Marginalized Viterbi Algorithm for Hierarchical Hidden Markov Models

Akira Hayashi[a,*], Kazunori Iwata[a], Nobuo Suematsu[a]

[a] *Graduate School of Information Sciences, Hiroshima City University, 3–4–1 Ozuka-Higashi, Asa-Minami-Ku, Hiroshima, 731–3194 Japan*

Abstract

The generalized Viterbi algorithm, a direct extension of the Viterbi algorithm for hidden Markov models (HMMs), has been used to find the most likely state sequence for hierarchical HMMs. However, the generalized Viterbi algorithm finds the most likely whole level state sequence rather than the most likely upper level state sequence. In this paper, we propose a marginalized Viterbi algorithm, which finds the most likely upper level state sequence by marginalizing lower level state sequences. We show experimentally that the marginalized Viterbi algorithm is more accurate than the generalized Viterbi algorithm in terms of upper level state sequence estimation.

Keywords: time series data, hierarchical HMM, finding the most likely state sequence, generalized Viterbi algorithm, marginalized Viterbi algorithm

1. Introduction

Hidden Markov models (HMMs) [1], known for their success in voice recognition, have been widely used to analyze time series data. Fine et al. [2] proposed hierarchical hidden Markov models (HHMMs) as a generalization of HMMs with a hierarchical state space. An HHMM may be represented using a tree structure, where each state at a non-leaf node, called an internal state, is itself a dynamical probabilistic model. Therefore, the internal states of an HHMM emit sequences rather than a single symbol. An HHMM generates sequences by recursive activation of a substate of an internal state, until a leaf node state, called a production state, is reached. Production states are the only states that actually output symbols through the usual HMM mechanism. The original inference algorithm for HHMMs is not efficient, taking $O(T^3)$ time where T is the length of the observation sequence. Murphy et al. [3] devised a dynamic

*Corresponding author

Email addresses: akira@hiroshima-cu.ac.jp (Akira Hayashi), kiwata@hiroshima-cu.ac.jp (Kazunori Iwata), suematsu@hiroshima-cu.ac.jp (Nobuo Suematsu)

Bayesian network (DBN) representation for HHMMs, thanks to which a linear time ($O(T)$) inference algorithm is now available.

HHMMs can naturally represent the multiple time scale structure of many time series data (for example, voice has three time scale structures: word sequence, phone sequence, and sub-phone sequence), and are gaining much attention in the research community. Some of the applications of HHMMs are hand written character recognition [2], information extraction from texts [4], and video analysis [5, 6].

The problem of finding the most likely state sequence from an observation sequence [1] is important and has many applications. To find the most likely state sequence for HHMMs, the generalized Viterbi algorithm (GVA) [2, 3], a direct extension of the Viterbi algorithm for HMMs [7, 1], has been used. However, GVA finds the most likely whole level state sequence, but not the most likely upper level state sequence.

In this paper, we propose a marginalized Viterbi algorithm (MVA) to overcome the problem associated with GVA. MVA finds the most likely upper level state sequence by marginalizing lower level state sequences. For example, MVA will find the most likely sequence of "word" states in speech recognition by marginalizing the irrelevant "phone" and "sub-phone" state sequences, thus avoiding the problems associated with words having several pronunciations [8] [1]

To explain our motivation for marginalizing irrelevant lower level states, consider the simple two level static hierarchical model in Fig. 1. The model can be seen as a Gaussian mixture speaker model for speaker identification [9], where the top level state, q^1, stands for a speaker s, and the second level state, q^2, stands for a component c of the Gaussian mixture model:

$$\begin{cases} p(q^2 = c|q^1 = s) = \pi_c^s \\ p(o = \mathbf{x}|q^1 = s, q^2 = c) = \mathcal{N}(\mathbf{x}|\mu_c^s, \mathbf{\Sigma}_c^s) \end{cases}, \tag{1}$$

where $\pi_c^s \geq 0$ is the weight of c and satisfies $\sum_c \pi_c^s = 1.0$, and $\mathcal{N}(\mathbf{x}|\mu_c^s, \mathbf{\Sigma}_c^s)$ is a Gaussian density with mean vector μ_c^s and covariance matrix $\mathbf{\Sigma}_c^s$. Given an observation $o = \mathbf{x}$, the most likely estimation for the speaker identification is

$$\hat{s} = \underset{s}{\operatorname{argmax}}\, p(q^1 = s|o = \mathbf{x}), \tag{2}$$

where $p(q^1 = s|o = \mathbf{x})$ is obtained by

$$p(q^1 = s|o = \mathbf{x}) = \sum_c p(q^1 = s, q^2 = c|o = \mathbf{x}), \tag{3}$$

that is, by marginalizing q^2, an irrelevant second level state.

[1] MVA cannot, however, find the most likely word sequence since it does not marginalize over word segmentation boundaries.

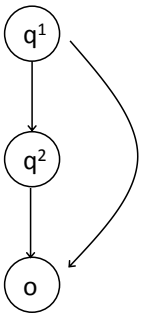

Figure 1: two level static hierarchical model

In this paper, we also propose a fast approximation algorithm for MVA. We show using experiments that MVA is more accurate than GVA in terms of upper level state sequence estimation.

MVA was developed in our lab and first introduced in [10] as a conference proceedings paper with a limited audience. The main theme of the previous paper is not MVA, but HHCRFs, discriminative models corresponding to HHMMs. Most of the results presented in the current paper are new, including the fast approximation algorithm for MVA and the detailed comparison of GVA and MVA through experiments.

Our paper is organized as follows. We explain HHMMs in Section 2. We then explain GVA and MVA in Section 3. In Section 4, we compare the performances of GVA and MVA through experiments. We summarize the paper in Section 5.

2. HHMMs

An HHMM may be represented using a tree structure, and generates sequences by recursive activation of a substate of a non-leaf node until a leaf node state, called a production state, is reached. Production states are the only states that actually emit output symbols. The original inference algorithm for HHMMs is not efficient, taking $O(T^3)$ time. Murphy et al. [3] devised a dynamic Bayesian network (DBN) representation for HHMMs, thanks to which a linear time ($O(T)$) inference algorithm is now available.

2.1. Overview of HHMMs

An HHMM is represented as a tree structure as shown in Figure 2. The circles, trapezoids, and rectangles in the figure stand for internal states, production states, and end states, respectively. The arrows connecting the states represent state transitions. A solid line indicates a horizontal transition to a state within the same level, a broken line indicates a vertical transition to a child state in the next level, and a dotted line indicates a transition to an end state, after which control is returned to the calling parent state. The state at

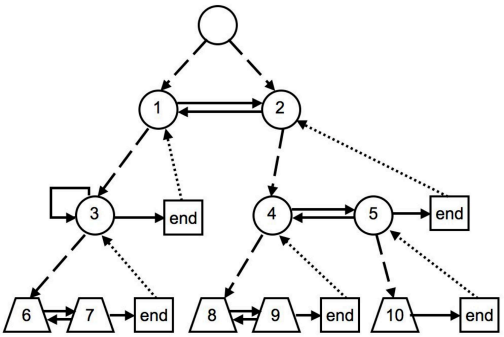

Figure 2: Example of an HHMM with a three-level hierarchy.

the top of the hierarchy is called the root node. The level for the root node is 0, and a sequence of state transitions starts at the root state.

An HHMM generates a sequence of observations as follows.

(Step 1) Start: we start from the root node at time $t = 1$.

(Step 2) Vertical transition : a transition occurs from the current state (an internal state) to a child state in the lower level. If the destination is an internal state, further transitions to lower level states occur until a production state is reached.

(Step 3) Output symbol emission: the production state emits an output symbol o_t. Time t is incremented by 1.

(Step 4) Horizontal transition: a transition to a state within the same level occurs. If the destination is an internal state, we go back to Step 2, and if the destination is a production state, we go back to Step 3. If the destination is an end state, we proceed to Step 5.

(Step 5) Forced transition: A forced transition occurs to the upper level parent state which has initiated the current level state transitions, and we go back to Step 4.

Fine et al. [2], as well as proposing HHMMs, developed an algorithm for state estimation on the basis of the inside-outside algorithm. This algorithm is not efficient, however, and the time for state estimation and also for the most likely state sequence estimation is $O(T^3)$, where T is the length of the observation sequence.

2.2. Representing HHMMs as DBNs

Murphy et al. [3] devised a dynamic Bayesian network (DBN) representation for HHMMs. A Bayesian network (BN) is a directed acyclic graph representing conditional independence relationships between random variables, and a DBN

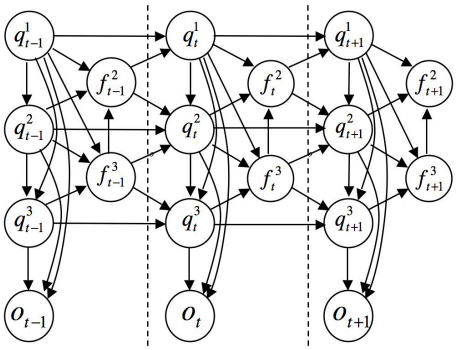

Figure 3: DBN representation of a three-level HHMM, which draws state-transitions from time $t-1$ to $t+1$.

is an extension of a BN to a random process, where the random variables are dependent on time t. Thanks to the DBN representation, linear time ($O(T)$) algorithms for state estimation and the most likely state sequence estimation have become available.

We show a DBN representation of a three-level HHMM in Figure 3. (We assume for simplicity that all production states are in the bottom level of the hierarchy.) The random variable o_t in the figure stands for the output from a production state at time t ($t = 1, \ldots, T$). The output of an HHMM can be either discrete or continuous, but we consider the case of discrete symbol output in this paper. The state of the HHMM in level d and at time t is denoted by q_t^d ($d \in \{1, \ldots, D\}$), where d is the hierarchy index: the top level has $d = 1$, and the bottom level has $d = D$.

f_t^d is an indicator variable which is equal to 1 if q_t^d has transitioned to its end state, and is 0 otherwise. The indicator variables play an important role in representing an HHMM as a DBN. As we explained in the previous subsection, a transition to an end state leads to a state transition in the upper level. In other words, $f_t^d = 1$ implies a possible state change in level $d-1$. In addition, if $f_t^d = 1$ then $f_t^{d'} = 1$ for all $d' > d$; hence the number of indicator variables that equal 0 denotes the level of the hierarchy we are currently in.

We now explain the state transition probabilities and the discrete output probabilities of an HHMM. The set of these probabilities constitutes the model parameters of an HHMM and completely defines the HHMM. Note that $f_t^d = 1$ implies not only that q_{t+1}^{d-1}, the state in level $d-1$ at time $t+1$, may change from q_t^{d-1} as mentioned above, but also that the value for q_{t+1}^d, the state in level d at time $t+1$, is determined by a vertical transition. We show below the state transition probabilities and the discrete output probabilities of an HHMM:

$$p(q_t^d{=}j'|q_{t-1}^d{=}j, f_{t-1}^{d+1}{=}b, f_{t-1}^d{=}f, q_t^{1:d-1}{=}\vec{i}) = \begin{cases} \delta(j=j') & \text{if } b=0, \\ A_{\vec{i}}^d(j,j') & \text{if } b=1 \text{ and } f=0, \\ \pi_{\vec{i}}^d(j') & \text{if } b=1 \text{ and } f=1, \end{cases} \tag{4}$$

$$p(f_t^d=1|q_t^d=j, q_t^{1:d-1}=\vec{i}, f_t^{d+1}=b) = \begin{cases} 0 & \text{if } b=0, \\ Ae_{\vec{i}}^d(j) & \text{if } b=1, \end{cases} \tag{5}$$

$$p(o_t=k|q_t^{1:D}=\vec{i}) = B_{\vec{i}}(k), \quad 1 \le k \le K, \tag{6}$$

where $q_t^{1:d} = (q_t^1, \ldots, q_t^d)$ is a vector consisting of the states in levels 1 through d at time t, and is denoted by $\vec{i}$ [2]. In Eq.(5), $d \ge 2$ is assumed. We assume that $f_0^1 = 1$ so that a state transition occurs at time $t = 1$. We also assume that $f_t^{D+1} = 1$ so that a state transition occurs in the bottom level at each time point.

$\delta(j = j')$ in Eq.(4) is 1 if $j = j'$, and is 0 if $j \neq j'$. If we assume that the vector of higher-up state variables at time t, $q_t^{1:d-1}$, is $\vec{i}$, then $A_{\vec{i}}^d(j,j')$ is the horizontal transition probability from state j to state j' in level d, $\pi_{\vec{i}}^d(j')$ is the vertical transition probability to state j' in level d, and $Ae_{\vec{i}}^d(j)$ is the horizontal transition probability in level d from state j to an end state. $B_{\vec{i}}(k)$ is the probability to output the k-th symbol when $q_t^{1:D}$ is $\vec{i}$. In Eq.(6), K is the number of output symbols.

3. Finding the most likely state sequence

3.1. GVA

GVA is a direct extension of the Viterbi algorithm to HHMMs that finds the most likely sequence of states in all levels, including those in the lower levels.

Let us define the symbols we use in explaining GVA. Let $Q^{1:D} = (q_1^{1:D}, ..., q_T^{1:D})$ be a sequence of state vectors, where $q_t^{1:D}(1 \le t \le T)$ is the vector of state variables from level 1 to level D at time t, and let $F^{2:D} = (f_1^{2:D}, ..., f_T^{2:D})$ be a sequence of indicator vectors, where $f_t^{2:D}(1 \le t \le T)$ is the vector of indicator variables from level 2 to level D at time t. Let $O = (o_1, ..., o_T)$ be a sequence of observations, where $o_t(1 \le t \le T)$ is the observation symbol at time t.

The most likely state sequence which GVA finds, $(\hat{Q}^{1:D}, \hat{F}^{2:D})$ is defined as follows:

$$(\hat{Q}^{1:D}, \hat{F}^{2:D}) \triangleq \underset{Q^{1:D}, F^{2:D}}{\operatorname{argmax}} P(Q^{1:D}, F^{2:D}|O). \tag{7}$$

To find the most likely state sequence, $(\hat{Q}^{1:D}, \hat{F}^{2:D})$, given an observation sequence $O = \{o_1, ..., o_T\}$, we define $\delta_t(\vec{q}, \vec{f})$ as follows:

$$\delta_t(\vec{q}, \vec{f}) \triangleq \max_{q_{1:t-1}^{1:D}, f_{1:t-1}^{2:D}} \log P\big(q_{1:t-1}^{1:D}, f_{1:t-1}^{2:D}, q_t^{1:D} = \vec{q}, f_t^{2:D} = \vec{f}, o_1, o_2, \cdots, o_t\big). \tag{8}$$

[2] We suppose that $q_t^{1:d-1} = \vec{i}$ stands for the root node in level 0, when $d = 1$ in Eq.(4).

$\delta_t(\vec{q},\vec{f})$ is the log probability of the most likely state sequence that starts from the initial state, emits $o_1,...,o_t$, and ends at time t in state $(\vec{q},\vec{f})$. By induction, we can rewrite Eq.(8) as

$$\delta_t(\vec{q},\vec{f}) = \max_{\vec{q'},\vec{f'}}\{\delta_{t-1}(\vec{q'},\vec{f'}) + \log P(q_t^{1:D}=\vec{q}, f_t^{2:D}=\vec{f} \mid q_{t-1}^{1:D}=\vec{q'}, f_{t-1}^{2:D}=\vec{f'})\} + \log P(o_t \mid q_t^{1:D}=\vec{q}). \quad (9)$$

To actually retrieve the state sequence, we keep track of $\vec{q'},\vec{f'}$ which maximizes the right hand side of Eq.(9) for each state $(\vec{q},\vec{f})$ at each time $t \geq 2$. We do this via the array $\phi_t(\vec{q},\vec{f})$. The whole procedure for finding the most likely state sequence can now be stated as follows.

(Step 1) **Initialization:** for $t = 1$,

$$\delta_1(\vec{q},\vec{f}) = \log P(q_1^{1:D}=\vec{q}, f_1^{2:D}=\vec{f}) + \log P(o_1|q_1^{1:D}=\vec{q}), \quad \forall\vec{q},\forall\vec{f}. \quad (10)$$

(Step 2) **Recursion:** for $t = 2,...,T$,

$$\delta_t(\vec{q},\vec{f}) = \max_{\vec{q'},\vec{f'}}\{\delta_{t-1}(\vec{q'},\vec{f'}) + \log P(q_t^{1:D}=\vec{q}, f_t^{2:D}=\vec{f} \mid q_{t-1}^{1:D}=\vec{q'}, f_{t-1}^{2:D}=\vec{f'})\} + \log P(o_t \mid q_t^{1:D}=\vec{q}), \quad \forall\vec{q},\forall\vec{f}, \quad (11)$$

$$\phi_t(\vec{q},\vec{f}) = \operatorname*{argmax}_{\vec{q'},\vec{f'}}\{\delta_{t-1}(\vec{q'},\vec{f'}) + \log P(q_t^{1:D}=\vec{q}, f_t^{2:D}=\vec{f} \mid q_{t-1}^{1:D}=\vec{q'}, f_{t-1}^{2:D}=\vec{f'})\} + \log P(o_t \mid q_t^{1:D}=\vec{q}), \quad \forall\vec{q},\forall\vec{f}. \quad (12)$$

(Step 3) **Termination:**

$$\log \hat{P} = \max_{\vec{q},\vec{f}} \delta_T(\vec{q},\vec{f}), \quad (13)$$

$$(\hat{q}_T^{1:D}, \hat{f}_T^{2:D}) = \operatorname*{argmax}_{\vec{q},\vec{f}} \delta_T(\vec{q},\vec{f}). \quad (14)$$

(Step 4) **Path (state sequence) backtracking:**

for $t = T-1, T-2, ..., 1$,

$$(\hat{q}_t^{1:D}, \hat{f}_t^{2:D}) = \phi_{t+1}(\hat{q}_{t+1}^{1:D}, \hat{f}_{t+1}^{2:D}). \quad (15)$$

The time complexity of GVA is $O(T)$.

3.2. MVA for two-level HHMMs

In hierarchical models, upper level states usually convey more important information. MVA finds the most likely upper level state sequence by marginalizing the lower level state sequences. In this subsection, we explain MVA for two-level HHMMs for simplicity. We will explain MVA for general D-level HHMMs in the appendix.

Let $Q^1 = q^1_{1:T}$, $Q^2 = q^2_{1:T}$, and let $F^2 = f^2_{1:T}$. The most likely upper level state sequence, $(\hat{Q}^1, \hat{F}^2)$, is defined as follows [3]:

$$(\hat{Q}^1, \hat{F}^2) \triangleq \underset{Q^1, F^2}{\operatorname{argmax}} P(Q^1, F^2, O) = \underset{Q^1, F^2}{\operatorname{argmax}} \sum_{Q^2} P(Q^{1:2}, F^2, O). \tag{16}$$

Before explaining MVA, let us define *segments*. Suppose we are given $\{Q^1 = q^1_{1:T}, F^2 = f^2_{1:T}, Q^2 = q^2_{1:T}\}$, a state sequence from time 1 to time T. We call $\{q^1_{t_1:t_2}, f^2_{t_1:t_2}, q^2_{t_1:t_2}\}$, which is a partial sequence of states between time t_1 and t_2, a segment with t_1, t_2 as the start time and the end time for the segment, when $\{q^1_{t_1:t_2}, f^2_{t_1:t_2}, q^2_{t_1:t_2}\}$ satisfies the following conditions:

$$\begin{cases} t_1 = 1 \text{ or } f^2_{t_1-1} = 1, \\ f^2_{t_1:t_2-1} = 0 \text{ if } \ t_2 - t_1 \geq 2, \\ f^2_{t_2} = 1. \end{cases} \tag{17}$$

Simply speaking, a segment is a partial sequence of states between the time just after the state in level 2 transitions to its end state and the time when the state in level 2 transitions to its end state again. See Fig. 4. Note that level 1 state transitions occur only at segment boundaries, and that the state in level 1 does not change within a segment (i.e., between the start and the end time of a segment). Therefore, the level 1 state sequence from time 1 to time T can be specified by the end times (or the start times) of all the segments and the level 1 state for each of the segments.

Computing $\{\delta_t(i) \mid 1 \leq i \leq N^1, 1 \leq t \leq T\}$ defined below plays a central role in MVA:

$$\begin{aligned} \delta_t(i) &\triangleq \max_{q^1_{1:t-1}, f^2_{1:t-1}} \log P(q^1_{1:t-1}, f^2_{1:t-1}, q^1_t = i, f^2_t = 1, o_{1:t}) \\ &= \max_{q^1_{1:t-1}, f^2_{1:t-1}} \log \sum_{q^2_{1:t}} P(q^1_{1:t-1}, f^2_{1:t-1}, q^2_{1:t-1}, q^1_t = i, f^2_t = 1, q^2_t, o_{1:t}), \end{aligned} \tag{18}$$

where N^1 is the total number of level 1 states. $\delta_t(i)$ is the log probability of the most likely level 1 state sequence which starts from the initial state, emits $o_1, ..., o_t$, and ends at time t, when the level 1 state is i and the level 2 state transitions to its end state (i.e. t is an end time of a segment).

[3] For some applications, we may want to marginalize the indicator variable sequences, F^2, as well.

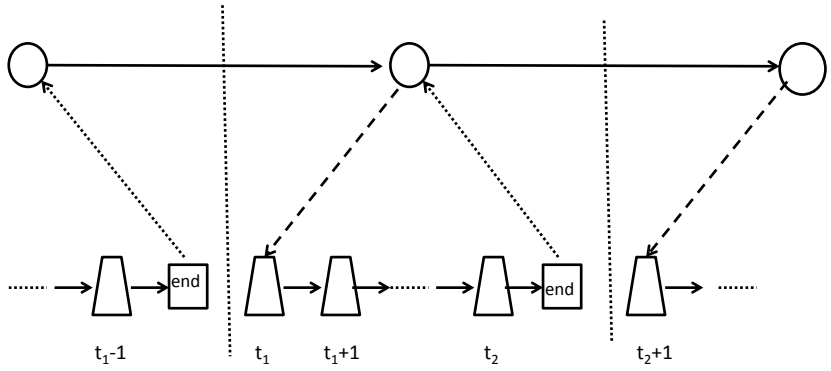

Figure 4: A segment that starts at t_1 and ends at t_2.

Given an observation sequence, O, and the indicator variable at time T, $f_T^2 = \tilde{f}_T^2$, the most likely level 1 state sequence can be found by the following dynamic program. We assume $\tilde{f}_T^2 = 1$ for simplicity.

(Step 1) **Initialization:** for $t = 1$,

$$\delta_1(i) = \log P(q_1^1 = i, f_1^2 = 1, o_1), \qquad \forall i, \tag{19}$$

$$\phi_1(i) = (0, 0), \qquad \forall i. \tag{20}$$

(Step 2) **Recursion:** for $t = 2, ..., T$,

$$\delta_t(i) = \max\big\{\alpha_{0,t}(i), \max_{j,\tau:1\le\tau<t}\big(\delta_\tau(j) + \log A_0^1(j,i) + \alpha_{\tau,t}(i)\big)\big\}, \ \forall i, \tag{21}$$

$$\phi_t(i) = \begin{cases} (0,0) & \text{if } \delta_t(i) = \alpha_{0,t}(i), \\ (j^*,\tau^*) = \operatorname{argmax}_{j,\tau:1\le\tau<t}(\delta_\tau(j) + \log A_0^1(j,i) + \alpha_{\tau,t}(i)) & \text{otherwise,} \end{cases} \quad \forall i, \tag{22}$$

where

$$\begin{aligned} \alpha_{\tau,t}(i) &= \log P(f_{\tau+1:t-1}^2 = 0, f_t^2 = 1, o_{\tau+1:t} | f_\tau^2 = 1, q_{\tau+1}^1 = i) \\ &= \log \sum_{q_{\tau+1:t}^2} P(f_{\tau+1:t-1}^2 = 0, f_t^2 = 1, q_{\tau+1:t}^2, o_{\tau+1:t} \mid f_\tau^2 = 1, q_{\tau+1}^1 = i), \\ &\qquad \tau \ge 1, \\ \alpha_{0,t}(i) &= \log P(q_{1:t}^1 = i, f_{1:t-1}^2 = 0, f_t^2 = 1, o_{1:t}). \end{aligned}$$

(Step 3) **Backtracking**

$$\hat{q}_T^1 = \operatorname*{argmax}_i \delta_T(i) \ , \hat{f}_T^2 = 1, t = T. \tag{23}$$

while $t > 0$ do

1) $(\hat{q}^1_\tau, \tau) = \phi_t(\hat{q}^1_t), \hat{f}^2_\tau = 1$

$\hat{q}^1_{\tau+1:t-1} = \hat{q}^1_t, \hat{f}^2_{\tau+1:t-1} = 0$ if $\tau + 1 \leq t - 1$

2) $t \leftarrow \tau$

endwhile

In the above, $A^1_0(\cdot, \cdot)$ is the horizontal transition probability for the level 1 states, the subscript τ in $\alpha_{\tau,t}(i)$ is the end time of the segment just before the segment which ends at time t, and $\alpha_{\tau,t}(i)$ is the log probability that the subsequence of observations $o_{\tau+1:t}$ is emitted by any segment whose level 1 state is i. That is, $\alpha_{\tau,t}(i)$ is the marginalization over the lower level state sequence of the joint log probability that $o_{\tau+1:t}$ is emitted by a lower level state sequence generated by the upper level state i. All of $\{\alpha_{\tau,t}(i)|1 \leq i \leq N^1, 1 \leq \tau < t\}$ can be computed by the backward procedure for HMMs in $O(t)$ total time[1]. $\phi_t(i)$ is a variable used for backtracking, containing both the level 1 state, j, of the previous segment and the end time, τ, for the previous segment.

While the time complexity of GVA is $O(T)$, the time complexity of MVA is $O(T^2)$. This is the cost we have to pay to find the most likely level 1 state sequence.

3.3. Fast approximation algorithm for MVA

In this subsection, we explain a fast approximation algorithm for MVA, which we call AVA (Approximate Viterbi Algorithm). AVA finds the most likely level 1 state sequence whose segments are not longer than L. AVA is based on the dynamic programming in Section 3.2. While computing $\{\delta_t(i)|1 \leq i \leq N^1, 1 \leq t \leq T\}$ defined in Eq.(21) plays a central role in MVA, computing $\{\tilde{\delta}_t(i)|1 \leq i \leq N^1, 1 \leq t \leq T\}$ defined below plays a central role in AVA.

$$\tilde{\delta}_t(i) \triangleq \begin{cases} \max\{\alpha_{0,t}(i), \max_{j,\tau:\max\{1,t-(L-1)\}\leq\tau<t}(\tilde{\delta}_\tau(j) + \log A^1_0(j,i) + \alpha_{\tau,t}(i))\} \\ \qquad t \leq L \\ \max_{j,\tau:\max\{1,t-(L-1)\}\leq\tau<t}(\tilde{\delta}_\tau(j) + \log A^1_0(j,i) + \alpha_{\tau,t}(i)) \quad \textit{otherwise} \end{cases} \tag{24}$$

Note that while the range of τ in MVA is $1 \leq \tau < t$ in Eq.(21), it is $\max\{1, t - (L-1)\} \leq \tau < t$ in AVA because the length of a segment is less than or equal to L.

When we set $L = T$, AVA becomes exact and is equivalent to MVA. The time complexity of AVA, considering L to be a constant, is $O(LT) = O(T)$.

4. Experiments

We carry out four experiments. In experiment 1, we compare GVA and MVA in terms of accuracy. In experiment 2, we determine when MVA is much more accurate than GVA. In experiment 3, we evaluate AVA in terms of accuracy and efficiency. These three experiments all use artificial data. In experiment 4, we compare GVA and MVA using real data.

4.1. Overview of the experiment

4.1.1. Experimental Data

Artificial data to be used in experiments 1, 2 and 3 are randomly generated from two-level HHMMs. The initial state probabilities, $\{\pi_0^1(i) \mid 1 \leq i \leq N^1\}$ (N^1 is the total number of states at level 1), the state transition probabilities, $\{A_0^1(i,i') \mid 1 \leq i,i' \leq N^1\}$, for level 1, the initial state probabilities, $\{\pi_i^2(j) \mid 1 \leq i \leq N^1, 1 \leq j \leq N^2\}$ (N^2 is the total number of states in level 2), and the state transition probabilities, $\{A_i^2(j,j') \mid 1 \leq i \leq N^1, 1 \leq j,j' \leq N^2\}$, are all sampled from Dirichlet distributions with concentration parameters α_k equal to 1.0 in experiments 1 and 2, and 0.1 in experiment 3. The state ending probabilities, $\{A_i^2(j,end) \mid 1 \leq i \leq N^1, 1 \leq j \leq N^2\}$, are all 0.1 in experiments 1 and 2, and are sampled from the uniform distribution over the interval $(0,1)$ in experiment 3. The output probabilities, $\{B_{(i,j)}(k) \mid 1 \leq i \leq N^1, 1 \leq j \leq N^2, 1 \leq k \leq K\}$, are also sampled from Dirichlet distributions with concentration parameters α_k equal to 1.0 in experiments 1 and 2, and 0.1 in experiment 3.

A concentration parameter of 1.0 in the Dirichlet distribution results in all sets of probabilities being equally likely (i.e. a uniform distribution). The state ending probabilities are set to 0.1 in experiments 1 and 2 to make the segments longer and make upper level state estimation easier. The state ending probabilities are set to be random in experiment 3 to generate both short and long segments. This makes upper level state estimation more difficult. Therefore, the concentration parameters of the Dirichlet distributions are set to 0.1 in experiment 3 (which causes generated probabilities of approximately 0 or 1), to make state estimation easier.

In experiment 4, we use the same data that Skounakis et al. [4] used for information extraction.

4.1.2. Performance Metrics

We estimate the upper level state sequences[4] using GVA and MVA. The performance of the algorithms is evaluated in terms of two accuracy rates:

- Accuracy rate 1 is computed from the sequences:

$$\text{accuracy rate 1} \triangleq \frac{N_{seq}^{corr}}{N_{seq}},$$

  where N_{seq} is the total number of sequences, and N_{seq}^{corr} is the total number of sequences for which the upper level states are always correctly estimated.

- Accuracy rate 2 is computed from the times:

$$\text{accuracy rate 2} \triangleq \frac{N_{times}^{corr}}{N_{times}},$$

[4] An upper level state sequence can be either $q_{1:T}^1$ or $(q_{1:T}^1, f_{1:T}^2)$, but we use the latter, $(q_{1:T}^1, f_{1:T}^2)$, as an upper level state sequence.

Table 1: Results of Experiment 1 ($N^1 = 2, N_\lambda = N_{\lambda'} = 4$, the numbers in parentheses are standard deviations).

		Accuracy Rate 1 (%)		Accuracy Rate 2 (%)	
K	T	GVA	MVA	GVA	MVA
2	10	30.6 (7.2)	**32.7** (6.2)	64.8 (11.9)	**69.1** (8.9)
	20	9.4 (4.3)	**11.5** (3.4)	56.0 (15.4)	**64.7** (11.3)
	50	0.5 (0.7)	**0.6** (0.8)	58.1 (16.6)	**65.1** (12.7)
4	10	31.7 (6.5)	**34.0** (6.4)	68.3 (8.8)	**71.1** (8.1)
	20	11.9 (4.2)	**13.4** (3.7)	66.6 (12.8)	**72.1** (9.2)
	50	0.4 (0.6)	**0.7** (0.8)	64.6 (12.6)	**71.0** (8.2)
6	10	32.1 (5.9)	**34.2** (5.4)	68.7 (8.2)	**71.2** (7.3)
	20	12.3 (3.5)	**13.3** (3.7)	70.0 (9.3)	**72.6** (8.2)
	50	0.8 (1.0)	**0.9** (0.9)	69.4 (12.1)	**73.8** (8.0)
8	10	34.1 (6.2)	**36.0** (5.6)	70.4 (7.8)	**73.2** (6.7)
	20	12.3 (3.9)	**13.7** (3.6)	69.4 (8.0)	**72.7** (6.1)
	50	0.6 (0.7)	**0.7** (0.9)	70.0 (9.0)	**73.8** (7.5)
10	10	33.5 (5.1)	**35.3** (5.0)	71.2 (7.3)	**73.5** (6.5)
	20	12.4 (3.7)	**13.3** (3.9)	71.1 (7.6)	**73.3** (6.6)
	50	0.7 (0.9)	**0.8** (1.0)	72.3 (8.4)	**75.9** (6.7)

where N_{times} is the sum of the sequence lengths, and N_{times}^{corr} is the sum of the times at which the upper level states are correctly estimated.

4.2. Experiment 1

In experiment 1,we compare GVA and MVA in terms of the two accuracy rates.

We set $N^1 = 2$. Let λ and λ' be the lower level models (considered as HMMs) whose parent states are states 1 and 2 in level 1, respectively. In experiment 1, we consider the case where λ and λ' have the same number of states.

- N_λ and $N_{\lambda'}$, the total number of states in λ and λ', are both four.
- K is $2, 4, 6, 8$ or 10.
- We generate 100 sequences of length $T = 10$, $T = 20$, and $T = 50$ for each trial.
- The accuracy rates are averaged over 100 trials.

We show the results in Table 1. MVA outperforms GVA in terms of both accuracy rate 1 and accuracy rate 2. For both algorithms, accuracy rate 1 decreases as T becomes longer. Accuracy rates 1 and 2 both increase as K becomes larger. The reason is as follows. Since the symbol output probabilities are sampled from Dirichlet distributions whose concentration parameters are 0.1, the probabilities are close to 0 or 1, and therefore it becomes easier to estimate states from observations when there are more output symbols.

Table 2: Results of Experiment 2 ($N_\lambda = 8, N_{\lambda'} = 4$, the numbers in parentheses are standard deviations).

(a) Accuracy Rate 1 (%)

K	T	GVA			MVA		
2	10	25.3	(10.2)	-5.3	**32.3**	(7.0)	-0.4
	20	7.5	(4.4)	-1.9	**11.4**	(3.7)	-0.2
	50	0.4	(0.6)	-0.1	**0.6**	(0.8)	0.0
4	10	26.5	(8.0)	-5.3	**33.8**	(5.1)	-0.2
	20	9.1	(4.7)	-2.7	**13.0**	(3.8)	-0.4
	50	0.3	(0.6)	-0.1	**0.5**	(0.7)	-0.2
6	10	28.4	(8.1)	-3.8	**34.5**	(5.7)	+0.3
	20	9.2	(4.4)	-3.0	**13.0**	(3.7)	-0.4
	50	0.4	(0.7)	-0.4	**0.7**	(0.8)	-0.2
8	10	29.8	(7.7)	-4.3	**35.1**	(4.9)	-0.9
	20	10.2	(4.7)	-2.1	**14.5**	(4.5)	+0.8
	50	0.4	(0.6)	-0.2	**0.7**	(0.8)	0.0
10	10	28.6	(7.1)	-4.9	**34.4**	(5.7)	-0.9
	20	9.9	(4.1)	-2.5	**13.3**	(3.3)	0.0
	50	0.4	(0.7)	-0.3	**0.9**	(0.9)	+0.1

4.3. Experiment 2

In experiment 2, we determine when MVA is much more accurate than GVA in terms of accuracy rate 1.

We set $N^1 = 2$, as in experiment 1. In experiment 2, we consider the case where λ and λ' have different numbers of states. N_λ and $N_{\lambda'}$ are eight and four, respectively. All other settings are the same as in experiment 1.

We show the results (accuracy rate 1) of experiment 2 in Table 2. The figures in parentheses show the differences from the accuracy rates in experiment 1. When we compare the results with those from experiment 1, we notice that while the accuracy rates of GVA are lower, the accuracy rates of MVA do not change very much. We conjecture that the reason is as follows. Since GVA finds the most likely whole level state sequence including the lower level states, a model which has fewer states (λ'), i.e. the upper level state 2, is more likely to be chosen by the algorithm because of the (possibly) larger transition probabilities in level 2. On the other hand, MVA, which marginalizes the lower level state, does not have this problem. This is why there is a big difference in the accuracy rates of the two algorithms.

4.4. Experiment 3

In experiment 3, we evaluate AVA in terms of accuracy rate 1 and efficiency.

We set $L = 9$ for AVA. We compute accuracy rate 1 for each of the following two cases:

Case 1: Samples with a segment which is longer than L are excluded from the test data.

Table 3: Results of Experiment 3: Accuracy rate 1 (%) for cases 1 and 2 (the numbers in parentheses are standard deviations).

		Case 1						Case 2					
K	T	GVA		MVA		AVA		GVA		MVA		AVA	
2	10	17.3	(19.6)	17.9	(19.8)	**18.9**	(19.6)	24.9	(21.4)	**25.6**	(21.4)	23.7	(20.6)
	20	5.0	(9.2)	5.1	(9.4)	**5.4**	(9.4)	9.1	(13.8)	**9.3**	(13.7)	6.6	(10.9)
	50	2.9	(12.4)	3.0	(12.4)	**3.4**	(12.8)	3.4	(10.0)	**3.5**	(10.3)	1.3	(4.1)
4	10	30.0	(20.4)	30.6	(20.3)	**31.2**	(20.3)	31.1	(24.7)	**31.3**	(24.5)	28.1	(22.8)
	20	15.9	(19.4)	16.2	(19.4)	**16.7**	(19.6)	16.3	(20.1)	**16.5**	(20.2)	13.2	(17.3)
	50	4.6	(13.1)	4.7	(13.2)	**4.8**	(13.2)	**4.0**	(11.1)	**4.0**	(11.1)	3.1	(10.5)
6	10	36.6	(26.0)	36.9	(25.8)	**37.4**	(25.8)	36.8	(22.8)	**37.5**	(22.9)	34.2	(21.9)
	20	14.7	(17.5)	15.1	(17.8)	**15.8**	(18.0)	19.3	(21.2)	**19.5**	(21.3)	16.4	(18.7)
	50	4.4	(11.2)	4.7	(11.5)	**5.0**	(11.7)	**5.7**	(11.7)	**5.7**	(11.7)	3.1	(8.8)
8	10	37.3	(23.4)	37.8	(23.9)	**39.1**	(23.6)	37.4	(23.3)	**37.8**	(23.5)	35.6	(22.6)
	20	19.4	(22.7)	19.4	(22.7)	**19.8**	(22.8)	20.0	(19.7)	**20.2**	(19.8)	16.4	(18.8)
	50	5.7	(12.4)	5.8	(12.4)	**6.2**	(13.1)	4.9	(10.0)	**5.0**	(10.0)	3.3	(8.2)
10	10	39.7	(22.5)	40.2	(22.8)	**40.8**	(23.0)	42.2	(25.7)	**43.0**	(25.9)	41.7	(25.3)
	20	21.6	(24.0)	21.9	(24.2)	**22.9**	(24.4)	20.9	(22.8)	**21.2**	(22.8)	18.3	(21.8)
	50	3.9	(9.5)	4.0	(9.6)	**4.4**	(19.7)	7.9	(14.7)	**8.1**	(14.9)	4.2	(10.5)

Case 2: Samples with a segment which is longer than L are included in the test data.

In comparing accuracy, all the settings are the same as in experiment 1. In comparing efficiency, we set $N^1 = 2$, $N^2 = 4$, and $K = 4$, and measure the computation times for sequences with $T = 10, 20, 50, 100, 200, 500$.

We show the accuracy results from the experiment in Table 3. The accuracy rate of AVA is higher than that of MVA in case 1 and is lower in case 2.

We conjecture that the reason why AVA is more accurate than the exact algorithm, i.e., MVA in case 1, is that while MVA finds the most likely sequence without a condition on segment length, AVA finds the most likely sequence from sequences whose segments are shorter than or equal to L. Of course, the sequence found by MVA has higher likelihood, but when samples with a segment which is longer than L are excluded (as in case 1), the maximal segment length of the true sequence (i.e., the sample sequence) must be shorter than or equal to L. Hence, the sequence found by MVA cannot be correct if its maximal segment length is longer than L. This is why AVA is more accurate than method 1 in case 1.

We show the computation times in Figure 5. We can confirm from the figure that while the computation time of MVA, an exact algorithm without approximation, is $O(T^2)$, the computation time of AVA is $O(T)$, and thus AVA is much faster than MVA.

Combining the accuracy and efficiency results of the experiment, we conclude that AVA is very effective when the maximal segment length, L, is known in advance and is much shorter than the sequence length, T.

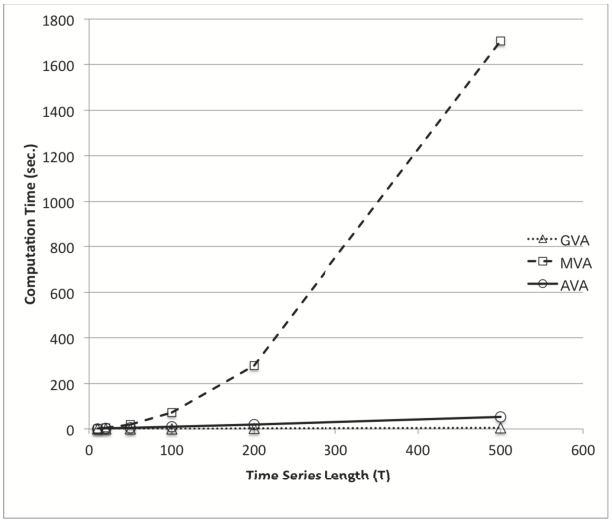

Figure 5: Results of Experiment 3: Computation Time

4.5. Experiment 4

In this experiment, we use the same data that Skounakis et al. [4] used for information extraction. Information extraction is the task of automatically extracting instances of specified classes or relations from text, for which Skounakis et al. proposed a method using hierarchical hidden Markov models. In the models they use, the upper level states represent phrases, and the lower level states represent parts of speech.

The three data sets are:

- Protein-Location (PL) data set (730 sentences),
- Gene-Disorder (GD) data set (825 sentences),
- Protein-Protein (PP) data set (5270 sentences).

We estimate the upper level state sequence $q^1_{1:T}$ using GVA and MVA, and compare their accuracy rates. Since we need to learn the HHMM from the data in this experiment, we divide each of data set into training data sentences and test data sentences. We use five-fold cross validation.

The results in Table 4 show that MVA outperforms GVA in both accuracy rate 1 and accuracy rate 2.

5. Conclusions

The generalized Viterbi algorithm, a direct extension of the Viterbi algorithm for HMMs, has been used to find the most likely state sequence for hierarchical HMMs. However, the generalized Viterbi algorithm finds the most likely whole

Table 4: Results for Experiment 4

	Accuracy Rate 1 (%)		Accuracy Rate 2 (%)	
	GVA	MVA	GVA	MVA
PL	7.6	**7.8**	83.9	**84.5**
GD	11.9	**13.1**	91.3	**92.3**
PP	8.9	**9.4**	88.9	**89.3**

level state sequence, but not the most likely upper level state sequence. In this paper, we have proposed a marginalized Viterbi algorithm that finds the most likely upper level state sequence by marginalizing the lower level state sequences. Using experiments, we have shown that the marginalized Viterbi algorithm is more accurate than the generalized Viterbi algorithm in terms of upper level state sequence estimation.

Appendix A. MVA for general *D*-level HHMMs

In this section, we explain MVA for general D-level HHMMs.

Let $1,\ldots,U$ be the upper levels, and let $U+1,\ldots,D$ be the lower levels. The most likely upper level state sequence, $(\hat{Q}^{1:U},\hat{F}^{2:U+1})$, is defined as follows:

$$\begin{aligned}(\hat{Q}^{1:U},\hat{F}^{2:U+1}) &\triangleq \underset{Q^{1:U},F^{2:U+1}}{\operatorname{argmax}} P(Q^{1:U},F^{2:U+1},O)\\ &= \underset{Q^{1:U},F^{2:U+1}}{\operatorname{argmax}} \sum_{Q^{U+1:D}} \sum_{F^{U+2:D}} P(Q^{1:D},F^{2:D},O).\end{aligned} \tag{A.1}$$

Computing $\{\delta_t(q^{1:U},f^{2:U})|\forall q^{1:U},\forall f^{2:U},1\le t\le T\}$ defined below plays a central role in MVA. We consider sequences which start from the initial state, emit $o_1,...,o_t$, and end at time t, when the upper level states are $(q^{1:U},f^{2:U})$ and the lower level states transition to their end states (i.e. $f_t^{U+1:D}=1$). $\delta_t(q^{1:U},f^{2:U})$ is the log probability of the most likely upper level state sequence after marginalizing the lower level state sequences, and is formally defined as follows:

$$\begin{aligned}&\delta_t(q^{1:U},f^{2:U})\\ &\quad= \max_{q_{1:t-1}^{1:U},f_{1:t-1}^{2:U},f_{1:t-1}^{U+1}} \log P(q_{1:t-1}^{1:U},q_t^{1:U}=q^{1:U},f_{1:t-1}^{2:U},f_t^{2:U}=f^{2:U},f_{1:t-1}^{U+1},f_t^{U+1}=1,o_{1:t})\\ &\quad= \max_{q_{1:t-1}^{1:U},f_{1:t-1}^{2:U},f_{1:t-1}^{U+1}} \log \sum_{q_{1:t}^{U+1:D}} \sum_{f_{1:t}^{U+2:D}} P(q_{1:t-1}^{1:U},q_t^{1:U}=q^{1:U},q_{1:t}^{U+1:D},f_{1:t-1}^{2:U},\\ &\qquad\qquad f_t^{2:U}=f^{2:U},f_{1:t-1}^{U+1},f_t^{U+1}=1,f_{1:t}^{U+2:D},o_{1:t}).\end{aligned} \tag{A.2}$$

Given an observation sequence, $O=o_{1:T}$, and the indicator variable in level $U+1$ at time T, $f_T^{U+1}=\tilde{f}_T^{U+1}$, the most likely upper level state sequence can be found by the following dynamic program. We assume $\tilde{f}_T^{U+1}=1$ for simplicity.

(Step 1) **Initialization: for** $t = 1$**,**

$$\delta_1(q^{1:U}, f^{2:U}) = \log P(q_1^{1:U} = q^{1:U}, f_1^{2:U} = f^{2:U}, f_1^{U+1} = 1, o_1)$$

$$= \sum_{q_1^{U+1:D}} \sum_{f_1^{U+2:D}} \log P(q_1^{1:U} = q^{1:U}, q_1^{U+1:D}, f_1^{2:U} = f^{2:U}, f_1^{U+1} = 1, f_1^{U+2:D}, o_1),$$

$$\forall q^{1:U}, \forall f^{2:U}, \quad \text{(A.3)}$$

$$\phi_1(q^{1:U}, f^{2:U}) = ((\vec{0}, \vec{0}), 0), \qquad \forall q^{1:U}, \forall f^{2:U}. \quad \text{(A.4)}$$

(Step 2) **Recursion: for** $t = 2, ..., T$**,**

$$\delta_t(q^{1:U}, f^{2:U}) = \max\Big\{ \alpha_{0,t}(q^{1:U}, f^{2:U}), \max_{\bar{q}^{1:U}, \bar{f}^{2:U}, 1 \le \tau < t} \big(\delta_\tau(\bar{q}^{1:U}, \bar{f}^{2:U}) + \log A^{1:U}((\bar{q}^{1:U}, \bar{f}^{2:U}), q^{1:U}) + \alpha_{\tau,t}(q^{1:U}, f^{2:U}) \big) \Big\}, \ \forall q^{1:U}, \forall f^{2:U}, \quad \text{(A.5)}$$

$$\phi_t(q^{1:U}, f^{2:U}) = \begin{cases} ((\vec{0}, \vec{0}), 0) & \text{if} \quad \delta_t(q^{1:U}, f^{2:U}) = \alpha_{0,t}(q^{1:U}, f^{2:U}) \\ \operatorname{argmax}_{\bar{q}^{1:U}, \bar{f}^{2:U}, 1 \le \tau < t} (\delta_\tau(\bar{q}^{1:U}, \bar{f}^{2:U}) + log A^{1:U}\big((\bar{q}^{1:U}, \bar{f}^{2:U}), q^{1:U}\big) \\ \qquad + \alpha_{\tau,t}(q^{1:U}, f^{2:U})\big) \quad \text{otherwise}, \end{cases}$$

$$\forall q^{1:U}, \forall f^{2:U}. \quad \text{(A.6)}$$

where

$$\alpha_{\tau,t}(q^{1:U}, f^{2:U}) = \log P(f_{\tau+1:t-1}^{U+1} = 0, f_t^{U+1} = 1, o_{\tau+1:t} \mid q_{\tau+1}^{1:U} = q^{1:U}, f_t^{2:U} = f^{2:U}, f_\tau^{U+1} = 1) \qquad \tau \ge 1,$$

$$\alpha_{0,t}(q^{1:U}, f^{2:U}) = \log P(q_1^{1:U} = q^{1:U}, f_1^{2:U} = f^{2:U}, f_{1:t-1}^{U+1} = 0, f_t^{U+1} = 1, o_{1:t}).$$

(Step 3) **Backtracking:**

$$(\hat{q}_T^{1:U}, \hat{f}_T^{2:U}) = \operatorname*{argmax}_{q^{1:U}, f^{2:U}} \delta_T(q^{1:U}, f^{2:U}), \hat{f}_T^{U+1} = 1, t = T. \quad \text{(A.7)}$$

while $t > 0$ do

1) $((\hat{q}_\tau^{1:U}, \hat{f}_\tau^{2:U}), \tau) = \phi_t(\hat{q}_t^{1:U}, \hat{f}_t^{2:U}), \hat{f}_\tau^{U+1} = 1, \hat{q}_{\tau+1:t-1}^{1:U} = \hat{q}_t^{1:U}, \hat{f}_{\tau+1:t-1}^{2:U} = \vec{0}, \hat{f}_{\tau+1:t-1}^{U+1} = 0,$ if $\tau + 1 \le t - 1$

2) $t \leftarrow \tau$

endwhile

In the above, $A^{1:U}(\cdot,\cdot)$ is the horizontal transition probabilities for the upper level states, the subscript τ in $\alpha_{\tau,t}(q^{1:U}, f^{2:U})$ is the end time of the segment[5] just before the segment which ends at time t, and $\alpha_{\tau,t}(q^{1:U}, f^{2:U})$ is the log probability that the subsequence of observations $o_{\tau+1:t}$ is emitted by any segment whose upper level states are $q^{1:U}$ and whose upper level indicator variables at time t are $f^{2:U}$. In other words, $\alpha_{\tau,t}(q^{1:U}, f^{2:U})$ is the marginalization over the lower level state sequence, $(q_{\tau+1:t}^{U+1:D}, f_{\tau+1:t}^{U+1:D})$, of the joint log probability that $o_{\tau+1:t}$ is emitted by a lower level state sequence generated by the upper level state $q^{1:U}$. $\alpha_{\tau,t}(q^{1:U}, f^{2:U})$ can be computed by the (generalized) backward procedure for HMMs [1]. $\phi_t(q^{1:U}, f^{2:U})$ is a variable used for backtracking, and contains the upper level states, $\bar{q}^{1:U}$, of the previous segment, the upper level indicator variables, $\bar{f}^{2:U}$, at time τ and the end time, τ, for the previous segment.

While the time complexity of GVA is $O(T)$, the time complexity of MVA is $O(T^2)$.

References

[1] L. Rabiner, B.-H. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.

[2] S. Fine, Y. Singer, N. Tishby, The hierarchical hidden Markov model: Analysis and applications, Machine Learning 32 (1) (1998) 41–62.

[3] K. Murphy, M. Paskin, Linear time inference in hierarchical HMMs, Advances in Neural Information Processing Systems 14 (2001) 833–840.

[4] M. Skounakis, M. Craven, S. Ray, Hierarchical hidden Markov models for information extraction, in: Proceedings of 18th International Joint Conference on Artificial Intelligence, 2003, pp. 427–433.

[5] D. Q. Phung, T. V. Duong, S. Venkatesh, H. H. Bui, Topic transition detection using hierarchical hidden Markov and semi-Markov models, in: Proceedings of 13th ACM International Conference on Multimedia, 2005, pp. 11–20.

[6] N. T. Nguyen, D. Q. Phung, S. Venkatesh, H. Bui, Learning and detecting activities from movement trajectories using the hierarchical hidden Markov models, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, 2005, pp. 955–960.

[7] A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory 13 (2) (1967) 260–269.

[5] We generalize the notion of segments, which have been originally defined for two-level HHMMs, to D level HHMMs by viewing the upper levels (levels $1 \sim U$) as level 1, and the lower levels (levels $U+1 \sim D$) as level 2.

[8] D. Jurafsky, J. H. Martin, Speeach and Language Processing (2nd Ed.), Prentice Hall, 2008.

[9] D. A. Reynolds, Speaker identification and verification using gaussian mixture speaker models, Speech Communication 17 (1995) 91–108.

[10] T. Sugiura, N. Gotou, A. Hayashi, A discriminative model corresponding to hierarchical HMMs, in: Proc. 14th Int. Conf. Intell. Data Eng. and Automated Learning, 2007, pp. 375–384.