

# On Efficient Viterbi Decoding for Hidden semi-Markov Models

Ritendra Datta  
Penn State University  
University Park, PA, USA  
datta@cse.psu.edu

Jianying Hu  
IBM T.J. Watson Research Center  
Yorktown Heights, NY, USA  
jyhu@us.ibm.com

Bonnie Ray  
IBM China Research Lab  
Beijing, China  
bonnier@cn.ibm.com

## Abstract

We present algorithms for improved Viterbi decoding for the case of hidden semi-Markov models. By carefully constructing directed acyclic graphs, we pose the decoding problem equivalently as that of finding the longest path in it between specific pairs of nodes. We consider fully connected models as well as restrictive topologies and state duration conditions, and show that performance improves by a significant factor in all cases. We present detailed algorithms as well as theoretical results related to their running time.

## 1. Introduction

Hidden Markov models [10], in their most common form (first-order Markov), have state durations that follow geometric distributions. This is oftentimes too restrictive to model certain real-world phenomena. Hidden semi-Markov models (HsMM) [4], also referred to as generalized HMMs, are one class of HMMs which allow state duration to be explicitly modeled by a chosen distribution, thus increasing their flexibility. They find application speech recognition [11], video event detection [5], and more recently, protein structure prediction [1] and business analytics [3].

Unlike in the case of HMMs, the standard Viterbi algorithm [10] does not guarantee optimal solution to sequence decoding or likelihood estimation because the usual assumptions do not hold. Not only is decoding crucial for inference, it is also a key step in Viterbi-based model training [7] (also called segmental K-means algorithm) which is a fast, approximate alternative to Baum-Welch training [10]. Therefore, a performance improvement in Viterbi decoding will lead to faster inference as well as training.

Given an HMM and an observation sequence, the standard Viterbi algorithm to find the corresponding state sequence is known to have a running time of

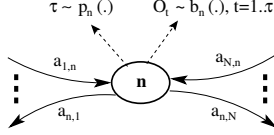
$O(NKT)$ , where  $N$  is the number of states,  $K$  is the average number of predecessors (in-degree) across states, and  $T$  is the sequence length. In the case of HsMM, the decoding time increases by a factor of  $D_{max}$  to  $O(NKTD_{max})$ , where  $D_{max}$  is the maximum allowed duration in any given state [9, 6, 8]. In the most general case, where there is no restriction on the state duration ( $D_{max} = T$ ), and all state transitions are possible ( $K \approx N$ ), the decoding time is  $O(N^2T^2)$ .

In this paper, we propose a new algorithm that improves the HsMM Viterbi decoding complexity. The algorithm, which essentially maps the decoding problem into finding the longest path in a directed acyclic graph (DAG) [2], also improves clarity in algorithm understanding and in the analysis of asymptotics. For the general case of HsMM topology, our decoding complexity is found to improve by a factor of  $\frac{NT}{N+T}$  and  $\frac{KD_{max}}{K+D_{max}}$  with and without restrictions respectively. For the specific case of HsMM topology where state skipping is disallowed (i.e.,  $K = 1$ ), improvement is by a factor of  $\frac{T^2}{T-N}$ . In the following sections, we describe the algorithms in detail, and analyze their running time.

## 2. Hidden Semi-Markov Model Basics

An HsMM has a finite set of states, say  $N$ . Transitions occur from state  $i$  to state  $j$  when transition probability  $a_{i,j}$  is positive. Due to its very nature,  $a_{i,i} = 0$  for all  $i$ . The HsMM topology is further characterized by  $K$ , the average number of predecessors to a state. If every state is reachable from every other,  $K = (N - 1)$ , and thus there are  $N(N - 1)$  possible transitions.

The HsMM generates observation sequences of length  $T$ ,  $\mathbf{O} = \{\mathbf{O}_1, \dots, \mathbf{O}_T\}$  following a Moore machine model, as shown in Fig. 1. The initial state of the model is entered according to the distribution  $\{\pi_1, \dots, \pi_N\}$ . Given that the system enters state  $n$ , state duration  $\tau \sim p_n(\cdot)$  is sampled, and a set of  $\tau$  observations  $\mathbf{O}_t \sim b_n(\cdot)$  are sampled independently. Here,  $d_n(\cdot)$  and  $b_n(\cdot)$  are dura-



**Figure 1. HsMM sequence generation by Moore machine model.**

tion and emission distributions respectively, for state  $n$ . A maximum value  $D_{max}$  for state duration  $\tau$  may be specified. The complete parameter set is denoted  $\lambda = (N, K, D_{max}, \{\pi_n\}, \{a_{i,j}\}, \{b_n\}, \{p_n\})$ . Given a model  $\lambda$  and an observation sequence  $\mathbf{O}$ , the decoding problem is to find the most likely underlying state sequence  $Q^* = \{q_1, \dots, q_T\}$ . The value  $\mathcal{L}^*$  of this maximum likelihood is also needed, especially for training:

$$Q^* = \arg \max_Q Pr(Q, \mathbf{O} | \lambda)$$

$$\mathcal{L}^* = \max_Q Pr(Q, \mathbf{O} | \lambda)$$

Let us denote by  $\delta_t(n)$  the likelihood of the most probable state sequence corresponding to the first  $t$  observations, that ends in state  $n$ . This is formally given as

$$\delta_t(n) = \max_{q_1, \dots, q_{t-1}} Pr(q_1, \dots, q_{t-1}, q_t = n, \mathbf{O}_1, \dots, \mathbf{O}_t | \lambda)$$

which can be computed efficiently using the recursion

$$\delta_t(n) = \max_{\tau_n} \left( \max_j \delta_{t-\tau_n}(j) a_{j,n} \right) p_n(\tau_n) \prod_{u=1}^t b_n(\mathbf{O}_u)$$

The maximum over  $\delta_T(n)$  yields  $\mathcal{L}^*$ , and by backtracking, we get  $Q^*$ . For each  $(t, n)$  pair, time taken to compute  $\delta_t(n)$  is  $O(KD_{max})$ . Computed over all  $t$  and  $n$  values, this standard decoding algorithm thus takes  $O(NTKD_{max})$  time, as reported in [9, 6], or  $O(N^2T^2)$  in the unconstrained case.

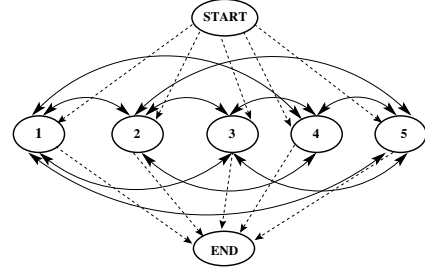
### 3. Improved Viterbi Decoding

To improve performance, we formulate equivalent problems of finding longest path in DAGs. We describe decoding algorithms, first for the most generic HsMM topology, and then for a more restrictive case.

#### 3.1. Fully Connected Topology

The fully connected  $N$ -state HsMM topology is shown in Fig. 2, where any start/end state is allowed and every state transition is possible, i.e.,  $K = N - 1$ . Furthermore, there is no restriction on state duration, i.e.,  $D_{max} = T$  when decoding a length  $T$  sequence.

Given the HsMM parameter set  $\lambda$  and an observation sequence  $\mathbf{O} = \{\mathbf{O}_1, \dots, \mathbf{O}_T\}$ , we construct a DAG as shown in Fig. 3. Specifically, for every  $(t, n)$  pair, where  $t = 1, \dots, T+1$  and  $n = 1, \dots, N$ , two types of nodes,  $U(t, n)$  (elliptical) and  $V(t, n)$  (rectangular), are introduced, along with special nodes START and END. Node types  $U$  and  $V$  are interpreted as follows:



**Figure 2. Example of an HsMM with fully connected topology ( $N = 5$  states).**

- $U(t, n)$ : The current state is  $n$  and a jump is only possible to node  $V(t', n)$ ,  $t' > t$ , which is equivalent to the HsMM generating  $(t' - t)$  observations in state  $n$ , before making another transition.
- $V(t, n)$ : The HsMM has completed generating observations and can now move to node  $U(t, n')$ , i.e., make a transition to HsMM state  $n'$ , where  $n' \neq n$ .

We denote by  $w(N_1 \rightarrow N_2)$  the weight on the directed edge from node  $N_1$  to  $N_2$ . Weights are assigned as follows, in logarithms. For  $n = 1 \dots N$ ,

$$w(\text{START} \rightarrow U(1, n)) = \log \pi_n.$$

For  $n = 1 \dots N$ ,  $t = 1 \dots T$ , and  $t' = t + 1 \dots T + 1$ ,

$$w(U(t, n) \rightarrow V(t', n)) = \log p_n(t' - t) + \sum_{x=t}^{t'-1} \log b_n(\mathbf{O}_x).$$

For  $n=1 \dots N$ ,  $t=2 \dots T$ , and  $n'=1 \dots N$ ,  $n' \neq n$ ,

$$w(V(t, n) \rightarrow U(t, n')) = \log a_{n,n'}.$$

Finally, for  $n = 1 \dots N$ ,

$$w(V(T+1, n) \rightarrow \text{END}) = 0. \quad (1)$$

Node pairs not covered above do not have edges between them, making the graph a DAG. Any path from START to END in this DAG corresponds to a valid state sequence for  $\mathbf{O}$ . The most likely state sequence  $Q^*$  corresponds to the *longest path* among them, and the likelihood  $\mathcal{L}^*$  corresponds to its length.

We find the longest path in this DAG, assign this length to  $\mathcal{L}^*$ , and backtrack on the longest path thus found so as to get the state sequence  $Q^*$ . For example, if the longest path found in such a graph for  $T = 5$  is

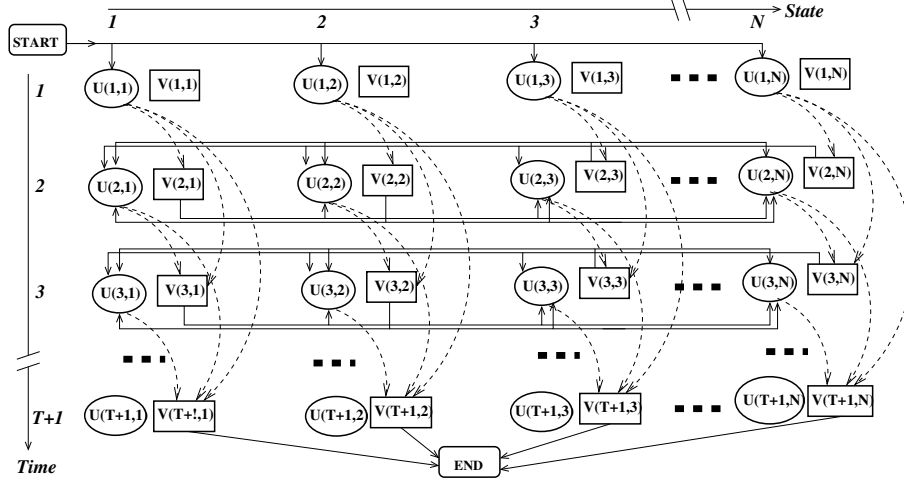
$$\text{START} \rightarrow U(1, 2) \rightarrow V(3, 2) \rightarrow U(3, 4) \rightarrow V(6, 4) \rightarrow \text{END}$$

then the optimal state sequence for  $\{\mathbf{O}_1, \dots, \mathbf{O}_5\}$  is  $Q^* = \{2, 2, 4, 4, 4\}$ . The path length is

$$\mathcal{L}^* = \log \left( \pi_2 p_2(2) \prod_{x=1}^2 b_2(\mathbf{O}_x) a_{2,4} p_4(3) \prod_{x=3}^5 b_4(\mathbf{O}_x) \right)$$

which can be easily seen to be the correct formulation for state likelihood  $Pr(Q^*, \mathbf{O} | \lambda)$ .

**3.1.1. Decoding Complexity.** The running time for our algorithm is dependent on (a) any pre-computation, (b) construction of the DAG, and (c) finding the longest



**Figure 3. Schematic representaiton of the DAG we construct for sequence decoding for fully connected HsMM. A path from START to END corresponds to a candidate decoding. Note that some nodes are drawn for illustrative purposes only.**

path in the DAG. For efficient construction of the DAG, we define a term  $\gamma_n(t) = \sum_{x=1}^t \log b_n(\mathbf{O}_x)$  which is the log probability of the observation sequence  $\{\mathbf{O}_1, \dots, \mathbf{O}_t\}$  in state  $n$ . This is computed recursively,

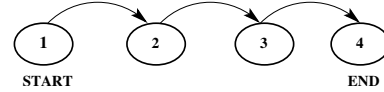
$$\gamma_n(t) = \gamma_n(t-1) + \log b_n(\mathbf{O}_t), \quad (2)$$

$n = 1 \dots N, t = 1 \dots T$ , where initial condition is  $\gamma_n(0) = 0$ . This pre-computation step takes  $O(NT)$ . For the graph to be constructed, there are  $NT$  nodes each of type U and V. Originating from each U node are upto  $T$  edges, and from each V node upto  $N$  edges. The START and END nodes have  $N$  edges associated with them. Therefore, the total number of edges is  $\approx NT(N+T) + 2N$ . The DAG construction step essentially involves assignment of edge weights, which can all be computed in constant time  $O(1)$  except for  $w(U(t,n) \rightarrow V(t',n))$ . A naïve approach would take  $O(T)$  to compute  $\sum_{x=t}^{t'-1} \log b_n(\mathbf{O}_x)$ , but pre-computed  $\gamma_n(t)$  values allow for  $O(1)$  computation by using

$$\begin{aligned} \sum_{x=t}^{t'-1} \log b_n(\mathbf{O}_x) &= \sum_{x=1}^{t'-1} \log b_n(\mathbf{O}_x) - \sum_{x=1}^{t-1} \log b_n(\mathbf{O}_x) \\ &= \gamma_n(t'-1) - \gamma_n(t-1) \end{aligned} \quad (3)$$

Thus, DAG construction complexity is  $O(NT(N+T))$ . The final step is to find the longest path in this DAG. We know [2] that if the topological ordering of nodes is known, then finding the DAG shortest path takes  $O(|edges| + |nodes|)$ . Moreover, we can find the longest path using this algorithm by negating the edge weights first. We do not need to topological sort the nodes because we know it by construction, as follows:

$$\begin{aligned} &\text{START} \rightarrow V(1,1), \dots, V(1,N) \rightarrow U(1,1), \dots, U(1,N) \\ &\rightarrow V(2,1), \dots, V(2,N) \rightarrow U(2,1), \dots, U(2,N), \dots, \text{END} \end{aligned}$$



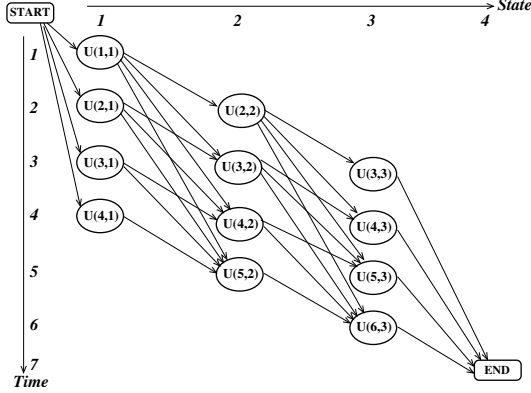
**Figure 4. Example of a HsMM structure with state skipping disallowed ( $N = 4$ ).**

Since  $|nodes| \approx 2NT$  and  $|edges| \approx NT(N+T) + 2N$ , the longest path time complexity is  $O(NT(N+T))$ . Therefore, the overall computation time is also  $O(NT(N+T))$ . If the maximum state duration  $D_{max}$  is specified, and the HsMM topology is less than fully connected (specified by  $K$ ), it is easy to show that decoding complexity then is  $O(NT(K + D_{max}))$ .

### 3.2. No State Skipping Topology

For certain applications of HsMM training and sequence decoding, a simpler topology may suffice in modeling an underlying process. For example, business project resource allocations over time may be conceived as being generated by an HsMM with states signifying project phases, where no phase can be skipped [3]. In such cases, a strict start/end topology with state skipping disallowed, shown in Fig. 4 may be appropriate.

For this restricted topology, decoding run-time can be improved further. The model parameters here are  $\lambda = (N, D_{max}, \{b_n\}, \{p_n\})$ . Note that by default,  $K = 1, \pi_1 = 1, p_{ij} = 0$  for all  $j > 1$ , and  $a_{i,j} = 1$  only for  $j = i + 1$ , zero otherwise. Given an observation sequence  $\mathbf{O}$  of length  $T \geq N$ , we again construct a DAG as shown in the example in Fig. 5. Specifically, only U nodes are needed here because state skipping is disallowed. The node set consists of  $\{U(t,n)\}, n = 1, \dots, N-1, t = n, \dots, n + (T-N)$ , and special nodes



**Figure 5. Schematic representation of a DAG we construct for sample decoding problem ( $N = 4$ ,  $T = 7$ ) for a strict HsMM topology with state skipping disallowed.**

START and END. Edge weights, for those edges that exist, are defined as follows. For  $t = 1 \dots (T - N) + 1$ ,

$$w(\text{START} \rightarrow U(t, 1)) = \log p_1(t) + \sum_{x=1}^t \log b_1(\mathbf{O}_x).$$

For  $n' = 1 \dots N - 1$ ,  $n = n' + 1$ ,  $t = n' \dots n + (T - N)$ , and  $t' = t + 1 \dots n + (T - N)$ ,

$$w(U(t, n') \rightarrow U(t', n)) = \log p_n(t' - t) + \sum_{x=t+1}^{t'} \log b_n(\mathbf{O}_x).$$

Finally, for  $t = N - 1, \dots, T - 1$ ,

$$w(U(t, N - 1) \rightarrow \text{END}) = \log p_n(T - t) + \sum_{x=t+1}^T \log b_n(\mathbf{O}_x).$$

Again, the optimal state sequence  $Q^*$  corresponds to the longest path in this DAG, and  $\mathcal{L}^*$  is its length. For example, if the longest path in the DAG in Fig. 5 is

$$\text{START} \rightarrow U(2, 1) \rightarrow U(3, 2) \rightarrow U(6, 3) \rightarrow \text{END}$$

then the optimal state sequence corresponding to  $\{\mathbf{O}_1, \dots, \mathbf{O}_7\}$  is  $Q^* = \{1, 1, 2, 3, 3, 3, 4\}$ .

**3.2.1. Decoding Complexity.** The running time for this algorithm depends on the same three steps as before. We again pre-compute  $\gamma_n(t)$  values using the recursion in Eq. 2, which takes  $O(NT)$  time. For the construction of the DAG, we note that  $|nodes| \approx N(T - N)$ , and from each node, upto  $T - N$  edges originate, so  $|edges| \approx N(T - N)^2$ . The edge weight calculations are all done in  $O(1)$  time using Eq. 3, which makes the DAG construction take  $O(N(T - N)^2)$ . Finally, we can again use the DAG shortest path algorithm [2] which takes  $O(|nodes| + |edges|)$  time. To get longest path, we negate the edge weights, and then provide a topological ordering of the nodes, known by construction:

$$\text{START} \rightarrow U(1, 1), \dots, U(1, 1 + (T - N)) \rightarrow$$

$$\rightarrow U(N - 1, N - 1), \dots, U(N - 1, T - 1) \rightarrow \text{END}$$

Since  $|nodes| \approx N(T - N)$  and  $|edges| \approx N(T - N)^2$ , the decoding time for this topology is  $O(N(T - N)^2)$ .

## 4. Conclusions

We have proposed algorithms that improve Viterbi decoding of sequences for HsMM, over the previously known running time of  $O(N^2T^2)$ , by formulating equivalent DAG longest path problems. For the fully connected topology, our algorithm achieves a performance of  $O(NT(N + T))$ . For more restricted topologies with  $K$  predecessors per state on average and maximum state specified by  $D_{max}$ , our running time is  $O(NT(K + D_{max}))$ . For a very simple topology that allows no state-skipping, our algorithm brings down the decoding complexity to  $O(N(T - N)^2)$ . Improvements to the forward-backward algorithm, used for Baum-Welch training, have been shown before [12], with running time of  $O(NT(K + D_{max}))$ , but we are unaware of explicit attempts at improving HsMM Viterbi decoding.

## References

- [1] Z. Aydin, Y. Altunbasak, and M. Borodovsky. Protein secondary structure prediction for a single-sequence using hidden semi-markov models. *BMC Bioinformatics*, 7:178–178, 2007.
- [2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [3] R. Datta, J. Hu, and B. Ray. Building project taxonomies for resource demand forecasting. In *Proc. Workshop on Data Mining for Business, PAKDD*, 2007.
- [4] D. Ferguson. Variable duration models for speech. In *Proc. Symposium on the Application of HMMs to Text and Speech*, 1980.
- [5] S. Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. In *Proc. ICCV*, 2003.
- [6] N. Hughes, L. Tarassenko, and S. Roberts. Markov models for automated ecg interval analysis. In *Proc. NIPS*, 2003.
- [7] B.-H. Juang and L. Rabiner. The segmental k-means algorithm for estimating parameters of hidden markov models. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 38(9):1639–1641, 1990.
- [8] C. Mitchell, M. Harper, and L. Jamieson. On the complexity of explicit duration hmm's. *IEEE Trans. Speech and Audio Processing*, 3(3):213–217, 1995.
- [9] K. Murphy. *Hidden semi-Markov models*. Technical Report, MIT AI Lab, 2002.
- [10] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.
- [11] M. Russell and R. Moore. Explicit modelling of state occupancy in hidden markov models for automatic speech recognition. In *Proc. ICASSP*, 1985.
- [12] S. Yu and H. Kobayashi. An efficient forward-backward algorithm for an explicit-duration hidden markov model. *IEEE Signal Processing Letters*, 10(1):11–14, 2003.