

Customer Event Rate Estimation Using Particle Filters

Harsha Honnappa¹

¹ Applied Research Group,
Satyam Computer Services Ltd.,
Indian Institute of Science (IISc),
Bangalore, India.

Email: Harsha_Honnappa@satyam.com

Abstract

Estimating the rate at which events happen has been studied under various guises and in different settings. We are interested in the specific case of consumer-initiated events or transactions (credit/debit card transactions, mobile phone calls, online purchases, etc.), and the modeling of such behavior, in order to estimate the rate at which such transactions are made. In this paper, we detail a model of such events and a Bayesian approach, utilizing Sequential Monte Carlo technology, to online estimation of the event rate from event observations alone.

Keywords: Event Rate Estimation, Markov Jump Process, Particle Filter, Cox Process, Poisson Process.

1 Introduction

Modeling consumer behavior is advantageous for a multitude of business problems - targeted advertising/marketing, fraud detection, click-stream analysis, etc. There are a plethora of statistics that one can 'mine' or extract from this data ¹. One statistic that is particularly significant, especially since (individual) consumer events are time-sequenced, is the rate at which such events occur. The rate is a particularly useful metric in detecting changing consumer behavior.

Indeed, there have been several approaches to estimating the (mean) rate at which events occur. In (Lambert et al., 2001), a very detailed algorithm for estimating the rate using a 'controlled' version of the exponentially weighted moving average (EWMA) time-series model is described. Scott et al, study click-rates with a Markov Modulated Poisson Process (MMPP), in (Scott and Smyth, 2003). In (Weinberg et al., 2006) a Markov Chain Monte Carlo (MCMC) approach to determining the rate of a doubly stochastic Poisson process is detailed, based on call center data. Similar approaches have been used in other proprietary systems.

In this paper, our aim is to detail a different approach to rate estimation. We assume that the rate itself is unobservable, since it is not exactly a physically manifested signal. Thus, we can view the

rate estimation problem as a latent state estimation or, in signal processing parlance, a filtering problem. As we shall see, in this case we are dealing with a non-linear filtering problem. Further, we want to estimate the rate, in an online fashion, without storing much data.

Latent state space models are widely used in many applications. The problem that this paper aims to solve is estimating this state using only observations and a knowledge of a *state space model*. This filtering problem has been solved in a few cases (and indeed optimally), and the most famous example of such a filter is the Kalman filter, which optimally estimates the state in the case of a linear, Gaussian model; i.e., when the state process is a Gaussian process and the relation between observations and states is a linear model, driven by Gaussian noise. As we shall see, in our model the underlying state process cannot be a Gaussian process. Instead, we assume that the rate process follows a piecewise deterministic Markov process (PDMP), see (Davis, 1984). More specifically, we assume that the rates follow a piecewise continuous Markov process, or a Markov jump process (MJP). Further, we assume a Poisson observation model, in which the number of events occurring within a given time interval is Poisson distributed, dependent upon the underlying rate process.

Given our modeling assumptions, the solution to the filtering problem requires sophisticated algorithms. Here, we consider a Bayesian approach to solving the problem - using Sequential Monte Carlo (SMC) methods. SMC methods, more commonly known as particle filtering (PF), estimate the posterior density of the (latent) states given the observations, by a cloud of weighted samples or *particles*. These particles are updated and propagated using sequential importance sampling (SIS) and MCMC methods. For a very good introduction to PF's see (Arulampalam et al., 2002) and (Cappé et al., 2007). Generic particle filtering approaches usually used assume that, for every observation, there is a corresponding underlying state; i.e., the state process evolves stochastically at every time instant. However, in our model, we assume that the underlying state changes at random instants of time and that it remains constant between these instants of time. Thus, the rate at which state changes occur is different from the rate at which events are observed. Put another way, we do not know the number of jumps in the state process. This requires a different paradigm of PF's.

Variable rate particle filters (VRPF), (Godsill and Vermaak, 2005) and (Godsill et al., 2007), are a specific type of particle filter that allows for different

Copyright ©2008, Australian Computer Society, Inc. This paper appeared at the Seventh Australasian Data Mining Conference (AusDM 2008), Glenelg, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 87, John F. Roddick, Jiuyong Li, Peter Christen and Paul Kennedy, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

¹We shall henceforth refer to the transactions or consumer-initiated events, noted above, as *events*.

rates of evolution for the observations and the state. VRPF's define a 'neighborhood' for each sample observed, and sample *stopping times*, or jump-times, and states from a predefined importance density until the neighborhood for a given sample is *complete*, in some well defined manner. This framework is exactly what is required for estimating event rates, given our modeling assumptions. Here we detail how the VRPF algorithm can be applied to this problem, with appropriate definition of neighborhoods and importance density. We make some enhancements to the basic algorithm laid out in (Godsill et al., 2007), by incorporating *moves* into the filter to improve particle diversity.

The rest of this paper is organized as follows. In section 2, we describe the MJP model we assume for the rate process and the corresponding state evolution density functions. We also describe the Poisson observation model, and give a brief mathematical treatment of the the specific type of Poisson process we assume for this model. Next, in section 3, we describe the VRPF algorithm that we adopt in this paper. We also briefly discuss particle filters and the general idea behind them, for completeness. In section 4, we present results of experiments we conducted comparing the VRPF and EWMA algorithms and comparisons of the individual VRPF algorithms, incorporating moves and without moves. Finally, we conclude in section 5 with a listing of future research directions we intend to pursue and a brief discussion of the implications and importance of this model on estimating customer event rates.

2 The Model

There are numerous algorithms that aim to solve the latent state estimation problem. However, we need to first model the latent state process. There are a couple of dimensions to this general modeling problem and listing them will help in setting up our own problem. First, the state space itself can be discrete or continuous. Secondly, the event sequencing (or time-ordering) can be, again, discrete or continuous. As we shall see, it is useful to model the state process by a continuous-time, continuous state-space model, with a point process observation model, in our problem.

An example would help set the stage for a more formal explanation. Consider credit card usage. Generally, a histogram of the time of usage of the cards, across a portfolio of card users and over a 24 hour period, would be a bell curve. Clearly, there are periods of the day during which consumers tend to transact more, and other times during which they transact less. For an individual user, the time scale is probably much different, but even there, individual consumers would tend to use their cards more over the weekend (shopping, movies etc.) than during the working week. Thus, there are clear categories of usage. However, even though there maybe a finite number of usage categories, the quantification of this usage (in terms of a rate) does not necessarily have to be in a finite or discrete set of values. To illustrate, suppose that a cardholder uses her card more often over a weekend, implying a higher rate. Even though she exhibits a fairly stationary behavior, it is not true that she has the exact same rate *every* weekend. The rate quanta can vary over a continuous range of values. Further, one cannot assume that the cardholders state changes abruptly at a *set* time instance (even though it is assumed that the state can change at an instant). Thus, even though the

number of state changes over a given time period maybe denumerable, the instants of the changes and number of state changes, and the magnitude changes of the state process are random. Figure 1 is an illustration of this conceptual model.

We can see that there is a compelling case for modeling such consumer behavior with continuous time stochastic processes. Let us now look at a more formal presentation, and start with a description of our model of consumer behavior.

2.1 An Observation Model

As mentioned in the introduction to this section, generally there are a denumerable and random number of events during any given period of time. We shall model the event observations as a type of Markov process - a Poisson process, with rate $\lambda > 0$; see (Ross, 2007, Ch. 5) for a good introduction. Note that Poisson processes are a type of continuous-time Markov chain, where in the states of the chain are the number of events seen up to a given instant of time. These processes are also called as counting processes, and also as pure-birth processes. Poisson processes have some very useful properties -

- The number of events over a given time interval is Poisson distributed - $P(N(t+s) - N(s) = n) = e^{-\lambda t} \frac{(\lambda t)^n}{n!}$. Where, $n \in \{0, 1, 2 \dots\}$.
- The Poisson process has stationary and independent increments; i.e., $N(t+s) - N(s)$ is independent of $N(s+u) - N(u)$ where, $u < s < t$.
- The time between events is exponentially distributed.

Note that in the basic, or homogeneous, Poisson process the rate, λ , is assumed to be a constant. However, in our problem, the rate is assumed to change at random instants of time to random locations. Thus, we need to turn to a more sophisticated model, a doubly stochastic Poisson process or Cox process, (Daley and Vere-Jones, 2003), in which the rate is assumed to be some stochastic process. There are several interesting applications of Cox processes, including modeling insurance risk and securities risk, see (Dassios and Jang, 2003) and (Lando, 1997) for illustrative examples.

2.1.1 Cox Process

We shall follow Bremaud's definition of a Cox process, (Bremaud, 1981) (also used in (Dassios and Jang, 2003)). Let, (Ω, \mathcal{F}, P) be a probability space, where \mathcal{F} consists of all σ -algebras \mathcal{F}_t . Suppose N_t is a Poisson process adapted to \mathcal{F}_t . Let λ_t be a non-negative process that is \mathcal{F}_t adapted, and

$$\int_0^t \lambda_s ds < \infty \text{ a.s.}$$

If $\forall 0 \leq t_1 \leq t_2$ and $u \in \mathcal{R}$,

$$E \left\{ e^{iu(N_{t_2} - N_{t_1})} | \mathcal{F}_{t_2} \right\} = \exp \left\{ (e^{iu} - 1) \int_{t_1}^{t_2} \lambda_s ds \right\}$$

Then, the probability of n events in time interval $t_2 - t_1$ is Poisson distributed, given the rate process over

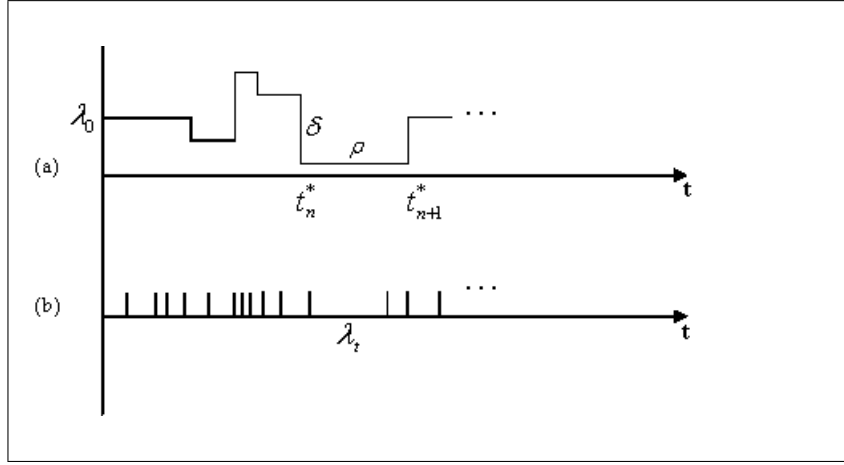


Figure 1: The conceptual model of a Poisson observation model and a Markov Jump Process rate process. (a) depicts the state process, where λ_0 is the initial value, ρ and δ are model parameters governing the jump-time Poisson process, and the state-magnitude Markov chain resp. and t_n^* and t_{n+1}^* are two jump times, (b) depicts the events sequence, where events are represented by marks on the time axis. The rate at which events occur is shown as λ_t which is the realization of the rate process at time t .

$$t_1 \leq s \leq t_2,$$

$$P \{N_{t_2} - N_{t_1} = n | \lambda_s, t_1 \leq s \leq t_2\} = \frac{e^{-\int_{t_1}^{t_2} \lambda_s ds} \left(\int_{t_1}^{t_2} \lambda_s ds \right)^n}{n!}$$

Thus, in order to employ the Cox process as a valid observation model, a model for the state evolution process, λ_t , needs to be chosen. In the next section we will consider an appropriate model for our problem.

2.2 The State Evolution Model

Markovian state evolution models are uniquely represented by a state transition probability - $P(Y|X)$. Where, crudely, P represents the probability of moving to state Y , given that the system is in state X at the previous step. This state transition probability is also represented by the use of a *transition function*, $k_{s,t}(dy, x)$, where $s(< t)$ are the times at which one considers the process evolution.

There are many choices of processes for describing the state evolution. A good example would be a *random walk*, where in the current state is a random perturbation of the state at the previous time instant-

$$\lambda_{t+1} = \lambda_t + \nu_{t+1} \quad (1)$$

Where, ν_{t+1} is the random perturbation or noise component, and λ is the state. It is quite simple to verify the Markovian nature of such a process. Further, if the noise is assumed to be standard normal, $\mathcal{N}(0,1)$, and $\lambda_0 = 0$, then this process represents Brownian motion (of course subject to some conditions that allow its description in continuous time).

In (Lambert et al., 2001) the authors describe a multiplicative noise state evolution model -

$$\lambda_{t+1} = \nu_{t+1} \lambda_t \quad (2)$$

Where,

$$\nu \sim \Gamma(\alpha, \alpha)$$

And λ_0 is assumed to be known.

In (Weinberg et al., 2006), the authors describe the behavior of calls arriving at a call center at a large North American bank. From their analysis, it is clear that the rate at which calls arrive has different stages, and that these vary from day-to-day. The authors model this behavior with a doubly stochastic Poisson process, with the rate evolution governed by -

$$\lambda_{t+1} = w_{d_{t+1}} v + \epsilon_{t+1} \quad (3)$$

Where, $w_{d_{t+1}}$ is the proportion of calls occurring in the period of interest, v is an estimate of the number of daily volume on the day of interest and ϵ_{t+1} is a random perturbation. The most interesting aspect of this model is the fact that the model accommodates both inter- and intra-day variation in the rate, thus allowing for greater control of the model.

However, as described in the introduction to this section, it appears more plausible that the states of consumers undergo a slow evolution. Further, we assume that the state can change only a finite number of times in a given time period (say, a day). We model the state evolution with a Markov process, and thus introduce the assumption that the future state is conditionally independent of the past, given the present.

Given the hypotheses stated above it does not appear reasonable to assume that the state evolution follows some type of a *diffusion process*. A pure-diffusion process assumes that the state is continuously evolving at each time instant. This violates the assumption of a denumerable number of state changes in a given period of time. Jump-diffusion processes appear to be a more plausible model, since one can assume that the state changes are represented by the jumps in the process, and that the process then evolves around the magnitude jumped to. However, this precludes any notion of a slow evolution, since the state still changes at each time instant.

A more plausible model for the state evolution process is to assume it follows a piecewise deterministic Markov process (PDMP). Specifically, we assume that

the process is a Markov jump process (MJP). Note, however, we do not pursue the PDMP description directly, and instead merely point out the connection between MJP's and PDMP's. Describing PDMP's is outside the purview of this paper, and the interested reader is directed to (Davis, 1984). Figure 1 shows an example of how the state process may vary. We assume that the state process is a pure-jump process, with constant trajectory, or flow, between the jumps. Our assumption that the state of a user does not change significantly, once in a given state, means that MJP's are a good model for representing such behavior. A brief review of MJP's and their properties follows.

2.2.1 Markov Jump Process (MJP)

A jump process is a stochastic process that changes its magnitude only at random instants of time. Generally, any process with piecewise-constant trajectories between jumps is called as a jump process. Markov Jump Processes impose a Markovian structure on the states, by requiring that the future states are conditionally independent of the past, given the present. Linked to this is an important result that shows that a jump process is Markovian *iff* the time between jumps is exponentially distributed - $F_\lambda(t)$, $\forall \lambda \in S$, with parameter $\rho(\lambda(t))$.

The basic characteristics of MJP's include -

- A stochastic kernel $k(\lambda, A)$ on $S \times \mathcal{B}(S)$ ($A \in \mathcal{B}(S)$) that satisfies the condition, $k(\lambda, \lambda) = 0$. This ensures that there actually *is* a jump at each jump time.
- A bounded, non-negative function of the state, $\rho(\cdot)$, that controls the rate at which jumps occur. We assume that the rate at which jumps occur is much lower than the rate process magnitude, $\lambda(t)$.

Intuitively, a continuous-state MJP can be thought of as starting in some position λ_0 at time t_0 , and remains in that state till an exponentially distributed time t_1 , when it jumps to a new location λ_1 according to the transition kernel, $k(d\lambda_1, \lambda_0)$. It then remains in the state λ_1 until the next exponentially distributed jump time, t_2 , and again the process jumps to a new location, following the transition kernel $k(d\lambda_2, \lambda_1)$. This continues till the end of the period of interest, or ad infinitum.

We can think of the MJP as being the composition of a Poisson (point) process, N , and a Markov chain, M , $\Lambda = N \circ M$. Now, we can model the state by a tuple, $\lambda = (\tau, \theta)$, where, τ is the jump time and θ is the state-magnitude. Further, we assume that the state evolution density can be decomposed in the following manner -

$$f(\tau_t, \theta_t | \tau_{t-1}, \theta_{t-1}) = f(\tau_t | \tau_{t-1}, \theta_{t-1}) f(\theta_t | \theta_{t-1}) \quad (4)$$

We assume that the future jump time is conditionally independent of the future state-magnitude, given the present jump time and the present state-magnitude, and the future state-magnitude is conditionally independent of the present jump time. This assumption fits in nicely with our assumption that the state process is a MJP. This is a very general description of the process evolution. In order to adapt this description to the problem of event rate estimation, we make a few further assumptions that will fully describe the model -

- We assume that users generally have a fairly moderate rate of usage, with jumps being not too high or low.
- The high rate states tend to be of short duration (or 'bursty').
- High rate states are usually followed by a drop down to a more moderate rate.
- A low rate state is followed by a moderate state with high probability.

Based on these, still, rather general assumptions, we model user behavior by considering the following forms for the state evolution densities.

2.2.2 Jump Time Distribution

As stated earlier, one of the properties of MJP's is that the time between jumps is exponentially distributed. The parametrization of this distribution is assumed to dependent upon the present state-magnitude in the following manner.

$$\frac{1}{\rho(\theta_t)} \exp\left\{-\frac{\tau_{t+1} - \tau_t}{\rho(\theta_t)}\right\} \quad (5)$$

Where, the mean rate of the jumps, as a function of the state-magnitude is given as,

$$\rho(\theta_t) = \frac{\alpha}{\theta_t}$$

Here, $\alpha \in \mathfrak{R}$. As stated earlier, we assume that the rate at which jumps occur is much lower than the event rate. α allows us to control the jump rate. Thus, one can see that a large rate value will produce a very low future jump time, and vice versa.

2.2.3 State-magnitude Distribution

In the description of a MJP, it was noted that the process is associated with a transition kernel which determines the evolution of the state-magnitude, θ . We use a random walk type model, (1), to describe the relation between the future and present state-magnitude. We define the random perturbation as -

$$\Delta\theta_{t+1} \triangleq \theta_{t+1} - \theta_t \quad (6)$$

Note that $\theta \in \mathfrak{R}^+$. This implies that the difference between the present and future magnitudes is bounded below -

$$-\theta_t \leq \Delta\theta_{t+1} < \infty$$

We could model $\Delta\theta_{t+1}$ by a shifted Gamma distribution (the density function of which exists), where the shift parameter is $-\theta_t$. However, as defined, MJP's require the probability of jumping to the same location should be zero, $k(\lambda, \{\lambda\}) = 0$. That is, $\Delta\theta_{t+1} \neq 0$. We accommodate this by modeling the density function as a mixture of two shifted gamma distributions, such that the density of the mixture at $\Delta\theta_{t+1} = 0$ approaches zero, for a fixed θ_t . By a slight abuse of mathematical propriety, we will define the first mixture component density, c_1 , as that having most of its density in the region $[-\theta_t, 0)$, and the second component, c_2 , as that having support, $[0, \infty)$. While there is a small probability that $\Delta\theta_{t+1} = 0$, by carefully choosing the parameters of the gamma distribution and the

mixture probabilities, it is possible to minimize this.

Now, recalling the assumptions we made regarding the user behavior, we assume that high or low rate magnitudes will be followed by a return to a more ‘moderate’ rate. We model this behavior by considering an appropriate probability mass function for the components, which is dependent upon the present state-magnitude. In order to achieve this, we require an appropriate function, π , such that, $\pi : \mathbb{R}^+ \rightarrow [0, 1]$. We choose an exponential function, with a suitable parametrization incorporating θ_t , as this function -

$$\pi(1|\theta_t) = 1 - \exp\left\{-\frac{\theta_t}{\eta}\right\} \quad (7)$$

$$\pi(2|\theta_t) = 1 - \pi(1|\theta_t) \quad (8)$$

Where, η parametrizes the mean magnitude to which the process returns to. For example, consider a situation with a large η ($\eta \gg 1$). Then, if θ_t is small, there is a greater probability of jumping to a larger value, in the next jump (since $\pi(2|\theta_t) > \pi(1|\theta_t)$), and vice versa. We call this model as an *exponentially modulated Gamma mixture*. Finally, the state-magnitude transition density function is given by -

$$f(\theta_{t+1}|\theta_t) = \pi(1|\theta_t) \Gamma(\delta, \frac{1}{\delta\theta_t}) + \pi(2|\theta_t) \Gamma(\delta, \frac{1}{\delta\theta_t}, \theta_t) \quad (9)$$

Where, $\Gamma(\cdot, \cdot)$ is the gamma distribution and $\Gamma(\cdot, \cdot, \cdot)$ is the shifted gamma distribution, with the third parameter the shift. $\delta (\in \mathbb{R}^+)$ is the gamma distribution parametrization.

A final observation to be made is that the state-magnitude is assumed to have a Gamma prior distribution, with parametrization δ . Thus, we have a description of the the latent state process transition density, (4). We will use this description in designing an estimation procedure for the state from observations. As stated earlier, the state evolution process is not directly observable, but only through the Cox process observation model. Further, we would like to estimate this process (or at least the mean) in an online fashion. We accomplish this estimation with a Particle Filter. We describe the filter design and algorithm in the next section.

3 A Particle Filter Solution

Non-linear state space models are described by a state evolution model,

$$x_t = a(x_{t-1}, \nu_t) \Leftrightarrow f(x_t|x_{t-1})$$

And an observation model,

$$y_t = b(x_t, \iota_t) \Leftrightarrow f(y_t|x_t)$$

Where, $f(x_t|x_{t-1})$ and $f(y_t|x_t)$ are the transition density (a.k.a. the transition kernel) and the observation density, resp. $a(\cdot)$ and $b(\cdot)$ are some non-linear functions, and ν and ι are the driving noise processes. We have detailed our modeling assumptions in the previous section. Now, we are interested in estimating the *posterior density*², $f(x_{0:T}|y_{0:T})$. Using Bayes rule we get -

$$\begin{aligned} f(x_{0:T}|y_{0:T}) &= \frac{f(y_{0:T}|x_{0:T})f(x_{0:T})}{\int f(y_{0:T}|x_{0:T})f(x_{0:T})d\mathbf{x}} \\ &= \frac{f(x_0)f(y_0|x_0) \prod_{i=1}^T f(x_i|x_{i-1})f(y_i|x_i)}{\int f(y_{0:T}|x_{0:T})f(x_{0:T})d\mathbf{x}} \end{aligned} \quad (10)$$

However, evaluating this expression requires evaluating the integral, which is intractable in most cases. One can use a simulation based approach to estimating this density, using importance sampling (IS). IS uses a carefully designed *auxiliary* density, $q(\cdot)$, called as the importance function, whose support covers the support of the target density function. Then, the expectation, $E(g(X))$, of some function $g(\cdot)$ of random variable X , can be estimated by using N IID, weighted, samples drawn from the importance distribution -

$$\begin{aligned} E[g(X)] &= \int_{\mathcal{D}(x)} g(x)f_X(x)dx \\ &= \int_{\mathcal{D}(x)} g(x)\frac{f_X(x)}{q(x)}q(x)dx \\ &\approx \sum_{i=0}^{N-1} w_i g(x_i) \end{aligned} \quad (11)$$

Where, w_i are the normalized importance weights, N is the number of samples and x_i are samples drawn from the importance density. The un-normalized importance weights, \tilde{w}_i , are defined as -

$$\tilde{w}_i = \frac{f_X(x)}{q(x)} \quad (12)$$

These weights are then normalized before being used in (11) -

$$w_i = \frac{\tilde{w}_i}{\sum_{i=0}^{N-1} \tilde{w}_i}$$

It can be shown that the estimate (11) converges to the true expectation, $E[g(X)]$, in the limit, $N \rightarrow \infty$. The standard IS procedure can be extended to a sequential situation, where the weights are estimated and updated at each instant an observation is made. This is called as the sequential importance sampling (SIS) algorithm, and forms a subset of sequential Monte Carlo (SMC) methods. SIS forms the basis of particle filter (PF) algorithms.

3.1 Generic Particle Filters

PF’s are a specific instance of SIS algorithms used to estimate the the posterior density (and associated expectation functions), using a set of samples, or particles, drawn from the importance density. Essentially one can think of drawing particles as drawing a sample path from the importance density. The importance weights are given by -

$$\tilde{w}_t^{(i)} = \frac{f(x_{0:t_n}^{(i)}|y_{0:t_n})}{q(x_{0:t_n}^{(i)}|y_{0:t_n})} \quad i = 1 \dots N \quad (13)$$

²We shall implicitly assume that the density function exists for all distributions of interest to us.

Where, $x^{(i)}$ is the i^{th} particle drawn from $q(\cdot)$. By assuming that the importance density can be decomposed as -

$$q(x_{0:t_n}|y_{0:t_n}) = q(x_{0:t-1}|y_{0:t-1})q(x_t|x_{t-1}, y_t) \quad (14)$$

i.e., the particle value at t is dependent only on the observation at t and the particle value at the previous time instant. Crudely, one can say that the sample path is extended by sampling from the second part of the importance function decomposition. From (10), and the Markovian nature of the state process, we can see that the posterior density function can be expressed as -

$$f(x_{0:t}|y_{0:t}) \propto f(y_t|x_t) f(x_t|x_{t-1}) f(x_{0:t-1}|y_{0:t-1}) \quad (15)$$

Now, substituting (14) and (15) into (13), and dropping the terms that do not depend upon the state sequence, we get -

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{f(y_t|x_t^{(i)})f(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|y_t, x_{t-1}^{(i)})} \quad (16)$$

Thus, we get a sequential update equation for the importance weight, which depends only upon the present particle, the current observation and the previous particle. At each observation instant, we draw a sample from the second part of the RHS of (14), and update the weight according to (16). However, this basic algorithm has problems with its performance. Specifically, it suffers from weight degeneracy, where most of the weight is concentrated on a few particles after a few iterations of the algorithm. This problem can be worked around by re-sampling the particles either on every iteration, or according to some criterion (described later on).

Note that in the generic particle filter, we know that the dimension of the sample space of the posterior probability distribution is strictly increasing; i.e., $\dim(S_{t-1}) < \dim(S_t)$. That is, we know that at each observation instance, there is exactly one latent state variable value. However, this is not the situation with our problem. Recall that we assume a Markov jump process (MJP) model for the state evolution process, that changes state at a rate much lower than the state-magnitude at a given instant of time. Thus, we have a situation where the number of state variable samples at each observation instance is unknown. Here, we must assume that at each instance, we have the same sample space, S , but the support of the posterior density function, E_t , is increasing. Thus, we require different algorithms to solve this problem.

There have a few attempts at solving this type of filtering problem. The Variable Rate Particle Filter (VRPF), described in (Godsill and Vermaak, 2005) and expanded in (Godsill et al., 2007), works by sampling stopping times, such that every observation has a *complete neighborhood*, where neighborhood is some well defined region around each observation. In (Del Moral et al., 2006), a general technique for sampling from a sequence of distributions that are defined on the same underlying sample space, called Sequential Monte Carlo (SMC) Samplers is defined. Applications of this algorithm to PDMP's and jumping processes is described in (Whiteley et al., 2007). We

adopt the VRPF algorithm in this paper, and describe some simple extensions to the basic algorithm described in (Godsill et al., 2007). We describe the VRPF algorithm next.

3.2 Variable Rate Particle Filter

For brevity, a brief description of the VRPF follows; details of the algorithm are available in (Godsill and Vermaak, 2005) and (Godsill et al., 2007). One of the assumptions made about the state evolution model is that the rate at which jumps occur is much lower than the state-magnitude itself. Thus, we require a way of redefining (10) and (15), to incorporate this fact. First, the posterior distribution that we are interested in is defined as -

$$f(x_{0:k_t^\square}|y_{0:t}) = f((\tau, \theta)_{0:k_t^\square}|y_{0:t}) \quad (17)$$

Where, k_t^\square is the index of the last jump time, τ_k , greater than the observation time, t ; i.e.,

$$k_t^\square = \min\{k : \tau_k > t\}$$

Recall from (4) that the state is a tuple, (τ, θ) , composed of the jump time and the state-magnitude resp.. This leads to a definition of the *neighborhood* of the t^{th} observation, y_t as -

$$\aleph_t = \left\{ x_k : \tau_{k_{t-1}^\square+1} < \dots < t < \tau_{k_t^\square} \right\} \quad (18)$$

Accompanying the neighborhood structure, we also define a neighborhood function, $\hat{\phi}(t)$, that helps compute a neighborhood structure from the state-magnitude values. There are many possibilities for this function. In our case, we adopt the interpolation function defined in (Godsill et al., 2007), as it fits our problem well -

$$\hat{\phi}(t) = \frac{\theta_k(\tau_k - t) + \theta_{k-1}(t - \tau_{k-1})}{\tau_k - \tau_{k-1}}, \quad \tau_{k-1} \leq t < \tau_k$$

Now, using Bayes rule and the Markovian nature of our model, (17) can be expanded as -

$$\begin{aligned} f(x_{0:k_t^\square}|y_{0:t}) &= \frac{f(y_{0:t}|x_{0:k_t^\square}) f(x_{0:k_t^\square})}{f(y_{0:t})} \\ &\propto f(y_t|x_{\aleph_t})f(x_{k_{t-1}^\square+1:k_t^\square}|x_{k_{t-1}^\square})f(x_{0:k_{t-1}^\square}|y_{0:t-1}) \end{aligned} \quad (19)$$

(19) is the VRPF analogue of (15). Now, we can choose an appropriate importance density function, $q(x_{0:k_t^\square}|y_{0:t})$, such that it factorizes as in (14) -

$$q(x_{0:k_t^\square}|y_{0:t}) = q(x_{k_{t-1}^\square+1:k_t^\square}|x_{k_{t-1}^\square}, y_t)q(x_{0:k_{t-1}^\square}|y_{0:t-1}) \quad (20)$$

The importance weight update is then obtained by combining (19) and (20) in the equivalent of (13) -

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \frac{f(y_t|x_{\aleph_t}^{(i)})f(x_{k_{t-1}^\square+1:k_t^\square}^{(i)}|x_{k_{t-1}^\square}^{(i)})}{q(x_{k_{t-1}^\square+1:k_t^\square}^{(i)}|x_{k_{t-1}^\square}^{(i)}, y_t)} \quad (21)$$

Where, the samples $x_k^{(i)}$ are drawn from the importance density -

$$x_{k_{t-1}^\square+1:k_t^\square}^{(i)} \sim q(x_{k_{t-1}^\square+1:k_t^\square}^{(i)} | x_{k_{t-1}^\square}^{(i)}, y_t)$$

From (19) we see that the posterior distribution depends upon the number of jumps, k_t^\square , made up to the t^{th} observation. This is a random variable itself, and this fact will have a bearing upon the design of the particle filter algorithm. Essentially the algorithm comes down to sampling jump time proposals from the importance density until the neighborhood of the current observation is *complete*. In our problem, the current observation y_t is basically the time between the latest observation and the last observation. Thus, if the last observation was at time instant v_{t-1} , then the instant of the current observation is $y_t + v_{t-1}$. Thus, completing the neighborhood requires drawing jump-times and corresponding state-magnitudes from the importance density till one of the jump-times is greater than $y_t + v_{t-1}$.

The algorithm performance hinges to a great extent on the choice of importance density, $q(\cdot)$. A simple and often used choice for the importance density is the state evolution density, (4). This type of a particle filter is called as a *bootstrap filter*. As noted in (Arulampalam et al., 2002), the state space is explored without any knowledge of the current observation. This could make the algorithm susceptible to perform poorly, when there are outliers in the data. However, since we assume that most consumers tend to have a fairly stable behavior, this might not cause a problem for us, and using the prior transition density as the importance density should not be a bad choice.

We will incorporate re-sampling into the algorithm. Re-sampling is performed when the Effective Sample Size (ESS) falls below some pre-defined threshold. ESS measures the number of samples that have significant weight, and is defined as -

$$N_{ess} = \frac{N_s}{1 + Var(w_k^*)}$$

Where, w^* is the *true* importance weight, and N_s is the number of particles. The ESS is approximated by,

$$\hat{N}_{ess} = \frac{1}{\sum (w_k^i)^2}$$

Particles are resampled using the sample-with-replacement regime, with the normalized weights as pseudo sample probabilities. Post re-sampling the weights of the sampled particles is set equal to $\frac{1}{N}$. The basic VRPF algorithm with re-sampling is listed in **Algorithm 1**.

Over time, the re-sampling regime suffers from a lack of sample diversity. As the re-sampling occurs, there is a greater tendency to sample the same few particles. Thus, even though the weight degeneracy is eliminated, another problem crops up. In order to correct for this problem, authors in the past have suggested adding *moves* to the resampled particles, see (Gilks and Berzuini, 2001), involving an MCMC kernel. The idea behind using moves is to adjust the position of the resampled particles, so that they are all not at the same location, and with the same

Algorithm 1 Bootstrap VRPF algorithm

- Initialize Particles

- 1: **for** i in 1 to N **do**
- 2: Set $\tau_0^{(i)} = 0$
- 3: Draw $\theta_0^{(i)} \sim \Gamma(\beta, \frac{1}{\beta})$
- 4: **end for**

- Start Algorithm

Require: T and N

1. Re-sampling

- 1: Compute $\hat{N}_{ess} = \frac{1}{\sum (w_k^i)^2}$
- 2: **if** $\hat{N}_{ess} < T$ **then**
- 3: **for** i in 1 to 10 **do**
- 4: Resample particle i with replacement, with probability $\sim w^{(i)}$
- 5: Set $w^{(i)} = \frac{1}{N}$
- 6: (optional) Move according to the prior dynamical density (4)
- 7: **end for**
- 8: **else**
- 9: Continue
- 10: **end if**

2. Propagation

- 1: **for** i in 1 to N **do**
 - 2: **Complete Neighborhood**
 - 3: $j = -1$
 - 4: **repeat**
 - 5: Increment j
 - 6: $\tau_{k_{t-1}^\square+j}^{(i)} \sim f(\tau | \tau_{k_{t-1}^\square+j}^{(i)}, \theta_{k_{t-1}^\square+j}^{(i)})$
 - 7: $\theta_{k_{t-1}^\square+j}^{(i)} \sim f(\theta | \theta_{k_{t-1}^\square+j}^{(i)})$
 - 8: Augment Sample Path
 - 9: **until** $\tau_{k_{t-1}^\square+j}^{(i)} > y_t$
 - 10: **Weight Computation**
 - 11: $\tilde{w}_t^{(i)} \propto w_{t-1}^{(i)} f(y_t | x_{\hat{\delta}_{t-1}})$
 - 12: **end for**
-

state-magnitude. This way, the diversity of the particles is ensured. Here we detail a simple particle movement regime.

The re-sampling regime involves moving the last sample in a particle according to a constrained prior dynamical density, at each re-sampling instant -

$$\begin{aligned} x_{k_t^\square}^{(i)} &\sim f(\tau, \theta | \tau_{k_t^\square-1}, \theta_{k_t^\square-1}, t) \\ &= f(\tau | \tau_{k_t^\square-1}, \theta_{k_t^\square-1}, \tau > t) f(\theta | \theta_{k_t^\square-1}) \end{aligned}$$

That is, the last sample in the particle sample path is moved to a new location according to the prior dynamical density, such that the jump instant is greater than the current observation instant. Next, we describe the experiments conducted and the results on these experiments.

4 Experiments and Results

We compare our algorithm to a standard exponentially weighted moving average (EWMA) filter, commonly used in time series filtering. As pointed out in the introduction, (Lambert et al., 2001) describe an algorithm for estimating the rate using a controlled version of the EWMA filter, such that a separate EWMA filter is maintained for each time interval of interest. We consider the simple version of the EWMA. We first briefly describe the EWMA and issues arising in its use, then describe the experiments run and finally present the results.

4.1 Exponentially Weighted Moving Average Filter

The EWMA is a very simple model that just averages the last n values of the observations to generate the current estimate of the mean value of the random variable of interest -

$$\bar{Y}_n = (1 - \alpha)\bar{Y}_{n-1} + \alpha x_n, \quad 0 < \alpha < 1$$

Where, Y_n and Y_{n-1} are the current and previous (resp.) estimated mean time between observed events, x_n is the current observation (time since the last transaction) and α is a fixed weighting value. The mean event rate we estimate is the inverse of the mean time between events. The question is, how does one choose the value of the weight α ? A general heuristic is to choose α such that $\alpha = \frac{2}{n+1}$. However, there is no set guideline and describing some point estimation method like maximum likelihood for this parameter is outside the purview of this paper.

4.2 Experiments

We ran the VRPF algorithm and the EWMA on data generated using the model described in section 2. Figure 2 shows the sample path that we consider in this test, along with the observation instants. This test can be thought of as a sanity check on the efficacy of the model, and also serves as a test-bed to compare a traditional solution (EWMA), with the proposed algorithm. We use the mean squared error (MSE) as the criterion to compare the algorithms -

$$MSE := \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2$$

Where, n is the number of observations, x is the actual observation value and \hat{x} is the estimated value.

In order to generate the data, we set up the model with $\delta = 2.0$, $\alpha = 10.0$ and $\eta = 10.0$. Using this parametrization, we generated data such that there are exactly 10 jumps in the state process. We can see that figure 2 depicts different types of behavior - (from left to right on the time scale) jumping between moderate values, a jump to a very low rate, followed by a spike in the rate.

We use the same parametrization in the prior dynamical density in the VRPF. We present results for two versions of the algorithm, as noted above. First, we use simple re-sampling with no moves at each iteration of the algorithm, and a second version with a very simple move based on the prior dynamical density, (4). In the latter case we also present results with re-sampling when the effective sample size (ESS) falls below 0.4, and re-sampling at each iteration. In order to compare the VRPF algorithm with the EWMA, we ran the algorithms on the dataset above once, with the VRPF's initialized with 10,000 particles. For the EWMA, we ran the simulation multiple times, using different values for α in the set $\{0.1, 0.2 \dots 0.9\}$. We found that the best value of α in a mean-squared error sense is 0.2.

4.3 Results

Figure 3 shows the performance of the algorithms over the entire sample path. Figures 4(a), 4(b) and 4(c) show the performance at the rate spike, for clarity. We compare the EWMA to each of the VRPF algorithm flavors. It is clear that the particle filters are very close in performance to the EWMA, and indeed tend to converge to the actual value a lot quicker than the EWMA. However, we also see that the particle filter shows much more variance in the spikes, compared to the EWMA. This is most pronounced in the case of the VRPF algorithms with moves. Thus, incorporating moves into the algorithm certainly helps it converge faster, but it also tends to make it a bit more 'volatile'. The table below, table 1, shows the comparison of the algorithms using the MSE criterion. Clearly, the VRPF algorithms, VRPF_Move_ESS and VRPF_NoMove, perform better than the EWMA. Interestingly the VRPF with plain re-sampling at each iteration performed much better than the algorithms with moves.

Here, VRPF_Move_ESS denotes the VRPF algorithm with moves and re-sampling when ESS falls below 0.4, VRPF_Move_NoESS is the VRPF algorithm with moves and re-sampling at each iteration and VRPF_NoMove denotes the VRPF algorithm with only re-sampling at each iteration. In order to compare the VRPF algorithms themselves, we ran the algorithms 30 times on the same dataset, and estimated

Algorithm	MSE
EWMA	19.0839
VRPF_Move_ESS	15.4523
VRPF_Move_NoESS	25.8216
VRPF_NoMove	9.28208

Table 1: Comparison of EWMA to VRPF Algorithms

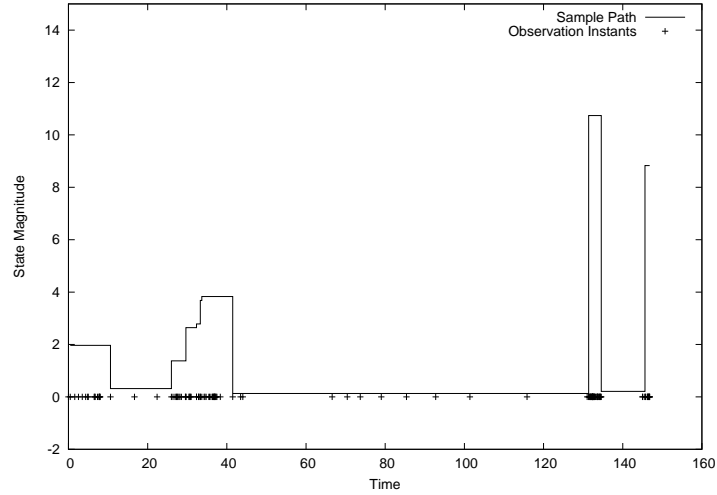


Figure 2: The figure shows the sample path of the state process that we use to compare the algorithms. The sample path is the solid line (-) and the observation instants are the '+'.

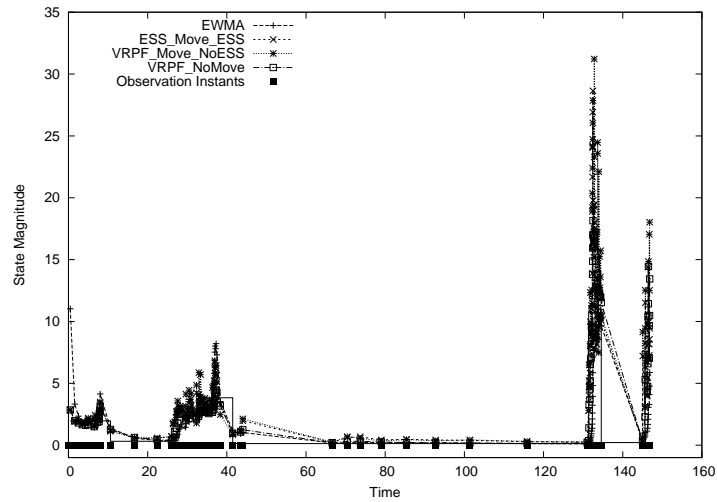


Figure 3: The relative performance of the algorithms on the entire sample path shown in figure 2. We show the estimated mean process.

the mean MSE over the runs. For this experiment, we changed the number of particles to 5,000. The mean MSE is estimated using-

$$\widehat{MSE} := \frac{1}{n} \sum_{j=1}^n \left\{ \frac{1}{m} \sum_{i=1}^m (x_{ji} - \hat{x}_{ji})^2 \right\}$$

Where, m is the number times the algorithms are run, n , as before, is the number of observations. The table below, table 2, summarizes the results of this test.

Figure 5 shows the performance of the VRPF algorithms at the rate spike. We can clearly see that the VRPF with no moves, figure 5(c), shows significant variation, as indicated by the large 95% confidence bands at each observation. Thus, even though we happened to obtain a good performance in the first experiment, the VRPF_NoMove is clearly a bad choice. The VRPF_Move_NoESS, figure 5(b), too, shows significant variance in the performance and the best algorithm would in fact be the VRPF with moves, and re-sampling when ESS falls below 0.4.

Algorithm	\widehat{MSE}
VRPF_MOVE_ESS	16.1922
VRPF_MOVE_NoESS	25.2604
VRPF_NoMove	24.7914

Table 2: Comparison of VRPF Algorithms

5 Discussion and Future Work

We have detailed a sequential Monte Carlo approach to estimating the rate at which customer initiated events are made. We assume that the events are governed by a doubly stochastic Poisson process. We assume that the rate process follows a Markov jump process, and we detailed an appropriate model of customer behavior. Based on the Poisson observation model and the Markov jump process model for the rate process, we showed how to use a particle filter,

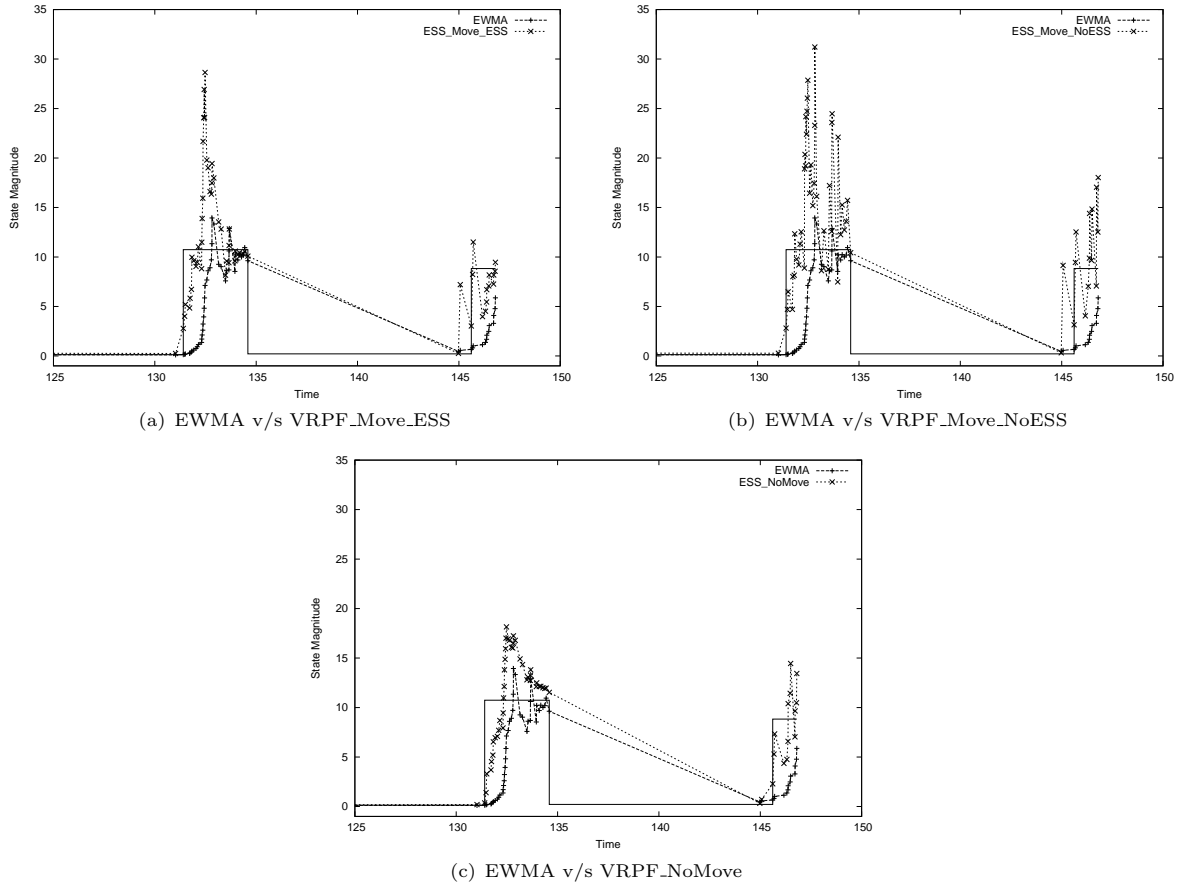


Figure 4: Relative Performance at the spike in the rate sample path. Here we compare the EWMA algorithm with each of the EWMA algorithm flavors. It is interesting to note that the EWMA_NoMove algorithm shows a much smoother behavior, with lesser variance in its estimates, compared to the other two algorithms. However, as seen in figure 5(c), below, it has significant variation from run to run.

the Variable Rate Particle Filter, to estimate the latent state. Our experiments show that the particle estimates of the mean rate is significantly better than the EWMA estimate of the state process sample, based on a mean squared error criterion.

5.1 Implications Of The Proposed Method

As noted in the introduction section, event rate estimation provides significant inputs towards solving many business problems, including fraud detection, click-stream analysis, targeted advertising etc. Consider, as an exemplar of these problems, fraud detection in credit cards. The events (card transactions) occur in a time-ordered fashion, with each consumer having a fairly stable transaction behavior. One of the measures of this stable behavior is the rate at which those transactions are made. Of course, in order to classify transactions as fraudulent or not, it is necessary to include a vast number of interesting features. However, knowledge of a customers historical transaction behavior is indispensable in detecting fraudulent behavior. The event rate is, arguably, the most important measure of past behavior. The change in transaction rate, in fact, can be an important marker for early detection of fraud events, as an indicator of abnormal consumer behavior. As another example, consider the case of click-stream analysis. By knowing how often a person browsing a website is likely to click on a link, it is possible to optimize the amount of caching to be done, for instance, to serve up webpages to millions of users simultaneously. Another example is in estimating

the amount of time a website patron spends on a particular page, thereby helping to optimize the amount of ad-spend on a particular page. Once again, knowledge of the event/transaction rate is very useful.

Thus, one can see that estimating the event rate is crucial to solving many important business problems. The estimate has to be as accurate as possible to be useful in any analytics used to solve these business problems. Our solution is to make some justifiable assumptions regarding the stochastic process that models the rate. Poisson models of events are widely used, and are justifiable by the great flexibility they afford. But, of course, the rate process is not observable, and has to be estimated from observations of the events alone. We showed that using a Bayesian method, Sequential Monte Carlo (SMC), the estimation of the rate from event observations is possible and with much better performance compared to the widely used EWMA approach. It is important to note that this paper lays out an extremely flexible framework for more accurate estimates of the rate process. Indeed, the experiments conducted, though on experimental data, clearly indicate the power and efficacy of the approach.

5.2 Future Directions

As future work, there are quite a few directions to consider -

- a) Recent results in (Whiteley et al., 2007) show that the SMC samplers framework, (Del Moral

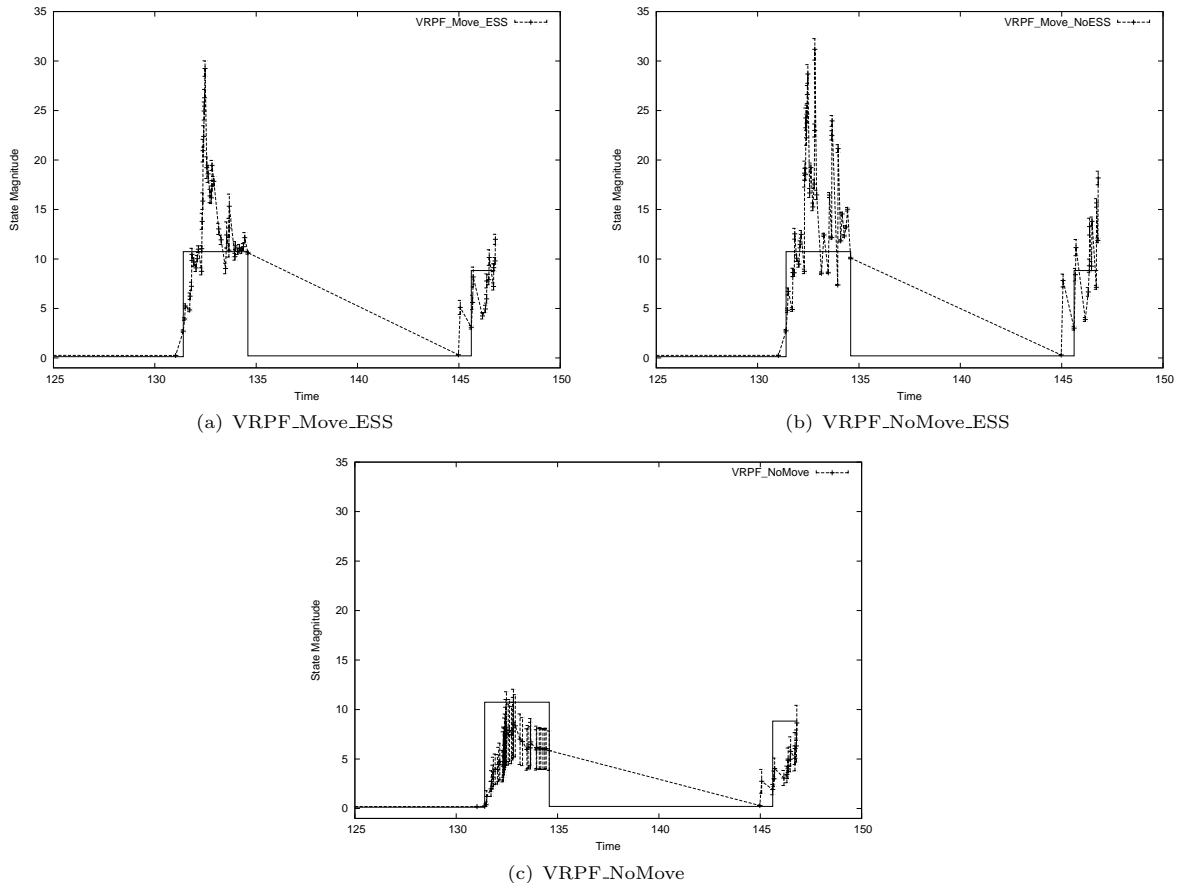


Figure 5: Relative Performance of the VRPF algorithms alone, at the spike in the rate sample path, over a 30 repeat run of the experiment. Also indicated are the 95% confidence bands for each estimate. Note that the VRPF_NoMove algorithm, (c), shows very significant confidence bands, compared to the other two algorithm flavors.

et al., 2006), is a better way of approaching particle filtering of MJP's. We intend to investigate this approach.

- b) The current work does not make any explicit reference to the time scale over which the state estimation is being done. In the introduction we mentioned a couple of approaches to solving the rate estimation problem in Poisson observed events - (Lambert et al., 2001) and (Weinberg et al., 2006). Both of these approaches incorporate controls into the models, so that the intra-day and inter-day rate variations can be captured. We need to introduce such controls into the model and the estimation algorithm.
- c) The results shown are on experimental data alone. It would be interesting to evaluate the approach on empirical data.
- d) Investigating the design of better importance densities for the MJP type setting. This point is related to a) above, since the SMC samplers framework allows for the design of better proposals, by considering mixtures of proposal kernels.
- e) The algorithm assumes knowledge of the parametrization of the jump time and state-magnitude distributions. This would require significant amount of data understanding in real world scenarios. Thus, there is also space to improve the algorithm by making it adaptive, and learn the distribution parameters from data.

References

- Arulampalam, M., Maskell, S., Gordon, N., Clapp, T., Sci, D., Organ, T. and Adelaide, S. (2002), 'A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking', *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]* **50**(2), 174–188.
- Bremaud, P. (1981), *Point Processes and Queues: Martingale Dynamics*, Springer-Verlag.
- Cappé, O., Godsill, S. and Moulines, E. (2007), 'An overview of existing methods and recent advances in sequential Monte Carlo', *Proceedings of the IEEE* **95**(5), 899–924.
- Daley, D. and Vere-Jones, D. (2003), *An Introduction to the Theory of Point Processes*, Springer.
- Dassios, A. and Jang, J. (2003), 'Pricing of catastrophe reinsurance and derivatives using the Cox process with shot noise intensity', *Finance and Stochastics* **7**(1), 73–95.
- Davis, M. (1984), 'Piecewise-deterministic Markov processes: a general class of nondiffusion stochastic models', *J. Roy. Statist. Soc. Ser. B* **46**(3), 353–388.
- Del Moral, P., Doucet, A. and Jasra, A. (2006), 'Sequential Monte Carlo samplers', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **68**(3), 411–436.

- Gilks, W. and Berzuini, C. (2001), 'Following a moving target-Monte Carlo inference for dynamic Bayesian models', *Journal of the Royal Statistical Society: Series B (Methodological)* **63**(1), 127–146.
- Godsill, S. and Vermaak, J. (2005), 'Variable rate particle filters for tracking applications', *Statistical Signal Processing, 2005 IEEE/SP 13th Workshop on* pp. 1280–1285.
- Godsill, S., Vermaak, J., Ng, W. and Li, J. (2007), 'Models and Algorithms for Tracking of Maneuvering Objects Using Variable Rate Particle Filters', *Proceedings of the IEEE* **95**(5), 925–952.
- Lambert, D., Pinheiro, J. and Sun, D. (2001), 'Estimating Millions of Dynamic Timing Patterns in Real Time.', *Journal of the American Statistical Association* **96**(453).
- Lando, D. (1997), 'Modelling bonds and derivatives with default risk', *Mathematics of Derivative Securities* pp. 369–393.
- Ross, S. (2007), *Introduction to Probability Models*, Academic Press.
- Scott, S. and Smyth, P. (2003), 'The Markov Modulated Poisson Process and Markov Poisson Cascade with Applications to Web Traffic Modeling.', *Bayesian Statistics, 7*.
- Weinberg, J., Brown, L. and Stroud, J. (2006), 'Bayesian forecasting of an inhomogeneous Poisson process with applications to call center data', *Journal of the American Statistical Association*.
- Whiteley, N., Johansen, A. and Godsill, S. (2007), 'Monte Carlo Filtering of Piecewise Deterministic Processes', *Technical Report CUED/F-INFENG/TR-592, University of Cambridge, Department of Engineering*.