.

Abstract

Variable-length codes can provide compression for data communication. Such codes may be used not only when the source statistics is known but also when we do not know the source probability distribution, and a source with equal symbol probabilities (equiprobable symbols) can or has to be assumed. This paper presents variable-length codes with code words that differ in length by at most one code symbol. Such codes suit the efficient encoding of sources with equiprobable symbols. We accommodate non-binary codes and present an iterative algorithm for the construction of such codes. We also calculate the average codeword length for such codes, which extends Krichevski's result for binary codes [5]. Finally, we propose a scheme that allows the code to be communicated efficiently from transmitter to receiver.

1 Introduction

Variable-length codes may be used for data compression in serial communication. In these cases, the encoding is to an extent determined by the frequencies of occurrence of the source symbols. A well-known example of this approach is the algorithm for the construction of Huffman codes [4, 5]. If we have source symbols with equal probabilities, the resulting Huffman code will have codewords that differ at most by one code symbol from each other. This means that we may use alternative construction mechanisms for the code, such as the two algorithms presented in the next section.

2 Code Construction

The first construction algorithm we present here is an iterative algorithm. It is similar to the recursive "copy-k-times-and-append" algorithm called T-augmentation, which is used in the construction of simple and/or generalized T-Codes [1, 2, 3]. The main difference between T-augmentation and the algorithm presented here is that there is no "copy" operation and that the codeset appended is always just the alphabet. As such, the algorithm presented below is merely iterative but not recursive.

Let A be the alphabet that is to be used on the communication channel, consisting of N_0 symbols. Further let n be the number of source symbols that we wish to encode.

The initial code set S_0 is chosen to be the alphabet A, i.e., $S_0 = A$, where the subscript indicates the level of code construction, starting at 0. Correspondingly, S_i denotes the code set obtained after i iterative construction steps.

We further denote by w'_i a codeword from S_i that is of minimum length (i.e., for which there are no shorter codewords in S_i), and by w''_i a codeword from S_i of maximum length. We will see that for the code sets under consideration here, $|w''_i| = |w'_i| + 1$ or $|w''_i| = |w'_i|$ if the code set is a block code.

S_0	S_1	S_2	S_3	S_4
0	~~0~~			
1	1	~~1~~		
2	2	2	~~1~~	
	00	00	00	0~~0~~
	01	01	01	01
	02	02	02	02
		10	10	10
		11	11	11
		12	12	12
			20	20
			21	21
			22	22
				000
				001
				002

Figure 1: Iterative construction of a variable-length code with codewords of length 2 and 3 from a ternary alphabet

If we delete any word w_0 from S_0, add it as a prefix to a copy of each of the N_0 code symbols from S_0, then we get a prefix-free code set S_1 with $N_1 = (N_0 - 1) + N_0$ codewords. We may iteratively repeat this step using the following algorithm, starting from a set at level i:

1. choose a w'_i
2. $S_{i+1} = w'_i S_0 \cup S_i \backslash w'_i$, where $w'_i S_0 = \{x | x = w'_i z, z \in S_0\}$
3. If the number of codewords in S_{i+1} is smaller than n, increment i and goto step 1, otherwise continue.
4. if the number of codewords in S_{i+1} is larger than n, delete codewords of length $|w''_i|$ until the code set contains exactly n codewords.

The number of codewords in S_i, N_i may be recursively determined as

$$N_i = N_{i-1} + N_0 - 1, \tag{1}$$

or directly as

$$N_i = N_0(i+1) - i. \tag{2}$$

Thus, the algorithm is capable of generating arbitrarily large code sets. We note that the code sets thus constructed are complete, i.e., any sufficiently long stream of symbols from A has a decoding over any S_i.

Table 1 illustrates the construction of such a variable-length code based on a ternary channel alphabet $A = \{0, 1, 2\}$.

The above algorithm may be "shortened" as follows[1]:

1. generate the complete block code of all codewords of length $|w'_i| = \lfloor \log_{N_0} n \rfloor$.
2. apply the previous algorithm starting from this set.

In our example in Table 1, using the shortened algorithm is equivalent to first creating S_3 directly as a block code and then finishing the algorithm as previously discussed.

While the computational complexity of the first algorithm may be higher, the second is more expensive in terms of resources required.

[1] $\lfloor q \rfloor$ and $\lceil q \rceil$ denote the largest/smallest integer that is smaller/larger than or equal to q

3 Efficiency of Variable-Length Coding for Sources with Equal Symbol Probabilities

In practice, there are situations when the number n of source symbols is smaller than the number N_i of codewords in S_i, but larger than than the number N_{i-1} of codewords in S_{i-1}. So one or more of the longer codewords in S_i are not required in the encoding. It follows from Equation (1) that there are at most $N_0 - 2$ codewords in S_i that are surplus to requirements.

Taking this into account, we may now calculate the average codeword length for a source with n symbols with equal probabilities $P(n) = \frac{1}{n}$. The classical entropy of the source is given by $H(n) = \log_{N_0} n$ and may not be an integer. On the other hand, at some levels of the algorithm, the code set may well be a block code (e.g., the set S_3 in Table 1).

If we choose such a level where $|w_i''| = |w_i'| = \lfloor \log_{N_0} n \rfloor$, then we can create the required code set S_{i+c} at level $i + c$ by eliminating c words of length $\lfloor \log_{N_0} n \rfloor$ from S_i and adding b words with the length $\lceil \log_{N_0} n \rceil$.

Theorem 3.1 *The average codeword length $L_{N_0}(n)$ of a code for a source with n symbols of equal probability is given by*

$$L_{N_0}(n) = \lfloor \log_{N_0} n \rfloor + \frac{1}{n(N_0 - 1)} \left\{ N_0(n - N_0^{\lfloor \log_{N_0} n \rfloor}) + (N_0^{\lceil \log_{N_0} n \rceil + 1} - n) \mathrm{mod}(N_0 - 1) \right\}. \tag{3}$$

Proof: if we eliminate c codewords from nearest block code set of length $\lfloor \log_{N_0} n \rfloor$, the remaining number of words, a, with this length is given by

$$a = N_0^{\lfloor \log_{N_0} n \rfloor} - c. \tag{4}$$

Each of the previously deleted codewords of length $\lfloor \log_{N_0} n \rfloor$ gives rise to N_0 possible new codewords of length $\lfloor \log_{N_0} n \rfloor + 1$. Thus, each deletion and subsequent prefixing can add up to $N_0 - 1$ codewords to the set. In total, we require $n - N_0^{\lfloor \log_{N_0} n \rfloor}$ codewords more than in the block code. The number of deletions c that we require is thus given by

$$c = \left\lceil \frac{n - N_0^{\lfloor \log_{N_0} n \rfloor}}{N_0 - 1} \right\rceil. \tag{5}$$

As we require the final set to contain n words, we need to add

$$b = n - a \tag{6}$$

codewords. That may be less than the $N_0 c$ codewords we are now able to add. The number of "surplus" codewords, $N_0 c - b$, may be calculated by "counting backwards" from the next longer block code:

$$N_0 c - b = (N_0^{\lfloor \log_{N_0} n \rfloor + 1} - n) \mathrm{mod}(N_0 - 1). \tag{7}$$

Thus b is also given by

$$b = N_0 c - (N_0^{\lfloor \log_{N_0} n \rfloor + 1} - n) \mathrm{mod}(N_0 - 1). \tag{8}$$

Substituting Equation (4) for c, we obtain

$$b = N_0(N_0^{\lfloor \log_{N_0} n \rfloor} - a) - (N_0^{\lfloor \log_{N_0} n \rfloor + 1} - n) \mathrm{mod}(N_0 - 1) \tag{9}$$

and then, substituting Equation (6) for a:

$$b = N_0(N_0^{\lfloor \log_{N_0} n \rfloor} - N + b) - (N_0^{\lfloor \log_{N_0} n \rfloor + 1} - n) \mathrm{mod}(N_0 - 1). \tag{10}$$

Solving for b, we get:

$$b = \frac{N_0 n - N_0^{\lfloor \log_{N_0} n \rfloor + 1} + (N_0^{\lfloor \log_{N_0} n \rfloor + 1} - n) \bmod (N_0 - 1)}{N_0 - 1}, \tag{11}$$

i.e., the number of codewords of length $\lfloor \log_{N_0} n \rfloor + 1$. From Equation (6), this also gives us the number of codewords of length $\lfloor \log_{N_0} n \rfloor$:

$$a = \frac{-n + N_0^{\lfloor \log_{N_0} n \rfloor + 1} - (N_0^{\lfloor \log_{N_0} n \rfloor + 1} - n) \bmod (N_0 - 1)}{N_0 - 1}. \tag{12}$$

The average length of the codewords in our final set is given by:

$$L_{N_0}(n) = \frac{a \lfloor \log_{N_0} n \rfloor + b(\lfloor \log_{N_0} n \rfloor + 1)}{n}. \tag{13}$$

Substituting Equations (12) and (11) for a and b, we get

$$L_{N_0}(n) = \lfloor \log_{N_0} n \rfloor + \frac{1}{n(N_0 - 1)} \left\{ N_0(n - N_0^{\lfloor \log_{N_0} n \rfloor}) + (N_0^{\lceil \log_{N_0} n \rceil + 1} - n) \bmod (N_0 - 1) \right\}. \tag{14}$$

$\square$

In the binary case, where $N_0 = 2$, Equation (3) simplifies to Krichevski's result [5]:

$$L_2(n) = \lfloor \log_2 n \rfloor + \frac{2}{n}(n - 2^{\lfloor \log_2 n \rfloor}). \tag{15}$$

Note that the mod-term disappears as a binary code set constructed according to our rules is always complete.

Example 3.2 *Table 2 illustrates the construction of a code set with $n = 16$ codewords over a ternary alphabet ($A = \{0, 1, 2\}$, $N_0 = 3$).*

Its average codeword length is given by Equation (3):

$$L_3(16) = \lfloor \log_3 16 \rfloor + \frac{1}{16(3-1)} \left\{ 3(16 - 3^{\lfloor \log_3 16 \rfloor}) + (3^{\lceil \log_3 16 \rceil + 1} - 16) \bmod (3 - 1) \right\} \approx \quad 2.69. \tag{16}$$

We can verify this by simply counting the number of codewords of length 2 and 3 in $S_{7'}$ and calculating the average codeword length as follows:

$$\begin{aligned} L_3(16) &= \frac{\textit{no. of codewords of length 2} \times 2 + \textit{no. of codewords of length 3} \times 3}{16} \\ &= \frac{5 \times 2 + 11 \times 3}{16} \\ &\approx 2.69. \end{aligned} \tag{17}$$

4 Coding the Code

One problem often encountered in the use of variable-length codes is that they may vary depending on the source that is used.

In our case, such a variability could occur if the coding system has to support a number of equiprobable sources with different cardinality. If a lexicographical ordering of codeword assignments can be agreed on, and we agree to choose the w_i' in step 1 of our iterative algorithm in a quasi-lexicograhical manner, all we need to communicate to the receiver is the cardinality of the source. The coding/decoding tree and codeword assignments may easily be deduced from this. Alternatively, we may communicate the lexicograpically last codeword used as it conveys the same information.

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	$S_{7'}$
0	$\cancel{0}$							
1	1	$\cancel{1}$						
2	2	2	$\cancel{2}$					
	00	00	00	$0\cancel{0}$				
	01	01	01	01	$0\cancel{1}$			
	02	02	02	02	02	$0\cancel{2}$		
		10	10	10	10	10	$1\cancel{0}$	
		11	11	11	11	11	11	11
		12	12	12	12	12	12	12
			20	20	20	20	20	20
			21	21	21	21	21	21
			22	22	22	22	22	22
				000	000	000	000	000
				001	001	001	001	001
				002	002	002	002	002
					010	010	010	010
					011	011	011	011
					012	012	012	012
						020	020	020
						021	021	021
						022	022	022
							100	100
							101	101
							102	$10\cancel{2}$

Figure 2: Construction of a code set with 16 codewords of from a ternary alphabet. Note that the intermediate set S_3 is the next smaller block code and the one codeword has to be removed from S_7 to yield the final set $S_{7'}$

If the cardinality (or the length of the last codeword) are á priori unknown to both transmitter and receiver, we can't just transmit the last codeword as the receiver will not know how long it is going to be, and whether (in a continuous data stream) it is receiving the codeword or subsequent data. Hence, we require an efficient scheme that permits the transmission of this information.

The scheme proposed here is a "mixed scheme". It first transmits the encoded length of the shorter of the two codeword lengths. This determines the length of the longer codewords (except in the case of block codes). Let us first consider codes where $|w_i''| = |w_i'| + 1$.

For such codes, we might thus just send the last codeword "as is". This leaves the problem of unanimously transmitting the length of the shorter codewords. For a binary channel alphabet, this could be done by transmitting $1^{|w_i'|-1}0$, i.e., $|w_i'| - 1$ ones followed by a zero, where $|w_i'|$ is the length of the shorter codewords.

Unfortunately, this is inefficient for non-binary channel alphabets. For these, we may opt for the following method: without loss of generality, presume that the symbols of the channel alphabet are denoted by the numbers $0 \ldots N_0 - 1$. Then we may communicate $|w_i'|$ as follows:

1. start the length count L at zero: $L = 0$.
2. if $|w_i'| - L \geq N_0$, transmit the symbol N_0 and increment L by N_0 and repeat this step. Else transmit $|w_i'| - L$.

The receiver now adds all numbers received until the first number smaller than N_0 is received. This yields $|w_i'|$. We can then send the last codeword "as is".

This approach is not suitable for block codes, though: if our code is a block code, we have $|w_i''| = |w_i'|$ and thus the last codeword would be one symbol shorter than we expect. However, this is merely a result of transmitting the last codeword *used.* If we instead transmit the last codeword from a set whose cardinality is increased by one, we circumvent the problem — its length always equals $|w_i'| + 1$. Thus, for block codes, we send the codeword $0^{|w_i'|}N_0$. In this, we exploit the fact that our code construction algorithm can never yield a code set that has only one code word of length $|w_i'| + 1$.

Example 4.1 *Consider the ternary code $S_{7'}$ in the last column of Table 2. For this code, $|w_i'| = 2$ and $|w_i''| = 3$. Lexicographically, the last codeword used is* 102. *If we were to extend our code further, the "next" codeword would be* 110. *We first encode $|w_i'|$ as* 20. *This is followed by* 110 *such that the transmission starts as* 20110...

5 Conclusions

This report explained the construction and efficiency evaluation of variable-length codes with two distinct codeword lengths over arbitrary alphabets. These codes may be used for efficient encoding of sources with equiprobable symbols. Possible application areas are conceivable in discrete telecommuncation systems where a particular alphabet cardinality is favoured by the communication technology used. The report shows that the construction of the codes, the evalutation of their efficiency, and their communication to a receiver are easy.

References

[1] M. R. Titchener: *Generalized T-Codes: an Extended Construction Algorithm for Self-Synchronizing Variable-Length Codes*, IEE Proceedings – Computers and Digital Techniques, 143(3), June 1996, pp. 122-128.

[2] U. Guenther, P. Hertling, R. Nicolescu, and M. R. Titchener: *Representing Variable-Length Codes in Fixed-Length T-Depletion Format in Encoders and Decoders*, Journal of Universal Computer Science, 3(11), November 1997, pp. 1207–1225.

[3] U. Guenther: *Data Compression and Serial Communication with Generalized T-Codes*, Journal of Universal Computer Science, V. 2, N 11, 1996, pp. 769-795.

[4] D. Huffman: *A Method for the Construction of Minimum Redundancy Codes*, Proc. Inst. Radio Eng. 40, September 1952, pp. 1098–1101.

[5] R. Krichevski: *Compression and search of information*. Radio and Communication, Moscow, 1989, p. 168.

[6] V. Maevski, F. Blocski, A. Novak et al.: *Digital systems of transmission*, Communication, Moscow, 1978, pp. 77–78.