EE477 Universal Schemes in Information Theory Lecture 4 - 10/06/2011

Context Tree Weighting (CTW)

Lecturer: Haim Permuter *Scribe: Idoia, Alexandros and Albert*

1 Motivation

Assume we have a sequence $x^N = (x_1, x_2, \ldots, x_N)$ drawn from $P_a(x^N)$ that we want to compress. If the probability distribution of the source is known, we can use a compression algorithm that makes use of this probability to perform the compression, like Arithmetic or Huffman coding. In the case where the actual distribution $P_a(x^N)$ is unknown, we can still compress the sequence by using a universal compression algorithm like Lempel Ziv or the CTW that do not need the probability distribution of the source.

On the other hand, if we are interested in estimating the probability distribution of x^N, since we know that a good compressor implies a good estimate of $P_a(x^N)$, we could use the compression result of LZ to implicitly compute the probability distribution. The CTW algorithm does it better, because it actually estimates the probabilities before the compression, so that we use them to our purposes.

Assume that we have a sequence x^N and we need a good estimate of the probability $P_L(x^N)$. To address this problem, we are going to rely on the fact that a good compressor of the sequence implies a good estimate of $P_a(x^N)$. The latter will be shown in the following sections. Depending on the assumptions that we make on the statistical properties of the source we use different approaches. For example, for $i.i.d$ sources, we are going to present the Arithmetic Coding, and for more general models the Context Tree Weighting.

2 Good universal compressor implies good universal estimator

Let x^N be the sequence whose probability distribution $P_a(x^N)$ we need to estimate. As stated above, we are going to use ideas from a good compressor in order to achieve this. The base of the logarithm in this discussion is always 2.

Assume that we have a good compressor of the sequence. Let $L(x^N)$ be the length of the codeword $c(x^N)$. Let $P_L(x^N)$ be the probability distribution that the compressor uses in order to compress the sequence x^N. The latter has the meaning of the estimate of the probability distribution $P_a(x^n)$. We know that, in order to achieve minimum expected code length the code uses $L(x^N) = \log \frac{1}{P_L(x^N)} \Rightarrow P_L(x^N) = 2^{-L(x^N)}$. Yet, we also need $P_L(x^N)$ to sum to 1 since it is a probability distribution. Therefore, let $k_N = \sum_{x_N} 2^{-L(x^N)}$ to be the normalization factor. We know that $k_N \leq 1$ from the Kraft Inequality. We get that:

$$P_L(x^N) = \frac{2^{-L(x^N)}}{k_N} \Rightarrow$$
$$L(x^N) = -\log(k_N) + \log \frac{1}{P_L(x^N)}$$

Now, lets define the individual redundancy $\rho(x^N)$ of a source code, relative to the actual source:

$$\rho(x^N) = L(x^N) - \log \frac{1}{P_a(x^N)}$$

Similarly we can define the average redundancy $\bar{\rho}(x^N)$ as the average value of $\rho(x^N)$ over all possible sequences x^N.

$$\begin{aligned}
\bar{\rho}_n &= \mathbb{E}[\rho(X^N)] \\
&= \sum_{x^N} P_a(x^N) \left[L(x^N) - \log \frac{1}{P_a(x^N)} \right] \\
&= \sum_{x^N} P_a(x^N) \left[-\log k_N + \log \frac{P_a(x^N)}{P_L(x^N)} \right] \\
&= -\log k_N + \sum_{x^N} P_a(x^N) \log \frac{P_a(x^N)}{P_L(x^N)} \\
&= -\log k_N + D(P_a || P_L)
\end{aligned}$$

Clearly, we get the following properties,

- $\bar{\rho}_N \geq 0$, since $k_N \leq 1 \Rightarrow -\log(k_N) \geq 0$ and the K-L divergence $D(\cdot||\cdot)$ is always non-negative.
- If $\bar{\rho}_N \to 0$ then $k_N \to 1$ and $D(P_a(x^N)||P_L(x^N)) \to 0$.
  A good compressor is going to have a small $\bar{\rho}_n$. Therefore, if we have a good compressor, we also have a good estimate of the real distribution $P_a(x^N)$. This means that we can only achieve $\bar{\rho}_N = 0$ if the source code uses the actual distribution to code the sequence.

3 Arithmetic Coding

Assume a lexicographical ordering over the sequences. Let $Q(x^N) = \sum_{\tilde{x}^N < x^N} P_a(\tilde{x}^N)$ be the cumulative probability of x^N. We associate to a source sequence x^N a source interval $I(x^N) = [Q(x^N), Q(x^N) + P_a(x^N)]$. Note that these intervals are disjoint and that their union is $[0, 1)$. Also, assume that we have a codeword $c = (c_1, c_2, \ldots, c_N)$. For every binary codeword define $.c = 0.c_1c_2\cdots c_N = c_1 2^{-1} + c_2 2^{-2} + \cdots + c_N 2^{-N}$. Each codeword has associated an interval $J(c) = [.c, .c + 2^{-L}]$.

The idea of arithmetic coding is to choose for a given source sequence x^N the codeword $c(x^N)$ with a code interval $J(c(x^N))$ inside $I(x^N)$. In order to do that we can take:

$$\begin{aligned}
L(x^N) &= \lceil \log(\frac{1}{P_a(x^N)}) \rceil + 1 \\
.c &= \lceil Q(x^N) \cdot 2^{L(x^N)} \rceil \cdot 2^{-L(x^N)}
\end{aligned}$$

Then, it follows that:

$$\begin{aligned}
.c + 2^{-L(x^N)} &= (\lceil Q(x^N) 2^{L(x^N)} \rceil + 1) 2^{-L(x^N)} \\
&\leq Q(x^N) + 2^{-L(x^N)+1} \\
&= Q(x^N) + 2^{-(\lceil \log \frac{1}{P_a(x^N)} \rceil + 1) + 1} \\
&\leq Q(x^N) + P_a(x^N)
\end{aligned}$$

Therefore, $Q(x^N) \leq .c \leq .c + 2^{-L(x^N)} \leq Q(x^N) + P_a(x^N)$, and $J(c) = [.c, .c + 2^{-L(x^N)}] \subset [Q(x^N), Q(x^N) + P_a(x^N)]$, which is the desired result.

Assuming that the actual probability distribution $P_a(x^N)$ is known, the redundancy using Arithmetic Coding is given by $\rho(x^N) = L(x^N) - \log \frac{1}{P_a(x^N)} = \lceil \log(\frac{1}{P_a(x^N)}) \rceil + 1 - \log \frac{1}{P_a(x^N)} < 2$, and it follows that $\bar{\rho_n} < 2$.

We observe that Arithmetic Coding achieves codeword lengths that are very close to the optimal codeword lengths $\log \frac{1}{P_a(x^n)}$.

4 One unknown Parameter

In the previous section we have seen that if we know the true distribution of the source we can achieve $\bar{\rho} < 2$ using Arithmetic Coding. Now suppose that we have an *i.i.d.* source $X^N \sim Bern(\theta)$, where the parameter θ is unknown. Notice that if we knew θ then we would be in the previous case. The question now is whether it is still possible to design a source code which has acceptable individual redundancies for all sequences x^N. The answer turns out to be affirmative: We can apply Arithmetic Coding with a coding distribution equal to an estimated distribution $P_e(x^N)$ formulated by the Krichevsky and Trofimov (KT) Estimation [1].

First we are going to present the Krichevsky-Trofimov Estimation and then we will show that using the KT Estimation together with Arithmetic Coding we can actually achieve $\rho(x^N) < (1+\frac{1}{2}\log N)+2 = \frac{1}{2}\log N+3$. The term $(1+\frac{1}{2}\log N)$ is the price we pay for not knowing the parameter θ (it is called *parameter redundancy*) and the other term is the redundancy for using Arithmetic Coding over the estimated distribution $P_e(x^N)$.

4.1 Krichevsky-Trofimov (KT) Estimation

Suppose that we have an *i.i.d.* source $X^N \sim Bern(\theta)$, but we do not know θ. We would like to estimate θ based on the sequence x^N. One way to estimate θ is by using the KT Estimation. Let a and b denote the number of 0's and 1's in x^N, respectively, so that $a + b = N$. Assume that we use as a prior probability for θ the Dirichlet $(1/2, 1/2)$. That is:

$$f(\theta) = \frac{1}{\pi\sqrt{\theta(1-\theta)}},\ \theta \in (0,1).$$

The use of the above distribution as a prior knowledge for the parameter θ is not unique. Yet, it leads to iterative formulas for calculating the estimated posterior distribution and to nice properties. Using the above assumptions we get:

$$\begin{aligned} P_e(x^n) &= P_e(a,b) \\ &= \int P_e(a,b|\theta) f(\theta) d\theta \\ &= \int (1-\theta)^a \theta^b f(\theta) d\theta \\ &= \int \frac{1}{\pi\sqrt{\theta(1-\theta)}} (1-\theta)^a \theta^b d\theta \end{aligned}$$

The above estimator of the posterior probability has the following properties. The KT Estimator $P_e(\alpha, \beta)$

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

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

  Note that

$$P_e(a,b) = P_e(a+1,b) + P_e(a,b+1)$$

- for $a+b \geq 1$ It satisfies the following inequality:

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

Note that we could use any other prior knowledge for the parameter θ or another estimator, like the Maximum Likelihood (ML) that leads to the following formulas:

$$\operatorname*{argmax}_\theta P_e(a,b|\theta) = \frac{b}{a+b} \tag{2}$$
$$\max_\theta P_e(a,b|\theta) = (\frac{a}{a+b})^a (\frac{b}{a+b})^b \tag{3}$$

Yet the above estimator does not offer as a simple recursive formula as it happens with the KT Estimator.

4.2 Arithmetic Coding and KT Estimator

Now that we have explained the KT Estimator we are going to use Arithmetic Coding on the estimated distribution $P_e(x^N)$ and hope that we do not pay much for not knowing the actual parameter θ. Recall that $\rho(x^N) \leq 2$ for Arithmetic Coding with known probability distribution. Thus, with respect to the estimated distribution we pay a maximum of 2 bits in the individual redundancy:

$$L(x^N) < \log \frac{1}{P_e(\alpha,\beta)} + 2 \tag{4}$$

Surprisingly, under the model of one unknown parameter, we can also find an upper bound for the individual redundancy with respect to the actual distribution:

$$
\begin{aligned}
\rho(x^N) &= L(x^N) - \log \frac{1}{P_a(x^N)} \\
&\overset{(a)}{<} \log \frac{1}{P_e(a,b)} - \log \frac{1}{P_a(x^N)} + 2 \\
&= -\log P_e(a,b) + \log(1-\theta)^a \theta^b + 2 \\
&\overset{(b)}{\leq} \log \frac{(1-\theta)^a \theta^b}{\frac{1}{2\sqrt{a+b}}(\frac{a}{a+b})^a (\frac{b}{a+b})^b} + 2 \\
&\overset{(c)}{\leq} \log \frac{(\frac{a}{a+b})^a (\frac{b}{a+b})^b}{\frac{1}{2\sqrt{a+b}}(\frac{a}{a+b})^a (\frac{b}{a+b})^b} + 2 \\
&= \frac{1}{2}\log(a+b) + 1 + 2 \\
&= \frac{1}{2}\log N + 3,
\end{aligned}
$$

where

- (a) follows from (4).
- (b) follows from the lower bound of $P_e(\alpha, \beta)$ of (1).
- (c) follows from (2).

Hence, even if we have one unknown parameter the individual redundancy is never larger than $\frac{1}{2}\log(N) + 3$ for all sequences x^N and all $\theta \in [0, 1]$.

5 Tree Source

If a source is memoryless, each new source symbol is generated according to the same parameter θ. In a more complex situation we can assume that the parameter for generating the next symbol depends on the most recent source symbols. A Tree Source is a nice concept to describe such sources. It consists of a set $\mathcal{S}$ of suffixes that together form a tree. To each suffix (leaf) $s \in \mathcal{S}$ of the tree there corresponds a parameter θ_s. Therefore for a Tree Source we have $|\mathcal{S}|$ parameters that denote all the possible dependencies between the next symbol and the past. The probability of the next symbol being one depends on the suffix in $\mathcal{S}$ of the semi-infinite sequence of past source symbols.

Definition 1. *A context of a source symbol x_t is a suffix of the semi-infinite sequence $\ldots x_{t-2}x_{t-1}$ that precedes it.*

Define S as the set of prefixes and $\Theta_S = \{\theta_s = P(x_t = 1 | x_{t-D}^{t-1} = s) : s \in S\}$. Then we can say that the probability of x^N is as follows,

$$
\begin{aligned}
P_a(x^N) &= \prod_{s \in S} P_a(a_s, b_s) \\
&= \prod_{s \in S} \theta_s^b (1-\theta_s)^a
\end{aligned}
$$

where a_s is the number of 0's right after the string s. For example, for $x^N = 01\mathbf{10}11\mathbf{10}0\mathbf{10}1$ and $s = 10$ we have that $a_s = 1$ and $b_s = 2$.

Example 1 Consider the following Tree Source where $\mathcal{S} = \{1, 10, 00\}$, and let $P(x_t = 1|s = x_{t-2}x_{t-1} = 00) = 0.5$, $P(x_t = 1|s = 10) = 0.3$ and $P(x_t = 1|s = 1) = 0.2$. Given the prefix 1, suppose $x^N = 01101$. Then, the probability of x^N given the prefix $s = 1$ can be computed as follows:

$$\begin{aligned} P_a(01101|1) &= P(0|1)P(1|10)P(1|1)P(0|1)P(1|10) \\ &= 0.8 \times 0.3 \times 0.2 \times 0.8 \times 0.3 \end{aligned}$$

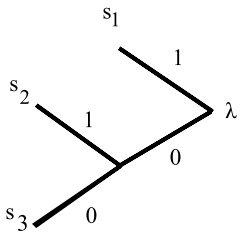

Figure 1: Tree Structure of the source in Example 1.

Some comments:

- The root λ of the tree source corresponds to the empty context. In that case the source is memoryless.
- If a tree source of depth D is complete then this is similar as saying that the source is a Markov source of order D.
- If we have a tree source of depth D then we can always add more leafs in order to make it complete and create a Markov source of order D.

5.1 Known Model, Unknown Parameters

Now, suppose that we only know the structure of the tree but we do not know the leaves. This means that we do not know any of the parameters θ_s. This problem is just a generalization of the problem described in section 4 where we did not know one parameter θ. Specifically, all symbols that correspond to the same suffix $s \in \mathcal{S}$ form a memoryless subsequence whose statistics are determined by the unknown parameter θ_s. For this subsequence we simply use the KT-estimator that we introduced in section 4.

$$
\begin{aligned}
\rho(x^N) &= L(x^N) - \log \frac{1}{P_a(x^N)} \\
&\overset{(a)}{<} \log(\frac{1}{P_e(x^N)}) - \log \frac{1}{P_a(x^N)} + 2 \\
&\overset{(b)}{=} \log(\frac{1}{\prod_{s\in\mathcal{S}} P_e(a_s, b_s)}) - \log \frac{1}{P_a(x^N)} + 2 \\
&= \sum_{s\in\mathcal{S}} \log(\frac{\theta_s^{b_s}(1-\theta_s)^{a_s}}{P_e(a_s, b_s)}) + 2 \\
&\overset{(c)}{\leq} \sum_{s\in\mathcal{S}} \left(\frac{1}{2}\log(a_s + b_s) + 1\right) + 2 \\
&\overset{(d)}{\leq} |S| \left(\frac{1}{2}\log\frac{\sum_{s\in\mathcal{S}}(a_s + b_s)}{|S|} + 1\right) + 2 \\
&= |\mathcal{S}| \left(\frac{1}{2}\log\frac{N}{|\mathcal{S}|} + 1\right) + 2,
\end{aligned}
$$

where

- (a) follows from the fact that we use Arithmetic Coding with the estimated probability $P_e(x^N)$ that the encoder knows, and thus the price is only 2 bits.
- (b) follows because $P_e(x^N) = \prod_{s\in\mathcal{S}} P_e(a_s, b_s)$.
- (c) follows if we consider each term of the summation separately and use (2).
- (d) follows from the concavity of the log-sum function.

We can make the following three observations:

- We observe again that the result qualitatively is the same as in section 4. The price for using the Arithmetic coding on the estimated model is 2 bits, and the $O(\log(N))$ term is the price for not knowing the parameters θ_s.
- Since $|S|$ is constant, it is clear that $\frac{1}{N}\rho(x^N) \to 0$ as $N \to \infty$.
- $|\mathcal{S}| = 1$ means that we only have one unknown parameter θ. This is the same model as in section 4.
- The above upper bound is actually the optimal value that we can get. The reasoning is as follows: Let $T(\mathcal{S})$ denote family of all tree source on $\mathcal{S}$. Given a specific tree model $\mathcal{S}$, but with unknown the set Θ_s it is proven that [4]:

$$
\begin{aligned}
\max_{x^N}[l(x^N) - \min_{P\in T(\mathcal{S})} \log(\frac{1}{p(x^N))}] \geq \frac{|\mathcal{S}|}{2}\log(N) + o(1) \Rightarrow \\
\min_{all\ \ schemes}\left[\max_{x^N}[l(x^N) - \min_{P\in T(\mathcal{S})} \log(\frac{1}{p(x^N))}]\right] = \frac{|\mathcal{S}|}{2}\log(N) + o(1)
\end{aligned}
$$

  This means that the worst case individual redundancy is actually bounded by $\frac{|\mathcal{S}|}{2}\log(N)$ and some constant terms. We observe that using KT-estimation we can actually achieve this bound.

5.2 Unknown Model - Context Tree

The framework that we are going to consider in this section is the one where we do not know the Tree Source itself, but we do know the maximum depth D of the Tree. This means that we are not aware of the suffixes and the kind of dependance that exists between the source symbols, but we know that there is not a dependance of larger than D time steps. In this case we can use a Context Tree to compute the appropriate coding distribution. That is, we are trying to find an estimated distribution based on the observed sequence and then use Arithmetic coding on that distribution. In order to do this cleverly we will define a weighted coding distribution which takes into account all the possible tree sources that could lead to the sequence that we observe. We hope that using this technique, the individual redundancy that we will get is small as in the previous sections. We will see that this is the case. Notice that although the CTW gets a smaller redundancy than Lempel Ziv, as a counterpart the latter does not require to know D ahead of time, while the CTW does.

Lets start by defining the context tree.

Definition 2. *The context tree $\mathcal{T}_\mathcal{D}$ 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 $s \in \mathcal{T}_\mathcal{D}$ with $l(s) < \mathcal{D}$, 'splits up' in to two nodes $0s$ and $1s$. The node s is called the parent of the nodes $0s$ and $1s$, who in turn are the children of s. To each node $s \in \mathcal{T}_\mathcal{D}$, we assosiate $a_s \geq 0$ and $b_s \geq 0$ which is the number of zeros and ones respectively up to that point. For the children $0s$ and $1s$ of parent node s, the counts must satisfy $a_{0s} + a_{1s} = a_s$ and $b_{0s} + b_{1s} = b_s$.*

A Context Tree is a complete tree up to depth D. It is complete because we do not know the real tree that we are trying to approximate. Depending on the sequence we 'learn' the parameters θ_s and we perform weighting over all the tree structures that could have generated this sequence. At this point we repeat the meaning of the word context: A context of a source symbol x_t is a suffix of the semi-infinite sequence $\ldots, x_{t-2}x_{t-1}$ that precedes it. So, a context tree up to depth D means that we know that our source has no memory more than D. Note that the context tree is well-defined since no matter what it is the suffix of x_t, there is a parameter that determines how x_t was generated. As before, each node s in the context tree is associated with the subsequence of source symbols that occurred after the context s.

Example 2 First, we have to define the depth D of the tree, and fill up the tree up to that depth. For example, take $D = 2$, and consider the sequence $x^N = 01 \mid 1010100$.

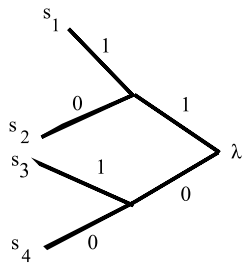

Figure 2: Tree Structure of Example 2.

Recall that a and b are the number of 0's and 1's in x^N, respectively, and a^s and b^s represent the number of 0's and 1's in x^N right after a string s, respectively. After completing the tree, we compute:

$a = 4$ ($x^N = 01 \mid 1\mathbf{0}1\mathbf{0}1\mathbf{00}$)
$b = 3$ ($x^N = 01 \mid \mathbf{1}0\mathbf{1}0100$)
$a^0 = 1$ ($x^N = 01 \mid 101010\mathbf{0}$)
$b^0 = 2$ ($x^N = 01 \mid 10\mathbf{1}0100$)
$a^1 = 3$ ($x^N = 01 \mid 1\mathbf{0}1\mathbf{0}100$)

$b^1 = 0$ ($x^N = 01 \mid 1010100$)
$a^{00} = 0$ ($x^N = 01 \mid 1010100$)
$b^{00} = 0$ ($x^N = 01 \mid 1010100$)
$a^{10} = 1$ ($x^N = 01 \mid 1010100$)
$b^{10} = 2$ ($x^N = 01 \mid 1010100$)
$a^{01} = 2$ ($x^N = 01 \mid 1010100$)
$b^{01} = 0$ ($x^N = 01 \mid 1010100$)
$a^{11} = 0$ ($x^N = 01 \mid 1010100$)
$b^{11} = 0$ ($x^N = 01 \mid 1010100$)

Now that we have explained the formulation of the Context Tree we are going to show how we can get a good estimation of the actual probability distribution of the source that will give a nice upper bound for the individual redundancy of any source sequence. Remember that the encoder and the decoder do not know neither the model of the tree (that is the set of suffixes $\mathcal{S} \in \mathcal{C}_{\mathcal{D}}$), nor the parameter vector $\Theta_{\mathcal{S}}$. They only know that the depth is less or equal to D. The idea behind the usage of Context Tree is described in the following corollary.

Corollary 3. *Suppose that a source x^N is distributed according to $P_c^1(x^N)$ or $P_c^2(x^N)$. Then, one can achieve a redundancy of 1 bit using the weighted distribution*

$$P_c^w(x^N) = \frac{P_c^1(x^N) + P_c^2(x^N)}{2} \tag{5}$$

Exercise:
Compute the worst case redundancy in the case where we use the distribution P_c^1 to encode x^N when the truth distribution of the source is P_c^2 instead.

Proof Assume that we encode using $P_c^w(x^N)$. Then,

$$L(x^N) = \log \frac{2}{P_1(x^N) + P_2(x^N)},$$

and the individual redundancy for $i \in \{1, 2\}$ and $j = \{1, 2\} \backslash \{i\}$ is upper bounded by:

$$\begin{aligned}
\rho_i(x^N) &= L(x^N) - \log \frac{1}{P_i(x^N)} \\
&= \log \frac{2P_i(x^N)}{P_1(x^N) + P_2(x^N)} \\
&= \log \frac{2}{1 + \frac{P_j(x^N)}{P_i(x^N)}} \\
&\leq 1
\end{aligned}$$

If the two distributions are the same then we observe that the individual redundancy is 0, as expected. If not, the price that we pay from the weighting is only 1 bit. If we stated differently, the bound on the codeword length increases by 1 bit. In practice the increase is far less, especially if $P_c^1(x_1^N)$ and $P_c^2(x_1^N)$ are approximately equal. Note that, if after observing x_1^N we select the i that minimizes $P_c^i(x_1^N)$, we lose exactly 1 bit. This bit is now needed to specify the source index. □

Using the above result, in the context tree, we are trying to find a weighted probability of a node of the tree in order to take into account that we do not know the model. Specifically we are going to use the following weighting. The weighting is not unique. However we prefer a simple recursive weighting.

Definition 4. *Given a tree source of depth D and $\mathcal{S} = (s_1, s_2, \ldots, s_n)$, the weight at a specific node $\Gamma(s)$ is given by:*

$$\Gamma_D(s) = \begin{cases} 0 & , \text{ if } s \in \mathcal{S} \\ 1\Gamma(0s)\Gamma(1s) & , \text{ if } s \notin \mathcal{S} \\ \emptyset & , \text{ if } l(s) = D \end{cases} \tag{6}$$

Observe that the weight will depend only on the set of strings, not in the probability of the leaves. Every tree source has a weight $\Gamma(\mathcal{S}) = |\Gamma(\lambda)|$ associated to it, which is upper bounded by the number of nodes, and therefore by 2^D. The main characteristic is that you can find it in a recursive way.

Example 3 Consider for example the following tree source with maximum depth 3:
Then, its weight is given by

$$\begin{aligned} \Gamma_3(\mathcal{S}) &= |\Gamma(\lambda)| \\ &= |1\Gamma(0)\Gamma(1)| \\ &= |11\Gamma(00)\Gamma(10)0| \\ &= |11000| \\ &= 5 \end{aligned}$$

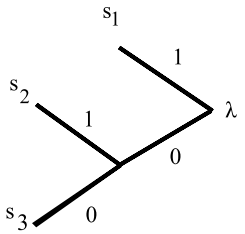

Figure 3: Tree structure of Example 3.

Definition 5. *To each node $s \in \mathcal{C}_\mathcal{D}$, we assign a weighted probability P_w^s which is defined as*

$$P_w^s = \begin{cases} \frac{1}{2}P_e^s(a_s, b_s) + \frac{1}{2}P_w^{s0}P_w^{s1} & , \text{ if } l(s) \neq D \\ P_e^s(a_s, b_s) & , \text{ if } l(s) = D \end{cases} \tag{7}$$

What the above definition says is that if the suffix s is on the leafs of the tree, then the probability of the current node is the estimated probability $P_e^s(a_s, b_s)$. However, if this is not true, then the probability P_w^s is the average between the estimated probability and the product of the weighted probabilities that correspond to the children. The weighted probabilities of a node can be regarded as a weighting over the estimated probabilities corresponding to all the (sub-) models that live above this node. This point of view of the above algorithm is shown in the next lemma.

Lemma 6. *The weighted probability P_w^s of a node $s \in \mathcal{T}_\mathcal{D}$ with $l(s) = d$ for $0 \leq d \leq D$ satisfies*

$$P_w^s = \sum_{\mathcal{U} \in \mathcal{C}_{\mathcal{D}-d}} 2^{-\Gamma_{\mathcal{D}-d}(\mathcal{U})} \prod_{u \in \mathcal{U}} P_e(a_{us}, b_{us})$$

with $\sum_{\mathcal{U} \in \mathcal{C}_{\mathcal{D}-d}} 2^{-\Gamma_{\mathcal{D}-d}(\mathcal{U})} = 1$. *The summation is over all complete and proper suffix sets $\mathcal{U}$.*

Proof

$$\begin{aligned}
P_w^s &= \frac{1}{2} P_e(a_s, b_s) + \frac{1}{2} P_w^{0s} P_w^{1s} \\
&\overset{(a)}{=} \frac{1}{2} P_e(a_s, b_s) \\
&+ \frac{1}{2} \Big(\sum_{\mathcal{V} \in \mathcal{C}_{D-d}} 2^{-\Gamma_{D-d}(\mathcal{V})} \prod_{v \in \mathcal{V}} P_e(a_{v0s}, b_{v0s}) \Big) \Big(\sum_{\mathcal{W} \in \mathcal{C}_{D-d}} 2^{-\Gamma_{D-d}(\mathcal{W})} \prod_{w \in \mathcal{W}} P_e(a_{w1s}, b_{w1s}) \Big) \\
&\overset{(b)}{=} 2^{-1} P_e(a_s, b_s) + \sum_{\mathcal{W}, \mathcal{V} \in \mathcal{C}_{D-d}} 2^{-1-\Gam_{D-d}(\mathcal{V})-\Gamma_{D-d}(\mathcal{W})} \prod_{v \in \mathcal{V} \times 0 \cup \mathcal{W} \times 1} P_e(a_{us}, b_{us}) \\
&\overset{(c)}{=} \sum_{\mathcal{U} \in \mathcal{C}_{D-d+1}} 2^{-\Gamma_{D-d+1}(\mathcal{U})} \prod_{u \in \mathcal{U}} P_e(a_{us}, b_{us}),
\end{aligned}$$

where (a) follows by using the induction hypothesis, (b) follows by rearranging the terms and (c) follows from (7). The conclusion is that the hypothesis also holds for $d-1$, and by induction for all $0 \leq d \leq D$.

The fact $\sum_{\mathcal{U} \in \mathcal{C}_{D-d}} 2^{-\Gamma_{D-d}(\mathcal{U})} = 1$ can be proved similarly if we note that

$$\sum_{\mathcal{U} \in \mathcal{C}_{D-d+1}} 2^{-\Gamma_{D-d+1}(\mathcal{U})} = \frac{1}{2} + \frac{1}{2} \Big(\sum_{\mathcal{V} \in \mathcal{C}_{D-d}} 2^{-\Gamma_{D-d}(\mathcal{V})} \Big) \Big(\sum_{\mathcal{W} \in \mathcal{C}_{D-d}} 2^{-\Gamma_{D-d}(\mathcal{W})} \Big) = \frac{1}{2} + \frac{1}{2} = 1 \tag{8}$$

□

Now we show some properties of P_w^s.

1. Let P_w^λ the estimated weighted probability at the root λ.

$$P_w^\lambda = \sum_{\mathcal{U} \in \mathcal{C}_\mathcal{D}} 2^{-\Gamma_\mathcal{D}(\mathcal{U})} \prod_{u \in \mathcal{U}} P_e(a_u, b_u) \geq 2^{-\Gamma_D(\lambda)} \prod_{s \in \mathcal{S}} P_e(a_s, b_s) \tag{9}$$

2. If $x_{t-l(s)+1}^t = s$,

$$P_w^s(x^t, x_{t+1} = 0) + P_w^s(x^t, x_{t+1} = 1) = P_w^s(x^t) \tag{10}$$

3. If $x_{t-l(s)+1}^t \neq s$,

$$P_w^s(x^t, x_{t+1} = 0) = P_w^s(x^t, x_{t+1} = 1) \tag{11}$$

Finally, we want to compute what we lose by the Context Tree Weighting algorithm. Assume $P_a(x^N)$ is the actual probability (that we do not know) and $P_c(x^N)$ is the probability obtained at the root of the tree (the estimated probability of observing x^N which is obtained from the CTW algorithm).

$$\begin{aligned}\rho(x^N) &= L(x^N) - \log \frac{1}{P_a(x^N)} \\ &\overset{(a)}{=} \log \frac{\prod_{s\in\mathcal{S}} P_e(a_s,b_s)}{P_w^\lambda(x^N)} + \log \frac{P_a(x^N)}{\prod_{s\in\mathcal{S}} P_e(a_s,b_s)} + L(x^N) - \log \frac{1}{P_w^\lambda(x^N)} \\ &\overset{(b)}{<} \Gamma_D(\lambda) + |S| \left(\frac{1}{2}\log\frac{N}{|S|} + 1\right) + 2,\end{aligned}$$

where

- (a) follows by adding and subtracting the terms $\log\left(\prod_{s\in\mathcal{S}} P_e(a_s,b_s)\right)$ and $\log \frac{1}{P_w^\lambda(x^N)}$. Notice that the dominating part is $|S|\left(\frac{1}{2}\log\frac{N}{|S|}+1\right)$.
- (b) follows for the following reasons:
  - We perform Arithmetic Coding in the distribution $P_w^\lambda(x^N)$. Therefore:
$$L(x^N) - \log\frac{1}{P_w^\lambda(x^N)} < 2$$
  - From relation (9) we get that:
$$\log \frac{\prod_{s\in\mathcal{S}} P_e(a_s,b_s)}{P_w^\lambda(x^N)} \le \Gamma_D(\lambda)$$

    This represents the price that we pay because we do not know the actual model of the tree source. We note again that $\Gamma_D(\lambda)$ is upper bounded by the number of nodes in the tree source (2^D) and that it is independent of N.
  - From section 4 where we knew the model and not the parameters we get that:
$$\log \frac{P_a(x^N)}{\prod_{s\in\mathcal{S}} P_e(a_s,b_s)} < |S|\left(\frac{1}{2}\log\frac{N}{|S|}+1\right)$$

Note that:

- Even in a source coding problem of a tree source where we only know the maximum depth, Context tree weighting achieves the optimal worst case individual redundancy.

References

[1] R.E. Krichevsky and V.K. Trofimov, "The Perfomance of Universal Encoding", IEEE Trans. Information Theory, vol. IT-27, pp. 199-207, March 1981.

[2] T.M. Cover and J.A. Thomas, "Elements of Information". New York: John Wiley, 1991

[3] Frans Willems, Yuri Shtarkov, and Tjalling Tjalkens, "Reflections on The Context-Tree Weighting Method: Basic Properties", IEEE Information Theory Society Newsletter, Vol. 47, No. 1, March 1997

[4] J. Rissanen, "Universal coding, information, prediction, and estimation", IEEE Trans. on IT, vol. 30, pp. 629-636, July 1984.