Stony Brook University

Log-linear Model Based Tree and Latent Variable Model for Count Data

A Dissertation presented

by

Yuan Zhao

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Statistics)

Stony Brook University

August 2016

Stony Brook University

The Graduate School

Yuan Zhao

We, the dissertation committe for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Hongshik Ahn - Dissertation Advisor
Professor, Department of Applied Mathematics and Statistics

Stephen Finch - Chairperson of Defense
Professor, Department of Applied Mathematics and Statistics

Il Memming Park - Dissertation Coadvisor
Assistant Professor, Department of Neurobiology and Behavior

Sangjin Hong - Outside Member
Professor, Department of Electrical and Computer Engineering

This dissertation is accepted by the Graduate School

Nancy Goroff
Interim Dean of the Graduate School

Abstract of the Dissertation

Log-linear Model Based Tree and Latent Variable Model for Count Data

by

Yuan Zhao

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Statistics)

Stony Brook University

2016

Events that occur randomly over time or space result in count data. Poisson models are widely used for analyses. However, simple log-linear forms are often insufficient for complex relationship between variables. Thus we study tree-structured log-linear models and latent variables models for count data. First, we consider extending Poisson regression for independent observations. Decision trees exhibit the advantage of interpretation. Constant fits are too simple to interpret within strata nonetheless. We hence propose to embed log-linear models to decision trees, and use negative binomial distribution for overdispersion. Second, we consider latent variable models for point process observation in neuroscience. Neurons fire sequences of electrical spikes as signals which can naturally be treated as point processes disregarding the analog difference. Large scale neural recordings have shown evidences of low-dimensional nonlinear dynamics which describe the neural computations implemented by a large neuronal network. Sufficient redundancy of population activity would give us access to the underlying neural process of interest while observing only a small subset of neurons for understanding how neural systems work. Thus we propose a probabilistic method that recovers the latent trajectories non-parametrically under a log-linear generative model with minimal assumptions. Third, we are aim to model the continuous dynamics to further

understand the neural computation. Theories of neural computation are characterized by dynamic features such as fixed points and continuous attractors. However, reconstructing the corresponding low-dimensional dynamical system from neural time series are usually difficult. Typical linear dynamical system and autoregressive models either are too simple to reflect complex features or sometimes extrapolate wildly. Thus we propose a flexible nonlinear time series model that directly learns the velocity field associated with the dynamics in the state space and produces reliable future predictions in a variety of dynamical models and in real neural data.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I am especially indebted to Dr. Hongshik Ahn and Dr. Il Memming Park. This work would not have been possible without their academic mentorship and support. I am grateful to all of those with whom I have had the pleasure to study and work during the four years.

Chapter 1

Introduction

Events that occur randomly over time or space result in count data. Models of count data are used to explain and predict the number of occurrences of an event. The count data are usually heteroscedastic and right skewed. Poisson models are widely used for analyzing count data. However, simple log-linear forms are often insufficient for complex relationship between responses and explanatory variables.

In this study, we consider two specific problems in the field of count data: one is extending log-linear models of independent observations in biostatistics; the other pertains to latent variable models of point process observations in neuroscience.

1.1 Poisson log-linear models

A simple distribution modeling counts is Poisson [1]. The density function of the Poisson distribution is

$$f(y_i;\mu_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}, \quad y_i = 0,1,2,\ldots;\mu_i > 0 \tag{1.1}$$

where y_i is the i-th response, μ_i is the mean.

Poisson distribution belongs to the exponential family. McCullagh and Nelder [85] proposed the generalized linear models (GLMs) that adopt exponential family as distributions of response variables, and model functions of the mean, which extends ordinary regression models. Log-linear models are widely used Poisson regression models to analyze count with independent variables. Such models are of the form of the logarithm of the Poisson mean as linear form of the covariates,

$$\log(\mu_i) = \eta_i = \mathbf{x}_i'\boldsymbol{\beta}. \tag{1.2}$$

The advantage of GLMs is that it provides a unified framework of modeling, estimating, hypothesis testing and predicting for a family of distributions.

The model functions of the mean can be extended beyond linear forms. Hastie and Tibshirani [48, 49] introduced general additive models which replace the linear form by a sum of smoother functions that are estimated applicable to any likelihood-based regression model.

1.2 Greater variability in count

The variance of a Poisson random variable equals its mean. However, count data often exhibit greater variability than Poisson assumption [106, 54]. This phenomenon is called overdispersion. Overdispersion has various causes. Suppose that the response is Poisson distributed at each level of the covariates. Then the variance equals the mean when the relevant covariates

are controlled, but the heterogeneity causes the variance exceeding the mean when it is not controlled. McCullagh and Nelder [85] gave other scenarios in which intra-cluster variability leads to overdispersion. One might observe a Poisson process over an interval whose length is random rather than fixed, or the data might be produced by a clustered Poisson process where each event contributes a random amount to the total.

A model that allows extra-Poisson variations is desirable for such data. Ignoring occurrence of overdispersion in data may lead to an underestimation of variances and further anti-conservative results of tests. Such errors can exaggerate the association between the responses and factors.

Breslow [14] proposed a modification of the iteratively re-weighted least squares that adopts extra-Poisson variation. It introduces a random error term with zero mean and unknown constant variance into the log-linear form. It provides an equation to estimate the variance recursively, and a moment estimator as well. It proposes the iterative procedure of alternatively doing the re-weighted least squares estimation and the variance estimation until certain condition is satisfied.

Wedderburn [114] proposed to use quasi-likelihood approach to overcome this disadvantage that is mainly used for fitting linear or nonlinear regression models. Instead of specifying a distribution, it only assumes the relationship between the mean and variance. Then the quasi-likelihood function $Q(y_i, \mu_i)$ is defined by the score function,

$$\frac{\partial Q(y_i, \mu_i)}{\partial \mu_i} = \frac{y_i - \mu_i}{V(\mu_i)}. \tag{1.3}$$

The parameter $\boldsymbol{\beta}$ can be estimated by solving the equation,

$$S(\boldsymbol{\beta}) = \partial Q/\partial \boldsymbol{\beta} = 0 \tag{1.4}$$

where S is the quasi-score function.

Suppose the variance of y_i satisfies $\text{var}(y_i) = \phi V(\mu_i)$ for known V and unknown ϕ. It represents the overdispersion if $\phi > 1$. The maximum quasi-likelihood estimate of $\boldsymbol{\beta}$ and μ_i will not be affected by the value of ϕ so that $\hat{\boldsymbol{\beta}}$ can be calculated by setting ϕ to 1. In a Poisson model, with $V(\mu_i) = \mu_i$, the maximum quasi-likelihood estimate is identical to the maximum likelihood estimate. So the estimated covariance matrix of $\hat{\boldsymbol{\beta}}$ becomes ϕ times that in the Poisson model. This implies that the Poisson model may not be appropriate when overdispersion arises since the estimated covariance matrix will affect the tests. Usually ϕ is an unknown parameter to be estimated from the data. Wedderburn [114] suggested a moment estimation

$$\hat{\phi} = \frac{1}{n-p}\sum_{i=1}^{n}\frac{(y_i - \mu_i)^2}{V(\mu_i)}. \tag{1.5}$$

Nelder and Pregibon [87] extended quasi-likelihood to allowing for the comparison of various forms of the components of a generalized linear model. The extended quasi-likelihood function Q^+ is defined as

$$Q^+(y_i;\mu_i) = -\frac{1}{2}\log(2\pi\phi V(y_i)) - \frac{1}{2}D(y_i;\mu_i)/\phi \tag{1.6}$$

where D is the deviance and ϕ is the dispersion parameter. The estimates of

$\boldsymbol{\beta}$ are the same as those obtained from quasi-likelihood whereas the estimate of ϕ by maximizing Q^+ is $D(y_i;\hat{\mu}_i)/n$.

Chen and Ahn [19] investigated the approach of quasi-likelihood. They extended quasi-likelihood and pseudo-likelihood to estimating and testing the mean parameters with respect to two models, $\text{var}(y) = \mu^\theta(1+\phi\mu)$ and $\text{var}(y) = \mu^\theta(1+\tau)$. They discussed that the quasi-likelihood yields consistent estimates for the mean parameter with reduced efficiency even if the structure of the variance is not correctly specified.

Another approach dealing with overdispersion is the negative binomial model. Comparing to Poisson distribution, a negative binomial distribution has an additional parameter that scales the variance. Thus it provides flexibility over the Poisson model. The negative binomial model can be considered as a Poisson model with a gamma distributed error residing in its mean in a multiplicative manner [50]. Suppose the counts follow the Poisson distribution conditional on ν_i:

$$f(y_i;\mu_i,\nu_i) = \frac{e^{-\mu_i\nu_i}(\mu_i\nu_i)^{y_i}}{y_i!} \tag{1.7}$$

where y_i is the observed number of count for $i = 1, 2, \ldots, n$ and ν_i is the one-parameter gamma distributed error term with rate and shape θ. Integrating μ, we obtain the marginal density of y_i as

$$\begin{aligned} f(y_i;\mu_i,\theta) &= \int_0^\infty \frac{e^{-\mu_i\nu_i}(\mu_i\nu_i)^{y_i}}{y_i!}\frac{\theta^\theta}{\Gamma(\theta)}\nu_i^{\theta-1}e^{-\theta\nu_i}\,d\nu_i \\ &= \frac{\Gamma(y_i+\theta)}{\Gamma(y_i+1)\Gamma(\theta)}\left(1-\frac{\mu_i}{\mu_i+\theta}\right)^\theta\left(\frac{\mu_i}{\mu_i+\theta}\right)^{y_i} \end{aligned}$$

which is the p.m.f. of a negative binomial random variable with mean μ_i and

variance $\mu_i(1+\mu_i/\theta)$. This model is as known as NB2. There are other forms of negative binomial models. NB1 model is similar to NB2 except that the variance is $\mu_i + \frac{\mu_i}{\theta}$ [50].

Given θ fixed and re-parameterizing $p_i = \frac{\mu_i}{\mu_i+\theta}$, the p.m.f. can be expressed in exponential family form as

$$f(y_i;\mu_i,\theta) = \exp\left\{ y_i \log(p_i) + \theta \log(1-p_i) + \log \frac{\Gamma(y_i+\theta)}{\Gamma(y_i+1)\Gamma(\theta)} \right\} \qquad (1.8)$$

with log link.

We transform θ into $\phi = 1/\theta$ for convenience in the thesis. If the regression model is specified in terms of μ, say $\mu = \mu(\beta)$, and if ϕ is an unknown constant, then it will no longer be of exponential family and hence the estimating equations for β are in general different from those obtained by iteratively re-weighted least squares.

Lawless [72] studied negative binomial regression models that can handle extra-Poisson variation. He stated that the simplest way of obtaining maximum likelihood estimates $(\hat{\boldsymbol{\beta}}, \hat{\phi})$ was to maximize the log-likelihood with respect to $\boldsymbol{\beta}$ by fixing the value of dispersion parameter ϕ. This leads to the estimates $\tilde{\boldsymbol{\beta}}(\phi)$ and the profile likelihood $l(\tilde{\boldsymbol{\beta}}(\phi), \phi)$. Then one can estimate ϕ from the profile likelihood.

The maximum likelihood estimation of the coefficients and the dispersion parameter are asymptotically independent. It was found that under mild conditions on the distribution of the response, $\hat{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\beta}}$ are both consistent estimators of $\boldsymbol{\beta}$. For some positive ϕ, $\tilde{\phi}$ gives a consistent estimation.

Hilbe [50] mentioned an algorithm that is called a ML negative binomial to obtain the estimate of θ, and then uses it as constant in the GLM algorithm.

Cameron and Trivedi [16] also proposed a test based on the standardized dispersion statistic

$$S = \frac{\sum_{i=1}^{n}[(y_i - \hat{\mu}_i)^2 - y_i]}{\sqrt{2\sum_{i=1}^{n}\hat{\mu}_i^2}} \tag{1.9}$$

that is asymptotically standard normal under the hypothesis that y_i's are independent Poisson random variables.

1.3 Segment log-linear models

A single log-linear model is often insufficient for complex data. Relationship beyond linearity between the response and explanatory variables or variability may vary across strata. Hence stratifying the data according to covariates can provide insight into the nature of the response and explanatory variables. The data in strata involve few covariates and might be more homogeneous.

Tree-structured regressions feature highlights of decision trees: interpretability and predictive power in nonlinear regression relationship. The importance of predictive power has been dimmed by modern approaches such as boosting and random forests [47] which often perform better than trees. However, the interpretability of single tree is still valuable in form of graphical representations of complex regression problems. Chaudhuri et al. [18] studied Poisson regression tree. Hongshik Ahn [55] proposed tree-structured methods for over-dispersed binomial data. Choi, Ahn, and Chen [21] modified the former

method with the quasi-likelihood approach for fitting extra-Poisson models.

There have been increasing interests in the incorporation of simple regression models into trees. The motivation is by the fact that the nodes of classical trees use constant fits which tends to grow large trees and thus they are hard to interpret [17]. Methods attaching simple regression models to terminal nodes or splitting in non-terminal nodes were proposed. Kim and Loh [64], Loh [75], and Chan and Loh [17] use parametric models in terminal nodes. Loh and Shih [76] employ parametric models to obtain splits in non-terminal nodes. Choi, Ahn, and Chen [21] modified the tree-structured methods for over-dispersed count data using quasi-likelihood approach and moment estimator, and developed a test of overdispersion. Zeileis, Hothorn, and Hornik [120] embed recursive partitioning into parametric models with fluctuation tests.

In a simple generalized linear model, all the observations have one common coefficient vector. Since we partition only a single variable at one time, it reduces to a scalar coefficient in our problem,

$$\begin{aligned} \ln \mu_1 &= \beta_0 + \beta_1 x_1 \\ &\vdots \\ \ln \mu_j &= \beta_0 + \beta_j x_j \\ &\vdots \\ \ln \mu_n &= \beta_0 + \beta_n x_n \end{aligned} \tag{1.10}$$

where $x_1 \dots x_n$ are the values of n observations. Note that the $\beta_1 \dots \beta_n$ correspond to not different variables but observations here. This is different from

usual notation. We formulate the sufficiency of linearity in terms of such hypothesis test,

$$\mathrm{H}_0 : \beta_1 = \cdots = \beta_n = \beta \tag{1.11}$$

If the linear relationship is sufficient, that is all the coefficients are the same, then the hypothesis is accepted. Otherwise there will be more than one distinct coefficients if the linearity is broken at some point. It is natural to ask whether the relationship between responses and explanatory variables changes over particular ordering. A large number of literature have been written since the CUMSUM test [99]. Werner Ploberger [115] generalize the CUMSUM test to dependent and heteroskedastic OLS residuals. Kuan and Hornik [67] introduce a generalized and unifying framework without a specific alternative for linear regressions. Hjort and Koning [51] propose tests based on maximum likelihood scores. Zeileis and Hornik [119] suggest generalized M-fluctuation tests incorporating a variety of existing tests.

Inspired by these ideas, we embed log-linear models into trees by partitioning the explanatory variables. In our work, a segmented log-linear model is fitted by computing a tree in which every node is associated with a log-linear model using maximum likelihood estimation. The corresponding parameter instability is tested for each node to assess whether a variable should be used for partitioning. The recursive partitioning allows for non-linear relationships. Moreover, the use of log-linear models provides with scientific insight and interpretability. We apply our method on the data for lung-cancer mortality among the 115 counties in Missouri during the period from 1972 to 1981 [109,

110][1]. The resulted regression trees will shown in Chapter 2.

1.4 Neural spike trains are point processes

Ion channels spanning on the neuronal membrane control the inflow and outflow of ions. By controlling the flowing ions, a neuron is able to change the membrane potential and generates action potentials. An action potential is a roughly 100 mV fluctuation in the electrical potential across the cell membrane that lasts for about 1 ms. Action potentials are the only form of membrane potential fluctuation that can travel down nerve fibers over long distances. Neurons use such electrical pulses as signals to communicate with one another. These signals are also called spikes.

Neural signals are usually obtained by electrodes. While recording, a hollow glass electrode filled with conducting electrolyte is connected to a neuron. The potentials are captured and compared with ones from a reference electrode placed in the extracellular medium. Action potential sequences are recorded either intra- or extracellularly.

Neurons respond to various stimulus such as light, sound or motor actions. The attributes of stimulus are represented and transmitted by firing sequences of spikes in various temporal patterns. Such sequences of spikes are usually called spike trains. It is important to study the relationship between stimulus and response in order to understand how the stimuli are encoded in the sequences.

Generally the spikes are abstracted as identical discrete events disregarding

[1]The data can be obtained from `http://www.ams.stonybrook.edu/hahn/research/tree.html`

durations, amplitudes and shapes in neural study. Neuronal responses can vary from trial to trial even given the same stimulus. This variability comes from many different sources and make spikes stochastic. Point processes naturally describe spike trains in terms of events.

A point process is modeling the occurrences of some phenomenon at the time epochs $\{t_i\}$ with i in some suitable index set [27]. For a neural spike train this would be the set of individual spike times. For a sequence of n spikes, we denote the spike times by t_i with $i = 1, 2, \ldots, n$. The spike train can also be represented as a sum of infinitesimally narrow, idealized spikes in the form of Dirac δ functions,

$$\rho(t) = \sum_{i=1}^{n} \delta(t - t_i) \tag{1.12}$$

where $\rho(t)$ is called the neural response function and used to re-express sums over spikes as integral over time.

Another equivalent way of describing a point process is counting events. Counting spikes over a time interval $[0, t]$ gives the function

$$N(t) = \int_0^t \rho(\tau)\, d\tau. \tag{1.13}$$

The firing rate also characterizes spike trains defined as mean count over an time interval,

$$r = \frac{N(t)}{t}. \tag{1.14}$$

However this is insufficient to determine the distribution of spikes. To find the distribution, starting with the simplest case — constant rate over time, then n spikes are generated in equal probability over a fixed interval. Divide the time

t into m equal-spaced bins, $\Delta t = t/m$. Assuming that Δt is small enough to involve at most one spike in each bin, each bin becomes a Bernoulli variable where exists a spike or not. Then the number of spikes over the entire interval is binomial distributed,

$$\begin{aligned} P\{N(t) = n\} &= \frac{m!}{n!(m-n)!}(r\Delta t)^n(1 - r\Delta t)^{m-n} \\ &= \frac{\lambda^n}{n!}\frac{m!}{m^n(m-n)!}(1 - \lambda/m)^{-n}(1 - \lambda/m)^m \end{aligned} \tag{1.15}$$

where $\lambda = r\Delta t$. As $\Delta t \to 0$,

$$P\{N(t) = n\} \to \frac{\lambda^n e^{-\lambda}}{n!}. \tag{1.16}$$

Therefore the spike count in an interval is Poisson distributed for constant rate. Such a process is called homogeneous Poisson process.

Generally a realistic firing rate would not be constant over time. Spike trains exhibit various temporal correlation structures. The distribution of a spike train is determined by the joint probability density of spikes of the sequence. The joint probability density can be characterized in terms of the conditional intensity function,

$$\lambda(t \mid H_t) = \lim_{\Delta t \to 0} \frac{P\{N(t + \Delta t) - N(t) = 1 \mid H_t\}}{\Delta t} \tag{1.17}$$

where H_t is the history of the sample path and any covariates up to time t. An inhomogeneous Poisson process is a special case that preserves the assumption of independent spike arrival times. The spike count in any interval is Poisson

distributed. For a particular ordering $0 \leq t_1 \leq t_2 \leq \ldots \leq t_n \leq t$, the probability of the sequence is given as

$$P\{t_1, t_2, \ldots, t_n\} = \exp\left(-\int_0^t \lambda(\tau)\, d\tau\right) \prod_{i=1}^{n} \lambda(t_i). \tag{1.18}$$

Note that the probability should be divided by $n!$ if the times are not ordered.

Although such processes are defined in continuous time, they can be discretized by binning to time series. For example, a discretized homogeneous Poisson process with evenly-spaced bins are series of Poisson random variables.

Classical analyses of neural spike trains typically average the responses over repeated trials that are presumably time-locked to a stereotypical computation process. Denote the average neural response function by $\bar{\rho}(t)$, the time-variant firing rate is estimated by

$$r(t) = \frac{1}{\Delta t} \int_t^{t+\Delta t} \bar{\rho}(\tau)\, d\tau. \tag{1.19}$$

For sufficiently small Δt, $r(t)\Delta t$ is the average spike count in the time interval $[t, t + \Delta t]$ over multiple trials.

However, the complex neural responses are not necessarily time-locked nor precisely repeated from trial to trial. The sequences of spikes often reflect a mixture of both neuronal intrinsic dynamics and temporal features of the stimulus. The mean trajectory loses the observable variations in the internal processes that manifest in behavior such as error trials, broad reaction time and change of mind. For example, the firing rates in trial-averaged responses could arise from instantaneous jumps at different times on different trials in

the macaque lateral intra-parietal (LIP) area during decision-making [70]; The standard deviation of duration between events increases with their mean [60]; decision-makers would change of mind in some trials without additional information [102]. In addition, it is difficult to disambiguate different possible neural implementations of computation from the average trajectory since they may only differ in their trial-to-trial variability. Churchland et al. [22] unveils neural computations that cannot be discerned from measures of average firing rate by an analysis of neural response variance. Therefore, if we wish to understand how neural computation is implemented in neural populations, it is necessary to learn from individual trials.

1.5 Recover latent trajectories from spike trains

Advanced techniques enable recording from large subpopulations of neurons which facilitate single-trial analysis. Footprints of underlying low-dimensional dynamics have been revealed behind large-scale recordings, that is, a small number of common factors often explain most of the dependence among neurons. The firing patterns produced by a large population of neurons in cortex lie in a space of lower dimension [88]. Variability in responses of sensory neurons arising from fluctuations in excitability was found due to factors such as arousal, attention and adaptation which are not purely sensory [42]. A common sequential structure contributes to variations in population timing patterns with different stimuli [77]. All these evidences support the idea that a large neuronal network is implementing necessary computations described by continuous low-dimensional nonlinear dynamics.

Although we can only observe a small subset of neurons at one time, sufficient amount of redundancy in the population activity would allow us access to the internal computation process of interest. Thus, it is necessary to deduce the latent trajectories from neural time series in order to understand if and how neural systems operate in this regime. Latent trajectories recovered from motor cortex suggest that these methods can provide insight to the coding and preparation of planned reaching behavior [103, 24, 23]. Latent dynamical trajectories also elucidate the low-dimensional noise structure of neural codes and computations [86, 44, 103, 30].

Several statistical approaches have been developed for extracting latent dynamical trajectories that describe the observed neural activities in populations. With the assumption of low-dimensionality and discretization, the inference of latent dynamical trajectories is a dimensionality-reduction method for multivariate time series, akin to Kalman smoothing or factor analysis [62]. Given the sequences of observations, the task is to infer a shared, low-dimensional latent process that explains much of the variation in high-dimensional observations. For example, Koyama et al. [66] uses Laplace's method to approximate the mean and variance of the posterior density in state-space models; Pfau, Pnevmatikakis, and Paninski [95] models the firing rate driven by linear combination of latent trajectory and history of spikes; Archer et al. [3] propose a state-space model of logarithm of firing rate which incorporate the stimulus and interactions into the linear dynamics through quadratic form and interaction; Frigola, Chen, and Rasmussen [36] defines the state transition function by Gaussian process.

In our study, we pay attention to two typical classes of methods.

A large number of neural trajectory inference algorithms assume linear dynamical system [91, 80, 15, 25] in the latent space which lack nonlinear features ubiquitous in neural computation. We refer to those with Poisson likelihood as PLDS (Poisson Linear Dynamical System). Denote the latent state by $\mathbf{x}_t$ at time t. PLDS defines a Gaussian linear dynamics with an external drive $\mathbf{u}_t$ as

$$\begin{aligned} \mathbf{x}_1 &\sim \mathcal{N}(\mathbf{x}_0, \mathbf{Q}_0) \\ \mathbf{x}_t \mid \mathbf{x}_{t-1} &\sim \mathcal{N}(\mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t, \mathbf{Q}) \end{aligned} \tag{1.20}$$

where $\mathbf{A}$ is the state transition matrix, matrix $\mathbf{B}$ is for control-input, $\mathbf{Q}_0$ and $\mathbf{Q}$ are covariance matrices.

Although the assumption of linear dynamics advantages computational tractability, it can also be overly simplistic: interesting neural computations are naturally implemented as nonlinear dynamics, and evidence points to nonlinear dynamics in the brain in general.

The Gaussian process factor analysis (GPFA) method [118, 69] relaxes the assumption of linearity and imposes a general Gaussian process to nonparametrically infer the latent trajectories.

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [101]. It is completely specified by its mean function

$$m(t) = \mathbb{E}(x_t), \tag{1.21}$$

and covariance function

$$k(t_1, t_2) = \mathbb{E}[(x_{t_1} - m(t_1))(x_{t_2} - m(t_2))], \tag{1.22}$$

where x_t is a Gaussian random variable.

The covariance function plays an important role in a Gaussian process. It encodes the assumption about the correlation structure we wish to learn. GPFA chooses a modified squared exponential covariance function,

$$k_{\text{GPFA}}(t_1, t_2) = \sigma_f^2 \exp\left(-\frac{(t_1 - t_2)^2}{2\tau^2}\right) + \sigma_n^2 \cdot \delta_{t_1, t_2} \tag{1.23}$$

where σ_f^2 is the signal variance, τ is the timescale, σ_n^2 is the noise variance and δ_{t_1,t_2} equals 1 if $t_1 = t_2$ and 0 otherwise. A typical squared exponential covariance function only includes the first exponential term. It provides general smoothness and is probably the most widely-used kernel within the kernel machines field.

Despite of the advantage of Gaussian process, GPFA assumes Gaussian observations on square root of spike count in time bins. These observations are driven by the latent Gaussian process. As we mentioned above, point process observations are more appropriate to spike trains. Even if the spike count could be approximately Gaussian statistically when it is of a high value, a Gaussian distribution would do poorly due to rare spikes per bin in the millisecond-range time scale.

To overcome the shortcoming and exploit the advantages of point process and Gaussian process, we propose a generative model where the point process

observation are driven by Gaussian process. Precisely we bin the observations with equal time-interval and assume conditional independence of bins so that we obtain Poisson likelihood. The detail will be discussed in Chapter 3.

We have assumed the generation of spikes is driven by the dynamics in the model. Yet we are only able to observe the spike trains in experiment, while the dynamics of interest are latent. We hence do Bayesian inference on the trajectories.

Bayesian inference provides a framework of making conclusions about unobserved quantities in terms of observations. Denote the observation by $\mathbf{y}$ and the trajectory by $\mathbf{x}$. Given a prior $p(\mathbf{x})$, Bayes' rule yields the posterior density as

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \tag{1.24}$$

It updates as more evidence or information become available. In neural context, $\mathbf{y}$ represents a spike train and $\mathbf{x}$ represents the latent trajectory.

The posterior is obtained analytically if the marginal density $p(\mathbf{y})$ is available given prespecified prior $p(\mathbf{x})$ and likelihood $p(\mathbf{y} \mid \mathbf{x})$. If the posterior is in the same family as the prior, a closed-form the posterior is given conveniently without explicit marginal. Then the prior is called conjugate prior for the very likelihood. However, this is not always the case. The marginal density is still required otherwise by integration (continuous random variables) or summation (discrete random variables) of the numerator over the hidden variable. However the integral or sum may not have a closed-form in practice. It makes the exact calculation of the posterior impractical.

In our generative model, the point process observation has conditional

Poisson likelihood. The Gaussian prior is not conjugate with respect to it. So the price we pay is a non-conjugate prior and, consequently, we turn to approximation methods.

A family of approximation techniques called variational Bayes has been widely used which originates from the calculus of variations [7]. It is a method to find the function that optimizes a certain functional. Decompose the logarithm of marginal density in the following way,

$$\ln p(\mathbf{y}) = \ln \frac{p(\mathbf{x} \mid \mathbf{y})}{p(\mathbf{x} \mid \mathbf{y})} p(\mathbf{y}) \tag{1.25}$$

$$= \ln \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x} \mid \mathbf{y})} \tag{1.26}$$

$$= \ln p(\mathbf{x}, \mathbf{y}) - \ln p(\mathbf{x} \mid \mathbf{y}) \tag{1.27}$$

introduce a new distribution $q(\mathbf{x})$,

$$= \ln \frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{x})} - \ln \frac{p(\mathbf{x} \mid \mathbf{y})}{q(\mathbf{x})} \tag{1.28}$$

take expectation with respect to q on both sides,

$$= \int q(\mathbf{x}) \ln \frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{x})} \, d\mathbf{x} - \int q(\mathbf{x}) \ln \frac{p(\mathbf{x} \mid \mathbf{y})}{q(\mathbf{x})} \, d\mathbf{x} \tag{1.29}$$

$$= \int q(\mathbf{x}) \ln \frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{x})} \, d\mathbf{x} + \int q(\mathbf{x}) \ln \frac{q(\mathbf{x})}{p(\mathbf{x} \mid \mathbf{y})} \, d\mathbf{x}. \tag{1.30}$$

Note that the second term on the right hand side is the Kullback-Leibler (KL) divergence of $p(\mathbf{x} \mid \mathbf{y})$ from $q(\mathbf{x})$ denoted by $\mathrm{KL}(q \| p)$. The KL divergence is a measure of the difference between two probability distributions. It equals

to zero if and only if the two distribution are equal almost everywhere. Since the KL divergence is always nonnegative, the first term is a lower bound of the logarithm of marginal. Denote the lower bound by $\mathcal{L}(q)$. It is also called evidence lower bound (ELBO).

The distribution $q(\mathbf{x})$ equals to the posterior density if the KL divergence equals to zero. It implies that $q(\mathbf{x})$ can be a feasible approximation to the posterior as long as the divergence is minimized. However, the intractable posterior prevents the calculation of KL divergence. Instead we can maximize the lower bound with respect to the distribution $q(\mathbf{x})$ equivalently. The family of $q(\mathbf{x})$ must be restricted for optimization. One way of restriction is to use a parametric distribution determined by a set of parameters. Then the problem becomes nonlinear optimization for optimal values of the parameters.

There are plenty of methods solving the optimization problem. Under differentiability of objective functions, gradient descent and Newton-type methods are widely used. Newton's method advantages better rate of convergence thanks to the curvature information. It also brings information matrix as by-product that happens to be the negative Hessian of likelihood objective.

Since we impose Gaussian prior on the trajectory, the finite sample of Gaussian process can be treated as a multivariate normal random vector with covariance matrix defined by the corresponding covariance function. For general smoothness, we choose the squared exponential covariance function as GPFA does. The smoothness is then determined by the time-scale. However, the smoothness leads the covariance matrix to be ill-conditioned such as almost singular. The computation of its inverse or solution of a linear system

in the optimization problem is prone to numerical errors. Most widely used strategies involves incomplete Cholesky factorization [41]. The idea is to compute a low rank sparse matrix $\mathbf{G}$ that is close to the exact Cholesky factor, i.e.

$$\mathbf{K} \approx \mathbf{G}\mathbf{G}^{*}. \tag{1.31}$$

We will discuss how those techniques are incorporated and other detail of our method in Chapter 3. There we validate our method in contrast to PLDS and GPFA and apply to a real neural recording from primary visual cortex of monkey.

1.6 Dynamical models for interpretation

Understanding how the neural dynamics evolve can give us insight of the mechanism of neural computation. There are many proposed theories and models of how the brain works, however, we have been in the data poor regime since the dawn of electro-physiology, where theories of neural computation were only partially supported by experimental data. Continuous dynamical systems theory lends itself as a framework for both qualitative and quantitative understanding of neural models [45, 78, 58, 107].

A dynamical system are defined by differential equations. Differential equations describe the evolution of systems in continuous time. The temporal behavior that concerns us only involves the derivatives with respect to time that pertain to ordinary differential equations (ODE). Generally a n-variate

dynamical system described by ODEs is given as

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, \ldots, x_n) \\ &\vdots \\ \dot{x}_n &= f_n(x_1, \ldots, x_n) \end{aligned} \tag{1.32}$$

where the overdots denote the differentiation with respect to time t, $\dot{x}_i \equiv dx_i/dt$. The system is linear if the functions f_i are linear in all the x's.

Models of neural computation are often implemented as attractor dynamics where the convergence to one of the attractors represents the result of computation. The models also are often reduced to low-dimensional system, ignoring the fast dynamic modes. The key features of dynamics are multiple stable and unstable fixed points, bifurcation, meta-stability, chaos and robustness to noise. Stable fixed points, also known as attractors, are important because the system will converge to them or, in the presence of noise, dwell near them. Unstable fixed points come in more varieties, as either repellers or saddle points that funnel a large volume of phase space through. Slow points in areas of phase space are not true fixed points but merely points of very slow movement [107]. When the neuron is resting, it has a stable equilibrium. Small perturbations causes small excursions from the equilibrium, while large perturbations are amplified by the neuron's intrinsic dynamics and result in spikes. Sufficiently strong current into a neuron excites periodic spike firing. In terms of dynamical system, the state has a limit circle. Equilibria and limit cycles can both exist and transit from one to the other by a transient input. Such transitions correspond to a bifurcation of neuronal dynamics. Neurons

are excitable near bifurcations from resting to spiking. Synchronized chaos generated by the internal dynamics of a large neural network is investigated in [45].

Recent advances in the neural recording capabilities allow us for the first time to access large number of neurons in the mammalian brain. Ultimately, brain is a computing machine where its internal dynamics implement computations at various time scales, and to learn how the brain works, we must connect the theories with the experimental time series at appropriate scales. The methods of dynamics recovery bring low-dimensional and smooth time series as mentioned in the previous section. To connect those dynamical theories with neural time series data, we need to be able to fit a model that has the expressive power to identify such features of a dynamical system. Despite the wide adoption of dynamical systems theory in theoretical neuroscience, solving the inverse problem, that is, reconstructing meaningful dynamics from neural time series, has been challenging.

One of the key ingredients we are missing is an expressive dynamical model that is easily interpretable and controllable. Numerous linear and nonlinear time series models emerge in the field. However most of them are not readily interpretable due to their parameterization, and often produces wild extrapolation which is not suitable for scientific study confidently recovering the dynamics is of great interest.

In this study, we aim to build an interpretable dynamics model to reverse-engineer the neural implementation of computation.

Clearly linear dynamical system models (e.g., PLDS [80]) can have at most

one fixed point, so that they cannot predict the trajectories such as meta-stable states, saddles, fixed points, hyperplane attractors, and slow points, or furthermore exhibit important features of nonlinear theories of neural computation, even with static nonlinear transformations (Hammerstein-style model) [4].

Typical approaches of using nonlinear autoregressive models [90]. Eikenberry and Marmarelis [31] proposed a new variant of Volterra-type model with a nonlinear autoregressive component as a framework for describing the process of action potential generation by the neuron membrane potential. The proposed model was applied to input-output data generated by the Hodgkin-Huxley equations [53].

Consider a general d-dimensional continuous nonlinear dynamical system driven by external input,

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}) \tag{1.33}$$

where $\mathbf{x} \in \mathbb{R}^d$ represent the dynamic trajectory, and $F : \mathbb{R}^d \times \mathbb{R}^{d_i} \to \mathbb{R}^d$ fully defines the dynamics in the presence of input drive $\mathbf{u} \in \mathbb{R}^{d_i}$. We aim to learn the essential part of the dynamics F from a collection of trajectories sampled at frequency $1/\Delta$.

Assuming a separable, linear input interaction, $F(\mathbf{x}, \mathbf{u}) = F_x(\mathbf{x}) + F_u(\mathbf{x})\mathbf{u}$, many nonlinear take a general form:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \epsilon_t, \tag{1.34}$$

where $\mathbf{B}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^{d \times d_i}$ is the linearization of F_u around $\mathbf{x}$ and ϵ_t denotes model mismatch noise of order $\mathcal{O}(\Delta^2)$. This form is widely used, and

called nonlinear autoregressive with eXogenous inputs (NARX) model where $\mathbf{f}$ assumes various function forms (e.g. neural network, radial basis function network [20], or Volterra series [31]).

One subclass of (4.3) use a locally linear expansion [90, 113]:

$$\mathbf{x}_{t+1} = \mathbf{A}(\mathbf{x}_t)\mathbf{x}_t + \mathbf{b}(\mathbf{x}_t) + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \epsilon_t \tag{1.35}$$

where $\mathbf{A}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ and $\mathbf{b}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ is the linearization of $\mathbf{f}$ around $\mathbf{x}$. For example, $\{\mathbf{A}, \mathbf{B}\}$ are parametrized with an RBF network in multivariate RBF-ARX model of [38, 90], and $\{\mathbf{A}, \mathbf{b}, \mathbf{B}\}$ are parametrized with sigmoid neural networks in [113].

Our work assumes slow continuous dynamics allowing us to propose a flexible nonlinear time series model that directly learns the velocity field, and builds on a specific parameterization. The particular parameterization yields to better interpretations: identifying fixed points and ghost points are easy, and so is the linearization of the dynamics around those points for stability and manifold analyses.

We further parameterize the velocity field using a finite number of basis functions. Standard multilayer feed-forward networks with as few as one single hidden layer and arbitrary bounded and non-constant basis function are universal approximation to continuous functions on compact subsets of $\mathbb{R}^n$ [56]. Simple networks can represent a wide variety of interesting functions.

One class of the widely used basis functions are radial basis functions. A radial basis function (RBF) is a real-valued function whose value depends only

on the distance from some center $\mathbf{c}$, so that $\phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x}-\mathbf{c}\|)$. One common used type of RBF are Gaussian,

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}}, \tag{1.36}$$

where r is the distance.

The artificial neural network that uses radial basis function as activation functions is called a radial basis function network. A typical radial basis function network consists of three layers: an input layer, a hidden layer and an output layer. The input layer is a vector of real numbers $\mathbf{x}$. The hidden layer contains multiple radial basis functions. The output layer is a linear combination of radial basis functions of the inputs and parameters, which is

$$\varphi(\mathbf{x}) = \sum_{i=1}^{n} w_i \phi(\|\mathbf{x} - \mathbf{c}_i\|), \tag{1.37}$$

given the hidden layer has n nodes. We use a normalized radial basis function networks in this study. The definition will be given in Chapter 4.

In addition, we add a global contractional component. These features encourages the model to focus on interpolating dynamics within the support of the training trajectories. We show the expressive power of our modeling framework by characterizing dynamical features of various computational models.

Chapter 2

Log-linear Model Based Tree

Poisson regression is a widely-used tool for analyzing counting observations. A common form of it is the log-linear model. The assumptions of linearity and Poisson have not only advantages but also limitations. Relaxing such assumptions is desirable when the data are complex and overdispersed.

Decision trees stratify the data hierarchically which gives insight and interpretability of data. Meanwhile, negative binomial distribution as an extension to Poisson distribution allows modeling overdispersed data. We embed log-linear models with negative binomial likelihood into nodes of a tree for those benefits.

2.1 Log-Linear Model

A log-linear model takes the logarithm of response mean as a linear combination of the covariates,

$$\log \mu_i = \eta_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}, \tag{2.1}$$

where μ_i the mean of i-th observation and $x_{i1} \dots x_{ip}$ are the corresponding covariates. The response is often treated as Poisson distributed and the maximum likelihood estimation (MLE) is widely used for parameters. The MLE's derived likelihood equation is

$$\sum_{i=1}^{n} \frac{(y_i - \mu_i)x_{ij}}{\text{var}(y_i)} \frac{\partial \mu_i}{\partial \eta_i} = 0.$$

One can obtain the estimates of $\boldsymbol{\beta}$ by solving it. Newton-Raphson method and Fisher scoring are two usual ways to find the solution [1].

Newton-Raphson method starts with an initial value for the solution. It iteratively keeps updating the solution by approximating the function to be maximized in a neighborhood of the previous solution until the process converges to the maximum,

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - (\mathcal{H}^{(t)})^{-1} \nabla L(\boldsymbol{\beta}^{(t)})$$

where $\nabla L(\boldsymbol{\beta})$ and $\mathcal{H}$ are the gradient and Hessian matrix of the log-likelihood function respectively.

Fisher scoring resembles the Newton-Raphson method using the expected information instead of the observed information by taking expectation of the Hessian matrix,

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + (\mathcal{I}^{(t)})^{-1} \nabla L(\boldsymbol{\beta}^{(t)}).$$

A generalized linear model with canonical link, such as a Poisson log-linear model, has the identical observed information to the expected information.

Fisher scoring also gives the asymptotic covariance matrix besides the estimation, but the observed information usually is easier to calculate.

Another variety is iterative reweighted least squares (IRLS). It can be shown that

$$\mathcal{I} = \mathbf{X}'\mathbf{W}\mathbf{X}$$

where $\mathbf{W}$ is the diagonal matrix with elements $w_i = (\partial\mu_i/\partial\eta_i)^2/\mathrm{var}(y_i)$. Then Fisher scoring formula can be rearranged as

$$\boldsymbol{\beta}^{(t+1)} = (\mathbf{X}'\mathbf{W}^{(t)}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{(t)}\mathbf{z}^{(t)}$$

where $z_i = \eta_i + (y_i - \mu_i)(\partial\eta_i/\partial\mu_i)$. Since the initial value does not require $\boldsymbol{\beta}$, the process can simply begin with the observation $\mathbf{y}$ as the initial estimate of $\boldsymbol{\mu}$.

The negative binomial models allow for overdispersion. The variance function is given as

$$V(\mu_i) = \mu_i + \phi\mu_i^2,$$

where ϕ ($\phi > 0$) is the parameter that characterizes overdispersion. Since negative binomial distributions have the parameter ϕ twisted with $\boldsymbol{\mu}$, an iterative ML procedure for estimation is given in Algorithm 1.

2.2 Growing a Tree

We build tree by recursive partitioning as classical decision trees. Instead of using a constant fit in a node, we fit a log-linear model for the data within

Algorithm 1 Pseudocode for NB estimation

procedure NB($\mathbf{y}, maxit, tol$)
  $\hat{\boldsymbol{\mu}}^{(0)} \leftarrow \arg\max \mathcal{L}_{pois}(\boldsymbol{\mu}; \mathbf{y})$
  $\hat{\phi}^{(0)} \leftarrow \arg\max \mathcal{L}_{nb}(\phi; \mathbf{y}, \hat{\boldsymbol{\mu}}^{(0)})$
  $i \leftarrow 1$
  while true **do**
    $\hat{\boldsymbol{\mu}}^{(i)} \leftarrow \arg\max \mathcal{L}_{nb}(\boldsymbol{\mu}; \mathbf{y}, \hat{\phi}^{(i-1)})$
    $\hat{\phi}^{(i)} \leftarrow \arg\max \mathcal{L}_{nb}(\phi; \mathbf{y}, \hat{\boldsymbol{\mu}}^{(i)})$
    $\Delta \leftarrow |\mathcal{L}_{nb}(\mathbf{y}, \hat{\boldsymbol{\mu}}^{(i)}, \hat{\phi}^{(i)}) - \mathcal{L}_{nb}(\mathbf{y}, \hat{\boldsymbol{\mu}}^{(i-1)}, \hat{\phi}^{(i-1)})|$
    if $\Delta < tol$ **then**
      return $\hat{\boldsymbol{\mu}}^{(i)}$, $\hat{\phi}^{(i)}$
    end if
    $i \leftarrow i + 1$
    if $i > maxit$ **then**
      return max iteration reached.
    end if
  end while
end procedure

the node. In this study, we only consider binary trees.

Starting from the root node, we recursively partition the data into subsets which consist of branches. The procedure is described as follows:

1. Fit a log-linear model once to all observations in the current node.

2. Assess whether a split is necessary. Stop if it is not.

3. Search the split point that locally optimizes the objective function.

4. Split the node into children nodes and repeat the procedure.

The objective function in generalized linear models is usually the (negative) likelihood function to maximize (minimize). To be consistent with this objective, we split the node at the value that maximizes the sum of likelihood of the children nodes.

To assess the necessity of a split, we formulate a hypothesis test. Suppose the data contain n observations and the log-linear model takes the form of

$$\mathbb{E}(y_i) = \exp(\mathbf{x}_i^\top \boldsymbol{\beta}_i). \tag{2.2}$$

The split is considered necessary if the null hypothesis of

$$\mathrm{H}_0 : \boldsymbol{\beta}_1 = \cdots = \boldsymbol{\beta}_n = \boldsymbol{\beta} \tag{2.3}$$

is rejected, where $\boldsymbol{\beta}_n$ is the parameter vector for the n-th observation. To control the size of tree, we control the probability of a Type I error of the test. The Type I error is defined as the probability of a unnecessary splitting.

The hypothesis can be expressed in another way if the value where the splitting happens is given. Denote the splitting variable by s, and its splitting value by v (for continuous variables) or c (for categorical variables). We introduce the dummy variable d in model

$$\log \mu = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \beta d, \tag{2.4}$$

where

$$d = \mathrm{I}(s \leq v)s, \tag{2.5}$$

or

$$d = \mathrm{I}(s \in c). \tag{2.6}$$

Then we test the hypothesis

$$\mathrm{H}_0 : \beta = 0 \text{ vs } \mathrm{H}_1 : \beta \neq 0. \tag{2.7}$$

Note that it is not necessary to split a categorical variable which is already considered as an independent variable in the log-linear model, because the coded variable (not the dummy variable for splitting) is equivalent to stratification.

An intuitive value of v is the one determined by the splitting rule that is maximizing the likelihood. However, such a value estimated from the data may violate the distribution of test statistic under the null hypothesis and make test incorrect (high probability of a type I error) [97]. We use the median value if the variable is continuous for the division [97]. If the variable is categorical, we add the variable as a regressor directly.

2.2.1 Test for Splitting

There are many methods of testing $\beta = 0$ in (2.7). In context of generalized linear models, Wald test and likelihood ratio test are often used [1]. Besides we can also use bootstrap tests. We define parametric bootstrapped LRT test as

1. Compute the LRT statistic λ from the data.
2. Generate bootstrap samples from the null model.
3. Form a $1 - \alpha$ confidence interval with the statistic λ_b from the samples generated from the null distribution.

4. Reject H_0 if the above confidence interval does not contain λ.

The above tests needs a prespecified splitting value. Although testing a nearly equal-size split would avoid deficiency of small sample size, it might be inappropriate in some cases. Hence we use another class of tests for parameter instability that work with (2.3).

Let $u(y \mid \boldsymbol{\beta})$ be the score function. Under H_0,

$$W_n(t, \boldsymbol{\beta}) = \frac{1}{\sqrt{n}} \sum_{i \leq [nt]} u(y_i, \boldsymbol{\beta}) \rightarrow_d Z(t), \quad 0 \leq t \leq 1 \tag{2.8}$$

where $Z(t)$ is a Gaussian process with zero mean and covariance function $\min(t_1, t_2)B(\boldsymbol{\beta})$ and $B(\boldsymbol{\beta})$ is the variance of score. Then a stochastic process defined as

$$M_n(t) = B(\boldsymbol{\beta})^{-1/2} W(t, \boldsymbol{\beta}) \tag{2.9}$$

converges to a p-dimensional Brownian motion.

Define two test statistics [119] as

$$\frac{1}{n} \sum_{i=1}^{n} \left\| M_n\left(\frac{i}{n}\right) \right\|_2^2 \tag{2.10}$$

and

$$\max_{t,k} |M_{n,k}(t)| \tag{2.11}$$

The idea behind such tests is that if any observation has an different parameter value from the others, the test statistics will deviate far away from the zero mean. We denote (2.10) by T_{L_2} and (2.11) by T_{L_∞}. The critical values of (2.10) are tabulated in [46], and the calculation of (2.11) is given in [115].

2.2.2 Simulation Study

In this section, we conducted a simulation study with the models from [21]. The models are defined as follows,

- Model 3:

$$\mu = \exp(-0.6931 + 0.497x),\ x \sim U(0, 5)$$

- Model 6:

$$\mu = \begin{cases} \exp(-0.6931 + 0x) & x \leq 2.5 \\ \exp(-3.178 + 0.994x) & x > 2.5 \end{cases},\ x \sim U(0, 5)$$

- Model 7:

$$\mu = \exp(|x|),\ x \sim U(-2, 2)$$

- Model 8:

$$\mu = \begin{cases} \exp(2x + 4) & x \leq -1 \\ \exp(2|x|) & -1 < x \leq 1 \\ \exp(-2x + 4) & x > 1 \end{cases},\ x \sim U(-2, 2)$$

- Model 9:

$$\mu = \exp(-0.5 + 0.8x_1 - 0.5x_2),\ x_1, x_2 \sim U(0, 3)$$

- Model 10:

$$\mu = \exp((-1)^c(0.4x_1 - 0.4x_2),\ c \in \{0,1\}, x_1 \sim U(0,5), x_2 \sim U(0,2)$$

Among those models, Model 3 and 9 do not need split, Model 6, 7 and 9 need one split, and Model 8 needs 3 splits.

The responses are generated from Poisson distribution without overdispersion and from the negative binomial distribution with corresponding mean and overdispersion parameter $\phi = 1$. Poisson likelihood is used for all tests when $\phi = 0$ and negative binomial likelihood is used for Wald, LRT and bootstrap tests and quasi-Poisson is used for the remaining two instability tests when $\phi = 1$.

We follow the same setting as that in [21]. We simulate 200 samples from each model. Each sample contains 120 observations except that it is 300 for overdispersed Model 8 to adapt multiple splits and high variance due to overdispersion. Bootstrap size is 1000. The significance level is 0.05.

The probability of a Type I error and Power of the simulation study are shown in Table 2.1. The Wald test is not performed on categorical variables (Model 10) since the coding might be more than one dummy variable. It only shows the probability of at least one split for the bootstrapped LRT to compare with LRT. The first three tests perform poorly on Model 8 because of the particular function form. The function is symmetric about 0 and so are the observations which lead to similar estimation before and after splitting, a line parallel to the x-axis so that the tests do not reject the null.

Table 2.1: Probability of a Type I error and Power

	Model	# of splits	Wald	LRT	P.Boot.LRT	T_{L_∞}	T_{L_2}
$\phi = 0$	3	>0	0.055	0.055	0.045	0.015	0
	6	1	0.11	0.125	0.08 (# > 0)	0.72	0.71
		2	0	0		0	0
		3	0.005	0.005		0.005	0
	7	1	0.985	0.985		0.985	0.985
		2	0.015	0.015		0.015	0.015
	8	1	0.01	0.01	0.035 (# > 0)	0.04	0.04
		2	0.045	0.045		0.385	0.51
		≥3	0.045	0.045		0.247	0.347
	9	>0	0.045	0.045	0.085 (# > 0)	0.015	0.015
	10	>0	NA	1	1 (# > 0)	1	1
$\phi = 1$	3		0.06	0.055	0.05 (# > 0)	0.03	0.025
	6	1	0.15	0.15	0.114 (# > 0)	0.41	0.315
		2	0.005	0		0.005	0
	7	1	0.995	0.995		0.315	0.95
		2	0.005	0.005		0.005	0.015
	8	1	0.005	0.01	0.03 (# > 0)	0.095	0.041
		2	0.035	0.02		0.333	0.424
		≥3	0.015	0.02		0.199	0.251
	9	>0	0.075	0.06	0.023 (# > 0)	0.05	0.035
	10	>0	NA	0.94	1 (# > 0)	0.835	0.85

2.3 Test for Over-dispersion

Though the negative binomial distribution is used to deal with overdispersion, it is still necessary to know if the data are truly overdispersed in practice. Suppose the variance function is $\text{var}(\mu) = \mu(1 + \phi\mu)$. Over-dispersion exists if $\phi > 0$. Then we have the hypothesis:

$$\text{H}_0 : \ \phi = 0 \text{ vs } \text{H}_1 : \ \phi > 0. \tag{2.12}$$

Here we consider several tests for overdispersion. The first one is the likelihood ratio test (LRT),

$$T = -2\left[\ell_{\text{Poisson}}(\mu|\mathbf{x}) - \ell_{\text{negative binomial}}(\mu, \phi|\mathbf{x})\right] \tag{2.13}$$

where $\ell(\mu, \phi|\mathbf{x})$ is the log likelihood [50]. The test statistic T converges in distribution to chi-squared distribution with 1 degree of freedom as sample size goes to infinity. We reject H_0 if $T > \chi^2_{\alpha,1}$.

Secondly, if the variance function takes another form as $\text{var}(y) = \varphi\mu$, overdispersion exists if $\varphi > 1$. The new dispersion parameter φ can be estimated through the Pearson χ^2 statistic

$$\hat{\varphi} = \frac{\chi^2_p}{n - p}. \tag{2.14}$$

Base on the estimator, we propose a nonparametric bootstrap test. Reject no overdispersion if 1 is on the left of the bootstrap confidence interval.

Thirdly a score test was given in [16, 117] as

$$T = \left(\frac{\sum_{i=1}^{n}((y_i - \hat{\mu}_i)^2 - y_i)}{\sqrt{2\sum_{i=1}^{n}\hat{\mu}_i^2}} \right)^2 \overset{n\to\infty}{\to} \chi_1^2 \tag{2.15}$$

It is obvious that overdispersion exists if the test statistic deviates far beyond 1.

We also do a simulation study for those tests. The simulation was conducted with different combinations of sample size (20, 50, 100), value of overdispersion (0, 0.05, 0.1, 0.2), number of covariates (2, 4, 6, 8) and logarithm of ground mean (2, 2, 4, 5). The covariates are uniformly random in $[-1, 1]$ and the coefficients are sampled from standard normal distribution except β_0. We used log-linear model to obtain the mean of response and generate responses by Poisson or negative binomial random variables according to the overdispersion value. The result is given in Table 2.2.

2.4 Missouri Lung Cancer

The Missouri lung cancer data contain death, age, sex and populations of 115 counties in each age and sex category. The counties were given in coded format and thus the true names are unidentifiable.

The raw response is the number of deaths. Obviously it is proportional to the population (size). Hence the mortality rate is modeled instead. The interest is the association between the mortality rate and covariates. The regression variables are sex and age. Sex contains only two categories, female and male. It is not necessarily a splitting variable. The ages are grouped

Table 2.2: Probability of a Type I Error and Power of Test for Over-dispersion

n	$\log \beta_0$	ϕ	p	LRT	Boot Quasi	LM
20	2	0	2	0	0	0.01
50	2	0	2	0.02	0.01	0.02
20	4	0	2	0	0	0
50	4	0	2	0	0.01	0.01
20	2	0.05	2	0.01	0.01	0.01
50	2	0.05	2	0.43	0.4	0.5
20	4	0.05	2	0.01	0.01	0.04
50	4	0.05	2	0.48	0.31	0.55
20	2	0.2	2	0.27	0.18	0.33
50	2	0.2	2	0.95	0.94	0.95
20	4	0.2	2	0.14	0.11	0.19
50	4	0.2	2	0.7	0.5	0.78
20	2	0	4	0	0	0
50	2	0	4	0	0.02	0.02
20	4	0	4	0	0	0.04
50	4	0	4	0.02	0.01	0.03
20	2	0.05	4	0.01	0	0.04
50	2	0.05	4	0.16	0.11	0.2
20	4	0.05	4	0.5	0.18	0.56
50	4	0.05	4	0.18	0.11	0.23
20	2	0.2	4	0.05	0.01	0.03
50	2	0.2	4	0.03	0.04	0.07
20	4	0.2	4	0.09	0.03	0.15
50	4	0.2	4	1	1	1
20	2	0	8	0	0	0
50	2	0	8	0	0	0
20	4	0	8	0	0	0
50	4	0	8	0.01	0	0.03
20	2	0.05	8	0	0	0
50	2	0.05	8	0.02	0.01	0.01
20	4	0.05	8	0	0	0
50	4	0.05	8	0.32	0.03	0.12
20	2	0.2	8	0	0	0
50	2	0.2	8	0.69	0.13	0.68
20	4	0.2	8	0.05	0	0.03
50	4	0.2	8	0.9	0.68	0.72

into 45–54, 55-64, 65–74 and 75+. It is naturally coded as ordered if only linear relationship with the mortality rate is considered. We code it as 45, 55, 65 and 75. Since the size is considered the exposure, it is not included as a regression variable. However, size is a splitting variable because it may be a representation of other geographic information. So sex and age are used as regression variables, and age and size are used as splitting variables.

Trees with two models are fitted. One uses Poisson likelihood and the other uses negative binomial likelihood to assess overdispersion. The minimum terminal sizes are chosen to be 5 and 20. The stopping rule uses meanL2 test.

The results are shown in Figure 2.1 for Poisson (20 min. terminal node size) and negative binomial (5 and 20 min. terminal node size) and Figure 2.2 for Poisson (5 min. terminal size). When the minimum terminal node size is set to 5, the Poisson tree splits node 6 into two sub nodes. The fitted log-linear models are shown in Tables 2.3, 2.4 and 2.5.

Sex is always significant in all terminal nodes. It implies the mortality rates are different between males and females, and positive estimates suggest that males have higher chance to die than females from lung cancer. The root node is split at age 65 for both models. Age is insignificant in the whole sample. It is significant for people who are younger than 65, but not for the older people. In the negative binomial tree, overdispersion is significant for the whole sample, but it is insignificant in nodes 10 and 11 in Figure 1. These nodes contain people who are at least 65 years old and living in counties with low population.

Table 2.3: Poisson (min. terminal = 5)

Node	Parameter	Estimate	p value
1	(Intercept)	-7.9151	0.0000
	SexM	1.5223	0.0000
	Age	0.0452	0.0000
4	(Intercept)	-10.3281	0.0000
	SexM	1.2928	0.0000
	Age	0.0927	0.0000
5	(Intercept)	-9.8033	0.0000
	SexM	1.1466	0.0000
	Age	0.0894	0.0000
7	(Intercept)	-4.9236	0.0000
	SexM	1.6740	0.0000
	Age	0.0025	0.3305
8	(Intercept)	-5.6923	0.0000
	SexM	1.7105	0.0000
	Age	0.0077	0.1377
9	(Intercept)	-5.4495	0.0000
	SexM	1.8727	0.0000
	Age	0.0051	0.1145

2.5 Summary

In this chapter, we embed log-linear models with negative binomial likelihood into tree to relax the assumption of linearity and allow overdispersion.

To control the size of tree, tests for parameter instability are introduced in the procedure of tree growth. The node splits only if the sufficiency of linear relationship is rejected by the test. Besides, we test the overdispersion to see if extra-Poisson is present.

We validate our method with simulation study and apply it to modeling mortality rate of Missouri lung cancer cases. The result shows significance of effects and overdispersion change along the path of tree which are not reveal

Table 2.4: Poisson (min. terminal = 20)

Node	Parameter	Estimate	p vlaue
1	(Intercept)	-7.9151	0.0000
	SexM	1.5223	0.0000
	Age	0.0452	0.0000
5	(Intercept)	-9.7674	0.0000
	SexM	1.0900	0.0000
	Age	0.0885	0.0000
7	(Intercept)	-4.9236	0.0000
	SexM	1.6740	0.0000
	Age	0.0025	0.3305
8	(Intercept)	-6.0290	0.0000
	SexM	1.1292	0.0000
9	(Intercept)	-5.3129	0.0000
	SexM	1.4444	0.0000
10	(Intercept)	-5.6923	0.0000
	SexM	1.7105	0.0000
	Age	0.0077	0.1377
11	(Intercept)	-5.4495	0.0000
	SexM	1.8727	0.0000
	Age	0.0051	0.1145

by the simple Poisson model.

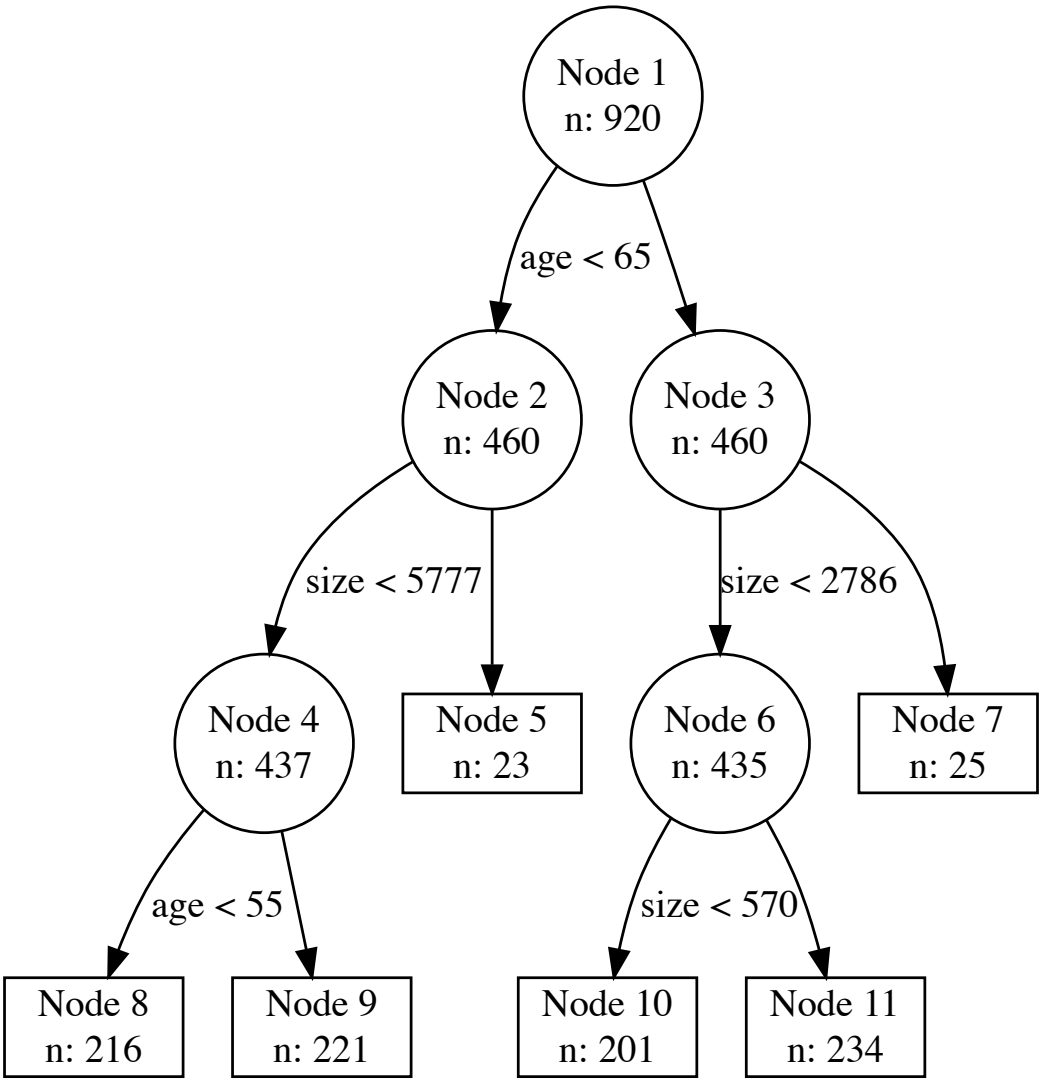


Figure 2.1: Regression tree of Missouri lung cancer. Poisson (min. terminal = 20), negative binomial (min. terminal = 5, 20). The circles represent intermediate nodes, and the boxes represent the terminal nodes. The number of node and sample size are printed in nodes. The splitting variable and value are printed on the left branches.

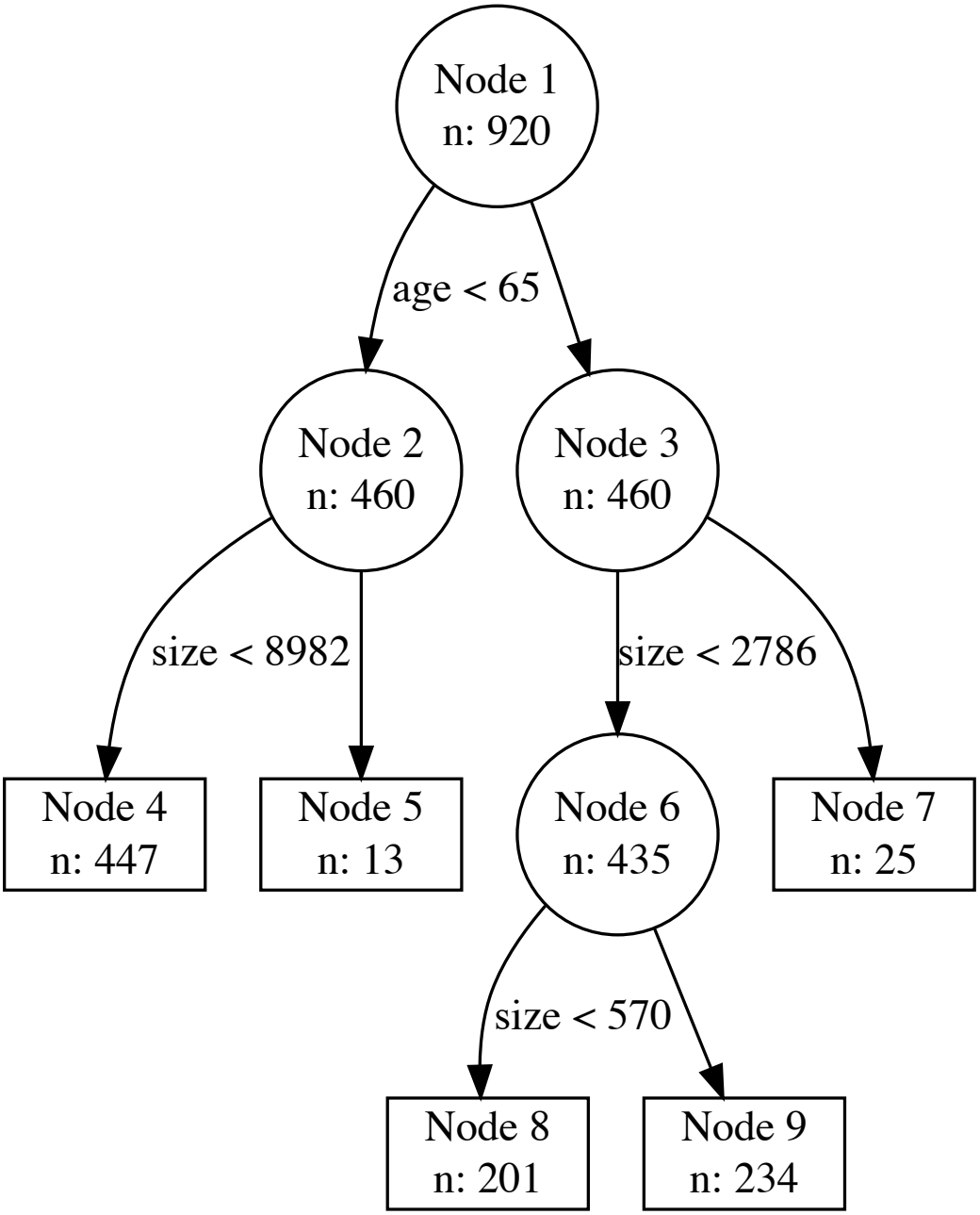


Figure 2.2: Regression tree of Missouri lung cancer. Poisson (min. terminal = 5)

Table 2.5: Negative binomial (min. terminal = 5, 20)

Node	Parameter	Estimate	p value
1	(Intercept)	-7.9034	0.0000
	ϕ	0.1069	0.0000
	SexM	1.4402	0.0000
	Age	0.0458	0.0000
5	(Intercept)	-9.5688	0.0000
	ϕ	0.1424	0.0000
	SexM	1.0601	0.0000
	Age	0.0849	0.0000
7	(Intercept)	-4.8927	0.0000
	ϕ	0.0022	0.0000
	SexM	1.6735	0.0000
	Age	0.0020	0.5539
8	(Intercept)	-6.0290	0.0000
	ϕ	0.0304	0.0215
	SexM	1.1292	0.0000
9	(Intercept)	-5.3129	0.0000
	ϕ	0.0222	0.0000
	SexM	1.4444	0.0000
10	(Intercept)	-5.7075	0.0000
	ϕ	0.0000	0.6323
	SexM	1.7105	0.0000
	Age	0.0079	0.1359
11	(Intercept)	-5.4769	0.0000
	ϕ	0.0071	0.0584
	SexM	1.8734	0.0000
	Age	0.0055	0.1284

Chapter 3

Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains

A small number of shared factors, or a low-dimensional trajectory can explain the interdependence of simultaneously recorded population of neurons when governed by underlying low-dimensional dynamics. Recovering these latent trajectories, particularly from single-trial population recordings, may help to understand the dynamics that drive neural computation. However, inferring trajectories from data is a difficult statistical problem. Here, we propose a practical and efficient inference method, called the variational latent Gaussian process (vLGP). The vLGP combines a generative model with a history-dependent point process observation together with a smoothness prior on the latent trajectories. It improves upon earlier methods for recovering latent trajectories, which assume either observation models inappropriate for point processes or linear dynamics. We compare and validate vLGP on both simulated datasets and population recordings from the primary visual cortex (V1). In the V1 dataset, we find that vLGP achieves substantially higher performance than previous methods for predicting omitted spike trains, as well as capturing both the toroidal topology of visual stimuli space, and the

noise-correlation. These results show that vLGP is a robust method with a potential to reveal hidden neural dynamics from large-scale neural recordings.

3.1 Generative model

Suppose we simultaneously observe spike trains from N neurons. Let $(y_{t,n})_{t=1,\dots,T}$ denote the spike count time-series from the n-th neuron for a small time bin. We assume the following parametric form of the conditional intensity function $\lambda^*(\cdot)$ for the point process likelihood [27, 80]:

$$\begin{aligned} \log p(y_{t,n} \,|\, \mathbf{x}_t, \mathbf{h}_{t,n}, \boldsymbol{\alpha}_n, \boldsymbol{\beta}_n) &= y_{t,n} \log \lambda^*(t, n \,|\, \mathbf{h}_{t,n}) - \lambda^*(t, n \,|\, \mathbf{h}_{t,n}), \\ \lambda^*(t, n \,|\, \mathbf{h}_{t,n}) &= \exp\left(\boldsymbol{\alpha}_n^\top \mathbf{x}_t + \boldsymbol{\beta}_n^\top \mathbf{h}_{t,n}\right), \end{aligned} \tag{3.1}$$

where $\mathbf{x}_t$ is a latent process and $\mathbf{h}_{t,n} = [1, y_{t-p,n}, y_{t-p+1,n}, \dots, y_{t-1,n}]^\top$ denotes the spike history vector [108, 96]. Each neuron is directly influenced by the observed self-history[1] with weight $\boldsymbol{\beta}_n$ and also driven by the common latent process with weight $\boldsymbol{\alpha}_n$ (Fig. 3.1). Neurons are conditionally independent otherwise: all trial-to-trial variability is attributed either to the latent process or individual point process noise (c.f., [42, 30, 74]).

The vector $\mathbf{x}_t$ denotes the L-dimensional latent process at time t. We assume that $L \ll N$, since we are looking for a small number of latent processes that explain the structure of a large number of observed neurons. The vector $\boldsymbol{\beta}_n$ consists of the weights of the spike history and a time-independent bias term of the log firing rate for each neuron, and $\mathbf{h}_{t,n}$ is a vector of length $(1+p)$

[1] It is straightforward to add external covariates similar to the self-history in this point process regression [92].

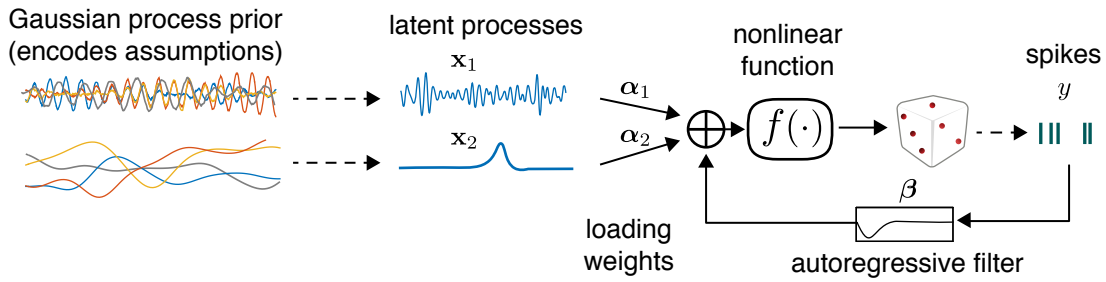


Figure 3.1: Generative model schematic for one neuron driven by two latent processes. Every neuron in the observed population are driven by the same set of latent processes. The inferred latent processes are more likely to be smooth, as assumed by the smooth Gaussian process prior. The point nonlinearity is fixed to be exponential $f(\cdot) = \exp(\cdot)$.

containing the dummy value 1 for the bias and p time-step spike self-history. This parametrization assumes that at most p bins in the past influence the current intensity.

Under conditional independence, the joint distribution (data likelihood) of N spike trains is given by,

$$p(y_{1\ldots T,1\ldots N} \,|\, \mathbf{x}_{1\ldots T}, \boldsymbol{\alpha}_{1\ldots N}, \boldsymbol{\beta}_{1\ldots N}) = \prod_{t=1}^{T} \prod_{n=1}^{N} p(y_{t,n} \,|\, \mathbf{x}_t, \mathbf{h}_{t,n}, \boldsymbol{\alpha}_n, \boldsymbol{\beta}_n). \tag{3.2}$$

Note that this model is not identifiable (see later sections for further discussions): $\boldsymbol{\alpha}_n^T \mathbf{x}_t = (\boldsymbol{\alpha}_n^T \mathbf{C})(\mathbf{C}^{-1}\mathbf{x}_t) = \boldsymbol{\alpha'}_n^T \mathbf{x'}_t$ where $\mathbf{C}$ is an arbitrary $L \times L$ invertible matrix. Also, the mean of latent process $\mathbf{x}$ can be traded off with the bias term in $\boldsymbol{\beta}$.

Our assumptions about the latent process — namely the smoothness over time in this paper — are encoded in the prior distribution over the latent process. We use the popular Gaussian process (GP) framework [101] for flexible

prior design of each dimension $x_l(t)$ independently:

$$x_l(t) \sim \mathcal{GP}(\mu_l, \kappa_l) \tag{3.3}$$

where $\mu_l(t)$, and $\kappa_l(t, s)$ are mean and covariance functions, respectively. When time is discretized, the GP prior reduces to a multi-variate Gaussian distribution over the latent time series. We use the following form:

$$p(\mathbf{x}_l) = \mathcal{N}(\mathbf{x}_l \,|\, \mathbf{0}, \mathbf{K}_l), \qquad l = 1, \ldots, L. \tag{3.4}$$

For the analyses in this manuscript, we choose the squared exponential covariance function [101] for general smoothness over time,

$$\mathrm{cov}(x_{t,l}, x_{s,l}) = \sigma_l^2 \exp(-\omega_l (t - s)^2). \tag{3.5}$$

where σ_l and ω_l are hyperparameters corresponding to the magnitude and inverse time scale of the latent process, respectively.

3.2 Variational inference

Our goal is to infer the posterior distribution over the latent process and fit the model parameters given the observed data. By Bayes' theorem, the posterior distribution of the latent process is,

$$p(\mathbf{x}_{1\ldots L} \,|\, \mathbf{y}_{1\ldots N}) = \frac{p(\mathbf{y}_{1\ldots N} \,|\, \mathbf{x}_{1\ldots L}) p(\mathbf{x}_{1\ldots L})}{p(\mathbf{y}_{1\ldots N})}, \tag{3.6}$$

However, unlike in GPFA, the posterior under a point process likelihood and Gaussian process prior does not have an analytical form [91]. Consequently, we must turn to an approximate inference technique. We employ variational inference, which aims to find an approximate distribution $q(\mathbf{x})$ of the intractable true posterior $p(\mathbf{x} \mid \mathbf{y})$. We can introduce this approximate posterior into the likelihood by re-writing it as,

$$\log p(\mathbf{y}_{1\ldots N}) = \mathcal{E}_q[\log p(\mathbf{y}_{1\ldots N})] = \mathcal{E}_q\left[\log \frac{p(\mathbf{y}_{1\ldots N}, \mathbf{x}_{1\ldots L})}{q(\mathbf{x}_{1\ldots L})} \cdot \frac{q(\mathbf{x}_{1\ldots L})}{p(\mathbf{x}_{1\ldots L} \mid \mathbf{y}_{1\ldots N})}\right] \tag{3.7}$$

$$= \underbrace{\mathcal{E}_q\left[\log \frac{p(\mathbf{y}_{1\ldots N}, \mathbf{x}_{1\ldots L})}{q(\mathbf{x}_{1\ldots L})}\right]}_{\mathcal{L}(q)} + \underbrace{\mathcal{E}_q\left[\log \frac{q(\mathbf{x}_{1\ldots L})}{p(\mathbf{x}_{1\ldots L} \mid \mathbf{y}_{1\ldots N})}\right]}_{D_{\mathrm{KL}}(q\|p)}, \tag{3.8}$$

where $\mathcal{E}_q$ denotes an expectation over $q(\mathbf{x})$, and $D_{\mathrm{KL}}(q\|p)$ is the Kullback-Leibler divergence, which measures the difference in the true posterior and its variational approximation. Since $D_{\mathrm{KL}}(q\|p)$ is non-negative, $\mathcal{L}(q)$ is the lower bound for the marginal likelihood. Finding an approximate posterior q close to the true posterior by minimizing the Kullback-Leibler divergence is equivalent to maximizing the lower bound $\mathcal{L}(q)$.[2]

We further assume the q distribution factorizes into Gaussian distributions with respect to each dimension of the latent process, so that

$$q(\mathbf{x}_{1\ldots L}) = \prod_{l=1}^{L} \mathcal{N}(\mathbf{x}_l \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l). \tag{3.9}$$

[2] $\mathcal{L}(q)$ is often called the Evidence Lower BOund (ELBO).

We then obtain

$$\begin{aligned}\mathcal{L}(q) =& \sum_{t=1}^{T}\sum_{n=1}^{N}\mathcal{E}_q[\log p(y_{t,n} \mid \mathbf{x}_t, \mathbf{h}_{t,n}, \boldsymbol{\alpha}_n, \boldsymbol{\beta}_n)] - \sum_{l=1}^{L}\mathcal{E}_q\left[\log \frac{q(\mathbf{x}_{1\ldots L} \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}{p(\mathbf{x}_{1\ldots L} \mid \mathbf{K}_l)}\right] \\ =& \sum_{t=1}^{T}\sum_{n=1}^{N}[y_{t,n}(\boldsymbol{\alpha}_n^\top \boldsymbol{\mu}_t + \boldsymbol{\beta}_n^\top \mathbf{h}_{t,n}) - \exp(\boldsymbol{\alpha}_n^\top \boldsymbol{\mu}_t + \boldsymbol{\beta}_n^\top \mathbf{h}_{t,n} + \frac{1}{2}\boldsymbol{\alpha}_n^\top \boldsymbol{\Sigma}_t \boldsymbol{\alpha}_n)] \\ & - \frac{1}{2}\sum_{l=1}^{L}[\boldsymbol{\mu}_l^\top \mathbf{K}_l^{-1}\boldsymbol{\mu}_l + \operatorname{tr}(\mathbf{K}_l^{-1}\boldsymbol{\Sigma}_l) - \log\det(\mathbf{K}_l^{-1}\boldsymbol{\Sigma}_l) - T].\end{aligned} \tag{3.10}$$

where T is the number of total time steps, and each temporal slice $\boldsymbol{\mu}_t$ is a vector of posterior means of the L latent variables at time t. Each temporal slice $\boldsymbol{\Sigma}_t$ is a diagonal matrix whose diagonal contains the variances of the L latent variables at time t.

Variational inference for the entire posterior over latents, parameters, and hyperparameters all be formulated in terms of maximizing (3.10). We sequentially update all parameters coordinate-wise; each conditional update turns out to be a convex-optimization problem except for hyperparameters as explained below. The algorithm is summarized in Algorithm 2.

Our algorithm scales linearly in space $\mathcal{O}(Ts)$ and time $\mathcal{O}(Tr^2L)$ per iteration (for a fixed hyperparameter) where $s = \max(rL, pN)$ thanks to the rank-r incomplete Cholesky factor of the prior covariance matrix. For comparison, time complexity of GPFA is $\mathcal{O}(T^3L^3)$, and that of PLDS is $\mathcal{O}(T(L^3 + LN))$.

3.2.1 Posterior over the latent process

The variational distribution q_l is assumed to be Gaussian and thus determined only by its mean $\boldsymbol{\mu}_l$ and covariance $\boldsymbol{\Sigma}_l$. The optimal solution is therefore

Algorithm 2 Pseudocode for vLGP inference

procedure vLGP($\mathbf{y}_{1\ldots T}$, $\mathbf{h}_{1\ldots T,1\ldots N}$, $\sigma^2_{1\ldots L}$, $\omega_{1\ldots L}, tol, k$)
  $\mathbf{G}_l = \text{ichol}(\sigma_l^2, \omega_l)$, $l = 1 \ldots L$ ▷ construct incomplete Cholesky decomposition [5]
  Initialize $\boldsymbol{\alpha}_n$ and $\boldsymbol{\mu}_l$ by factor analysis
  $\boldsymbol{\beta}_n \leftarrow (\mathbf{h}^\top_{1\ldots T,n}\mathbf{h}_{1\ldots T,n})^{-1}\mathbf{h}^\top_{1\ldots T,n}\mathbf{y}_{1\ldots T}$, $n = 1 \ldots N$ ▷ linear regression
  while true **do**
    for $l \leftarrow 1, \ldots, L$ **do**
      $\lambda_{t,n} \leftarrow \boldsymbol{\alpha}_n^\top \boldsymbol{\mu}_t + \boldsymbol{\beta}_n^\top \mathbf{h}_{t,n} + \frac{1}{2}\boldsymbol{\alpha}_n^\top \boldsymbol{\Sigma}_t \boldsymbol{\alpha}_n$, $t = 1 \ldots T, n = 1 \ldots N$
      $\mathbf{u}_l \leftarrow \mathbf{G}_l\mathbf{G}_l^\top(\mathbf{y} - \boldsymbol{\lambda})\boldsymbol{\alpha}_l - \boldsymbol{\mu}_l^{old}$
      $\mathbf{B}_l \leftarrow \mathbf{G}_l^\top \text{diag}(\mathbf{W}_l)\mathbf{G}_l$
      $\boldsymbol{\mu}_l^{new} \leftarrow \boldsymbol{\mu}_l^{old} + [\mathbf{I}_T - \mathbf{G}_l\mathbf{G}_l^\top\mathbf{W}_l + \mathbf{G}_l\mathbf{B}_l(\mathbf{I}_r + \mathbf{B}_l)^{-1}\mathbf{G}_l^\top\mathbf{W}_l]\mathbf{u}_l$ ▷ Newton-step for $\boldsymbol{\mu}$
      $\boldsymbol{\mu}_l^{new} \leftarrow (\boldsymbol{\mu}_l^{new} - \bar{\boldsymbol{\mu}}_l^{new})$ ▷ constrain $\boldsymbol{\mu}$
    end for
    for $n \leftarrow 1, \ldots, N$ **do**
      $\lambda_{t,n} \leftarrow \boldsymbol{\alpha}_n^\top \boldsymbol{\mu}_t + \boldsymbol{\beta}_n^\top \mathbf{h}_{t,n} + \frac{1}{2}\boldsymbol{\alpha}_n^\top \boldsymbol{\Sigma}_t \boldsymbol{\alpha}_n$, $t = 1 \ldots T, n = 1 \ldots N$
      $\boldsymbol{\alpha}_n^{new} \leftarrow \boldsymbol{\alpha}_n^{old} + [(\boldsymbol{\mu} + \mathbf{V} \circ \boldsymbol{\alpha}_n^{old})^\top \text{diag}(\boldsymbol{\lambda}_n)(\boldsymbol{\mu} + \mathbf{V} \circ \boldsymbol{\alpha}_n^{old}) + \text{diag}(\mathbf{V}^\top\boldsymbol{\lambda}_n)]^{-1}[\boldsymbol{\mu}^\top\mathbf{y}_n - (\boldsymbol{\mu} + \mathbf{V} \circ \boldsymbol{\alpha}_n^{old})^\top\boldsymbol{\lambda}_n]$ ▷ Newton-step for $\boldsymbol{\alpha}$
      $\boldsymbol{\beta}_n^{new} \leftarrow \boldsymbol{\beta}_n^{old} + [\mathbf{h}_n^\top \text{diag}(\boldsymbol{\lambda}_n)\mathbf{h}_n]^{-1}\mathbf{h}_n^\top(\mathbf{y}_n - \boldsymbol{\lambda}_n)$ ▷ Newton-step for $\boldsymbol{\beta}$
    end for
    $\boldsymbol{\alpha}_l^{new} \leftarrow \boldsymbol{\alpha}_l^{new}/\|\boldsymbol{\alpha}_l^{new}\|$, $l = 1 \ldots L$ ▷ constrain $\boldsymbol{\alpha}$
    $W \leftarrow \boldsymbol{\lambda}\boldsymbol{\alpha}^{2\top}$ ▷ update diagonals of W
    $\mathbf{B}_l \leftarrow \mathbf{G}_l^\top \text{diag}(\mathbf{W}_l)\mathbf{G}_l$, $l = 1 \ldots L$
    $\mathbf{V}_{1\ldots T,l} \leftarrow [\mathbf{G}_l \circ (\mathbf{G}_l - \mathbf{G}_l\mathbf{B}_l + \mathbf{G}_l\mathbf{B}_l(I_k + \mathbf{B}_l)^{-1}\mathbf{B}_l)]\mathbf{1}$, $l = 1 \ldots L$
    Optimize hyperparameters with the gradient in (3.35) and update $\mathbf{G}_{1\ldots L}$ every k iterations
    if $\|(\boldsymbol{\mu}^{new}_{1\ldots L}, \boldsymbol{\alpha}^{new}_{1\ldots N}, \boldsymbol{\beta}^{new}_{1\ldots N}) - (\boldsymbol{\mu}^{old}_{1\ldots L}, \boldsymbol{\alpha}^{old}_{1\ldots N}, \boldsymbol{\beta}^{old}_{1\ldots N})\| < tol$ **then**
      break
    end if
    $\boldsymbol{\mu}^{old}_{1\ldots L} \leftarrow \boldsymbol{\mu}^{new}_{1\ldots L}$, $\boldsymbol{\alpha}^{old}_{1\ldots N} \leftarrow \boldsymbol{\alpha}^{new}_{1\ldots N}$, $\boldsymbol{\beta}^{old}_{1\ldots N} \leftarrow \boldsymbol{\beta}^{new}_{1\ldots N}$
  end while
end procedure

obtained by

$$\boldsymbol{\mu}^{\star}_{1\ldots L}, \boldsymbol{\Sigma}^{\star}_{1\ldots L} = \underset{\boldsymbol{\mu}_{1\ldots L}, \boldsymbol{\Sigma}_{1\ldots L}}{\arg\max}\, \mathcal{L}(q), \tag{3.11}$$

while holding other parameters and hyperparameters fixed.

Denote the expected firing rate of neuron n at time t by $\lambda_{t,n}$,

$$\lambda_{t,n} = \mathcal{E}_q\left[\lambda^*(t, n \,|\, \mathbf{h}_{t,n})\right] = \exp\left(\boldsymbol{\beta}_n^\top \mathbf{h}_{t,n} + \boldsymbol{\alpha}_n^\top \boldsymbol{\mu}_t + \frac{1}{2}\boldsymbol{\alpha}_n^\top \boldsymbol{\Sigma}_t \boldsymbol{\alpha}_n\right). \tag{3.12}$$

The optimal $\boldsymbol{\mu}_l$ can be obtained by the Newton-Raphson method. The gradient and Hessian are given as

$$\nabla_{\boldsymbol{\mu}_l}\mathcal{L} = \sum_{t,n}(y_{t,n} - \lambda_{t,n})a_{n,l}\mathbf{e}_t - \mathbf{K}_l^{-1}\boldsymbol{\mu}_l, \tag{3.13}$$

$$\nabla^2_{\boldsymbol{\mu}_l}\mathcal{L} = -\sum_{t,n}\lambda_{t,n}a_{n,l}^2\mathbf{e}_t\mathbf{e}_t^\top - \mathbf{K}_l^{-1}. \tag{3.14}$$

where $\mathbf{e}_t$ is a vector of length T with value 1 at t and zero elsewhere. Note that the Hessian is negative definite, and hence this is a convex optimization given the other arguments and $\lambda_{t,n}$. In each iteration, the update is

$$\boldsymbol{\mu}_l^{new} = \boldsymbol{\mu}_l^{old} - (\nabla^2_{\boldsymbol{\mu}_l}\mathcal{L})^{-1}(\nabla_{\boldsymbol{\mu}_l}\mathcal{L}). \tag{3.15}$$

If we set the derivative w.r.t. $\boldsymbol{\Sigma}_l$ to 0,

$$\nabla_{\boldsymbol{\Sigma}_l}\mathcal{L} = -\frac{1}{2}\sum_{t,n}\lambda_{n,t}a_{n,l}^2\mathbf{e}_t\mathbf{e}_t^\top - \frac{1}{2}\mathbf{K}_l^{-1} + \frac{1}{2}\boldsymbol{\Sigma}_l^{-1} = 0, \tag{3.16}$$

we obtain the optimal covariance,

$$\boldsymbol{\Sigma}_l = \left(\mathbf{K}_l^{-1} + \sum_{t,n} \lambda_{t,n} a_{n,l}^2 \mathbf{e}_t \mathbf{e}_t^\top \right)^{-1} \tag{3.17}$$

$$= \left(\mathbf{K}_l^{-1} + \mathbf{W}_l \right)^{-1}. \tag{3.18}$$

where $\mathbf{W}_l = \sum_{t,n} \lambda_{t,n} a_{n,l}^2 \mathbf{e}_t \mathbf{e}_t^\top$ is a diagonal matrix. Therefore, there is no need for optimization of the covariance. This simple form of variational posterior covariance has been noted before [89]. Also note that $\nabla^2_{\boldsymbol{\mu}_l} \mathcal{L} = -\boldsymbol{\Sigma}_l^{-1}$.

There is a redundancy between the bias term in $\boldsymbol{\beta}$ and the mean $\boldsymbol{\mu}$. During optimization, we constrain the latent mean $\boldsymbol{\mu}$ by zero-centering, and normalize the loading $\boldsymbol{\alpha}$ by its max-norm latent-wise.

The prior covariance matrix $\mathbf{K}_l$ is large ($T \times T$) and is often severely ill-conditioned. We only keep a truncated incomplete Cholesky factor $\mathbf{G}$ [5] of size $T \times r$ where r is the rank of the resulting approximation,

$$\mathbf{K}_l \approx \mathbf{G}_l \mathbf{G}_l^\top, \tag{3.19}$$

which provides both a compact representation and numerical stability. Now, we derive key quantities that are necessary for a memory-efficient and numerically stable implementation. For convenience and without ambiguity, we omit the subscript l of all vectors and matrices below. By the matrix inversion lemma [101], we have

$$\boldsymbol{\Sigma} = (\mathbf{K}^{-1} + \mathbf{W})^{-1} = \mathbf{K} - \mathbf{K}(\mathbf{W}^{-1} + \mathbf{K})^{-1}\mathbf{K}. \tag{3.20}$$

and applying the lemma again

$$(\mathbf{W}^{-1} + \mathbf{K})^{-1} = \mathbf{W} - \mathbf{W}\mathbf{G}(\mathbf{I} + \mathbf{B})^{-1}\mathbf{G}^\top\mathbf{W}, \tag{3.21}$$

where $\mathbf{B} = \mathbf{G}^\top\mathbf{W}\mathbf{G}$. We obtain two useful identities as a result:

$$\boldsymbol{\Sigma} = \mathbf{G}\mathbf{G}^\top - \mathbf{G}\mathbf{B}\mathbf{G}^\top + \mathbf{G}\mathbf{B}(\mathbf{I} + \mathbf{B})^{-1}\mathbf{B}\mathbf{G}^\top, \tag{3.22}$$

$$\mathbf{K}^{-1}\boldsymbol{\Sigma} = \mathbf{I} - \mathbf{W}\mathbf{G}\mathbf{G}^\top + \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{B})^{-1}\mathbf{B}\mathbf{G}^\top. \tag{3.23}$$

With (3.22) and (3.23), we can avoid large matrices in above equations such as,

$$\mathrm{tr}[\mathbf{K}^{-1}\boldsymbol{\Sigma}] = T - \mathrm{tr}[\mathbf{B}] + \mathrm{tr}[\mathbf{B}(\mathbf{I} + \mathbf{B})^{-1}\mathbf{B}], \tag{3.24}$$

$$\log\det[\mathbf{K}^{-1}\boldsymbol{\Sigma}] = \log\det[\mathbf{I} - \mathbf{B} + \mathbf{B}(\mathbf{I} + \mathbf{B})^{-1}\mathbf{B}], \tag{3.25}$$

$$\mathrm{diag}(\boldsymbol{\Sigma}) = [\mathbf{G} \circ (\mathbf{G} - \mathbf{G}\mathbf{B} + \mathbf{G}\mathbf{B}(\mathbf{I}_k + \mathbf{B})^{-1}\mathbf{B})]\mathbf{1}, \tag{3.26}$$

$$\boldsymbol{\Sigma}\nabla_{\boldsymbol{\mu}}\mathcal{L} = (\mathbf{I} - \mathbf{G}\mathbf{G}^\top\mathbf{W} + \mathbf{G}\mathbf{B}(\mathbf{I}_k + \mathbf{B})^{-1}\mathbf{G}^\top\mathbf{W})\mathbf{u}, \tag{3.27}$$

where $\mathbf{1}$ is the all-ones vector, and $\mathbf{u} = \mathbf{G}\mathbf{G}^\top(\mathbf{y} - \boldsymbol{\lambda})\boldsymbol{\alpha}_l - \boldsymbol{\mu}$. In addition, by the one-to-one correspondence between $\mathbf{W}$ and $\boldsymbol{\Sigma}$, we use the diagonal of $\mathbf{W}$ as a representation of $\boldsymbol{\Sigma}$ in the algorithm. The derivations of useful identities are in Appendix A.1.

3.2.2 Weights

Denote the temporal slices of $\boldsymbol{\Sigma}_l$'s by $T \times L$ matrix $\mathbf{V}$. The optimal weights $\boldsymbol{\alpha}_n$ and $\boldsymbol{\beta}_n$ given the posterior over the latents can be obtained by the Newton-

Raphson method with the following derivatives and Hessians,

$$\nabla_{\mathbf{a}_n}\mathcal{L} = \boldsymbol{\mu}^\top(\mathbf{y}_n - \boldsymbol{\lambda}_n) - \operatorname{diag}(\mathbf{V}^\top\boldsymbol{\lambda}_n)\mathbf{a}_n, \tag{3.28}$$

$$\nabla^2_{\mathbf{a}_n}\mathcal{L} = -(\boldsymbol{\mu} + \mathbf{V}\circ\mathbf{1}\mathbf{a}_n^\top)^\top \operatorname{diag}(\boldsymbol{\lambda}_n)(\boldsymbol{\mu} + \mathbf{V}\circ\mathbf{1}\mathbf{a}_n^\top) - \operatorname{diag}(\mathbf{V}^\top\boldsymbol{\lambda}_n), \tag{3.29}$$

and

$$\nabla_{\boldsymbol{\beta}_n}\mathcal{L} = \mathbf{h}_n^\top(\mathbf{y}_n - \boldsymbol{\lambda}_n), \tag{3.30}$$

$$\nabla^2_{\boldsymbol{\beta}_n}\mathcal{L} = -\mathbf{h}_n^\top \operatorname{diag}(\boldsymbol{\lambda}_n)\mathbf{h}_n. \tag{3.31}$$

The updating rules are

$$\boldsymbol{\alpha}_n^{new} = \boldsymbol{\alpha}_n^{old} - (\nabla^2_{\boldsymbol{\alpha}_n}\mathcal{L})^{-1}\nabla_{\boldsymbol{\alpha}_n}\mathcal{L}, \tag{3.32}$$

$$\boldsymbol{\beta}_n^{new} = \boldsymbol{\beta}_n^{old} - (\nabla^2_{\boldsymbol{\beta}_n}\mathcal{L})^{-1}\nabla_{\boldsymbol{\beta}_n}\mathcal{L}. \tag{3.33}$$

Once again, both Hessians are negative definite, and hence in the territory of convex optimization.

3.2.3 Hyperparameters

One way to choose hyperparameters is to maximize the marginal likelihood w.r.t. the hyperparameters. Since the marginal likelihood is intractable in the vLGP model, we instead maximize (3.10) once again given the parameters and posterior. Interestingly, this objective function takes the same form as the one that is maximized in the GPFA's hyperparameters updates.

We write the squared-exponential covariance kernel as,

$$\mathbf{K}_l = \sigma_l^2 \exp(-\omega_l \mathbf{D}), \tag{3.34}$$

where $\mathbf{D}$ is the matrix of squared distances of each time pair. Hyperparameters σ^2 and ω corresponds to prior variance and inverse (squared) time scale. We optimize the log-transformed hyperparameter for those are positive. To the j-th transformed hyperparameter of the l-th latent dimension, θ_{lj}, the derivative is given as

$$\frac{\partial \mathcal{L}}{\partial \theta_{lj}} = \operatorname{tr}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{K}_l}\frac{\partial \mathbf{K}_l}{\partial \theta_{lj}}\right), \tag{3.35}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_l} = \frac{1}{2}\left(\mathbf{K}_l^{-1}\boldsymbol{\mu}_l\boldsymbol{\mu}_l^\top\mathbf{K}_l^{-1} + \mathbf{K}_l^{-1}\boldsymbol{\Sigma}_l\mathbf{K}_l^{-1} - \mathbf{K}_l^{-1}\right). \tag{3.36}$$

The optimal value can be found by common gradient algorithms for each latent dimension independently.

The above derivation of the hyperparameter optimization technique assumes a fixed posterior and parameters. Thus it requires complete prior covariance matrices and explicit posterior covariance matrices rather than low-rank decompositions. In order to avoid numerical singularity, we add a small quantity to the diagonal of prior covariance matrices. It would be extremely costly to use these complete covariance matrices for long, consecutive time series. Therefore, we randomly take many shorter temporal subsamples of the posterior for fast computation [118]. One hyperparameter iteration is performed every fixed number of iterations of posterior and parameter optimization.

3.3 Results

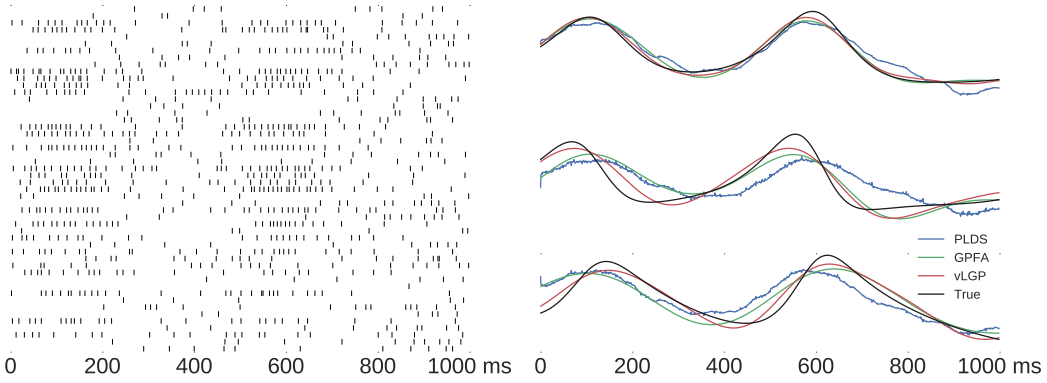
We apply our inference algorithm to two simulated systems and one real dataset. We compare our method (vLGP) against GPFA and PLDS.

3.3.1 Evaluation

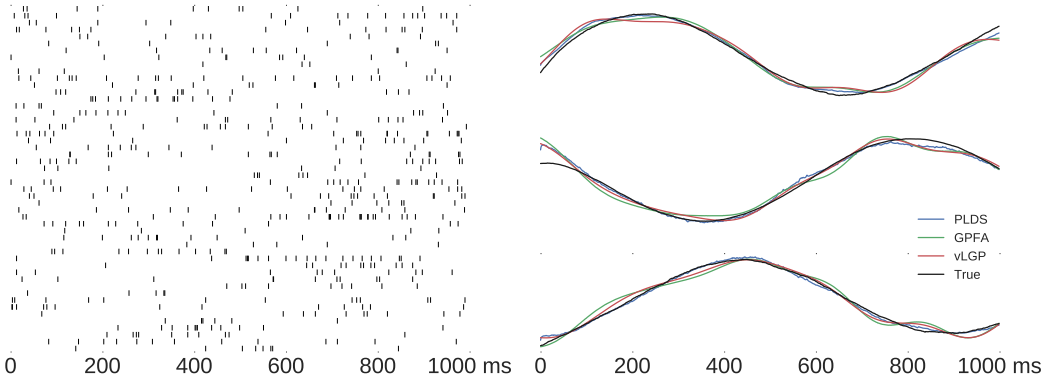
We use a leave-one-neuron-out prediction likelihood to compare models. For each dataset comprising of several trials, we choose one of the trials as test trial and the others as training trials. First, the weights and posterior are inferred from the training trials. Next, we leave one neuron out of the test trial and make inference on the posterior by the remaining neurons with the weights estimated from the training trials. Then the spike train of the left-out neuron are predicted by the model given the weights estimated from the training trials and the posterior inferred from the test trial. We do this procedure on each neuron of one trial and each trial of one sample. Finally we obtain the prediction of all spike trains in the sample.

For simulated datasets, we know the true latent process that generates observations. Since latent space is only identifiable up to affine transformation, we quantify using the angle between subspaces [15, 95]. However, due to possible mismatch in the point nonlinearity, the subspace can be distorted. To account for this mismatch, we use the mean Spearman's rank correlation that allows invertible monotonic mapping in each direction. The Spearman's rank correlation between the posterior and true latent trajectory gives a measure of the goodness of the posterior. If the correlation is large, the posterior recovers

more information about the underlying trajectory.



(a) Lorenz attractor with refractory period



(b) Linear dynamical system (LDS) with soft-rectified Poisson observation.

Figure 3.2: Spike trains from 50 simultaneously observed neurons, and corresponding 3-dimensional latent trajectory. **(Left)** Simulated spike trains from each corresponding system. See (4.13) and (3.38) for the exact generative model. **(Right)** True and inferred 3-dimensional latent processes. vLGP and GPFA infers smooth posterior, while noticeable high-frequency noise is present in the PLDS inference.

3.3.2 Simulation

We simulate two datasets: one with deterministic nonlinear dynamics, and one with linear dynamics and model-mismatched nonlinear observation. Each

dataset consists of 5 samples and each sample contains 10 trials from 50 neurons which last for 1 sec. We choose a bin size of 1 ms.

In the first dataset, the latent trajectories are sampled from the Lorenz dynamical system with the time step of 0.0015. This 3-dimensional system is defined by the following set of equations,

$$\begin{aligned} \dot{x} &= 10(y - x), \\ \dot{y} &= x(28 - z) - y, \\ \dot{z} &= xy - 2.667z. \end{aligned} \tag{3.37}$$

Spike trains are simulated by (3.1) with 10 ms suppressive history filter given the latent trajectories.

In the second dataset, Poisson spike trains are simulated from a 3-dimensional linear dynamical system (LDS) defined as

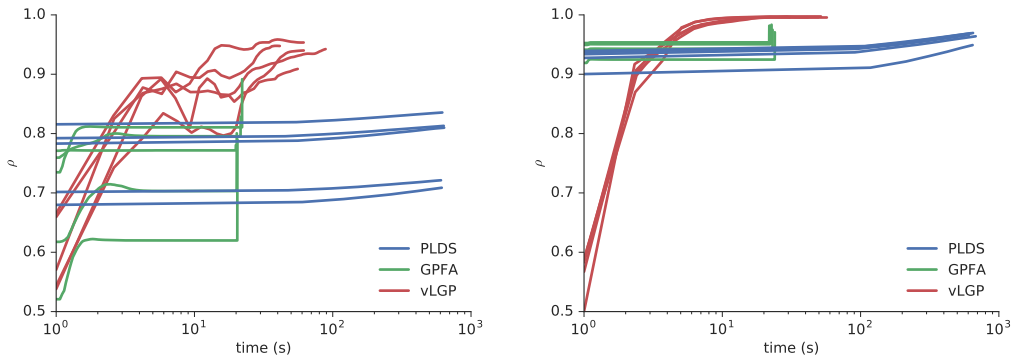
$$\begin{aligned} y_{t,n} \,|\, \mathbf{x}_t &\sim \text{Poisson}(\log(1 + \exp(\mathbf{c}_n^\top \mathbf{x}_t + \mathbf{d}_n)) \\ \mathbf{x}_0 &\sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{Q}_0) \\ \mathbf{x}_{t+1} \,|\, \mathbf{x}_t &\sim \mathcal{N}(\mathbf{A}\mathbf{x}_t + \mathbf{b}_t, \mathbf{Q}). \end{aligned} \tag{3.38}$$

Figure 3.2 shows one trial from each dataset and corresponding inferred posterior mean latent trajectory. The posterior means are rotated toward the true latent subspace. The PLDS inference (blue) looks the farthest away from the true Lorenz latent relatively but much closer to the LDS latent because the true latent meets its assumption. However, PLDS inference lacks of smoothness. The GPFA inference (green) is better than PLDS for Lorenz latent but

shows deviations from the true LDS latent. The smoothness is kept in the inference. The inference of our model (red) are very close to the true latent in both cases along the time while being smooth at the same time.

The Spearman's rank correlation between the posterior and true latent trajectory gives a measure of the goodness of the posterior. If the correlation is large, the posterior recovers more information about the underlying trajectory. For one sample, we compute the correlation between the posterior and true latent after concatenate all the trials along time. Since the models do not assume continuity between trials and impose constrains on the posterior mean, the concatenation is performed before computing the correlation. Note that the GPFA divides each trial into small time segments for estimating the loading matrix and bias. It breaks the continuity within each trial. Only the final iteration infers each trial as whole. Thus the correlations of the final iterations jumps up in the figures. However, those sudden changes do not affect for comparison so the correlations are still meaningful. The resulting correlation is an overall measure of the whole sample.

Figure 3.3 shows the Spearman's rank correlation between the posterior mean and true latent versus running time (log scale). The figures shows our model (vLGP) resulted in overall larger correlation than the PLDS and GPFA for almost all samples of both datasets after the algorithms end. PLDS uses nuclear norm penalized rate estimation as initialization [95]. The rank correlation from PLDS inference stayed near the initial value through the optimization. Both the GPFA and our model use factor analysis as initialization [118]. Note that the GPFA divides each trial into small time segments for estimating



(a) Lorenz

(b) LDS

Figure 3.3: Performance comparison on simulated datasets. **(a,b)** Convergence speed of each algorithm in terms of inferred rank correlation between the true generative latent time series and the inferred mean posterior. GPFA is the fastest, and PLDS converges very slowly. vLGP achieves the largest correlation, yet an order of magnitude faster than PLDS. The origin of time is shifted to 1 for convenience.

the loading matrix and bias. It breaks the continuity within each trial. Only the final iteration infers each trial as whole. Thus the correlations of the final iterations jumps up in the figures. It is obvious that our model makes much improvement to the result of factor analysis in terms of the rank correlation.

We use the log-likelihood on the leave-one-neuron-out to quantify predictive performance on the spike trains, as described in the evaluations section. The likelihood of test spike train is normalized with respect to that of a baseline model which assumes a homogeneous Poisson process to obtain, the prediction log-likelihood (PLL), given as,

$$\text{PLL} = \frac{\left[\sum_{t,n}(y_{y,n}\log(\lambda_{t,(-n)}) - \lambda_{t,(-n)})\right] - \left[\sum_{t,n}(y_{y,n}\log(\bar{y}) - \bar{y})\right]}{(\#\text{ of spikes})\log(2)}, \qquad (3.39)$$

where $\lambda_{t,(-n)}$ is the leave-neuron-out prediction to the firing rate of neuron n

at time t, and $\bar{y}$ is the population mean firing rate. Poisitive PLL implies the model predicts better than mean firing rate, and higher PLL implies better prediction. PLL has a unit of *bits per spike*, and is widely used to quantify spike train prediction [96].

In Table 3.1, we compare the three models for each dataset. Since GPFA assumes a Gaussian likelihood, it is incompatible to compare directly using a point process likelihood. We use linear rectifier to convert the GPFA predictions to non-negative rates, then compute PLL [3]. Denote the linear predictor by η omitting the neuron, time and model. Specifically, The predicted rate is given by,

$$\lambda = \begin{cases} \log(1+\exp(a\eta))/a & \text{GPFA (rectifier link)} \\ \exp(\eta + \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{V}\boldsymbol{\alpha}) & \text{PLDS and vLGP} \end{cases}, \tag{3.40}$$

where a control the "softness" of the rectifier. We choose the possible largest value that guarantees positive firing rates.

3.3.3 V1 population recording

We apply our method to a large scale recording to validate that vLGP picks up meaningful known signals, and investigate the population-wide trial-to-trial variability structure. We use the dataset [43] where 72 different equally spaced directional drifting gratings were presented to an anesthetized monkey for 50 trials each (array-5, 148 simultaneously recorded single units). We use 63 V1 neurons by only considering neurons with tuning curves that could be well

[3] We tried square link function for GPFA initially. However, it often produces deterimental predictions due to large negative predictions.

Table 3.1: Predictive log-likelihood (PLL)

Dataset	Sample	PLDS	GPFA (rectifier)	vLGP
Lorenz	1	0.41	-0.22	0.58
	2	0.50	-0.52	0.74
	3	0.50	-0.68	0.74
	4	0.51	-0.83	0.76
	5	0.44	-0.78	0.66
LDS	1	0.79	0.72	0.83
	2	0.94	0.87	0.98
	3	0.99	0.91	1.03
	4	0.97	0.92	1.01
	5	0.97	0.91	1.01
V1	N=63	0.81	0.97	0.99
	N=148	1.28	1.29	1.35

approximated ($R^2 \geq 0.75$) by bimodal circular Gaussian functions (the sum of two von Mises functions with different preferred orientations, amplitudes and bandwidths) according to [43]. We do not include the stimulus drive in the model, in hopes that the inferred latent processes would encode the stimulus. We used bin size of 1 ms.

We use 4-fold cross-validation to determine the number of latents. A 15–dimensional model is fitted to a subsample composed of the first trial of each direction at first. In each fold, we use its estimated parameter to infer the latent process from another subsample composed of the second trial of each direction. The inference is made by leaving a quarter of neurons out, and we predict the spike trains of the left-out neurons given the first k ($k = 1 \ldots 15$) orthogonalized latent process corresponding to k-dimension. This procedure led us to choose 5 as the dimension since the predictive log-likelihood reached its maximum.

We refit a 5-dimensional vLGP model using the subsample of the first trials.

To quantify how much the model explains, we report pseudo-R^2 defined as

$$R^2 = 1 - \frac{\mathrm{LL}_{\text{saturated}} - \mathrm{LL}_{\text{model}}}{\mathrm{LL}_{\text{saturated}} - \mathrm{LL}_{\text{null}}} \tag{3.41}$$

where $\mathrm{LL}_{\text{null}}$ refers to the log-likelihood of population mean firing rate model (single parameter). The pseudo-R^2 our model (vLGP with 5D latents) is 20.88%. This model explains with shared variability through the latents, and heterogeneity of baseline firing of individual neurons. For a baseline model with only per neuron noise component (and no shared latent), the pseudo-R^2 is 6.77%.

Table 3.1 shows the PLLs based on two subsets. The first one is 4 trials (0°, 90°, 180°, 270°) of the subset of 63 neurons with 5-dimensional latent process. The second one is 10 trials (5 trials of 0° and 5 trials of 90°) of all 148 neurons with 4-dimensional latent process.

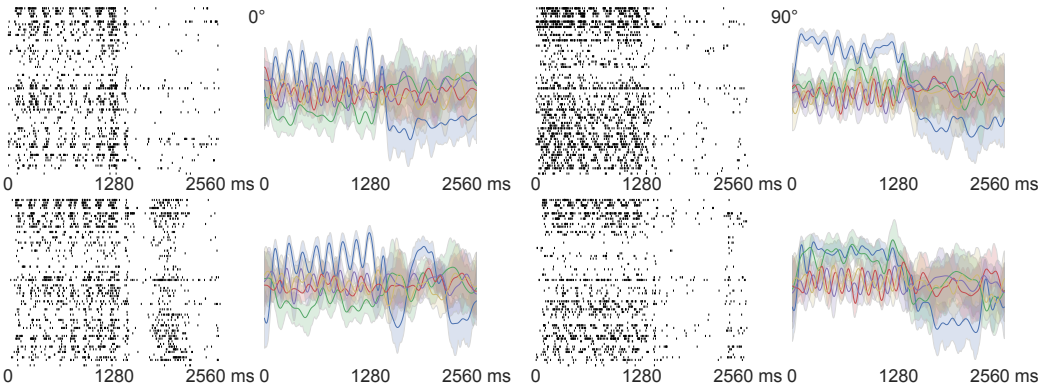


Figure 3.4: Single trial spike trains and inferred latent. The visual stimulus was only on for the first half of the trial. The left two columns are the spike trains and respective inferred latent of 2 trials of 0°. The right ones are 2 trials of 90°.

Although the parameters are estimated from a subsample, we can use them

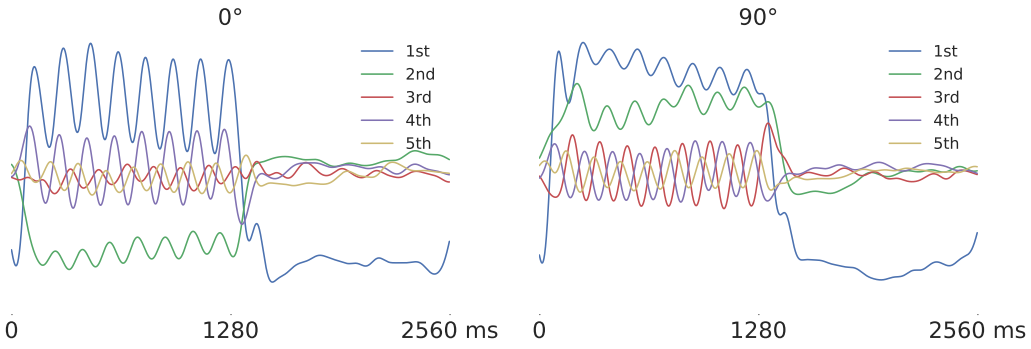


Figure 3.5: Inferred latent processes averaged for two stimulus directions (0° and 90°). Latents are rotated to maximize the power captured by each latent in decreasing order.

to infer the latent process of all trials of all 72 directions. Figure 3.4 shows inferred latent processes for two trials for two directions. We rotate the inferred latent process by the singular value decomposition (SVD) Variational posterior distribution over the latents are shown for each trial. During the second half of the trial when the stimulus was off, and the firing rate was lower, the uncertainty in the latent processes increases. There are visible trial-to-trial variability in the spike trains which are reflected in the variations of latents.

First we investigate how the "signal"—defined as visual stimuli—is captured by the latent processes. We average the inferred latent processes over 50 trials with identical spatiotemporal stimuli (Fig. 3.5). Since the stimuli are time-locked, corresponding average latent trajectory should reveal the time-locked population fluctuations driven by the visual input. We concatenate the average latent processes along the dimension of time. Then we orthogonalize it by SVD. The dimensions of orthogonalized one are ordered by the respective singular values. The latent process of a single trial is also rotated to the same

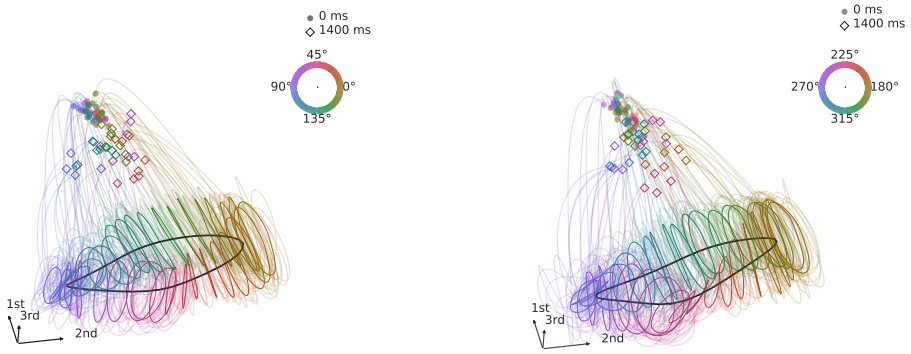


Figure 3.6: 3D projection of mean latent trajectories given each orientation. We plot the first three singular vectors of the inferred latent corresponding to the signal interval (0–1400 ms) colored by orientation. The colored circles are cycle averages that visualize the temporal phase of oscillation per direction, and form an approximate torus. The black circle visualizes the circular orientation that goes through the center of the torus. The left side shows 0–180° and the right side shows 180–360°.

subspace.

Furthermore, we visualized the trajectories in 3D (see supplementary online video[4]) that show how signal and noise are dynamically encoded in the state space. Figure 3.6 shows the projection of average latent process corresponding to each orientation to the first 3 principal components. The projection topologically preserves the orientation tuning in the V1 population. There are two continuous circular variables in the stimuli space to be encoded: orientation and temporal phase of oscillation. The simplest topological structure of neural encoding is a torus, and we observe a toroidal topology (highlighed as rings of cycle averages).

To see if our model captures the noise correlation structure through the latents as one would predict from recent studies of cortical population ac-

[4] `https://www.youtube.com/watch?v=CrY5AfNH1ik`

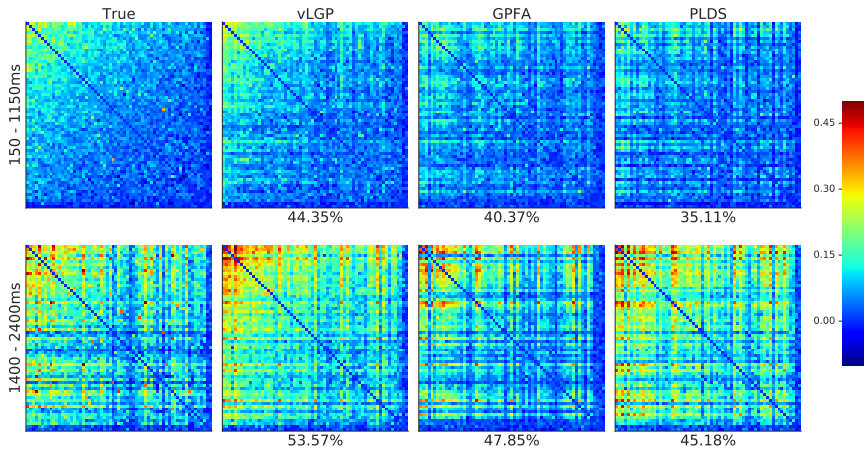


Figure 3.7: Noise-correlation analysis. The pairwise noise correlations between all neurons were calculated during the stimulus period (top, 150–1150 ms) and off-stimulus period (bottom, 1400–2400 ms). The time bin size is 50 ms. Neurons are sorted by the total noise-correlation defined as the row sum of stimulus-driven noise correlation matrix (top-left). The model-explained power percentages are shown on the bottom of each matrix.

tivity [42, 30, 74], we calculated pairwise correlations between all neurons. We simulated spike trains by model-predicted firing rates of all trials with 0° and 90° stimulus. To remove the signal, we subtracted the mean over 50 trials for each direction of stimulus. Figure 3.7 shows the correlation matrices during the stimulus period (150–1150 ms) and off-stimulus period (1400–2400 ms). The neurons are sorted by the total correlations during the stimulus period. The power of model-explained noise correlation is defined as $(1 - \|\mathbf{C}_{\text{model}} - \mathbf{C}_{\text{true}}\|_{\text{F}})/\|\mathbf{C}_{\text{true}}\|_{\text{F}}$ where $\mathbf{C}$ is the zero-diagonal correlation matrix w.r.t. its subscript and $\|\cdot\|_{\text{F}}$ is the Frobenius norm. The proposed model explains more noise correlation in contrast to GPFA and PLDS for both periods.

These results show that vLGP is capable of capturing both the signal — repeated over multiple trials — and noise — population fluctuation not time locked to other task variables — present in the cortical spike trains.

3.4 Summary

We propose vLGP, a method that recovers low-dimensional latent trajectories from high-dimensional time series in this chapter. Inferring latent trajectories from single trials provides a flexible framework for studying internal processes that are not time-locked. It can perform dimensionality reduction on a single trial basis and allows decomposition of neural signals into a small number of temporal signals and their relative contribution to the population signal.

We compare our method to two widely used latent state space modeling

tools in neuroscience, GPFA [118] and PLDS [80]. Our method allows point process observations in contrast to GPFA and is faster than PLDS. Yet it shows superiority in capturing the spatio-temporal structures in the neural data to both PLDS and GPFA.

We also apply our method to real electrophysiological recordings, V1 population recording driven by fixed stimulus. The inferred latents contain meaningful information about the external stimuli, encoding both orientation and temporal modulation.

The proposed method has potential application in many areas, and it will be particularly useful in discovering how specific neural computations are implemented as neural dynamics.

Chapter 4

Interpretable Nonlinear Dynamic Modeling of Neural Trajectories

A central challenge in neuroscience is understanding how neural system implements computation through its dynamics. In this chapter, we propose a nonlinear time series model aimed at characterizing interpretable dynamics from neural trajectories. Our model incorporates prior assumption about globally contractional dynamics to avoid overly enthusiastic extrapolation outside of the support of observed trajectories. We will show that our model can recover qualitative features of the phase portrait such as attractors, slow points, and bifurcation, while also producing reliable long-term future predictions in a variety of dynamical models and in real neural data.

4.1 Model

Consider a general d-dimensional continuous nonlinear dynamical system driven by external input,

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}) \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ represent the dynamic trajectory, and $F : \mathbb{R}^d \times \mathbb{R}^{d_i} \to \mathbb{R}^d$ fully defines the dynamics in the presence of input drive $\mathbf{u} \in \mathbb{R}^{d_i}$. We aim to learn the essential part of the dynamics F from a collection of trajectories sampled

at frequency $1/\Delta$.

Our work builds on extensive literature in nonlinear time series modeling. Assuming a separable, linear input interaction, $F(\mathbf{x}, \mathbf{u}) = F_x(\mathbf{x}) + F_u(\mathbf{x})\mathbf{u}$, a natural nonlinear extension of autoregressive model is to use a locally linear expansion of (4.1) [90, 113]:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{A}(\mathbf{x}_t)\mathbf{x}_t + \mathbf{b}(\mathbf{x}_t) + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \epsilon_t \tag{4.2}$$

where $\mathbf{b}(\mathbf{x}) = F_x(\mathbf{x})\Delta$, $\mathbf{A}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is the Jacobian matrix of F_x at $\mathbf{x}$ scaled by time step Δ, $\mathbf{B}(\mathbf{x}) : \mathbb{R}^{d_i} \to \mathbb{R}^d$ is the linearization of F_u around $\mathbf{x}$, and ϵ_t denotes model mismatch noise of order $\mathcal{O}(\Delta^2)$. For example, $\{\mathbf{A}, \mathbf{B}\}$ are parametrized with an RBF network in multivariate RBF-ARX model of [38, 90], and $\{\mathbf{A}, \mathbf{b}, \mathbf{B}\}$ are parametrized with sigmoid neural networks in [113]. Note that $\mathbf{A}(\cdot)$ is not guaranteed to be the Jacobian of the dynamical system (4.1) since $\mathbf{A}$ and $\mathbf{b}$ also change with $\mathbf{x}$. In fact, the functional form for $\mathbf{A}(\cdot)$ is not unique, and a powerful function approximator for $\mathbf{b}(\cdot)$ makes $\mathbf{A}(\cdot)$ redundant and over parameterizes the dynamics.

Note that (4.2) is a subclass of a general nonlinear model:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \epsilon_t, \tag{4.3}$$

where $\mathbf{f}, \mathbf{B}$ are the discrete time solution of F_x, F_u. This form is widely used, and called nonlinear autoregressive with eXogenous inputs (NARX) model where $\mathbf{f}$ assumes various function forms (e.g. neural network, radial basis function network [20], or Volterra series [31]).

We propose to use a specific parameterization,

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{g}(\mathbf{x}_t) + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \epsilon_t \\ \mathbf{g}(\mathbf{x}_t) &= \mathbf{W}_g \boldsymbol{\phi}(\mathbf{x}_t) - e^{-\tau^2}\mathbf{x}_t \\ \mathrm{vec}(\mathbf{B}(\mathbf{x}_t)) &= \mathbf{W}_B \boldsymbol{\phi}(\mathbf{x}_t) \end{aligned} \tag{4.4}$$

where $\boldsymbol{\phi}(\cdot)$ is a vector of r continuous basis functions,

$$\boldsymbol{\phi}(\cdot) = (\phi_1(\cdot), \ldots, \phi_r(\cdot))^\top. \tag{4.5}$$

Note the inclusion of a global leak towards the origin whose rate is controlled by τ^2. The further away from the origin (and as $\tau \to 0$), the larger the effect of the global contraction. This encodes our prior knowledge that the neural dynamics are limited to a finite volume of phase space, and prevents solutions with nonsensical runaway trajectories.

The function $\mathbf{g}(\mathbf{x})$ directly represents the velocity field of an underlying smooth dynamics (4.1), unlike $\mathbf{f}(\mathbf{x})$ in (4.3) which can have convoluted jumps. We can even run the dynamics backwards in time, since the time evolution for small Δ is reversible (by taking $\mathbf{g}(\mathbf{x}_t) \approx \mathbf{g}(\mathbf{x}_{t+1})$), which is not possible for (4.3), since $\mathbf{f}(\mathbf{x})$ is not necessarily an invertible function.

Fixed points $\mathbf{x}^*$ satisfy $\mathbf{g}(\mathbf{x}^*)+\mathbf{B}(\mathbf{x}^*)\mathbf{u} = 0$ for a constant input $\mathbf{u}$. Far away from the fixed points, dynamics is locally just a flow (rectification theorem) and largely uninteresting. The Jacobian $J = \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}$ provides linearization of the dynamics around the fixed points (via Hartman-Grobman theorem), and the corresponding fixed point is stable if all eigenvalues of J are negative.

We can further identify local minima of $\|\mathbf{g}\|$ which include the ghost points in addition to the fixed points. The flow around the ghost points can be extremely slow [107], and can exhibit signatures of computation through metastable dynamics [100]. Continuous attractors (such as limit cycles) are also important features of neural dynamics which exhibit spontaneous oscillatory modes. We can easily identify attractors by simulating the model.

4.2 Estimation

We define the mean squared error as the loss function

$$L(\mathbf{W}_g, \mathbf{W}_B, \mathbf{c}_{1\ldots r}, \sigma_{1\ldots r}) = \frac{1}{T}\sum_{t=0}^{T-1} \|\mathbf{g}(\mathbf{x}_t) + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \mathbf{x}_t - \mathbf{x}_{t+1}\|_2^2, \tag{4.6}$$

where we use normalized squared exponential radial basis functions

$$\phi_i(\mathbf{z}) = \frac{\exp\left(-\frac{\|\mathbf{z}-\mathbf{c}_i\|_2^2}{2\sigma_i^2}\right)}{\epsilon + \sum_{i=1}^{r}\exp\left(-\frac{\|\mathbf{z}-\mathbf{c}_i\|_2^2}{2\sigma_i^2}\right)}, \tag{4.7}$$

with centers $\mathbf{c}_i$ and corresponding kernel width σ_i. The small constant $\epsilon = 10^{-7}$ is to avoid numerical 0 in the denominator.

We estimate the parameters $\{\mathbf{W}_g, \mathbf{W}_B, \tau, \mathbf{c}, \boldsymbol{\sigma}\}$ by minimizing the loss function through gradient descent (Adam [65]) implemented within TensorFlow [82]. We initialize the matrices $\mathbf{W}_g$ and $\mathbf{W}_B$ by truncated standard normal distribution, the centers $\{\mathbf{c}_i\}$ by the centroids of the K-means clustering on the training set, and the kernel width σ by the average euclidean distance between the centers.

4.3 Inferring Theoretical Models of Neural Computation

We apply the proposed method to a variety of low-dimensional neural models in theoretical neuroscience. Each theoretical model is chosen to represent a different mode of computation.

4.3.1 Fixed point attractor and bifurcation for binary decision-making

Perceptual decision-making and working memory tasks are widely used behavioral tasks where the task typically involve a low-dimensional decision variable, and subjects are close to optimal in their performance. To understand how the brain implements such neural computation, many competing theories have been proposed [6, 79, 81, 39, 84, 116, 40]. We implemented the two dimensional dynamical system from [116] where the final decision is represented by two stable fixed points corresponding to each choice. The stimulus strength (coherence) nonlinearly interacts with the dynamics (see appendix for details), and biases the choice by increasing the basin of attraction (Fig. 4.1). We encode the stimulus strength as a single variable held constant throughout each trajectory as in [116].

The model with 10 basis functions learned the dynamics from 90 training trajectories (30 per coherence $c = 0, 0.5, -0.5$). We visualize the log-speed as colored contours, and the direction component of the velocity field as arrows in Fig. 4.1. The fixed/slow points are shown as red dots, which ideally should be at the crossing of the model nullclines given by solid lines. For each coherence,

two novel starting points were simulated from the true model and the estimated model in Fig. 4.1. Although the model was trained with only low or moderate coherence levels where there are 2 stable and 1 unstable fixed points, it predicts bifurcation at higher coherence and it identifies the ghost point (lower right panel).

We compare the model (4.4) to the following "locally linear" (LL) model,

$$\begin{aligned} \mathbf{x}_{t+1} =& \mathbf{A}(\mathbf{x}_t)\mathbf{x}_t + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \mathbf{x}_t \\ \text{vec}(\mathbf{A}(\mathbf{x}_t)) =& \mathbf{W}_A \boldsymbol{\phi}(\mathbf{x}_t) \\ \text{vec}(\mathbf{B}(\mathbf{x}_t)) =& \mathbf{W}_B \boldsymbol{\phi}(\mathbf{x}_t) \end{aligned} \tag{4.8}$$

in terms of training and prediction errors in Table 4.1. Note that there is no contractional term. We train both models on the same trajectories described above. Then we simulate 30 trajectories from the true system and trained models for coherence $c = 1$ with the same random initial states within the unit square and calculate the mean squared error between the true trajectories and model-simulated ones as prediction error. The other parameters are set to the same value as training. The LL model has poor prediction on the test

Table 4.1: Model errors

Model	Training error	Prediction error: mean (std)
(4.4)	4.06E-08	0.002 (0.008)
(4.8)	2.04E-08	0.244 (0.816)

set. This is due to unbounded flow out of the phase space where the training data lies (see Fig. A.1 in the appendix).

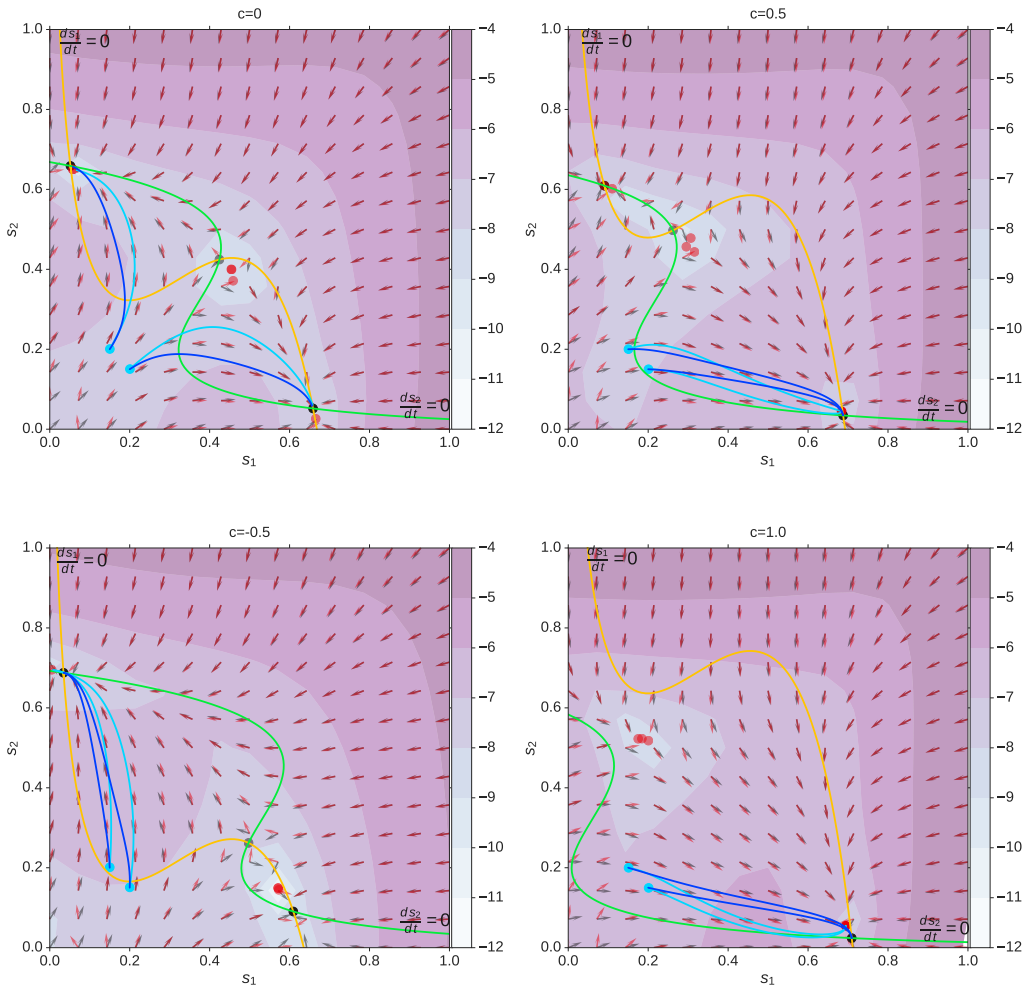


Figure 4.1: Wong and Wang's 2D dynamics model for perceptual decision-making [116]. We train the model with 90 trajectories (uniformly random initial points within the unit square, 0.5 s duration, 1 ms time step) with different input coherence levels $c = \{0, 0.5, -0.5\}$ (30 trajectories per coherence). The yellow and green lines are the true nullclines. The black arrows represent the true velocity fields (direction only) and the red arrows are model-predicted ones. The black and gray circles are the true stable and unstable fixed points, while the red ones are local minima of model-prediction (includes fixed points and slow points). The background contours are model-predicted $\log\|\frac{d\mathbf{s}}{dt}\|_2$. We simulated two 1 s trajectories each for true and learned model dynamics. The trajectories start from the cyan circles. The blue lines are from the true model and the cyan ones are simulated from trained models. Note that we do not train our model on trajectories from the bottom right condition ($c = 1$).

4.3.2 Nonlinear oscillator model

One of the most successful application of dynamical systems in neuroscience is in the biophysical model of a single neuron. We study the FitzHugh-Nagumo (FHN) model which is a 2-dimensional reduction of the Hodgkin-Huxley model [58]:

$$\dot{v} = v - \frac{v^3}{3} - w + I, \tag{4.9}$$

$$\dot{w} = 0.08(v + 0.7 - 0.8w), \tag{4.10}$$

where v is the membrane potential, w is a recovery variable and I is the magnitude of stimulus current. The FHN has been used to model the up-down states observed in the neural time series of anesthetized auditory cortex [26].

We train the model with 50 basis functions on 100 simulated trajectories with uniformly random initial states within the unit square $[0, 1] \times [0, 1]$ and driven by injected current generated from a 0.3 mean and 0.2 standard deviation white Gaussian noise. The duration is 200 and the time step is 0.1.

In electrophysiological experiments, we only have access to $v(t)$, and do not observe the slow recovery variable w. Delay embedding allows reconstruction of the phase space under mild conditions [61]. We build a 2D model by embedding $v(t)$ as $(v(t), v(t - 10))$, and fit the dynamical model (Fig. 4.2b). The phase space is distorted, but the overall prediction of the model is good given a fixed current (Fig. 4.2b). Furthermore, the temporal simulation of $v(t)$ for white noise injection shows reliable long-term prediction (Fig. 4.2c). We also test the model in a regime far from the training trajectories, and the dynamics

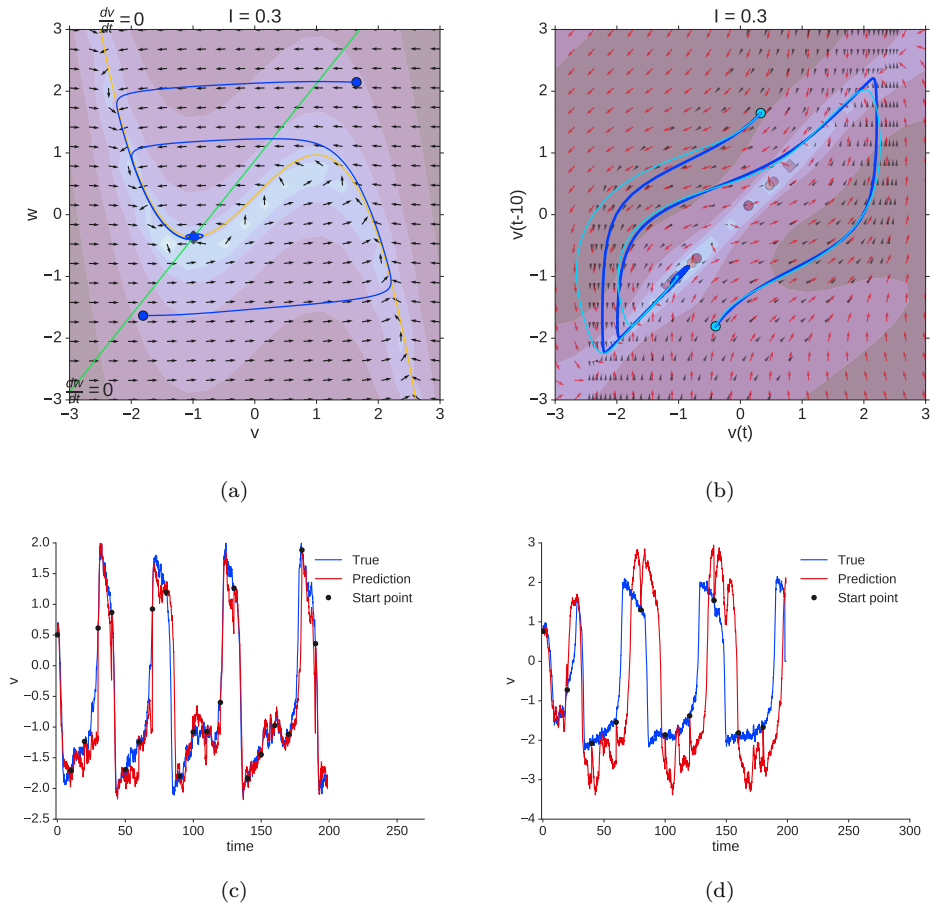


Figure 4.2: FitzHugh-Nagumo model. **(a)** Direction (black arrow) and log-speed (contour) of true velocity field. Two blue trajectories starting at the blue circles are simulated from the true system. The yellow and green lines are null-clines of v and w. The diamond is a spiral point. **(b)** 2-dimensional embedding of v model-predicted velocity field (red arrow and background contour). The black arrows are true velocity field. There are a few model-predicted slow points in light red. The blue lines are the same trajectories as the ones in (a). The cyan ones are simulated from trained model withe the same initial states of the blue ones. **(c)** 100-step prediction every 100 steps using a test trajectory generated with the same setting as training. **(d)** 200-step prediction every 200 steps using a test trajectory driven by sinusoid input with 0.5 standard deviation white Gaussian noise.

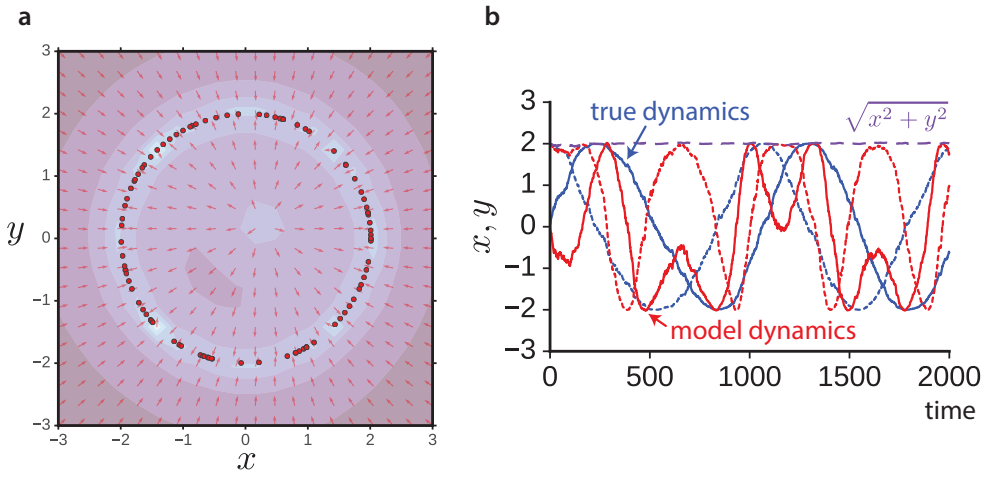


Figure 4.3: **(a)** Model-predicted (red arrows) velocity field for both direction and log-speed, model-predicted fixed points (red circles). **(b)** One trajectory from the true model, and one trajectory from the fit model. The trajectory remains on the circle for both (purple dashed line). Both are driven by the same input, and starts at same initial state, however, they quickly diverge.

does not diverge away from reasonable region of the phase space (Fig. 4.2d).

4.3.3 Ring attractor dynamics for head direction network

Continuous attractors such as line and ring attractors are often used as models for neural representation of continuous variables [81, 107]. For example, the head direction neurons are tuned for the angle of the animal's head direction, and a bump attractor network with ring topology is proposed as the dynamics underlying the persistently active set of neurons [94]. Here we use the following 2 variable reduction of the ring attractor system:

$$\tau_r \dot{r} = r_0 - r, \tag{4.11}$$

$$\tau_\theta \dot{\theta} = I(t), \tag{4.12}$$

where θ represents the head direction driven by input $I(t)$, and r is the radial component representing the overall activity in the bump. The computational role of this ring attractor is to be insensitive to the noise in the r direction, while integrating the differential input in the θ direction. In the absence of input, the head direction θ does a random walk around the ring attractor. The ring attractor consists of a continuum of stable fixed points with a center manifold.

We train the model with 50 basis functions on 150 trajectories (Fig. 4.4). The duration is 5 and the time step is 0.01. The parameters are set as $r_0 = 2$, $\tau_r = 0.1$ and $\tau_\theta = 1$. The initial states are uniformly random within $(r, \theta) \in [0, 4] \times [0, 2\pi]$. The inputs are constant angles evenly spaced in $[-\pi, \pi]$ with standard Gaussian noises added.

From the trained model, we can identify a number of fixed points arranged around the ring attractor (Fig. 4.3a). The true ring dynamics model has one negative eigenvalue, and one zero-eigenvalue in the Jacobian, but the analysis of the trained model are all saddles (one positive, and one negative eigenvalue). The fixed points allows the state to remain close to the ring attractor, however, this results in an imperfect integration of input. This is demonstrated in Fig. 4.3b, as the true model and the trained model are both attracted to the ring, however show qualitatively different behavior for the same input.

4.3.4 Chaotic dynamics

Chaotic dynamics (or near chaos) has been postulated to support asynchronous states in the cortex [45], and neural computation over time by gen-

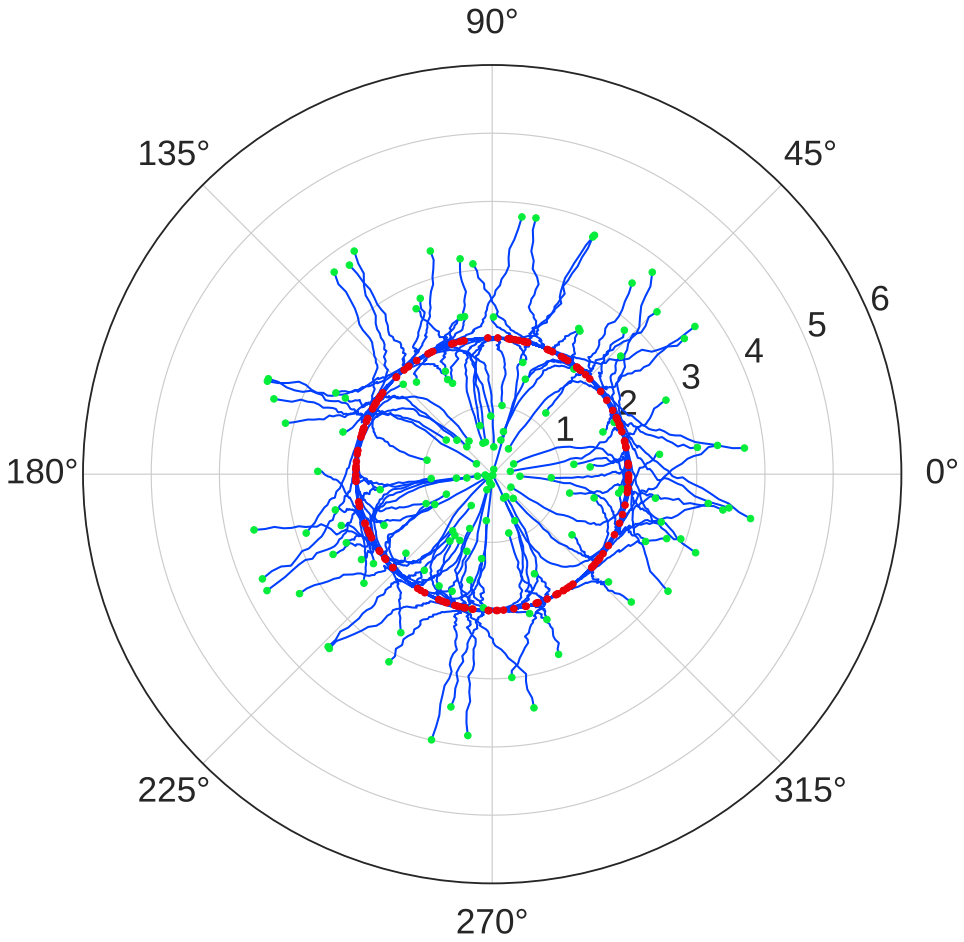


Figure 4.4: 150 training trajectories for the ring attractor. Green circles are initial states and red circles are final states.

erating rich temporal patterns [78, 68]. We consider the 3D Lorenz attractor as an example chaotic system. We simulate 20 trajectories from,

$$\begin{aligned} \dot{x} &= 10(y - x), \\ \dot{y} &= x(28 - z) - y, \\ \dot{z} &= xy - \frac{8}{3}z. \end{aligned} \tag{4.13}$$

The initial state of each trajectory is standard normal. The duration is 200 and the time step is 0.04. The first 300 transient states of each trajectory are discarded. We use the 19 trajectories for training and the last one for test. We train a model with 10 basis functions. Figure 4.5a shows the direction of prediction. The vectors represented by the arrows start from current states and point at the next future state. The predicted vectors (red) overlap the true vectors (blue) implying the one-step-ahead predictions are close to the true values in both speed and direction. Panel (b) gives an overview that the prediction resembles the true trajectory. Panel (c) shows that the prediction is close to the true value up to 200 steps.

4.4 Learning V1 neural dynamics

We use a set of trajectories obtained from a Gaussian process latent dynamical model. The latent trajectory model infers a 5-dimensional trajectory that describes a large scale V1 population recording. The recording was from an anesthetized monkey that 72 different equally spaced directional drifting gratings were presented to for 50 trials each (63 well tuned neurons out of 148

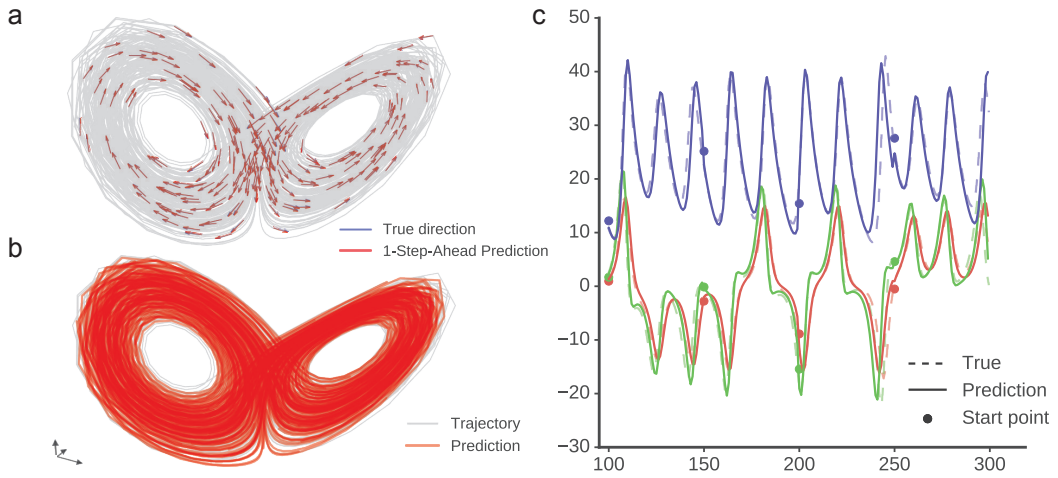
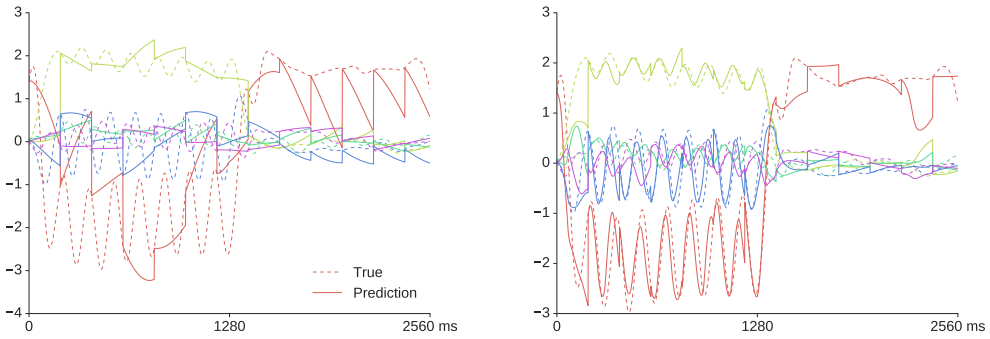


Figure 4.5: **(a)** Vector plot of 1-step-ahead prediction on one Lorenz trajectory (test). **(b)** 50-step prediction every 50 steps on one Lorenz trajectory (test). **(c)** A 200-step window of **(b)** (100-300). The dashed lines are the true trajectory, the solid lines are the prediction and the circles are the start points of prediction.

simultaneously recorded single units from [43]). Each trial lasts for 2.56 s and the stimulus was presented during the first half.

We train our model with 50 basis functions on the mean trajectories for 71 directions, and use 1 direction for testing. The input were 3 dimension: two boxcar indicating the stimulus direction $(\sin\theta, \cos\theta)$, and one corresponding to a low-pass filtered stimulus onset indicator. Figure 4.6 shows the prediction of the best linear dynamical system (LDS) for the 71 directions, and the nonlinear prediction from our model. Although the LDS is widely used for smoothing the latent trajectories, it clearly is not a good predictor for the nonlinear trajectory of V1 (Fig. 4.6a).



(a) LDS prediction

(b) Proposed model prediction

Figure 4.6: V1 latent dynamics prediction. Models trained on 71 average trajectories for each directional motion are tested on the 1 unseen direction. We divide the average trajectory at 0° into 200 ms segments and predict each whole segment from the starting point of the segment. Note the poor predictive performance of linear dynamical system (LDS) model.

4.5 Summary

In summary, we present a novel complementary approach to studying the neural dynamics of neural computation and show that it successfully learns low-dimensional dynamics of well-known dynamical models of neural computation with various features such as fixed points, bifurcation, ring attractor and chaotic attractor. In addition, it can model nonlinear latent trajectories extracted from high-dimensional neural time series. Applications of the proposed method are not limited to neuroscience, but should be useful for studying other slow low-dimensional nonlinear dynamical systems from observations [28].

Chapter 5

Discussion

In this thesis, we focus on modeling count data. Two specific problems have been discussed. The first involves independent counting observations in biostatistics, and the second pertains to point process observations in neuroscience. To deal with those problems, we extend the Poisson log-linear model, and propose a latent variable model together with a nonlinear dynamic model in correspondence.

5.1 Log-Linear Model Based Tree

Poisson regressions play an important role in the classical framework of count data analyses. The log-linear model as a subclass of GLM is a standard device for interpreting the association between response and explanatory variables. It has been well-studied and shows many advantages such as tractability and interpretability. The assumptions, however, are not always satisfied in practice. The relationship between dependent and independent variables is often complex.

There are many sophisticated methods that deal with nonlinearity such as GAM [49]. Those powerful nonparametric techniques somehow lack of inter-

pretability. Here we consider another way to relax the linearity by stratification. In other words, we break a single log-linear model into pieces. Such stratification is repeated recursively, which leads to the tree structure. The tree partitions the data by certain conditions. It results in a graphical representation of stratification. In a tree, each node represents a subset of data and the attached branches are substratification within those subset.

Traditional decision trees use simple statistics, such as mean, median and mode, as the representation of the very subset. This advantages computational efficiency and power especially in prediction and classification problems, but it loses information within each node. Ensemble methods are proposed to improve the accuracy such as random forest and boosting. Those methods trade interpretability for prediction power.

We replace the simple statistic (mean, median or mode) of the nodes with log-linear models. Within each node, there is a log-linear model for the stratum. Then the tree can be considered as a piecewise log-linear model. In consistent with maximum likelihood estimation for log-linear models, we partitions the variable in the way that maximizes the likelihood of two branches. Without controlling the size, the tree grows to saturation of which the terminal (leaf) nodes are single individual observations. Apparently overfitting happens if a simple log-linear model is sufficient for the data. So it is imperative to remove unnecessary splits to control the size. This can be done forward (look-ahead) and backward (prune). We only consider forward methods in this study. We assess splits by testing for sufficiency of a simple log-linear model. One node is splitted only if the null hypothesis is rejected. Then controlling the

tree size becomes controlling the Type I error. Different tests were compared on simulated data from several typical models.

The responses are usually treated as Poisson distributed in log-linear models. The Poisson distribution features the same mean and variance. Overdispersion however occurs sometimes. It leads to incorrect testing result which has higher probability of rejection due to underestimation of variance. We use negative binomial instead of Poisson for overdispersed data.

We apply the method on Missouri lung cancer data with both Poisson and negative binomial distributions. The resulted trees are the same when the minimal size of node is 20 and different when the size is 5. They show that the effect of sex is significant in all strata but the effect of age is not in certain nodes. This is not revealed in the root node which is the simple log-linear model. Besides, in the negative binomial tree, the overdispersion become weaker as the nodes go deeper, which suggests that the overdispersion may come from mixture.

The likelihood ratio test is competitive to the last two parameter instability tests in most simulation cases. The shortcoming is that it relies on prespecified breakpoint. We use median in purpose of balanced size, but it causes failure in some cases (Model 8). A desirable breakpoint is the true one used in split. The distribution of the test statistic is no longer χ^2_1 under null hypothesis since the breakpoint is estimated from data. Thus the test will be biased and high likely to reject the null. Therefore a future work would be to find the null distribution of the likelihood ratio with the breakpoint which maximize the likelihood.

5.2 Variational Latent Gaussian Process for Recovering Single-Trial Trajectories from Population Spike Trains

To neuroscience problem, we propose vLGP, a method that recovers low-dimensional latent trajectories from high-dimensional time series. Methods inferring latent state-space are different from trial averaging ones that only recover the average trajectories time-locked to an external observation [24, 13]. Inferring latent trajectories from single trials provides a flexible framework for studying internal processes that are not time-locked. Higher-order tasks such as decision-making, attention, and memory recall are well suited to be analyzed via latent trajectories. It performs dimensionality reduction on a single trial basis and allows decomposition of neural signals into a small number of temporal signals and their relative contribution to the population signal.

In our study, we compare our method to two widely used latent state-space modeling tools in neuroscience, GPFA [118] and PLDS [80]. Unlike GPFA, vLGP allows generalized linear model observations which is suitable for a wide class of point process observations. Moreover, vLGP is faster than PLDS which assumes a Poisson process observation given the latent trajectory, yet it shows performance in capturing the spatio-temporal structures in the neural data superior to both PLDS and GPFA.

To test its validity in real electrophysiological recordings, we use V1 population recording driven by fixed stimulus as a litmus test for vLGP. Our inferred latent trajectories contain meaningful information about the external stimuli, encoding both orientation and temporal modulation.

We have only applied smoothness encoded in the GP prior in this study, but a plethora of GP kernels are available [101, 104]. For example, to capture prior assumptions about periodicity in some of the latent processes, we can use spectral kernels [111]. This can be particularly useful for capturing internal neural oscillations [35]. In addition, it is straightforward to incorporate additional covariates such as external stimuli [92] or local field potential [63] to vLGP. The proposed method has potential application in many areas, and it will be particularly useful in discovering how specific neural computations are implemented as neural dynamics.

5.3 Interpretable Nonlinear Dynamic Modeling of Neural Trajectories

Following the latent dynamics recovery, fitting an expressive model that produces robust prediction is in connection to the dynamical theories of neural computation with neural time series data. The interpretability is needed such that signatures of neural computation from the theories can be identified by its qualitative features.

Our method successfully learns low-dimensional dynamics in contrast to fitting a high-dimensional recurrent neural network models in previous approaches [81, 107, 68]. Our proposed model works well for well-known dynamical models of neural computation with various features: chaotic attractor, fixed point dynamics, bifurcation, line/ring attractor, and a nonlinear oscillator. In addition, our method can model nonlinear latent trajectories extracted from high-dimensional neural time series.

The critical assumption is that the dynamics consists of a continuous and slow flow. This allows us to parameterize the velocity field directly, which reduces the complexity of the nonlinear function approximation, and makes it easy to identify the fixed/slow points. An additional structural assumption is the existence of a global contractional dynamics. This regularizes and encourages the dynamics to occupy a finite phase volume around the origin.

Learning continuous attractor dynamics requires more basis functions, and more careful regularization. Although we learn the continuous attractor, the modeled ring dynamics cannot integrate the input correctly. Our plan is to design basis functions and explore a deeper architecture that can represent arbitrary limit cycles and meta-stable dynamics parsimoniously without losing flexibility.

Visualization techniques for arbitrary trajectories from a nonlinear system such as recurrence plots were often difficult to understand. We decompose the velocity field into speed and direction, and overlay fixed/slow points found by numerically minimizing the speed. This is however difficult for higher-dimensional dynamics. So dimensionality reduction and visualization that preserves essential dynamic features are left for future directions.

Our next goal is to incorporate this model with a multivariate point process observation model in Chapter 3 and control internal neural states [113].

Bibliography

[1] A. Agresti. *Categorical Data Analysis*. 2nd Edition. John Wiley & Sons, Apr. 2003.

[2] H. Ahn et al. "Classification by Ensembles from Random Partitions of High-Dimensional Data". In: *Computational Statistics & Data Analysis* 51 (2007), pp. 6166–6179.

[3] E. W. Archer et al. "Low-Dimensional Models of Neural Population Activity in Sensory Cortical Circuits". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 343–351.

[4] E. Archer et al. "Black Box Variational Inference for State Space Models". In: *ArXiv* (Nov. 2015).

[5] F. R. Bach and M. I. Jordan. "Kernel Independent Component Analysis". In: *Journal of Machine Learning Research* 3.1 (Jan. 2002), pp. 1–48.

[6] O. Barak et al. "From Fixed Points to Chaos: Three Models of Delayed Discrimination". In: *Progress in neurobiology* 103 (Apr. 2013), pp. 214–222.

[7] C. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[8] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. *Variational Inference: A Review for Statisticians*. Mar. 2016. arXiv: `1601.00670`.

[9] L. Breiman. "Bagging Predictors". In: *Machine learning* 140 (1996), pp. 123–140.

[10] L. Breiman. "Heuristics of instability and stabilization in model selection". In: *The annals of statistics* 24 (1996), pp. 2350–2383.

[11] L. Breiman. *Out-of-Bag Estimation*. Tech. rep. 1996.

[12] L. Breiman. "Random Forests". In: *Machine learning* 45 (2001), pp. 5–32.

[13] W. Brendel, R. Romo, and C. K. Machens. "Demixed Principal Component Analysis". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. 2011, pp. 2654–2662.

[14] N. Breslow. "Extra-Poisson Variation in Log-Linear Models". In: *Applied statistics* 33 (1984), pp. 38–44.

[15] L. Buesing, J. Macke, and M. Sahani. "Spectral Learning of Linear Dynamics from Generalised-Linear Observations with Application to Neural Population Data". In: *Advances in Neural Information Processing Systems 25*. 2012, pp. 1691–1699.

[16] A. C. Cameron and P. K. Trivedi. "Econometric Models Based on Count Data. Comparisons and Applications of Some Estimators and Tests". In: *Journal of Applied Econometrics* 1.1 (1986), pp. 29–53.

[17] K.-Y. Chan and W.-Y. Loh. "LOTUS: An Algorithm for Building Accurate and Comprehensible Logistic Regression Trees". In: *Journal of Computational and Graphical Statistics* 13.4 (2004), pp. 826–852.

[18] P. Chaudhuri et al. "Generalized Regression Trees". In: *Statistica Sinica* 5.1986 (1995), pp. 641–666.

[19] J. Chen and H. Ahn. "Fitting Mixed Poisson Regression Models Using Quasi-Ikelihood Methods". In: *Biometrical journal* 38 (1996), pp. 81–96.

[20] S. Chen et al. "Practical Identification of NARMAX Models Using Radial Basis Functions". In: *International Journal of Control* 52.6 (Dec. 1990), pp. 1327–1350.

[21] Y. Choi, H. Ahn, and J. Chen. "Regression Trees for Analysis of Count Data with Extra Poisson Variation". In: *Computational statistics & data analysis* (2005).

[22] A. K. Churchland et al. "Variance as a Signature of Neural Computations During Decision Making". In: *Neuron* 69.4 (Feb. 2011), pp. 818–831.

[23] M. M. Churchland et al. "Cortical Preparatory Activity: Representation of Movement or First Cog in a Dynamical Machine?" In: *Neuron* 68.3 (Nov. 2010), pp. 387–400.

[24] M. M. Churchland et al. "Neural Population Dynamics During Reaching". In: *Nature* 487.7405 (July 2012), pp. 51–56.

[25] J. P. Cunningham and B. M. Yu. "Dimensionality Reduction for Large-Scale Neural Recordings". In: *Nature Neuroscience* 17.11 (2014), pp. 1500–1509.

[26] C. Curto et al. "A Simple Model of Cortical Dynamics Explains Variability and State Dependence of Sensory Responses in Urethane-Anesthetized Auditory Cortex". In: *The Journal of Neuroscience* 29.34 (Aug. 2009), pp. 10600–10612.

[27] D. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*. Probability and Its Applications. Springer-Verlag New York, 2002.

[28] B. C. Daniels and I. Nemenman. "Automated Adaptive Inference of Phenomenological Dynamical Models". In: *Nature Communications* 6 (Aug. 2015), pp. 8133+.

[29] P. Deb and P. K. Trivedi. "Demand for Medical Care by the Elderly: A Finite Mixture Approach". In: *Journal of Applied Econometrics* 12 (1997), pp. 313–336.

[30] A. S. Ecker et al. "State Dependence of Noise Correlations in Macaque Primary Visual Cortex". In: *Neuron* 82.1 (Apr. 2014), pp. 235–248.

[31] S. E. Eikenberry and V. Z. Marmarelis. "A Nonlinear Autoregressive Volterra Model of the Hodgkin–Huxley Equations". In: *Journal of Computational Neuroscience* 34.1 (2013), pp. 163–183.

[32] J. Engel. "Models for Response Data Showing Extra-Poisson Variation". In: *Statistica Neerlandica* 38.3 (1984), pp. 159–167.

[33] H. L. Fernandes et al. "Saliency and Saccade Encoding in the Frontal Eye Field During Natural Scene Search". In: *Cerebral Cortex* 24.12 (2014), pp. 3232–3245.

[34] R. W. Friedrich and G. Laurent. "Dynamic Optimization of Odor Representations by Slow Temporal Patterning of Mitral Cell Activity". In: *Science* 291.5505 (Feb. 2001), pp. 889–894.

[35] P. Fries et al. "Modulation of Oscillatory Neuronal Synchronization by Selective Visual Attention". In: *Science* 291.5508 (Feb. 2001), pp. 1560–1563.

[36] R. Frigola, Y. Chen, and C. Rasmussen. "Variational Gaussian Process State-Space Models". In: *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 3680–3688.

[37] J. Gama. “Functional Trees for Classification”. In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. 2001, pp. 147–154.

[38] M. Gan et al. “A Locally Linear RBF Network-Based State-Dependent Ar Model for Nonlinear Time Series Modeling”. In: *Information Sciences* 180.22 (Nov. 2010), pp. 4370–4383.

[39] S. Ganguli et al. “One-Dimensional Dynamics of Attention and Decision Making in LIP”. In: *Neuron* 58.1 (Apr. 2008), pp. 15–25.

[40] M. S. Goldman. “Memory Without Feedback in a Neural Network”. In: *Neuron* 61.4 (Feb. 2009), pp. 621–634.

[41] G. Golub and C. Van Loan. *Matrix Computations*. Matrix Computations. Johns Hopkins University Press, 2012.

[42] R. L. T. Goris, J. A. Movshon, and E. P. Simoncelli. “Partitioning Neuronal Variability”. In: *Nature Neuroscience* 17.6 (June 2014), pp. 858–865.

[43] A. B. Graf et al. “Decoding the Activity of Neuronal Populations in Macaque Primary Visual Cortex”. In: *Nature neuroscience* 14.2 (Feb. 2011), pp. 239–245.

[44] R. M. Haefner et al. “Inferring Decoding Strategies from Choice Probabilities in the Presence of Correlated Variability”. In: *Nature neuroscience* 16.2 (Feb. 2013), pp. 235–242.

[45] D. Hansel and H. Sompolinsky. “Synchronization and computation in a chaotic neural network”. In: *Physical Review Letters* 68.5 (Feb. 1992), pp. 718–721.

[46] B. E. Hansen. “Testing for parameter instability in linear models”. In: *Journal of Policy Modeling* 14.4 (1992), pp. 517–533.

[47] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York, 2009.

[48] T. Hastie and R. Tibshirani. “Generalized Additive Models”. In: *Statistical science* (1986), pp. 297–310.

[49] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Vol. 43. CRC Press, 1990.

[50] J. Hilbe. *Negative Binomial Regression*. Cambridge University Press, 2011, p. 553.

[51] N. L. Hjort and A. Koning. “Tests For Constancy Of Model Parameters Over Time”. In: *Journal of Nonparametric Statistics* 14.1-2 (2002), pp. 113–132.

[52] T. K. Ho. “The random subspace method for constructing decision forests”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20 (1998), pp. 832–844.

[53] A. L. Hodgkin and A. F. Huxley. “A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve”. In: *The Journal of Physiology* 117.4 (1952), pp. 500–544.

[54] T. R. Holford. “The Estimation of Age, Period and Cohort Effects for Vital Rates”. In: *Biometrics* 39.2 (1983), pp. 311–324.

[55] J. J. C. Hongshik Ahn. “Tree-Structured Logistic Model for Over-Dispersed Binomial Data with Application to Modeling Developmental Effects”. In: *Biometrics* 53.2 (1997), pp. 435–455.

[56] K. Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257.

[57] E. M. Izhikevich and R. FitzHugh. “FitzHugh-Nagumo model”. In: 1.9 (2006), p. 1349.

[58] E. M. Izhikevich. *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. Computational Neuroscience. San Francisco, CA, USA: MIT Press, 2010.

[59] J. A. S. James N. Morgan. “Problems in the Analysis of Survey Data, and a Proposal”. In: *Journal of the American Statistical Association* 58.302 (1963), pp. 415–434.

[60] M. Jazayeri and M. N. Shadlen. “Temporal context calibrates interval timing”. In: *Nature Neuroscience* 13.8 (June 2010), pp. 1020–1026.

[61] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 2003.

[62] J. C. Kao et al. “Single-Trial Dynamics of Motor Cortex and Their Applications to Brain-Machine Interfaces”. In: *Nature Communications* 6.7759 (July 2015).

[63] R. C. Kelly et al. “Local Field Potentials Indicate Network State and Account for Neuronal Response Variability”. In: *Journal of Computational Neuroscience* 29.3 (Dec. 2010), pp. 567–579.

[64] H. Kim and W.-Y. Loh. “Classification Trees With Unbiased Multiway Splits”. In: *Journal of the American Statistical Association* 96.454 (2001), pp. 589–604.

[65] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *ArXiv* (2014).

[66] S. Koyama et al. “Approximate Methods for State-Space Models.” In: *Journal of the American Statistical Association* 105.489 (Mar. 2010), pp. 170–180. arXiv: 1004.3476.

[67] C.-M. Kuan and K. Hornik. “The Generalized Fluctuation Test: A Unifying View”. In: *Econometric Reviews* 14.2 (1995), pp. 135–161.

[68] R. Laje and D. V. Buonomano. “Robust Timing and Motor Patterns by Taming Chaos in Recurrent Neural Networks”. In: *Nat Neurosci* 16.7 (July 2013), pp. 925–933.

[69] K. C. Lakshmanan et al. “Extracting Low-Dimensional Latent Structure from Time Series in the Presence of Delays”. In: *Neural Computation* 27.9 (Sept. 2015), pp. 1825–1856.

[70] K. W. Latimer et al. “Single-Trial Spike Trains in Parietal Cortex Reveal Discrete Steps During Decision-Making”. In: *Science* 349.6244 (2015), pp. 184–187.

[71] G. Laurent. “Olfactory Network Dynamics and the Coding of Multidimensional Signals”. In: *Nature reviews. Neuroscience* 3.11 (2002), pp. 884–895.

[72] J. Lawless. “Negative Binomial and Mixed Poisson Regression”. In: *The Canadian Journal of Statistics* 15 (1987), pp. 209–225.

[73] N. Lim et al. “Classification of High-Dimensional Data with Ensemble of Logistic Regression Models”. In: *Journal of biopharmaceutical statistics* 20 (2009), pp. 160–171.

[74] I.-C. Lin et al. “The Nature of Shared Cortical Variability”. In: *Neuron* 87.3 (Aug. 2015), pp. 644–656.

[75] W.-Y. Loh. “Regression Trees with Unbiased Variable Selection and Interaction Detection”. In: *Statistica Sinica* (2002), pp. 361–386.

[76] W.-Y. Loh and Y.-S. Shih. “Split Selection Methods for Classification Trees”. In: *Statistica Sinica* 7.4 (1997), pp. 815–840.

[77] A. Luczak, P. Bartho, and K. D. Harris. “Gating of Sensory Input by Spontaneous Cortical Activity”. In: *Journal of Neuroscience* 33.4 (Jan. 2013), pp. 1684–95.

[78] W. Maass, T. Natschläger, and H. Markram. "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations". In: *Neural Computation* 14 (2002), pp. 2531–2560.

[79] C. K. Machens, R. Romo, and C. D. Brody. "Flexible Control of Mutual Inhibition: A Neural Model of Two-Interval Discrimination". In: *Science* 307.5712 (Feb. 2005), pp. 1121–1124.

[80] J. H. Macke et al. "Empirical Models of Spiking in Neural Populations". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 1350–1358.

[81] V. Mante et al. "Context-Dependent Computation by Recurrent Dynamics in Prefrontal Cortex". In: *Nature* 503.7474 (Nov. 2013), pp. 78–84.

[82] Martn Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[83] M. E. Mazurek and M. N. Shadlen. "Limits to the Temporal Fidelity of Cortical Spike Rate Signals". In: *Nat Neurosci* 5.5 (May 2002), pp. 463–471.

[84] M. E. Mazurek et al. "A Role for Neural Integrators in Perceptual Decision Making". In: *Cerebral Cortex* 13.11 (Nov. 2003), pp. 1257–1269.

[85] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989.

[86] R. Moreno-Bote et al. "Information-Limiting Correlations". In: *Nature Neuroscience* 17.10 (Oct. 2014), pp. 1410–1417.

[87] J. A. Nelder and D. Pregibon. "An Extended Quasi-Likelihood Function". In: *Biometrika* 74 (1987), pp. 221–232.

[88] M. Okun et al. "Diverse Coupling of Neurons to Populations in Sensory Cortex". In: *Nature* 521.7553 (May 2015), pp. 511–515.

[89] M. Opper and C. Archambeau. "The Variational Gaussian Approximation Revisited". In: *Neural Computation* 21.3 (Sept. 2008), pp. 786–792.

[90] T. Ozaki. *Time Series Modeling of Neuroscience Data*. CRC Press, Jan. 2012.

[91] L. Paninski et al. “A New Look at State-Space Models for Neural Data”. In: *Journal of Computational Neuroscience* 29.1-2 (Aug. 2010), pp. 107–126.

[92] I. M. Park et al. “Encoding and Decoding in Parietal Cortex During Sensorimotor Decision-Making”. In: *Nature Neuroscience* 17.10 (Oct. 2014), pp. 1395–1403.

[93] M. Park and J. W. Pillow. “Receptive Field Inference with Localized Priors”. In: *PLoS Comput Biol* 7.10 (Oct. 2011), pp. 1–16.

[94] A. Peyrache et al. “Internally Organized Mechanisms of the Head Direction Sense”. In: *Nature Neuroscience* 18.4 (Mar. 2015), pp. 569–575.

[95] D. Pfau, E. A. Pnevmatikakis, and L. Paninski. “Robust Learning of Low-Dimensional Dynamics from Large Neural Ensembles”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. 2013, pp. 2391–2399.

[96] J. W. Pillow et al. “Spatio-Temporal Correlations and Visual Signaling in a Complete Neuronal Population”. In: *Nature* 454 (2008), pp. 995–999.

[97] R. E. Quandt. “Tests of the Hypothesis that a Linear Regression System Obeys Two Separate Regimes”. In: *Journal of the American Statistical Association* 55.290 (1960), pp. 324–330.

[98] J. R. Quinlan. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[99] J. M. E. R. L. Brown J. Durbin. “Techniques for Testing the Constancy of Regression Relationships over Time”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 37.2 (1975), pp. 149–192.

[100] M. I. Rabinovich et al. “Transient Cognitive Dynamics, Metastability, and Decision Making”. In: *PLoS Computational Biology* 4.5 (May 2008), e1000072+.

[101] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning Series)*. The MIT Press, Nov. 2005.

[102] A. Resulaj et al. “Changes of Mind in Decision-Making”. In: *Nature* 461.7261 (Sept. 2009), pp. 263–266.

[103] P. T. Sadtler et al. “Neural Constraints on Learning”. In: *Nature* 512.7515 (Aug. 2014), pp. 423–426.

[104] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive computation and machine learning. MIT Press, 2002.

[105] L. Song, P. Langfelder, and S. Horvath. "Random Generalized Linear Model: A Highly Accurate and Interpretable Ensemble Predictor". In: *BMC bioinformatics* 14 (2013), p. 5.

[106] R. G. Stevens, S. H. Moolgavkar, and J. A. Lee. "Temporal trends in breast cancer". In: *American journal of epidemiology* 115.5 (1982), pp. 759–777.

[107] D. Sussillo and O. Barak. "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks". In: *Neural Computation* 25.3 (Dec. 2012), pp. 626–649.

[108] W. Truccolo et al. "A Point Process Framework for Relating Neural Spiking Activity to Spiking History, Neural Ensemble and Extrinsic Covariate Effects". In: *J. Neurophysiol* 93.2 (2005), pp. 1074–1089.

[109] R. K. Tsutakawa. "Estimation of Cancer Mortality Rates: A Bayesian Analysis of Small Frequencies". In: *Biometrics* 41.1 (1985), pp. 69–79.

[110] R. K. Tsutakawa. "Mixed Model for Analyzing Geographic Variability in Mortality Rates". In: *Journal of the American Statistical Association* 83.401 (1988), pp. 37–42.

[111] K. R. Ulrich et al. "GP Kernels for Cross-Spectrum Analysis". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 1990–1998.

[112] J. M. Vesin. "An Amplitude-Dependent Autoregressive Model Based on a Radial Basis Functions Expansion". In: *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*. Vol. 3. IEEE, Apr. 1993, pp. 129–132.

[113] M. Watter et al. "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 2746–2754.

[114] R. Wedderburn. "Quasi-Likelihood Functions, Generalized Linear Models, and the Gauss—Newton Method". In: *Biometrika* 61 (1974), pp. 439–447.

[115] W. K. Werner Ploberger. "The Cusum Test with Ols Residuals". In: *Econometrica* 60.2 (1992), pp. 271–285.

[116] K.-F. Wong and X.-J. Wang. “A Recurrent Network Mechanism of Time Integration in Perceptual Decisions”. In: *The Journal of Neuroscience* 26.4 (Jan. 2006), pp. 1314–1328.

[117] Z. Yang et al. “Testing approaches for overdispersion in poisson regression versus the generalized poisson model”. In: *Biometrical Journal* 49.4 (2007), pp. 565–584.

[118] B. M. Yu et al. “Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity”. In: *Journal of neurophysiology* 102.1 (July 2009), pp. 614–635.

[119] A. Zeileis and K. Hornik. “Generalized M-Fluctuation Tests for Parameter Instability”. In: *Statistica Neerlandica* 61.4 (2007), pp. 488–508.

[120] A. Zeileis, T. Hothorn, and K. Hornik. “Model-Based Recursive Partitioning”. In: *Journal of Computational and Graphical Statistics* 17.2 (2008), pp. 492–514.

[121] A. Zeileis, C. Kleiber, and S. Jackman. “Regression Models for Count Data in R”. In: *Journal of Statistical Software* 27 (2008), pp. 1–25.

Appendix

A.1 Derivation of vLGP Equations

The equations used in the algorithm of vLGP heavily use the incomplete Cholesky decomposition

$$\mathbf{K} = \mathbf{G}_{T\times k}\mathbf{G}^\top, \tag{A.1}$$

and the matrix inversion lemma

$$(\mathbf{K}^{-1} + \mathbf{W})^{-1} = \mathbf{K} - \mathbf{K}(\mathbf{W}^{-1} + \mathbf{K})^{-1}\mathbf{K}, \tag{A.2}$$

$$(\mathbf{W}^{-1} + \mathbf{K})^{-1} = \mathbf{W} - \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}. \tag{A.3}$$

The posterior variance and some other quantities are given as follows,

$$\begin{aligned}
\mathbf{\Sigma} =&(\mathbf{K}^{-1} + \mathbf{W})^{-1}\\
=&\mathbf{K} - \mathbf{K}(\mathbf{W}^{-1} + \mathbf{K})^{-1}\mathbf{K}\\
=&\mathbf{G}\mathbf{G}^\top - \mathbf{G}\mathbf{G}^\top(\mathbf{W}^{-1} + \mathbf{G}\mathbf{G}^\top)^{-1}\mathbf{G}\mathbf{G}^\top\\
=&\mathbf{G}\mathbf{G}^\top - \mathbf{G}\mathbf{G}^\top(\mathbf{W} - \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W})\mathbf{G}\mathbf{G}^\top\\
=&\mathbf{G}\mathbf{G}^\top - \mathbf{G}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top + \mathbf{G}\mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top
\end{aligned} \tag{A.4}$$

$$\begin{aligned}
\mathbf{K}^{-1}\mathbf{\Sigma} =&\mathbf{I}_T - (\mathbf{W}^{-1} + \mathbf{K})^{-1}\mathbf{K}\\
=&\mathbf{I}_T - (\mathbf{W}^{-1} + \mathbf{G}\mathbf{G}^\top)^{-1}\mathbf{G}\mathbf{G}^\top\\
=&\mathbf{I}_T - (\mathbf{W} - \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W})\mathbf{G}\mathbf{G}^\top\\
=&\mathbf{I}_T - \mathbf{W}\mathbf{G}\mathbf{G}^\top + \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top
\end{aligned} \tag{A.5}$$

$$\begin{aligned}
\mathbf{\Sigma}\mathbf{K}^{-1} =&\mathbf{I}_T - \mathbf{K}(\mathbf{W}^{-1} + \mathbf{K})^{-1}\\
=&\mathbf{I}_T - \mathbf{G}\mathbf{G}^\top(\mathbf{W}^{-1} + \mathbf{G}\mathbf{G}^\top)^{-1}\\
=&\mathbf{I}_T - \mathbf{G}\mathbf{G}^\top(\mathbf{W} - \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W})\\
=&\mathbf{I}_T - \mathbf{G}\mathbf{G}^\top\mathbf{W} + \mathbf{G}\mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W})
\end{aligned} \tag{A.6}$$

$$
\begin{aligned}
&\mathrm{tr}[\mathbf{K}^{-1}\mathbf{\Sigma}] \\
=&\mathrm{tr}[\mathbf{I}_T - \mathbf{W}\mathbf{G}\mathbf{G}^\top + \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top] \\
=&\mathrm{tr}[\mathbf{I}_T] - \mathrm{tr}[\mathbf{W}\mathbf{G}\mathbf{G}^\top] + \mathrm{tr}[\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top] \\
=&T - \mathrm{tr}[\mathbf{G}^\top\mathbf{W}\mathbf{G}] + \mathrm{tr}[\mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}]
\end{aligned} \tag{A.7}
$$

$$
\begin{aligned}
&\ln\det[\mathbf{K}^{-1}\mathbf{\Sigma}] \\
=&\ln\det[\mathbf{I}_T - (\mathbf{W} - \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W})\mathbf{G}\mathbf{G}^\top] \\
=&\ln\det[\mathbf{I}_k - \mathbf{G}^\top(\mathbf{W} - \mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W})\mathbf{G}] \\
=&\ln\det[\mathbf{I}_k - \mathbf{G}^\top\mathbf{W}\mathbf{G} + \mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}]
\end{aligned} \tag{A.8}
$$

Diagonal

$$
\begin{aligned}
\mathrm{diag}(\mathbf{\Sigma}) =&\mathrm{diag}[\mathbf{G}\mathbf{G}^\top - \mathbf{G}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top + \mathbf{G}\mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top] \\
=&\mathrm{diag}[\mathbf{G}(\mathbf{G}^\top - \mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top + \mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top)] \\
=&[\mathbf{G} \circ (\mathbf{G}^\top - \mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top + \mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G}\mathbf{G}^\top)^\top]\mathbf{j}_k \\
=&[\mathbf{G} \circ (\mathbf{G} - \mathbf{G}\mathbf{G}^\top\mathbf{W}\mathbf{G} + \mathbf{G}\mathbf{G}^\top\mathbf{W}\mathbf{G}(\mathbf{I}_k + \mathbf{G}^\top\mathbf{W}\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{W}\mathbf{G})]\mathbf{j}_k
\end{aligned} \tag{A.9}
$$

The term $\mathbf{G}^\top\mathbf{W}\mathbf{G}$ is frequently used.

A.2 Wong and Wang's Dynamics

We used the following setting in the simulation of Wong and Wang's two-variable dynamics.

$$\frac{ds_i}{dt} = -\frac{s_i}{\tau_s} + (1 - s_i)\gamma H_i \tag{A.10}$$

$$H_i = \frac{ax_i - b}{1 - \exp[-d(ax_i - b)]} \tag{A.11}$$

$$x_1 = J_{N,11}s_1 - J_{N,12}s_2 + I_0 + I_1 \tag{A.12}$$

$$x_2 = J_{N,22}s_2 - J_{N,21}s_1 + I_0 + I_2 \tag{A.13}$$

$$I_i = J_{A,ext}\mu_0\left(1 \pm \frac{c}{100\%}\right) \tag{A.14}$$

where $i = 1, 2$, $a = 270(\mathbf{\Sigma}\mathrm{nC})^{-1}, b = 108\mathrm{Hz}, d = 0.154\mathrm{s}, \gamma = 0.641, \tau_s = 100\mathrm{ms}, J_{N,11} = J_{N,22} = 0.2609\mathrm{nA}, J_{N,12} = J_{N,21} = 0.0497\mathrm{nA}, J_{A,ext} = 0.00052\mathrm{nA}\cdot\mathrm{Hz}^{-1}, \mu_0 = 30\mathrm{Hz}$.

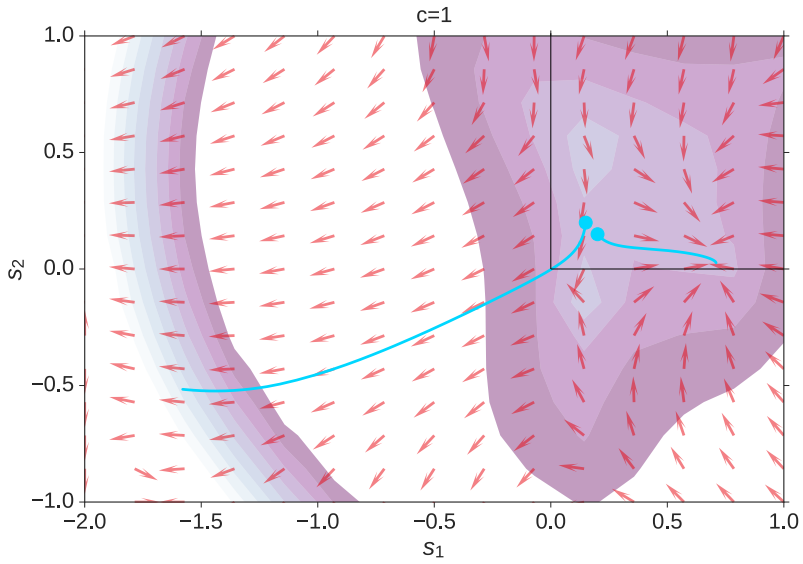


Figure A.1: Failure mode of unregularized locally linear model: 1 s simulation from $\mathbf{x}_{t+1} = \mathbf{A}(\mathbf{x}_t)\mathbf{x}_t + \mathbf{B}(\mathbf{x}_t)\mathbf{u}_t + \mathbf{x}_t$ model (fitted to Wong and Wang's dynamics).