Fast Adaptive Arithmetic Code for Large Alphabet Sources with Asymmetrical Distributions

Boris Ryabko [1]
Siberian State University of Telecommunication and Computer Science; Novosibirsk, 630102, Russia
e-mail: ryabko@neic.nsk.su

Jorma Rissanen
Tampere International Center for Signal Processing at the Technical University of Tampere, Finland
e-mail: Jrrissanen@aol.com

Abstract — **We address the problem of constructing an adaptive arithmetic code in the case where the source alphabet is large and there are lots of different symbols with equal counts of occurrence. For an alphabet of N symbols and r distinct symbol weights we describe a code for which the number of operations of encoding and decoding is equal to $c\log r + c_1$ instead of $c\log N + c_2$ as in previous arithmetic codes, c, c_1, c_2 are constants. When r is small relative to N - which is the case for most practical coding problems on large alphabets - the encoding and decoding speed of the suggested code will be substantially greater than with known methods.**

Arithmetic coding was introduced by Rissanen [6] and now it is one of the most popular methods of source coding. The advantage of arithmetic coding over other coding techniques is that it achieves arbitrarily small coding redundancy per source symbol at less computational effort than any other method (the redundancy is defined as the difference between the mean codeword length and the source entropy). Furthermore, arithmetic coding may easily be applied to sources with unknown statistics, when being combined with adaptive models. Often in such cases a source alphabet is quite large (or even infinite) but the main part of its letters is seldom used if ever, whereas a few letters are frequent. This fact is well known and has been used in Information Theory [5]. In this report we take advantage of the same properties in the design of a fast arithmetic code.

Let us consider a memoryless source generating letters from the alphabet $A = \{a_1, ..., a_N\}$ with unknown probabilities. Let the source generate a message $x_1x_2\ldots$ and let $\nu^t(a)$ denote the count of letter a occurring in the word $x_1\ldots x_t$. After the first t letters $x_1\ldots x_t$ have been processed the following letter x_{t+1} needs to be encoded. In the most popular version of the arithmetic code the current estimated probability distribution is taken as $p^t(\nu^t(a)) = (\nu^t(a)+c)/(t+Nc)$, where c is a constant (as a rule c is 1 or 1/2). Let $x_{t+1} = a_i$, and let the interval $[\alpha, \beta)$ represent the word $x_1\ldots x_t$. Then the word $x_1\ldots x_t x_{t+1}$, $x_{t+1} = a_i$, will be encoded by the interval $[\alpha+(\beta-\alpha)q_i^t, \alpha+(\beta-\alpha)q_{i+1}^t)$, where $q_i^t = \sum_{j=1}^{i-1} p^t(a_j)$. When the size of the alphabet N is large the calculation of q_i^t is the most time consuming part in the encoding process. The time of a "naive" method of updating the cumulative probabilities is proportional to N. Jones [2] and Ryabko [7] have independently suggested two different algorithms which perform all necessary transitions between individual and cumulative probabilities in $O(\log N)$ operations. Later many such algorithms have been suggested and developed, see [3, 8, 1, 4].

Let $A_k^t, k = 0, 1, ...,$ denote the set of letters whose count of occurrence in the word $x_1\ldots x_t$ equals k. Informally, the codeword of the letter $x_{t+1} = a_i$ consists of two parts. The first part encodes the set A_k^t that contains a_i, and the second part encodes the number of the element a_i in the set A_k^t. It turns out that it is easy to encode and decode letters in the sets A_k^t, and we should use time consuming operations for encoding the sets A_k^t, only. Let $k_1, k_2, ..., k_{r_t}$ be integers such that $A_{k_i}^t$ are nonempty, and let $k_1 < k_2 < ... < k_{r_t}$. Define the values: $P_{k_i}^t = |A_{k_i}^t|\, p^t(k_i)$, $\quad Q_i^t = \sum_{j=1}^{i-1} P_{k_j}^t$. The coding interval for the word $x_1\ldots x_t x_{t+1}$ is calculated as follows

$$[\alpha+(\beta-\alpha)(Q_j^t+(\delta(a_i)-1)p^t(k_j)), \alpha+(\beta-\alpha)(Q_j^t+\delta(a_i)p^t(k_j))),$$

where $\delta(a_i)$ is the ordinal of a_i in the subset $A_{k_j}^t$. We describe shortly decoding with the new method. Suppose that the letter x_{t+1} is to be decoded. There are two steps required: first, the algorithm finds the set $A_{k_j}^t$ with the usual arithmetic code that contains the (unknown) letter a_i. The ordinal of the letter a_i is calculated as follows: $\delta() = \lfloor (code(x_{t+1}...) - Q_j^t)/p^t(j) \rfloor$, where $code(x_{t+1}...)$ is the number that encodes the word $x_{t+1}x_{t+2}...$. The main properties of the suggested code can be formulated as follows.

Theorem.

Let the described code be applied to encode and decode symbols $x_1x_2\ldots$ generated by an N letter source. Then the number of operations for the encoding and decoding of the letter x_t is given by $T = c_1\log r_t + c_2$, where c_1, c_2 are constants and r_t is the number of different counts of occurrence of letters from A in the word $x_1\ldots x_{t-1}$.

References

[1] P. Fenwick, "A new data structure for cumulative probability tables," *Software – Practice and Experience,* vol. 24, no. 3, pp. 327–336, 1994.

[2] D. W. Jones, "Application of splay trees to data compression", *Communications of the ACM*, v 31, n. 8, pp.996-1007, 1988.

[3] A. Moffat, "Linear time adaptive arithmetic coding",*IEEE Trans. on Information Theory* v.36, no. 2, pp.401-406, 1990.

[4] A. Moffat, "An improved data structure for cumulative probability tables",*Software – Practice and Experience*,v.29, no. 7, pp.647-659,1999.

[5] A.Moffat,A.Turpin, " Efficient Construction of Minimum-Redundancy Codes for Large Alphabets".*IEEE Trans. Inform. Theory,* vol. IT-44, no. 4, pp. 1650–1657, 1998.

[6] J.Rissanen, "Generalized Kraft inequality and arithmetic coding", *IBM J. Res. Dev.,* vol. 20, pp. 198–203, 1976.

[7] B. Ya.Ryabko,"A fast sequential code." *Dokl. Akad. Nauk SSSR* v.306 (1989), no. 3, pp.548–552 (Russian); translation in *Soviet Math. Dokl.*, v. 39 (1989), no. 3, pp. 533–537.

[8] B.Ryabko, "A fast on-line adaptive code". *IEEE Trans. Inform. Theory,* vol. IT-38,no. 4, pp. 1400 - 1404, 1992.

[1] Supported by the Russian Foundation of Basis Research under the Grant no. 99-01-00586, INTAS under the Grant 00-738 and Tampere International Center for Signal Processing, Finland.