JUMP-Means: Small-Variance Asymptotics for Markov Jump Processes

Jonathan H. Huggins* JHUGGINS@MIT.EDU
Karthik Narasimhan* KARTHIKN@MIT.EDU
Ardavan Saeedi* ARDAVANS@MIT.EDU
Vikash K. Mansinghka VKM@MIT.EDU
Computer Science and Artificial Intelligence Laboratory, MIT
*These authors contributed equally and are listed alphabetically.

Abstract

Markov jump processes (MJPs) are used to model a wide range of phenomena from disease progression to RNA path folding. However, maximum likelihood estimation of parametric models leads to degenerate trajectories and inferential performance is poor in nonparametric models. We take a small-variance asymptotics (SVA) approach to overcome these limitations. We derive the small-variance asymptotics for parametric and nonparametric MJPs for both directly observed and hidden state models. In the parametric case we obtain a novel objective function which leads to non-degenerate trajectories. To derive the nonparametric version we introduce the gamma-gamma process, a novel extension to the gamma-exponential process. We propose algorithms for each of these formulations, which we call *JUMP-means*. Our experiments demonstrate that JUMP-means is competitive with or outperforms widely used MJP inference approaches in terms of both speed and reconstruction accuracy.

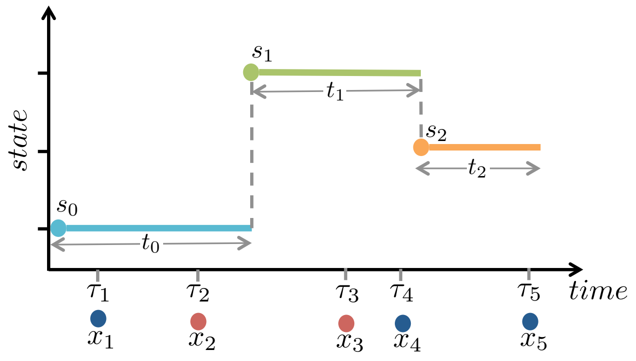
1. Introduction

Markov jump processes (MJPs) are continuous-time, discrete-state Markov processes in which state durations are exponentially distributed according to state-specific rate parameters. A stochastic matrix controls the probability of transitioning between pairs of states. MJPs have been used to construct probabilistic models either when the state of a system is observed directly, such as with disease progression (Mandel, 2010) and RNA path folding (Hajiaghayi et al., 2014), or when the state is only observed indirectly, as in corporate bond rating (Bladt & Sørensen, 2009). For example, consider the important clinical task of analyzing physiological signals of a patient in order to detect abnormalities. Such signals include heart rate, blood pressure, respiration, and blood oxygen level. For an ICU patient, an abnormal state might be the precursor to a cardiac arrest event while for an epileptic, the state might presage a seizure (Goldberger et al., 2000). How can the latent state of the patient be inferred by a Bayesian modeler, so that, for example, an attending nurse can be notified when a patient enters an abnormal state? MJPs offer one attractive approach to analyzing such physiological signals.

Applying an MJP model to physiological signals presents a challenge: the number of states is unknown and must be inferred using, for example, Bayesian nonparametric methods. However, efficient inference in nonparametric MJP models is a challenging problem, where existing methods based on particle MCMC scale poorly and mix slowly (Saeedi & Bouchard-Côté, 2011). Current optimization-based methods such as expectation maximization (EM) are inapplicable if the state size is countably infinite; hence, they cannot be applied to Bayesian nonparametric MJP models, as we would like to do for physiological signals.

Furthermore, although MJPs are viewed as more realistic than their discrete-time counterparts in many fields (Rao & Teh, 2013), degenerate solutions for the maximum likelihood (ML) trajectories for both directly and indirectly observed cases (Perkins, 2009), and non-existence of the ML transition matrix (obtained from EM) for some indirectly observed cases (Bladt & Sørensen, 2009) present inferential challenges. Degenerate ML trajectories occur when some of the jump times are infinitesimal, which severely undermines the practicality of such approaches. For instance, a trajectory which predicts a patient's seizure for an

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).



Notation
M: number of states
π: initial state distribution
P: state transition matrix, with entries $p_{ss'}$
λ_s: transition rate for state s
$\mathcal{U} = (s_0, t_0, s_1, t_1, \dots, s_{K-1}, t_{K-1}, s_K)$: MJP trajectory
$\mathcal{S}$: the states corresponding to $\mathcal{U}$
$\mathcal{T}$: the times corresponding to $\mathcal{U}$
$\mathcal{O} = \{(\tilde{t}_i, \tilde{s}_i)\}$: observation times and states of DOMJP*
$\boldsymbol{\tau} = (\tau_1, \dots, \tau_L)$: observation times of HMJP
$\mathcal{X} = (x_1, \dots, x_L)$: observations of HMJP
ρ_{sn}: probability of observing $x_\ell = n$ when in state s

Figure 1. **Left:** Illustrative example for an HMJP (Section 3.2) with three hidden states ($M = 3$) and two possible observation values ($N = 2$). The observations $\mathcal{X}$, their times $\boldsymbol{\tau}$, an (arbitrary) sample MJP trajectory $\mathcal{U} = (s_0, t_0, s_1, t_1, s_2, t_2)$. **Right:** Notation used for parametric MJPs. *DOMJP = directly observed MJP.

infinitesimal amount of time is of limited use to the medical staff. Fig. 3 shows an example of the degeneracy problem.

In this paper, we take a *small-variance asymptotics* (SVA) approach to develop an optimization-based framework for efficiently estimating the most probable trajectories (states) for both parametric and nonparametric MJP-based models. Small-variance asymptotics has recently proven to be useful in estimating the parameters and inferring the latent states in rich probabilistic models. SVA extends the well-known connection between mixtures of Gaussians and k-means: as the variances of the Gaussians approach zero, the *maximum a posteriori* solution to the mixture of Gaussians model degenerates to k-means solution (Kulis & Jordan, 2012). The same idea can be applied to obtain well-motivated objective functions that correspond to a latent variable model for which scalable inference via standard methods like MCMC is challenging. SVA has been applied to (hierarchical) Dirichlet process mixture models (Kulis & Jordan, 2012; Jiang et al., 2012), Bayesian nonparametric latent feature models (Broderick et al., 2013), hidden Markov models (HMMs), and infinite-state HMMs (Roychowdhury et al., 2013).

We apply the SVA approach to both parametric and Bayesian nonparametric MJP models to obtain what we call the *JUMP-means* objective functions. In the parametric case, we derive a novel objective function which does not suffer from maximum likelihood's solution degeneracy, leading to more stable and robust inference procedures in both the directly observed and hidden state cases. Infinite-state MJPs (iMJPs) are constructed from the hierarchical gamma-exponential process (HΓEP) (Saeedi & Bouchard-Côté, 2011). In order to apply SVA to iMJPs, we generalize the HΓEP to obtain the first deterministic procedure (we know of) for inference in Bayesian nonparametric MJPs.

We evaluate JUMP-means on several synthetic and real-world datasets in both the parametric and Bayesian nonparametric cases. JUMP-means performs on par with or better than existing methods, offering an attractive speed-accuracy tradeoff. We obtain significant improvements in the non-parametric case, gaining up to a 20% reduction in mean error on the task of observation reconstruction. In summary, the JUMP-means approach leads to algorithms that 1) are applicable to MJPs with Bayesian nonparametric priors; 2) provide non-degenerate solutions for the most probable trajectories; and 3) are comparable to or outperform other standard methods of inference both in terms of speed and reconstruction accuracy.

2. Background

2.1. Markov Jump Processes

A *Markov jump process* (MJP) is defined by (a) a finite (or countable) state space, which we identify with the integers $[M] \triangleq \{1, \dots, M\}$; (b) an initial state probability distribution π; (c) a (stochastic) state transition matrix P with $p_{ss} = 0$ for all $s \in [M]$; and (d) a state dwell-time rate vector $\boldsymbol{\lambda} \triangleq (\lambda_1, \dots, \lambda_M)$. The process begins in a state $s_0 \sim \pi$. When the process enters a state s, it remains there for a dwell time that is exponentially distributed with parameter λ_s. When the system leaves state s, it transitions to state $s' \neq s$ with probability $p_{ss'}$.

A *trajectory* of the MJP is a sequence of states and a dwell time for each state, except for the final state: $\mathcal{U} \triangleq \mathcal{U}_T \triangleq (s_0, t_0, s_1, t_1, \dots, s_{K-1}, t_{K-1}, s_K)$. Implicitly, K (and thus $\mathcal{U}$) is a random variable such that $t_{K-1} < T$ and the system is in state s_K at time T. Let $\mathcal{S} \triangleq \mathcal{S}_T \triangleq (s_0, s_1, \dots, s_K)$ and $\mathcal{T} \triangleq \mathcal{T}_T \triangleq (t_0, t_1, \dots, t_{K-1})$ be the sequences of states and times corresponding to $\mathcal{U}$. The probability of a trajectory is given by

$$p(\mathcal{U} \mid \pi, P, \boldsymbol{\lambda}) = \mathbb{1}[t_. \leq T] e^{-\lambda_{s_K}(T - t_.)} \pi_{s_0} \times \prod_{k=1}^{K} \lambda_{s_{k-1}} e^{-\lambda_{s_{k-1}} t_{k-1}} p_{s_{k-1} s_k}, \tag{1}$$

where $t_\cdot \triangleq \sum_{k=0}^{K-1} t_k$ and $\mathbb{1}[\cdot]$ is the indicator function. In many cases when the states are directly observed, the initial state and the final state are observed, in which case it is straightforward to obtain a likelihood from (1).

A hidden state MJP (HMJP) is an MJP in which the states are observed indirectly according to a likelihood model $p(x \mid s)$, $s \in [M]$, $x \in X$, where X is some observation space. The times of the observations $\boldsymbol{\tau} = (\tau_1, \dots, \tau_L)$ are chosen independent of $\mathcal{U}$, so the probability of the observations $\mathcal{X} \triangleq (x_1, \dots, x_L)$ is given by $p(\mathcal{X} \mid \mathcal{U}, \boldsymbol{\tau}) = \prod_{\ell=1}^{L} p(x_\ell \mid s_{\tau_\ell})$, where, with an abuse of notation, we write s_τ for the state of the MJP at time τ.

2.2. Previous Approaches to MJP Inference

There are a number of existing approaches to inference and learning in MJPs. An expectation-maximization (EM) algorithm can be derived, but it cannot be applied to models with countably infinite states, so it is not suitable for iMJPs (Lange, 2014) (iMJPs are detailed in Section 4). Moreover, with discretely observed data, the maximum-likelihood estimate with finite entries for the transition matrix obtained from EM may not exist (Bladt & Sørensen, 2005).

Maximum likelihood inference amounts to finding $\max_{\mathcal{U}} \ln p(\mathcal{U} \mid \pi, P, \boldsymbol{\lambda})$, which can be carried out efficiently using dynamic programming (Perkins, 2009). However, maximum likelihood solutions for the trajectory are degenerate: only an infinitesimal amount of time is spent in each state, except for the state visited with the smallest rate parameter (i.e., longest expected dwell time). Such a solution is unsatisfying and unintuitive because the dwell times are far from their expected values. Thus, maximum likelihood inference produces results that are unrepresentative of the model behavior.

Markov chain Monte Carlo methods have also been developed, but these can be slow and their convergence is often difficult to diagnose (Rao & Teh, 2013). Recently, a more efficient Monte Carlo method was proposed in Hajiaghayi et al. (2014) which is based on particle MCMC (PMCMC) methods (Andrieu et al., 2010). This approach addresses the issue of efficiency, but since it marginalizes over the jump points, it cannot provide probable trajectories.

2.3. Small-variance Asymptotics

Consider a Bayesian model $p(D \mid Z, \theta, \sigma^2)p(Z, \theta)$ in which the likelihood terms contain a variance parameter σ^2. Given some data D, a point estimate for the parameters θ and latent variables Z of the model can obtained by maximizing the posterior $p(Z, \theta \mid D, \sigma^2) \propto p(D \mid Z, \theta, \sigma)p(Z, \theta)$, resulting in a *maximum a posteriori* (MAP) estimate. In the SVA approach (Broderick et al., 2013), the MAP optimization is considered in the limit as the likelihood variance parameter is taken to zero: $\sigma^2 \to 0$. Typically, the small-variance limit leads to a much simpler optimization than the MAP optimization with non-zero variance. For example, the MAP objective for a Gaussian mixture model simplifies to the k-means objective.

3. Parametric MJPs

3.1. Directly Observed MJP

Consider the task of inferring likely state/dwell-time sequences given $\mathcal{O} = \{(\tilde{t}_i, \tilde{s}_i)\}_{i=1}^{I}$, the times at which the system was directly observed and the states of the system at those times. For simplicity we assume that $\tilde{t}_0 = 0$ and that all times are in the interval $[0, T]$. Let $s(\mathcal{U}, t)$ be the state of the system following trajectory $\mathcal{U}$ at time t. The likelihood of a sequence is

$$\ell(\mathcal{U} \mid \mathcal{O}, P, \boldsymbol{\lambda}) = \mathbb{1}[t_\cdot \le T] \prod_{i=1}^{I} \mathbb{1}[s(\mathcal{U}, \tilde{t}_i) = \tilde{s}_i] \\ \times \left(\prod_{k=1}^{K} \lambda_{s_{k-1}} e^{-\lambda_{s_{k-1}} t_{k-1}} p_{s_{k-1} s_k} \right) e^{-\lambda_{s_K}(T - t_\cdot)} \tag{2}$$

We also place a gamma prior on the rate parameters $\boldsymbol{\lambda}$ (detailed below). Instead of relying on MAP estimation, we apply a small variance asymptotics analysis to obtain a more stable objective function. Following (Jiang et al., 2012), we scale the distributions by an inverse variance parameter β and then maximize the scaled likelihood and prior in the limit $\beta \to \infty$ (i.e., as the variance goes to zero).

Scaling the exponential distribution $f(t; \lambda) = \lambda \exp(-\lambda t)$ produces the two-parameter family with

$$\ln f(t; \lambda, \beta) = \\ -\beta \left(\lambda t - \ln t - \ln \lambda - \frac{\beta \ln \beta - \ln \Gamma(\beta)}{\beta} + \frac{\ln t}{\beta} \right),$$

which is the density of a gamma distribution with shape parameter β and rate parameter $\beta\lambda$. Hence, the mean of the scaled distribution is $\frac{1}{\lambda}$ and its variance is $\frac{1}{\lambda^2 \beta}$. Letting $F(t; \lambda, \beta)$ denote the CDF corresponding to $f(t; \lambda, \beta)$, we have $1 - F(t; \lambda, \beta) = \frac{\Gamma(\beta, \beta\lambda t)}{\Gamma(\beta)}$, where $\Gamma(\cdot, \cdot)$ is the upper incomplete gamma function.

The multinomial distribution is scaled by the parameter $\hat{\beta} \triangleq \xi\beta$. Writing the likelihood with the scaled exponential families (and dropping indicator variables) yields:

$$\ell(\mathcal{U} \mid \mathcal{O}, P, \boldsymbol{\lambda}) \\ \propto \exp\left\{ -\beta \left(\frac{\ln \Gamma(\beta) - \ln \Gamma(\beta, \beta \lambda_{s_K} t_\cdot)}{\beta} \right.\right. \\ \left.\left. + \sum_{k=0}^{K-1} \left(\xi \ln p_{s_k s_{k+1}} + \lambda_{s_k} t_k - \ln \lambda_{s_k} t_k \right) \right.\right. \tag{3}$$

$$+ \sum_{k=0}^{K-1} \left(-\frac{\beta \ln \beta - \ln \Gamma(\beta)}{\beta} + \frac{\ln t_k}{\beta} \right) \Bigg) \Bigg\} .$$

The modified likelihood is for a jump process which is no longer Markov when $\beta \neq 1$. We also place a $\mathsf{Gam}(\alpha_\lambda, \alpha_\lambda \mu_\lambda)$ prior on each λ_i and set $\alpha_\lambda = \xi_\lambda \beta$. It can be shown (see the Supplementary Material for details) that, when $\beta \to \infty$, the MAP estimation problem becomes

$$\begin{aligned} \min_{\mathcal{U}, \boldsymbol{\lambda}, P} \Bigg\{ & \xi \sum_{k=0}^{K-1} \ln p_{s_k s_{k+1}} + \sum_{k=0}^{K-1} (\lambda_{s_k} t_k - \ln \lambda_{s_k} t_k - 1) \\ & + \mathbb{1}[\lambda_{s_K} t_\cdot \geq 1](\lambda_{s_K} t_\cdot - \ln \lambda_{s_K} t_\cdot - 1) \\ & + \xi_\lambda \sum_{s=1}^{M} (\mu_\lambda \lambda_s - \ln \lambda_s - 1) \Bigg\}. \end{aligned} \tag{4}$$

The optimization problem (4) is very natural and offers far greater stability than maximum likelihood optimization. As with maximum likelihood, the $\ln p_{s_k s_{k+1}}$ terms penalize transitions with small probability. The term $h(t_k) \triangleq \lambda_{s_k} t_k - \ln \lambda_{s_k} t_k - 1$ is convex and minimized when $t_k = 1/\lambda_{s_k}$, the expected value of the dwell time for state s_k. As $t_k \to 0$, $h(t_k)$ approaches ∞, while for $t_k \gg 1/\lambda_{s_k}$, $h(t_k)$ grows approximately linearly. Thus, times very close to zero are heavily penalized while times close to the expected dwell time are penalized very little. The term $\mathbb{1}[\lambda_{s_K} t_\cdot \geq 1](\lambda_{s_K} t_\cdot - \ln \lambda_{s_K} t_\cdot - 1)$ penalizes the time $t_\cdot$ spent in state s_K so far in the same manner as a regular dwell time when $t_\cdot$ is *greater* than the expected value of the dwell-time. However, when $t_\cdot$ is *less* than the expected value there is no cost, which is quite natural since the system may remain in state s_K for longer than $t_\cdot$ — i.e., there should not be a large penalty for $t_\cdot$ being less than its expected value. Finally, parameters ξ_λ and μ_λ have a very natural interpretation (cf. (8) below): they correspond to *a priori* having ξ_λ dwell times of length μ_λ for each state.

Comparison to Maximum Likelihood MJP trajectories estimated using maximum likelihood (MLE) are usually trivial, with the system spending almost all its time in a single state (with the smallest λ), with infinitesimal dwell times for the other states. This poor behavior of MLE is due to the fact that the mode of $\mathsf{Exp}(\lambda)$, which is favored by the MLE, is 0, even though the mean is $1/\lambda$.[1] The SVA optimization, on the other hand, does give trajectories that are representative of the true behavior because the SVA terms of the form $\lambda t - \ln(\lambda t) - 1$ are optimized at $1/\lambda$ (i.e., at the mean of $\mathsf{Exp}(\lambda)$). We demonstrate the superior behavior of the SVA in the concrete example of estimating disease progression in patients in Section 5.

[1]Note that placing priors on the rate parameters, as we do, does not affect the degeneracy of the ML trajectory.

3.2. Hidden State MJP

For an HMJP, the likelihood of a valid trajectory is

$$\begin{aligned} p(\mathcal{U} \mid \mathcal{X}, \boldsymbol{\tau}, P, \boldsymbol{\lambda}) = & \left(\prod_{\ell=1}^{L} p(x_\ell \mid s_{\tau_\ell}) \right) \\ & \times \left(\prod_{k=1}^{K} \lambda_{s_{k-1}} e^{-\lambda_{s_{k-1}} t_{k-1}} p_{s_{k-1} s_k} \right) e^{-\lambda_{s_K}(T - t_\cdot)}. \end{aligned} \tag{5}$$

Hence, the only difference between the directly observed case and the HMJP is the addition of the observation likelihood terms. Because multinomial observations are commonly used in MJP applications, that is the case we consider here. Let N denote the number of possible observations and ρ_{sn} be the probability of observing $x_\ell = n$ when $s_{\tau_\ell} = s$. The observation likelihoods are scaled in the same manner as the transition probabilities, but with $\hat{\beta} = \zeta \beta$. Thus, for the HMJP, we obtain:

$$\begin{aligned} \min_{\mathcal{U}, \boldsymbol{\lambda}, P, \rho} \Bigg\{ & \zeta \sum_{\ell=1}^{L} \ln \rho_{s_{\tau_\ell} x_\ell} + \xi \sum_{k=0}^{K-1} \ln p_{s_k s_{k+1}} \\ & + \sum_{k=0}^{K-1} (\lambda_{s_k} t_k - \ln \lambda_{s_k} t_k - 1) \\ & + \mathbb{1}[\lambda_{s_K} t_\cdot \geq 1](\lambda_{s_K} t_\cdot - \ln \lambda_{s_K} t_\cdot - 1) \\ & + \xi_\lambda \sum_{s=1}^{M} (\mu_\lambda \lambda_s - \ln \lambda_s - 1) \Bigg\}. \end{aligned} \tag{6}$$

3.3. Algorithm

Optimizing the JUMP-means objectives in (4) and (6) is non-trivial due to the fact that we do not know the number of jumps in the MJP, and the combinatorial explosion in the sequences with the number of jump points. The terms involving the continuous variables t_k (dwell times) present an additional complexity.

We therefore resort to an alternating minimization procedure to optimize the JUMP-means objective function, similar in spirit to the one used in Roychowdhury et al. (2013). In each iteration of the optimization process, we first use a modified Viterbi algorithm to obtain the most likely state sequence. Then, we use convex optimization to distribute the jump points optimally with respect to the values from $\boldsymbol{\lambda}$ for the current state sequence.

Directly Observed MJP When optimizing (4), there may be many sequences ($\mathcal{O}$'s) available, representing distinct realizations of the process. We use the following algorithm to optimize (4):

1. Initialize the state transition matrix P and rate vector $\boldsymbol{\lambda}$ with uniform values.

2. For every observation sequence $\mathcal{O}$, instantiate the jump points by adding one jump point between every pair of observations, in addition to the start and end points.
3. For each $\mathcal{O}$, use a modified Viterbi algorithm. to find the best state sequence to optimize (4), while keeping the jump points fixed. The modified algorithm includes the dwell time penalty terms, which are dependent upon the assignment of states to the time points.
4. Optimize the dwell times t_k with the state sequences of the trajectories fixed.
5. Optimize P and $\boldsymbol{\lambda}$ with the other variables fixed. The optimal values can be obtained in closed form. For example, if there is only a single observation sequence $\mathcal{O}$ with corresponding inferred trajectory $\mathcal{S}$, then

$$p_{mj} = \frac{n_{mj}}{\sum_{j=1}^{M} n_{mj}}, \quad m, j \in [M] \tag{7}$$

$$\lambda_m = \frac{\xi_\lambda + \sum_k \mathbb{1}[s_k = m]}{\xi_\lambda \mu_\lambda + \sum_k \mathbb{1}[s_k = m] t_k}, \tag{8}$$

   where n_{mj} denotes to the number of transitions from state m to state j in $\mathcal{S}$.
6. Repeat steps 3-5 until convergence.

Beam Search Variant We note that the optimization procedure just described is restrictive since the number of jump points is fixed and the jump points are constrained by the observation boundaries. To eliminate this, we also tested a beam search variant of the algorithm to allow for the creation and removal of jump points, but found it did not have much impact in our experiments.

Hidden State MJP The algorithm to optimize the hidden state MJP JUMP-means objective (6) is similar to that for optimizing (4), but with three modifications. First, in place of $\mathcal{O}$, we have the indirect observations of the states $\mathcal{X}$. Second, observation likelihood terms containing $\boldsymbol{\rho}$ are included in the objective minimized by the Viterbi optimization (step 3). Finally, an additional update is performed in step 5 for each of the observation distributions $\boldsymbol{\rho}_m$:

$$\rho_{mn} = \frac{\sum_\ell \mathbb{1}[s_{\tau_\ell} = m]\mathbb{1}[x_\ell = n]}{\sum_\ell \mathbb{1}[s_{\tau_\ell} = m]} \tag{9}$$

for $m \in [M]$ and $n \in [N]$. If each $\boldsymbol{\rho}_m \triangleq (\rho_{m1}, \dots, \rho_{mN})$ is initialized to be uniform, then the algorithm converges to a poor local minimum, so we add a small amount of random noise to each uniform $\boldsymbol{\rho}_m$.

4. Bayesian Nonparametric MJPs

We now consider the Bayesian nonparametric MJP (iMJP) model. The iMJP is based on the hierarchical gamma-exponential process (HΓEP), which is constructed from the gamma-exponential process (ΓEP). We denote the Moran gamma process with base measure H and rate parameter γ by $\Gamma\mathrm{P}(H, \gamma)$ (Kingman, 1993). The HΓEP generates a state/dwell-time sequence $s_0, t_1, s_1, t_2, s_2, t_3, s_3, \dots$ (with s_0 assumed known) according to (Saeedi & Bouchard-Côté, 2011):

$$\mu_0 \sim \Gamma\mathrm{P}(\alpha_0 H_0, \gamma_0), \tag{10}$$

$$\mu_i \,|\, \mu_0 \overset{\text{i.i.d.}}{\sim} \Gamma\mathrm{P}(\mu_0, \gamma), \quad i = 1, 2, \dots, \tag{11}$$

$$s_k \,|\, \{\mu_i\}_{i=0}^{\infty}, \mathcal{U}_{k-1} \sim \bar{\mu}_{s_{k-1}}, \tag{12}$$

$$t_k \,|\, \{\mu_i\}_{i=0}^{\infty}, \mathcal{U}_{k-1} \sim \mathsf{Exp}(\|\mu_{s_{k-1}}\|), \tag{13}$$

where H_0 is the base probability measure, α_0 is a concentration parameter, $\mathcal{U}_k \triangleq (s_0, t_1, s_1, t_2, s_2, \dots, t_{k-1}, s_k)$, $\bar{\mu}_i \triangleq \mu_i / \|\mu_i\|$, and $\|\mu\|$ denotes the total mass of the measure μ. As in the parametric case, we must replace the exponential distribution in (13) with the scaled exponential distribution. After an appropriate scaling of the rest of the hyperparameters, we obtain the hierarchical gamma-gamma process (HΓΓP). The definition and properties of the HΓΓP are given in the Supplementary Material.

Let M denote the number of used states, K_m the number of transitions out of state m, and μ_{ij} the mass on the j-th component of the measure μ_i. For $0 \le i \le M, 1 \le j \le M$, let $\bar{\pi}_{ij} \triangleq \bar{\mu}_{ij}$ and for $0 \le i \le M$, let $\bar{\pi}_{i,M+1} \triangleq 1 - \sum_{j=1}^{M} \bar{\mu}_{ij}$. Let $\boldsymbol{t}_m^* \triangleq (t_{m1}^*, \dots, t_{mK_m}^*)$ be the waiting times following state m and define $t_{m\cdot}^* \triangleq \sum_{j=1}^{K_m} t_{mj}^*$. In order to retain the effects of the hyperparameters in the asymptotics, set $\alpha_0 = \exp(-\xi_1 \beta)$ and $\gamma_0 = \kappa_0 = \xi_2$. It can then be shown that (see the Supplementary Material for details), when $\beta \to \infty$, the iMJP MAP estimation problem becomes

$$\begin{aligned} \min_{K, \mathcal{U}_K, \bar{\pi}, \boldsymbol{\rho}} \quad & \zeta \sum_{\ell=1}^{L} \ln \rho_{s_{\tau_\ell} x_\ell} + \xi \sum_{k=1}^{K} \ln \bar{\pi}_{s_{k-1} s_k} \\ & + \xi_1 M + \sum_{m=1}^{M} \xi_2 \mathrm{KL}(\bar{\pi}_0 || \bar{\pi}_m) \\ & - \sum_{m=1}^{M} \left\{ \textstyle\sum_{j=1}^{K_m} \ln t_{mj}^* - K_m \ln\left([\gamma + t_{m\cdot}^*]/K_m\right) \right\}. \end{aligned} \tag{14}$$

Like its parametric counterpart, the Bayesian nonparametric cost function penalizes dwell durations very close to zero via the $\ln t_{mj}^*$ terms. In addition, there are penalties for the number of states and the state transitions. The observation likelihood term in (14) favors the creation of new states to minimize the JUMP-means objective, while the state penalty $\xi_1 M$ and the non-linear penalty term $K_m \ln\left([\gamma + t_{m\cdot}^*]/K_m\right)$ counteracts the formation of a long tail of states with very few data points. The γ hyperparameter introduces an additional, nonlinear cost for each additional state — if a state is occupied for $\Omega(\gamma)$ time, then the γ term for that state does not have much effect on the cost.

Table 1. Statistics and Mean observation reconstruction error for the various models on different datasets. **Key:** BL = Baseline; P = parametric; SVA = JUMP-means; NP = nonparametric; DO = directly observed; H = hidden; MS = multiple sclerosis data set; MIMIC = blood pressure data set. *Best result obtained by running EM with various number of hidden states (up to 12).

	Data Set			Mean Error (%)				
	# Points	Held Out	# States	BL	EM	MCMC	SVA	PMCMC
Synthetic 1 (P-DO)	10,000	30 %	10	69.7	**40.2**	**41.9**	**41.2**	-
Synthetic 2 (P-H)	10,000	30 %	5	51.8	**42.9**	74.6	46.5	-
MS (P-DO)	390	50 %	3	51.2	**26.2**	48.1	**25.4**	-
MIMIC (NP-H)	2,208	25 %	-	42.3	25.7*	-	**24.3**	30.9

The KL divergence terms between $\bar{\pi}_0$ and $\bar{\pi}_m$ arise from the hierarchical structure of the prior, biasing the transition probabilities $\bar{\pi}_m$ to be similar to the prior $\bar{\pi}_0$.

4.1. Algorithm

For the iMJP case, we have the extra variables M and $\{\bar{\pi}_m\}_{m=0}^M$ to optimize. In addition, the number of variables to optimize depends on the number of states in our model. The major change in the algorithm from the parametric case is that we must propose and then accept or reject the addition of new states. We propose the following algorithm for optimizing the iMJP:

(1) Initialize $\boldsymbol{\rho}$, $\bar{\pi}_0$ and $\bar{\pi}_1$ with uniform values and set the number of states $M = 1$.

(2) For each observation sequence, apply the Viterbi algorithm and update the times using the new objective function in (14), analogously to steps (3) and (4) in the parametric algorithm.

(3) Perform MAP updates for $\boldsymbol{\rho}$ (as in (9)) and $\bar{\pi}$:

$$\bar{\pi}_{mj} = \frac{\xi n_{mj} + \xi_2 \bar{\pi}_{0j}}{\xi \sum_{j=1}^{M} n_{mj} + \xi_2 \bar{\pi}_{0j}}, \quad m, j \in [M] \tag{15}$$

$$\bar{\pi}_{0j} \propto \prod_{m=1}^{M} \bar{\pi}_{mj}^{1/M}, \quad j \in [M]. \tag{16}$$

(4) For every state pair $m, m' \in [M]$, form a new state $M+1$ by considering all transitions from m to m' and reassigning all observations x_ℓ that were assigned to m' to the new state. Update $\bar{\pi}$ and $\boldsymbol{\rho}$ to estimate the overall objective function for every new set of $M+1$ states formed in this way and accept the state set that minimizes the objective. If no such set exists, do not create a new state and revert back to the old $\bar{\pi}$ and $\boldsymbol{\rho}$.

(5) Repeat steps 2-4 until convergence.

Remark. If instead of multinomial observations we have Gaussian observations, the parameter $\boldsymbol{\rho}_s$ is replaced with the mean parameter μ_s. In this case, we update the mean for each state using the data points assigned to the state, similar to the procedure for k-means clustering (see, e.g., Jiang et al. (2012); Roychowdhury et al. (2013)).

5. Experiments

In this section we provide a quantitative analysis of the JUMP-means algorithm and compare its performance on synthetic and real datasets with standard inference methods for MJPs. For evaluation, we consider multiple sequences of discretely observed data and randomly hold out a subset of the data. We report reconstruction error for performance comparison.

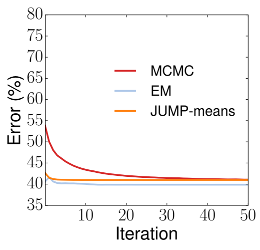
5.1. Parametric Models

For the parametric models, we compare JUMP-means to maximum likelihood estimation of the MJP parameters learned by EM (Asger & Ledet, 2005), the MCMC method proposed in Rao & Teh (2013) and a simple baseline where we ignore the sequential structure of the data. We run three sets of experiments (2 synthetic, 1 real) for our evaluation.

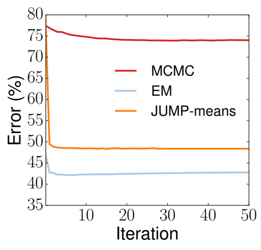
Synthetic 1: Directly Observed States For evaluating the model on a directly observed process, we generate 100 different datasets randomly from various MJPs with 10 states. To generate each dataset, we first generate the rows of the transition probability matrix and transition rates independently from $\mathsf{Dir}(1)$ and $\mathsf{Gam}(1,1)$, respectively. Next, given the rates and transition probabilities for each dataset, we sample 500 sequences of length 20. We hold out 30% of the observations at random for testing reconstruction error.

We run JUMP-means by initializing the algorithm with a uniform transition matrix P and set the rate vector $\boldsymbol{\lambda}$ to be all ones. We run 300 iterations of the algorithm described in Section 3.3; each iteration is one scan through all the sequences. We set the hyperparameters ξ, ξ_λ, and μ_λ equal to 1, 1, and .5, respectively. For MCMC, we initialize the jump points using the time points of the observations. We place independent $\mathsf{Dir}(1)$ priors on P and independent $\mathsf{Gam}(1,1)$ priors on $\boldsymbol{\lambda}$. We initialize EM with a uniform P and an all-ones $\boldsymbol{\lambda}$. We run both MCMC and EM for 300 iterations, then reconstruct observations using the Bayes estimator approximated from the 300 posterior samples. For our baseline we use the most common observation in the dataset as an estimate of the missing observations.

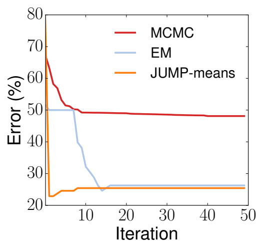
Table 1 gives the mean reconstruction error across sequences for the various methods. Note that JUMP-means performs better than MCMC, and is almost on par with EM.



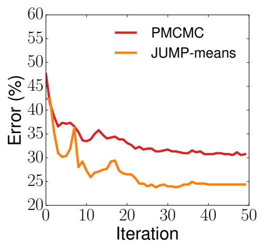
(a)



(b)



(c)



(d)

Figure 2. Mean error vs iterations for **(a)** Synthetic 1; **(b)** Synthetic 2; **(c)** MS; and **(d)** MIMIC datasets. In each case the JUMP-means algorithms have better or comparable performance to other standard methods of inference in MJPs. Mean error vs CPU runtime plots can be found in the Supplementary Material.

Fig. 2(a) shows the average error across all the datasets for each method versus number of iterations. In terms of CPU time, each iteration of JUMP-means (Java), EM (Java), and MCMC (Python) takes 0.3, 1.61 and 42 seconds, respectively. We also ran experiments with the beam search variant described in Section 3.3; however, we did not obtain any significant improvement in results.

Synthetic 2: Hidden States For the hidden state case, we generate 100 different datasets for MJPs with 5 hidden and 5 observed states, with varying parameters as above. In each dataset there are 500 sequences of length 20. In addition to parameters in the directly observed case, we generate observation likelihood terms for each state from $\mathrm{Dir}(1)$.

We initialize the transition probabilities and the rate vectors for JUMP-means, MCMC and EM in a fashion similar to the directly observed case. For the observation likelihood $\boldsymbol{\rho}$, we use $\mathrm{Dir}(1)$ as a prior for MCMC, uniform distributions for EM initialization and a uniform probability matrix with a small amount of random noise for JUMP-means initialization. We set $\xi, \xi_\lambda, \mu_\lambda$ as before and ξ to 1.

We run each algorithm for 300 iterations. For JUMP-means, we use the hidden state MJP algorithm described in Section 3.3. Table 1 and Fig. 2(b) again demonstrate that JUMP-means outperforms MCMC by a large margin and performs comparably to EM. The poor performance of MCMC is due to slow mixing over the parameters and state trajectories. The slow mixing is a result of the coupling between the latent states and the observations, which is induced by the observation likelihood.

Disease Progression in Multiple Sclerosis (MS) Estimating disease progression and change points in patients with Multiple Sclerosis (MS) is an active research area (see, e.g., Mandel (2010)). We can cast the progression of the disease in a single patient as an MJP, with different states representing the various stages of the disease. Obtaining the most-likely trajectory for this MJP can aid in understanding the disease progression and enable better care.

For our experiments, we use a real-world dataset collected from a phase III clinical trial of a drug for MS. This dataset

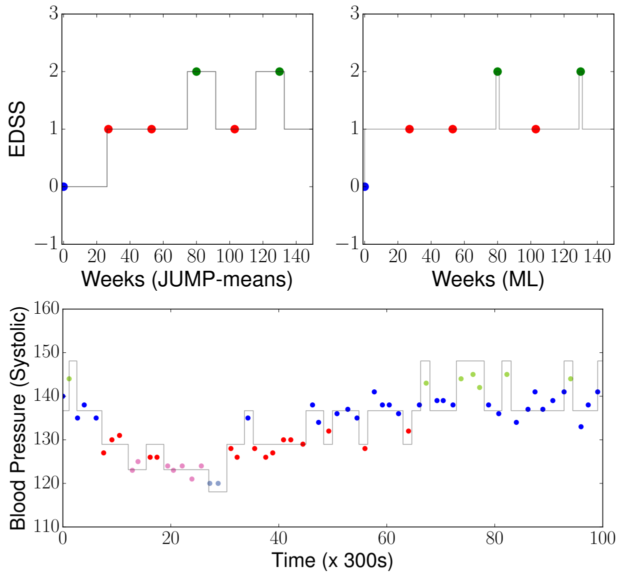


Figure 3. **Top:** Latent trajectories inferred by JUMP-means and ML estimate for a patient in the MS dataset. **Bottom:** Latent trajectory inferred by JUMP-means for a patient in MIMIC dataset.

tracks the progression of the disease for 72 different patients over three years. We randomly hold out 50% of the observations and evaluate on the observation reconstruction task. The observations are values of a disability measure known as EDSS, recorded at different time points. Initialization and hyperparameters are the same as Synthetic 1.

Table 1 shows that JUMP-means significantly outperforms MCMC, achieving almost a 50% relative reduction in reconstruction error. JUMP-means again achieves comparable results with EM. Fig. 3 (top panel) provides an example of the latent trajectories from JUMP-means and maximum likelihood estimate for a single patient. The MLE trajectory includes two infinitesimal dwell times, which do not reflect realistic behavior of the system (since we do not expect a patient to be in a disease state for an infinitesimal amount of time). On the other hand, the trajectory produced by JUMP-means takes into account the dwell times of the various stages of the disease and provides a more reasonable picture of its progression.

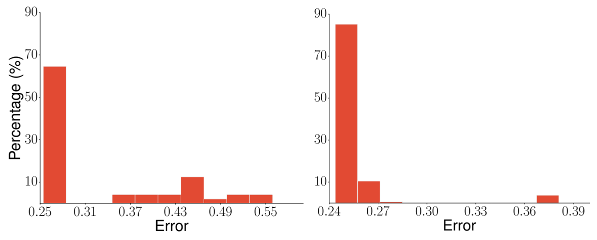


Figure 4. Histograms of error reconstruction for runs with different hyperparameter settings for (a) MS (P-DO, 48 settings), and (b) MIMIC (NPB-H, 1125 settings) datasets.

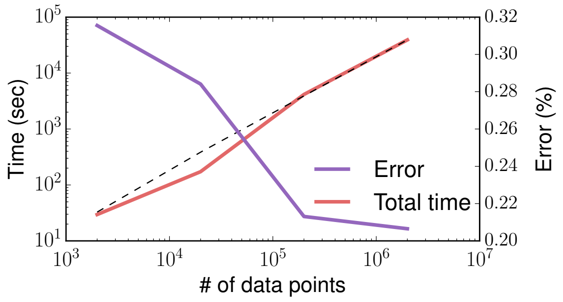


Figure 5. Runtime and error of nonparametric JUMP-means algorithm with increasing synthetic data size. The runtime scales linearly with data size (dashed black line).

5.2. Nonparametric Model

Vital Signs Monitoring (MIMIC) We now consider a version of the problem of understanding physiological signals discussed in the introduction. We use data from the MIMIC database (Goldberger et al., 2000; Moody & Mark, 1996), which contains recordings of several vital parameters of ICU patients. Specifically, we consider blood pressure readings of 69 ICU patients collected over a 24-hour period and sub-sample observation sequences of length 32 for each patient, keeping the start and end times fixed.[2] For testing, we randomly hold out $\sim$25% of the observations.

To initialize JUMP-means, we choose uniform matrices for $\boldsymbol{\rho}, \bar{\pi}_0$ and $\bar{\pi}_1$ and set $M = 1$. The hyperparameters γ and ξ_1 are set to 5, while ζ, ξ, and ξ_2 are set to 0.005. Using a Gaussian likelihood model for the observations, we run our model for 50 iterations. We compare with particle MCMC (PMCMC) (Andrieu et al., 2010) and EM. PMCMC is a state-of-the-art inference method for iMJPs (Saeedi & Bouchard-Côté, 2011), which we run for 300 iterations with 100 particles. For PMCMC, we first categorize the readings into the standard four categories for blood pressure provided by NIH[3]. We run EM with a number of hidden states from 1 to 12 and report the best performance among all the results. For initializing the EM, we use the same setting as the Synthetic 2 case.

For evaluation, we consider the time point of a test observation and categorize the mean of the latent state at this time point (using the same categories obtained above) to compare against the actual category. Table 1 shows that JUMP-means significantly outperforms PMCMC and obtains a 21% relative reduction in average error rate. Fig. 2(c) plots the error against iterations of both algorithms. In terms of CPU time, each iteration of JUMP-means (Java) and PMCMC (Java) takes 0.17 and 1.95 seconds, respectively. Compared to EM's error rate of 25.7%, JUMP-means reaches a rate of 24.3% without the need to separately train for different number of states. The second-best result for the EM had an error of 45%, which shows the importance of model selection when using EM.

Fig. 3 (bottom) provides an example of the latent trajectory inferred by JUMP-means. The observations are uniquely colored by the latent state they are assigned. We note that the model captures different levels of blood pressure readings and provides a non-degenerate latent trajectory.

Hyperparameters A well-known problem when applying SVA methods is that there are a number of hyperparameters to tune. In our objective functions, some of these hyperparameters (γ, μ_λ, and ξ_λ) have natural interpretations so prior knowledge and common sense can be used to set them, but others do not. Fig. 4 shows histograms over the errors we obtain for runs of JUMP-means on the MS and MIMIC datasets with different settings. We can see that a significant fraction of the runs converge to the minimum error, while some settings — in particular when the hyperparameters were of different orders of magnitude — led to larger errors. Hence, the sensitivity study indicates the robustness of JUMP-means to the choice of hyperparameters.

Scaling Fig. 5 shows the total runtime and reconstruction error of the non-parametric JUMP-means algorithm on increasingly large amounts of synthetic data. The algorithm is able to handle up to a million data points with the runtime scaling linearly with data size. Furthermore, the error rate decreases significantly as the amount of data increases. See the Supplementary Material for further details.

6. Conclusion

We have presented JUMP-means, a new approach to inference in MJPs using small-variance asymptotics. We derived novel objective functions for parametric and Bayesian nonparametric models and proposed efficient algorithms to optimize them. Our experiments demonstrate that JUMP-means can be used to obtain high-quality non-degenerate estimates of the latent trajectories. JUMP-means offers attractive speed-accuracy tradeoffs for both parametric and nonparametric problems, and achieved state-of-the-art reconstruction accuracy on nonparametric problems.

[2]We use a small dataset for testing since PMCMC cannot easily scale to larger datasets.

[3]`http://www.nhlbi.nih.gov/health/health-topics/topics/hbp`

Acknowledgments

Thanks to Monir Hajiaghayi, Matthew Johnson, and Tejas Kulkarni for helpful comments and discussions. JHH was supported by the U.S. Government under FA9550-11-C-0028 and awarded by the DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

References

Andrieu, C., Doucet, A., and Holenstein, R. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

Asger, H. and Ledet, J. J. Statistical inference in evolutionary models of dna sequences via the em algorithm. *Statistical Applications in Genetics and Molecular Biology*, 4(1):1–22, 2005.

Bladt, M. and Sørensen, M. Statistical inference for discretely observed markov jump processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(3):395–410, 2005.

Bladt, M. and Sørensen, M. Efficient estimation of transition rates between credit ratings from observations at discrete time points. *Quantitative Finance*, 9(2):147–160, 2009.

Broderick, T., Kulis, B., and Jordan, M. I. MAD-Bayes: MAP-based Asymptotic Derivations from Bayes. In *International Conference on Machine Learning*, 2013.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

Hajiaghayi, M., Kirkpatrick, B., Wang, L., and Bouchard-Côté, A. Efficient Continuous-Time Markov Chain Estimation. In *International Conference on Machine Learning*, 2014.

Jiang, K., Kulis, B., and Jordan, M. I. Small-variance asymptotics for exponential family dirichlet process mixture models. In *NIPS*, pp. 3167–3175, 2012.

Kingman, J. F. C. *Poisson Processes*. Oxford Studies in Probability. Oxford University Press, 1993.

Kulis, B. and Jordan, M. I. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. In *International Conference on Machine Learning*, 2012.

Lange, J. *Latent Continuous Time Markov Chains for Partially-Observed Multistate Disease Processes*. PhD thesis, 2014.

Mandel, M. Estimating disease progression using panel data. *Biostatistics*, 11(2):304–316, 2010.

Moody, G. and Mark, R. A database to support development and evaluation of intelligent intensive care monitoring. In *Computers in Cardiology, 1996*, pp. 657–660, Sept 1996.

Perkins, T. J. Maximum likelihood trajectories for continuous-time markov chains. In *NIPS*, pp. 1437–1445, 2009.

Rao, V. and Teh, Y. W. Fast MCMC sampling for Markov jump processes and extensions. *The Journal of Machine Learning Research*, 14(1):3295–3320, 2013.

Roychowdhury, A., Jiang, K., and Kulis, B. Small-Variance Asymptotics for Hidden Markov Models. In *Advances in Neural Information Processing Systems*, pp. 2103–2111, 2013.

Saeedi, A. and Bouchard-Côté, A. Priors over recurrent continuous time processes. In *NIPS*, volume 24, pp. 2052–2060, 2011.