# Poisson Extension of Gaussian Process Factor Analysis for Modelling Spiking Neural Populations

Thesis

submitted in partial fulfillment of the requirements of the degree

Master of Science

Graduate School of Neural Information Processing

Faculty of Science
Faculty of Medicine

Eberhard-Karls-Universität Tübingen

Presented by

Hooram Nam

from Busan, South Korea
Tübingen, 31, August, 2015

Thesis Advisor: Dr. Jakob Macke

Department of
Neural Computation and Behaviour
Max Planck Institute for Biological Cybernetics, Tübingen

I affirm that I have written the dissertation myself and have not used any sources and aids other than those indicated.

I affirm that I have not included data generated in one of my laboratory rotations and already presented in the respective laboratory report.

Date/Signature:

## Contents

## 1 Abstract

Advances in multi-cell recording techniques allow neuroscientists to record neural activity from large populations of neurons. A popular approach to analysing neural population activity is to identify a smooth, low-dimensional summary of the population activity. Gaussian process factor analysis (GPFA) is a powerful technique for neural dimensionality reduction. However, it is based on a simplifying assumption that is not appropriate for neural spike trains: Spike counts conditioned on the intensity are Gaussian distributed, which fails to capture the discrete nature of neural spike counts or the mean-variance relationships typically observed in neural spike trains. Here we propose to extend GPFA using a more realistic assumption for spike generation: spike counts conditioned on the intensity are Poisson distributed, and the intensity is a non-linear function of the latent state. We present and compare two methods for performating inference of the latent state in this model, using a Laplace-approximation or variational inference. We find that both of these methods give very similar results, and that the variational algorithm is much more expensive computationally. To speed up parameter learning, we employ a stochastic version of EM, and show that it requires less computational resources when fitting the model on a large data-set. Python code implementing the method is available at `github.com/mackelab/poisson-gpfa`.

## 2 Introduction

Neuroscience has been undergoing an information explosion with modern experimental techniques allowing unprecedented insights into the structure and function of neural circuits. These advances open the possibility of studying the statistical structure of neural activity in large populations of neurons, and of using these insights in clinical applications such as neural prosthetics. However, understanding the complex data generated by neurophysiological experiments is a challenging task that requires powerful statistical methods.

Among the various neural data made available by technologies such as EEG, MEG and fMRI, of particular interest is the spiking activities of neural populations. Using micro multi-electrodes such as tetrodes and Utah arrays, researchers can gain insights into information processing in many neurons' simultaneous activities at the millisecond resolution. These new data have aided breakthroughs in neural computation and neural prosthetics, partly owing their success to advances in dimensionality reductions techniques (Cunningham and Byron, 2014).

Dimensionality reduction has been a popular analysis tool that offers insights into information processing in large populations of recorded neurons. Low dimensional representations of the activities of many neurons provide means to visualize the internal dynamics linked to observed behaviour. By obtaining a low dimensional summary of the recorded neural population's activity on a trial-to-trial basis, one can meaningfully assess the effects that experimental conditions have on the neural activity. This is particularly relevant in cognitive tasks such as motor planning and decision making, where the neural responses are more a reflection of the internal states than external stimuli (Churchland et al., 2007).

Probabilistic latent state-space models are well suited for the task of extracting low dimensional representation of the high dimensional neural spiking activities on a trial-by-trial basis (Paninski et al., 2010). The general framework behind latent state-space models is the assumption of the latent, unobserved variables that influence the outcomes of the observed variables. In latent state-space models, one must at large make two assumptions: 1) about the process that governs the dynamics of the latent state, and 2) about the way in which the latent states influce the observed spike counts. The second assumption is sometimes called the 'noise' model, as the observed spike counts are modelled to be samples from some probability distribution conditioned on the latent state, and the choice of this spike generation process influences the trial-to-trial variability in the sampled data. Depending on the assumptions regarding the evolution of the latent state and the observation process linking the latent to observed space, the model can capture various statistics of the observed data. Such probabilistic latent state-space models are particularly relevant regarding the view that neural population receive 'common inputs' from unobserved (latent) parts of the network (Kulkarni and Paninski, 2007; Vidne et al., 2012).

Various state-space models exist for the task of neural dimensionality reduction. The current state-of-the-art method is Gaussian process factor analysis (GPFA), where the evolution of the latent states is assumed to be a Gaussian process. In addition, the model assumes that observed spike counts conditioned on the latent state is Gaussian distributed (Byron et al., 2009). While GPFA is powerful and popular, it rests upon an inappropriate assumption for real spike trains; conditioned on the latent state, GPFA assumes Gaussian distributed spike counts, which allows negative, non-integer spike counts to be sampled from the model. In addition, variance of real spike counts increases with the mean, a property not captured by the Gaussian assumption for spike generation (Figure 1).

The GPFA model can be extended to capture the aforementioned properties of real spike counts by replacing the Gaussian assumption for spike generation with a Poisson process, whose intensity is a non-linear function of the latent state. The new model with Poisson spike generation process can be fit using the expectation-maximization (EM) algorithm. Unlike GPFA, the posterior distribution has no closed form expression. Thus we approximate the posterior with a Gaussian distribution using 1) Laplace approximation and 2) dual variational inference as outlined in (Emtiyaz Khan et al., 2013). In addition, we implemented a

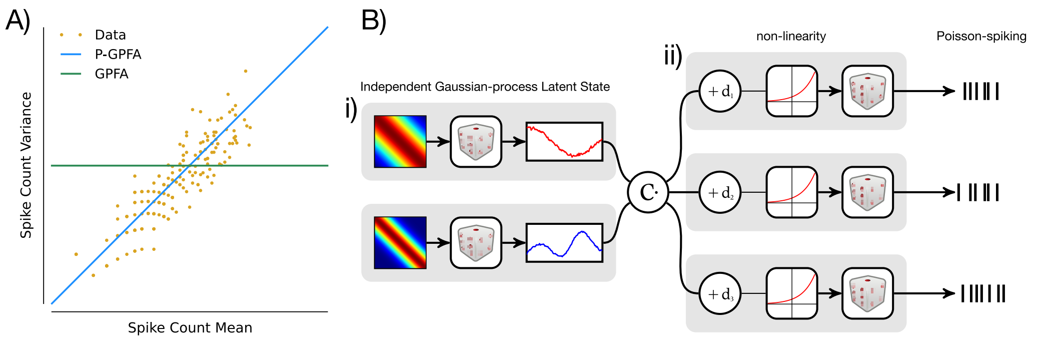

*Figure 1:* ***A) Motivation.*** *Variance of a neuron's spike counts increases with firing rate (brown, data from (Stevenson et al., 2011)). In GPFA, spike counts conditioned on intensity are assumed to be Gaussian distributed, which does not capture this mean-var relationship (green). P-GPFA, on the other hand, assumes the spike counts to be Poisson distributed, and does capture the increasing variance of real neurons as a quadratic function of the mean (blue) (equation 39) .* ***B) Generative Model.*** *The model assumes a doubly stochastic process for spike generation. We first draw entire time-courses of independent Gaussian process latent states with a squared exponential kernel (i). At each time point $t$, the $p$ dimensional latent state $x$ is projected to the observed space via the parameter $C$. Then a bias term $d$ is added, controlling the mean firing rates of the neurons. The sum $Cx+d$ is then passed through the exponential nonlinearity, which serves as the intensity of the Poisson process that generates observed spike counts (ii).*

stochastic version of EM to speed up the process of model parameter estimation. We will present a performance comparison of the two inference methods, as well as a comparison of a number of EM variants made available by the combination of inference methods and stochastic/full EM. Finally, we will show a comparison of GPFA and P-GPFA on neurophysiological data from (Stevenson et al., 2011).

## 3 Methods

### 3.1 Model Specification

In this section, we describe a model for neural populations' spike count data that we call *Poisson observation Gaussian process factor analyzers* (P-GPFA) model. Neural spike counts are denoted $y^k_{it}, i = 1 \dots q, t = 1 \dots T$ from $q$ neurons and $T$ time bins on trial $k$, $k = 1 \dots N$. Thus, $y^k_{i,:}$ is the vector of spike counts of neuron $i$ across time, and $y^k_{:,t} = y^k_t$ the vector of spike counts across the population at time $t$. We will use the notation $\mathbf{y}^k = \text{vec}(y^k)$ to denote the vector of size $qT \times 1$ in which the first $q$ entries are $y^k_{:,1}$, and also drop the index $k$ when it gets too cluttered. We will denote the set of observed spike counts $D = \{\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^N\}$.

The hidden state of the population is modelled to be a $p$-dimensional Gaussian process $x$. A priori, the dimensions are independent, and dimension $i$ has covariance $K_i$. We denote by $x^k_{it}$ the state of the $i$th dimension of the GP at time $t$ of trial $k$, by $x^k_{i,:}$ the evolution of state $i$ across time, and by $x^k_{:,t} = x^k_t$ the multivariate state at time $t$. The Gaussian process of $i$th latent dimension is specified by the squared exponential kernel,

$$K_i(t_1, t_2) = \sigma_s^2 \exp\left(\frac{-(t_1 - t_2)^2}{2\tau_i^2}\right) + \sigma_n^2 \delta_{t_1,t_2}, \tag{1}$$

with the timescale constant $\tau_i$. The GP noise variance $\sigma_n^2$ is set to $1 \times 10^{-3}$ and $\sigma_s^2 = 1 - \sigma_n^2$ as is the case in (Byron et al., 2009). The entire time course of the $i$th latent state is drawn from the Gaussian process

$$x^k_{i,:} \sim \mathcal{GP}(0, K(\tau_i)). \tag{2}$$

We also use the notation $\mathbf{x}^k = \text{vec}(x^k)$ to represent the whole latent process in a vector. Here, the first $p$ entries of $\mathbf{x}^k$ are $x^k_{:,1}$. Hence, the prior covariance $K$ over $\mathbf{x}$ has a 'stripe' structure: It consists of $T \times T$ diagonal blocks $D^{s,t}$, where $D^{s,t}_{ii} = K_i(s,t)$. If all processes have the same kernel $K_1$, then $K = K_1 \otimes \mathbb{I}_p$. Conditioned on the latent states at time $t$, the spike counts are assumed to be Poisson distributed:

$$y^k_{i,t} | x^k_{:,t} \sim \mathcal{P}\left(y^k_{i,k} | \exp\left(C_i x^k_{:,t} + d_i\right)\right). \tag{3}$$

Here, $C$ is a $q \times p$ matrix representing the mapping from the latent state up to the spike-rates, and $d$ is a $p$ dimensional vector of bias terms, which controls the mean firing rates of the neurons. Thus the model is fully specified by the parameter set $\theta = \{\tau, C, d\}$.

## 3.2 Inference

The problem of inference is concerened with estimating the latent states given the observed neural activity. Here, we are not only interested in finding the most likely sequence of latent states, but also the uncertainty associated with the estimated states. In this regard, we seek the full posterior *distribution* over $\mathbf{x}$, given the spike count data $\mathbf{y}$ and a guess of parameters $\theta = \{C, d, \tau\}$. The exponential nonlinearity in the intensity of the Poisson process renders the posterior distribution $p(\mathbf{x}|\mathbf{y})$ non-Gaussian, however it is unimodal and log-concave (Boyd and Vandenberghe, 2004). Therefore, there is reason to be optimistic that a Gaussian approximation would work well; we will present two methods of approximating the posterior with a Gaussian distribution, $p(\mathbf{x}|\mathbf{y}) \approx q(\mathbf{x}|\mu, \Sigma) = \mathcal{N}(\mathbf{x}|\mu, \Sigma)$, where we denote the approximate posterior by $q(\mathbf{x}|\mu, \Sigma)$. The first method is Laplace approximation, a simple heuristic scheme where we set the mode of the (log) posterior as the mean $\mu$ of the approximate distribution. In the second method, we find $\mu$ and $\Sigma$ of $q(\mathbf{x})$ that minimizes the distance between the true posterior and the approximate Gaussian distribution as measured by Kullback-Leibler Divergence (Emtiyaz Khan et al., 2013).

### 3.2.1 Inference using Laplace Approximation

To find the mode of the log-posterior, we run a gradient based optimization algorithm on the unnormalized log-posterior given by:

$$\begin{aligned}
L(\mathbf{x}^k|\mathbf{y}^k;\theta) &:= \log p(\mathbf{x}^k|\mathbf{y}^k;\theta) & (4)\\
&= \log p(\mathbf{y}^k|\mathbf{x}^k;\theta) + \log p(\mathbf{x}^k;\theta) + const & (5)\\
&= \sum_{i,t} \log \mathcal{P}\left(y^k_{it} | \exp((Cx^k_t)_i + d_i)\right) + \log \mathcal{N}\left(\mathbf{x}^k|0, K\right) + const & (6)\\
&= \mathbf{e}^\top_{(qT)} \exp\left(\tilde{C}\mathbf{x}^k + \mathbf{d}\right) - \mathbf{y}^{k\top}\left(\tilde{C}\mathbf{x}^k + \mathbf{d}\right) + \frac{1}{2}\mathbf{x}^{k\top} K^{-1} \mathbf{x}^k + const, & (7)
\end{aligned}$$

with exp on matrices applied elementwise, $\mathbf{e}_{qT}$ is a vector of $qT$ ones, $\tilde{C} = \mathbb{I}_T \otimes C$, and $\mathbf{d} = \mathbf{e}_T \otimes d$. Having obtained the gradient and Hessian of this quantity, we can speed up the optimization process using the first and second order information. We set the mean $\mu$ of the approximate distribution $q(\mathbf{x})$ as the mode of the log-posterior, and set the covariance $\Sigma$ of $q(\mathbf{x})$ as the negative inverse Hessian of the log-posterior evaluated at the mode. The mean and covariance of the approximate posterior $q(\mathbf{x}^k|\mu^k, \Sigma^k)$ of trial $k$ are thus

$$\begin{aligned}
\mu^k &= \underset{\mathbf{x}^k}{\text{argmin}}\left(-L(\mathbf{x}^k|\mathbf{y}^k;\theta)\right) \text{ and} & (8)\\
\Sigma^k &= -\nabla^2_{\mathbf{x}} \log L(\mathbf{x}^k|\mathbf{y}^k;\theta)|_{\mathbf{x}^k=\mu^k}. & (9)
\end{aligned}$$

### 3.2.2 Gaussian Variational Inference

While Laplace approximation allows easy and efficient calculation of the approximate posterior, it has been shown that other principled methods of approximation provide more accurate approximations of the posterior (Nickisch and Rasmussen, 2008). One such method is Gaussian variational inference (Emtiyaz Khan et al., 2013), where we find the approximate posterior mean $\mu$ and $\Sigma$ that maximizes a tight lower bound on the log marginal likelihood. Such a lower bound is given by Jenson's inequality:

$$\mathcal{L}(\mathbf{y},\theta) := E_{q(\mathbf{x})}\left[\log \frac{\prod_k p(\mathbf{y}_k|\mathbf{x}_k;\theta)p(\mathbf{x};\theta)}{q(\mathbf{x};\theta)}\right] \leq \log p(\mathbf{y}) \tag{10}$$

We write the lower bound as the sum of two terms and maximize it over $\mu$ and $\Sigma$.

$$\mathcal{L}(\mu,\Sigma,\mathbf{y},\theta) = -D_{KL}[q(\mathbf{x};\mu,\Sigma)||p(\mathbf{x};\theta)] - \sum_{k=1}^{N} \mathbf{E}_{q(x_k;\mu,\Sigma)}\left[-\log p(\mathbf{y}^k|\mathbf{x}^k;\theta)\right] \tag{11}$$

$$\hat{\mu},\hat{\Sigma} = \underset{\mu,\Sigma}{\operatorname{argmax}}\, \mathcal{L}(\mu,\Sigma,\mathbf{y},\theta). \tag{12}$$

The two terms in equation 11 are given by (up to additive constants):

$$D_{KL}[q(\mathbf{x})||p(\mathbf{x})] = \frac{1}{2}\left(\log|\boldsymbol{\Sigma}| - \operatorname{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{K}) - \boldsymbol{\mu}^\top\mathbf{K}^{-1}\boldsymbol{\mu}\right) \tag{13}$$

$$\mathrm{E}_{q(\mathbf{x})}[\log P(\mathbf{y}|\mathbf{x})] = \mathbf{y}^\top\left(\tilde{C}\mu + \tilde{\mathbf{d}}\right) - \mathbf{e}_{(qT)}^\top \exp\left(\tilde{C}\mu + \tilde{\mathbf{d}} + \frac{1}{2}\mathbf{diag}(\tilde{C}\Sigma\tilde{C}^\top)\right). \tag{14}$$

One can in principle optimize $\mathcal{L}$ directly over the posterior mean and covariance $\mu,\Sigma$. However this is computationally expensive because $\Sigma$ has $\mathcal{O}(p^2T^2)$ entries and that it has to be constrained to be a symmetric, semi-definite matrix. Fortunately we can circumvent direct optimization over $\mu$ and $\Sigma$ by framing the objective as a dual optimization problem that exploits convexity property of the variational Gaussian approximations (Emtiyaz Khan et al., 2013). In the reduced dual problem, we minimze the objective function $\mathcal{D}(\lambda)$ over $\lambda \geq 0$:

$$\lambda^* = \underset{\lambda\geq 0}{\operatorname{argmin}}\, \mathcal{D}(\lambda) \tag{15}$$

$$\mathcal{D}(\lambda) = \frac{1}{2}(\lambda - \mathbf{y})^\top\tilde{C}K\tilde{C}^\top - (\tilde{C}\mu + \tilde{\mathbf{d}})^\top(\lambda - \mathbf{y}) \tag{16}$$

$$+ \frac{1}{2}\log|\Sigma(\lambda)| + \sum_{n=1}^{qT}\lambda_n(\log\lambda_n - 1). \tag{17}$$

$\lambda$ is a variable of size $\mathcal{O}(qT)$, as opposed to $\mathcal{O}(q^2T^2)$ for optimizing directly over $\mu$ and $\Sigma$. We used a pseudo-Newton method lBFGS (Boyd and Vandenberghe, 2004) to find the optimum of equation 15. The posterior mean and covariance as a function of the dual variable $\lambda$ are given as:

$$\Sigma^{-1}(\lambda) = K^{-1} + \tilde{C}\operatorname{diag}(\lambda)\tilde{C}^\top \tag{18}$$

$$\mu(\lambda) = -K\tilde{C}^\top(\lambda - \mathbf{y}). \tag{19}$$

## 3.3 Learning

### 3.3.1 Full EM

To learn the model parameters given observed spike counts, we implemented variants of EM (expectation maximization). In general, EM is an iterative method that finds the maximum likelihood estimate of the parameter of a latent-variable model. In the E-step, we hold the model parameters $\theta$ fixed and optimize the lower bound over the posterior distributions $p(\mathbf{x}|\mathbf{y})$ [sec 3.2]. In the M-step, we hold the latent variables fixed and optimize the lower bound over the model parameters $\theta$. Repeated application of the two steps gives a sequence of parameters that converge to a maximum of the log-likelihood. The algorithm is however susceptible to finding local optimum, thus we implemented a sensible parameter initialization using Poisson PCA. Typically, in standard EM, the algorithm processes inference on all of available trials in each iteration. This process can be computationally demanding on data with many trials, especially using Variational inference [fig 3F]. Thus we implemented stochastic EM, where we only process a smaller subset of available trials in each iteration.

The goal is to maximize the marignal log likehood $\log p(\mathbf{y})$ over the model parameters $\theta$. We instead maximize the same lower bound as in equation 10, which can alternatively be written as

$$\mathcal{L}(\theta) = \mathrm{E}_{q(\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{y}) - \log q(\mathbf{x})\right]. \tag{20}$$

Given a posterior approximation $q(\mathbf{x})$ either using Laplace or Variational inference, $\mathcal{L}(\theta)$ can be written as:

$$\begin{aligned}
\mathcal{L}(\theta) &= \sum_{k=1}^{N}\int q(\mathbf{x}^k)\log P(\mathbf{x}^k,\mathbf{y}^k;\theta)d\mathbf{x}^k + \mathrm{const}(\theta) && (21)\\
\mathcal{L}(\theta) &= \sum_{k=1}^{N}\int q(\mathbf{x}^k)\log P(\mathbf{y}^k|\mathbf{x}^k;\theta)d\mathbf{x}^k + \int q(\mathbf{x}^k)\log P(\mathbf{x}^k|\theta)d\mathbf{x}^k + \mathrm{const}(\theta) && (22)\\
&=: \mathcal{L}_{obs}(C,d) + \mathcal{L}_{GP}(\tau) + \mathrm{const}(\theta). && (23)
\end{aligned}$$

Hence the M-step cost function can be expressed as the sum of two cost functions that we can optimize separately. The first term corresponds to the parameters related to the Poisson observation process, namely the latent-to-observed-space mapping $C$ and the bias $d$ for the mean firing rates of neurons. The second term corresponds to the latent Gaussian process with kernel parameter $\tau$.

Dropping constants which are independent of $\theta$ and using our approximation to the posterior, this can be written as

$$\begin{aligned}
\mathcal{L}_{obs}(C,d) &= \sum_k \mathbf{e}_{(qT)}^{\top}\exp\left(\mathbf{d} + \tilde{C}\mu^k + \frac{1}{2}\mathrm{diag}^{-1}\left(\tilde{C}\Sigma^k\tilde{C}^{\top}\right)\right) - \mathbf{y}^{k\top}\left(\tilde{C}\mu^k + \mathbf{d}\right). && (24)\\
\mathcal{L}_{GP}(\tau) &= \frac{N}{2}\mathrm{logdet}(K) + \frac{1}{2}\mathrm{tr}\left(K^{-1}\sum_k\left(\Sigma^k + \mu^k\mu^{k\top}\right)\right). && (25)\\
&=: \mathcal{L}(C,d) + \mathcal{L}(\tau). && (26)
\end{aligned}$$

Thus the estimated model parameter $\theta^n = \{C^n, d^n, \tau^n\}$ of the $n$th iteration is

$$\begin{aligned}
C^n, d^n &= \underset{C,d}{\mathrm{argmax}}\,\mathcal{L}_{obs}(C,d) && (27)\\
\tau^n &= \underset{C,d}{\mathrm{argmax}}\,\mathcal{L}_{obs}(C,d) && (28)
\end{aligned}$$

Optimizing the cost function for $\tau$ (equation 25) is entirely analogous to GPFA (Byron et al., 2009).

### 3.3.2 Stochastic EM

To fit the model using EM on data with many trials, performing inference on all available trials in each iteration can be infeasible. To speed up the fitting process on such data, we devised a number of stochastic variants of EM. In our methods, the stochasticity comes from subsampling of available trials in each iteration. The variants differ in ways the algorithms 'remember' previously processed trials. In the stochastic setting, without a mechanism to remember the learned parameters between iterations, the final estimated parameters will be overfit to the batch of trials processed in the last iteration. Various algorithms for stochastic online EM in similar context have been previously studied. In the following sections, we will discuss some of the methods presented in (Cappé and Moulines, 2009; Lange, 1995; Sato, 1999; Titterington, 1984).

In our stochastic setting, each iteration of EM only processes a batch of $N^* < N$ trials instead of all available trials. Denote the set of subsamples in each iteration by $D^* := \{\mathbf{y}^{*1}, \mathbf{y}^{*2}, \ldots, \mathbf{y}^{*N^*}\} \subset D$. In the E-step, we perform inference on the selected $N^*$ trials and obtain the posterior $q(\mathbf{x}^k|\mu^k, \Sigma^k)$. The lower bound on marginal likelihood $\mathcal{L}^*(\theta)$ in the stochastic M-step is calculated over the subsampled $N^*$ trials:

$$\mathcal{L}^*(\theta) = \sum_{\mathbf{y}^k \in D^*} q(\mathbf{x}^k) \log P(\mathbf{x}^k, \mathbf{y}^k; \theta) d\mathbf{x}^k + const(\theta). \tag{29}$$

### 3.3.2.1 Newton's Method

Taking the cues from (Cappé and Moulines, 2009; Lange, 1995; Titterington, 1984), we replace the M-step of the EM algorithm by Newton update:

$$\hat{\theta}_n = \hat{\theta}_{n-1} - \gamma_n \left[\nabla^2_\theta \mathcal{L}^*(\theta|\theta_{n-1})|_{\theta=\theta_{n-1}}\right]^{-1} \nabla_\theta \mathcal{L}^*(\theta|\theta_{n-1})|_{\theta=\theta_{n-1}}. \qquad \textbf{`grad'} \text{ method} \tag{30}$$

where $\gamma_n$ is a step size ($\gamma = 1$ corresponding to the actual Newton update). The gradient of the lower bound $\mathcal{L}^*$ is available in closed form, while we use 4th order central difference method to approximate the Hessian. To ensure convergence, we use a decreasing sequence of step sizes $\{\gamma_n\}$ such that $\sum_{n=1}^{\infty} \gamma_n = \infty$ (Cappé and Moulines, 2009; Sato, 1999).

### 3.3.2.2 Proximal Algorithm

We consider the application of proximal algorithms (Parikh and Boyd, 2013) to the optimization problem at hand. Much like Newton's method is a standard tool in optimization, proximal algorithms can be viewed as an analogous tool for such problems. At the base, the algorithms consist of repeated applications of the proximal operator

$$\mathbf{prox}_f(\nu) = \underset{x}{\operatorname{argmin}} \left(f(x) + (1/2)||x - \nu||^2\right). \tag{31}$$

The definition indicates that $\mathbf{prox}_f(\nu)$ is a point that finds a compromise between minimizing $f$ and being near to $\nu$. Applying this idea to our problem, we consider the M-step as the evaluation of the proximal operator $\mathbf{prox}_{\mathcal{L}^*}(\theta_{n-1})$. Thus we minize the cost function $\mathcal{L}^*$ while panelizing the distance between subsequent estimates of the model parameters $\theta$. The squared norm is defined with respect to a positive semidefinite cone $\Sigma_\pi$. This is equivalent to assuming a Gaussian prior $\theta_n \sim \mathcal{N}(\theta_{n-1}, \Sigma_{\pi,\theta})$ on the cost function $\mathcal{L}^*$ in the $n$th iteration. The regularizer panelizes the distance between the previous estimate $\theta_{n-1}$ and the current estimate $\theta_n$. By following a scheduled learning rate $\lambda_n$ with $\lambda_n > 0$ and $\sum_{n=1}^{\infty} \lambda_n = \infty$, we gradually increase this panelty over the iterations to ensure the convergence of parameters (Parikh and Boyd, 2013). The new parameter update equation in $n$th iteration is thus:

$$\theta_n = \underset{\theta}{\operatorname{argmax}} \left(\mathcal{L}(\theta|\theta_{n-1}) - \frac{1}{2}\lambda_n(\theta - \theta_{n-1})^\top \Sigma_{\pi,\theta}^{-1}(\theta - \theta_{n-1})\right). \tag{32}$$

For the choice of $\Sigma_{\pi,\theta}$, we implemented two options:

$$\Sigma_{\pi,\theta} = \mathbf{I}_\theta \qquad \text{- } \textbf{`diag'} \text{ method} \tag{33}$$

$$\Sigma_{\pi,\theta} = -\left(\nabla^2 \tilde{L}(\theta)|_{\theta=\theta_{n-1}}\right)^{-1}. \qquad \text{- } \textbf{`hess'} \text{ method} \tag{34}$$

### 3.4 Initialization by Poisson-PCA

From (Buesing et al., 2012), assuming $y|z \sim \mathcal{P}(y|exp(z))$, the covariance of $z$ is given by

$$\Sigma_z = \log(\Sigma_y + \mu_y\mu_y^\top - \text{diag}(\mu_y)) - \log(\mu_y\mu_y^\top), \tag{35}$$

where $\Sigma_y$ is the empirical covariance matrix of the raw spike counts of all trials, and $\mu_y$ is the empirical mean vector of the spike counts of the neurons across all trials. To initialize $C$, we perform PCA on $\Sigma_z$ and set $d = \log(\mu_y)$.

## 4 Results

### 4.1 Results on Simulated Data

#### 4.1.1 Comparison of Inference Methods

We compared the two methods of posterior approximation (sections 3.2.1 and 3.2.2) on simulated data. First we simulated a trial with 3 latent dimensions, 20 neurons lasting 1 second binned at 10 ms resolution. The approximated posterior means of the two methods are similar (fig 2A). Using the true parameter, the inferred posterior covariances are similar to each other, as well as to the covariance of 10000 MCMC samples (Murray et al., 2009) (fig 2B).

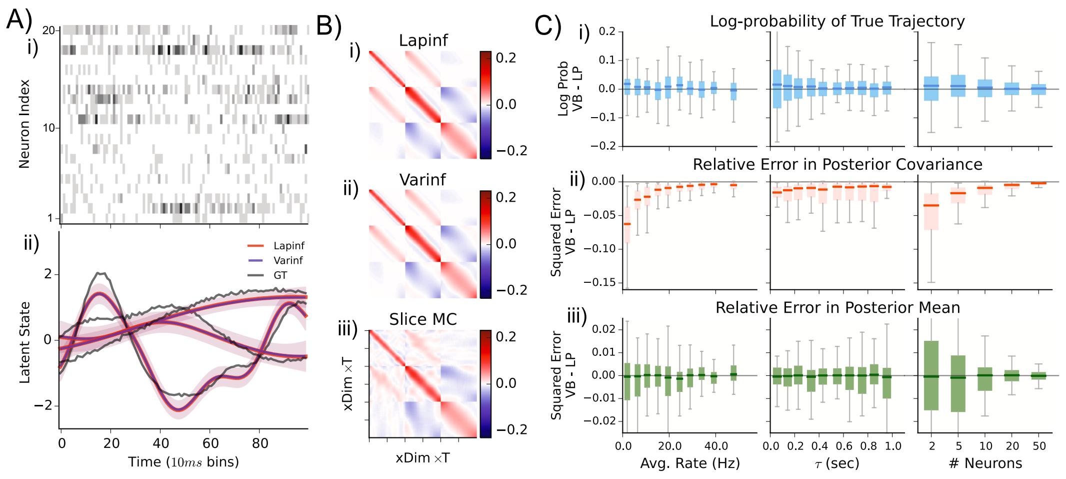

*Figure 2:* ***Variational and Laplace inference give similar results. A) Simulated spike counts and state-reconstruction.*** *i) Simulated spike raster from 20 neurons. ii) Three true state-trajectories (gray) and posterior mean estimates of two inference methods. Both variational (dashed blue) and Laplace approximations (red) have similar reconstructions of the mean trajectory. Shaded regions indicate estimated posterior uncertainty around the mean, and they are also vary similar in this case.* ***B) Posterior Covariance.*** *Inferred posterior covariance of the two methods are nearly identical to each other (i,ii), and are similar to 10,000 elliptical slice sample covariance (iii).* ***C) Performance of the two methods as a function of experiment parameters.*** *We simulated 1000 trials of varying firing rates, latent timescale, and number of neurons. Probabilities of the true trajectory under the two posterior assumptions have difference centered at zero across experiment settings (i). We computed the normalized error of the posterior covariance and mean relative respectively to the slice sample covariance (ii) and true trajectories (iii). There are systematic differences in posterior covariance estimates of the two methods (ii).*

To look for systematic differences between the two inference methods, we compared the quality of the variational and Laplace approximations on 1000 simulated trials. The trials had firing rates ranging between 1Hz and 50Hz, number of neurons $q$ ranging between 2 and 50, and the latent Gaussian process time scale $\tau$ varying between 0 (exclusive) and 1 second (fig 2C). We compared the quality of the two methods as a function of $q, \tau$ and population average firing rate. As a measure for assessing approximation quality, we first evaluated the log-probability $\log q(x_{1:T})$ of the true (unobserved) latent trajectory $x_{1:T}$ under the two different approximate posteriors (fig 2C, first row). That is,

$$\begin{aligned}\text{log Prob VB-LP} &= \log P(\mathbf{x}_{GT}|\mu_{VB}, \Sigma_{VB}) - \log P(\mathbf{x}_{GT}|\mu_{LP}, \Sigma_{LP}) && (36)\\ &= \log \mathcal{N}(\mathbf{x}_{GT}|\mu_{VB}, \Sigma_{VB}) - \log \mathcal{N}(\mathbf{x}_{GT}|\mu_{LP}, \Sigma_{LP}). && (37)\end{aligned}$$

For a good posterior approximation, the true trajectories have higher probability than under a poor one. The two methods produce difference of probability centered at zero across the test variables. Second, we calculated the squared error difference between the two posterior covariance approximations relative to the MCMC sample covariance (fig 2C second row). On trials with low firing rate and smaller number of neurons, the variational method performed better at inferring posterior covariance. Third, we compared the square error difference between the two posterior mean approximations relative to the ground truth latent trajectories (fig 2C, bottom row). Across the tested variables, the difference between the errors of the two methods are centered at zero. We conclude that both are effect methods for state inference, however, given the lower computational cost of Laplace approximation, it may in many cases be preferable to variational inference.

### 4.1.2 Comparison of Parameter Learning Methods

We tested the performance of various fitting options made available by combinations of Laplace/variational inference methods (E-step), full/stochastic parameter update methods (M-step), and random/P-PCA intialization methods (fig 3). 100 datasets were generated, each with 40 trials lasting 400ms binned at 20ms, 2 latent states, and 30 neurons. For stochastic EM, we performed inference on five trials in each iteration of EM and only show results using the **'diag'** method. We used a number of model fitting methods on each of 100 datasets and compared the fits on a number of error measures. The quality of estimated $C$ can be accessed by the angle between the subspaces spanned by columns of true and estimated $C$ (fig 3A). In addition, we have analyitical expressions for the mean and covariance of the spike counts, which we can compare to the sample mean and covariance of the data (fig 3B,C). We plotted the squared error between the true and estimated spike count mean/covariance. Given a set of model parameters $C, d$, and $\tau$, the expected spike count mean and covariance are given by the equations (Buesing et al., 2012):

$$\mathrm{E}[y|C, d] =: m = \exp\left(\frac{1}{2}\mathrm{diag}(CC^\top) + d\right) \tag{38}$$

$$\mathrm{Cov}[y|C, d] = mm^\top \exp(CC^\top) + \mathrm{diag}(m) - mm^\top. \tag{39}$$

Another measure of the quality of a model fit is the leave-one-out prediction error on the training set. Given a set of estimated model parameters, we can perform inference on a given trial data with a given neuron's activity removed. Using the inferred latent trajectory and the estimated model parameters, we can predict the Poisson intensity of the removed neuron at a given time. The squared error between the predicted intensity and the ommited neuron's actual activity is summed across neurons and trials. This error is compared between the fitting methods, relative to the prediction error of the ground truth parameters (fig 3D). In addition, we compared the lower bound $\mathcal{L}$ on the marginal log likelihood across the fitting methods, relative to that of the ground truth parameters (fig 3E,F). We included the constant terms of $\mathcal{L}$ that do not depend on $\theta$.

In our experiment, the full version of EM generally performs better than the stochastic variant across all measures of fit. Moreover, performing Variational inference in the E-step

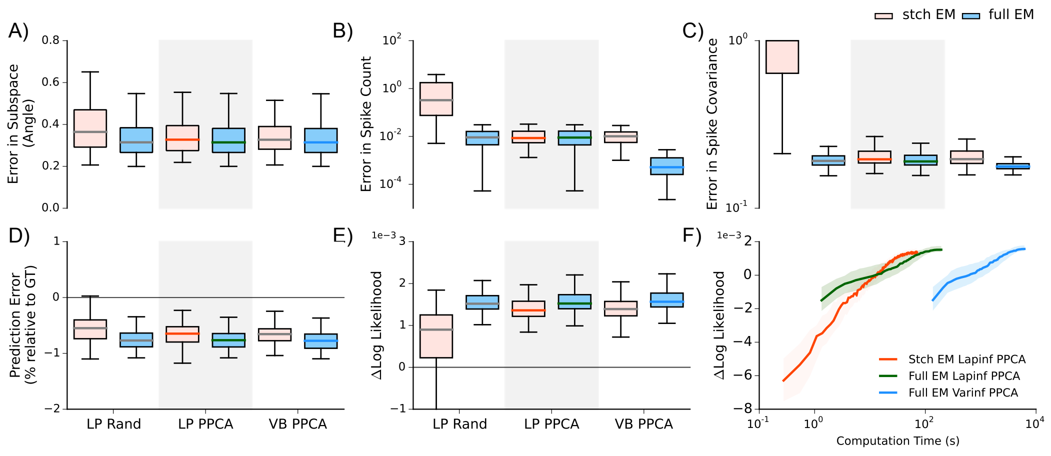

*Figure 3:* ***Comparison of proposed fitting algorithms:*** *We tested the proposed fitting methods on 100 simulated datasets (40 trials/dataset, 400 ms trial duration, 20 ms bin size, 2 latent states, 30 neurons, 250 EM iterations).* ***A-E)*** *Various error measures of optimal fits learned by proposed fitting methods. The default fitting method is highlighted with red median lines. Theoretically best performing fitting method is highlighted with blue median lines.* ***F)*** *Course of model log likelihood over the EM iterations plotted against computation time. Computation time required for the default method is 2 orders of magnitude less than the best method, for comparatively small sacrifice in performance.*

leads to better a fit than using Laplace inference, despite the apparent similarities in mean and covariance of inferred trajectories. The best performing method is to use variational inference and to perform full EM. With Laplace inference, initialization using P-PCA affects the performance of the stochastic method but has no significant effect in full EM.

We assessed the relationship between the required computation time and the fit performance measured by the marginal lower bound (fig 3F). In full EM, using Laplace inference (green) is about two orders of magnitude faster than variational inference (blue). Using either methods of inference, the implemented stochastic EM brings a speed up by the factor (# available trials)/(# trials batch) in comparison to full EM, at the cost of performance loss shown in figure 3.

## 4.2 Results on Neurophysiological Data

In this section, we demonstrate statistics of neural data that GPFA and P-GPFA capture. Analysis was performed on data from (Stevenson et al., 2011), containing activities of 201 neurons in M1 of a monkey. The data was collected while the monkey performed trials of center-out reach task. After cutting the trials to 2000 ms and counting the spikes in 100 ms bins, we fit both GPFA and P-GPFA to the data. We then sampled from the two models fit to the data, and compared the sampled data to the neural data (fig 4). Both models can capture the mean and variance of each neuron's firing rate collapsed across trials (fig 4A). GPFA assumes a Gaussian observation model, where the spike counts conditioned on latent state are drawm from a Gaussian with a specified firing rate offset $d$ and diagonal covariance $R$. Thus GPFA explicitly models the mean-variance relationship of individual neurons. In P-GPFA however, there is no explicit parameter for the variance of each neuron's spike counts. Despite the lack of an explicit parameter for variance of spike counts, P-GPFA can accurately describe the mean-variance relationship of each neuron's activty.

The statistics captured in figure 4A is collaped over trials, meaning that the values shown are measured over the entire training set. Significant trial-to-trial variability of this

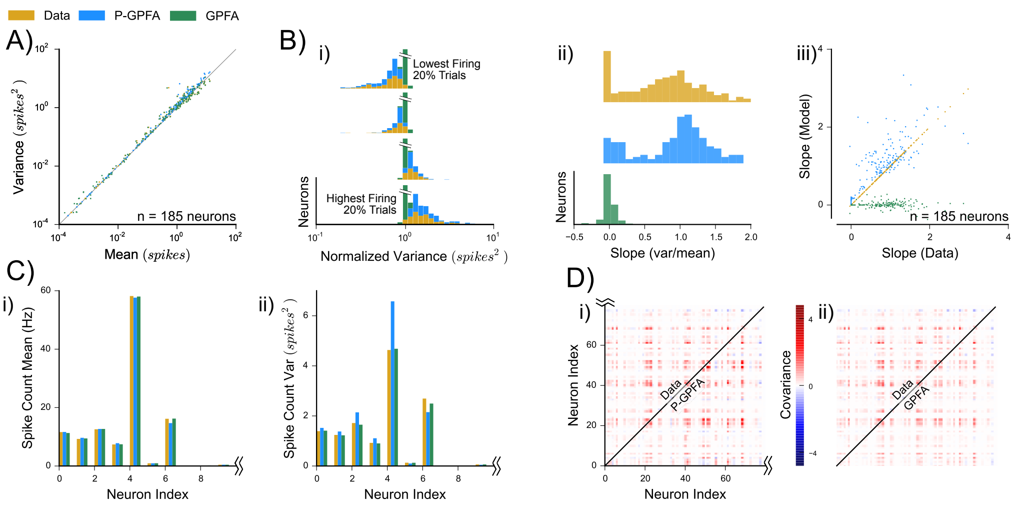

*Figure 4:* ***Comparison between P-GPFA and GPFA:*** *We fit GPFA and P-GPFA to data from (Stevenson et al., 2011), and compare the real data to samples drawn from the two models fit to data. (194 trials, 187 neurons, 2000ms trial duration, 100ms bin size)* ***A) Neuron-by-neuron mean-variance relationship.*** *Both models nicely capture the neurons' spike count mean and variance.* ***B) Trial-by-trial fluctuations in mean-var relationship.*** *We investigate trial-by-trial fluctuation in mean-variance relationship, ignoring the neuron-by-neuron difference. For each neuron, we separately split all trials into 5 bins, from low spiking to high spiking. For each neuron in each bin, we plot a histogram of normalized variance relative to the variance of the middle bin (i). The middle one reduces to a single point (not shown). In the data, higher firing rate trials have higher variance. P-GPFA captures this, while GPFA does not. Using the same data as in (i), we fit a straight line to the means and variances for each neuron in each bin. The distributions of the slopes are plotted in (ii,iii). For the real data and P-GPFA samples, the slopes are centered at 1, whereas for GPFA, the slopes are centered on 0 (ii).* ***C) True and sampled spike count mean and variance.*** *The firing rates of each neuron learned in both models reflect the true firing rates (i), as well as spike count variances (ii). Note that GPFA explicitly models the spike count mean and variance of each neuron, while P-GPFA has no explicit parameter for spike count variance. Only the first 10 neurons are shown.* ***D) True and sampled spike count covariances.*** *Both models capture spike count covariance well. Only the first 80 neurons are shown due to resolution.*

relationship is present, which we demonstrate in figure 4B. In our analysis, for *each neuron*, we create a 'long trial', which consists of the neuron's activity acorss all trials concatenated in time. This long trial is cut into a specified number of bins $m$. The bins are sorted by the mean spike count, and we calculate the mean and variance of the spike counts in each of the bins. Thus, for each neuron, we have $m$ measurements of mean and variance corresponding to $m$ bins. For all neurons, we cut the long trial into 5 bins, and normalize the variance in each bin by the variance of the middle bin. In neural data, we observe that the normalized variance of spike counts are higher in bins with higher firing rate (fig 4B.i). Data sampled from P-GPFA exhibit this property, while sampled data from GPFA does not. Further, we can investigate how the spike count variance varies as a function of mean spike count. To this end, we fit a straight line through the five pairs of mean and variances for each neuron. The slopes of the straight lines reveal neural activity's fluctuations in gain (fig 4B.ii,iii). The distribution of the slopes has a mode at 1 for the real date, as well as the sampled data from P-GPFA. For the data sampled from GPFA, the distribution of the slopes is narrowly peaked at zero.

## 5 Discussion

We found that the two posterior approximation methods outlined in section 3.2 perform similarly (fig 2). The two inference methods are ways to approximate the posterior distribution with a Gaussian pdf, since the true posterior has no closed form solution in our model. In general, performing exact inference is one of the main challenges of probabilistic models with a Gaussian process prior. Other methods of posterior inference in models adopting Gaussian process priors have been suggested. For example, in (Filippone and Girolami, 2014), it is shown that Pseudo Marginal approach to Monte Carlo sampling of the posterior leads to more accurate results than expectation propagation (EP), which performs better than the Laplace method (Kuss and Rasmussen, 2005).

It was pointed out that in our model, Laplace approximation requires optimization of a size $\mathcal{O}(pT)$ variable and Variational approximation requires optimization of a size $\mathcal{O}(qT)$ variable. In either cases, for large $T$, inference of all trials in each iteration can be very slow. To address this issue, we resorted to stochastic EM, where we only perform inference on a small number of trials in each iteration of EM. An alternative is to perform inference in the Fourier domain as in (Rabinowitz et al., 2015). Here, they proposed a very similar model for neural responses with a Gaussian process prior and Poisson observation process. Their model's key difference to our model is that it models responses of a *single* neuron, whereas P-GPFA models a neural population's response.

We presented parameter update methods of EM for our model. In addition to the traditional EM, we detailed the implementation of several stochastic versions of EM to minimize the impact of long computation time required for inference. In full EM, we showed that using Laplace inference reduces the computation time by two orders of magnitude than using Variational inference. In addition, we showed that our stochastic EM leads to an order of magnitude less computation time than full EM when used with Laplace inference, notwithstanding a slightly worse model fit than full EM. The reduction in computation time is proportional to the ratio of total available trials to the number of trials processed in each iteration of stochastic EM. When the algorithm is used to process much larger dataset, one can gain an arbitrary reduction (down to the computation time required for inference of one trial) in computation time by using stochastic EM.

In conclusion, we presented a probabilistic state-space model P-GPFA for spiking activities of neural population, improving upon the previous state-of-the-art model GPFA. We conclude that P-GPFA captures known statistics of neural spikes more realistically than GPFA, and presented model fitting techniques that can be readily applied to neural data.

## References

Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

Lars Buesing, Jakob H Macke, and Maneesh Sahani. Spectral learning of linear dynamics from generalised-linear observations with application to neural population data. In *Advances in neural information processing systems*, pages 1682–1690, 2012.

M Yu Byron, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In *Advances in neural information processing systems*, pages 1881–1888, 2009.

Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

Mark M Churchland, M Yu Byron, Maneesh Sahani, and Krishna V Shenoy. Techniques for extracting single-trial activity patterns from large-scale neural recordings. *Current opinion in neurobiology*, 17(5):609–618, 2007.

John P Cunningham and M Yu Byron. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 2014.

Mohammad Emtiyaz Khan, Aleksandr Aravkin, Michael Friedlander, and Matthias Seeger. Fast dual variational inference for non-conjugate latent gaussian models. In *Proceedings of The 30th International Conference on Machine Learning*, pages 951–959, 2013.

Maurizio Filippone and Mark Girolami. Pseudo-marginal bayesian inference for gaussian processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(11): 2214–2226, 2014.

Jayant E Kulkarni and Liam Paninski. Common-input models for multiple neural spike-train data. *Network: Computation in Neural Systems*, 18(4):375–407, 2007.

Malte Kuss and Carl Edward Rasmussen. Assessing approximate inference for binary gaussian process classification. *The Journal of Machine Learning Research*, 6:1679–1704, 2005.

K. Lange. A gradient algorithm locally equivalent to the em algorithm. *J. Roy. Statist. Soc. Ser. B*, 57(2):425–437, 1995.

Iain Murray, Ryan Prescott Adams, and David JC MacKay. Elliptical slice sampling. *arXiv preprint arXiv:1001.0175*, 2009.

Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 9(10), 2008.

Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of computational neuroscience*, 29(1-2):107–126, 2010.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):123–231, 2013.

Neil C Rabinowitz, Robbe LT Goris, Johannes Ballé, and Eero P Simoncelli. A model of sensory neural responses in the presence of unknown modulatory inputs. *arXiv preprint arXiv:1507.01497*, 2015.

Masa-aki Sato. Fast learning of on-line em algorithm. *Rapport Technique, ATR Human Information Processing Research Laboratories*, 1999.

Ian H Stevenson, Anil Cherian, Brian M London, Nicholas A Sachs, Eric Lindberg, Jacob Reimer, Marc W Slutzky, Nicholas G Hatsopoulos, Lee E Miller, and Konrad P Kording. Statistical assessment of the stability of neural movement representations. *Journal of neurophysiology*, 106(2):764–774, 2011.

D Michael Titterington. Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 257–267, 1984.

Michael Vidne, Yashar Ahmadian, Jonathon Shlens, Jonathan W Pillow, Jayant Kulkarni, Alan M Litke, EJ Chichilnisky, Eero Simoncelli, and Liam Paninski. Modeling the impact of common noise inputs on the network activity of retinal ganglion cells. *Journal of computational neuroscience*, 33(1):97–121, 2012.