Uncertain observation times

Shaunak Chatterjee and Stuart Russell

Computer Science Division,
University of California, Berkeley
Berkeley, CA 94720, USA
{shaunakc,russell}@cs.berkeley.edu

Abstract. Standard temporal models assume that observation times are correct, whereas in many real-world settings (particularly those involving human data entry) noisy time stamps are quite common. Serious problems arise when these time stamps are taken literally. This paper introduces a modeling framework for handling uncertainty in observation times and describes inference algorithms that, under certain reasonable assumptions about the nature of time-stamp errors, have linear time complexity.

1 Introduction

Real-world stochastic processes are often characterized by discrete-time state-space models such as hidden Markov models, Kalman filters, and dynamic Bayesian networks. In all of these models, there is a hidden (latent) underlying Markov chain and a sequence of observable outputs, where (typically) the observation variables depend on the corresponding state variables. Crucially, the *time* of an observation variable is not considered uncertain, so that the observation is always attached to the right state variables.

In practice, however, the situation is not always so simple—particularly when human data entry is involved. For example, a patient in an intensive care unit (ICU) is monitored by several sensors that record physiological variables (e.g., heart rate, breathing rate, blood pressure); for these sensors, the time stamps are reliable. In addition, the ICU nurse records annotated observations of patient state ("agitated," "coughing," etc.) and events ("suctioned," "drew blood," "administered phenylephrine," etc.). Each such annotation includes an accurate *data entry time* (generated by the data recording software) and a manually reported *event time* that purports to measure the *actual* event time. For example, at 11.00 the nurse may include in an hourly report the assertion that phenylephrine was administered at 10.15, whereas in fact the event took place at 10.05.

Such errors matter when their magnitude is non-negligible compared to the time-scale of the underlying process. For example, phenylephrine is a fast-acting vasopressor that increases blood pressure in one or two minutes. In the situation described above, a monitoring system that takes the reported event time of 10.15 literally would need to infer another explanation for the rapid rise in blood pressure at 10.06 (perhaps leading to a false diagnosis) and might also infer that

the drug injected at 10.15 was not in fact phenylephrine, since it had no observed effect on blood pressure in the ensuing minutes. Such errors in observation times would also cause serious problems for a learning system trying to learn a model for the dynamical system in question; moreover, reversals in the apparent order of events can confuse attempts to learn causal relations or expert policy rules. It would be undesirable, for example, to learn the rule that ICU nurses inject phenylephrine in response to an unexplained rise in blood pressure.

Similar examples of potentially noisy time stamps are found in manual data entry in biological labs, industrial plants, attendance logs, intelligence operations, and active warfare. These examples share a common trait—a sequence of manually entered observations complements continually recorded observations (spectrometer readings, CCTV footage, surveillance tapes, etc) that are temporally accurate. The process of reconstructing historical timelines suffers from "time-stamp" errors in all observation sequences—carbon dating, co-located artifacts, and contemporary sources may give incorrect or inexact ("near the end of the reign of ...") dates for events.

In this work, we present an extension of the hidden Markov model that allows for time-stamp errors in some or all observations. As one might expect, we include random variables for the data entry time, the manually reported event time, and the actual event time, and these connect the observation variable itself to the appropriate state variables via multiplexing. Of particular interest are the assumptions made about the errors—for example, the assumption that *event ordering* among manually reported events in a given reporting stream is not jumbled. We show that, under certain reasonable assumptions, inference in these models is tractable—the complexity of inference is $O(MS^2T)$, where M is the window size of the time stamp uncertainty, S is the state space size of the HMM and T is the length of the observation sequence.

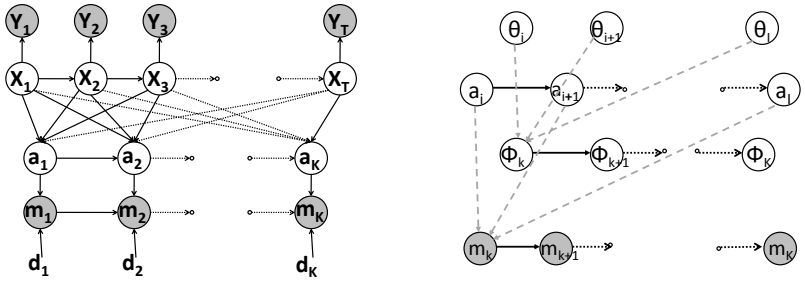
There has been a lot of work on state space models with multiple output sequences. Some authors have modeled observation sequences as non-uniform subsamples of single latent trajectory ([6, 4]) and thereby combined information sources. Others, namely [1, 2] (asynchronous HMMs (AHMMs)) and [5] (pair HMMs), have proposed alignment strategies for the different sequences using a common latent trajectory. AHMMs ([1]) are closely related to our work. However, the assumptions they make for the generative model of the less frequent observation sequence are different from ours and are not suited to the applications we have described. Also, in our case, the annotations come with noisy time stamps, which help us localize our search for the true time stamp. We also handle missing reports and false reports, which cannot be modeled in AHMMs.

The paper begins (Section 2) with the basic modeling framework for uncertainty in observation times. Section 3 presents a modified forward–backward algorithm for the basic model. Section 4 extends the model to accommodate unreported events and false reports of events, and Section 5 describes an exact inference algorithm for this extended model. The complexity of the exact algorithm is analyzed in Section 6 and some simplifications and approximations are

proposed. Section 7 presents some experiments to highlight the performance of the different algorithms.

2 Extending HMMs

A hidden Markov model (HMM) is a special case of the state space model where the latent variable is discrete. As shown in Figure 1(a), $\mathbf{X} = \{X_1, X_2, \ldots, X_T\}$ is a Markov process of order one and X_t is the hidden (latent) variable at time step t. There are two different observation sequences. $\mathbf{Y}$ is the variable observed at every time step (and is assumed to have the correct time stamp). Y_t corresponds to the observation at time t. $Y_{t_1:t_2}$ refers to the sequence of Y_t from $t = t_1$ to $t = t_2$ ($t_1 \leq t_2$). In the ICU, $\mathbf{Y}$ could be the various sensors hooked up to the patient. The other sequence of observations is less frequent and can be thought of as analogous to *annotations* or manual entries of *events*. In a sequence of T time-steps, there are K annotations ($K < T$) which mark K events. m_k represents the (potentially erroneous) time stamp of the *report* corresponding to the k^{th} event. a_k represents the actual time of occurrence of the k^{th} event. d_k is the time at which the time stamp data for the k^{th} (i.e. m_k) event was entered. In the ICU example from Section 1, m_k is 10:15, a_k is 10:05 and d_k is 11:00. d_k can be a parameter for the error model of the time stamp (i.e. $p(m_k|m_{k-1}, a_k)$). For instance, the noisy time stamp m_k can be no greater than d_k, if we exclude anticipatory data entry. M_k is the window of uncertainty of the k^{th} event and denotes the possible values of a_k (around m_k). So, if we assume that the nurse can err by at most 15 minutes, M_k is from 10:00 to 10:30.



(a) (b)

Fig. 1. (a) The extended hidden Markov model with actual and measured times of events. All X's are potential parents of each a_k and the connections depend on the values of X_i. Certain dependencies are denoted by solid lines, while value-dependent ones are dotted. (b) The generalized noisy time stamp hidden Markov model with actual and measured times of events. $\mathbf{X}$ and $\mathbf{Y}$ have been omitted for simplicity (they are identical to Figure 1(a)). Dependencies are only shown completely for ϕ_j and m_j. Color coding indicates definite dependencies (black) and value-dependent ones (gray).

A key assumption is that *the time stamps of events are chronologically ordered*. This restriction is analogous to the monotonicity of the mapping in time imposed on sequence matching in dynamic time warping [9]. Thus, m_k is strictly greater than m_{k-1}. This assumption holds vacuously if the events are identical and non-distinguishable. It also holds in several real-life scenarios.

The next important point is that there is a deterministic relationship between a_k's and $\mathbf{X}$. For clarity of presentation, let us consider the case where X_t is a binary random variable. $X_t = 1$ is the state corresponding to an event or annotation and $X_t = 0$ is the state representing a non-event. The generalization to the case where the state space is of size S is straight-forward and is presented in the supplementary material. a_k is the smallest i such that $\sum_{j=1}^{i} X_j = k$. The complete likelihood model is as follows:

$$p(X_{1:T}, Y_{1:T}, a_{1:K}, m_{1:K}) = \\ p(X_1)p(Y_1|X_1)\prod_{t=2}^{T} p(X_t|X_{t-1})p(Y_t|X_t)p(a_{1:K}|X_{1:T})\prod_{k=1}^{K} p(m_k|m_{k-1}, a_k)$$

For notational convenience, assume $m_0 = 0$. Since the a_k's are deterministically determined by the sequence $\mathbf{X}$, $p(a_{1:K}|X_{1:T})$ is zero for all $a_{1:K}$ instantiations except the one which corresponds to the given $X_{1:T}$ chain. Also, only those $X_{1:T}$ instantiations which have exactly K events will have non-zero probability support from the evidence $m_{1:K}$. In a later model, we will relax these constraints.

For now, we assume that every annotation corresponds to an event and every event has been recorded/annotated. Thus, it is justified to only consider latent variable trajectories with exactly K events. The inference task is to compute the posterior distributions of X_i and a_k conditioned on all the evidence available (namely $\mathbf{Y}$ and $m_{1:K}$). In the next section we describe an efficient algorithm for this task.

3 The modified forward-backward algorithm

The notation used in this section will be very similar to the standard notation used in the $\alpha - \beta$ forward backward algorithm as presented in [3]. $\alpha(a_k = t) = p(a_k = t, Y_{1:t}, m_{0:k})$ and will be simply written as $\alpha(a_k)$ when the context is clear. Thus, $\alpha(a_k)$ denotes the joint probability of all given data upto time a_k and the value of a_k itself. $\beta(a_k) = p(Y_{a_k+1:T}, m_{k+1:K}|a_k, m_k)$ represents the conditional probability of all future data given the value of a_k and m_k. Let $\mathcal{L}(a_k, a_{k+1}) = p(a_{k+1}, Y_{a_k+1:a_{k+1}}|a_k)$. This likelihood term can be simplified by

$$\begin{aligned}\mathcal{L}(a_k, a_{k+1}) &= p(a_{k+1}, Y_{a_k+1:a_{k+1}}|a_k) \\ &= \sum_{X_{a_k+1:a_{k+1}}} p(a_{k+1}, X_{a_k+1:a_{k+1}}, Y_{a_k+1:a_{k+1}}|a_k) \\ &= \prod_{t=a_k+1:a_{k+1}} p(Y_t|X_t)p(X_t|X_{t-1}),\end{aligned}$$

where $X_{a_k+1:a_{k+1}} = \{0, 0, \ldots, 0, 1\}$ since $p(a_{k+1}|a_k, X_{a_k+1:a_{k+1}}) = 0$ for every other $X_{a_k+1:a_{k+1}}$ sequence. The α update step is

$$\begin{aligned}\alpha(a_k) &= p(a_k, Y_{1:a_k}, m_{0:k}) \\ &= \sum_{a_{k-1}} \sum_{X_{1:a_k}} p(a_k, a_{k-1}, X_{1:a_k}, Y_{1:a_k}, m_{0:k}) \\ &= p(m_k|m_{k-1}, a_k) \sum_{a_{k-1}} \alpha(a_{k-1}) \mathcal{L}(a_{k-1}, a_k).\end{aligned}$$

The backward (smoothing) step is as follows:

$$\begin{aligned}\beta(a_k) &= p(Y_{a_k+1:T}, m_{k+1:K}|a_k, m_k) \\ &= \sum_{a_{k+1}} p(a_{k+1}, Y_{a_k+1:T}, m_{k+1:K}|a_k, m_k) \\ &= \sum_{a_{k+1}} \beta(a_{k+1}) p(m_{k+1}|m_k, a_{k+1}) \mathcal{L}(a_k, a_{k+1}).\end{aligned}$$

Given these definitions, the standard rule for computing the posterior still holds.

$$\gamma(a_k) = p(a_k|Y_{1:T}, m_{0:K}) \propto \alpha(a_k)\beta(a_k).$$

3.1 Computing $\gamma(X_i)$ from $\gamma(a_k)$

The final step of the algorithm would be to compute the conditional distributions of the hidden state variables X_i from $\gamma(a_k)$. This computation is straight-forward since X_i can only be 1 if in a chain, there exists a k such that $a_k = i$. It should also be noted that $a_k = i$ denotes that X_i is the k^{th} 1 in the sequence. Therefore, the $\mathbf{X}$ sequences contributing to $\gamma(a_k = i)$ and $\gamma(a_{k'} = i)$ are disjoint when $k \neq k'$. So the probability of an event at time i is just equal to the probability of any of the K events occurring at time i. The posterior distribution of X_i is given by

$$\gamma(X_i) = p(X_i = 1|Y_{1:T}, m_{0:K}) = \sum_{k=1}^{K} \gamma(a_k = i)$$

3.2 Tractable error models for m_k

In our analysis, we have conditioned the error model of m_k on the time stamp of the previous report m_{k-1} and the actual time of the k^{th} event a_k. The time of data entry d_k can also be a parameter in this conditional distribution and we could additionally condition on a_{k-1}. We cannot include any previous events or reports since that would destroy the first-order Markovian dynamics that we need for our analysis. However, with the allowed parameters, very flexible error models can be created. m_{k-1} as a parent can be used to model an expected gap between two reports. a_k, a_{k-1} and m_{k-1} together could be used to specify a (stochastic) relationship between the relative timings of events and their reports. Two sample error models are shown in Figure 2.

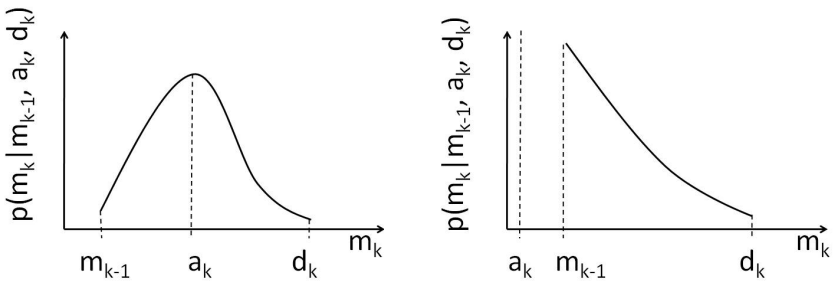


Fig. 2. Sample probability distributions for $p(m_k|m_{k-1}, a_k)$. The possible values of m_k are bounded by m_{k-1} (to satisfy the monotonicity constraint) and d_k (to exclude anticipatory entries), with some bias for a_k.

3.3 Complexity of the algorithm

Our analysis has assumed that there are K events in T time steps. Let us also assume (for simplicity) that all uncertainty windows are of the same size, i.e. $\forall k, |M_k| = M$. Let the maximum possible interval between $t_1 \in M_k$ and $t_2 \in M_{k+1}$ be I_k. Then the computation of $(\mathcal{L}(t_1, t_2)$ for all values of $\{t_1, t_2 : t_1 \in M_k, t_2 \in M_{k+1}, t_1 < t_2\}$ is an $O(M^2 + I_k)$ operation.

Once the relevant $\mathcal{L}(t_1, t_2)$ and $\alpha(a_k)$ values have been computed, the computation of $\alpha(a_{k+1})$ is an $O(M^2)$ operation. Thus the total complexity of the modified forward step is $O(KM^2 + \sum_k I_k)$. If we assume that only a constant number of uncertainty windows can overlap, then $\sum_k I_k = O(T)$ and $MK \leq O(T)$. Thus, the total complexity expression simplifies to $O(MT)$. The modified backward (or β) step has a similar complexity. Computing $\gamma(a_k)$ and $\gamma(X_i)$ are both $O(MK)$ operations. Thus, the overall complexity is $O(MT)$.

If we consider an HMM with $S + 1$ states, where state S corresponds to the annotation state, then the computation of $\mathcal{L}(t_1, t_2)$ becomes an $(M^2S^2 + MS^2I_k)$ operation. The other steps have the same complexity as in the previous analysis, so the overall complexity becomes $O(MS^2T)$. Thus, we see an M-fold increase in the inference complexity over a regular HMM.

The space complexity is $O(KM^2)$ for storing the relevant $\mathcal{L}(t_1, t_2)$ values and $O(KM)$ for storing the α, β and γ values. Thus, it is independent of the state space size. The algorithm can be trivially extended to handle cases with more than one type of event.

4 Unreported events and false reports

The model in section 2 assumes that every event is reported (with a possibly erroneous time stamp). However, in real life, events often go unreported. An example of this in the ICU setting would be a nurse forgetting to make an entry

of a drug administration because the recording was done in a batch fashion. Many events in history might go unreported by a historian if she does not come across sufficient evidence which warrants a report. Thus, negligence and ignorance would be primary causes for unreported events. Precisely speaking, an unreported event is an event (some $X_t = 1$) which does not generate an m_k.

Previously, we also assumed that every report corresponds to an actual event. This is also often violated in reality. False reports can occur when one event is entered twice (making one of them a false report) or more. Misinterpretation of observations could also lead to false reporting as in the case of historians often drawing contentious conclusions. In the model, a false report would correspond to an m_k which was not generated by any event.

We wish to extend our model to handle both of these artifacts. To this end, we introduce some new variables in the original model. Let us still assume that there are K reports of events. In addition, let us hypothesize I actual events. I can be chosen using prior knowledge about the problem (the rate of false reports and missed reports). For each hypothesized event a_i, we introduce a binary variable θ_i. $\theta_i = 0$ indicates that the event a_i is unreported, while $\theta_i = 1$ indicates that a_i has been reported and thus generates some m_k. $\mathbf{\Theta} = \{\theta_\mathbf{1}, \ldots, \theta_\mathbf{I}\}$ denotes the set of all θ_i. Now, for each report m_k, we introduce a new variable ϕ_k whose range is $\{0, 1, \ldots, I\}$. If the report m_k is generated by the event a_i then $\phi_k = i$. In other words, ϕ_k is the index of the (reported) event corresponding to the report m_k. As is obvious, $p(\phi_k = i|\theta_i = 0) = 0$. $\phi_k = 0$ means m_k is a false report. $\mathbf{\Phi}$ is the set of all ϕ_j. The generalized model is shown in Figure 1(b).

The deterministic relationship between $\mathbf{X}$ and $\mathbf{a}$ remains unaffected. The prior on θ_i can be problem-specific. For our analysis, we assume it is a constant. Let $p(\theta_i = 0) = \delta_i$. The conditional probability table for ϕ_j is as follows:

$$p(\phi_k|\phi_{k-1}, \theta_{1:I}) = \begin{cases} \epsilon_k, & \text{if } \phi_k = 0 \\ 1 - \epsilon_k, & \text{if } \phi_k = i, \theta_i = 1, \theta_{\phi_{k-1}:i-1} = 0 \end{cases}$$

The prior probability of a false report (modeled currently with a constant ϵ_k) can also be modeled in more detail to suit a specific problem. However, if m_k is not a false report (currently an event with probability $1 - \epsilon_k$), then ϕ_k is deterministically determined by ϕ_{k-1} and $\mathbf{\Theta}$. When $\phi_k = 0$, m_k is no longer parameterized by a_{ϕ_k}. The new distribution is represented as $\tilde{p}(m_k|m_{k-1})$.

5 Exact inference algorithm for the generalized model

We shall briefly explore the effect of a particular choice of I in Section 7. For inference in this generalized model, there is an added layer of complexity. We now have to enumerate all possible instances of $\mathbf{\Theta}$ and $\mathbf{\Phi}$. A meaningful decomposition of the posterior distribution (in the lines of the the standard forward-backward algorithm) and using dynamic programming could be a potential solution. All elements of $\mathbf{\Theta}$ are independent and hence enumerating all possibilities is infeasible. $\mathbf{\Phi}$ is a better proposition because there are dependencies that can be exploited - either the report is false (i.e. ϕ_k is 0) or it corresponds to an event after the previous reported actual event (i.e. $\phi_k > \phi_{k-1}$). We will use this

key fact to divide all possible instantiations of $\mathbf{\Phi}$ into some meaningful sets. Our main objective is to compute the posterior distribution $p(a_i|Y_{1:T}, m_{1:K})$, from which we can compute the posterior distribution of each X_i as described in Section 3.1. The posterior distribution for a_i is

$$\begin{aligned}\gamma(a_i) &= p(a_i|Y_{1:T}, m_{0:K}) \\ &\propto p(a_i, Y_{1:T}, m_{0:K}) \\ &= \sum_{\mathbf{\Phi}} p(a_i, \mathbf{\Phi}, Y_{1:T}, m_{0:K}).\end{aligned}$$

Now we will describe a way to partition the possible instantiations of $\mathbf{\Phi}$ which will then be used to formulate the forward and backward steps.

5.1 Partitioning the Φ sequences

Theorem 1. *For any i, such that $0 < i \leq I$, consider the following sets of ϕ sequences: $\mathcal{S}_0 = \{\phi_1 > i\}$; $\mathcal{S}_1 = \{\phi_1 \leq i$ and $\phi_2 > i\}$; $\mathcal{S}_2 = \{\phi_2 \leq i$ and $\phi_3 > i\}; \ldots$ $\mathcal{S}_K = \{\phi_K \leq i\}$*

The sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_K$ are disjoint and exhaustively cover all valid instantiations of $\mathbf{\Phi}$.

Proof. Intuitively, the set $\mathcal{S}_k$ corresponds to all the cases where the first i events generate the first k reports and the $k+1^{th}$ report is a true report.

Clearly, any sequence in $\mathcal{S}_0$ cannot belong to any other set. Any sequence ϕ belonging to $\mathcal{S}_1$ will have $\phi_2 > i$ and hence cannot belong to $\mathcal{S}_2$. Also, any sequence belonging to $\mathcal{S}_k$ will have $\phi_k \leq i$, which would imply $\phi_2 \leq i$. Thus ϕ cannot be in any $\mathcal{S}_k$ for $k \geq 2$. Similar arguments can be presented to show that $\forall k_1, k_2, \mathcal{S}_{k_1} \cap \mathcal{S}_{k_2} = \emptyset$. One important point to note is that all sequences in $\mathcal{S}_1$ have $\phi_2 \neq 0$, which means that ϕ_2 is not a false report in those cases.

Let ϕ be a valid instantiation of $\mathbf{\Phi}$. Now we have to show that every ϕ lies in some $\mathcal{S}_k$. The sequence $\phi = \{0, 0, \ldots, 0\}$ lies in $\mathcal{S}_K$. In every other sequence, there is at least some $\phi_j > 0$. If $\phi_j > i$, then that sequence belongs to $\mathcal{S}_{j-1}$. Thus, we have proved that the proposed partition of all valid instances of $\mathbf{\Phi}$ is both disjoint and exhaustive. Note that this partition is not unique, and the pivot (currently set to i) can be any value between 1 and I. □

5.2 Defining forward-backward steps

Now we can use the partitions $\mathcal{S}_k$ to define an efficient dynamic program to compute the posterior distribution of a_i. As we saw earlier,

$$\begin{aligned}\gamma(a_i) &\propto \sum_{\mathbf{\Phi}} p(a_i, \mathbf{\Phi}, Y_{1:T}, m_{1:K}) \\ &= \sum_{k=0}^{K} \sum_{\phi \in \mathcal{S}_k} p(a_i, \phi, Y_{1:T}, m_{1:K}) \\ &= \sum_{\phi \in \mathcal{S}_0} p(a_i, \phi, Y_{1:T}, m_{1:K}) + \sum_{k=1}^{K} \sum_{\phi \in \mathcal{S}_k} p(a_i, \phi, Y_{1:T}, m_{1:K})\end{aligned}$$

Let us denote $\sum_{\phi \in \mathcal{S}_k} p(a_i, \phi, Y_{1:T}, m_{1:K})$ by P_k. We can compute P_k by further decomposing it.

$$\begin{aligned}P_k &= \sum_{\phi_k \leq i} \sum_{\phi_{k+1} > i} p(a_i, \phi_k, \phi_{k+1}, Y_{1:T}, m_{1:K}) \\ &= \sum_{\phi_k \leq i} p(a_i, \phi_k, Y_{1:a_i}, m_{1:k}) \sum_{\phi_{k+1} > i} p(Y_{a_i+1:T}, m_{k+1:K}, \phi_{k+1} | a_i, m_k) \\ &= \alpha(a_i, m_k)\beta(a_i, m_k)\end{aligned}$$

Due to lack of space, we skip the detailed derivation of the update equations for the α and β expressions. Intuitively $\alpha(a_i, m_k)$ is the probability of the trajectories where the first k reports are associated with the first i events, whereas $\beta(a_i, m_k)$ is the probability of the trajectories where the last $K - k$ reports are associated with the last $I - i$ events. The initialization steps for α and β are straightforward. The order in which the α and β variables are computed is identical to other well-known dynamic programs of a similar structure ([7, 10]).

5.3 Computing $\gamma(a_i)$ and $\gamma(X_t)$

Once $\alpha(a_i, m_k)$ and $\beta(a_i, m_k)$ are computed for $\forall i, k$ s.t. $i \in \{1, 2, \ldots, I\}$ and $k \in \{0, 1, \ldots, K\}$, we can compute $\gamma(a_i)$ and $\gamma(X_t)$ by the following

$$\gamma(a_i) = \sum_{k=0}^{K} \alpha(a_i, m_k)\beta(a_i, m_k); \qquad \gamma(X_t) = \sum_{i=1}^{I} \gamma(a_i = t)$$

5.4 Multiple report sequences

Consider a scenario where there are R historians and each of them have their own set of annotations of historical events replete with time stamp conflicts. Since all historians do not concur on which events took place, there are instances of missed reports as well as false reports (assuming there is a set of actual events that took place). A simplifying assumption we make is that the historians reach

their conclusions independently based solely upon the latent state sequence ($\mathbf{X}$) and do not consult one another.

In this case, the addition to the generalized model from Section 4 is that the single report sequence $m_{1:K}$ is now replaced by R report sequences $m^{(r)} = m^{(r)}_{1:K_r}$, where $r \in \{1, 2, \ldots, R\}$. The key feature of the model which makes inference tractable (and very similar to the single report sequence case) is that *given the hidden state sequence* $\mathbf{X}$, $m^{(r_1)}$ *is independent of* $m^{(r_2)}$.

The posterior distribution of a_i is computed as follows:

$$\begin{aligned}
\gamma(a_i) &= \sum_{\mathbf{\Phi^{(1):(R)}}} p(a_i, \mathbf{\Phi^{(1):(R)}}, m^{(1):(R)}, Y_{1:T}) \\
&= p(a_i, Y_{1:T}) \prod_{r=1}^{R} \sum_{\mathbf{\Phi^{(r)}}} p(\mathbf{\Phi^{(r)}}, m^{(r)} | a_i, Y_{1:T}) \\
&\propto p(a_i, Y_{1:T})^{-R+1} \prod_{r=1}^{R} \sum_{\mathbf{\Phi^{(r)}}} p(\mathbf{\Phi^{(r)}}, m^{(r)}, a_i, Y_{1:T}) \\
&= p(a_i, Y_{1:T})^{-R+1} \prod_{r=1}^{R} \gamma(a_i^{(r)})
\end{aligned}$$

The $\gamma(a_i^{(r)})$ will be computed as before. $p(a_i, Y_{1:T})$ is proportional to $p(a_i|Y_{1:T})$ which can be computed using a standard forward-backward algorithm. $\gamma(X_t)$ is computed as before.

6 Complexity and simplifications

The algorithm presented in the previous section, while exact, is computationally very expensive. We now analyze the computational complexity of the exact algorithm and present some simplifications and possible approximation schemes.

6.1 Complexity Analysis

In the model where a_i corresponded to m_i, an uncertainty window resulted from the error model $p(m_i|m_{i-1}, a_i)$. If the error model suggested that m_i could only be within $M/2$ time units of a_i on either side, then this resulted in an uncertainty window of size M for a_i centered at m_i. However, when events can go unreported and reports can be false, the uncertainty window of a_i becomes much larger since we no longer know which (if any) m_k it corresponds to. The safe bet is to assume that $0 < a_i < T$ as long as it satisfies the monotonicity constraint (i.e. $a_{i-1} < a_i < a_{i+1}$). Thus, the uncertainty window in the worst case is $O(T)$.

If there are I hypothesized events and K reports (in the single report sequence case), then the complexity of the α computation step is $O(IKT^2)$. This is of course prohibitively expensive. However, there is a simplifying case.

6.2 Shifts in data entry

In the ICU setting, the nurse often enters data once an hour. A safe assumption is that all report(s) generated during the period between consecutive data entries correspond to the events in that same period. Let there be $\bar{I}$ hypothesized events and $\bar{K}$ reports in the time span $\bar{T}$ between two data entries. Then we can run the exact inference algorithm locally within the time span. The computational complexity will be $O(\bar{I}\bar{K}\bar{T}^2)$ for one time span. Over the entire time period T there will be $T/\bar{T}$ such time spans. Thus, the total computational complexity reduces to $O(\bar{I}\bar{K}\bar{T}T)$ which is much more tractable. *If the time span $\bar{T}$ is a constant, then the inference complexity is linear in T.*

6.3 Approximate inference

Another possible way to reduce the inference complexity is to not consider all possible trajectories of $\mathbf{\Phi}$. If the probability of a missed report (δ_i) and false report (ϵ_k) are small, then $\alpha(a_i, m_k)$ and $\beta(a_i, m_k)$ have significant non-zero values only when i and k are close to one another. We could potentially zero out all α and β values corresponding to $|i-k| > c$ where c is some threshold. This would naturally reduce the uncertainty window of event a_i which can now only be associated with some m_k where $i - c \leq k \leq i + c$. This means it suffices if $I = K + c$. If the reduced window size is $O(M_c)$ then the computational complexity of the algorithm becomes $O((K+c)KM_c^2 + T)$.

7 Experiments

For our experiments, we have primarily focused on simulations. The main reason for this choice is that we do not know the ground truth (correct time stamp of events) in the ICU data that we have been working on. Conversely we know the ground truth for our simulations and hence can evaluate our posterior inference results.

7.1 Simple model simulations

We set up an HMM with two states whose emission distributions were Gaussian with means $\mu_0 = -1$ and $\mu_1 = -3$ and standard deviation 0.5. The error model was $p(m_k|m_{k-1}, a_k) \sim \mathcal{N}(a_k, \sigma)$ with the window-size $M = 15$. The Gaussian was truncated at m_{k-1} and d_k. A standard HMM treats the time stamps as accurate (equivalent to a noise model with $\sigma = 0$). With this model, we generated data for $T = 1000$ for different values of σ (increasing steps of 0.5 from 0 to 5. We repeated this exercise 20 times to generate more simulations and remove random effects. All results are averaged over the 20 simulations and the bars indicate one standard deviation.

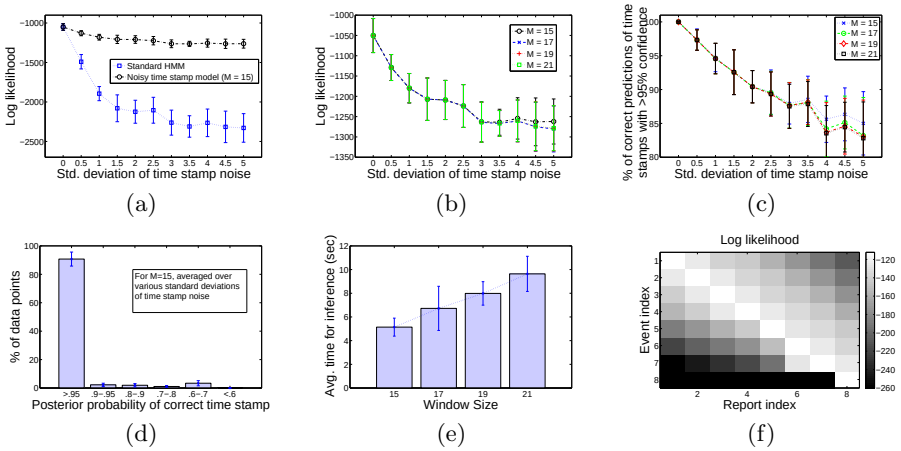


Fig. 3. (a) Average log likelihood of the simulated data using the two models. (b) Average log likelihood for models with various window sizes M. (c) % of high confidence correct time stamp inferences for varying window sizes. (d) Almost all time stamps are predicted with at least 60% accuracy. (e) Average time taken by the inference algorithms for different M. (f) Heat map of Q.

Increase in Likelihood. One objective of using the HMM with the noisy time stamp error model extension is to provide a better explanation for the data. This can be measured in terms of the likelihood. The average log likelihood of the data computed by a standard HMM and the noisy time stamp model are shown in Figure 3(a). The inference algorithms were run with the same transition and observation parameters used to generate the data.

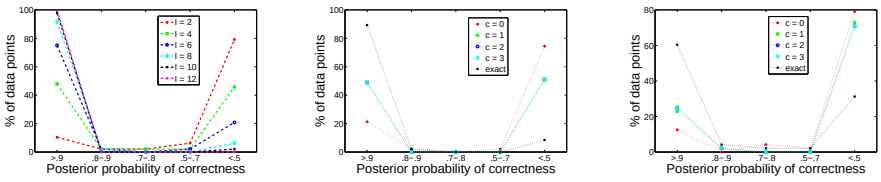
The difference in the two likelihoods increases as the variance of the time stamp noise increases, since this makes noisy time stamps more likely. The trend was similar for other values of $\{\mu_0, \mu_1\}$ and the two plots came closer as the two means became similar. Also, noteworthy is the fact that the likelihood of the data under the noisy model changes very little even as the noise increases - thus indicating robustness.

Next, we ran the inference algorithm with different window sizes ($M = 17$, 19 and 21). The likelihood of the data did not change significantly as shown in Figure 3(b). The time taken by the inference algorithm is also linear in the window size M as shown in Figure 3(e).

Accuracy of posterior inference. Another objective of the model is to accurately infer the correct time stamps of events. This will lead to better learning of the event characteristics. After computing the posterior distribution $\gamma(\mathbf{X})$, we looked at $\gamma(X_t)$ corresponding to all t which were correct time stamps of events (i.e. a_i). Figure 3(c) shows the percentage of events where the $\gamma(X_t)$ value exceeds .95. The percentage varies between 85% and 100% with the performance

degrading as the noise in the time stamp increases. *We can also see that the accuracy is not sensitive to the window size used in the inference.*

Figure 3(d) shows that there are almost no correct time stamps t where the $\gamma(X_t)$ value goes below .6. Thus, we do not miss any event completely. However, there are also some rare false positives. These result because the observation at the event's correct time stamp is not peaked enough to warrant a time stamp movement hypothesis in terms of likelihood.



(a) (b) (c)

Fig. 4. (a) Higher number of hypothesized events I has a high recall of correct time stamps. (b) Prediction accuracy of the diagonal approximation scheme. δ = .05 (c) δ = .3

7.2 Model with Missing and false reports

We generated simulation data using the generalized model for $T = 100$ and various values of δ and ϵ. For all settings, higher values of I had a high recall as seen in Figure 4(a). Although it would seem like a safe bet to set I high, this also leads to a lot of false positives, since a lot of events have to be hypothesized and accounted for. One possible approach to find a good I could be the likelihood measure. We consistently found that the data likelihood peaked at the value of I which corresponded to the correct number of events.

Another observation we made for small values of δ and ϵ was that most of the α and β values were concentrated along the diagonal. e take a look at the following matrix Q where $Q(i,k) = \sum_{t=1:T} \alpha(a_i = t, m_k)\beta(a_i = t, m_k)$ in Figure 3(f).

α and β entries were only considered along the (skewed) diagonal and c diagonals around that - the scheme described in Section 6.3. As we increase c from 0 (only the skewed diagonal) to larger values (more diagonals), our time stamp prediction accuracy increases as shown in Figure 4(b) and (c). However, the accuracy in the presence of these approximations is more when δ is smaller.

8 Conclusion

In this paper, we have proposed two model extensions of the HMM to deal with noisy time stamps of events. These models have inference algorithms quite similar in structure to the forward-backward algorithm used for inference in HMMs. It is easy to see how this model can be used in an EM setup to learn the error

model $p(m_k|m_{k-1}, a_k)$ or the transition model for $\mathbf{X}$ or the emission model for $\mathbf{Y}$. The algorithm is linear in T with one-to-one correspondence between events and reports. In other cases, certain reasonable assumptions can get it back to linear time. Noisy time stamps are pervasive in data - especially data recorded by humans (machines can also occasionally have logging errors). Algorithms which try to learn about human expertise will always have to deal with such data.

Looking ahead, it will be interesting to consult with doctors and run experiments on real data from the ICU. Another interesting direction is to model events which have a finite duration (and hence potential overlap). Such events could also be modeled with continuous time Bayesian networks (CTBNs) [8].

Acknowledgements We would like to acknowledge NSF (IIS-0904672 RI: Hierarchical Decision Making for Physical Agents) and DARPA (DSO contract FA8650-11-1-7153: Open-Universe Theory for Bayesian Information and Decision Systems) for their support and the anonymous reviewers for their comments and suggestions.

References

1. Samy Bengio. An asynchronous hidden markov model for audio-visual speech recognition. In *Advances in Neural Information Processing Systems, NIPS 15*. MIT Press, 2003.
2. Samy Bengio and Yoshua Bengio. An em algorithm for asynchronous input/output hidden markov models, 1996.
3. C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
4. A. Coates, P. Abbeel, and A.Y. Ng. Learning for control from multiple demonstrations. *Proceedings of 25th international conference on Machine learning*, 2008.
5. Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. Biological sequence analysis: probabilistic models of proteins and nucleic acids. cambridge univ, 1998.
6. J. Listgarten, R.M. Neal, S.T. Roweis, and A. Emili. Multiple alignment of continuous time series. In *Advances in Neural Information Processing Systems*, pages 817–824. MIT Press, 2005.
7. S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 1970.
8. U. Nodelman, C.R. Shelton, and D. Koller. Continuous time bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 378–387, 2002.
9. H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.
10. T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 1981.