# Generalized Linear Model

By Anqi Wu

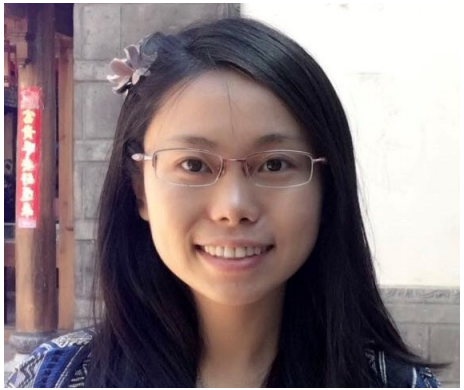
# Who is Anqi Wu?

- Postdoc at Columbia with Liam Paninski and John Cunningham

- PhD at Princeton with Jonathan Pillow

- Research interest: neural sensory encoding, latent variable models for large-scale neural recordings, fMRI decoding, behavior analysis

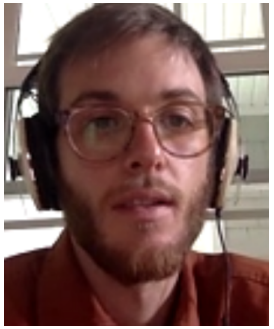
- Modeling: probabilistic graphical model, Gaussian process, latent variable/dynamic model, Bayesian deep learning, Bayesian optimization and active learning



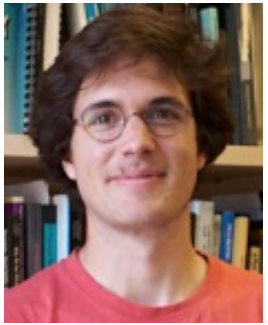
# D4 Team

Jakob Macke

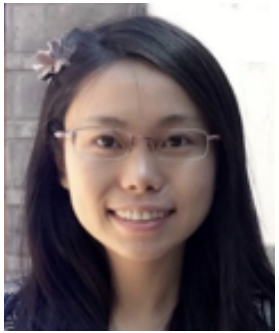
Cristina Savin



Ari Benjamin



Pierre Etienne Fiquet



Anqi Wu

Il Memming Park

# Roadmap of Week 1 Day 4

Tutorial part 1

- Linear Gaussian model
- Poisson GLM

Tutorial part 2

- Logistic regression
- Regularization
  - Ridge (L2)
  - Lasso (L1)

# Linear Gaussian Model

On day 3, we talked about the multiple linear model :

Vector form

neural response

$$y_i = \boldsymbol{\theta}^\top \boldsymbol{x}_i \qquad \forall i = 1, ..., N$$

$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, ...]^\top$ linear weights

$\boldsymbol{x}_i = [1, x_{i,1}, x_{i,2}, ...]^\top$ multiple stimulus features (e.g., orientation, contrast, etc.)

number of data points

Matrix form

$$\boldsymbol{y} = X\boldsymbol{\theta}$$

Index i

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} — & \boldsymbol{x}_0 & — \\ — & \boldsymbol{x}_1 & — \\ — & \boldsymbol{x}_2 & — \\ & \vdots & \end{bmatrix} \quad \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix}$$

design matrix

If we assume the observation noise to be Gaussian, then

Vector form

neural response

Gaussian noise $\eta \sim \mathcal{N}(0, \sigma^2)$

$$y_i = \boldsymbol{\theta}^\top \boldsymbol{x}_i + \eta \quad \forall i = 1, ..., N$$

$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, ...]^\top$ linear weights

$\boldsymbol{x}_i = [1, x_{i,1}, x_{i,2}, ...]^\top$ multiple stimulus features (e.g., orientation, contrast, etc.)

number of data points

Matrix form

$$\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\eta}$$

Index i

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} — & \boldsymbol{x}_0 & — \\ — & \boldsymbol{x}_1 & — \\ — & \boldsymbol{x}_2 & — \\ & \vdots & \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix} + \begin{bmatrix} \eta_0 \\ \eta_1 \\ \eta_2 \\ \vdots \end{bmatrix}$$

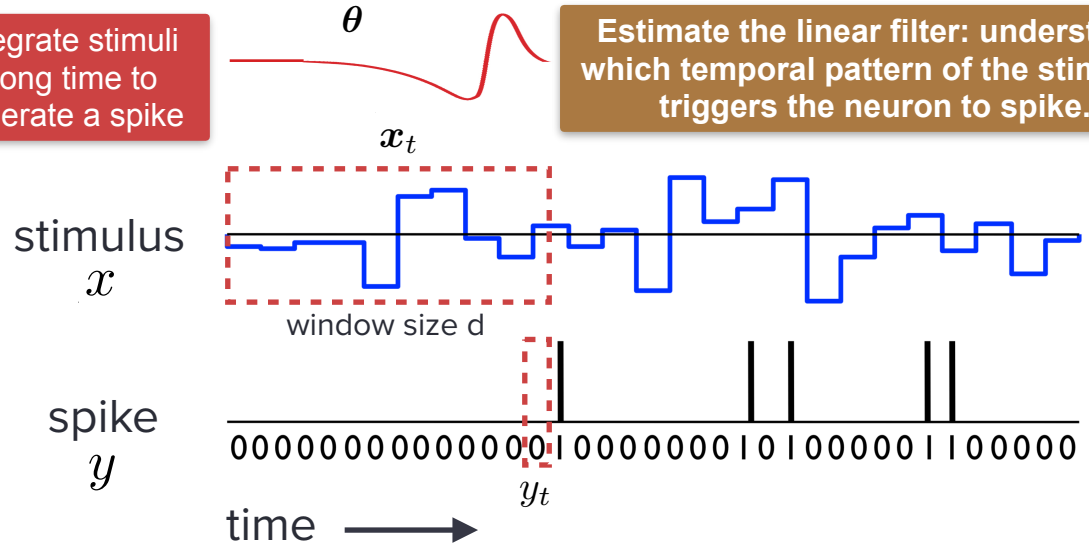
$\eta \sim \mathcal{N}(0, \sigma^2)$

design matrix

# Temporal Filtering Model

Task: predict neural spikes from stimuli at all time points

Integrate stimuli along time to generate a spike




**Estimate the linear filter: understand which temporal pattern of the stimulus triggers the neuron to spike.**

window size

$$\lambda_t = \boldsymbol{\theta}^\top \boldsymbol{x}_t = \sum_{i=0}^{d-1} \theta_i x_{t-i}$$

$$y_i = \boldsymbol{\theta}^\top \boldsymbol{x}_i + \eta$$

firing rate at time t — linear filter — vector stimulus at time t

$$y_t = \lambda_t + \eta_t$$

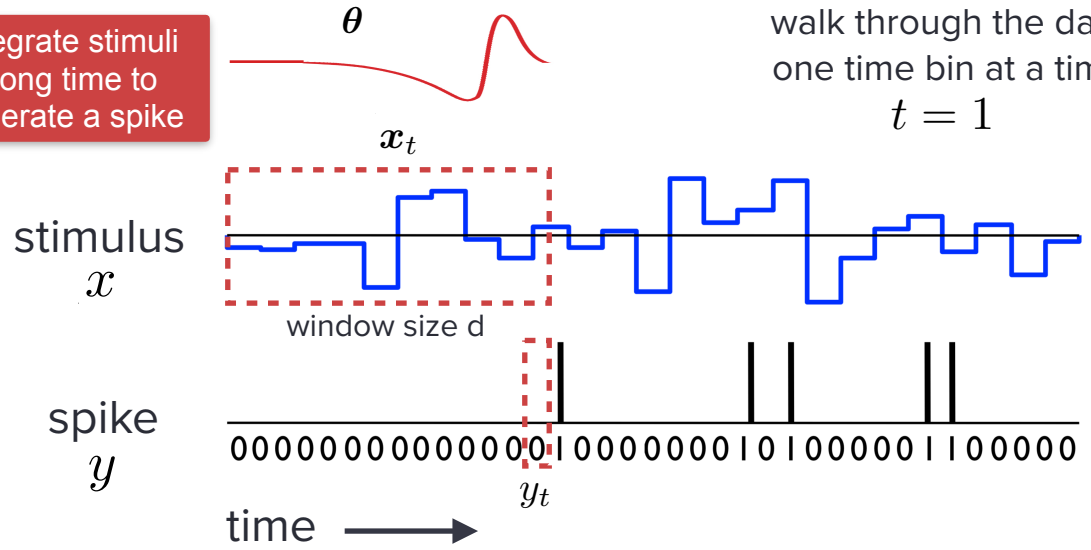
spike at time t — Gaussian noise

source: http://pillowlab.princeton.edu/teaching/statneuro2018/slides/lec09_GLMs1.pdf

# Temporal Filtering Model

Task: predict neural spikes from stimuli at all time points

Integrate stimuli along time to generate a spike




walk through the data one time bin at a time

$t = 1$

window size

$$\lambda_t = \boldsymbol{\theta}^\top \boldsymbol{x}_t = \sum_{i=0}^{d-1} \theta_i x_{t-i}$$

firing rate at time t — linear filter — vector stimulus at time t

$$y_t = \lambda_t + \eta_t$$

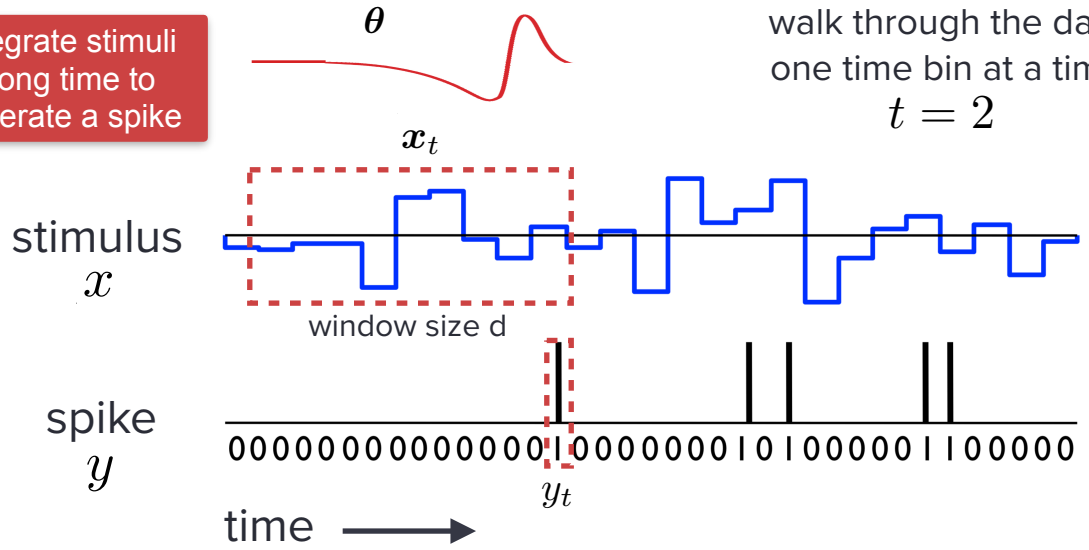
spike at time t — Gaussian noise

source: http://pillowlab.princeton.edu/teaching/statneuro2018/slides/lec09_GLMs1.pdf

# Temporal Filtering Model

Task: predict neural spikes from stimuli at all time points

Integrate stimuli along time to generate a spike




window size

$$\lambda_t = \boldsymbol{\theta}^\top \boldsymbol{x}_t = \sum_{i=0}^{d-1} \theta_i x_{t-i}$$

firing rate at time t

linear filter

vector stimulus at time t

$$y_t = \lambda_t + \eta_t$$

spike at time t

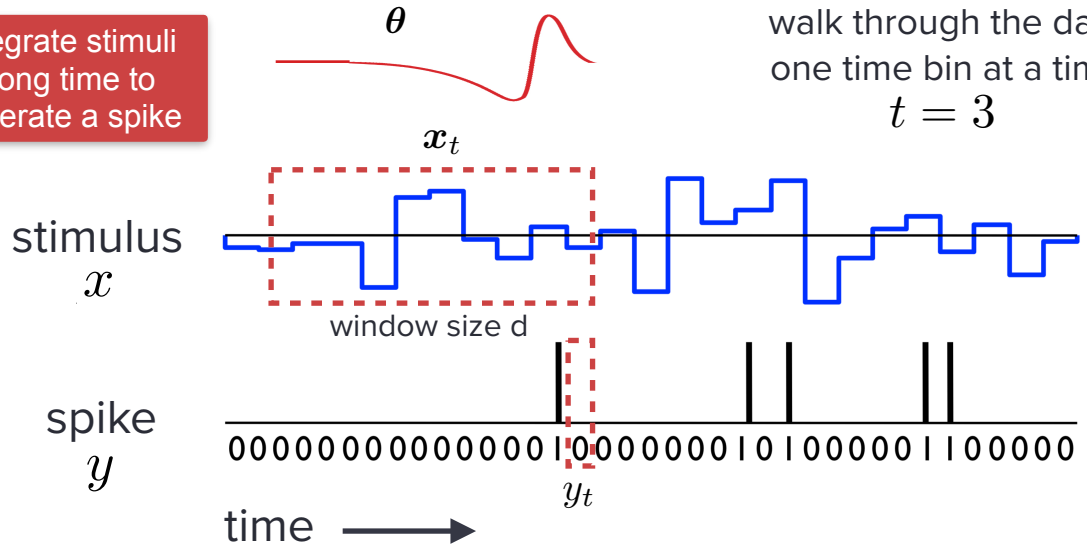
Gaussian noise

source: http://pillowlab.princeton.edu/teaching/statneuro2018/slides/lec09_GLMs1.pdf

# Temporal Filtering Model

Task: predict neural spikes from stimuli at all time points

Integrate stimuli along time to generate a spike




walk through the data one time bin at a time

$t = 3$

window size

$$\lambda_t = \boldsymbol{\theta}^\top \boldsymbol{x}_t = \sum_{i=0}^{d-1} \theta_i x_{t-i}$$

firing rate at time t

linear filter

vector stimulus at time t

$$y_t = \lambda_t + \eta_t$$

spike at time t

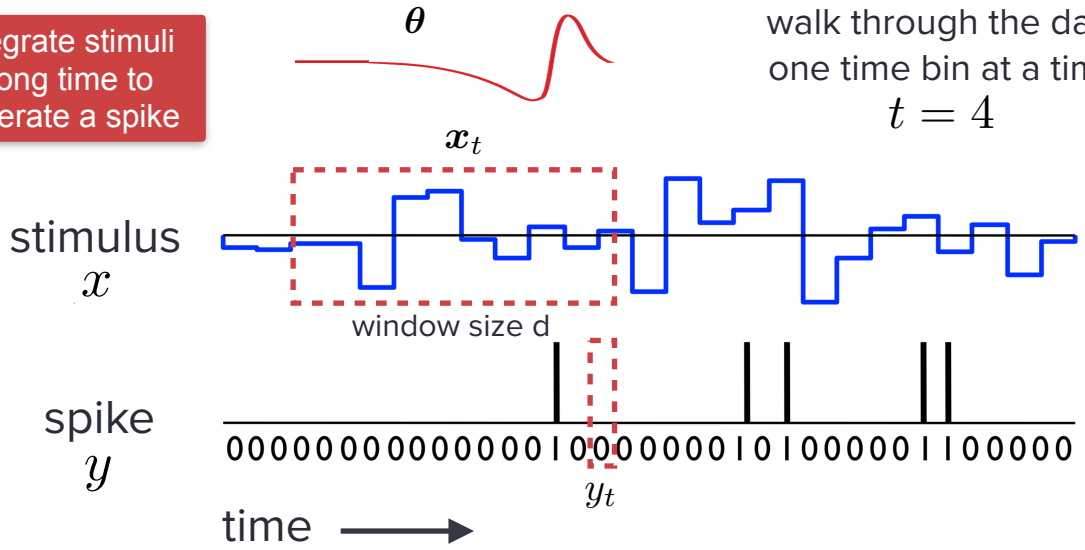
Gaussian noise

source: http://pillowlab.princeton.edu/teaching/statneuro2018/slides/lec09_GLMs1.pdf

# Temporal Filtering Model

Task: predict neural spikes from stimuli at all time points

Integrate stimuli along time to generate a spike




window size

$$\lambda_t = \boldsymbol{\theta}^\top \boldsymbol{x}_t = \sum_{i=0}^{d-1} \theta_i x_{t-i}$$

firing rate at time t

linear filter

vector stimulus at time t

$$y_t = \lambda_t + \eta_t$$

spike at time t

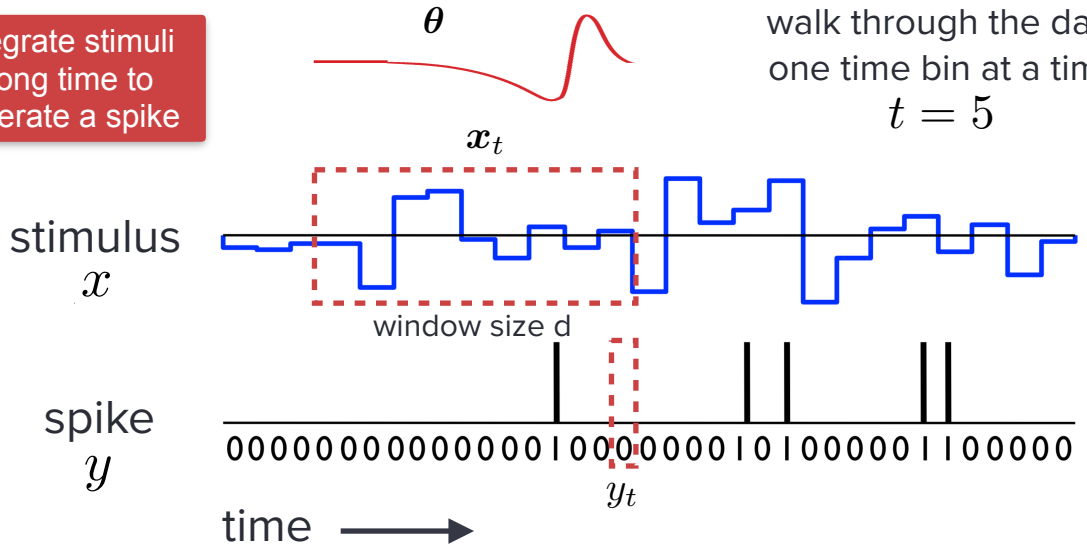
Gaussian noise

source: http://pillowlab.princeton.edu/teaching/statneuro2018/slides/lec09_GLMs1.pdf

# Temporal Filtering Model

Task: predict neural spikes from stimuli at all time points

Integrate stimuli along time to generate a spike




window size

$$\lambda_t = \boldsymbol{\theta}^\top \boldsymbol{x}_t = \sum_{i=0}^{d-1} \theta_i x_{t-i}$$

firing rate at time t — linear filter — vector stimulus at time t

$$y_t = \lambda_t + \eta_t$$

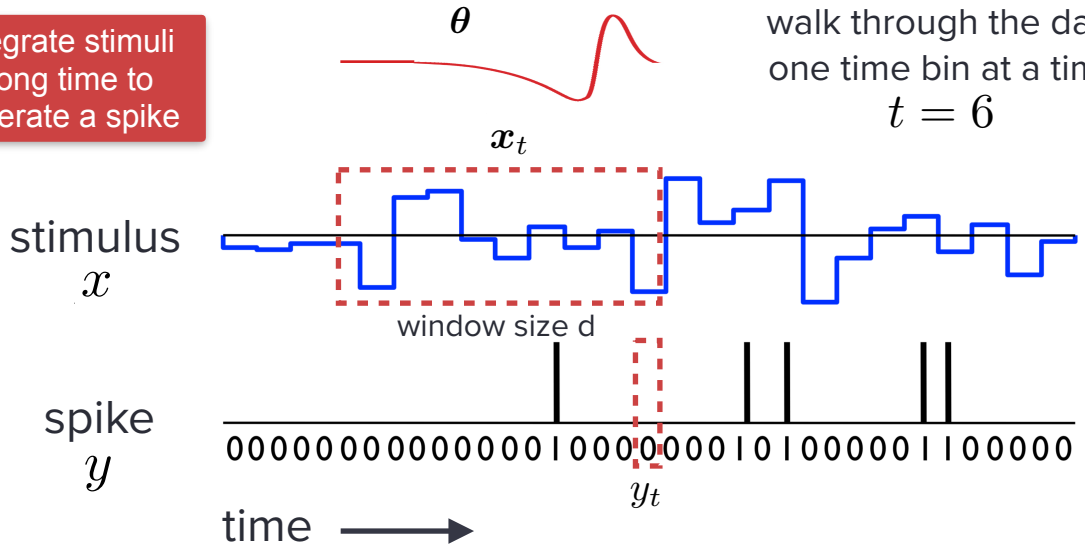
spike at time t — Gaussian noise

source: http://pillowlab.princeton.edu/teaching/statneuro2018/slides/lec09_GLMs1.pdf

# Temporal Filtering Model

Task: predict neural spikes from stimuli at all time points

Integrate stimuli along time to generate a spike




window size

$$\lambda_t = \boldsymbol{\theta}^\top \boldsymbol{x}_t = \sum_{i=0}^{d-1} \theta_i x_{t-i}$$

firing rate at time t — linear filter — vector stimulus at time t

$$y_t = \lambda_t + \eta_t$$

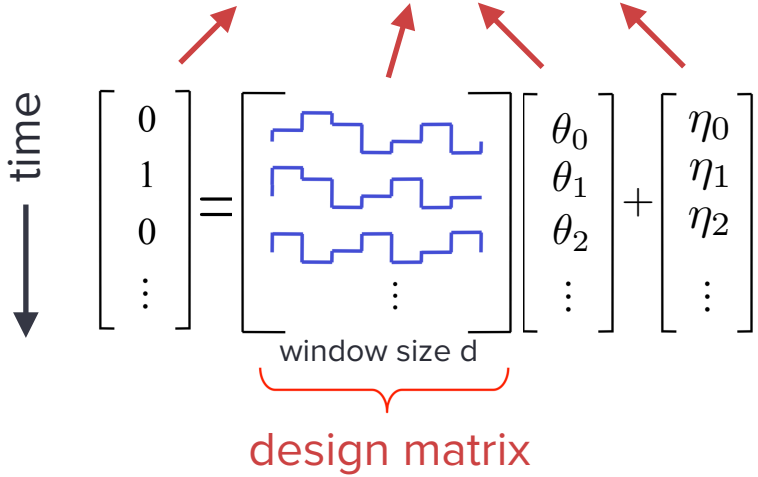
spike at time t — Gaussian noise

source: http://pillowlab.princeton.edu/teaching/statneuro2018/slides/lec09_GLMs1.pdf

# Temporal Filtering Model

Build up to the following matrix version along time axis:

$$\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\eta}$$




MSE solution, ignoring the noise $\boldsymbol{\eta}$ (revisit D3):

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \, ||X\boldsymbol{\theta} - \boldsymbol{y}||_2^2 = \sum_{t=1}^{T} (y_t - \boldsymbol{\theta}^\top \boldsymbol{x}_t)^2$$

Differentiate and set to zero

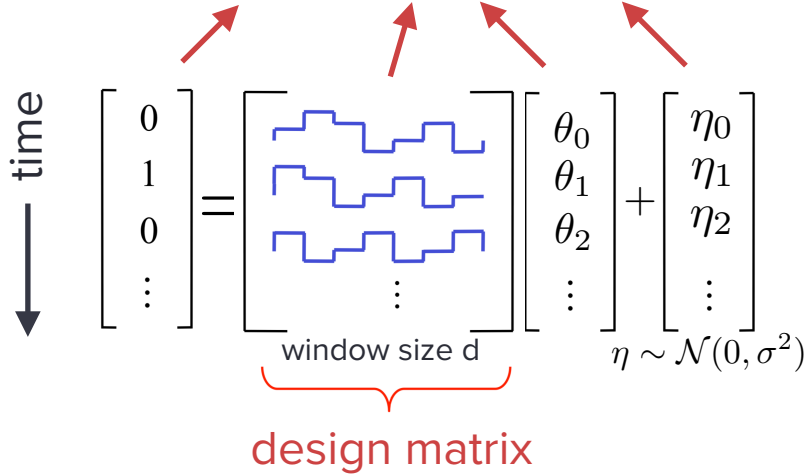
$$\frac{\partial \, ||X\boldsymbol{\theta} - \boldsymbol{y}||_2^2}{\partial \, \boldsymbol{\theta}} = 2X^\top (X\boldsymbol{\theta} - \boldsymbol{y}) = 0$$

$$\Rightarrow \; \boldsymbol{\theta}_{\text{MSE}} = \underbrace{(X^\top X)}_{\text{stimulus covariance}}^{-1} \underbrace{X^\top \boldsymbol{y}}_{\text{spike-triggered avg (STA)}}$$

# Temporal Filtering Model

Build up to the following matrix version along time axis:

$$\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\eta}$$




MLE solution (revisit D3):

the number of time points

window size

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}} \ \log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y}) = -\frac{Td}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}(\boldsymbol{y} - X\boldsymbol{\theta})^\top(\boldsymbol{y} - X\boldsymbol{\theta})$$

Differentiate and set to zero

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y})}{\partial \boldsymbol{\theta}} = -\frac{1}{\sigma^2}X^\top(X\boldsymbol{\theta} - \boldsymbol{y}) = 0$$

$$\Rightarrow \boldsymbol{\theta}_{\text{MLE}} = (X^\top X)^{-1}X^\top \boldsymbol{y}$$

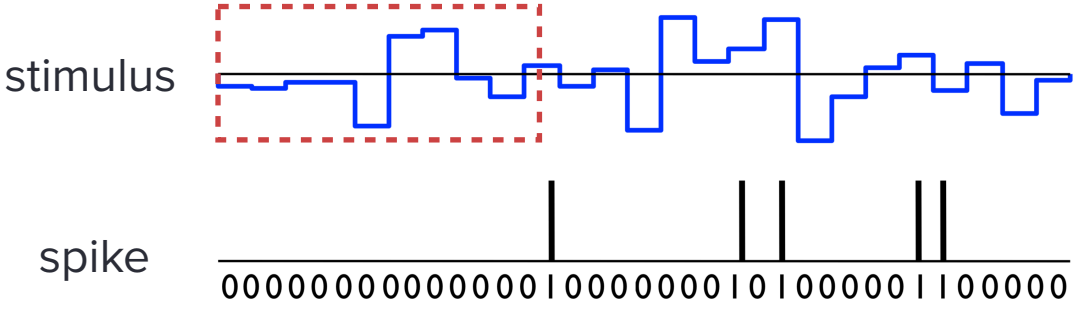
(same as MSE when the noise is Gaussian)

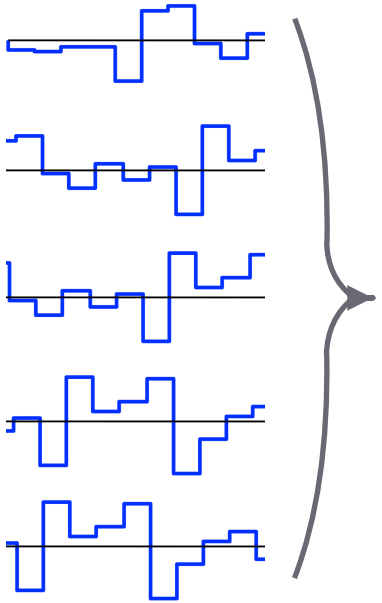
stimulus covariance

spike-triggered avg (STA)

# Spike-triggered Average (STA)

$$\boldsymbol{\theta}_{\text{STA}} = X^\top \boldsymbol{y}$$




In general, $\boldsymbol{\theta}_{\text{MLE}}$ is better than $\boldsymbol{\theta}_{\text{STA}}$ .

# Notebook

In the following notebook, you will

1. Formulate the design matrix from the stimulus vector.
2. Fit the linear Gaussian model to the stimulus and spike count data.
3. Predict spike counts using the fitted linear Gaussian model.

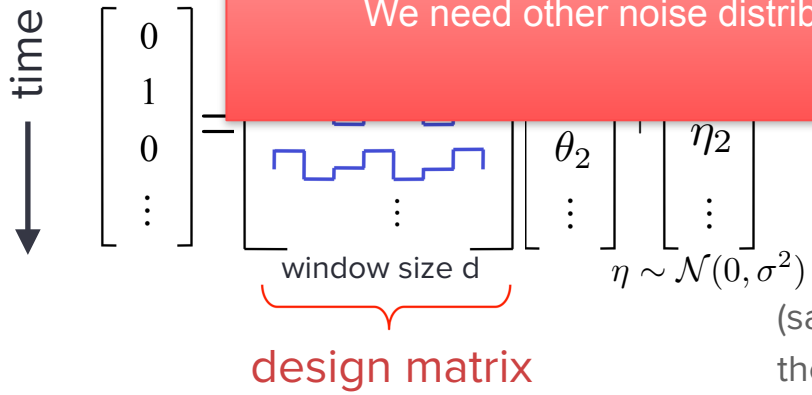
Enjoy!!

# Generalized Linear Model

# Temporal Filtering Model

Build up to following matrix version along time axis:

The noise is not Gaussian!
The neural response is spike count which is non-negative and discrete!
We need other noise distribution and nonlinear function.




design matrix

$$-\frac{1}{2\sigma^2}(\boldsymbol{y} - X\boldsymbol{\theta})^\top(\boldsymbol{y} - X\boldsymbol{\theta})$$

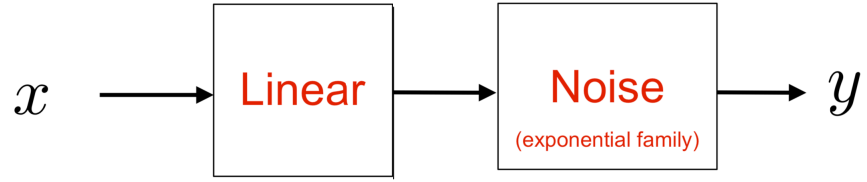
…ts and $d$ is the window

$$\frac{\partial \log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y})}{\partial\, \boldsymbol{\theta}} = -\frac{1}{\sigma^2}X^\top(X\boldsymbol{\theta} - \boldsymbol{y}) = 0$$

$$\Rightarrow \boldsymbol{\theta}_{\text{MLE}} = \underbrace{(X^\top X)^{-1}}_{\text{stimulus covariance}} \underbrace{X^\top \boldsymbol{y}}_{\text{spike-triggered avg (STA)}}$$

(same as MSE when the noise is Gaussian)

Linear model




Example: linear Gaussian model $y = \theta x + \eta$ where $\eta \sim \mathcal{N}(0, \sigma^2)$

Generalized linear model

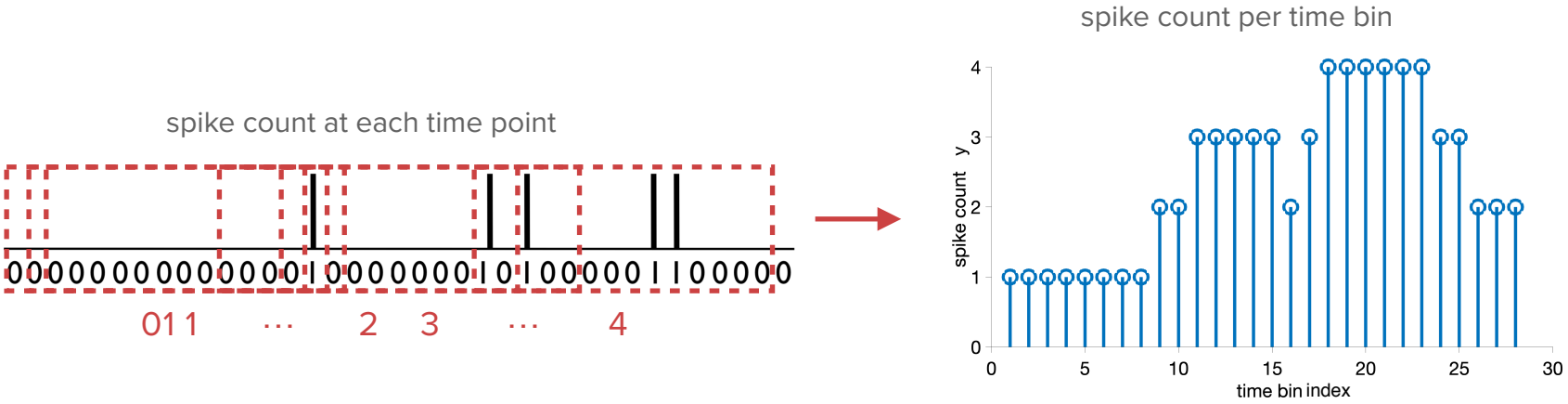

Example: nonlinear Gaussian model $y = f(\theta x) + \eta$ where $\eta \sim \mathcal{N}(0, \sigma^2)$

nonlinear
$f^{-1}$: link function

Poisson GLM $y \sim \text{Poisson}\left(f(\theta x)\right)$ for spike train encoding

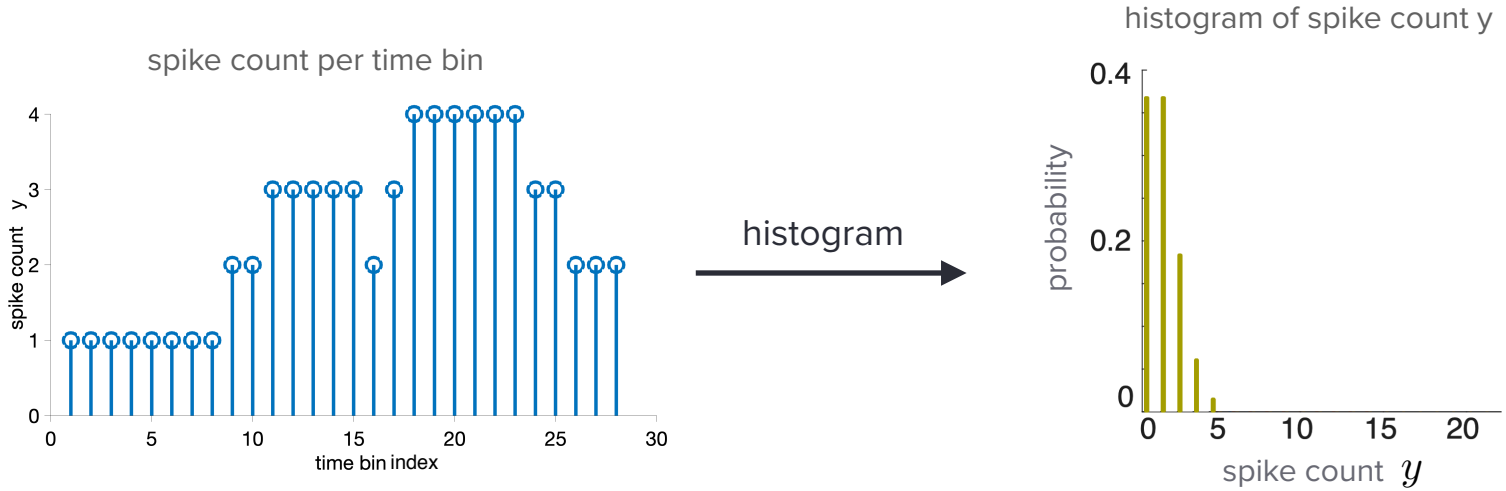
# Poisson GLM

**Poisson distribution** is used to model the number of events (spikes) occurring within a given time interval.



# Poisson GLM

**Poisson distribution** is used to model the number of events (spikes) occurring within a given time interval.



# Poisson GLM

**Poisson distribution** is used to model the number of events (spikes) occurring within a given time interval.

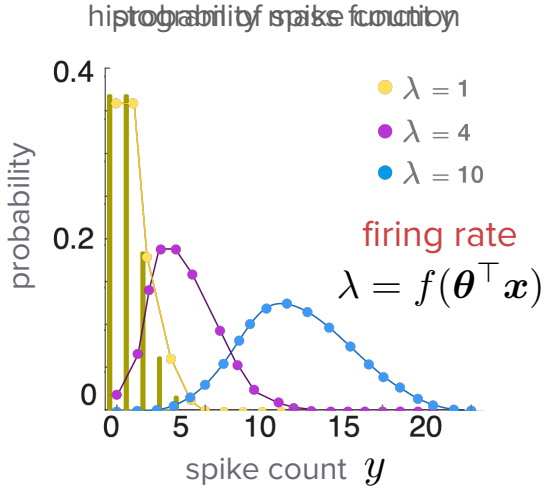
The probability mass function (pmf) of $Y$ is given by

$$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!}$$

unknown parameter, >0

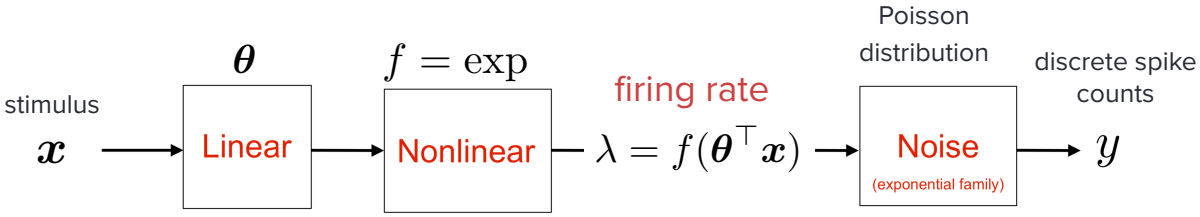
observed discrete spike count y=0,1,2,3,...

not contain $\lambda$



# Poisson GLM

## Temporal filtering model




Given the probability mass function for Poisson distribution $P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!}$

We can now assume the encoding distribution to be

for time t $p(y_t|\boldsymbol{\theta}, \boldsymbol{x}_t) = P(Y = y_t|\lambda_t = f(\boldsymbol{\theta}^\top \boldsymbol{x_t})) = \frac{[f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)]^{y_t}}{y_t!} e^{-f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)}$

# Poisson GLM

## Temporal filtering model

for time t $$p(y_t|\boldsymbol{\theta}, \boldsymbol{x}_t) = P(Y = y_t|\lambda_t = f(\boldsymbol{\theta}^\top \boldsymbol{x_t})) = \frac{[f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)]^{y_t}}{y_t!} e^{-f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)}$$

log likelihood $$\log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y}) = \log p(\boldsymbol{y}|X, \boldsymbol{\theta}) = \log \prod_{t=1}^{T} p(y_t|\boldsymbol{\theta}, \boldsymbol{x}_t) = \log \prod_{t=1}^{T} \frac{[f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)]^{y_t}}{y_t!} e^{-f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)}$$

$$= \sum_{t=1}^{T} \log \frac{[f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)]^{y_t}}{y_t!} + \sum_{t=1}^{T} \log e^{-f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)}$$

$$= \sum_{t=1}^{T} y_t \log f(\boldsymbol{\theta}^\top \boldsymbol{x}_t) - \sum_{t=1}^{T} \log y_t! - \sum_{t=1}^{T} f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)$$

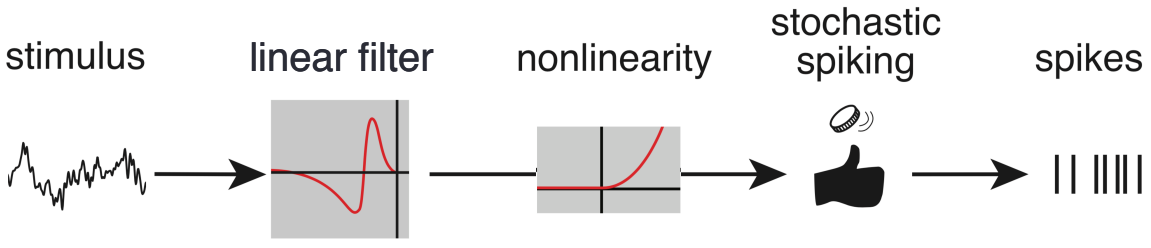
$$= \sum_{t=1}^{T} \left(y_t \log f(\boldsymbol{\theta}^\top \boldsymbol{x}_t) - f(\boldsymbol{\theta}^\top \boldsymbol{x}_t)\right) + const = \boldsymbol{y}^\top \log f(X\boldsymbol{\theta}) - \mathbf{1}^\top f(X\boldsymbol{\theta}) + const$$

# Poisson GLM

## Temporal filtering model

log likelihood $$\log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y}) = \log p(\boldsymbol{y}|X, \boldsymbol{\theta}) = \boldsymbol{y}^\top \log f(X\boldsymbol{\theta}) - \mathbf{1}^\top f(X\boldsymbol{\theta}) + const$$

- Solving $\frac{\partial \log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y})}{\partial \boldsymbol{\theta}} = 0$ has no closed-form
- $-\log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y})$ is a convex function
- Convex optimization such as gradient descent



# Notebook

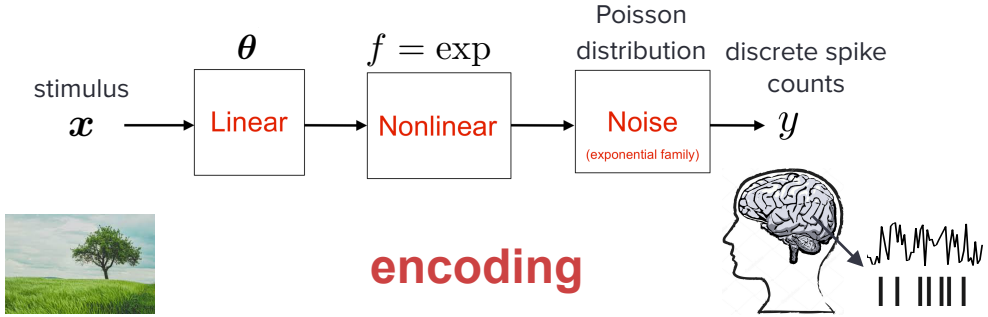
In the following notebook, you will

1. Continue the fitting for the same stimulus and spike count data.
2. Fit Poisson GLM by optimization and predict spike counts.
3. Compare the prediction from linear Gaussian and Poisson GLM.

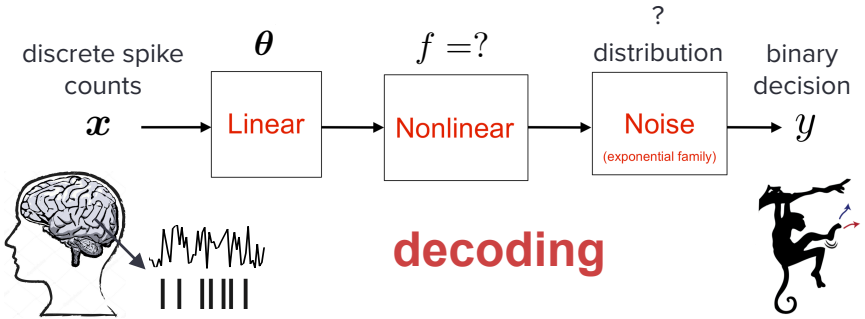
Enjoy!!

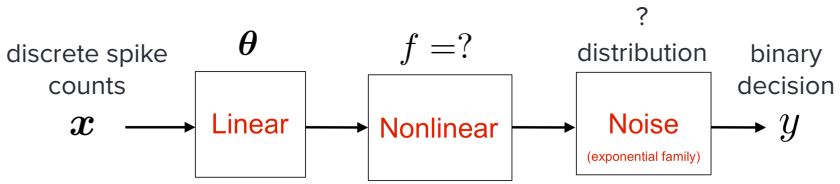
# Logistic Regression

Previously we introduce how to use Poisson GLM to **encode** neural spike trains from stimuli.

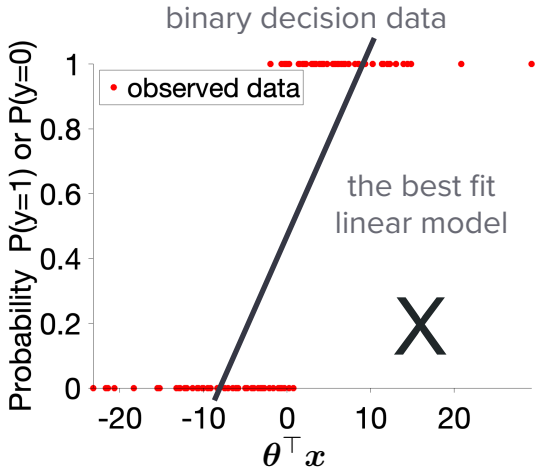


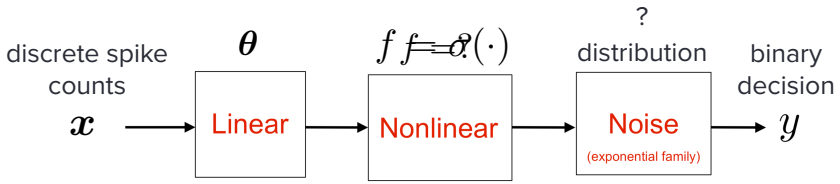

Here we will talk about **decoding** a binary decision of an animal from spike train data using a specific instantiation of GLM.



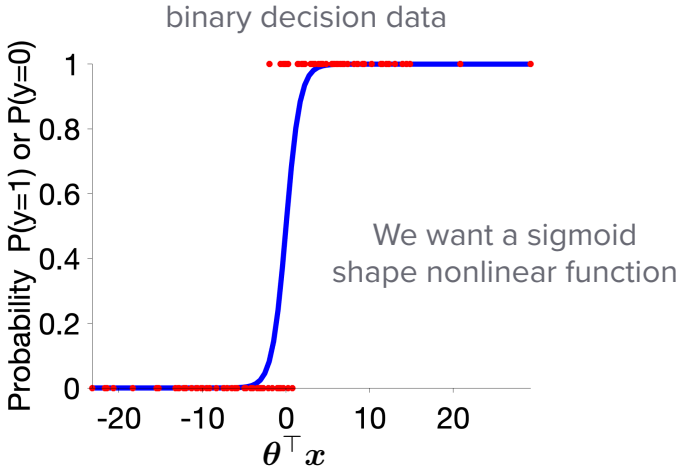



## Link function $f = ?$





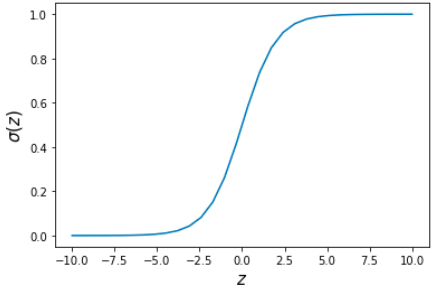

## Link function $f = ?$



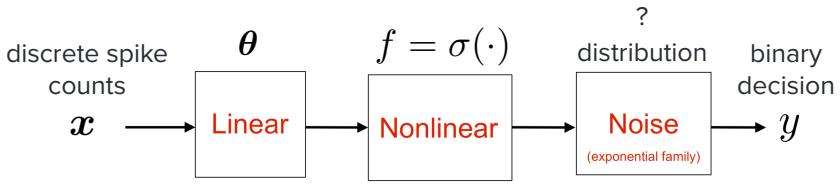

We want a sigmoid shape nonlinear function

We define $f = \sigma(\cdot)$ which is a "squashing" function called the sigmoid function.

$$\sigma(z) = \frac{1}{\exp(-z) + 1}$$




Notice $0 \leq f(\boldsymbol{\theta}^\top \boldsymbol{x}) = \frac{1}{\exp(-\boldsymbol{\theta}^\top \boldsymbol{x}) + 1} \leq 1$




## Distribution of the observation noise

$f(\boldsymbol{\theta}^\top \boldsymbol{x}) = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}) = \dfrac{1}{\exp(-\boldsymbol{\theta}^\top \boldsymbol{x}) + 1}$ only gives us a probability-like value, not a binary decision.

**Bernoulli distribution:** generate a binary value with some input probability value.

single coin flip

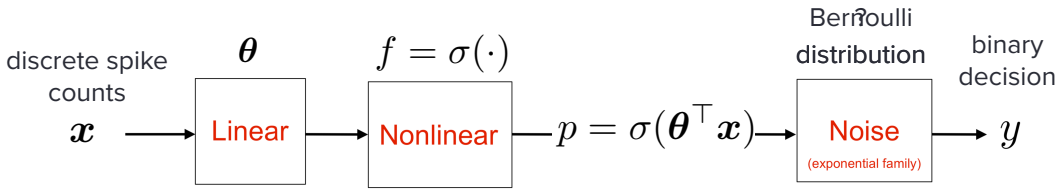
| outcome y: | head | tail |
|---|---|---|
| probability: | p | 1-p |

The probability mass function for y is

$$P(y|p) = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{if } y = 0 \end{cases}$$

Alternatively,

$$P(y|p) = p^y (1-p)^{1-y}$$




## Distribution of the observation noise

Given the probability mass function $P(y|p) = p^y(1-p)^{1-y}$

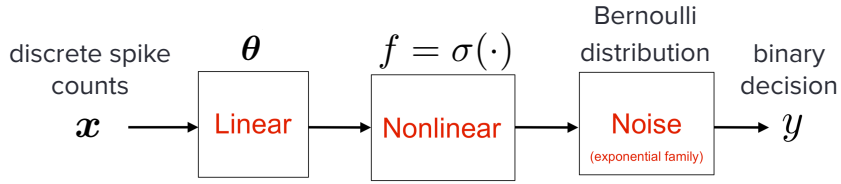
We can write down the decoding distribution as

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = P(y_i|p_i = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i)) = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i)^{y_i}(1 - \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i))^{1-y_i}$$

for all data points

$$p(\boldsymbol{y}|X, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \prod_{i=1}^{N} \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i)^{y_i}(1 - \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i))^{1-y_i}$$

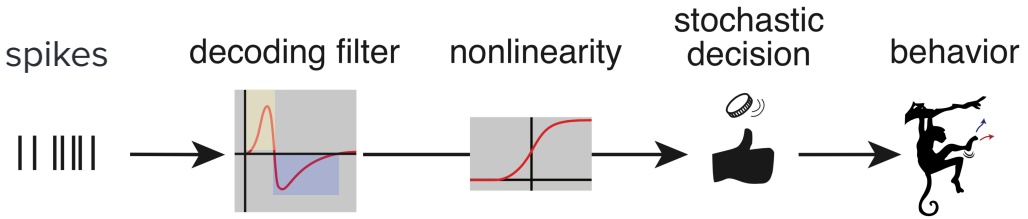
# Bernoulli GLM *(Logistic Regression)*




Decoding distribution $$p(\boldsymbol{y}|X,\boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i,\boldsymbol{\theta}) = \prod_{i=1}^{N} \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i)^{y_i}(1-\sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i))^{1-y_i}$$

log likelihood $$\log \mathcal{L}(\boldsymbol{\theta}|X,\boldsymbol{y}) = \log p(\boldsymbol{y}|X,\boldsymbol{\theta}) = \sum_{i=1}^{N} y_i \log \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i) + (1-y_i)\log\,(1-\sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_i))$$

- no closed form solution, need optimization
- **logistic regression**: closely related to linear regression, but for discrete outcomes.



# Different GLMs can solve different problems

| Output type | Likelihood | Nonlinearity | | |
|---|---|---|---|---|
| real values | Gaussian $\mathrm{P}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(y-\mu)^2}{2\sigma^2}$ | identity $\mu = \boldsymbol{\theta}^\top \mathbf{x}$ |  | Linear regression |
| discrete counts 0,1,2,3... | Poisson $\mathrm{P}(y) = \frac{\lambda^y \exp(-\lambda)}{y!}$ | exponential $\lambda = \exp\left(\boldsymbol{\theta}^\top \mathbf{x}\right)$ |  | Poisson GLM |
| binary 0,1 | Bernoulli $\mathrm{P}(y) = p^y (1-p)^{1-y}$ | logistic $p = \sigma\left(\boldsymbol{\theta}^\top \mathbf{x}\right)$ |  | Logistic regression (Bernoulli GLM) |

# Notebook

In the following notebook, you will

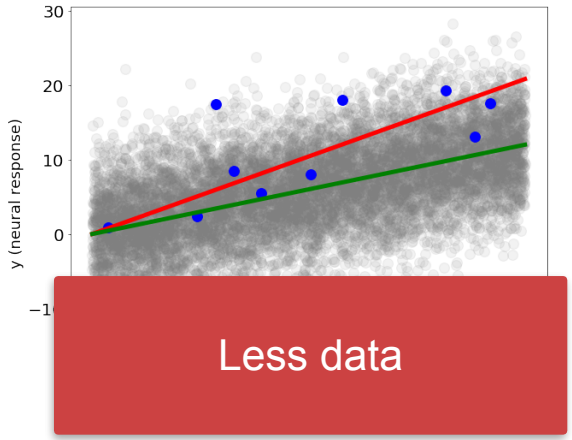
1. Inspect the sigmoid function.
2. Use Scikit-learn to fit the logistic regression model to a simulated decision making data.

Enjoy!!

# Regularization

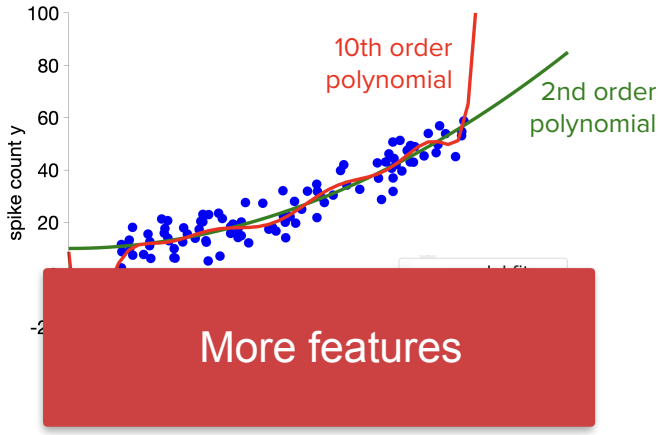
# Revisit Overfitting

Linear regression overfits when **data sample is small and different from true data distribution.**




Polynomial regression overfits when the polynomial order is high (**high model complexity**)

$$y = \sum_{p=0}^{P} \theta_p x^p$$



# Regularization

**Regularization** shrinks the parameters $\boldsymbol{\theta}$ in GLMs towards 0 — this can reduce overfitting.

Example: linear Gaussian model $\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\eta}$

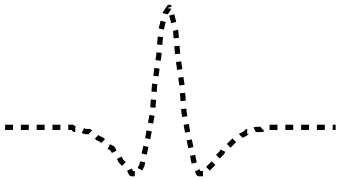

L2 (a.k.a. "ridge") regularization:

- penalize squared parameters
- suppress all features

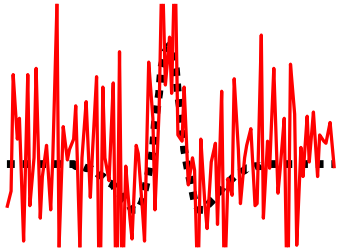
hyperparameter

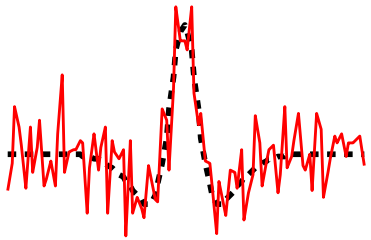
$$\log \mathcal{L}' = \log \mathcal{L} - \frac{\beta}{2} \sum_i \theta_i^2$$

true $\boldsymbol{\theta}$



$\boldsymbol{\theta}_{\text{MLE}} = (X^\top X)^{-1} X^\top \boldsymbol{y}$





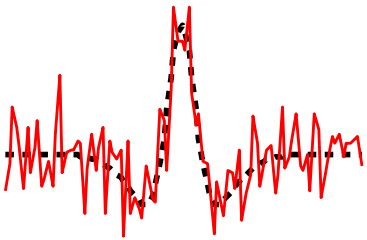
# Regularization for Linear Gaussian Model

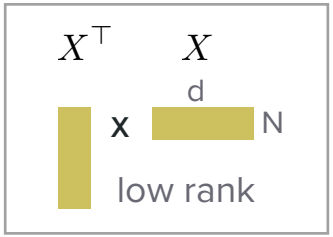
L2 (a.k.a. "ridge") regularization:

- penalize squared parameters
- suppress all features

hyperparameter

$$\log \mathcal{L}' = \log \mathcal{L} - \frac{\beta}{2}\sum_i \theta_i^2$$






Formally, if $\log \mathcal{L}$ is the log likelihood of linear Gaussian

$$\log \mathcal{L}'(\boldsymbol{\theta}|X,\boldsymbol{y}) = \log \mathcal{L}(\boldsymbol{\theta}|X,\boldsymbol{y}) - \frac{\beta}{2}\sum_i \theta_i^2$$

new quadratic term

$$= -\frac{Nd}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}(\boldsymbol{y}-X\boldsymbol{\theta})^\top(\boldsymbol{y}-X\boldsymbol{\theta}) - \frac{\beta}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}$$

linear Gaussian on D3

Differentiate and set to zero

$$\frac{\partial \log \mathcal{L}'(\boldsymbol{\theta}|X,\boldsymbol{y})}{\partial\, \boldsymbol{\theta}} = -\frac{1}{\sigma^2}X^\top(X\boldsymbol{\theta}-\boldsymbol{y}) - \beta\boldsymbol{\theta} = 0$$

$$\Rightarrow \boldsymbol{\theta}_{\text{Ridge}} = (X^\top X + \sigma^2\beta I)^{-1}X^\top\boldsymbol{y}$$

$$\boldsymbol{\theta}_{\text{MLE}} = (X^\top X)^{-1}X^\top\boldsymbol{y}$$

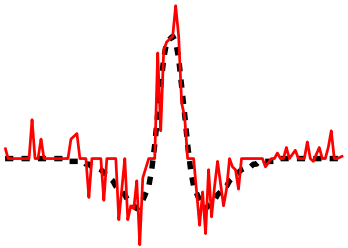
# Regularization for Linear Gaussian Model

L1 (a.k.a, “lasso”) regularization:

- penalize the absolute value of parameters
- encourages sparse parameters
- selects informative features

hyperparameter

$$\log \mathcal{L}' = \log \mathcal{L} - \frac{\beta}{2} \sum_i |\theta_i|$$



Formally, if $\log \mathcal{L}$ is the log likelihood of linear Gaussian model

$$\log \mathcal{L}'(\boldsymbol{\theta}|X, \boldsymbol{y}) = \log \mathcal{L}(\boldsymbol{\theta}|X, \boldsymbol{y}) - \frac{\beta}{2} \sum_i |\theta_i|$$

$$= -\frac{Nd}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} (\boldsymbol{y} - X\boldsymbol{\theta})^\top (\boldsymbol{y} - X\boldsymbol{\theta}) - \frac{\beta}{2} \sum_i |\theta_i|$$

no closed form solution, even for linear Gaussian need optimization

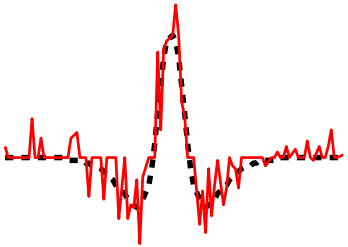
# Regularization for Linear Gaussian Model

L1 (a.k.a, “lasso”) regularization:

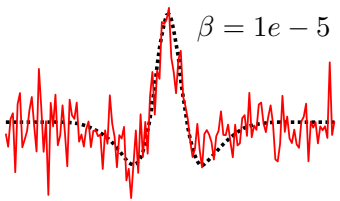
- penalize the absolute value of parameters
- encourages sparse parameters
- selects informative features

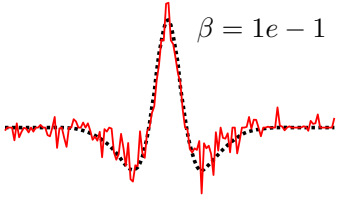
hyperparameter

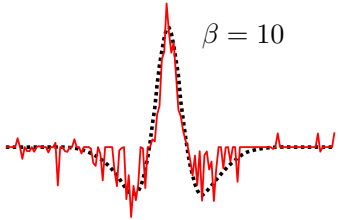
$$\log \mathcal{L}' = \log \mathcal{L} - \frac{\beta}{2} \sum_i |\theta_i|$$

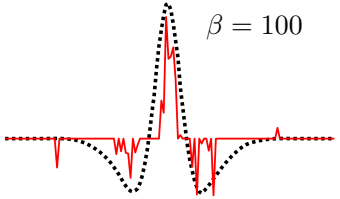


Different $\beta$ would lead to different sparsity levels



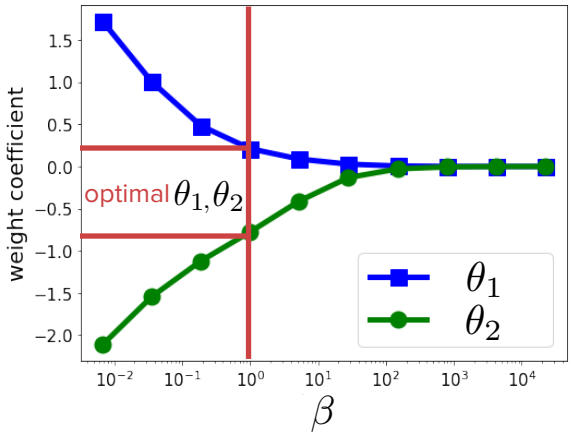



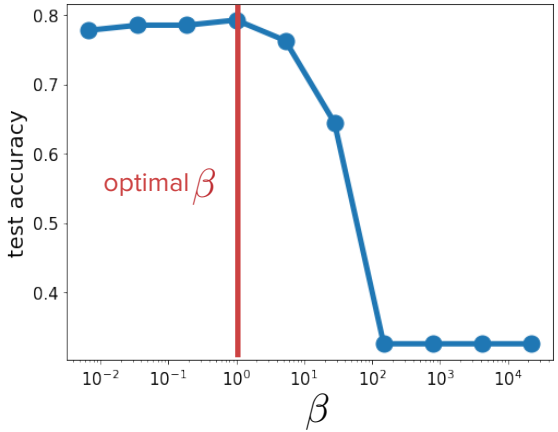
# Regularization for Logistic Regression

Similarly, we could add L2 (ridge) or L1 (lasso) penalty to the log likelihood of logistic regression.

logistic regression + **L2** regularization



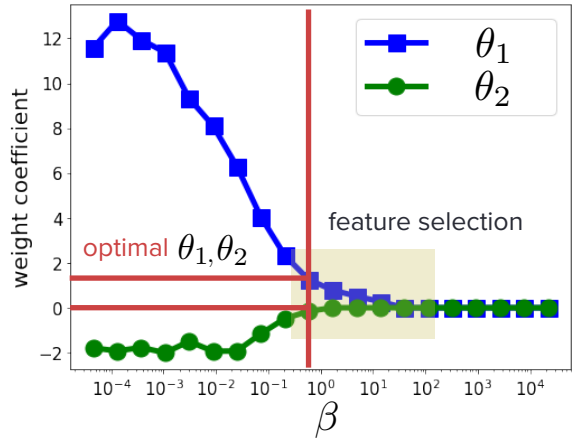

accuracy performance on test data



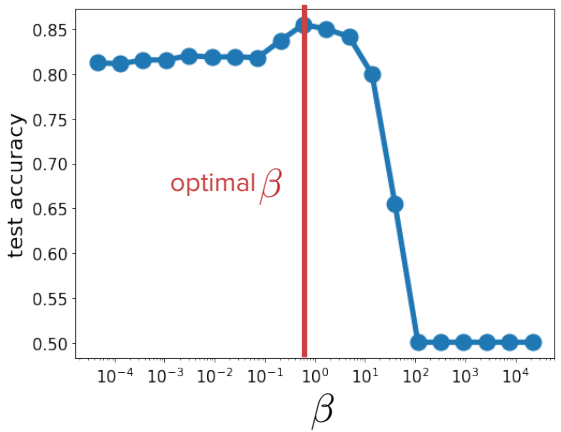
# Regularization for Logistic Regression

Similarly, we could add L2 (ridge) or L1 (lasso) penalty to the log likelihood of logistic regression.

logistic regression + **L1** regularization




accuracy performance on test data



# Notebook

In the following notebook, you will

1. See how the hyper parameter affects model fitting for logistic regression with L1 and L2 penalties.
2. Run cross validation to select the optimal hyperparameters for both L1 and L2.

Enjoy!!