

Context Tree Weighting

Peter Sunehag and **Marcus Hutter**



2013

Motivation

- ▶ Context Tree Weighting (CTW) is a Bayesian Mixture of the huge class of variable-order Markov processes.
- ▶ It is principled *and* computationally efficient.
- ▶ It leads to excellent predictions and compression in practice and in theory.
- ▶ It can (and will) be used to approximate Solomonoff's universal prior $\xi_U(x)$.

Prediction of I.I.D Sequences

- ▶ Suppose that we have a sequence which we believe to be I.I.D. but we do not know the probabilities.
- ▶ If x has been observed n_x times, then we can use the (generalized Laplace rule, Dirichlet(α) prior) estimate $Pr(x_{n+1} = x | x_{1:n}) = \frac{n_x + \alpha}{n + M\alpha}$, where M is the size of the alphabet and $\alpha > 0$ is a smoothing constant.
- ▶ We use the special case of **binary alphabet** and $\alpha = 1/2$ (Jeffrey's=Beta(1/2) prior, \approx minimax optimal).
- ▶ The probability of a 0 if we have previously seen a zeros and b ones hence is $\frac{a+1/2}{a+b+1}$ and the probability of a 1 is $\frac{b+1/2}{a+b+1}$

Joint Prob. for I.I.D with Beta(1/2) Prior

- ▶ The joint prob. of sequence is product of individual probabilities independent of order:

$$Pr(x_1, \dots, x_n) = Pr(x_{\pi(1)}, \dots, x_{\pi(n)}) \quad \forall \text{ permutations } \pi.$$

- ▶ We denote the probability of a zeros and b ones with $P_{kt}(a, b)$

$$\text{▶ } P_{kt}(a+1, b) = P_{kt}(a, b) \frac{a+1/2}{a+b+1}, \quad P_{kt}(0, 0) = 1.$$

$$\text{▶ } P_{kt}(a, b+1) = P_{kt}(a, b) \frac{b+1/2}{a+b+1}, \quad Pr(x_{1:n}) = P_{kt}(a, b).$$

$a \backslash b$	0	1	2	3	4	...
0	1	1/2	3/8	5/16	35/128	...
1	1/2	1/8	1/16	5/128	7/256	...
2	3/8	1/16	3/128	3/256	7/1024	...
3	5/16	5/128	3/256	5/1024	5/2048	...

$$\text{▶ Example: } Pr_{kt}(0011) = \frac{1/2}{1} \cdot \frac{1+1/2}{2} \cdot \frac{1/2}{3} \cdot \frac{1+1/2}{4} = Pr_{kt}(0101) = Pr_{kt}(1001) = \dots = Pr_{kt}(2, 2) = \frac{3}{128}$$

$$\text{▶ Direct: } Pr(x_{1:n}) = \prod_{t=1}^n Pr(x_t | x_{<t}) = P_{kt}(a, b) = \int_{\theta} Pr_{\theta}(x) \text{Beta}_{1/2}(\theta) d\theta = \frac{1}{\pi} \int \theta^{a-1/2} (1-\theta)^{b-1/2} d\theta = \frac{(a-1/2)!(b-1/2)!}{(a+b)! \pi}$$

Using Context

- ▶ If you hear a word that you believe is either "cup" or "cop", then you can use context to decide which one it was.
- ▶ If you are sure they said "Wash the" just before, then it is probably "cup".
- ▶ If they said "Run from the", it might be "cop".
- ▶ We will today look at models, which have a short term memory dependence.
- ▶ In other words models that remember the last few observations.

Markov Models

- ▶ Suppose that we have a sequence x_t of elements from some finite alphabet
- ▶ Suppose that $Pr(x_t | x_{1:t-1}) = Pr(x_t|x_{t-1})$ is always true, then we have a **Markov Model**. At most the latest observation matters
- ▶ If $Pr(x_t | x_{1:t-1}) = Pr(x_t|x_{t-k:t-1})$, then we have a **k:th order Markov Model**. At most the k latest observations matter
- ▶ Given that we have chosen k and a (several) sequence(s) of observations, then for every context (string of length k) we can estimate probabilities for all possible next observations using the KT-estimator.
- ▶ We define the probability of the event of having a 0 after s given that we have, in this context, previously seen a_s zeros and b_s ones to be $\frac{a_s+1/2}{a_s+b_s+1}$ and the probability of a one to be $\frac{b_s+1/2}{a_s+b_s+1}$
- ▶ **Formally:** $[x]_s := (x_t : x_{t-k:t-1} = s) \equiv$ all those x_t with k -context $x_{t-k:t-1} = s$. $a_s = \#\{0 \text{ in } [x]_s\}$, $b_s = \#\{1 \text{ in } [x]_s\}$.

Markov-KT Examples

- ▶ Compute $Pr_k(x) = (\frac{1}{2})^k \prod_{s \in \{0,1\}^k} P_{kt}(a_s, b_s)$
for Example $x = 0010100111101010$.

- ▶ $k = 0 : a = b = 8$.

$$Pr_0(0010100111101010) = P_{kt}(8, 8) \approx 402 \times 2^{-27}$$

- ▶ $k = 1 : a_0 = 2, b_0 = 5, a_1 = 5, b_1 = 3$.

$$Pr_1(0 \ 010100111101010) = \frac{1}{2} P_{kt}(2, 5) P_{kt}(5, 3) = \frac{1}{2} \cdot \frac{9}{2048} \cdot \frac{45}{32768} = 405 \cdot 2^{-27}$$

- ▶ $k = 2 :$

s	00	01	10	11
a_s	0	4	1	1
b_s	2	1	3	2
$[x]_s$	11	00100	1011	110

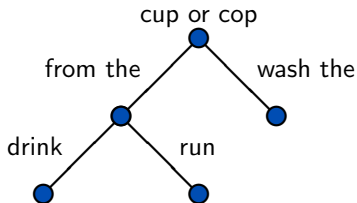
$$Pr_2(00 \ 10100111101010) = (\frac{1}{2})^2 P_{kt}(0, 2) P_{kt}(4, 1) P_{kt}(1, 3) P_{kt}(1, 2) = \frac{1}{4} \cdot \frac{3}{8} \cdot \frac{7}{256} \cdot \frac{5}{16} \cdot \frac{1}{16} = 105 \cdot 64 \cdot 2^{-27}$$

Choosing k

- ▶ If we decide to model our sequence with a k :th order Markov model we then need to pick k .
- ▶ How long dependencies exist?
- ▶ Which contexts have we seen enough to make good predictions from? Shorter contexts appear more often.
- ▶ Solution 1: Use MDL to select k .
- ▶ Solution 2: We can take the **Bayesian approach** and have a mixture over all orders.
- ▶ We can choose a prior (initial mixture weights) that **favors shorter orders (simpler models)**, e.g. $P(k) = \frac{1}{k(k+1)}$.

Context Trees

- ▶ It can be natural to consider **contexts of different lengths** in a model, depending on the situation.
- ▶ For example if we have heard **"from the"** and want to decide if the next word is **"cup"** or **"cop"** it is useful to also know if it is **"drink from the"** or **"run from the"** while if we have the context **"wash the"** it might be enough.
- ▶ Having **redundant parameters** lead to a need for more data to find good parameter estimates. With small amount of data for a context, it is better to be shallow.



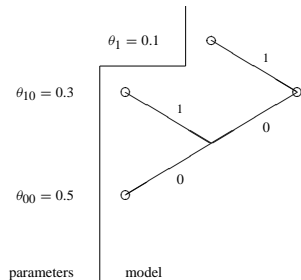
Tree Source

Example: Tree $\mathcal{T} \hat{=} \{00, 10, 1\}$

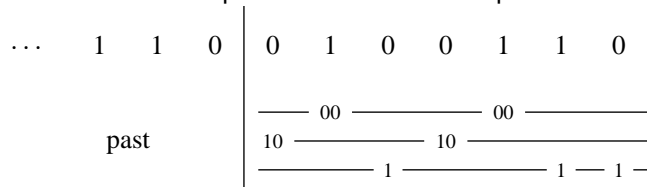
$$P(x_t = 1 | \dots x_{t-1} = 1) = \theta_1$$

$$P(x_t = 1 | \dots x_{t-2} = 0, x_{t-1} = 0) = \theta_{00}$$

$$P(x_t = 1 | \dots x_{t-2} = 1, x_{t-1} = 0) = \theta_{10}$$



Contexts \mathcal{T} in sequence 01001100 with past ...110:



$P_{\mathcal{T}}(0100110 | \dots 110) = \frac{3}{8} \cdot \frac{3}{8} \cdot \frac{1}{16}$, where $\frac{3}{8}$, $\frac{3}{8}$, $\frac{1}{16}$ are the P_{kt} probabilities of subsequences 11, 00, 010 corresponding to leaves/contexts 00, 10, 1, respectively.

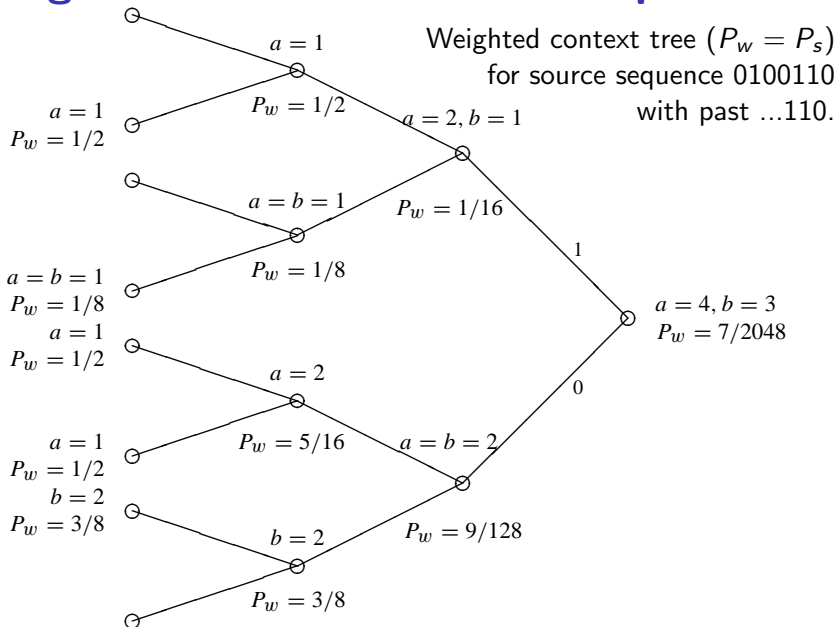
Context Tree Weighting (CTW)

- ▶ Given a tree we have defined contexts such that we are always exactly in one context. Given data we estimate probabilities for the next observation given the context.
- ▶ We do not know the most appropriate tree. So we have to estimate it or Bayes-average over all trees.
- ▶ The number of trees (and weights) increases double exponentially with the depth.
- ▶ The CTW algorithm resolves this problem.
- ▶ It stores two numbers (for the binary case) per node in the full tree of the given maximal depth and calculates a third through a recursive formula.
- ▶ The stored numbers are just counts and are easy to update.

Context Tree Weighting

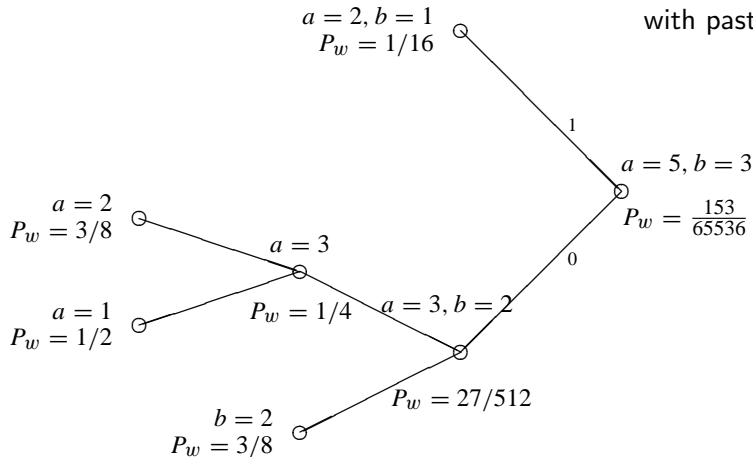
- ▶ To define a probability of a finite sequence, CTW uses a recursive formula that starts at the leafs of the tree and moves towards the root where the probability of the whole sequence will appear.
- ▶ The next few slides will explain it with an example.
- ▶ Every node in the full tree corresponds to a context s , e.g. 010.
- ▶ We let a_s be the number of times that a 0 has followed s and b_s is the number of times 1 has followed.
- ▶ For every context that corresponds to a node in the full tree we will define a number P_s . $P_{root} \equiv P_\epsilon \equiv P_{CTW}$ will be the probability for the whole sequence.
- ▶ For a leaf we let $P_s = P_{kt}(a_s, b_s)$.
- ▶ Not leaf: $P_s = \frac{1}{2}(P_{kt}(a_s, b_s) + P_{0s}P_{1s})$

Weighted Context Tree – Example



Weighted Context Tree Update—Example

Updated path of weighted context tree
for 0100110 followed by 0
with past ...110.

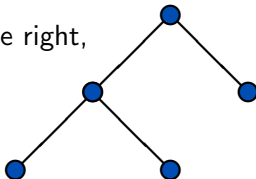


Coding Context Trees

- ▶ We can rewrite the probability P_{CTW} as a Bayesian mixture over trees with a prior based on a coding scheme for trees:
- ▶ Let \mathcal{T} denote a tree and $CL(\mathcal{T})$ the code length for \mathcal{T}
- ▶ Let tree prior $P(\mathcal{T}) = 2^{-CL(\mathcal{T})}$. Then

$$P_{CTW} = \sum_{\mathcal{T}} 2^{-CL(\mathcal{T})} \prod_{s \in \mathcal{T}} P_{kt}(a_s, b_s)$$

- ▶ The coding scheme is based on the following recursion:
 1. Code(tree) = Code(root-node)
 2. Code(Internal-node) = 1 Code(left-child) Code(right-child)
 3. Code(leaf-node) = 0
- ▶ 11000 is the code of the tree on the right, so the code length is 5.



Coding Redundancy

- ▶ Arithmetic Coding w.r.t. P_{CTW} gives code for x of length $CL_{CTW}(x) = \log 1/P_{CTW}(x)$.
- ▶ A coding scheme's redundancy is $CL(x_{1:T}) - \log 1/Pr(x_{1:T})$ where Pr is the true source.
- ▶ We are interested in the expected redundancy
- ▶ The expected redundancy, unlike the actual redundancy, is at least 0. The truth has the lowest (i.e. 0) expected redundancy.
- ▶ The **Rissanen lower bound** says that the expected redundancy is at least $\frac{1}{2} \log T$ per parameter.
- ▶ If the sequence is generated from a (stationary) tree (Markov) source \mathcal{T} with S leaves, then for large enough T the **expected redundancy of CTW is less than $CL(\mathcal{T}) + \frac{S}{2} \log T$** .
- ▶ CTW achieves the Rissanen lower bound
- ▶ This is essentially a corollary of the general **(continuous+discrete) "entropy bounds"** from UAI book (Hutter) / previous lectures.

Derivation of Coding Redundancy

The bound can be derived directly or follows from the generic bounds for Bayesian Sequence Prediction [Hut05]:

- ▶ Inst. Rel. Entropy: $d_t(x_{<t}) = \sum_{x_t \in \mathcal{X}} \mu(x_t | x_{<t}) \log \frac{\mu(x_t | x_{<t})}{\xi(x_t | x_{<t})}$
- ▶ Total Relative Entropy: $D_T = \sum_{t=1}^T \mathbf{E}[d_t]$ is the **expected coding redundancy** from using ξ instead of (the true) μ , i.e. $D_T = \mathbf{E}_\mu[CL_\xi(x_{1:T}) - CL_\mu(x_{1:T})]$
- ▶ Countable family of models: $D_T \leq \log w_\mu^{-1}$
- ▶ Continuous family of models:
 - ▶ $D_T \leq \log w_\mu^{-1} + \frac{S}{2} \log \frac{T}{2\pi} + \frac{1}{2} \log \det j_T + O(1)$
 - ▶ CTW is a continuous mixture with prior $w_\mu = 2^{-CL(\mathcal{T})} g(\theta)$ where \mathcal{T} is the tree, θ represents the probability parameters and $g(\theta)$ is the Beta(1/2, 1/2) prior density on the parameters.
- ▶ $\frac{1}{2} \log \det j_T = O(1)$ for tree sources so $D_T \leq CL(\mathcal{T}) + \frac{S}{2} \log T + O(1)$

Conclusion / Properties

- ▶ CTW is a simple algorithm for mixing contexts of different length to make predictions.
- ▶ The computational complexity for calculating the probability of a sequence is **linear** in the sequence length.
- ▶ CTW has **good practical performance** for many purposes, e.g. **text compression**.
- ▶ Given that the truth is a tree source there are bounds for how much worse CTW is than using the (unknown) true model.
- ▶ When the true model is not known there are limits (a lower bound) to how close we can be to the performance of the true model.
- ▶ CTW asymptotically achieves this **theoretically optimal performance**.

Extensions and Augmentations of CTW

- ▶ **Context Tree Switching** switches between arbitrary distributions from a finite set. This enlarges the base class by allowing switching between different distributions. [VNHB12]
- ▶ **Active/conditional version** for MC-AIXI-CTW. [VNH⁺11]
- ▶ **Non-binary CTW** or one CTW per binarization bit.

Improving the underlying KT Estimator

- ▶ **Adaptive KT** discounts old observations to deal with non-stationary smoothly changing environments. [OHSS12]
- ▶ **Window KT** uses limited past window to deal with piecewise i.i.d. environments. [SSH12]
- ▶ **Replace KT by better iid estimators** with e.g. finite redundancy for unused symbols (e.g. KT0 and SAD). [Hut13]

Literature

- (WST95) Willems, Shtarkov, and Tjalkens (1995) The Context-Tree Weighting Method: Basic Properties, 41. IEEE Transactions on Information Theory.
- (WST97) Willems, F.M.J., Shtarkov, Y.M. and Tjalkens, T.J. (1997). Reflections on the prize paper 'The context-tree weighting method: Basic properties'. Newsletters of the IEEE.
- (VNH+11) J. Veness, K. S. Ng, M. Hutter, W. Uther, and D. Silver. A Monte Carlo AIXI approximation, Journal of Artificial Intelligence Research, 40:95–142, 2011.
- (Hut05) M. Hutter. Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin, 2005.