Modeling Interleaved Hidden Processes

Niels Landwehr NIELS.LANDWEHR@CS.KULEUVEN.BE
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A 3001 Heverlee, Belgium

Abstract

Hidden Markov models assume that observations in time series data stem from some hidden process that can be compactly represented as a Markov chain. We generalize this model by assuming that the observed data stems from *multiple* hidden processes, whose outputs interleave to form the sequence of observations. Exact inference in this model is NP-hard. However, a tractable and effective inference algorithm is obtained by extending structured approximate inference methods used in factorial hidden Markov models. The proposed model is evaluated in an activity recognition domain, where multiple activities interleave and together generate a stream of sensor observations. It is shown to be more accurate than a standard hidden Markov model in this domain.

1. Introduction

Hidden Markov models (HMMs) are among the most popular approaches for modeling time series data, and have seen widespread application in areas such as speech recognition, bioinformatics, or robotics. They assume that observed data stems from a hidden process which is stationary and Markov. However, in some application domains this single-process model is not appropriate. Consider for instance a log of web server requests, and assume we have no definite knowledge about which request has been issued by which user (e.g. because of proxy use). Clearly, there is no single hidden Markov process that accounts for the sequence of observed requests. Instead, there are multiple processes, one per user, which interleave to generate the sequence of observations. Another example, and the main motivation for the work presented in this paper, is *activity recognition*: the task of inferring a user's current activity from a stream of dense sensor data. In many situations, users switch back and forth between multiple activities, which causes sensor observations associated with the individual activities to interleave in time. The specific scenario considered in this paper is that objects used in activities of daily living are equipped with small RFID tags, which are picked up by a wearable RFID reader whenever a user interacts with the object. The task is to infer the sequence of activities carried out given the observed object interactions. In the light of recent advances in RFID technology, which allow tags to be cheaply mass-produced and readers to be made wearable, such application scenarios are attracting increasing research interest from both academia and industry (Wang et al., 2007).

HMMs have been widely used in activity recognition: activities are modeled as hidden states that emit the object tags observed by the RFID reader (Patterson et al., 2005). This is an appropriate model if activities are atomic and carried out sequentially. In many domains, however, activities are hierarchically structured, as sets of *basic* activities can be grouped into *high-level* activities. High-level activities typically interleave in time as a user is switching between them, as illustrated in Figure 1. In this example, a user is having breakfast, which consists of high-level activities makeToast, makeJuice, and getNews with corresponding basic activities. The domain could be modeled with a standard HMM by "flattening" the three activities into one process with 7 states, but in this case part of the problem structure would be lost. Alternatively, we can model the activities as three *different* processes which interleave in time. This has the advantage of decoupling transition dynamics within one high-level activity from the interleaving behavior, yielding a more concise representation with fewer parameters.

In this paper, we present a probabilistic model in which observations are generated by multiple, interleaved hidden processes. The hidden processes are stationary Markov chains, and the switching mechanism by which they interleave is again Markov. Although there exists a large body of related work, to the best

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

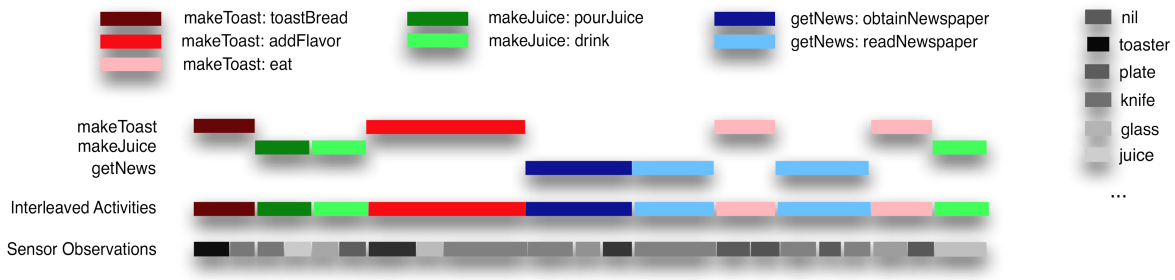

Figure 1. Interleaving in an activity recognition domain. Three high-level activities (makeToast, makeJuice, getNews) with corresponding basic activities are interleaved in time as a user switches between them. Different activities can produce identical sensor observations, and therefore neither the interleaving nor the actual activities are directly observable from the sensor data.

of our knowledge this interleaved setting has not been addressed before. A simplified version has been discussed in (Batu et al., 2004); however, tractable inference algorithms are not explored in this work. Simplicial mixtures of Markov chains, which employ a generative semantics similar to latent Dirichlet allocation, also address a similar problem (Girolami & Kabán, 2003). However, they restrict the constituent processes to be Markov rather than hidden Markov. A further class of models assumes several hidden processes that run in parallel, and that observations stem from their joint state. Examples include *factorial hidden Markov models* (Ghahramani & Jordan, 1997), *hidden Markov decision trees* (Jordan et al., 1996), *coupled hidden Markov models* (Brand, 1997) and *mixed hidden Markov models* (Altman, 2007). In contrast to our approach, these models focus on factorizing complex state spaces into cross-products of simpler components, rather than modeling interleaved processes. Another related technique are *switching state-space models* (SSSMs) (Ghahramani & Hinton, 1998), in which several processes run in parallel and an additional *switch* variable selects one *active* process from which the current observation is generated. SSSMs are different in that processes run concurrently, while an interleaving of processes is characterized by the fact that an inactive process is stopped and only resumes when it becomes active again. This creates additional dependencies between processes which cannot be modeled in a SSSM. Finally, *hierarchical hidden Markov models* model hierarchical structure within the hidden process that generates the observations (Fine et al., 1998). However, the component processes cannot interleave, and thus the model is not appropriate in our domain.

The next section introduces the proposed model more formally. Afterwards, we discuss the key problem of hidden state inference: given a sequence of observations, find the most likely configuration of hidden processes to have generated the data. Unfortunately, exact inference can be shown to be NP-hard; however, efficient structured approximate inference techniques can be applied (Section 3). Finally, the proposed technique is evaluated in an activity recognition domain, and shown to outperform standard HMM-based approaches (Section 4).

2. The Model

Let $Y_1, ..., Y_T$ denote a sequence of observations, where the Y_t take on one of D discrete values. A hidden Markov model μ (Rabiner, 1989) defines a sequence $X_1, ..., X_T$ of hidden state variables, with $X_t \in \{1, ..., K\}$ and K the number of different states the hidden process can take on. To simplify notation, assume that there is a special start state 0 the process is in at time $t = 0$, that is, $X_0 = 0$. The first transition is from X_0 to $X_1 \in \{1, ..., K\}$, and afterwards the first output Y_1 is emitted. The HMM is characterized by initial state probabilities $a_{0i} = P(X_1 = i \mid X_0 = 0)$, state transition probabilities $a_{ij} = P(X_t = j \mid X_{t-1} = i)$ for $t \geq 2$ and emission probabilities $b_{il} = P(Y_t = l \mid X_t = i)$ for $t \geq 1$. The joint distribution of observations $\mathbf{Y} = Y_1, ...Y_T$ and hidden states $\mathbf{X} = X_1, ..., X_T$ is given by

$$P(\mathbf{X}, \mathbf{Y}) = \prod_{t=1}^{T} P(X_t \mid X_{t-1}) P(Y_t \mid X_t).$$

We will also refer to $\mathbf{X}$ as the hidden process that generated the observations $\mathbf{Y}$.

We propose a model for multiple, interleaved hidden processes. Intuitively, an additional *switching process* controls a token that is handed from process to process, and determines which of the processes is active at a particular point t in time. The active process tran-

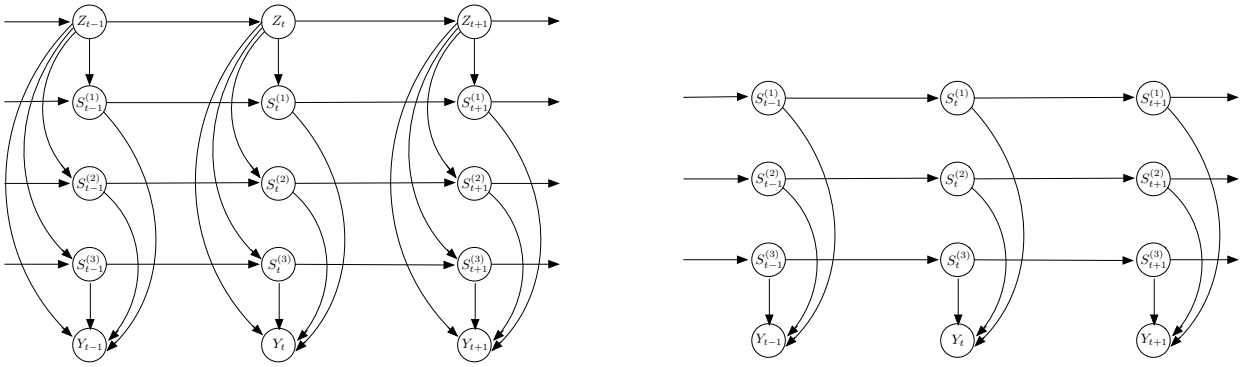

Figure 2. Interleaved mixture of hidden Markov models (left) and factorial hidden Markov model (right) in dynamic Bayesian network notation (for $M = 3$).

sitions to a new state and outputs the observation Y_t, while all other processes remain "frozen" in time.

More formally, let $\mu_1, ..., \mu_M$ be hidden Markov models with initial state probabilities $a_{0i}^{(m)}$, transition probabilities $a_{ij}^{(m)}$ and emission probabilities $b_{il}^{(m)}$. For ease of notation, we assume the number of states K is identical for all μ_m, but the model trivially generalizes to processes with state spaces of different size. Let furthermore $\bar{\mu}$ be a Markov process with states $\{1, ..., M\}$, initial state probabilities d_{0i} and transition probabilities d_{ij}. Let Z_t denote a random variable representing the state of $\bar{\mu}$ at time t, and $S_t^{(m)}$ denote random variables representing the state of process μ_m at time t for $1 \leq m \leq M$. $Z_t \in \{1, ..., M\}$ determines the *active* process at time t, and we will refer to $\bar{\mu}$ as the *switching process*. At every step t in time, a new active process is sampled from $\bar{\mu}$ with probability $P(Z_t = j \mid Z_{t-1} = i) = d_{ij}$. Afterwards, the states of $\mu_1, ..., \mu_M$ are updated according to

$$P(S_t^{(m)} = j \mid S_{t-1}^{(m)} = i, Z_t = k) = \begin{cases} a_{ij}^{(m)} & k = m; \\ \delta_{ij} & k \neq m, \end{cases} \tag{1}$$

where $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \neq j$. In other words, a process μ_m transitions to a new state with probability given by its transition matrix if it is active at time t, and stays in its old state otherwise.

Finally, the probability of emitting symbol Y_t is

$$P(Y_t = l \mid S_t^{(1)} = i_1, ..., S_t^{(M)} = i_M, Z_t = k) = b_{i_k l}^{(k)} \tag{2}$$

That is, it is given by the emission probability of the process that is active at time t. Let $\mathbf{S}_t = S_t^{(1)}, ..., S_t^{(M)}$, $\mathbf{Z} = Z_1, ..., Z_T$ and $\mathbf{S} = \mathbf{S}_1, ..., \mathbf{S}_T$. Then

$$P(\mathbf{Z}, \mathbf{S}, \mathbf{Y}) = \prod_{t=1}^{T} P(Z_t \mid Z_{t-1}) P(Y_t \mid \mathbf{S}_t, Z_t) \prod_{m=1}^{M} P(S_t^{(m)} \mid S_{t-1}^{(m)}, Z_t) \tag{3}$$

We will refer to this model as an *interleaved mixture of hidden Markov models*. It is represented by the dynamic Bayesian network structure given in Figure 2 (left). The model is structurally related to a factorial hidden Markov model (Ghahramani & Jordan, 1997), shown in Figure 2 (right). However, the structure is extended by the additional chain of Z_t nodes that determine the currently active process. Although the structure is densely connected, the set of parameters is simply the union of the parameter sets of the constituent HMMs $\mu_1, ..., \mu_M$ and the switching process $\bar{\mu}$.

The following alternative interpretation of the model can be given. Let $\mathbf{z}$ denote an interleaving[1] and let $t_1^m, ..., t_{T^m}^m$ denote the sequence positions for which $z_t = m$. That is, $\mathbf{Y}_{\downarrow \mu_m} = Y_{t_1^m}, ..., Y_{t_{T^m}^m}$ is the projection of $\mathbf{Y}$ to elements generated by μ_m, and $\mathbf{S}_{\downarrow \mu_m} = S_{t_1^m}^{(m)}, ..., S_{t_{T^m}^m}^{(m)}$ the corresponding hidden state variables. It is easily verified that

$$P(\mathbf{Z}, \mathbf{S}, \mathbf{Y}) = P(\mathbf{Z}) \prod_{m=1}^{M} P_{\mu_m}(\mathbf{Y}_{\downarrow \mu_m}, \mathbf{S}_{\downarrow \mu_m}),$$

where $P_{\mu_m}(\mathbf{Y}_{\downarrow \mu_m}, \mathbf{S}_{\downarrow \mu_m})$ is the joint distribution of hidden states $\mathbf{S}_{\downarrow \mu_m}$ and observations $\mathbf{Y}_{\downarrow \mu_m}$ in the original HMM μ_m. This reformulation gives rise to an intuitive approach for sampling from $\mathbf{Y}$: first sample an interleaving pattern $\mathbf{z}$ from $\bar{\mu}$, and afterwards $\mathbf{Y}_{\downarrow \mu_m}$ from μ_m for $1 \leq m \leq M$.

[1] In general, we denote random variables with uppercase letters, and their instantiations with lower-case letters.

3. Inference and Parameter Estimation

A key task in the activity recognition domains we have in mind is *hidden state inference*: find

$$(\mathbf{z}^*, \mathbf{s}^*) = \operatorname*{argmax}_{\mathbf{z},\mathbf{s}} P(\mathbf{z}, \mathbf{s} \mid \mathbf{y}) \tag{4}$$

for a given sequence $\mathbf{y}$ of observations. This involves simultaneously finding a segmentation of $\mathbf{y}$ into subsequences $\mathbf{y}_{\downarrow\mu_m}$ generated by μ_m (the $\mathbf{z}^*$), and most likely hidden states for $\mathbf{y}_{\downarrow\mu_m}$ in μ_m (the $\mathbf{s}^*$).

3.1. Exact Inference

Two special cases of the problem are trivial. For $M = 1$, the model coincides with a hidden Markov model, and the Viterbi algorithm returns the most likely hidden states in time $O(K^2T)$. Moreover, if the output symbol sets of $\mu_1, ..., \mu_M$ are disjoint, the interleaving $\mathbf{z}$ is directly observable, and $\mathbf{s}$ can be obtained by running M instances of Viterbi in time $O(MK^2T)$.

The more interesting case of $M \geq 2$ and non-disjoint output symbol sets is inherently more difficult due to its combinatorial nature—the M constituent chains are coupled via the switching process and observations, and thus cannot be handled independently. Accordingly, exact graphical model inference (e.g. with the junction tree algorithm) applied to the model in Figure 2 (left) has costs exponential in M, because the cliques at Y_t are of size $O(M)$. In fact, for *general* graphical model structures of this form there is no tractable inference algorithm available. However, the conditional distributions $P(Y_t \mid \mathbf{S}_t, Z_t)$ have a particularly simple form, which could make the problem easier. Unfortunately, this is not the case:

Theorem. *Exact inference for interleaved mixtures of hidden Markov models is NP-hard.*

The theorem is proved by reduction from the strongly NP-hard *3-partition problem* (Garey & Johnson, 1975):

Problem (3-partition problem)**.** *Let S be a multiset of $M = 3N$ positive integers. Is there a partition of S into subsets $S_1, ..., S_N$ of size 3 each such that the sum over the integers in each subset is the same?*

A detailed proof is omitted for lack of space. Intuitively, the relationship is that an interleaving of $\mu_1, ..., \mu_M$ "partitions" a given sequence into the parts generated by the different processes (cf. Figure 1). Note that a key issue is the strong NP-hardness of 3-partition: the problem is NP-hard even if numbers in the input are given in unary notation (or, equivalently, if integers in S are polynomially bounded in M).

3.2. Approximate Inference

Approximate inference in graphical models has received much attention, and a variety of techniques are available. The most simple class of methods are Markov chain Monte Carlo (MCMC) approaches. In Gibbs sampling, for instance, iterative conditional resampling of random variables defines a Markov process whose stationary distribution—under certain conditions—will be the conditional distribution in Equation (4). However, MCMC is not an effective inference method in our case, because the Markov process defined by the Gibbs sampler is not ergodic. There can be two state configurations with positive probability that cannot be transformed into each other by single-variable changes without passing through an invalid (probability zero) configuration, such as any configuration with $S_{t-1}^{(m)} \neq S_t^{(m)}$ but $Z_t \neq m$. This effectively traps the Gibbs sampler in a subspace of all configurations and prevents MCMC convergence.

The problem is that Gibbs sampling, by updating only one variable at a time, ignores the specific model structure. Instead, we have to resort to approximate inference methods that better exploit model structure. Examples include structured variational approximations (Ghahramani & Jordan, 1997) and an iterative approximate inference method known as the chainwise Viterbi algorithm (Saul & Jordan, 1999). These algorithms are used in factorial HMMs for computing EM statistics and hidden state inference. In the rest of the Section, we present an extension of chainwise Viterbi for solving the problem given by Equation (4).

The idea behind chainwise Viterbi is to repeatedly solve tractable sub-problems of the (intractable) global optimization problem. For factorial hidden Markov models, the natural sub-problem to solve is to optimize hidden states in one chain $\mathbf{S}^{(m)} = S_1^{(m)}, ..., S_T^{(m)}$ conditioned on the current states of the other chains:

$$\begin{aligned}\mathbf{s}_{new}^{(m)} &= \operatorname*{argmax}_{\mathbf{s}^{(m)}} P(\mathbf{s}^{(m)} \mid \{\mathbf{s}^{(l)} : l \neq m\}, \mathbf{y}) \\ &= \operatorname*{argmax}_{\mathbf{s}^{(m)}} P(\mathbf{s}^{(1)}, ..., \mathbf{s}^{(M)}, \mathbf{y}).\end{aligned}$$

In the dynamic Bayesian network representing an interleaved mixture of HMMs (Figure 2, left), there are two different types of hidden chains: the chains $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$ representing the constituent processes $\mu_1, ..., \mu_M$ and the chain $\mathbf{Z}$ representing the switching process $\bar{\mu}$. Assume first that $\mathbf{Z}$ is kept fix, and the goal is to conditionally optimize a chain $\mathbf{S}^{(m)}$. This is straightforward: for a given interleaving pattern, the chains $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$ become independent given $\mathbf{Z}$ and $\mathbf{Y}$ due to the special form of the conditional distribu-

Algorithm 1 Chainwise Viterbi for interleaved mixtures of hidden Markov models

Input: model $\mathcal{M}$, observations $\mathbf{Y}$
$(\mathbf{S}, \mathbf{Z}) :=$ consistent-configuration($\mathcal{M}$)
while not converged **do**
  choose $m, n \in \{1, ..., M\}, m \neq n$
  let $(\mathbf{S}^{(m)}, \mathbf{S}^{(n)}, \mathbf{Z}) := \underset{\mathbf{S}^{(m)}, \mathbf{S}^{(n)}, \mathbf{Z}}{\operatorname{argmax}} P(\mathbf{S}, \mathbf{Z}, \mathbf{Y})$
end while
return $\mathbf{S}, \mathbf{Z}$

tions $P(Y_t \mid \mathbf{S}_t, Z_t)$, cf. Equation (2). They can thus be optimized independently with standard Viterbi.

We therefore focus on the task of optimizing $\mathbf{Z}$ given $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$. A straightforward update

$$\mathbf{z}_{new} = \underset{\mathbf{z}}{\operatorname{argmax}} P(\mathbf{z}, \mathbf{s}^{(1)}, ..., \mathbf{s}^{(M)}, \mathbf{y})$$

is not very effective: as a process μ_m can only change state at time t if it is active, we know from $S_t^{(m)} \neq S_{t-1}^{(m)}$ that $Z_t = m$. Thus, the joint state of $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$ essentially determines $\mathbf{Z}$. To change the state of Z_t from m to n, it is necessary to also update $S_t^{(m)}$ and $S_t^{(n)}$ to reflect that μ_n is now active at time t. The solution is to jointly optimize *two* constituent chains $\mathbf{S}^{(m)}$, $\mathbf{S}^{(n)}$ and the switching chain $\mathbf{Z}$ by

$$(\mathbf{s}_{new}^{(m)}, \mathbf{s}_{new}^{(n)}, \mathbf{z}_{new}) = \underset{\mathbf{s}^{(m)}, \mathbf{s}^{(n)}, \mathbf{z}}{\operatorname{argmax}} P(\mathbf{s}, \mathbf{y}, \mathbf{z}). \qquad (5)$$

Intuitively speaking, this update allows to re-assign observations that have so far been attributed to process μ_m to process μ_n, by changing some Z_t from m to n and updating $\mathbf{S}^{(m)}$ and $\mathbf{S}^{(n)}$ accordingly. If it is repeatedly applied with different process indices m, n, the interleaving can be arbitrarily revised. Algorithm 1 describes this chainwise update scheme in pseudocode. The method consistent-configuration($\mathcal{M}$) initializes the states of the hidden variables to some positive-probability configuration[2]. When choosing $m, n \in \{1, ..., M\}$ different strategies are possible; we assume the algorithm repeatedly cycles through all pairs $n \neq m$. If the update step (5) is implemented exactly, $P(\mathbf{s}, \mathbf{z}, \mathbf{y})$ will increase unless the hidden state configuration is left unchanged. Thus, the algorithm will always converge (though not necessarily to the true global optimum).

An efficient implementation of the update step (5) is crucial for fast inference. This can be achieved by dynamic programming in the spirit of the Viterbi algorithm (Rabiner, 1989). Moreover, the particularly restrictive form of the model (basically, that only the active chain changes state at any point in time) can be exploited. This allows much faster inference than for *general* graphical models with the DAG structure given in Figure 2 (left), as will be briefly outlined now.

To simplify notation, assume that $n = 1$ and $m = 2$. In analogy to the Viterbi algorithm, define

$$\delta_{ijk}[t] = \max_{\mathbf{D}} P(\mathbf{D}, S_t^{(1)} = i, S_t^{(2)} = j, Z_t = k, \mathbf{Y}, \mathbf{S}^{(3)}, ..., \mathbf{S}^{(M)})$$

with

$$\mathbf{D} = \{S_1^{(1)}, ..., S_{t-1}^{(1)}, S_1^{(2)}, ..., S_{t-1}^{(2)}, Z_1, ..., Z_{t-1}\}.$$

Initialization of $\delta_{ijk}[1]$ is straightforward. For the recursive definition of $\delta_{ijk}[t]$, let

$$C[k] = \prod_{m=3}^{M} P(S_t^{(m)} = s_t^{(m)} \mid S_{t-1}^{(m)} = s_{t-1}^{(m)}, Z_t = k),$$

where $s_{t-1}^{(m)}$, $s_t^{(m)}$ for $m \geq 3$ denote the current values of the fixed chains $\mu_3, ..., \mu_M$. Now two cases have to be considered. If $k \geq 3$, chains $1, 2$ cannot have changed state, and

$$\delta_{ijk}[t] = \max_{k'=1,...,M} \delta_{ijk'}[t-1] d_{k'k} b_{sy}^{(k)} C[k]$$

with $s = S_t^{(k)}$ and $y = Y_t$. This quantity can be computed in time $O(M)$. If $k \in \{1, 2\}$, we have to take into account state changes on the chains being optimized. Assume without loss of generality that $k = 1$. Now

$$\delta_{ij1}[t] = \max_{k'=1,...,M} \max_{i'=1,...,K} \delta_{i'jk'}[t-1] d_{k'1} a_{i'i}^{(1)} b_{iy}^{(1)} C[1],$$

with $y = Y_t$. This quantity can be computed in time $O(KM)$. There are $O(K^2MT)$ values of the form $\delta_{ijk}[t]$ to compute. However, time for computing all values is bounded by $O(K^2M(M+K)T)$, as the case $k \in \{1, 2\}$ only appears $O(K^2T)$ times.

The maximum probability of a hidden state configuration is

$$\max_{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{z}} P(\mathbf{s}, \mathbf{z}, \mathbf{y}) = \max_{ijk} \delta_{ijk}[T],$$

and a maximizing configuration is found by keeping track of where maxima occur in backtrace variables.

It is instructive to compare the complexity of the outlined chainwise Viterbi algorithm to inference in an HMM where hidden states are "flattened" into a single process. This HMM has a state space of size KM, and standard Viterbi has thus complexity $O(K^2M^2T)$, similar to the $O(K^2M(M+K)T)$ for a single update step in chainwise Viterbi. However, several such update steps will be needed before convergence.

[2] This is trivial if observation probabilities are always non-zero, as e.g. in Laplace-smoothed models.

3.3. Parameter Estimation

There are different possible settings for learning the proposed model from data. In the activity recognition setting discussed in Section 4, both sensor observations and activities are given for the training set. In this fully observable case maximum-likelihood model parameters can essentially be determined by counting. More generally, if the interleaving is known for the training data (that is, we know which part of each sequence has been generated by which process), the problem reduces to independently estimating the parameters of $\mu_1, ..., \mu_M$ with the standard Baum-Welch algorithm (Rabiner, 1989). In an unsupervised learning setting, expectation-maximization including the unknown interleaving $\mathbf{Z}$ is a natural choice. However, for the same reasons as discussed in Section 3, exact computation of the expectation step will be infeasible. In factorial hidden Markov models, this problem is solved elegantly by a structured variational approximation, and exploring variational inference methods for the interleaved mixture model presented in this paper is an interesting direction for future work. A simple alternative is to employ *hard EM*: instead of computing exact expectations, hidden states are set to their max-likelihood values given the observations, and expectations determined by counting. Together with the chainwise Viterbi algorithm discussed in Section 3.2 this yields a tractable method which is straightforward to implement.

4. Experimental Evaluation

The proposed model has been evaluated in an activity of daily living (ADL) recognition domain, where the goal is to infer a user's activity from a stream of dense RFID sensor data. The dataset has been collected in a real RFID environment at Intel Research Seattle (Landwehr et al., 2007). Objects are equipped with small RFID tags, and the user is wearing a lightweight RFID reader in a bracelet around the wrist. Whenever the reader comes close (10–15 centimeters) to a tagged object, the object tag is recorded. The sequence of observed tags thus indicates the objects a user has been interacting with while performing the activity.

We recorded activities involved in making breakfast at home, as this domain showcases the kind of interleaving behavior we are interested in (cf. Figure 1). The dataset consists of 20 sequences of RFID tag observations collected from 5 different persons having breakfast. Sequences are hand-labeled with the true current activity based on a human observer. There are 18 basic activities organized into 6 high-level activities, 24 different classes of tagged objects (including *nil* if no object was observed), and a total of 4597 timepoints to be classified. Timepoints at which no activity is taking place and activities with a coverage of less than 1% were removed, leaving 14 activities and 3545 timepoints in the dataset. The average number of segments into which a high-level activity is broken up because of interleaving is 3.95. There is significant overlap between observations associated with different activities, either because the same object is used in different activities or noise in the sensor data. More specifically, the average overlap in the set of observations associated with two different activities is 40.6%.

A standard approach in ADL recognition is based on HMMs: each basic activity corresponds to a hidden state, and sensor data to observations. In the described domain this means that all activities are "flattened" into one hidden process, and their hierarchical structure is lost. This approach will serve as a baseline, denoted by HMM. Alternatively, high-level activities can be modeled as separate hidden processes using the model described in Section 2. Here we consider a slight extension of this model: state transition probabilities in the active hidden process μ_{Z_t} depend not only on the previous state but also on whether or not the process has just become active; that is, $Z_t \neq Z_{t-1}$. The motivation for this extension is that high-level activities are typically interrupted at a point where the basic activity changes as well. It is straightforward to generalize the model and algorithms discussed in Section 2 and Section 3 to include this dependency.

Each high-level activity A is represented as a process μ_A, and the state space of μ_A are the basic activities associated with A. Note that the method, when applied to a given observation sequence, will automatically chose the (approximately) most likely subset of high-level activities that explains the observations. This model, together with the approximate inference technique discussed in Section 3.2 will be denoted as HMMmix. In the chainwise Viterbi algorithm, hidden states are initialized to the most likely activity given the current sensor observation (as observed in the training data). Furthermore, a version with exact inference (denoted HMMmix*) is run for comparison.

The experimental study seeks to answer the following two questions:

(Q1) Does reconstruction accuracy increase if high-level activities are modeled as separate processes?

(Q2) Does the approximate inference algorithm for HMMmix yield results similar to exact inference?

The rationale behind (**Q1**) is that modeling high-level

Table 1. Average cross-validated accuracy for MAJORITY, MAJORITY/OBSERVATION, HMM, HMMMIX and HMMMIX* on the ADL dataset. • indicates that result for HMMMIX is significantly better than result for other method (paired two-sided t-test, $p = 0.05$).

Method	Accuracy
MAJORITY	21.2 ± 25.4•
MAJORITY/OBSERVATION	71.4 ± 10.3•
HMM	84.0 ± 9.8•
HMMMIX	$\mathbf{86.0} \pm 8.6$
HMMMIX*	$\mathbf{86.0} \pm 8.6$

activities as separate processes will capture transition dynamics more concisely, as it decouples dynamics within a high-level activity from the switching dynamics. This is reflected in the number of model parameters: The "flattened" HMM representation requires $O((MK)^2) = O(M^2K^2)$ parameters to specify transition dynamics, while HMMMIX only requires $O(M^2 + MK^2)$ parameters.

To evaluate the different approaches, we performed a leave-one-sequence-out cross-validation. On the respective training set, models are estimated from fully observable training data, i.e., information on both sensor observations and activities is available. Given a test sequence, the most likely joint state of hidden variables in the model is determined, yielding a prediction of the current basic activity at every point in time. This is compared against the known true activity, and average prediction accuracy is computed. Table 1 shows reconstruction accuracy for HMM, HMMMIX and HMMMIX*. Additionally, accuracy for always predicting the most frequent activity (MAJORITY), and the most frequent activity given a particular sensor observation (MAJORITY/OBSERVATION) are shown. HMMMIX significantly outperforms HMM (paired two-sided t-test, $p = 0.05$), and predictions made by HMMMIX and HMMMIX* are identical in this experiment. This affirmatively answers questions **Q1** and **Q2**. Figure 3 shows the convergence behavior of chainwise Viterbi. The normalized log-likelihood of the current configuration of hidden states and the reconstruction accuracy given by this configuration are plotted as a function of the algorithm iteration. As expected, both likelihood and accuracy increase as the algorithm repeatedly revises the current interleaving. Furthermore, convergence occurs after a small number of iterations.

There are two sources of information for predicting the activity at a point t in time: the current sensor observation, and transition dynamics for activities

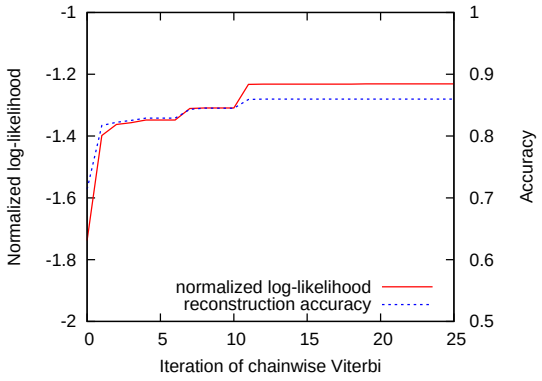

Figure 3. Normalized log-likelihood and reconstruction accuracy of the current hidden state configuration as a function of the number of iterations in chainwise Viterbi. Results are averaged over all test sequences.

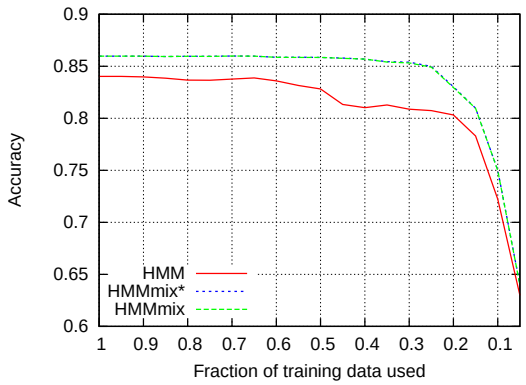

Figure 4. Reconstruction accuracy as a function of the fraction γ of training sequences used to estimate model transition parameters, while emission probabilities are estimated from all available training data. Results are averaged over 5 runs of cross-validation.

(which capture the influence of past and future observations on the current prediction). The MAJORITY/OBSERVATION approach already performs well; this indicates that much information is obtained simply from the current sensor observation. To further investigate the influence of transition dynamics on reconstruction accuracy, the following experiment was carried out. When estimating a model from data, only a randomly selected fraction γ of the training sequences is used to estimate transition probabilities, while all available data is used to estimate emission probabilities. Figure 4 shows reconstruction accuracy as a function of γ. The experiment confirms that HMMMIX outperforms HMM, and that approximate inference gives solutions very close to those of exact inference (solutions differ slightly, but the curves for HMMMIX and HMMMIX* in Figure 4 are indistin-

guishable). Moreover, the difference between HMM and HMMmix is most pronounced if only 20% to 40% of training sequences are used to estimate transition parameters. This supports the hypothesis that the more concise representation of transition dynamics in HMMmix (with fewer model parameters) explains its superior performance, as a concise representation matters most if training data is sparse.

5. Conclusions and Related Work

We have introduced a model for interleaved mixtures of hidden processes, which was shown to be superior to a single-process model in an activity recognition domain. The model should be generally applicable in situations where only the interleaved output of several independent processes can be observed. Related work includes several extensions of hidden Markov models (as discussed in Section 1), and activity recognition approaches based on HMMs such as (Patterson et al., 2005) and (Zhang et al., 2007) or dynamic Bayesian networks (Wang et al., 2007). The proposed method not only labels sequence positions but returns a structured parse of the sequence in terms of a set of hidden processes. Thus, it is also related to segmentation models, grammar-based approaches, and more generally models for predicting structured data (see (Bakir et al., 2007) for an overview). Directions for future work include semi- and unsupervised learning settings, and testing the model in different domains and on larger activity recognition datasets.

Acknowledgments The author would like to thank Matthai Philipose and Intel Research Seattle for making the activity recognition dataset available, and Luc De Raedt, Ingo Thon, Bernd Gutmann and Siegfried Nijssen for helpful discussions. The comments from the anonymous reviewers also helped to improve the paper. This work was supported by the Research Foundation-Flanders (FWO-Vlaanderen), and GOA/08/008 project "Probabilistic Logic Learning".

References

Altman, R. M. (2007). Mixed hidden Markov models: An extension of the hidden Markov model to the longitudinal data setting. *Journal of the American Statistical Association*, *102*, 201–210.

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

Batu, T., Guha, S., & Kannan, S. (2004). Inferring Mixtures of Markov Chains. *Proceedings of the 17th Annual Conference on Learning Theory*.

Brand, M. (1997). *Coupled hidden Markov models for modeling interactive processes* (Technical Report 405). MIT Media Lab.

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

Garey, M. R., & Johnson, D. S. (1975). Complexity Results for Multiprocessor Scheduling under Resource Constraints. *SIAM Jour. Comp.*, *4*, 397–411.

Ghahramani, Z., & Hinton, G. E. (1998). *Switching State-Space Models* (Technical Report). Department of Computer Science, University of Toronto.

Ghahramani, Z., & Jordan, M. I. (1997). Factorial Hidden Markov Models. *Mach. Lear.*, *29*, 245–273.

Girolami, M., & Kabán, A. (2003). Simplicial Mixtures of Markov Chains: Distributed Modelling of Dynamic User Profiles. *Proc. of the 17th Ann. Conference on Neural Information Processing Systems*.

Jordan, M. I., Ghahramani, Z., & Saul, L. K. (1996). Hidden Markov Decision Trees. *Proceedings of the 9th Conference on Advances in Neural Information Processing Systems*.

Landwehr, N., Gutmann, B., Thon, I., Philipose, M., & De Raedt, L. (2007). Relational Transformation-based Tagging for Human Activity Recognition. *Proc. of the Intern. Workshop on Knowledge Discovery from Ubiquitous Data Streams*.

Patterson, D., Fox, D., Kautz, H., & Philipose, M. (2005). Fine-Grained Activity Recognition by Aggregating Abstract Object Usage. *Proc. of the 9th IEEE Intern. Symp. on Wearable Computers*.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Saul, L. K., & Jordan, M. I. (1999). Mixed Memory Markov Models: Decomposing Complex Stochastic Processes as Mixtures of Simpler Ones. *Machine Learning*, *37*, 75–87.

Wang, S., Pentney, W., Popescu, A.-M., Choudhury, T., & Philipose, M. (2007). Common Sense Based Joint Training of Human Activity Recognizers. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Zhang, W., Chen, F., Xu, W., & Cao, Z. (2007). Decomposition in hidden Markov models for activity recognition. *Multimedia Content Anal. and Mining*.