Learning Continuous-Time Hidden Markov Models for Event Data

Yu-Ying Liu, Alexander Moreno, Shuang Li, Fuxin Li, Le Song, and James M. Rehg

Abstract The Continuous-Time Hidden Markov Model (CT-HMM) is an attractive modeling tool for mHealth data that takes the form of events occurring at irregularly-distributed continuous time points. However, the lack of an efficient parameter learning algorithm for CT-HMM has prevented its widespread use, necessitating the use of very small models or unrealistic constraints on the state transitions. In this paper, we describe recent advances in the development of efficient EM-based learning methods for CT-HMM models. We first review the structure of the learning problem, demonstrating that it consists of two challenges: (1) the estimation of posterior state probabilities and (2) the computation of end-state conditioned expectations. The first challenge can be addressed by reformulating the estimation problem in terms of an equivalent discrete time-inhomogeneous hidden Markov model. The second challenge is addressed by exploiting computational methods traditionally used for continuous-time Markov chains and adapting them to the CT-HMM domain. We describe three computational approaches and analyze the tradeoffs between them. We evaluate the resulting parameter learning methods in simulation and demonstrate the use of models with more than 100 states to analyze disease progression using glaucoma and Alzheimer's Disease datasets.

Introduction

The analysis of mobile health data can utilize a wide range of modeling and analysis tools for stochastic signals. One particularly attractive choice is the latent state model, which encodes measurement signals via the temporal evolution of a hidden state vector which emits the observations. Latent states define a level of abstraction over measured signals. States can be defined to correspond to behavioral constructs such as stress or craving, which are then connected to the underlying measurements

Y.-Y. Liu • A. Moreno (✉) • S. Li • L. Song • J.M. Rehg
Georgia Institute of Technology, Atlanta, GA, USA
e-mail: yuyingliu0823@gmail.com; alexander.f.moreno@gatech.edu; sli370@gatech.edu; lsong@cc.gatech.edu; rehg@gatech.edu

F. Li
2077 Kelley Engineering Center, Oregon State University, Corvallis, OR 97331, USA
e-mail: lif@eecs.oregonstate.edu

J.M. Rehg et al. (eds.), *Mobile Health*, DOI 10.1007/978-3-319-51394-2_19

via an observation model. The observation model also provides a means to describe the stochastic variability in the measurement sequence. Furthermore, any prior knowledge or constraints on the temporal evolution of the latent states can be captured by a model of the state dynamics. The interpretability of latent state models is an attractive feature. Since the latent states have a direct interpretation in the context of an experiment, the examination of latent state trajectories (following model fitting) is a potentially-powerful tool for gaining insight into complex temporal patterns. This is particularly important if the probability distributions obtained from latent state modeling are to be used in subsequent analysis steps, such as adjusting the tailoring variables in a mobile health intervention.

A standard latent variable model for mobile health data is the Hidden Markov Model (HMM). The Discrete Time HMM (DT-HMM) is widely used in speech recognition, robotics, signal processing, and other domains. It assumes that measurement data arrives at a fixed, regular sampling rate, and associates each measurement sample with an instantiated hidden state variable. The DT-HMM is an effective model for a wide range of time series data, such as the outputs of accelerometers, gyroscopes, and photoplethysmographic sensors. However, the fixed sampling rate assumptions that underlie the DT-HMM make it an inappropriate model choice for data that is distributed irregularly in time, such as event data. A classic example of a mobile health paradigm that generates event data is the use of Ecological Momentary Assessment (EMA) to ascertain the cognitive or emotional state of a participant. When an EMA is triggered, the participant is asked to respond to a number of questions using a smartphone interface. Since an EMA can be triggered at arbitrary times throughout the day, EMA data are most effectively modeled as event data. Even when EMAs are triggered at regular intervals, the participant usually has the option to postpone their response to the EMA (if they are driving or otherwise unavailable), and in addition the participant can choose to provide additional EMA datapoints at any time. In addition to EMA, many mHealth markers which are extracted from time series sensor data, such as periods of high stress or craving, also constitute event data since they can arise at any time.

A further disadvantage of using DT-HMMs to model event data is the fact that transitions in the hidden state are assumed to occur at the sample times. Since event data may be distributed sparsely in time, a more flexible model would allow hidden state transitions to occur between observations. One potential approach to using DT-HMMs with event data would be to set the sampling period fine enough to describe the desired state dynamics and then use a missing data model to address the fact that many sample times will not have an associated measurement. While this approach is frequently-used, it has several undesirable properties. First, the treatment of missing measurements can be both inefficient and inaccurate when the number of observations is sparse relative to the sampling rate. On the other hand, if the discretization is too coarse, many transitions could be collapsed into a single one, obscuring the actual continuous-time dynamics. Second, the sparsity of measurement can itself change over time. For example, during a demanding work week the frequency of high stress events could be high, while during a vacation the frequency of events could be much lower. The need to tradeoff between the temporal granularity at which state transitions can occur and the number of missing

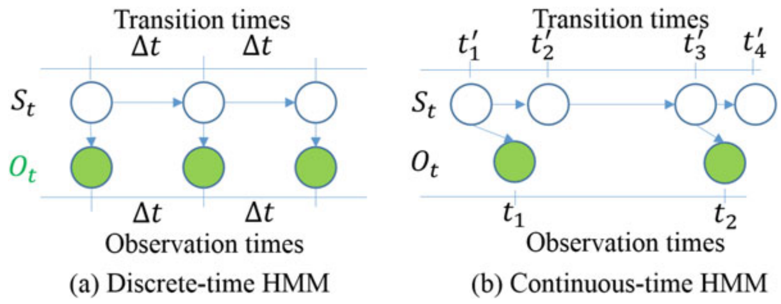

Fig. 1 The DT-HMM and the CT-HMM. In the DT-HMM, the observations O_t and state transitions S_t occur at fixed time intervals Δ_t, and the states S_t are the only source of latent information. In the CT-HMM, the observations O_t arrive at irregular time intervals, and there are two sources of latent information: the states S_t and the transition times $(t'_1, t'_2, \ldots)$ between the states

measurements which must be handled is a consequence of using a discrete time model to describe an inherently sparse, continuous-time measurement process.

A *Continuous-Time* HMM (CT-HMM) is an HMM in which both the transitions between hidden states and the arrival of observations can occur at arbitrary (continuous) times [7, 13]. It is therefore suitable for modeling a wide range of event data that is irregularly-sampled in time, including both mHealth data and clinical measurements [3, 17, 31]. However, the additional modeling flexibility provided by CT-HMM comes at the cost of a more complex inference procedure. In CT-HMM, not only are the hidden states unobserved, but the *transition times* at which the hidden states are changing are also unobserved. Moreover, multiple unobserved hidden state transitions can occur between two successive observations. Figure 1 gives a graphical model comparison of the CT-HMM and a regular HMM. The process of learning the parameters of a CT-HMM model from data is significantly more challenging computationally than the standard DT-HMM learning problem. There has been relatively little prior work on CT-HMM parameter learning. An approach by Jackson directly maximizes the data likelihood [13], but this method is limited to very small model sizes. A general Expectation-Maximization (EM) framework for continuous-time dynamic Bayesian networks, of which CT-HMM is a special case, was introduced in [24], but that work did not address the question of efficient learning. In general, the lack of an efficient parameter learning method for CT-HMM has been a barrier to the wide-spread use of this model [16], particularly for problems with large state spaces (hundreds of states or more).

This article describes a computational framework for CT-HMM learning which can efficiently handle a large number of states within an EM framework. It is based on [18], but includes additional algorithmic details and analysis of the computational cost of model learning. Further, we have improved the complexity of one of the approaches by a factor of the number of states. We begin in section "Continuous-Time Markov Chain" by introducing the mathematical definition of the Continuous-Time Markov Chain (CTMC). In a CTMC, the states are directly observable and there is no measurement process. It turns out that the key computations that are required for CT-HMM learning also arise in fitting CTMC models to data [12, 21, 28]. Section "Continuous-Time Hidden Markov Model" describes the addition of a measurement process which extends the CTMC model into a CT-HMM, and introduces the key equations that arise in parameter learning. Multiple approaches to the problem using EM are presented in section "EM Algorithms for CT-HMM". These approaches differ in the specific computational methods used in the E-step, and represent different approaches to solving the core computational problem that underlies EM for CT-HMM. In section "Experimental Results", we describe the results from an experimental evaluation of CT-HMM using both simulation studies and real-world clinical datasets. These results demonstrate the practical utility of CT-HMM for clinical data modeling. Note that our software implementation is available from our project website.[1]

Continuous-Time Markov Chain

A continuous-time Markov chain (CTMC) is defined by a finite and discrete state space S, a state transition rate matrix Q, and an initial state probability distribution π. The elements q_{ij} in Q describe the rate at which the process transitions from state i to j for $i \neq j$, and q_{ii} are specified such that each row of Q sums to zero ($q_i = \sum_{j \neq i} q_{ij}$, $q_{ii} = -q_i$) [7]. In a time-homogeneous process, in which the q_{ij} are independent of t, the sojourn time in each state i is exponentially-distributed with parameter q_i: $f_i(t) = q_i e^{-q_i t}$ with mean $1/q_i$. The probability that the process's next move is from state i to state j is given by q_{ij}/q_i. If a realization of the CTMC is *fully* observed, it means that one can observe every state transition time $(t'_0, t'_1, \ldots, t'_{V'})$, and the corresponding states $Y' = \{y_0 = s(t'_0), \ldots, y_{V'} = s(t'_{V'})\}$, where $s(t)$ denotes the state at time t. In that case, the complete likelihood (CL) of the data is

$$CL = \prod_{v'=0}^{V'-1} (q_{y_{v'}, y_{v'+1}} / q_{y_{v'}})(q_{y_{v'}} e^{-q_{y_{v'}} \tau_{v'}}) = \prod_{v'=0}^{V'-1} q_{y_{v'}, y_{v'+1}} e^{-q_{y_{v'}} \tau_{v'}}$$
$$= \prod_{i=1}^{|S|} \left[\prod_{j=1, j \neq i}^{|S|} q_{ij}^{n_{ij}} \right] e^{-q_i \tau_i} \quad (1)$$

where $\tau_{v'} = t'_{v'+1} - t'_{v'}$ is the time interval between two transitions, n_{ij} is the number of transitions from state i to j, and τ_i is the total amount of time the chain remains in state i.

In general, a realization of the CTMC is observed only at *discrete and irregular* time points $(t_0, t_1, \ldots, t_V)$, corresponding to a state sequence Y, which are *distinct* from the transition times. As a result, the Markov process between two consecutive observations is *hidden*, with potentially many unobserved state transitions. Thus, both n_{ij} and τ_i are unobserved. To express the likelihood of the incomplete

[1] http://www.cbs.gatech.edu/CT-HMM

observations, we can utilize a discrete time hidden Markov model by defining a state transition probability matrix for each time interval t, $P(t) = e^{Qt}$, where $P_{ij}(t)$, the entry (i,j) in $P(t)$, is the probability that the process is in state j after time t, given that it is in state i at time 0. This quantity takes into account all possible intermediate state transitions and timing between i and j which are not observed. Then the likelihood of the data is

$$L = \prod_{v=0}^{V-1} P_{y_v,y_{v+1}}(\tau_v) = \prod_{v=0}^{V-1} \prod_{i,j=1}^{|S|} P_{ij}(\tau_v)^{\mathbb{I}(y_v=i,y_{v+1}=j)} \tag{2}$$

where $\tau_v = t_{v+1} - t_v$ is the time interval between two observations, $\mathbb{I}(\cdot,\cdot)$ is the indicator function that is 1 if both arguments are true, otherwise it is 0. Note that there is no analytic maximizer of L, due to the structure of the matrix exponential, and direct numerical maximization with respect to Q is computationally challenging. This motivates the use of an EM-based approach.

An EM algorithm for CTMC learning is described in [21]. Based on Eq. (1), the expected complete log-likelihood takes the form

$$\sum_{i=1}^{|S|} \left[\sum_{j=1,j\neq i}^{|S|} \log(q_{ij}) \mathbb{E}[n_{ij}|Y,\hat{Q}_0] \right] - q_i \mathbb{E}[\tau_i|Y,\hat{Q}_0] \tag{3}$$

where $\hat{Q}_0$ is the current estimate for Q, and $\mathbb{E}[n_{ij}|Y,\hat{Q}_0]$ and $\mathbb{E}[\tau_i|Y,\hat{Q}_0]$ are the expected state transition count and total duration given the incomplete observation Y and the current transition rate matrix $\hat{Q}_0$, respectively. Once these two expectations are computed in the E-step, the updated $\hat{Q}$ parameters can be obtained via the M-step as

$$\hat{q}_{ij} = \frac{\mathbb{E}[n_{ij}|Y,\hat{Q}_0]}{\mathbb{E}[\tau_i|Y,\hat{Q}_0]}, i \neq j \quad \text{and} \quad \hat{q}_{ii} = -\sum_{j\neq i} \hat{q}_{ij}. \tag{4}$$

Now the main computational challenge is to evaluate $\mathbb{E}[n_{ij}|Y,\hat{Q}_0]$ and $\mathbb{E}[\tau_i|Y,\hat{Q}_0]$. By exploiting the properties of the Markov process, the two expectations can be decomposed as [6]:

$$\begin{aligned}
\mathbb{E}[n_{ij}|Y,\hat{Q}_0] &= \sum_{v=0}^{V-1} \mathbb{E}[n_{ij}|y_v, y_{v+1}, \hat{Q}_0] \\
&= \sum_{v=0}^{V-1} \sum_{k,l=1}^{|S|} \mathbb{I}(y_v = k, y_{v+1} = l) \mathbb{E}[n_{ij}|y_v = k, y_{v+1} = l, \hat{Q}_0] \\
\mathbb{E}[\tau_i|Y,\hat{Q}_0] &= \sum_{v=0}^{V-1} \mathbb{E}[\tau_i|y_v, y_{v+1}, \hat{Q}_0]
\end{aligned}$$

$$= \sum_{v=0}^{V-1} \sum_{k,l=1}^{|S|} \mathbb{I}(y_v = k, y_{v+1} = l) \mathbb{E}[\tau_i | y_v = k, y_{v+1} = l, \hat{Q}_0]$$

Thus, the computation reduces to computing the end-state conditioned expectations $\mathbb{E}[n_{ij}|y_v = k, y_{v+1} = l, \hat{Q}_0]$ and $\mathbb{E}[\tau_i|y_v = k, y_{v+1} = l, \hat{Q}_0]$, for all $k, l, i, j \in S$. These expectations are also a key step in CT-HMM learning, and section "EM Algorithms for CT-HMM" presents our approach to computing them.

Continuous-Time Hidden Markov Model

In this section, we describe the continuous-time hidden Markov model (CT-HMM) for disease progression and our approach to CT-HMM learning.

Model Description

In contrast to CTMC, where the states are directly observed, none of the states are directly observed in CT-HMM. Instead, the available observational data o depends on the hidden states s via the measurement model $p(o|s)$. In contrast to a conventional HMM, the observations $(o_0, o_1, \ldots, o_V)$ are only available at irregularly-distributed continuous points in time $(t_0, t_1, \ldots, t_V)$. As a consequence, there are two levels of hidden information in a CT-HMM. First, at observation time, the state of the Markov chain is hidden and can only be inferred from measurements. Second, the state transitions in the Markov chain between two consecutive observations are also hidden. As a result, a Markov chain may visit multiple hidden states before reaching a state that emits a noisy observation. This additional complexity makes CT-HMM a more effective model for event data in comparison to HMM and CTMC. But as a consequence the parameter learning problem is more challenging. We believe we are the first to present a comprehensive and systematic treatment of efficient EM algorithms to address these challenges.

A *fully observed* CT-HMM contains four sequences of information: the underlying state transition time $(t'_0, t'_1, \ldots, t'_{V'})$, the corresponding state $Y' = \{y_0 = s(t'_0), \ldots, y_{V'} = s(t'_{V'})\}$ of the hidden Markov chain, and the observed data $O = (o_0, o_1, \ldots, o_V)$ at time $T = (t_0, t_1, \ldots, t_V)$. Their joint complete likelihood can be written as

$$CL = \prod_{v'=0}^{V'-1} q_{y_{v'}, y_{v'+1}} e^{-q_{y_{v'}} \tau_{v'}} \prod_{v=0}^{V} p(o_v|s(t_v)) = \prod_{i=1}^{|S|} \left[\prod_{j=1, j \neq i}^{|S|} q_{ij}^{n_{ij}} \right] e^{-q_i \tau_i} \prod_{v=0}^{V} p(o_v|s(t_v)) \tag{5}$$

We make two simplifying assumptions. First, we assume that the observation time is independent of the states and the state transition times. Second, we assume that individual state trajectories are homogeneous, in that all sequences share the same global rate and emission parameters, which do not vary over time. With the first assumption, we do not require any further assumptions on the distribution of observation times. Furthermore, the observation time is not informative of the state.

We will focus our development on the estimation of the transition rate matrix Q. Estimates for the parameters of the emission model $p(o|s)$ and the initial state distribution π can be obtained from the standard discrete time HMM formulation [26], but with time-inhomogeneous transition probabilities, which we describe in section "Computing the Posterior State Probabilities". That is, the transition rates stay constant, but in the discrete-time formulation, the transition probabilities vary over time.

Parameter Estimation

We now describe an EM-based method for estimating Q from data. Given a current parameter estimate $\hat{Q}_0$, the expected complete log-likelihood takes the form

$$L(Q) = \left\{ \sum_{i=1}^{|S|} \left[\sum_{j=1, j\neq i}^{|S|} \log(q_{ij}) \mathbb{E}[n_{ij}|O, T, \hat{Q}_0] \right] - q_i \mathbb{E}[\tau_i|O, T, \hat{Q}_0] \right\} \tag{6}$$

$$+ \sum_{v=0}^{V} \mathbb{E}[\log p(o_v|s(t_v))|O, T, \hat{Q}_0]. \tag{7}$$

In the M-step, taking the derivative of L with respect to q_{ij} yields

$$\hat{q}_{ij} = \frac{\mathbb{E}[n_{ij}|O, T, \hat{Q}_0]}{\mathbb{E}[\tau_i|O, T, \hat{Q}_0]}, i \neq j \quad \text{and} \quad \hat{q}_{ii} = -\sum_{j\neq i} \hat{q}_{ij}. \tag{8}$$

The challenge lies in the E-step, where we compute the expectations of n_{ij} and τ_i conditioned on the observation sequence. The expectation for n_{ij} can be expressed in terms of the expectations between successive pairs of observations as follows:

$$E[n_{ij}|O, T, \hat{Q}_0] = \sum_{s(t_1),\ldots,s(t_V)} p(s(t_1),\ldots,s(t_V)|O, T, \hat{Q}_0) \mathbb{E}[n_{ij}|s(t_1),\ldots,s(t_V), \hat{Q}_0] \tag{9}$$

$$= \sum_{s(t_1),\ldots,s(t_V)} p(s(t_1),\ldots,s(t_V)|O, T, \hat{Q}_0) \sum_{v=1}^{V-1} \mathbb{E}[n_{ij}|s(t_v), s(t_{v+1}), \hat{Q}_0] \tag{10}$$

$$= \sum_{s(t_1),\ldots,s(t_V)} \sum_{v=1}^{V-1} p(s(t_1),\ldots,s(t_V)|O,T,\hat{Q}_0)\mathbb{E}[n_{ij}|s(t_v),s(t_{v+1}),\hat{Q}_0] \tag{11}$$

$$= \sum_{v=1}^{V-1} \sum_{s(t_v),s(t_{v+1})} p(s(t_v),s(t_{v+1})|O,T,\hat{Q}_0)\mathbb{E}[n_{ij}|s(t_v),s(t_{v+1}),\hat{Q}_0]$$

by marginalization (12)

$$= \sum_{v=1}^{V-1} \sum_{k,l=1}^{|S|} p(s(t_v)=k,s(t_{v+1})=l|O,T,\hat{Q}_0)$$
$$\mathbb{E}[n_{ij}|s(t_v)=k,s(t_{v+1})=l,\hat{Q}_0] \tag{13}$$

In a similar way, we can obtain an expression for the expectation of τ_i:

$$\mathbb{E}[\tau_i|O,T,\hat{Q}_0] = \sum_{v=1}^{V-1} \sum_{k,l=1}^{|S|} p(s(t_v)=k,s(t_{v+1})=l|O,T,\hat{Q}_0)$$
$$\mathbb{E}[\tau_i|s(t_v)=k,s(t_{v+1})=l,\hat{Q}_0]. \tag{14}$$

Note that, in contrast to the CTMC case, during CT-HMM learning we cannot observe the states directly at the observation times. Therefore, while the sum of expectations is weighted via indicator variables in the CTMC case, the weights are probabilities obtained through inference in the CT-HMM case.

The key to efficient computation of the expectations in Eqs. (13) and (14) is to exploit the structure of the summations. These summations have an inner-outer structure, which is illustrated in Fig. 2. The key observation is that the measurements partition the continuous timeline into intervals. It is therefore sufficient to compute the distribution over the hidden states at two successive observations, denoted by $p(s(t_v) = k, s(t_{v+1}) = l|O,T,\hat{Q}_0)$, and use these probabilities to weight the expectation over unobserved state transitions, which we refer to as the *end-state conditioned expectations* $\mathbb{E}[n_{ij}|s(t_v) = k, s(t_{v+1}) = l, \hat{Q}_0]$ and $\mathbb{E}[\tau_i|s(t_v) = k, s(t_{v+1}) = l, \hat{Q}_0]$. We present three methods that can be used to compute the end-state conditioned expectations in section "EM Algorithms for CT-HMM". We now describe our approach to computing the hidden state probabilities at the observations.

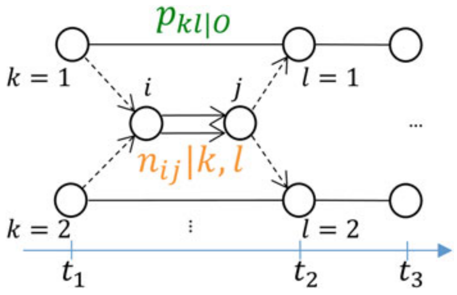

Fig. 2 Illustration of the decomposition of the expectation calculations (Eq. 13) according to their inner-outer structure, where k and l represent the two possible end-states at successive observation times (t_1, t_2), and i, j denotes a state transition from i to j within the time interval. $p_{kl|O}$ represents $p(s(t_v) = k; s(t_{v+1}) = l|O, T, \hat{Q}_0)$ and $n_{ij}|k, l$ denotes $E[n_{ij}|s(t_v) = k, s(t_{v+1}) = l, \hat{Q}_0]$ in Eq. (13)

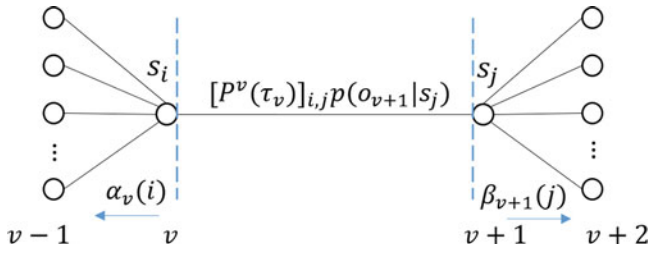

Fig. 3 Illustration of the computation of the posterior state probabilities $p(s(t_v) = k, s(t_{v+1}) = l|O, T, \hat{Q}_0)$. An equivalent time-inhomogeneous HMM is formed where the state transition probability matrix varies over time (denoted as $P^v(\tau_v)$ here). α and β are the forward and backward variables used in the forward-backward algorithm [26]

Computing the Posterior State Probabilities

The challenge in efficiently computing $p(s(t_v) = k, s(t_{v+1}) = l|O, T, \hat{Q}_0)$ is to avoid the explicit enumeration of all possible state transition sequences and the varying time intervals between intermediate state transitions (from k to l).

The key is to note that the posterior state probabilities are only needed at the times where we have observation data. We can exploit this insight to reformulate the estimation problem in terms of an equivalent discrete *time-inhomogeneous* hidden Markov model. This is illustrated in Fig. 3.

Specifically, given the current estimate $\hat{Q}_0$, O and T, we divide the timeline into V intervals, each with duration $\tau_v = t_v - t_{v-1}$. We then make use of the transition property of CTMC, and associate each interval v with a state transition matrix $P^v(\tau_v) := e^{\hat{Q}_0\tau_v}$. Together with the emission model $p(o|s)$, this results in a discrete time-inhomogeneous hidden Markov model with joint likelihood:

$$\prod_{v=1}^{V}[P^v(\tau_v)]_{(s(t_{v-1}),s(t_v))}\prod_{v=0}^{V}p(o_v|s(t_v)). \tag{15}$$

The formulation in Eq. (15) allows us to reduce the computation of $p(s(t_v) = k, s(t_{v+1}) = l|O, T, \hat{Q}_0)$ to familiar operations. The forward-backward algorithm [26] can be used to compute the posterior distribution of the hidden states, which we refer to as the *soft* method. This gives the probabilities $p(s(t_v) = k, s(t_{v+1}) = l|O, T, \hat{Q}_0)$, which sum to 1 over k and l. Alternatively, the MAP assignment of hidden states obtained from the Viterbi algorithm can provide an approximate distribution, which we refer to as the *hard* method. This gives $p(s(t_v) = k, s(t_{v+1}) = l|O, T, \hat{Q}_0) = 1$ for a single value of k and l, and $p(s(t_v) = k, s(t_{v+1}) = l|O, T, \hat{Q}_0) = 0$ for all the others. The forward-backward and Viterbi algorithms are then the same as in [26], except that we replace the transition matrix with $P^v(\tau_v)$ for each observation.

The *hard* method is potentially faster, but is less accurate in the case of multimodal posteriors. The *soft* method is more accurate, but requires expectation calculations for every k and l. Note that the *hard* method is only faster when the computation of the end-state conditioned expectations for a single start and end state is less expensive than computing them for all states, which we will see is not always the case.

EM Algorithms for CT-HMM

Pseudocode for the EM algorithm for CT-HMM parameter learning is shown in Algorithm 1. Multiple variants of the basic algorithm are possible, depending upon the choice of method for computing the end-state conditioned expectations, along with the choice of *hard* or *soft* decoding for obtaining the posterior state probabilities in Eq. (15).

The remaining step in finalizing the EM algorithm is to discuss the computation of the end-state conditioned expectations (ESCE) for n_{ij} and τ_i from Eqs. (13) and (14), respectively. The first step is to express the expectations in integral form, following [11]:

$$\mathbb{E}[n_{ij}|s(0) = k, s(t) = l, Q] = \frac{q_{i,j}}{P_{k,l}(t)} \int_0^t P_{k,i}(x)P_{j,l}(t-x)\,dx \tag{16}$$

$$\mathbb{E}[\tau_i|s(0) = k, s(t) = l, Q] = \frac{1}{P_{k,l}(t)} \int_0^t P_{k,i}(x)P_{i,l}(t-x)\,dx. \tag{17}$$

From Eq. (16), we define $\tau_{k,l}^{i,j}(t) = \int_0^t P_{k,i}(x)P_{j,l}(t-x)dx = \int_0^t (e^{Qx})_{k,i}(e^{Q(t-x)})_{j,l}\,dx$, while $\tau_{k,l}^{i,i}(t)$ can be similarly defined for Eq. (17) (see [24] for a related construction). Three primary methods for computing $\tau_{k,l}^{i,j}(t)$ and $\tau_{k,l}^{i,i}(t)$ have been proposed in the CTMC literature: an eigendecomposition based method, which we refer to as *Eigen*, a method called *uniformization* (*Unif*), and a method from Van Loan [30]

Algorithm 1: CT-HMM Parameter Learning (Soft/Hard)

```
1: Input: data $O = (o_0, \ldots, o_V)$ and $T = (t_0, \ldots, t_V)$, state set $S$, edge set $L$, initial guess of $Q$
2: Output: transition rate matrix $Q = (q_{ij})$
3: Find all time intervals between events $\tau_v = t_{v+1} - t_v$ for $v = 1, \ldots, V-1$, where
   $t_1 = t_0 = 0$
4: Compute $P(\tau_v) = e^{Q\tau_v}$ for each $\tau_v$
5: repeat
6:     Compute $p(v, k, l) = p(s(t_v) = k, s(t_{v+1}) = l | O, T, Q)$ for all $v$, and the
       complete/state-optimized data likelihood $l$ by using Forward-Backward (soft) or Viterbi
       (hard)
7:     Use Expm, Unif or Eigen method to compute $\mathbb{E}[n_{ij}|O, T, Q]$ and $\mathbb{E}[\tau_i|O, T, Q]$
8:     Update $q_{ij} = \frac{\mathbb{E}[n_{ij}|O,T,Q]}{\mathbb{E}[\tau_i|O,T,Q]}$, and $q_{ii} = -\sum_{i \neq j} q_{ij}$
9: until likelihood $l$ converges
```

for computing integrals of matrix exponentials, which we call *Expm*. *Eigen* and *Unif* both involve expressing the terms $P_{k,i}(x)P_{j,l}(t-x)$ as summations and then integrating the summations. Eigen utilizes an eigendecomposition-based approach, while Unif is based on series approximations. *Expm* notes a connection between the integrals and a system of differential equations, and solves the system. We describe each method, show how to improve the complexity of the *soft Eigen* method, and discuss their tradeoffs.

Across the three methods, the bottleneck is generally matrix operations, particularly matrix multiplication. Our finding is that with our improvements, *soft Eigen* is the preferred method except in the case of an unstable eigendecomposition. It is efficient due to having few matrix multiplications and it is accurate due to being a soft method. We find in our experiments that it is very fast (see Fig. 5) and that the stability of *Eigen* is usually not a problem when using random initialization. However, in the case where *Eigen* is unstable in any iteration, the alternatives are *soft* Expm, which has the advantage of accuracy, and *hard* Unif, which is often faster. Note that one can switch back to *Eigen* again once the likelihood is increasing.

The Eigen Method

The calculation of the ESCE $\tau_{k,l}^{i,i}(t)$ and $\tau_{k,l}^{i,j}(t)$ can be done in closed-form if Q can be diagonalized via its eigendecomposition (the *Eigen* method [20, 21]). Consider the eigendecomposition $Q = UDU^{-1}$, where the matrix U consists of all eigenvectors associated with the corresponding eigenvalues of Q in the diagonal matrix $D = diag(\lambda_1, \ldots, \lambda_n)$. Then we have $e^{Qt} = Ue^{Dt}U^{-1}$ and the integral can be written as:

$$\tau_{k,l}^{i,j}(t) = \sum_{p=1}^{n} U_{kp}U_{pi}^{-1} \sum_{q=1}^{n} U_{jq}U_{ql}^{-1} \Psi_{pq}(t) \tag{18}$$

where the symmetric matrix $\Psi(t) = [\Psi_{pq}(t)]_{p,q\in S}$ is defined as:

$$\Psi_{pq}(t) = \begin{cases} te^{t\lambda_p} & \text{if } \lambda_p = \lambda_q \\ \frac{e^{t\lambda_p} - e^{t\lambda_q}}{\lambda_p - \lambda_q} & \text{if } \lambda_p \neq \lambda_q \end{cases} \tag{19}$$

We now describe a method for vectorizing the *Eigen* computation, which results in improved complexity in the *soft* case. Let $V = U^{-1}$, $\circ$ be the Hadamard (elementwise) product, and V_i^T refer to the ith column of V, and U_j the jth row of U, then

$$\tau_{k,l}^{i,j}(t) = [U[V_i^T U_j \circ \Psi]V]_{kl} \tag{20}$$

This allows us to perform only one matrix construction for all k, l, but still requires two matrix multiplications for each ij with an allowed transition or edge.[2]

We now show how to reuse the matrix-matrix products across edges and replace them by a Hadamard product to improve efficiency further. A similar idea was explored in [19], but their derivation is for the gradient calculation of a CTMC, which we extend to EM for CT-HMMs. The intuition is that since matrix multiplication is expensive, by rearranging matrix operations, we can do one matrix multiplication, cache it, and reuse it so that we only do elementwise matrix products for every possible transition combination i and j, rather than doing matrix multiplications for every such combination.

Let F be a matrix given by $F_{kl} = \frac{p(s(t_v)=k, s(t_{v+1})=l|O,T,\hat{Q}_0)}{P_{kl}(t)}$, and let $A^{ij} = V_i^T U_j \circ \Psi$. Then

$$\sum_{k,l=1}^{|S|} p(s(t_v) = k, s(t_{v+1}) = l|O, T, \hat{Q}_0)\mathbb{E}[n_{ij}|s(t_v) = k, s(t_{v+1}) = l, \hat{Q}_0] \tag{21}$$

$$= q_{ij} \sum_{k,l=1}^{|S|} ([U[V_i^T U_j \circ \Psi]V] \circ F)_{kl} \tag{22}$$

$$= q_{ij} \sum_{k,l=1}^{|S|} ([UA^{ij}V] \circ F)_{kl} \tag{23}$$

Now note these two properties of the trace and Hadamard product, which hold for any matrices A, B, C, D:

$$\sum_{ij} (A \circ B)_{ij} = \mathrm{Tr}(AB^T) \tag{24}$$

$$\mathrm{Tr}(ABCD) = \mathrm{Tr}(BCDA) \tag{25}$$

[2]Note that a version of Eq. (20) appears in [21], but that version contains a small typographic error.

Then

$$\sum_{k,l=1}^{|S|}([UA^{ij}V]\circ F)_{kl} = \mathrm{Tr}(UA^{ij}VF^T) \tag{26}$$

$$= \mathrm{Tr}(A^{ij}VF^TU) \tag{27}$$

$$= \sum_{kl}(A^{ij}\circ(VF^TU)^T)_{kl} \tag{28}$$

$$= \sum_{kl}(A^{ij}\circ\underbrace{(U^TFV^T)}_{\text{reuse}})_{kl} \tag{29}$$

The term U^TFV^T is not dependent on i,j: only A^{ij} is, and A^{ij} does not require any matrix products to construct. Thus for each event or time interval, the naïve implementation of (20) required two matrix products for each i,j that form an edge. Through the preceding construction, we can reduce this to only two matrix products in total. We replaced all the other matrix products with Hadamard products. This improves the complexity by a factor of S, the number of states. The case for the ESCE of the duration τ_i is similar. Letting $A^i = V_i^TU_i\circ\Psi$ and using the subscript v to denote the matrix constructed for observation v, the final expectations are

$$E[n_{ij}|O,T,\hat{Q}_0] = q_{ij}\sum_{v=1}^{V-1}\sum_{kl}(A_v^{ij}\circ(U^TF_vV^T))_{kl} \tag{30}$$

$$E[\tau_i|O,T,\hat{Q}_0] = \sum_{v=1}^{V-1}\sum_{kl}(A_v^{i}\circ(U^TF_vV^T))_{kl} \tag{31}$$

Note that the *Hard* eigen method avoids explicitly summing over all k and l states. The key matrix manipulation then is the construction of matrices where the rows and columns correspond to k and l, respectively. Therefore, *hard* eigen has the same complexity as *soft* eigen when it is formulated as in Eqs. (30) and (31). Thus, *soft* eigen is the preferred choice.

Computing the ESCE using the *Eigen* method Algorithm 2 presents pseudocode for our *Eigen* method using the Hadamard product and trace manipulations. The algorithm does two matrix multiplications for each observation, and does only Hadamard products for each state and edge after that.

Stability of the Eigen Method

In general, *soft Eigen* is the fastest soft method, but Qt can suffer from an ill-conditioned eigendecomposition which can prevent the method from being usable. In prior CTMC works [20, 21], Metzner et al. mention that the eigendecomposition

Algorithm 2: Eigen Algorithm for ESCE

```
 1: Perform eigendecomposition $Q = UDV$ using balancing, where $V = U^{-1}$
 2: for $v = 1$ to $V - 1$ do
 3:     Compute $\tau_v = t_{v+1} - t_v$, set $t = \tau_v$
 4:     Compute $\Psi$ with $t = \tau_v \Rightarrow O(S^2)$
 5:     Compute $F_{k,l} = \frac{p(s(t_v)=k, s(t_{v+1})=l|O,T,\hat{Q}_0)}{P(t)}$
 6:     Compute $B = U^T F V^T$
 7:     for each state $i$ in S do
 8:         $A = V_i^T U_i \circ \Psi$
 9:         $E[\tau_i|O,T,Q] += \sum_{k,l=1}^{|S|} (A \circ B)_{kl} \Rightarrow O(S^2)$
10:     end for
11:     for each edge $(i,j)$ in $L$ do
12:         $A = V_i^T U_j \circ \Psi \Rightarrow O(S^2)$
13:         $\mathbb{E}[n_{ij}|O,T,Q] += q_{ij} \sum_{k,l=1}^{|S|} (A \circ B)_{kl} \Rightarrow O(S^2)$
14:     end for
15: end for
```

can potentially be ill-conditioned, but do not characterize the scope of this problem, which we discuss now in more detail. Both the eigenvalue and eigenvector estimation problems can be ill-conditioned. For the eigenvalue problem, the primary issue is the condition number of the eigenvector matrix. This follows from the Bauer-Fike Theorem [4], which gives a bound on the error in estimating the eigenvalues (as a result of a perturbation ΔQ of the Q matrix):

$$\min_{\lambda \in \lambda(Q)} |\lambda - \mu| \leq ||U|| \cdot ||\Delta Q|| \cdot ||U^{-1}|| \tag{32}$$

$$= \kappa(U)||\Delta Q||. \tag{33}$$

The error between an eigenvalue μ of $Q + \Delta Q$ and the true eigenvalue λ is bounded by the matrix norm of the perturbation, $||\Delta Q||$, and the condition number $\kappa(U)$ of the eigenvector matrix U of Q. We now discuss the impact of each of these two terms. The perturbation of Q, $||\Delta Q||$, is often due to rounding error and thus depends on the norm of Q. A class of methods known as balancing or diagonal scaling [25] can help reduce the norm of Q. In our experiments, balancing did not provide a significant improvement in the stability of the eigendecomposition, leading us to conclude that rounding error was not a major factor. The condition number $\kappa(U)$ captures the structural properties of the eigenvector matrix. We found empirically that certain pathological structures for the Q matrix, such as sparse triangular forms, can produce poor condition numbers. We recommend initializing the Q matrix at the start of EM with randomly-chosen values in order to prevent the inadvertent choice of a poorly-conditioned U. We found that uniform initialization, in particular, was problematic, unless random perturbations were added.

Having discussed the eigenvalue case, we now consider the case of the eigenvectors. For an individual eigenvector r_j, the estimation error takes the form

$$\Delta r_j = \sum_{k \neq j} \frac{l_k \Delta Q r_j}{\lambda_j - \lambda_k} r_k + O(||\Delta Q||^2), \tag{34}$$

where l_k are left eigenvectors, r_j and r_k are right eigenvectors, and λ_j, λ_k are eigenvalues of Q (see [5] for details). Thus the stability of the eigenvector estimate degrades when eigenvalues are closely-spaced, due to the term $\lambda_j - \lambda_k$ in the denominator. Note that this condition is problematic for the ESCE computation as well, as can be seen in Eq. (19). As was the case for the eigenvalue problem, care should be taken in initializing Q.

In summary, we found that randomly initializing the Q matrix was sufficient to avoid problems at the start of EM. While it is difficult in general to diagnose or eliminate the possibility of stability problems during EM iterations, we did not encounter any significant problems in using the *Eigen* approach with random initialization in our experiments. We recommend monitoring for a decrease in the likelihood and switching to an alternate method for that iteration in the event of a problem. One can switch back to *Eigen* once the likelihood is increasing again.

Expm Method

Having described an eigendecomposition-based method for computing the ESCE, we now describe an alternative approach based on a classic method of Van Loan [30] for computing integrals of matrix exponentials. In this approach, an auxiliary matrix A is constructed as $A = \begin{bmatrix} Q & B \\ 0 & Q \end{bmatrix}$, where B is a matrix with identical dimensions to Q. It is shown in [30] that

$$\int_0^t e^{Qx} B e^{Q(t-x)} dt = (e^{At})_{(1:n),(n+1):(2n)} \tag{35}$$

where n is the dimension of Q. That is, the integral evaluates to the upper right quadrant of e^{At}. Following [12], we set $B = I(i,j)$, where $I(i,j)$ is the matrix with a 1 in the (i,j)th entry and 0 elsewhere. Thus the left hand side reduces to $\tau_{k,l}^{i,j}(t)$ for all k, l in the corresponding matrix entries, and we can leverage the substantial literature on numerical computation of the matrix exponential. We refer to this approach as *Expm*, after the popular Matlab function. This method can be seen as expressing the integral as a solution to a differential equation. See Sect. 4 of [12] for details.

The most popular method for calculating matrix exponentials is the Padé approximation. As was the case in the *Eigen* method, the two issues governing the accuracy of the Padé approximation are the norm of Q and the eigenvalue spacing. If the norms are large, scaling and squaring, which involves exploiting the identity $e^A = (e^{A/m})^m$ and using powers of two for m, can be used to reduce the norm. To understand the role of Eigenvalue spacing, consider that the Padé approximation involves two series expansions $N_{pq}(Qt)$ and $D_{pq}(Qt)$, which are used to construct the matrix exponential as follows:

$$e^{Qt} \approx [D_{pq}(Qt)]^{-1} N_{pq}(Qt) \tag{36}$$

When the eigenvalue spacing *increases*, $D_{pq}(Qt)$ becomes closer to singular, causing large errors [22, 23].

The maximum separation between the eigenvalues is bounded by the Gershgorin Circle Theorem [9], which states that all of the eigenvalues of a rate matrix lie in a circle in the complex plane centered at the largest rate, with radius equal to that rate. That is, all eigenvalues $\lambda \in \lambda(Q)$ lie in $\{z \in \mathbb{C} : |z - \max q_i| \leq \max q_i\}$. This construction allows us to bound the maximum eigenvalue spacing of Qt (considered as a rate matrix). Two eigenvalues cannot be further apart than twice the absolute value of the largest magnitude diagonal element. Further, scaling and squaring helps with this issue, as it reduces the magnitude of the largest eigenvalue. Additional details can be found in [10, 22, 23].

Because scaling and squaring can address any stability issues associated with the Padé method, we conclude that *Expm* is the most stable method for computing the ESCE. However, we find it to be dramatically slower than *Eigen* (especially given our vectorization and caching improvements), and so it should only be used if *Eigen* fails.

The *Expm* algorithm does not have an obvious hard variant. Hard variants involve calculating expectations conditioned on a single start state k and end-state l for the interval between the observations. However, *Expm*, by virtue of using the Padé approximation of the matrix exponential, calculates it for all k and l. The output of the matrix exponential gives a matrix where each row corresponds to a different k and each column a different l. Developing a hard variant would thus require a method for returning a single element of the matrix exponential more efficiently than the entire matrix. One direction to explore would be the use of methods to compute the action of a matrix exponential $e^{At}x$, where x is a vector with a single 1 and 0's elsewhere, without explicitly forming e^{At} (see [2]).

Computing the ESCE using the *Expm* method Algorithm 3 presents pseudocode for the *Expm* method for computing end-state conditioned expectations. The algorithm exploits the fact that the A matrix does not change with time t. Therefore, when using the *scaling and squaring* method [10] for computing matrix exponentials, one can easily cache and reuse the intermediate powers of A to efficiently compute e^{At} for different values of t.

Uniformization

We now discuss a third approach for computing the ESCE. This was first introduced by Hobolth and Jensen [12] for the CTMC case, and is called *uniformization* (*Unif*). *Unif* is an efficient approximation method for computing the matrix exponential

Algorithm 3: Expm Algorithm for ESCE

```
1: for $v = 1$ to $V - 1$ do
2:     $\tau_v = t_{v+1} - t_v$, set $t = \tau_v$
3:     for each state $i$ in $S$ do
4:         $D_i = \frac{(e^{At})_{(1:n),(n+1):(2n)}}{P_{kl}(t)}$, where $A = \begin{bmatrix} Q & I(i,i) \\ 0 & Q \end{bmatrix}$
5:         $\mathbb{E}[\tau_i|O,T,Q] += \sum_{(k,l)\in L} p(s(t_v) = k, s(t_{v+1}) = l|O,T,\hat{Q}_0)(D_i)_{k,l}$
6:     end for
7:     for each edge $(i,j)$ in $L$ do
8:         $N_{ij} = \frac{q_{ij}(e^{At})_{(1:n),(n+1):(2n)}}{P_{kl}(t)}$, where $A = \begin{bmatrix} Q & I(i,j) \\ 0 & Q \end{bmatrix}$
9:         $\mathbb{E}[n_{ij}|O,T,Q] += \sum_{(k,l)\in L} p(s(t_v) = k, s(t_{v+1}) = l|O,T,\hat{Q}_0)(N_{ij})_{k,l}$
10:    end for
11: end for
```

$P(t) = e^{Qt}$ [12, 14]. It gives an alternative description of the CTMC process and illustrates the relationship between CTMCs and DTMCs (see [27]). The idea is that instead of describing a CTMC by its rate matrix, we can subdivide it into two parts: a Poisson process $\{N(t) : t \geq 0\}$ with mean $\hat{q}$, where $N(t)$ refers to the number of events under the Poisson process at time t, and a DTMC and its associated transition matrix R. The state of the CTMC at time t is then equal to the state after $N(t)$ transitions under the DTMC transition matrix R. In order to represent a CTMC this way, the mean of the Poisson process and the DTMC transition matrix must be selected appropriately.

Define $\hat{q} = \max_i q_i$, and matrix $R = \frac{Q}{\hat{q}} + I$, where I is the identity matrix. Then,

$$e^{Qt} = e^{\hat{q}(R-I)t} = \sum_{m=0}^{\infty} R^m \frac{(\hat{q}t)^m}{m!} e^{-\hat{q}t} = \sum_{m=0}^{\infty} R^m Pois(m; \hat{q}t), \tag{37}$$

where $Pois(m; \hat{q}t)$ is the probability of m occurrences from a Poisson distribution with mean $\hat{q}t$. The expectations can then be obtained by directly inserting the e^{Qt} series into the integral:

$$\tau_{k,l}^{i,i} = \sum_{m=0}^{\infty} \frac{t}{m+1} [\sum_{n=0}^{m} (R^n)_{ki} (R^{m-n})_{il}] Pois(m; \hat{q}t) \tag{38}$$

$$\tau_{k,l}^{i,j} = \frac{R_{ij} \sum_{m=1}^{\infty} [\sum_{n=1}^{m} (R^{n-1})_{ki} (R^{m-n})_{jl}] Pois(m; \hat{q}t)}{P_{kl}(t)} \tag{39}$$

The main difficulty in using *Unif* in practice lies in determining the truncation point for the infinite sum. However, for large values of $\hat{q}t$, we have $Pois(\hat{q}t) \approx \mathcal{N}(\hat{q}t, \hat{q}t)$, where $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution and one can then bound the truncation error from the tail of Poisson by using the cumulative normal distribution [28].

Algorithm 4: Unif Algorithm for ESCE

```
1: Set $\hat{t} = \max t_\Delta$; set $\hat{q} = \max_i q_i$.
2: Let $R = Q/\hat{q} + I$. Compute $R, R^2, \ldots, R^{\hat{M}}, \hat{M} = \ulcorner 4 + 6\sqrt{\hat{q}\hat{t}} + (\hat{q}\hat{t}) \urcorner \Rightarrow O(\hat{M}S^3)$
3: for $v = 1$ to $V - 1$ do
4:     $\tau_v = t_{v+1} - t_v$, set $t = \tau_v$
5:     $M = \ulcorner 4 + 6\sqrt{\hat{q}t} + (\hat{q}t) \urcorner$;
6:     for each state $i$ in $S$ do
7:         $E[\tau_i | s(0) = k, s(t) = l, Q] = \frac{\sum_{m=0}^{M} \frac{t}{m+1} [\sum_{n=0}^{m} (R^n)_{ki} (R^{m-n})_{il}] Pois(m; \hat{q}t)}{P_{kl}(t)} \Rightarrow O(M^2)$
8:         $E[\tau_i | O, T, Q] += p(s(t_v) = k, s(t_{v+1}) = l | O, T, \hat{Q}_0) E[\tau_i | s(0) = k, s(t) = l]$
9:     end for
10:    for each edge $(i, j)$ in $L$ do
11:        $E[n_{ij} | s(0) = k, s(t) = l, Q] = \frac{R_{ij} \sum_{m=1}^{M} [\sum_{n=1}^{m} (R^{n-1})_{ki} (R^{m-n})_{jl}] Pois(m; \hat{q}t)}{P_{kl}(t)} \Rightarrow O(M^2)$
12:        $E[n_{ij} | O, T, Q] += p(s(t_v) = k, s(t_{v+1}) = l | O, T, \hat{Q}_0) E[n_{ij} | s(0) = k, s(t) = l]$
13:    end for
14: end for
15: Soft: $O(\hat{M}S^3 + VS^3M^2 + VS^2LM^2)$; Hard: $O(\hat{M}S^3 + VSM^2 + VLM^2)$
```

Our implementation uses a truncation point at $M = \ulcorner 4 + 6\sqrt{\hat{q}t} + (\hat{q}t) \urcorner$, which is suggested in [28] to have error bound of 10^{-8}.

Computing the ESCE using the *Unif* method Algorithm 4 presents pseudocode for the *Unif* method for computing end-state conditioned expectations. The main benefit of *Unif* is that the R sequence ($R, R^2, \ldots, R^{\hat{M}}$) can be precomputed (line 2) and reused, so that no additional matrix multiplications are needed to obtain all of the expectations. One main property of *Unif* is that it can evaluate the expectations for only the two specified end-states, and it has $O(M^2)$ complexity, which is not related to S (when given the precomputed R matrix series).

One downside of *Unif* is that if $\hat{q}_i t$ is very large, so is the truncation point M. The computation can then be very time consuming. We find that *Unif*'s running time performance depends on the data and the underlying Q values. The time complexity analysis is detailed in Algorithm 4 line 15. This shows that the complexity of *soft Unif* is unattractive, while *hard Unif* may be attractive if *Eigen* fails due to instability.

Summary of Time Complexity

To compare the computational cost of different methods, we conducted an asymptotic complexity analysis for the five combinations of *hard* and *soft* EM with the methods *Expm*, *Unif*, and *Eigen* for computing the ESCE. The complexities are summarized in Table 1. *Eigen* is the most attractive of the soft methods at $O(VS^3 + VLS^2)$, where V is the number of visits, S is the number of states, and L is the number of edges. Its one drawback is that the eigendecomposition may

Table 1 Time complexity comparison of all methods in evaluating all required expectations under *Soft/Hard* EM

Complexity	Expm	Unif	Eigen
Soft EM	$O(VS^4 + VLS^3)$	$O(MV^3 + VS^3M^2 + VS^2LM^2)$	$O(VS^3 + VLS^2)$
Hard EM	$O(VS^4 + VLS^3)$	$O(MS^3 + VSM^2 + VLM^2)$	N/A

S number of states, L number of edges, V number of visits, M the largest truncation point of the infinite sum for *Unif*, set as $\ulcorner 4 + 6\sqrt{\hat{q}\hat{t}} + (\hat{q}\hat{t})\urcorner$, where $\hat{q} = \max_i q_i$, and $\hat{t} = max_\Delta \tau_v)$

become ill-conditioned at any iteration. However, in our experiments, with a random initialization, we found Eigen to be successful, and other papers have found similar results [21], although generally with a smaller number of states. If *Eigen* fails, *Expm* provides an alternative soft method, and *Unif* provides an alternative hard method. *Hard Unif* is often faster than *Expm* in practice, so we recommend running that first to get a sense of how long *Expm* will take, and if it is feasible, run *Expm* afterwards.

The time complexity comparison between *Expm* and *Unif* depends on the relative size of the state space S and M, where $M = \ulcorner 4 + 6\sqrt{\max_i q_i t} + (\max_i q_i t)\urcorner$ is the largest truncation point of the infinite sum used in *Unif* (see Table 1). It follows that *Unif* is more sensitive to $\max_i q_i t$ than *Expm* (quadratic vs. log dependency). This is because when *Expm* is evaluated using the scaling and squaring method [10], the number of matrix multiplications depends on the number of applications of matrix scaling and squaring, which is $\ulcorner \log_2(||Qt||_1/\theta_{13})\urcorner$, where $\theta_{13} = 5.4$ (the *Pade* approximant with degree 13). If scaling of Q is required [10], then we have $\log_2(||Qt||_1) \leq \log_2(S \max_i q_i t)$. Therefore, the running time of *Unif* will vary with $\max q_i t$ more dramatically than *Expm*.

When selecting an EM variant, there are three considerations: stability, time, and accuracy. Overall, *soft Eigen* offers the best tradeoff between speed and accuracy. However, if it is not stable, then *soft Expm* will generally have higher accuracy than *hard Unif*, but may be less efficient.

In some applications, event times are distributed irregularly over a discrete timescale. For example, hospital visits may be identified by their date but not by their time. In that case, the interval between two events will be a discrete number. In such cases, events with the same interval, e.g. with 5 days between visits, can be pooled and the ESCE can be computed once for all such events. See [18] for details, including the supplementary material for complexity analysis.

Experimental Results

We evaluated our EM algorithms in simulation (section "Simulation Performance on a 5-state Complete Digraph") and on two real-world datasets (section "Application of CT-HMM to Analyzing Disease Progression"): a glaucoma dataset using a model with 105 states and an Alzheimer's Disease dataset with 277 states. This is a significant advance in the ability to work with large models, as previous CT-HMM works [13, 16, 31] employed fewer than 100 states. We initialized the rate matrix uniformly with a small random perturbation added to each element. The random perturbation avoids a degenerate configuration for the *Eigen* method, while uniform initialization makes the runtimes comparable across methods. We used balancing for the eigendecomposition. Our timing experiments were run on an early 2015 MacBook Pro Retina with a 3.1 GHz Intel Core i7 processor and 16 GB of memory.

Simulation Performance on a 5-state Complete Digraph

We evaluated the accuracy of all methods on a 5-state complete digraph with synthetic data generated under different noise levels. Each q_i was randomly drawn from $[1, 5]$ and then q_{ij} was drawn from $[0, 1]$ and renormalized such that $\sum_{j \neq i} q_{ij} = q_i$. The state chains were generated from Q, such that each chain had a total duration around $T = \frac{100}{\min_i q_i}$, where $\frac{1}{\min_i q_i}$ is the largest mean holding time. The data emission model for state i was set as $\mathcal{N}(i, \sigma^2)$, where σ varied under different noise level settings.

The observations were then sampled from the state chains with rate $\frac{0.5}{\max_i q_i}$, where $\frac{1}{\max_i q_i}$ is the smallest mean holding time, which was ensured to be dense enough to make the chain identifiable. A total of 10^5 observations were sampled. The convergence threshold for EM was a change in the relative data likelihood of $\leq 10^{-8}$. The average 2-norm relative error $\frac{||\hat{q}-q||}{||q||}$ was used as the performance metric, where $\hat{q}$ is a vector of the learned q_{ij} parameters, and q is the ground truth.

The simulation results from five random runs are given in Table 2. *Expm*, *Unif*, and *Eigen* produced nearly identical results, and so they are combined in the table, which focuses on the difference between hard and soft variants. We found *Eigen* to be stable across all runs. All soft methods achieved significantly higher accuracy than hard methods, especially for higher observation noise levels. This can be attributed to the maintenance of the full hidden state distribution, leading to improved robustness to noise.

Application of CT-HMM to Analyzing Disease Progression

In the next set of experiments, we used the CT-HMM to analyze and visualize disease progression patterns from two real-world datasets of glaucoma and Alzheimer's

Table 2 The average 2-norm relative error from five random runs on a 5-state complete digraph under varying measurement noise levels

Error	$\sigma = 1/4$	$\sigma = 3/8$	$\sigma = 1/2$	$\sigma = 1$	$\sigma = 2$
Soft	0.026 ± 0.008	0.032 ± 0.008	0.042 ± 0.012	0.199 ± 0.084	0.510 ± 0.104
Hard	0.031 ± 0.009	0.197 ± 0.062	0.476 ± 0.100	0.857 ± 0.080	0.925 ± 0.030

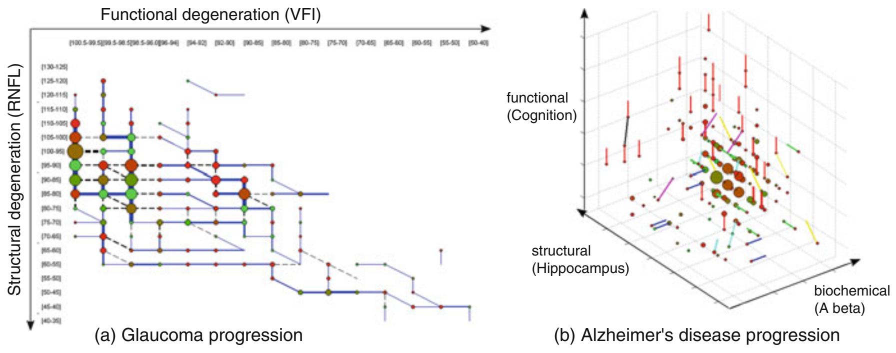

Fig. 4 Visualization of disease progression from two datasets: (**a**) Nodes represent states of glaucoma, with the node color encoding the average sojourn time (*red* to *green*: 0–5 years and above). The *blue* links between nodes indicate the most probable (i.e. strongest) transitions between adjacent states, selected from among the three allowed transitions (i.e., down, to the right, and diagonally). The line width and the node size reflect the expected count of patients passing through a transition or state. (**b**) The representation for AD is similar to (**a**) with the strongest transition from each state being coded as follows: $A\beta$ direction (*blue*), hippo (*green*), cog (*red*), $A\beta$+hippo (*cyan*), $A\beta$+cog (*magenta*), hippo+cog (*yellow*), $A\beta$+hippo+ cog(*black*). The node color represents the average sojourn time (*red* to *green*: 0–3 years and above). http://www.cbs.gatech.edu/CT-HMM

Disease (AD). Both are examples of degenerative disorders where the time course of the disease plays an important role in its etiology and treatment. We demonstrate that CT-HMM can yield insight into disease progression, and we compare the timing results for learning across our family of methods.

We begin by describing a 2D CT-HMM for glaucoma progression. Glaucoma is a leading cause of blindness and visual morbidity worldwide [15]. This disease is characterized by a slowly progressing optic neuropathy with associated irreversible structural and functional damage. We use a 2D-grid state space model defined by successive value bands of the two main glaucoma markers, Visual Field Index (VFI) (functional marker) and average RNFL (Retinal Nerve Fiber Layer) thickness (structural marker) with forwarding edges (see Fig. 4a).

Our glaucoma dataset contains 101 glaucomatous eyes from 74 patients followed for an average of 11.7 ± 4.5 years, and each eye has at least five visits (average 7.1 ± 3.1 visits). There were 63 distinct time intervals. The state space is created so that most states have at least five raw measurements mapped to them. All states that are in a direct path between two successive measurements are instantiated, resulting in 105 states.

In Fig. 4a, we visualize the model trained using the entire glaucoma dataset. Several dominant paths can be identified: there is an early stage containing RNFL thinning with intact vision (blue vertical path in the first column). At RNFL range [80, 85], the transition trend reverses and VFI changes become more evident (blue horizontal paths). This L shape in the disease progression supports the finding in [32] that RNFL thickness of around 77 microns is a tipping point at which functional deterioration becomes clinically observable with structural deterioration. Our 2D CT-HMM model reveals the non-linear relationship between structural and functional degeneration, yielding insights into the progression process.

We now demonstrate the use of CT-HMM to visualize the temporal interaction of disease markers of Alzheimer's Disease (AD). AD is an irreversible neuro-degenerative disease that results in a loss of mental function due to the degeneration of brain tissues. An estimated 5.3 million Americans have AD, and there is no known method for the prevention or cure of the condition [29]. It could be beneficial to visualize the relationship between clinical, imaging, and biochemical markers as the pathology evolves, in order to better understand AD progression and develop treatments.

In this experiment, we analyzed the temporal interaction among the three kinds of markers: amyloid beta ($A\beta$) level in cerebral spinal fluid (CSF) (a bio-chemical marker), hippocampus volume (a structural marker), and ADAS cognition score (a functional marker). We obtained the *ADNI* (The *Alzheimer's Disease Neuroimaging Initiative*) dataset from [29].[3] Our sample included patients with mild cognitive impairment (MCI) and AD who had at least two visits with all three markers present, yielding 206 subjects with an average of 2.38 ± 0.66 visits traced in 1.56 ± 0.86 years. The dataset contained three distinct time intervals at 1 month resolution. A 3D gridded state space consisting of 277 states with forwarding links was defined such that for each marker, there were 14 bands that spanned its value range. The procedure for constructing the state space and the definition of the data emission model is the same as in the glaucoma experiment. Following CT-HMM learning, the resulting visualization of Alzheimer's disease in Fig. 4b supports recent findings that a decrease in the A level of CSF (blue lines) is an early marker that precedes detectable hippocampus atrophy (green lines) in cognition-normal elderly [8]. The CT-HMM disease model and its associated visualization can be used as an exploratory tool to gain insights into health dynamics and generate hypotheses for further investigation by biomedical researchers.

Figure 5 gives the average runtime comparison for a single EM iteration between *soft Expm*, *soft Eigen*, and *hard Unif* for both datasets. *Soft Eigen* with our improvements is 26 times faster than *soft Expm* for the glaucoma experiment, and 35 times faster for the AD experiment. *Hard Unif* is slightly slower than *soft Eigen*. We did not include *soft Unif* due to its poor complexity or *hard Eigen* due to its minimal computational benefit in comparison to *soft Eigen*.

[3]Data were obtained from the ADNI database (adni.loni.usc.edu). The ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer's disease (AD). For up-to-date information, see http://www.adni-info.org.

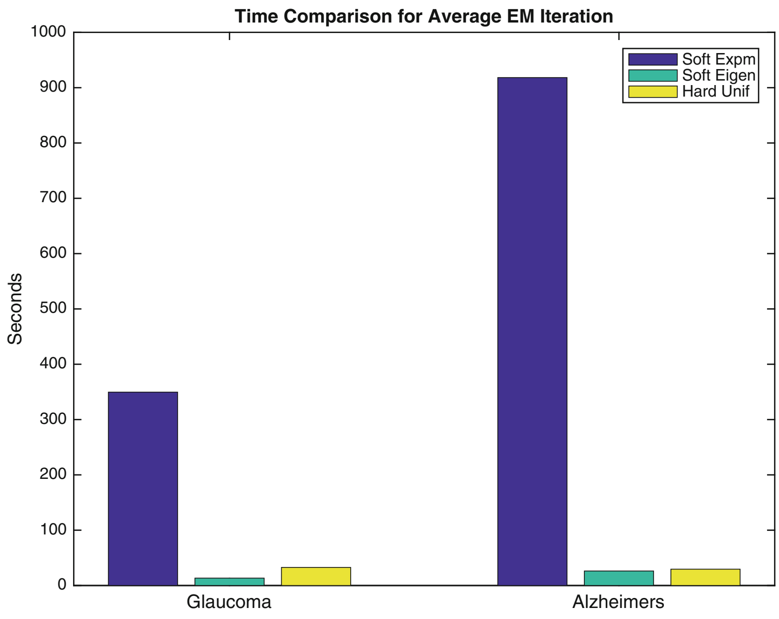

Fig. 5 Time comparison for the average time per iteration between *soft Expm*, *soft Eigen* and *hard Unif* for both experiments. *Soft Eigen* is the fastest method, over an order of magnitude faster than *soft Expm* in both cases. Thus, it should be used unless the eigendecomposition fails, in which case there is a tradeoff between *soft Expm* for accuracy and *hard Unif* for speed

Conclusion

This article introduces novel EM algorithms for CT-HMM learning which leverage recent approaches [12] for evaluating the end-state conditioned expectations in CTMC models. We improve upon the efficiency of the *soft Eigen* method, demonstrating in our experiments a 26–35 times speedup over *Expm*, the next fastest soft method. To our knowledge, we are the first to develop and test the *Expm* and *Unif* methods for CT-HMM learning. We present time complexity analysis for all methods and provide experimental comparisons under both soft and hard EM frameworks. We conclude that *soft Eigen* is the most attractive method overall, based on its speed and its accuracy as a soft method, unless it suffers from an unstable eigendecomposition. We did not encounter significant stability issues in our experiments. We evaluated our EM algorithms on two disease progression datasets for glaucoma and Alzheimer's Disease, and demonstrated that the CT-HMM can provide a novel tool for visualizing the progression of these diseases. The software implementation of our methods is available from our project website.[4]

[4]http://www.cbs.gatech.edu/CT-HMM

In future work, we plan to explore the use of CT-HMMs in modeling event data in a mobile health context, including the analysis of EMA data and moments of high stress or craving identified from mobile sensor data. Other future directions include the combination of event data with regularly-sampled data in a joint model, the incorporation of covariates to model heterogeneous populations, and explicitly incorporating event times into the model. In addition, more work could be done to improve the computational efficiency of the *Expm* and *Unif* methods. As an example, [1] describes potentially more efficient ways to compute *Expm* by noting that the upper right corner of the matrix solution is a Frechét derivative, which has its own Padé approximation. It appears that the Hadamard and trace manipulations we introduced could be applied to this approach as well. Scaling and squaring would cancel much of the benefit, so it would have to be replaced by balancing, which has the same goal of reducing the matrix norm. Additional improvements in efficiency would support the development of large-scale state models.

Acknowledgements Portions of this work were supported in part by NIH R01 EY13178-15 and by grant U54EB020404 awarded by the National Institute of Biomedical Imaging and Bioengineering through funds provided by the Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The research was also supported in part by NSF/NIH BIGDATA 1R01GM108341, ONR N00014-15-1-2340, NSF IIS-1218749, NSF CAREER IIS-1350983, and funding from the Georgia Tech Executive Vice President of Research Office and the Center for Computational Health.

Appendix: Derivation of *Vectorized Eigen*

In [20, 21], it is stated without proof that the naïve *Eigen* is equivalent to *Vectorized Eigen*. Here we present the derivation. Let

$$\tau_{k,l}^{i,j}(t) = \sum_{p=1}^{n} U_{kp}U_{pi}^{-1} \sum_{q=1}^{n} U_{jq}U_{ql}^{-1}\Psi_{pq}(t) \tag{40}$$

where the symmetric matrix $\Psi(t) = [\Psi_{pq}(t)]_{p,q\in S}$ is defined as:

$$\Psi_{pq}(t) = \begin{cases} te^{t\lambda_p} & \text{if } \lambda_p = \lambda_q \\ \frac{e^{t\lambda_p}-e^{t\lambda_q}}{\lambda_p-\lambda_q} & \text{if } \lambda_p \neq \lambda_q \end{cases} \tag{41}$$

Letting $V = U^{-1}$, this is equivalent to

$$\tau_{k,l}^{i,j}(t) = [U[V_i^T U_j \circ \Psi]V]_{kl} \tag{42}$$

To see why, first, note that for the outer product,

$$V_i^T U_j \circ \Psi = \begin{pmatrix} U^{-1}_{1,i} U_{j,1} \Psi_{1,1} & \cdots & U^{-1}_{1,i} U_{j,n} \Psi_{1,n} \\ \vdots & & \vdots \\ U^{-1}_{n,i} U_{j,1} \Psi_{n,1} & \cdots & U^{-1}_{n,i} U_{j,n} \Psi_{n,n} \end{pmatrix} \tag{43}$$

Then

$$U[V_i^T U_j \circ \Psi] = \begin{pmatrix} U_{1,1} & \cdots & U_{1,n} \\ \vdots & & \vdots \\ U_{n,1} & \cdots & U_{n,n} \end{pmatrix} \begin{pmatrix} U^{-1}_{1,i} U_{j,1} \Psi_{1,1} & \cdots & U^{-1}_{1,i} U_{j,n} \Psi_{1,n} \\ \vdots & & \vdots \\ U^{-1}_{n,i} U_{j,1} \Psi_{n,1} & \cdots & U^{-1}_{n,i} U_{j,n} \Psi_{n,n} \end{pmatrix} \tag{44}$$

$$= \begin{pmatrix} \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} U_{j,1} \psi_{p,1} & \cdots & \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} U_{j,n} \psi_{p,n} \\ \vdots & & \vdots \\ \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} U_{j,1} \psi_{p,1} & \cdots & \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} U_{j,n} \psi_{p,n} \end{pmatrix} \tag{45}$$

$$U[V_i^T U_j \circ \Psi]U^{-1} = \begin{pmatrix} \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} U_{j,1} \psi_{p,1} & \cdots & \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} U_{j,n} \psi_{p,n} \\ \vdots & & \vdots \\ \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} U_{j,1} \psi_{p,1} & \cdots & \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} U_{j,n} \psi_{p,n} \end{pmatrix} \cdot \begin{pmatrix} U^{-1}_{1,1} & \cdots & U^{-1}_{1,n} \\ \vdots & & \vdots \\ U^{-1}_{n,1} & \cdots & U^{-1}_{n,n} \end{pmatrix} \tag{46}$$

$$= \begin{pmatrix} \sum_{q=1}^n \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} U_{j,q} U^{-1}_{q,1} \Psi_{p,q} & \cdots & \sum_{q=1}^n \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} U_{j,q} U^{-1}_{q,n} \Psi_{p,q} \\ \vdots & & \vdots \\ \sum_{q=1}^n \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} U_{j,q} U^{-1}_{q,1} \Psi_{p,q} & \cdots & \sum_{q=1}^n \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} U_{j,q} U^{-1}_{q,n} \Psi_{p,q} \end{pmatrix} \tag{47}$$

$$= \begin{pmatrix} \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} \sum_{q=1}^n U_{j,q} U^{-1}_{q,1} \Psi_{p,q} & \cdots & \sum_{p=1}^n U_{1,p} U^{-1}_{p,i} \sum_{q=1}^n U_{j,q} U^{-1}_{q,n} \Psi_{p,q} \\ \vdots & & \vdots \\ \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} \sum_{q=1}^n U_{j,q} U^{-1}_{q,1} \Psi_{p,q} & \cdots & \sum_{p=1}^n U_{n,p} U^{-1}_{p,i} \sum_{q=1}^n U_{j,q} U^{-1}_{q,n} \Psi_{p,q} \end{pmatrix} \tag{48}$$

So that

$$[U[V_i^T U_j \circ \Psi(t)]U^{-1}]_{kl} = \sum_{p=1}^n U_{k,p} U^{-1}_{p,i} \sum_{q=1}^n U_{j,q} U^{-1}_{q,l} \Psi_{p,q}(t) \tag{49}$$

as desired.

References

1. Al-Mohy, A.H., Higham, N.J.: Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation. SIAM Journal on Matrix Analysis and Applications **30**(4), 1639–1657 (2009)
2. Al-Mohy, A.H., Higham, N.J.: Computing the action of the matrix exponential, with an application to exponential integrators. SIAM Journal on Scientific Computing **33**(2), 488–511 (2011)
3. Bartolomeo, N., Trerotoli, P., Serio, G.: Progression of liver cirrhosis to HCC: An application of hidden Markov model. BMC Medical Research Methodology **11**(38) (2011)
4. Bauer, F.L., Fike, C.T.: Norms and exclusion theorems. Numerische Mathematik **2**(1), 137–141 (1960)
5. Bindel, D., Goodman, J.: Principles of scientific computing (2009)
6. Bladt, M., Sørensen, M.: Statistical inference for discretely observed Markov jump processes. J. R. Statist. Soc. B **39**(3), 395–410 (2005)
7. Cox, D.R., Miller, H.D.: The Theory of Stochastic Processes. Chapman and Hall, London (1965)
8. Fagan, A.M., Head, D., Shah, A.R., et. al: Decreased CSF A beta 42 correlates with brain atrophy in cognitively normal elderly. Ann Neurol. **65**(2), 176–183 (2009)
9. Golub, G.H., Van Loan, C.F.: Matrix computations, vol. 3. JHU Press (2012)
10. Higham, N.: Functions of Matrices: Theory and Computation. SIAM Press (2008)
11. Hobolth, A., Jensen, J.L.: Statistical inference in evolutionary models of DNA sequences via the EM algorithm. Statistical Applications in Genetics and Molecular Biology **4**(1) (2005)
12. Hobolth, A., Jensen, J.L.: Summary statistics for endpoint-conditioned continuous-time Markov chains. Journal of Applied Probability **48**(4), 911–924 (2011)
13. Jackson, C.H.: Multi-state models for panel data: The MSM package for R. Journal of Statistical Software **38**(8) (2011)
14. Jensen, A.: Markoff chains as an aid in the study of Markoff processes. Skand. Aktuarietidskr **36**, 87–91 (1953)
15. Kingman, S.: Glaucoma is second leading cause of blindness globally. Bulletin of the World Health Organization **82**(11) (2004)
16. Leiva-Murillo, J.M., Rodrıguez, A.A., Baca-Garcıa, E.: Visualization and prediction of disease interactions with continuous-time hidden Markov models. In: Advances in Neural Information Processing Systems (2011)
17. Liu, Y., Ishikawa, H., Chen, M., et al.: Longitudinal modeling of glaucoma progression using 2-dimensional continuous-time hidden Markov model. In: Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 444–451 (2013)
18. Liu, Y.Y., Li, S., Li, F., Song, L., Rehg, J.M.: Efficient learning of continuous-time hidden Markov models for disease progression. In: Proc. Twenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS 15). Montreal, Canada (2015)
19. McGibbon, R.T., Pande, V.S.: Efficient maximum likelihood parameterization of continuous-time Markov processes. The Journal of Chemical Physics **143**(3), 034,109 (2015)
20. Metzner, P., Horenko, I., Schütte, C.: Generator estimation of Markov jump processes. Journal of Computational Physics **227**, 353–375 (2007)
21. Metzner, P., Horenko, I., Schütte, C.: Generator estimation of Markov jump processes based on incomplete observations nonequidistant in time. Physical Review E **76**(066702) (2007)
22. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix. SIAM Review **20**(4), 801–836 (1978)
23. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Review **45**(1), 3–49 (2003)
24. Nodelman, U., Shelton, C.R., Koller, D.: Expectation maximization and complex duration distributions for continuous time Bayesian networks. In: Proc. Uncertainty in AI (UAI 05) (2005)

25. Osborne, E.: On pre-conditioning of matrices. Journal of the ACM (JACM) **7**(4), 338–345 (1960)
26. Rabinar, L.R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE **77**(2) (1989)
27. Ross, S.M.: Stochastic Processes. John Wiley, New York (1983)
28. Tataru, P., Hobolth, A.: Comparison of methods for calculating conditional expectations of sufficient statistics for continuous time Markov chains. BMC Bioinformatics **12**(465) (2011)
29. The Alzheimer's Disease Neuroimaging Initiative: http://adni.loni.usc.edu
30. Van Loan, C.: Computing integrals involving the matrix exponential. IEEE Trans. Automatic Control **23**, 395–404 (1978)
31. Wang, X., Sontag, D., Wang, F.: Unsupervised learning of disease progression models. Proceeding KDD **4**(1), 85–94 (2014)
32. Wollstein, G., Kagemann, L., Bilonick, R., et al.: Retinal nerve fibre layer and visual function loss in glaucoma: the tipping point. Br J Ophthalmol **96**(1), 47–52 (2012)