Context Tree Switching

Joel Veness[†] Kee Siong Ng[‡] Marcus Hutter[‡] Michael Bowling[†]

[†]University of Alberta, Edmonton, Canada
[‡]Australian National University, Canberra, Australia

Abstract

This paper describes the Context Tree Switching technique, a modification of Context Tree Weighting for the prediction of binary, stationary, n-Markov sources. By modifying Context Tree Weighting's recursive weighting scheme, it is possible to mix over a strictly larger class of models without increasing the asymptotic time or space complexity of the original algorithm. We prove that this generalization preserves the desirable theoretical properties of Context Tree Weighting on stationary n-Markov sources, and show empirically that this new technique leads to consistent improvements over Context Tree Weighting as measured on the Calgary Corpus.

1 Introduction

Context Tree Weighting [Willems et al., 1995] is a well-known, universal lossless compression algorithm for binary, stationary, n-Markov sources. It provides a striking example of a technique that works well both in theory and practice. Similar to Prediction by Partial Matching [Cleary et al., 1984], Context Tree Weighting (CTW) uses a context tree data structure to store statistics about the current data source. These statistics are recursively combined by *weighting*, which leads to an elegant algorithm whose worst-case performance can be characterized by an analytic regret bound that holds for *any* finite length data sequence, as well as asymptotically achieving (in expectation) the lower bound of Rissanen [1984] for the class of binary, stationary n-Markov sources.

This paper explores an alternative recursive weighting procedure for CTW, which weights over a strictly larger class of models without increasing the asymptotic time or space complexity of the original algorithm. We call this new procedure the Context Tree Switching (CTS) algorithm, which we investigate both theoretically and empirically.

2 Background

We begin with some notation and definitions for binary data generating sources. Our binary alphabet is denoted by $\mathcal{X} := \{0, 1\}$. A binary string $x_1 x_2 \ldots x_n \in \mathcal{X}^n$ of length n is denoted by $x_{1:n}$. The prefix $x_{1:j}$ of $x_{1:n}$, $j \leq n$, is denoted by $x_{\leq j}$ or $x_{<j+1}$. The empty string is denoted by ϵ. The concatenation of two strings s and r is denoted by sr. If $\mathcal{S}$ is a set of strings and $r \in \{0, 1\}$, then $\mathcal{S} \times r := \{sr : s \in \mathcal{S}\}$. We will also use $l(s)$ to denote the length of a string s.

2.1 Probabilistic Binary Sources

We define a probabilistic data generating source ρ to be a set of functions $\rho_n : \mathcal{X}^n \to [0,1]$, for $n \in \mathbb{N}$, satisfying the constraint that $\rho_n(x_{1:n}) = \sum_{y \in \mathcal{X}} \rho_{n+1}(x_{1:n}y)$ for all $x_{1:n} \in \mathcal{X}^n$, with base case $\rho_0(\epsilon) = 1$. As the meaning is always clear from the argument to ρ, we drop the subscripts on ρ from here onwards. Under this definition, the conditional probability of a symbol x_n given previous data $x_{<n}$ is defined as $\rho(x_n|x_{<n}) := \rho(x_{1:n})/\rho(x_{<n})$ if $\rho(x_{<n}) > 0$, with the familiar chain rule $\rho(x_{1:n}) = \prod_{i=1}^{n} \rho(x_i|x_{<i})$ now following.

2.2 Coding and Redundancy

A source code $c : \mathcal{X}^* \to \mathcal{X}^*$ assigns to each possible data sequence $x_{1:n}$ a binary codeword $c(x_{1:n})$ of length $l_c(x_{1:n})$. The typical goal when constructing a source code is to minimize the lengths of each codeword while ensuring that the original data sequence $x_{1:n}$ is always recoverable from $c(x_{1:n})$. Given a data generating source μ, we know from Shannon's Source Coding Theorem that the optimal (in terms of expected code length) source code c uses codewords of length $-\log_2 \mu(x_{1:n})$ bits for all $x_{1:n}$. This motivates the notion of the *redundancy* of a source code c given a sequence $x_{1:n}$, which is defined as $r_c(x_{1:n}) := l_c(x_{1:n}) + \log_2 \mu(x_{1:n})$. Provided the data generating source is known, near optimal redundancy can essentially be achieved by using arithmetic encoding [Witten et al., 1987]. More precisely, using a_μ to denote the source code obtained by arithmetic coding using probabilistic model μ, the resultant code lengths are known to satisfy

$$l_{a_\mu}(x_{1:n}) < \lceil -\log_2 \mu(x_{1:n}) \rceil + 2, \tag{1}$$

which implies that $r_{a_\mu}(x_{1:n}) < 2$ for all $x_{1:n}$. Typically however, the true data generating source μ is unknown. The data can still be coded using arithmetic encoding with an alternate model ρ, however now we expect to use an extra $\mathbb{E}_\mu \left[\log_2 \mu(x_{1:n})/\rho(x_{1:n})\right]$ bits to code the random sequence $x_{1:n} \sim \mu$.

2.3 Weighting and Switching

This section describes the two fundamental techniques, *weighting* and *switching*, that are the key building blocks of Context Tree Weighting and the new Context Tree Switching algorithm.

2.3.1 Weighting

Suppose we have a finite set $\mathcal{M} := \{\rho_1, \rho_2, \ldots, \rho_N\}$, for some $N \in \mathbb{N}$, of candidate data generating sources. Consider now a source coding distribution ξ defined as

$$\xi(x_{1:n}) := \sum_{\rho \in \mathcal{M}} w_0^\rho \rho(x_{1:n}) \tag{2}$$

formed by weighting each model by a real number $w_0^\rho > 0$ such that $\sum_{\rho \in \mathcal{M}} w_0^\rho = 1$. Notice that if some model $\rho^* \in \mathcal{M}$ is a good source coding distribution for a data sequence $x_{1:n}$, then provided n is sufficiently large, ξ will be a good coding distribution, since

$$-\log_2 \xi(x_{1:n}) = -\log_2 \sum_{\rho \in \mathcal{M}} w_0^\rho \rho(x_{1:n}) \leq -\log_2 w_0^\rho \rho(x_{1:n}) = -\log_2 w_0^\rho - \log_2 \rho(x_{1:n}) \tag{3}$$

holds for all $\rho \in \mathcal{M}$. Therefore, we would only need at most an extra $-\log_2 w_0^{\rho^*}$ bits, an amount independent of n, to transmit $x_{1:n}$ using ξ instead of the best model ρ^* in $\mathcal{M}$. An important special case of this result is when $|\mathcal{M}| = 2$ and $w_0^{\rho_1} = w_0^{\rho_2} = \frac{1}{2}$, when only 1 extra bit is required.

2.3.2 Switching

While weighting provides an easy way to combine models, as an ensemble method it is somewhat limited in that it only guarantees performance in terms of the best *single* model in $\mathcal{M}$. It is easy to

imagine situations where this would be insufficient in practice. Instead, one could consider weighting over *sequences* of models chosen from a fixed base class $\mathcal{M}$. Variants of this fundamental idea have been considered by authors from quite different research communities. Within the data compression community, there is the Switching Method and the Snake algorithm [Volf and Willems, 1998]. Similar approaches were also considered in the online learning community, in particular the Fixed-Share [Herbster and Warmuth, 1998] algorithm for tracking the best expert over time. From the machine learning community, related ideas were investigated in the context of Bayesian model selection, giving rise to the Switch Distribution [van Erven et al., 2007]. The setup we use draws most heavily on [van Erven et al., 2007], though there appears to be considerable overlap amongst the approaches.

Definition 1. *Given a finite model class $\mathcal{M} = \{\rho_1, \ldots, \rho_N\}$ with $N > 1$, for all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{X}^n$, the Switch Distribution with respect to $\mathcal{M}$ is defined as*

$$\tau_\alpha(x_{1:n}) := \sum_{i_{1:n} \in \mathcal{I}_n(\mathcal{M})} w(i_{1:n}) \prod_{k=1}^{n} \rho_{i_k}(x_k | x_{<k}) \tag{4}$$

where the prior over model sequences is recursively defined by

$$w(i_{1:n}) := \begin{cases} 1 \quad \textit{if} \quad i_{1:n} = \epsilon \\ \frac{1}{N} \quad \textit{if} \quad n = 1 \\ w(i_{<n}) \times \left((1 - \alpha_n) \mathbb{I}[i_n = i_{n-1}] + \frac{\alpha_n}{|\mathcal{M}|-1} \mathbb{I}[i_n \neq i_{n-1}] \right) \textit{ otherwise,} \end{cases}$$

with each switch rate $\alpha_k \in [0, 1]$ for $1 < k \leq n$, and $\mathcal{I}_n(\mathcal{M}) := \left\{ x \in \{1, 2, \ldots, N\}^n \right\}$.

Now, using the same argument to bound $-\log_2 \tau_\alpha(x_{1:n})$ as we did with $-\log_2 \xi(x_{1:n})$ in Section 1, we see that the inequality

$$-\log_2 \tau_\alpha(x_{1:n}) \leq -\log_2 w(i_{1:n}) - \log_2 \rho_{i_{1:n}}(x_{1:n}) \tag{5}$$

holds for any sequence of models $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$, with $\rho_{i_{1:n}}(x_{1:n}) := \prod_{k=1}^{n} \rho_{i_k}(x_k | x_{<k})$. By itself, Equation 5 provides little reassurance since the $-\log_2 w(i_{1:n})$ term might be large. However, by decaying the switch rate over time, a meaningful upper bound on $-\log_2 w(i_{1:n})$ that holds for any sequence of model indices $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$ can be derived.

Lemma 1. *Given a base model class $\mathcal{M}$ and a decaying switch rate $\alpha_t := \frac{1}{t}$ for $t \in \mathbb{N}$,*

$$-\log_2 w(i_{1:n}) \leq (m(i_{1:n}) + 1) \left(\log_2 |\mathcal{M}| + \log_2 n \right),$$

for all $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$, where $m(i_{1:n}) := \sum_{k=2}^{n} \mathbb{I}[i_k \neq i_{k-1}]$ denotes the number of switches in $i_{1:n}$.

Proof. See Appendix B. □

Now by combining Equation 5 with Lemma 1 and taking the minimum over $\mathcal{I}_n(\mathcal{M})$ we get the following upper bound on $-\log_2 \tau_\alpha(x_{1:n})$.

Theorem 1. *Given a base model class $\mathcal{M}$ and switch rate $\alpha_t := \frac{1}{t}$ for $t \in \mathbb{N}$, for all $n \in \mathbb{N}$,*

$$-\log_2 \tau_\alpha(x_{1:n}) \leq \min_{i_{1:n} \in \mathcal{I}_n(\mathcal{M})} \left\{ (m(i_{1:n}) + 1) \left[\log_2 |\mathcal{M}| + \log_2 n \right] - \log_2 \rho_{i_{1:n}}(x_{1:n}) \right\}.$$

Thus if there exists a good coding distribution $\rho_{i_{1:n}}$ such that $m(i_{1:n}) \ll n$ then τ_α will also be a good coding distribution. Additionally, it is natural to compare the performance of switching to weighting in the case where the best performing sequence of models satisfies $m(i_{1:n}) = 0$. Here an extra cost of $O(\log n)$ bits is incurred, which is a small price to pay for a significantly larger class of models.

Algorithm A direct computation of Equation 4 is intractable. For example, given a data sequence $x_{1:n}$ and a model class $\mathcal{M}$, the sum in Equation 4 would require $|\mathcal{M}|^n$ additions. Fortunately, the structured nature of the model sequence weights $w(i_{1:n})$ can be exploited to derive Algorithm 1, whose proof of correctness can be found in Appendix A. Assuming that every conditional probability can be computed in constant time, Algorithm 1 runs in $\Theta(n|\mathcal{M}|)$ time and uses only $\Theta(|\mathcal{M}|)$ space. Furthermore, only $\Theta(|\mathcal{M}|)$ work is required to process each new symbol.

Algorithm 1 SWITCH DISTRIBUTION - $\tau_\alpha(x_{1:n})$

Require: A finite model class $\mathcal{M} = \{\rho_1, \ldots, \rho_N\}$ such that $N > 1$
Require: A weight vector $(w_1, \ldots, w_N) \in \mathbb{R}^N$, with $w_i = \frac{1}{N}$ for $1 \leq i \leq N$
Require: A sequence of switching rates $\{\alpha_2, \alpha_3, \ldots, \alpha_n\}$

$r \leftarrow 1$
for $i = 1$ to n **do**
  $r \leftarrow \sum_{j=1}^{N} w_j \rho_j(x_i|x_{<i})$
  $k \leftarrow (1 - \alpha_{i+1})N - 1$
  for $j = 1$ to N **do**
    $w_j \leftarrow \frac{1}{N-1}\left[\alpha_{i+1} r + k w_j \rho_j(x_i|x_{<i})\right]$
  end for
end for
return r

Discussion The above switching technique can be used in a variety of ways. For example, drawing inspiration from Volf and Willems [1998], multiple probabilistic models (such as PPM and CTW) could be combined with this technique, with the conditional probability $\tau_\alpha(x_n|x_{<n})$ of each symbol x_n given by the ratio $\tau_\alpha(x_{1:n})/\tau_\alpha(x_{<n})$. This seems to be a direct improvement over the Switching Method [Volf and Willems, 1998], since similar theoretical guarantees are obtained, while additionally reducing the time and space required to process each new symbol x_n from $O(n)$ to $O(|\mathcal{M}|)$. This, however, is not the focus of our paper. Rather, the improved computational properties of Algorithm 1 motivated us to investigate whether the Switch Distribution can be used as a replacement for the recursive weighting operation inside CTW. It is worth pointing out that the idea of using a switching method recursively inside a context tree had been discussed before in Appendix A of [Volf, 2002]. This discussion focused on some of the challenges that would need to be overcome in order to produce a "Context Tree Switching" algorithm that would be competitive with CTW. The main contribution of this paper is to describe an algorithm that achieves these goals both in theory and practice.

2.4 Context Tree Weighting

As our new Context Tree Switching approach extends Context Tree Weighting, we must first review some of CTW's technical details. We recommend [Willems et al., 1995, 1997] for more information.

2.4.1 Overview

Context Tree Weighting is a binary sequence prediction technique that works well both in theory and practice. It is a variable order Markov modeling technique that works by computing a "double mixture" over the space of *all* Prediction Suffix Trees (PSTs) of bounded depth $D \in \mathbb{N}$. This involves weighting (see Section 2.3.1) over all PST structures, as well as integrating over all possible parameter values for each PST structure. We now review this process, beginning by describing how an unknown, memoryless, stationary binary sources is handled, before moving on to describe how memory can be

added through the use of a Prediction Suffix Tree, and then finishing by showing how to efficiently weight over all PST structures.

2.4.2 Memoryless, Stationary, Binary Sources

Consider a sequence $x_{1:n}$ generated by successive Bernoulli trials. If a and b denote the number of zeroes and ones in $x_{1:n}$ respectively, and $\theta \in [0,1] \subset \mathbb{R}$ denotes the probability of observing a 1 on any given trial, then $\Pr(x_{1:n} \,|\, \theta) = \theta^b(1-\theta)^a$. One way to construct a distribution over $x_{1:n}$, in the case where θ is unknown, is to weight over the possible values of θ. A good choice of weighting can be obtained via an objective Bayesian analysis, which suggests using the weighting $w(\theta) := \text{Beta}(\frac{1}{2},\frac{1}{2}) = \pi^{-1}\theta^{-1/2}(1-\theta)^{-1/2}$. The resultant estimator is known as the Krichevsky-Trofimov (KT) estimator [Krichevsky and Trofimov, 1981]. The KT probability of a binary data sequence $x_{1:n}$ is defined as

$$\xi_{KT}(x_{1:n}) := \int_0^1 \theta^b(1-\theta)^a \, w(\theta) \, d\theta, \tag{6}$$

for all $n \in \mathbb{N}$. Furthermore, $\xi_{KT}(x_{1:n})$ can be efficiently computed online using the identities

$$\xi_{KT}(x_n = 0 \mid x_{<n}) = \frac{a + 1/2}{a+b+1}, \quad \xi_{KT}(x_n = 1 \mid x_{<n}) = \frac{b+1/2}{a+b+1} \tag{7}$$

in combination with the chain rule $\xi_{KT}(x_{1:n}) = \xi_{KT}(x_n|x_{<n}) \times \xi_{KT}(x_{<n})$.

Parameter Redundancy The parameter redundancy of the KT estimator can be bounded uniformly. Restating a result from Willems et al. [1995], one can show that for all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{X}^n$, for all $\theta \in [0,1]$,

$$\log_2 \frac{\theta^b(1-\theta)^a}{\xi_{KT}(x_{1:n})} \leq \tfrac{1}{2}\log_2(n) + 1. \tag{8}$$

This result plays an important role in the analysis of both CTW and CTS.

2.4.3 Variable-length Markov, Stationary, Binary Sources

A richer class of data generating sources can be defined if we let the source model use memory. A finite, variable order, binary Markov model [Begleiter et al., 2004] is one such model. This can equivalently be described by a binary Prediction Suffix Tree (PST). A PST is formed from two main components: a *structure*, which is a binary tree where all the left edges are labeled 1 and all the right edges are labeled 0; and a set of real-valued parameters within $[0,1]$, with one parameter for every leaf node in the PST structure. This is now formalized.

Definition 2. *A suffix set $\mathcal{S}$ is a collection of binary strings. $\mathcal{S}$ is said to be proper if no string in $\mathcal{S}$ is a suffix of any other string in $\mathcal{S}$. $\mathcal{S}$ is complete if every semi-infinite binary string $\cdots x_{n-2}x_{n-1}x_n$ has a suffix in $\mathcal{S}$. $\mathcal{S}$ is of bounded depth $D \in \mathbb{N}$ if $l(s) \leq D$ for all $s \in \mathcal{S}$.*

A binary Prediction Suffix Tree structure is uniquely described by a complete and proper suffix set. For example, the suffix set associated with the PST in Figure 1 is $\mathcal{S} := \{1, 10, 00\}$, with each suffix $s \in \mathcal{S}$ describing a path from a leaf node to the root.

Definition 3. *A PST is a pair $(\mathcal{S}, \Theta_{\mathcal{S}})$, where $\mathcal{S}$ is a suffix set and $\Theta_{\mathcal{S}} := \{\theta_s : \theta_s \in [0,1]\}_{s\in\mathcal{S}}$. The depth of a suffix set $\mathcal{S}$ is defined as $d(\mathcal{S}) := \max_{s\in\mathcal{S}} l(s)$. The context with respect to a suffix set $\mathcal{S}$ of a binary sequence $x_{1:n} \in \mathcal{X}^n$ is defined as $\phi_{\mathcal{S}}(x_{1:n}) := x_{k:n}$, where k is the unique integer such that $x_{k:n} \in \mathcal{S}$.*

Notice that $\phi_{\mathcal{S}}(x_{1:n})$ may be undefined when $n < d(\mathcal{S})$. To avoid this problem, from here onwards we adopt the convention that the first $d(\mathcal{S})$ bits of any sequence are held back and coded separately. By denoting these bits as $x_{D-1} \ldots x_{-1}x_0$, our previous definition of $\phi_{\mathcal{S}}(x_{1:n})$ is always well defined.

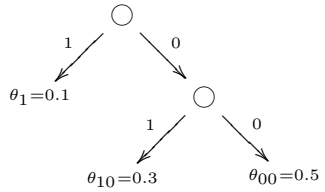

Figure 1: An example prediction suffix tree

Semantics A PST $(\mathcal{S}, \Theta_{\mathcal{S}})$ maps each binary string $x_{1:n}$ with $n \geq d(\mathcal{S})$ to a parameter value $\theta_{\phi_{\mathcal{S}}(x_{1:n})}$, with the intended meaning that $\Pr(x_{n+1} = 1 \mid x_{1:n}) = \theta_{\phi_{\mathcal{S}}(x_{1:n})}$. For example, the PST in Figure 1 maps the string 1110 to $\theta_{\phi_{\mathcal{S}}(1110)} = \theta_{10} = 0.3$, which means the next bit after 1110 takes on a value of 1 with probability 0.3, and a value of 0 with probability 0.7. If we let b_s and a_s denote the number of times a 1 and 0 is seen in context s respectively, this gives

$$\Pr(x_{1:n} \mid \mathcal{S}, \Theta_{\mathcal{S}}) := \prod_{s \in \mathcal{S}} \theta^{b_s} (1 - \theta_s)^{a_s}. \tag{9}$$

Unknown Parameters Given a PST with known structure $\mathcal{S}$ but unknown parameters $\Theta_{\mathcal{S}}$, a good coding distribution can be obtained by replacing each unknown parameter value $\theta_s \in \Theta_{\mathcal{S}}$ with a KT estimator. If we let $x^s_{1:n}$ denote the (possibly non-contiguous) subsequence of data $x_{1:n}$ that matches context $s \in \mathcal{S}$, this gives

$$\Pr(x_{1:n} \mid \mathcal{S}) := \prod_{s \in \mathcal{S}} \xi_{KT}(x^s_{1:n}). \tag{10}$$

This choice is justified by the analysis of Willems et al. [1995]. If we let

$$\gamma(k) := \begin{cases} k & \text{if } 0 \leq k < 1 \\ \frac{1}{2}\log_2(k) + 1 & \text{if } k \geq 1, \end{cases}$$

the parameter redundancy of a PST with known structure $\mathcal{S}$ can be bounded by

$$\log_2 \frac{\Pr(x_{1:n} \mid \mathcal{S}, \Theta_{\mathcal{S}})}{\Pr(x_{1:n} \mid \mathcal{S})} \leq |\mathcal{S}| \gamma(\tfrac{n}{|\mathcal{S}|}). \tag{11}$$

2.4.4 Weighting Over Prediction Suffix Trees

The Context Tree Weighting algorithm combines the data partitioning properties of a PST, a carefully chosen weighting scheme, and the distributive law to efficiently weight over the space of PST structures of bounded depth $D \in \mathbb{N}$. We now introduce some notation to make this process explicit.

Definition 4. *The set of all complete and proper suffix sets of bounded depth D is denoted by $\mathcal{C}_D$, and is given by the recurrence*

$$\mathcal{C}_D := \begin{cases} \{\{\epsilon\}\} & \text{if } D = 0 \\ \{\{\epsilon\}\} \cup \{\mathcal{S}_1 \times 1 \cup \mathcal{S}_2 \times 0 : \mathcal{S}_1, \mathcal{S}_2 \in \mathcal{C}_{D-1}\} & \text{if } D > 0. \end{cases} \tag{12}$$

Notice that $|\mathcal{C}_D|$ grows roughly double exponentially in D. For example, $|\mathcal{C}_0| = 1, |\mathcal{C}_1| = 2, |\mathcal{C}_2| = 5, |\mathcal{C}_3| = 26, |\mathcal{C}_4| = 677, |\mathcal{C}_5| = 458330$, which means that some ingenuity is required to weight over all $\mathcal{C}_D$ for any reasonably sized D. This comes in the form of a weighting scheme that is derived from a natural prefix coding of the structure of a PST. It works as follows: given a PST structure with depth no more than D, a pre-order traversal of the tree is performed. Each time an internal node is encountered for the first time, a 1 is written down. Each time a leaf node is encountered, a 0 is written if the depth of the leaf node is less than D, otherwise nothing is written. For example, if $D = 3$, the code for the

model shown in Figure 1 is 10100; if $D = 2$, the code for the same model is 101. We now define the cost $\Gamma_D(\mathcal{S})$ of a suffix set $\mathcal{S}$ to be the length of its code. One can show that $\sum_{\mathcal{S}\in\mathcal{C}_D} 2^{-\Gamma_D(\mathcal{S})} = 1$; i.e. the prefix code is complete. Thus we can now define

$$\textsc{ctw}_D(x_{1:n}) := \sum_{\mathcal{S}\in\mathcal{C}_D} 2^{-\Gamma_D(\mathcal{S})} \prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n}). \tag{13}$$

Notice also that this choice of weighting imposes an Ockham-like penalty on large PST structures.

Recursive Decomposition If we let $\mathcal{K}_D := \{0,1\}^*$ denote the set of all possible contexts for class $\mathcal{C}_D$, $x^c_{1:n}$ denote the subsequence of data $x_{1:n}$ that matches context $c \in \mathcal{K}_D$, and define $\textsc{ctw}^\epsilon_D(x_{1:n}) := \textsc{ctw}_D(x_{1:n})$, we can decompose Equation 13 into (see [Willems et al., 1995])

$$\textsc{ctw}^c_D(x_{1:n}) = \tfrac{1}{2}\,\xi_{KT}(x^c_{1:n}) + \tfrac{1}{2}\,\textsc{ctw}^{0c}_{D-1}(x_{1:n})\,\textsc{ctw}^{1c}_{D-1}(x_{1:n}), \tag{14}$$

for $D > 0$. In the base case of a single node (i.e. weighting over $\mathcal{C}_0$) we have $\textsc{ctw}^c_0(x_{1:n}) = \xi_{KT}(x^c_{1:n})$.

Computational Properties The efficiency of CTW derives from Equation 14, since the double mixture can be maintained incrementally by applying it $D+1$ times to process each new symbol. Therefore, using the Context Tree Weighting algorithm, only $O(nD)$ time is required to compute $\textsc{ctw}_D(x_{1:n})$. Furthermore, only $O(D)$ work is required to compute $\textsc{ctw}_D(x_{1:n+1})$ from $\textsc{ctw}_D(x_{1:n})$.

Theoretical Properties Using Equation 3, the model redundancy can be bounded by

$$-\log_2 \textsc{ctw}_D(x_{1:n}) = -\log_2 \left(\sum_{\mathcal{S}\in\mathcal{C}_D} 2^{-\Gamma_D(\mathcal{S})} \prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n}) \right) < \Gamma_D(\mathcal{S}) - \log_2 \prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n}).$$

This can be combined with the parameter redundancy specified by Equation 11 to give

$$-\log_2 \textsc{ctw}_D(x_{1:n}) < \Gamma_D(\mathcal{S}) + |\mathcal{S}|\gamma\left(\tfrac{n}{|\mathcal{S}|}\right) - \log_2 \Pr(x_{1:n} \,|\, \mathcal{S}, \Theta_\mathcal{S}) \tag{15}$$

for any $\mathcal{S} \in \mathcal{C}_D$. Finally, combining Equation 15 with the coding redundancy bound given in Equation 1 leads to the main theoretical result for CTW.

Theorem 2 (Willems et al. [1995])**.** *For all $n \in \mathbb{N}$, given a data sequence $x_{1:n} \in \mathcal{X}^n$ generated by a binary PST source $(\mathcal{S}, \Theta_\mathcal{S})$ with $\mathcal{S} \in \mathcal{C}_D$ and $\Theta_\mathcal{S} := \{\theta_s : \theta_s \in [0,1]\}_{s\in\mathcal{S}}$, the redundancy of CTW using context depth $D \in \mathbb{N}$ is upper bounded by $\Gamma_D(\mathcal{S}) + |\mathcal{S}|\gamma\left(\frac{n}{|\mathcal{S}|}\right) + 2$.*

3 Context Tree Switching

Context Tree Switching is a natural combination of CTW and switching. To see this, first note that Equation 14 allows us to interpret CTW as a recursive application of the weighting method of Section 2.3.1. Recalling Theorem 1, we know that a careful application of switching essentially preserves the good properties of weighting, and may even work better provided some rarely changing sequence of models predicts the data well. Using a class of PST models, it seems reasonable to suspect that the best model may change over time; for example, a large PST model might work well given sufficient data, but before then a smaller model might be more accurate due to its smaller parameter redundancy. The main insight behind CTS is to weight over all *sequences* of bounded depth PST structures by recursively using the efficient switching technique of Section 2.3.2 as a replacement for Equation 14. This gives, for all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{X}^n$, the following recursion for $D > 0$,

$$\textsc{cts}^c_D(x_{1:n}) := \sum_{i_{1:n_c}\in\{0,1\}^{n_c}} w_c(i_{1:n_c}) \prod_{k=1}^{n_c} \left[\mathbb{I}[i_k{=}0]\, \frac{\xi_{KT}([x^c_{1:n}]_{1:k})}{\xi_{KT}([x^c_{1:n}]_{<k})} + \mathbb{I}[i_k{=}1] \frac{\textsc{cts}^{0c}_{D-1}(x_{1:t_c(k)})}{\textsc{cts}^{0c}_{D-1}(x_{<t_c(k)})} \frac{\textsc{cts}^{1c}_{D-1}(x_{1:t_c(k)})}{\textsc{cts}^{1c}_{D-1}(x_{<t_c(k)})} \right] \tag{16}$$

for $c \in \mathcal{K}_D$, where $n_c := l(x^c_{1:n})$ and $t_c(k)$ is the smallest integer such that $l(x^c_{1:t_c(k)}) = k$. In the base cases we have $\text{CTS}^c_0(x_{1:n}) := \xi_{KT}(x^c_{1:n})$ and $\text{CTS}^c_D(\epsilon) := 1$ for any $D \in \mathbb{N}$, $c \in \mathcal{K}_D$.

We now specify the CTS algorithm, which involves describing how to maintain Equation 16 efficiently at each internal node of the context tree data structure, as well as how to select an appropriate sequence of switching rates (which defines $w_c(i_{1:n_c})$) for each context. Also, from now onwards, we will use $\text{CTS}_D(x_{1:n})$ to denote the top-level mixture $\text{CTS}^\epsilon_D(x_{1:n})$.

3.1 Algorithm

CTS repeatedly applies Algorithm 1 to efficiently maintain Equation 16 at each distinct context. This requires maintaining a context tree, where each node representing context c contains six entries: $\xi_{KT}(x^c_{1:n})$ and associated a_c, b_c counts, $\text{CTS}^c_D(x_{1:n})$ and two weight terms k_c and s_c which we define later. Initially the context tree data structure is empty. Now, given a new symbol x_n, having previously seen the data sequence $x_{<n}$, the context tree is traversed from root to leaf by following the path defined by the current context $\phi_D(x_{<n}) := x_{n-1}x_{n-2}\ldots x_{n-D}$. If, during this process, a prefix c of $\phi_D(x_{<n})$ is found to not have a node representing it within the context tree, a new node is created with $k_c := 1/2$, $s_c := 1/2$, $a_c = 0$, and $b_c = 0$. Next, the symbol x_n is processed, by applying in order, for all nodes corresponding to contexts $c \in \{\phi_D(x_{1:n}), \ldots, \phi_1(x_{1:n}), \epsilon\}$, the following update equations

$$\begin{aligned}
\text{CTS}^c_D(x_{1:n}) &\leftarrow k_c\,\xi_{KT}(x^c_n|x^c_{<n}) + s_c\,z^c_D(x_n|x_{<n}) \\
k_c &\leftarrow \alpha^c_{n+1}\,\text{CTS}^c_D(x_{1:n}) + (1-2\alpha^c_{n+1})\,k_c\,\xi_{KT}(x^c_n \mid x^c_{<n}) \\
s_c &\leftarrow \alpha^c_{n+1}\,\text{CTS}^c_D(x_{1:n}) + (1-2\alpha^c_{n+1})\,s_c\,z^c_D(x_n \mid x_{<n}),
\end{aligned}$$

for $D > 0$, where $\xi_{KT}(x^c_n|x^c_{<n}) := \xi_{KT}(x^c_{1:n})/\xi_{KT}(x^c_{<n})$ and

$$z^c_D(x_n|x_{<n}) := \left[\text{CTS}^{0c}_{D-1}(x_{1:n})/\text{CTS}^{0c}_{D-1}(x_{<n})\right]\left[\text{CTS}^{1c}_{D-1}(x_{1:n})/\text{CTS}^{1c}_{D-1}(x_{<n})\right],$$

proceeding from the leaf node back to the root. In the base case we have $\text{CTS}^c_0(x_{1:n}) := \xi_{KT}(x^c_{1:n})$. In addition, for each relevant context, $\xi_{KT}(x^c_{1:n})$ is updated by applying Equation 7 and incrementing either a_c or b_c by 1. As CTS is identical to CTW except for its constant time recursive updating scheme, its asymptotic time and space complexity is the same as for CTW.

Setting the Switching Rate The only part of Equation 16 we have not yet specified is how to set the switching rate α^c_n. With Theorem 1 in mind, our first thought was to use $\alpha^c_n = n_c^{-1}$. However this choice gave poor empirical performance. Furthermore, with this choice we were unable to find a redundancy bound competitive with Equation 15. Instead, a much better alternative was to set $\alpha^c_n = n^{-1}$ for *any* sub-context. The next result justifies this choice.

Theorem 3. *For all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{X}^n$, for all $D \in \mathbb{N}$, we have*

$$-\log_2 \text{CTS}_D(x_{1:n}) < \Gamma_D(\mathcal{S}) + [d(\mathcal{S})+1]\log_2 n + |\mathcal{S}|\gamma(\tfrac{n}{|\mathcal{S}|}) - \log_2 \Pr(x_{1:n} \mid \mathcal{S}, \Theta_\mathcal{S}), \qquad (17)$$

for any pair $(\mathcal{S}, \Theta)$ where $\mathcal{S} \in \mathcal{C}_D$ and $\Theta_\mathcal{S} := \{\theta_s : \theta_s \in [0,1]\}_{s\in\mathcal{S}}$.

Proof. See Appendix C. □

This is a very strong result, since it holds for all binary PST models of maximum depth D, and all possible data sequences, without making any assumptions (probabilistic or otherwise) on how the data is generated. Additionally, Theorem 3 lets us state a redundancy bound for CTS when it is combined with an arithmetic encoder to compress data generated by a binary, n-Markov, stationary source.

Corollary 1. *For all $n \in \mathbb{N}$, given a data sequence $x_{1:n} \in \mathcal{X}^n$ generated by a binary PST source $(\mathcal{S}, \Theta_\mathcal{S})$ with $\mathcal{S} \in \mathcal{C}_D$ and $\Theta_\mathcal{S} := \{\theta_s : \theta_s \in [0,1]\}_{s\in\mathcal{S}}$, the redundancy of CTS using a context depth $D \in \mathbb{N}$ is upper bounded by $\Gamma_D(\mathcal{S}) + [d(\mathcal{S})+1]\log_2 n + |\mathcal{S}|\gamma(\frac{n}{|\mathcal{S}|}) + 2$.*

	bib	book1	book2	geo	news	obj1	obj2	paper1	paper2	paper3	paper4	paper5	paper6	pic	progc	progl	progp	trans
CTW_{48}	2.25	**2.31**	2.12	**5.01**	2.78	**4.63**	3.19	2.84	2.59	2.97	3.50	3.73	2.99	**0.90**	3.00	2.11	2.24	2.09
CTS_{48}	**2.23**	2.32	**2.10**	5.05	**2.77**	4.70	**3.16**	**2.78**	**2.56**	**2.95**	**3.48**	**3.70**	**2.93**	0.91	**2.94**	**2.05**	**2.12**	**1.95**
CTW^*_{48}	1.83	**2.18**	**1.89**	4.53	2.35	3.72	2.40	2.29	2.23	2.5	2.82	2.93	2.37	0.80	2.33	1.65	1.68	1.44
CTS^*_{48}	**1.79**	2.19	**1.89**	**4.18**	**2.33**	**3.65**	**2.33**	**2.27**	**2.22**	**2.48**	**2.78**	**2.90**	**2.36**	**0.77**	**2.32**	**1.59**	**1.62**	**1.37**
CTS^*_{160}	1.77	2.18	1.86	4.17	2.31	3.64	2.30	2.26	2.21	2.48	2.78	2.90	2.35	0.77	2.30	1.54	1.56	1.31

Table 1: Performance (average bits per byte) of CTW and CTS with a fixed D on the Calgary Corpus

Comparing the redundancy bounds in Equation 17 with Equation 15, we see that CTS bound is slightly looser, by an additive $[d(\mathcal{S}) + 1] \log_2 n$ term. However this is offset by the fact that CTS weights over a much larger class than CTW. If the underlying data isn't generated by a single binary PST source, it seems reasonable to suspect that CTS may perform better than CTW. Notice too that as n gets large, both methods have $O(\log_2 n)$ redundancy behavior for stationary, D-Markov sources.

4 Experimental Results

We now investigate the performance of Context Tree Switching empirically. For this we measured the performance of CTS on the well known Calgary Corpus - a collection of files widely used to evaluate compression algorithms. The results (in average bits per byte) are shown in Table 1.

The results for CTW_{48}, CTS_{48}, CTS^*_{48} and CTS^*_{160} were generated from our own implementation[1], which used a standard binary arithmetic encoder to produce the compressed files. The CTW_{48}, CTS_{48} methods refer to the base CTW and CTS algorithms using a context depth of D=48 (6 bytes). Both methods used the KT estimator at leaf nodes, and contained no other enhancements. CTS^*_{48} and CTS^*_{160} referred to our enhanced versions of CTS. These used the binary decomposition method from [Willems and Tjalkens, 1997] and a technique similar to count halving, which multiplied a_c and b_c by a factor of 0.98 during every update. Additionally, s_c and k_c were initialized to 0.925 and 0.075 respectively for each $c \in \mathcal{K}_D$ upon node creation. The remaining CTW^*_{48} results are from a state-of-the-art CTW implementation made public by algorithm's original creators [Willems, 2011]. This version features important enhancements such as replacing the KT estimator with the Zero-Redundancy estimator, binary decomposition for byte oriented data, weighting only at byte boundaries and count halving [Willems and Tjalkens, 1997]. Various combinations of these CTW enhancements were also tried with CTS, but were found to be slightly inferior to the CTS^* method described above.

PPM^*	CTW	PPMZ	CTS^*	DEPLUMP
2.09	1.99	1.93	1.93	**1.89**

Table 2: Weighted (by filesize) Average Bits per Byte on the Calgary Corpus

The first two rows in Table 1 compare the performance of the base CTW and CTS algorithms. Here we see that CTS generally outperforms CTW, in some cases producing files that are 7% smaller. In the cases where it is worse, it is only by a margin of 1%. The third and fourth rows compare the performance of the enhanced versions of CTW and CTS. Again we see similar results, with CTS performing better by up to 8%; in the single case where it is worse, the margin is less than 1%. Finally, Table 2 shows the performance of CTS (using D=160) relative to the results reported in [Gasthaus et al., 2010]. Here we see that CTS's performance is excellent, comparable with modern PPM techniques such as PPMZ [Bloom, 1998] and only slightly inferior to the recent DEPLUMP algorithm.

[1] Available at: `http://jveness.info/software/cts-v1.zip`

5 Conclusion

This paper has introduced Context Tree Switching, a universal algorithm for the compression of binary, stationary, n-Markov sources. Experimental results show that the technique gives a small but consistent improvement over regular Context Tree Weighting, without sacrificing its theoretical guarantees. We feel our work is interesting since it demonstrates how a well-founded data compression algorithm can be constructed from switching. Importantly, this let us narrow the performance gap between methods with strong theoretical guarantees and those that work well in practice.

A natural next step would be investigate whether CTS can be extended for binary, k-Markov, *piecewise stationary* sources. This seems possible with some simple modifications to the base algorithm. For example, the KT estimator could be replaced with a technique that works for unknown, memoryless, piecewise stationary sources, such as those discussed by Willems [1996], Willems and Krom [1997]. Theoretically characterizing the redundancy behavior of these combinations, or attempting to derive a practical algorithm with provable redundancy behavior for k-Markov, piecewise stationary sources seems an exciting area for future research.

References

Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

C. Bloom. Solving the problem of context modelling". `http://www.cbloom.com/papers/ppmz.pdf`, 1998.

John G. Cleary, Ian, and Ian H. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications*, 32:396–402, 1984.

J. Gasthaus, F. Wood, and Y. W. Teh. Lossless compression based on the sequence memoizer. In *Data Compression Conference*, 2010.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, August 1998.

R. Krichevsky and V. Trofimov. The performance of universal encoding. *Information Theory, IEEE Transactions on*, 27(2):199–207, 1981.

J. Rissanen. Universal coding, information, prediction, and estimation. *Information Theory, IEEE Transactions on*, 30(4):629 – 636, jul 1984. ISSN 0018-9448. doi: 10.1109/TIT.1984.1056936.

Tim van Erven, Peter Grünwald, and Steven de Rooij. Catching Up Faster in Bayesian Model Selection and Model Averaging. *Neural Information Processing Systems (NIPS)*, 2007.

P. Volf. *Weighting techniques in data compression: Theory and algorithms*. PhD thesis, Eindhoven University of Technology, 2002.

Paul A. J. Volf and Frans M. J. Willems. Switching between two universal source coding algorithms. In *In Data Compression Conference*, pages 491–500, 1998.

F. Willems and M. Krom. Live-and-die coding for binary piecewise i.i.d. sources. In *Information Theory. 1997. Proceedings., 1997 IEEE International Symposium on*, page 68, jun-4 jul 1997. doi: 10.1109/ISIT.1997.612983.

F. Willems and T.J. Tjalkens. Complexity Reduction of the Context-Tree Weighting Algorithm: A Study for KPN Research. *Tech. Rep. EIDMA Report RS.97.01*, 1997.

F. M. J. Willems. CTW website. `http://www.ele.tue.nl/ctw/`, 2011.

Frans Willems, Yuri Shtarkov, and Tjalling Tjalkens. Reflections on "The Context Tree Weighting Method: Basic properties". *Newsletter of the IEEE Information Theory Society*, 47(1), 1997.

Frans M. J. Willems. Coding for a binary independent piecewise-identically-distributed source. *IEEE Transactions on Information Theory*, 42:2210–2217, 1996.

Frans M.J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The Context Tree Weighting Method: Basic Properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.

Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30: 520–540, June 1987. ISSN 0001-0782.

Appendix A. Correctness of Algorithm 1

This section proves the correctness of Algorithm 1. We begin by first proving a lemma.

Lemma 2. *If $w_{j,t}$ denotes the weight w_j at the beginning of iteration t in Algorithm 1, the identity*

$$w_{j,t} = \sum_{i_{<t}} w(i_{<t}j) \prod_{k=1}^{t-1} \rho_{i_k}(x_k \mid x_{<k}),$$

holds for all $t \in \mathbb{N}$.

Proof. We use induction on t. In the base case, we have

$$w_{j,1} = \sum_{i_{<1}} w(i_{<t}j) = w(\epsilon j) = w(j) = \tfrac{1}{N},$$

which is what is required. Letting r_t denote the value assigned to r on iteration t, for the inductive case we have

$$\begin{aligned}
w_{j,t+1} =& \tfrac{1}{N-1}[\alpha_{t+1} r_t + (N(1-\alpha_{t+1}) - 1) w_{j,t}\, \rho_j(x_t \mid x_{<t})] \\
=& \tfrac{\alpha_{t+1}}{N-1} \sum_{j=1}^{N} \left[\sum_{i_{<t}} w(i_{<t}j) \prod_{k=1}^{t-1} \rho_{i_k}(x_k \mid x_{<k}) \right] p_j(x_t \mid x_{<t}) + \\
& \tfrac{N(1-\alpha_{t+1})-1}{N-1} \left[\sum_{i_{<t}} w(i_{<t}j) \prod_{k=1}^{t-1} \rho_{i_k}(x_k \mid x_{<k}) \right] \rho_j(x_t \mid x_{<t}) \\
=& \tfrac{\alpha_{t+1}}{N-1} \sum_{i_{1:t}} w(i_{1:t}) \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}) + \tfrac{N(1-\alpha_{t+1})-1}{N-1} \sum_{i_{<t}} w(i_{<t}j) \left[\prod_{k=1}^{t-1} \rho_{i_k}(x_k \mid x_{<k}) \right] \rho_j(x_t \mid x_{<t}) \\
=& \sum_{i_{1:t} \mid i_t \neq j} w(i_{1:t}) \tfrac{\alpha_{t+1}}{N-1} \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}) + \sum_{i_{1:t} \mid i_t = j} w(i_{<t}j) \tfrac{\alpha_{t+1}}{N-1} \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}) + \\
& \sum_{i_{1:t} \mid i_t = j} w(i_{<t}j) \tfrac{N(1-\alpha_{t+1})-1}{N-1} \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}) \\
=& \sum_{i_{1:t} \mid i_t \neq j} w(i_{1:t}) \tfrac{\alpha_{t+1}}{N-1} \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}) + \sum_{i_{1:t} \mid i_t = j} w(i_{<t}j)(1-\alpha_{t+1}) \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}) \\
=& \sum_{i_{1:t}} w(i_{1:t}j) \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}).
\end{aligned}$$

□

Theorem 4. *$\forall n \in \mathbb{N}, \forall x_{1:n} \in \mathcal{X}^n$, Algorithm 1 computes $\tau_\alpha(x_{1:n})$.*

Proof. Letting $w_{j,t}$ denote the weight w_j at the beginning of iteration t, Algorithm 1 returns

$$\begin{aligned}
\sum_{j=1}^{N} w_{j,t} \rho_j(x_t \mid x_{<t}) &= \sum_{j=1}^{N} \sum_{i_{<t}} w(i_{<t}j) \left[\prod_{k=1}^{t-1} \rho_{i_k}(x_k \mid x_{<k}) \right] \rho_j(x_t \mid x_{<t}) \\
&= \sum_{i_{1:t}} w(i_{1:t}) \prod_{k=1}^{t} \rho_{i_k}(x_k \mid x_{<k}) \\
&= \tau_\alpha(x_{1:t}),
\end{aligned}$$

where the first equality follows from Lemma 2. □

Appendix B. Proof of Lemma 1

Lemma 1. *Given a base model class $\mathcal{M}$ and a decaying switch rate $\alpha_t := \frac{1}{t}$ for $t \in \mathbb{N}$,*

$$-\log_2 w(i_{1:n}) \leq (m(i_{1:n}) + 1)\left(\log_2 |\mathcal{M}| + \log_2 n\right),$$

for all $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$, where $m(i_{1:n}) := \sum_{k=2}^{n} \mathbb{I}[i_k \neq i_{k-1}]$ denotes the number of switches in $i_{1:n}$.

Proof. Consider an arbitary $i_{1:n} \in \mathcal{I}_n(\mathcal{M})$. Now, letting m denote $m(i_{1:n})$, we have

$$\begin{aligned}
-\log_2 w(i_{1:n}) &= \log_2 |\mathcal{M}| - \log_2 \prod_{t=2}^{n} \tfrac{\alpha_t}{|\mathcal{M}|-1}\mathbb{I}[i_t \neq i_{t-1}] + (1-\alpha_t)\mathbb{I}[i_t = i_{t-1}] \\
&\leq \log_2 |\mathcal{M}| - \log_2 \prod_{t=2}^{n} \tfrac{1}{n(|\mathcal{M}|-1)}\mathbb{I}[i_t \neq i_{t-1}] + \tfrac{t-1}{t}\mathbb{I}[i_t = i_{t-1}] \\
&\leq \log_2 |\mathcal{M}| - \log_2 \left(n^{-m}(|\mathcal{M}|-1)^{-m} \prod_{t=2}^{n-m} \tfrac{t-1}{t} \right) \\
&= \log_2 |\mathcal{M}| + m \log_2 n + m \log_2 (|\mathcal{M}| - 1) + \log_2 (n-m) \\
&\leq (m+1)[\log_2 |\mathcal{M}| + \log_2 n].
\end{aligned}$$

□

Appendix C. Proof of Theorem 3

Theorem 3. *For all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{X}^n$, for all $D \in \mathbb{N}$, we have*

$$-\log_2 \textsc{cts}_D(x_{1:n}) \leq \Gamma_D(\mathcal{S}) + [d(\mathcal{S})+1]\log_2 n + |\mathcal{S}|\gamma(\tfrac{n}{|\mathcal{S}|}) - \log_2 \Pr(x_{1:n} \,|\, \mathcal{S}, \Theta_{\mathcal{S}}),$$

for any pair $(\mathcal{S}, \Theta)$ where $\mathcal{S} \in \mathcal{C}_D$ and $\Theta_{\mathcal{S}} := \{\theta_s \,:\, \theta_s \in [0,1]\}_{s\in\mathcal{S}}$.

Proof. Consider an arbitrary $\mathcal{S} \in \mathcal{C}_D$ and $\Theta_{\mathcal{S}} = \{\theta_s \,:\, \theta_s \in [0,1]\}_{s\in\mathcal{S}}$. Now define $\tilde{\mathcal{S}} \subset \mathcal{K}_D$ as the set of contexts that index the internal nodes of PST structure $\mathcal{S}$. Observe that, for all $n \in \mathbb{N}$ and for all $x_{1:n} \in \mathcal{X}^n$, by dropping the sum in Equation 16 we can conclude

$$\textsc{cts}^c_D(x_{1:n}) \geq \begin{cases} w_c(1_{1:n_c})\,\textsc{cts}^{0c}_{D-1}(x_{1:n})\,\textsc{cts}^{1c}_{D-1}(x_{1:n}) & \text{if } c \notin \mathcal{S} \\ w_c(0_{1:n_c})\,\xi_{KT}(x^c_{1:n}) & \text{if } c \in \mathcal{S} \text{ and } D > 0 \\ \xi_{KT}(x^c_{1:n}) & \text{if } D = 0, \end{cases} \tag{18}$$

for any sub-context $c \in \mathcal{S} \cup \tilde{\mathcal{S}}$. Next define $\mathcal{S}' := \{s \in \mathcal{S} \,:\, l(s) < D\}$. Now, by repeatedly applying Equation 18, starting with $\textsc{cts}_D(x_{1:n})$ (which recall is defined as $\textsc{cts}^\epsilon_D(x_{1:n})$) and continuing until no more $\textsc{cts}(\cdot)$ terms remain, we can conclude

$$\begin{aligned}
\textsc{cts}_D(x_{1:n}) &\geq \left(\prod_{c\in\tilde{\mathcal{S}}} w_c(1_{1:n_c})\right)\left(\prod_{s\in\mathcal{S}'} w_s(0_{1:n_s})\right)\left(\prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n})\right) \\
&= \left(\prod_{k=0}^{d(\mathcal{S})} \prod_{c\in\mathcal{S}'\cup\tilde{\mathcal{S}}\,:\,l(c)=k} w_c(1_{1:n_c})\right)\left(\prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n})\right) \\
&\geq \left(2^{-\Gamma_D(\mathcal{S})} \prod_{k=0}^{d(\mathcal{S})} \prod_{c\in\mathcal{S}'\cup\tilde{\mathcal{S}}\,:\,l(c)=k} \frac{w_c(1_{1:n_c})}{w_c(1_{1:\min(n_c,1)})}\right)\left(\prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n})\right) \\
&\geq \left(2^{-\Gamma_D(\mathcal{S})} \prod_{k=0}^{d(\mathcal{S})} \prod_{t=2}^{n} \frac{t-1}{t}\right)\left(\prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n})\right) \\
&= 2^{-\Gamma_D(\mathcal{S})} n^{-(d(\mathcal{S})+1)} \left(\prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n})\right).
\end{aligned}$$

The first equality follows by noting that Definition 1 implies $w_c(0_{1:t}) = w_c(1_{1:t})$ for all $t \in \mathbb{N}$ and rearranging. The second inequality follows from $|\mathcal{S}' \cup \tilde{\mathcal{S}}| = \Gamma_D(\mathcal{S})$, $w_c(1) = \frac{1}{2}$ and that either $w_c(1_{1:n_c}) = w_c(\epsilon) = 1$ if $n_c = 0$ or $w_c(1_{1:n_c}) = \frac{1}{2} \times \ldots$ for $n_c > 0$. The last inequality follows from the observation that the context associated with each symbol in $x_{1:n}$ matches at most one context $c \in \mathcal{S}' \cup \tilde{\mathcal{S}}$ of each specific length $0 \leq k \leq d(\mathcal{S})$. The final equality follows upon simplification of the telescoping product. Hence,

$$-\log_2 \textsc{cts}(x_{1:n}) \leq \Gamma_D(\mathcal{S}) + [d(\mathcal{S})+1]\log_2 n - \log_2 \left(\prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n})\right). \tag{19}$$

Finally, the proof is completed by noting that Equation 11 implies

$$-\log_2 \left(\prod_{s\in\mathcal{S}} \xi_{KT}(x^s_{1:n})\right) \leq |\mathcal{S}|\gamma(\tfrac{n}{|\mathcal{S}|}) - \log_2 \Pr(x_{1:n} \,|\, \mathcal{S}, \Theta_{\mathcal{S}}),$$

and then combining the above with Equation 19. □