Representing hierarchical POMDPs as DBNs, with applications to mobile robot navigation

Kevin P. Murphy
www.ai.mit.edu/∼murphyk

4 November 2002

1 Hierarchical HMMs

Hierarchical HMMs (HHMMs) [FST98] generalize HMMs by allowing the states of the stochastic automaton to emit single observations or strings of observations. Those that emit single observations are called "production states", and those that emit strings are termed "abstract states". The strings emitted by abstract states are themselves governed by sub-HHMMs, which can be called recursively. When the sub-HHMM is finished, control is returned to whichever state it was called from; the calling context is memorized using a depth-limited stack.

This call and return semantics is similar to recursive transition networks (RTNs) and stochastic context free grammars (SCFGs). However, HHMMs are finite state models (the depth of the hierarchy is fixed), and hence are less powerful than RTNs and SCFGs, which can model unbounded recursion. The advantage of HHMMs is that inference and learning is faster than in SCFGs. In particular, inference in an HHMM can be done in $O(T)$ time [MP01], where T is the length of the sequence, whereas inference in SCFGs (using the inside-outside algorithm) takes $O(T^3)$ time [JM00]. This also means learning, which uses inference as a subroutine, is faster.

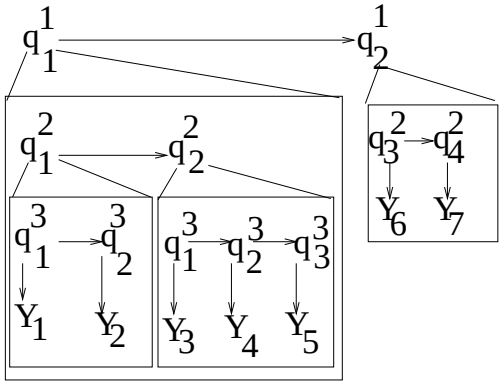

Figure 1: A sequence $y_{1:7}$ is hierarchically divided into segments, each of which is modelled by an HMM. The transition/observation distributions of the sub-HMMs may be conditioned on their immediate context, or their entire surrounding context. In a context free system, information is not allowed to flow across segment boundaries, but this restriction can be lifted.

HHMMs are closely related to semi-Markov HMMs, segment HMMs, abstract HMMs, embedded HMMs, and a variety of other models. Please see Figure 1 for the basic idea, and [Mur02, ch.2] for details.

2 Hierarchical POMDPs

A hierarchical POMDP [TRM01, PT02] is just an HHMM with inputs and a cost/reward function. For any POMDP, there are three problems to solve: planning, inference and (model) learning. Planning means computing a mapping

(called a policy) from belief states, $P(X_t|y_{1:t}, u_{1:t})$, to actions, so as to maximize the expected reward. (X_t is the hidden state, Y_t is the observation, and U_t is the action.) We shall not consider the planning problem in this paper. Inference comes in two forms: filtering, which means computing the belief state online, and smoothing, which means computing $P(X_t|y_{1:T}, u_{1:T})$ offline. The output of filtering is fed to the controller (policy); the output of smoothing is fed to the (offline) learner, which tries to find maximum likelihood estimates of the model parameters. For the purposes of inference and learning, we assume the controller is fixed, so the input sequence is always known; hence we can ignore the reward function.

3 Example of an HPOMDP applied to robot navigation

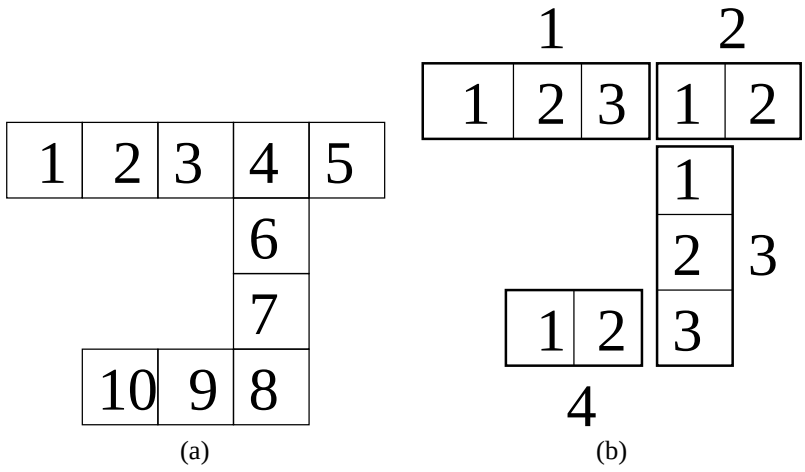

Figure 2: (a) A small flat map, and (b) a hierarchical version. Boxes group together corridors. Numbers outside the boxes specify which corridor, numbers inside the boxes specify which cell within the corridor.

We now give a simple example of a 2-level HPOMDP from the domain of mobile robot navigation. Consider the grid-world shown in Figure 2. On the left is a flat representation of space, and on the right is a hierarchical representation. Clearly there is a 1:1 mapping between the flat and the hierarchical representation. However, the hierarchical representation is easier to learn and to plan with: learning is easier because we can potentially re-use sub-models (this is a form of parameter tying), and planning is easier because we can work at a higher level of abstraction, i.e., we can reason in terms of "macro" actions, such as "go down corridor", instead of primitive actions such as "go forward 1m".[1] We now discuss how to create such hierarchical models.

In the flat representation, the state (position) of the robot can be represented by a single discrete random variable, $Q_t \in \{1, \ldots, 10\}$.[2] In the hierarchical representation, the position of the robot is defined using two variables, denoted by $Q_t^2 \in \{1, 2, 3, 4\}$ and $Q_t^1 \in \{1, 2, 3\}$; these represent which corridor the robot is in, and which cell within the corridor, respectively. The corridor states are abstract, because they emit sequences of observations; the cell states are concrete, and emit single observations. For simplicity, let us suppose each concrete state emits a unique symbol: $(Q^2 = 1, Q^1 = 1)$ emits 'a', $(Q^2 = 1, Q^1 = 2)$ emits 'b', etc. Also, let us focus attention on a smaller problem, consisting just of corridors 1 and 2. Suppose the robot has two actions, move-left and move-right, and suppose these succeed only 90% of the time. The resulting hierarchical state transition diagram is shown in Figure 3, and is modelled after Figure 3.2 of [The02, p.65].

Let us illustrate the generative process by sampling from the state transition diagram (see Figure 4). First we (simulating the HHMM) pick a corridor to start in, say $Q_1^2 = 1$. Then we make a "vertical transition" into a cell within

[1] In addition, the higher levels of the hierarchy usually have lower entropy (i.e., we are usually more certain of which corridor we are in than which cell within a corridor), which further simplifies planning. However, we do not discuss the planning problem (how to choose actions) in this paper.

[2] We can keep track of the orientation of the robot using a separate random variable, θ_t. This is orthogonal to the issue of hierarchical representation of space, so we shall ignore it. Note, however, that the DBN representation lets us factor out orientation from position in a way which is impossible in the standard HHMM framework.

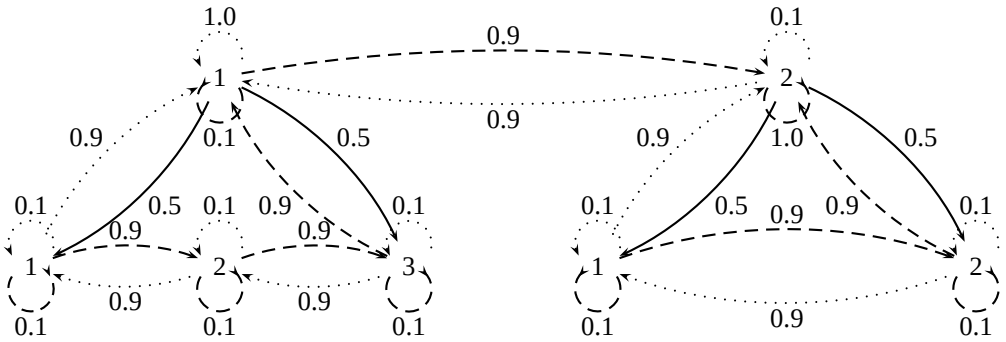

Figure 3: State transition diagram for a 2-level HPOMDP, corresponding to the map in Figure 2. Dotted arcs represent the go-left action, dashed arcs represent the go-right action, solid arcs (labelled with 0.5) represent entering a sub-HHMM under either action. Most actions only succeed with probability 0.9; failure means the state does not change.

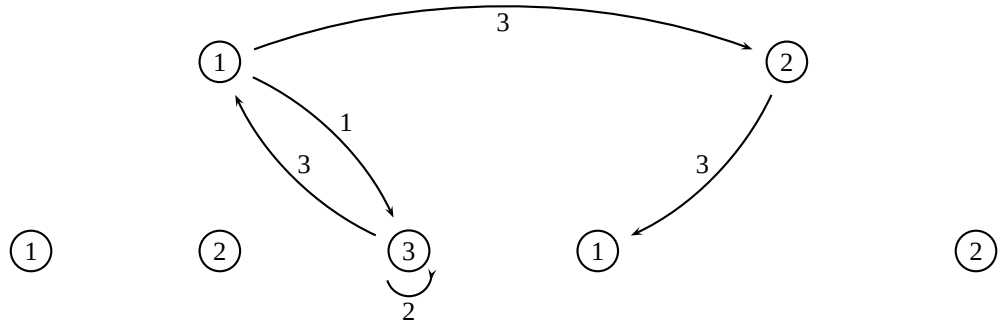

Figure 4: A sample path through the state transition diagram of Figure 3. The arcs labelled '3' represent leaving sub-HMM 1, making a horizontal transition from abstract state 1 to abstract state 2, and then making a vertical transition into sub-HMM 2.

corridor 1; in this case, we can either enter 1 or 3, say we choose $Q_1^1 = 3$. Now the state of the system is fully defined, namely $(Q_1^2 = 1, Q_1^1 = 3)$, so we emit observation 'c'. Suppose the robot decides to take the go-left action; with 90% probability we transition to cell 2, and with 10% probability we remain in cell 3 (this represents the fact that the robot actions are not perfectly reliable). Suppose the action fails, so we remain in cell 3, i.e., $(Q_2^2 = 1, Q_2^1 = 3)$; hence at time $t = 2$, we generate another observation 'c'. Suppose the next action the robot chooses is go-right. With 90% probability, this will let us exit this sub-HMM (corridor), and return control to the abstract state we came from. Suppose the go-right action succeeds. Now that the sub-HMM has finished, we are free to choose a new abstract state (corridor); suppose we transition to $Q_3^2 = 2$. From here, suppose we make a vertical transition into cell $Q_3^1 = 1$; then we emit 'd'. And so on.

4 Representing a 2-level HPOMDP as a DBN

We can represent a 2-level HPOMDP as a DBN as shown in Figure 5. $U_t \in \{\leftarrow, \rightarrow\}$ represents the go-left or go-right action (control), which we consider as an exogeneous random variable (i.e., we do not model what causes it); $Q_t^2 \in \{1, 2\}$ and $Q_t^1 \in \{1, 2, 3\}$ represent the corridor and cell (note that if $Q_t^2 = 2$, we restrict the state-space $Q_t^1 \in \{1, 2\}$, since corridor 2 has length 2); O_t can either be a discrete or continuous observation; and F_t^1 is a binary variable that indicates if the sub-HMM at level 1 has finished or not, i.e., if we have just left a corridor. We will explain

this more precisely below.

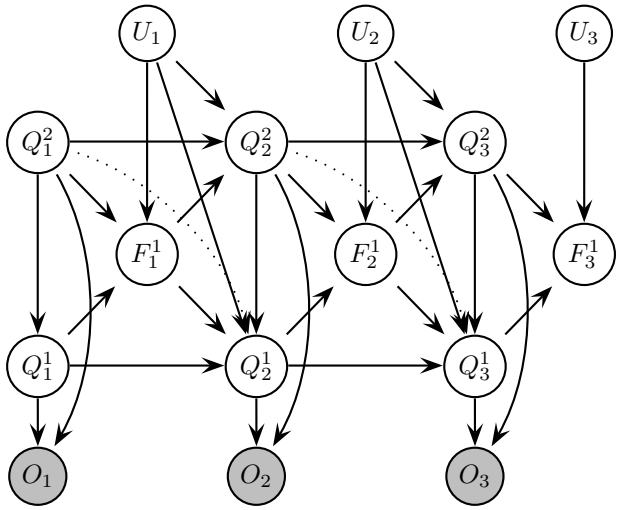

Figure 5: A 2-level HPOMDP represented as a DBN. The dotted arc is non-standard, and will be explained in Section 4.3.2. At time t, the system is in state (Q_t^2, Q_t^1), the agent takes action U_t, and F_t^1 turns on or off (representing exiting or not from a sub-HMM); finally the new state at $t+1$ is chosen, based on the old state, F_t^1 and U_t. Notice that F_t^1 does not affect O_t: the termination of a sub-process is not directly visible, and must be inferred.

Before defining the parameters of the model, below we show the assignment of values to the nodes which corresponds to the sample path shown in Figure 4.

U_t	$\leftarrow$	$\rightarrow$	$-$
A_t	1	1	2
F_t	0	1	0
C_t	3	3	1
O_t	c	c	d

This corresponds to the following scenario: the agent starts in corridor 1, cell 3 and observes 'c'; then it chooses to move left, but this action fails, so it remains in corridor 1, cell 3 and again observes 'c'; then it chooses to move right, which causes it to leave corridor 1 (represented by $F_2 = 1$) and enter corridor 2, cell 1, where upon it observes 'd'.

4.1 Abstract (top level) nodes

The conditional probability distribution (CPD) for the abstract node is defined as follows:

$$P(Q_t^2 = j | U_{t-1} = u, Q_{t-1}^2 = i, F_{t-1}^1 = f) = \begin{cases} T_u^2(i,j) & \text{if } f = 1 \\ \delta(i,j) & \text{if } f = 0 \end{cases}$$

If $F_{t-1}^1 = 0$, it means we have not exited a corridor (finished the sub-HMM), so the abstract state cannot change. Otherwise, the new abstract state is determined by the transition matrix for the abstract level, which we denote by $T_u^2(i,j)$. For example, if u is the go-right action, we have

$$T_{\rightarrow}^2 = \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix}$$

The $T_{\rightarrow}^2$ and $T_{\leftarrow}^2$ matrices encode the top level topology of the environment. We can write out the whole CPD explicitely as follows, where $-$ denotes a "don't care" symbol.

U_{t-1}	Q^2_{t-1}	F^1_{t-1}	$P(Q^2_t = 1, 2)$
$\rightarrow$	1	1	$(0.1, 0.9)$
$\rightarrow$	2	1	$(0.0, 1.0)$
$\leftarrow$	1	1	$(1.0, 0.0)$
$\leftarrow$	2	1	$(0.9, 0.1)$
-	1	0	$(1.0, 0.0)$
-	2	0	$(0.0, 1.0)$

4.2 Termination nodes for bottom level

The CPD for F^1_t, $P(F^1_t = 1|Q^2_t = k, Q^1_t = i, U_t = u)$, is the probability of exiting corridor k from cell i under action u. For example, $P(F^1_t = 1|U_t = \rightarrow, Q^2_t = 1, Q^1_t = \cdot) = [0, 0, 0.9]$, which means we have a 90% chance of exiting from cell 3 of corridor 1 if we take the go-right action. We can explicitly define the complete CPD as follows. (Some rows are labelled "undefined" because corridor 2 only contains 2 concrete states, not 3.) Those values of Q^1_t for which F^1_t has a non-zero probability of turning on will be called exit states, since it is possible to leave the sub-HMM via them.

U_t	Q^2_t	Q^1_t	$P(F^1_t = 0, 1)$
$\rightarrow$	1	1	$(1.0, 0.0)$
$\rightarrow$	1	2	$(1.0, 0.0)$
$\rightarrow$	1	3	$(0.1, 0.9)$
$\rightarrow$	2	1	$(1.0, 0.0)$
$\rightarrow$	2	2	$(0.1, 0.9)$
$\rightarrow$	2	3	undefined
$\leftarrow$	1	1	$(0.1, 0.9)$
$\leftarrow$	1	2	$(1.0, 0.0)$
$\leftarrow$	1	3	$(1.0, 0.0)$
$\leftarrow$	2	1	$(0.1, 0.9)$
$\leftarrow$	2	2	$(1.0, 0.0)$
$\leftarrow$	2	3	undefined

4.3 Concrete (bottom level) nodes

The CPD for the concrete node is defined as follows:

$$P(Q^1_t = j|Q^2_{t-1} = k, Q^1_{t-1} = i, Q^2_t = l, F^1_{t-1} = f, U_{t-1} = u) = \begin{cases} \pi^1_{u,k,i,l}(j) & \text{if } f = 1 \\ T^1_{u,l}(i, j) & \text{if } f = 0 \end{cases}$$

We will consider the $f = 0$ and $f = 1$ cases separately.

4.3.1 Horizontal transitions

If $F^1_{t-1} = 0$, it means we have not finished the sub-HMM, i.e., we are still inside a corridor; in this case, we define $T^1_{u,l}(i, j)$ to be the (horizontal) transition matrix for corridor l given the control u. (Note that if $F^1_{t-1} = 0$, then $Q^2_{t-1} = Q^2_t$, so the dotted link from Q^2_{t-1} to Q^1_t is redundant.) For example,

$$T^1_{\rightarrow,1} = \begin{pmatrix} 0.1 & 0.9 & 0 \\ 0 & 0.1 & 0.9 \\ 0 & 0 & 1 \end{pmatrix}$$

The bottom right entry is 1.0 because moving right from cell 3 in corridor 1 causes us to leave the sub-model (F^1 to turn on) with 90% probability, which is already accounted for in the CPD for F^1; if we fail to exit, we remain in cell 3 with probability 1.

Note that all corridors of the same length will have the same transition matrix. Shorter corridors will have more zeros, representing unreachable states, but still have the same chain-like structure. Hence we can define the transition matrix in terms of the structure and a single parameter, representing the reliability of the robot's actions.

We can explicitly define the complete transition matrix as follows.

$C_{t-1} = i$	$U_{t-1} = u$	$A_t = k$	$T^C_{u,k}(i, \cdot)$
1	$\rightarrow$	1	$(0.1, 0.9, 0.0)$
2	$\rightarrow$	1	$(0.0, 0.1, 0.9)$
3	$\rightarrow$	1	$(0.0, 0.0, 1.0)$
1	$\rightarrow$	2	$(0.1, 0.9, 0.0)$
2	$\rightarrow$	2	$(0.0, 1.0, 0.0)$
3	$\rightarrow$	2	undefined
1	$\leftarrow$	1	$(1.0, 0.0, 0.0)$
2	$\leftarrow$	1	$(0.9, 0.1, 0.0)$
3	$\leftarrow$	1	$(0.0, 0.9, 0.1)$
1	$\leftarrow$	2	$(1.0, 0.0, 0.0)$
2	$\leftarrow$	2	$(0.9, 0.1, 0.0)$
3	$\leftarrow$	2	undefined

4.3.2 Vertical transitions

Now consider the case that $F^1_{t-1} = 1$. In the original HHMM formulation, once we have exited a corridor, we are free to choose a new abstract state, and then a new concrete state within it, but the new concrete state only depends on the new abstract state and not on any previous information. (The reason is that the previous subroutine has finished, and its internal state, (Q^2_{t-1}, Q^1_{t-1}), is no longer available. This is similar to a context free assumption.)

This is clearly a problem for the robot navigation domain, because whether we enter cell 1 or 3 within corridor 2 depends on from which cell we exited corridor 1. Hence we need to add the dotted link from Q^2_{t-1} to Q^1_t shown in Figure 5. (If $F^1_{t-1} = 0$, this link is uninformative, since $Q^2_{t-1} = Q^2_t$.) We then define the vertical transition vector as

$$P(Q^1_t = j | U_{t-1} = u, Q^2_{t-1} = k, Q^1_{t-1} = i, Q^2_t = l, F_{t-1} = 1) = \pi^1_{u,k,i,l}(j)$$

However, we need to be very careful: defining a mapping from (Q^2_{t-1}, Q^1_{t-1}) to (Q^2_t, Q^1_t) requires specifying $O\left((|Q^2| \times |Q^1|)^2\right)$ parameters, where $|Q^2|$ is the number of corridors (abstract states), and $|Q^1|$ is the length of the longest corridor ($|Q^2| = 2$ and $|Q^1| = 3$ in our example). This is no better than specifying the flat model directly!

Fortunately, this mapping is very sparse. In particular, we only need to define $\pi^C_{u,k,i,l}(j)$ when $(Q^2_{t-1} = k, Q^1_{t-1} = i)$ represents an exit state, for only in such states can $F^1_t = 1$. Furthermore, for each such exit state, we can restrict U_{t-1} to those actions that cause an exit, and we can restrict Q^2_t to those abstract states that could result from such an action (i.e., to entry points). In our example, we have

$U_{t-1} = u$	$Q^2_{t-1} = k$	$Q^1_{t-1} = i$	$Q^2_t = l$	$\pi^1_{u,k,i,l}(\cdot)$	comments
$\rightarrow$	1	3	1	$(0, 0, 1)$	fail to enter LHS of cor 2
$\rightarrow$	1	3	2	$(1, 0, 0)$	enter LHS of cor 2
$\rightarrow$	2	2	2	$(0, 1, 0)$	bounce off RHS of cor 2
$\leftarrow$	1	1	1	$(1, 0, 0)$	bounce off LHS of cor 1
$\leftarrow$	2	1	1	$(0, 0, 1)$	enter RHS of cor 1
$\leftarrow$	2	1	2	$(1, 0, 0)$	fail to enter RHS of cor 1
					otherwise undefined

5 HHMMs with multiple levels

It is easy to generalize the DBN to handle HPOMDPs with multiple levels. (For simplicity, we ignore the input nodes, and focus on HHMMs.) For example, Figure 6 shows a DBN for a 3 level hierarchy, where the position of the robot is specified by (Q^3, Q^2, Q^1), representing floor, corridor and cell. Intuitively, the downward going arcs between the Q nodes represent the fact that one HMM calls another as a subroutine, and the upward going arcs between the F nodes represent signals from below that allow higher levels to change state only when the lower levels have finished.

The CPDs are defined as before, except for the F nodes. The HMM at level d is only able to finish if its child process has finished, i.e., F_t^d is only able to turn on if $F_t^{d-1} = 1$. Hence we define

$$P(F_t^d = 1 | Q_t^d = i, Q_t^{1:d-1} = k, F_t^{d-1} = b) = \begin{cases} 0 & \text{if } b = 0 \\ \omega_k^d(i) & \text{if } b = 1 \end{cases}$$

where $\omega_k^d(i)$ is the probability of exiting form state i within the k'th sub-HMM at level d. (Think of b as representing the signal from below.)

If each HMM at each level can be in one of K states, then there are K^d possible states at level d. The resulting state-transition diagram has a tree-like structure, as shown in Figure 7. Such a full model has $O(K^{D+1})$ parameters, where D is the depth of the hierarchy. For example, at level 3, we must define $T_k^3(i, j)$ for each sub-HMM, $k = 1, \ldots, 4$, i.e., 4 matrices of size 2×2; in general, at the bottom level, we have K^{D-1} matrices of size $K \times K$.

Since there might be too many parameters to specify in a large model, it is possible to remove some of the arcs from Q nodes higher up the hierarchy. The resulting state transition diagram then changes from tree-like to DAG-like. For example, Figure 8 shows the state transition diagram that would result if we removed the $Q^3 \rightarrow Q^1/F^1$ arcs: essentially we are saying that the transitions amongst states at level 1 are independent of the calling state at level 3. In other words, the topology of cells within a given corridor is the same on all floors. If in addition we remove the $Q^3 \rightarrow O$ arc, we are saying that the appearance of corridors is independent of floor. (Although this is a less plausible assumption in the navigation domain, in speech recognition it is common to assume the appearance of the subphones (level 1) is independent of the words (level 3) given the phones (level 2).)

Removing arcs from higher up Q nodes is a way to re-use sub-models in different contexts (share substructure). This reduces the amount of data required to learn the model. In general, we can make the conditional independencies "context specific" [BFGK96], so that we share sub-models in some higher level states, but not others. For example, floors 7 and 8 might be the same, but not floors 8 and 9. It is hard to represent such dynamic Bayesian multinets [Bil00] graphically, since we must include arcs if there is any dependence, even if only in rare contexts; however, the state transition diagram (which represents non-zero parameters) can make such complex patterns explicit.

6 Learning

6.1 Parameter learning

We can compute a local maximum likelihood parameter estimate using EM in the usual way: in the E step we compute the family marginals, $P(X_i, \text{Pa}(X_i) | O_{1:T})$, extract the expected sufficient statistics (ESS) from this, and sum the ESS up over sequences; in the M step, we maximise the parameters of each CPD independently, given its ESS. Below, we give the equations for the M step for the different node types, assuming a tabular (non-parametric) representation.

For notational simplicity, we make several assumptions: (1) that there is a single observation sequence, $O_{1:T}$; if there is more than one sequence, just sum over each sequence; (2) that $d > 1$; for $d = 1$, just eliminate the conditioning on $Q_t^{1:d-1} = k$; (3) that there are no inputs; if there are, just condition on them; (4) that the system is context-free, in the sense discussed in Section 4.3.2; if not, just condition on Q_{t-1}^{d-1}.

We can estimate the probability of a horizontal transition from state i to j at level d by counting the expected number of times we make such a transition within a segment, i.e., given that $F_{t-1}^d = 0$, indicating that i and j belong to the same sub-HMM:

$$\begin{aligned} \xi(t, d, i, k, j) &\stackrel{\text{def}}{=} P(Q_{t-1}^d = i, F_{t-1}^d = 0, Q_t^{1:d-1} = k, Q_t^d = j | O_{1:T}) \\ \hat{\tilde{A}}_k^d(i, j) &= \frac{\sum_{t=2}^T \xi(t, d, i, k, j)}{\sum_{j'} \sum_{t=2}^T \xi(t, d, i, k, j')} \end{aligned}$$

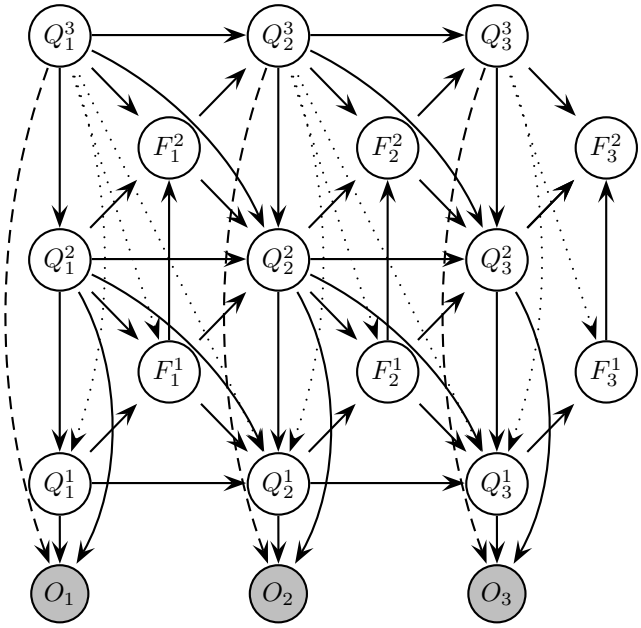

Figure 6: A 3-level HHMM represented as a DBN. If we omit the dotted arcs from Q^1, then we share the transition matrices for level 1 across level 3 models. If we omit the dashed arcs from Q^1, then we share the observation matrices for level 1 across level 3 models.

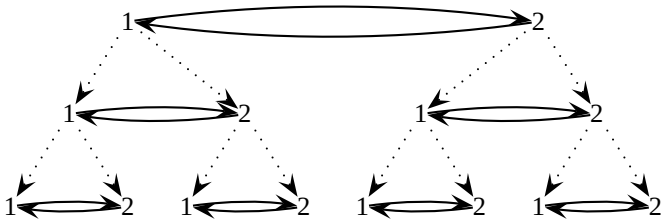

Figure 7: A full state-transition diagram for a 3-level HHMM, where each sub-HMM has 2 possible states. Dotted arcs represent vertical transitions, solid arcs represent horizontal transitions. Self transitions are not shown, to reduce clutter.

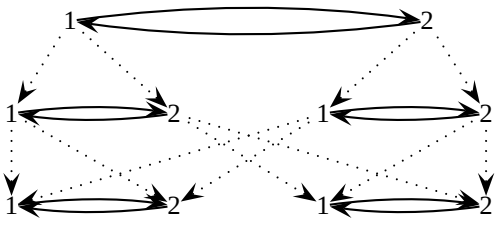

Figure 8: A state-transition diagram for a 3-level HHMM, where we share the bottom level HMMs. The vertical and horizontal transition probabilities at the bottom level depend only on the calling state at level 2, not at the top level.

We can estimate the probability of a vertical exit from state i at at level d by counting the expected number of times we are in state i and then the finish flag turns on:

$$\begin{aligned}\xi_e(t,d,k,i) &\stackrel{\text{def}}{=} P(Q_t^{1:d-1}=k, Q_t^d=i, F_t^d=1|O_{1:T}) \\ \hat{\tau}_k^d(i) &= \frac{\sum_{t=1}^T \xi_e(t,d,k,i)}{\sum_{i'}\sum_{t=1}^T \xi_e(t,d,k,i')}\end{aligned}$$

We can estimate the probability of a vertical entrance to state i at at level d by counting the expected number of times we are in state i, given that the finish flag was on:

$$\begin{aligned}\chi(t,d,k,i) &\stackrel{\text{def}}{=} P(F_{t-1}^d=1, Q_t^{1:d-1}=k, Q_t^d=i|O_{1:T}) \\ \chi(1,d,k,i) &\stackrel{\text{def}}{=} P(Q_1^{1:d-1}=k, Q_1^d=i|O_{1:T}) \\ \hat{\pi}_k^d(j) &= \frac{\sum_{t=1}^T \chi(t,d,k,i)}{\sum_{i'}\sum_{t=1}^T \chi(t,d,k,i')}\end{aligned}$$

If we model the observation density as a multinomial, which depends on the entire state vector, we can estimate it very easily as follows:

$$\begin{aligned}\gamma(t,k,o) &\stackrel{\text{def}}{=} P(Q_t^{1:D}=k, O_t=o|O_{1:T}) \\ \hat{B}(k,o) &= \frac{\sum_{t=1}^T \gamma(t,k,o)}{\sum_{o'}\sum_{t=1}^T \gamma(t,k,o')}\end{aligned}$$

A similar equation can be derived for Gaussian distributions.

Note that the F nodes represent the segmentation, and allow us to learn the parameters of each sub-model. If we flattened the DBN to an HMM, by combining the Q nodes and eliminating the F nodes, we would not be able to do learning.

If we know the segmentation, the distribution over the F nodes become delta-functions, and we can train each HMM independently. It is also possible that we know the segmentation at the top-level, but not the bottom-level; we simply include evidence on the appropriate set of F nodes. Similarly, if we know that each training case is a complete sequence (as opposed to a subsequence), we can add evidence that $F_T^d = 1$ for all d, to force all HMMs to finish at the last slice.

As mentioned above, many of these parameters are only defined for certain contexts (values of k). If the matrices are initialized to 0 for the invalid k's, they will remain 0 during the re-estimation process. Hence structural knowledge can be used to massively speed up parameter estimation.

6.2 Structure learning

By the structure of an HHMM, we mean the graph structure of the transition diagram (e.g., Figure 3), not the graph structure of the DBN, which is fixed i.e., we want to learn the number of HMMs, the number of states in each HMM, the allowable horizontal transitions within each HMM, and the allowable vertical transitions between levels.

The problem of determining the number of states and the set of non-zero horizontal transitions at any given level is equivalent to the problem of learning the structure of an HMM. A bottom-up approach to learning HMM structure is presented in [SO92]. They start with an HMM that has one state for every position in every sequence (hence it assumes the training set consists of a set of short strings). They then consider the benefit of merging all possible pairs of states (where the benefit is measured by the increase in the model posterior), and greedily perform the best merge. Clearly this algorithm cannot cannot be used to learn a model from a single long (unsegmented) sequence, nor can it be applied to sequences of continuous-valued data. A top-down, state-splitting approach is described in [FM00], where they use cross validation to determine if a local change to the model is beneficial.

A much more efficient way of learning HMM structure, which avoids discrete search, is to do MAP estimation of A_k with an entropic prior, followed by parameter pruning [Bra99]. The entropic prior has the form $P(\theta) \propto e^{-H(\theta)}$, and hence discourages parameters which are "uncertain" given the data; such parameters can be safely pruned from the model during EM. This approach assumes that we can afford to initialize the HMM with a fully connected topology and a large number of states, which might be problematic in the HHMM setting.

Now we turn to learning the vertical structure. An incoming vertical arc from state k to state i can be pruned if $\pi_k^d(i) \approx 0$. In principle, we could use an entropic prior to encourage sparse vertical connectivity as well. However, the real problem is detecting shared sub-structure, i.e., when we should do parameter tying. One idea would be to extend the model merging approach of [SO92], so that we merge HMMs instead of states. We need some kind of similarity metric on HMMs (e.g., [JR85]) to decide if we should merge them.

Structure learning in general HHMMs seems very hard. We are currently investigating hierarchical learning in a simpler class of models, based on the techniques of [dM96].

7 HPOMDPs in BNT

The example HPOMDP in Figure 3 has been implemented in BNT[3]: see the directory `BNT/examples/dynamic/HHMM/Map`. The file `mk-map-hhmm.m` makes the DBN; `sample-from-map` samples values of the nodes, using a controller than moves all the way to the right then all the way to the left; `learn-map` uses EM to learn the parameters from a single sequence of length 100, where the structure is hard-coded by initializing illegal transitions to 0, as in [The02, p.84]. Extensions are straightforward.

References

[BFGK96] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-Specific Independence in Bayesian Networks. In *UAI*, 1996.

[Bil00] J. Bilmes. Dynamic Bayesian multinets. In *UAI*, 2000.

[Bra99] M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11:1155–1182, 1999.

[dM96] C. de Marcken. *Unsupervised language acquisition*. PhD thesis, MIT AI lab, 1996.

[FM00] Dayne Freitag and Andrew McCallum. Information extraction with HMM structures learned by stochastic optimization. In *AAAI/IAAI*, pages 584–589, 2000.

[FST98] S. Fine, Y. Singer, and N. Tishby. The hierarchical Hidden Markov Model: Analysis and applications. *Machine Learning*, 32:41, 1998.

[JM00] D. Jurafsky and J. H. Martin. *Speech and language processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.

[JR85] B. H. Juang and L. R. Rabiner. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, 64(2):391–408, 1985.

[MP01] K. Murphy and M. Paskin. Linear time inference in hierarchical HMMs. In *NIPS*, 2001.

[Mur02] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, Dept. Computer Science, UC Berkeley, 2002.

[PT02] J. Pineau and S. Thrun. An integrated approach to hierarchy and abstraction for POMDPs. *J. of AI Research*, 2002. Submitted.

[SO92] A. Stolcke and S. M. Omohundro. Hidden Markov Model Induction by Bayesian Model Merging. In *NIPS-5*, 1992.

[The02] G. Theocharous. *Hierarchical Learning and Planning in Partially Observable Markov Decision Processes*. PhD thesis, Dept. Comp. Sci., Michigan State Univ., 2002.

[TRM01] G. Theocharous, K. Rohanimanesh, and S. Mahadevan. Learning Hierarchical Partially Observed Markov Decision Process Models for Robot Navigation. In *ICRA*, 2001.

[3] The Bayes Net Toolbox for Matlab is available at `www.cs.berkeley.edu/~murphyk/Bayes/bnt.html`.