A Tutorial of The Context Tree Weighting Method: Basic Properties

Zijun Wu

November 9, 2005

Abstract

In this tutorial, we try to give a tutorial overview of The Context Tree Weighting Method. We confine our discussion to binary bounded memory tree sources and describe a sequential universal data compression procedure, which achieves a desirable coding distribution for tree sources with unknown model and unknown parameters. Computational and storage complexity of the proposed procedure are both linear in the source sequence length.

1. Introduction

In our class, we learned Huffman codes. For small source alphabets, though, we have efficient coding only if we use long blocks of source symbols. It is therefore desirable to have an efficient coding procedure that works for long blocks of source symbols. Huffman coding is not ideal for this situation. Arithmetic coding achieves this goal. When no priori probability distribution knowledge of the source is available, one uses universal coding algorithm. For finite memory source model, Weinberger, Ziv and Lempel developed a sequential algorithm for universal coding, where artificial parameter K is involved. To avoid choosing parameter, Willems, Shtarkov and Tjalkens proposed a weighting method. This method estimates the source probability given past symbols and then combines it with arithmetic algorithm. We will outline the basics of the method in this tutorial.

2. Binary Bounded Memory Tree Sources

A binary tree source generates a sequence $x_{-\infty}^{\infty}$ of digits assuming values in the alphabet $\{0,1\}$. We denote by x_m^n the sequence $x_m x_{m+1} \cdots x_n$ and allow m and n to be infinitely large. For n<m the sequence is empty, denoted by ϕ.

The statistical behavior of a binary finite memory tree source can be described by means of a suffix set S. This suffix set is a collection of binary strings s(k), with k=1,2,…,|S|. We require it to be proper and complete. Properness of the suffix set implies that no string in S is a suffix of any other string in S. Completeness guarantees that each semi-infinite sequence (string) $\cdots x_{n-2}x_{n-1}x_n$ has a suffix that belongs to S. This suffix is unique since S is proper.

A bounded memory tree source has a suffix set S that satisfies $l(s) \le D$ for all $s \in S$. We say that the source has memory not larger than D.

We define a suffix function $\beta_S(\cdot)$, which maps semi-infinte sequences onto their unique suffix s in S.

Definition 1: The actual next-symbol probabilities for a bounded memory tree source with suffix set S and parameter vector Θ_S are

$$P_a(X_1 = 1 \mid x_{t-D}^{t-1}, S, \Theta_S) = 1 - P_a(X_t = 0 \mid x_{t-D}^{t-1}, S, \Theta_s) = \theta_{\beta_S}(x_{t-D}^{t-1}) \quad \text{for all t.}$$

The actual block probabilities are now products of actual next-symbol probabilities, i.e.

$$P_a(X_1^t = x_1^t \mid x_{1-D}^0, S, \Theta_S) = \prod_{\tau=1}^{t} P_a(X_\tau = x_\tau \mid x_{\tau-D}^{\tau-1}, S, \Theta_S)$$

All sources with the same suffix set are said to have the same model. The set of all tree models having memory not larger than D is called the model class C_D. It is possible to specify a model in this model class by a natural code by encoding the suffix set S recursively. The code of S is the code of the empty string λ. The code of a string s is void if l(s) = D; otherwise, it is 0 if $s \in S$ and 1 followed by the codes of the strings 0s and 1s if $s \notin S$. If we use this natural code, the number of bits that are needed to specify a model $S \in C_D$ is equal to $\Gamma_D(S)$, where

Definition 2: $\Gamma_D(S)$, the cost of a model S with respect to model class C_D is defined as

$$\Gamma_D(S) = |S| - 1 + |\{s : s \in S,\ l(s) \neq D\}|$$

where it is assumed that $S \in C_D$.

So far, we've proposed a model for the source. And we already know arithmetic coding is a good sequential method in the sense that its individual coding redundancy is no more than 2 bits. Now our task is to find a distribution estimate which bridges between the model and arithmetic coding method. In the following, we define the coding redundancy, describe the redundancy upper bound of arithmetic coding, and then a weighting method as a distribution estimate.

3. Codes and Redundancy

We assume that both the encoder and the decoder have access to the past source symbols $x_{1-D}^0 = x_{1-D} \cdots x_{-1} x_0$, so that implicitly the suffix that determines the probability distribution of the first source symbols, is available to them. We denote the functional relationship between source sequence and codeword to be $c^L(x_1^T \mid x_{1-D}^0)$. The length of the codeword, in binary digits, is denoted as $L(x_1^T \mid x_{1-D}^0)$. We restrict ourselves to prefix codes here.

The codeword lengths $L(x_1^T \mid x_{1-D}^0)$ determine the individual redundancies.

Definition 3: The individual redundancy $\rho(x_1^T \mid x_{1-D}^0, S, \Theta_S)$ of a sequence x_1^T given the past symbols x_{1-D}^0, with respect to a source with model $S \in C_D$ and parameter vector Θ_S, is defined as

$$\rho(x_1^T \mid x_{1-D}^0, S, \Theta_S) = L(x_1^T \mid x_{1-D}^0) - \log \frac{1}{P_a(x_1^T \mid x_{1-D}^0, S, \Theta_S)}$$

We only consider sequences with positive probability $P_a(x_1^T \mid x_{1-D}^0, S, \Theta_S)$.

4. Arithmetic Coding

Suppose that the encoder and decoder both have access to, what is called the coding distribution

$$P_C(x_1^t),\ x_1^t \in \{0,1\}^t,\ t = 0,1,\cdots,T$$

We require that this distribution satisfies

$$P_C(\phi) = 1,$$

$$P_C(x_1^{t-1}) = P_C(x_1^{t-1}, X_t = 0) + P_C(x_1^{t-1}, X_t = 1), \qquad \text{for all } x_1^t \in \{0,1\}^t,\ t = 0,1,\cdots,T$$

and $P_C(x_1^T) > 0$, for all possible $x_1^T \in \{0,1\}^T$ (1)

Thereom 1: Given a coding distribution

$$P_C(x_1^t),\ x_1^t \in \{0,1\}^t,\ t = 0,1,\cdots,T$$

The Elias algorithm achieves codeword lengths $L(x_1^T)$ that satisfy

$$L(x_1^T) < \log \frac{1}{P_C(x_1^T)} + 2$$

for all possible $x_1^T \in \{0,1\}^T$. The codeword form a prefix code.

We say that the coding individual coding redundancy is always less than 2 bits. The Elias algorithm combines an acceptable coding redundancy with a desirable sequential implementation. The number of operations is linear in the source sequence length T. If we are ready to accept a loss of at most 2 bits coding redundancy, we are now left with the problem of finding good, sequentially available, coding distributions.

5. Probability Estimation

The probability that a memoryless source with parameter θ generates a sequence with a zeros and b ones is $(1-\theta)^a \theta^b$. If we weight this probability over all θ with a $(\frac{1}{2}, \frac{1}{2})$-Dirichlet distribution we obtain the so-called Krichevsky-Trofimov estimate (see [5]).

Definition 4: The Krichevsky-Trofimov estimated probability for a sequence containing $a \geq 0$ zeros and $b \geq 0$ ones is defineds as

$$P_e(a,b) = \int_0^1 \frac{1}{\pi\sqrt{(1-\theta)\theta}} (1-\theta)^a \theta^b d\theta$$

This estimator has properties that are listed in the lemma that follows.

Lemma 1: The K-T probability estimator $P_e(a,b)$

1) can be computed sequentially, i.e., $P_e(0,0) = 1$, and for $a \geq 0$ and $b \geq 0$

$$P_e(a+1,b) = \frac{a+\frac{1}{2}}{a+b+1} \cdot P_e(a,b) \quad \text{and}$$

$$P_e(a,b+1) = \frac{b+\frac{1}{2}}{a+b+1} \cdot P_e(a,b) \qquad (2)$$

2) satisfies, for $a+b \geq 1$, the following inequality:

$$P_e(a,b) = \frac{1}{2} \cdot \frac{1}{\sqrt{a+b}} (\frac{a}{a+b})^a (\frac{b}{a+b})^b$$

6. Coding for an Unknown Tree Source

A. Definition of the Context-Tree Weighting Method

Consider the case where we have to compress a sequence which is (supposed to be) generated by a tree source, whose suffix set S ∈ C$_D$ and parameter vector Θ_S are unknown to the encoder and the decoder. We will define a weighted coding distribution for this situation.

Definition 5: The context tree T_D is a set of nodes labeled s, where s is a (binary) string with length $l(s)$ such that $0 \leq l(s) \leq D$. Each node s∈ T_D with $l(s) < D$, "splits up" into two nodes, 0s and 1s. The counts must satisfy $a_{0s} + a_{1s} = a_s$ and $b_{0s} + b_{1s} = b_s$

Definition 6: To each node s∈ T_D, we assign a weighted probability P_w^s which is defined as: $$P_w^s \triangleq \begin{cases} \frac{1}{2} P_e(a_s,b_s) + \frac{1}{2} P_w^{0s} P_w^{1s} & for\ 0 \leq l(s) \leq D \\ P_e(a_s,b_s) & for \quad l(s) = D \end{cases} \qquad (3)$$

The context tree together with the weighted probabilities of the nodes is called a weighted context tree.

We define our weighted coding distribution as

$P_C(x_1^t \mid x_{1-D}^0) \triangleq P_w^\lambda(x_1^t \mid x_{1-D}^0)$ for all $x_1^t \in \{0,1\}^t$, t = 0, 1,$\cdots$, T, where λ is the root node of the context tree T_D. (4)

We can check that the weighted coding distribution satisfies (1).

B. An Upper Bound on the Redundancy

Definition 7: Let

$$\gamma(z) \triangleq \begin{cases} z & for\ 0 \le z < 1 \\ \frac{1}{2}\log z + 1 & for\ \ z \ge D \end{cases}$$

The basic result concerning the context-tree weighting technique can be stated now.

Theorem 2: The individual redundancies with respect to any source with model $S \in C_D$ and parameter vector Θ_S are upper-bounded by

$\rho(x_1^T \mid x_{1-D}^0, S, \Theta_S) < \Gamma_D(S) + |S|\gamma(\frac{T}{S}) + 2$ for all $x_1^T \in \{0,1\}^T$, for any sequence of past symbols x_{1-D}^0. The three terms on the right hand side of the inequality represents upper bound of model redundancy, parameter redundancy and coding redundancy respectively.

Corollary: Using the coding distribution in (4), the codeword lengths $L(x_1^T \mid x_{1-D}^0)$ are upper bounded by

$$L(x_1^T \mid x_{1-D}^0) < \min_{S \in C_D}(\min_{\Theta_S} \log \frac{1}{P_a(x_1^T \mid x_{1-D}^0, S, \Theta_S)} + \Gamma_D(S) + |S|\gamma(\frac{T}{S})) + 2$$

C. Implementation of the Context-Tree Weighting Method

1) Encoding and Decoding: We assume that a node $s \in T_D$ contains the pair (a_s, b_s), the estimated probability $P_e(a_s, b_s)$ and the weighted probability P_w^s. When a node is created, the counts a_s and b_s are made 0, the probabilities $P_e(a_s, b_s)$ and P_w^s are made 1. We then use the update scheme as indicated by (2) & (3) to get the coding distribution. Then the encoding and decoding procedure follows through with the Elias algorithm.

2) Complexity Issues: For each symbol x_t we have to visit D+1 nodes. Some of these nodes have to be created first. From this it follows that the total number of allocated nodes cannot be more than $T(D+1)$. This makes the storage complexity not more than linear in T. Note also that the number of nodes cannot be more than $2^{D+1} - 1$, the total number of nodes in T_D.

The computational complexity, i.e. the number of additions, multiplications, and divisions, is proportional to the number of nodes that are visited, which is $T(D+1)$. Therefore, this complexity is also linear in T.

REFERENCES

[1] T. M. Cover and J. A. Thomas, Elements of Information Theory, New York: Wiley, 1991.

[2] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” IEEE Trans.
Inform. Theory, vol. IT-23, NO.3, May 1977.

[3] M. J. Weinberger, A. Lempel, and J. Ziv, “A sequential algorithm for the universal coding of
finite memory sources,” IEEE Trans. Inform. Theory, vol. 38, pp. 1002-1014, May 1992.

[4] G. G. Langdon, An introduction to arithmetic coding. IBM Journal of Research and
Development, 28: 135-149, 1984

[5] J. Rissanen and G. G. Langdon, “Universal modeling and coding,” IEEE Trans. Inform.Theory,
vol. IT-27, pp. 12-23, Jan. 1981.