CSE/STAT 416

Regularization – Ridge Regression

Karthik Mohan
University of Washington
June 28, 2021

Logistics/ Announcements

- Homework 1 extended to Wednesday night 9 pm due to heat wave/power outage issues for some of the folks
- Homework 2 will go out Wednesday
- GradeScope submissions active - Concept portion of homework and learning reflections submitted here
- Rest of submissions happen on EdStem (programming and check points)
- Any questions?

NOTE ON POLLS

- Those watching this lecture recording
  - Only one poll can be activated at a time
  - Workaround: After lecture fix: Combine polls into one and activate that one

RECAP LAST LECTURE

- Choosing the right model complexity
- Overfitting vs underfitting
- Linear regression vs polynomial regression
- Train set error vs test

Model determinism

- Can running the same ML model on the same data produce different results?
- What are the sources of randomness? (HW0 example of R^2)
- How do we make the model be deterministic?

Choosing Complexity

We can't just choose the model that has the lowest **train** error because that will favor models that overfit!

It then seems like our only other choice is to choose the model that has the lowest **test** error (since that is our approximation of the true error)

- This is almost right, but now we don't have a good estimate of the true error anymore.
- We didn't technically train the model on the test set (that's good), but we chose **which model** to use based on the performance of the test set.
    - It's no longer a stand in for "the unknown" since we probed it many times to figure out which model would be best.

Poll Everywhere

Think

1 min

pollev.com/
karthikmohan088

ML MODEL FIT PART 1

Suppose we fit three ML models to solve the spam detection problem - Model A, Model B and Model C. The 3 models use the same data set but have a different ML algorithm doing the training.

Let TrainErr[X] and TestErr[X] denote the Train and Test error of model X. Suppose TrainErr[B] > TrainErr[A] and TestErr[B] > TestErr[A]

Which of the following statements are true?

a) Model A overfits as compared to Model B

b) We need to increase the model complexity for Model B

c) Model A has a higher model complexity than Model B

d) None of the above

Poll Everywhere

Think

1 min

pollev.com/
karthikmohan088

ML MODEL FIT PART 2

Suppose we fit three ML models to solve the spam detection problem - Model A, Model B and Model C. The 3 models use the same data set but have a different ML algorithm doing the training.

Let TrainErr[X] and TestErr[X] denote the Train and Test error of model X. Now assume TrainErr[A] < TrainErr[C] and TestErr[C] < TestErr[A]

Which of the following statements are true?

a) Model A overfits as compared to Model C

b) We need to lower the model complexity for Model A

c) Model C has a higher model complexity than Model A

c) None of the above

Choosing Complexity

We will talk about two ways to pick the model complexity without ruining our test set.

- Using a validation set
- Doing cross validation

Validation Set

So far we have divided our dataset into train and test

We can't use Test to choose our model complexity, so instead, break up Train into ANOTHER dataset

Validation Set

The process generally goes

```
train, validation, test = split_data(dataset)
for each model complexity p:
    model = train_model(model_p, train)
    val_err = error(model, validation)
    keep track of p with smallest val_err
return best p + error(model, test)
```

Validation Set

Pros

Easy to describe and implement

Pretty fast

- Only requires training a model and predicting on the validation set for each complexity of interest

Cons

Have to sacrifice even more training data! ☹️

Poll Everywhere

Think

1 min

pollev.com/
karthikmohan088

ML MODEL SPLITS

Suppose we want to build an ML model for spam detection in emails.

The data set had 1000 emails with 100 of them spam.

What would be a good way to split the data into training and test ?

a) 600 emails in training data and 400 emails in test data

b) 500 emails in training data and 500 emails in test data

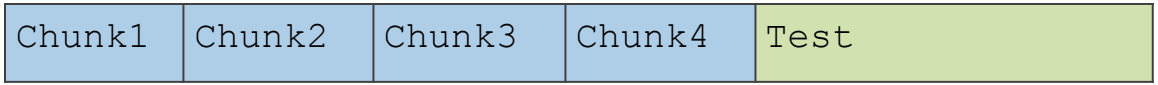
c) 800 emails in training data and 200 emails in test data

d) 950 emails in training data and 50 emails in test data

Cross-Validation

Clever idea: Use many small validation sets without losing too much training data.

Still need to break off our test set like before. After doing so, break the training set into chunks.



For a given model complexity, train it times. Each time use all but one chunk and use that left out chunk to determine the validation error.

80-20 split. Take the 80% split and do a k-fold cross-validation on it.

Cross-Validation

The process generally goes

```
chunk_1, …, chunk_k, test = split_data(dataset)
for each model complexity p:
    for i in [1, k]:
        model = train_model(model_p, chunks - i)
        val_err = error(model, chunk_i)
    avg_val_err = average val_err over chunks
    keep track of p with smallest avg_val_err
return model trained on train with best p +
error(model, test)
```

Cross-Validation

Pros

Don’t have to actually get rid of any training data!

Cons

Can be a bit slow. For each model complexity, trains k models!

For best results, need to make k really big

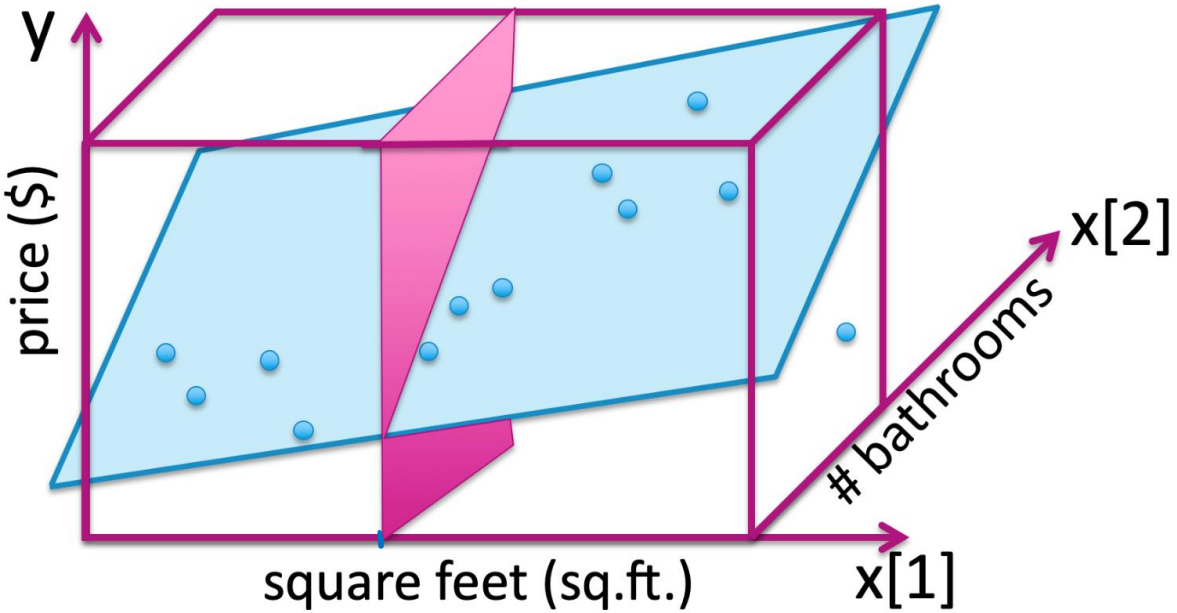
- Theoretical best estimator is to use $k = n$
  - Called "Leave One Out Cross Validation”
- In practice, people use $k = 5$ to 10

Interpreting Coefficients

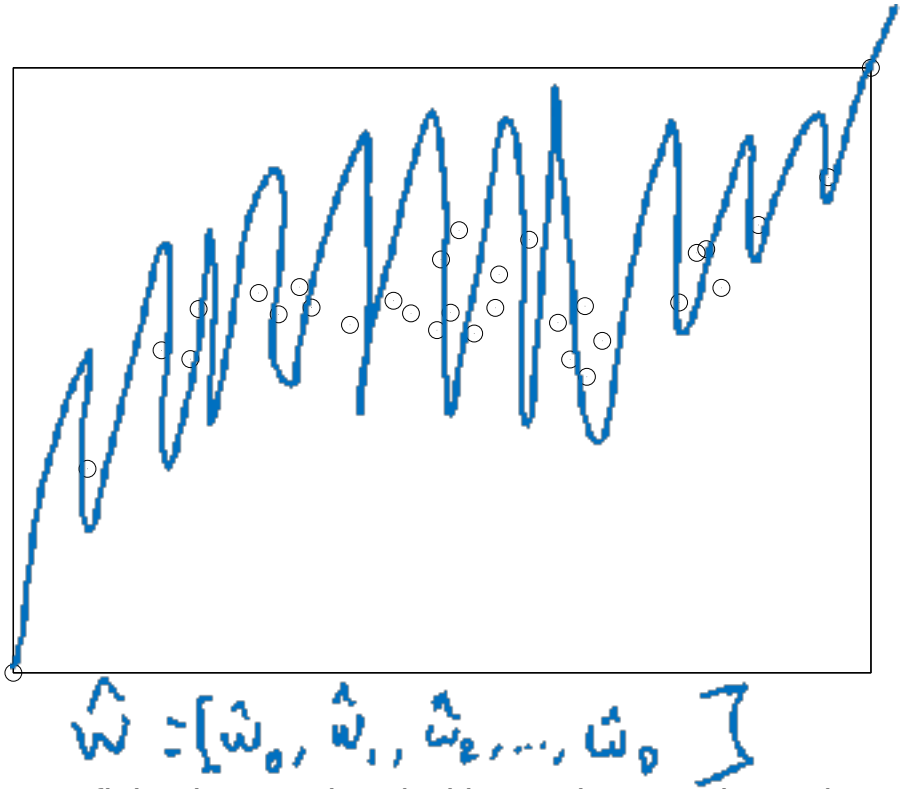
Interpreting Coefficients – Multiple Linear Regression

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x[1] + \hat{w}_2 x[2]$$

Fix



Overfitting



$\hat{w} = [\hat{w}_0, \hat{w}_1, \hat{w}_2, \ldots, \hat{w}_p]$

Often, overfitting is associated with very large estimated parameters !

$|\hat{w}_j| >> 0$

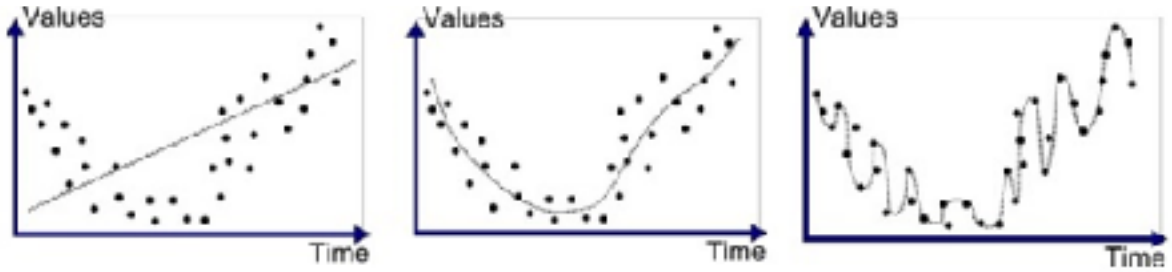
Poll Everywhere

Think

1 min

pollev.com/
karthikmohan088

Regression Fit



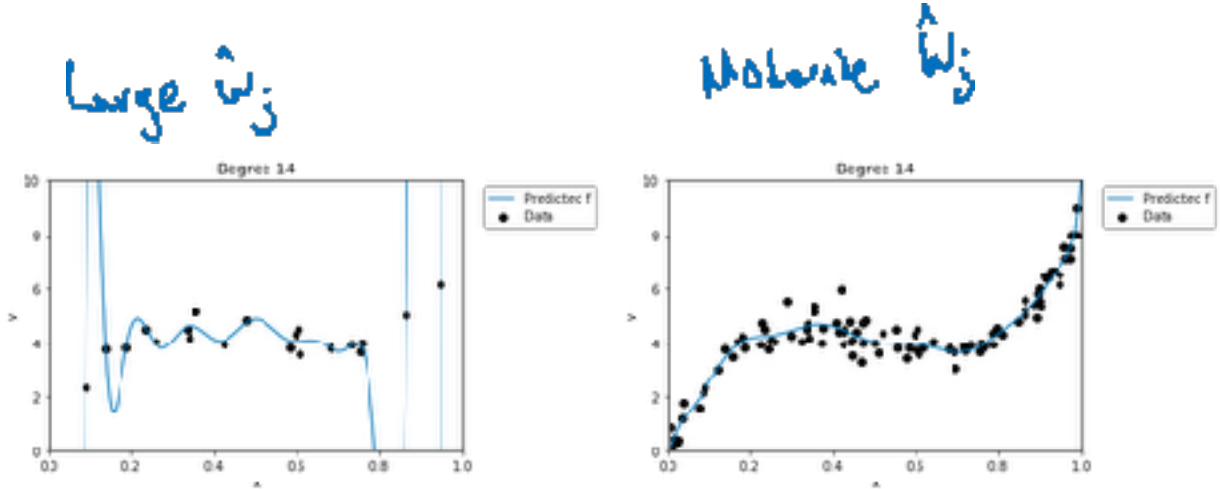
Consider the above fits through 3 different regression models - Models A,B & C.
Which of these following statements are true?

a) Model A overfits
b) Model B is a good fit
c) Model C overfits
d) Model A has high bias
e) Model C has high bias
f) Model A has high variance
g) Model A is a linear regression model

Number of Features

Overfitting is not limited to polynomial regression of large degree. It can also happen if you use a large number of features!

Why? Overfitting depends on how much data you have and if there is enough to get a representative sample for the complexity of the model.



Poll Everywhere

Think

1 min

pollev.com/karthikmohan088

NUMBER OF FEATURES

Consider the housing prices problem. Let's say the data had raw features that include columns for square footage, # bedrooms and location of the house. There are two models fit to the data. Model A does a linear fit over the training data. Model B does a polynomial fit of degree 3 specifically on square footage and linear fit on other features.

How many parameters do Model A and Model B have?

a) 3 and 5

b) 3 and 6

c) 4 and 6

d) 4 and 5

e) 4 and 7

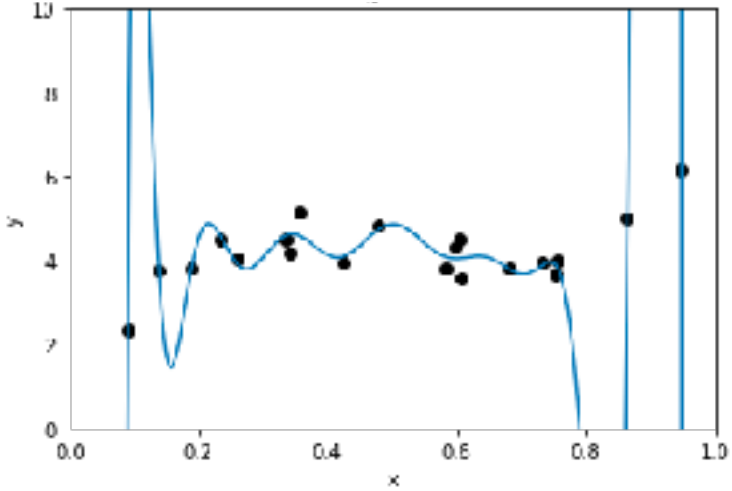
f) None of the above

Number of Features

How do the number of features affect overfitting?

1 feature

Data must include representative example of all pairs to avoid overfitting

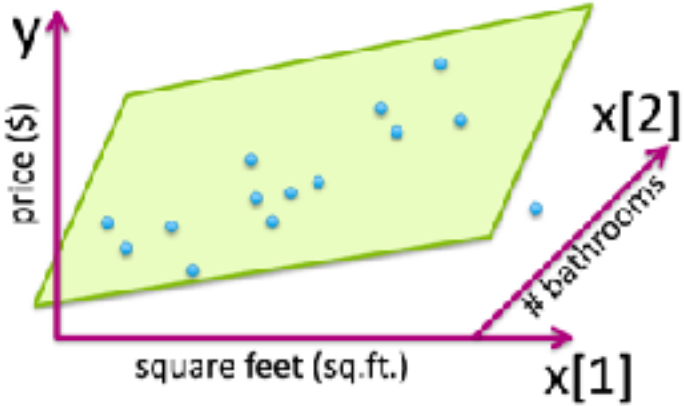


Number of Features

How do the number of features affect overfitting?

D features

Data must include representative example of all combos to avoid overfitting!



Introduction to the **Curse of Dimensionality**.
We will come back to this later in the quarter!

Prevent Overfitting

Last time, we saw we could use cross validation / validation set to pick which model complexity to use

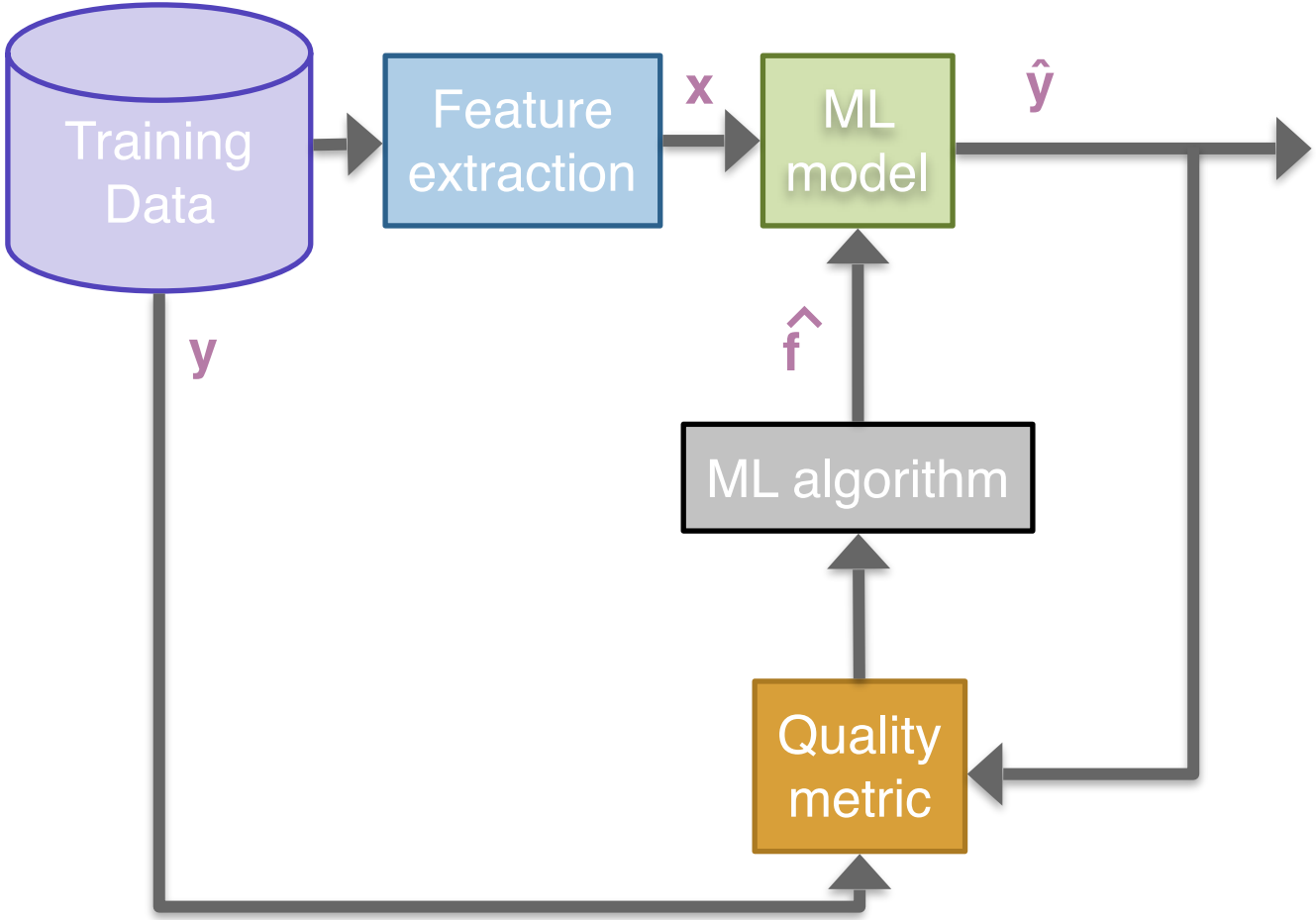
- In the case of polynomial regression, we just chose degree
- For deciding which or how many features to use, there are a lot of choices!
  - For inputs, there are subsets of those inputs!

What if we use a model that wasn't prone to overfitting?

- Big Idea: Have the model self-regulate to prevent overfitting by making sure its coefficients don't get "too large"

This idea is called **regularization**.

Regularization


Training Data
Feature extraction
x
ML model
ŷ
y
f̂
ML algorithm
Quality metric

Regularization

Our case: $L(w) = RSS(w)$

Before, we used the quality metric that minimized loss

$$\hat{w} = \min_{w} L(w)$$

Change quality metric to balance loss with measure of overfitting

- $L(w)$ is the measure of fit
- $R(w)$ measures the magnitude of coefficients

$$\hat{w} = \min_{w} L(w) + \lambda R(w)$$

How do we actually measure the magnitude of coefficients?

Magnitude

$w = [w_0, w_1, \ldots, w_D]$

$R(w) =$ measure of overfitting

Come up with some number that summarizes the magnitude of the coefficients in .

Sum?

$R(w) = \sum_{j=0}^{D} w_j$

Doesn't work

$w = [1000000, -1000000]$

$R(w) = 0$

Sum of absolute values?

$R(w) = \sum_{j=0}^{D} |w_j| \triangleq \|w\|_1$

L1 norm (come back Wed!)

Sum of squares?

$R(w) = \sum_{j=0}^{D} w_j^2 \triangleq \|w\|_2^2$

L2 norm (today)

Ridge Regression

Change quality metric to minimize

$$\hat{w} = \min_{w} RSS(W) + \lambda \|w\|_2^2$$

λ is a tuning parameter that changes how much the model cares about the regularization term.

What if $\lambda = 0$?

$\hat{w} = \min_{w} RSS(w)$

$\hookrightarrow \hat{w}_{LS}$

often called ordinary least squares (OLS)

What if $\lambda = \infty$? $\Rightarrow \hat{w} = \vec{0}$

If any $w_j \neq 0$, $RSS(w) + \lambda \|w\|_2^2 = \infty$

If all $w_j = 0$, $RSS(w) + \lambda \|w\|_2^2 = RSS(w) < \infty$

therefore, will choose $\hat{w} = \vec{0}$

λ in between?

$$0 \leq \|w\|_2^2 \leq \|\hat{w}_{LS}\|_2^2$$

Effect of regularization on bias/ variance

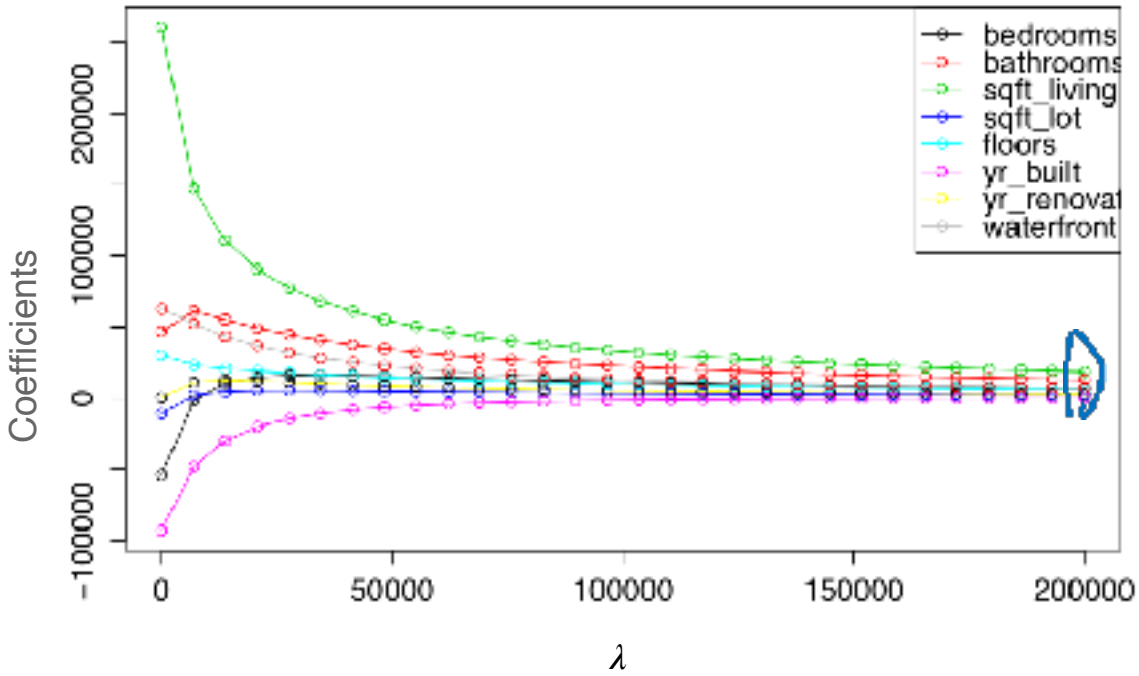
- When lambda is 0, the model has ——— bias and ——— variance ? (high/low)
- When lambda is infinity, the model has ——— bias and ——— variance ?

Effect of regularization on bias/ variance

- When lambda is 0, the model has **lower** bias and **higher** variance ? (high/low)
- When lambda is infinity, the model has **higher** bias and **lower** variance ?
- Does regularization (i.e. with lambda > 0) increase or decrease the model complexity?

Brain Break

Coefficient Paths for Ridge Regression



Choosing λ

For any particular setting of λ, use Ridge Regression objective

$$\widehat{w}_{ridge} = \min_{w} RSS(w) + \lambda||w_{1:D}||_2^2$$

If λ is too small, will overfit to **training set**. Too large, $\widehat{w}_{ridge} = 0$.

How do we choose the right value of λ? We want the one that will do best on **future data.** This means we want to minimize error on the validation set.

Don't need to minimize $RSS(w) + \lambda||w_{1:D}||_2^2$ on validation because you can't overfit to the validation data (you never train on it).

Another argument is that it doesn't make sense to compare those values for different settings of λ. They are in different "units" in some sense.

Choosing λ

Hyperparameter tuning

The process for selecting λ is exactly the same as we saw with using a validation set or using cross validation.

for λ in λs:

  Train a model using using Gradient Descent

$$\widehat{w}_{ridge(\lambda)} = \min_{w} RSS_{train}(w) + \lambda||w_{1:D}||_2^2$$

  Compute validation error

$$validation\_error = RSS_{val}(\widehat{w}_{ridge(\lambda)})$$

  Track λ with smallest $validation\_error$

Return λ^* & estimated future error $RSS_{test}(\widehat{w}_{ridge(\lambda^*)})$

There is no fear of overfitting to validation set since you never trained on it! You can just worry about error when you aren't worried about overfitting to the data.

Regularization

At this point, I've hopefully convinced you that regularizing coefficient magnitudes is a good thing to avoid overfitting!

You:

We might have gotten a bit carried away, it doesn't ALWAYS make sense…

The Intercept

For most of the features, looking for large coefficients makes sense to spot overfitting. The one it does not make sense for is the **intercept**.

We shouldn't penalize the model for having a higher intercept since that just means the value units might be really high!

Two ways of dealing with this

- Change the measure of overfitting to not include the intercept

- Center the values so they have mean 0
  - This means forcing w_0 to be small isn't a problem

Scaling Features

The other problem we looked over is the “scale” of the coefficients.

Remember, the coefficient for a feature increase per unit change in that feature (holding all others fixed in multiple regression)

Consider our housing example with $(sq.ft., price)$ of houses

- Say we learned a coefficient $\widehat{w}_1$ for that feature
- What happens if we change the unit of x to square **miles?** Would $\widehat{w}_1$ need to change?
  - It would need to get bigger since the prices are the same but its inputs are smaller

This means we accidentally penalize features for having large coefficients due to having small value inputs!

Scaling Features

Fix this by **normalizing** the features so all are on the same scale!

$$\tilde{h}_j(x_i) = \frac{h_j(x_i) - \mu_j(x_1, \dots, x_N)}{\sigma_j(x_1, \dots, x_N)}$$

Where

The mean of feature j:

$$\mu_j(x_1, \dots, x_N) = \frac{1}{N}\sum_{i=1}^{N} h_j(x_i)$$

The standard devation of feature j:

$$\sigma_j(x_1, \dots, x_N) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(h_j(x_i) - \mu_j(x_1, \dots, x_N)\right)^2}$$

Important: Must scale the test data and all future data using the means and standard deviations **of the training set!**

- Otherwise the units of the model and the units of the data are not comparable!

Recap

Theme: Use regularization to prevent overfitting

Ideas:

- How to interpret coefficients
- How overfitting is affected by number of data points
- Overfitting affecting coefficients
- Use regularization to prevent overfitting
- How L2 penalty affects learned coefficients
- Visualizing what regression is doing
- Practicalities: Dealing with intercepts and feature scaling