Statistical Inference for Hidden Markov Tree Models and Application to Wavelet Trees
Jean-Baptiste Durand, Paulo Gonçalves

To cite this version:

HAL Id: inria-00072339
https://inria.hal.science/inria-00072339
Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Statistical Inference for Hidden Markov Tree Models
and Application to Wavelet Trees

Jean-Baptiste Durand — Paulo Gonçalvès

N° 4248
Septembre 2001
Statistical Inference for Hidden Markov Tree Models and Application to Wavelet Trees

Jean-Baptiste Durand, Paulo Gonçalvès

Thème 4 — Simulation et optimisation de systèmes complexes
Projet is2

Rapport de recherche n° 4248 — Septembre 2001 — 30 pages

Abstract: The hidden Markov tree model was introduced by Crouse, Nowak and Baraniuk (1998) for modeling non-independent, non-Gaussian wavelet transform coefficients. In their article, they developed an inductive algorithm, called “upward-downward” algorithm, for likelihood computation. They also introduced Expectation Maximization algorithms for likelihood maximization. These algorithms are subject to the same numerical limitations as the “forward-backward” procedure for hidden Markov chains. In this report, we develop efficient variants of the “upward-downward” and EM algorithms, inspired by Devijver’s “conditional forward-backward” recursion (1985). Thus, the inference algorithms limitations for hidden Markov trees are considerably reduced. Moreover, as the hidden states restoration problem has no known solution for hidden Markov trees, we present the MAP algorithm for this model. The interest of those algorithms is illustrated by an application to signal processing.

Key-words: hidden Markov tree model, maximum likelihood estimation, hidden states restoration, maximum a posteriori algorithm, upward-downward algorithm
Inférence statistique pour les modèles
d’arbres de Markov cachés
et application au traitement du signal


Mots-clés : modèle d’arbre de Markov caché, estimation par maximum de vraisemblance, restauration des états cachés, maximum a posteriori, algorithme haut-bas
1 Introduction

The hidden Markov tree model (HMT) was introduced by Crouse, Nowak and Baraniuk (1998). The context of their work was the modeling of statistical dependencies between wavelet coefficients in signal processing, for which variables are organised in a natural tree structure. Applications of such a model are: image segmentation, signal classification, denoising and image document categorization, among other examples (see Hyekho and Baraniuk, 1999; Diligenti, Frasconi and Gori, 2001). This model shares similarities with hidden Markov chains: both are mixture models, parameterized by a transition matrix and parameters of conditional distributions given hidden states. Both models can be identified through the EM algorithm, involving a forward-backward recursion. And in both cases this recursion involves joint probabilities which tend towards zero exponentially fast as the number of data increases causing underflow problems on computers.

Our first aim is to adapt the forward-backward algorithm of Devijver (1985) to the hidden Markov tree model to answer this numerical limitation. This algorithm called conditional upward-downward recursion is based on the computation of conditional probabilities instead of joint probability densities for hidden Markov chains, thus overcoming the computational limitations of the standard algorithm. However the adaptation to hidden Markov trees is not straightforward and the resulting algorithm involves an additional step consisting in computing the hidden states marginal distribution.

Then we present the Maximum A Posteriori algorithm (MAP) for hidden Markov tree models. This algorithm is important for the restoration of hidden states, which can be useful in itself. It is analogous to the Viterbi algorithm for hidden Markov chains. As far as we know, this algorithm is the first proposed solution to the restoration problem for hidden Markov trees. It also makes use of the conditional probabilities and we provide an original proof for this solution, giving an interpretation for the variables involved.

This paper is organized as follows. The hidden Markov tree model is introduced in Section 2. We also present the three problems related to the general hidden Markov models, i.e. training, likelihood determination and state estimation. Their practical interest have been pointed out by Rabiner (1989) in his tutorial. A classical solution for the first two problems is summarized in Section 3. A parallel is drawn between the resulting algorithm and the forward-backward algorithm for hidden Markov chains. We examine the adaptation of Devijver’s algorithm (1985) in Section 4. A solution for the states estimation problem is proposed in Section 5. An illustration based on simulations is provided in Section 6. In this section, the interest of the HMT model in signal processing is briefly discussed. Section 7 contains some concluding remarks. Appendices A, B and C contain the justification for the conditional upward-downward formulae. Appendix D proves the optimality of the hidden tree obtained by the restoration step described in Section 5.
2 HMT model

We use the general notation $P()$ to denote either a probability measure or a probability function density, the true nature of $P()$ being obvious from the context. In the same way, when dealing with joint probabilities or pdfs, we use the notation $P()$. This notation makes any assumption on the discrete or continuous nature of the observed process unnecessary.

Let $W = (W_1, \ldots, W_n)$ refer to the observed data. They are supposed to be also indexable as a tree rooted in $W_1$. For convenience’s sake, we suppose that each non terminal node has at least to children and that the length of the path joining any terminal node to the root is a constant, called the tree depth and denoted by $J_0$. Our work can easily be extended to any type of tree at the cost of tedious notation. These variables are said to define a hidden Markov tree if and only if they fulfill the five assumptions:

- $\forall u \in \{1, \ldots, n\}$, $W_u$ arise from a mixture of distributions with density

$$P(W_u = w) = \sum_{i=1}^{K} P(S_u = i)P_{\theta_i}(w)$$

where $S$ is a discrete variable with $K$ states, denoted $\{1, \ldots, K\}$.

- $(S_1, \ldots, S_n)$ has the same indexation structure as $W$. Thus it can be indexed as a tree rooted in $S_1$. For each $u$ in $\{2, \ldots, n\}$ we denote the parent of $S_u$ by $S_{\rho(u)}$. If $S_u$ is not a leaf, we denote its children by $(S_{c_1}^u, \ldots, S_{c_{\nu(u)}}^u)$ where $\nu(u)$ represents the children number for node $S_u$. We also denote the children’s indexes set by $c(u)$. This notation is illustrated in Figure 1.

- $P(S_u|\{S_{w'}|w' \neq u\}) = P(S_u|S_{\rho(u)}, S_{c_1}^u, \ldots, S_{c_{\nu(u)}}^u)$ (Markov tree property),

- $P(W_1, \ldots, W_n|S_1, \ldots, S_n) = \prod_{u=1}^{n} P(W_u|S_1, \ldots, S_n)$,

- $\forall u \in \{1, \ldots, n\}$ $P(W_u|S_1, \ldots, S_n) = P(W_u|S_u)$.

We refer to the last two properties as conditional independence properties. It is important at this stage to distinguish between the indexation structure of $(W_1, \ldots, W_n)$ or $(S_1, \ldots, S_n)$ and the influence diagram which involves all variables $(S_1, \ldots, S_n, W_1, \ldots, W_n)$. The influence diagram is a graphical way for describing conditional independence relations between variables (see Smyth, Heckerman and Jordan, 1996). The influence diagram for HMT model is shown in Figure 2. We chose to represent the independence relations between variables by a directed acyclic graph to match the original parameterization of Crouse, Nowak and Baraniuk (1999). However, it can be seen from Smyth, Heckerman and Jordan (1996) that the model obtained by dropping all the directions on the edges in Figure 2 has the same independance properties as the directed model, as each node has one parent at the most.

Such a model is characterized by the following parameters:

INRIA
• The distribution for the root node \( S_1 \pi = (\pi_k)_{k \in \{1, \ldots, K\}} \) and the transition probabilities \( A = (a^{rm}_{u(r)}, u)_{u \in \{2, \ldots, n\}, r \in \{1, \ldots, K\}, m \in \{1, \ldots, K\}} \), defined by
\[ a^{rm}_{u(r)} = P(S_u = m | S_{\rho(u)} = r). \] In the rest of this paper, we suppose that the transition matrix does not depend on \( u \), which is a realistic assumption in most applications and makes the notation lighter. Thus we have \( P(S_u = m | S_{\rho(u)} = r) = a_{rm}. \) We denote \( P(S_u = m | S_{\rho(u)} = r) = a_{rm} \) by \( A. \)

• The parameters of the mixture components \( (\theta_1, \ldots, \theta_K) \), such as
\[ P(W_u = w | S_u = k) = P_{\theta_k}(w), \]
where \( P_{\theta} \) belongs to a parametric distribution family. For example, \( P_{\theta} \) can be the density of a Gaussian distribution. Then \( \theta = (\mu, \Sigma) \) denotes the mean and variance matrix of the Gaussian distribution.

We denote \( \lambda = (\pi, A, \theta_1, \ldots, \theta_K) \). The first problems adressed here are the computation and the maximization of the likelihood \( L_{\omega}(\lambda) = P_{\lambda}(w) \). For the second problem we can resort to the Expectation Maximization (EM) algorithm of Dempster, Laird and Rubin (1977). The \( E \) step of the resulting algorithm essentially consists of computing conditional probabilities \( P(S_u = k | W) \) and \( P(S_{\rho(u)} = i, S_u = k | W) \). It is based on an inductive
algorithm named upward-downward algorithm, similar to the forward-backward algorithm for hidden Markov chains.

3 Upward-downward algorithm

We introduce the following notation:

- $T_u$ is the subtree with root at node $u$. Thus $T_1$ is the entire tree.
- If $T_i$ is a subtree of $T_u$ then $T_u \setminus i$ is the set of nodes in $T_u$ which are not in $T_u$.

The upward-downward algorithm has been developed by Crouse et al. (1998). The upward step consists of computing the joint conditional probability of each subtree $T_u$ of the entire tree and starts from the terminal nodes. The downward step consists of computing the joint probability of $T_1$ where each subtree has been removed in turn. It starts from the root of the entire tree. The authors define the following variables:

\[
\begin{align*}
\beta_u(k) &= P(T_u|S_u = k) \\
\beta_{u, p(u)}(k) &= P(T_u|S_{p(u)} = k) \\
\beta_{p(u) \setminus u}(k) &= P(T_{p(u)} \setminus u|S_{p(u)} = k) \\
\alpha_u(k) &= P(S_u = k, T_{1 \setminus u})
\end{align*}
\]

The upward step is described by algorithm 1 and the downward step by algorithm 2. The likelihood is given by the formula:

\[
\forall u \in \{1, \ldots, n\} \quad P(w) = P(w_1, \ldots, w_n) = \sum_{k=1}^{K} \beta_u(k) \alpha_u(k)
\] (1)
Algorithm 1 UP step, linking up the \( \beta \) variables

1. **Initialization**
   
   For all leaves \( W_u \) of \( T_1 \)
   
   for all \( k = 1, \ldots, K \) do
   
   \( \beta_u(k) = P_{\theta_u}(w_u) \)
   
   end for
   
   for all \( j = 1, \ldots, K \) do
   
   \( \beta_{u,\rho(u)}(j) = \sum_{i=1}^{K} \beta_u(i)a_{j i} \)
   
   end for
   
2. **Induction**

   for all \( s = 1, \ldots, J_0 - 1 \) do
   
   for all nodes \( W_u \) at scale \( J_0 - s \) do
   
   for all \( k = 1, \ldots, K \) do
   
   \( \beta_u(k) = P_{\theta_u}(w_u) \prod_{t \in c(u)} \beta_{t,u}(k) \)
   
   for all \( t \in c(u) \) do
   
   \( \beta_{u \setminus t}(k) = \frac{\beta_u(k)}{\beta_{t,u}(k)} \)
   
   end for
   
   end for
   
   for all \( j = 1, \ldots, K \) do
   
   \( \beta_{u,\rho(u)}(j) = \sum_{i=1}^{K} \beta_u(i)a_{j i} \) (except at root node)
   
   end for
   
   end for
   
end for
Algorithm 2 \textit{DOWN} step, linking up the $\alpha$ and $\beta$ variables

1. Initialization
   \begin{align*}
   &\text{for all } k = 1, \ldots, K \text{ do} \\
   &\quad \alpha_1(k) = \pi_k \\
   &\text{end for}
   \end{align*}

2. Induction
   \begin{align*}
   &\text{for all } s = 2, \ldots, J_0 \text{ do} \\
   &\quad \text{for all nodes } W_u \text{ at scale } s \text{ do} \\
   &\quad\quad \text{for all } k = 1, \ldots, K \text{ do} \\
   &\quad\quad\quad \alpha_u(k) = \sum_{i=1}^{K} \alpha_{p(u)(i)} a_{ik} \beta_{p(u)\backslash u}(i) \\
   &\quad\quad\text{end for} \\
   &\quad \text{end for} \\
   &\text{end for}
   \end{align*}

It is worth noting that the right member of equation (1) does not depend on $u$. It means that the likelihood of the parameter can be calculated by splitting the tree at any node. The conditional probabilities used in the EM algorithm are

\[
\begin{align*}
P(S_u = k|w) &= \frac{\beta_u(k) \alpha_u(k)}{P(w)} \\
P(S_{p(u)} = i, S_u = k|w) &= \frac{\beta_u(k) a_{ik} \alpha_{p(u)(i)} \beta_{p(u)\backslash u}(i)}{P(w)}
\end{align*}
\]

Table 1 points out the similarities between the \textit{upward-downward} algorithm for hidden Markov trees and the \textit{forward-backward} algorithm for hidden Markov chains.

As for hidden Markov chains (see Levinson et al., 1983) the joint probability densities $\alpha_u(i)$ and $\beta_u(i)$ satisfy:

\[
\lim_{u \to \infty} \beta_u(i) = 0 \quad \text{and} \quad \lim_{u \to \infty} \alpha_u(i) = 0
\]

and the rate of convergence is exponential. This property causes underflow problems when executing the algorithm. A typically observed limitation for $n$ on Matlab 5.2 is $n = 127$ (i.e. a depth tree limited to 7). In the next section we present an algorithm overcoming this difficulty.

4 \textbf{Upward-downward algorithm using conditional probabilities}

In order to avoid underflow problems with hidden Markov chains, Devijver (1985) suggests to use, instead of the forward variables $P(S_t = k, W_1, \ldots, W_t)$ and the backward variables
$P(W_{t+1}, \ldots, W_n | S_t = k)$, the conditional variables:

$$P(S_t = k | W_1, \ldots, W_t) \quad \text{and} \quad \frac{P(W_{t+1}, \ldots, W_n | S_t = k)}{P(W_{t+1}, \ldots, W_n | W_1, \ldots, W_t)}$$

The conditional backward variables computation needs the conditional forward variables. Thus it is necessary to run the forward algorithm first.

A natural adaptation of this method would be to use the following variables for hidden Markov trees:

$$P(S_u = k | T_{1 \setminus u}) \quad \text{and} \quad \frac{P(T_u | S_u = k)}{P(T_u | T_{1 \setminus u})} \quad (2)$$

The difficulty comes from the fact that as in hidden Markov chains, it would be necessary to run the down step before the up step. This is in conflict with the fact that the down step uses the results of the up step (see algorithm 2).

The main idea in our adaptation of the conditional forward-backward algorithm is to reverse the roles of the $\alpha$ and $\beta$ variables. The use of conditional probabilities implies the definition of a scaling factor $\mathcal{M}_u$ as in Devijver’s algorithm. However, in our case, $\mathcal{M}_u$ comes from the normalization of the $\beta$ variables (instead of the $\alpha$ variables for hidden Markov chains). This scaling factor will play a role in the computation of the conditional downward variables denominators $P(T_{1 \setminus u} | T_u)$, as shown by the proof in Appendix B. Moreover, the conditional forward-backward algorithm involves the computation of the marginal distribution of each hidden variable. This can be achieved by a preliminary step based on a downward exploration of the hidden tree. Thus we could call this a conditional downward-upward-downward algorithm. This term accounts for the substantial modification of the original upward-downward and conditional forward-backward algorithms.
In the case of the conditional algorithm, as in Devijver’s algorithm, the value of the likelihood can not be inferred from the $\tilde{\alpha}$ and the $\tilde{\beta}$ variables. Instead, its computation involves the scaling factor $M_u$. In practice, we use the quantity $\log(M_u)$ to dynamically update a variable denoted by $l_u$ in the conditional up step defined in algorithm 4 below. We prove in Appendix C that

$$l_u = \log(P(T_u)) - \sum_{W_i \in \mathcal{F}(T_u)} \log(P(W_i)),$$

where $\mathcal{F}(T_u)$ denotes the set of observed variables located in the leaves of $T_u$ (see Figure 1). We use the following conditional probabilities instead of the probabilities involved in (2):

$$P(S_u = k|T_u) = P(S_{\rho(u)} = k|T_u) = \frac{P(T_u|S_u = k)}{P(T_u|S_{\rho(u)} = k)}$$

As in standard upward-downward algorithms, the variables $\tilde{\beta}_u(k)$, $\tilde{\beta}_{u,\rho(u)}(k)$, $\tilde{\beta}_u\setminus\rho(u)(k)$, $\tilde{\alpha}_u(k)$ and $M_u$ are computed inductively by algorithms 4 and 5. As proved in Appendices A, B and C, the following equations hold:

$$\tilde{\beta}_u(k) = P(S_u = k|T_u)$$

$$\tilde{\beta}_{u,\rho(u)}(k) = P(S_{\rho(u)} = k|T_u)$$

$$M_u = \frac{P(T_u)}{\prod_{i=1}^{n_u} P(T_{c_i}^u)} = P(W_u|T_{c_1}^u, \ldots, T_{c_{n_u}}^u) \text{ if } u \text{ is not a leaf}$$

$$\tilde{\alpha}_u(k) = \frac{P(T_1|S_u = k)}{P(T_1|T_u)}$$

$$l_u = \log(P(T_u)) - \sum_{W_i \in \mathcal{F}(T_u)} \log(P(W_i))$$

Now we define $\tilde{\beta}_{u\setminus\rho(u)}(k)$ as $\frac{\tilde{\beta}_{u}(k)}{\beta_{u,\rho(u)}(k)}$.

The corresponding new upward-downward algorithm includes the three steps described by algorithms 3, 4 and 5 (’\$’ denotes the matrix product).

The computation of the loglikelihood and the conditional probabilities involved in the EM algorithm can be inferred from the equations below. Proof of equation (9) is given in Appendix C. Equations (10) and (11) follow directly from the classical formulae in Table 1 and from the conditional variables definition.

$$\log(P(T_i)) = l_1 + \sum_{W_u \in \mathcal{F}(T_i)} \log(P(W_u))$$

$$P(S_u = k|w) = \tilde{\alpha}_u(k) \tilde{\beta}_u(k)$$

$$P(S_{\rho(u)} = i, S_u = k|w) = \frac{\tilde{\beta}_u(k)}{P(S_u = k)} \alpha_{\rho(u)}(i) \tilde{\beta}_{u\setminus\rho(u)}(i) P(S_{\rho(u)} = i)$$

INRIA
Algorithm 3 Computation of the distribution of hidden states

1. Initialization
   \[ \text{for all } k = 1, \ldots, K \text{ do} \]
   \[ P(S_t = k) = \pi_k \]
   \[ \text{end for} \]

2. Induction
   \[ \text{for all } s = 2, \ldots, J_0 \text{ do} \]
   \[ \text{for all nodes } W_u \text{ at scale } s \text{ do} \]
   \[ [P(S_u = 1), \ldots, P(S_u = K)] = [P(S_{h_u} = 1), \ldots, P(S_{h_u} = K)] \cdot A_u \]
   \[ \text{end for} \]
   \[ \text{end for} \]

where the probabilities \((P(W_u))_{W_u \in \mathcal{F}(T_1)}\) have been computed in the initialization step of the upward algorithm.

Table 2 points out the similarities and differences between the conditional upward-downward algorithm for hidden Markov trees and the conditional forward-backward algorithm for hidden Markov chains. As for Devijver’s algorithm, the execution of the above procedure does not cause underflow problems. Its implementation in Matlab allows to handle trees having more than 8000 nodes (i.e. the maximal depth of the tree is at least 13).

We prove in Appendices A and B that the \( \hat{\beta} \) and \( \hat{\alpha} \) variables defined by the algorithms 4 and 5 satisfy equations (4) to (7). We also prove equation (9) in Appendix C. It is worth noting that this new algorithm makes use of a scaling factor due to the conditional probabilities, as in Devijver and of the marginal distribution of hidden states due to the inversion between \( \alpha \) and \( \beta \). The first step of the algorithm (see algorithm 3) has a complexity order of \( O(nK^2) \). As the standard algorithm also has a \( O(nK^2) \) complexity, the complexity order of the conditional upward-downward algorithm remains \( O(nK^2) \).

5 MAP algorithm

Let \( T_u \) denote the subtree of the entire tree rooted in node \( u \) and \( \mathcal{H}_u \) the set of hidden variables corresponding to \( T_u \) (see Figure 2). The aim of the MAP algorithm is to find the optimal hidden tree \( \hat{h}_1 = (s_1, \ldots, s_n) \) maximizing \( P(\mathcal{H}_1 = h_1 | T_1) \) and the value \( \hat{P} \) of the maximum. The MAP algorithm for hidden Markov trees is defined by algorithm 6. It is based on the conditional upward algorithm 4. As a consequence, it requires the computation of the hidden states marginal distribution and the scaling factors. Theses factors appear as constants in the conditional MAP algorithm and do not actually need to be computed. Their purpose is to make the \( \delta \) variables interpretable in a probabilistic way.
Algorithm 4 Conditional UP step

1. Initialization
   For all leaves $W_u$ of $T_1$
   
   for all $k = 1, \ldots, K$
   
   $\tilde{\beta}_u(k) = \frac{P_k(u_x)P(S_u=i)}{\sum_{i=1}^K P_k(u_x)P(S_u=i)}$
   
   end for
   
   for all $j = 1, \ldots, K$
   
   $\tilde{\beta}_{u,\rho(u)}(j) = \left[ \sum_{i=1}^K \tilde{\beta}_u(i) \frac{a_{ij}}{P(S_u=i)} \right] P(S_{\rho(u)} = j)$
   
   end for
   
   $l_i = 0$

2. Induction

   for all $s = 1, \ldots, J_0 - 1$

   for all nodes $W_u$ at scale $J_0 - s$

   $M_u = \sum_{k=1}^K P_k(u_x) \prod_{t \in c(u)} \tilde{\beta}_u(k)$

   $l_u = \log(M_u) + \sum_{t \in c(u)} l_t$

   for all $k = 1, \ldots, K$

   $\tilde{\beta}_u(k) = \frac{P_k(u_x) \prod_{t \in c(u)} \tilde{\beta}_u(k)}{P(S_u=k) \times 1-M_u}$

   for all $t \in c(u)$

   $\tilde{\beta}_{u \setminus t}(k) = \frac{\tilde{\beta}_u(k)}{\tilde{\beta}_{u,\rho(u)}(k)}$

   end for

   end for

   for all $j = 1, \ldots, K$

   $\tilde{\beta}_{u,\rho(u)}(j) = \left[ \sum_{i=1}^K \tilde{\beta}_u(i) \frac{a_{ij}}{P(S_u=i)} \right] P(S_{\rho(u)} = j)$ (except at root node)

   end for

end for

end for
Algorithm 5 Conditional $\text{DOWN}$ step

1. Initialization
   \[\tilde{\alpha}_1(k) = 1\]
   \[
   \forall k = 1, \ldots, K \quad \text{end for}
   \]

2. Induction
   \[\forall s = 2, \ldots, J_0 \quad \text{end for}
   \]
   \[
   \forall k = 1, \ldots, K \quad \text{end for}
   \]

\[
\tilde{\alpha}_u(k) = \left[ \sum_{i=1}^{K} \tilde{\alpha}_u(i) a_{u,\rho(u)}^k \tilde{\beta}_{\rho(u)}(i) P(S_{\rho(u)} = i) \right] \frac{1}{P(S_u = k)}
\]

<table>
<thead>
<tr>
<th>Hidden Markov Chains</th>
<th>Hidden Markov Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional forward variables :</td>
<td>Conditional forward variables :</td>
</tr>
<tr>
<td>$\tilde{\alpha}_t(k) = P(S_t = k</td>
<td>W_1, \ldots, W_t)$</td>
</tr>
<tr>
<td>$\tilde{\beta}_u(k) = P(S_u = k</td>
<td>T_u)$</td>
</tr>
<tr>
<td>$\tilde{\beta}<em>{\rho(u)}(k) = P(S</em>{\rho(u)} = k</td>
<td>T_u)$</td>
</tr>
<tr>
<td>Conditional backward variables :</td>
<td>Conditional backward variables :</td>
</tr>
<tr>
<td>$\tilde{\beta}<em>t(k) = \frac{P(W</em>{t+1}, \ldots, W_{n}</td>
<td>S_t = k)}{P(W_{t+1}, \ldots, W_{n}</td>
</tr>
<tr>
<td>Conditional probabilities</td>
<td>Conditional probabilities</td>
</tr>
<tr>
<td>$P(S_t = k</td>
<td>w) = \tilde{\beta}_t(k) \tilde{\alpha}_t(k)$</td>
</tr>
</tbody>
</table>

Table 2: Similarities between conditional forward-backward and upward-downward algorithms
Algorithm 6 MAP algorithm

1. Initialization

For all leaves \( W_u \) of \( T_1 \)

for all \( k = 1, \ldots, K \) do

\( \delta_u(k) = \hat{\beta}_u(k) \)

end for

for all \( j = 1, \ldots, K \) do

\( \delta_{u,r(u)}(j) = \max_{1 \leq i \leq K} \left[ \delta_u(i) \frac{\theta_{ji}}{P(U_S = i)} \right] P(S_{r(u)} = j) \)

\( \psi_u(j) = \arg \max_{1 \leq i \leq K} \left[ \delta_u(i) \frac{\theta_{ji}}{P(U_S = i)} \right] \)

end for

2. Induction

for all \( s = 1, \ldots, J_0 - 1 \) do

for all nodes \( W_u \) at scale \( J_0 - s \) do

for all \( k = 1, \ldots, K \) do

\( \delta_u(k) = P_{u}(w_u) \prod_{j \in u} \delta_{u,j}(k) \)

end for

for all \( j = 1, \ldots, K \) do

\( \delta_{u,r(u)}(j) = \max_{1 \leq i \leq K} \left[ \delta_u(i) \frac{\theta_{ji}}{P(U_S = i)} \right] P(S_{r(u)} = j) \) (except at root node)

\( \psi_u(j) = \arg \max_{1 \leq i \leq K} \left[ \delta_u(i) \frac{\theta_{ji}}{P(U_S = i)} \right] \)

end for

end for

3. Termination

\( \hat{P} = \max_{1 \leq i \leq K} \delta_1(i) \)

\( \hat{s}_1 = \arg \max_{1 \leq i \leq K} \delta_1(i) \)

4. “Downward-tracking”

(creation of the tree from root)

for all \( s = 2, \ldots, J_0 \) do

for all nodes \( W_u \) at scale \( s \) do

for all \( u = 2, \ldots, n \) do

\( \hat{s}_u = \psi_u(\hat{s}_{r(u)}) \)

end for

end for

end for
We prove in Appendix D that the tree defined by \((\hat{s}_u)_u\) is maximizing

\[ P(S_1 = s_1, \ldots, S_n = s_n|T_1). \]

The initial MAP algorithm for non independent mixture models is due to Viterbi. It is originally intended to analyze Markov processes observed in memoryless noise. The justification of this algorithm given in Forney (1973) is based on a graphical argument. The maximization of the conditional probabilities is proved to be equivalent to finding the shortest path in a graph with weighted edges. This proof could be adapted in the context of hidden Markov trees. We could also define a non-conditional MAP algorithm (algorithm 7) which would be an application of Dawid’s algorithm for graphical models (see Smyth, Heckerman and Jordan, 1996). The proof would be more direct but would not provide any interpretation for the variables involved.

However, we give a more analytical proof which has the advantage of giving an interpretation of the variables involved in algorithm 6, although the details of the demonstration are tedious. The proof is based on the following statements. The \(\delta_u\) and \(\delta_{u, p(u)}\) variables satisfy

\[
\delta_u(k) = \max_{h_{c_1}^u, \ldots, h_{c_{\delta_u}}^u} P(H_{c_1}^u = h_{c_1}^u, \ldots, H_{c_{\delta_u}}^u = h_{c_{\delta_u}}^u, S_u = k|T_u) \quad (12)
\]

\[
\delta_{u, p(u)}(k) = \max_{h_u} P(H_u = h_u, S_{p(u)} = k|T_u) \quad (13)
\]

From these equations, it can be seen that the \(\delta\) variables are joint probabilities going towards zero when \(u\) increases. To avoid underflow problems, the easiest solution is then to use \(\log(\delta)\) instead of \(\delta\). As only products and maxima are necessary for the MAP algorithm, the adaptation is straightforward. As the quantities \(\pi_r, P_\theta(w)\) or \(a_{rm}\) involved in algorithm 6 may be equal to 0, we consider that in this case the logarithmic value is \(-\infty\).

As a conclusion, both conditional and non-conditional versions of the MAP algorithm lead to underflow problems if we do not use a logarithmic variant. As the non-conditional algorithm is computationally less expensive, we would recommend this version in practice.

6 An application to signal processing

In this section, we develop one example of application, illustrating the interest of the hidden Markov tree model. Let \(x = (x_1, \ldots, x_T)\) be a realization of a piecewise constant (Hölder) regularity process, for example a piecewise homogeneous fractional Brownian motion. The local regularity of a function (or of the trajectory of a stochastic process) is defined as Mallat (1999): the function \(f\) has local regularity \(k < h < k + 1\), at time \(t\), if there exists two constant \(0 < C < \infty\) and \(0 < t_0\) as well as a polynomial \(P_k\) of order \(k\), such that for all \(t - t_0 < l < t + t_0\)

\[ |f(l) - P_k(l)| < C|l - t|^h \]

We assume that \(T = 2^M\) and that from \(t = 1\) to \(t = T_0\) with \(1 \leq T_0 < T\), the local regularity of the process is \(H = H_0\) and from \(t = T_0 + 1\) to \(t = T\), its local regularity is \(H = H_1\). Our aim is to estimate \(H_0, H_1\) and \(T_0\).
Algorithm 7 Non conditional MAP algorithm

1. Initialization
   For all leaves $W_i$ of $T_i$
   for all $k = 1, \ldots, K$ do
      $\gamma_u(k) = \beta_u(k)$
   end for
   for all $j = 1, \ldots, K$ do
      $\gamma_{u,p(u)}(j) = \max_{1 \leq i \leq K} \gamma_u(i)a_{ji}$
      $\xi_u(j) = \arg \max_{1 \leq i \leq K} \gamma_u(i)a_{ji}$
   end for

2. Induction
   for all $s = 1, \ldots, J_0 - 1$ do
      for all nodes $W_i$ at scale $J_0 - s$ do
         for all $k = 1, \ldots, K$ do
            $\gamma_u(k) = P_{h_i}(w_i) \prod_{t \in c(u)} \gamma_{i,t}(k)$
         end for
         for all $j = 1, \ldots, K$ do
            $\gamma_{u,p(u)}(j) = \max_{1 \leq i \leq K} \gamma_u(i)a_{ji}$
            (except at root node)
            $\xi_u(j) = \arg \max_{1 \leq i \leq K} \gamma_u(i)a_{ji}$
         end for
      end for
   end for

3. Termination
   $\hat{P} = \max_{1 \leq i \leq K} \gamma_1(i)$
   $\hat{s}_1 = \arg \max_{1 \leq i \leq K} \gamma_1(i)$

4. “Downward-tracking”
   (creation of the tree from root)
   for all $s = 2, \ldots, J_0$ do
      for all nodes $W_i$ at scale $s$ do
         for all $u = 2, \ldots, n$ do
            $\hat{s}_u = \xi_u(\hat{s}_{p(u)})$
         end for
      end for
   end for
Our method is based on a multiresolution analysis of $x$. As a first step, we compute an orthonormal discrete wavelet transform of $x$, through the following inner product: 
\[(w_n^m)_{1 \leq m \leq J_0, 0 \leq n \leq 2^m - 1}, \text{ with } w_n^m = \sum_{k=1}^{2^m} x_k 2^{m/2} \psi(2^m k - n) \text{ where } J_0 \text{ corresponds to the finest scale. As in Crouse, Nowak and Baraniuk (1998), we adopt a statistical approach to wavelet-based signal processing. This means that we process the signal } x \text{ by operating on its wavelet coefficients } (w_n^m)_{m,n} \text{ and that we consider these coefficients as realizations of random variables } (W_n^m)_{m,n}. \text{ The authors justify the use of a hidden Markov binary tree model for the wavelet coefficients instead of an independent Gaussian model by the two following arguments:}

- the key dependencies between wavelet coefficients are modeled by a latent Markov structure,
- the non-Gaussian nature of the wavelet coefficients is modeled by a mixture of Gaussian distributions.

We recall that the path of a H-fbm has local Hölder regularity $H$ almost surely almost everywhere. Hence from (Jaffard 1991, Flandrin 1992 and Wornell 1992) the random variables $W_n^m$ of its wavelet decomposition are normally, identically distributed and centered with variance:

\[
\text{var}(W_n^m) = \sigma^2 2^{-m(2H+1)}
\]

In our simple test signal, the local regularity being $H_0$ for $1 \leq t \leq T_0$ and $H_1$ for $T_0 + 1 \leq t \leq T$, we consider a two states model with conditional distribution

\[(W_n^m | S_n^m = i) \sim N(0, \sigma_i^2 2^{-m(2H_i+1)})\]

Thus we model the distribution of $(w_n^m)_{m,n}$ by the following hidden Markov tree model:

- $W_n^m$ arise from a mixture of distributions with density

\[
f(W_n^m = w_n^m) = \sum_{i=0}^{1} P(S_n^m = i) f_{\theta_i}(w_n^m)
\]

where $S_n^m$ is a discrete variable with 2 states, denoted $\{0, 1\}$ and $f_{\theta_i}(w_n^m)$ is the Gaussian distribution density with mean 0 and variance $\sigma_i^2 2^{-m(2H_i+1)}$,

- $(S_n^m)_{m,n}$ is a Markov binary tree (i.e. each non-terminal node has exactly two children). Its root node distribution is denoted by $\pi$. In this application, we choose an homogeneous Markov tree model, i.e. a model where the transition matrix $A$ does not depend on $(n,m)$.

- the wavelet coefficients are independent conditionally to the hidden states.

As in Section 2, we denote the observed tree $(W_n^m)_{m,n}$ by $T_1$ and the hidden tree $(S_n^m)_{m,n}$ by $H_1$.

In the case of an abrupt regularity jump at time $T_0$, the hidden tree model $(T_1, H_1)$ satisfies the two following properties:
for each subtree $\mathcal{H}_t$ of $\mathcal{H}_1$, there exists $i$ in $\{0,1\}$ such as the left subtree of $\mathcal{H}_t$ is entirely in state $i$ or its right subtree is entirely in state $i$.

- if $S^h_{t_1}$ and $S^h_{t_2}$ are two leaves with $t_1 < t_2$ such as $S^h_{t_1} = S^h_{t_2} = i$ then for all $t$ between $t_1$ and $t_2$, $S^h_t = i$.

To detect the local regularity jump, we compute the discrete wavelet transform $w_n^\mu$ of the signal using a compact support Daubechies wavelet with regularity 2 over $J_0 = M$ scales. Then we estimate the model parameters by the EM algorithm. The $H_t$ and $\sigma_i$ parameters are estimated at the M step with a procedure adapted from the maximum likelihood estimation by Wornell and Oppenheim (1992). Thus we obtain $A$, $\pi$, $\sigma_0$, $\sigma_1$, $H_0$ and $H_1$. The jump detection is performed by hidden states restoration under the two constraints above, using the Viterbi algorithm. We obtain a value for the hidden tree $\mathcal{H}_1$, such as exactly one subtree $\mathcal{H}_t$ of $\mathcal{H}_1$ is in state $i$ and $\mathcal{H}_1 \setminus t$ is in state $1 - i$. Thus there is only one leave $S^h_{t^*}$ such as $S^h_{t^*} \neq S^h_{t^* + 1}$. The jump time $T_0$ is estimated by:

$$T_0 = 2.0t^*$$

In practice, to avoid a too severe discontinuity in the path at the transition time $T_0$, and to ensure that at any point $t$, the local regularity $H(t)$ is correctly defined, we synthetize a multifractional brownian motion as proposed and defined in (Levy-Vehel and Peltier, 1995), with a continuous transitional Hölder regularity (Figure 3):

$$\forall t \in \{1,\ldots,1024\} \quad H(t) = 0.1 \tanh(-20 + \frac{40(t - 1)}{1023}) + 0.5$$

Thus we consider that $H_0 = 0.4$ and $H_1 = 0.6$. We then construct the process $x = (x(t))_{t=1,\ldots,1024}$ with local regularity given by (14). One realization path of such process is shown in Figure 4 a).

![Figure 3: Local regularity parameter](image-url)
Figure 4 b) shows the result of unconstrained hidden states restoration. Y-axis of the plot represents the tree depth, with root at the bottom line. Figure 4 c) shows the result of constrained hidden states restoration. The border between both states is used to locate the transition time $T_0$ in $H(t)$. The estimated parameters are $\hat{H}_0 = 0.3009$, $\hat{H}_1 = 0.6649$ and $\hat{T}_0 = 520$.

![Graph](image)

**Figure 4:** The signal and the hidden tree associated to wavelet decomposition

Whereas the estimates of $H_0$ and $H_1$ are imprecise and the amplitude of transition is overestimated, due to the few amount of time-samples for each state, the discrimination achieved by our method is satisfactory, concerning the mixture components separation and also in accordance with the performances discussed in Wornell and Oppenheim. It is worth noting that the method used for the estimation of $H_i$ and $\sigma_i$ suffers from the same limitations as the algorithm described in Wornell and Oppenheim (1992). The use of an alternative method to the parameter estimation by likelihood maximization could result in some improvements in the results.
7 Conclusion

In this paper, we propose new efficient tools for hidden Markov tree models identification, inspired by the similarities between hidden Markov trees and hidden Markov chains. The conditional upward-downward algorithm is no substantial improvement in the theory of hidden Markov trees. But it is a computational improvement which allows to dramatically increase the size of the trees which can be handled by the EM algorithm.

The Viterbi algorithm is a more significative contribution for inference in hidden Markov trees, as hidden state restoration is a key point for model interpretation and comprehension. Although the aim of this paper is not to discuss about the interest of hidden Markov tree modeling for wavelet analysis in signal processing, we show an example where hidden state restoration allows discrimination between two regimes of a simulated signal.

In this direction, we could tackle another big issue inherent to scaling law estimation. In most real world application the scaling law (self-similarity, long range dependance,...) is satisfied only within some finite scale range (also called inertial range). Aside this interval, not only the scale parameter $H$ can change, but also the whole model can fail to apply. It is therefore crucial to identify this inertial range, as it provides a valuable information on the underlying physical system. In our framework, we could take advantage of the transition probabilities between states, estimated on the data, to first automatically determine a plausible scale interval on which the power-law variance model is valid, and second to provide us with a confidence factor assessing accuracy on the estimated value of $H$.

On the other hand, it could be possible to exploit the similarities between hidden Markov tree models and stochastic context-free grammars (Lari and Young, 1990; Jelinek and Mercer, 1992) and adapt our conditional algorithm to tackle the computational difficulties for these models.

In conclusion, our work is a contribution to efficient parameter estimation for hidden Markov tree models.

Acknowledgments

The authors acknowledge helpful advice and discussion about inference algorithms in mixture models with Gilles Celeux. They are most grateful to Yann Guédon for all his valuable remarks on this work.
Appendices

As a first step, we prove by induction in Appendix A that the \( \tilde{\beta} \) variables defined by the conditional upward-downward algorithm satisfy equations (4)–(6). Thus, the \( \tilde{\beta} \) variables are proved to be conditional probabilities. In Appendix B we show that the \( \tilde{\alpha} \) variables satisfy equation (7). The proof of equation (9) is given in Appendix C. These equations allow the loglikelihood computation of a parameter. Appendix D contains the proof that algorithm 6 gives the hidden tree with highest conditional probability.

A Up step

We first prove by induction on the depth of \( \mathcal{T}_u \) that the \( \tilde{\beta} \) variables defined by algorithm 4 are equal to the following conditional probabilities

\[
\tilde{\beta}_u(k) = P(S_u = k|\mathcal{T}_u) \quad \text{and} \quad \tilde{\beta}_{u,\rho(u)}(k) = P(S_{\rho(u)} = k|\mathcal{T}_u) \tag{15}
\]

Let \( J_0 \) be the depth of \( \mathcal{T}_1 \) with the convention that the root of a tree is at depth 1 and the leaves at depth \( J_0 \) (see Figure 1).

**Induction assumption**: for each \( r \leq j \) and for each node \( W_u \) of \( \mathcal{T}_1 \) at depth \( J_0 - r \)

\[
\tilde{\beta}_u(k) = P(S_u = k|\mathcal{T}_u) \tag{16}
\]

**Proof for \( j = 0 \)**

For all leaves \( W_u \) of \( \mathcal{T}_1 \), by definition of \( \tilde{\beta}_u(k) \) we have

\[
P(S_u = k|\mathcal{T}_u) = \frac{P(S_u = k, W_u = w_u)}{P(W_u = w_u)} = \frac{P(W_u = w_u|S_u = k)P(S_u = k)}{\sum_{l=1}^{K} P(W_u = w_u|S_u = l)P(S_u = l)} = \tilde{\beta}_u(k)
\]

**Induction**

For all remaining nodes \( W_u \) of \( \mathcal{T}_1 \),

\[
P(S_{\rho(u)} = k|\mathcal{T}_u) = \frac{\tilde{\beta}_{u,\rho(u)}(k)P(S_{\rho(u)} = k)}{P(\mathcal{T}_u)} \tag{17}
\]

\[
= \left[ \sum_{i=1}^{K} P(\mathcal{T}_u|S_u = i)a_{ki} \right] \frac{P(S_{\rho(u)} = k)}{P(\mathcal{T}_u)} = \left[ \sum_{i=1}^{K} P(S_u = i|\mathcal{T}_u) \frac{a_{ki}}{P(S_u = i)} \right] P(S_{\rho(u)} = k) \tag{18}
\]

RR n° 4248
\[
\sum_{i=1}^{K} \tilde{\beta}_u(i) \frac{a_{ki}}{P(S_u = i)} P(S_{p,u} = k) = \tilde{\beta}_{u,p(u)}(k) \\
\]

(19)

as \(\beta_{u,p(u)}(k) = P(T_u|S_{p,u} = k)\) by algorithm 1, whence equation (17). Equation (18) also comes from algorithm 1. Equation (19) comes from our induction assumption (16) and equation (20) from algorithm 4. Now

\[
P(S_u = k|T_u) = \frac{\beta_u(k)P(S_u = k)}{P(T_u)} \\
= \frac{P_{\bar{\theta}_u}(w_u)P(S_u = k)\prod_{t=1}^{n_u} \prod_{t=1}^{n_c} P(T_u^t|S_u = k)}{P(T_u)\prod_{t\in c(u)} P(S_u = k)^{n_u-1}} \\
= \frac{P_{\bar{\theta}_u}(w_u)}{P(T_u)}P(S_u = k)^{n_u-1} \prod_{t\in c(u)} \tilde{\beta}_{t,u}(k) \prod_{t\in c(u)} P(T_u^t) \\
\]

(23)

Equations (21) and (22) come from algorithm 1. Equation (23) results from (20) and assumption (16). Moreover, as \(\sum_{t=1}^{K} P(S_u = k|T_u) = 1\) we have

\[
\prod_{t\in c(u)} \frac{P(T_u^t)}{P(T_u)} = \left[\sum_{t=1}^{K} \frac{P_{\bar{\theta}_u}(w_u)}{P(S_u = k)^{n_u-1}} \prod_{t\in c(u)} \tilde{\beta}_{t,u}(l) \right]^{-1} \\
= [M_u]^{-1} \text{ by definition.} \\
\]

(24)

It follows from equation (23) that:

\[
P(S_u = k|T_u) = \frac{P_{\bar{\theta}_u}(w_u)}{M_u P(S_u = k)^{n_u-1}} \prod_{t\in c(u)} \tilde{\beta}_{t,u}(k) = \tilde{\beta}_u(k) \\
\]

As a result we have:

\[
\forall u \in \{1, \ldots, n\} \quad \forall k \in \{1, \ldots, K\} \\
\tilde{\beta}_u(k) = P(S_u = k|T_u) \text{ and } \tilde{\beta}_{u,p(u)}(k) = P(S_{p,u} = k|T_u) \\
\]

which justifies the up step of the conditional upward-downward algorithm.

**B Down step**

We now prove by induction that the \(\tilde{\alpha}\) variables defined by algorithm 5 satisfy the following assumption.
Induction assumption: for each node $W_u$ of $T_1$ at depth $j$,

$$\tilde{\alpha}_u(k) = \frac{P(T_1 \setminus u | S_u = k)}{P(T_1)}$$  \hspace{1cm} (26)

As $\alpha_u(k) = P(T_1 \setminus u, S_u = k)$ by algorithm 2, we notice that this assumption is equivalent to:

$$\tilde{\alpha}_u(k) = \frac{P(T_1 \setminus u | S_u = k)P(T_u)}{P(T_1)} = \frac{\alpha_u(k)P(T_u)}{P(S_u = k)P(T_1)}$$

**Proof for $j = 1$**

$\forall k \in \{1, \ldots, K\}$ we have by algorithm 2:

$$\frac{\alpha_1(k)P(T_1)}{P(S_1 = k)P(T_1)} = \frac{\pi_k}{\pi_k} = 1 = \tilde{\alpha}_1(k)$$

**Induction**

By the standard down step (cf. algorithm 2) we have

$$\alpha_u(k) = \sum_{i=1}^{K} a_{ik} \beta_{\rho(u) \setminus u}(i) \alpha_{\rho(u)}(i)$$  \hspace{1cm} (27)

where by definition of $\beta_{\rho(u) \setminus u}(i)$ in algorithm 2

$$\beta_{\rho(u) \setminus u}(i) = P(T_{\rho(u)} \setminus u | S_{\rho(u)} = i)$$

$$= \beta_{\rho(u)}(i)$$  \hspace{1cm} (28)

$$= \frac{\beta_{\rho(u)}(i)P(T_{\rho(u)})}{P(T_u)}$$  \hspace{1cm} (29)

$$= \beta_{\rho(u) \setminus u}(i)\frac{P(T_{\rho(u)})}{P(T_u)}$$  \hspace{1cm} (30)

Equation (28) comes from the standard up step (cf. algorithm 1). Equation (29) is a consequence of equations (15), (17) and (21). Equation (30) comes from the definition of $\beta_{\rho(u) \setminus u}(i)$. It follows from algorithm 5 that

$$\tilde{\alpha}_u(k) = \sum_{i=1}^{K} a_{ik} \tilde{\alpha}_{\rho(u)}(i) \tilde{\beta}_{\rho(u) \setminus u}(i) \frac{P(S_{\rho(u)} = i)}{P(S_u = k)}$$

$$= \sum_{i=1}^{K} a_{ik} \frac{P(T_{1 \setminus u} | S_{\rho(u)} = i)P(T_{\rho(u)})}{P(T_1)} \beta_{\rho(u) \setminus u}(i) \frac{P(T_u)}{P(T_{\rho(u)})} \frac{P(S_{\rho(u)} = i)}{P(S_u = k)}$$  \hspace{1cm} (31)
\[
\begin{align*}
&= \frac{P(T_u)}{P(T_1)P(S_u = k)} \sum_{i=1}^{K} a_{ik} P(T_{1\backslash u}, S_{\rho(u)} = i) \beta_{\rho(u)\backslash u}(i) \\
&= \frac{P(T_u)}{P(T_1)P(S_u = k)} \alpha_u(k) 
\end{align*}
\]

(32)

We used our induction assumption (26) and equation (30) to derive (31). Equation (32) comes from (27).

It follows that

\[
\forall u \in \{1, \ldots, n\}, \quad \forall k \in \{1, \ldots, K\}
\]

\[
\tilde{\alpha}_u(k) = \frac{P(T_{1\backslash u} \mid S_u = k)}{P(T_{1\backslash u} \mid T_u)}
\]

which completes the induction, hence the justification for the down step of the conditional upward-downward algorithm.

C E step

The aim of this appendix is to prove equation (9). The proof establishes that

\[
l_u = \log(P(T_u)) - \sum_{W_i \in F(T_u)} \log(P(W_i))
\]

for all non-terminal nodes.

Likelihood computation

Recall that \( F(T_u) \) denotes the set of observed variables located in the leaves of \( T_u \) (see Figure 1). The equation

\[
\log(P(T_u)) = l_u + \sum_{W_i \in F(T_u)} \log(P(W_i))
\]

is proved by induction on the depth of \( T_u \).

Induction assumption: for each node \( W_u \) of \( T_1 \) at depth \( J_0 - j \)

\[
l_u = \log(P(T_u)) - \sum_{W_i \in F(T_u)} \log(P(W_i))
\]

(33)

Proof for \( j = 1 \)

For each node \( W_u \) of \( T_1 \) at depth \( J_0 - 1 \), by construction,

\[
l_u = \log(M_u) + \sum_{t \in c(u)} l_t = \log(M_u)
\]

as \( l_u = 0 \) for the leaves. Thus by definition of \( M_u \)

\[
l_u = \log(P(T_u)) - \sum_{t \in c(u)} \log(P(T_t)) = \log(P(T_u)) - \sum_{W_i \in F(T_u)} \log(P(W_i))
\]

INRIA
**Induction:** if assumption (33) is satisfied for each node at depth $J_0 - j$, then for each node $W_u$ at depth $J_0 - (j + 1)$, it follows directly from equation below

$$\forall t \in c(u) \quad l_t = \log(P(T_t)) - \sum_{W_t \in \mathcal{F}(T_t)} \log(P(W_t))$$

and from the definition of $l_u$ and $\mathcal{M}_u$ (see algorithm 5 and equation (25)) that $l_u = \log(P(T_u)) - \sum_{W_t \in \mathcal{F}(T_t)} \log(P(W_t))$ for node $u$ and therefore for all non-terminal nodes.

## D MAP algorithm

This appendix proves that the hidden tree resulting from algorithm 6 is maximizing $P(\mathcal{H}_1 = h_1 | T_1)$. As for hidden Markov chains, the idea of the proof is to consider the $\delta$ variables used in this algorithm as maximal conditional probabilities of subtrees of $T_1$. Moreover the algorithm must keep a map of the path run along. This is achieved by the $\psi$ function which gives the optimal state of a hidden node again when its parent’s state is known. This interpretation of our MAP algorithm is justified below.

We recall that for any node $u$, $\mathcal{H}_u$ denotes the hidden subtree rooted in $S_u$ and $\left( \mathcal{H}_u^1, \ldots, \mathcal{H}_u^k \right)$ denotes the subtrees of $\mathcal{H}_u$. We first assert that the $\delta_u$ variables maximize the conditional probabilities of a given hidden tree by maximising the states of its children. This is formally expressed and then proved by induction on the depth of $T_u$ by the following proposition.

**Proposition 1** If $u$ is not a leaf of $T_u$ then $\forall k \in \{1, \ldots, K\}$

$$\delta_u(k) = \max_{h_{c_1}^u, \ldots, h_{c_n}^u} P(\mathcal{H}_{c_1}^u = h_{c_1}^u, \ldots, \mathcal{H}_{c_n}^u = h_{c_n}^u, S_u = k | T_u)$$

$$\delta_{u, \rho(u)}(k) = \max_{h_u} P(\mathcal{H}_u = h_u, S_{\rho(u)} = k | T_u)$$

**Induction assumption:** for each node $W_u$ of $T_1$ at depth $J_0 - j$

$$\forall k \in \{1, \ldots, K\} \quad \delta_u(k) = \max_{h_{c_1}^u, \ldots, h_{c_n}^u} P(\mathcal{H}_{c_1}^u = h_{c_1}^u, \ldots, \mathcal{H}_{c_n}^u = h_{c_n}^u, S_u = k | T_u) \quad (34)$$

**Proof for $j = 1$**

For each node $W_u$ of $T_1$ at depth $J_0$, by construction,

$$\delta_{u, \rho(u)}(k) = \max_{1 \leq i \leq K} \left[ P(S_u = i | W_u) \frac{P(S_u = i | S_{\rho(u)} = k)}{P(S_u = i)} \right] P(S_{\rho(u)} = k)$$

$$= \max_{1 \leq i \leq K} P(W_u | S_u = i) P(S_{\rho(u)} = k) \frac{P(S_u = i, S_{\rho(u)} = k)}{P(W_u)}$$

$$= \max_{1 \leq i \leq K} \frac{P(W_u | S_u = i, S_{\rho(u)} = k)}{P(W_u)} (35)$$

$$= \max_{1 \leq i \leq K} P(S_u = i, S_{\rho(u)} = k | W_u) = \max_{h_u} P(\mathcal{H}_u = h_u, S_{\rho(u)} = k | T_u) \quad (36)$$
Equation (35) is a consequence of conditional independence properties of HMT. The depth of node $W_{\rho(u)}$ is $J_0 - 1$ and it follows from (36) and from the definition of $\delta_{\rho(u)}(k)$ that

$$
\delta_{\rho(u)}(k) = \frac{P(W_{\rho(u)}|S_{\rho(u)} = k)}{M_{\rho(u)} P(S_{\rho(u)} = k)^{n_{\rho(u)} - 1}} \prod_{t=1}^{n_{\rho(u)}} \max_{j_t \leq k} P(S_{\rho(u)} = j_t, S_{\rho(u)} = k|W_{\rho(u)}^t)
$$

Using the definition of $M_{\rho(u)}$ and conditional independence properties, we obtain:

$$
\frac{P(W_{\rho(u)}|S_{\rho(u)} = k)}{M_{\rho(u)} P(S_{\rho(u)} = k)^{n_{\rho(u)} - 1}} \prod_{t=1}^{n_{\rho(u)}} \max_{j_t \leq k} P(S_{\rho(u)} = j_t, S_{\rho(u)} = k|W_{\rho(u)}^t)

= P(S_{\rho(u)} = j_1, ..., S_{\rho(u)} = j_{n_{\rho(u)}}, S_{\rho(u)} = k|\mathcal{T}_{\rho(u)})
$$

Hence

$$
\delta_{\rho(u)}(k) = \max_{h_{c_1}^{\rho(u)}, ..., h_{c_{u+1}}^{\rho(u)}} P(\mathcal{H}_{c_1}^{\rho(u)} = h_{c_1}^{\rho(u)}, ..., \mathcal{H}_{c_{u+1}}^{\rho(u)} = h_{c_{u+1}}^{\rho(u)}, S_{\rho(u)} = k|\mathcal{T}_{\rho(u)})
$$

**Induction**: if assumption (34) is satisfied for each node at depth $J_0 - j$, then for each node $W_u$ at depth $J_0 - j$, by construction,

$$
\delta_{u,\rho(u)}(k) = \max_{1 \leq i \leq K} \left[ \delta_u(i) \frac{P(S_u = i|S_{\rho(u)} = k)}{P(S_u = i)} \right] P(S_{\rho(u)} = k)

= \max_{1 \leq i \leq K} \max_{h_{c_1}^{u}, ..., h_{c_{u+1}}^{u}} P(\mathcal{H}_{c_1}^{u} = h_{c_1}^{u}, ..., \mathcal{H}_{c_{u+1}}^{u} = h_{c_{u+1}}^{u}, S_u = i|\mathcal{T}_u) P(S_{\rho(u)} = k|S_u = i)
$$

by induction assumption (34)

It follows from conditional independence properties that

$$
P(\mathcal{H}_{c_1}^{u} = h_{c_1}^{u}, ..., \mathcal{H}_{c_{u+1}}^{u} = h_{c_{u+1}}^{u}, S_u = i|\mathcal{T}_u) P(S_{\rho(u)} = k|S_u = i)

= P(\mathcal{H}_{c_1}^{u} = h_{c_1}^{u}, ..., \mathcal{H}_{c_{u+1}}^{u} = h_{c_{u+1}}^{u}, S_u = i, S_{\rho(u)} = k|\mathcal{T}_u)
$$

Hence $\delta_{u,\rho(u)}(k) = \max P(\mathcal{H}_u = h_{u}, S_{\rho(u)} = k|\mathcal{T}_u)$ and it follows from the definitions of $\delta_{\rho(u)}(k)$ and $M_{\rho(u)}$ that:

$$
\delta_{\rho(u)}(k) = \prod_{h_u \in h_u} \delta_{u,\rho(u)}(k)

= \prod_{h_u \in h_u} \frac{P_{\theta_u}(w_{\rho(u)})}{P(\mathcal{T}_{\rho(u)}) P(S_{\rho(u)} = k)^{n_{\rho(u)} - 1}} \prod_{t \in (\rho(u))} P(\mathcal{T}_t) \prod_{t \in (\rho(u))} \max_{h_t} P(\mathcal{H}_t = h_t, S_{\rho(u)} = k|\mathcal{T}_t)
$$

It results from conditional independence properties that

$$
\prod_{h_u \in h_u} \frac{P_{\theta_u}(w_{\rho(u)})}{P(\mathcal{T}_{\rho(u)}) P(S_{\rho(u)} = k)^{n_{\rho(u)} - 1}} \prod_{t \in (\rho(u))} P(\mathcal{T}_t) \prod_{t \in (\rho(u))} \max_{h_t} P(\mathcal{H}_t = h_t, S_{\rho(u)} = k|\mathcal{T}_t)

= P(\mathcal{H}_{c_1}^{u} = h_{c_1}^{u}, ..., \mathcal{H}_{c_{u+1}}^{u} = h_{c_{u+1}}^{u}, S_{\rho(u)} = k|\mathcal{T}_u)
$$

INRIA
Thus
\[
\delta_{\rho(u)}(k) = \max_{\mu_{c_1}, \ldots, \mu_{c_{\rho(u)}}} P(\mathcal{H}_{c_1}^{\rho(u)} = h_{c_1}, \ldots, \mathcal{H}_{c_{\rho(u)}}^{\rho(u)} = h_{c_{\rho(u)}}, S_{\rho(u)} = k | T_{\rho(u)})
\]

This completes the proof of the assumption (34) for all nodes of \( T_1 \). The following property
\[
\delta_{u, \rho(u)}(k) = \max_{\mu_{u}} P(\mathcal{H}_{u} = h_{u}, S_{\rho(u)} = k | T_{u})
\]
shows that \( \delta_{u, \rho(u)}(k) \) is maximizing the joint conditional probability of the whole hidden subtree and its root when the observed subtree \( T_u \) is known.

The fact that the states tree \((\hat{s}_1, \ldots, \hat{s}_n)\) defined by the MAP algorithm is optimal remains to be shown. Let \( \hat{h}_1 = (\hat{s}_1, \ldots, \hat{s}_n) \) be a hidden tree such as \( P(\mathcal{H}_1 = \hat{h}_1 | T_1) \) (or equivalently \( P(\mathcal{H}_1 = \hat{h}_1, T_1) \)) is maximal. Then \( \forall u \in \{1, \ldots, n\} \), a characterization of \( s_u \) is:
\[
s_u = \arg \max_{1 \leq i \leq K} P(\mathcal{H}_{c_1}^{u} = h_{c_1}, \ldots, \mathcal{H}_{c_{\rho(u)}}^{u} = h_{c_{\rho(u)}}, S_{u} = s_u, \mathcal{H}_{1 \setminus u} = \hat{h}_{1 \setminus u}, T_1)
\]

(37)
The following property: \( \forall u \in \{1, \ldots, n\} \quad s_u = \hat{s}_u \), is proved by an induction on the depth of \( T_u \).

Induction assumption :
for each node \( W_u \) of \( T_1 \) at depth \( j \) \( s_u = \hat{s}_u \)
(38)

Proof for \( j = 1 \)
From the definition of \( \hat{s}_1 \) and by property (34), we have:
\[
\hat{s}_1 = \arg \max_{1 \leq i \leq K} \delta_1(i) = \arg \max_{1 \leq i \leq K} P(\mathcal{H}_{c_1}^{1} = h_{c_1}^1, \ldots, \mathcal{H}_{c_{\rho(1)}}^{1} = h_{c_{\rho(1)}}^1, S_1 = \hat{s}_1 | T_1) = s_1
\]

Induction :
Let \( T_{b(u)} \) be the set of subtrees of \( T_{\rho(u)} \) such as: \( T_{b(u)} \cup T_u \cup W_{\rho(u)} = T_{\rho(u)} \) (see Figure 1). Then for each node \( W_u \) of \( T_1 \) at depth \( j + 1 \),
\[
s_u = \arg \max_{1 \leq i \leq K} \max_{\mu_{c_1}, \ldots, \mu_{c_{\rho(u)}}} P(\mathcal{H}_{c_1}^{u} = h_{c_1}, \ldots, \mathcal{H}_{c_{\rho(u)}}^{u} = h_{c_{\rho(u)}}, S_u = k, \mathcal{H}_{1 \setminus \mu} = \hat{h}_{1 \setminus \mu}, T_1)
\]
by characterization (37). From conditional independance properties of HMT and the decompositions :
\[
T_1 = T_u \cup T_{1 \setminus u} \quad \text{and} \quad \mathcal{H}_{1 \setminus u} = \mathcal{H}_{1 \setminus \rho(u)} \cup \mathcal{H}_{b(u)} \cup \{S_{\rho(u)}\}
\]
we derive the following equation
\[
P(\mathcal{H}_{c_1}^{u} = h_{c_1}, \ldots, \mathcal{H}_{c_{\rho(u)}}^{u} = h_{c_{\rho(u)}}, S_u = k, \mathcal{H}_{1 \setminus u} = \hat{h}_{1 \setminus u}, T_1)
= P(\mathcal{H}_{c_1}^{u} = h_{c_1}, \ldots, \mathcal{H}_{c_{\rho(u)}}^{u} = h_{c_{\rho(u)}}, S_u = i, S_{\rho(u)} = \hat{s}_{\rho(u)}, T_u)
\]
\[
P(Y_{1 \setminus \rho(u)} = \hat{h}_{1 \setminus \rho(u)}, \mathcal{H}_{b(u)} = \hat{h}_{b(u)}, T_{1 \setminus u} | S_{\rho(u)} = \hat{s}_{\rho(u)})
\]
This equation results from recurrence assumption (38) and the second factor of its right hand side neither depends on \(i\) nor on the \(h_{c_i}^u\). Thus

\[
\sigma_u = \arg \max_{1 \leq i \leq K} \max_{h_{c_i}^u, \ldots, h_{c_n}^u} \mathbf{P}(\mathcal{H}_{c_1}^u = h_{c_1}^u, \ldots, \mathcal{H}_{c_n}^u = h_{c_n}^u, S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)} | \mathcal{T}_u)
\]

Now, using conditional independance properties, let us factorize the above equation to make \(\delta_u(i)\) appear

\[
P(\mathcal{H}_{c_1}^u = h_{c_1}^u, \ldots, \mathcal{H}_{c_n}^u = h_{c_n}^u, S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)} | \mathcal{T}_u)
\]

\[
= P(S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)} | \mathcal{T}_u) \prod_{i=1}^{n_u} P(\mathcal{H}_{c_i}^u = h_{c_i}^u | S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)}, \mathcal{T}_u)
\]

\[
= \frac{P(S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)}, \mathcal{T}_u)}{P(S_u = i)} \prod_{i=1}^{n_u} P(\mathcal{H}_{c_i}^u = h_{c_i}^u | S_u = i, \mathcal{T}_u) \tag{39}
\]

where

\[
P(S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)}, \mathcal{T}_u)
\]

\[
= P(\mathcal{T}_u | S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)}) P(S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)})
\]

\[
= \frac{P(\mathcal{T}_u, S_u = i)}{P(S_u = i)} P(S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)}) \tag{40}
\]

Equations (39) and (40) result from conditional independance properties. Combination of equations (39) and (40) gives:

\[
P(\mathcal{H}_{c_1}^u = h_{c_1}^u, \ldots, \mathcal{H}_{c_n}^u = h_{c_n}^u, S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)} | \mathcal{T}_u)
\]

\[
= \frac{P(S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)})}{P(S_u = i)} P(S_u = i) \prod_{i=1}^{n_u} P(\mathcal{H}_{c_i}^u = h_{c_i}^u | S_u = i, \mathcal{T}_u)
\]

\[
= \frac{P(S_u = i, S_{\rho(u)} = \hat{\delta}_{\rho(u)})}{P(S_u = i)} P(\mathcal{H}_{c_1}^u = h_{c_1}^u, \ldots, \mathcal{H}_{c_n}^u = h_{c_n}^u, S_u = i | \mathcal{T}_u)
\]

the last equation resulting from conditional independance properties. Hence, by characterization (34) of \(\delta_u(i)\) and by definition of \(\sigma_u\), we have:

\[
\sigma_u = \arg \max_{1 \leq i \leq K} \max_{h_{c_1}^u, \ldots, h_{c_n}^u} \mathbf{P}(\mathcal{H}_{c_1}^u = h_{c_1}^u, \ldots, \mathcal{H}_{c_n}^u = h_{c_n}^u, S_u = i | \mathcal{T}_u)
\]

\[
= \arg \max_{1 \leq i \leq K} \max_{h_{c_1}^u, \ldots, h_{c_n}^u} \left[ \delta_u(i) \cdot \frac{a_{\sigma(u), i}}{P(S_u = i)} \right] P(S_{\rho(u)} = \hat{\delta}_{\rho(u)})
\]

\[
= \psi_u(\hat{\delta}_{\rho(u)}) = \sigma_u
\]

This shows the optimality of the tree resulting from algorithm 6, which is thus proved to be the MAP algorithm.
References


