CSE/STAT 416

Cross Validation; Ridge Regression

Pemi Nguyen
University of Washington
April 4, 2022

Slides by Hunter Schafer

Who am I?

Pemi Nguyen
Lecturer
he/him
peming@cs

Background

- Former Teaching Assistant and Content Development Contributor for a lot of CSE courses (especially for CSE 446/546) for 7 quarters
- Former NLP Researcher at UWNLP on misinformation detection
- Software Engineer at Facebook (starting June 2022)
- Disability & Accessibility Advocate

Contact

- Course Content + Logistics: EdStem
- Personal Matters: peming@cs.washington.edu / Office Hours
- Let met know if I can help you with anything (academic struggles, internships, career advice, accommodations, etc.)

Fun (or not) facts:

- I used to fail classes when I started out in college (not the proudest thing). You can always re-bounce from failures.
- I did a lot of traveling to a lot of countries which helped me gain great perspectives in life, and appreciation for cultural diversity
- I have a severe hearing disability (since birth). If I can’t hear you well in person, please kindly repeat what you said ☺.

Recap week 1

Lecture 1:

Different ML tasks (supervised, unsupervised)

A focus on regression task, specifically linear regression

(x, y)

Different components of a ML pipeline

input

output = continuous

Lecture 2:

very reliable

Model complexity

not reliable

Train vs. Test vs. True error ← not possible to calculate

Overfitting and Underfitting

Bias-Variance Tradeoff

Error as a function of train set size

Choosing best model complexity

- Validation set
- Cross Validation

Lecture Logistics

To encourage participation and welcome questions from students attending live lectures, I will create an EdStem post pinned on top during every lecture.

Please write comments or questions that you have (and set them **anonymously** if you need). This looks like the Zoom functionality where you can write comments live. It’s harder for in-person lecture, but let’s just try this ☺

You’re also encouraged to verbally ask questions if you think it’s better doing that

I will pause occasionally during lectures to make sure I answer every question needed.

Administrivia

Homework 1 Due Friday night! Remember to do three parts

- Gradescope Quiz
- Written questions
- Programming

HW2 goes out on Friday (like regular)

- HW assignments are weighted equally!

Pairing results out today. We will email the ones that fill out directly.

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 Sundays (same required components each time)

Poll Everywhere

Think

1 Minute

pollev.com/cs416

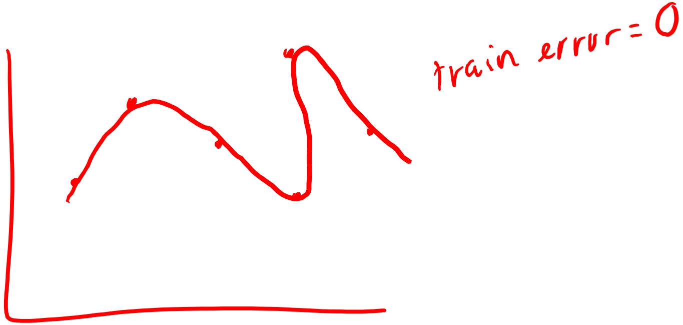
What describes overfitting?

Low train error, low test error

High train error, high test error

[✓ Low train error, high test error]

High train error, low test error



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. However, the test set has been **tampered**, thus is no longer is an unbiased estimate of the true error.
>
> 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.

NEVER EVER EVER touch the test set until the end. You only use it ONCE to evaluate the performance of the best model you have selected during training.

Choosing Complexity

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

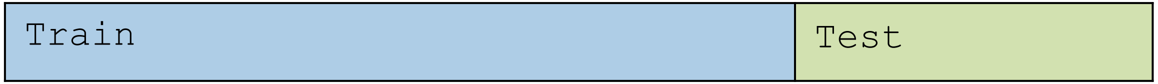
Using a validation set

Doing cross validation

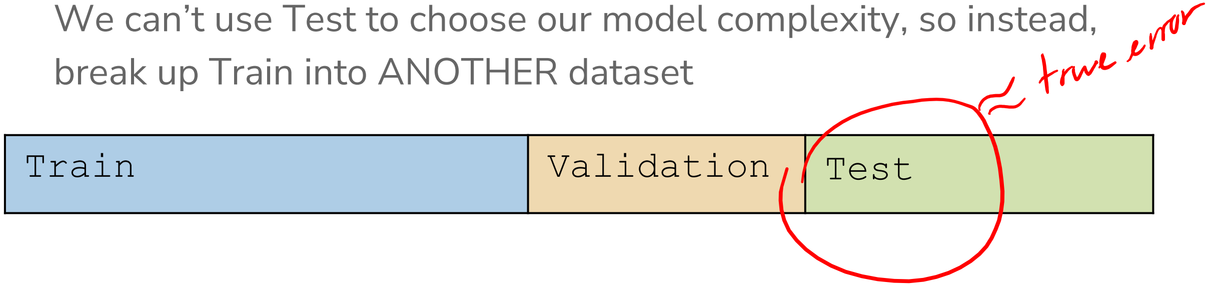
Note: Even though we use these two approaches for choosing the best model complexity, they are also used for **fine-tuning hyperparameters**, which we'll talk later.

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



We will pick the model that does best on validation. Note that this now makes the validation error of the "best" model a biased estimate of true error. The test error will be an unbiased estimate though since we never looked at it!

Validation Set

The process generally goes

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

Validation Set

Pros

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

Easy to implement

Very fast, and is widely used nowadays in Deep Learning

Cons

Have to sacrifice even more training data ⇐ suitable for Deep Learning

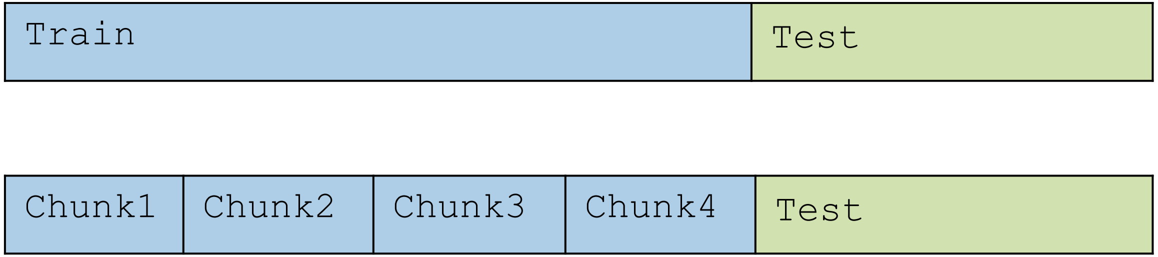
Chance of overfitting is less than when training on the full training set, but there's still a possibility of overfitting

Cross-Validation

(k-fold)

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

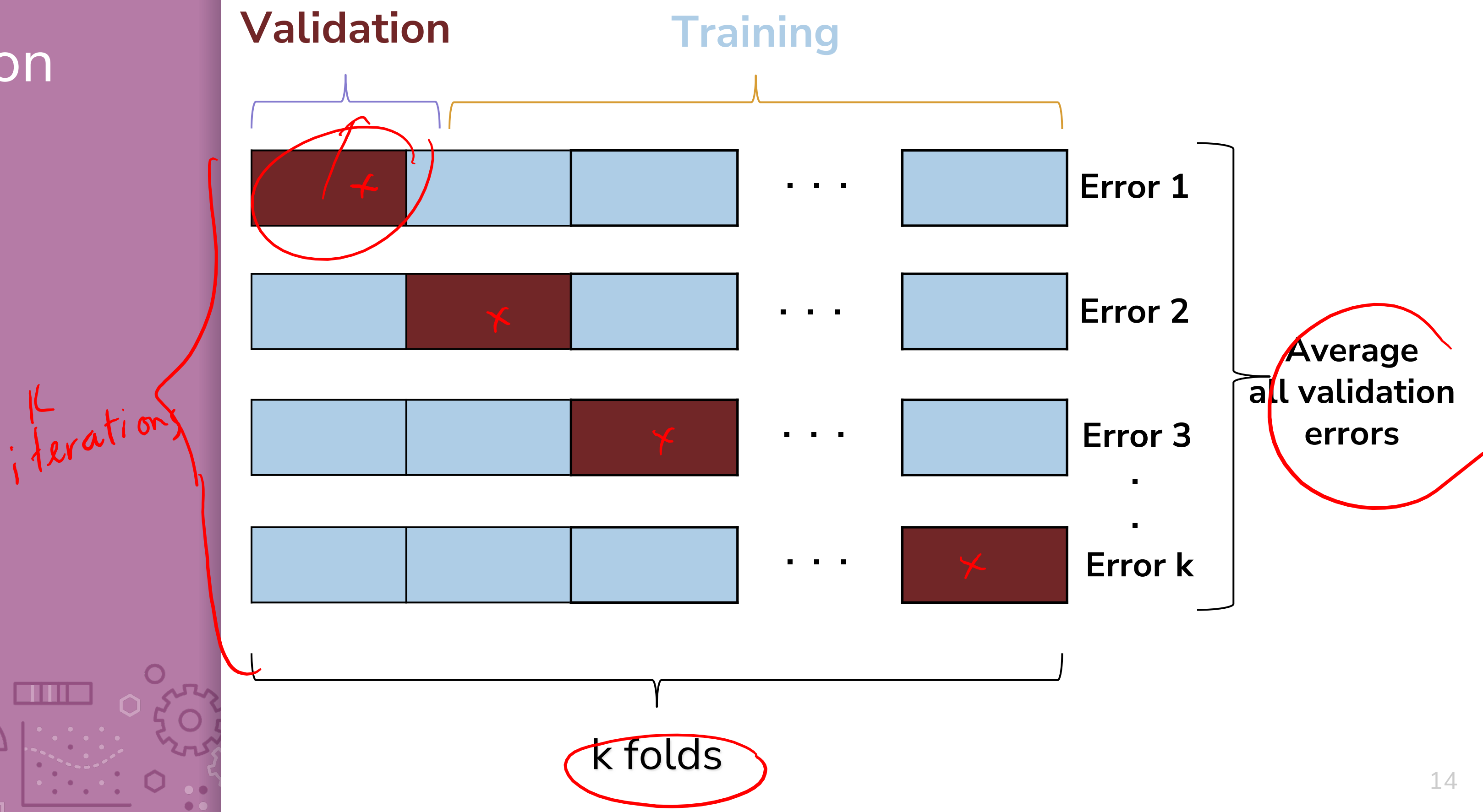
Still need to break off our test set like before. After doing so, break the training set into k chunks.



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

For each model complexity / a set of hyperparameters, perform Cross Validation with k iterations and get an average validation error



Cross-Validation

The process generally goes:

```
train_set, test_set = random_split(dataset)
randomly shuffle train_set
choose a specific k and split it into k groups *
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
retrain the model with best complexity p on the
full training set
return the error of that model on the test set
```

*: we expect to split the groups evenly, but the last one might not have the same number of datapoints like the rest of the training set

Cross-Validation

Pros

Prevent overfitting: By training the model on multiple folds instead of only 1 training set, this allow the model with the best generalization capabilities.

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

Cons

Very slow. For each model selection, we have to train k times

Very computationally expensive

The choice of k also follows bias-variance tradeoff.

Lower k: High bias / Higher k: High variance

In practice, people use k = 10 because it achieves a fine medium balance

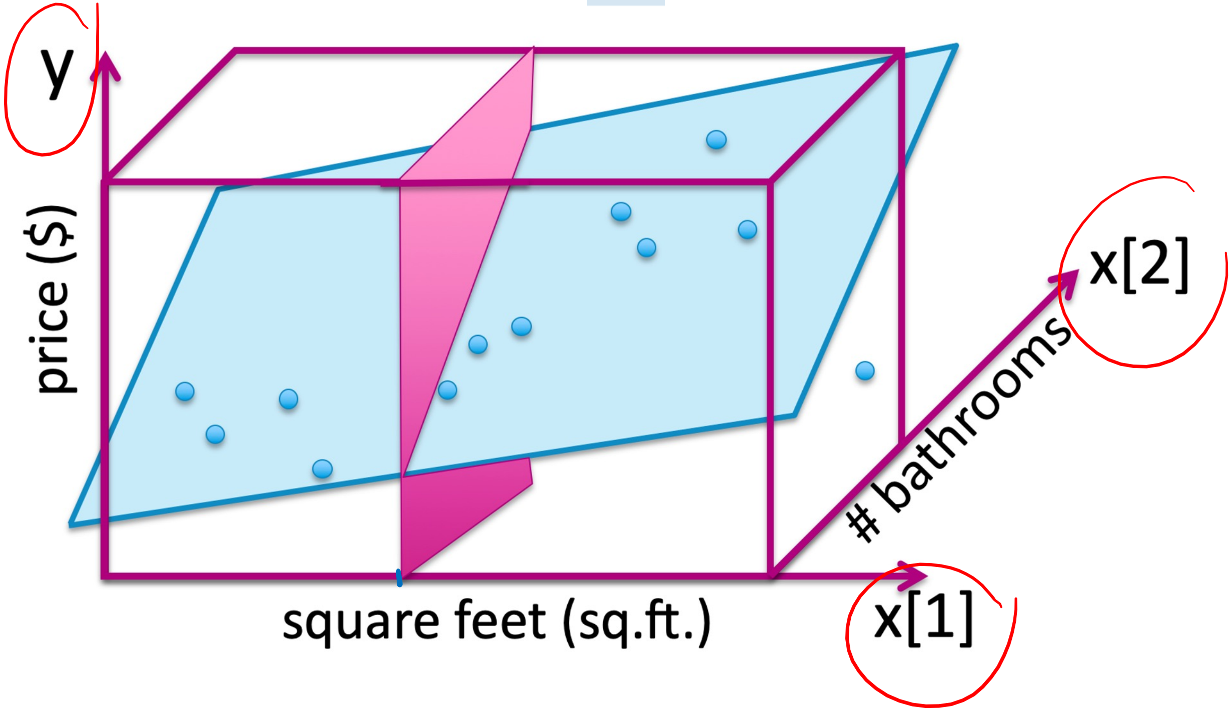
Pre-Lecture Video 2

Coefficients/Weights *and Overfitting*

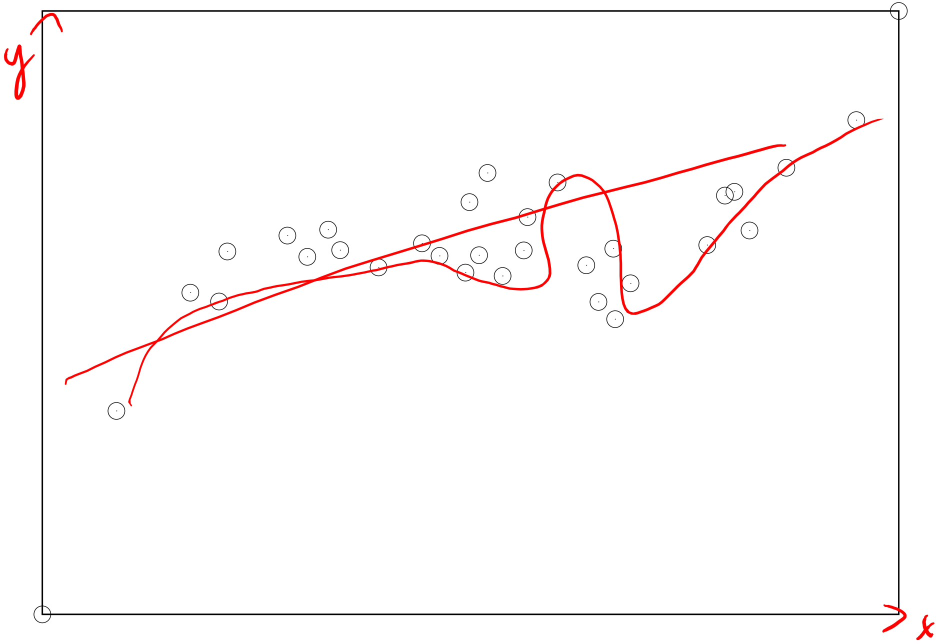
Interpreting Coefficients

Interpreting Coefficients – Multiple Linear Regression

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



Overfitting

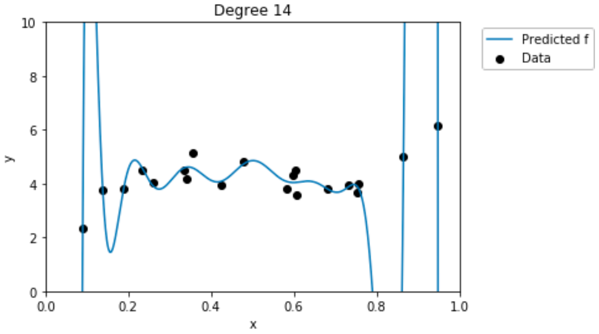


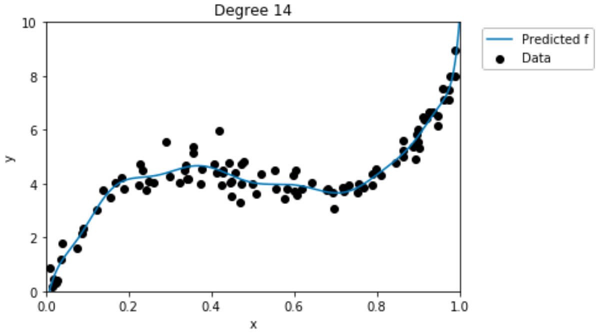
Often, overfitting is associated with very large estimated parameters $\hat{w}$!

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.





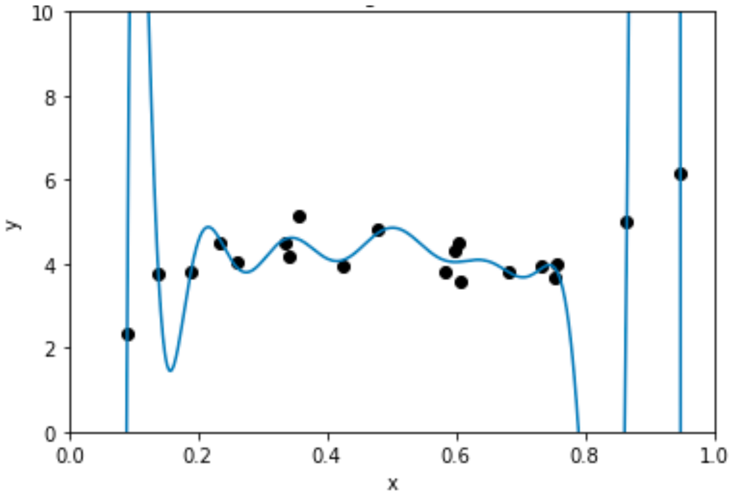
Number of Features

How do the number of features affect overfitting?

1 feature

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

HARD



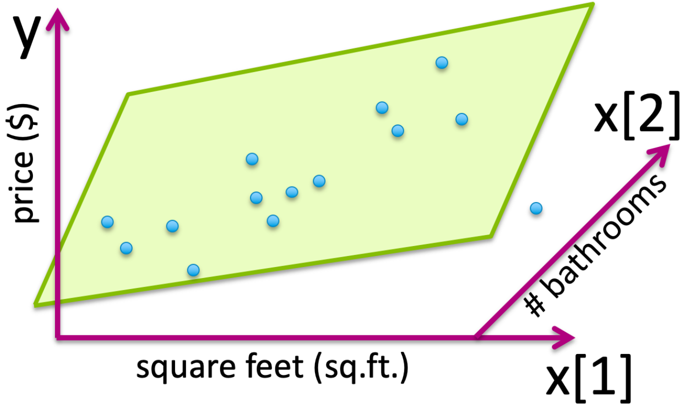
Number of Features

How do the number of features affect overfitting?

D features

Data must include representative example of all $((x[1], x[2], \ldots, x[D]), y)$ combos to avoid overfitting!

MUCH HARDER!!



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!

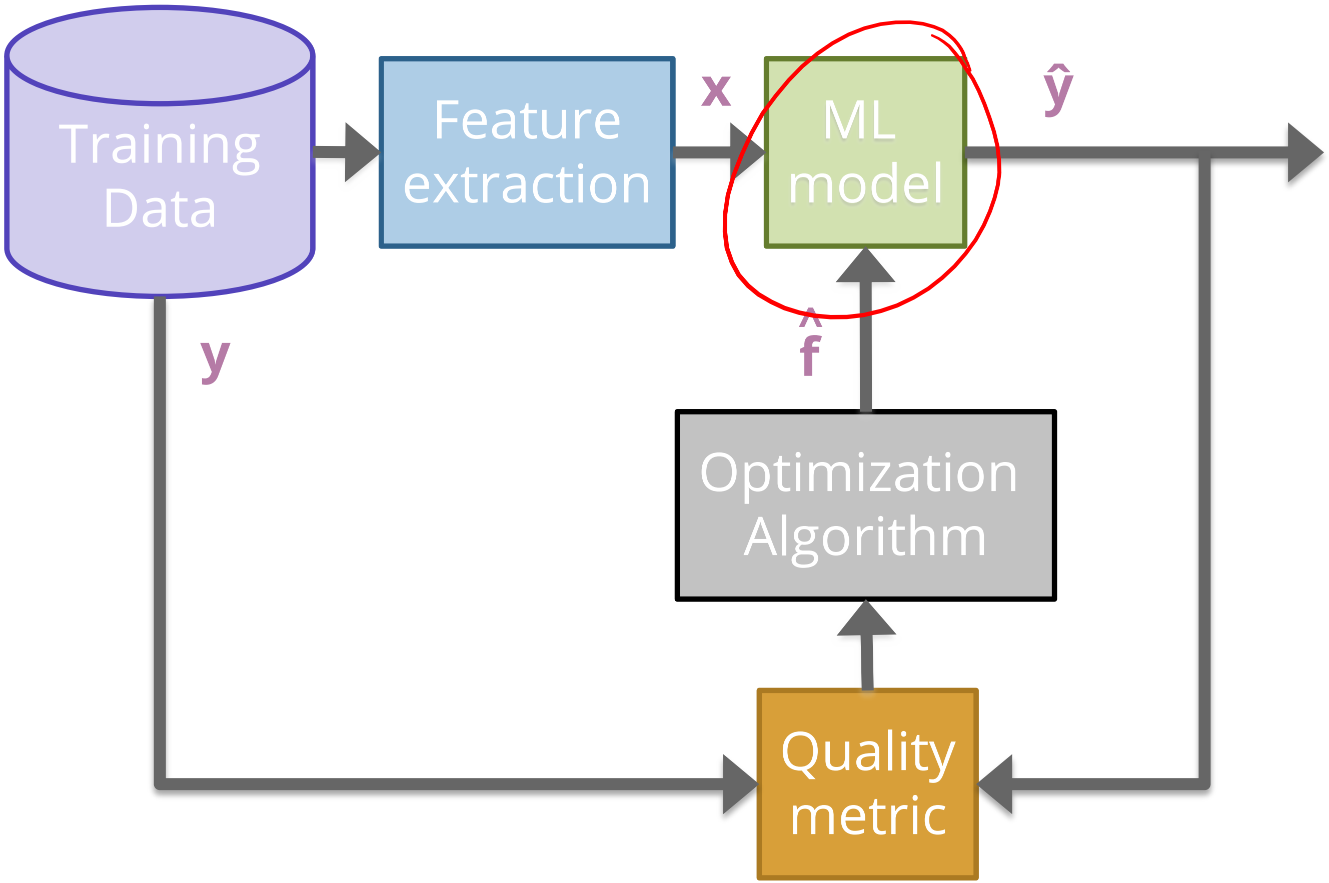
- For d inputs, there are 2^d subsets of those inputs!

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


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

Regularization

loss

Before, we used the quality metric that minimized loss

$$\hat{w} = \underset{w}{\operatorname{argmin}}\, \underline{L(w)}$$

Change quality metric to balance loss with measure of overfitting

$L(w)$ is the measure of fit

$R(w)$ measures the magnitude of coefficients

$$\hat{w} = \underset{w}{\operatorname{argmin}}\, L(w) \boxed{+ \underline{\lambda}\, \boxed{R(w)}}$$

λ: regularization parameter

How do we actually measure the magnitude of coefficients?

larger λ → more regularization effects

Magnitude

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

$w_1 = 1000$ ← big

$w_2 = -1000$ ← big

Sum?

$\cancel{R(w) =} w_1 + \dots w_d$

not a good approach.

Sum of absolute values?

$R(w) = |w_1| + \dots + |w_d| =$ — L_1-regularization

$= ||w||_1$ ← L_1-Norm

Sum of squares?

$R(w) = w_1^2 + \dots w_d^2$

$= ||w||_2^2$ ← L_2 norm

$\boxed{p\text{-norm}: \ ||w||_p^p = |w_1|^p + |w_2|^p + \dots |w_d|^p}$

Magnitude

Come up with some number that summarizes the magnitude of the weights w.

$$\hat{w} = \underset{w}{\operatorname{argmin}}\, MSE(w) + \underline{\lambda R(w)}$$

Sum?

$$R(w) = w_0 + w_1 + \cdots + w_d$$

~~Doesn't work because the weights can cancel out (e.g. $w_0 = 1000, w_1 = -1000$) which so $R(w)$ doesn't reflect the magnitudes of the weights~~

Sum of absolute values?

$$R(w) = |w_0| + |w_1| + \cdots + |w_d| = \|w\|_1$$

It works! We're using L1-norm, for L1-regularization (LASSO)

Sum of squares?

$$R(w) = |w_0|^2 + |w_1|^2 + \ \ldots + |w_d|^2 = w_0^2 + w_1^2 + \ \ldots + w_d^2 = \|w\|_2^2$$

It works! We're using L2-norm, for L2-regularization (Ridge Regression)

Note: Definition of p-Norm: $\|w\|_p^p = |w_0|^p + |w_1|^p + \ \ldots + |w_d|^p$

Ridge Regression

Change quality metric to minimize

$$\hat{w} = \underset{w}{\overset{\text{arg}}{\min}}\, MSE(W) + \lambda\|w\|_2^2 = L'(w)$$

← L_2-regularization

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

What if $\lambda = 0$?

$$\hat{w} = \arg\min_{w} MSE(w) = \hat{w}_{OLS}$$

What if $\lambda = \infty$?

Case 1: $w = \vec{0} \Rightarrow L'(w) = MSE(\vec{0}) + \lambda \cdot 0 = \boxed{MSE(\vec{0})}$

Case 2: At least 1 weight $w_i \neq 0$

$$\Rightarrow L'(w) = MSE(w) + \underbrace{\underbrace{\lambda}_{\infty} \cdot \underbrace{\|w\|_2^2}_{\neq 0}}_{\infty} = MSE(w) + \infty = \boxed{\infty}$$

λ ~~in between~~? $\Rightarrow \hat{w} = \vec{0}$ ← 1 solution → high bias / underfitting

Ridge Regression

Change quality metric to minimize

$$\hat{w} = \underset{w}{\operatorname{argmin}}\, L(w) = \underset{w}{\operatorname{argmin}}\, MSE(w) + \lambda\|w\|_2^2 = L'(w)$$

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

What if $\lambda = 0$?

$$\hat{w} = \underset{w}{\operatorname{argmin}}\, MSE(w) = \hat{w}_{OLS}$$

What if $\lambda = \infty$?

Case 1: $w = \vec{0}$, then $\|w\|_2^2 = 0$, which means:

$$L(w) = MSE(\vec{0}) = \text{constant}$$

Case 2: w contains at least 1 $w_i \neq 0$, then $\|w\|_2^2 > 0$:

$$L(w) = MSE(w) + \lambda\,\|w\|_2^2 \geq \lambda\,\|w\|_2^2 = \infty$$

Since we're trying to minimize $L(w)$, $\min L(w) = 0$, so $\hat{w} = \vec{0}$.

λ in between?

$$0 \leq \|w\|_2^2 \leq \|w_{OLS}\|_2^2$$

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.

$\hat{w} = \vec{0}$

When $\lambda = \infty$

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

Brain Break

Demo: Ridge Regression

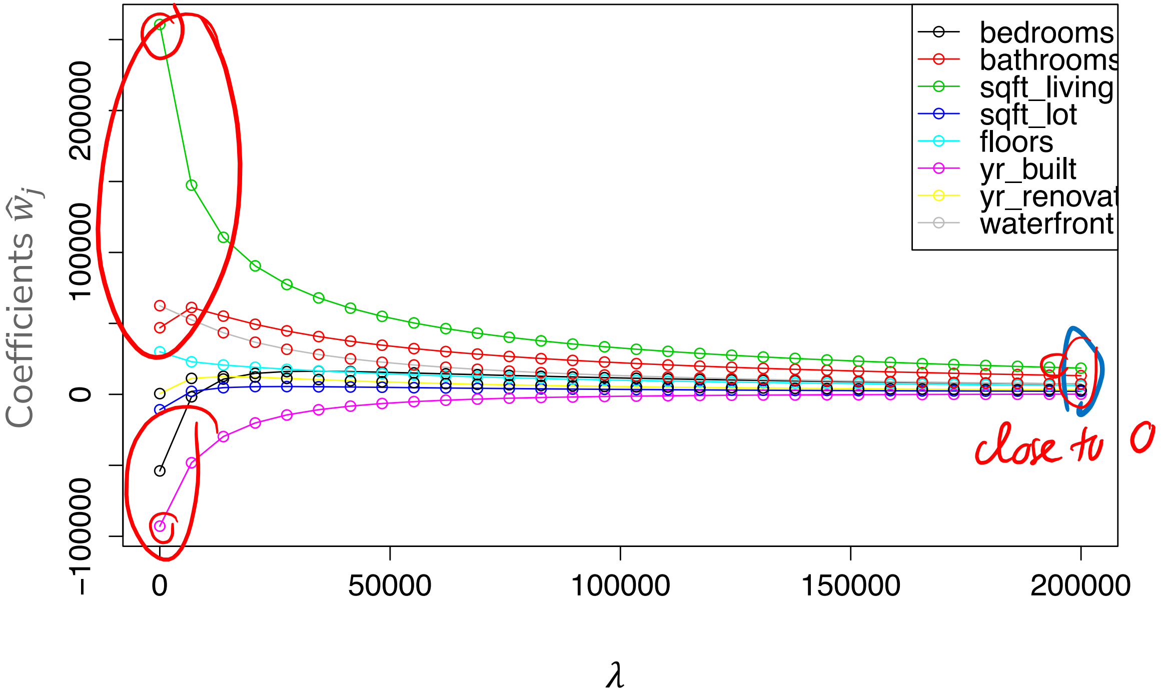
See Jupyter Notebook for interactive visualization.

Shows relationship between

- Regression line
- Mean Square Error
  - Also called Ordinary Least Squares
- Ridge Regression Quality Metric
- Coefficient Paths

Coefficient Paths

/L_2



Poll Everywhere

Think

2 minutes

pollev.com/cs416

What hyperparameters we have learned so far?

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!*

Hyperparameters

There are a number of hyperparameters such as:

Learning rate (in Gradient Descent)

Number of iterations (in Gradient Descent)

Regularization parameter λ

Number of folds (in Cross Validation)

A **hyperparameter** is a parameter whose value is used to control the learning process and is external to the model. By contrast, the values of model **parameters** are derived via training

Fine-tuning hyperparameters is the process of choosing an optimal set of hyperparameters for the training process. It’s painstaking process and there are still a lot of unknowns.

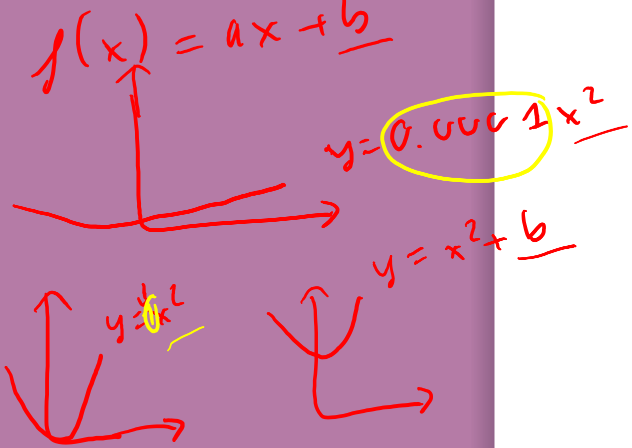
Regularization

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 (Bias term)



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! Also, the intercept doesn't affect the curvature of a loss function (it's just a linear scale).

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

One way of dealing with this

Change the measure of overfitting to not include the intercept

$$\underset{w_0, w_{rest}}{\operatorname{argmin}} MSE(\overset{\text{bias}}{\underline{w_0}}, w_{rest}) + \lambda ||w_{rest}||_2^2$$

Note: we're referring to bias here as the intercept term (constant) in mathematical functions. It's different from bias in bias and variance

Poll Everywhere

Think

1.5 Minute

pollev.com/cs416

$\hat{w} = 1$ $= 1000^2$

$x = 10\,m^2 = \frac{10}{(1000)^2}\,km^2$

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

Consider our housing example with $(\overset{x}{sq.ft.}, \overset{y}{price})$ 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 $\hat{w}_1$ for that feature
- What happens if we change the unit of x to square **miles?** Would $\hat{w}_1$ need to change?
  - 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:

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

area (m²)	no of bedroom
100	2
200	3

mean = 0

std = 1

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