Statistical Models for Neural Data: from Regression / GLMs to Latent Variables

Jonathan Pillow
Princeton Neuroscience Institute

Tutorial
Cosyne 2018
Retinal responses to white noise (ON parasol cells)

neural coding problem

stimulus \( x \)

membrane potential
spikes
imaging

neural activity \( y \)

- How are stimuli and actions encoded in neural activity?
- What aspects of neural activity carry information?
neural coding problem

Approach:
- develop flexible statistical models of $P(y|x)$
- quantify information carried in neural responses
neural coding problem

\[ P(y|x) \]

encoding models

“regression models”

• not restricted to sensory variables
neural coding problem

- Position (P)
- Head direction
- Speed (S)
- Theta phase

“external covariates”

\[ P(y|x) \]

encoding models

“regression models”

- not restricted to sensory variables

membrane potential
spikes
imaging

[Hardcastle et al 2015]
latent variable models

- capture hidden structure underlying neural activity
  (e.g. low-dimensional or discrete states)
latent variable models

• capture hidden dynamics underlying neural activity

latent variable
(unobserved or “hidden”)

spikes

membrane potential

imaging

latent dynamics

latent dynamical encoding models

\[ P(y_t | x_t) P(x_t | x_{t-1}) \]
model desiderata

- Linear, Gaussian
- GLM
- Sweet spot
- Multi-compartment Hodgkin-Huxley

Fittability / tractability
(can be fit to data)

Richness / flexibility
(capture realistic neural properties)
What is the code?

Why does the code take this form?

How is it implemented?

normative theories (e.g. “efficient coding”)

descriptive statistical models

anatomy, biophysics

$P(y|x)$
Outline

1. Spike count models & Maximum Likelihood
2. Spike train models (GLMs with spike history)
3. Multiple Spike Train Models (GLMs with coupling)
4. Regularization
5. Beyond GLM
6. Latent variable models
simple example #1: linear Poisson neuron

encoding model: \[ P(y|x, \theta) = \frac{1}{y!} \lambda^y e^{-\lambda} = \frac{1}{y!} (\theta x)^y e^{-(\theta x)} \]

\[ \lambda = \text{mean} = \text{variance} \]

spike rate \[ \lambda = \theta x \]

spike count \[ y \sim \text{Poisson}(\lambda) \]
mean(y) = \theta x
\text{var}(y) = \theta x

P(y|x)

conditional distribution

p(y|x = 5)
mean(y) = \theta x
var(y) = \theta x

conditional distribution

\begin{align*}
P(y|x) \\
p(y|x = 20)
\end{align*}
mean\( (y) = \theta x \)

\[ \text{var}(y) = \theta x \]

Conditional distribution

\[ P(y|x) \]

\[ p(y|x = 35) \]
Maximum Likelihood Estimation:

- given observed data \((Y, X)\), find \(\theta\) that maximizes \(P(Y|X, \theta)\)

\[
P(Y|X, \theta) = \prod_{i=1}^{N} P(y_i|x_i, \theta)
\]

Q: what assumption are we making about the responses?
A: conditional independence across trials!
Maximum Likelihood Estimation:

- given observed data \((Y, X)\), find \(\theta\) that maximizes 

\[
P(Y | X, \theta) = \prod_{i=1}^{N} P(y_i | x_i, \theta)
\]

all spike counts, all stimuli, parameters

Q: what assumption are we making about the responses?

A: conditional independence across trials!

Q: when do we call \(P(Y | X, \theta)\) a likelihood?

A: when considering it as a function of \(\theta\)!
Maximum Likelihood Estimation:

• given observed data \((Y, X)\), find \(\theta\) that maximizes \(P(Y \mid X, \theta)\)

\[ y \sim \text{Pois}(\theta x) \]
\[ \theta = 1.5 \]

• could in theory do this by turning a knob
Maximum Likelihood Estimation:

- given observed data \((Y, X)\), find \(\theta\) that maximizes \(P(Y|X, \theta)\)

\[
P(y|x) \sim \text{Poiss}(\theta x)
\]

\(\theta = 1\)

- could in theory do this by turning a knob
Maximum Likelihood Estimation:

- given observed data \((Y, X)\), find \(\theta\) that maximizes \(P(Y|X, \theta)\)

\[ P(y|x) \]

\[ y \sim \text{Poiss}(\theta x) \]

\[ \theta = 0.5 \]

- could in theory do this by turning a knob
Likelihood function:  $P(Y|X, \theta)$ as a function of  $\theta$

Because data are independent:

$$P(Y|X, \theta) = \prod_i P(y_i|x_i, \theta)$$

$$= \prod \frac{1}{y_i!} (\theta x_i)^{y_i} e^{-\theta x_i}$$
Likelihood function: $P(Y | X, \theta)$ as a function of $\theta$

Because data are independent:

$$P(Y | X, \theta) = \prod_i P(y_i | x_i, \theta)$$

$$= \prod \frac{1}{y_i!} (\theta x_i)^{y_i} e^{-\theta x_i}$$

log-likelihood:

$$\log P(Y | X, \theta) = \sum_i \log P(y_i | x_i, \theta)$$

$$= \sum y_i \log \theta - \theta x_i + c$$
\[
\log P(Y \mid X, \theta) = \sum_i \log P(y_i \mid x_i, \theta) \\
= \sum y_i \log \theta - \theta x_i + c \\
= \log \theta \left( \sum y_i \right) - \theta \left( \sum x_i \right)
\]

Do it: solve for $\theta$
\[ \log P(Y|X, \theta) = \sum_i \log P(y_i|x_i, \theta) \]
\[ = \sum y_i \log \theta - \theta x_i + c \]
\[ = \log \theta (\sum y_i) - \theta (\sum x_i) \]

- Closed-form solution when model in “exponential family”

\[
\frac{d}{d\theta} \log P(Y|X, \theta) = \frac{1}{\theta} \sum y_i - \sum x_i = 0
\]

\[ \Rightarrow \hat{\theta}_{ML} = \frac{\sum y_i}{\sum x_i} \]
Properties of the MLE (maximum likelihood estimator)

• **consistent**
  (converges to true $\theta$ in limit of infinite data)

• **efficient**
  (converges as quickly as possible, i.e., achieves minimum possible asymptotic error)
simple example #2: linear Gaussian neuron

spike rate $\mu = \theta x$

spike count $y \sim \mathcal{N}(\mu, \sigma^2)$

encoding model:

$$P(y|x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y - \theta x)^2}{2\sigma^2}}$$
\[
\text{mean}(y) = \theta x \\
\text{var}(y) = \sigma^2
\]
\[ P(y|x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta x)^2}{2\sigma^2}} \]

Log-Likelihood

\[ \log P(Y|X, \theta) = - \sum \frac{(y_i - \theta x_i)^2}{2\sigma^2} + c \]

Differentiate, set to zero, and solve for \( \theta \)
Log-Likelihood

\[ P(y|x, \theta) = \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{(y-\theta x)^2}{2\sigma^2}} \]

Log-Likelihood

\[ \log P(Y|X, \theta) = - \sum \frac{(y_i - \theta x_i)^2}{2\sigma^2} + c \]

\[ \frac{d}{d\theta} \log P(Y|X, \theta) = - \sum \frac{(y_i - \theta x_i)x_i}{\sigma^2} = 0 \]

Maximum-Likelihood Estimator:

\[ \hat{\theta}_{ML} = \frac{\sum y_i x_i}{\sum x_i^2} \]

("Least squares regression" solution)

(Recall that for Poisson, \( \hat{\theta}_{ML} = \frac{\sum y_i}{\sum x_i} \))
example #3: unknown neuron

Be the computational neuroscientist: what model would you use?
Example 3: unknown neuron

More general setup:

- $\lambda = f(\theta x)$: firing rate is nonlinear
- $y \sim Poiss(\lambda)$: Poisson firing

This is a GLM!
“basic” Poisson generalized linear model (GLM)

Linear-Nonlinear-Poisson (LNP) model

- stimulus $x(t)$
- dimensionality reduction
- nonlinear stretching
- noise
- stimulus filter $\theta$
- exponential nonlinearity $f$
- Poisson spiking $\lambda(t)$
- spike rate $\lambda = f(\vec{k} \cdot \vec{x})$
- spike count $y \sim \text{Poiss}(\lambda)$

- also known as a “cascade” model
What is a GLM?

Be careful about terminology:

GLM ≠ GLM

General Linear Model ≠ Generalized Linear Model

(Nelder 1972)
Moral:
Be careful when naming your model!
1. General Linear Model

Examples:

1. Gaussian
   \[ y = \theta \cdot x + \epsilon \]

2. Poisson
   \[ y \sim \text{Poisson}(\theta \cdot x) \]
2. Generalized Linear Model

Examples:
1. Gaussian
   \[ y = f(\vec{\theta} \cdot \vec{x}) + \epsilon \]
2. Poisson
   \[ y \sim \text{Poisson}(f(\vec{\theta} \cdot \vec{x})) \]
2. Generalized Linear Model

Terminology:

\[ \tilde{x} \rightarrow \text{Linear} \rightarrow \text{Nonlinear} \rightarrow \text{Noise} \rightarrow y \]

- \( \tilde{x} \) → Linear → Nonlinear → Noise → \( y \)
- \( \theta \) → “parameters”
- \( f \) → “distribution function”
- \( f^{-1} \) = “link function”
- \( f \) = “the nonlinearity”
Applying it to data

\[ y_t = \vec{k} \cdot \vec{x}_t + \text{noise} \]

response at time \( t \)

linear filter

vector stimulus at time \( t \)

stimulus

response

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0

time ————->
response at time $t$

\[ y_t = \vec{k} \cdot \vec{x}_t + \text{noise} \]

linear filter

vector stimulus at time $t$

walk through the data one time bin at a time

$t = 1$

stimulus

response

Time $y_t$
response at time \( t \)

walk through the data one time bin at a time

\[ y_t = \mathbf{k} \cdot \mathbf{x}_t + \text{noise} \]

linear filter

vector stimulus at time \( t \)

stimulus

response

t = 2

\( \mathbf{x}_t \)

time \( \rightarrow \)

\( y_t \)
The response at time $t$ is given by:

$$y_t = \vec{k} \cdot \vec{x}_t + \text{noise}$$

where $\vec{k}$ is the linear filter and $\vec{x}_t$ is the vector stimulus at time $t$.

Walk through the data one time bin at a time.

For $t = 3$, we have $\vec{x}_t$ as the stimulus.

The response is shown below.
response at time $t$

walk through the data one time bin at a time

$t = 4$

stimulus

$y_t = \vec{k} \cdot \vec{x}_t + \text{noise}$

linear filter

vector stimulus at time $t$

response

stimulus

response

time

$y_t$
response at time $t$

$$y_t = \mathbf{k} \cdot \mathbf{x}_t + \text{noise}$$

linear filter

vector stimulus at time $t$

walk through the data one time bin at a time

$t = 5$

stimulus

response

time $\rightarrow$ $y_t$

$y_t = \mathbf{k} \cdot \mathbf{x}_t + \text{noise}$
response at time $t$

$y_t = \mathbf{k} \cdot \mathbf{x}_t + \text{noise}$

linear filter

vector stimulus at time $t$

walk through the data one time bin at a time

$t = 6$

stimulus

response

time $\longrightarrow$

$y_t$
Build up to following matrix version:

\[ Y = X\vec{k} + \text{noise} \]

\[
\begin{bmatrix}
0 \\
0 \\
1 \\
\vdots
\end{bmatrix} = \begin{bmatrix}
\vdots \\
\vdots \\
\vdots
\end{bmatrix} \begin{bmatrix}
\vec{k}
\end{bmatrix}
\]

\text{design matrix}
Computing maximum likelihood estimate

\[ Y = X \tilde{k} + \text{noise} \]

1. "Linear-Gaussian" GLM:
\[ \hat{k} = \left( X^T X \right)^{-1} X^T Y \]
Computing maximum likelihood estimate

\[ Y = f(X \vec{k}) + \text{noise} \]

\[
\begin{bmatrix}
0 \\
1 \\
\vdots
\end{bmatrix}
= 
\begin{bmatrix}
\vdots \\
\vdots \\
\vdots
\end{bmatrix}
\begin{bmatrix}
\vec{k}
\end{bmatrix}
\]

2. **Poisson GLM:**  
\[ k = \text{glmfit}(X,Y,'\text{Poisson'}); \]

maximum likelihood fit

(assumes exponential nonlinearity by default)
Computing maximum likelihood estimate

\[ Y = f(X\vec{k}) + noise \]

3. **Bernoulli GLM**: \( \vec{k} = \text{glmfit}(X,Y,\text{'binomial'}); \)

outputs 0 and 1 (assumes **logistic** nonlinearity by default)

“logistic regression”
GLM summary

1. Linear-Gaussian GLM: \( Y \mid X, \mathbf{k} \sim \mathcal{N}(X \mathbf{k}, \sigma^2 I) \) \text{ continuous}

   log-likelihood: \[-\frac{1}{2\sigma^2} (Y - X \mathbf{k})^\top (Y - X \mathbf{k}) + \text{const}\]

   MLE: \( \hat{\mathbf{k}} = (X^T X)^{-1} X^T Y \)

2. Poisson GLM: \( y \mid \bar{x}, \mathbf{k} \sim \text{Pois}(f(\bar{x}_t \cdot \mathbf{k})) \) \text{ integer counts}

   log-likelihood: \( \mathcal{L} = Y^\top \log f(X \mathbf{k}) - 1^\top f(X \mathbf{k}) \)

3. Bernoulli GLM: \( y_t \mid \bar{x}_t, \mathbf{k} \sim \text{Ber}(f(\bar{x}_t \cdot \mathbf{k})) \) \text{ binary counts}

   log-likelihood: \( \mathcal{L} = Y^\top \log f(X \mathbf{k}) - (1 - Y)^\top \log(1 - f(X \mathbf{k})) \)

   “logistic regression” if \( f(x) = \frac{1}{1 + e^{-x}} \)
NEXT:

GLMs with spike-history and coupling
stimulus filter

Poisson spiking

stimulus $x(t)$

spike rate $\lambda(t) = f(k \cdot x(t))$

• problem: assumes spiking depends only on stimulus!
Poisson GLM with spike-history dependence

stimulus filter

exponential nonlinearity

probabilistic spiking

stimulus

\[ \lambda(t) = f(\vec{k} \cdot \vec{x}(t) + \vec{h} \cdot \vec{y}_{hst}(t)) \]

\[ = e^{\vec{k} \cdot \vec{x}(t)} \cdot e^{\vec{h} \cdot \vec{y}_{hst}(t)} \]

• output: no longer a Poisson process
• interpretation: “soft-threshold” integrate-and-fire model

(Truccolo et al 2004, Gerstner 2001)
Poisson GLM with spike-history dependence

• interpretation: “soft-threshold” integrate-and-fire model

traditional IF

“soft-threshold” IF
GLM dynamic behaviors

- irregular spiking

stimulus

filter outputs ("currents")

\[ p(\text{spike}) \]

post-spike filter \[ h(t) \]
GLM dynamic behaviors

- regular spiking

stimulus

filter outputs ("currents")

p(spike)

post-spike filter

h(t)
GLM dynamic behaviors

- adaptation

stimulus

filter outputs ("currents")

p(spike)

post-spike filter $h(t)$
GLM dynamic behaviors

- bursting

stimulus

filter outputs
("currents")

p(spike)

post-spike filter

$h(t)$
GLM dynamic behaviors (from Izhikevich)

(A) tonic spiking

(B) phasic spiking

(C) tonic bursting

(D) phasic bursting

(E) mixed mode

(F) spike frequency adaptation

(G) type I

(H) type II

(I) spike latency

(J) resonator

(K) integrator

(L) rebound spike

(M) rebound burst

(N) threshold variability

(O) bistability I

(P) bistability II

Figure 6: Suite of dynamical behaviors of Izhikevich and GLM neurons. Each panel, top to bottom: stimulus (blue), Izhikevich neuron response (black), GLM responses on five trials (gray), stimulus filter (left, blue), and post-spike filter (right, red). Black line in each plot indicates a 50 ms scale bar for the stimulus and spike response. (Differing timescales reflect timescales used for each behavior in original Izhikevich paper (Izhikevich, 2004)). Stimulus filter and post-spike filter plots all have 100 ms duration.

Weber & Pillow 2017
multi-neuron GLM

stimulus filter → + → exponential nonlinearity → probabilistic spiking

post-spike filter

stimulus

neuron 1

neuron 2
multi-neuron GLM

stimulus

stimulus filter

exponential nonlinearity

probabilistic spiking

post-spike filter

coupling filters

neuron 1

+ +

neuron 2

+ +

...
GLM equivalent diagram:

\[ \lambda_i(t) = \exp(k_i \cdot x(t) + \sum_j h_{ij} \cdot y(t)) \]
Example dataset

- stimulus = binary flicker
- parasol retinal ganglion cell spike responses
Example dataset

- stimulus = binary flicker
- parasol retinal ganglion cell spike responses
**Stimulus-only GLM**

- **Design matrix**
  - $X$

- **Spike response**
  - $Y$

**Model**

$$P(Y|X)$$

**Diagram Description**

- The design matrix $X$ is represented with a time lag and time dimension.
- The spike response $Y$ is shown as a vertical bar with varying intensities, indicating the model output.

**Note**: The diagram illustrates the relationship between the stimulus (design matrix $X$) and the resulting spike response $Y$, with the model $P(Y|X)$ showing the probabilistic relationship.
Stimulus + SpikeHistory GLM

\[ Y \sim X \]

model

\[ P(Y \mid X) \]
Stimulus + History + 3 Neuron Coupling GLM

design matrix

\[ X \]

spike response

\[ Y \]

model

\[ P(Y|X) \]
Fitting: Maximum Likelihood

Data

\[ x_t \]

\[ y^1_t \]

\[ \vdots \]

\[ y^n_t \]

GLM

\[ k \]

\[ h^1 \]

\[ \vdots \]

\[ h^n \]

- maximize log-likelihood for filters \( \{k, h_1, h_2, \ldots, h_n\} \)

- firing rate:
  \[ \lambda_t = f(\vec{x}_t \cdot \vec{k}) \]

- log-likelihood is concave
- no local maxima \([Paninski 04]\)

\[ \log P(Y|X) = \sum_t y_t \log \lambda_t - \lambda_t \]
convexity and concavity

- everywhere downward curvature
- everywhere upward curvature

- maximizing concave function \iff minimizing a convex function
- preclude existence of non-global local optima
capturing dependencies in multi-neuron responses

[Pillow et al 2008]

retinal receptive fields

cross-correlations

cell #

1  2  3  4  5  6  7  8  9  10

--- data
--- GLM
--- uncoupled GLM

75 sp/s

50 ms
Decoding

- estimate stimuli from the observed spike times
- tool for comparing different encoding models
Q: what was the stimulus?
Decide: response 2

Q: what was the stimulus?
Bayesian Decoding

Bayes’ rule: \[ P(x|y) \propto P(y|x)P(x) \]
Bayesian Decoding

Bayes’ rule:

\[
P(x|y) \propto P(y|x)P(x)
\]

\[
P(y_1|x) \cdots P(y_n|x)
\]

“independent” (uncoupled GLM)

vs.

\[
P(y_1, y_2, \ldots, y_n|x)
\]

“joint encoding” (coupled GLM)
Decoding Comparison

![Bar graph showing comparison between linear decoding and Bayesian decoding with and without coupling. The graph illustrates a 20% increase in log SNR (bits/s) with coupling compared to without coupling.](image)

[Pillow et al 2008]
Regularization
Modern statistics

• more dimensions than samples \( D \geq N \)

\[
\begin{bmatrix}
    y_1 \\
    \vdots \\
    y_N
\end{bmatrix}
= \begin{bmatrix}
    \overrightarrow{x}_1 \\
    \vdots \\
    \overrightarrow{x}_N
\end{bmatrix}
\begin{bmatrix}
    w_1 \\
    \vdots \\
    w_D
\end{bmatrix} + \text{noise}
\]

• fewer equations than unknowns!
• no unique solution
Simulated Example

- 100-element filter ($D=100$)
- 100 noisy samples ($N=100$)

```
true $w$
```

maximum likelihood

```
\[
\text{maximize} \quad \log p(data|w)
\]
```

“overfitting” - parameters fit to details in the training data that are not useful for predicting new data
Simulated Example

- 100-element filter (D=100)
- 100 noisy samples (N=100)

\[ \hat{w} = (X^T X + \lambda I)^{-1} X^T Y \]

true \( w \)

maximum likelihood

"ridge regression"

- biased, but gives improved performance for appropriate choice of \( \lambda \) (James & Stein 1960)
Simulated Example

- 100-element filter (D=100)
- 100 noisy samples (N=100)

true $w$

maximum likelihood

\[
\text{maximize}
\log p(\text{data}|w)
\]

“smoothed”

\[
\text{maximize}
\log p(\text{data}|w) - \lambda \sum (w_i - w_{i-1})^2
\]

Q: how to set the regularization strength $\lambda$?

Simplest answer: use cross-validation!
GLM tutorial (matlab):

code: https://github.com/pillowlab/GLMspiketraintutorial
data: available on request from pillow@princeton.edu

• tutorial1_PoissonGLM.m - fitting of a linear-Gaussian GLM and Poisson GLM (aka LNP model) to RGC neurons stimulated with temporal white noise stimulus.

• tutorial2_spikehistcoupledGLM.m - fitting of a Poisson GLM with spike-history and coupling between neurons.

• tutorial3_regularization_linGauss.m - regularizing linear-Gaussian model parameters using maximum a posteriori (MAP) estimation under two kinds of priors:
  ○ (1) ridge regression (aka "L2 penalty");
  ○ (2) L2 smoothing prior (aka "graph Laplacian").

• tutorial4_regularization_PoissonGLM.m - MAP estimation of Poisson-GLM parameters using same two priors as in tutorial3.
GLM summary

• linear ("dim reduction") + nonlinear + noise
• incorporate spike-history via "spike history" filter
• rich dynamical properties: refractoriness, bursting, adaptation
• incorporate correlations between neurons via "coupling" filters
• flexible tool for encoding & decoding analyses
• regularize to reduce overfitting (essential w/ correlated stimuli)
Beyond GLM
Volterra / Wiener Kernels

Taylor series expansion of a function $f(x)$ in $n$ dimensions

$$y = k_0 + \vec{k}_1 \cdot \vec{x} + \vec{x}^t K_2 \vec{x} + K_3 \cdot \vec{x}^3 + \ldots$$

- $k_0$: constant
- $\vec{k}_1$: vector
- $K_2$: matrix
- $K_3$: 3-tensor

# parameters:
- $k_0$: 1
- $\vec{k}_1$: $n$ (20)
- $K_2$: $n^2$ (400)
- $K_3$: $n^3$ (8000)

- from “systems identification” literature (1960s-70s)
- white noise stimuli
- estimate kernels using moments of spike-triggered stimuli

Lee & Schetzen 1965
Marmarelis & Naka 1972
Korenberg & Hunter 1986
Why are Volterra/Wiener models (generally) bad?

- no output nonlinearity
- polynomials give poor fit to neural nonlinearities (e.g., rectifying, saturating)
- responses may depend on more than one projection of stimulus!
- emphasis on dimensionality reduction
- no longer technically a GLM if fitting nonlinearity $f$
multi-filter LNP

Estimators:

- Spike-triggered covariance (STC) [de Ruyter & Bialek 1998, Schwartz et al 2006]
- Generalized Quadratic Model (GQM) [Park & Pillow 2011; Park et al 2013; Rajan et al 2013]
- maximally informative dimensions (MID) / maximum likelihood [Sharpee et al 2004] [Williamson et al 2015]
extending GLM to conductance-based model [Latimer et al 2014]

membrane dynamics

\[ \frac{dV}{dt} = g_l (E_l - V) + g_e (E_e - V) + g_i (E_i - V) \]

conductances

\[ g_e(t) = f_c(k_e \cdot x(t)) \]
\[ g_i(t) = f_c(k_i \cdot x(t)) \]

inst. spike rate

\[ \lambda(t) = f(V(t)) \]

- shunting inhibition
- adaptive changes in dynamics
extending GLM to conductance-based model

- intracellular recordings in macaque parasol RGCs (Fred Rieke)

Linear filters

- excitatory (from spikes)
- inhibitory (from spikes)
- excitatory (from conductance)
- inhibitory (from conductance)

measured conductances
fit to conductance ($R^2=0.83$)
fit to spikes ($R^2 = 0.63$)

fit to spikes ($R^2 = 0.63$)

fit to spikes ($R^2 = 0.51$)

[Latimer et al 2014]
many other biophysically oriented extensions

Nonlinear input model (NIM)
[McFarland, Cui, & Butts 2013]

Linear-Nonlinear-Kinetics (LNK)
[Ozuysal & Baccus 2014]

Linear-Nonlinear-feedback-delayed-sum-nonlinear-feedback
[Real, Asari, Gollisch & Meister 2017]
If you understand GLMs... you understand DNNs!

- stack many LNs on top of each other: LN LN LN LN LN P
- use gradient ascent to maximize likelihood
- use software (tensorflow, theano) to compute gradients (no more computing gradients by hand!)
- use a bunch of tricks (batches, noise, SGD, dropout, ...)
- do NOT worry about local maxima!
Modern machine learning far outperforms GLMs at predicting spikes

Ari S. Benjamin¹, Hugo L. Fernandes², Tucker Tomlinson³, Pavan Ramkumar²,⁴, Chris VerSteeg¹, Lee Miller¹,²,³, Konrad Paul Kording¹,²,³

macaque M1

Fig 2

Mean $pR^2$

Fig 3

Mean $pR^2$

Fig 4

Mean $pR^2$

Fig 5

Mean $pR^2$

macaque S1

Fig 6

Mean $pR^2$

hippocampus
Modern machine learning far outperforms GLMs at predicting spikes?

Ari S. Benjamin, Hugo L. Fernandes, Tucker Tomlinson, Pavan Ramkumar, Chris VerSteeg, Lee Miller, Konrad Paul Kording

GLMs

NNs

Fig 2

Mean $pR^2$

Fig 3

Mean $pR^2$

Fig 4

Mean $pR^2$

Fig 5

Mean $pR^2$

(No of course not!)

GLM is a special case of NN!
What if there’s no stimulus?
encoding models

What if there’s no stimulus?

$P(y|x)$ encoding models

membrane potential
spikes
imaging
neural activity $y$
latent variable models

latent variable
(unobserved or “hidden”)

$P(y|x)P(x)$
latent encoding models

membrane potential
spikes
imaging

neural activity $Y$
spike responses

Neuron index

Time

(simulated data)

[credit: Jakob Macke]
spike responses

inferred latent variables

Trajectories

Neuron index

[credit: Jakob Macke]
spike responses

Neuron index

Time

[credit: Jakob Macke]
latent variable models = GLMs where we don’t know $x$

good: find shared structure underlying $y$

can it be $x$?
Why are latent variable models hard to work with?

- hard to compute likelihood!
Why are latent variable models hard to work with?

- hard to compute likelihood!

Poisson GLM: fit using:

\[
\log P(Y|X) = \sum_t y_t \log \lambda_t - \lambda_t
\]
Why are latent variable models hard to work with?

- hard to compute likelihood!

sensory encoding model

\[ \log P(Y|X) = \sum_t y_t \log \lambda_t - \lambda_t \]

latent variable model

\[ \log P(Y) = \log \int P(Y|X)P(X)dx \]

requires an integral!
Why are latent variable models hard to work with?
- hard to compute likelihood!

fit using:
\[
\log P(Y) = \int \prod_{t=1}^{T} \left( p(y_t|x_t) p(x_t|x_{t-1}) \right) dx_1 dx_2 \cdots dx_T
\]

Note: The integral is high-dimensional and difficult to compute.
Fitting Latent Variable Models

1. Sampling (“MCMC”) - fully Bayesian inference

- procedure for sampling joint distribution: \( P(\theta, \{x\} \mid \{r\}, \{c\}) \)

  1) sample latents: \( x^{(i)} \sim p(x \mid r, c, \theta^{(i)}) \) conditional over latents

  2) sample parameters: \( \theta^{(i+1)} \sim p(\theta \mid r, c, x^{(i)}) \) conditional over parameters
Fitting Latent Variable Models

1. Sampling ("MCMC") - fully Bayesian inference
   • procedure for sampling joint distribution: $P(\theta, \{x\} \mid \{r\}, \{c\})$
     1) sample latents: $x^{(i)} \sim p(x \mid r, c, \theta^{(i)})$ conditional over latents
     2) sample parameters: $\theta^{(i+1)} \sim p(\theta \mid r, c, x^{(i)})$ conditional over parameters

2. Expectation maximization (EM)
   Alternate updating parameters and posterior over latents.
Fitting Latent Variable Models

1. Sampling ("MCMC") - fully Bayesian inference
   • procedure for sampling joint distribution: $P(\theta, \{x\} \mid \{r\}, \{c\})$
     1) sample latents: $x^{(i)} \sim p(x \mid r, c, \theta^{(i)})$ conditional over latents
     2) sample parameters: $\theta^{(i+1)} \sim p(\theta \mid r, c, x^{(i)})$ conditional over parameters

2. Expectation maximization (EM)
   Alternate updating parameters and posterior over latents.

3. Variational inference
   Optimize a lower bound on posterior over parameters
   Easy with modern probabilistic programming languages (STAN, Edward)
Latent Variable models are defined by two quantities:

\[
P(x) \quad \text{latent} \quad P(y|x) \quad \text{mapping} \]

\[
\text{Model}
\]

- Mixture of Gaussians ("clustering")
- Factor analysis (PCA is special case)
- Linear Dynamical Systems (LDS) ("Kalman filter")
- Hidden Markov Model ("HMM")

- discrete
- Gaussian
- linear Gaussian dynamics
- discrete transitions
variational latent Gaussian process (vLGP)

[Zhao & Park 2016]

latent

\[ P(x) \]

mapping

\[ P(y|x) \]

Gaussian Process

Poisson GLM
variational latent Gaussian process (vLGP)

- 63 simultaneously-recorded V1 neurons [Graf et al 2011]
- stimuli: drifting sinusoidal gratings

[Zhao & Park 2016]

Latent dynamics in V1 driven by drifting grating

arXiv: 1604.03053

Yuan Zhao & I. Memming Park
Stony Brook University

https://www.youtube.com/watch?v=CrY5AfNH1ik
variational latent Gaussian process (vLGP)

- 63 simultaneously-recorded V1 neurons [Graf et al 2011]
- stimuli: drifting sinusoidal gratings

[https://www.youtube.com/watch?v=CrY5AfNH1ik](https://www.youtube.com/watch?v=CrY5AfNH1ik)
Summary

• descriptive statistical “encoding” models
• seek to capture structure in data
• formal tools for comparing models
• encoding and decoding analyses via Bayes rule
• models are modular, easy to build / extend / generalize
Big Picture

- large-scale recording technology advancing rapidly
- lots of interesting structure in high-D neural data
- big opportunities in computational / statistical for developing new methods and models to find / exploit this structure!