LINEAR REGRESSION V

11.13.2020

HOMEWORK 5

* due a week from today!

RECAP

* np.linalg.lstsq – numpy function that does least squares regression

* R^2 is a measure of how good a regression model is

RECAP

* in-set vs. out-of-set evaluation of a regression model

  * in-set biased upwards (to 1.0 in even not-so-extreme cases!)

  * out-of-set unbiased (maybe?) or biased downwards, depending on assumptions

RECAP

* regression stability

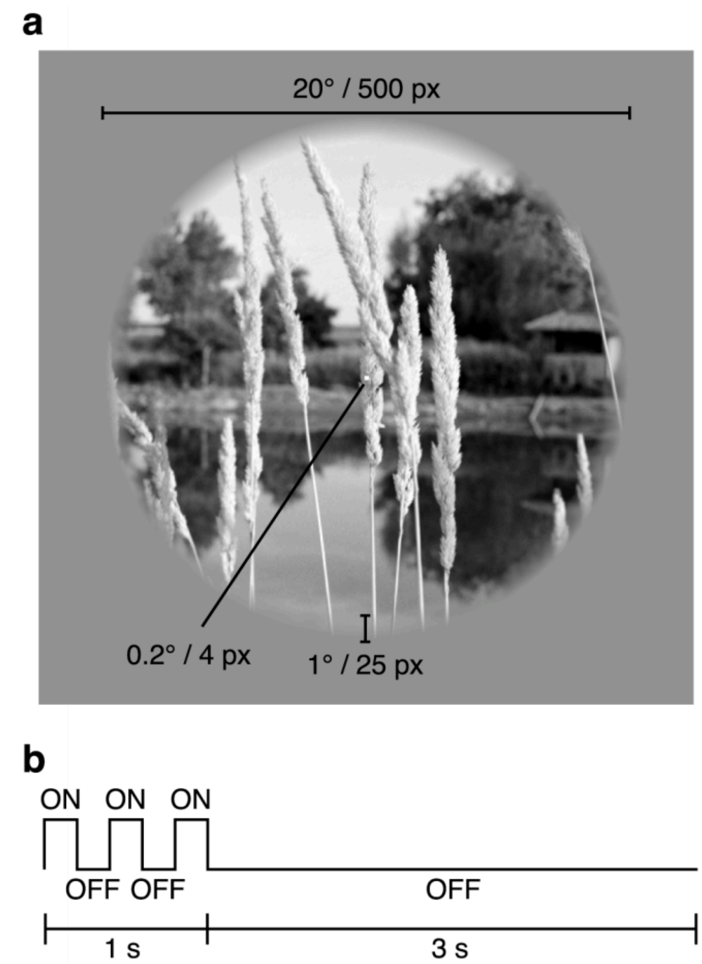
  * similar (*or linearly dependent*) regressors can cause instability in the weight estimates

  * can be assessed by looking at the **singular values** output by **lstsq**

REGRESSION IN PRACTICE

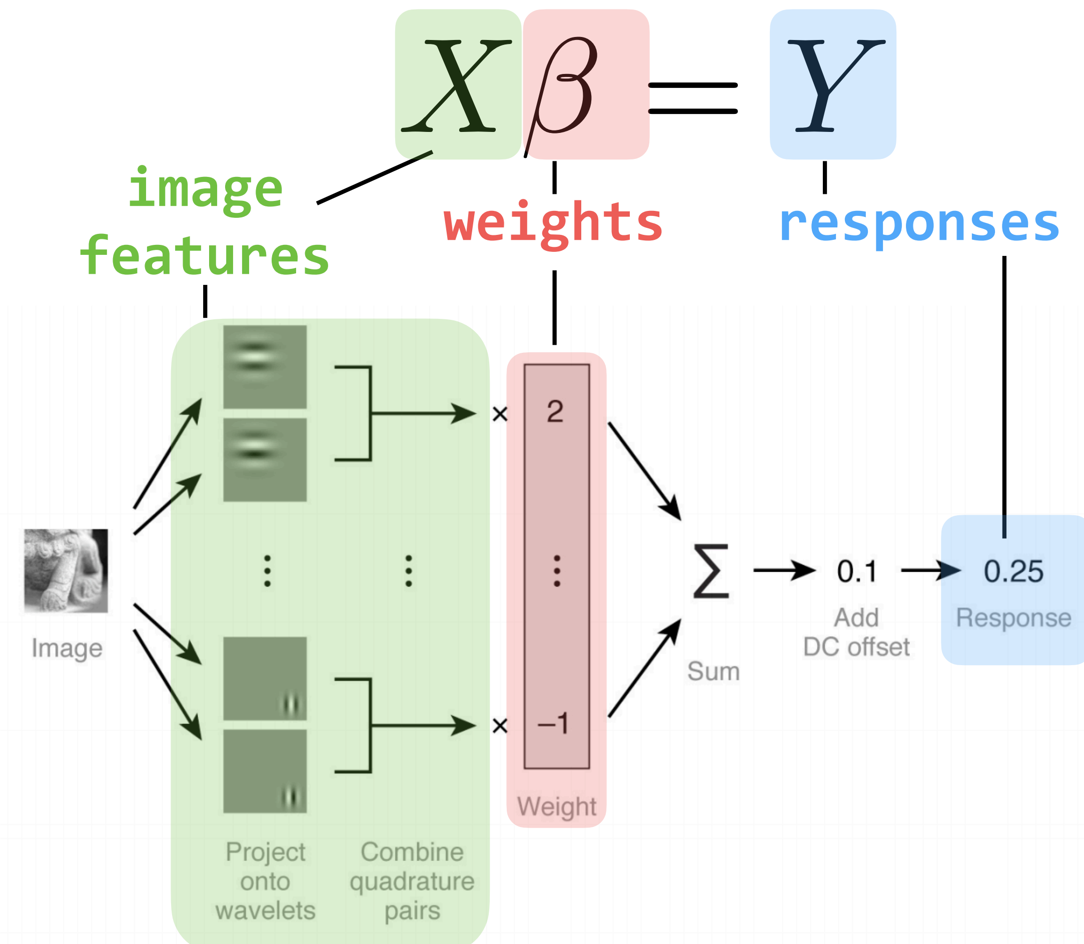
* show a subject 1750 images while recording brain responses using fMRI

* then use **regression** to determine what features in the image are represented in each brain area



Kay, Naselaris, Prenger, & Gallant (2008)

REGRESSION IN PRACTICE



Kay, Naselaris, Prenger, & Gallant (2008)

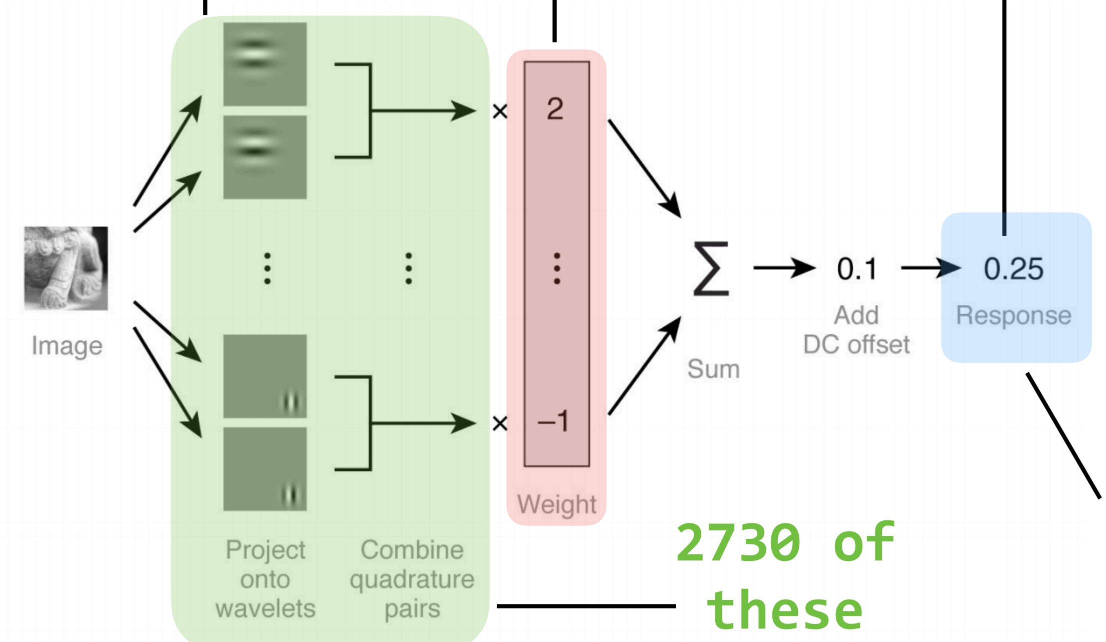
REGRESSION IN PRACTICE

$$X\beta = Y$$

image features

weights

responses



2730 of these

but only 1750 of these!

Kay, Naselaris, Prenger, & Gallant (2008)

STABILITY

* how do we correct unstable regression?

* remember the issue is that many different values for the weights can give us ~the same answer

STABILITY

* we correct unstable regression by making *assumptions* about what the weights should look like
  * this is called **regularization**
    * *imo this is the most important concept in all of machine learning*

STABILITY

* the most common assumption is that the weights should be *small*

* what does “small” mean? and how can we enforce “smallness”?

REGULARIZATION

* one way to get regularization is to modify the **error function** that we minimize to get the weights

* we want small weights, so we can make large weights look like an error!

REGULARIZATION

* recall the “least squares” error function:

$$Err(\beta) = \sum_{t=1}^{T} (y_t - x_t\beta)^2$$

REGULARIZATION

* we can modify this to add a *penalty* for large weights

$$Err(\beta) = \sum_{t=1}^{T} (y_t - x_t\beta)^2 + \lambda \sum_{i=1}^{P} \beta_i^2$$

REGULARIZATION

* now the error is the sum of a **loss term** and a **penalty term**

$$Err(\beta) = \boxed{\sum_{t=1}^{T}(y_t - x_t\beta)^2} + \boxed{\lambda\sum_{i=1}^{P}\beta_i^2}$$

REGULARIZATION

* it also introduces an extra parameter, λ, which is the ***regularization coefficient***, or, in this case, ***ridge coefficient***

$$Err(\beta) = \sum_{t=1}^{T}(y_t - x_t\beta)^2 + \boxed{\lambda}\sum_{i=1}^{P}\beta_i^2$$

REGULARIZATION

* what effect does adding this type of regularization have on a regression model?

1D EXAMPLE

subject

stimulus

* y = output of a neuron that you are measuring

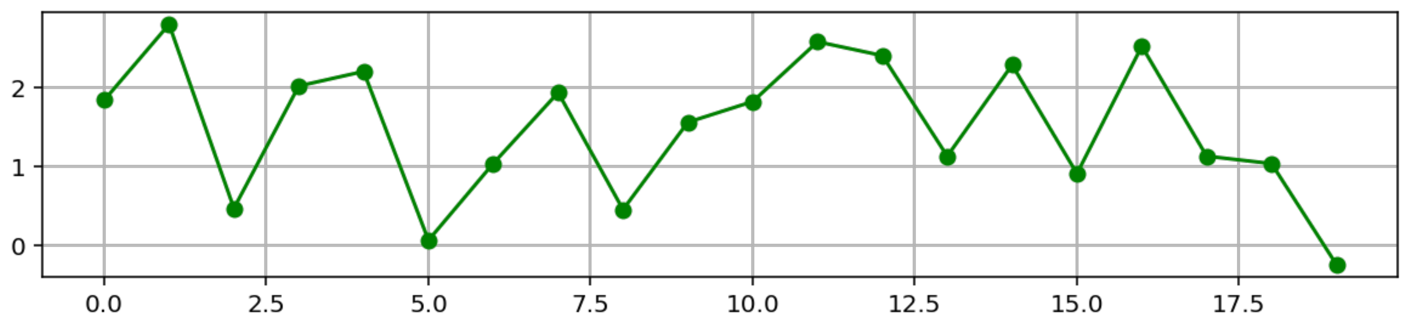
* x = how many times per second the screen flashes

1D EXAMPLE

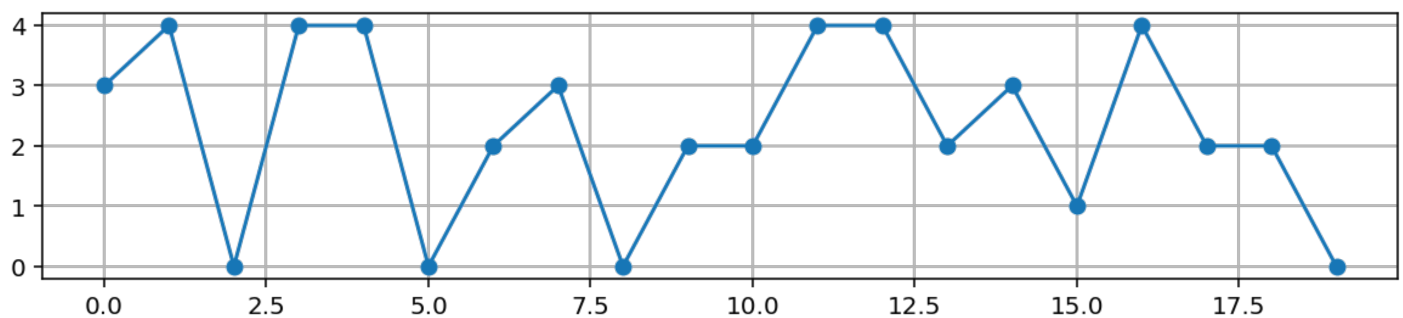
subject

stimulus

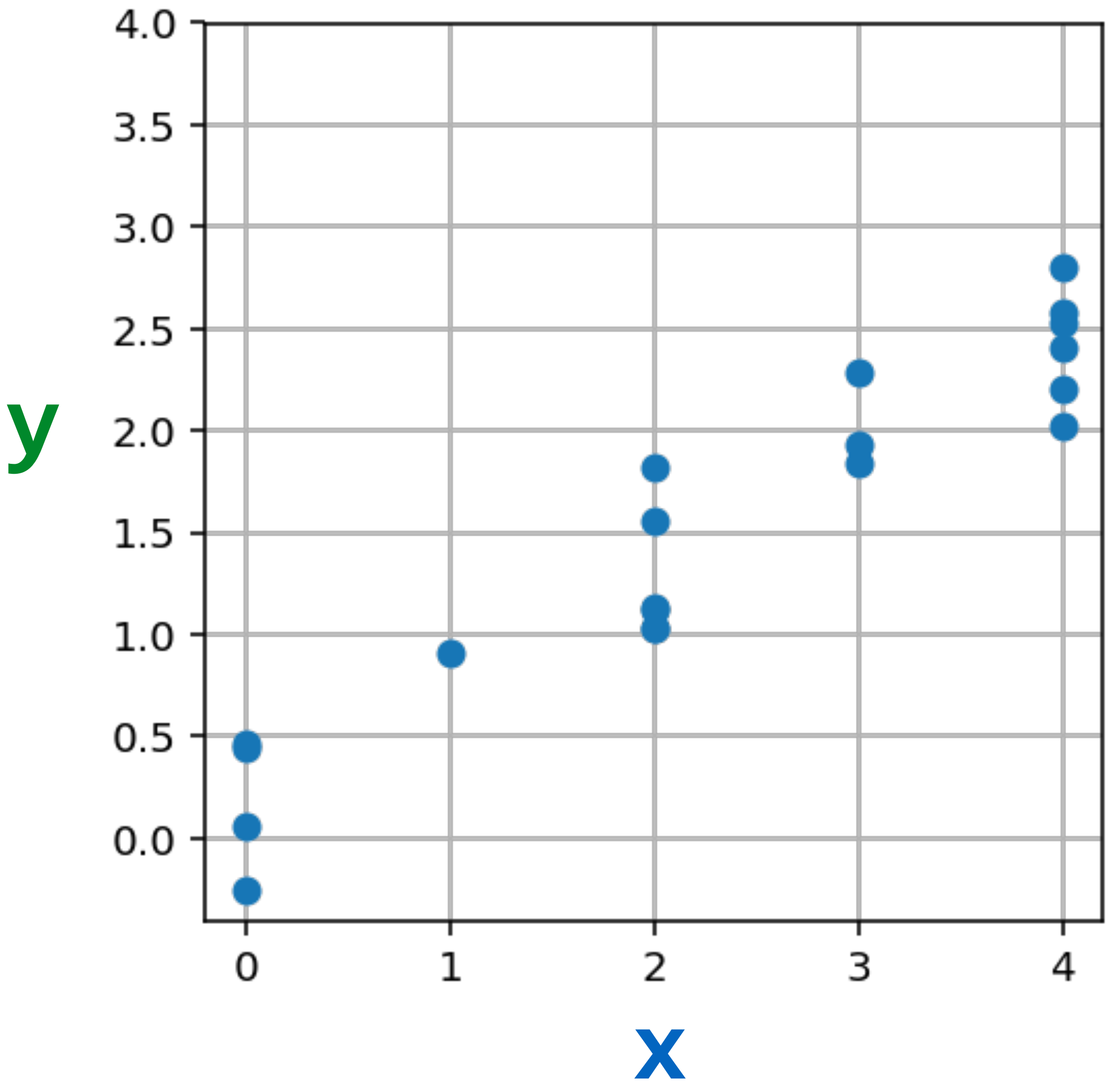
y



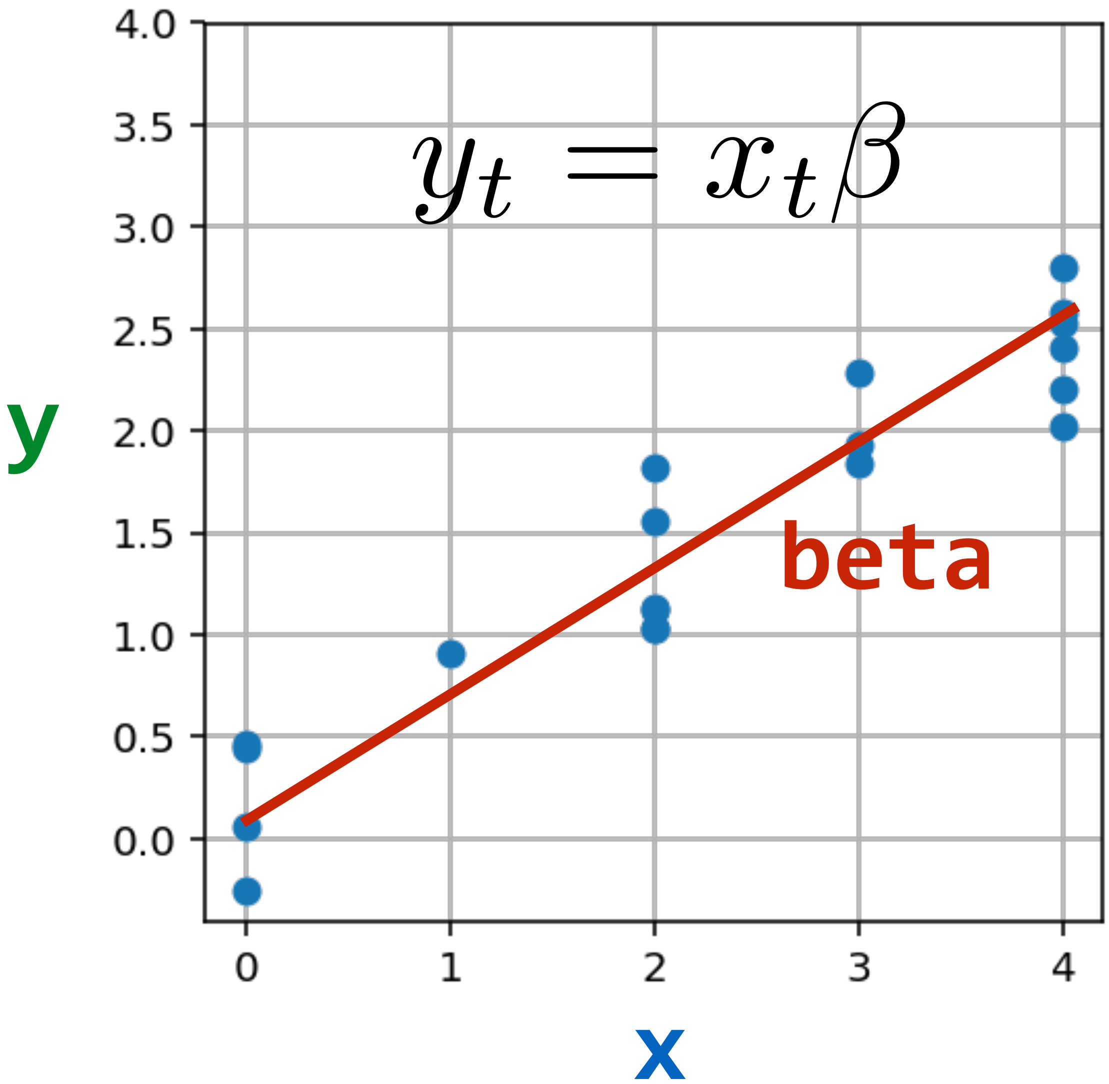
x



1D EXAMPLE

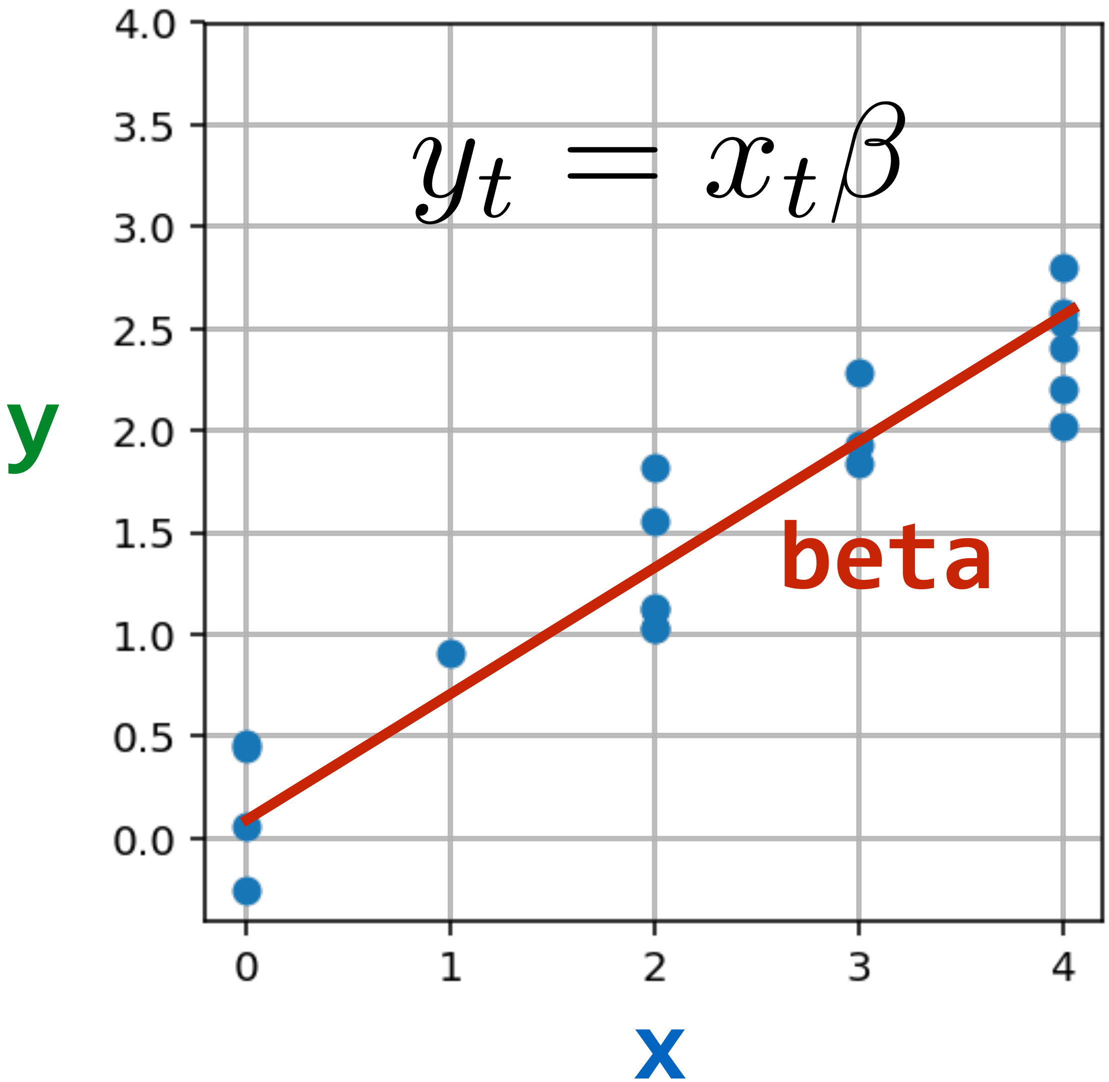


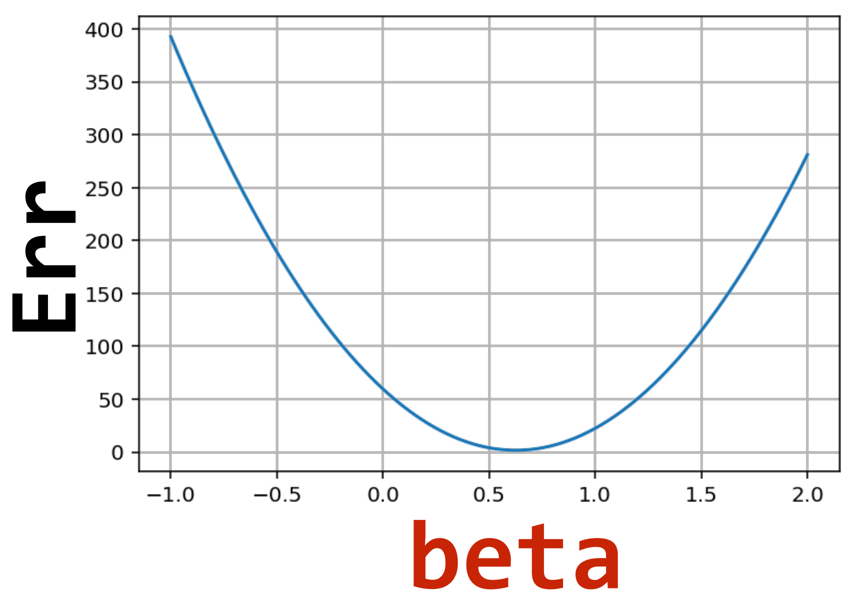
1D EXAMPLE



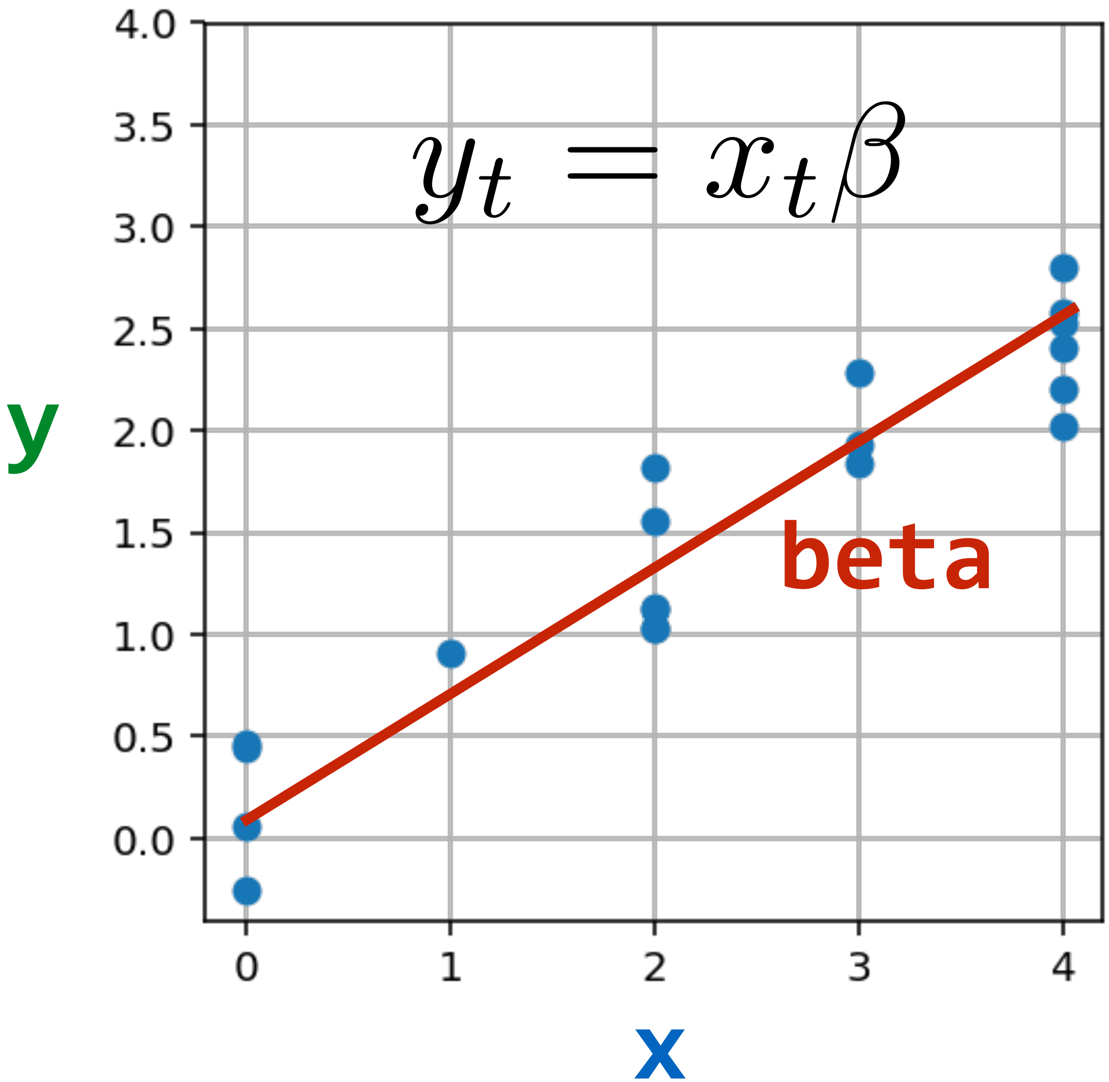
1D EXAMPLE

$$Err(\beta) = \sum_{t=1}^{T} (y_t - x_t\beta)^2$$

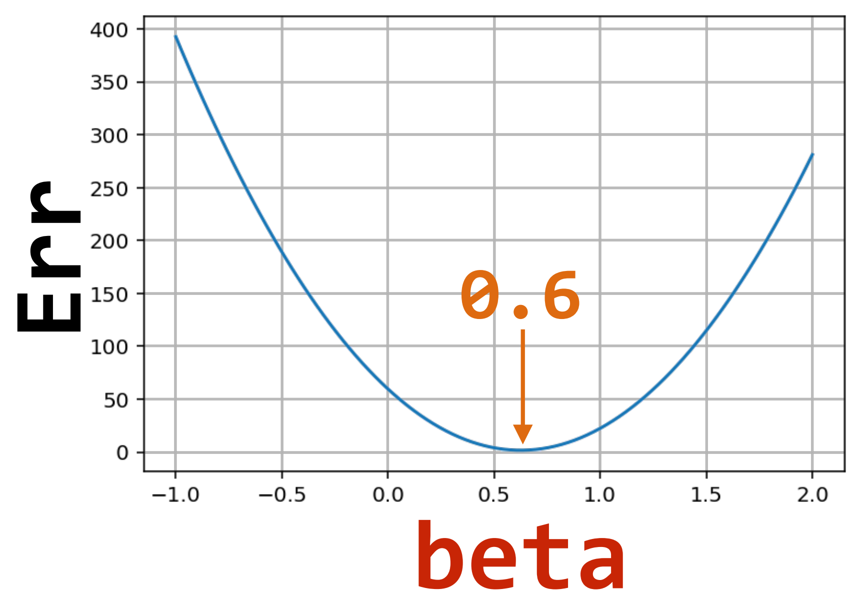




1D EXAMPLE

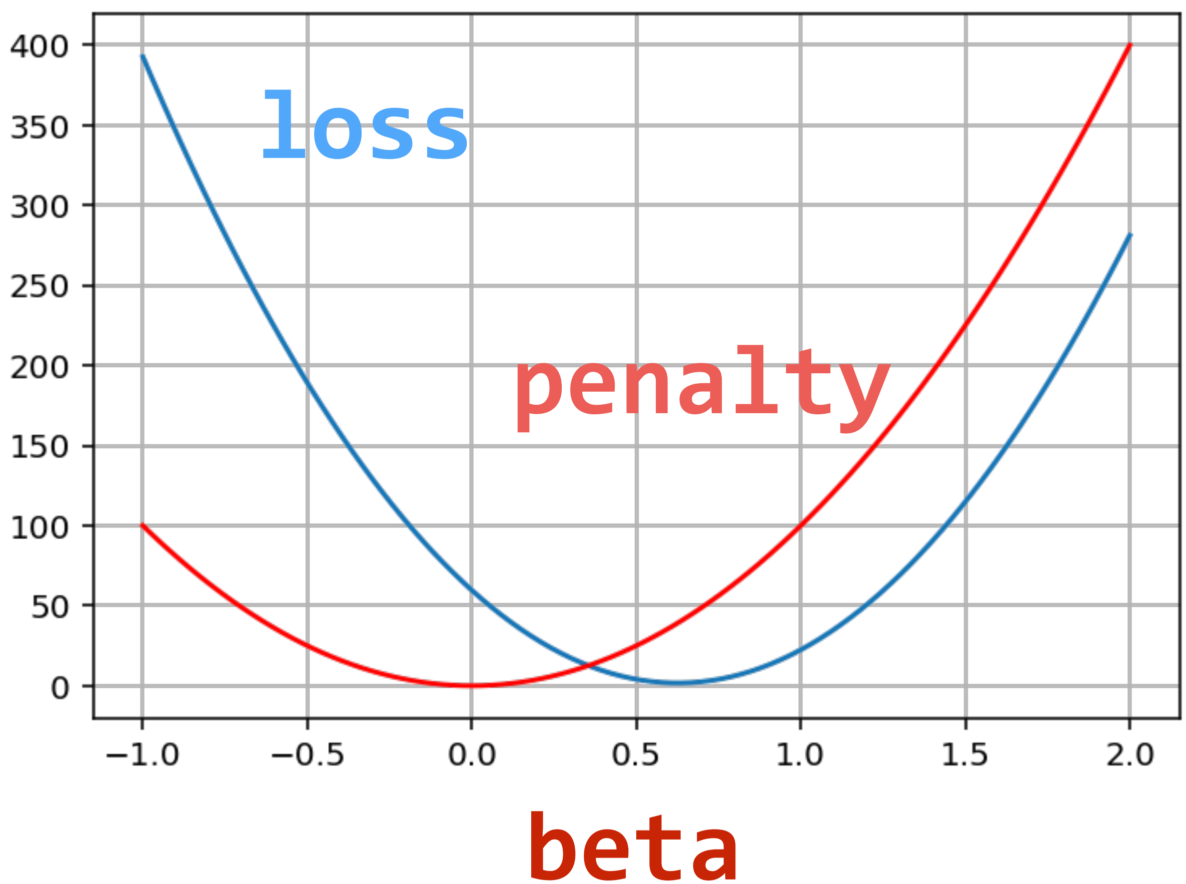


$$Err(\beta) = \sum_{t=1}^{T}(y_t - x_t\beta)^2$$



1D EXAMPLE

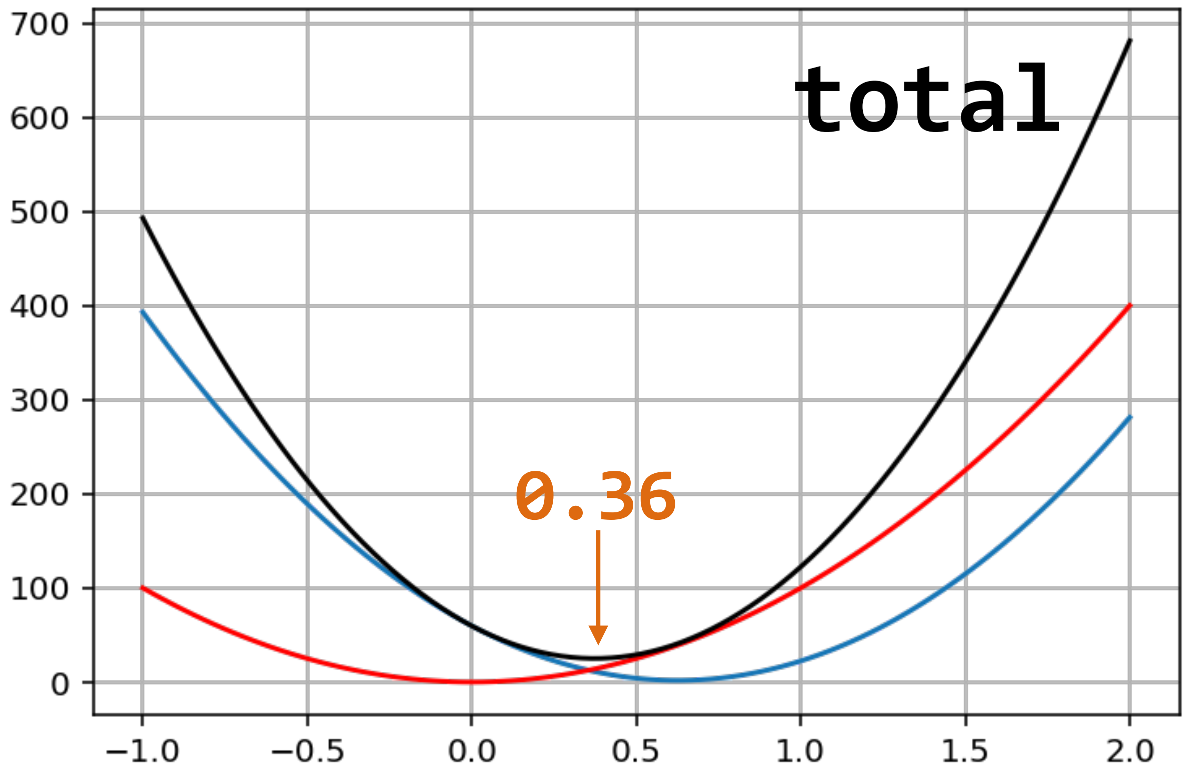
Regularization: $$Err(\beta) = \boxed{\sum_{t=1}^{T}(y_t - x_t\beta)^2} + \boxed{\lambda\beta^2}$$



1D EXAMPLE

Regularization:

$$Err(\beta) = \boxed{\sum_{t=1}^{T}(y_t - x_t\beta)^2} + \boxed{\lambda\beta^2}$$



beta

REGULARIZATION

* what effect does adding this type of regularization have on a regression model?

* it changes the shape of the error function to be more *circular* (and thus more stable)

REGULARIZATION

* what effect does this have on the weights?

* among the many ~equivalent sets of weights, it preferentially chooses those that are *small*

RIDGE REGRESSION

* this type of regularization (penalizing the sum of squared weights) is called **ridge regression**

* and because the ridge error function (loss + penalty) is parabolic, it has an analytic solution!

RIDGE REGRESSION

* a nice implementation is in scikit-learn (a package that we’ll talk more about next week) in **sklearn.linear_model.Ridge**

RIDGE REGRESSION

* but when doing ridge regression you have a new issue: how do you choose the ridge parameter, λ?

RIDGE REGRESSION

* if you train and test your regression model on the same piece of data, $\lambda=0$ is always going to be the best

  * ***~bogus~***

RIDGE REGRESSION

* if you train and test on different datasets (as discussed earlier) it’s better

  * but using your test data multiple times (to choose a parameter!) creates an issue of bias (aka **overfitting**)

RIDGE REGRESSION

* the correct solution is **cross-validation:**
  * break your dataset into training and test (X -> [X_trn, X_test])
  * further break up your training set(X_trn -> [X_fit, X_val])
  * fit weights using X_fit, choose λ based on performance on X_val, then finally test on X_test

END