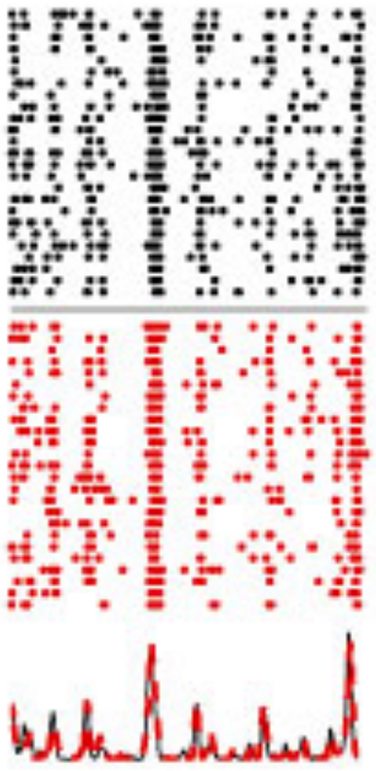
Neural encoding models for understanding neural population responses



Jonathan Pillow

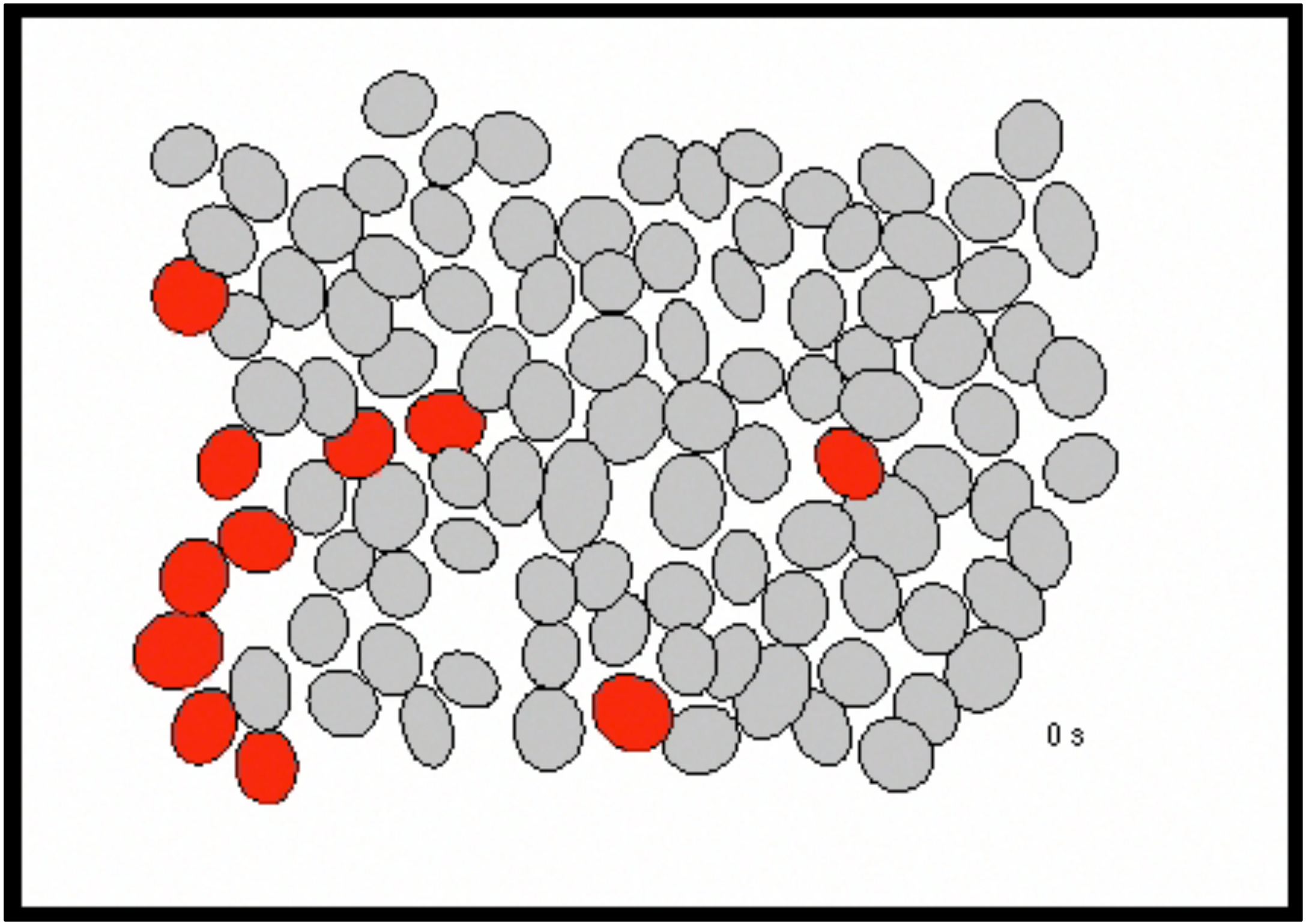
Princeton Neuroscience Institute, Psychology,
Center for Statistics & Machine Learning
Princeton University

Neural Data Science

CSHL summer course 2019

Retinal responses to white noise

(ON parasol cells)

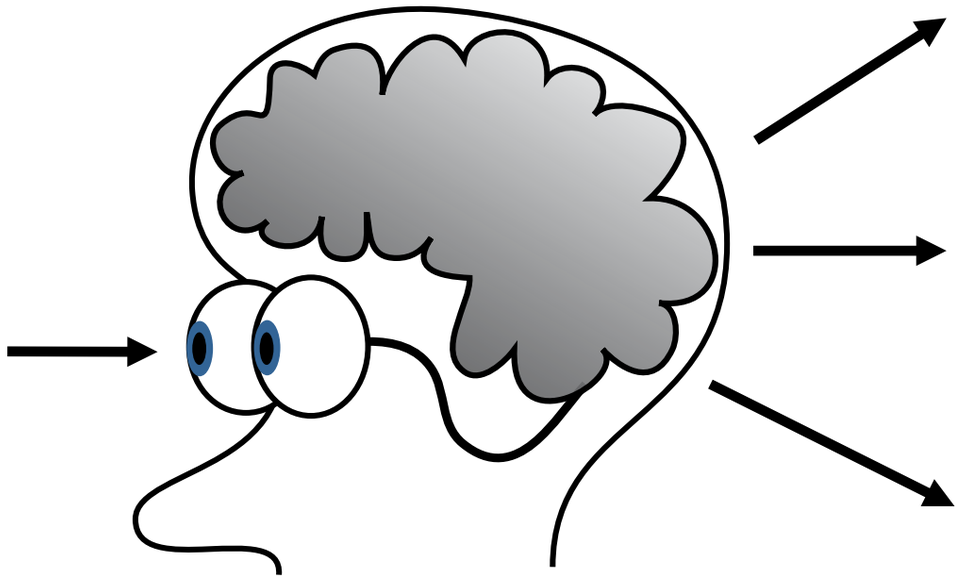


Shlens, Field, Gauthier, Greschner, Sher , Litke & Chichilnisky (2009).

neural coding problem

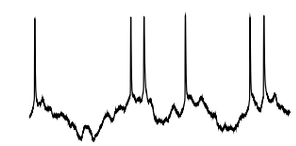
stimulus

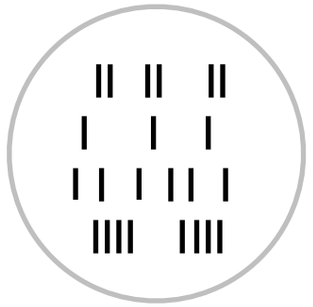
x

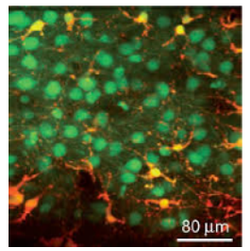


$P(y|x)$

"codebook"







membrane potential

spikes

imaging

neural activity

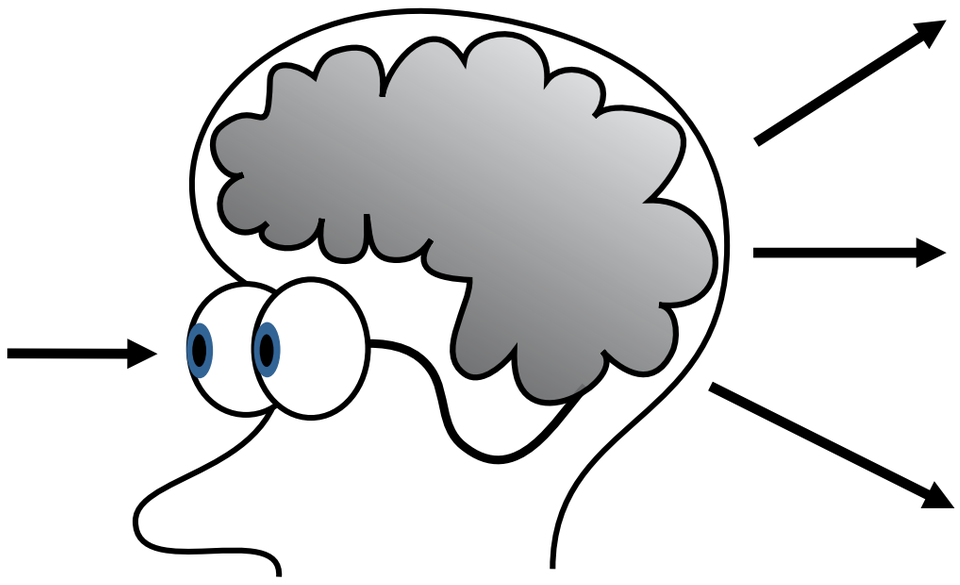
y

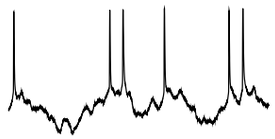
Q: What is the probabilistic relationship between stimuli and neural activity?

model-based approach

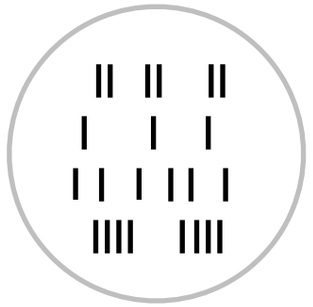
stimulus

x

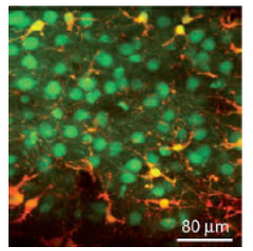




membrane potential



spikes



imaging

neural activity

y

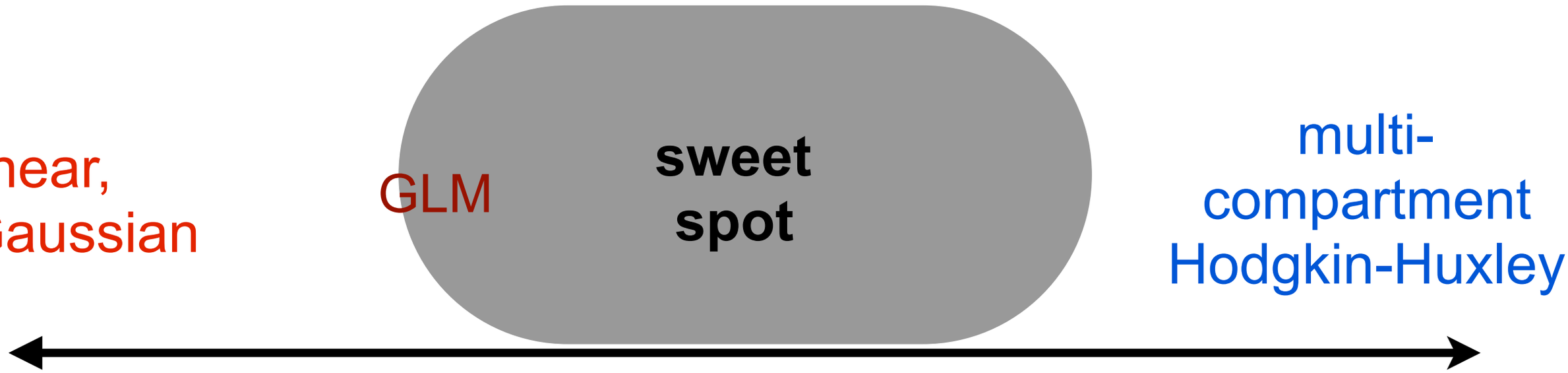
$$P_\theta(y|x) \approx P(y|x)$$

model

Goal: find model that closely approximates true encoding distribution

Question: what criteria for picking a model?

model desiderata



fittability /
tractability

(can be fit to data)

richness /
flexibility

(capture realistic neural properties)

Outline

1. maximum likelihood for simple encoding models

- spike-count codes

$$P(y|x,\theta)$$

2. Multi-neuron models

- Generalized linear model (GLM)

$$P(y_1, y_2, \ldots, y_n|x,\theta)$$

3. **Regularization** (after lunch)

- Ridge regression / Lasso

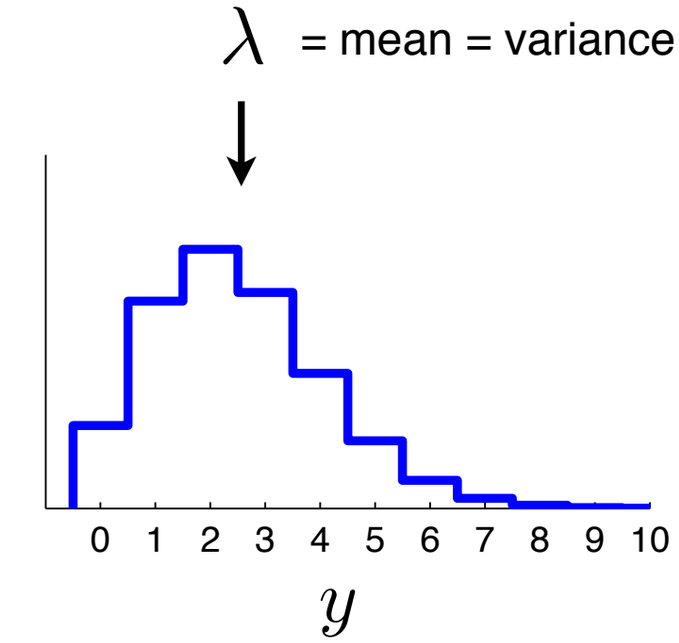
Example 1: linear Poisson neuron

spike count $y \sim Poiss(\lambda)$

spike rate $\lambda = \theta x$

parameter

stimulus

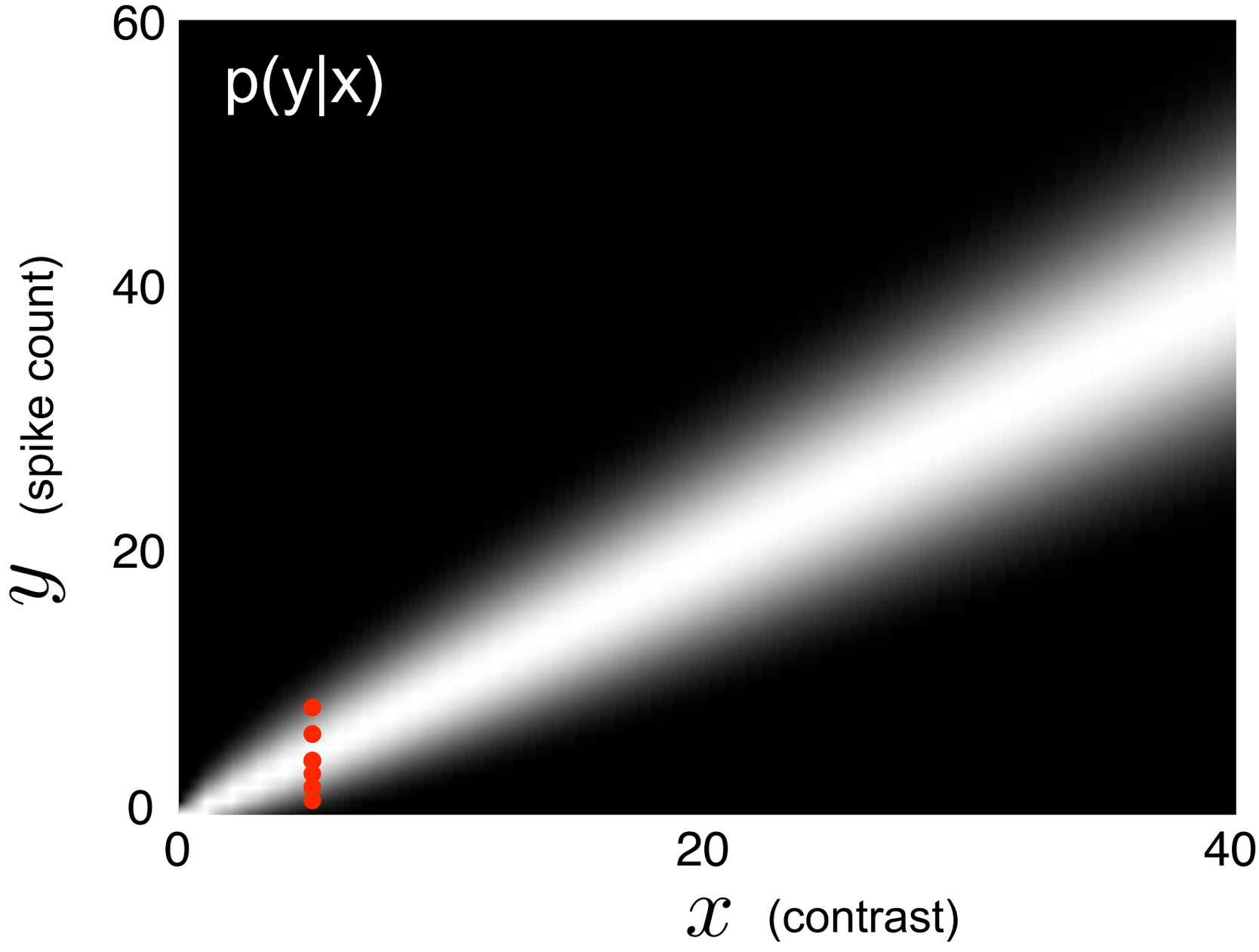


encoding model:

$$P(y|x,\theta) = \frac{1}{y!}\lambda^y e^{-\lambda}$$

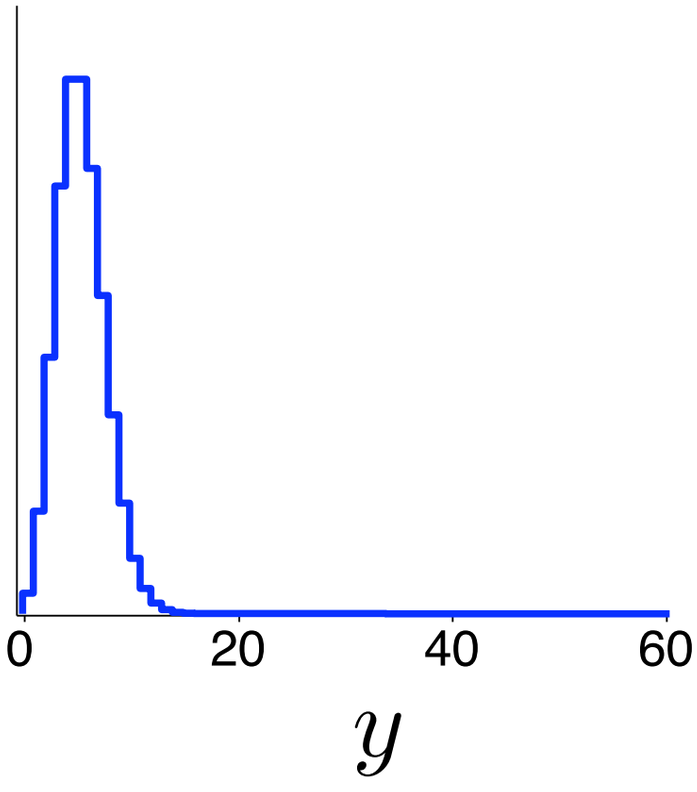
$$= \frac{1}{y!}(\theta x)^y e^{-(\theta x)}$$

$$\text{mean}(y) = \theta x$$
$$\text{var}(y) = \theta x$$

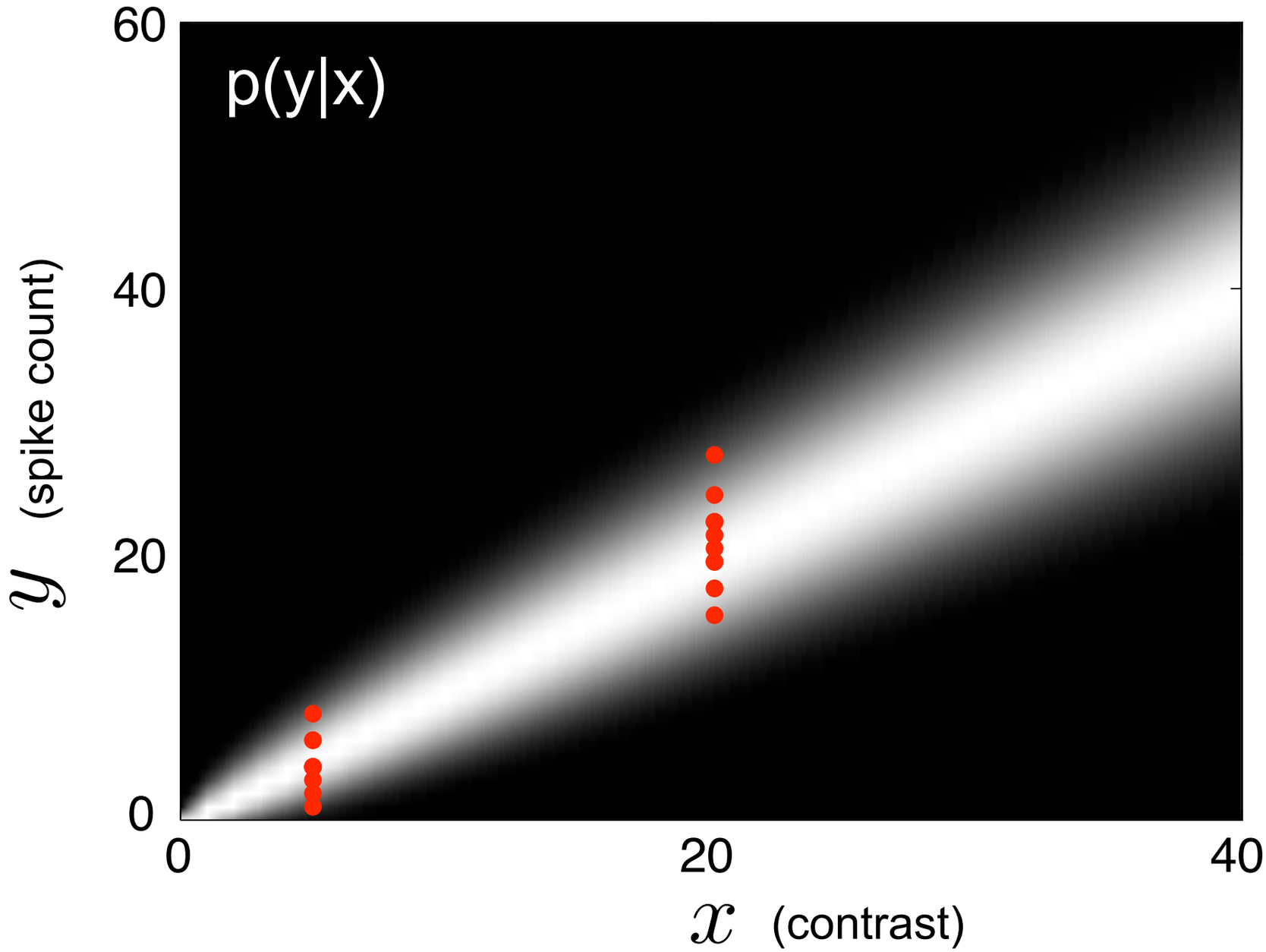


conditional distribution

$$p(y|x = 5)$$

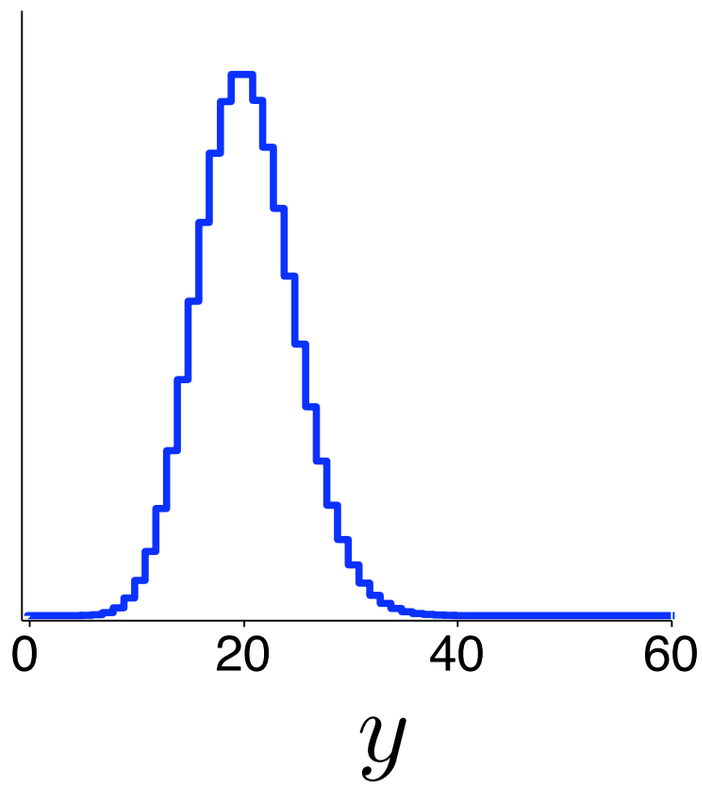


$$\mathrm{mean}(y) = \theta x$$
$$\mathrm{var}(y) = \theta x$$

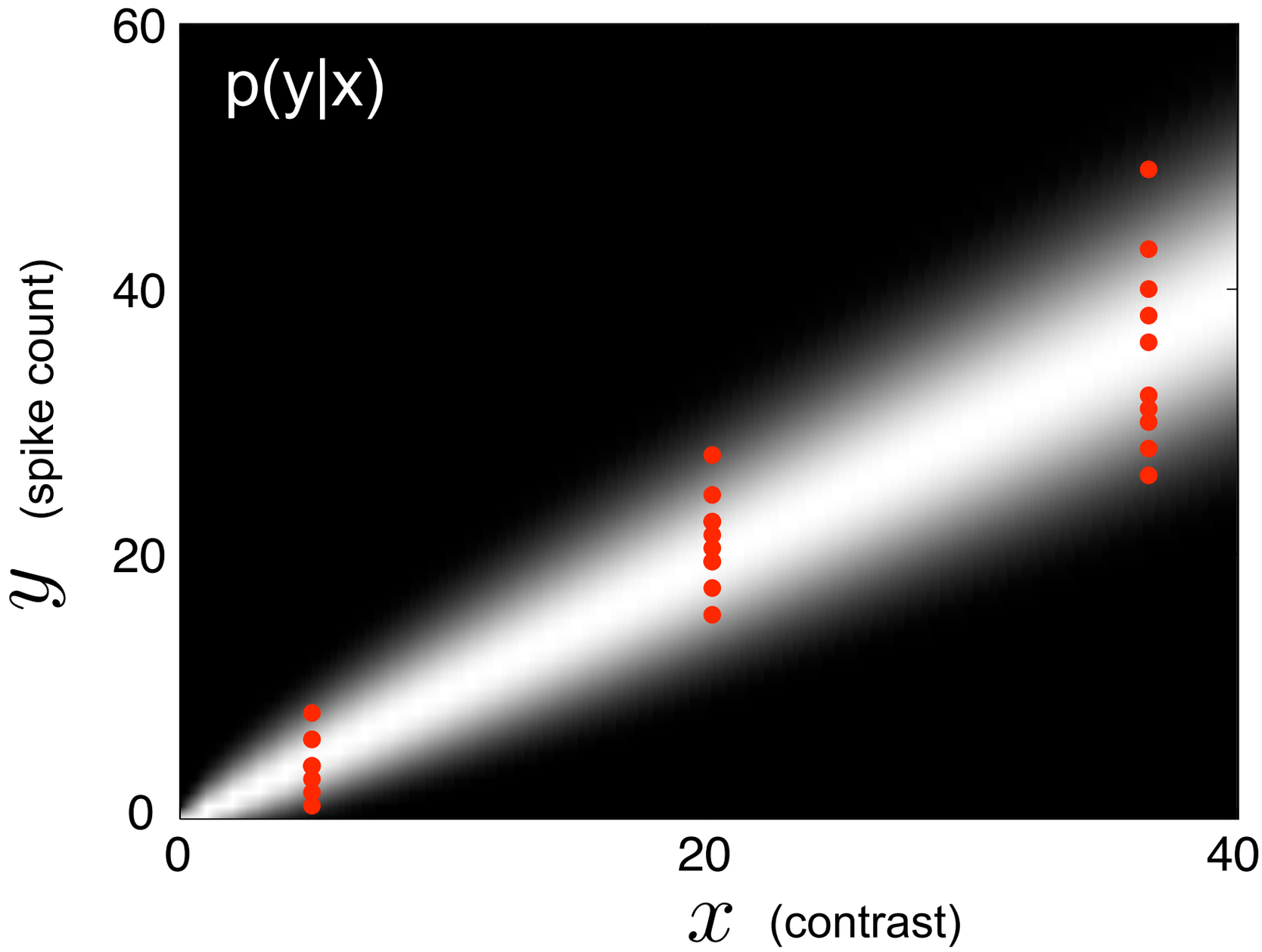


conditional distribution

$$p(y|x = 20)$$

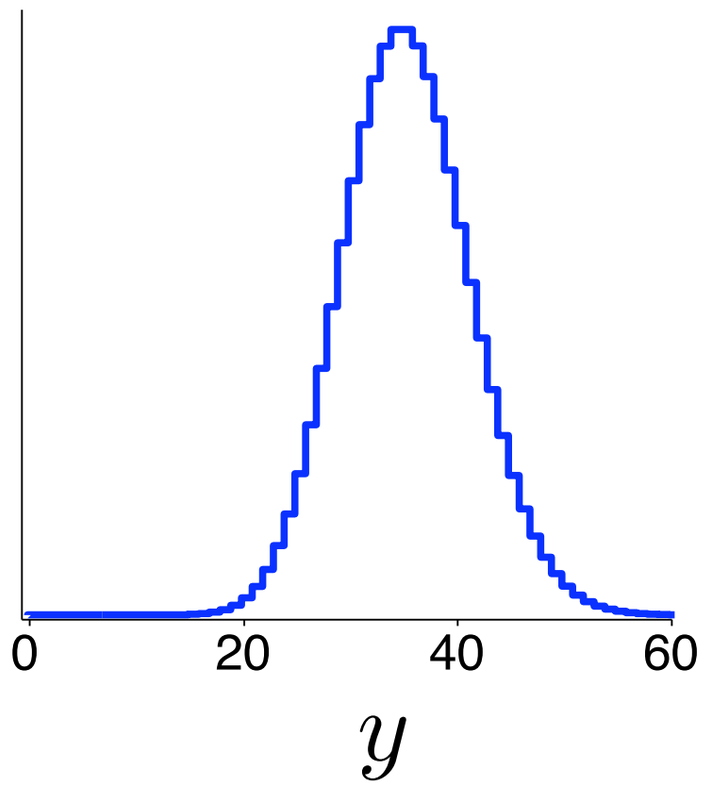


$$\text{mean}(y) = \theta x$$
$$\text{var}(y) = \theta x$$



conditional distribution

$$p(y|x = 35)$$



Maximum Likelihood Estimation:

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

all spike counts — all stimuli — parameters

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

single-trial probability

Q: what assumption are we making about the responses?

A: conditional independence across trials!

Maximum Likelihood Estimation:

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

all spike counts

all stimuli

parameters

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

single-trial probability

Q: what assumption are we making about the responses?

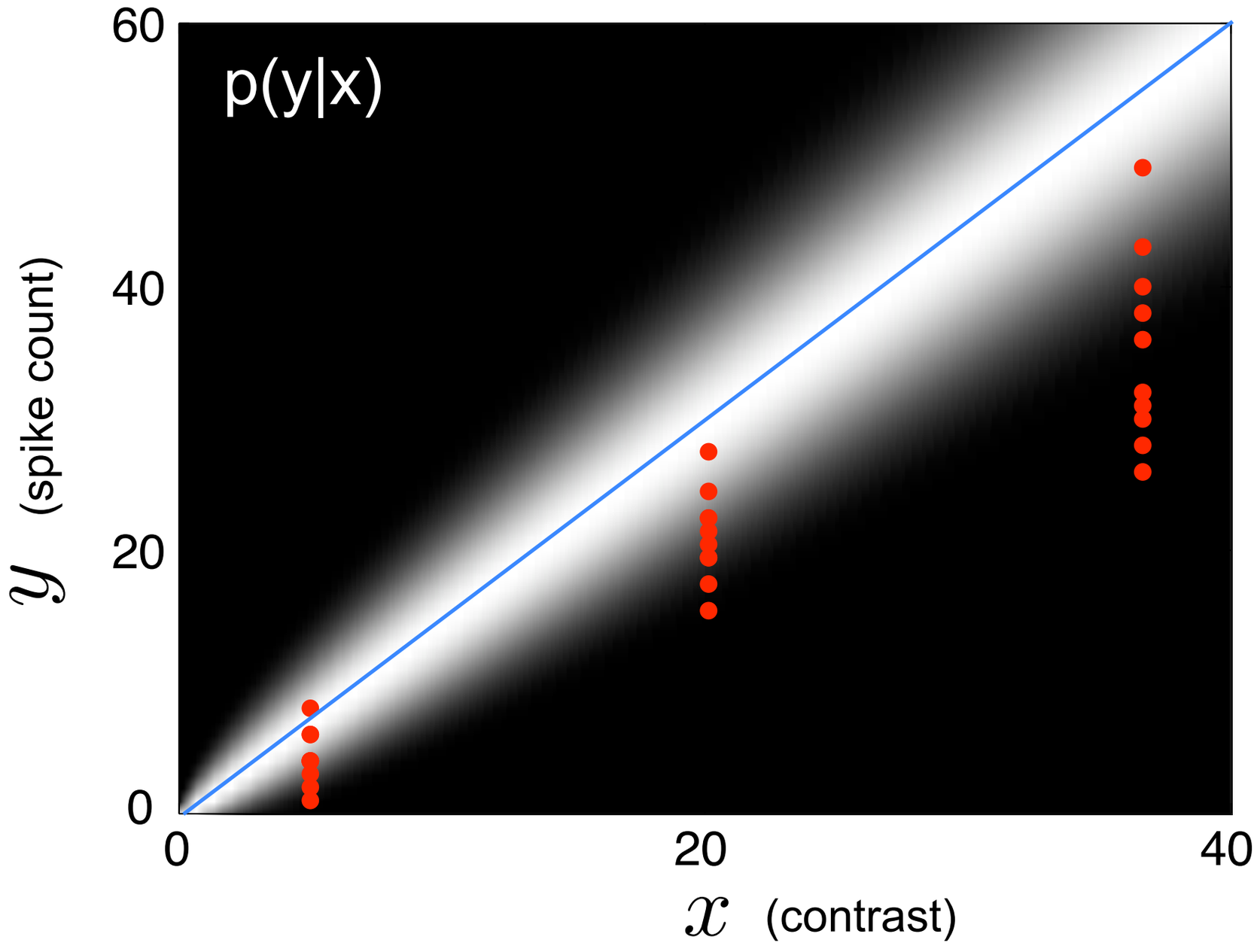
A: conditional independence across trials!

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

A: when considering it as a function of θ !

Maximum Likelihood Estimation:

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



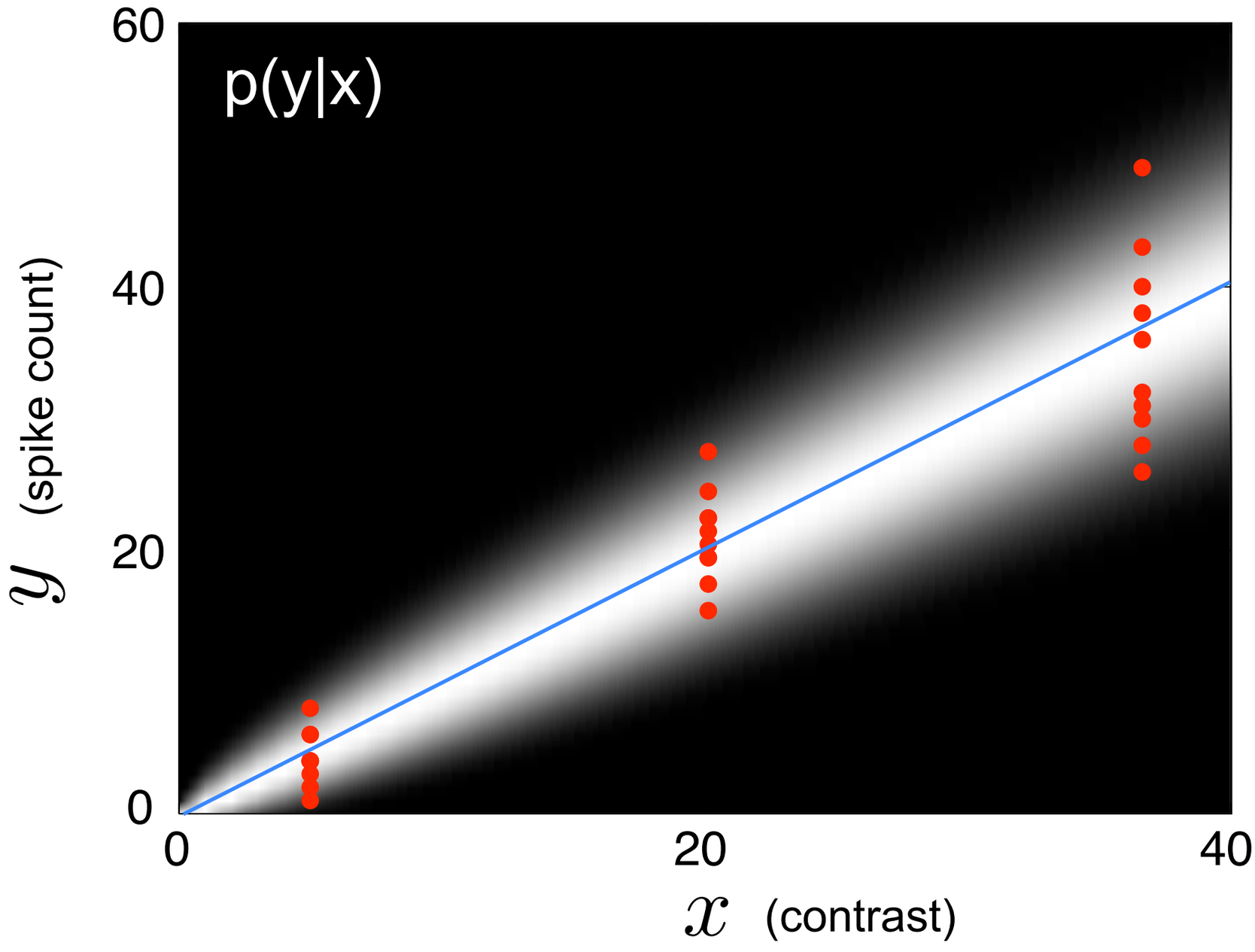
$$y \sim \text{Poiss}(\theta x)$$

$$\theta = 1.5$$

- could in theory do this by turning a knob

Maximum Likelihood Estimation:

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



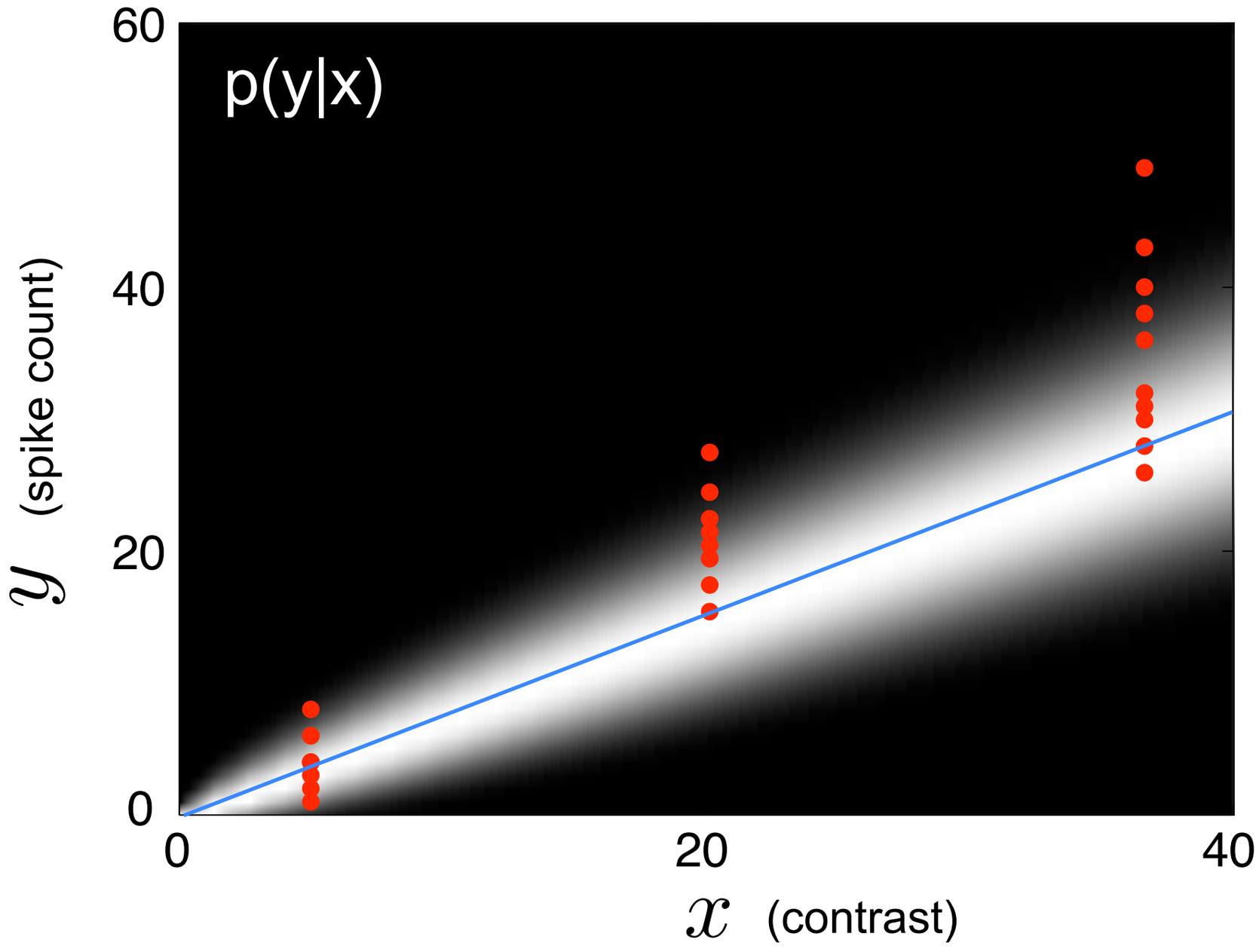
$$y \sim \text{Poiss}(\theta x)$$

$$\theta = 1$$

- could in theory do this by turning a knob

Maximum Likelihood Estimation:

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

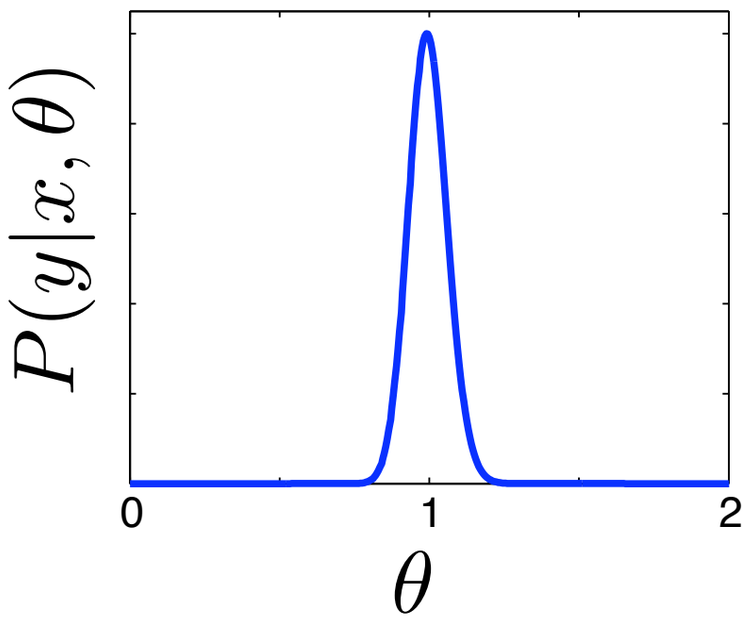


$$y \sim \text{Poiss}(\theta x)$$

$$\theta = 0.5$$

- could in theory do this by turning a knob

Likelihood function: $P(Y|X,\theta)$ as a function of θ



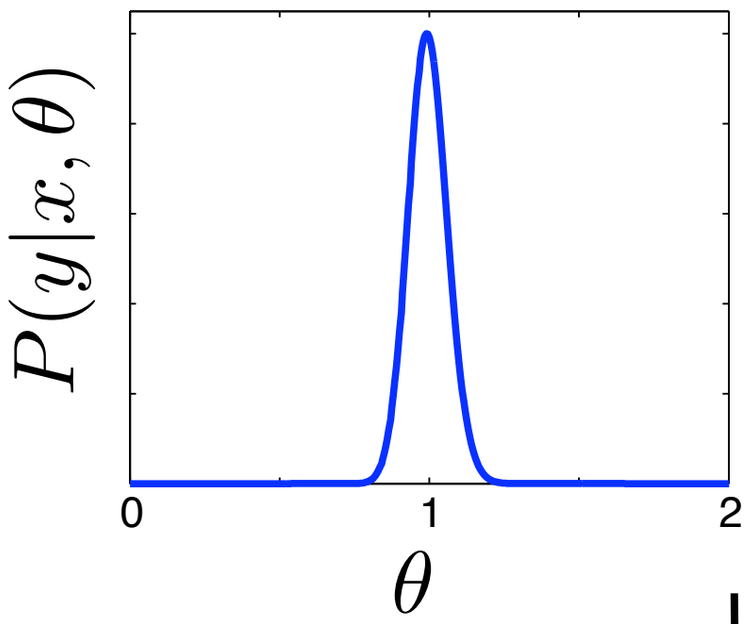
Because data are independent:

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

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

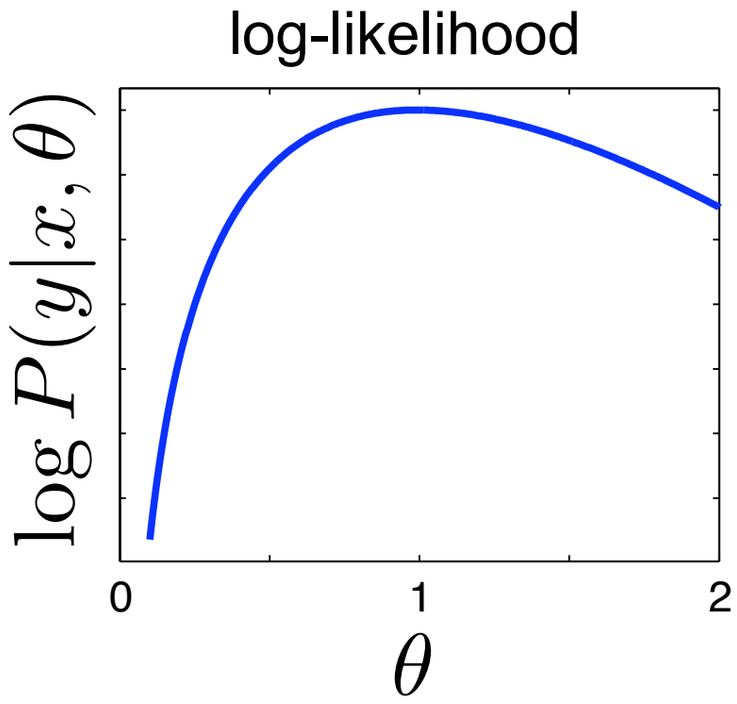
Likelihood function: $P(Y|X,\theta)$ as a function of θ

likelihood



log

log-likelihood



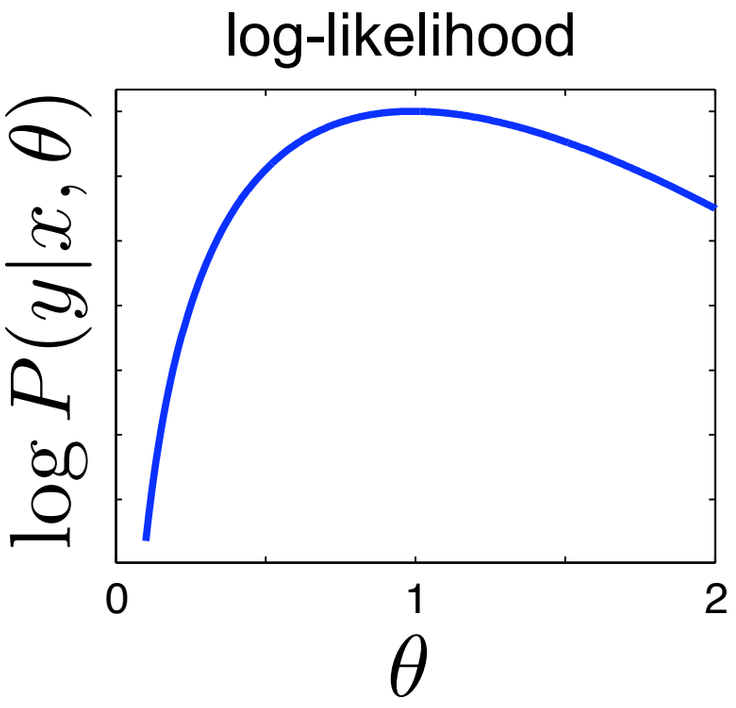
Because data are independent:

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

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

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

$$= \sum y_i \log\theta - \theta x_i + c$$

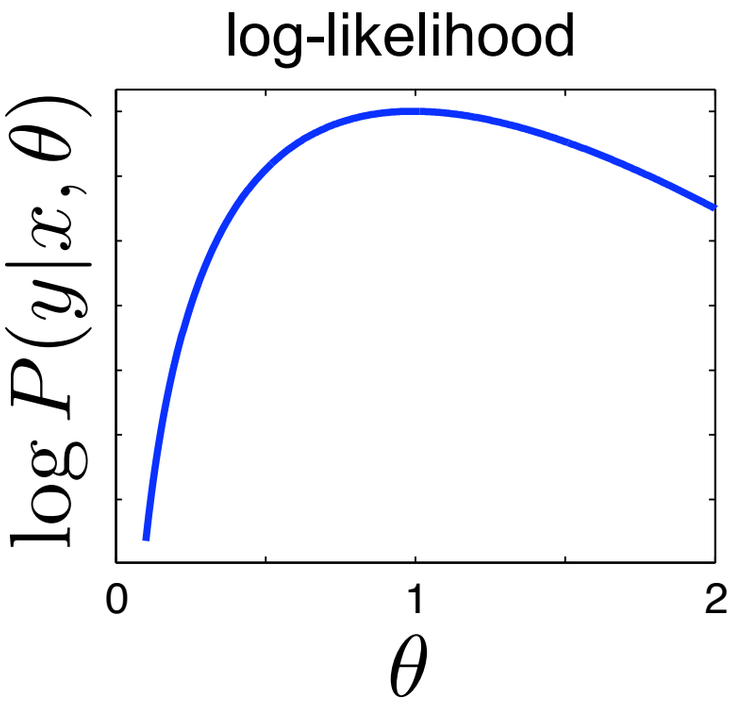


$$\log P(Y|X,\theta) = \sum_i \log P(y_i|x_i,\theta)$$

$$= \sum y_i \log\theta - \theta x_i + c$$

$$= \log\theta(\sum y_i) - \theta(\sum x_i)$$

Do it: solve for θ



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

$$= \sum y_i \log\theta - \theta x_i + c$$

$$= \log\theta(\sum y_i) - \theta(\sum x_i)$$

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

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

$$\Longrightarrow \quad \hat{\theta}_{ML} = \frac{\sum y_i}{\sum x_i}$$

Properties of the MLE (maximum likelihood estimator)

- consistent
  (converges to true θ in limit of infinite data)
- efficient
  (converges as quickly as possible,
  i.e., achieves minimum possible asymptotic error)

Example 2: linear Gaussian neuron

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

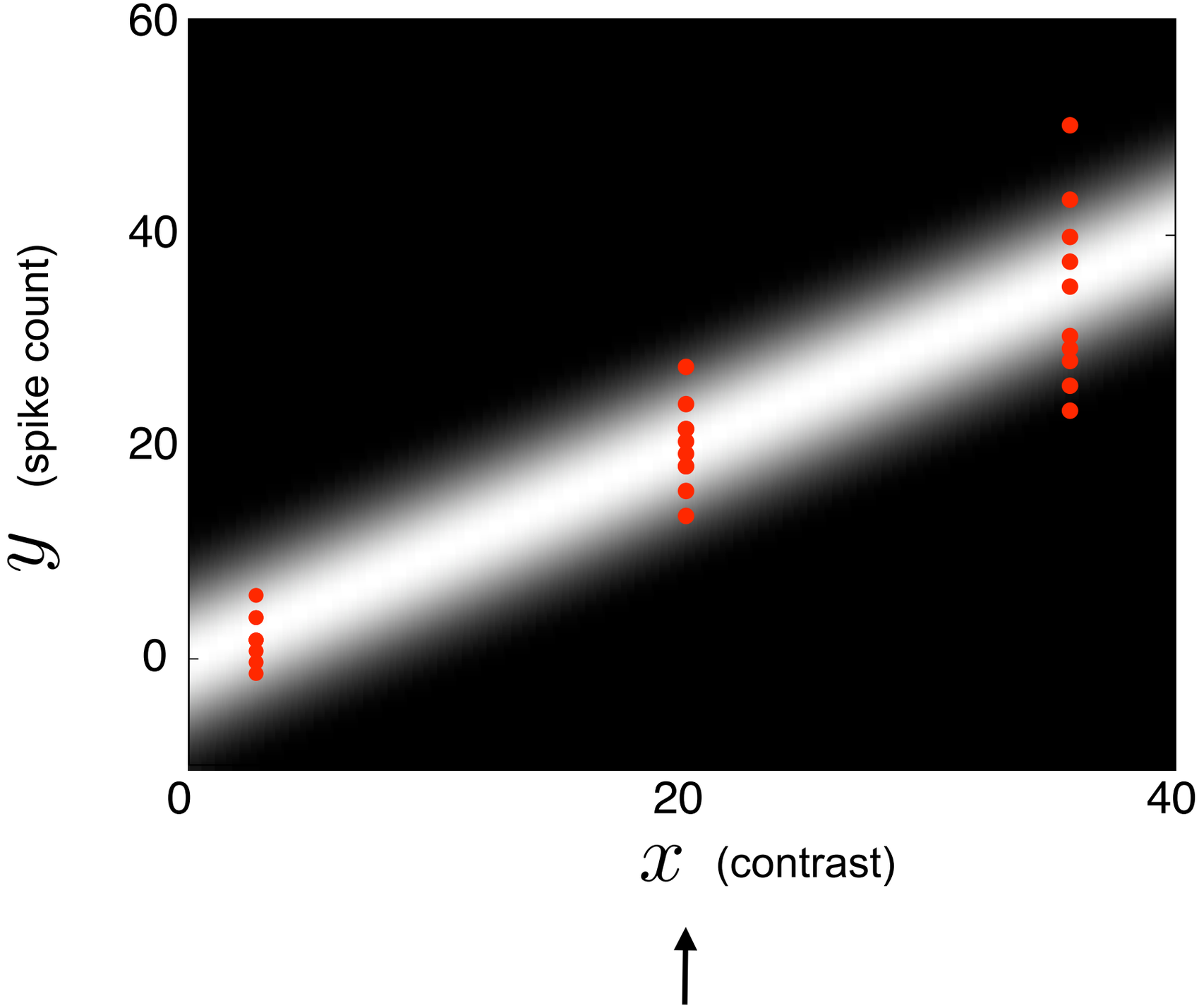
spike rate $\mu = \theta x$

parameter

stimulus

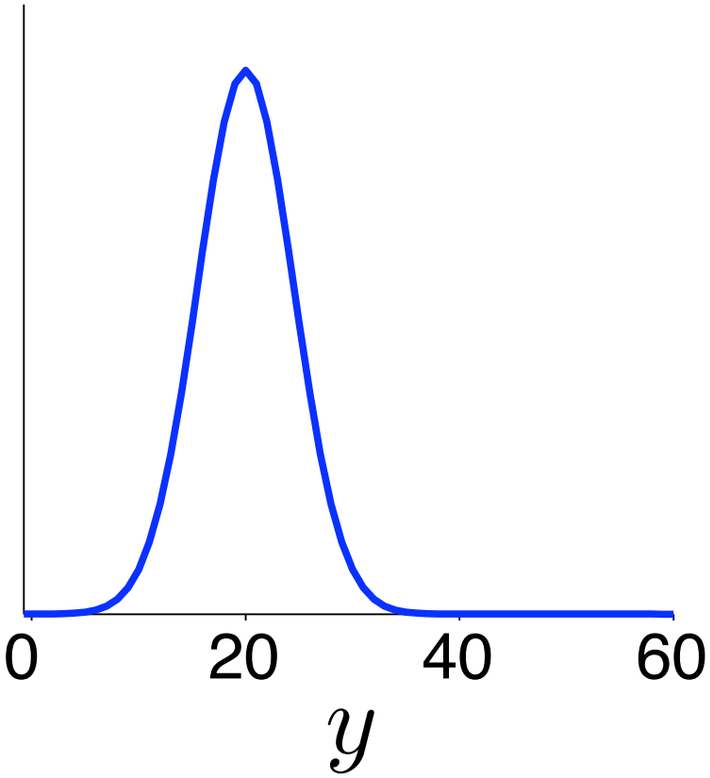
encoding model: $P(y|x,\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-\theta x)^2}{2\sigma^2}}$

$$\text{mean}(y) = \theta x$$
$$\text{var}(y) = \sigma^2$$



encoding distribution

$$p(y|x = 20)$$



All slices have same width

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

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

Do it: differentiate, set to zero, and solve.

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

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

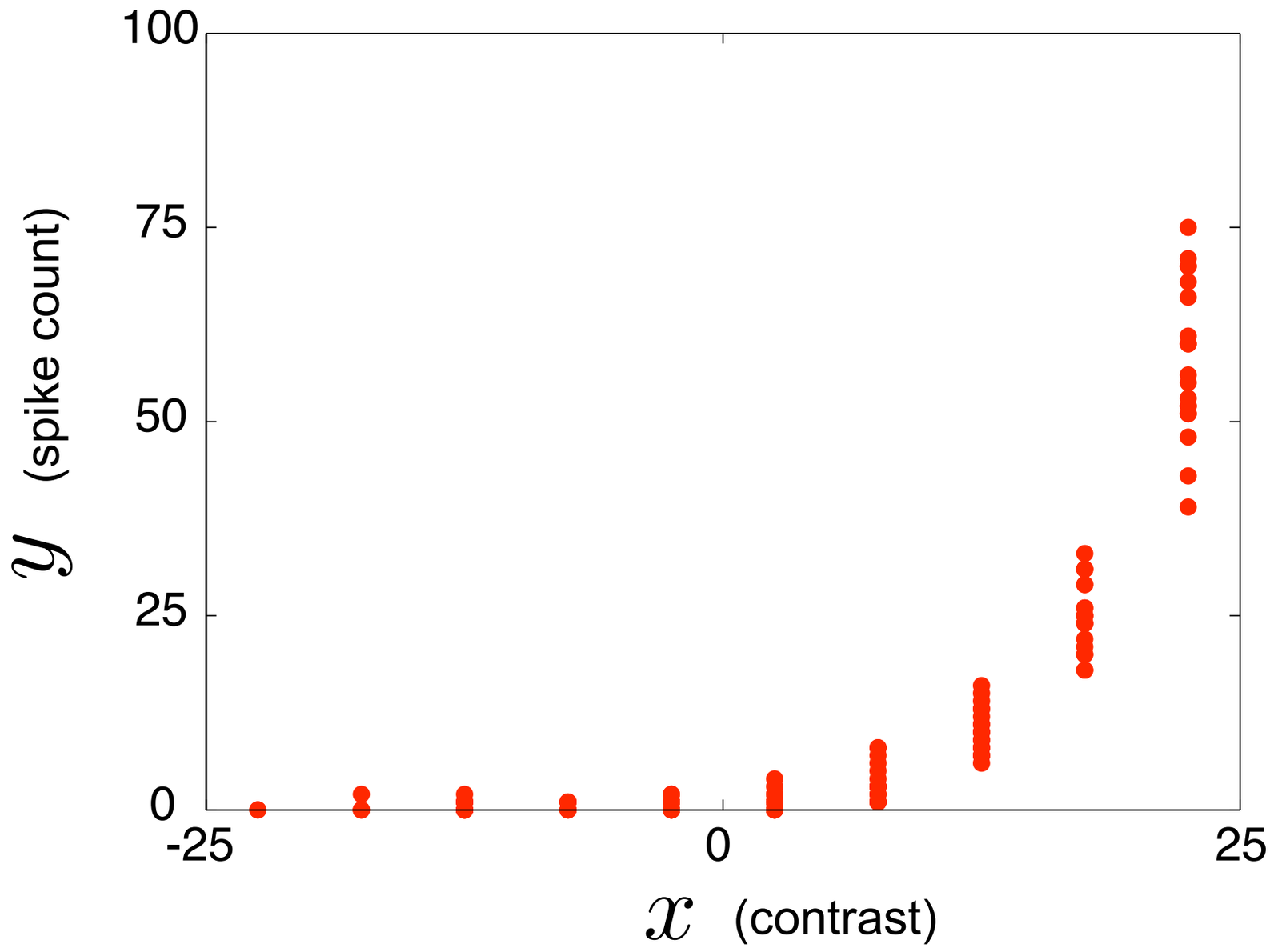
$$\frac{d}{d\theta} \log P(Y|X,\theta) = -\sum \frac{(y_i - \theta x_i)x_i}{\sigma^2} = 0$$

Maximum-Likelihood Estimator: $$\hat{\theta}_{ML} = \frac{\sum y_i x_i}{\sum x_i^2}$$

("Least squares regression" solution)

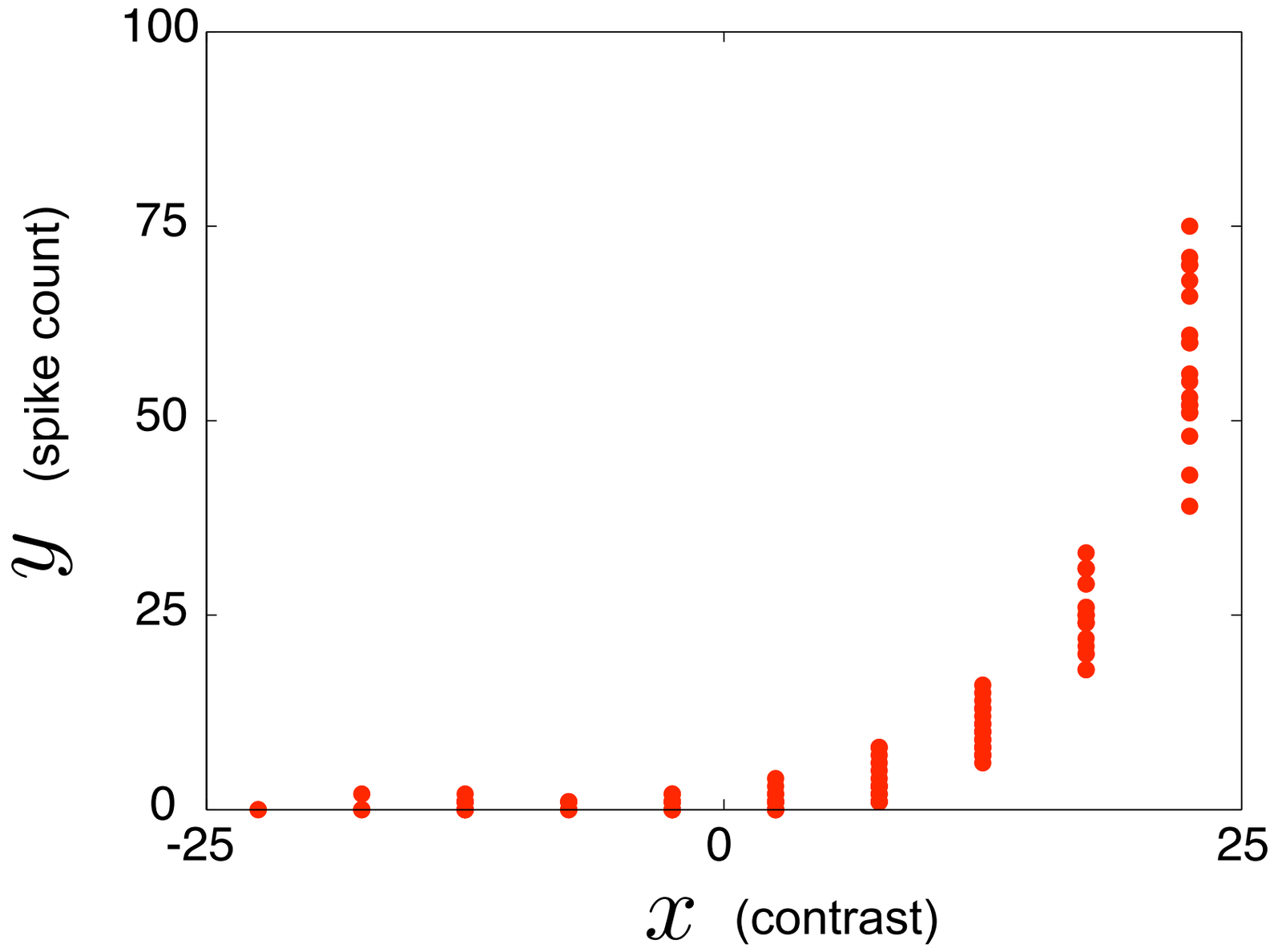
(Recall that for Poisson, $\hat{\theta}_{ML} = \frac{\sum y_i}{\sum x_i}$)

Example 3: unknown neuron



Be the computational neuroscientist: what model would you use?

Example 3: unknown neuron



More general setup: $y \sim Poiss(\lambda)$

$\lambda = f(\theta x)$

for some nonlinear function f

Quick Quiz:

The distribution P(y|x, θ) can be considered as a function of y, x, or θ.

spikes stimulus parameters

What is P(y|x, θ) :

1. as a function of y?

Answer: **encoding distribution** - probability distribution over spike counts

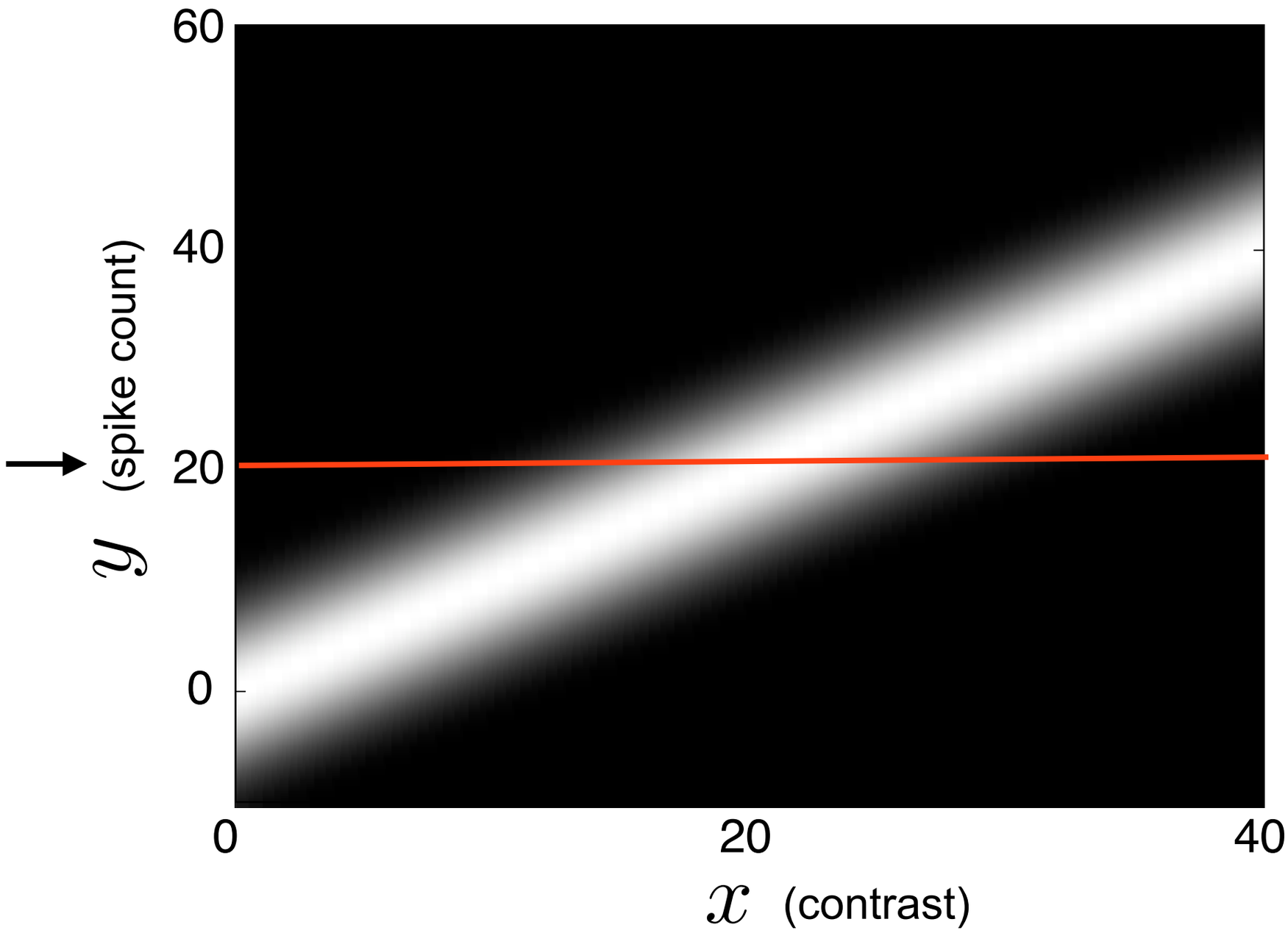
2. as a function of θ?

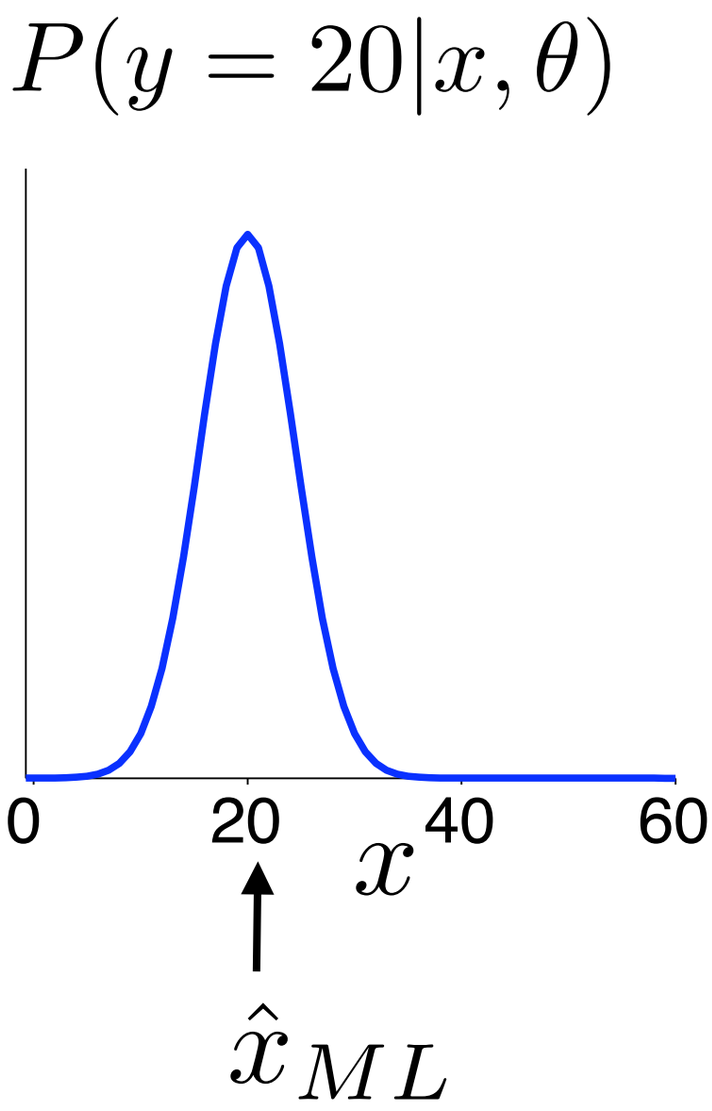
Answer: **likelihood function** - the probability of the data given model params

3. as a function of x?

Answer: **stimulus likelihood function** - useful for ML stimulus decoding!

What is this?





Stimulus likelihood function
(for decoding)

BREAK

Pascal Wallisch @Pascallisch · 13 May 2014

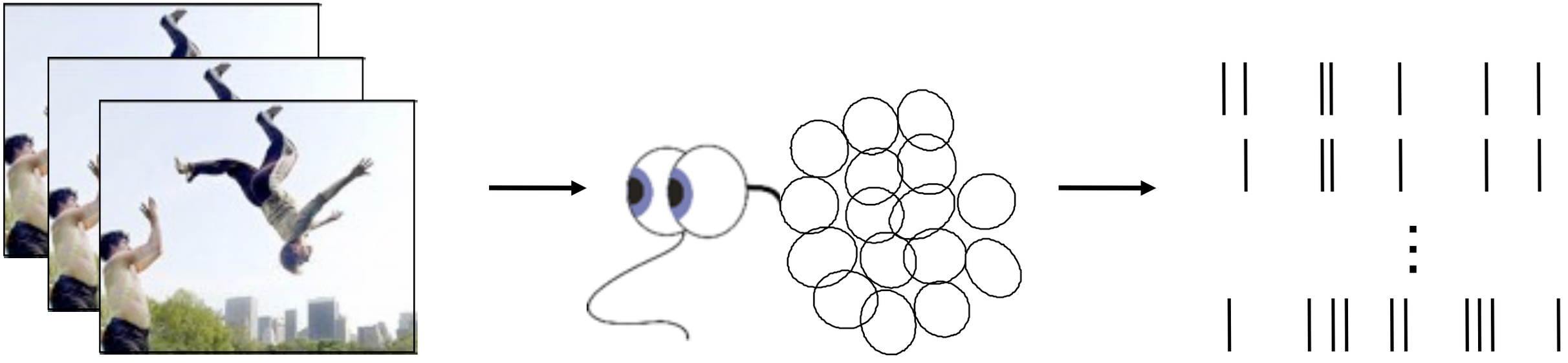
Every time you talk about "**data science**", you might as well say "I'm **stupid**". Is there a **science** without **data**?

2 3 8

Begin Part II:

- modeling neural populations and the "Generalized Linear Model" (GLM).

$$P(y_1, y_2, \ldots, y_n | x, \theta)$$



x

stimuli

y

spike trains

Note on GLMs

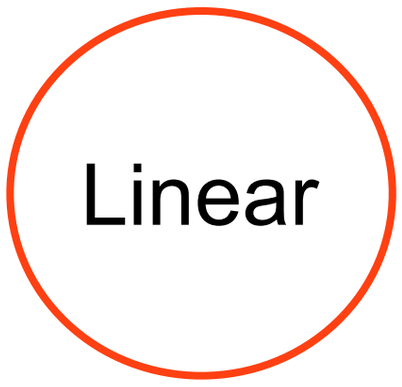
- Be careful about terminology:

GLM ≠ GLM

General Linear Model

Generalized Linear Model

(Nelder 1972)



2003 interview with John Nelder...

Stephen Senn: I must confess to having some confusion when I was a young statistician between general linear models and generalized linear models. Do you regret the terminology?

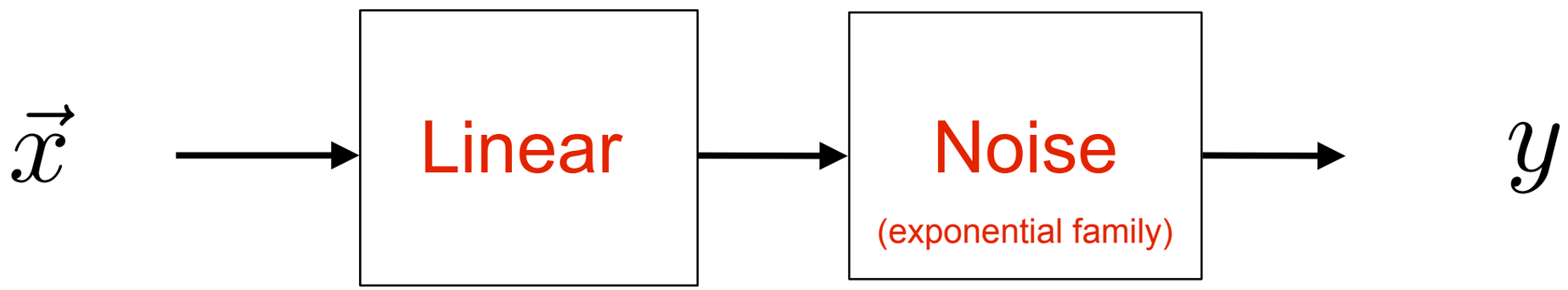
John Nelder: I think probably I do. I suspect we should have found some more fancy name for it that would have stuck and not been confused with the general linear model, although general and generalized are not quite the same. I can see why it might have been better to have thought of something else.

Senn, (2003). *Statistical Science*

Moral:

Be careful when naming your model!

1. General Linear Model



Examples:

1. Gaussian $y = \vec{\theta} \cdot \vec{x} + \epsilon$
2. Poisson $y \sim \text{Poiss}(\vec{\theta} \cdot \vec{x})$

2. Generalized Linear Model

$\vec{\theta}$ f

Examples:

1. Gaussian $y = f(\vec{\theta} \cdot \vec{x}) + \epsilon$
2. Poisson $y \sim \text{Poiss}(f(\vec{\theta} \cdot \vec{x}))$

From spike counts to spike trains:

response at time t

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

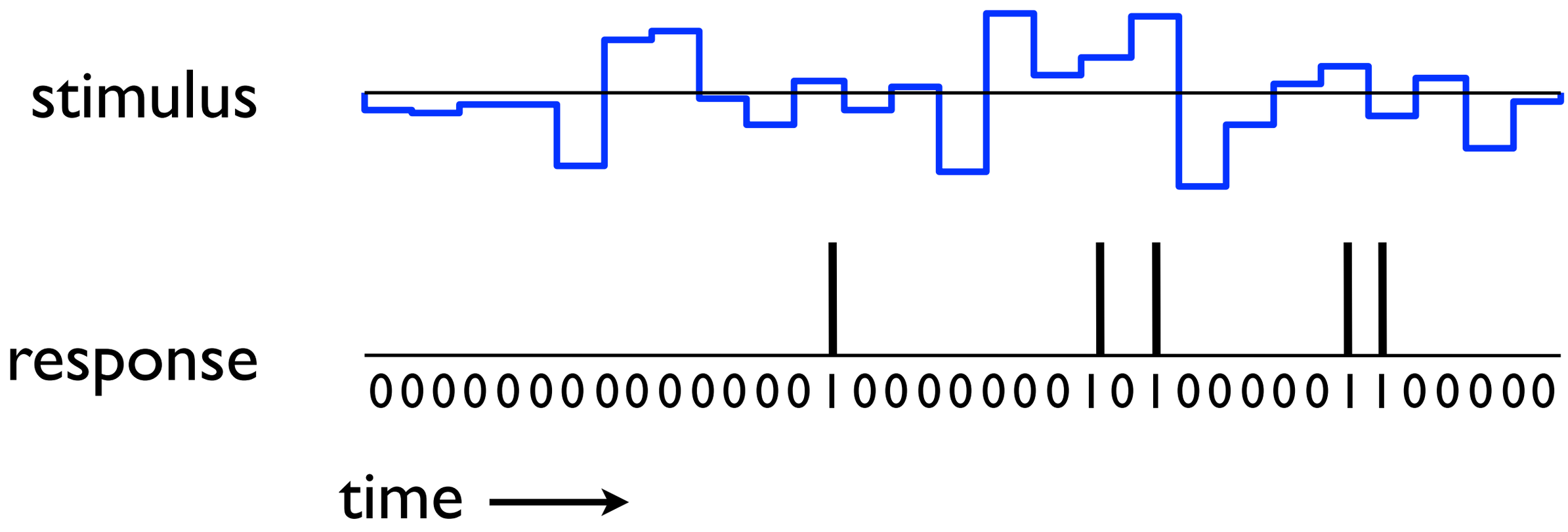
$N(0, \sigma^2)$

linear filter

vector stimulus at time t

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

first idea: linear-Gaussian model!



response at time t

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

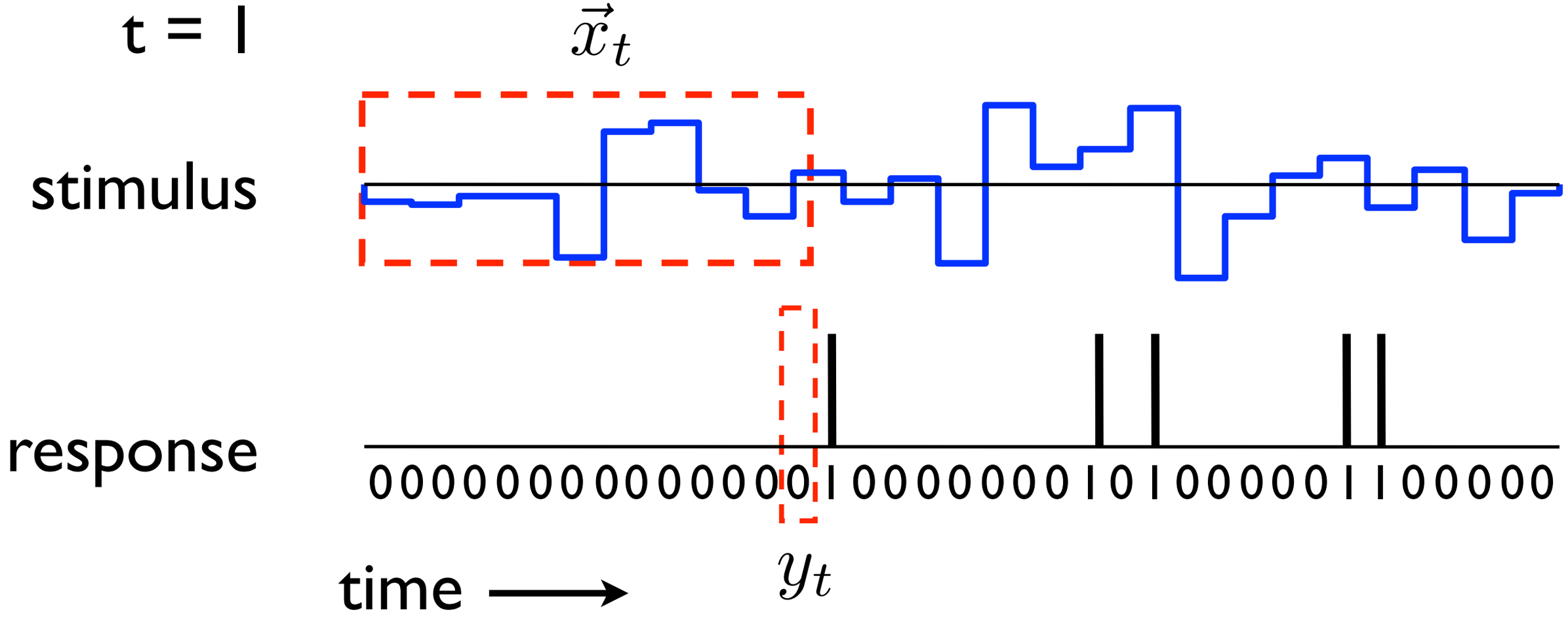
$N(0, \sigma^2)$

linear filter

vector stimulus at time t

walk through the data one time bin at a time

t = 1



response at time t

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

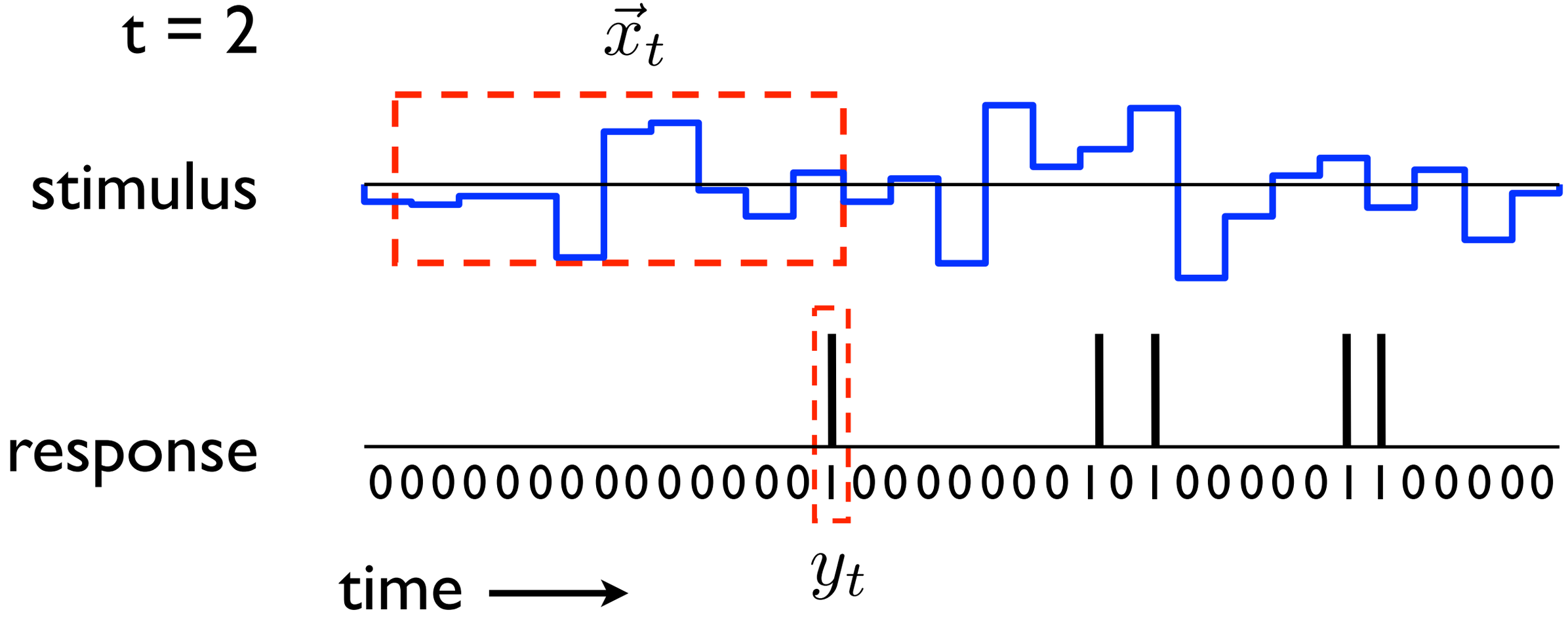
$N(0, \sigma^2)$

linear filter

vector stimulus at time t

walk through the data
one time bin at a time

t = 2



response at time t

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

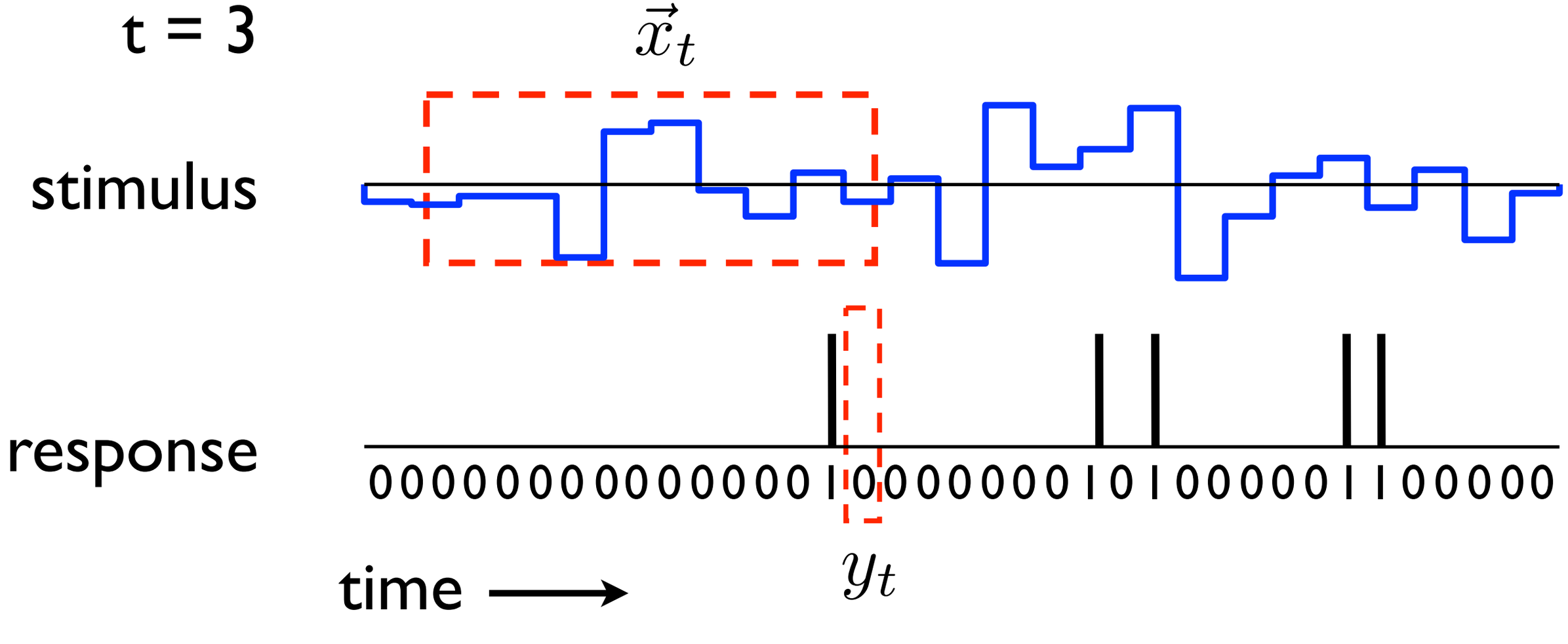
$N(0, \sigma^2)$

linear filter

vector stimulus at time t

walk through the data one time bin at a time

t = 3



response at time t

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

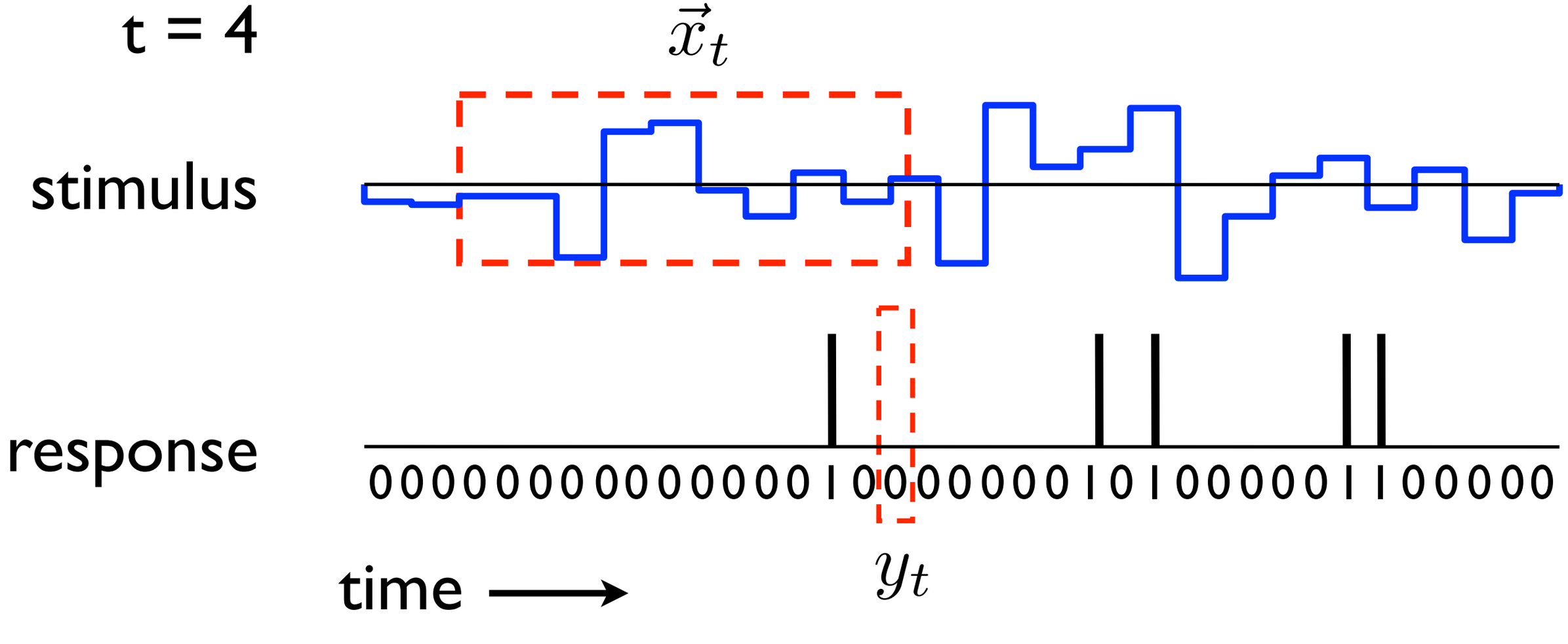
$N(0, \sigma^2)$

linear filter

vector stimulus at time t

walk through the data
one time bin at a time

t = 4



response at time t

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

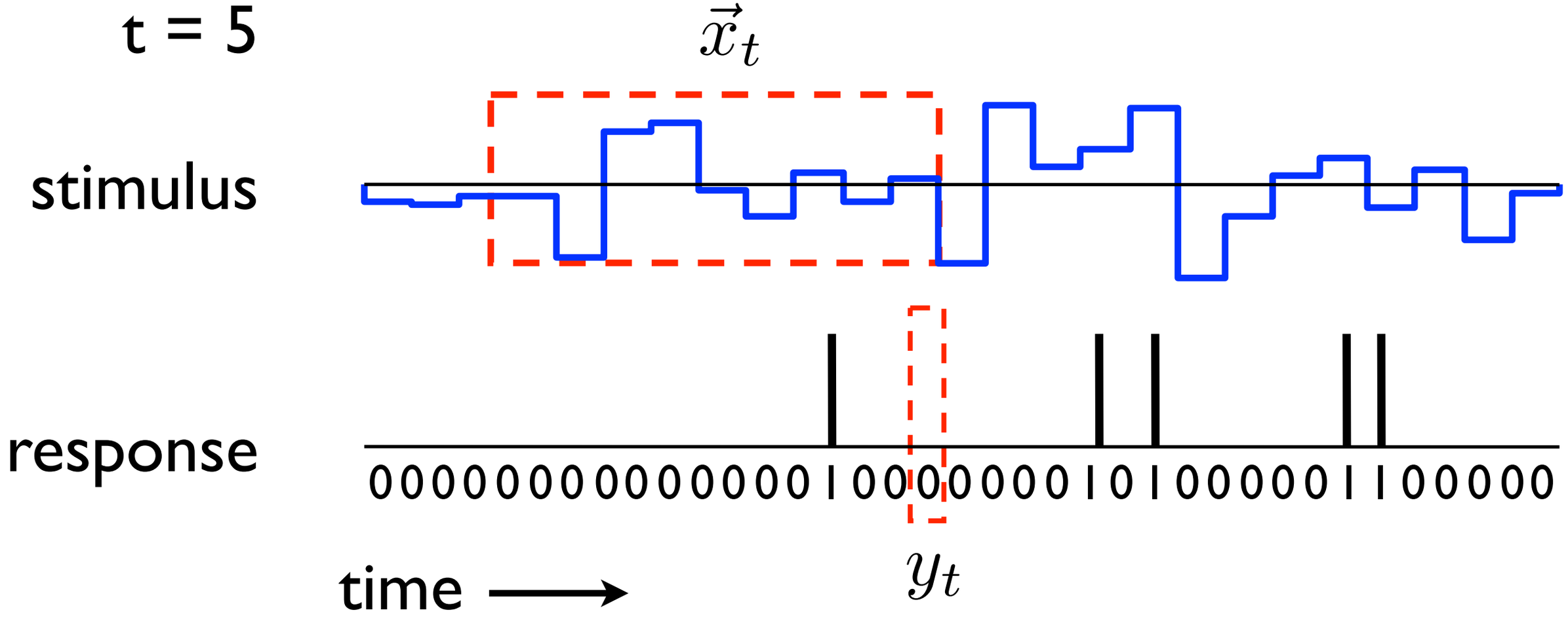
$N(0, \sigma^2)$

linear filter

vector stimulus at time t

walk through the data one time bin at a time

t = 5



response at time t

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

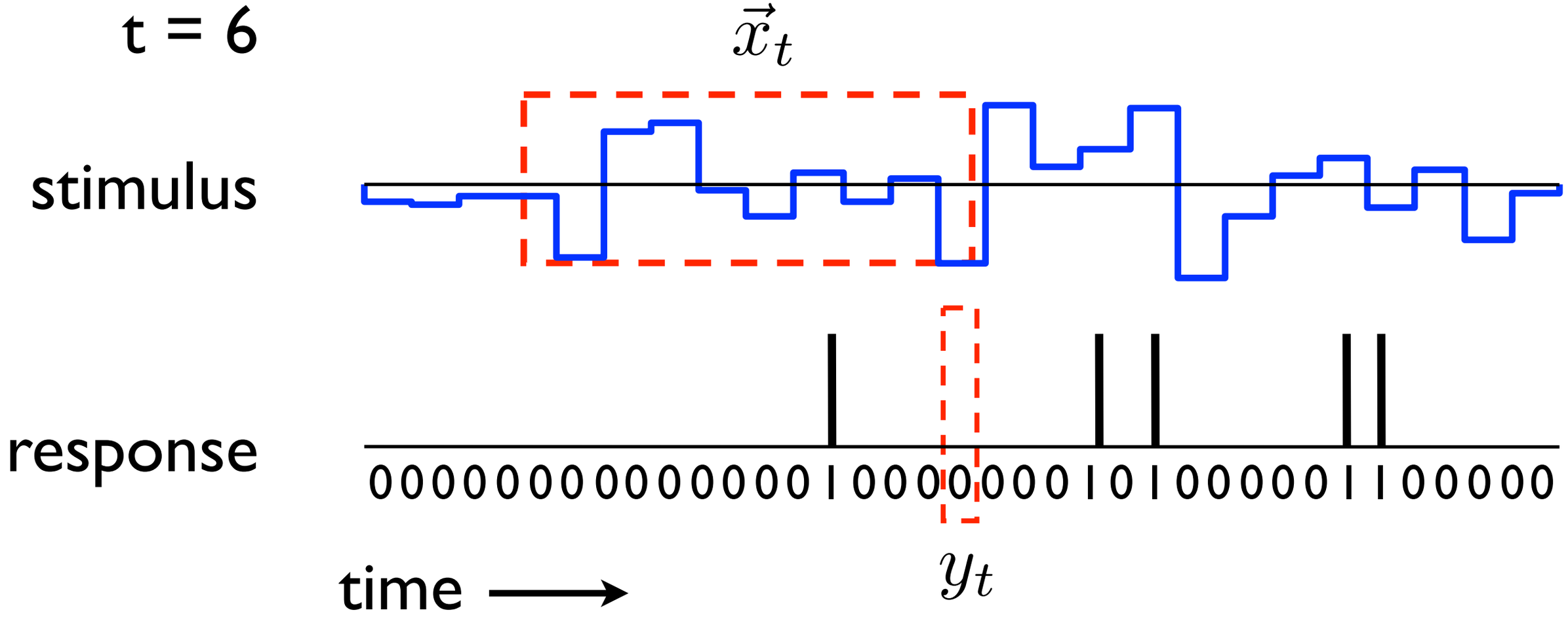
$N(0, \sigma^2)$

linear filter

vector stimulus at time t

walk through the data one time bin at a time

t = 6

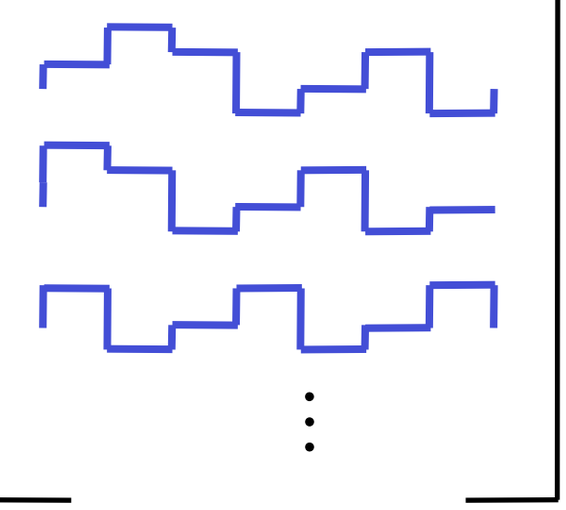


Build up to following matrix version:

$$Y = X\vec{k} + noise$$

time ↓

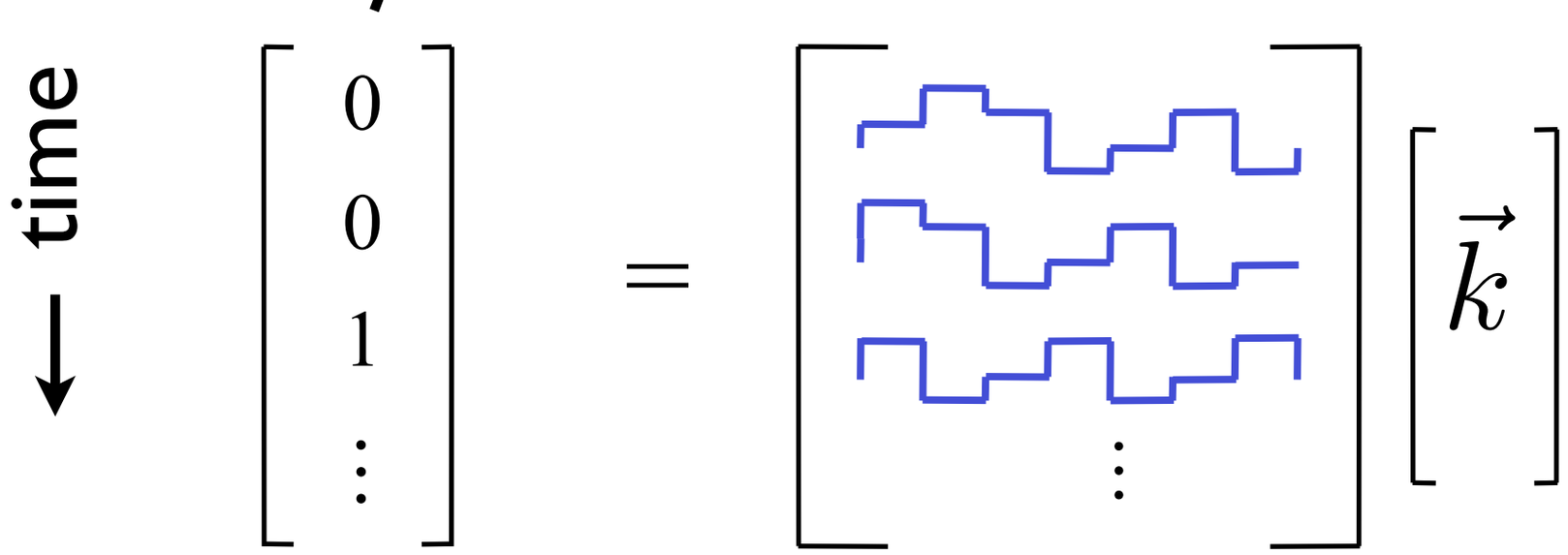
$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \\ & \vdots & \end{bmatrix} \begin{bmatrix} \vec{k} \\ \\ \\ \end{bmatrix}$$



design matrix

Build up to following matrix version:

$$Y = X\vec{k} + noise$$



1. "Linear-Gaussian" GLM: $\hat{k} = (\underbrace{X^T X}_{\text{stimulus covariance}})^{-1} \underbrace{X^T Y}_{\text{spike-triggered avg (STA)}}$

Build up to following matrix version:

$$Y = X\vec{k} + noise$$

time ↓

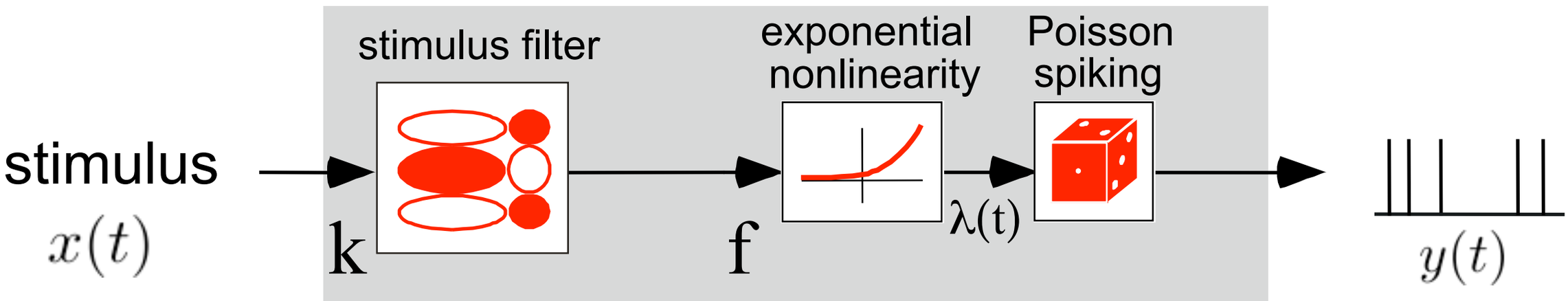
$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \\ & \vdots & \end{bmatrix} \begin{bmatrix} \\ \vec{k} \\ \\ \end{bmatrix}$$

1. "Linear-Gaussian" GLM: $\hat{k} = (X^T X)^{-1} X^T Y$

2. Poisson GLM: `k = glmfit(X,Y,'Poisson');`

maximum likelihood fit (assumes exponential nonlinearity by default)

Linear-Nonlinear-Poisson

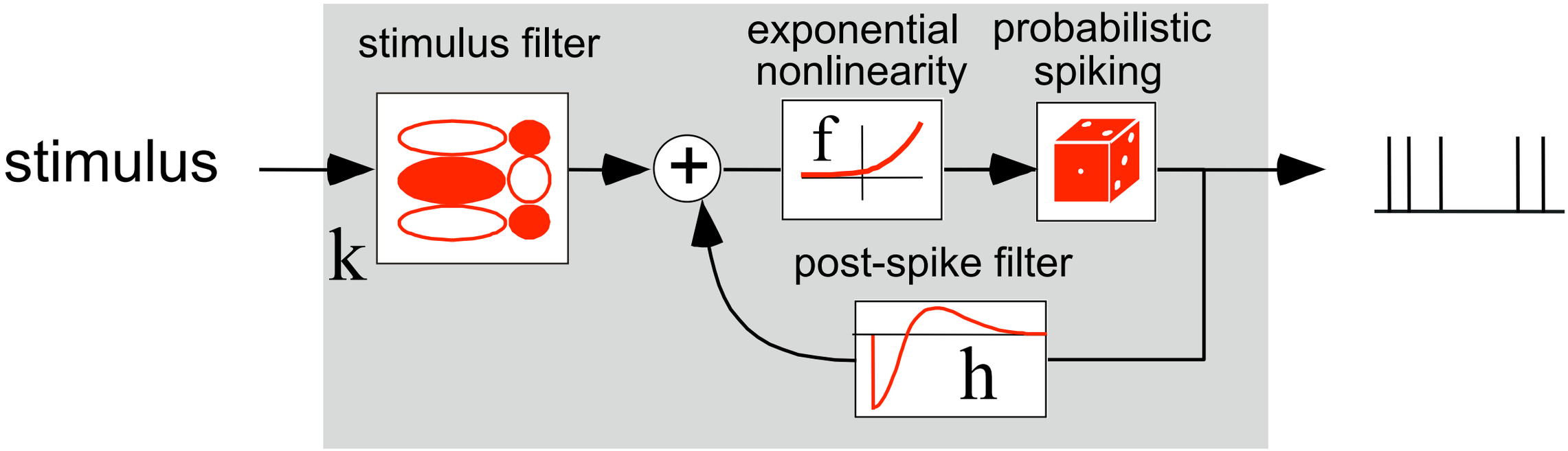


conditional intensity (spike rate) $\lambda(t) = f(k \cdot x(t))$

- output: Poisson process
- problem: assumes spiking depends only on stimulus!

(Truccolo et al 04)

Poisson GLM with spike-history dependence



conditional intensity (spike rate)

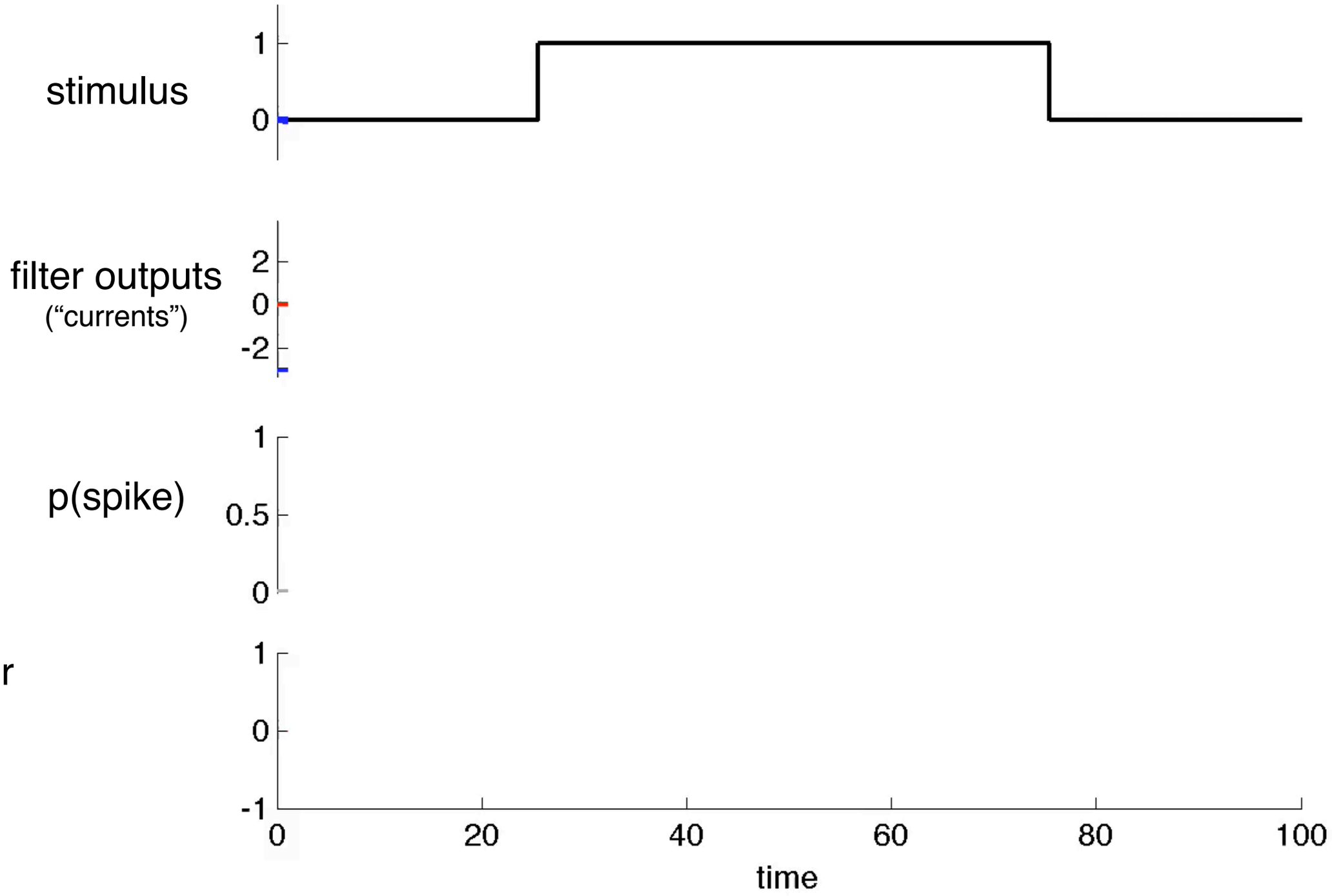
$$\lambda(t) = f(\vec{k} \cdot \vec{x}(t) + \vec{h} \cdot \vec{y}_{hst}(t))$$

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

- output: no longer a Poisson process

GLM dynamic behaviors

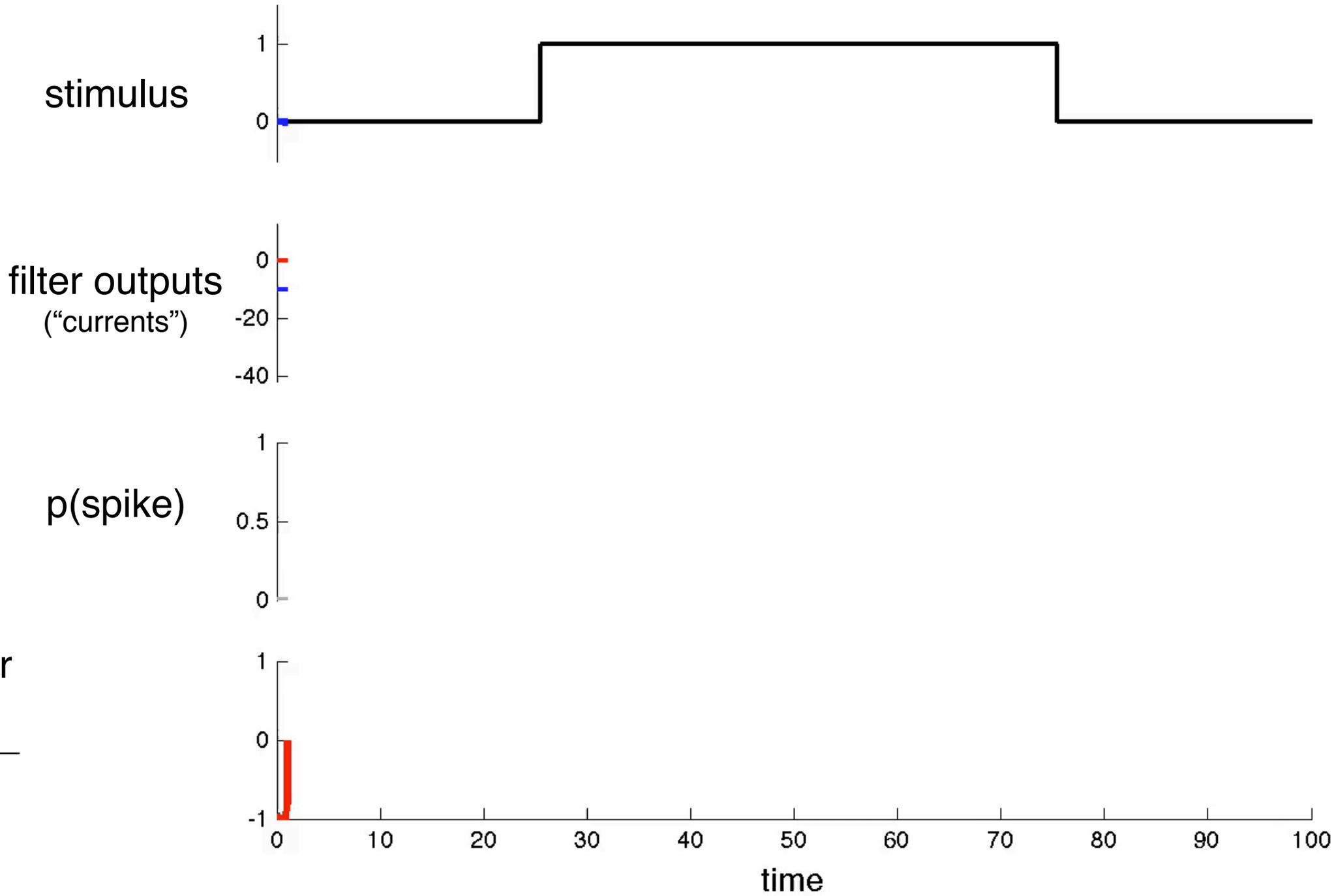
- irregular spiking



GLM dynamic behaviors

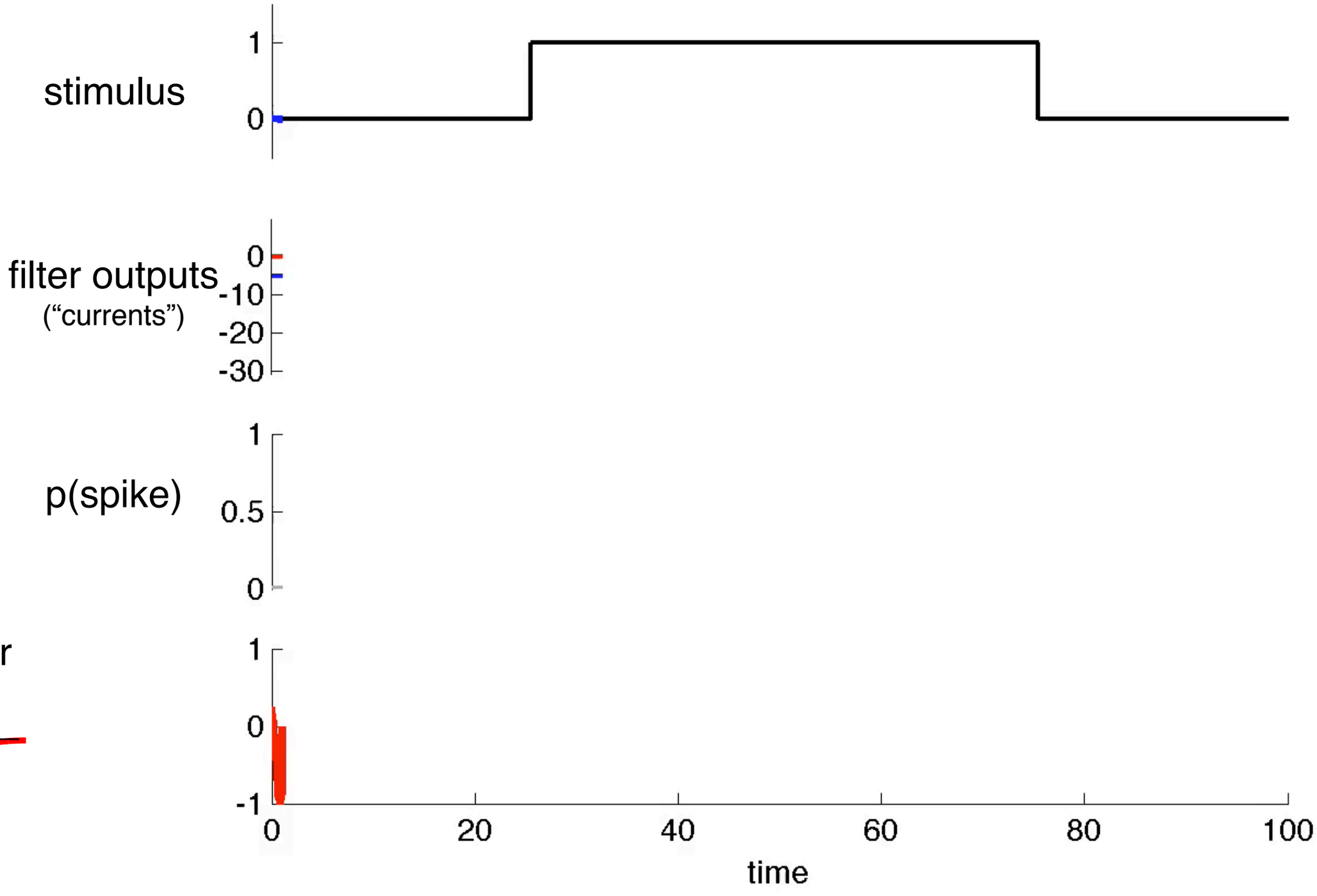
(Weber & Pillow 2016)

- regular spiking



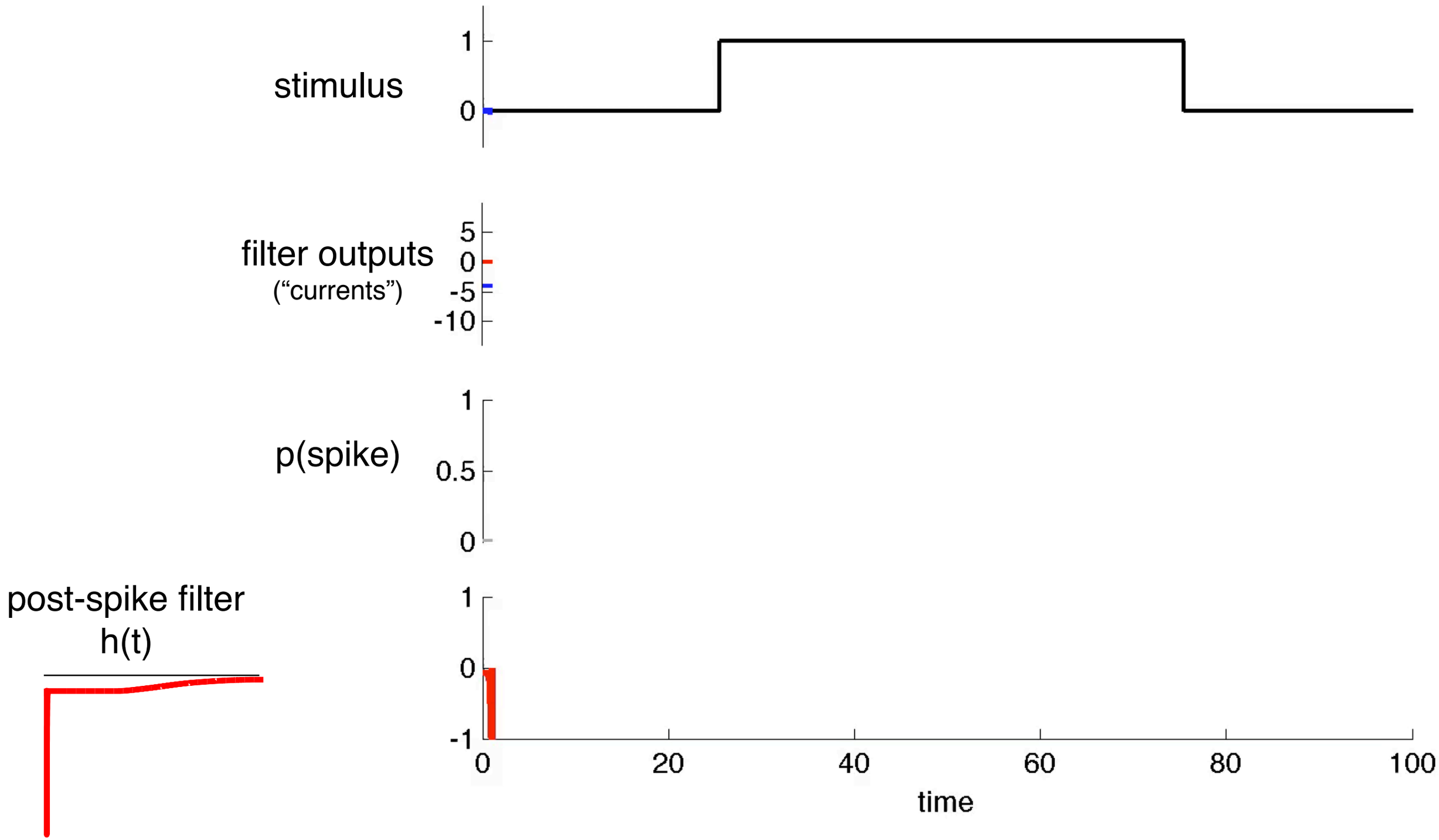
GLM dynamic behaviors

- bursting



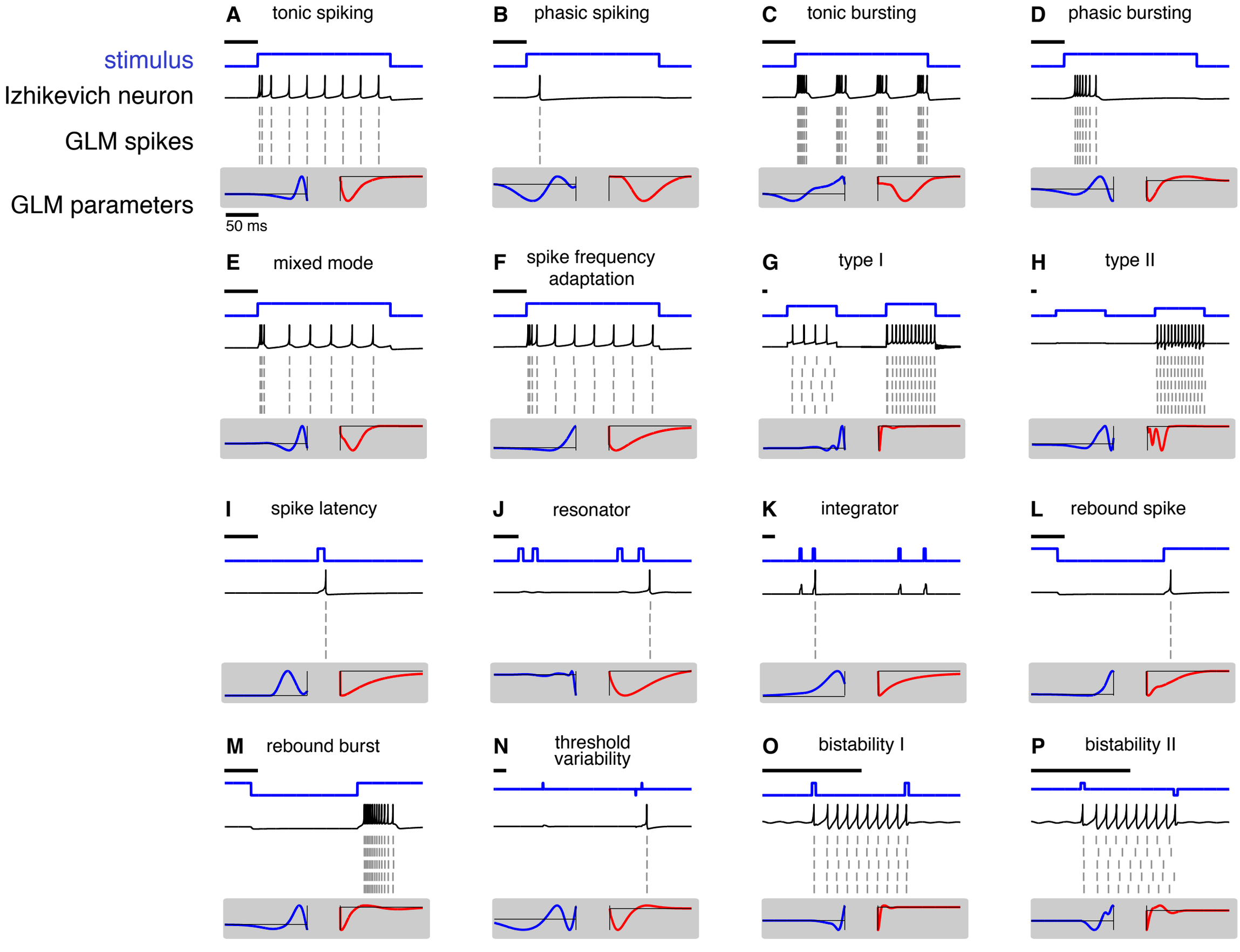
GLM dynamic behaviors

- adaptation

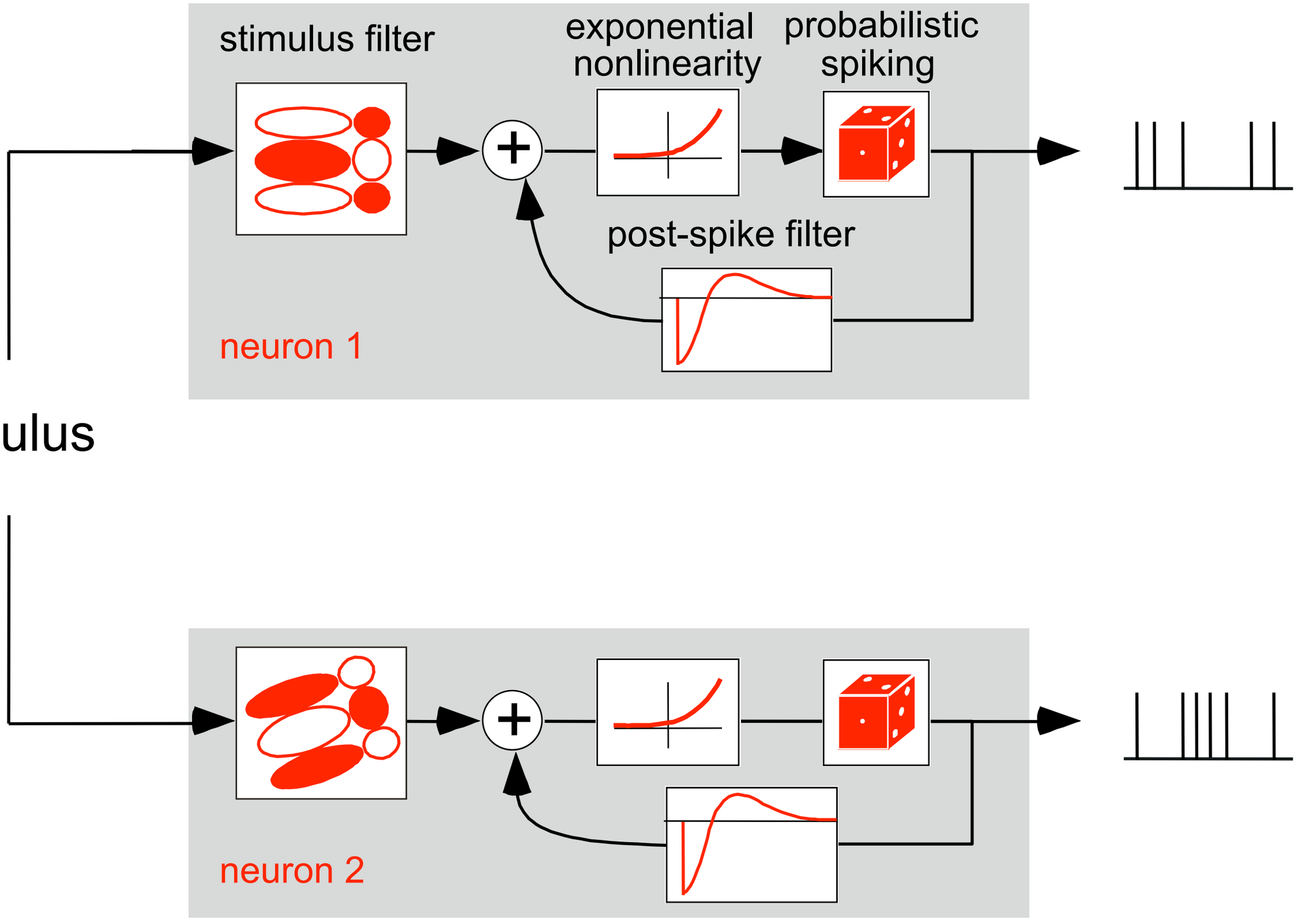


GLM dynamic behaviors (from Izhikevich)

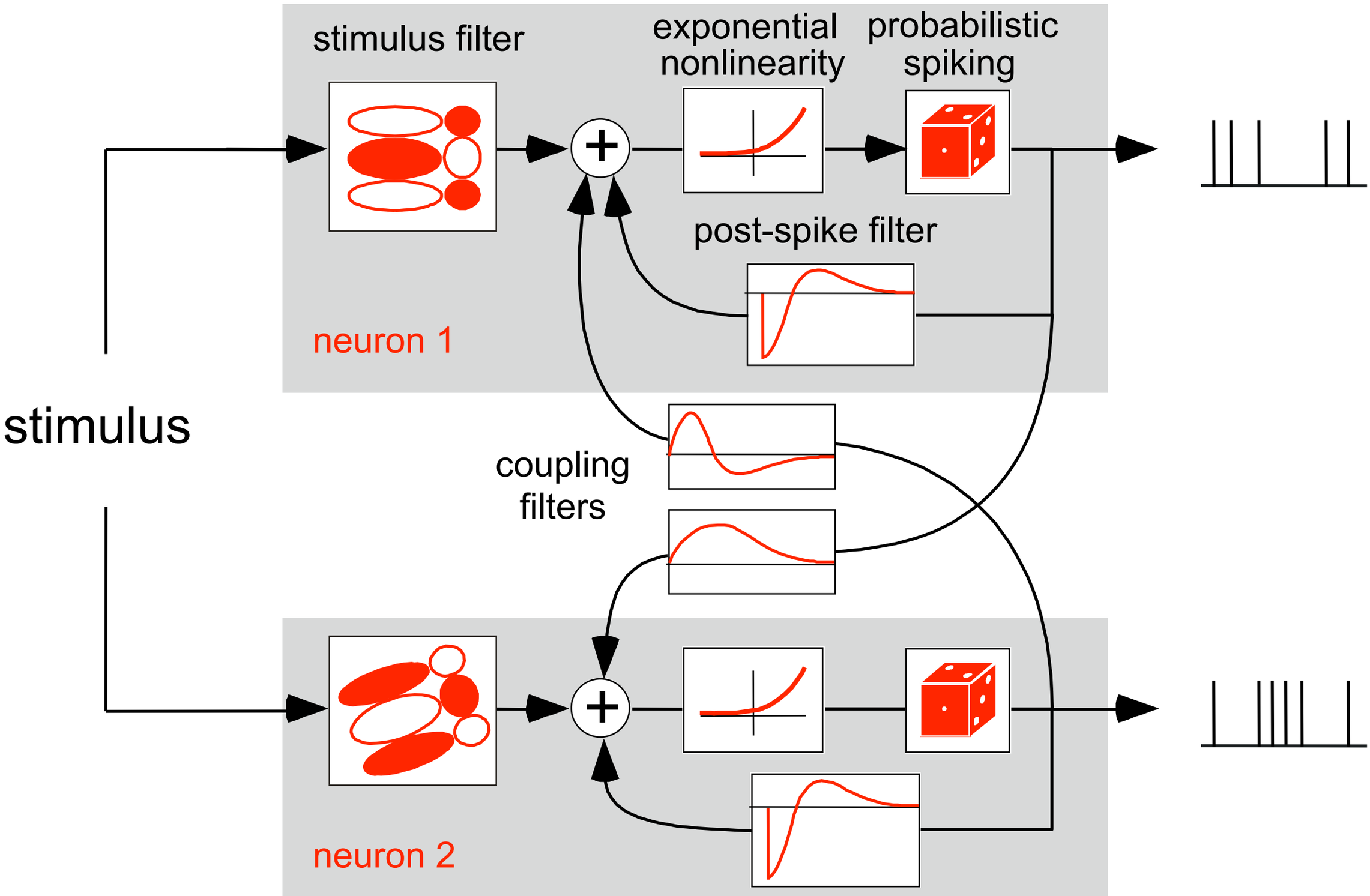
(Weber & Pillow 2017)



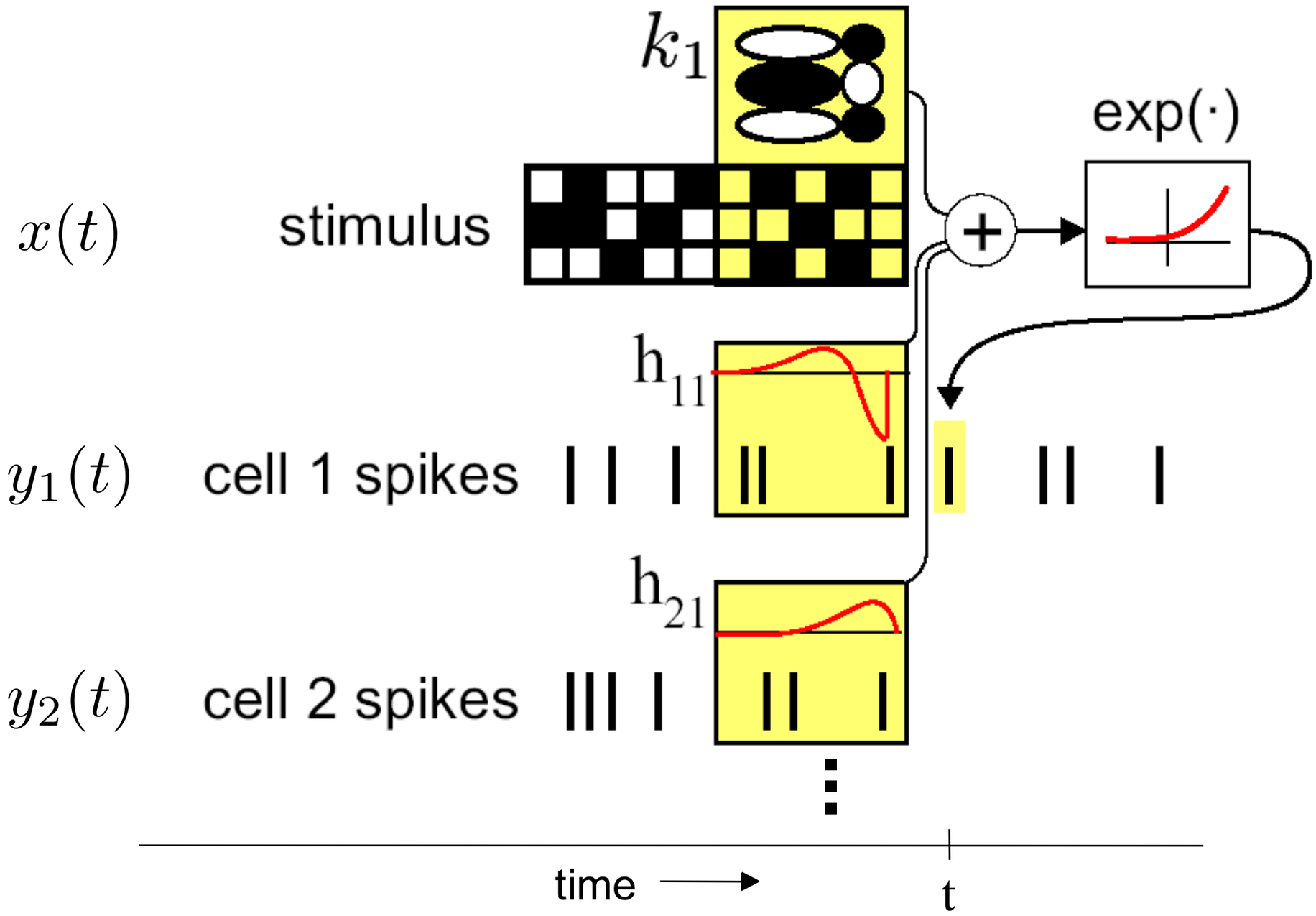
multi-neuron GLM



multi-neuron GLM



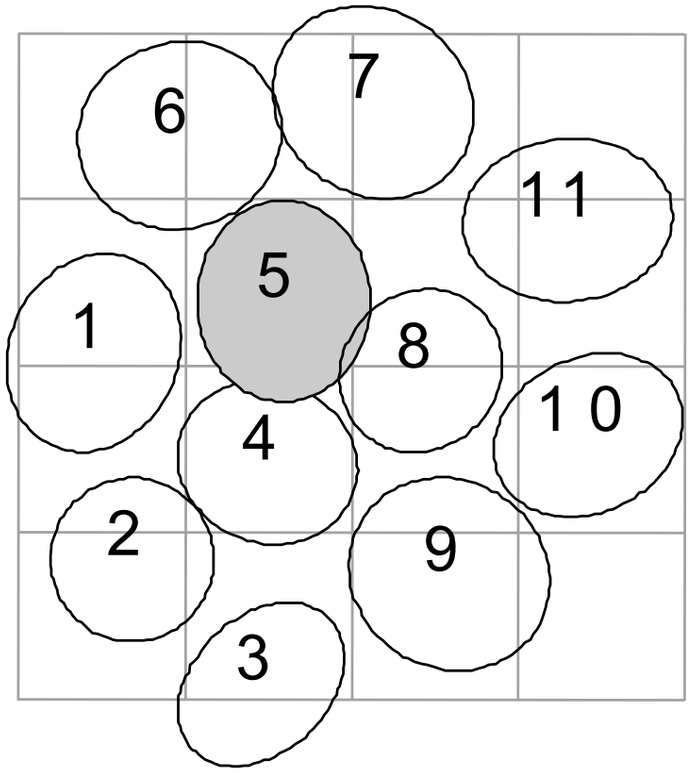
GLM equivalent diagram:

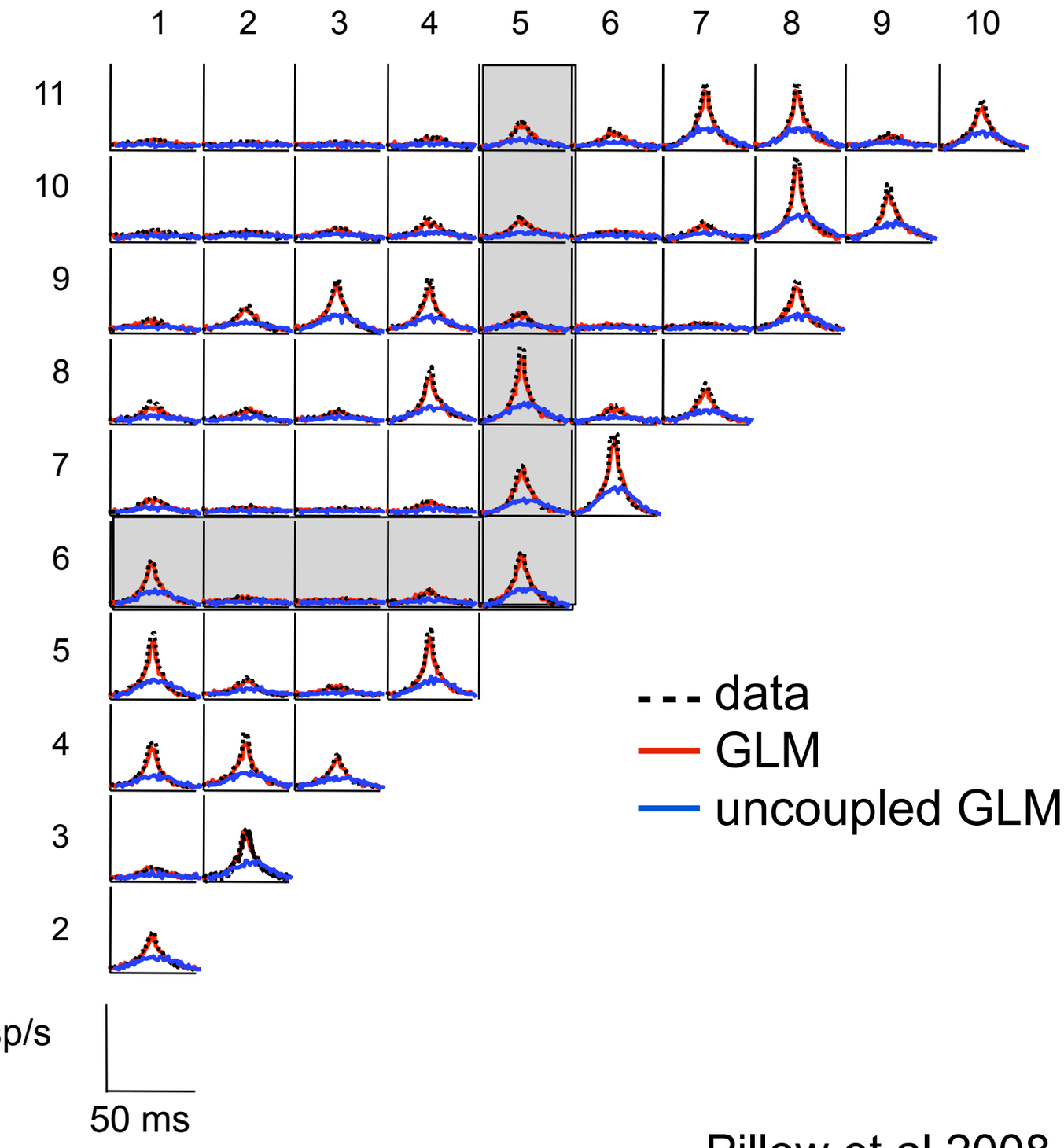


spike rate $\lambda_i(t) = \exp(k_i \cdot x(t) \; + \; \sum_j h_{ij} \cdot y(t))$

modeling correlation structure in neural spike trains

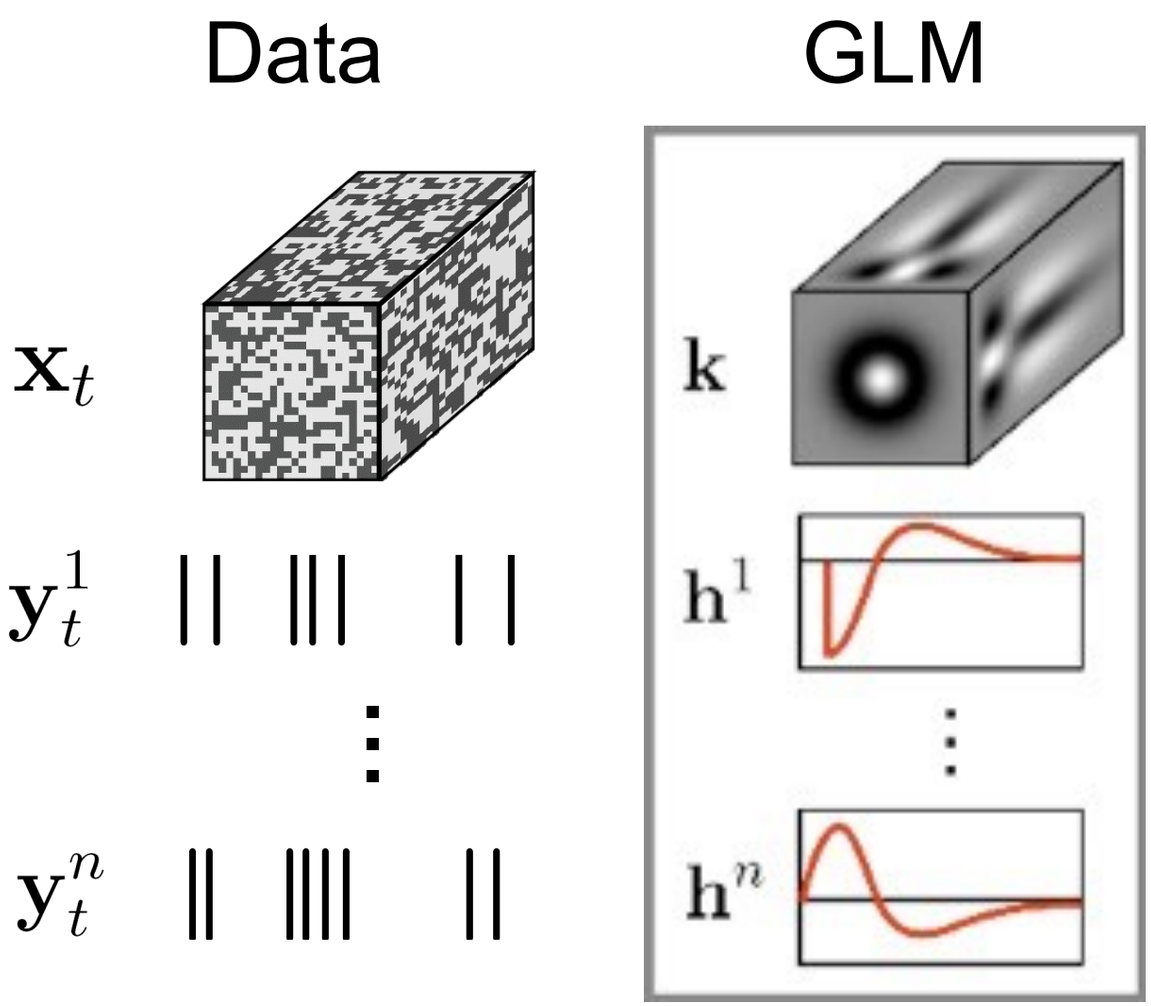
receptive fields





Pillow et al 2008

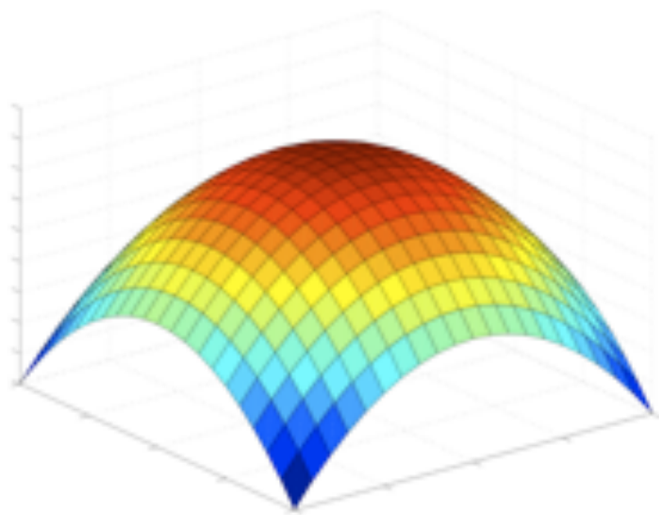
Fitting: Maximum Likelihood



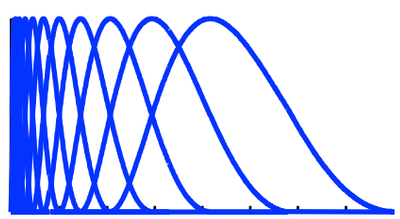
- find filters that maximize the log-conditional probability of the observed data

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

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



- smooth basis for coupling filters

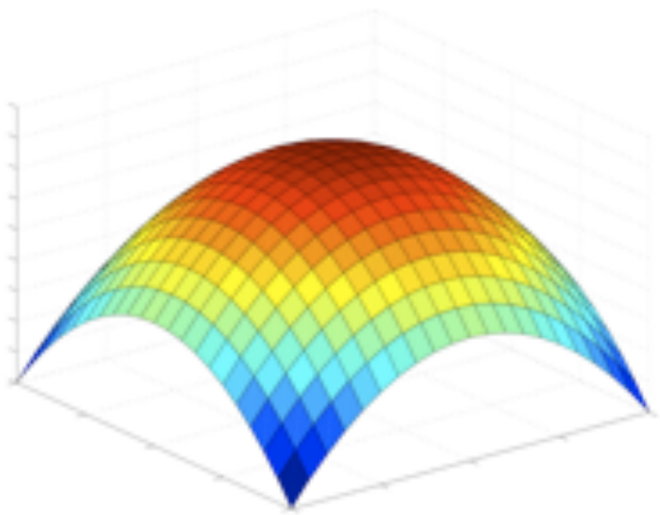


Pillow et al 2008

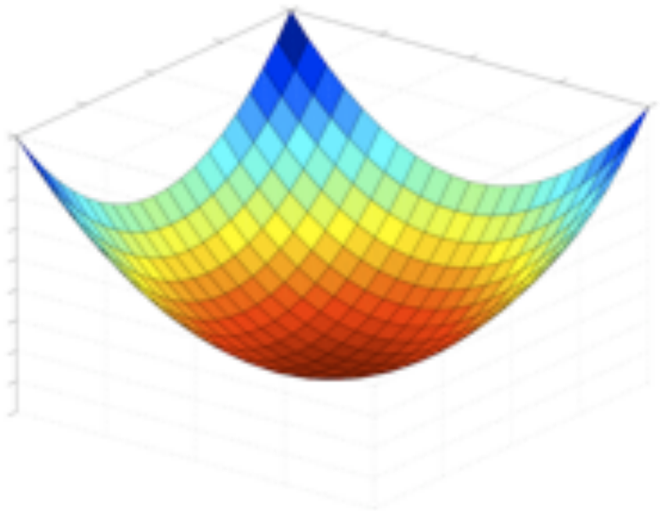
convexity and concavity

- Concavity: having everywhere downward curvature
- Convexity: having everywhere upward curvature
- “f = concave” iff “-f = convex”
- Maximizing concave function = minimizing a convex function
- both preclude (non-global) local optima

concave



convex

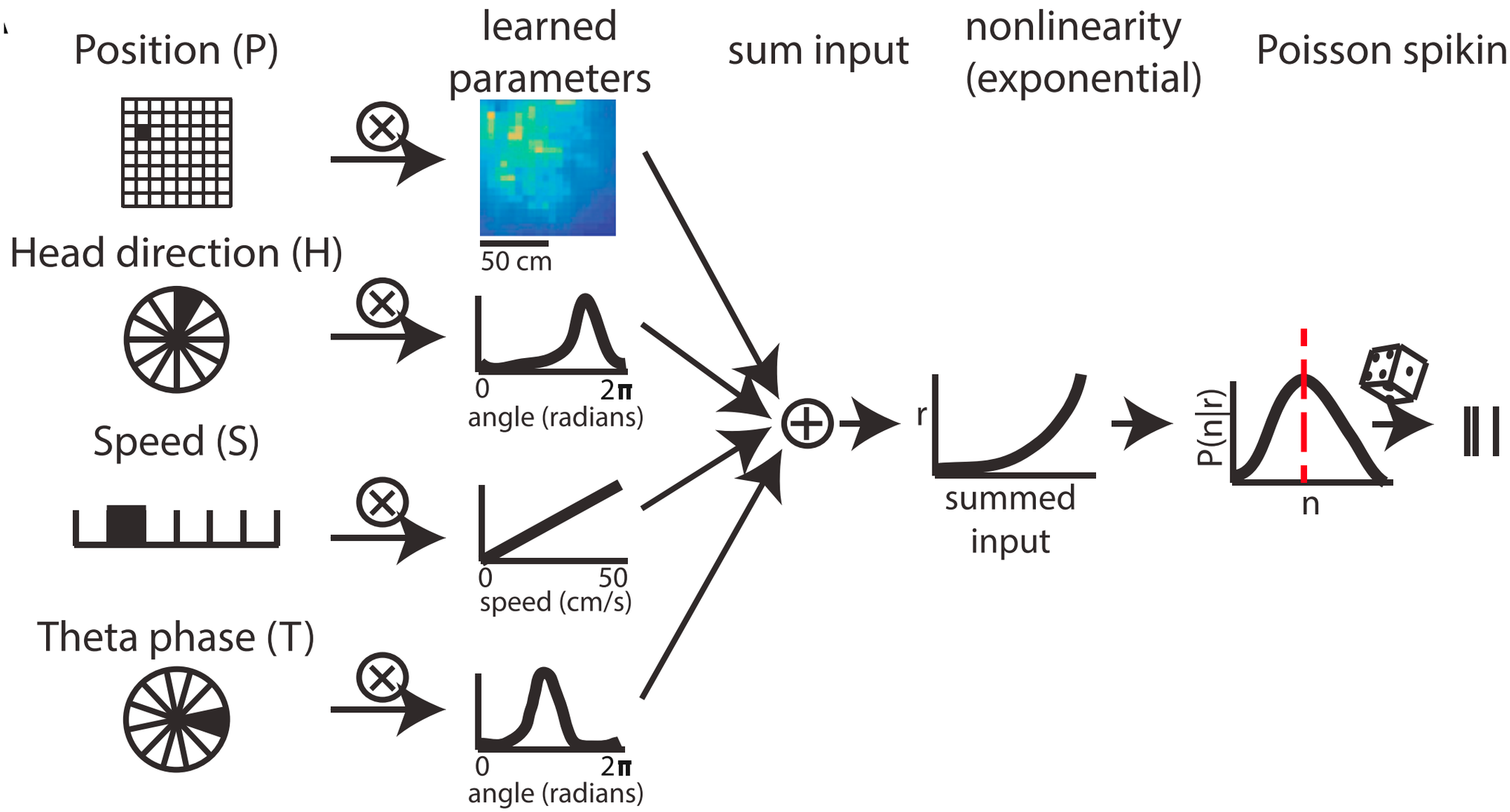


GLM applications & extensions

GLM applications:

Hardcastle, Ganguli & Giocomo 2017

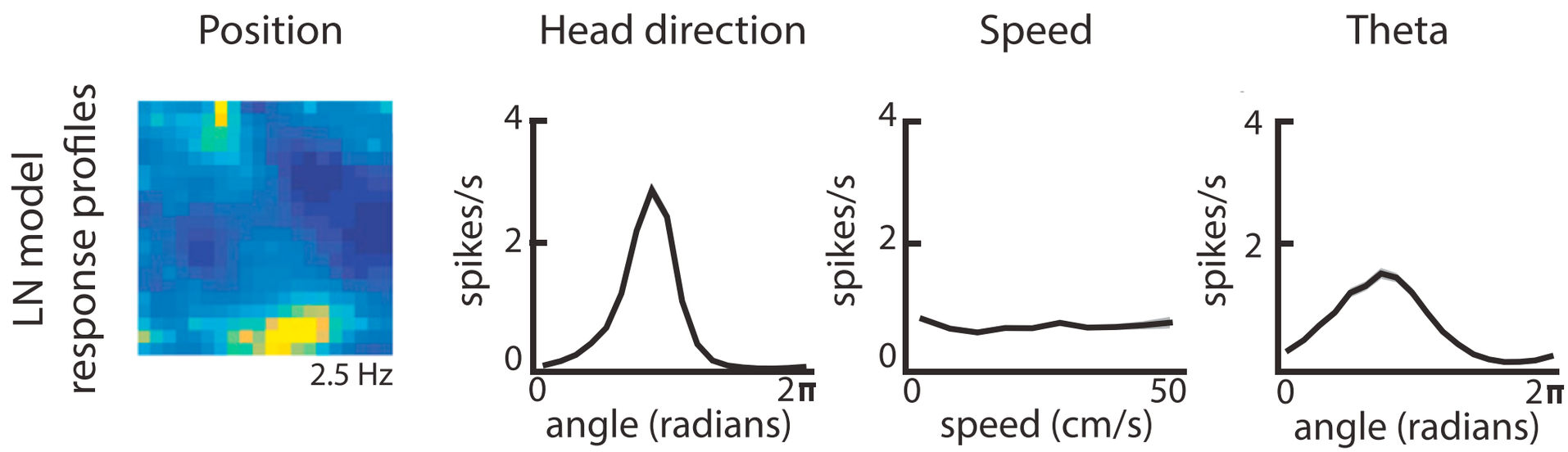
- GLM for coding in entorhinal cortex



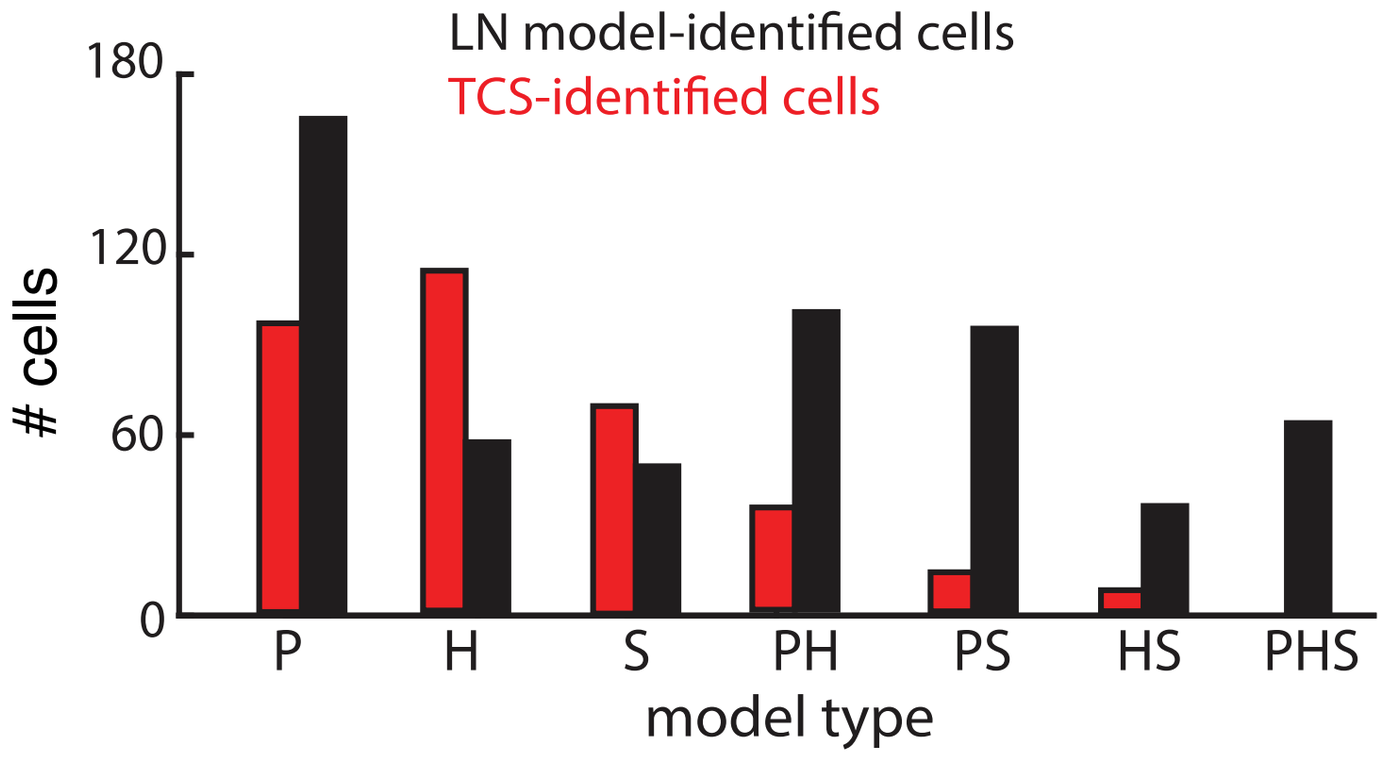
GLM applications:

Hardcastle, Ganguli & Giocomo 2017

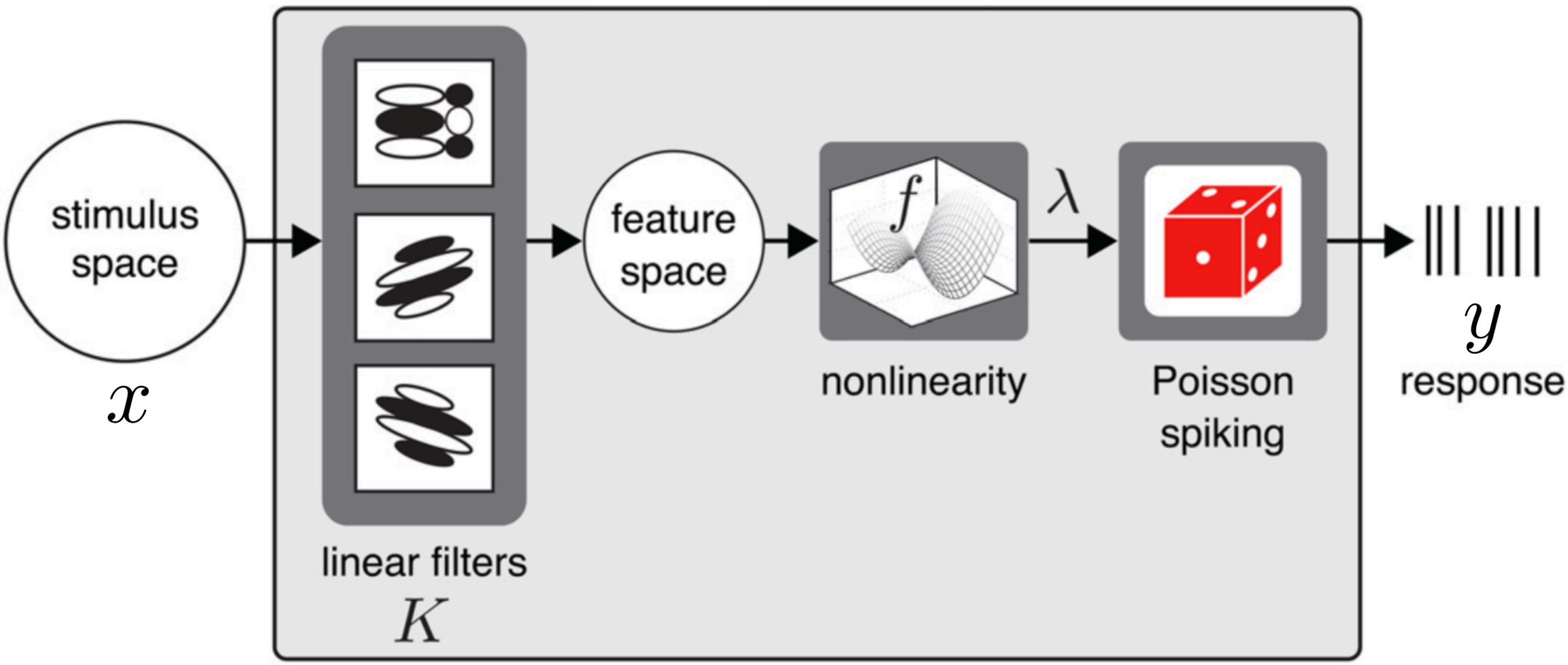
example cell:



population results:



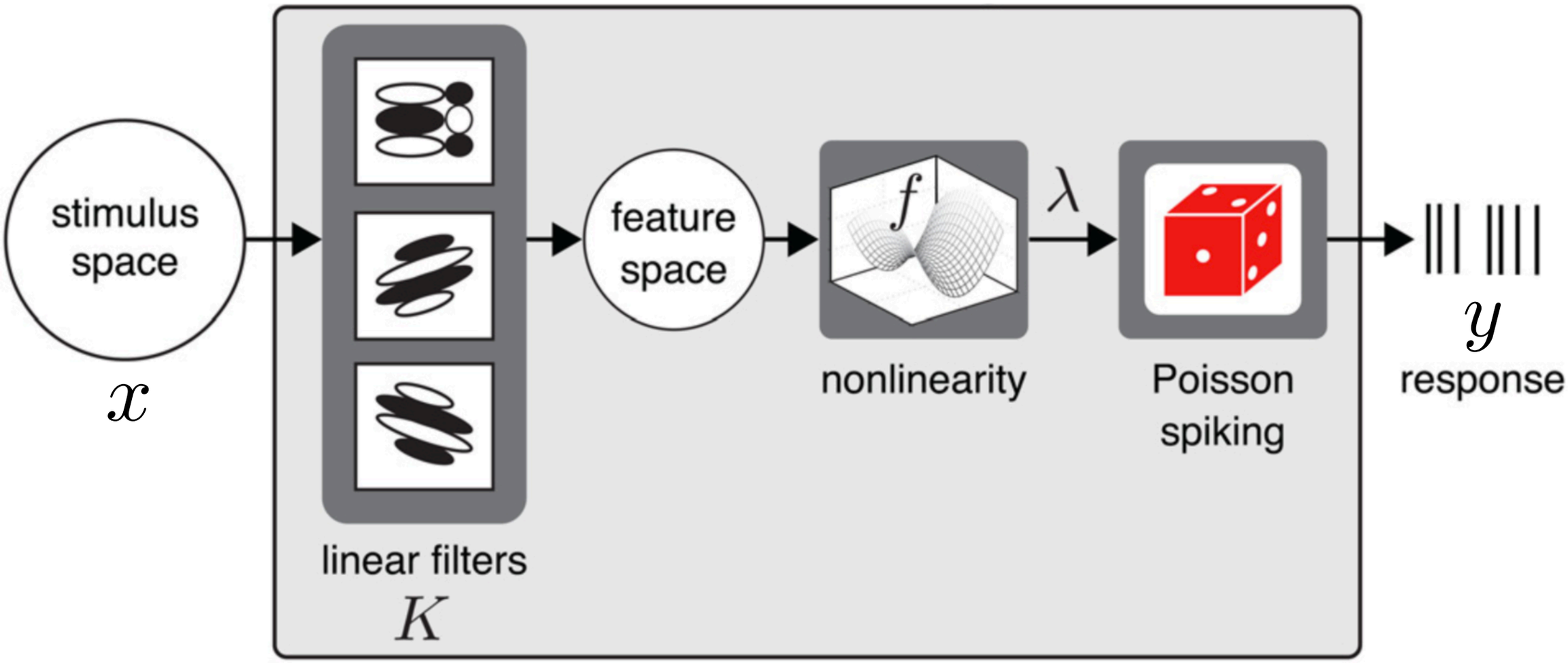
Beyond GLM: multi-filter LNP



- responses may depend on more than one projection of stimulus!
- emphasis on *dimensionality reduction*

- no longer technically a GLM if fitting nonlinearity *f*

Beyond GLM: multi-filter LNP



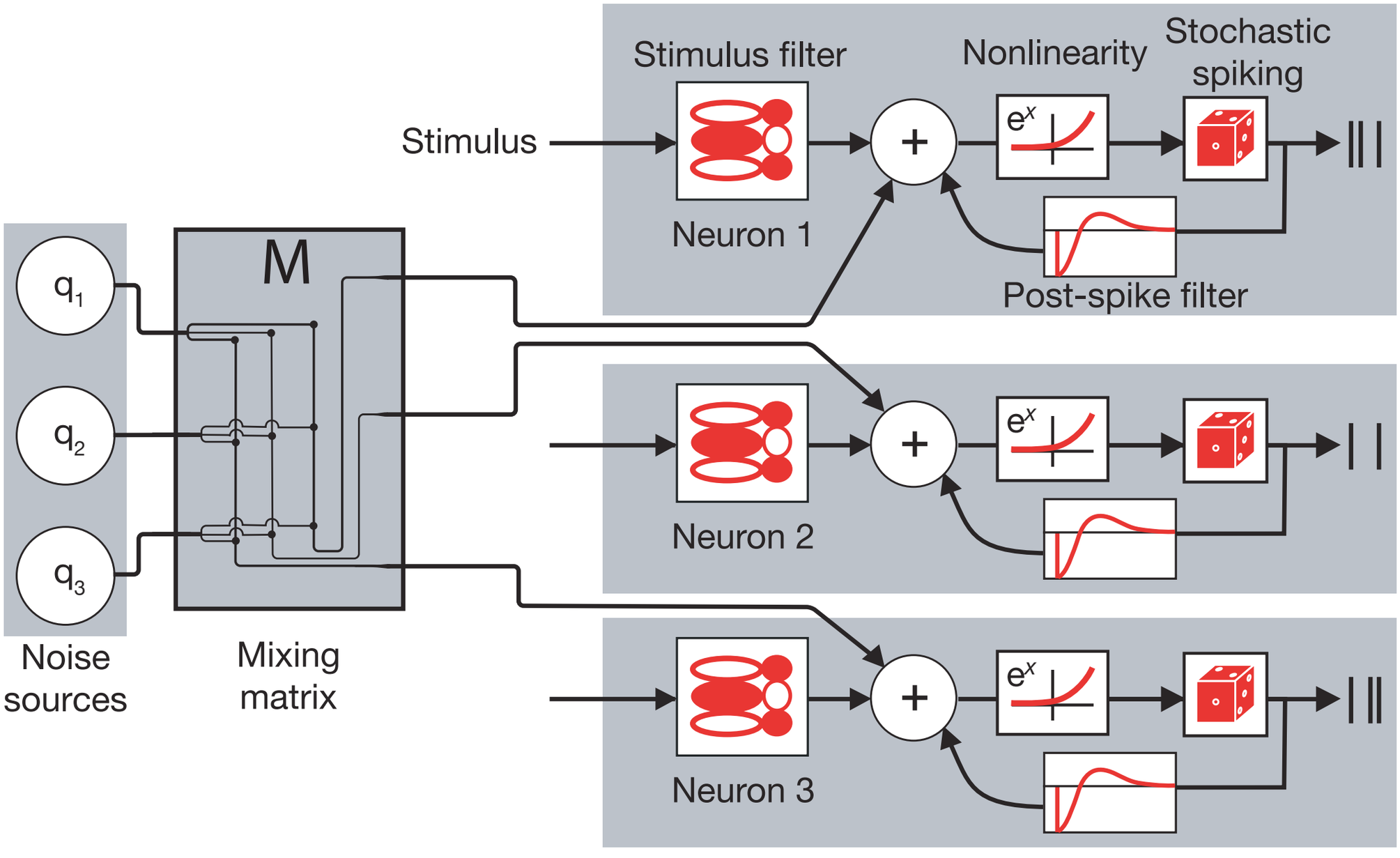
Estimators:

- 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]

“shared noise” (latent variable) models

Vidne et al 2012
Goris, Movshon, & Simoncelli 2014

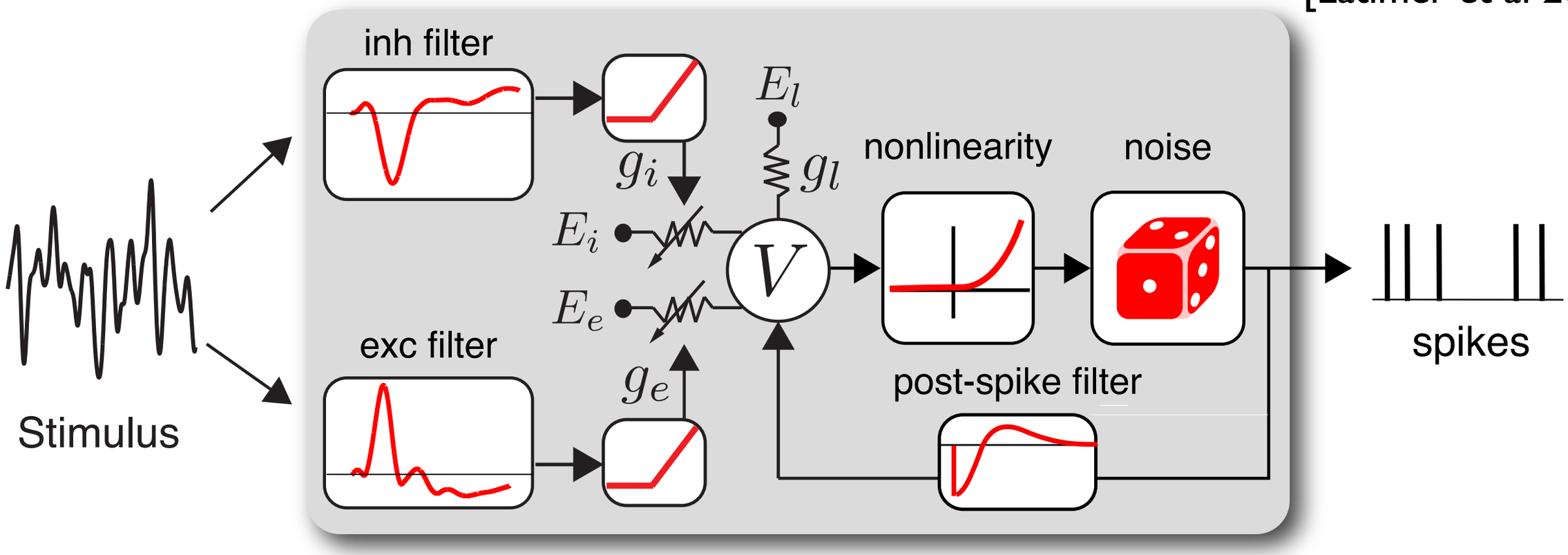


$$\lambda_t^i = \exp(\mu^i + \mathbf{k}^i \cdot \mathbf{x}_t^i + \mathbf{h}^i \cdot \mathbf{y}_t^i + \mathbf{M} \cdot q_t)$$

shared noise

extending GLM to conductance-based model

[Latimer et al 2014]



conductances

$$g_e(t) = f_c(k_e \cdot \mathbf{x}(t))$$
$$g_i(t) = f_c(k_i \cdot \mathbf{x}(t))$$

membrane dynamics

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

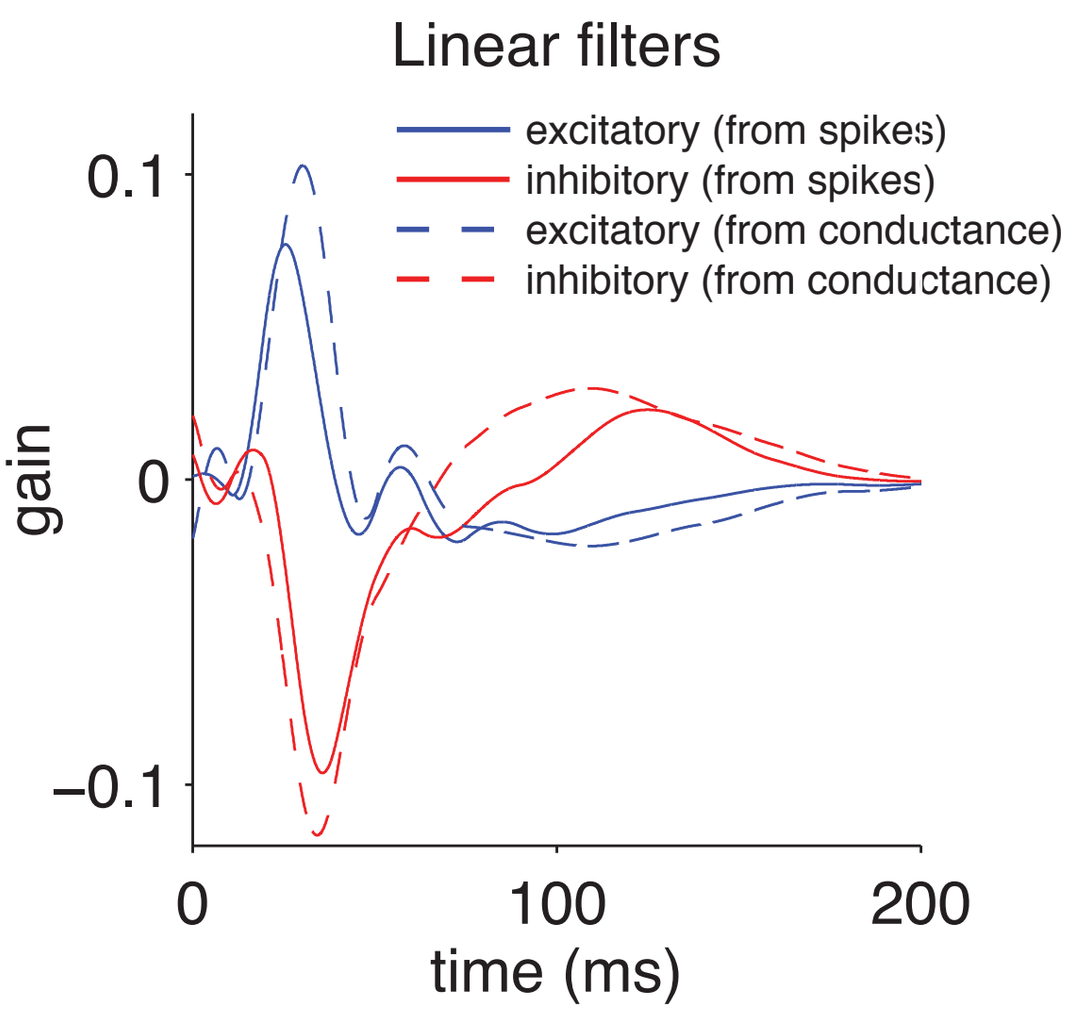
inst. spike rate

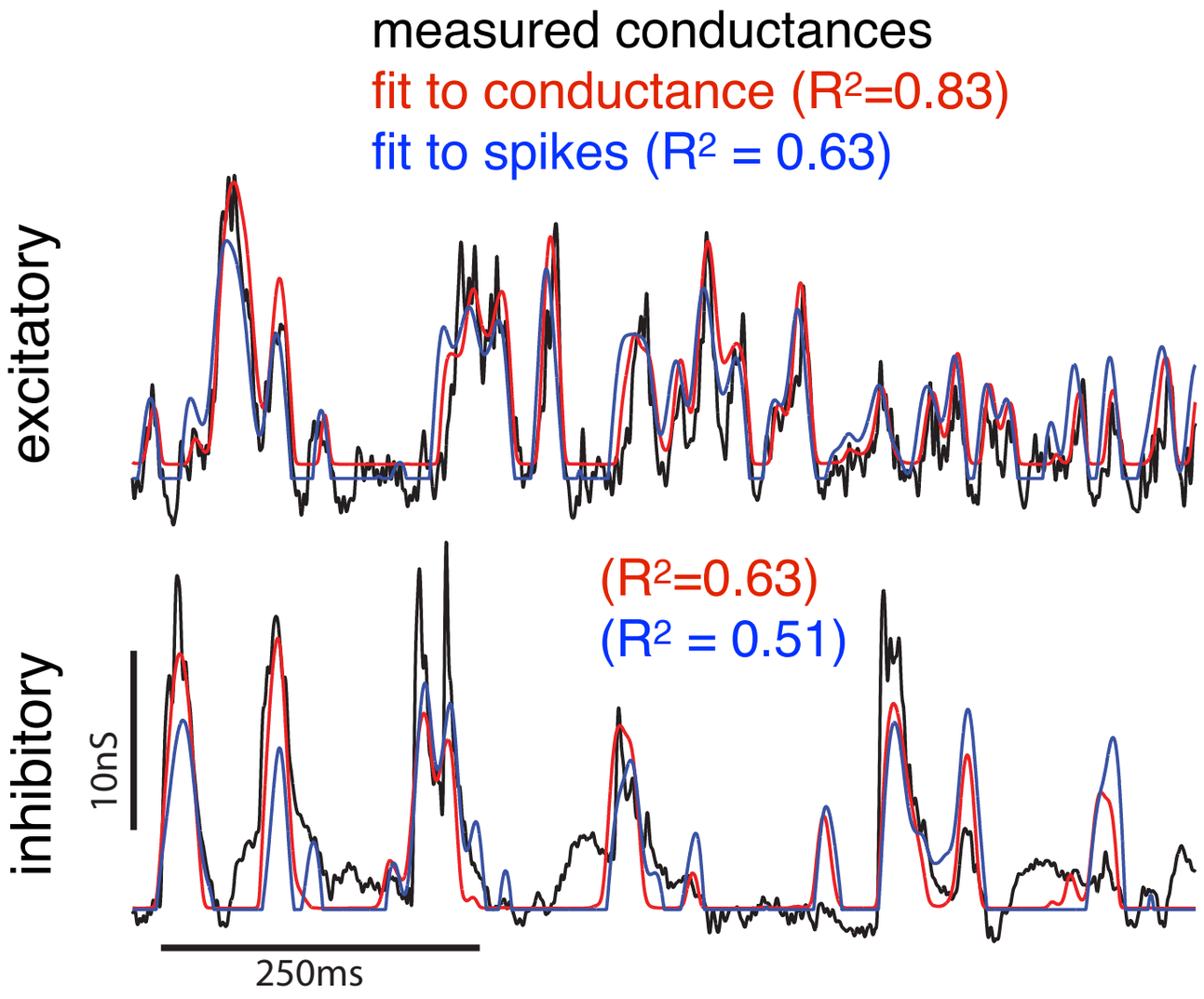
$$\lambda(t) = f(V(t))$$

- shunting inhibition
- adaptive changes in dynamics

extending GLM to conductance-based model

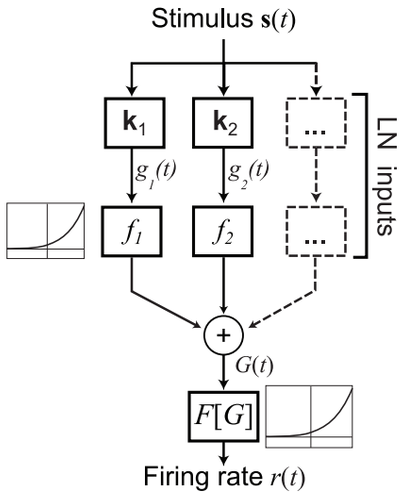
- intracellular recordings in macaque parasol RGCs (Fred Rieke)



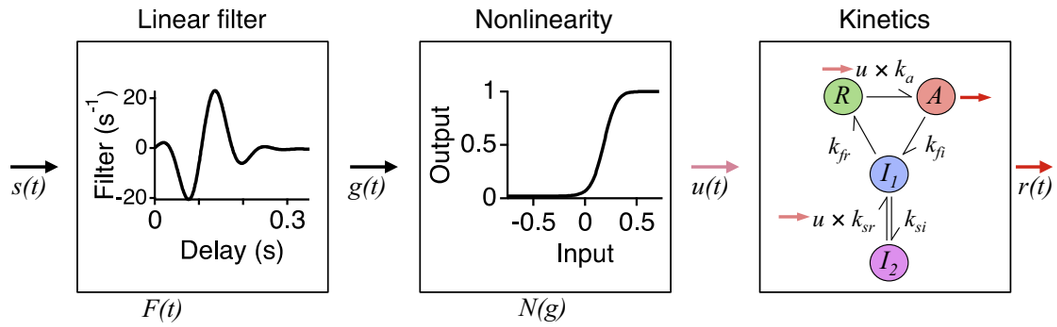


[Latimer et al 2014]

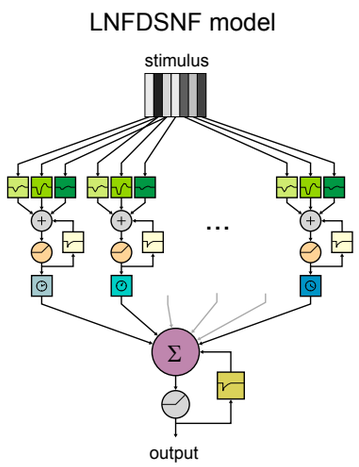
many other biophysically oriented extensions



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



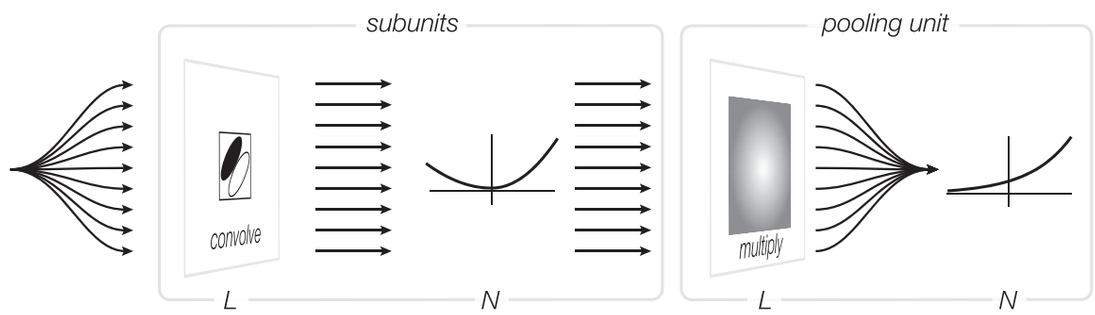
Linear-Nonlinear-Kinetics (LNK)
[Ozuysal & Baccus 2014]



LNFDSNF model

linear-nonlinear-feedback-delayed-sum-nonlinear-feedback

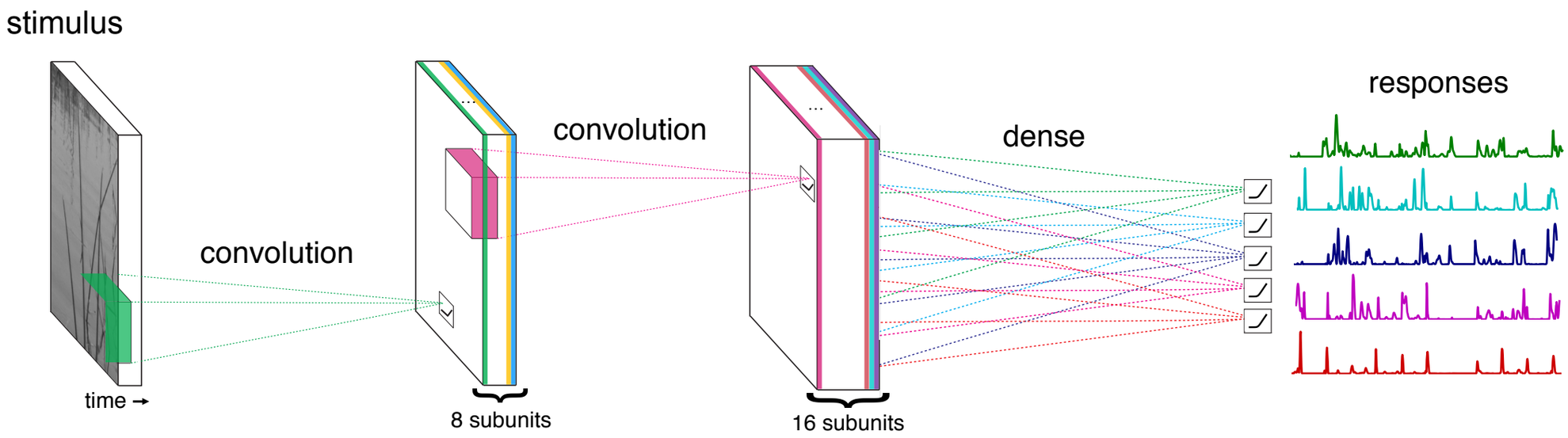
[Real, Asari, Gollisch & Meister 2017]



convolutional subunit model

[Vintch, Movshon & Simoncelli 2015,
Wu, Park & Pillow 2014]

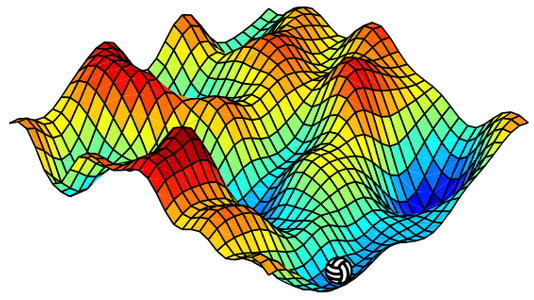
deep learning / deep neural networks (DNNs)



[Yamins et al 2014, Mcintosh et al 2016, Maheswaranathan et al 2017, Benjamin et al 2017, …]

If you understand GLMs… you understand DNNs!

- stack many LNs on top of each other: 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!



[credit: Jakob Macke]

Conclusions

- GLM: flexible encoding model for single & multi-neuron resps
- captures stimulus-dependence & correlations
  (caveat: for white noise stimuli!)
- Bayesian decoding: use encoding model to do "ideal observer" readout of spikes, role of correlations, timing, etc.

Big Picture

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

GLM summary

- linear ("predictor") + nonlinear ("link") + noise ("distribution")
- incorporate spike-history via "history" filter
- rich (non-Poisson) dynamical properties: refractoriness, bursting, adaptation, (others?)
- incorporate correlations between neurons via "coupling" filters
- log-concave likelihood function (for suitable f)
  $\Rightarrow$ tractable for fitting, even in high-dimensional parameter space

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!

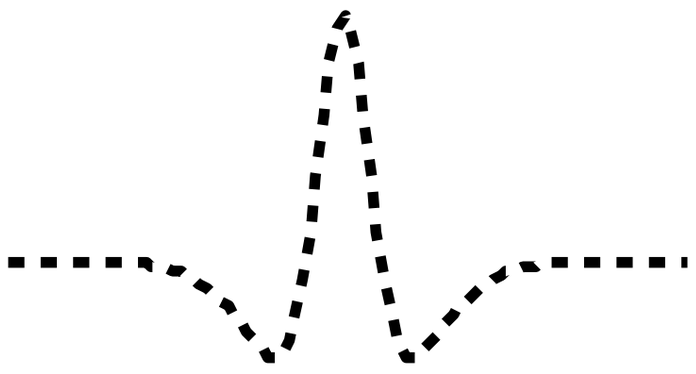
Regularization

Problem with maximum likelihood

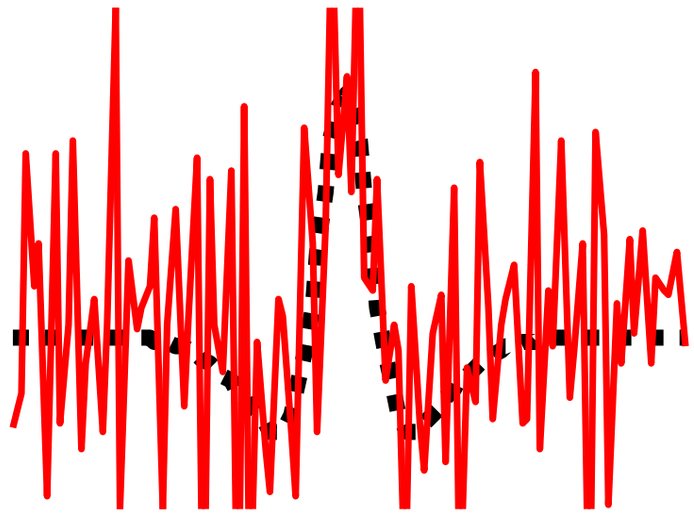
Simulated Example

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

true **w**



maximum likelihood



maximize

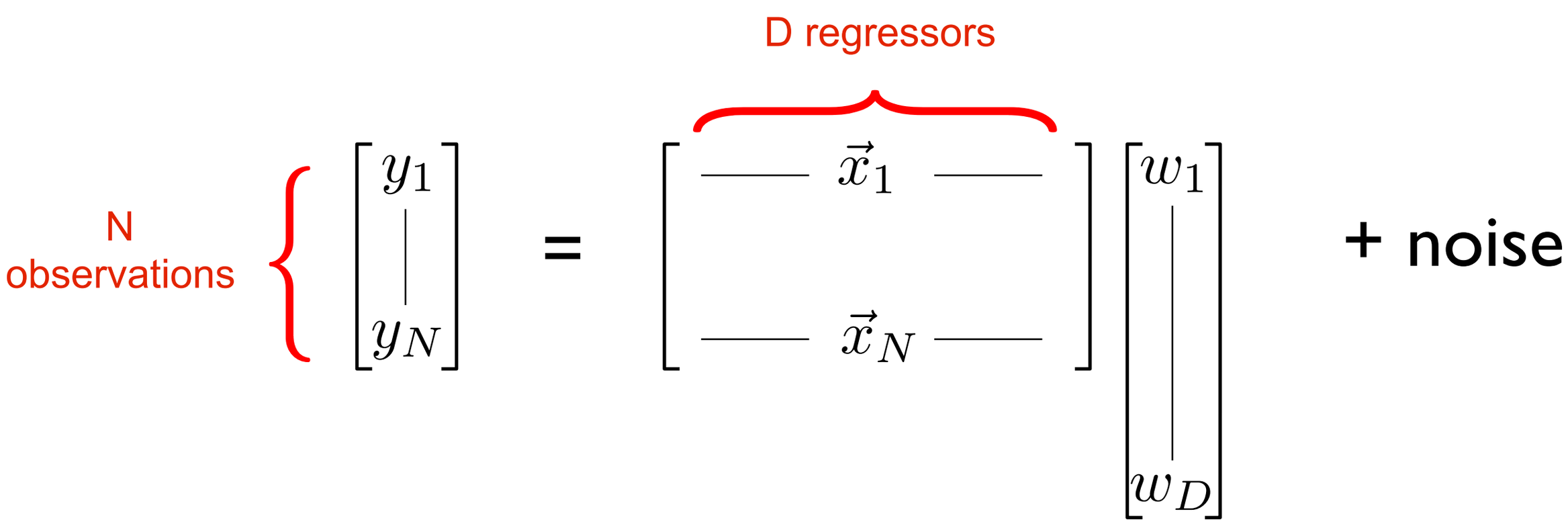
$$\log p(data|\mathbf{w})$$

- too many parameters + not enough data: "overfitting"

  parameters fit details in the training data that are NOT useful for predicting new data

Modern statistics

- more dimensions than samples $D \geq N$



- fewer equations than unknowns!
- no unique solution (ML estimate doesn't exist!)

Solution #1: ridge regression / Tikhonov regression / L2 penalization / L2 "shrinkage"

ridge parameter

$$\textit{minimize} \quad \underbrace{||Y - X\mathbf{w}||_2^2}_{\text{sum of squared errors}} + \lambda \underbrace{\sum_i w_i^2}_{\substack{\text{sum of squared} \\ \text{coefficients}}} = ||\mathbf{w}||_2^2$$

(aka squared L2 norm of **w**)

$\lambda \cdot$ "penalty"

Solution 1: ridge regression / Tikhonov regression / L2 penalization / L2 "shrinkage"

$$\textit{minimize} \quad ||Y - X\mathbf{w}||_2^2 + \lambda \sum_i w_i^2$$

Bayesian interpretation: *maximum a posteriori (MAP)* estimate

$$\hat{\mathbf{w}}_{MAP} = \arg\max_{\mathbf{w}} P(\mathbf{w}|data)$$

$$= \arg\max_{\mathbf{w}} \frac{P(data|\mathbf{w})P(\mathbf{w})}{P(data)}$$

$$= \arg\max_{\mathbf{w}} \underbrace{\log P(data|\mathbf{w})}_{\text{Gaussian likelihood}} + \underbrace{\log P(\mathbf{w})}_{\text{Gaussian prior}}$$

$$\mathcal{N}(y_i|\mathbf{x}_i^T\mathbf{w}, \sigma_n^2) \qquad \mathcal{N}(w_i; 0, \sigma_p^2)$$

Let's revisit the case of scalar parameter *w*:

likelihood: $$P(Y|X,w) = \prod_i \frac{1}{2\pi\sigma_n^2} \exp\left(-\frac{(y_i - x_i w)^2}{2\sigma_n^2}\right)$$

prior: $$P(w) = \frac{1}{2\pi\sigma_p^2} \exp\left(-\frac{w^2}{2\sigma_p^2}\right)$$

(1) Derive MAP estimator, i.e. maximum of $P(w|X,Y)$

(2) What is the ridge penalty λ in terms of σ_n^2 and σ_p^2 ?

recalling: $$\hat{w}_{map} = \arg\min_w ||Y - X\mathbf{w}||_2^2 + \lambda w^2$$

"Penalized log-likelihood" view:

$$\textit{minimize} \quad \underbrace{-\log P(data|\mathbf{w})}_{\text{negative log-likelihood}} + \underbrace{\lambda \sum_i w_i^2}_{\text{penalty}}$$

Bayesian view (MAP inference):

$$\textit{maximize} \quad \underbrace{\log P(data|\mathbf{w})}_{\text{log-likelihood}} + \underbrace{\log P(\mathbf{w})}_{\text{log-prior}}$$

(if we remove the prior, we're back to maximum likelihood / no regularization)

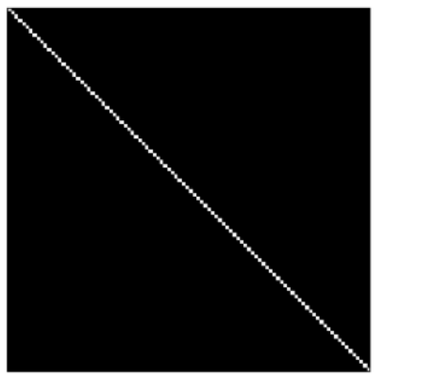
why is it called ridge regression?

$$p(w) = \mathcal{N}(0, C)$$

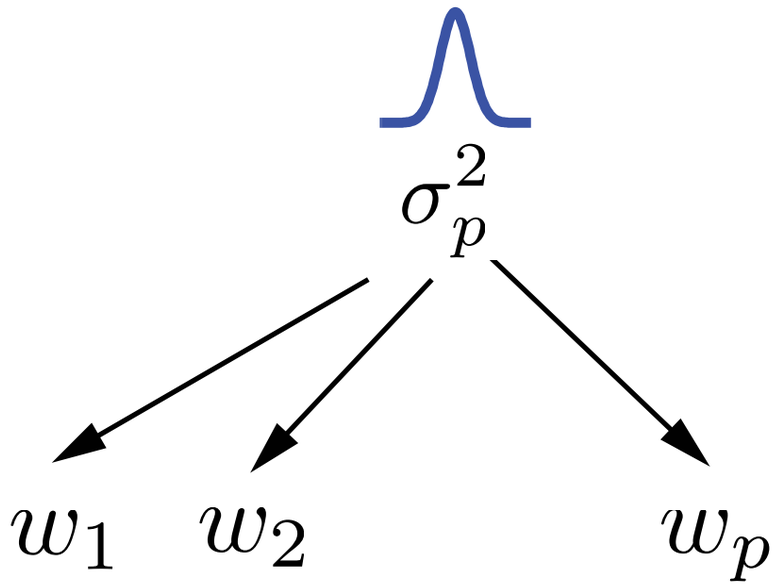
ridge controls
how ***small***
elements of w are

prior covariance

$$C = \sigma_p^2 I$$



ridge



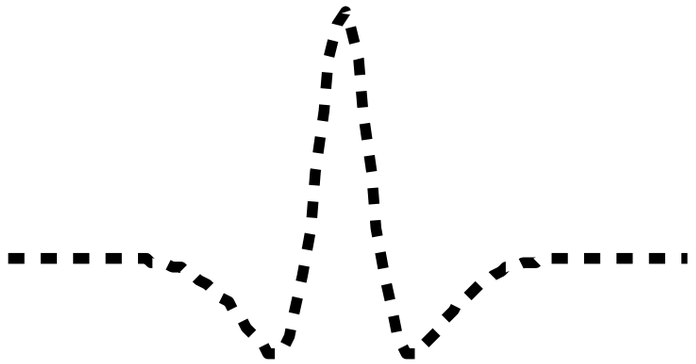
ridge regression
estimate

$$(X^\top X + \sigma^2 C^{-1})^{-1} X^\top Y$$

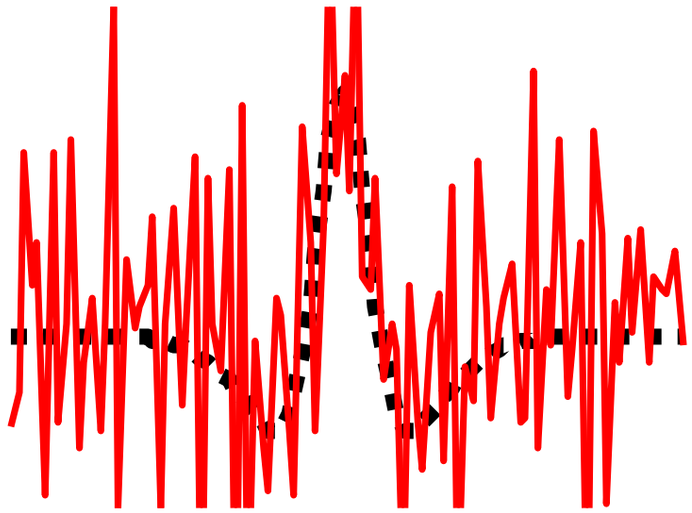
Simulated Example: ridge regression

$$\hat{\mathbf{w}} = (X^\top X + \lambda I)^{-1} X^\top Y$$

true **w**



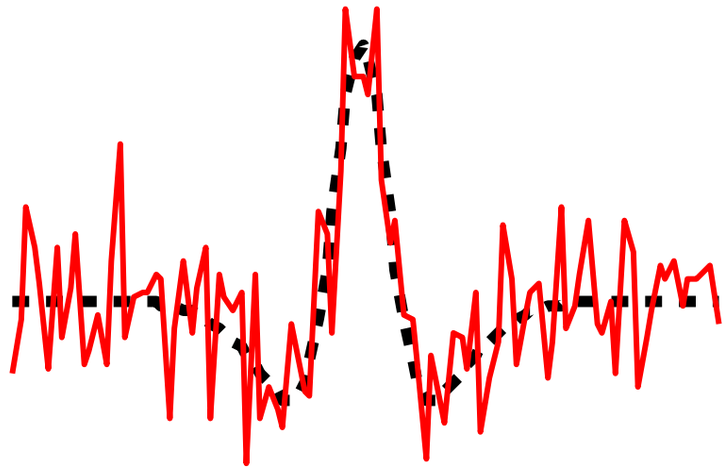
maximum likelihood



maximize

$$\log p(data|\mathbf{w})$$

"ridge regression"



maximize

$$\log p(data|\mathbf{w}) - \underbrace{\lambda \sum w_i^2}$$

penalty on big weights

Solution #2: Lasso / L1 penalization / L1 "shrinkage"

J. R. Statist. Soc. B (1996)
58, *No.* 1, *pp.* 267–288

Regression Shrinkage and Selection via the Lasso

By ROBERT TIBSHIRANI†

University of Toronto, Canada

[Received January 1994. Revised January 1995]

★ ❞ Cited by 28864

by comparison:

Tensorflow: A system for large-scale machine learning
M Abadi, P Barham, J Chen, Z Chen, A Davis… - … on Operating Systems …, 2016 - usenix.org
TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. **Tensor-Flow** uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many …
☆ ❞ Cited by 4562 Related articles All 21 versions Import into BibTeX »

ridge parameter

ridge regression: *minimize* $||Y - X\mathbf{w}||_2^2 + \lambda \sum_i w_i^2$

lasso parameter

lasso: *minimize* $||Y - X\mathbf{w}||_2^2 + \lambda \underbrace{\sum_i |w_i|}$

sum of abs values of coefficients $= ||\mathbf{w}||_1$

(aka L1 norm of **w**)

Lasso ("least absolute shrinkage and selection operator")

ridge parameter

ridge regression: *minimize* $||Y - X\mathbf{w}||_2^2 + \lambda \sum_i w_i^2$

lasso parameter

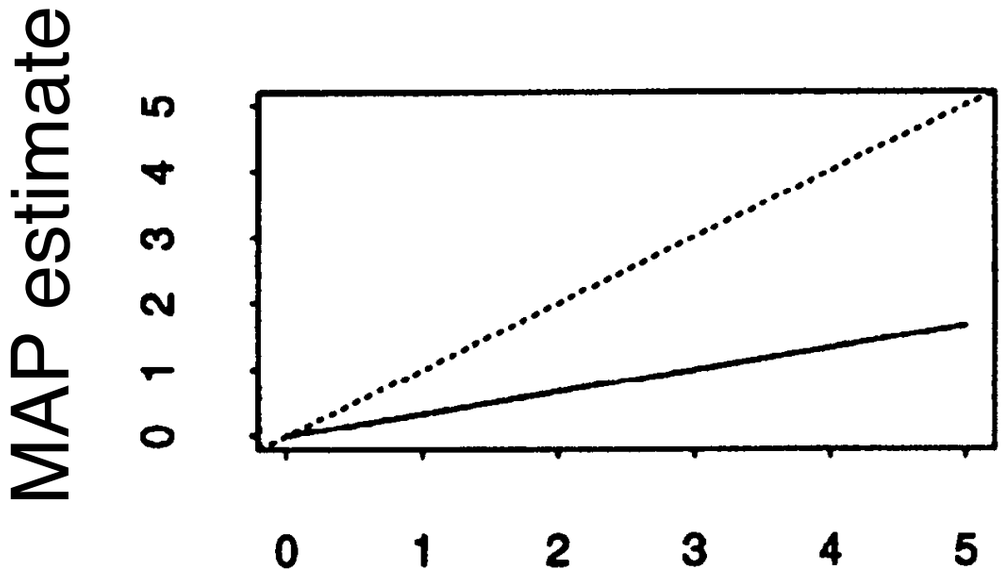
lasso: *minimize* $||Y - X\mathbf{w}||_2^2 + \lambda \sum_i |w_i|$

Bayesian interpretation: log prior

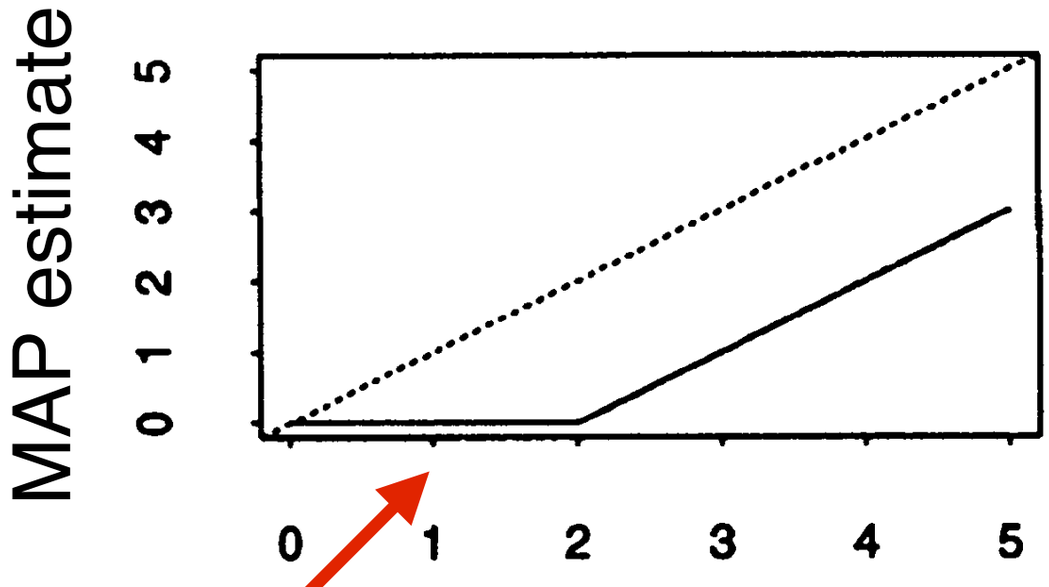
$$P(w_i) = \frac{1}{2}\lambda e^{-\lambda|w_i|}$$

(Laplace prior)

ridge regression
(L2 penalty)



lasso
(L1 penalty)

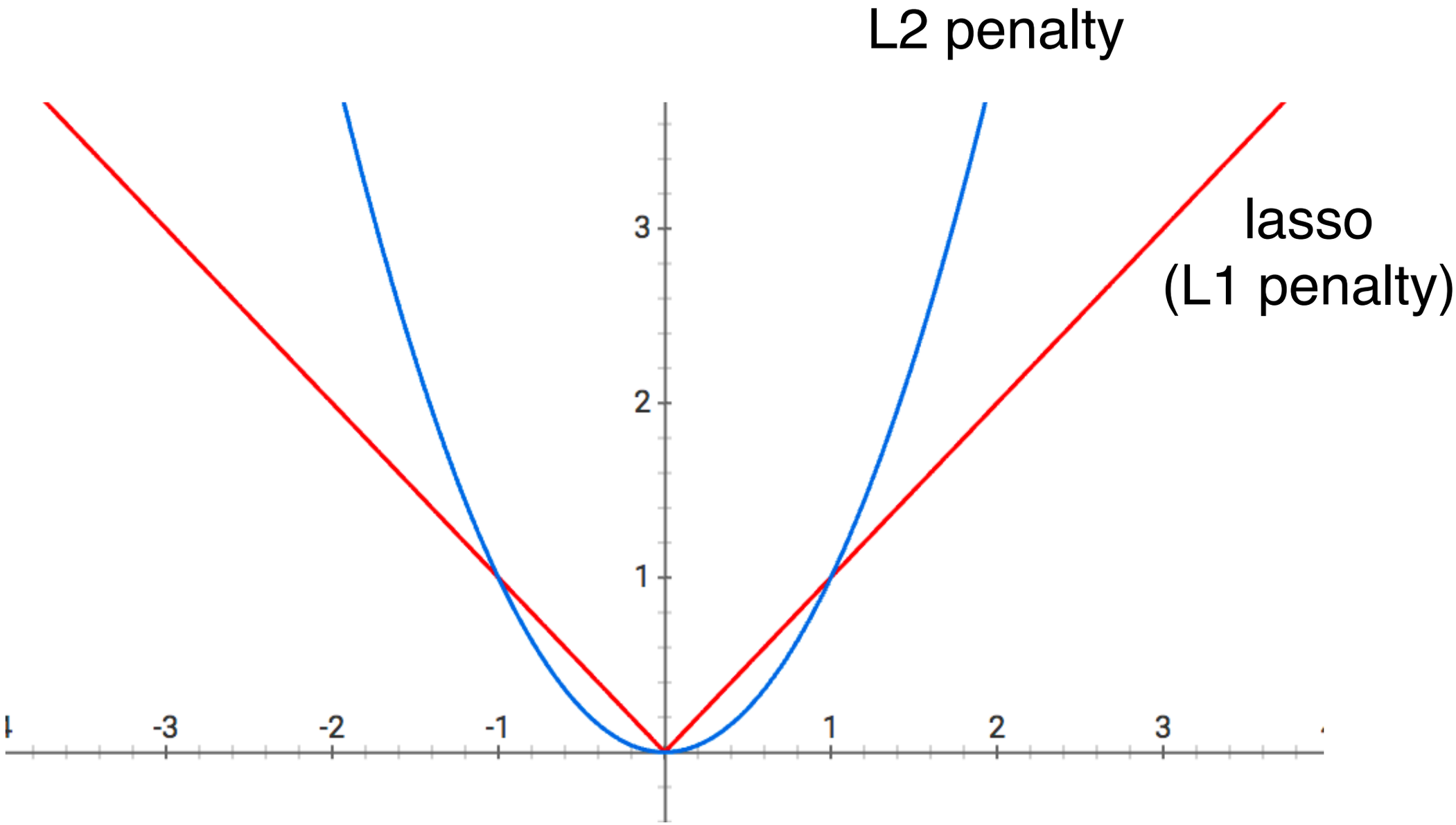


estimate is "shrunk" to zero!

$\Longrightarrow$

"sparse" solutions

Why does lasso achieve sparsity?

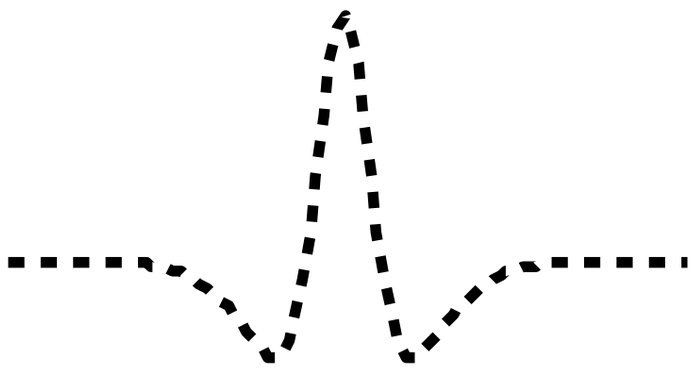


- derivative of L2 penalty falls to 0 at 0
- derivative of L1 penalty is constant on each side of 0

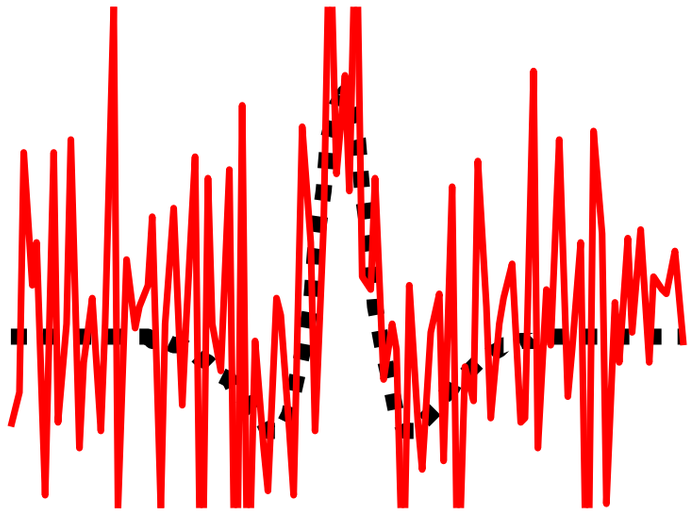
(hence is non-differentiable at 0; it requires special optimization techniques as a result!)

Simulated Example: lasso

true **w**



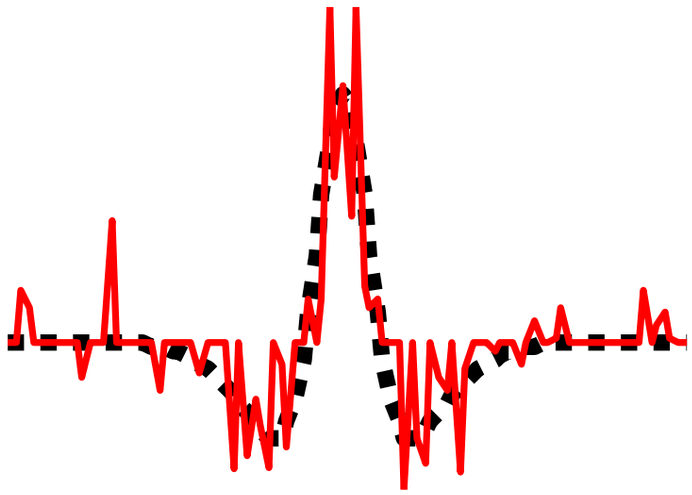
maximum likelihood



maximize

$$\log p(data|\mathbf{w})$$

lasso



maximize

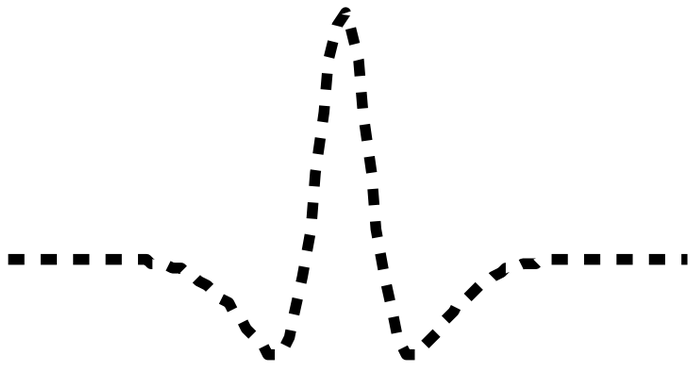
$$\log p(data|\mathbf{w}) - \lambda \underbrace{\sum_i |\mathbf{w}_i|}$$

sparsity penalty

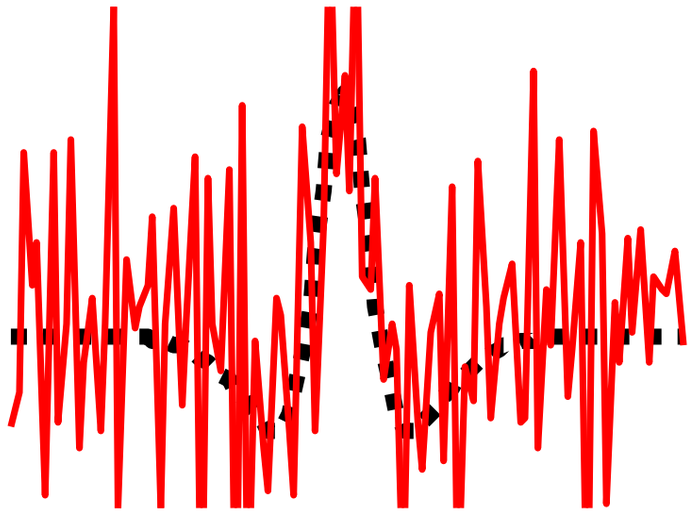
"compressed sensing" - good theoretical guarantees when w is sparse

Solution #3: Graph Laplacian / L2 smoothing

true **w**



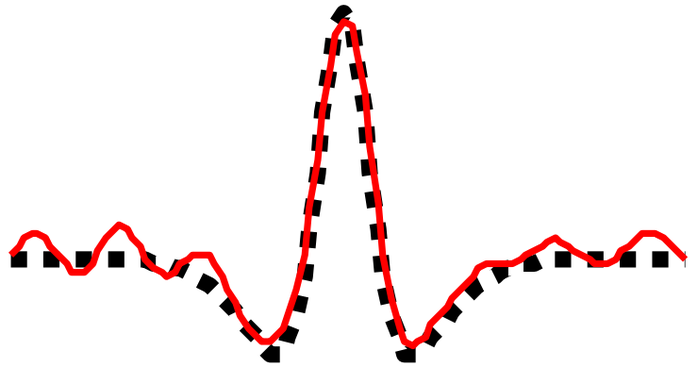
maximum likelihood



maximize

$$\log p(data|\mathbf{w})$$

“smoothed”



maximize

$$\log p(data|\mathbf{w}) - \lambda \sum (w_i - w_{i-1})^2$$

smoothness penalty

Q: how to set the regularization parameter λ ?

Simplest answer: use cross-validation!

(on board)

Tutorials

GLM tutorial:

https://github.com/pillowlab/GLMspiketraintutorial

- **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.

tutorial1_PoissonGLM.m

1. cd into correct directory ('pillow_GLMtutorials')
2. open tutorial: use `ctrl-enter`, `ctrl-down`

Dataset:

- retinal ganglion cell spike trains [Uzzell & Chichilnisky 2004]
- 2 OFF cells, 2 off cells
- full-field, binary white noise stimuli

tutorial1_PoissonGLM.m

1. cd into correct directory ('pillow_GLMtutorials')
2. open tutorial: use `ctrl-enter`, `ctrl-down`

Topics:

1-2. load data, bin spike trains
3. build design matrix (slow and fast versions)
4a. compute STA (for vizualization)
4b. *linear-Gaussian GLM* ("least-squares regression")
5. *Poisson GLM*: fit assuming exponential nonlinearity
6. *Poisson GLM*: non-parametric estimate of nonlinearity
7. model comparison: log-likelihood & mutual information
8. model comparison: AIC
9. simulate from fitted model

tutorial2_spikehistcoupledGLM.m

Topics:

1-2. load data & bin spike trains

3. build design matrix, now including spike history!

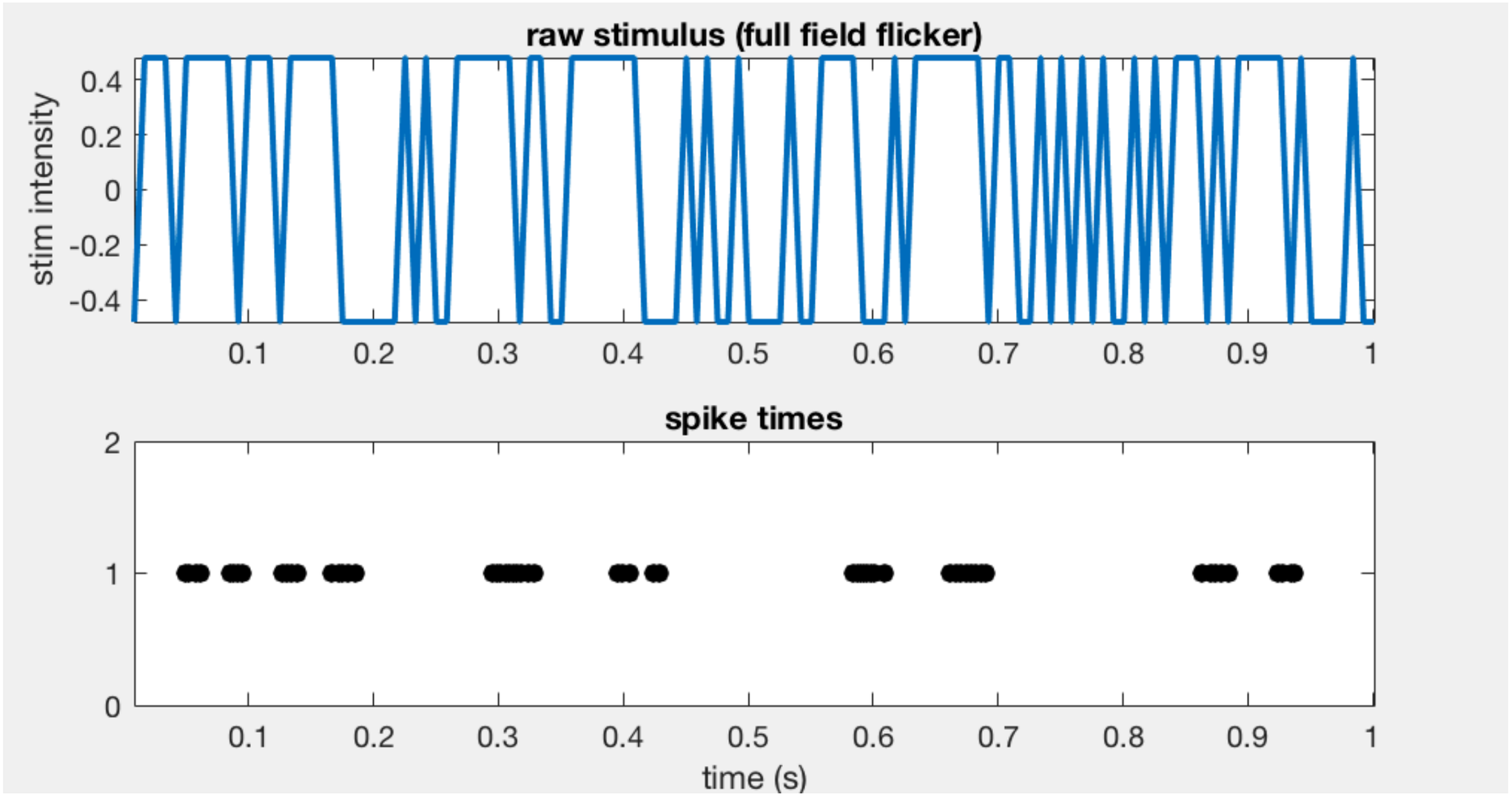
4. fit Poisson GLM with spike-history

5. fit coupled Poisson GLM (4 neuron spike-history)

6. model comparison: log-likelihood & AIC

Example dataset

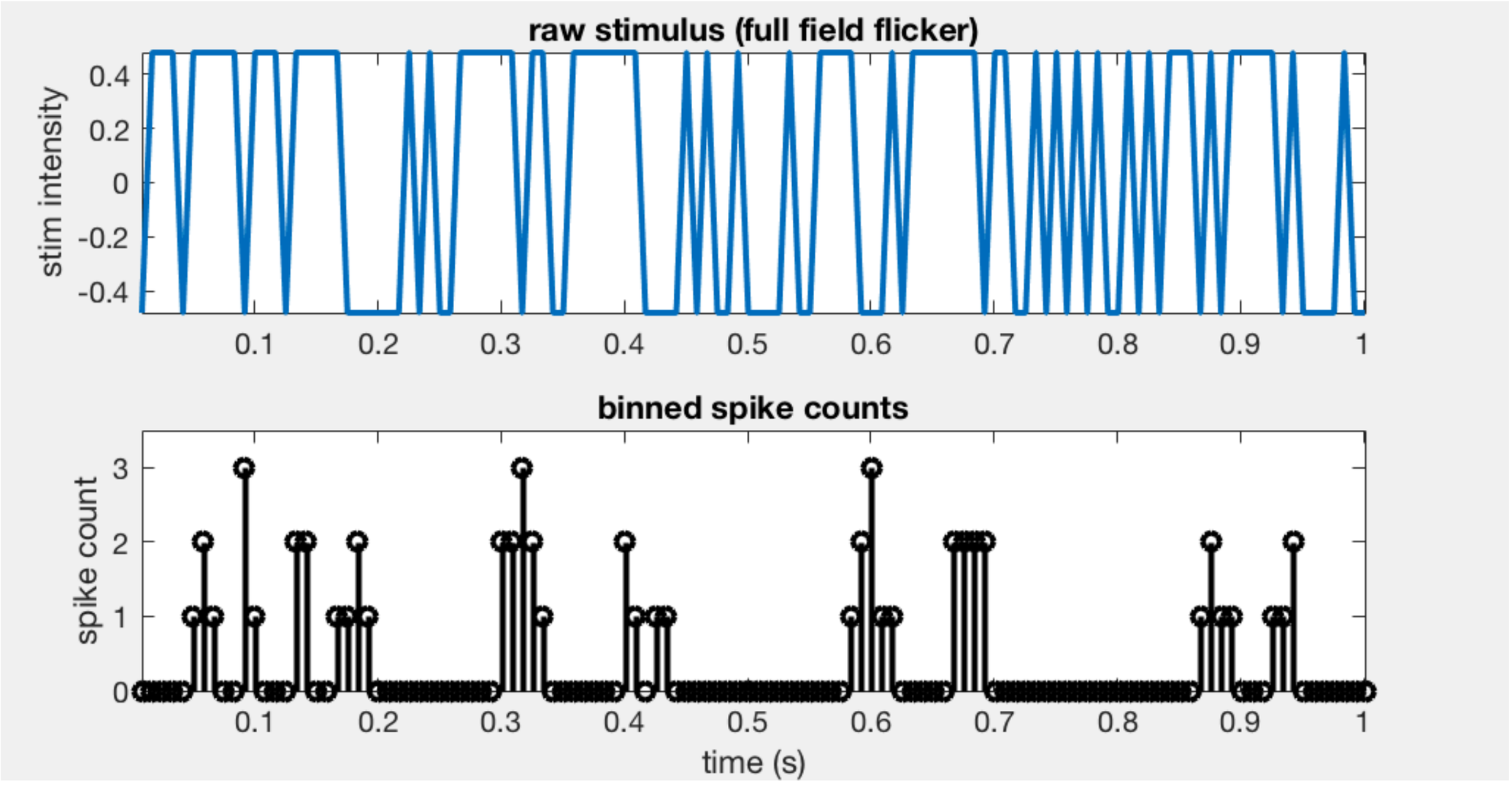
- stimulus = binary flicker
- parasol retinal ganglion cell spike responses



Uzzell et al (J Neurophys 04)

Example dataset

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

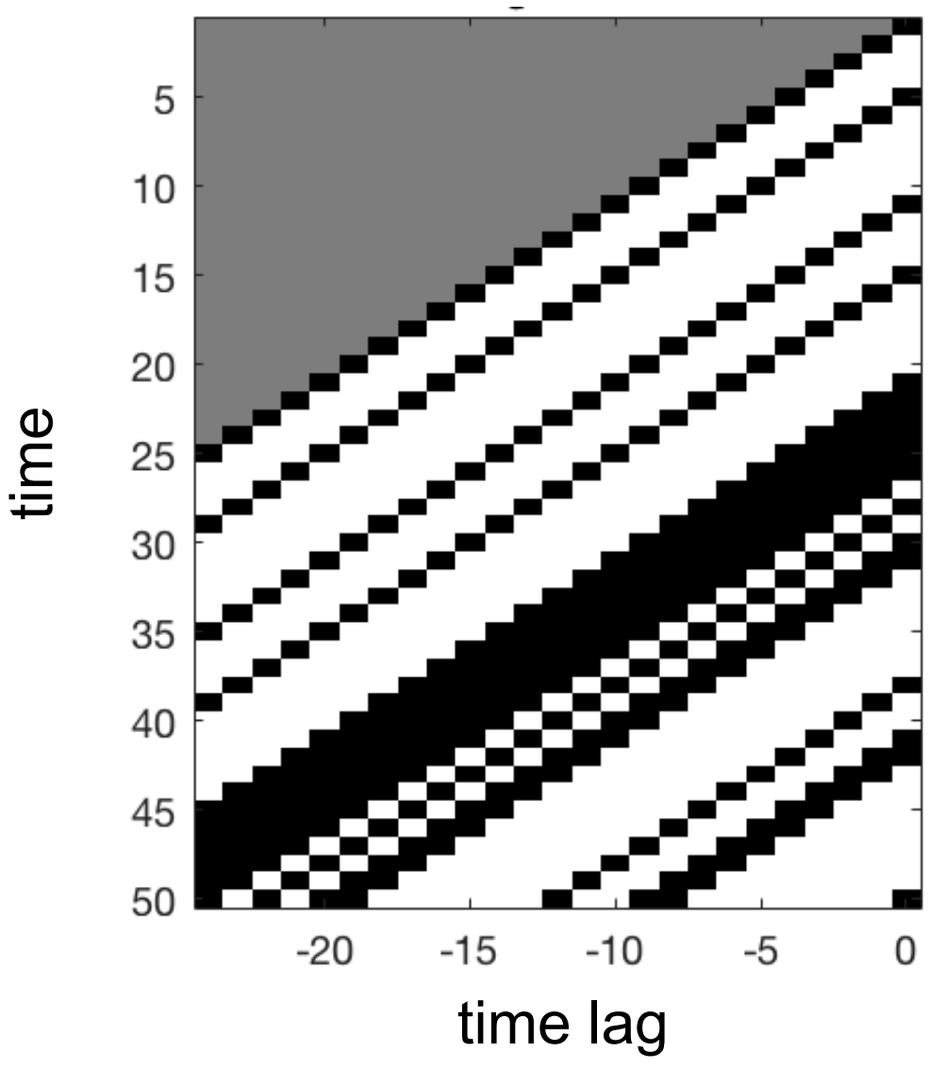


Uzzell et al (J Neurophys 04)

Stimulus-only GLM

design matrix

X



model

$P(Y|X)$

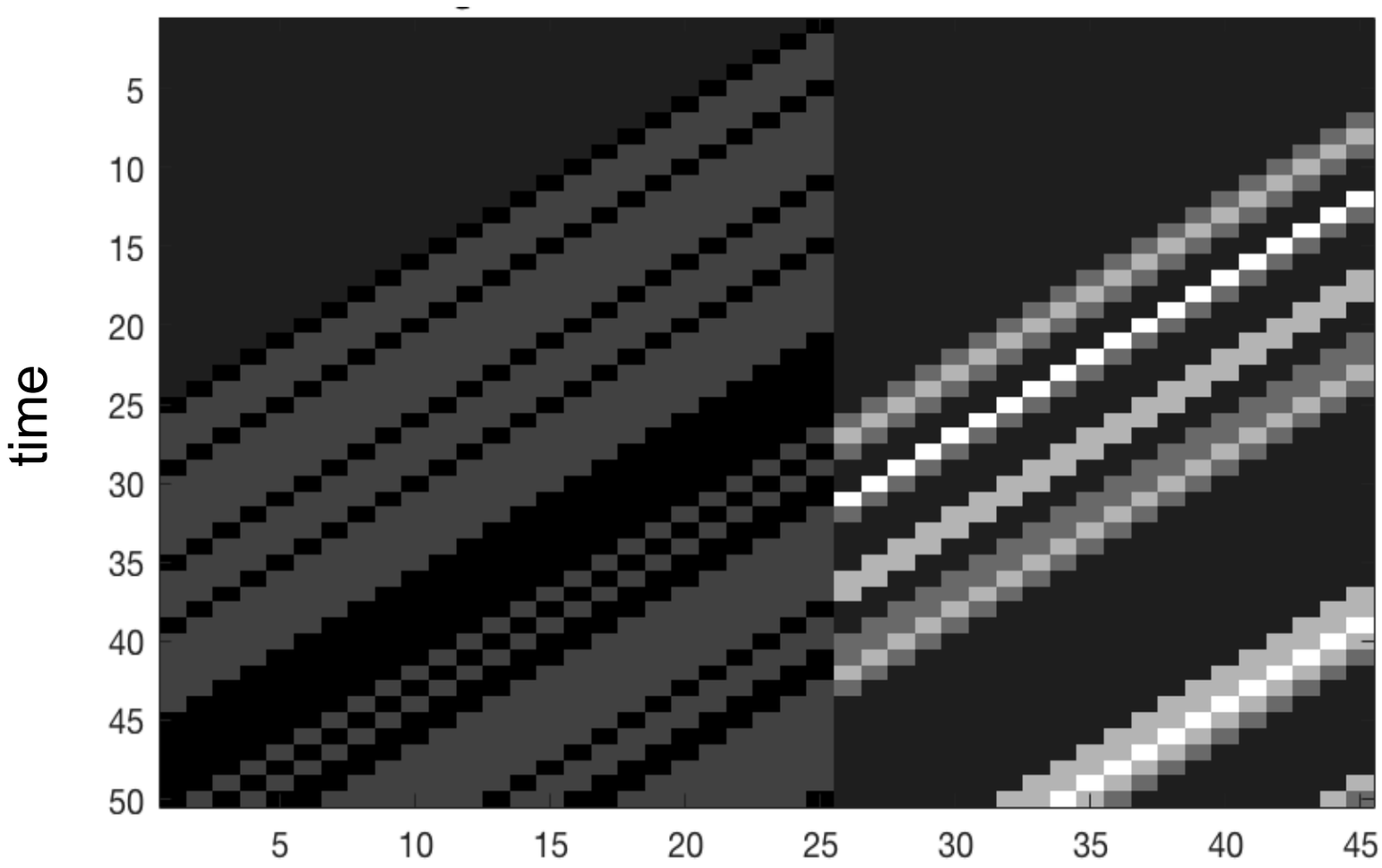
spike response

Y

Stimulus + SpikeHistory GLM

design matrix

X

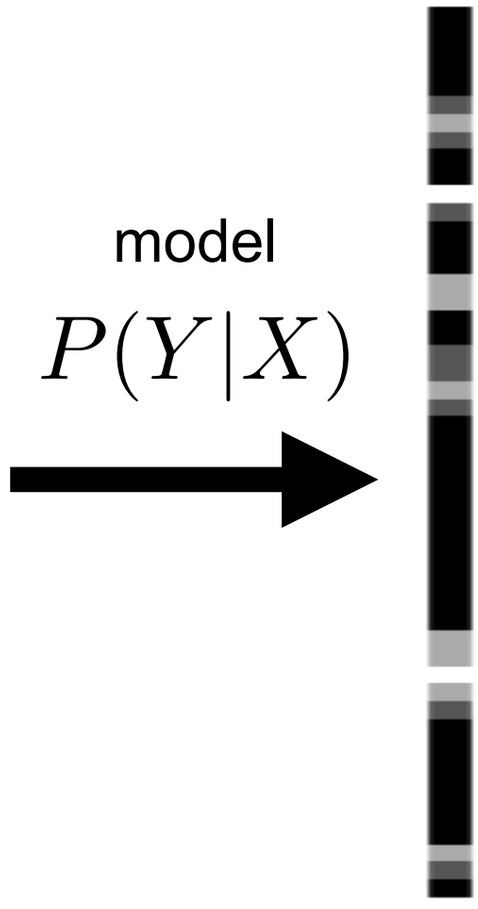


spike response

Y

model

$P(Y|X)$



Stimulus + History + 3 Neuron Coupling GLM

design matrix

X

spike response

Y