Fast Gaussian Process Methods for Point Process Intensity Estimation

John P. Cunningham JCUNNIN@STANFORD.EDU
Department of Electrical Engineering, Stanford University, Stanford, CA, USA 94305

Krishna V. Shenoy SHENOY@STANFORD.EDU
Department of Electrical Engineering and Neurosciences Program, Stanford University, Stanford, CA, USA 94305

Maneesh Sahani MANEESH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL London, WC1N 3AR, UK

Abstract

Point processes are difficult to analyze because they provide only a sparse and noisy observation of the intensity function driving the process. Gaussian Processes offer an attractive framework within which to infer underlying intensity functions. The result of this inference is a continuous function defined across time that is typically more amenable to analytical efforts. However, a naive implementation will become computationally infeasible in any problem of reasonable size, both in memory and run time requirements. We demonstrate problem specific methods for a class of renewal processes that eliminate the memory burden and reduce the solve time by orders of magnitude.

1. Introduction

Point processes with temporally or spatially varying intensity functions arise naturally in many fields of study. When the intensity function is itself a random process (often a Gaussian Process), the process is called a doubly-stochastic or Cox point process. Application domains including economics and finance (*e.g.* Basu & Dassios, 2002), neuroscience (*e.g.* Cunningham et al., 2008), ecology (*e.g.* Moller et al., 1998), and others.

Given observed point process data, one can use a Gaussian Process (GP) framework to infer an optimal estimate of the underlying intensity. In this paper we consider GP prior intensity functions coupled with point process observation models. The problem of intensity estimation then becomes a modification of GP regression and inherits the computational complexity inherent in GP methods (*e.g.* Rasmussen & Williams, 2006). The data size n will grow with the length (*e.g.* total time) of the point process. Naive methods will be $\mathcal{O}(n^2)$ in memory requirements (storing Hessian matrices) and $\mathcal{O}(n^3)$ in run time (matrix inversions and determinants). At one thousand data points (such as one second of millisecond-resolution data), a naive solution to this problem is already quite burdensome on a common workstation. At ten thousand or more, this problem is for all practical purposes intractable.

While applications of doubly-stochastic point processes are numerous, there is little work proposing solutions to the serious computational issues inherent in these methods. Thus, the development of efficient methods for intensity estimation would be of broad appeal. In this paper, we do not address the appropriateness of doubly-stochastic point process models for particular applications, but rather we focus on the significant steps required to make such modelling computationally tractable. We build on previous work from both GP regression and large-scale optimization to create a considerably faster and less memory intensive algorithm for doubly-stochastic point-process intensity estimation.

As part of the GP intensity estimation problem we optimize model hyperparameters using a Laplace approximation to the marginal likelihood or evidence. This requires an iterative approach which divides into two major parts. First, at each iteration we must find a modal (MAP) estimate of the intensity function. Second, we must calculate the approximate model evidence and its gradients with respect to GP hyperparameters. Both aspects of this problem present computational and memory problems. We develop methods to reduce the costs of both drastically.

We show that for certain classes of renewal process observation models, MAP estimation may be framed as a tractable convex program. To ensure nonnegativity in the intensity function we use a log barrier Newton method

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

(Boyd & Vandenberghe, 2004), which we solve efficiently by deriving decompositions of matrices with known structure. By exploiting a recursion embedded in the algorithm, we avoid many costly matrix inversions. We combine these advances with large scale optimization techniques, such as conjugate gradients (CG, as used by Gibbs & MacKay, 1997) and fast fourier transform (FFT) matrix multiplication methods.

To evaluate the model evidence, as well as its gradients with respect to hyperparameters, we again exploit the structure imposed by the renewal process framework to find an exact but considerably less burdensome representation. We then show that a further approximation loses little in accuracy, but makes the cost of this computation insignificant.

Combining these advances, we are able to reduce a problem that is effectively computationally infeasible to a problem with minimal memory load and very fast solution time. $\mathcal{O}(n^2)$ memory requirements are eliminated, and $\mathcal{O}(n^3)$ computation is reduced to modestly superlinear.

2. Problem Overview

Define $\mathbf{x} \in \mathbb{R}^n$ to be the intensity function (the high dimensional signal of interest); $\mathbf{x}$ is indexed by input[1] time points $\mathbf{t} \in \mathbb{R}^n$. Let the observed data $\mathbf{y} = \{y_0, \ldots, y_N\} \in \mathbb{R}^{N+1}$ be a set of $N+1$ time indices into the vector $\mathbf{x}$; that is, the ith point event occurs at time y_i, and the intensity at that time is x_{y_i}. Denote all hyperparameters by θ. In general, the prior and observation models are both functions of θ. The GP framework implies a normal prior on the intensity $p(\mathbf{x} \mid \theta) = \mathcal{N}(\mu\mathbf{1}, \Sigma)$, where the nonzero mean is a sensible choice because the intensity function is constrained to be nonnegative. Thus we treat μ as a hyperparameter ($\mu \in \theta$). The positive definite covariance matrix Σ (also a function of θ) is defined by an appropriate kernel such as a squared exponential or Ornstein-Uhlenbeck kernel (see Rasmussen & Williams, 2006, for a discussion of GP kernels). The point-process observation model gives the likelihood $p(\mathbf{y} \mid \mathbf{x}, \theta)$. In this work, we consider renewal processes (*i.e.* one-dimensional point processes with independent event interarrival times), a family of point processes that has both been well-studied theoretically and applied in many domains (Daley & Vere-Jones, 2002).

The GP prior is log concave in $\mathbf{x}$, and the nonnegativity constraint on intensity ($\mathbf{x} \succeq 0$) is convex (constraining $\mathbf{x}$ to be nonnegative is equivalent to solving an unconstrained problem where the prior on the vector $\mathbf{x}$ is a truncated multivariate normal distribution, but this is not the same as truncating the GP prior in the continuous, infinite dimensional function space; see Horrace, 2005). Thus, if the observation model is also log concave in $\mathbf{x}$, the MAP estimate $\mathbf{x}^*$ is unique and can be readily found using a log barrier Newton method (Boyd & Vandenberghe, 2004; Paninski, 2004). Renewal processes are simply defined by their interarrival distribution $f_z(z)$. A common construction for a renewal process with an inhomogeneous underlying intensity is to use the intensity rescaling $m(t_i \mid t_{i-1}) = \int_{t_{i-1}}^{t_i} x(u)du$ (in practice, a discretized sum of $\mathbf{x}$) (Barbieri et al., 2001; Daley & Vere-Jones, 2002). Accordingly, the density for an observation of event times $\mathbf{y}$ can be defined

$$\begin{aligned} p(\mathbf{y}) &= \prod_{i=1}^{N} p(y_i \mid y_{i-1}) \\ &= \prod_{i=1}^{N} |m'(y_i \mid y_{i-1})|\, f_z(m(y_i \mid y_{i-1})) \end{aligned} \quad (1)$$

by a change of variables for the interarrival distribution (Papoulis & Pillai, 2002). Since $m(t)$ is a linear transformation of the intensity function (our variables of interest), the observation model obeys log concavity as long as the distribution primitive $f_z(z)$ is log concave. Examples of suitable renewal processes include the inhomogeneous Poisson, gamma interval, Weibull interval, inverse Gaussian (Wald) interval, Rayleigh interval, and other processes (Papoulis & Pillai, 2002). For this paper, we choose one of these distributions and focus on its details. However, for processes of the form above, the implementation details are identical up to the forms of the actual distributions.

To solve the GP intensity estimation, we first find a MAP estimate $\mathbf{x}^*$ given fixed hyperparameters θ, and then we approximate the model evidence $p(\mathbf{y} \mid \theta)$ (for which we need $\mathbf{x}^*$) and its gradients in θ. Iterating these two steps, we can find the optimal model $\hat{\theta}$ (we do not integrate over hyperparameters). Finally, MAP estimation under these optimal hyperparameters $\hat{\theta}$ gives an optimal estimate of the underlying intensity. This iterative solution for $\hat{\theta}$ can be written:

$$\begin{aligned} \hat{\theta} &= \operatorname*{argmax}_{\theta} p(\theta) p(\mathbf{y} \mid \theta) \\ &\approx \operatorname*{argmax}_{\theta} p(\theta) p(\mathbf{y} \mid \mathbf{x}^*, \theta) p(\mathbf{x}^* \mid \theta) \frac{(2\pi)^{\frac{n}{2}}}{|\Lambda^* + \Sigma^{-1}|^{\frac{1}{2}}}, \end{aligned} \quad (2)$$

where the last term is a Laplace approximation to the intractable form of $p(\mathbf{y} \mid \theta)$, $\mathbf{x}^*$ is the mode of $p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$ (MAP estimate), and $\Lambda^* = -\nabla^2_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}, \theta) \mid_{\mathbf{x}=\mathbf{x}^*}$. The log concavity of our problem in $\mathbf{x}$ supports the choice of a Laplace approximation. Each of the two major steps in this algorithm (MAP estimation and model selection) involves computational and memory challenges. We address these challenges in Sections 4 and 5.

[1] In this work we restrict ourselves to a single input dimension (which we call time), as it aligns with the family of renewal processes in one-dimension. Some ideas here can be extended to multiple dimensions (*e.g.* if using a spatial Poisson process).

The computational problems inherent in GP methods have been well studied, and much progress has been made in sparsification (*e.g.* Quinonero-Candela & Rasmussen, 2005). Unfortunately, these methods do not apply directly to point process estimation, as there are no distinct training and test sets. The reader might wonder if a coarser grid would be adequate, thereby obviating the detailed methods developed here. We have found in experiments (not shown) that the sacrifice in accuracy required to allow reasonable computational tractability is large, and thus we do not consider the coarse grid a viable option. One could also consider re-expressing the problem in terms of the integrals $m(y_i \mid y_{i-1})$ appearing in Eq. 1. While this is possible in certain cases, it requires additional approximation. Finally, we note that the Laplace approximation is often inferior to Expectation Propagation (EP) (Kuss & Rasmussen, 2005) for GP methods. While many of the same techniques used here could also be used with EP, EP requires additional approximations and computational overhead. We find in experiments (not shown) that EP yields similar accuracy to the Laplace approximation in this domain, but EP incurs increased complexity and computational load.

3. Model Construction

To demonstrate our fast method, we choose the specific observation model of an inhomogeneous gamma interval process (Barbieri et al., 2001) (with hyperparameter $\gamma \in \theta$, $\gamma \geq 1$). If time has been discretized with precision Δ, this can be written

$$p(\mathbf{y} \mid \mathbf{x}, \theta) = \prod_{i=1}^{N} \left[\frac{\gamma x_{y_i}}{\Gamma(\gamma)} \left(\gamma \sum_{k=y_{i-1}}^{y_i - 1} x_k \Delta \right)^{\gamma - 1} \cdot \exp\left\{ -\gamma \sum_{k=y_{i-1}}^{y_i - 1} x_k \Delta \right\} \right], \quad (3)$$

(where we have ignored terms that scale with Δ). Let $f(\mathbf{x}) = -\log p(\mathbf{y} \mid \mathbf{x}, \theta) p(\mathbf{x} \mid \theta)$. Our MAP estimation problem is to minimize $f(\mathbf{x})$ subject to the constraint $\mathbf{x} \succeq 0$ (nonnegativity). In the log barrier method, we consider the above problem as a sequence of convex problems where we seek to minimize, at increasing values of τ, the (unconstrained) objective function

$$f_\tau(\mathbf{x}) = f(\mathbf{x}) - \sum_{k=1}^{n} \left(\frac{1}{\tau} \right) \log(x_k) \quad (4)$$

which has Hessian (positive definite by our log concave construction):

$$H = \nabla^2_{\mathbf{x}} f_\tau(\mathbf{x}) = \Sigma^{-1} + \Lambda, \quad \text{where} \quad \Lambda = B + D, \quad (5)$$

with $D = \mathbf{diag}(x_{y_0}^{-2}, \ldots, 0, \ldots, x_{y_i}^{-2}, \ldots, 0 \ldots, x_{y_N}^{-2}) + (\frac{1}{\tau})\mathbf{diag}(x_1^{-2}, \ldots, x_n^{-2})$ being positive definite and diagonal. B is block diagonal with N blocks $\hat{B}_i$:

$$\hat{B}_i = b_i b_i^T \quad \text{where} \quad b_i = \sqrt{(\gamma - 1)} \left(\sum_{k=y_{i-1}}^{y_i - 1} x_k \right)^{-1} \mathbf{1}. \quad (6)$$

B is thus block rank 1 (with the positive eigenvalue in each block corresponding to the eigenvector b_i). This matrix is key, as we exploit its structure to achieve improvements in computational performance.

4. MAP Estimation Problem

As outlined in Section 2, we first find the MAP estimate $\mathbf{x}^*$ for any model defined by hyperparameters θ. The log barrier method has the intensive requirements of calculating the objective Eq. 4, its gradient $\mathbf{g}$ (in $\mathbf{x}$), and the Newton step $\mathbf{x}_{nt} = -H^{-1}\mathbf{g}$. Each of these calculations is $\mathcal{O}(n^3)$ in run time and $\mathcal{O}(n^2)$ in memory. We show an approach that alleviates these burdens.

4.1. Finding the Newton Step $\mathbf{x}_{nt}$

First we consider the Hessian, $H = \Sigma^{-1} + \Lambda$, which itself contains the costly inverse Σ^{-1}. We would like to avoid this inversion of Σ entirely with the matrix inversion lemma (Sherman-Woodbury-Morrison formula):

$$\begin{aligned} -H^{-1} &= -(\Sigma^{-1} + \Lambda)^{-1} \\ &= -\Sigma + \Sigma R (I + R^T \Sigma R)^{-1} R^T \Sigma \end{aligned} \quad (7)$$

where R is any valid factorization such that $RR^T = \Lambda$. This decomposition preserves symmetry in the remaining matrix inverse (required for CG) and has advantageous numerical properties. With this form, instead of calculating $\mathbf{x}_{nt} = -H^{-1}\mathbf{g}$ directly, we need only multiply the rightmost expression in Eq. 7 with the gradient $\mathbf{g}$. Doing so requires the inversion $(I + R^T \Sigma R)^{-1}\mathbf{v}$ where $\mathbf{v} = R^T \Sigma \mathbf{g}$. CG allows us to avoid directly calculating matrix inverses and instead achieve the desired inversion by iteratively multiplying $(I + R^T \Sigma R)\mathbf{z}$ for different vectors $\mathbf{z}$ (Gibbs & MacKay, 1997).

It is common to precondition the CG method to reduce the number of iterations required for convergence. However, our experience with preconditioning (using both classic preconditioners and some of our own design) was that it actually degraded run-time performance. Preconditioners typically aim to improve the condition number of the Hessian, which indeed they do in this problem. However, the rapidity of CG convergence here is facilitated more by spectral concentration – many eigenvalues being equal or close to 1 – than by overall conditioning. Thus, we found it more effective to use CG inversion directly on $(I + R^T \Sigma R)$.

In general, however, finding the decomposition $\Lambda = RR^T$ is an $\mathcal{O}(n^3)$ operation, which would remove any computational benefit from this approach. For log concave renewal processes, we can derive a valid decomposition in closed form and linear computation time. Since Λ is block diagonal, we consider only one block without loss of generality. Calling this block $\hat{\Lambda}$, we know $\hat{\Lambda} = bb^T + \hat{D}$, where $\hat{D}$ is a diagonal block of the larger diagonal matrix D, and b is defined in Eq. 6. $\hat{D}$ is positive definite, so $T = \hat{D}^{-\frac{1}{2}}$ satisfies $T\hat{D}T = I$ (a similarity transform). Then, calling $\tilde{b} = Tb$, we have $T\hat{\Lambda}T = \tilde{b}\tilde{b}^T + I$. With this form, we see that the general structure of $T\hat{\Lambda}T$ is preserved under the desired matrix decomposition, up to scaling of the components:

$$(\alpha\tilde{b}\tilde{b}^T + I)(\alpha\tilde{b}\tilde{b}^T + I)^T = (\alpha^2\|\tilde{b}\|^2 + 2\alpha)\tilde{b}\tilde{b}^T + I \quad (8)$$

and we want to choose α such that (Eq. 8) equals $\tilde{b}\tilde{b}^T + I$. Using the quadratic formula to find this α, we see then that

$$\tilde{R} = \left(\frac{\sqrt{1 + \|\tilde{b}\|^2} - 1}{\|\tilde{b}\|^2}\right)\tilde{b}\tilde{b}^T + I \quad (9)$$

satisfies $T\hat{\Lambda}T = \tilde{R}\tilde{R}^T$. Since T is diagonal, it easily inverts to $T^{-1} = \hat{D}^{\frac{1}{2}}$. Then:

$$\hat{\Lambda} = T^{-1}\tilde{R}\tilde{R}^T T^{-1} = (T^{-1}\tilde{R})(T^{-1}\tilde{R})^T = \hat{R}\hat{R}^T. \quad (10)$$

To be explicit, we have found that

$$\hat{R} = \left(\frac{\sqrt{1 + \|\hat{D}^{-\frac{1}{2}}b\|^2} - 1}{\|\hat{D}^{-\frac{1}{2}}b\|^2}\right)bb^T\hat{D}^{-\frac{1}{2}} + \hat{D}^{\frac{1}{2}} \quad (11)$$

is a valid decomposition $\hat{R}\hat{R}^T = \hat{\Lambda}$. This decomposition can be seen as a partial rank-one (blockwise) update to a Cholesky factorization (Gill et al., 1974), in that $\hat{D}$ can trivially be factorized to $\hat{D}^{\frac{1}{2}}$. The final form is not, however, a Cholesky factorization, since $\hat{R}$ is not triangular (making a triangular factor would require additional computation and the explicit representation of the Cholesky matrix).

Since all of the products needed to construct $\hat{R}$ can be formed in $\mathcal{O}(m)$ time (where m is the size of the block), and since the larger matrix R can be formed by tiling the blocks $\hat{R}$, we have a total complexity for this decomposition of $\mathcal{O}(n)$. We can then use CG to find the solution to $(I + R^T\Sigma R)^{-1}(R^T\Sigma\mathbf{g})$. With this inversion calculated, we can perform the remaining forward multiplications in Eq. 7; this completes calculation of a Newton step.

In fact, we need not form the matrix R in memory. Instead, we retain each of its component elements (in Eq. 11), and reduce multiplication of a vector by R to a sequence of inner products and multiplications by diagonal matrices, all of which can be stored and calculated in $\mathcal{O}(n)$ time. Thus, we eliminate the need for $\mathcal{O}(n^2)$ storage, and we perform the relevant matrix multiplications in $\mathcal{O}(n)$ time. Since R can be multiplied in linear time, the complexity of multiplying vectors by $(I + R^T\Sigma R)$ depends on multiplying vectors by the covariance matrix Σ.

Since we have evenly spaced resolution of our data $\mathbf{x}$ in time indices t_i, Σ is Toeplitz. This matrix can be embedded in a larger circulant matrix, multiplication by which is simply a convolution operation of the argument vector with a row of this circulant matrix. Thus, the operation can be quickly done in $\mathcal{O}(n\log n)$ using frequency domain multiplications(Silverman, 1982). Further, we need never represent the matrix Σ; we only store the first row of the circulant matrix. Again we have eliminated $\mathcal{O}(n^2)$ memory needs. Other methods for fast kernel matrix multiplications include Fast Gauss Transforms (FGT) (Raykar et al., 2005) and kd-Trees (Shen et al., 2006; Gray & Moore, 2003). We note that the single input dimension (time) enables this Toeplitz structure, and thus an extension to multiple dimensions should use FGT or similar. The regular structure of the data points in any dimension make Σ multiplications very fast with such a method. Further, these methods avoid explicit representation of Σ. Here, the simple FFT approach for this one-dimensional problem significantly outperforms other (more general) methods in both speed and accuracy.

Finally, we note that the matrix $(I + R^T\Sigma R)$ is particularly well suited to CG. Although $R^T\Sigma R$ is full rank by definition, in practice its spectrum has very few large eigenvalues (typically fewer than N, the number of events). Loosely, the matrix looks like identity plus low rank. In practice, the CG method converges with high accuracy almost always in fewer than 50 steps (very often under 30). This is drastically fewer than the worst case of n steps (n of 10^3 to 10^4).

Instead of decomposing $\Lambda = RR^T$, one might have considered using the matrix inversion lemma to write $(\Sigma^{-1} + \Lambda)^{-1} = \Sigma - \Sigma\Lambda(\Lambda + \Lambda\Sigma\Lambda)^{-1}\Lambda\Sigma$. Indeed this valid form enables all of the CG and fast multiplication methods previously discussed. While it may seem that this form's ease of derivation (compared to the matrix decomposition in Eq. 11) warrants its use in general, the matrix to be inverted is poorly conditioned compared to $(I + R^T\Sigma R)$, and thus the inversion requires more CG steps. We have found in testing that the number of CG steps can roughly double. Thus, the decomposition of Eq. 11 is computationally worthwhile.

In this section, we have constructed a fast method for calculating the Newton step that costs $\mathcal{O}(n\log n)$ per CG step and incurs a very small number of CG steps. Also, we have avoided explicit representation of any matrix, so that memory requirements are only linear in the data size n, al-

lowing problem sizes of potentially millions of time steps. These two factors stand in contrast to the cubic run time and quadratic storage needs of a naive method.

4.2. Evaluating the Gradient and Objective

Calculating the objective $f_\tau(\mathbf{x})$ (Eq. 4) and its gradient (both required for the log barrier method) require finding $\Sigma^{-1}(\mathbf{x} - \mu\mathbf{1})$. Note that the kth iterate $\mathbf{x}^{(k)}$ (of the log barrier method) has the form

$$\begin{aligned}(\mathbf{x}^{(k)} - \mu\mathbf{1}) &= \mathbf{x}^{(k-1)} + t^{(k-1)}\mathbf{x}_{nt}^{(k-1)} - \mu\mathbf{1} \\ &= \sum_{j=1}^{k-1} t^{(j)}\mathbf{x}_{nt}^{(j)} + (\mathbf{x}^{(0)} - \mu\mathbf{1}) \quad (12)\end{aligned}$$

where $t^{(j)}$ and $\mathbf{x}_{nt}^{(j)}$ represent the jth iterates of the Newton step size t and the step $\mathbf{x}_{nt}$, and $\mathbf{x}^{(0)}$ is the algorithm initial point. The most logical starting point $\mathbf{x}^{(0)}$ is $\mu\mathbf{1}$, in which case the rightmost term in Eq. 12 drops out. Thus, letting $\mathbf{x}^{(0)} = \mu\mathbf{1}$ and using the form of $\mathbf{x}_{nt} = -H^{-1}\mathbf{g}$ with $-H^{-1}$ defined as in Eq. 7, we write:

$$\Sigma^{-1}(\mathbf{x}^{(k)} - \mu\mathbf{1}) = \sum_{j=1}^{k-1} t^{(j)}\Big(-\mathbf{g}^{(j)} + R^{(j)}(I + R^{(j)T}\Sigma R^{(j)})^{-1}R^{(j)T}\Sigma\mathbf{g}^{(j)}\Big). \quad (13)$$

In the earlier calculation of $\mathbf{x}_{nt}$ (Section 4.1), both of the right hand side arguments in Eq. 13 have already been found. As such, we have a recurrence that obviates the invertion of Σ, with no additional memory demands (Rasmussen & Williams, 2006).

The above steps reduce a naive MAP estimation (of any log concave renewal process) that requires cubic effort and quadratic storage to an algorithm that is modestly superlinear in run time and linear in memory requirements.

5. Model Selection Problem

Having now found $\mathbf{x}^*$ for any hyperparameters θ, the second major part of the problem is to find the negative logarithm of our approximation to the evidence $p(\mathbf{y} \mid \theta)$ in Eq. 2, and its gradients with respect to θ. The approximated log evidence can be written as:

$$\begin{aligned}&-\log\ p(\mathbf{y} \mid \theta) \approx -\log\ p(\mathbf{y} \mid \mathbf{x}^*) \\ &+\ \frac{1}{2}(\mathbf{x}^* - \mu\mathbf{1})^T\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\ +\ \frac{1}{2}\log|I + \Sigma\Lambda^*| \quad (14)\end{aligned}$$

(ignoring constants). Each of these terms has an explicit and an implicit gradient with respect to θ, where the latter result from the dependence of the MAP estimate $\mathbf{x}^*$ on the hyperparameters (such implicit gradients are typical for the use of Laplace approximation in GP learning; see Rasmussen & Williams, 2006, section 5.5.1). The implicit gradients in this problem are extremely computationally burdensome to calculate (requiring the trace of matrix inversions and matrix-matrix products for each element of $\mathbf{x}$). In empirical tests, we find implicit gradients to be quite small relative to the explicit gradients (often by several orders of magnitude). Ignoring these gradients is undesirable but essential to make this problem computationally feasible. Thus we consider only explicit gradients. This is a common approach for GP methods; see Rasmussen and Williams (2006).

Efficient computation of the first two terms of Eq. 14, as well as their gradients with respect to θ, can be achieved by the fast multiplication method and the recursion derived in Sec. 4. Specifically, the values of the first and second terms of Eq. 14 are calculated during the MAP estimation, so no additional memory or computation is necessary for them. The gradient of the first term is nonzero only with respect to γ and is linear in $\mathbf{x}$ (no matrix multiplications are required). Thus it can be quickly calculated with no additional memory demands. Computation of the gradient of the second term (the prior) can exploit the fact that we calculated $\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})$ in the final step of the MAP estimation. The gradient of this term with respect to μ is a simple inner product $\mathbf{1}^T(\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1}))$ (since we have already calculated the right side of this inner product, this computation is $\mathcal{O}(n)$ in run time and requires no additional memory). The gradient of this term with respect to a kernel hyperparameter θ_i (*e.g.* a lengthscale or variance) is:

$$\frac{d}{d\theta_i}\Big[\frac{1}{2}(\mathbf{x}^* - \mu\mathbf{1})^T\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\Big] = \frac{1}{2}\big(\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\big)^T\Big(\frac{d\Sigma}{d\theta_i}\Big)\big(\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\big). \quad (15)$$

Since we have $\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})$, this gradient only requires one matrix-vector multiplication. $\frac{d\Sigma}{d\theta_i}$ has the same Toeplitz structure as Σ and can thus be quickly multiplied. Thus, calculating the first two terms of Eq. 14 and their gradients adds no complexity to the method developed so far.

Only the term $\frac{1}{2}\log|I + \Sigma\Lambda^*|$ presents difficulty. Determinants in general require $\mathcal{O}(n^2)$ memory and $\mathcal{O}(n^3)$ solve time using a Cholesky or PLU factorization, so we must consider the problem more carefully. We examine the eigenstructure of $(I + \Sigma\Lambda^*)$. Since we are not trying to find a MAP estimate, there is no log barrier term (*i.e.* let $\tau \to \infty$); thus D (from Eq. 5) is rank N only. This means that $\Lambda^* = B + D$ (Eq. 6) is block outer product plus sub rank diagonal, so it is also rank deficient with block rank 2. Thus, it has $2N$ nonzero eigenvalues (two corresponding to each of the N events, one in each block from B and one in each block from D). Using the eigenvalue decomposition

$\Lambda^* = USU^T$, we see

$$\begin{aligned} \log|I + \Sigma\Lambda^*| &= \log|I + \Sigma USU^T| \\ &= \log|U^T||I + \Sigma USU^T||U| \\ &= \log|I + U^T\Sigma US|, \end{aligned} \tag{16}$$

since the orthogonal matrix U has determinant 1 and $U^TU = I$ by definition. Since Λ^* has rank $2N$, we know that S is diagonal with zeros on the last $n - 2N$ entries. By construction, the number of events N is much smaller than the total data size n. Since the determinant of a matrix is the product of its eigenvalues, the unit eigenvalue dimensions of $I + U^T\Sigma US$ can be ignored. We define $\overline{S}$ as the $2N \times 2N$ submatrix of S that is made up of the diagonal block with nonzero diagonal entries. Further define $\overline{U}$ as the corresponding $2N$ eigenvectors. Then, since the other dimensions of $U^T\Sigma US$ contribute nothing to the determinant, we have

$$\begin{aligned} \log|I + \Sigma\Lambda^*| &= \log|I + U^T\Sigma US| \\ &= \log|I + \overline{U}^T\Sigma\overline{U}\,\overline{S}| \\ &= \log|I + \overline{\Sigma}\,\overline{S}|, \end{aligned} \tag{17}$$

where I is now the $2N \times 2N$ identity, and we have further defined $\overline{\Sigma} = \overline{U}^T\Sigma\overline{U}$. Computationally, $\overline{\Sigma}$ is formed by multiplying Σ with the columns of $\overline{U}$. Since Λ^* is block rank 2, both matrices $\overline{S}$ and $\overline{U}$ can be found in closed form (N rank 2 eigendecompositions, one decomposition per block). This calculation of Eq. 17 requires $2N$ matrix multiplications which each have a run time cost of $\mathcal{O}(n\log n)$.

We can make a small approximation that simplifies this problem even further. Typically, N of these $2N$ eigenvalues are substantially larger than the other N. Each block of Λ^* contributes two nonzero eigenvalues. The larger is due to the diagonal entry $x_{y_i}^{-2}$ (from the matrix D) and is nearly axis aligned. The smaller eigenvalue is due to the outer product vector from block $\hat{B}_i$. Examination of the denominators in the definition of $\hat{B}_i$ and D in Eqs. 5 and 6 explains the difference in magnitude, since $x_{y_i}^2$ is much smaller than the square of sums denominator in $\hat{B}_i$. We approximate the eigenvector as the y_i axis and approximate its eigenvalue as the corresponding value in Λ^*. Then $\overline{S}$ is size $N \times N$. This savings is small, but importantly we can form $\overline{\Sigma} = \overline{U}^T\Sigma\overline{U}$ simply by picking out the N rows and columns of Σ corresponding to the event times y_i.

In this formulation, we are left with matrices of size $N \times N$ only, so we have some modest number of $\mathcal{O}(N^3)$ operations; this approach is considerably faster and scales better than the exact method above. We have also reduced $\mathcal{O}(n^2)$ storage to $\mathcal{O}(N^2)$. The following section elucidates the quality of this approximation.

To calculate the gradients with respect to this log determinant term, we also use the approximation of Eq.17. We call our approximate gradient of this term the gradient of the approximation in Eq. 17. This approximation can readily be differentiated with respect to the hyperparameters (again, typical for GP; see Rasmussen & Williams, 2006). Since these approximations are matrices in the event space N (not time space n), these gradients are quickly calculated with a handful of $\mathcal{O}(N^3)$ operations and with storage of $\mathcal{O}(N^2)$.

6. Results and Discussion

The methods developed here maintain computational accuracy while achieving massive speed-up and the elimination of memory burden. First, we have shown a fast method that achieves an accurate approximation of the MAP estimate $\mathbf{x}^*$ in much less time than a naive method. We have made all matrix multiplications implicit, thereby eliminating the memory burden of representing full matrices. We call this piece the "MAP Estimation." Second, we found the approximate model evidence, as well as its gradients, so as to perform model selection on the hyperparameters θ. These calculations, which involved the calculation of a log determinant and its gradients (Eq. 17), were achieved with matrices of significantly reduced dimension, again removing the storage demands of teh naive method. We call this piece the "log determinant approximation." These two pieces must be iterated (as described before Eq. 2) to find both the optimal model $\hat{\theta}$ and the optimal intensity $\mathbf{x}^*$. We call this iterative method (combining the two pieces above) the "full GP intensity estimation." We show here that each piece is fast and accurate, and finally that they combine to make an overall method that is considerably faster than a standard implementation, with minimal sacrifice to accuracy.

To demonstrate results, we pick six representative intensity functions, consisting of sinusoids of various amplitudes (5-100 events/second), means (15-150 events/second), frequencies (1-2 Hz), and lengths (0.5-10 seconds of millisecond resolution data, implying data sizes n of 500 to 10000). This set is by no means exhaustive, but it does indicate how this method outperforms a naive implementation in a range of scenarios. Our testing over many different intensity functions (including those in Cunningham et al., 2008) agrees with the results shown here. We simulate point process data $\mathbf{y}$ from these intensities, and we implement both the naive and the fast method on these process realizations.

All results are given for 2006era Linux (FC4) 64 bit workstations with 2-4GB of RAM running MATLAB (R14sp3, BLAS ATLAS 3.2.1 on AMD processors). The naive method was implemented in MATLAB. The fast method was similarly implemented in MATLAB with some use of the C-MEX interface for linear operations such as multiplication of a vector by the (implicitly represented) matrix R.

Table 1. Performance for fast and naive methods. Results averaged over 10 independent trials.

	Data Set					
	1	2	3	4	5	6
Data Size(n)	500	1000	1000	2000	4000	10000
Num. Events (N)[1]	20-30	30-40	140-160	55-70	55-70	140-160
MAP Estimation						
Fast Solve Time(s)	0.12	0.17	0.46	0.32	7.6	37.9
Naive Solve Time(s)	7.04	40.5	39.5	333	3704	1day[3]
Speed Up	**58**×	**232**×	**86**×	**1043**×	**493**×	**2000**×[3]
MS Error (Fast vs. Naive)[2]	4.3e-4	4.2e-4	2.1e-4	5.2e-6	6.1e-6	-
Avg. CG Iters.	6.4	5.5	16.2	8.1	29.9	49.7
Log Determinant Approximation						
Fast Solve Time(s)	6.5e-4	1.8e-3	1.9e-2	2.8e-3	2.8e-3	2.5e-2
Naive Solve Time(s)	0.24	1.02	0.97	5.7	34.7	540[3]
Speed Up	**375**×	**566**×	**52**×	**2058**×	**1.3e4**×	**2.2e4**×[3]
Avg. Acc. of Fast Approx.	99.1%	98.8%	99.8%	98.9%	99.7%	-
Avg. Model Selection Iters.	54.3	54.6	89.1	68.1	39.4	40.7
Full GP Intensity Estimation (Iterative Model Selection and MAP Estimation)						
Fast Solve Time(s)	4.4	7.1	30.3	18.7	128	423
Naive Solve Time(s)	443	3094	4548	2.4e4	1.5e5	1month[3]
Speed Up	**105**×	**451**×	**150**×	**1512**×	**1166**×	**1e4**×[3]
MS Error (Fast vs. Naive)[2]	0.10	0.03	10.8	0.01	0.01	-

[1] Entries show a range of data used.
[2] Squared norm of $x(t)$ is roughly 10^3 to 10^5, so these errors are insignificant.
[3] Unable to complete naive method; numbers estimated from cubic scaling.

First we demonstrate the utility of our fast MAP estimation method on problems of several different sizes and with different $\mathbf{x}$. We compare the fast MAP estimation to a naive implementation, demonstrating the average mean squared (MS) error (between the fast and naive estimates) and the average solve time. These results are found in the first part of Table 1. The squared norm of $\mathbf{x}$ is roughly 10^3 to 10^5, so the errors shown (the difference between the naive and fast methods) are vanishingly small. Thus, the fast MAP estimation gives an extremely accurate approximation of the naive MAP estimate. For all practical purposes, the fast MAP estimation method is exact.

The naive method scales in run time as the cube of data size n, as expected. The fast method and the speed-up factor do not appear to scale linearly in the data size. Indeed, run time depends heavily on the number of CG iterations required to solve the MAP estimation. This number of CG steps depends on problem size n, number of events N, and hyperparameters such as the lengthscale of the covariance matrix. Even so, major gains are achieved.

Second, we demonstrate our model selection accuracy and speed-up (the log determinant approximation). We run the full iterative fast method with both MAP estimation and evidence model selection. At each iterate of θ, we calculate evidence and its gradients using both the fast and naive methods. In the second section of Table 1, we show average solution times for calculating the log determinant in both naive and fast methods, and we compare their accuracy. For the sake of brevity, we demonstrate only the calculation of $\log |I + \Sigma\Lambda^*|$, not its gradients with respect to the hyperparameters. Those calculations show very similar speed-ups and are as well approximated. Thus, the log determinant is calculated to 99-100% accuracy with the naive method, and we have a highly accurate approximation.

Finally, the full intensity estimation problem requires iterative evidence calculations and MAP estimations, so we must also demonstrate the accuracy of the full fast method versus the full naive method. The last part of Table 1 shows this result (Full GP Intensity Estimation). We see that all data sets converge to quite similar results in both the fast and the naive methods, and the fast method enjoys significant speed-up. The MS errors shown compare the result

of the fast method to the result of the naive method and are very small compared to the squared norm of $\mathbf{x}$ (10^3 to 10^5).

We have demonstrated a method for inferring optimal intensity estimates from an observation of renewal process data, and we have exploited problem structure to make this method computationally attractive. As an extension, we also developed this fast GP technique for multiple observations $\mathbf{y}^{(i)}$ of the same underlying $\mathbf{x}$. It uses the same approach with comparable performance improvements. As such, we do not report it here.

Since we avoid all explicit representations of $n \times n$ matrices, our memory requirements are very minor for a problem of this size. The major run time improvements in Table 1 require effectively no loss of accuracy from an exact naive approach, and thus the additional technical complexity of this approach is well justified. Having fast, scalable methods for point process intensity estimation problems may mean the difference between theoretically interesting approaches and methods that become well used in practice.

Acknowledgments

This work was supported by NIH-NINDS-CRCNS-R01, the Michael Flynn SGF, NSF, Gatsby, CDRF, BWF, ONR, Sloan, and Whitaker. We thank Stephen Boyd for helpful discussions, Drew Haven for technical support, and Sandy Eisensee for administrative assistance.

References

Barbieri, R., Quirk, M., Frank, L., Wilson, M., & Brown, E. (2001). Construction and analysis of non-poisson stimulus-response models of neural spiking activity. *J Neurosci Methods*, *105*, 25–37.

Basu, S., & Dassios, A. (2002). A Cox process with log-normal intensity. *Insurance: Mathematics and Economics*, *31*, 297–302.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Cunningham, J. P., Yu, B. M., Shenoy, K. V., & Sahani, M. (2008). Inferring neural firing rates from spike trains using Gaussian processes. In *Advances in NIPS, 20*.

Daley, D., & Vere-Jones, D. (2002). *An introduction to the theory of point processes*. New York: Springer.

Gibbs, M., & MacKay, D. (1997). Efficient Implementation of Gaussian Processes. *Preprint*.

Gill, P., Golub, G., Murray, W., & Saunders, M. (1974). Methods for Modifying Matrix Factorizations. *Mathematics of Computation*, *28*, 505–535.

Gray, A., & Moore, A. (2003). Nonparametric density estimation: Toward computationsl tractability. *SIAM Int'l Conference on Data Mining.*.

Horrace, W. (2005). Some results on the multivariate truncated normal distribution. *J Multivariate Analysis*, *94*, 209–221.

Kuss, M., & Rasmussen, C. (2005). Assessing approximate inference for binary gaussian process classification. *Journal of Machine Learning Res.*, *6*, 1679–1704.

Moller, J., Syversveen, A., & Waagepetersen, R. (1998). Log Gaussian Cox processes. *Scandanavian J. of Stats.*, *25*, 451–482.

Paninski, L. (2004). Log-concavity results on Gaussian process methods for supervised and unsupervised learning. *Advances in NIPS*, *16*.

Papoulis, A., & Pillai, S. (2002). *Probability, random variables, and stochastic processes*. Boston: McGraw Hill.

Quinonero-Candela, J., & Rasmussen, C. (2005). A Unifying View of sparse approximate Gaussian process regression. *J. Machine Learning*, *6*, 1939–1959.

Rasmussen, C., & Williams, C. (2006). *Gaussian processes for machine learning*. Cambridge: MIT Press.

Raykar, V., Yang, C., Duraiswami, R., & Gumerov, N. (2005). Fast computation of sums of Gaussians in high dimensions. *University of Maryland Tech. Report CS-TR-4767/UMIACS-TR-2005-69*.

Shen, Y., Ng, A., & Seeger, M. (2006). Fast Gaussian Process Regression using KD-trees. *Advances in NIPS*, *18*.

Silverman, B. (1982). Kernel density estimation using the fast fourier transform. *Journal of Royal Stat. Soc. Series C: Applied Stat.*, *33*.