Deep Tree Models for ‘Big’ Biological Data

(Invited Paper)

Lambros Mertzanis
Department of Informatics
Athens University of Economics & Business, Greece
la.mertzanis@gmail.com

Athina Panotopoulou
Department of Computer Science
Dartmouth College, USA
athlon@cs.dartmouth.edu

Maria Skoularidou
MRC-Biostatistics Unit
University of Cambridge, UK
maria@mrc-bsu.cam.ac.uk

Ioannis Kontoyiannis
Department of Engineering
University of Cambridge, UK
i.kontoyiannis@eng.cam.ac.uk

Abstract—The identification of useful temporal dependence structure in discrete time series data is an important component of algorithms applied to many tasks in statistical inference and machine learning, and used in a wide variety of problems across the spectrum of biological studies. Most of the early statistical approaches were ineffective in practice, because the amount of data required for reliable modelling grew exponentially with memory length. On the other hand, many of the more modern methodological approaches that make use of more flexible and parsimonious models result in algorithms that do not scale well and are computationally ineffective for larger data sets. In this paper we describe a class of novel methodological tools for effective Bayesian inference for general discrete time series, motivated primarily by questions regarding data originating from studies in genetics and neuroscience.

Our starting point is the development of a rich class of Bayesian hierarchical models for variable-memory Markov chains. The particular prior structure we adopt makes it possible to design effective, linear-time algorithms that can compute most of the important features of the relevant posterior and predictive distributions without resorting to Markov chain Monte Carlo simulation. The origin of some of these algorithms can be traced to the family of Context Tree Weighting (CTW) algorithms developed for data compression since the mid-1990s. We have used the resulting methodological tools in numerous application-specific tasks (including prediction, segmentation, classification, anomaly detection, entropy estimation, and causality testing) on data from different areas of application. The results obtained compare quite favourably with those obtained using earlier approaches, such as Probabilistic Suffix Trees (PST), Variable-Length Markov Chains (VLMC), and the class of Markov Transition Distributions (MTD).

I. INTRODUCTION

The starting point of many biological studies is the examination and analysis of discrete sequence data; and an important first step is often the identification of patterns of statistical structure in the data. Perhaps the most well studied examples of such problems arise from questions in bioinformatics, statistical genetics, and neuroscience.

Identifying and quantifying dependence is not only crucial in performing application-specific tasks (such as segmentation, compression, classification, and so on), but it is also of great intrinsic interest, as it reveals important features of the underlying biological data-generating mechanisms. It is in part for this purpose that the need for higher-memory Markov models in a variety of applications has been noted many times in the past; see, e.g., [18, 16, 5]. But higher-order Markov models are typically impossible to estimate and use in practice, as the number of parameters involved grows exponentially with memory length. In order to overcome this obstacle, numerous approaches have been developed, using more effective, lower-dimensional model classes; see, e.g., the broad discussions in [4, 2, 10, 7].

The class of variable-memory Markov chain models we consider here were first introduced in Rissanen’s celebrated work [22, 19], and have also been employed (in some cases with minor variations) in the Probabilistic Suffix Tree (PST) [23, 1, 7] and Variable-Length Markov Chain (VLMC) [4, 3, 17] literature. A different class of parsimonious models for higher-order memory modelling, the Markov Transition Distribution (MTD) model, was introduced by Raftery in 1985 [18] and explored further in [2].

The main contribution of this work is the development of a Bayesian framework for inference within the class of variable-memory Markov models. The main result we present is an algorithm which can be employed to determine exactly the maximum a posteriori probability (MAP) model. In fact, the $k$-MAPT algorithm allows us to find not only the model with the highest posterior probability, but also the $k \geq 1$ a posteriori most likely models. The complexity of the algorithm is only linear in the sample size, but grows rapidly with $k$.

The prior structure we use is motivated, in part, by considerations related to Rissanen’s Minimum Description Length (MDL) principle [21]. The starting point of our development is the family of Context Tree Weighting (CTW) [29] and Context Tree Maximizing (CTM) [30, 31, 27] algorithms.

Since its introduction and application to data compression, the CTW algorithm and its many variants have also been applied to numerous different statistical tasks, including prediction [32], segmentation [11], reinforcement learning [25], network traffic analysis [12], turbo decoding [13], spam detection [24], and finance [6]. Biological applications can be found in several of the references cited above, as well as in [8, 9, 15].
In Section II we describe the hierarchical Bayesian model we consider, and Section III contains a description of the model selection algorithm k-MAPT. Finally, in Section IV we present brief experimental results illustrating the algorithm’s performance on simulated data. Our results build on earlier work described in [14]. Theoretical results are stated without proofs; these will be given in a subsequent publication.

II. TREE MODELS AND PRIOR SPECIFICATION

Consider the class of \( d \)th order, homogeneous Markov chains, with values in the alphabet \( A = \{0, 1, \ldots, m - 1\} \). The distribution of such a process \( \{X_n\} \) will be described in terms of the conditional distribution of each \( X_i, i \geq 1 \), given the previous \( d \) symbols \( X_{i-d}^{i-1} = (X_{i-d}, X_{i-d+1}, \ldots, X_i) \), where we write \( X_i^j \) for a vector of random variables \( (X_i, X_{i+1}, \ldots, X_j) \) and similarly \( x_i^j \in A^{j-i+1} \) for a string \( (x_i, x_{i+1}, \ldots, x_j) \) representing a realization of the random variables \( X_i^j \). The key element in specifying these distributions is a context function \( C : A^d \rightarrow T \), which maps each length-\( d \) context \( x_{i-d}^{i-1} \) to a (typically strictly) shorter suffix \( C(x_{i-d}^{i-1}) = x_{i-j}^{i-1} \) of itself, for some \( 0 \leq j \leq d \). Then the Markov property for \( \{X_n\} \) takes the form:

\[
P(x_n|x_{n-d+1}^0) = \prod_{i=1}^{n} P(x_i|x_{i-d}^{i-1} = n) = \prod_{i=1}^{n} P(x_i|C(x_{i-d}^{i-1})).
\]

The range \( T \) of \( C \) is a subset of \( \cup_{i=0}^{d-1} A^i \), where we adopt the convention that the set \( A^0 \) contains only the empty string \( \lambda \).

We assume that the set \( T \) is proper, namely, that no element in \( T \) is a proper suffix of any other, and that if some \( x_i^j \) is in the range of \( C \), then so is every string of the form \( (y, x_{i+1}, \ldots, x_j) \), for all \( y \in A \). Observe that, under these assumptions, the context function \( C \) is completely determined by its range \( T \), since, for any string \( x_{i-d}^{i-1} \) there is exactly one element of \( T \) which is a suffix \( x_{i-j}^{i-1} \) of \( x_{i-d}^{i-1} \).

To complete the specification of the distribution of \( \{X_n\} \), in addition to the context set \( T \), with every element \( s \in T \) we associate a vector \( \theta_s := (\theta_s(0), \theta_s(1), \ldots, \theta_s(m-1)) \), where the \( \theta_s(j) \) are nonnegative and sum to one. Then,

\[
P(x_n|x_{n-d+1}^0) = \prod_{i=1}^{n} \theta_s(C(x_{i-d}^{i-1}), x_i = a_s(j)) = \prod_{s \in T, j \in A} \theta_s(j)^{a_s(j)}, \quad (1)
\]

where, in the last expression, instead of taking the product sequentially in time, we took the product over all possible contexts \( s \in T \), and where the elements of each vector \( a_s = (a_s(0), a_s(1), \ldots, a_s(m-1)) \) are,

\[
a_s(j) = \text{# times } j \in A \text{ follows context } s \text{ in } x_n^0. \quad (2)
\]

We refer to the context set \( T \) as the model of \( \{X_n\} \) and observe that it may be represented as a tree, where the context corresponding to the empty string \( \lambda \) is the root of the tree. We also refer to \( \theta = \{\theta_s : s \in T\} \) as the parameters of \( \{X_n\} \).

Example. Consider a 5th order Markov chain on the alphabet \( A = \{0, 1, 2\} \) defined by the context tree \( T \) shown below, and by a collection of (known) parameters \( \theta = \{\theta_s : s \in T\} \), where \( \theta_s \) is a probability vector corresponding to leaf \( s \) in \( T \).

The likelihood of an arbitrary string is easily computable explicitly via (1). For example, with \( d = 5 \) and \( n = 12 \), the string,

\[
x_1^0 = 1, 0, 2, 1, 1, 0, 0, 1, 2, 0, 0, 2, 0, 1
\]

has probability given by (1) as:

\[
\theta_1(0)^2 \cdot \theta_1(2) \cdot \theta_2(0)^2 \cdot \theta_3(1) \cdot \theta_0(0) \cdot \theta_0(1) \cdot \theta_0(0) \cdot \theta_0(2) \cdot \theta_0(0) \cdot \theta_0(0) \cdot \theta_0(2) = \pi(T) := \pi_D(T) := \pi_D(T; \beta) := \alpha^{|T|-1} \beta^{|T|-L_D(T)}
\]

where \( \alpha := (1 - \beta)^1/(m-1) \), \( |T| \) denotes the number of leaves of \( T \), and \( L_D(T) \) denotes the number of leaves \( T \) has at depth \( D \). It is not hard to show that \( \pi_D(T, \beta) \) indeed defines a probability probability distribution on the set \( T(D) \) of all proper context trees of depth no greater than \( D \).

Prior on \( \theta \). Given a model \( T \), we define a prior distribution on models \( \theta \) (proper context sets, or the corresponding trees) of maximal depth no more than \( D \), as,

\[
\pi(T) := \pi_D(T) := \pi_D(T; \beta) := \alpha^{|T|-1} \beta^{|T|-L_D(T)},
\]

Finally, given \( T \) and the associated parameters \( \theta \), the likelihood of the observations is given as in (1),

\[
P(x_n|x_{n-d+1}^0, \theta, T) = \prod_{s \in T} \theta_s(j)^{a_s(j)},
\]

where, the \( a_s(j) \) are defined in (2) and, by convention, when we write \( \sum_{s \in T} \) or \( \prod_{s \in T} \), we take the corresponding sum or product over all the leaves \( s \) of the tree, not all its nodes. Also, in order to avoid cumbersome notation, in what follows we often write \( x \) for the string \( x_n^0 \) and suppress the dependence on its initial context \( x_{n-d+1}^0 \), so that, for example, we denote,

\[
P(x, \theta|T) = P(x_n^0|x_{n-d+1}^0, \theta) = \prod_{s \in T} \theta_s(j)^{a_s(j)}.
\]
Choice of $\beta$. In order to maintain an “exponential penalization” of large models, the value of $\beta$ needs to be adjusted for different alphabet sizes $m$. A simple calculation suggests the following practical rule: When $m$ is significantly larger than 2, then $\beta$ should be chosen to be close to $1 - 2^{-m+1}$ so that $\alpha \approx 1/2$; and when $m$ is equal to 2 or is not much larger (so that $2^{-1/(m-1)}$ is not close to 1), then $\beta$ should either be taken $\beta \approx 1 - 2^{-m+1}$ as before, or $\beta = 1/2$ for simplicity.

Marginal likelihood. An important and useful property of the prior specification is that the parameters $\theta$ are integrated out: With the vectors $A$, $x$, and $\pi$, the marginal likelihood $P(x|T)$ of the observations $x$ given a model $T$ is,

$$P(x|T) = \int P(x|\theta,T)\pi(\theta|T)d\theta = \prod_{s \in T} P_e(a_s),$$

with,

$$P_e(a_s) := \frac{\prod_{j=0}^{m-1} [(1/2)(3/2) \cdots (a_s(j) - 1/2)]}{(m/2)(m/2 + 1) \cdots (m/2 + M_s - 1)},$$

(3)

where $M_s := a_s(0) + a_s(1) + \cdots + a_s(m-1)$.

In terms of inference, the more interesting quantity is the model posterior distribution,

$$\pi(T|x) = \frac{P(x|T)\pi(T)}{P(x)}.$$

As usual, the main obstacle in the computation of $\pi(T|x)$ is the appearance of $P(x)$, which can be expressed as the weighted mean of the marginal likelihoods $P(x|T)$. We refer to $P(x)$ as the mean marginal likelihood of $x$, and denote it:

$$P^*_D(x) := \sum_{T \in \mathcal{T}(D)} \pi_D(T)P(x|T).$$

The difficulty in computing the mean marginal likelihood $P^*_D(x)$ comes from the fact that the class of variable-memory models is enormously rich, even for moderate (or even small) alphabet sizes $m$ and tree depths $D$: The number $|\mathcal{T}(D)|$ of models in the collection $\mathcal{T}(D)$ of all proper context trees of depth no greater that $D$ grows doubly exponentially in $D$.

Nevertheless, a modification of the CTW algorithm (not discussed further here) makes it possible to compute the mean marginal likelihood $P^*_D(x)$ precisely, without resorting to simulation. Moreover, the $k$-Maximum A Posteriori Probability Tree ($k$-MAPT) algorithm (described next) allows us to compute the $k$ a posteriori most likely tree models.

III. The k-MAPT Algorithm

The $k$-MAPT algorithm takes as input: Observations $x_{D+1}^n$; the size of the alphabet $m$; the maximum context depth $D$; the value of the prior parameter $\beta$; and the number $k$ of the $k$ a posteriori most likely models to be determined.

For the sake of clarity of exposition, we describe the $k$-MAPT algorithm in three stages.

Stage I. Preliminary steps.

(a) Build an $m$-ary tree $T_{\text{MMLA}}$, whose leaves are all the contexts $x_{D+1}^i$, $1 \leq i \leq n$, that appear in the observations $x_{D+1}^n$. If some node $s$ of $T_{\text{MMLA}}$ is at depth $d < D$ and some but not all of its children are in $T_{\text{MMLA}}$, then add all its remaining children as well, so that $T_{\text{MMLA}}$ is a proper tree.

(b) Compute the count vector $a_s$ as in (2), at each node $s$ of the tree $T_{\text{MMLA}}$ (not only at the leaves), and note that $a_s$ will be the all-zero vector for the additional leaves included in the last step of (a).

(c) Compute the probability $P_{e,s} := P_e(a_s)$ given by (3), at each node $s$ of the tree $T_{\text{MMLA}}$, with the convention that $P_e(a_s) = 1$ when $a_s$ is the all-zero count vector.

Stage II. Idealized $k$-MAPT algorithm.

(i) Let $T_{\text{MMLA}}$ be the complete $m$-ary tree at depth $D$; compute the count vectors $a_s$ and probabilities $P_{e,s} = P_e(a_s)$ at all nodes $s$ of $T_{\text{MMLA}}$ as in the preliminary steps (b), (c).

(ii) Starting at the leaves and proceeding towards the root, at each node $s$ we compute a list of $k$ maximal probabilities $P_m^i(s)$ and $k$ position vectors $e^i_s = (e^i_s(0), e^i_s(1), \ldots, e^i_s(m-1))$, for $i = 1, 2, \ldots, k$, where each $e^i_s(j)$ is an integer between 0 and $k$, recursively as follows.

(iia) At each leaf $s$, we let $P_m^i(s) = P_{e,s}$ and $e^i_s = (0, 0, \ldots, 0)$, where the all-zero vector $e^i_s$ indicates that $P_m^i(s)$ corresponds to the value of $P_{e,s}$ and does not depend on the children of $s$ (since there are none). For $i = 2, 3, \ldots, k$, we leave $P_m^i(s)$ and $e^i_s$ undefined.

(iib) At each node $s$ having only $m$ descendants (which are necessarily leaves), we compute the probability-position vector pairs $\beta P_{e,s} = (0, 0, \ldots, 0)$ and $(1 - \beta) \prod_{j=0}^{m-1} P_m^i(s)_{m,s,j}$ (where the all-1 vector indicates that the latter probability only depends on the first maximal probability of each of the children), and sort them as $P_m^i(s)$ and $P_m^i(s)$ in order of decreasing probability. For $i = 3, 4, \ldots, k$, we leave $P_m^i(s)$ and $e^i_s$ undefined.

(iic) A general internal node $s$ has $m$ children, where each child $s_j$ has a list of $k_j$ (for some $1 \leq k_j \leq k$) probability-vector pairs $P_m^j(s)$, $1 \leq i \leq k_j$. We compute the probability $\beta P_{e,s}$ with associated position vector $(0, 0, \ldots, 0)$, and all possible probability-position vector pairs,

$$(1 - \beta) \prod_{j=0}^{m-1} P_m^i(s_{m,s,j})_{m,s,j}, \quad (i_0, i_1, \ldots, i_{m-1}),$$

for all possible combinations of indices $1 \leq i_j \leq k_j$ for $0 \leq j \leq m - 1$. We then sort these $k' = 1 + k_0 \times k_1 \times \cdots \times k_{m-1}$ probabilities in order of decreasing probability, and rename the top $k$ of them as $P_m^i(s)$, for $i = 1, 2, \ldots, k$, together with their associated position vectors $e^i_s$. [Of course, if $k' < k$, after sorting we leave the remaining $k - k'$ probability-position vector pairs undefined.]

(iii) Having determined all maximal probabilities $P_m^i(s)$ for all nodes $s$ and $1 \leq i \leq k$, we now determine the “top $k$” trees $T_1^*, T_2^*, \ldots, T_k^*$ from the corresponding position vectors $e^i_s$. For each $i$ we repeat the following process, starting at the root
and proceeding until all available nodes of the tree $T_{\text{MMLA}}$ have been exhausted.

(iii) Depth $d = 0$. At the root node $\lambda$, we examine $e_{\lambda}^{(i)}$. If it is the all-zero vector, then $T_{\lambda}^*$ is the tree consisting of the root node only. Otherwise, we add to $T_{\lambda}^*$ the branch of $m$ children starting at the root, and proceed to examine each of the nodes corresponding to the $m$ children recursively.

(iii) Depth $d = 1$. Reaching node $s = j$ corresponding to the $j$th child of the root, means that $t = c_{(a)}^{(j)}(j)$ is nonzero for the appropriate index $u$ (corresponding to the position vector $c_{(a)}^{(u)}$ that was examined at node $s$). We examine $e_{s}^{(i)}$: If it is the all-zero vector, then we prune from $T_{s}^*$ all the descendants of $s$ and move to the next unexamined node; otherwise, we add to $T_{s}^*$ the branch of $m$ children starting at $s$, and proceed to examine each of the nodes corresponding to the $m$ children recursively.

(iii) General depth $1 \leq d \leq D - 1$. Reaching a node $s_j$ at depth $d$ from its parent node $s_i$ means that we decided to visit $s_j$ because $t = c_{(a)}^{(j)}(j)$ is nonzero for the appropriate index $u$ (corresponding to the position vector $c_{(a)}^{(u)}$ that was examined at node $s_i$). We examine $e_{s_j}^{(i)}$: If it is the all-zero vector, then we prune from $T_{s_j}^*$ all the descendants of $s_j$ and move to the next unexamined node; otherwise, we add to $T_{s_j}^*$ the branch of $m$ children starting at $s_j$, and proceed to examine each of the nodes corresponding to the $m$ children recursively.

(iii) Depth $d = D$. Reaching a node $s$ at depth $D$ means we have reached a leaf of $T_{\text{MMLA}}$, so we simply add $s$ to $T_{s}^*$ and proceed to the next unexamined node.

(iv) Output the $k$ resulting trees $T_{i}^*$ and the $k$ maximal probabilities at the root, $P_{m,\lambda}^{(i)}$, $i = 1, 2, \ldots, k$.

Stage III. Actual $k$-MAPT algorithm

(i) Initially, perform two preprocessing steps:

(ii) Execute the first two steps (i) and (ii) of the idealized $k$-MAPT algorithm on the complete $m$-ary tree of depth $D$, with all the count vectors $a_s$ assumed to be equal to zero, $a_s = (0, 0, \ldots, 0)$ for all $s$. Since all nodes at the same depth are identical, this process can be carried out effectively, by performing the relevant computations only at a single node (instead of all $m^d$ nodes) for each depth $0 \leq d \leq D$.

(iii) Build the tree $T_{\text{MMLA}}$ and compute the count vectors $a_s$ and the probabilities $P_{\pi,\lambda} = P_{\lambda}(a_s)$ at all nodes $s$ of $T_{\text{MMLA}}$, as in preliminary steps (a)–(c).

(iv) At each leaf $s$ at depth $D$, with a nonzero count vector $a_s$, let $P_{m,s}^{(i)} = P_{\pi,\lambda}$ and $c_{s}^{(i)} = (0, 0, \ldots, 0)$. For $i = 2, 3, \ldots, k$, we leave $P_{m,s}^{(i)}$ and $c_{s}^{(i)}$ undefined.

(v) Similarly, at each leaf $s$ at depth $D$, with an all-zero count vector $a_s$, let $P_{m,s}^{(1)} = P_{\pi,\lambda} = 1$, $c_{s}^{(1)} = (0, 0, \ldots, 0)$, and for $i = 2, 3, \ldots, k$, leave $P_{m,s}^{(i)}$ and $c_{s}^{(i)}$ undefined.

(vi) At each leaf $s$ at depth $d < D$, we let the list of the maximal probabilities $P_{m,s}^{(i)}$ and position vectors $c_{s}^{(i)}$ of $s$ be those that are computed for a node at depth $d$ in the preprocessing stage (ia).

(iid) Continue with steps (ii) and (iic) as in the idealized version of $k$-MAPT.

(iii) We perform the same steps as described in (iii)–(iid) of the idealized version, with the following addition:

(iii) While examining a node $s$ at depth $0 < d < D$ in step (iiib) or (iiic) of the idealized algorithm, we may reach a point where the algorithm dictates that we examine its $m$ children, when these children are not included in the tree $T_{\text{MMLA}}$. In that case, we add them to $T_{\text{MMLA}}$, and we define their corresponding maximal probabilities and position vectors according to the initialization described in step (iiib) or (iiic) above, depending on whether $d = D - 1$ or $d < D - 1$, respectively.

(iv) As in the idealized version, output the $k$ resulting trees $T_{i}^*$ and the $k$ maximal probabilities at the root, $P_{m,\lambda}^{(i)}$, $i = 1, 2, \ldots, k$.

\textbf{Theorem 3.1:} For any $\beta \geq 1/2$, the trees $T_{1}^*, T_{2}^*, \ldots, T_{k}^*$ produced by the $k$-MAPT algorithm are indeed the $k$ \textit{a posteriori} most likely trees.

Moreover, the $i$th maximal probability at the root satisfies:

$P_{m,\lambda}^{(i)} = P(x|T_{i}^*)_{\pi,D}(T_{i}^*) = P(x, T_{i}^*)$, $i = 1, 2, \ldots, k$.

\textbf{Remarks.}

1. It is not hard to show that the time-complexity of the $k$-MAPT algorithm is $O(nmk^mD)$. Hence, the complexity increases dramatically as we require more information about the “top” of the posterior $\pi(T|x)$ in model space. Also, note the memory required by the $k$-MAPT algorithm is $O(nmkD)$.

2. As mentioned earlier, in addition to identifying the $k$ \textit{a posteriori} most likely models, it is also possible to compute the mean marginal likelihood $P_D(x)$ of the observations, which makes it possible to then compute the Bayes factors and posterior odds for different models $T$. This is illustrated in the example presented in the next section.

3. Finally, some bibliographical comments are in order. The origin of some of the ideas behind the $k$-MAPT algorithm can be traced to the CTW algorithm as described in the unpublished manuscript [28]. A special case of the $k = 1$ version of the $k$-MAPT algorithm was introduced in [28, 30, 31]. The $k$-MAPT algorithm and the result of Theorem 3.1 are both new, although they were, in part, motivated by some remarks in [27].

As mentioned earlier, Rissanen’s early results in [19, 20] and their extensions in [26] form the real foundation of the present development. There, in addition to variable-memory Markov models, Rissanen also introduced the CONTEXT algorithm for fitting such a model to data. A different asymptotic analysis of CONTEXT was later carried out in the VLMC paper [4], and a more detailed examination of the algorithm in connection with model selection was given in [3]. In all these works, no Bayesian or other finite-sample interpretation is given for the resulting model. Finally, we mention that in the first PST paper [23], a different procedure called Learn-PSA, similar in spirit to CONTEXT, was described for estimating a variable-memory Markov model; see also the relevant results in [1].
IV. A SIMULATION EXAMPLE

We revisit the variable-memory chain \( \{X_t\} \) on the alphabet \( A = \{0, 1, 2\} \) discussed in the Introduction. The top \( k = 3 \) a posteriori most likely models were obtained by the \( k \)-MAPT algorithm from \( n = 10000 \) simulated samples. The algorithm’s parameters were \( m = 3, \beta = 3/4, \) and maximum depth \( D = 10. \)

The most likely model was found to be the true underlying model \( T_1^* \), shown in the Introduction; its posterior probability \( \pi(T_1^*|X) \approx 0.368 \) while its prior \( \pi(T_1^*) \approx 5.8 \times 10^{-6} \). For the second and third most likely models \( T_2^*, T_3^* \) shown below, we have the posterior odds, \( \pi(T_1^*|X)/\pi(T_2^*|X) \approx 6.2903 \) and \( \pi(T_1^*|X)/\pi(T_3^*|X) \approx 8.822. \)

It is worth noting perhaps that the correct model is identified, based on only \( n = 10^4 \) samples, among well over \( 10^{13000} \) possible models of maximum depth at most 10.

REFERENCES