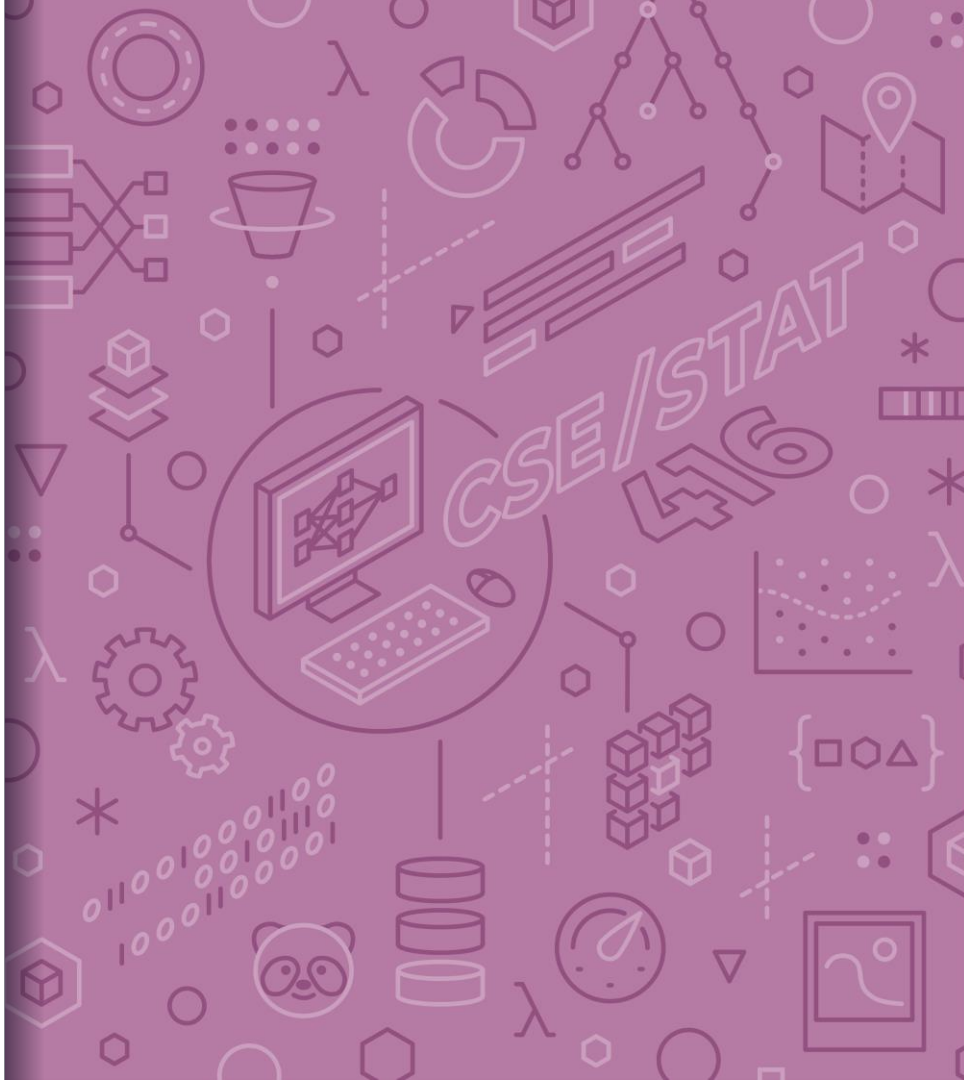
CSE/STAT 416

Regularization – Ridge Regression

Hunter Schafer
University of Washington
April 5, 2021

Music: Rex Orange County



Pre-Lecture Video 1

Cross Validation

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.

Choosing Complexity

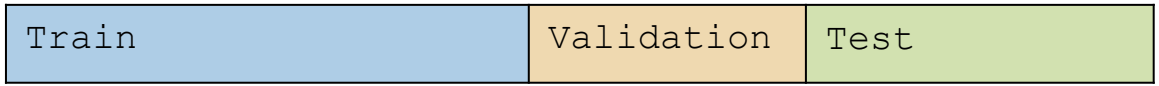
We will talk about two ways to pick the model complexity without ruining our test set.

- 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

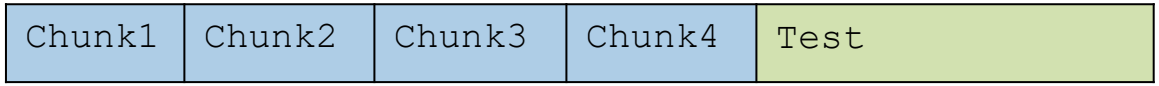
Cons

Have to sacrifice even more training data! ☹

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 k chunks.



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

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

Pre-Lecture Video 2

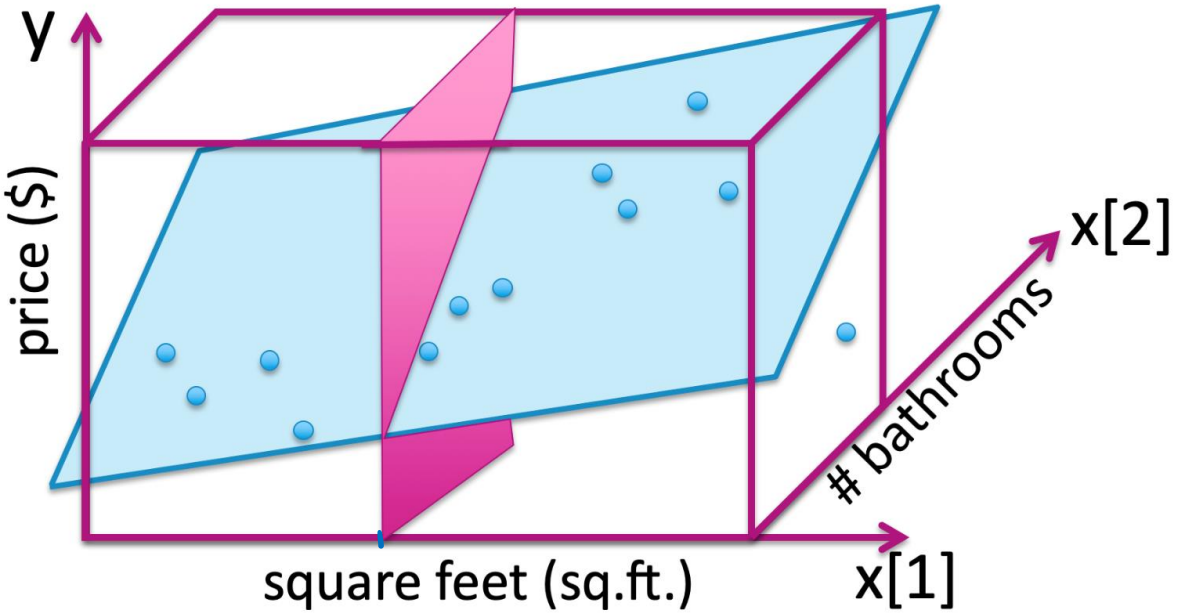
Coefficients and Overfitting

Interpreting Coefficients

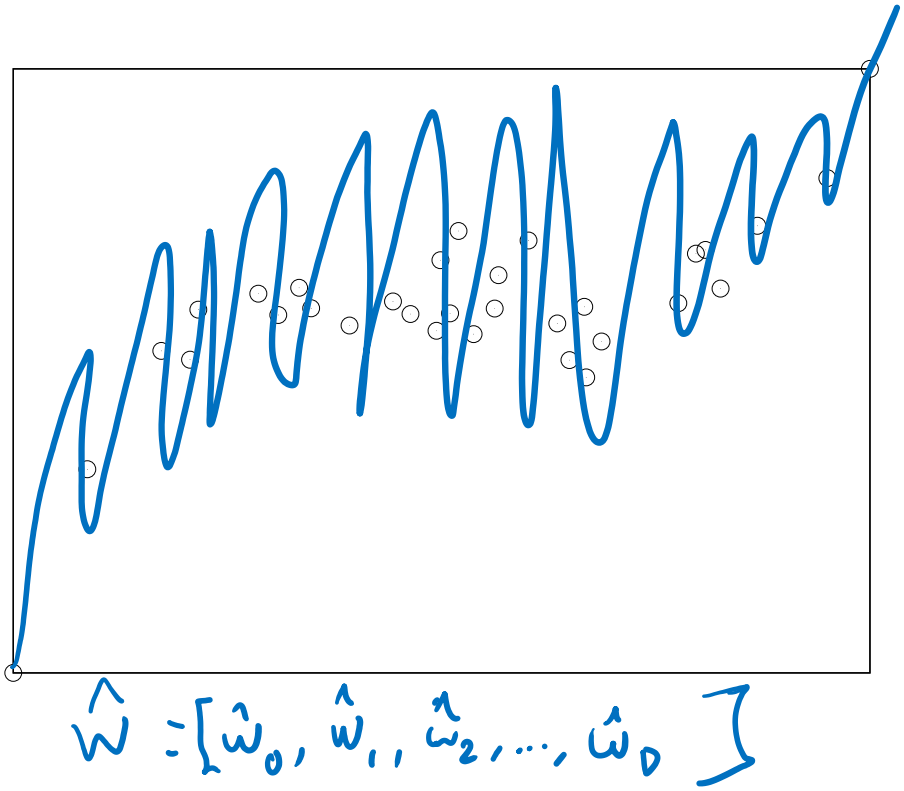
Interpreting Coefficients – Multiple Linear Regression

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

Fix



Overfitting



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

Often, overfitting is associated with very large estimated parameters $\hat{w}$!

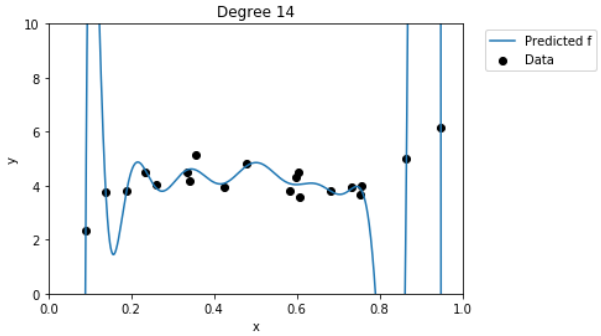
$|\hat{w}_j| >> 0$

Number of Features

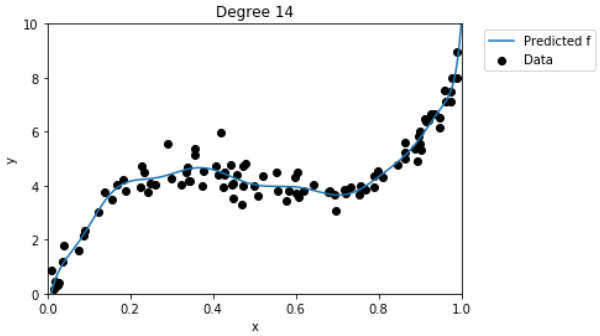
Overfitting is not limited to polynomial regression of large degree. It can also happen if you use a large number of features!

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.

Large $\hat{w}_j$



Moderate $\hat{w}_j$

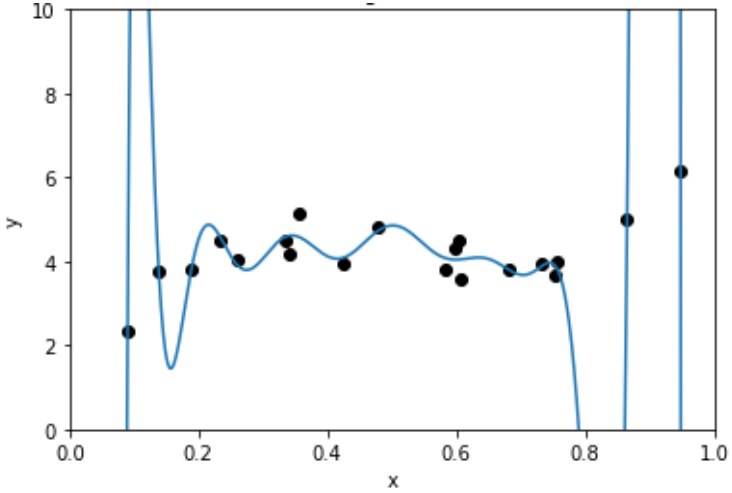


Number of Features

How do the number of features affect overfitting?

1 feature

Data must include representative example of all (x, y) pairs to avoid overfitting

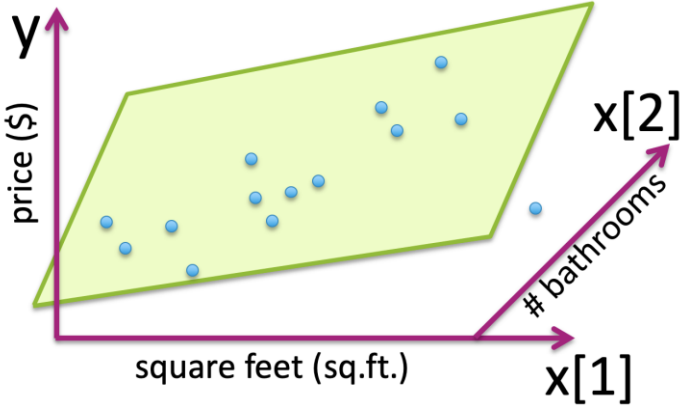


Number of Features

How do the number of features affect overfitting?

D features

Data must include representative example of all $\big((x[1], x[2], \ldots, x[D]), y\big)$ 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 p
- For deciding which or how many features to use, there are a lot of choices!
  - For d inputs, there are 2^d 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

Administrivia

Homework 1 Due tomorrow night! Remember to do both parts

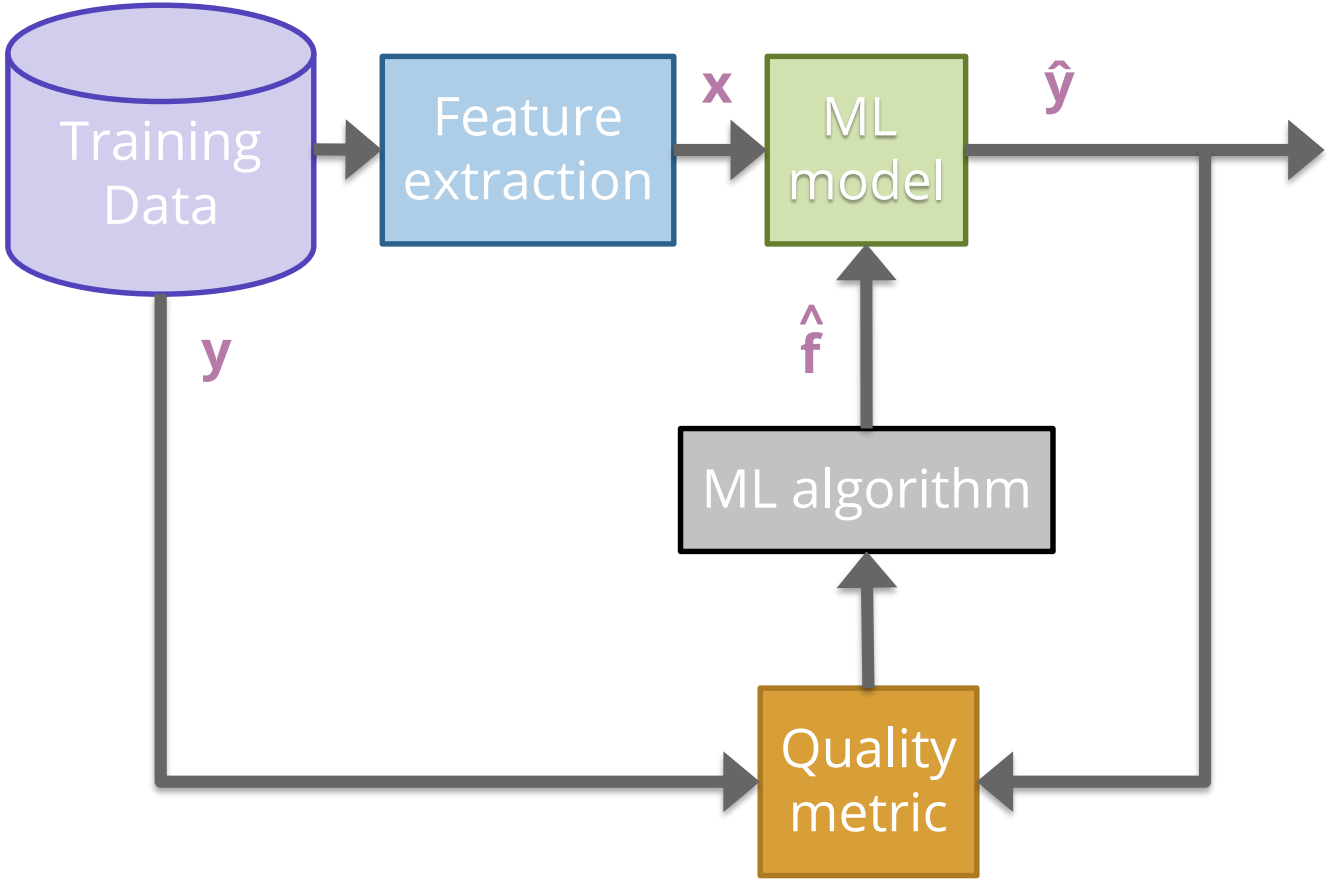
- [A1: Concept] Gradescope
- [A1: Programming] EdStem

HW2 goes out on Wednesday and is due next Tuesday (like regular)

- HW assignments are weighted equally!

Learning Reflections

- Great job on your first learning reflections! We saw a lot of great stuff while looking through those! Keep it up!
- Learning reflections due weekly on Fridays (same required components each time)


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 w.

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?

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

Poll Everywhere

Think

1.5 Minutes

pollev.com/cs416

How does λ affect the bias and variance of the model? For each underlined section, select "Low" or "High" appropriately.

When $\lambda = 0$

The model has **(Low / High)** Bias and **(Low / High)** Variance.

When $\lambda = \infty$

The model has **(Low / High)** Bias and **(Low / High)** Variance.

Poll Everywhere

Group

3 Minutes

pollev.com/cs416

How does λ affect the bias and variance of the model? For each underlined section, select "Low" or "High" appropriately.

When $\lambda = 0$ → Complex Model

The model has **(Low / High)** Bias and **(Low / High)** Variance.

When $\lambda = \infty$ → simple model

The model has **(Low / High)** Bias and **(Low / High)** Variance.

Brain Break

3 min -> 3:29

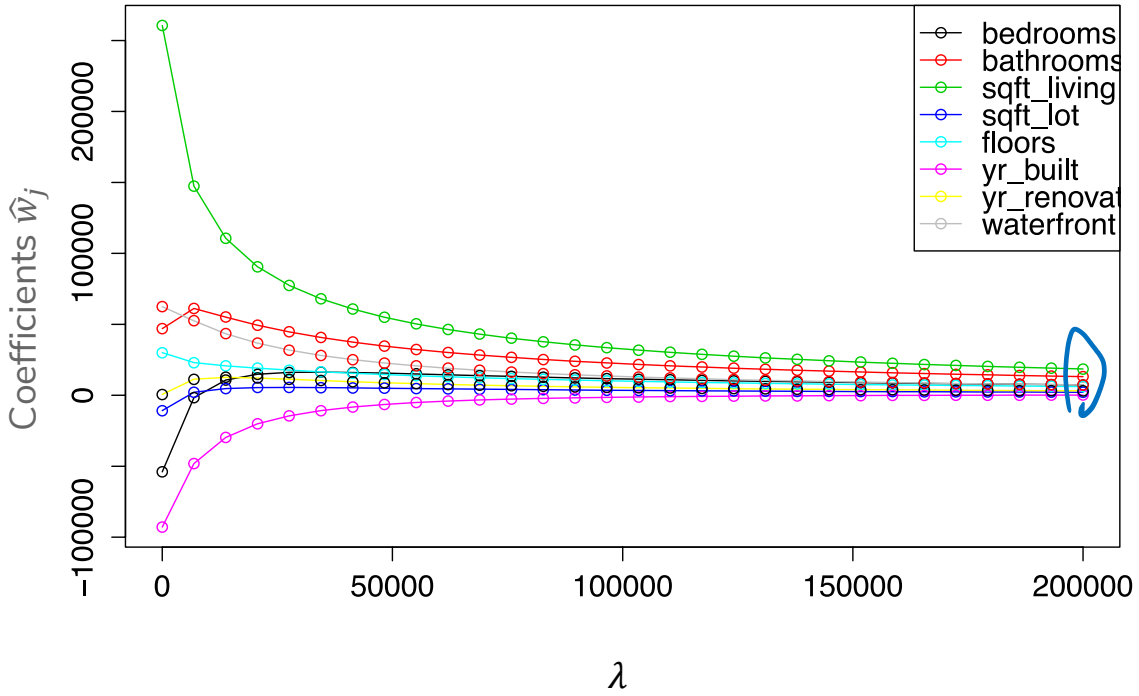
Demo: Ridge Regression

See Jupyter Notebook for interactive visualization.

Shows relationship between

- Regression line
- Residual Sum of Squares Quality Metric
  - Also called Ordinary Least Squares
- Ridge Regression Quality Metric
- Coefficient Paths

Coefficient Paths



Choosing λ

In the pre-lecture video for next time, we will talk about the procedure to choose the right λ.

- *Hint: It involves using a validation set or cross validation!*

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 y value units might be really high!

- My demo before does this wrong and penalizes w_0 as well!

Two ways of dealing with this

- Change the measure of overfitting to not include the intercept

$$\min_{w_0, w_{rest}} RSS(w_0, w_{rest}) + \lambda \left|\left|w_{rest}\right|\right|_2^2$$

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

Poll Everywhere

Think

1.5 Minute

pollev.com/cs416

How would the coefficient change if we change the scale of our feature?

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

- Say we learned a coefficient $\hat{w}_1$ for that feature
- What happens if we change the unit of x to square **miles?** Would $\hat{w}_1$ need to change?

a) The $\hat{w}_1$ in the new model with sqr. miles would be larger

b) The $\hat{w}_1$ in the new model with sqr. miles would be smaller

c) The $\hat{w}_1$ in the new model with sqr. miles would stay the same

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