State Sequence Analysis in Hidden Markov Models

Yuri Grinberg
Ottawa Hospital Research Institute
Ottawa, Ontario, Canada
ygrinberg@ohri.ca
www.perkinslab.ca

Theodore J. Perkins
Ottawa Hospital Research Institute
Ottawa, Ontario, Canada
tperkins@ohri.ca
www.perkinslab.ca

Abstract

Given a discrete time finite state hidden Markov model (HMM) and a sequence of observations, there are different ways to estimate the hidden behavior of the system. In this paper, the problem of finding the most probable state sequence is considered. The state sequence, as opposed to the state trajectory, specifies the sequence of states that the HMM visits but does not specify the dwelling times in these states. This inference problem is relevant in a variety of domains, like text analysis, speech recognition, or behavior recognition, where the exact timing of hidden state transitions is not nearly as important as the sequence of states visited. No existing algorithm addresses this inference question adequately. Leveraging previous work on continuous time Markov chains, we develop a provably correct algorithm, called *state sequence analysis*, that addresses this inference question in HMMs. We discuss and illustrate empirically the differences between finding the most probable state sequence directly and doing so through running the Viterbi algorithm and collapsing repetitive state visitations. Experimental results in two synthetic domains demonstrate that the Viterbi-based approach can be significantly suboptimal compared to state sequence analysis. Further, we demonstrate the benefits of the proposed approach on a real activity recognition problem.

1 INTRODUCTION

Hidden Markov models are a powerful and widely-used formalism for analyzing sequential information in a variety of domains, including speech recognition [Bahl et al., 1986, Rabiner, 1989], text analysis [Blei and Moreno, 2001], behavior recognition [Yamato et al., 1992, Nguyen et al., 2005], protein structure prediction [Sonnhammer et al., 1998], genomics [Haussler and Eeckman, 1996, Wang et al., 2007], and so on. As with graphical models generally, much of the utility of HMMs derives from our ability to use them to make estimates about hidden/unobserved variables based on observable variables. HMMs are often used as models of dynamical systems. In this context, the unobserved variables would be the true (underlying) state of the dynamical system. The observed variables would be something that we "see" when we observe or measure the system. The observations typically have some relationship to the underlying state, but the relationship may be imperfect or noisy. A typical task would then involve receiving some kind of observations of the system, and estimating the underlying states that generated those observations. Of course, an HMM can model sequential information that is not dynamical in nature, as seen for instance in applications in text analysis, genome annotation, etc. Regardless, the most common task is to estimate underlying states based on observations.

There are several different types of hidden state estimation problems for HMMs. Although we define HMMs formally in the next section, to distinguish different types of inference problems it is useful to introduce a small amount of notation here. We let X_t be a random variable denoting the underlying state at time t, and x_t a realization of that variable. Similarly, we let o_t be the observation at time t. Further, we let $X_{1:t} = (X_1, X_2, \ldots, X_t)$ be a random variable describing possible system *trajectories*, a realization of which is denoted $\mathbf{x}_{1:t} = (x_1, x_2, \ldots, x_t)$. Similarly, a series of observations is denoted $\mathbf{o}_{1:t} = (o_1, o_2, \ldots, o_t)$.

One of the fundamental HMM inference problems is to compute the probabilities of different underlying system states based on observations. More formally, if we receive a stream of observations, $o_1, o_2, o_3, \ldots$ then at each time $t = 1, 2, 3, \ldots$ we may want to compute the probabilities of different states x conditional on the observations: $\mathrm{P}(X_t = x|\mathbf{o}_{1:t})$. This can be done by the forward algorithm [Rabiner, 1989], which is very efficient, and indeed allows us to readily compute $\mathrm{P}(X_t = x|\mathbf{o}_{1:t})$ incrementally based on $\mathrm{P}(X_{t-1} = x'|\mathbf{o}_{1:t-1})$. If we are given an entire sequence of observations, $\mathbf{o}_{1:t}$, and we want to estimate the underlying state probabilities at all times, namely $\mathrm{P}(X_{t'} = x|\mathbf{o}_{1:t})$ for all x and $1 \le t' \le t$, then the forward-backward algorithm [Rabiner, 1989] gives us an efficient solution. From those probabilities, one can easily compute the most probable state at each time: $x_{t'}^{\max} = \arg\max_x \mathrm{P}(X_t' = x|\mathbf{o}_{1:t})$.

Each of these problems give us some information about underlying states based on observations, but they do not explicitly give us any pathway information. The forward and forward-backward algorithms, for example, give us only state probabilities. The sequence of most probable states, $x_1^{\max}, x_2^{\max}, x_3^{\max}, \ldots$, may or may not comprise a valid system path. That is, some transition $x_{t'}^{\max} \to x_{t'+1}^{\max}$ may not even be allowed under the dynamics. Thus, while these approaches are useful for estimating underlying states or "classifying" time points into different states, they are not directly useful for reconstructing underlying system paths.

For path reconstruction, by far the most common approach is the Viterbi algorithm [Rabiner, 1989], which is an efficient means for computing the maximum probability trajectory underlying a given sequence of observations: $\arg\max_{\mathbf{x}_{1:t}} \mathrm{P}(\mathbf{x}_{1:t}|\mathbf{o}_{1:t})$. Despite the many successes of the Viterbi approach (e.g., in speech recognition, activity recognition and bioinformatics, as described above), it has has been critiqued on a few different grounds. For one, like any *maximum a posteriori* (MAP) estimator, there is a question of how "representative" the maximum is. Intuitively, if the bulk of the posterior distribution contains trajectories that are "unlike" the MAP trajectory in some way, then the MAP trajectory can be misleading in our attempt to interpret observational data (see, e.g., [Lember and Koloydenko, 2014] and references therein). As a simple example, imagine the underlying state system is a series of independent flips of a biased coin that comes up heads with probability $p > 0.5$, and suppose we receive totally non-informative observations. Then the MAP trajectory is a series of all heads. But this path is "atypical" in a number of senses. For instance, we do expect some tails—in particular, about $n(1-p)$ of them where n is the number of flips. This and a great many other statistics about the series of coin flips are not represented in the MAP path. This is not a criticism of Viterbi per se, but merely of the practice of thinking or hoping that the MAP path is somehow representative of other probable paths as well.

In some other work, the authors have pointed out problems with MAP paths for stochastic continuous-time discrete-state systems [Perkins, 2009, Levin et al., 2012]. Such systems dwell in a state for random period of time, before moving on to a randomly-chosen next state. Although MAP paths can be efficiently computed [Perkins, 2009], solutions are non-typical in that they involve "instant" transitions through low latency states and dwells in high latency states. Similar issues can arise in discrete time HMMs, particularly, but not exclusively, when analyzing an HMM that arises from discretizing a continuous-time process.

A second problem with MAP paths is that they constitute an overly-specific inference. For example, in a simple speech recognition scenario, a MAP path might assign a word or phoneme to every timestep underlying the speech signal. However, transitions between words are not truly so crisp. Moreover, there is usually no point in assigning precise start and end times to each word or phoneme. If we hear, "The quick brown fox $\ldots$", what is the value of estimating that the word "The" ends and the word "quick" begins precisely 0.823 seconds into a spoken signal? It is the sequence of words spoken, and not their exact timing, that is important. Similarly for gesture recognition, activity recognition, etc.

Motivated by these criticisms of MAP trajectories, an alternative inference method, called *state sequence analysis*, was proposed earlier [Levin et al., 2012]. In this inference problem, the objective is to find the maximum a posteriori *sequence* of states, but averaging away (i.e. marginalizing over) the transition times between those states—essentially treating them as nuisance variables. In that work, however, the inference problem is solved for systems modeled as continuous-time Markov chains with initial and/or terminal probabilities; no observations during the time period of interest are allowed. In the present work, we study state sequence analysis for HMMs: the focus is on discrete time setting, where a series of noisy observations is allowed. We describe a provably-correct algorithm that finds the most probable state sequence given a trajectory of observations. Then, we demonstrate how state sequence analysis differs from Viterbi path inference on synthetic and real examples, and in particular (as it is designed to do) how state sequence analysis provides more accurate estimates of the sequence of states underlying a noisy series of observations.

2 BACKGROUND

Let $\mathcal{X}$ be a discrete finite state space and $\mathcal{O}$ the observation space of a hidden Markov model [Rabiner, 1989]. Let $\mathbf{T}$ be the transition matrix of this HMM, with $\mathbf{T}_{x,y}$ representing the probability of transitioning from state $x \in \mathcal{X}$ to state $y \in \mathcal{X}$, and $p_x(o)$ be the emission probability of observation $o \in \mathcal{O}$ in state $x \in \mathcal{X}$.

For a possible trajectory of states visited by a discrete time HMM we are interested in identifying what is the corresponding duration–free sequence of states, i.e. sequence of states with self-loops removed. For example, given the trajectory of states $\langle x, x, x, y, y, y, x \rangle$, the corresponding *state sequence* will be $\langle x, y, x \rangle$. We denote the probability that HMM trajectory follows the state sequence $\mathbf{s}$ given the sequence of n observations, as

$$\mathrm{P}(X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}) | \mathbf{o}_{1:n}),$$

where $\mathfrak{seq}_n(\mathbf{s})$ is a set of all length n trajectories whose duration–free sequence equals to $\mathbf{s}$. Where possible, we will use $\mathrm{P}(\mathbf{s}|\mathbf{o}_{1:n})$ as a shortcut to the above notation.

3 THE STATE SEQUENCE INFERENCE PROBLEM

Finding the most probable state sequence, which we call *state sequence analysis* (SSA), can be seen as a search problem that requires evaluation of probabilities of state sequences. Recall that, to find the most probable trajectory of states, a naive exhaustive enumeration quickly becomes infeasible because the number of possible trajectories grows exponentially with the length of observation sequence. Similarly, a naive implementation of the search for the most probable state sequence does not scale well. Even a single evaluation of a probability of a state sequence involves the summation over possibly large number of terms. Specifically, the number of terms in the set $\mathfrak{seq}_n(\mathbf{s})$ is equal to $\binom{n-1}{|\mathbf{s}|-1}$ for the state sequence $\mathbf{s}$.

Nevertheless, inferring the most probable trajectory can be done efficiently by a well–known Viterbi algorithm that uses dynamic programming to do the search and evaluation simultaneously. Although a Viterbi-like approach does not appear to be possible to address the question we pose, in what follows we develop search and evaluation algorithms that make the problem tractable. First, we identify a particular structure within the search space that allows us to prune large parts of the space as the search progresses. Second, we provide a recursive relation that is used to efficiently evaluate probabilities of new state sequences using dynamic programming. However, prior to delving into the algorithmic details, we begin with a little discussion about the similarities and differences between most probable state sequence and most probable state trajectory.

3.1 THE MOST PROBABLE STATE SEQUENCE AND TRAJECTORY

The definition of a state sequence presented in an earlier section leaves little doubt that the most probable state trajectory is not necessarily the same as the most probable state sequence, given a sequence of observations. Yet, to better understand the nature of their differences it might be helpful to pinpoint the cases where those two will, in fact, be equal. In the first case, consider a HMM in which probabilities of staying in any state are zero. Any state trajectory generated by this HMM will have no repetitive states, and therefore the only state sequences that have non zero probability of happening must be of the same length as the observation sequence. Hence, the most probable state trajectory and the most probable state sequence will be the same.

In the second case, suppose that the observations identify the underlying hidden states exactly (observation sequence is also Markov). It implies that there is only one state trajectory that could generate a given sequence of observations. Hence, there is only one state sequence explaining a given observation sequence; that state sequence can be computed by collapsing repetitive states appearing in the only possible state trajectory.

Of course, both of these cases are rather extreme, and in their exact form can rarely be found in practice. However, one can expect that the "closer" the given HMM is to one of those extremes, the less evident will the difference be between the most probable state sequence and state trajectory (in its collapsed form). This intuition is also backed up, to some extent, by the experimental results presented in later sections.

3.2 DOMINATION OF SEQUENCES

In [Levin et al., 2012], the authors develop a structural relation between state sequences of a continuous time Markov chain that allows them to avoid searching all of the space of sequences. In this section, we develop a similar relation between sequences in the setting of discrete time HMMs. The following derivation, which serves as a basis for this development, enables us to express the probability of longer state sequences in terms of shorter state sequences. Let $\mathbf{s}_x$ be a state sequence that ends with state x , and let $\mathbf{u}_y$ be a one step shorter subsequence of $\mathbf{s}_x$ such that $\mathbf{s}_x = \langle \mathbf{u}_y, x \rangle$. Then,

$$\begin{aligned}
\mathrm{P}(\mathbf{s}_x|\mathbf{o}_{1:n}) &= \textstyle\sum_{i=1}^{n-1} \mathrm{P}[X_{1:i} \in \mathfrak{seq}_i(\mathbf{u}_y); X_{i+1:n} = x|\mathbf{o}_{1:n}] \\
&= \textstyle\sum_{i=1}^{n-1} \mathrm{P}[X_{1:i} \in \mathfrak{seq}_i(\mathbf{u}_y) \mid \mathbf{o}_{1:n}] \\
&\qquad \cdot \mathrm{P}[X_{i+1:n} = x \mid \mathbf{o}_{1:n}; X_i = y] \\
&= \textstyle\sum_{i=1}^{n-1} \frac{\mathrm{P}[\mathbf{o}_{1:n} \mid X_{1:i} \in \mathfrak{seq}_i(\mathbf{u}_y)] \cdot \mathrm{P}[X_{1:i} \in \mathfrak{seq}_i(\mathbf{u}_y)]}{\mathrm{P}(\mathbf{o}_{1:n})} \\
&\qquad \cdot \mathrm{P}[X_{i+1:n} = x \mid \mathbf{o}_{i+1:n}; X_i = y] \\
&= \textstyle\sum_{i=1}^{n-1} \frac{\mathrm{P}[\mathbf{o}_{1:i} \mid X_{1:i} \in \mathfrak{seq}_i(\mathbf{u}_y)] \cdot \mathrm{P}[X_{1:i} \in \mathfrak{seq}_i(\mathbf{u}_y)]}{\mathrm{P}(\mathbf{o}_{1:i})} \\
&\qquad \cdot \frac{\mathrm{P}[\mathbf{o}_{i+1:n} \mid X_i = y]}{\mathrm{P}[\mathbf{o}_{i+1:n} \mid \mathbf{o}_{1:i}]} \\
&\qquad \cdot \mathrm{P}[X_{i+1:n} = x \mid \mathbf{o}_{i+1:n}; X_i = y] \\
&= \textstyle\sum_{i=1}^{n-1} \mathrm{P}[X_{1:i} \in \mathfrak{seq}_i(\mathbf{u}_y) \mid \mathbf{o}_{1:i}] \\
&\qquad \cdot \frac{\mathrm{P}[X_{i+1:n} = x; \mathbf{o}_{i+1:n} \mid X_i = y]}{\mathrm{P}[\mathbf{o}_{i+1:n} \mid \mathbf{o}_{1:i}]} \\
&= \textstyle\sum_{i=1}^{n-1} \mathrm{P}(\mathbf{u}_y \mid \mathbf{o}_{1:i}) \cdot \frac{\mathbf{T}_{y,x} \mathbf{T}_{x,x}^{n-i-1} \prod_{j=i+1}^{n} p_x(o_j)}{\mathrm{P}[\mathbf{o}_{i+1:n} \mid \mathbf{o}_{1:i}]}
\end{aligned} \tag{1}$$

Eq. (1) makes the separation between the parameters of the problem (HMM, observation sequence) and the shorter state sequence explicit. Specifically, the probability of longer state sequence equals to convolution of probabilities of shorter state sequence and the probability of staying in new state, normalized appropriately by the emission probabilities. As a result, the following definition of dominance between sequences turns out to be useful in speeding up the search.

Definition 1. *Given a sequence of observations $\mathbf{o}_{1:n}$, let $\mathbf{s}$ and $\mathbf{v}$ be sequences that start from the same state and end with the same state. Then we say that $\mathbf{s}$ dominates $\mathbf{v}$, and denote it by $\mathbf{s} \gg_n \mathbf{v}$, if*

$$\forall i \in \{1, ..., n\} : \mathrm{P}(\mathbf{s}|\mathbf{o}_{1:i}) \geq \mathrm{P}(\mathbf{v}|\mathbf{o}_{1:i}).$$

The dominance relation assures us that extensions of dominated sequences should never be explored within the search space, as the following lemma suggests.

Lemma 1. *Let $\mathbf{v}_y$ and $\mathbf{s}_y$ be two state sequences that start with the same state and end with state y. If $\mathbf{v}_y$ is dominated by $\mathbf{s}_y$ given an observation sequence $\mathbf{o}_{1:n-1}$ for some $n > 1$, then a one step extension of state sequence $\mathbf{v}_y$ will be dominated by some other state sequence, given an observation sequence $\mathbf{o}_{1:n}$.*

Proof. Observe that if $\mathbf{v}_y$ is dominated by $\mathbf{s}_y$ for an observation sequence $\mathbf{o}_{1:n-1}$, then the same holds for shorter observation sequences, e.g. $\forall i \in \{1, ..., n-1\} : \mathbf{o}_{1:i}$, simply following the definition of the dominance. Now, let $\langle \mathbf{v}_y, x\rangle$ be a one step extension of a state sequence $\mathbf{v}_y$. Since $\mathbf{v}_y$ is dominated by $\mathbf{s}_y$ for observation sequences $\mathbf{o}_{1:i}$ ($\forall i < n$), following equation (1) we get that

$$\forall i \leq n : \mathrm{P}(\langle \mathbf{v}_y, x\rangle|\mathbf{o}_{1:n}) \leq \mathrm{P}(\langle \mathbf{s}_y, x\rangle|\mathbf{o}_{1:n}).$$

□

Based on the above result, similarly to [Levin et al., 2012], we devise the algorithm that performs the search within the space of sequences by maintaining sets of non–dominated sequences only. The pseudo-code of the algorithm is provided in Algorithm 1 box. Each new sequence is checked against existing non–dominated sequences (line 6). If it appears to be non–dominated by any of those sequences, it is added to the set on non–dominated sequences (line 7). All sequences that are dominated by the new sequence in this set are removed (line 8), and all one step extensions of the new sequence will be checked by the algorithm later on (line 9).

3.3 COMPUTING THE PROBABILITY OF A STATE SEQUENCE

As mentioned above, performing the search within the space of state sequences will rely on evaluation of probabilities of those sequences. One approach to do so is based on implementing the recursive relation between sequences given in Eq. (1) using dynamic programming. Following this route it requires $O(n^2k)$ operations to evaluate the probability of a state sequence of length k given n observations. On the other hand, the following recursive relation can be used to evaluate probabilities in just $O(nk)$ operations. As previously, let $\mathbf{s}_x = \langle \mathbf{u}_y, x\rangle$ be a state sequence that consists of a shorter state sequence $\mathbf{u}_y$ and a last state x.

$$\begin{aligned}\mathrm{P}(\mathbf{s}_x|\mathbf{o}_{1:n}) &= \tfrac{\mathrm{P}[\mathbf{o}_{1:n-1}, o_n \mid X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x)] \cdot P[X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x)]}{\mathrm{P}(o_n \mid \mathbf{o}_{1:n-1}) \cdot \mathrm{P}(\mathbf{o}_{1:n-1})} \\ &= \tfrac{\mathrm{P}[\mathbf{o}_{1:n-1} \mid X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x)] \cdot \mathrm{P}[X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x)]}{\mathrm{P}(\mathbf{o}_{1:n-1})} \\ &\quad \cdot \frac{\mathrm{P}[o_n \mid X_n = x]}{\mathrm{P}(o_n \mid \mathbf{o}_{1:n-1})} \\ &= \mathrm{P}[X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x)|\mathbf{o}_{1:n-1}] \\ &\quad \cdot \frac{p_x(o_n)}{\mathrm{P}(o_n|\mathbf{o}_{1:n-1})}. \end{aligned} \tag{2}$$

Now, the first term in the product of Eq. (2) can be expanded as follows:

$$\begin{aligned}&\mathrm{P}[X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x) \mid \mathbf{o}_{1:n-1}] \\ &\quad = \mathrm{P}[X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x); X_{n-1} = y \mid \mathbf{o}_{1:n-1}] \\ &\qquad + \mathrm{P}[X_{1:n} \in \mathfrak{seq}_n(\mathbf{s}_x); X_{n-1} = x \mid \mathbf{o}_{1:n-1}] \\ &\quad = \mathrm{P}[X_{1:n-1} \in \mathfrak{seq}_{n-1}(\mathbf{u}_y); X_n = x \mid \mathbf{o}_{1:n-1}] \\ &\qquad + \mathrm{P}[X_{1:n-1} \in \mathfrak{seq}_n(\mathbf{s}_x); X_n = x \mid \mathbf{o}_{1:n-1}] \\ &\quad = \mathrm{P}(\mathbf{u}_y|\mathbf{o}_{1:n-1})\mathbf{T}_{y,x} + \mathrm{P}(\mathbf{s}_x|\mathbf{o}_{1:n-1})\mathbf{T}_{x,x}. \end{aligned} \tag{3}$$

Combining Eq. (2) and Eq. (3) we get,

$$\begin{aligned}\mathrm{P}(\mathbf{s}_x|\mathbf{o}_{1:n}) &= \frac{p_x(o_n)}{\mathrm{P}(o_n|\mathbf{o}_{1:n-1})} \\ &\quad \cdot \left[\mathrm{P}(\mathbf{u}_y|\mathbf{o}_{1:n-1})\mathbf{T}_{y,x} + \mathrm{P}(\mathbf{s}_x|\mathbf{o}_{1:n-1})\mathbf{T}_{x,x}\right]. \end{aligned} \tag{4}$$

Algorithm 1 State Sequence Analysis for HMMs

```
 1: Initialize $\forall x, y : ND_{x,y} = \emptyset$ - sets of non–dominated
    sequences for each pair of start and end states $x, y$
 2: Initialize Queue $CheckQ = \{\forall x : \langle x\rangle, \forall x, y : \langle x, y\rangle\}$
 3: while $CheckQ$ is not empty do
 4:     Fetch $\mathbf{s}_{x,y} \in CheckQ$
 5:     Compute $\forall i : \mathrm{P}(\mathbf{s}_{x,y}|\mathbf{o}_{1:i})$
 6:     if $\mathbf{s}_{x,y}$ is not dominated by $ND_{x,y}$ then
 7:         Add $\mathbf{s}_{x,y}$ to $ND_{x,y}$
 8:         Remove all $\mathbf{u}_{x,y} \in ND_{x,y}$ s.t. $\mathbf{s}_{x,y} \gg_n \mathbf{u}_{x,y}$
 9:         $\forall z \neq y$ : Add $\langle \mathbf{s}_{x,y}, z\rangle$ to $CheckQ$
10:     end if
11: end while
12: return $\arg\max_{\forall x,y:\mathbf{s} \in ND_{x,y}} \mathrm{P}(\mathbf{s}|\mathbf{o}_{1:n})$
```

4 EMPIRICAL EVALUATION

4.1 SYNTHETIC EVENT-DETECTION EXAMPLES

To demonstrate state sequence analysis, and its difference compared to the Viterbi algorithm, consider the HMM in Figure 1A. Trajectories begin in state S and can remain there through self-looping or move on either to state B or state E. State E is absorbing. When the system is in state B it can remain there through self-looping or proceed to state E. All states emit normal-distributed observations with standard deviation 1. However, the mean observation is zero in states S and E, and μ_B in state B. Thus, states S and E appear the same, whereas state B may appear different if $\mu_B \neq 0$.

At a high level, this example models event detection problems—for instance, detecting a security intrusion [Qiao et al., 2002], detecting specific gestures [Dardas and Georganas, 2011], detecting molecular events [Schreiber and Karplus, 2015], etc. Essentially, there is a series of noisy but harmless or uninteresting events, punctuated, rarely and for a short time, with relevant activity. But that relevant activity may still be subtle to detect, depending on how different it is from the background.

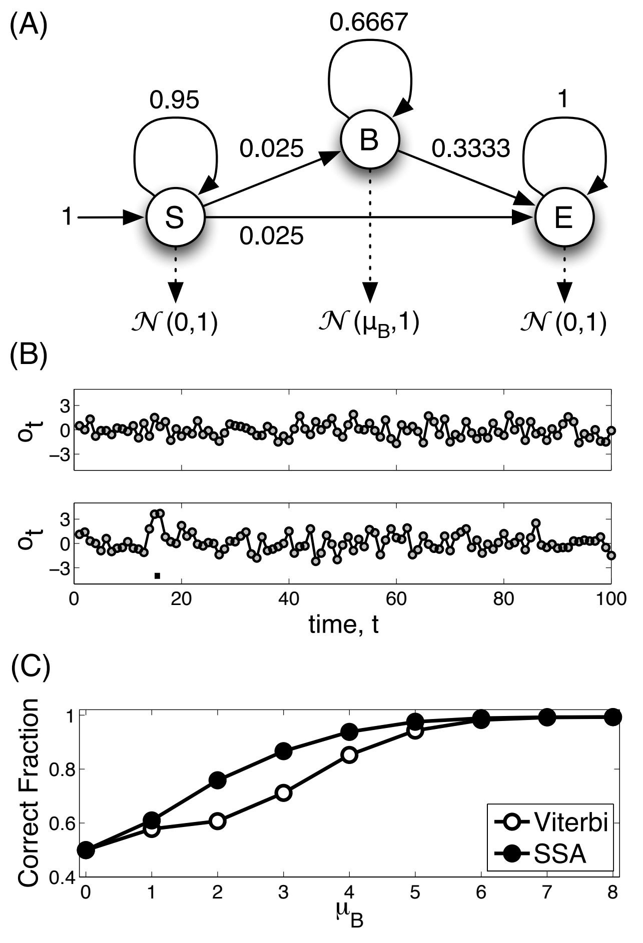
We simulated state-observation trajectories of 100 steps from the HMM. Because the chance of following the self-loop on state S is 0.95, the chance that a trajectory remains in that state for all 100 steps is $0.95^{99} \approx 0.0062$. Thus, almost all trajectories move on from the initial state, and they do so on average after 20 steps. From S, trajectories are equally likely to proceed to E or B, thus approximately half of all trajectories will contain a visit to state B, and half will not. Because the self-loop probability on state B is $2/3$, trajectories remain there on average for just three steps before moving on to state E. Figure 1B shows example observation trajectories that respectively do not and do include a visit to state B with $\mu_B = 4.5$. In the second trajectory, the visit to B happens on steps 15 and 16, as indicated by the black bar below.

We simulated 10,000 state-observation trajectories, and for each observation series applied the Viterbi algorithm and state sequence analysis. We "collapsed" the simulated state trajectory into the sequence of states visited, and likewise for the Viterbi solution. Then, we counted on how many of the 10,000 simulations Viterbi and/or state sequence analysis inferred the correct state sequence from the observation series. Figure 1C shows the fraction of correct state sequence inferences for each algorithm as a function of μ_B. When $\mu_B = 0$, there is no information to distinguish any of the states and so, unsurprisingly, both algorithms are right approximately half the time. Conversely, when μ_B is large, a brief visit to state B is so obvious that the sequence is clear, and both algorithms get nearly 100% of state sequence correct. In between, both algorithms have intermediate performance, but state sequence analysis is correct a greater fraction of the time. Figure 1D reports in greater detail the frequencies of simulated and inferred state sequences when $\mu_B = 5$. Both algorithms are correct more than 90% of the time, but state sequence analysis demonstrates substantially fewer "false positive" detections of state B than Viterbi does (66 versus 467), although it incurs a greater number of "false negatives" (inferring no B visit when there was one, 115 versus 36).

Figure 2A shows a similar problem where more than one event (visit to state B) can occur in a given trajectory, with a sample observation trajectory displayed in Figure 2B with $\mu_B = 4$. Again, we ran 10,000 simulations of 100 time steps to generate observation trajectories, and then applied Viterbi and state sequence analysis to estimate the underlying state sequences. Figure 2C shows the results. Similar to the previous example, when μ_B is high, so that events are clearly observed, both algorithms predict the number of events correctly. However, when μ_B is lower, state sequence analysis outperforms Viterbi. Unlike the previous example, state sequence analysis predicts correctly more often than Viterbi even when $\mu_B = 0$, so that observations are uninformative. With uninformative observations, state sequence analysis predicts the sequence with the highest a priori probability, which turns out to be $SBSBSBSBS$. By contrast, Viterbi computes the maximum probability trajectory to be $SS \ldots S$, but this corresponds to the state sequence S, which is of much lower probability. So, even with non-informative observations, state sequence analysis "guesses" correctly a greater fraction of the time. In part, this advantage extends to partially-informative observations. Figure 2D shows a heatmap of true numbers of events versus predicted numbers of events by the two algorithms, across our 10,000 simulations when $\mu_B = 2$. In this domain, we see that Viterbi tends to underestimate the number of events, whereas state sequence analysis appears much less biased.

4.2 ACTIVITY RECOGNITION DATASET

The increasing availability of various kinds of sensors allows us to collect and analyze data that was previously unthinkable. One place where sensors invade our lives is houses and apartments. The data collected from those sensors is the basis for designing various intelligent environments [Cook and Das, 2004, Augusto and Nugent, 2006] and several healthcare applications [Abowd et al., 2002, Suzuki et al., 2004]. In [van Kasteren et al., 2011], activity data was collected from several individuals living in an apartment/house. The data consists of sensor readings and activity annotations made either manually or automatically. The sensors installed around the house report, for example, open-close states of doors and cupboards, and pressure measurements on the couch, while the activities might in-



(D)

	Viterbi prediction				SSA prediction			
True sequence	S	SB	SE	SBE	S	SB	SE	SBE
S	0	0	60	6	0	0	66	0
SB	0	5	0	0	0	5	0	0
SE	0	0	4530	467	0	0	4931	66
SBE	0	0	36	4896	0	0	115	4817

Figure 1: Demonstration of State Sequence Analysis, and comparison with the Viterbi algorithm, on a simple detection problem. Based on a noisy time series, the problem is to infer the underlying sequence of states, which is approximately equivalent to determining whether the sequence contains a visit to the state B.

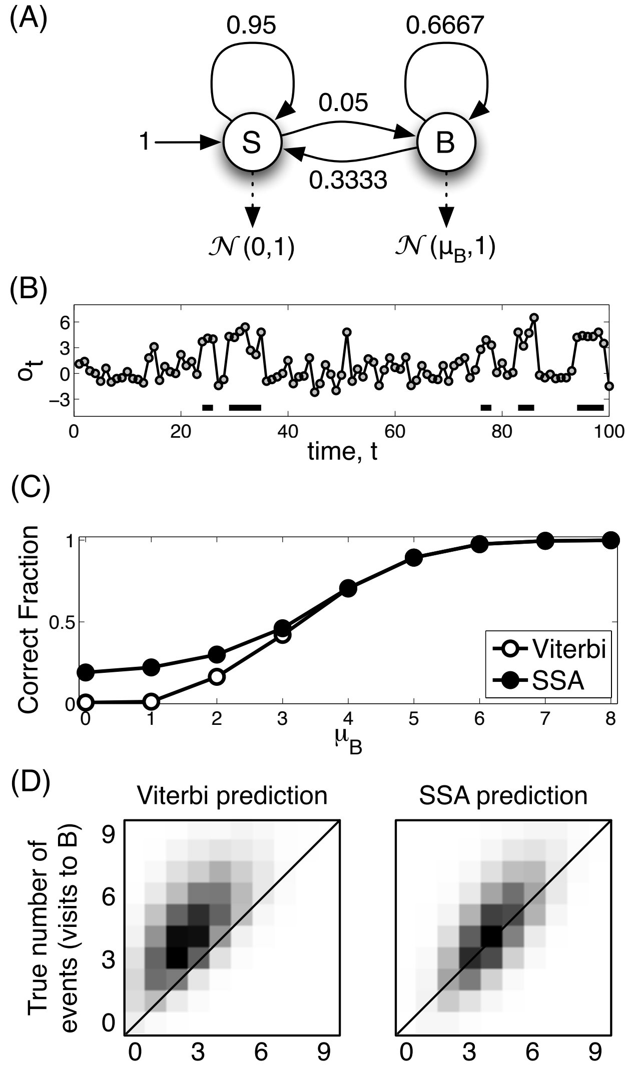


Figure 2: Comparison of State Sequence Analysis and Viterbi on a detection problem where multiple events per trajectory are possible.

Activities: *idle*, *leave house*, *use toilet*, *take shower*, *brush teeth*, *go to bed*, *prepare breakfast*, *prepare dinner*, *get snack*, *get drink*.
Sensors: *microwave*, *hall-toilet door*, *hall-bathroom door*, *cups cupboard*, *fridge*, *plates cupboard*, *front door*, *toilet flush*, *freezer*, *pans cupboard*, *groceries cupboard*, *hall-bedroom door*.

Table 1: House A: activities and sensors.

clude cooking, eating, taking a shower, etc. The authors trained several models that identify past activities given a time series of sensor readings. Modeling the data with an HMM and using the Viterbi algorithm to identify the activities was among the options that demonstrated competitive recognition performance. The reported results offer a useful benchmark to evaluate new methods of activity recognition in this setting.

Interestingly, the current problem formulation for activity recognition task around the house does not necessarily reflect the type of questions that the user is interested to find answers to. In particular, it is easy to imagine that what really matters to the user is the sequence of activities performed in a period of time, regardless of their duration. If an intelligent system is able to identify the activity perfectly at each point of time, one can obtain the sequence of activities simply by collapsing repetitive activities into a single value. Clearly, sensor readings do not always help to identify activities perfectly. As seen from the synthetic problem described earlier, when observations provide limited information about the underlying state, collapsing repetitive activities identified by the most probable state trajectory (Viterbi output) is often inferior to finding the most probable state sequence directly.

To illustrate the potential benefits of state sequence analysis on a realistic problem, we evaluate it on the largest out of three available datasets collected in [van Kasteren et al., 2011]. This dataset, named *House A*, contains 592 hours (nearly 25 days) of recorded activities and sensor readings of a single individual living in a one floor apartment. There are 14 sensors with binary outputs reporting different states of objects and 10 possible activities that the person can do at any single time period (see Table 1). A single record of activity and sensor readings was recorded every minute. To compare Viterbi and SSA approaches, 25 HMMs were trained, each on data from different sets of 24 days. These HMMs were then used, in combination with Viterbi or SSA, to make predictions on 24 trajectories (one hour each) taken from the remaining single day. For more details about the dataset see [van Kasteren et al., 2011].

As previously, we compared the performance of SSA to the results obtained by collapsing repetitive activities in the most probable state trajectory computed using the Viterbi algorithm. Out of 592 sample trajectories, Viterbi and SSA disagreed on 25 ($\approx 5\%$) of those. Further, we measured how well the produced state sequences match the true state sequences using the insert-delete string comparison[1]. On 7 trajectories the results of Viterbi produced a better score, while on 13 trajectories SSA performed better (for the other 5 the scores were equal). Although the difference might appear not very meaningful at first glance, there are several factors about the problem that need to be understood. First, some sensors identify the activities rather precisely, e.g., using toilet flush indicates that the toilet was used at that time, or opening the front door means that the person is leaving/coming back to the apartment. Second, most of the true state sequences are short (see Table 2, first line). In fact, around 75% of those contain only a single state, the majority of which are either *idling* (unidentified activity), *go to bed* or *leave house*. However, hours that include 3 activities or more are much more likely to produce sequences of observations on which Viterbi and SSA differ (see Table 2, second line). Considering that, it is expected to have a small number of trajectories on which Viterbi and SSA disagree. Among those, SSA provided a better prediction on nearly twice the number of trajectories on which Viterbi performed better. An example of a one hour trajectory where SSA improves over Viterbi inference is given in Figure 3. This is a typical example where sensor readings provide only indirect information about an activity that happens, which Viterbi algorithm fails to identify. From the computational perspective, it took about 5 minutes to run SSA algorithm on the entire dataset using a 3.40GHz CPU with 16GB memory machine, and a fraction of a second to find the most probable state trajectories (Viterbi).

To further establish the fact that SSA performs better as compared to Viterbi algorithm on this domain, we trained an HMM on the entire dataset and sampled a large number of state and observation trajectories. Specifically, for each starting state of an HMM we sampled 1000 one–hour–long trajectories and evaluated SSA and Viterbi on this simulated dataset. The results are presented in Figure 4. As expected, depending on the initial state of the HMM, there will be a different number of trajectories on which SSA and Viterbi state sequences are not equal. Moreover, depending on the initial state, the general performance of SSA and Viterbi, compared to each other, can be different. Nevertheless, in most cases SSA performed significantly better then Viterbi. In total, Viterbi scored better on 286 trajectories while SSA scored better on 494 trajectories, which amounts to more than 70% improvement in performance.

[1]This is similar to the well known edit/Levenshtein string distance, except that the replace operation is not directly allowed. In the context of activity recognition problems, it is better not to report a particular activity at all then to report a wrong activity.

length ≥ 1	length ≥ 2	length ≥ 3	length ≥ 4
100% (592)	$\approx 25\%$	$\approx 20\%$	$\approx 12\%$
100% (25)	96%	92%	52%

Table 2: House A: the percentage of state sequences of different length appearing in the dataset (line 1), and those whose Viterbi and SSA scores were different (line 2).

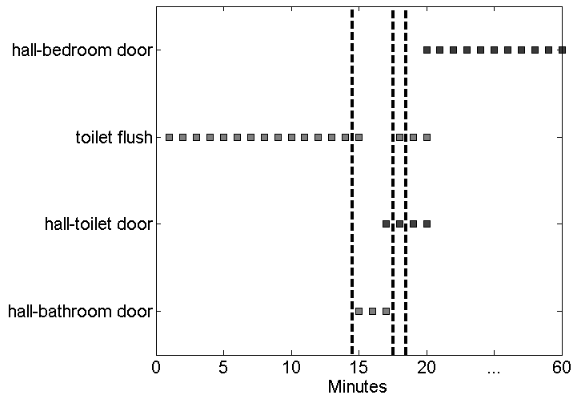


Figure 3: Example of a one hour trajectory of observations. Y axis enumerates binary sensor readings, and points on the plot identify active sensors. Note that we use the sensor readings representation that continues to be active until there is any change in other sensor readings (see more details in [van Kasteren et al., 2011]). Vertical dashed lines on the plots show the time of true activity change. The true underlying activity sequence is: *idle, brush teeth, use toilet, go to bed*. SSA outputs the true activity sequence, whose probability equals to 0.53. However, Viterbi's output omits the *brush teeth* activity, and the probability of resulting activity sequence is only 0.34.

5 CONCLUSION

Inference problem in Hidden Markov Models have received considerable attention, and several algorithms have been used in a variety of domains to infer the behavior of the underlying system this HMM represents, given a sequence of noisy observations. Typically, these algorithms estimate different quantities involving hidden variables (e.g., Viterbi, posterior decoder, most probable annotation sequence, etc. [Brejová et al., 2007, Lember and Koloydenko, 2014]) and therefore, at their core, address different inference problems. In this work, we point out that in a variety of domains none of these techniques is adequate enough to answer our question of interest, which is to find the most probable state sequence. Naturally, the closest existing approach that can be used to find the most probable state sequence is to find the most probable state trajectory using the Viterbi algorithm and collapse repetitive state visitations. However, by doing so, the recovered state sequence is not guaranteed to be the most probable state sequence. Moreover, the resulting state sequence might actually have a much lower probability of happening compared to the most probable one, as our experimental results suggest. In fact, those discrepancies are not surprising since, using Bayesian networks terminology, the Viterbi algorithm for HMMs produces *most probable explanation* (MPE) solution, while we are seeking to find a MAP solution different from MPE as we ignore dwelling time (variables) [Darwiche, 2009].

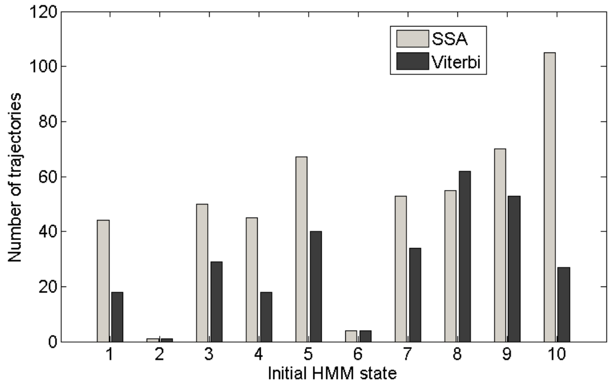


Figure 4: Comparison of SSA and Viterbi evaluated on simulated trajectories from *House A* dataset. X axis identifies the initial state of an HMM that the trajectories were sampled from. For each state, 1000 trajectories were sampled, and the performance of SSA was compared to Viterbi. As previously, trajectories on which the outputs of SSA and Viterbi differ were used to measure the performance by means of insert-delete string distance (Viterbi/SSA vs. true state sequence). Each bar in a category represents the number of trajectories on which the corresponding method scored better.

Building on earlier work where an algorithm to find the most probable state sequence in continuous time Markov chains was proposed [Levin et al., 2012], in this work we developed a state sequence analysis algorithm that finds the most probable state sequence of an HMM given a sequence of observations. The algorithm performs a search in the space of state sequences by evaluating those sequences and pruning parts of the search space by carefully maintaining a domination relationship between the sequences. We evaluated the algorithm on synthetic event–detection problems first to highlight its advantages over a Viterbi–based approach. Then, we used state sequence analysis to find the most probable sequence of activities based on a real activity recognition dataset collected from individuals performing different duties at home [van Kasteren et al., 2011]. Although the Viterbi algorithm performed reasonably well on this problem, the advantages of using state sequence analysis were nevertheless apparent.

Much remains to be done with respect to the analysis and evaluation of our proposed approach. One of the topics that requires further attention is the computational complexity of the algorithm. Although we did not experience difficulties running the algorithm on the problems presented in this paper, it is still not clear whether the problem of finding the most probable state sequence is polynomial or NP hard in general. Further, in many real dynamical systems that involve hidden variables, the model of choice is Hidden Semi-Markov Models (HSMM) [Yu, 2010]. It seems rather straightforward to adapt our algorithm to HSMMs, however experimental results are needed to verify its benefits and computational cost.

Acknowledgements

This work was supported in part by a Discovery grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) to TJP, and by an NSERC Postdoctoral Fellowship to YG.

References

[Abowd et al., 2002] Abowd, G. D., Bobick, A. F., Essa, I. A., Mynatt, E. D., and Rogers, W. A. (2002). The aware home: A living laboratory for technologies for successful aging. In *Proc. of the AAAI Workshop "Automation as Caregiver"*, pages 1–7.

[Augusto and Nugent, 2006] Augusto, J. C. and Nugent, C. D. (2006). *Designing smart homes: the role of artificial intelligence*, volume 4008. Springer Science & Business Media.

[Bahl et al., 1986] Bahl, L., Brown, P., de Souza, P. V., and Mercer, R. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing.*, volume 11, pages 49–52.

[Blei and Moreno, 2001] Blei, D. M. and Moreno, P. J. (2001). Topic segmentation with an aspect hidden Markov model. In *Proc. of ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 343–348. ACM.

[Brejová et al., 2007] Brejová, B., Brown, D. G., and Vinař, T. (2007). The most probable annotation problem in HMMs and its application to bioinformatics. *Journal of Computer and System Sciences*, 73(7):1060–1077.

[Cook and Das, 2004] Cook, D. and Das, S. (2004). *Smart environments: Technology, protocols and applications*, volume 43. John Wiley & Sons.

[Dardas and Georganas, 2011] Dardas, N. H. and Georganas, N. D. (2011). Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Trans. on Instrumentation and Measurement*, 60(11):3592–3607.

[Darwiche, 2009] Darwiche, A. (2009). *Modeling and reasoning with Bayesian networks*. Cambridge University Press.

[Haussler and Eeckman, 1996] Haussler, D. K. D. and Eeckman, M. G. R. F. H. (1996). A generalized hidden Markov model for the recognition of human genes in DNA. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, pages 134–142.

[Lember and Koloydenko, 2014] Lember, J. and Koloydenko, A. A. (2014). Bridging viterbi and posterior decoding: a generalized risk approach to hidden path inference based on hidden Markov models. *The Journal of Machine Learning Research*, 15(1):1–58.

[Levin et al., 2012] Levin, P., Lefebvre, J., and Perkins, T. J. (2012). What do molecules do when we are not looking? state sequence analysis for stochastic chemical systems. *Journal of The Royal Society Interface*, 9(77):3411–3425.

[Nguyen et al., 2005] Nguyen, N. T., Phung, D. Q., Venkatesh, S., and Bui, H. (2005). Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. In *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 955–960. IEEE.

[Perkins, 2009] Perkins, T. J. (2009). Maximum likelihood trajectories for continuous-time Markov chains. In *Advances in Neural Information Processing Systems*, pages 1437–1445.

[Qiao et al., 2002] Qiao, Y., Xin, X., Bin, Y., and Ge, S. (2002). Anomaly intrusion detection method based on HMM. *Electronics Letters*, 38(13):663–664.

[Rabiner, 1989] Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286.

[Schreiber and Karplus, 2015] Schreiber, J. and Karplus, K. (2015). Analysis of nanopore data using hidden Markov models. *Bioinformatics*, Epub ahead of print: Feb. 3.

[Sonnhammer et al., 1998] Sonnhammer, E. L., Von Heijne, G., Krogh, A., et al. (1998). A hidden Markov model for predicting transmembrane helices in protein sequences. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, pages 175–182.

[Suzuki et al., 2004] Suzuki, R., Ogawa, M., Otake, S., Izutsu, T., Tobimatsu, Y., Izumi, S.-I., and Iwaya, T. (2004). Analysis of activities of daily living in elderly people living alone: single-subject feasibility study. *Telemedicine Journal & E-Health*, 10(2):260–276.

[van Kasteren et al., 2011] van Kasteren, T., Englebienne, G., and Kröse, B. J. (2011). Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity recognition in pervasive intelligent environments*, pages 165–186. Springer.

[Wang et al., 2007] Wang, K., Li, M., Hadley, D., Liu, R., Glessner, J., Grant, S. F., Hakonarson, H., and Bucan, M. (2007). Penncnv: an integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data. *Genome research*, 17(11):1665–1674.

[Yamato et al., 1992] Yamato, J., Ohya, J., and Ishii, K. (1992). Recognizing human action in time-sequential images using hidden Markov model. In *Proc. IEEE Int. Conf. on Vision and Pattern Recognition*, pages 379–385.

[Yu, 2010] Yu, S.-Z. (2010). Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243.