Regression

Lucas Pinto
Assistant Professor
Department of Neuroscience
Northwestern University

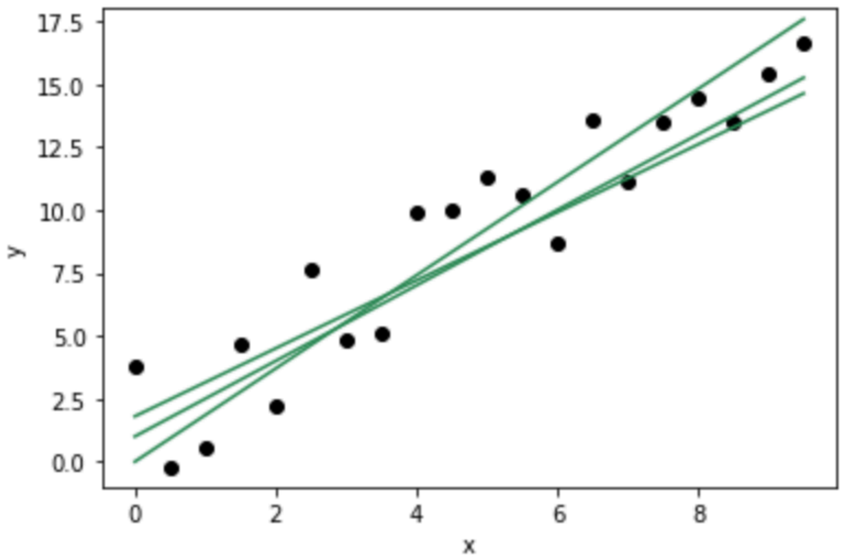
NUIN 443, 03/26/2024

Outline

- Linear regression
- Generalized linear models (GLMs)
- Mixed-effects models
- Preventing overfitting
- Predictor significance testing
- Parametrizing neuroscience data

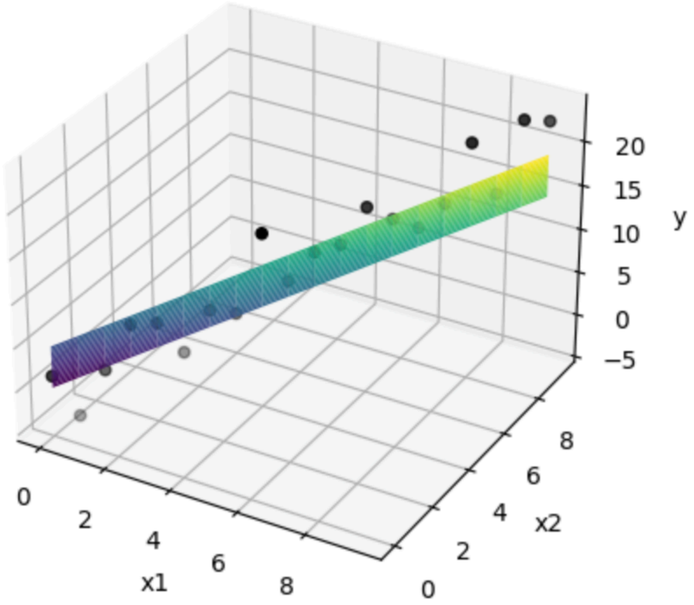
Linear regression

Linear regression intro



We want to estimate *y* by a linear function of the data *x*:

$$\hat{y} = \beta_0 + \beta_1 x$$



or, with more dimensions:

$$\hat{y} = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$$

For convenience
we will use matrix notation:

Predictor, design or input matrix

$$X = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{p,1} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_{1,n} & \dots & x_{p,n} \end{pmatrix}$$

n data points x *p* predictors
(*n* > *p*)

bias term is a column of ones since the offset is added to all data points

Weight vector

$$w = \begin{pmatrix} \beta_o \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

p x 1 predictors

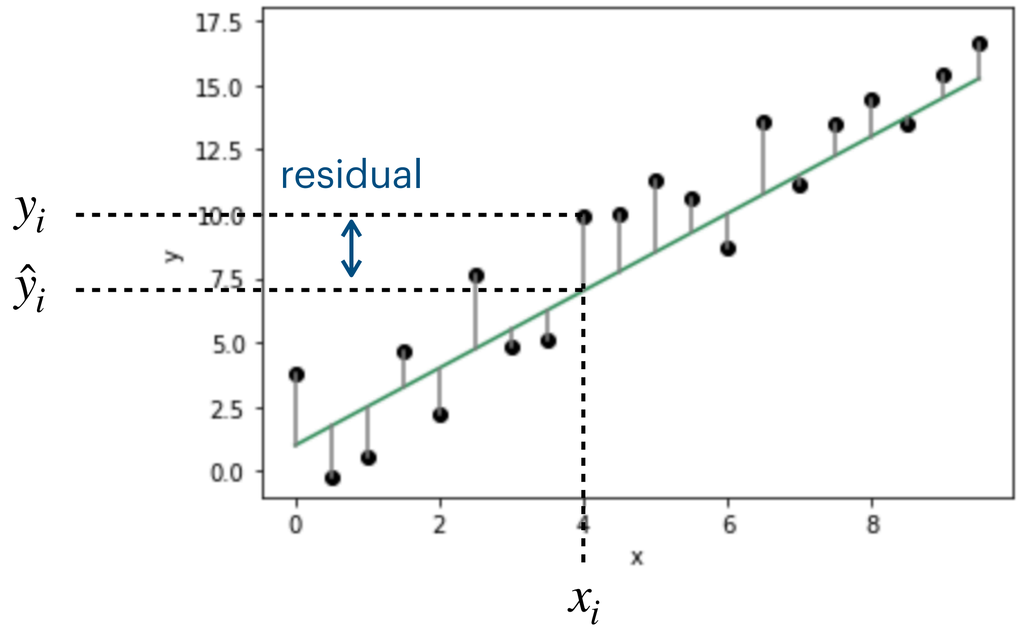
Response or output vector

$$\hat{y} = Xw$$

n x 1 data points

Cost function (aka loss): squared errors

But how do we choose among the many functions that approximate the data?



Residual sum of squares

We want to minimize:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Quadratic form guarantees that a minimum exists, but it may not be unique

Ordinary least squares

Under certain conditions, there is an analytical solution to minimizing *RSS*: OLS

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \sum_{i=1}^{n} (y_i - x_i\beta_i)^2$$

$$= ||y - Xw||^2$$

$$= (y - Xw)^T(y - Xw)$$

$$= y^Ty - (Xw)^Ty - y^T(Xw) + (Xw)^T(Xw)$$

$$= y^Ty - w^TX^Ty - y^TXw + w^TX^TXw$$

Since we want to minimize *RSS*, we can take the derivative w.r.t. *w* and set that to zero, obtaining the *normal equations*

$$\frac{\partial}{\partial w} RSS = -X^Ty - (y^TX)^T + 2X^TXw = 0$$

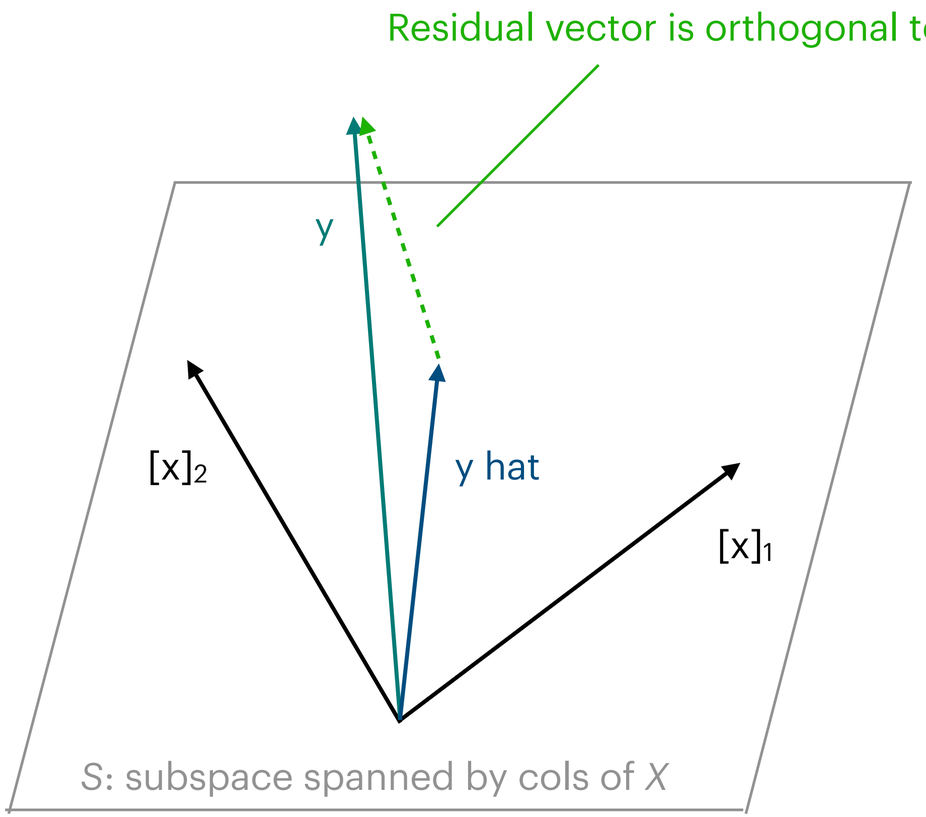
$$X^TXw = X^Ty$$

If X^TX is nonsingular, i.e. if $\det(X^TX)$ is nonzero, i.e. if all the columns of *X* are linearly independent, we can use OLS

$$w = (X^TX)^{-1}X^Ty$$

X corresponding to a singular X^TX is often called ill-conditioned

Geometric interpretation



Linear regression can also be thought of the projection of *y* onto the subspace spanned by the columns of *X*.

A projection matrix *P* is a square matrix that multiples a vector to project it onto a subspace spanned by the columns of *A*

$$P = A(A^T A)^{-1} A^T$$

$$w = (X^T X)^{-1} X^T y$$

$$\hat{y} = Xw$$

$$\hat{y} = X(X^T X)^{-1} X^T y = Py$$

Statistical interpretation

The data are assumed to follow a Gaussian distribution with mean $\hat{y}$

$$y_i \mid x_i \sim \mathcal{N}(\hat{y}, \sigma^2) \qquad \mathcal{N}(y, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

$$y_i \mid x_i \sim \mathcal{N}(Xw, \sigma^2)$$

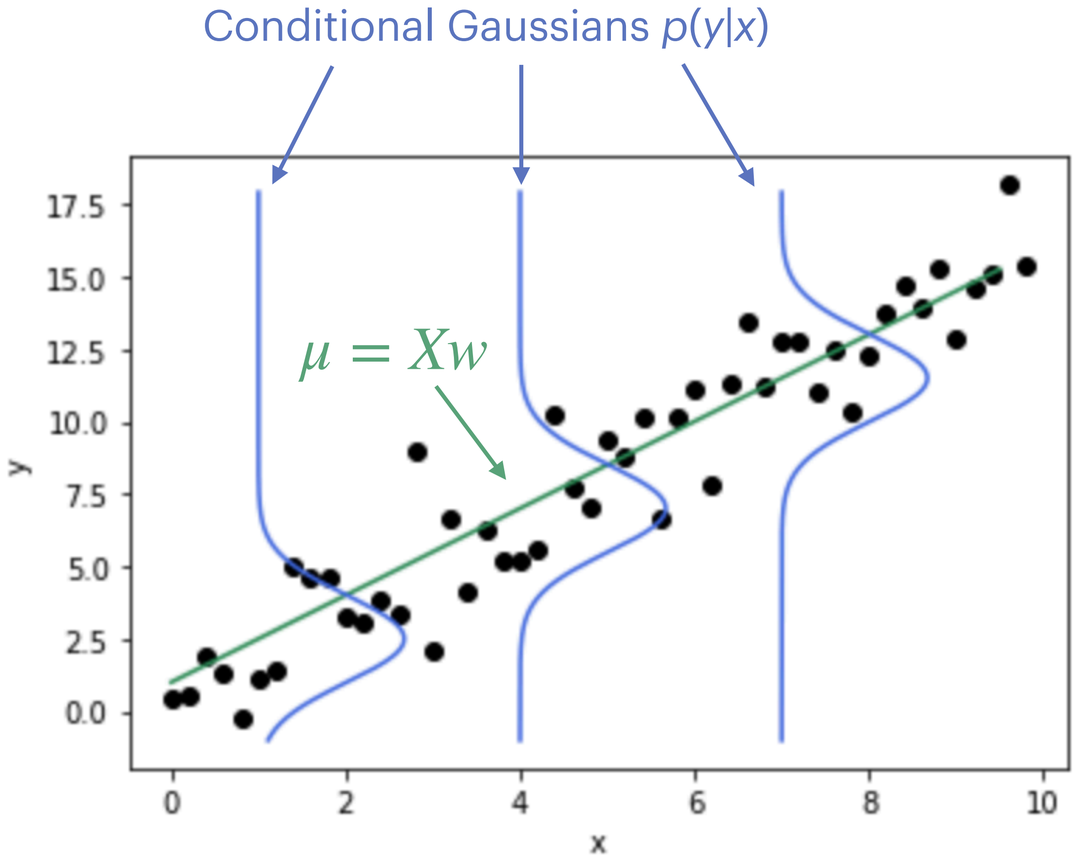
Because y_i are independent, we multiply their probability to obtain the likelihood function of the data

$$L(y \mid x, w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - x_i w)^2}{2\sigma^2}}$$

Take the log of the likelihood because sums are more convenient

$$\ell(y \mid x, w) = \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - x_i w)^2}{2\sigma^2}} \right)$$

$$= -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2} \boxed{\sum_{i=1}^{n} (y_i - x_i w)^2}$$

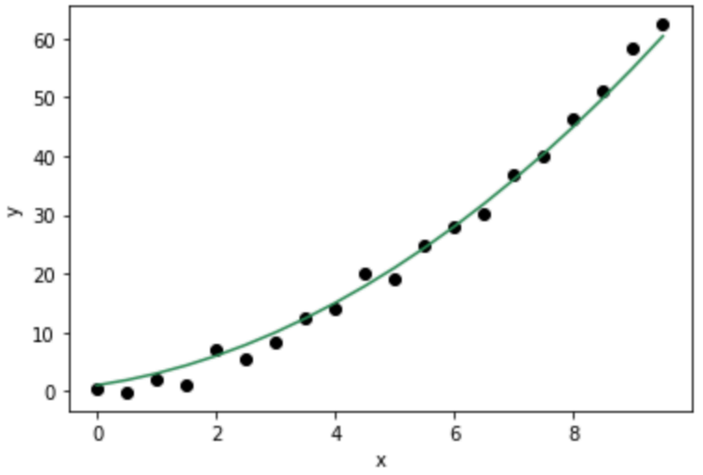


minimizing the residuals is equivalent to maximizing the likelihood of the model

Non-linear function approximators

- We can also approximate non-linear functions via non-linear transformations of the predictors themselves.
- For example, to approximate a quadratic function of x, we can add x^2 as an extra predictor.

$$X = \begin{pmatrix} 1 & x_{1,1} & x_{1,1}^2 \\ \vdots & \vdots & \vdots \\ 1 & x_{1,n} & x_{1,n}^2 \end{pmatrix}$$

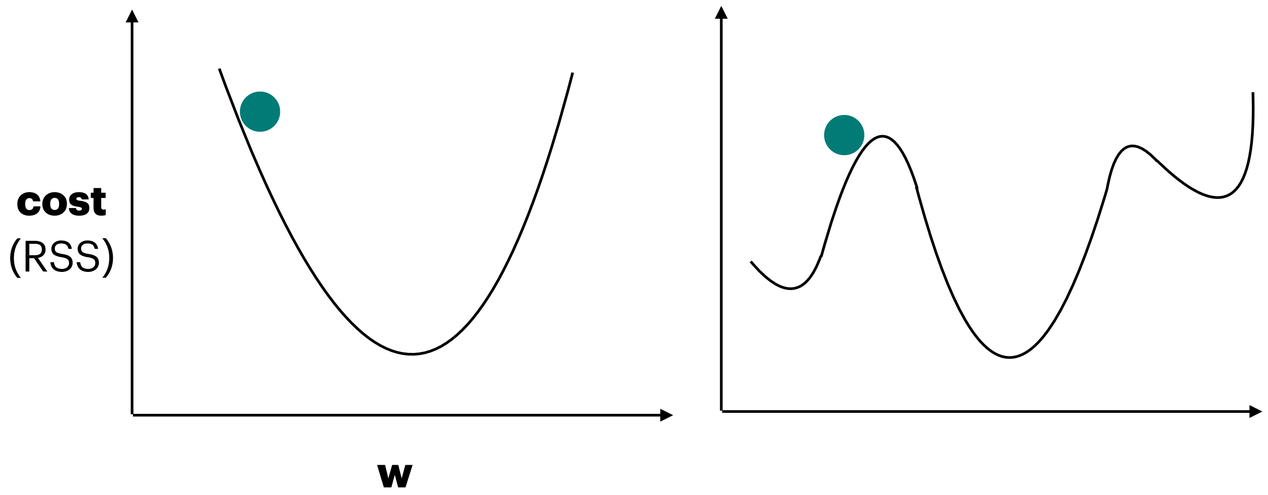


- In this case, $\hat{y}$ is still linear in X, and the math holds.
- A common use of this in neuroscience is basis functions. We will revisit this when we talk about model parametrization.

Gradient descent

- OLS is not feasible in most real-world data analysis scenarios. For one, inverting X^TX is very computationally expensive and unfeasible for large X's. The most common solution is to do gradient descent on the likelihood function (MLE).
- Gradient descent can also deal with cases where the columns of *X* are not fully independent. DISCOURAGED. You can run the fit but not fully trust the results.
- Best to condition X properly. Regularization also helps (back to this later)
- In practice, the most common implementation is *stochastic gradient descent*, which computes the gradient with respect to a random subset of the data for computational efficiency
- Gradient descent is a general algorithm, i.e. not restricted to regression. It can be used as long as the cost function is differentiable.

Parameter error landscape



The least mean squares algorithm

1. Iteratively compute the gradient (derivative) of the cost function (*RSS*) w.r.t. the regression weights

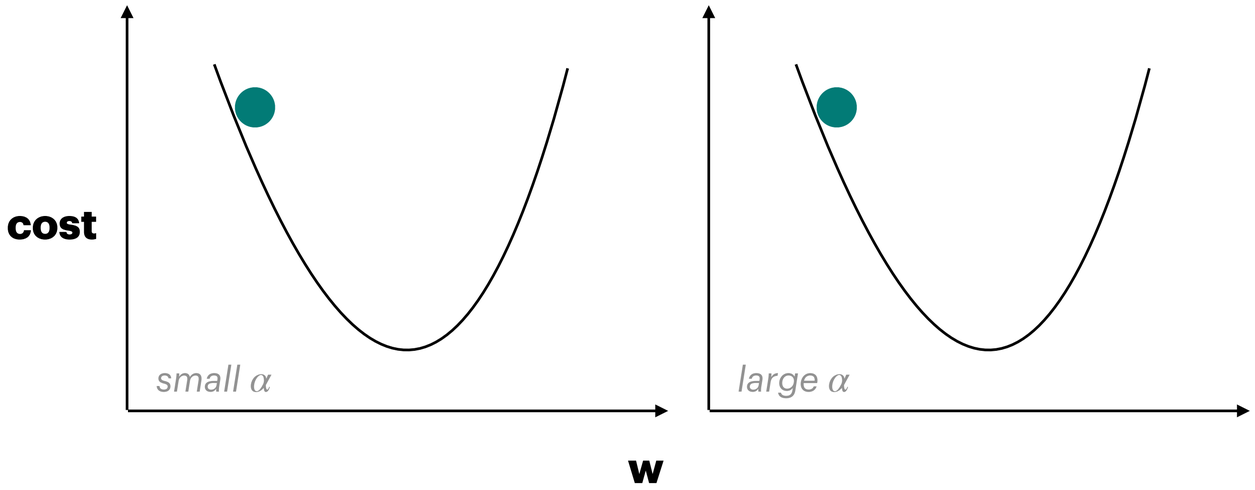
$$\frac{\partial}{\partial w} RSS = X^T(Xw - y)$$

2. Update the weights by moving them in the direction opposite to the steepest gradient (thus the negative sign)

$$w^{t+1} = w^t - \alpha \frac{\partial}{\partial w} RSS$$

$$w^{t+1} = w^t - \alpha X^T(Xw - y)$$

The amount of movement is additionally controlled by multiplying by α, a.k.a. *learning rate*, an important hyperparameter



3. Stop once you have reached criterion, typically some threshold for the magnitude of gradient change ϵ or maximum number of iterations

Generalized linear models

Least squares assumptions are often violated in neuroscience data

Residuals ϵ are normally distributed

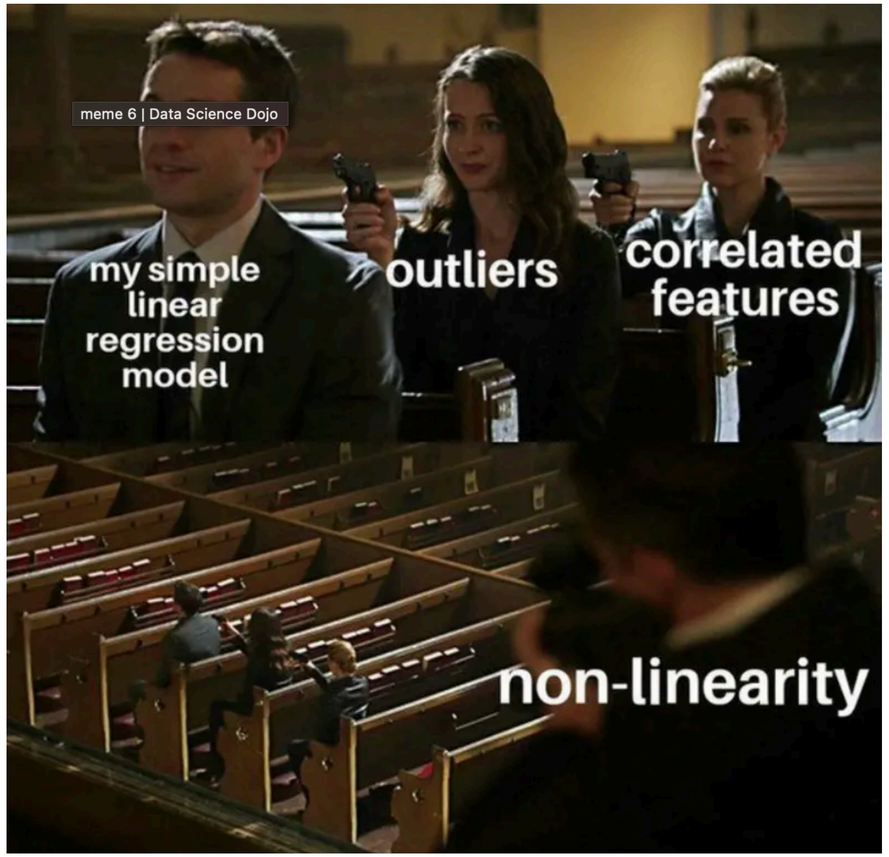
$$\epsilon \sim \mathcal{N}(0, \sigma^2 I_n)$$

With 0 mean (exogeneity)

With spherical errors:

Homoscedasticity: every observation has the same variance σ^2

Errors are uncorrelated (thus multiplication by identity matrix I_n)



GLM = Linear model + link function + noise distribution

In OLS, the conditional probability of an observation of the response variable $y_i$follows a normal distribution with mean $\hat{y}_i$. That is, OLS predicts the mean of a conditional Gaussian.

$$y_i \mid X_i \sim \mathcal{N}(\hat{y}_i, \sigma^2)$$

$$y_i \mid X_i \sim \mathcal{N}(X_i w, \sigma^2)$$

GLMs are a generalization that predicts the conditional means of some other distribution *P*. The means, in turn, are obtained via a function of the predictors, *f()*, s.t. *y* is linear in *f(Xw)* rather than *Xw*

$$y_i \mid X_i \sim P(f(X_i w))$$

Thus, two key ingredients:
1. *Noise distribution P*, coming from the exponential family (e.g., Gaussian, Gamma, Poisson, Bernoulli, Exponential)

2. A *link function g()*, such that:

$$E(y_i \mid X_i) = \hat{y}_i = \mu$$

$$g(\mu) = X_i w$$

$$\mu = g^{-1}(X_i w)$$

$g^{-1}()$ is the *mean function*, the inverse of the link

Note that OLS is just a case of GLM with link = identity and Gaussian noise

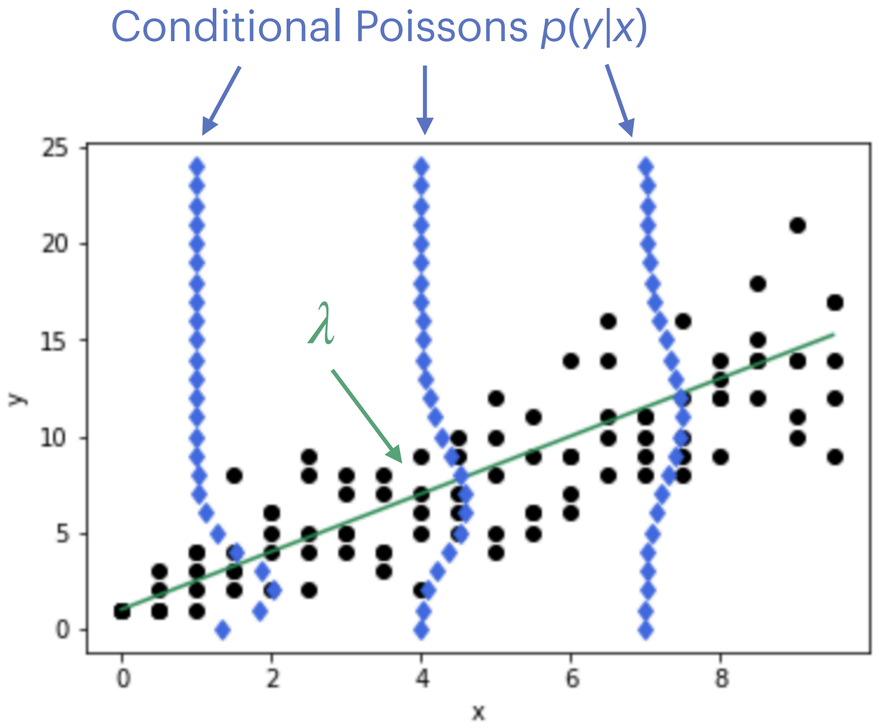
Case 1: Poisson GLM

Data *y*: non-negative integers
Example: spike counts
Link function: $g(\lambda) = \log(Xw)$ (Log enforces *y*>0)

Mean function: $\lambda = e^{Xw}$

Noise distribution: $Poisson(y \mid \lambda) = \frac{e^{-\lambda}\lambda^{y}}{y!}$

Statistical interpretation: $y_i \mid x_i \sim Poisson(e^{x_i w})$



Optimization:

GLMs are typically fit with maximum likelihood estimation (MLE) (i.e., minimizing the negative log likelihood).

$$L(y \mid x, w) = \prod_{i=1}^{n} \frac{\lambda_i^{y_i} e^{-\lambda_i}}{y_i!}$$

Again, log likelihood is more convenient

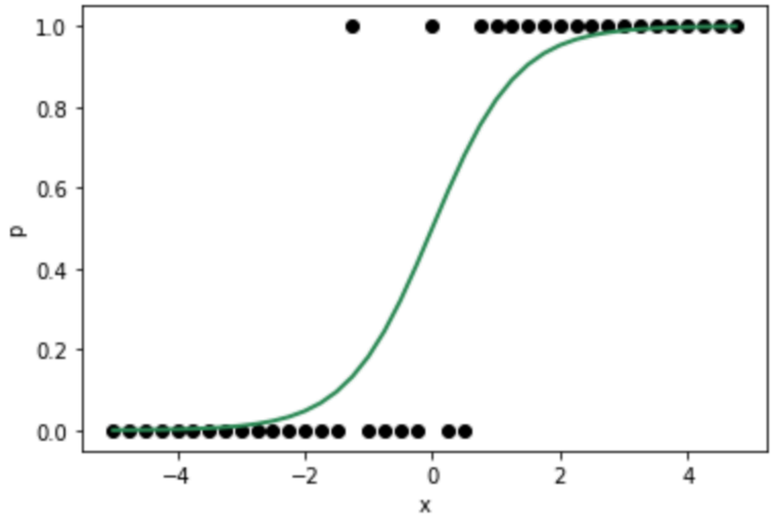
$$\begin{aligned}\ell(y \mid x, w) &= \sum_{i=1}^{n} y_i \log(\lambda_i) - \lambda_i - \log(y_i!) \\ &= \sum_{i=1}^{n} y_i \log(e^{x_i w}) - e^{x_i w} - \log(y_i!) \\ &= \sum_{i=1}^{n} y_i x_i w - e^{x_i w} - \log(y_i!)\end{aligned}$$

Take partial derivatives w.r.t. to weights and set to 0

$$\frac{\partial}{\partial w}\ell = \sum_{i=1}^{n} x_i(y_i - e^{x_i w}) = 0$$

Case 2: Bernoulli GLM (a.k.a. logistic regression)

Data y: binary variables {0,1}
Example: behavioral choices in 2-AFC tasks



Link function: $\log \frac{P(y=1)}{P(y=0)} = \log \frac{P(y=1)}{1-P(y=1)} = \log \frac{\mu}{1-\mu} = Xw$ (log odds ratio)

Mean function: $\mu = P(y=1 \mid X) = \frac{1}{1+e^{-Xw}}$ (logistic function)

Noise distribution: Bernoulli (some packages do Binomial, it's equivalent)

$$Bernoulli(y \mid p) = \begin{cases} p & \text{if } y = 1 \\ q = 1-p & \text{if } y = 0 \end{cases} = p^y (1-p)^{1-y}$$

Optimization:

$$L(y \mid w) = \prod_{i=1}^{n} p_i^{y_i} (1-p_i)^{1-y_i} = \prod_{i=1}^{n} p_i^{y_i} \frac{(1-p_i)}{(1-p_i)^{-y_i}} = \prod_{i=1}^{n} (1-p_i) \left(\frac{p_i}{1-p_i}\right)^{y_i}$$

$$\ell(y \mid w) = \sum_{i=1}^{n} \underbrace{\log(1-p_i)}_{= 1/1+e^{X_i w}} + y_i \underbrace{\log\left(\frac{p_i}{1-p_i}\right)}_{= X_i w} = \sum_{i=1}^{n} y_i X_i w - \log(1+e^{X_i w})$$

$$\frac{\partial}{\partial w} \ell = \sum_{i=1}^{n} X_i \left(y_i - \frac{e^{X_i w}}{1+e^{X_i w}} \right) = 0$$

$$f(x) = \log(1+e^x)$$

$$f'(x) = \frac{e^x}{1+e^x} = 1 - \frac{1}{1+e^x}$$

Link function & noise distribution cheat sheet

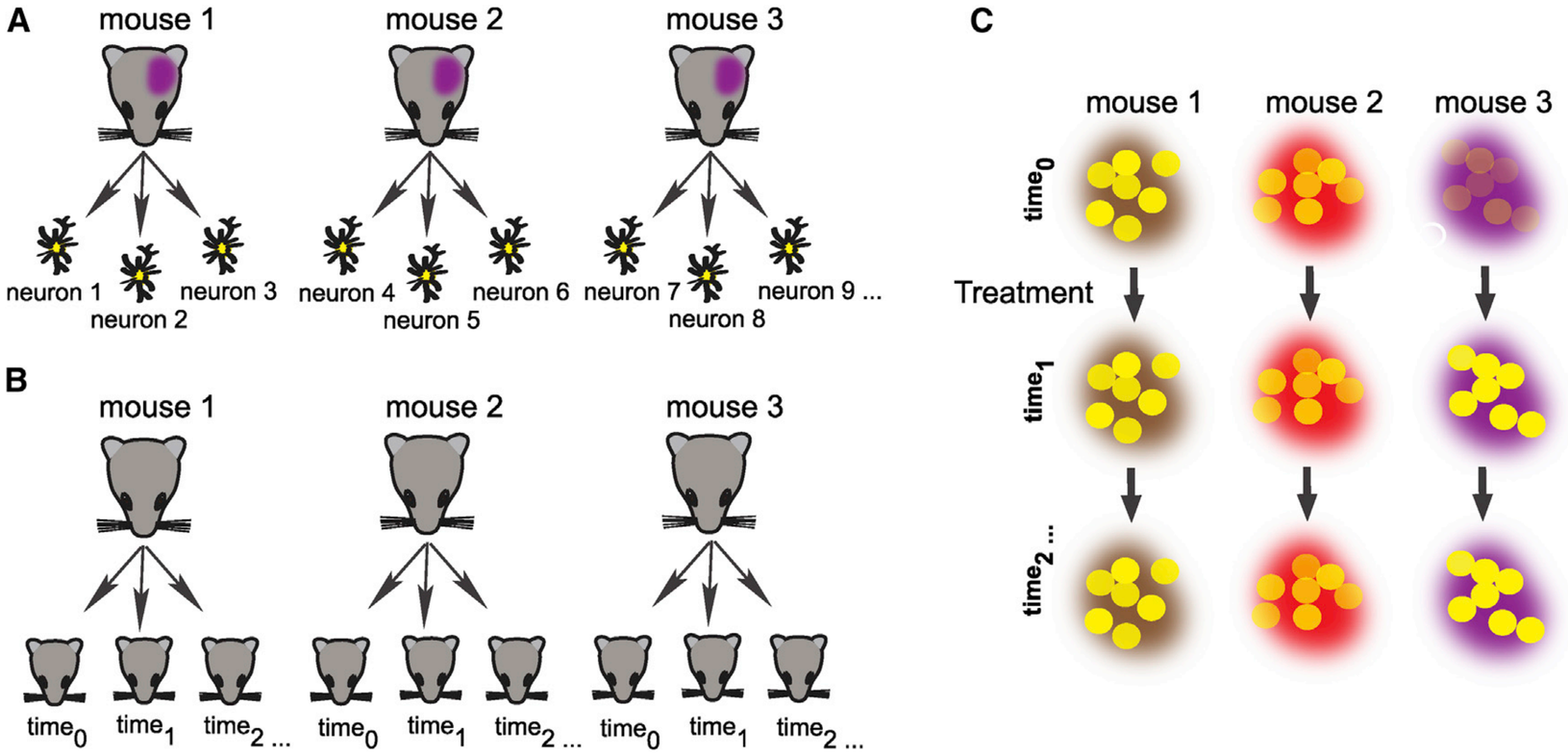
Common distributions with typical uses and canonical link functions

Distribution	Support of distribution	Typical uses	Link name	Link function, $\mathbf{X}\boldsymbol{\beta} = g(\mu)$	Mean function
Normal	real: $(-\infty, +\infty)$	Linear-response data	Identity	$\mathbf{X}\boldsymbol{\beta} = \mu$	$\mu = \mathbf{X}\boldsymbol{\beta}$
Exponential Gamma	real: $(0, +\infty)$	Exponential-response data, scale parameters	Negative inverse	$\mathbf{X}\boldsymbol{\beta} = -\mu^{-1}$	$\mu = -(\mathbf{X}\boldsymbol{\beta})^{-1}$
Inverse Gaussian	real: $(0, +\infty)$		Inverse squared	$\mathbf{X}\boldsymbol{\beta} = \mu^{-2}$	$\mu = (\mathbf{X}\boldsymbol{\beta})^{-1/2}$
Poisson	integer: $0, 1, 2, \ldots$	count of occurrences in fixed amount of time/space	Log	$\mathbf{X}\boldsymbol{\beta} = \ln(\mu)$	$\mu = \exp(\mathbf{X}\boldsymbol{\beta})$
Bernoulli	integer: $\{0, 1\}$	outcome of single yes/no occurrence	Logit	$\mathbf{X}\boldsymbol{\beta} = \ln\left(\frac{\mu}{1-\mu}\right)$	$\mu = \frac{\exp(\mathbf{X}\boldsymbol{\beta})}{1+\exp(\mathbf{X}\boldsymbol{\beta})} = \frac{1}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}$
Binomial	integer: $0, 1, \ldots, N$	count of # of "yes" occurrences out of N yes/no occurrences		$\mathbf{X}\boldsymbol{\beta} = \ln\left(\frac{\mu}{n-\mu}\right)$	
Categorical	integer: $[0, K)$	outcome of single *K*-way occurrence		$\mathbf{X}\boldsymbol{\beta} = \ln\left(\frac{\mu}{1-\mu}\right)$	
	K-vector of integer: $[0, 1]$, where exactly one element in the vector has the value 1				
Multinomial	*K*-vector of integer: $[0, N]$	count of occurrences of different types (1, ..., *K*) out of *N* total *K*-way occurrences			

https://en.wikipedia.org/wiki/Generalized_linear_model

Mixed-effects models

Clustered experimental design in neuroscience leads to many sources of correlations



Yu et al., 2022, *Neuron*

LME (linear mixed effects) **& GLMM** (generalized ...)

- Mixed effect models jointly estimate group-level *fixed effects* (the familiar *w*), and a set of *random effects u*, attributed to correlated variables that are not deemed relevant for the model (e.g., mouse ID, treatment day)

  $\hat{y} = Xw + Zu$ (LME) or $\hat{y} = P(f(Xw + Zu))$ (GLMM)

  Where it is typically assumed that $u \sim \mathcal{N}(0,\sigma^2)$. Thus, *u* contributes to variance around the fixed effects *w*.

- Note that in this simplest case *Z* is usually a *n x m* matrix of dummy variables indicating instances of the random effect, where *n* is the total number of data points and *m* is the number of instances (e.g. number of mice). In this case, the random effects only contributed *random intercepts* to the estimate of *y*.

$$Z = \begin{pmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{pmatrix}$$

- We can also add random slopes to the model by having model predictors for each of the *m* instances

  $\hat{y} = Xw + Zu + \sum_{i=1}^{m} X_m w_m$ (Note that in this formulation X_m should not have an intercept column)

- Naturally, the model can contain multiple sources of random effects.

- Downside: ME models can add many more parameters to be estimated, require more data.

- Given the gaussian assumption, GLMM can be optimized with MLE, but many packages instead use Bayesian estimation methods that Josh Glaser will cover

Preventing overfitting

L2 regularization: ridge regression

Especially in models with partially correlated predictors (very common in neuro), weights can acquire very large values to compensate for each other. Ridge regression improves that by penalizing the L2 norm of *w*. This is done by minimizing

$$||y - Xw||^2 + \lambda ||w||^2, \lambda \geq 0$$

where λ is a hyperparameter typically found with cross-validation

The ridge solution depends on weight magnitude, so predictors should be centered and normalized (z-scored). Also, we do not want to penalize the intercept, so we estimate it as the average of *y*, subtract the average from y and remove the column of 1s from *X*

$$y_s = y - \bar{y},\ X_s = (X - \bar{X})/std(X)$$

$$RSS(\lambda) = (y_s - X_s w)^T (y_s - X_s w) + \lambda w^T w$$

We can rearrange this into the familiar form

$$\hat{w}(\lambda) = (X_s^T X_s + \lambda I)^{-1} X_s^T y$$ where *I* is an identity matrix

this matrix is full rank even if X^TX isn't!
(by adding a constant to each column of X^TX)

Also, if X^TX is full rank, then $\hat{w}(\lambda) = \hat{w}/(1 + \lambda)$

L1 regularization: LASSO

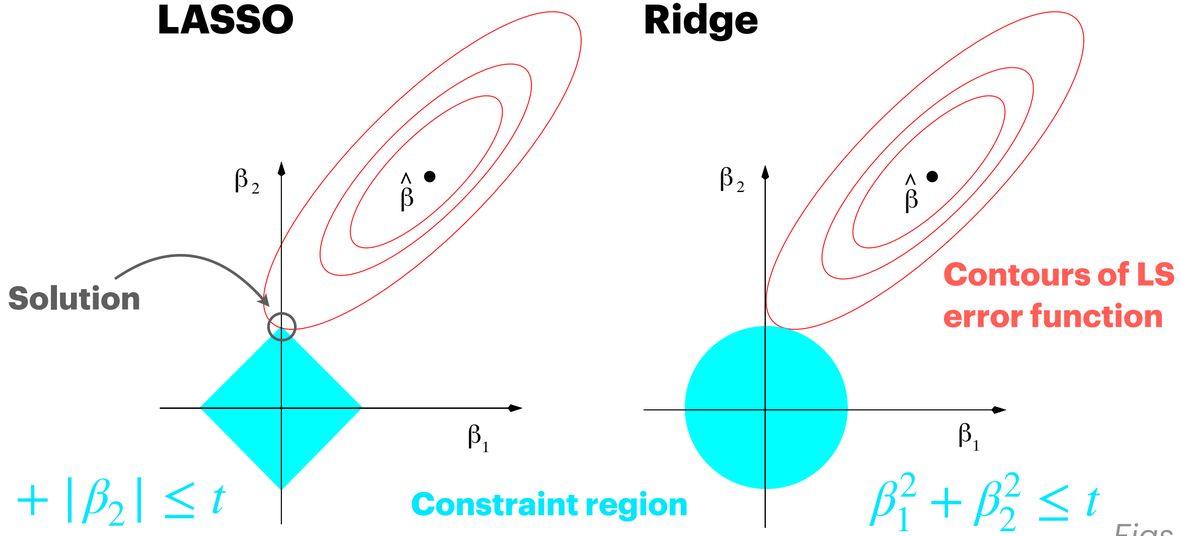
A related approach is to penalize the L1 norm of *w*. This is done by minimizing

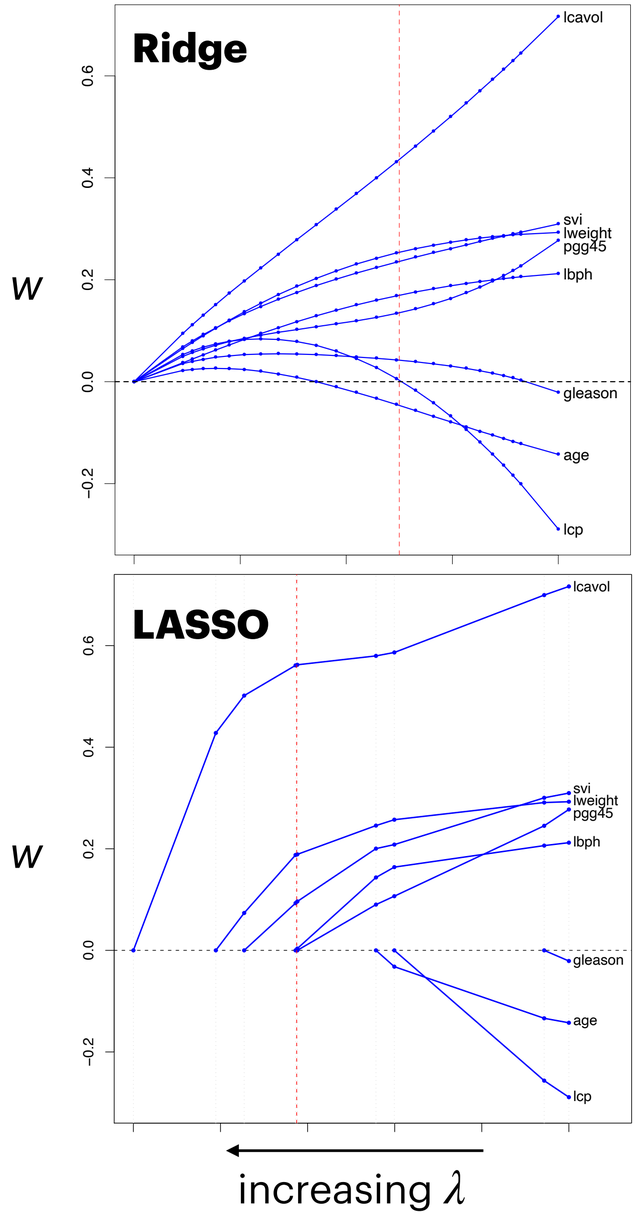
$$||y - Xw||^2 + \lambda |w|, \lambda \geq 0$$

where λ is again usually found with cross-validation

Taking the absolute value means that, unlike ridge, the solution for *w* is not linear in *y*, and cannot be derived analytically

LASSO is effectively a *sparseness constraint*



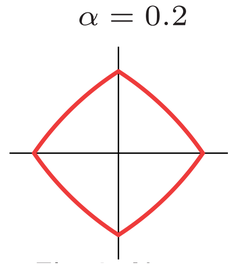


Figs from Hastie et al

Regularization: other methods

- Elastic net is a compromise between Ridge and LASSO with the penalty:

$$\lambda \sum_{j=1}^{p} \alpha w_j^2 + (1-\alpha)\,|w_j|,\ \lambda \geq 0$$



- Variable selection methods, e.g. Principal components regression:

1. Do PCA on X^TX to obtain matrix of eigenvectors V
2. Regress y against the m highest variance components of V, $m < p$
3. Project w back onto X via PC loadings

- Implicit methods, e.g., early stopping, dropout RNN training

Cross-validation

- How generalizable is the model? -> train and test on independent samples
- Particularly important for small *n* and/or large *p*

Wikipedia

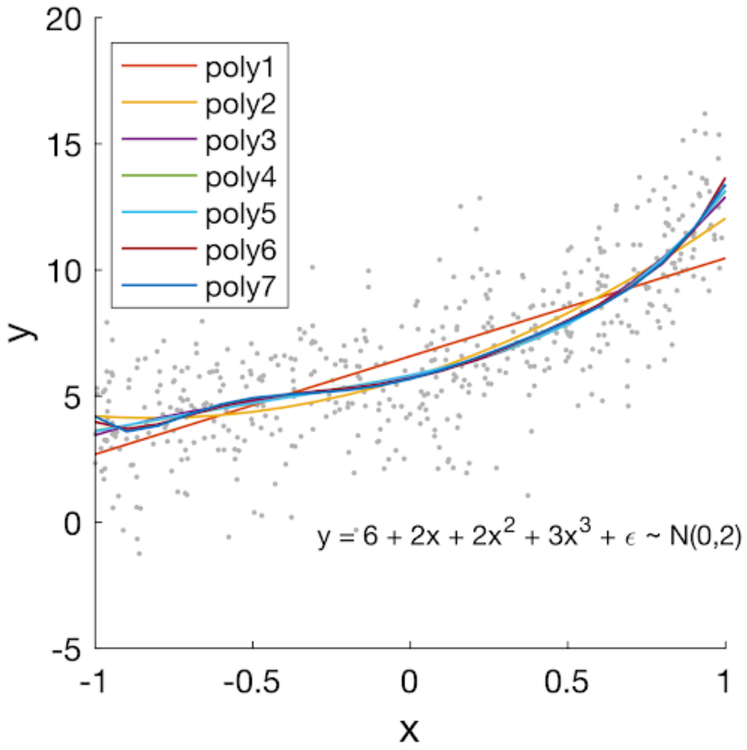
- Overall performance is some aggregate over iterations, e.g. mean or max
- Independence is important here. For example, calcium imaging data are temporally correlated due to sensor dynamics. Best to cross-validated at the level of trials instead of time points

Types:

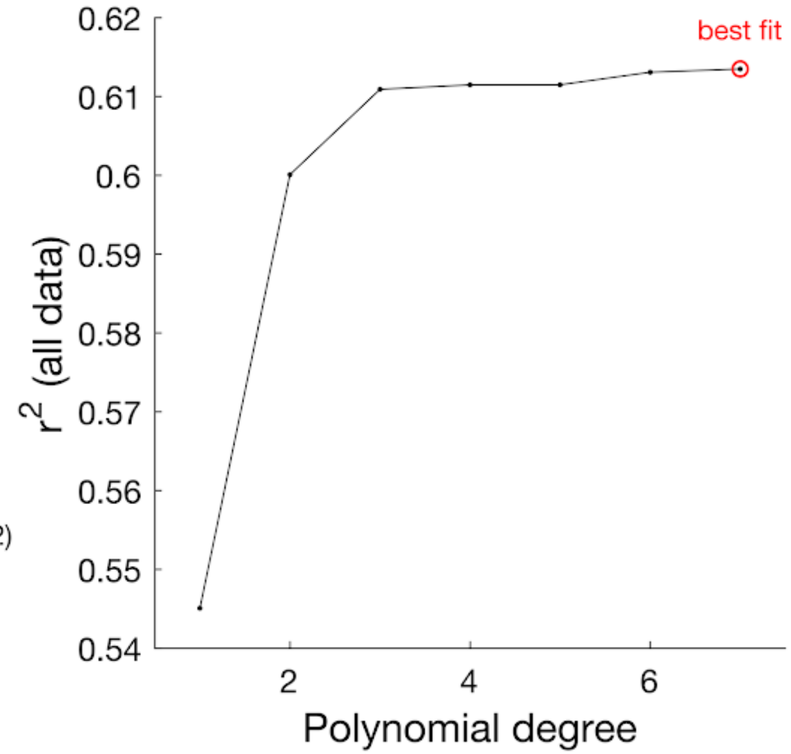
- *Leave-one-out*: drop one data point at a time, iterate over all data points (good for small *n*)
- *Leave-p-out*: generalization of leave-one-out
- *k*-fold: split data into *k* subsets, train on *k*–1 and test on 1. Do it *k* times. *k* = 10 is a popular choice, but depends on several factors, esp. resulting training set *n* w.r.t. *p*
- *Monte Carlo*: like *k*-fold, but with multiple random data splits instead of k. More robust and great for creating distributions for model comparison

Cross-validation: an illustration

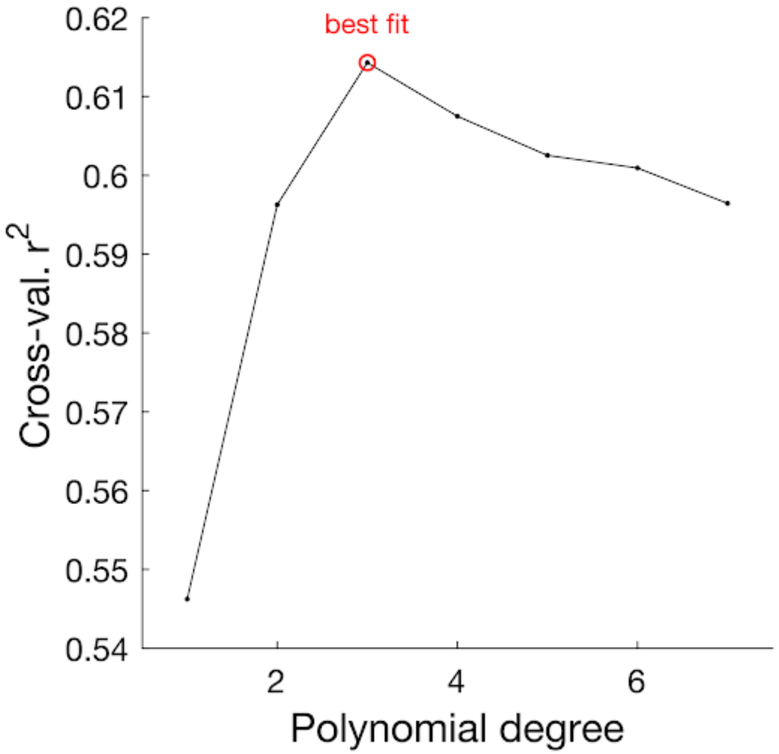
Data generated from a 3rd degree polynomial, fit with degrees 1 -



Without cross-validation



With cross-validation



Quantifying goodness of fit

▸ Coefficient of Determination

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Pros: intuitive (a perfect model will capture 100% of variance, $R^2 = 1$)
Cons: - only well behaved in OLS with an intercept term. E.g., can be negative in cross-validation
- does not account for # parameters

▸ Adjusted R^2

$$\bar{R}^2 = 1 - \frac{RSS/df_{res}}{TSS/df_{tot}} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

▸ Root mean squared error

$$RMSE = \sqrt{\frac{1}{n}RSS}$$

Pros: well behaved
Cons: - counterintuitive, magnitude depends on the scale of the data
- does not account for # parameters

▸ AIC (Akaike information criterion)

$$AIC = 2p - 2\ell$$

Pros: - related to log likelihood ℓ
- takes into account # parameters *p*
- Asymptotically equivalent to leave-one-out cross-val (Stone, *J R Stat Soc Series B Stat Methodol*, 1997)
Cons: - in cross-val, it's overkill to account for *p*
- only works for *n* >> *p*
- can only be used to compare models fit to identical *Y*
- can only be interpreted relative to AIC for other models (smaller the better):

$$Relative\ Likelihood = e^{(AIC_{min} - AIC_i)/2}$$

- proportional prob. that ith model maximizes ℓ

▸ BIC (Bayesian info criterion)

$$BIC = 2p\log(n) - 2\ell$$

Pros: - more stringent than AIC
Cons: - see AIC

▸ Correlation coefficient

$$r = corr(y, \hat{y}) = \frac{y^T\hat{y}}{\sigma_y\sigma_{\hat{y}}}$$

Pros: - intuitive
- easy to compute
- well behaved in cross-val
Cons: - does not account for # parameters (but cross-val)
- not strictly related to captured variance:

$$r^2 = corr(y, \hat{y})^2$$

- in OLS with an intercept term, = R^2 , *but not* in GLMs

▸ Model Information Index

$$MI = \frac{1}{n}\frac{\ell - \ell_0}{log(2)}$$, where $\ell_0 = \ell(Y|\bar{y})$

(Paninski et al, 2004, *J Neurosci*, 24:8551)
Pros: - units of bits / datapoint
- directly related to model likelihood
Cons: - harder to compute

Testing predictor significance

t and F statistics (OLS case)

t test for the null hypothesis that $w_j = 0$

- Assumptions: $\epsilon \sim \mathcal{N}(0,\sigma^2 I_n)$; w and σ^2 are independent

$$var(\hat{w}) = (X^T X)^{-1}\sigma^2 \text{ , where } \hat{\sigma^2} = \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(from plugging $(\hat{w}-w)^T(\hat{w}-w)$ in the OLS equation)

$$\hat{w} \sim \mathcal{N}(w, (X^T X)^{-1}\sigma^2)$$

$$z_j = \frac{\hat{w}_j}{\hat{\sigma}\sqrt{v_j}} \sim t_{n-p=1}$$, where v_j is jth diagonal element of $(X^T X)^{-1}$

$$z_j \to \mathcal{N} \text{ as } n \to \infty$$

- Large n tends to inflate significance

Confidence interval for w_j

$$(\hat{w} - z^{(1-\alpha)}\sqrt{v_j}\hat{\sigma} \text{ , } \hat{w} + z^{(1-\alpha)}\sqrt{v_j}\hat{\sigma})$$

where $z^{(1 - \alpha)}$ is *the* $(1 - \alpha)$ percentile of a Gaussian,
e.g.: $z^{(1 - 0.025)} = 1.96$

F test for nested models

Two models *a* and *b*, where model *b* contains a subset of the parameters in model *a*

$$F = \frac{(RSS_b - RSS_a)/(p_a - p_b)}{RSS_a - n - p_a - 1}$$

$$F \sim F_{p_a - p_b, n - p_a - 1}$$

The null hypothesis is that the models are not different, and a significant *F* indicates that the dropped parameters in model *b* are significant

Indeed, doing some simple plugging-in math, we can also show that the z_j statistic is equivalent to the *F* statistic for a model *b* that is only missing predictor *j*.

Dropping predictors

- Similar ideas can be applied in a non-parametric way, an increasingly popular approach in neuroscience
- For any GLM, you can just drop (or shuffle) the predictor and reassess goodness of fit
- Two possibilities:

1. Zero out the weight or shuffle the predictor, recompute $\hat{y}$ from the new Xw without refitting
   - In real world scenarios, cols of X may be partially correlated. If dropped predictor is one of these variables, not refitting may lead to overestimation of its significance (regularizing helps!)
   - Ceiling of a predictor's contribution

2. Same as 1, but with refitting
   - Refitting allows partially correlated variables to absorb variance that was attributed to dropped predictor
   - Floor of a predictor's contribution
   - More conservative approach, but more computationally expensive

- To obtain distributions for how well these different models do, we can just bootstrap the data (sample w/ replacement)
- The best is to this within a cross-validation framework
- The fairest comparison is to use exactly the same *y* for different models (set your random seeds!)

By the way

- We can use analogous methods to compare the *magnitude* of weights, as long as they are on the same scale
- Which reminds me: *always* z-score *X*!

Parametrizing neuroscience data

Neural data

- The activity of any given neuron or brain area is influenced by a number of measured and latent factors. Even in V1, neurons not only respond to visual stimuli, but also other sensory modalities, and their activity is modulated by the animal's behavioral state. They are also impacted by the firing of other neurons (beyond measured experimental parameters).
- They also have dynamics: responses to discrete events have delays and extend in time
- Given enough data, GLMs with multiple measured experimental parameters are a great way to estimate what a neuron (area) cares about
- However, a lot of these predictors are correlated (e.g., running speed and pupil diameter), so regularization and cross-validation are key

Temporal kernels

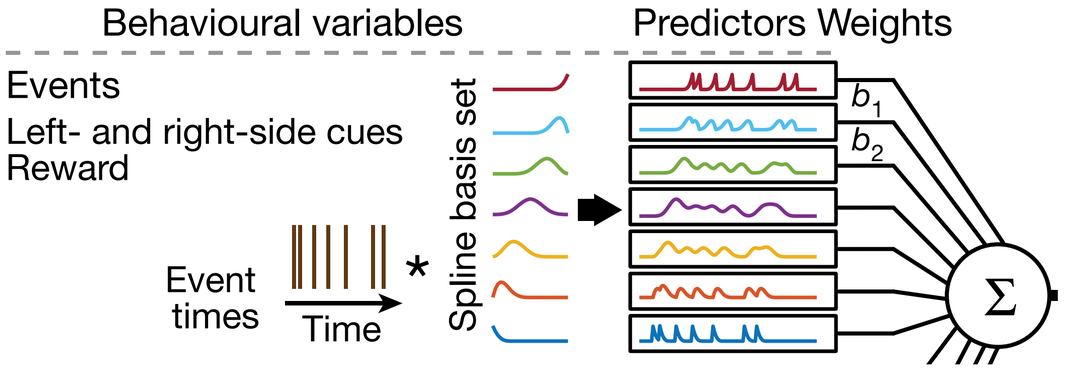
Time lags for discrete events

$$y = \begin{pmatrix} t_o \\ t_1 \\ t_2 \\ \vdots \end{pmatrix} \quad X = \begin{array}{c} \begin{matrix} t_0 & t_{-1} & t_{-2} \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \end{pmatrix} \end{array}$$

Pro: Non-parametric; captures kernel of any shape
Con: Lots of parameters

Basis sets for discrete events

Convolve discrete events with a set of orthogonal functions, e.g. spline, cosine bumps, polynomials



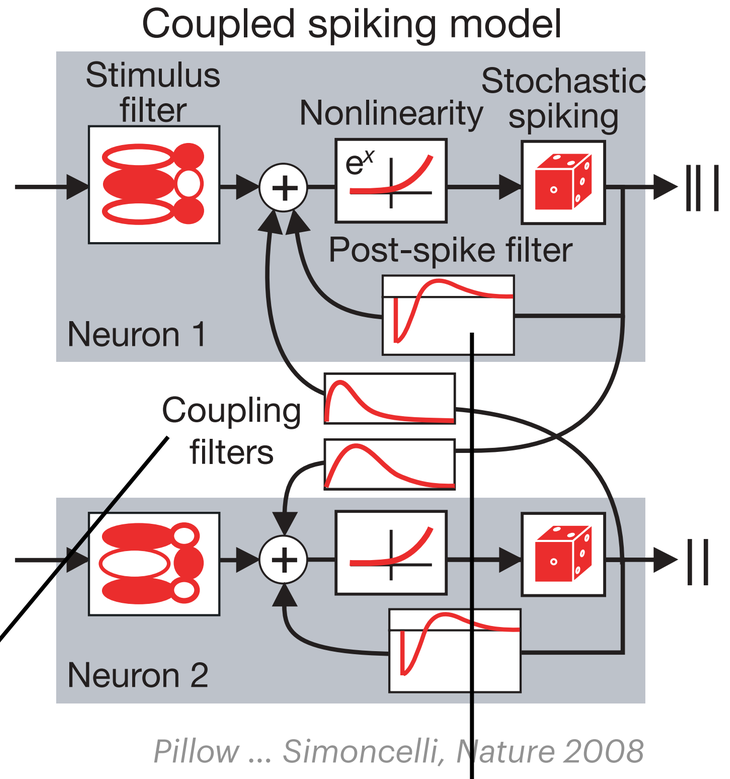
Engelhard ... Witten, Nature 2019

Pro: great for *p* budget
Con: smooths (but maybe that's good, too)

Continuous events: generally boxcar predictors, but also OK to do lags or non-linear predictor approximations

Coupling and auto-regressive predictors

- Neurons are also influenced by other neurons in ways that cannot be captured by external events (e.g. not every spike is related to a visual stimulus). We can capture that by explicitly modeling their interactions with other simultaneously recorded neurons.
- They also have their own history effects: e.g., refractory periods, characteristic timescales (auto-correlation functions). We can model these via auto-regressive predictors.
- Adding these typically improves model predictions quite a bit.



Pillow ... Simoncelli, Nature 2008

weight activity from other k-1 neurons at multiple time lags δ

$$\sum_{j=1}^{k-1} \sum_{t=-\delta}^{-1} y_{j,t} w_{j,t}$$

OR

do it with a set of basis functions θ

$$\sum_{j=1}^{k-1} \sum_{m=1}^{M} (y_k * \theta_m) w_{j,m}$$

Autoregressive

$$\sum_{t=-\delta}^{-1} y_{self(t)} w_t \quad \text{OR} \quad \sum_{m=1}^{M} (y_{self} * \theta_m) w_m$$

Behavioral data

- Behavior is also a function of multiple features of a task or environment.
- For example, in sensory 2-AFC tasks, stimulus identity (left vs. right side) or value (contrast) are obvious features to add.
- But choice is also influenced by things like trial history. An animal may be more likely to repeat a previous choice that led to a reward, even if that is not optimal.
- So for a task in which a mouse needs to lick towards the side with the highest-contrast grating, the model could look like this:

$$P(choose\ R) = \frac{1}{1 + e^{-A}}$$

$$A(t) = w_0 + c_R w_R + c_L w_L + choice_{t-1} w_{ch\ t-1} + (choice_{t-1} \times reward_{t-1}) w_{ch \times rew\ t-1}$$

intercept = side bias

Separate predictors for R and L contrast allow them to have different weights on choice. Could also have a single signed predictor (where L < 0)

Previous choice captures perseveration (could go n trials back)

Choice x reward interaction captures win-stay / lose-shift strategies

Treatment effects

- GLMs are a great way to estimate some treatment effect, e.g. the effect of optogenetically silencing a brain area on behavior (or neural activity)
- A common way of doing this is separately fitting models for control and treatment data. This keeps p smaller.
- Adding a single treatment predictor in the model is not a great idea, since it can only add an offset to $\hat{y}$.
- An alternative is to have a single model with treatment interaction terms with all predictors. Two downsides: 1) doubles p (OK with lots of data). 2) harder to interpret: the direction of the effect needs to be interpreted in light of the sign of the equivalent predictor without the interaction term.

Summary: regression best practices

- Lots of data! :-)
- Design X wisely: avoid correlated predictors, try to keep p small, and certainly $p << n$
- z-score X
- Pick link functions/noise distributions that match data statistics
- Use mixed-effects models for clustered data
- Do model selection and testing with cross-validation
- Regularize. L1 vs L2 depends on your hypothesis
- Test predictor significance by dropping and preferably refitting

Suggested reading

- Hastie, Tibshirani & Friedman (2017). The Elements of Statistical Learning, 2nd edition. Chapters 1–3.
- Hastie & Tibshirani (1986). Generalized Additive Models. *Statist. Sci.* 1:297-310
- Yu et al. (2022). Beyond t test and ANOVA: applications of mixed-effects models for more rigorous statistical analysis in neuroscience research. *Neuron* 110:21-35
- Journal club papers on canvas

The end