slido

Join at slido.com #3726997

ⓘ Start presenting to display the joining instructions on this slide.

Lecture 15

Cross Validation, Regularization

Different methods for ensuring the generalizability of our models to unseen data.

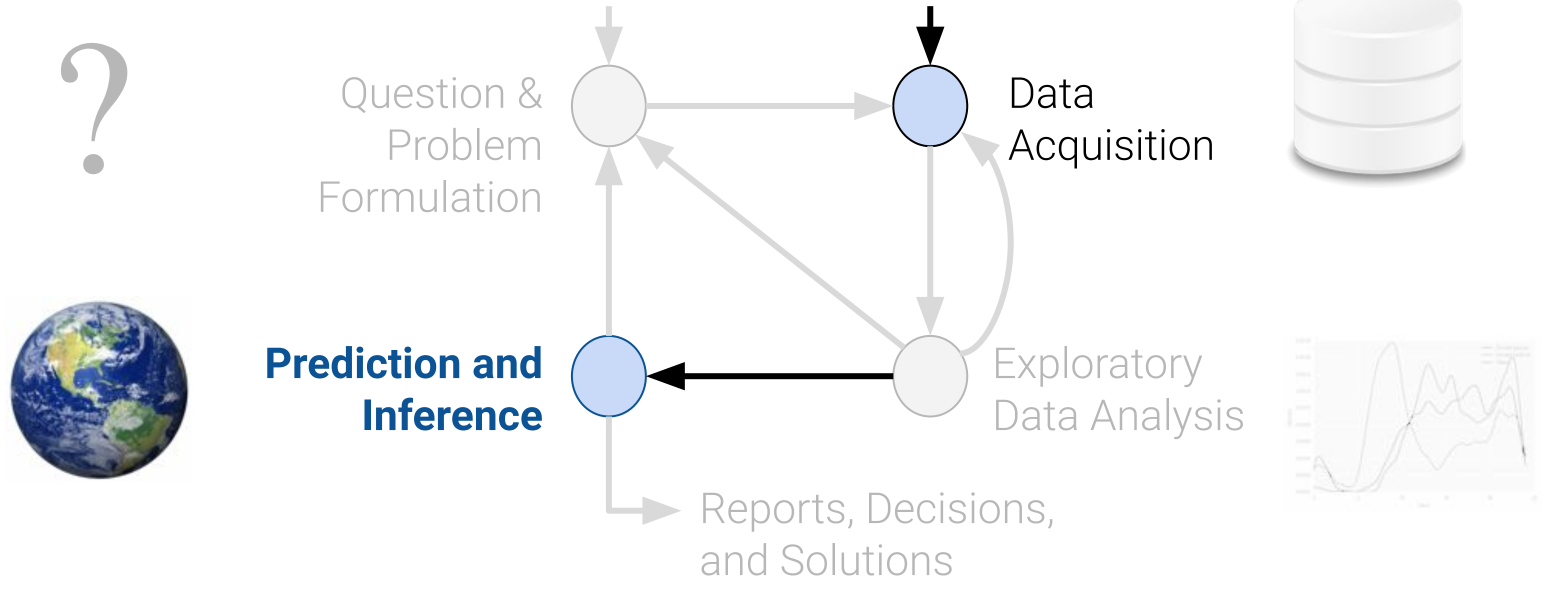
Data 100/Data 200, Spring 2023 @ UC Berkeley

Narges Norouzi and Lisa Yan

Content credit: Acknowledgments

Plan for Next Three Lectures: Model Selection

3726997


Question & Problem Formulation
Data Acquisition
Prediction and Inference
Exploratory Data Analysis
Reports, Decisions, and Solutions

(today)

Model Selection Basics: Cross Validation Regularization	**Probability I:** Random Variables Estimators	**Probability II:** Bias and Variance Inference/Multicollinearity

Today's Roadmap

Lecture 15, Data 100 Spring 2023

3726997

Cross Validation

- **The Holdout Method**
- K-Fold Cross Validation
- Test Sets

Regularization

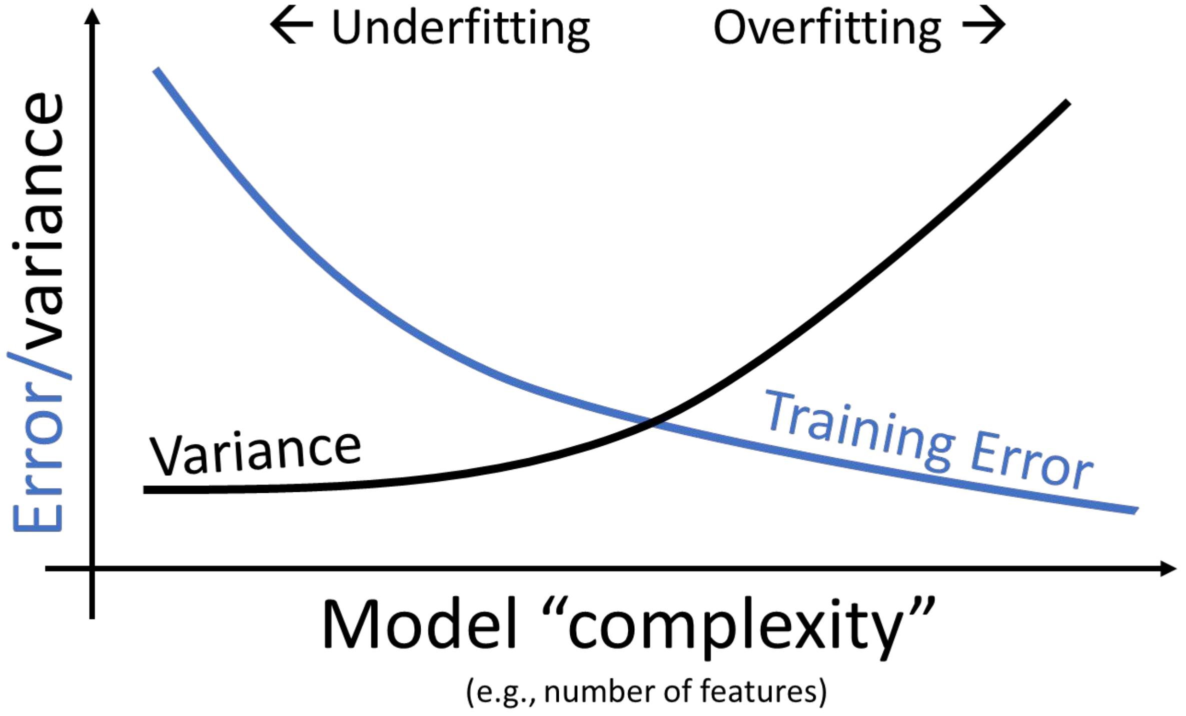
- L2 Regularization (Ridge)
- Scaling Data for Regularization
- L1 Regularization (LASSO)

Review: Error vs. Complexity

3726997

As we increase the complexity of our model:

- Training error decreases.
- Variance increases.

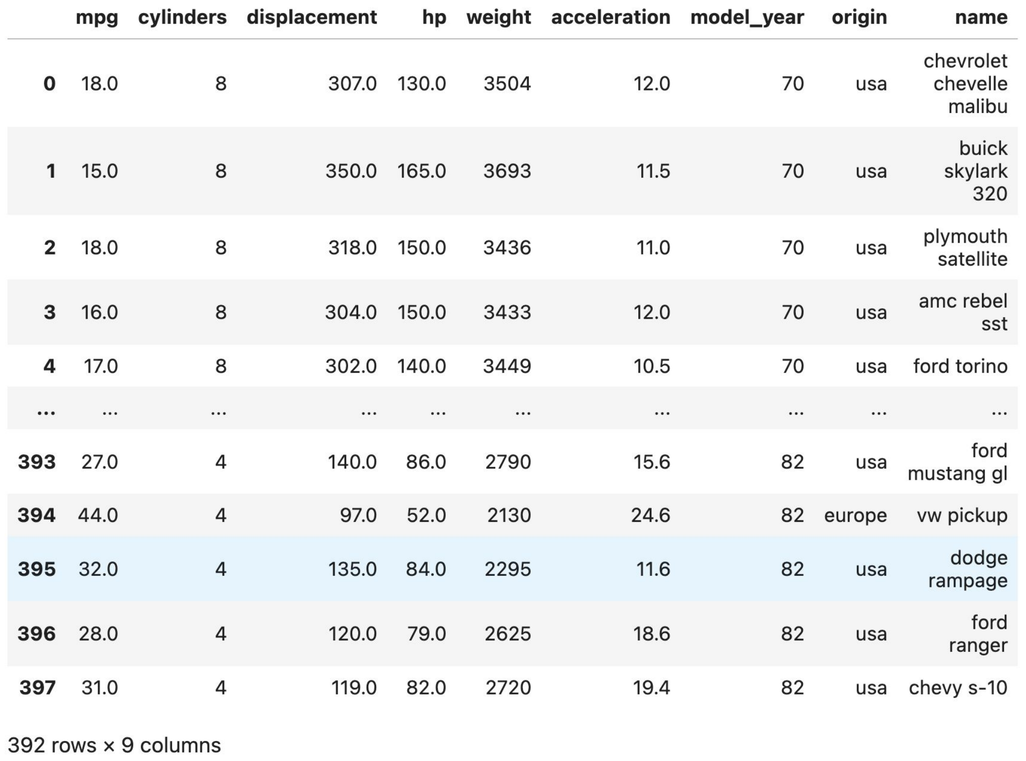


Review: Dataset

3726997

Today we will use the `mpg` dataset from the `seaborn` library.

The dataset has 392 rows and 9 column. Our task is to use some of the columns and their transformations to predict the value of the `mpg` column.



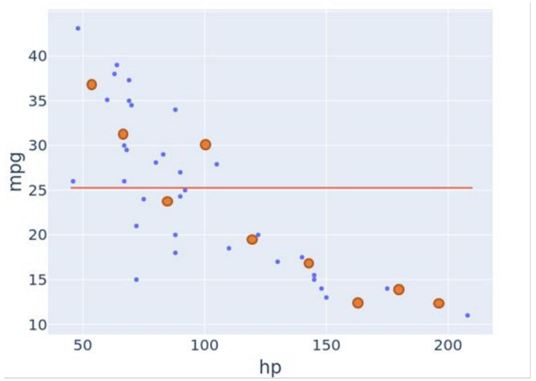
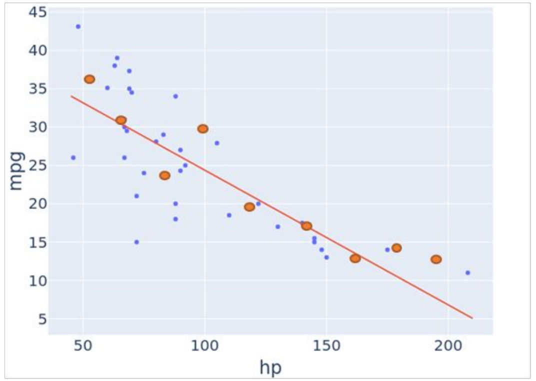
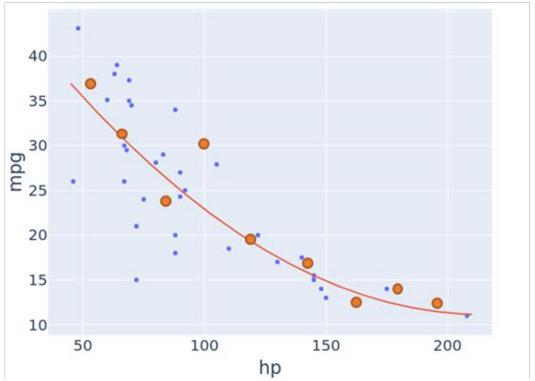
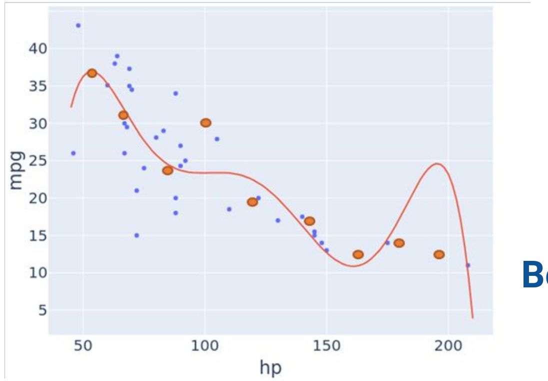
	mpg	cylinders	displacement	hp	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.0	2790	15.6	82	usa	ford mustang gl
394	44.0	4	97.0	52.0	2130	24.6	82	europe	vw pickup
395	32.0	4	135.0	84.0	2295	11.6	82	usa	dodge rampage
396	28.0	4	120.0	79.0	2625	18.6	82	usa	ford ranger
397	31.0	4	119.0	82.0	2720	19.4	82	usa	chevy s-10

392 rows × 9 columns

Review: Collecting More Data to Detect Overfitting

3726997

Suppose we use 35 sample to fit the regression model and collect the 9 new orange data points. We can compute MSE for our original models **without refitting using the new orange data points**.

k	MSE
0	72.091396
1	28.002727
2	25.835769
3	25.831592
4	25.763052
5	25.609403
6	23.269001

Original 35 data points

Best?

k	MSE
0	69.198210
1	31.189267
2	27.387612
3	29.127612
4	34.198272
5	37.182632
6	53.128712

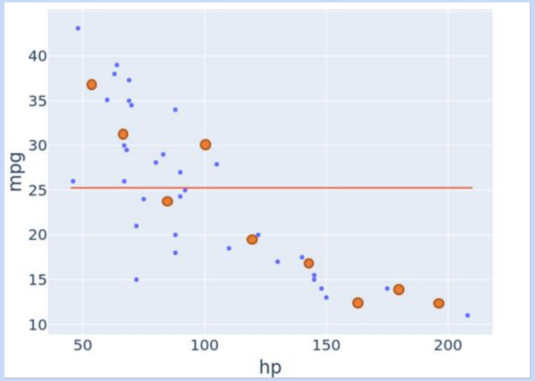
New 9 data points

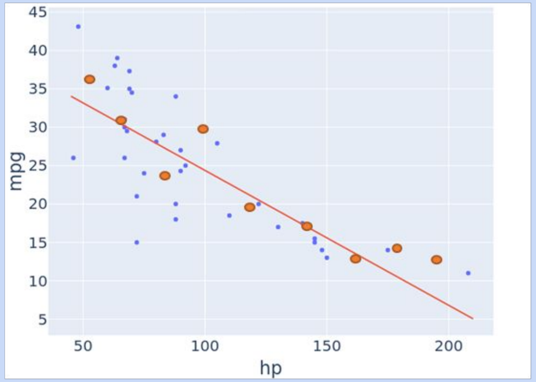
Best?

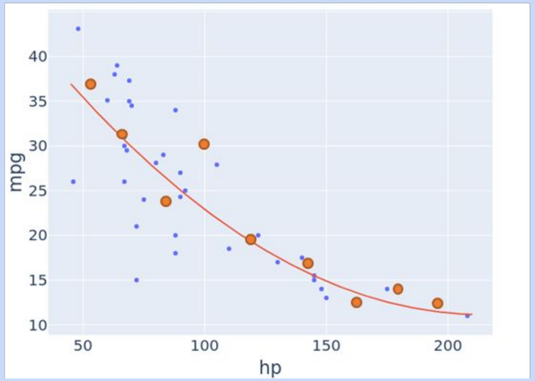
Review: Collecting More Data to Detect Overfitting

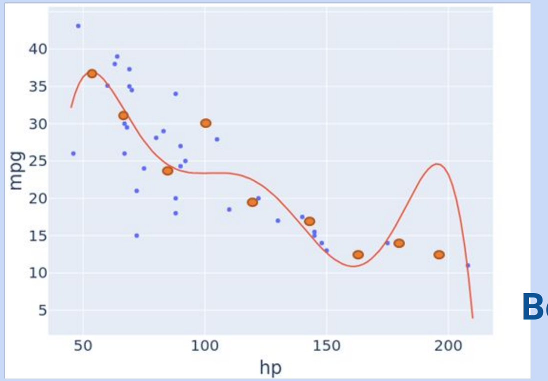
3726997

Which model do you like best? And why?









k	MSE
0	72.091396
1	28.002727
2	25.835769
3	25.831592
4	25.763052
5	25.609403
6	23.269001

Original 35 data points

Best?

k	MSE
0	69.198210
1	31.189267
2	27.387612
3	29.127612
4	34.198272
5	37.182632
6	53.128712

New 9 data points

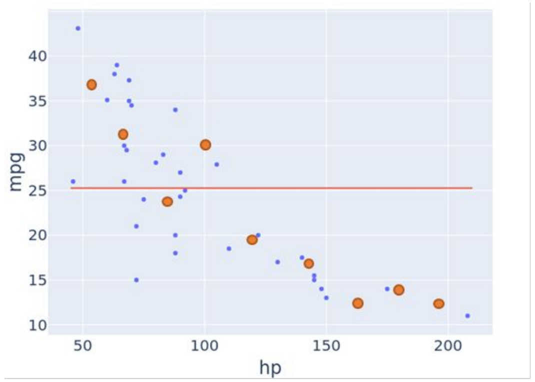
Best?

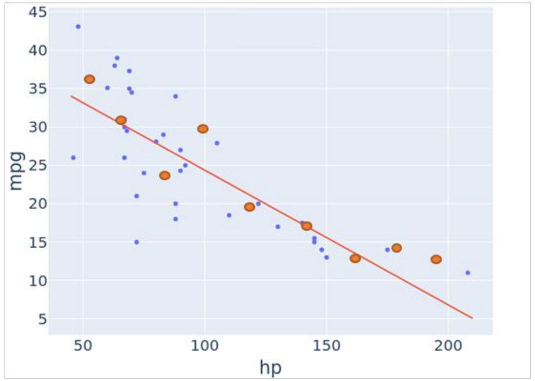
Review: Collecting More Data to Detect Overfitting

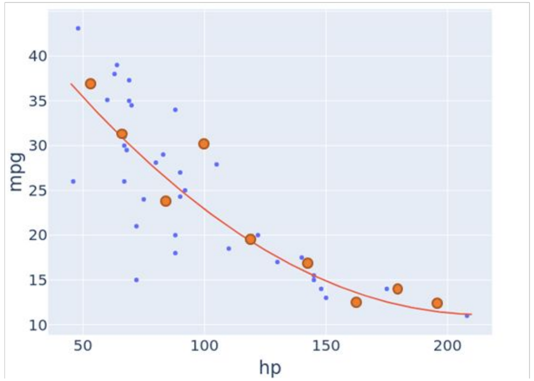
3726997

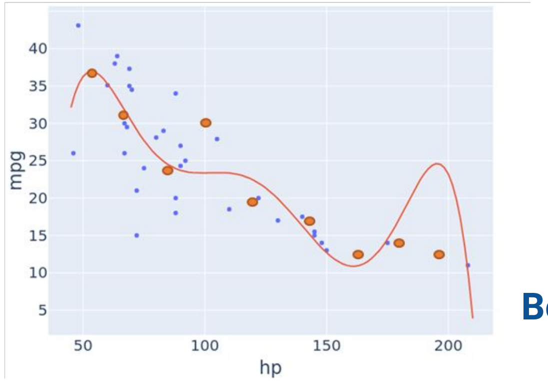
The order 2 model seems best to me.

- Performs best on data that has not yet been seen.









k	MSE
0	72.091396
1	28.002727
2	25.835769
3	25.831592
4	25.763052
5	25.609403
6	23.269001

Original 35 data points

Best?

k	MSE
0	69.198210
1	31.189267
2	27.387612
3	29.127612
4	34.198272
5	37.182632
6	53.128712

New 9 data points

Best?

Review: Collecting More Data to Detect Overfitting

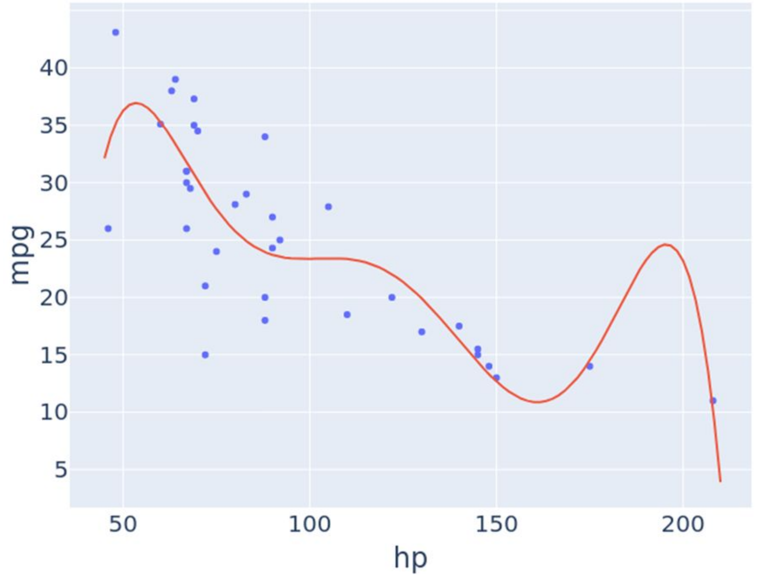
3726997

Suppose we have 7 models and don't know which is best.

- Can't necessarily trust the training error. We may have overfit!

We could wait for more data and see which of our 7 models does best on the new points.

- Unfortunately, that means we need to wait for more data. May be very expensive or time-consuming.
- "Will see an alternate approach next week."
  - As promised! Let's do it.



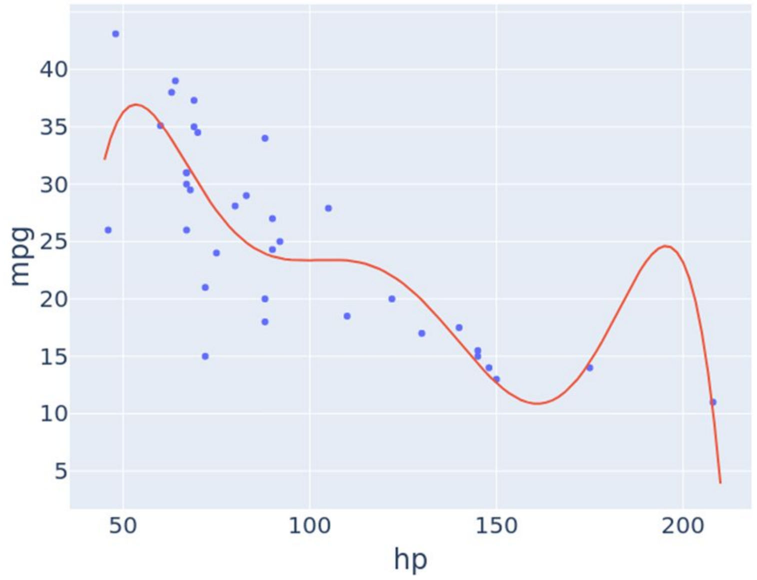
Idea 1: The Holdout Method

3726997

The simplest approach for avoiding overfitting is to keep some of our data secret from ourselves.

Example:

- Previous approach: We fit 7 models on all 35 of the available data points. Then waited for 9 new data points to decide which is best.
- Holdout Method: We **train** our models on all **25**/35 of the available data points. Then we **evaluate** the models' performance on the **remaining 10 data points**.
  - Data used to train is called the "**training set**".
  - **Held out data** is often called the "**validation set**" or "**development set**" or "**dev set**". These terms are all synonymous and used by different authors.

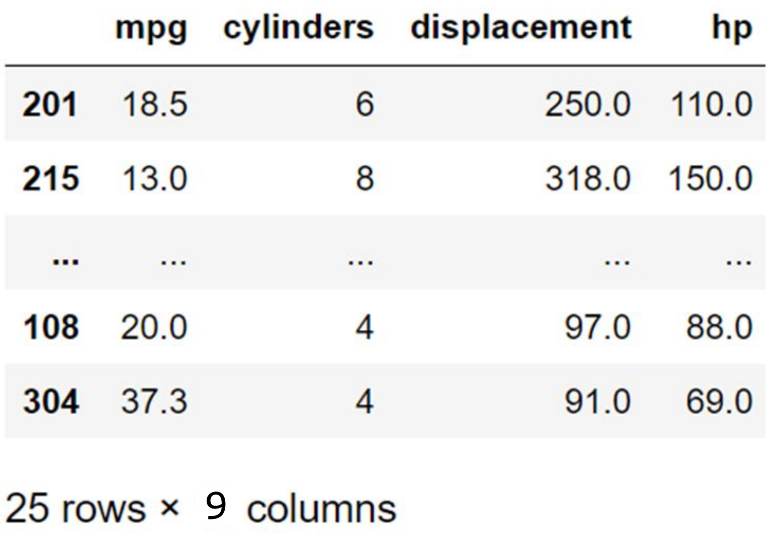


Holdout Set Demo

3726997

The code below splits our data into two sets of size 25 and 10.

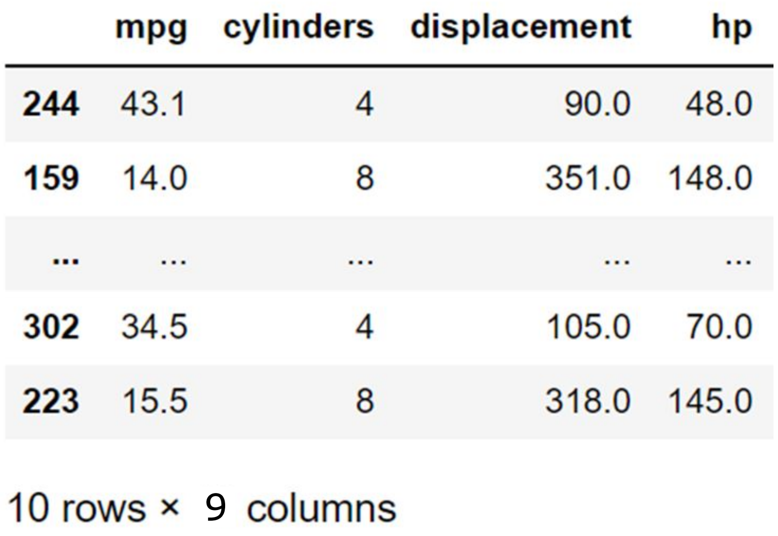
```
from sklearn.utils import shuffle
training_set, validation_set = np.split(shuffle(vehicle_data_sample_35), [25])
```



	mpg	cylinders	displacement	hp
201	18.5	6	250.0	110.0
215	13.0	8	318.0	150.0
...	...	...	...	...
108	20.0	4	97.0	88.0
304	37.3	4	91.0	69.0

25 rows × 9 columns

Used for **Training**



	mpg	cylinders	displacement	hp
244	43.1	4	90.0	48.0
159	14.0	8	351.0	148.0
...	...	...	...	...
302	34.5	4	105.0	70.0
223	15.5	8	318.0	145.0

10 rows × 9 columns

Used for **Evaluation**

Reflection: Shuffling

Question: Why did I shuffle first?

```
from sklearn.utils import shuffle
training_set, validation_set = np.split(shuffle(vehicle_data_sample_35), [25])
```

slido

Why did we shuffle the data before selecting the training and validation sets?

ⓘ Start presenting to display the poll results on this slide.

Reflection: Shuffling

Question: Why did I shuffle first?

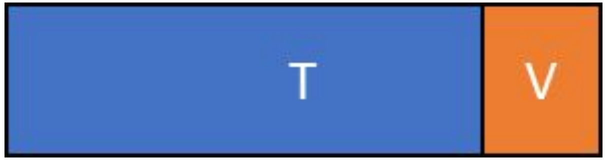
```
from sklearn.utils import shuffle
training_set, validation_set = np.split(shuffle(vehicle_data_sample_35), [25])
```

I'm using a large contiguous block of data as my validation set.

- If the set is sorted by something e.g. vehicle MPG, then my model will perform poorly on this unseen data.
  - Model will have never seen a high MPG vehicle.

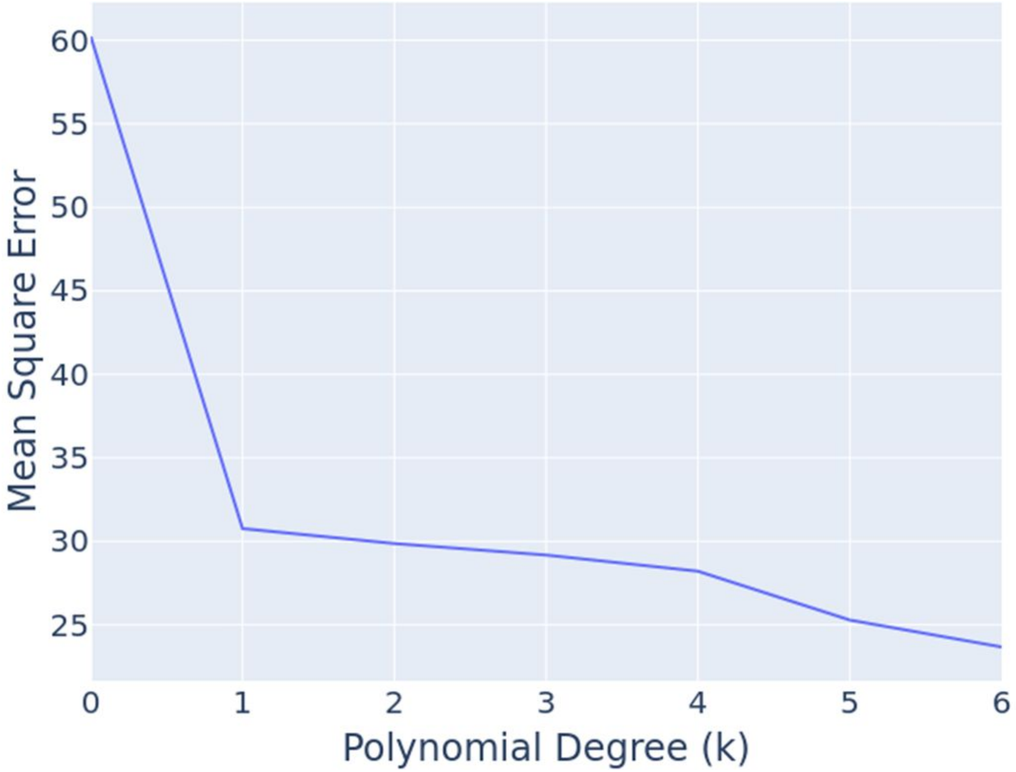
Shuffling prevents this problem.

- Alternate mathematically equivalent approach: Picking 10 samples randomly.



Hold Out Method Demo Step 1: Generating Models of Various Orders

First we **train** 7 models the experiment now on our **training set of 25 points**, yielding the MSEs shown below. As before, MSE decrease monotonically with model order.

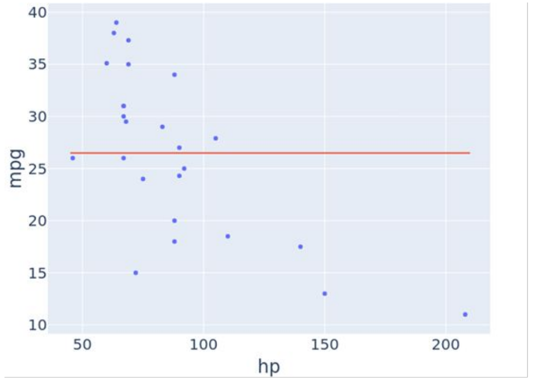


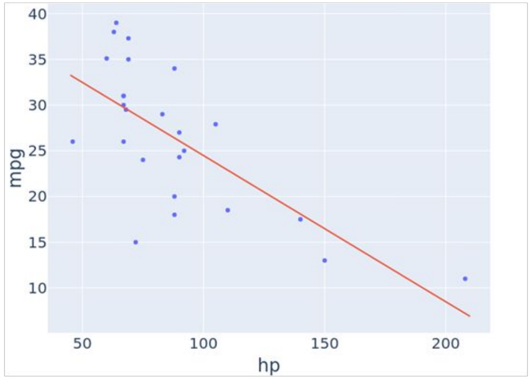
k	MSE
0	60.235744
1	30.756678
2	29.875269
3	29.180868
4	28.214850
5	25.290990
6	23.679651

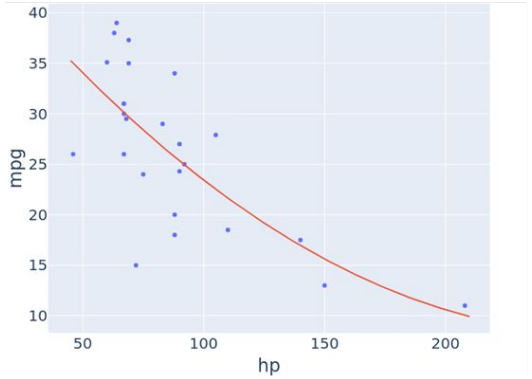
Hold Out Method Demo Step 1: Generating Models of Various Orders (Visualization)

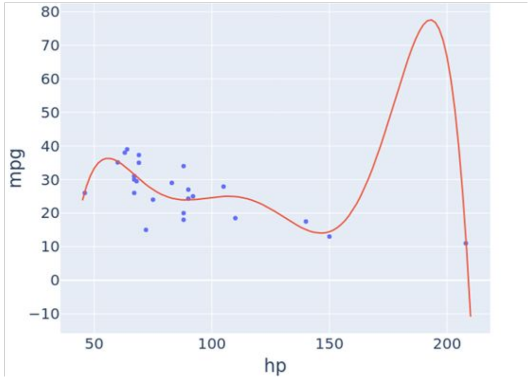
3726997

Below, we show the order 0, 1, 2, and 6 models trained on our **25 training points**.









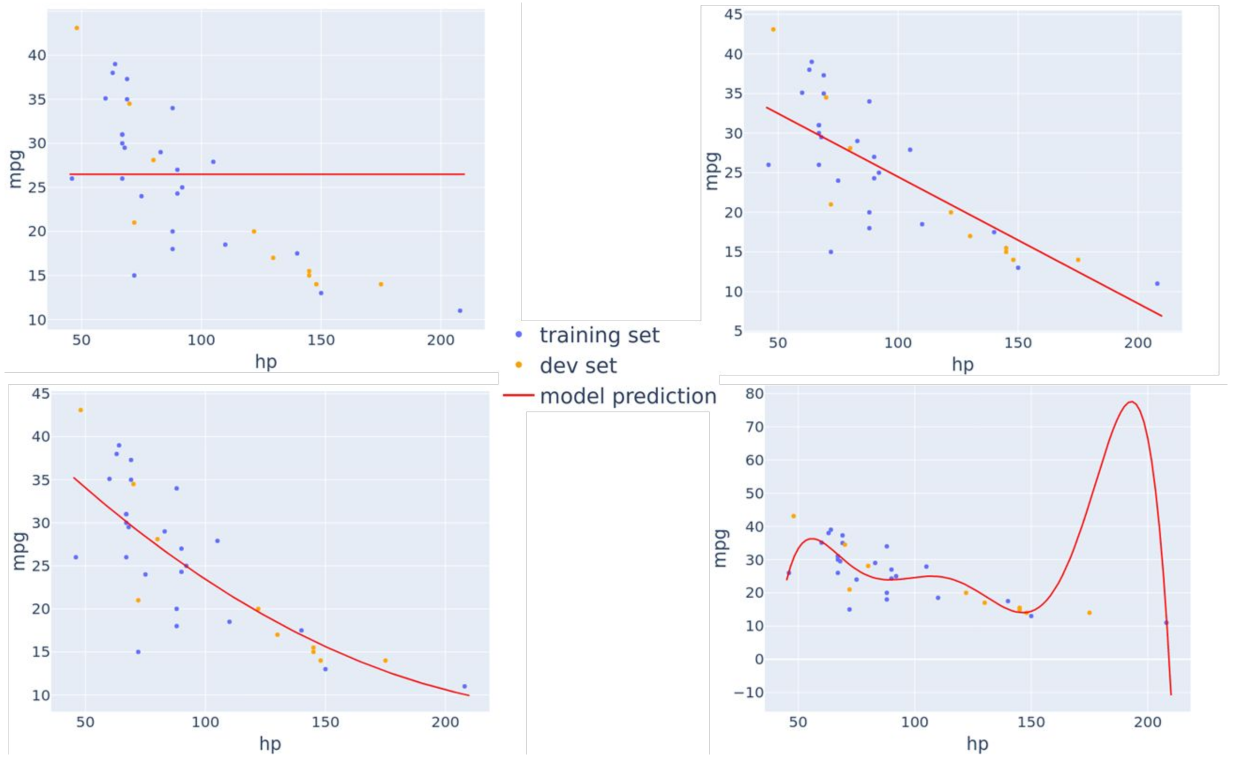
k	MSE
0	60.235744
1	30.756678
2	29.875269
3	29.180868
4	28.214850
5	25.290990
6	23.679651

Note: Our degree 6 model looks different than before. No surprise since variance is high and we're using a different data set.

Hold Out Method Demo Step 2: Evaluating on the Validation Set (a.k.a. Dev Set)

3726997

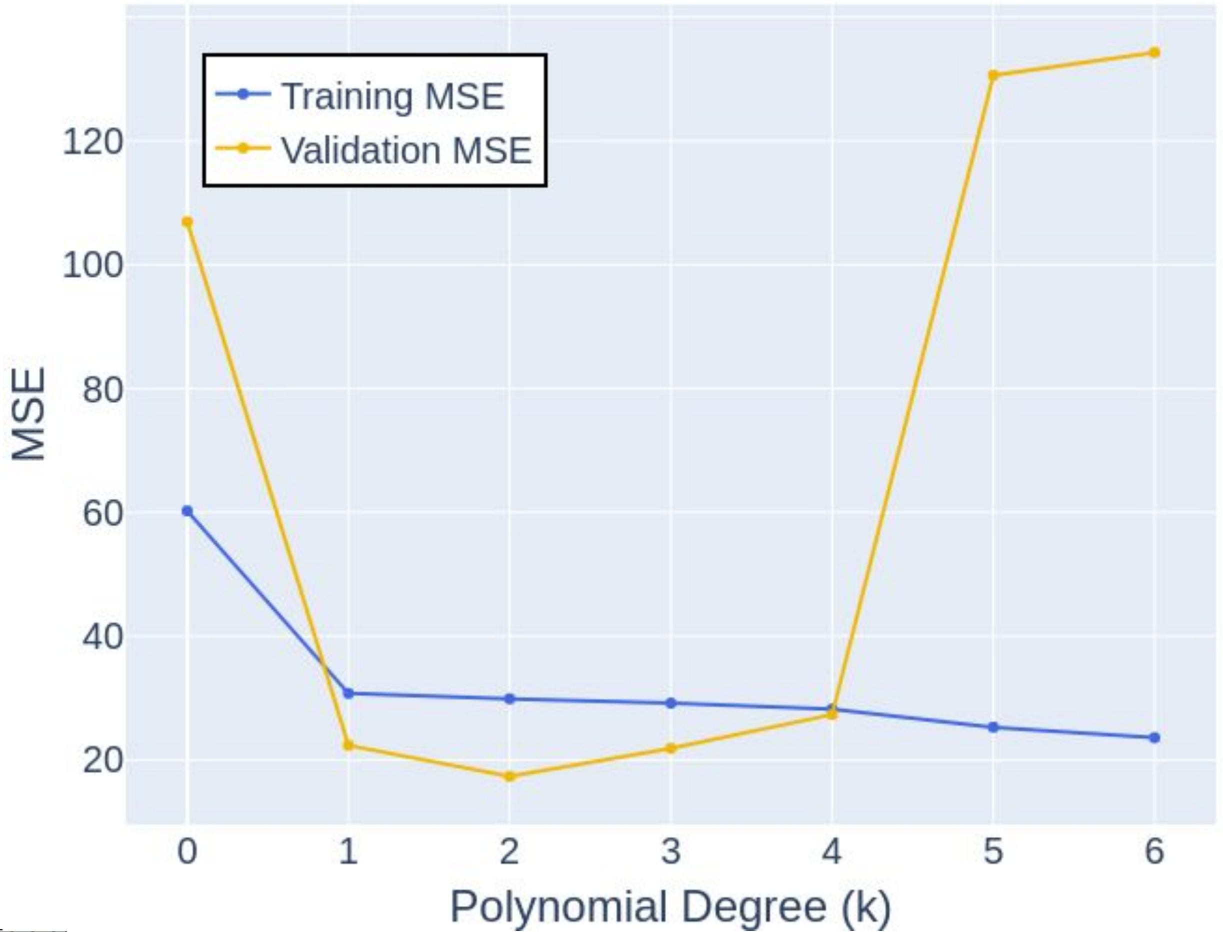
Then we compute MSE on our **10 validation set points** (in orange) for all 7 models **without refitting using these orange data points**. Models are only fit on the 25 training points.



k	Training MSE	Validation MSE
0	60.235744	106.925296
1	30.756678	22.363676
2	29.875269	17.331880
3	29.180868	21.889257
4	28.214850	27.340989
5	25.290990	130.599765
6	23.571025	129.209502

Evaluation: **Validation set MSE** is best for degree = 2!

Plotting Training and Validation MSE



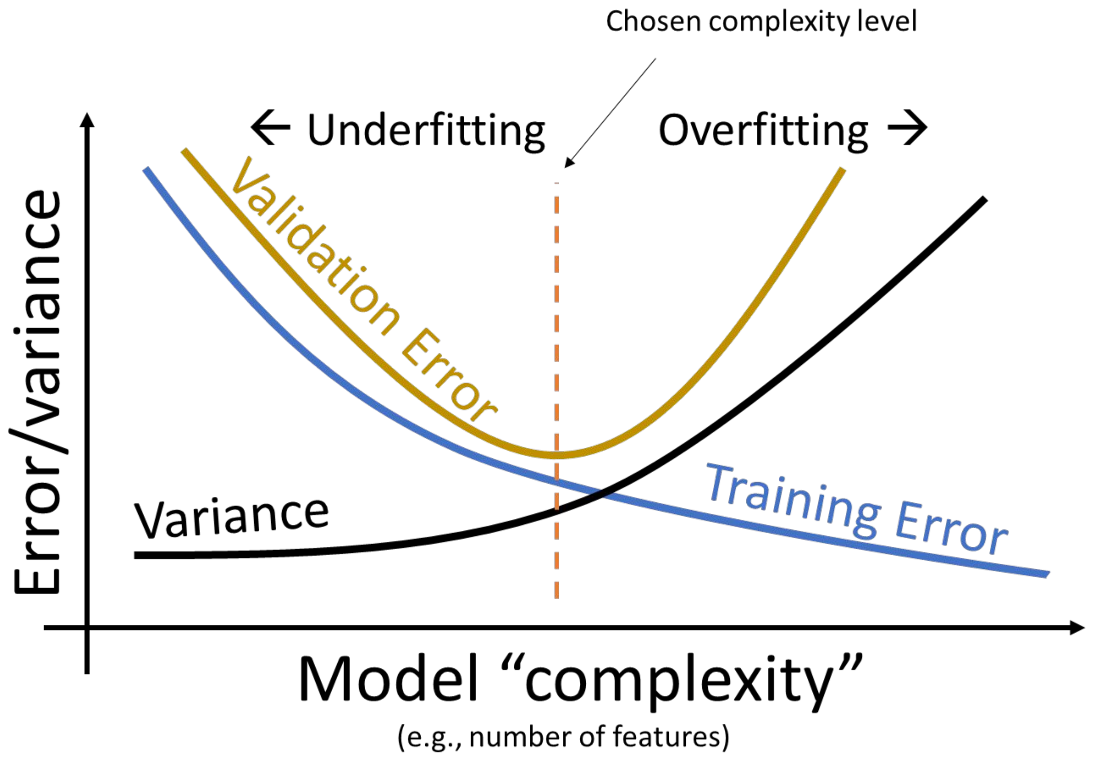
k	Training MSE	Validation MSE
0	60.235744	106.925296
1	30.756678	22.363676
2	29.875269	17.331880
3	29.180868	21.889257
4	28.214850	27.340989
5	25.290990	130.599765
6	23.571025	129.209502

Idealized Picture of Training and Validation Error

3726997

As we increase the complexity of our model:

- **Training error** decreases.
- Variance increases.
- Typically, **error on validation data** decreases, then increases.



We pick the model complexity that minimizes **validation set error**.

Hyperparameter: Terminology

3726997

In machine learning, a **hyperparameter** is a value that controls the learning process itself.

- For our example today, we built seven models, each of which had a hyperparameter called degree or k that controlled the order of our polynomial.

We use:

- The **training set** to **select parameters**.
- The **validation set** (a.k.a. development set) (a.k.a. cross validation set) to **select hyperparameters**, or more generally, between different competing models.

K-Fold Cross Validation

Lecture 15, Data 100 Spring 2023

Cross Validation

- The Holdout Method
- **K-Fold Cross Validation**
- Test Sets

Regularization

- L2 Regularization (Ridge)
- Scaling Data for Regularization
- L1 Regularization (LASSO)

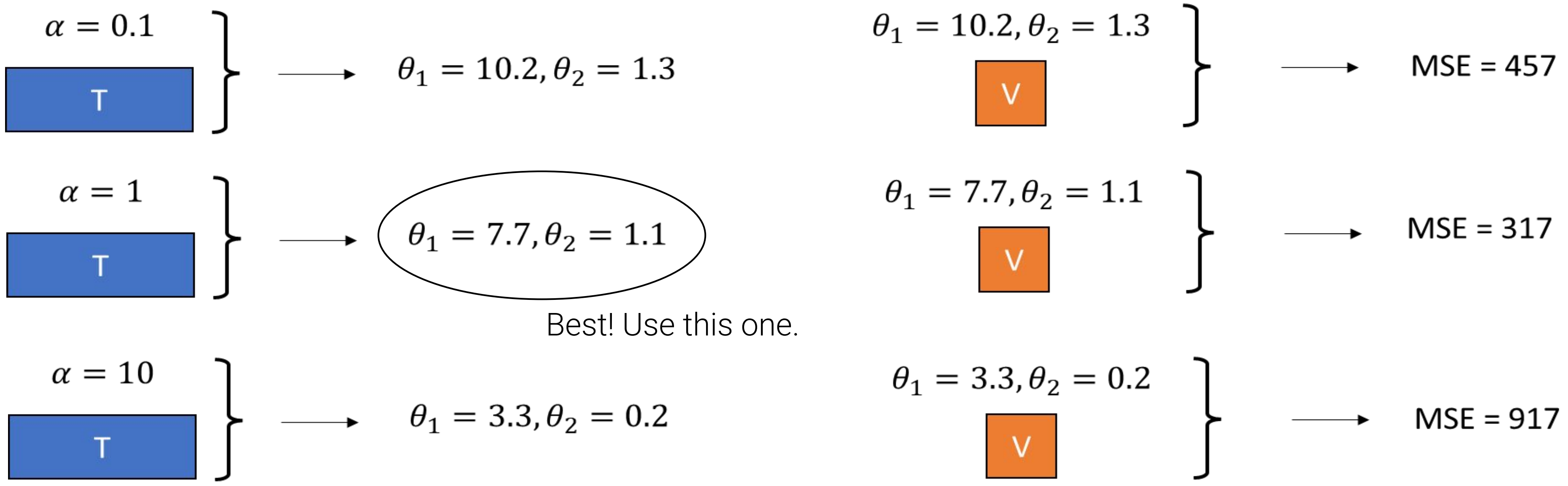
Another View of The Holdout Method

3726997

To determine the quality of a particular hyperparameter:

- **Train model** on **ONLY** the **training set. Quality** is model's error on ONLY the **validation set**.

Example, imagine we are trying to pick between three values of a hyperparameter α.



$\alpha = 0.1$
T
$\theta_1 = 10.2, \theta_2 = 1.3$
$\theta_1 = 10.2, \theta_2 = 1.3$
V
MSE = 457
$\alpha = 1$
T
$\theta_1 = 7.7, \theta_2 = 1.1$
$\theta_1 = 7.7, \theta_2 = 1.1$
V
MSE = 317
Best! Use this one.
$\alpha = 10$
T
$\theta_1 = 3.3, \theta_2 = 0.2$
$\theta_1 = 3.3, \theta_2 = 0.2$
V
MSE = 917

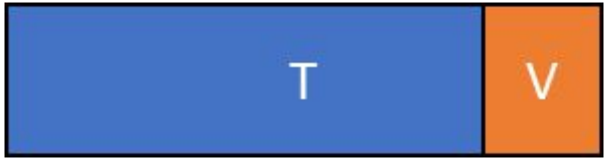
Another View of The Holdout Method

3726997

In the Holdout Method, we set aside the validation set at the beginning, and our choice is fixed.

- **Train model** on **ONLY** the **training set. Quality** is model's error on ONLY the **validation set**.

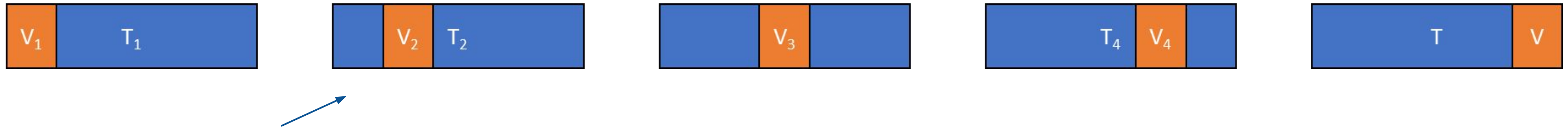
Example below where the last 20% is used as the validation set.



Thought Experiment

3726997

If we decided (arbitrarily) to use non-overlapping contiguous chunks of 20% of the data, there are 5 possible “chunks” of data we could use as our validation set, as shown below.



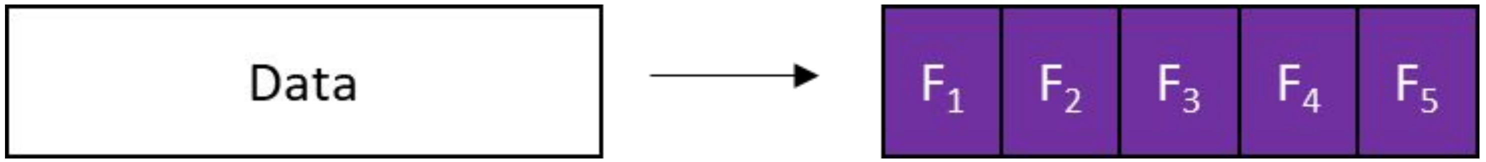
Use first 20% and last 60% to train, remaining 20% as validation set.

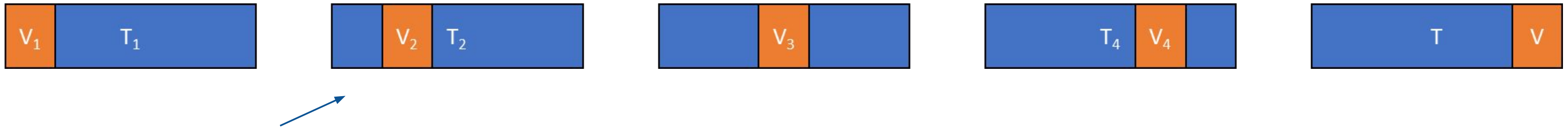
Thought Experiment

3726997

If we decided (arbitrarily) to use non-overlapping contiguous chunks of 20% of the data, there are 5 possible "chunks" of data we could use as our validation set, as shown below.

- The common term for these chunks is a "fold".
  - For example, for chunks of size 20%, we have 5 folds.

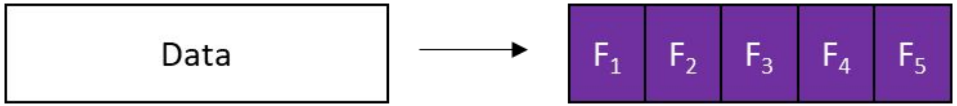




Use folds 1, 3, 4, and 5 to train, and use fold 2 as validation set.

K-Fold Cross Validation

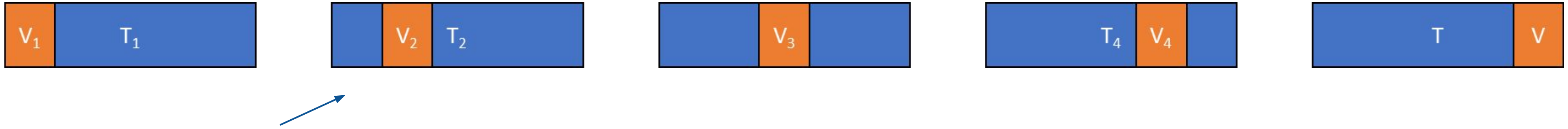
In the k-fold cross-validation approach, we split our data into k equally-sized groups (often called folds).



Given k folds, to determine the quality of a particular hyperparameter:

- Pick a fold, which we'll call the validation fold. Train model on all but this fold. Compute error on the validation fold.
- Repeat the step above for all k possible choices of validation fold.
- Quality is the average of the k validation fold errors.

Example for k = 5:



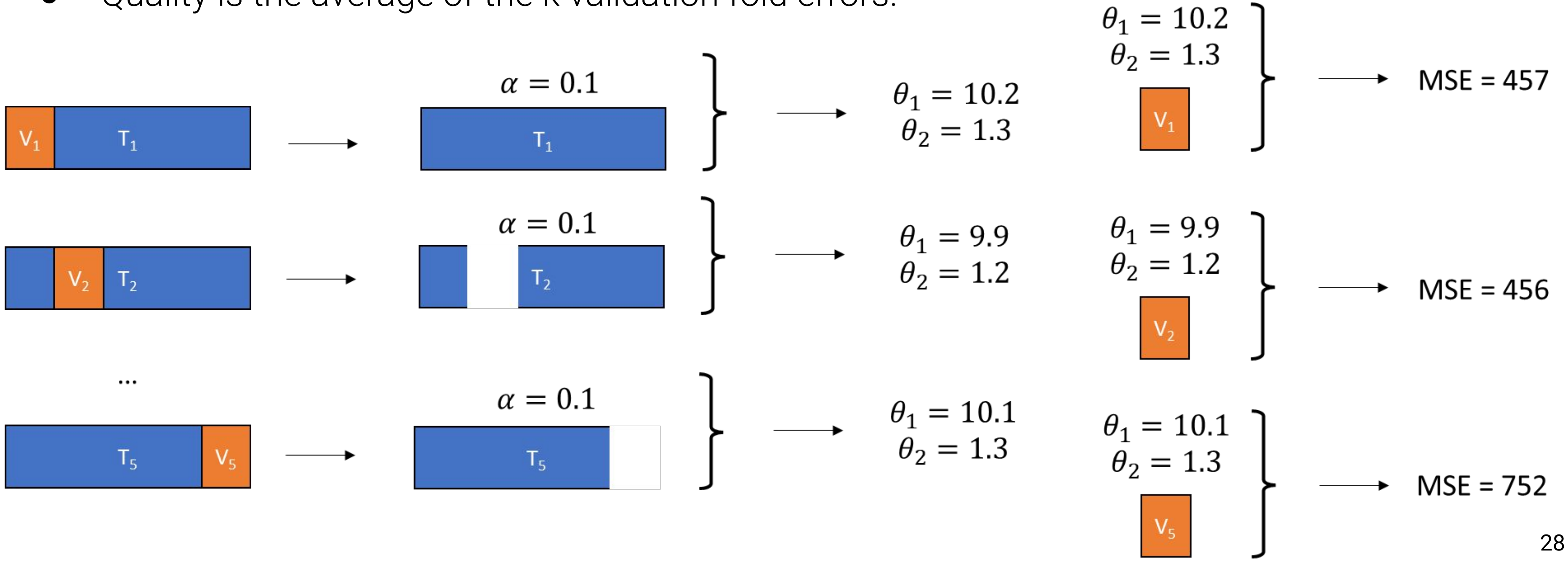
Use folds 1, 3, 4, and 5 to train, and use fold 2 as validation set.

5-Fold Cross Validation Demo

3726997

Given k folds, to determine the quality of a particular hyperparameter, e.g. $\alpha = 0.1$:

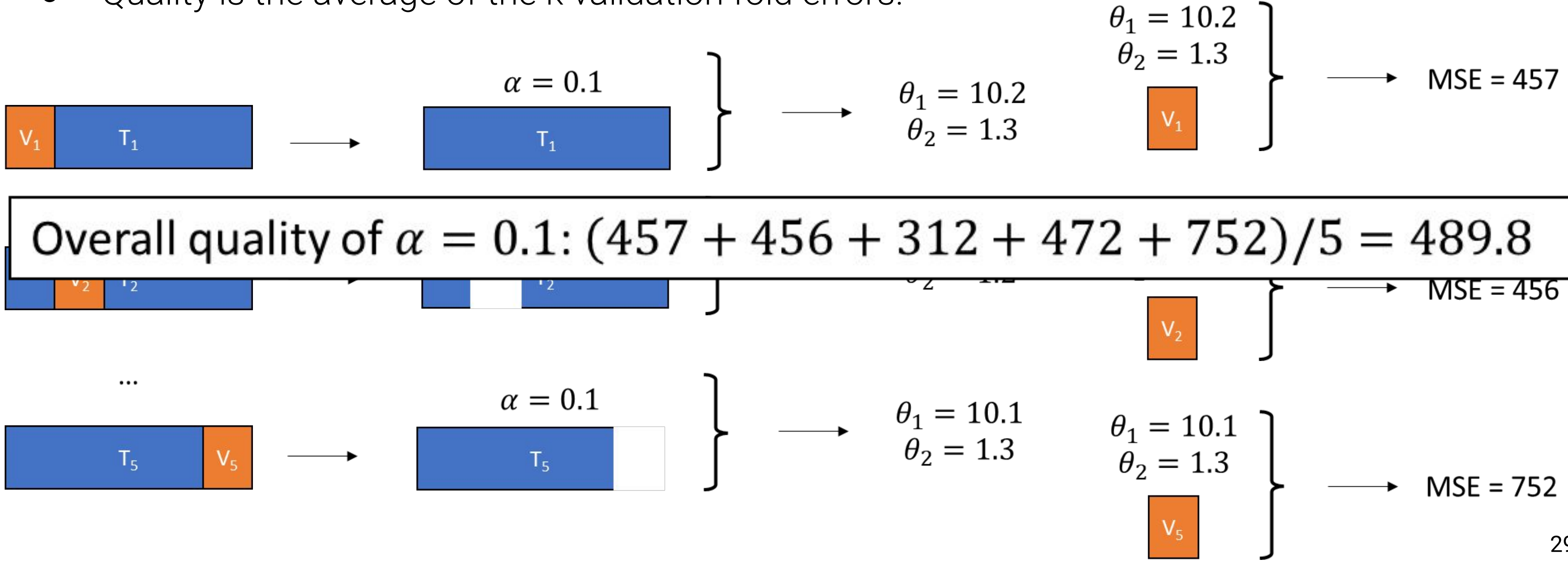
- Pick a fold, which we'll call the validation fold. Train model on all but this fold. Compute error on the validation fold.
- Repeat the step above for all k possible choices of validation fold.
- Quality is the average of the k validation fold errors.



5-Fold Cross Validation Demo

Given k folds, to determine the quality of a particular hyperparameter, e.g. alpha = 0.1:

- Pick a fold, which we'll call the validation fold. Train model on all but this fold. Compute error on the validation fold.
- Repeat the step above for all k possible choices of validation fold.
- Quality is the average of the k validation fold errors.



Overall quality of $\alpha = 0.1$: $(457 + 456 + 312 + 472 + 752)/5 = 489.8$

Test Your Understanding: How Many MSEs?

3726997

Suppose we pick k = 3 and we have 4 possible hyperparameter values α=[0.01, 0.1, 1, 10].

- How many total MSE values will we compute to get the quality of α=10?
- How many total MSE values will we compute to find the best α?

slido

Suppose we pick k = 3 and we have 4 possible hyperparameter values alpha=[0.01, 0.1, 1, 10]. How many total MSE values will we compute to get the quality of alpha=10?

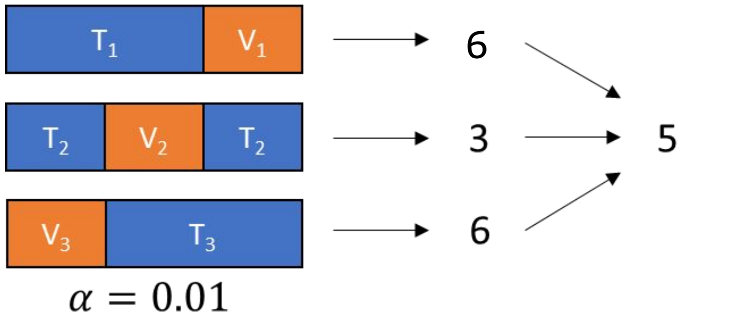
ⓘ Start presenting to display the poll results on this slide.

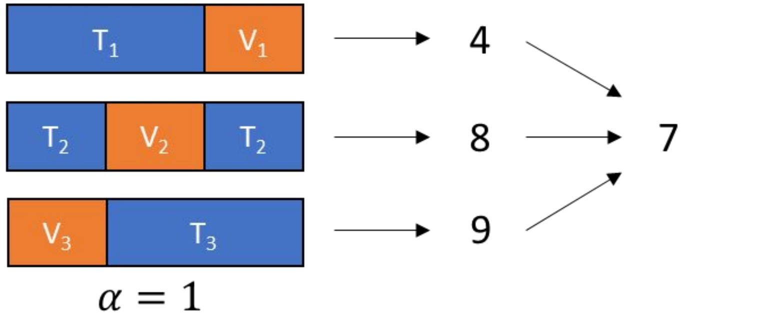
Test Your Understanding: How Many MSEs?

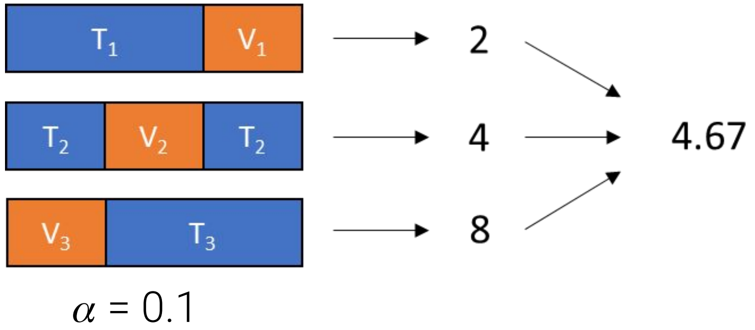
3726997

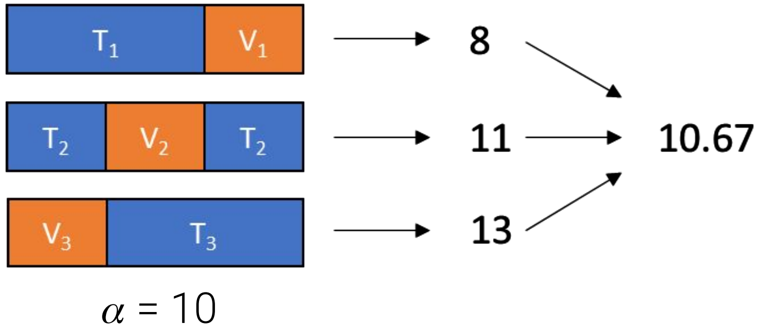
Suppose we pick k = 3 and we have 4 possible hyperparameter values α=[0.01, 0.1, 1, 10].

- How many total MSE values will we compute to get the quality of α=10? 3
- How many total MSE values will we compute to find the best α? 12







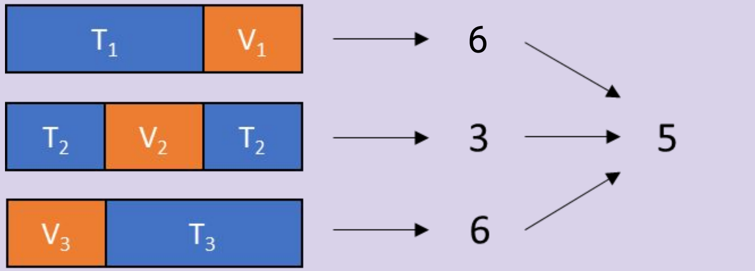


Test Your Understanding: Selecting Alpha

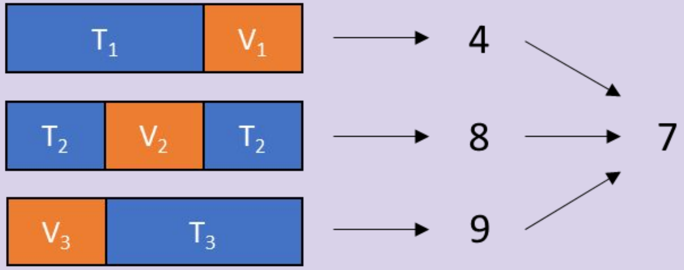
3726997

Which α should we pick?

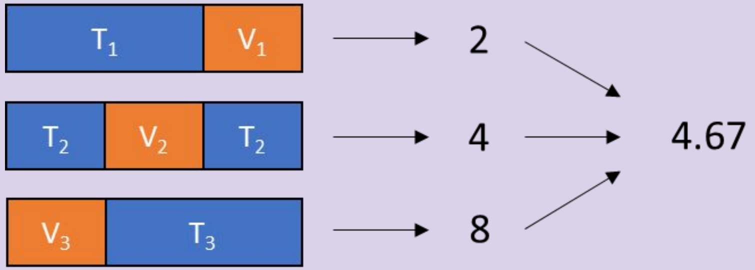
What fold (or folds) should we use as our training set for computing our final model parameters θ?



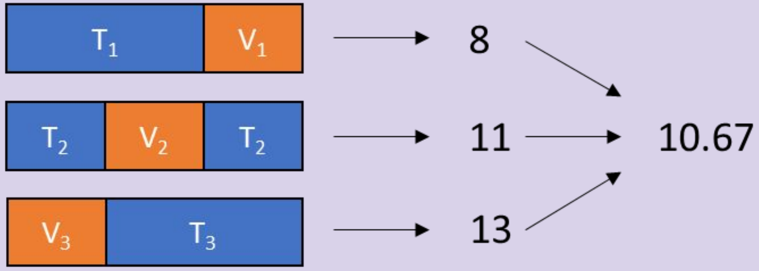
$\alpha = 0.01$



$\alpha = 1$



$\alpha = 0.1$



$\alpha = 10$

slido

Which alpha should we pick?

ⓘ Start presenting to display the poll results on this slide.

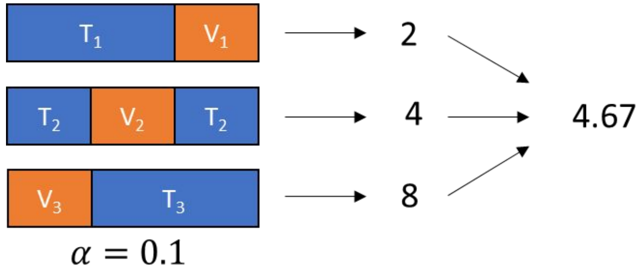
Test Your Understanding: Selecting Alpha

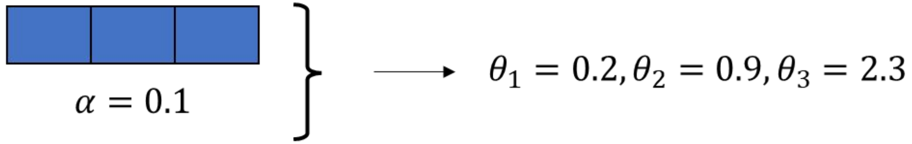
3726997

Which α should we pick? 0.1

What fold (or folds) should we use as our training set for computing our final model parameters θ?

- There's no reason to prefer any fold over any other. In practice, best to train all model on all of the data, i.e. use all 3 folds.





Picking K

3726997

Typical choices of k are 5, 10, and N, where N is the amount of data.

- k=N is also known as “leave one out cross validation”, and will typically give you the best results.
  - In this approach, each validation set is only one point.
  - Every point gets a chance to get used as the validation set.
- k=N is also very expensive, require you to fit a huge number of models.

Ultimately, the tradeoff is between k and computation time.

Cross Validation Summary

When selecting between models, we want to pick the one that we believe would generalize best on unseen data. Generalization is estimated with a "**cross validation score**"*.

- When selecting between models, keep the model with the best **score**.

Two techniques to compute a "**cross validation score**":

- The Holdout Method: Break data into a separate **training set** and **validation set**.
  - Use **training set** to **fit** parameters (thetas) for the model.
  - Use **validation set** to **score** the model.
  - Also called "Simple Cross Validation" in some sources.
- k-Fold Cross Validation: Break data into k contiguous non-overlapping "folds".
  - Perform k rounds of Simple Cross Validation, except:
    - Each fold gets to be the **validation set** exactly once.
    - The final **score** of a model is the **average validation score** across the k trials.

*Equivalently, I could have said "**cross validation loss**" instead of "**cross validation score**".

Test Sets

Lecture 15, Data 100 Spring 2023

3726997

Providing an Estimate of Model Generalization to the World

3726997

Suppose we're researchers building a state-of-the-art regression model.

- After months of work and after comparing billions of candidate models, we find the model with the best **validation set loss**.

Now we want to report this model out to the world so it can be compared to other models.

- Our **validation set loss** is not an unbiased estimator of its performance!
- Instead, we'll run our model just one more time on a special **test set**, that we've never seen or used for any purpose whatsoever.

Why Are Validation Set Errors Biased?

3726997

Analogy:

- Imagine we have a golf ball hitting competition. Whoever can hit the ball the farthest wins.
- Suppose we have the best 10000000 golfers in the world **play a tournament**. There are probably many roughly equal players near the top.
- When we're done, we want to provide an unbiased estimate of our best golfer's distance in yards.
- Using the **tournament results** may be biased, as the the winner maybe got just a bit lucky (maybe they had favorable wind during their rounds).
- Better unbiased estimate: Have the winner **play one more trial and report their score**.

Test Sets

Test sets can be something that we generate ourselves. Or they can be a common dataset whose solution is unknown.

In real world machine learning competitions, competing teams share a **common test set**.

- To avoid accidental or intentional overfitting, the **correct predictions for the test set are never seen by the competitors**.

Creating a Test Set Ourselves

3726997

We can do this easily in code. As before, we shuffle using scikit-learn then split using numpy.

```
# Splitting the data into training,  validation, and test set
train_set, val_set, test_set = np.split(shuffle(vehicle_data_sample_35), [25, 30])
```

Then we use `np.split`, now providing two numbers instead of one. For example, the code above splits the data into a **Training**, **Validation**, and **Test** set.

- Recall that a **validation set** is just another name for a **development set**.
- **Training set** used to pick parameters.
- **Validation set** used to pick hyperparameters (or **pick between different models**).
- **Test set** used to provide an **unbiased MSE at the end**.

Test Set Terminology in Real World Practice

Warning: The terms “**test set**” and “**validation set**” are sometimes used interchangeably.

- You’ll see authors saying things like “then we used a **test set** to select hyperparameters”.
    - While this violates my personal definition of **test set**, it’s clear to me what they meant, namely: “we used a **holdout set** to select hyperparameters”.
    - This is a terminological confusion, not a procedural error! They didn’t do anything wrong.
        - Imagine they said “We used a **bloop blop set**”. Same thing , just weird name.
    - The error would be if they claimed later that the loss on their **test set** was unbiased. Since “**validation set**” error and actually completely unseen “**test set**” errors are typically very close, this terminology error is very minor.

In practice, you may not need a **test set** at all!

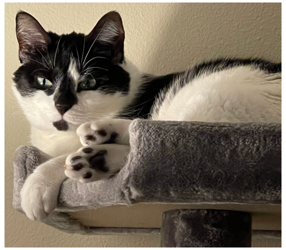
- If all you need to do is pick the best model, and you don’t care about providing a numerical measure of model quality, you don’t need a **test set**.

Validation Sets and Test Sets in Real World Practice

3726997

Standard **validation sets** and **test sets** are used as standard benchmarks to compare ML algorithms.

- Example: ImageNet is a dataset / competition used to compare to different image classification and localization algorithms on 1000 object classes (a “cat”).
  - 1,281,167 **training images**. Images and correct label provided.
  - 50,000 **validation images**. Images and correct label provided.
  - 100,000 **test images**. Images provided, but no correct label.



- When writing papers, researchers report their performance on the **validation images**.
  - This set is a “**validation set**” with respect to the entire global research community.
  - Research groups cannot report the **test error** because they cannot compute it!
- When ImageNet was a competition, the **test set** was used to rank different algorithms.
  - Researchers provide their predictions for the **test set** to a central server.
  - Server (which knows the labels) reports back a **test set score**.
  - Best **test set score** wins.

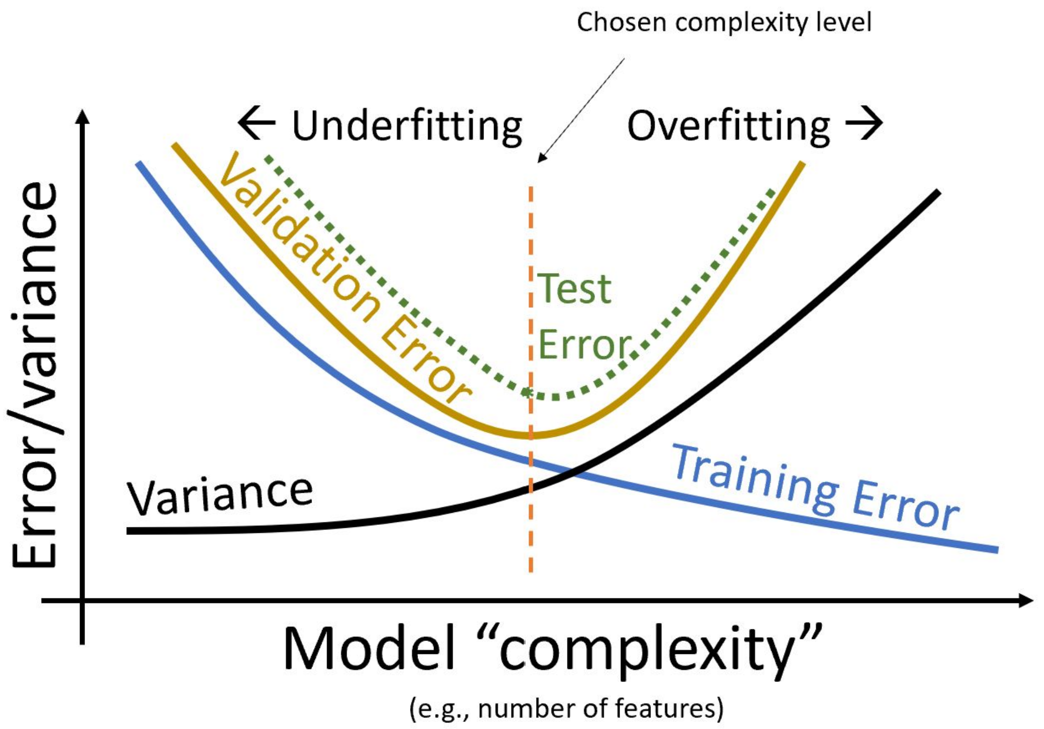
Note: Since the competition uses the **test set scores** compare image classification algorithms, the best **test set score** is no longer an unbiased estimate of the best algorithm’s performance.

Idealized Training, Validation, and Test Error

3726997

As we increase the complexity of our model:

- **Training error** decreases.
- Variance increases.
- Typically, **validation error** decreases, then increases.
- The **test error** is the essentially the same thing as the **validation error**! Only difference is that we are much restrictive about computing the **test error**
  - Don't get to see the whole curve!



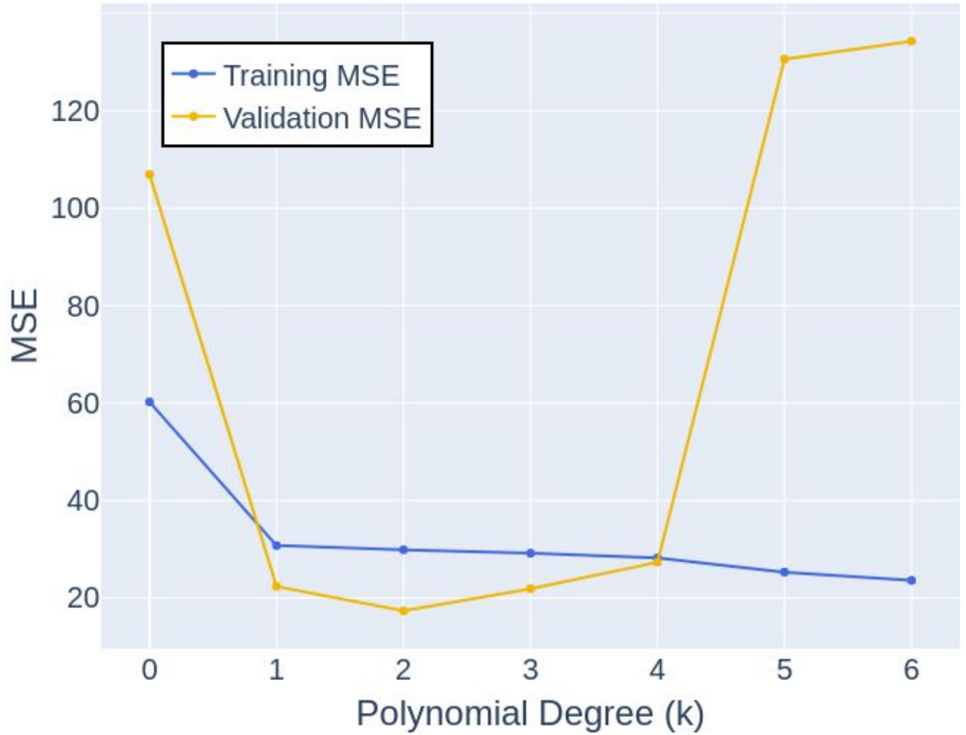
L2 Regularization (Ridge)

Lecture 15, Data 100 Spring 2023

Earlier

3726997

We saw how we can select model complexity by choosing the hyperparameter that minimizes **validation error**. This **validation error** can be computed using the **Holdout Method** or **K-Fold Cross Validation**.

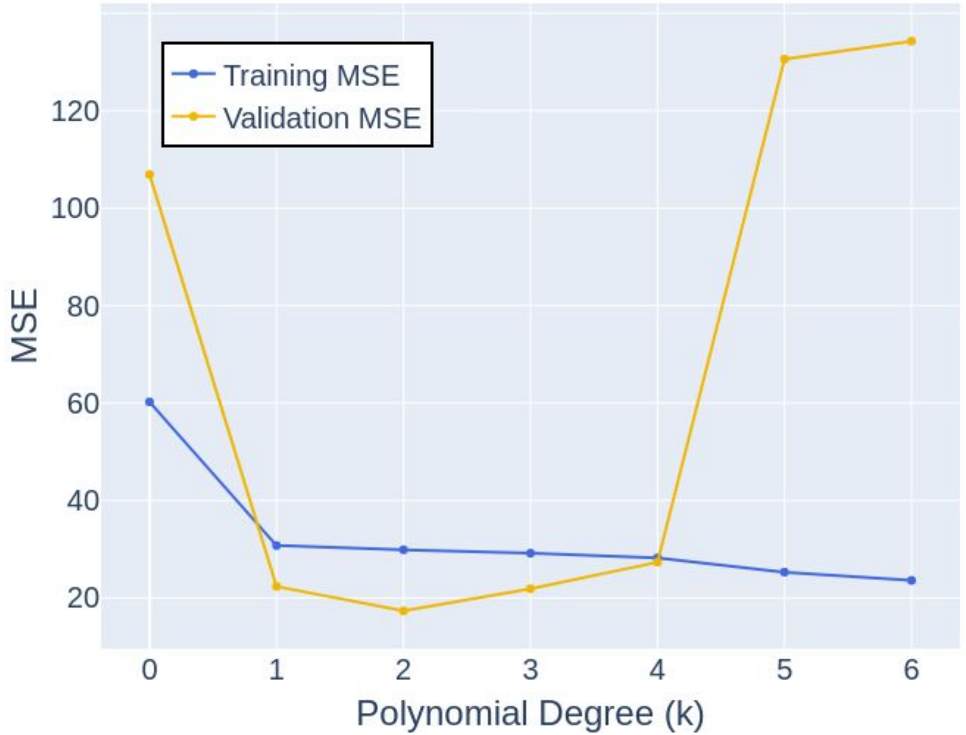


k	Training MSE	Validation MSE
0	60.235744	106.925296
1	30.756678	22.363676
2	29.875269	17.331880
3	29.180868	21.889257
4	28.214850	27.340989
5	25.290990	130.599765
6	23.571025	129.209502

Earlier

For the example below, our hyperparameter was the polynomial degree.

- Tweaking the “complexity” is simple, just increase or decrease the degree.



k	Training MSE	Validation MSE
0	60.235744	106.925296
1	30.756678	22.363676
2	29.875269	17.331880
3	29.180868	21.889257
4	28.214850	27.340989
5	25.290990	130.599765
6	23.571025	129.209502

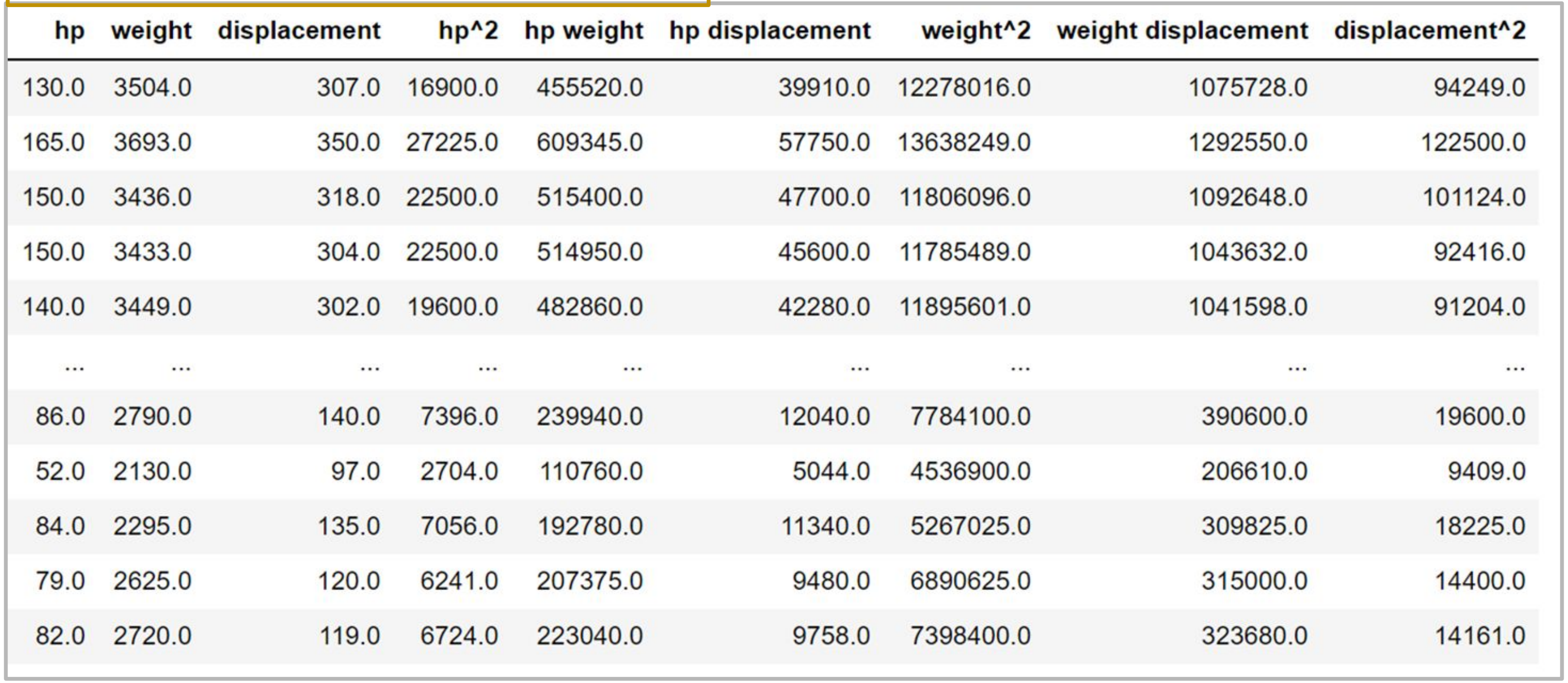
A More Complex Example

3726997

Suppose we have a dataset with 9 features.

- We want to decide which of the 9 features to include in our linear regression.

```
vehicle_data_with_squared_features
```



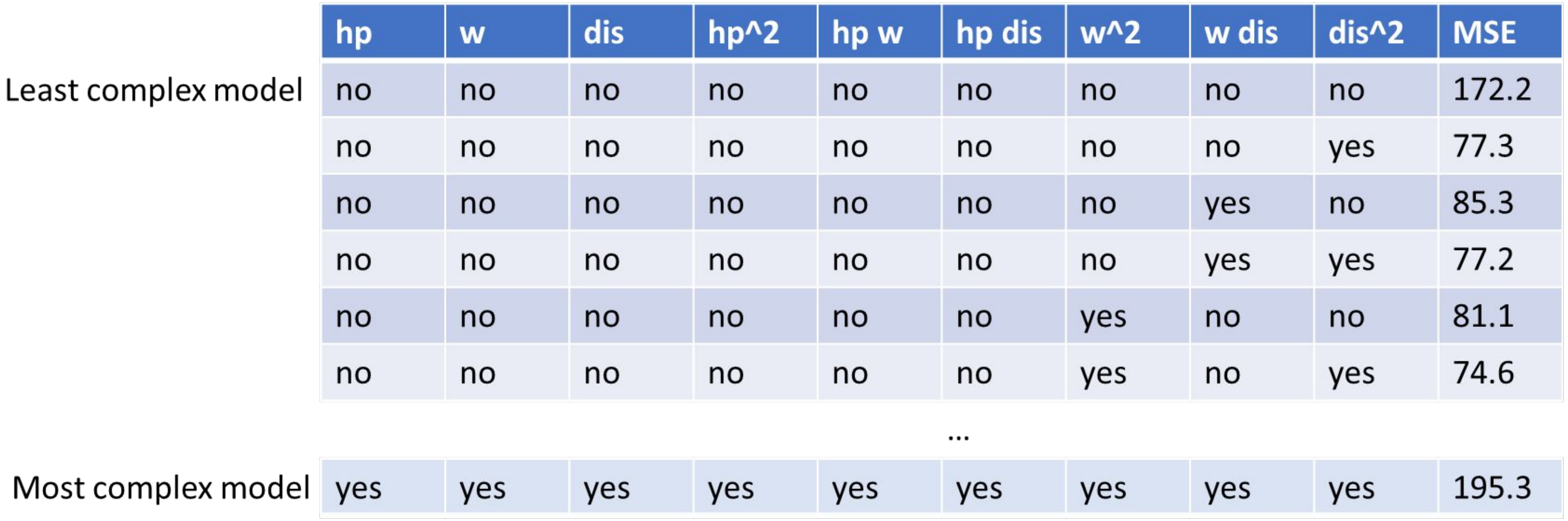
hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0

Tweaking Complexity via Feature Selection

With 9 features, there are 2^9 different models.One approach:

- For each of the 2^9 linear regression models, compute the **validation MSE**.
- Pick the model that has the lowest **validation MSE**.

Runtime is exponential in the number of parameters!



	hp	w	dis	hp^2	hp w	hp dis	w^2	w dis	dis^2	MSE
Least complex model	no	no	no	no	no	no	no	no	no	172.2
	no	no	no	no	no	no	no	no	yes	77.3
	no	no	no	no	no	no	no	yes	no	85.3
	no	no	no	no	no	no	no	yes	yes	77.2
	no	no	no	no	no	no	yes	no	no	81.1
	no	no	no	no	no	no	yes	no	yes	74.6
						...				
Most complex model	yes	yes	yes	yes	yes	yes	yes	yes	yes	195.3

Tweaking Complexity via Feature Selection

3726997

Alternate Idea: What if we use all of the features, but only a little bit?

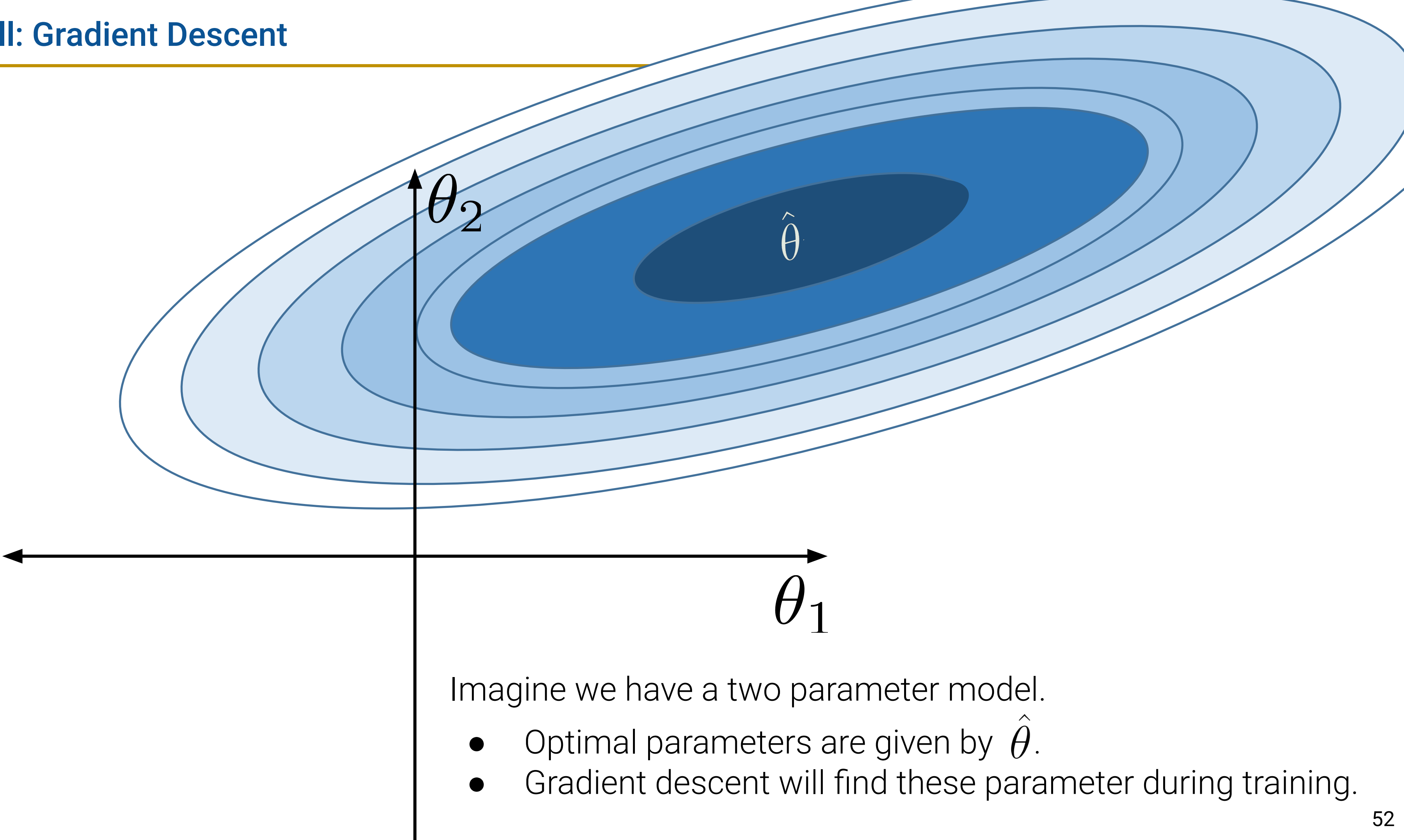
- Let's see a simple example for a 2 feature model.
- Will return to this 9 feature model later.

	hp	w	dis	hp^2	hp w	hp dis	w^2	w dis	dis^2	MSE
Least complex model	no	no	no	no	no	no	no	no	no	172.2
	no	no	no	no	no	no	no	no	yes	77.3
	no	no	no	no	no	no	no	yes	no	85.3
	no	no	no	no	no	no	no	yes	yes	77.2
	no	no	no	no	no	no	yes	no	no	81.1
	no	no	no	no	no	no	yes	no	yes	74.6

...

Most complex model	yes	yes	yes	yes	yes	yes	yes	yes	yes	195.3

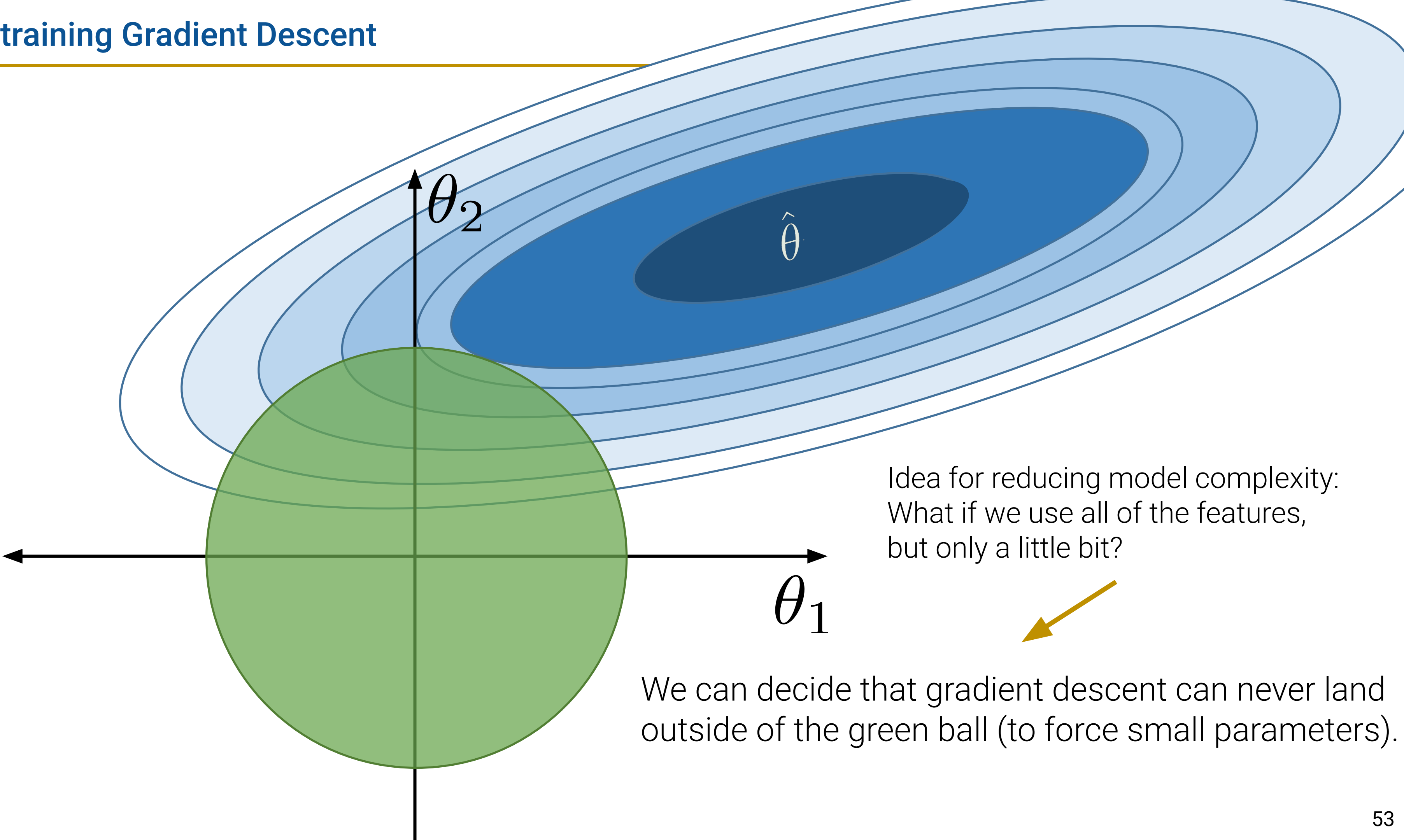
Recall: Gradient Descent



Imagine we have a two parameter model.

- Optimal parameters are given by $\hat{\theta}$.
- Gradient descent will find these parameter during training.

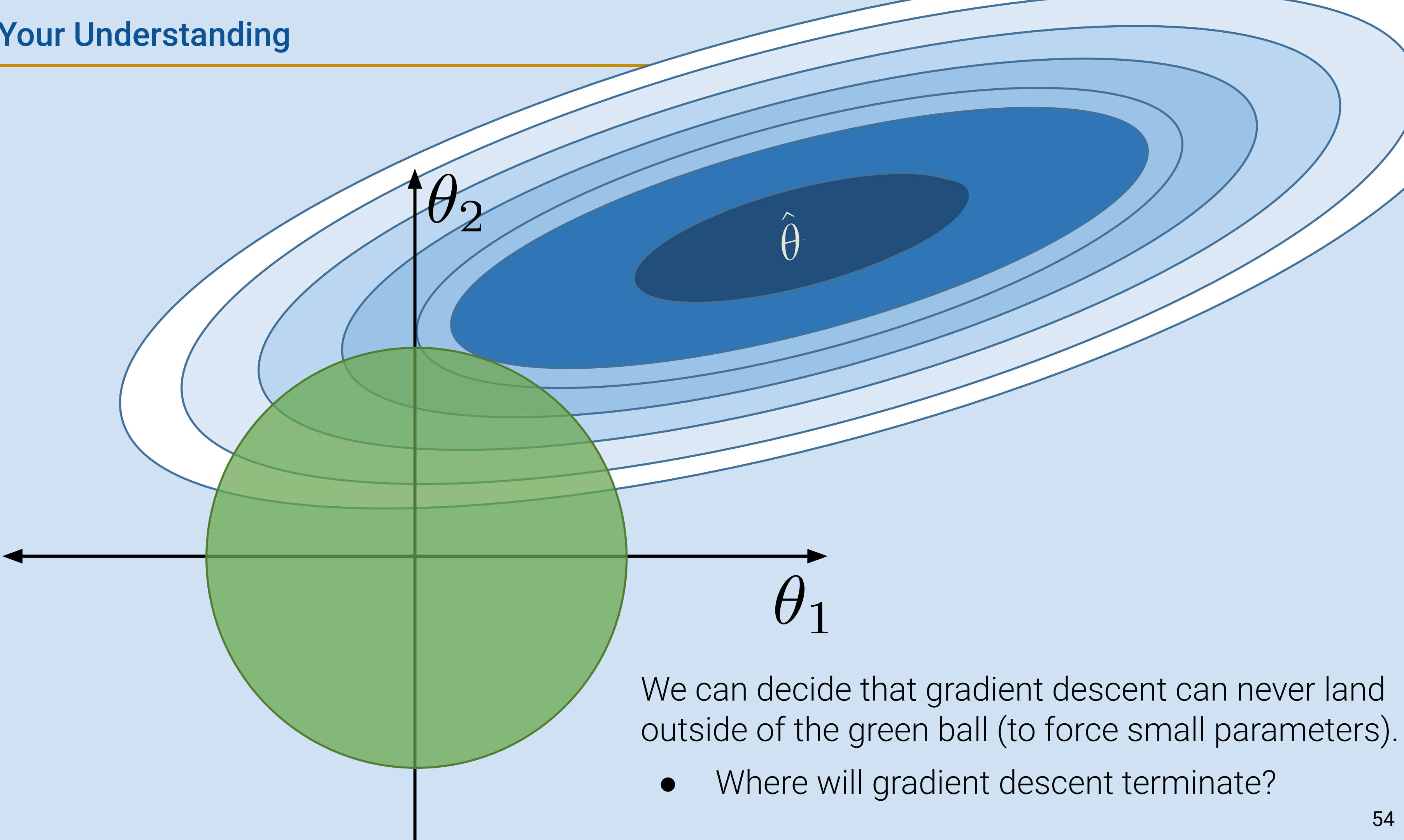
Constraining Gradient Descent



Idea for reducing model complexity: What if we use all of the features, but only a little bit?

We can decide that gradient descent can never land outside of the green ball (to force small parameters).

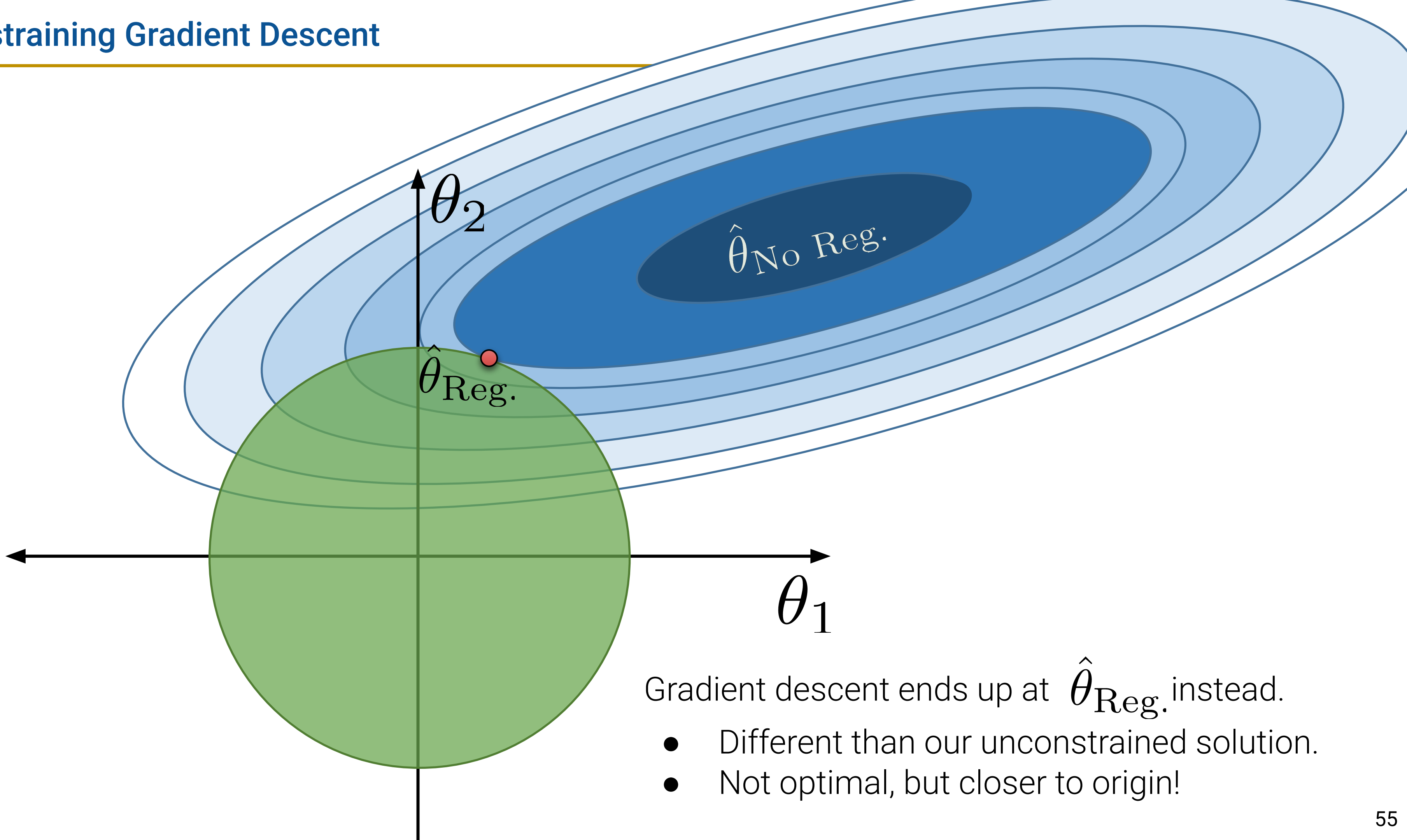
Test Your Understanding



We can decide that gradient descent can never land outside of the green ball (to force small parameters).

- Where will gradient descent terminate?

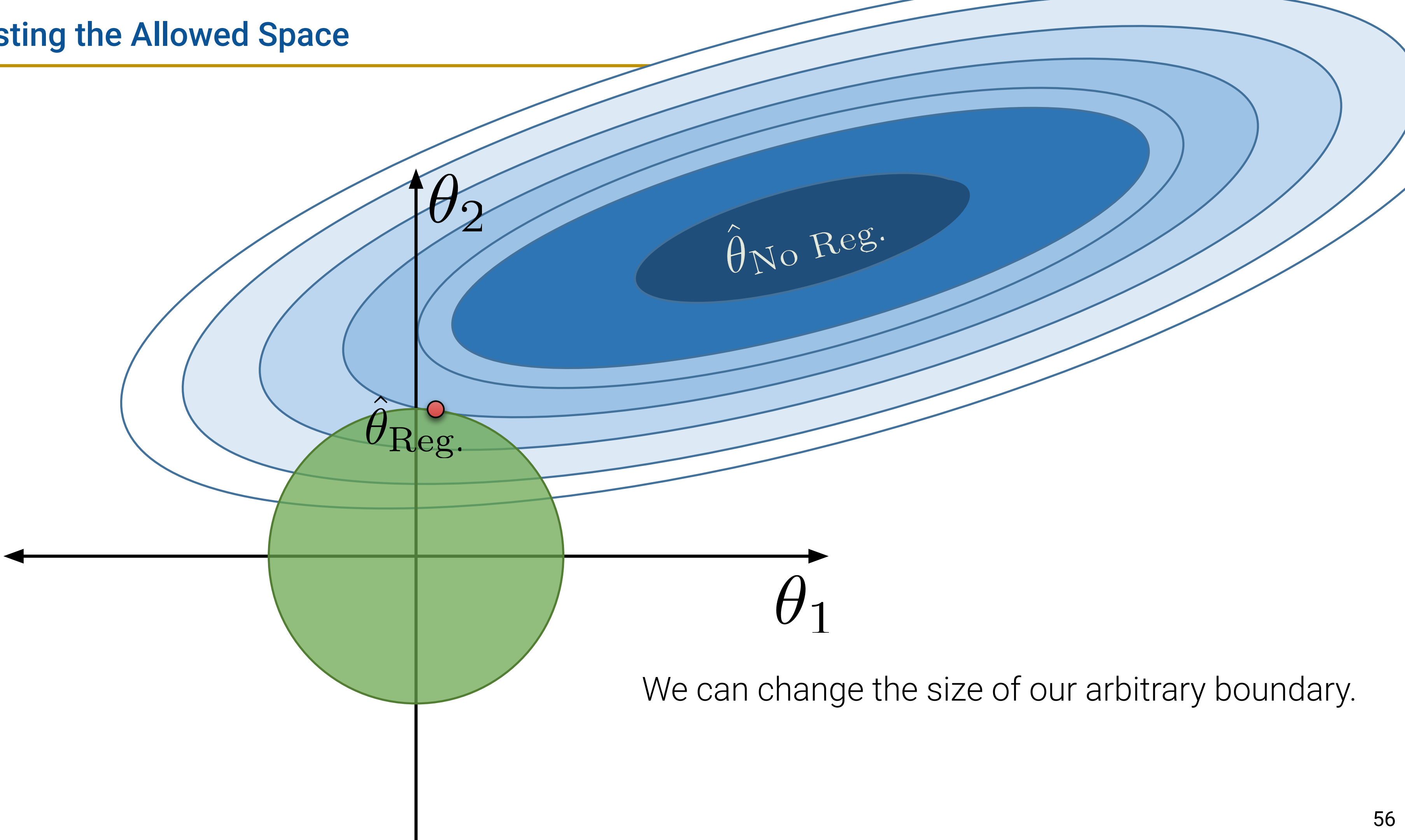
Constraining Gradient Descent



Gradient descent ends up at $\hat{\theta}_{\text{Reg.}}$ instead.

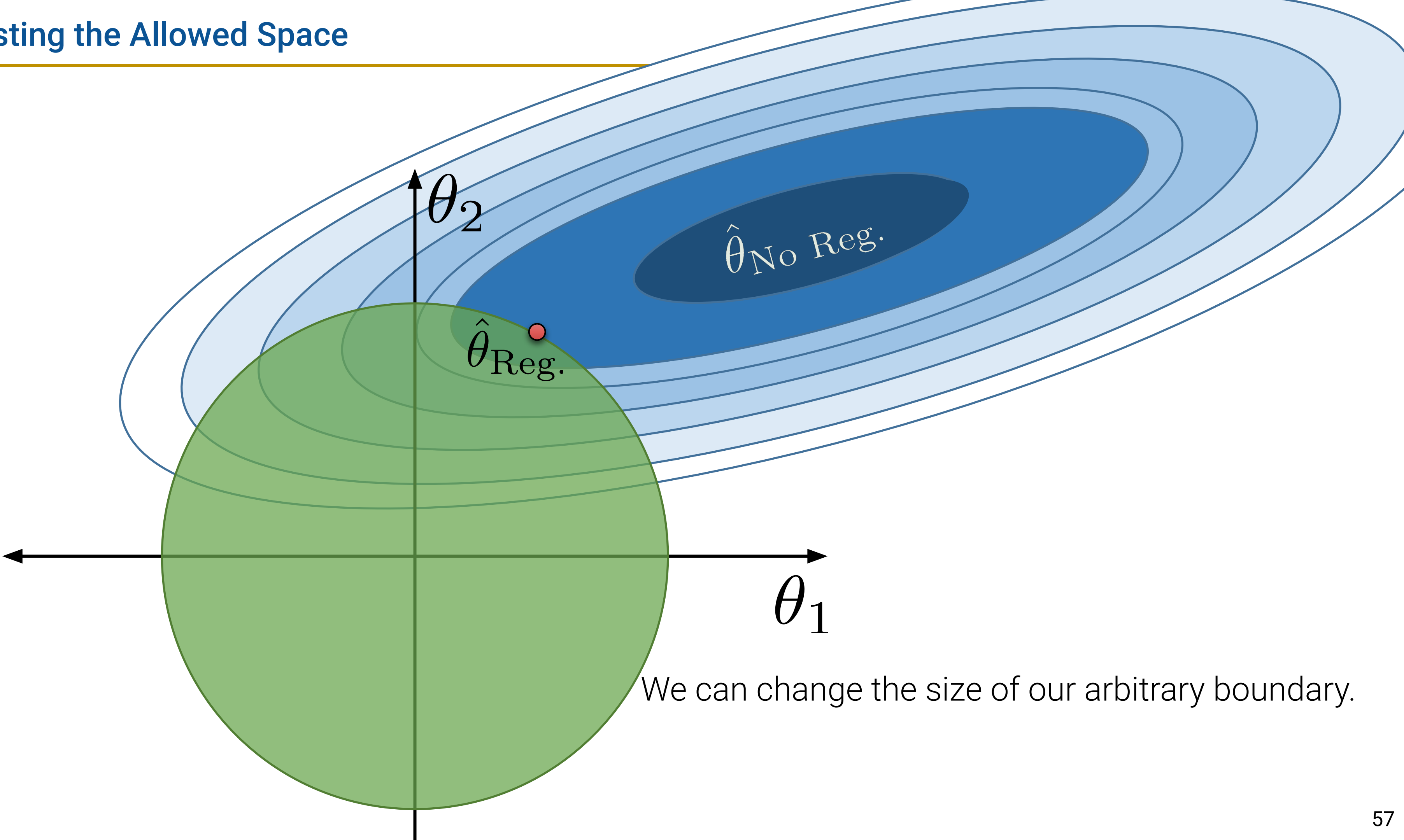
- Different than our unconstrained solution.
- Not optimal, but closer to origin!

Adjusting the Allowed Space



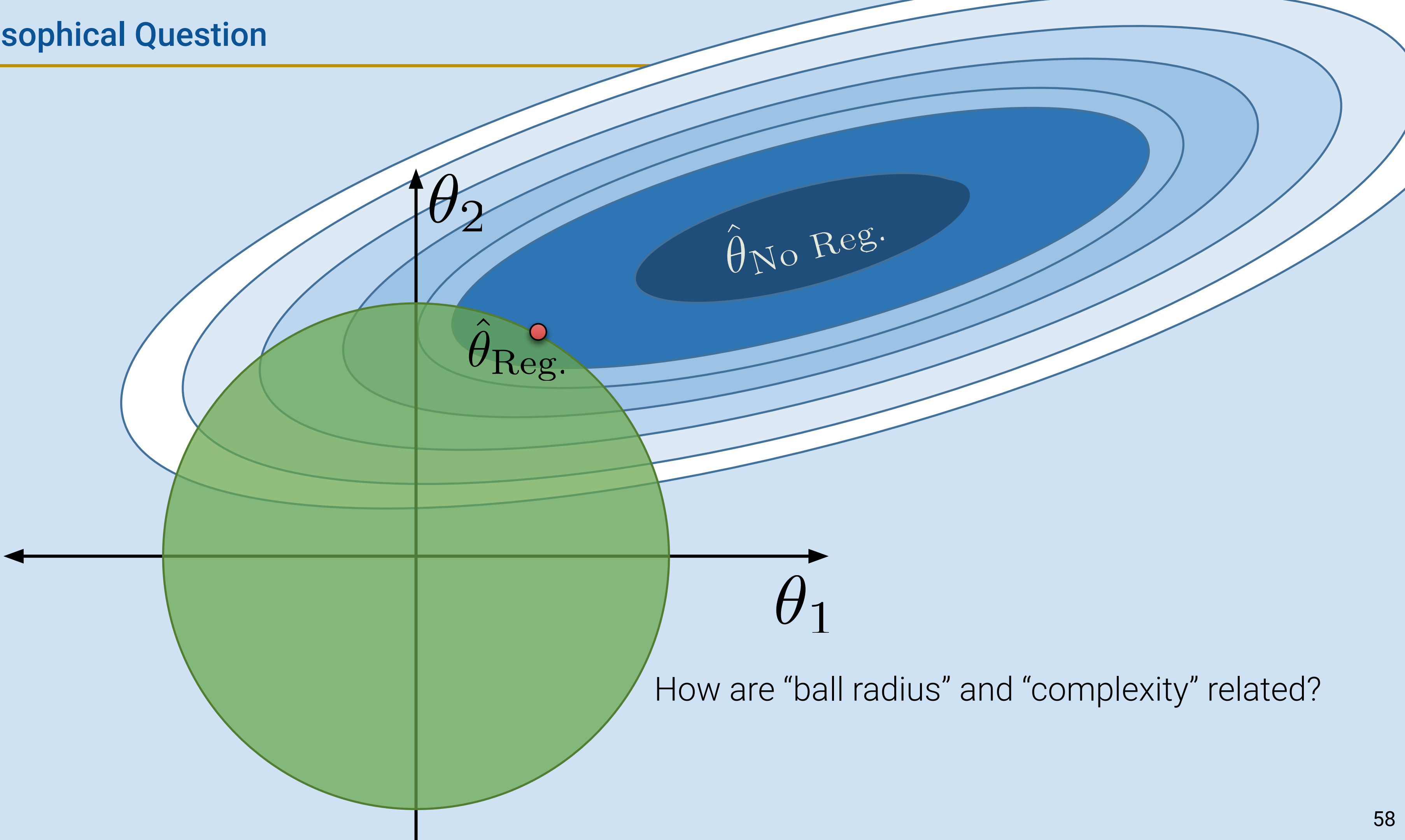
We can change the size of our arbitrary boundary.

Adjusting the Allowed Space



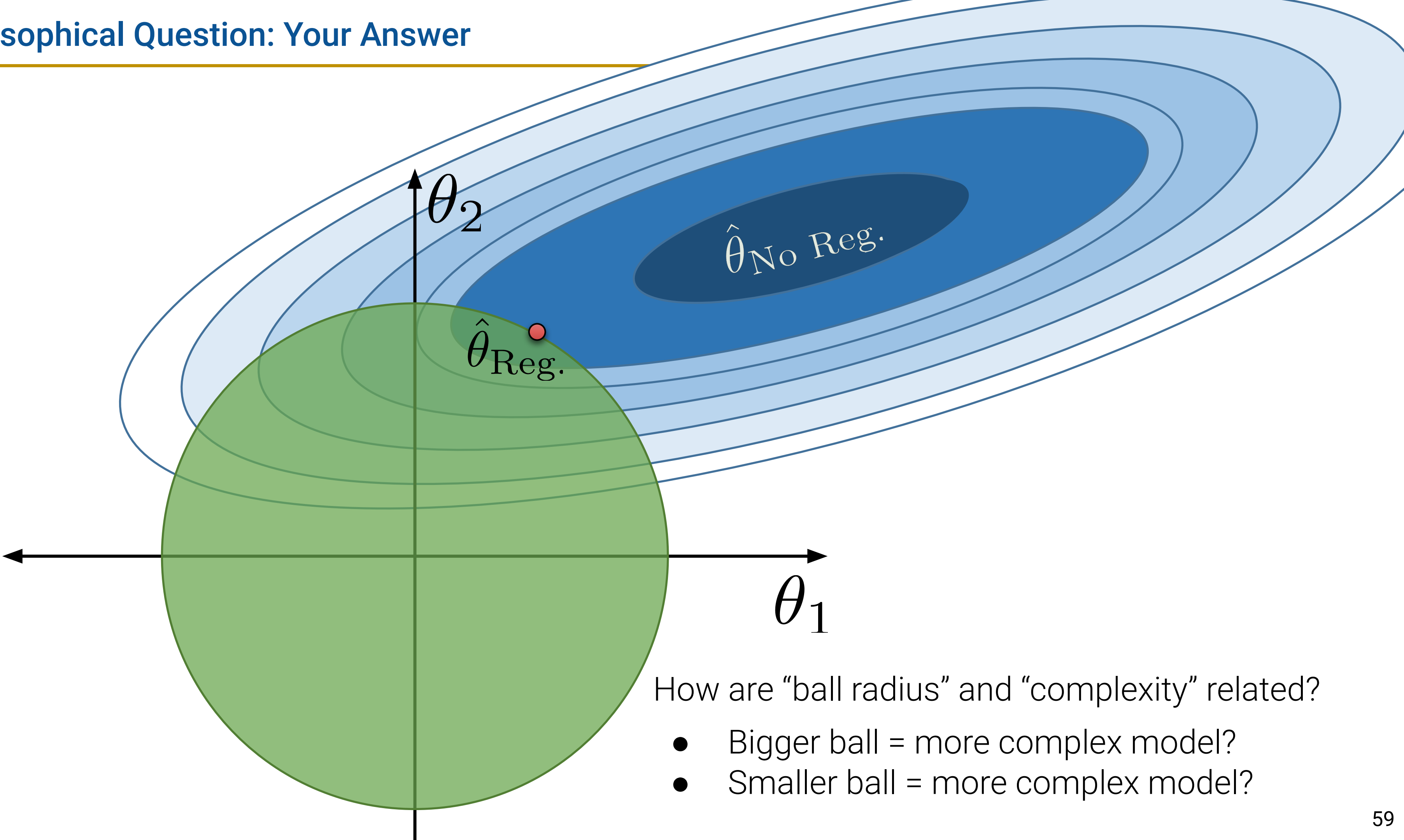
We can change the size of our arbitrary boundary.

Philosophical Question



How are “ball radius” and “complexity” related?

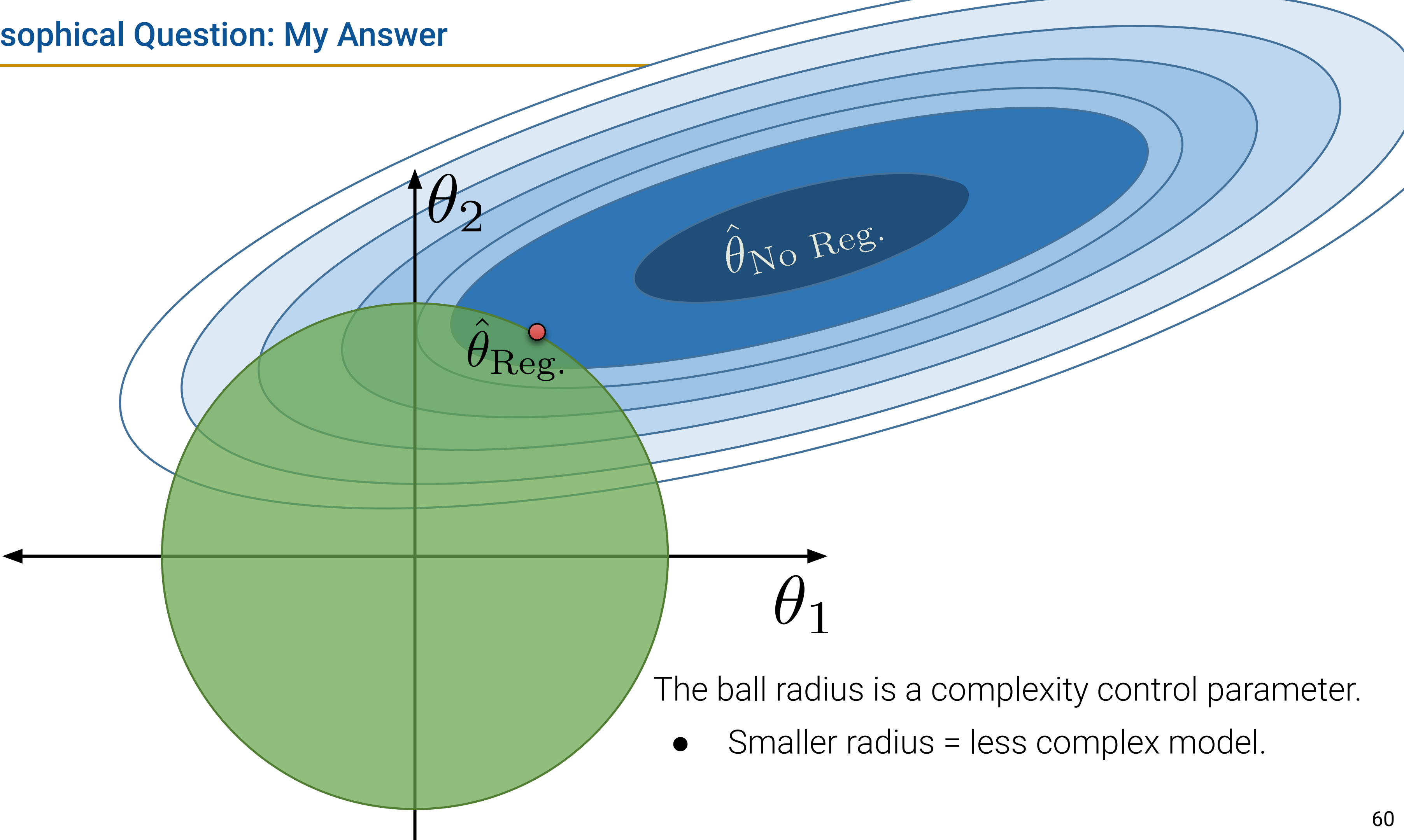
Philosophical Question: Your Answer



How are “ball radius” and “complexity” related?

- Bigger ball = more complex model?
- Smaller ball = more complex model?

Philosophical Question: My Answer



The ball radius is a complexity control parameter.

- Smaller radius = less complex model.

Test Your Understanding

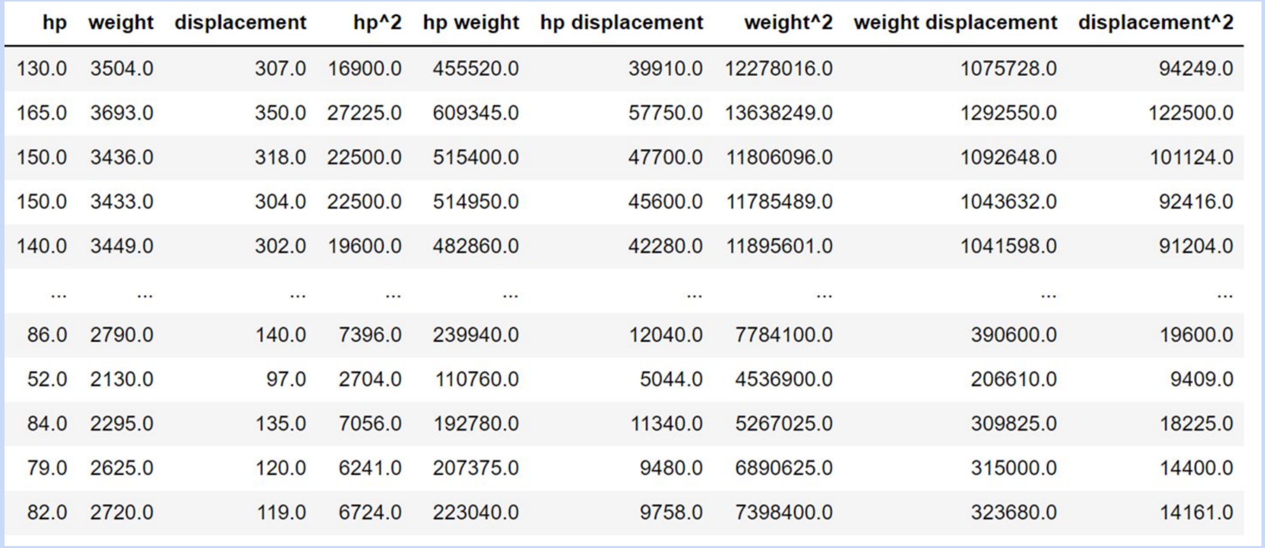
3726997

Let's return to our 9 feature model from before (d = 9).

- If we pick a very small ball radius, what kind of model will we have?

a. A model that only returns zero.
b. A constant model.
c. Ordinary least squares.
d. Something else.

$$\hat{y} = \theta_0 + \theta_1 \phi_1 + \ldots + \theta_d \phi_d$$



hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0

slido

Based on the concept of reducing the complexity of the model by constraining the parameters to be within a ball centered at the origin, what kind of model will we have if we pick a very small radius?

ⓘ Start presenting to display the poll results on this slide.

Test Your Understanding Answer

3726997

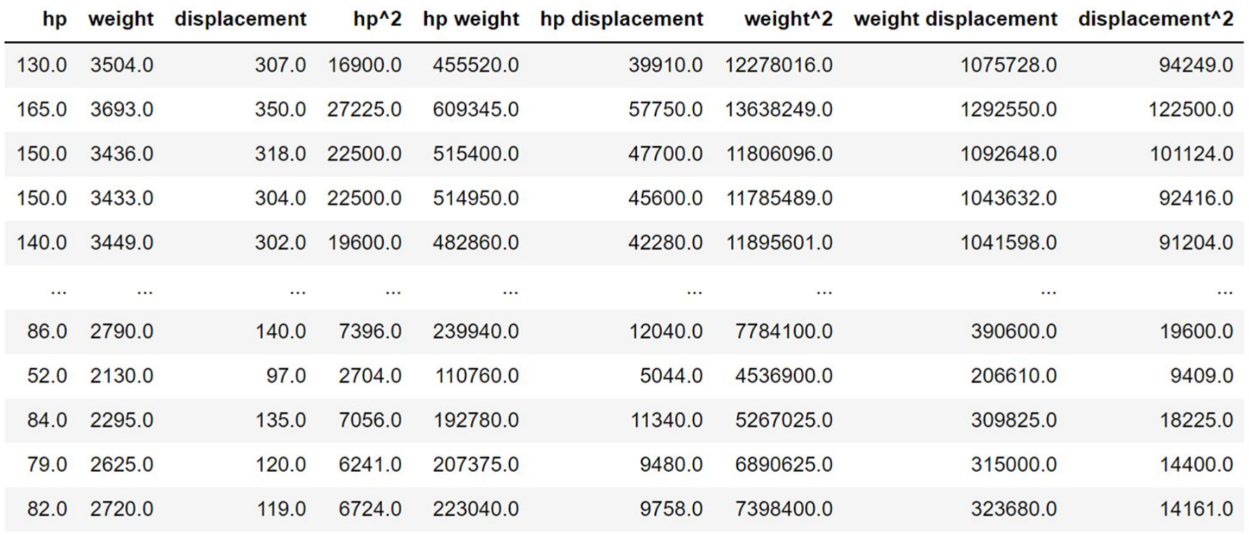
Let’s return to our 9 feature model from before (d = 9).

- If we pick a very small ball radius, what kind of model will we have?

a. A model that only returns zero.
b. A constant model.
c. Ordinary least squares.
d. Something else.

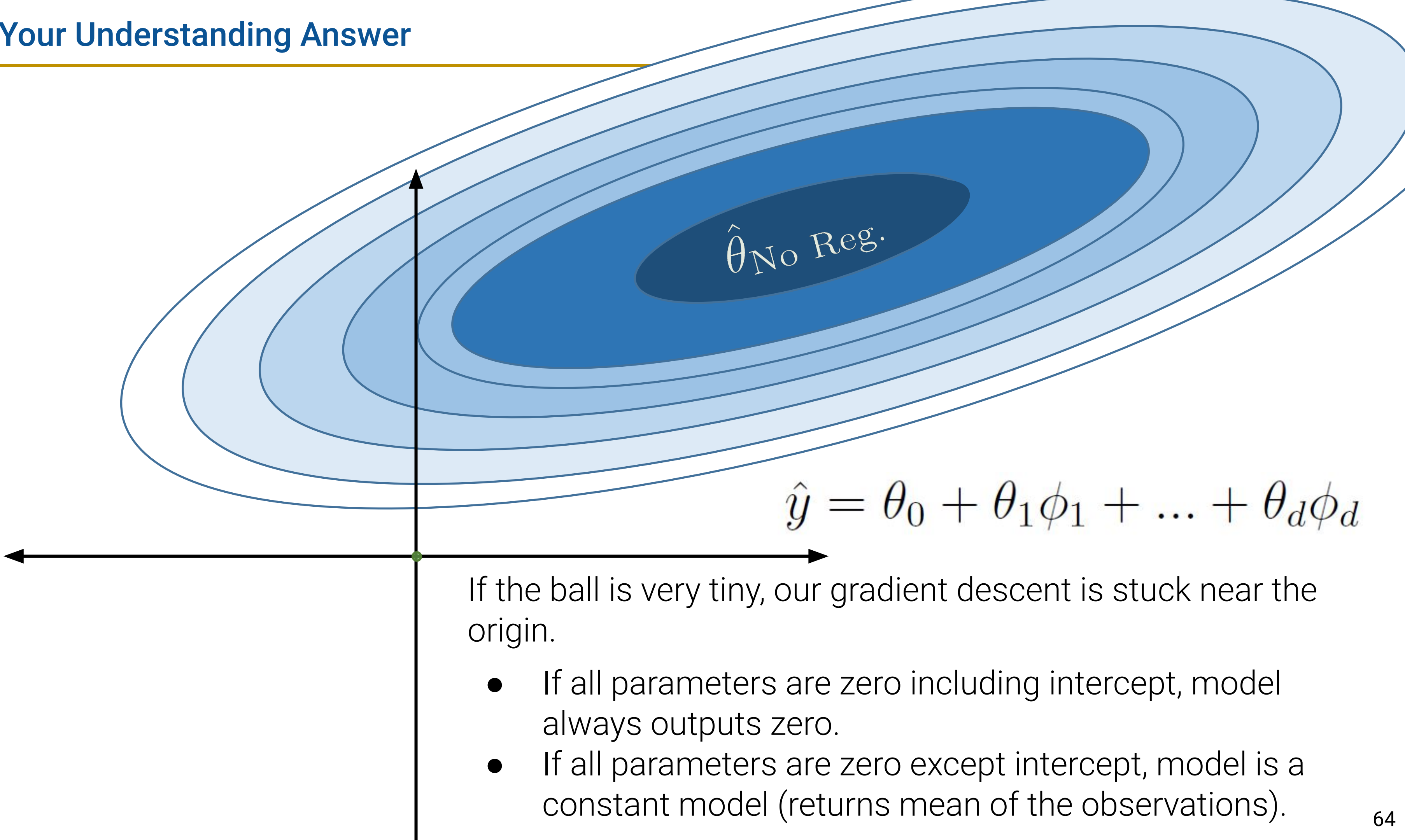
$$\hat{y} = \theta_0 + \theta_1\phi_1 + \ldots + \theta_d\phi_d$$

Answer: It depends!



hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0

Test Your Understanding Answer

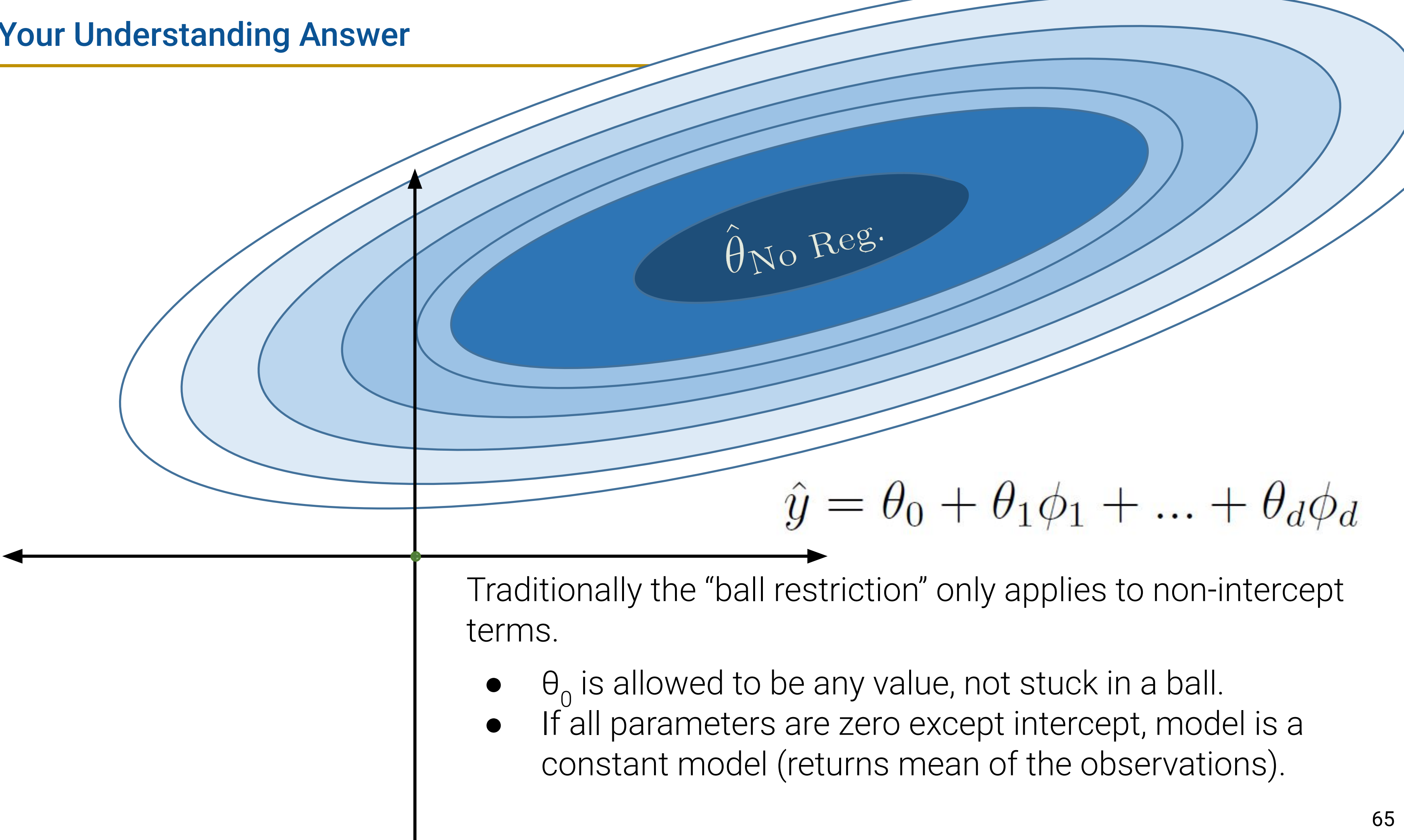


$$\hat{y} = \theta_0 + \theta_1\phi_1 + ... + \theta_d\phi_d$$

If the ball is very tiny, our gradient descent is stuck near the origin.

- If all parameters are zero including intercept, model always outputs zero.
- If all parameters are zero except intercept, model is a constant model (returns mean of the observations).

Test Your Understanding Answer



$$\hat{y} = \theta_0 + \theta_1\phi_1 + \ldots + \theta_d\phi_d$$

Traditionally the “ball restriction” only applies to non-intercept terms.

- θ_0 is allowed to be any value, not stuck in a ball.
- If all parameters are zero except intercept, model is a constant model (returns mean of the observations).

Test Your Understanding

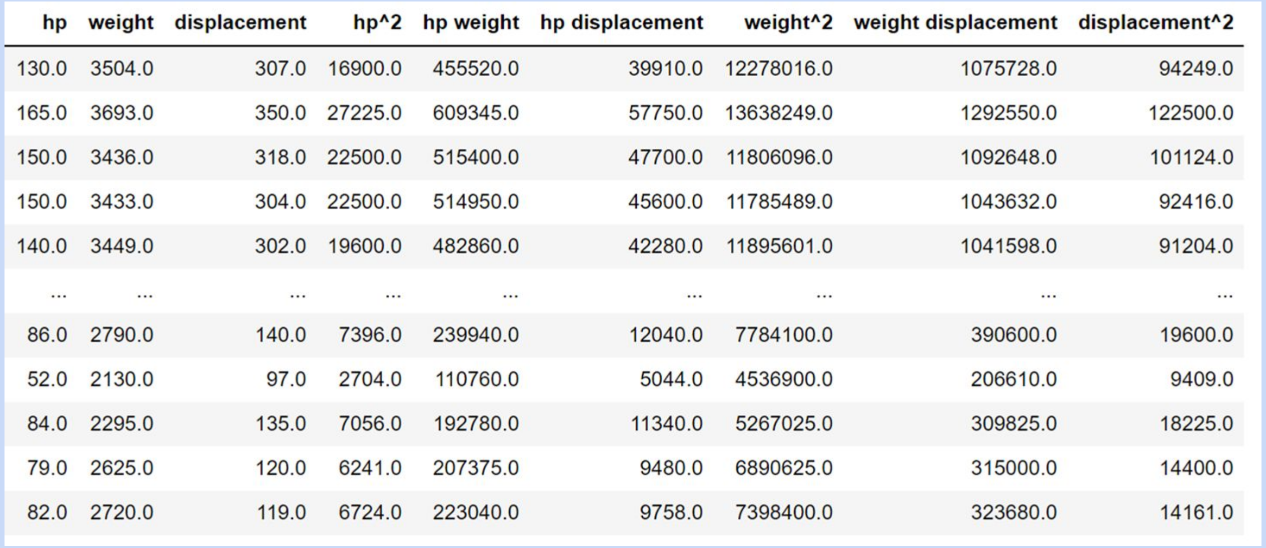
3726997

Back to our 9 feature model from before (d = 9).

- If we pick a very large ball radius, what kind of model will we have?

a. A model that only returns zero.
b. A constant model.
c. Ordinary least squares.
d. Something else.

$$\hat{y} = \theta_0 + \theta_1\phi_1 + \ldots + \theta_d\phi_d$$



hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0

Test Your Understanding Answer

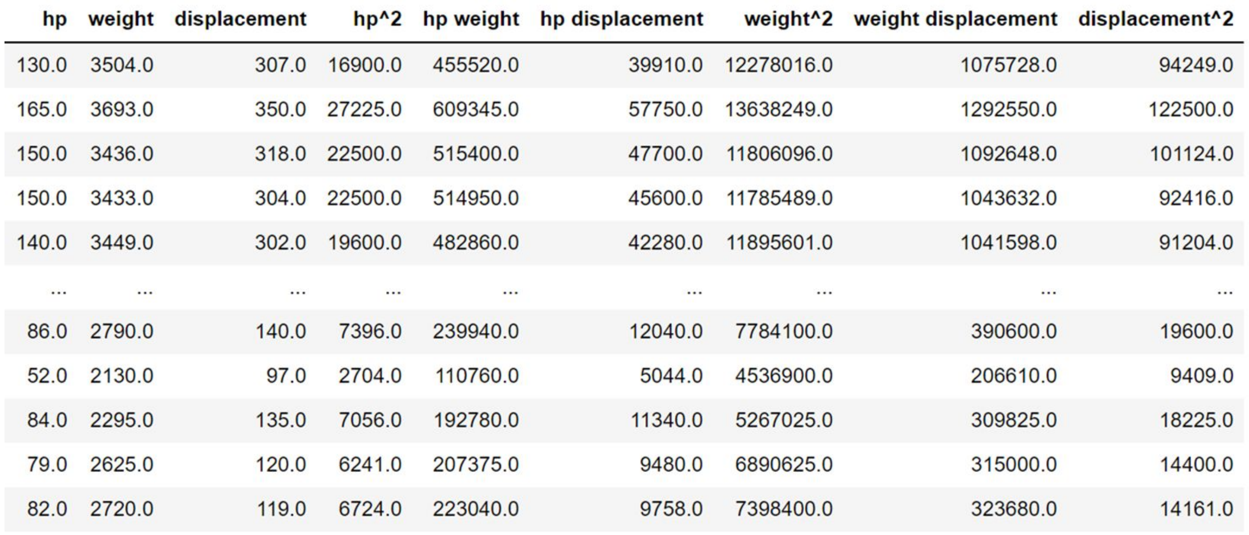
3726997

Back to our 9 feature model from before (d = 9).

- If we pick a very large ball radius, what kind of model will we have?

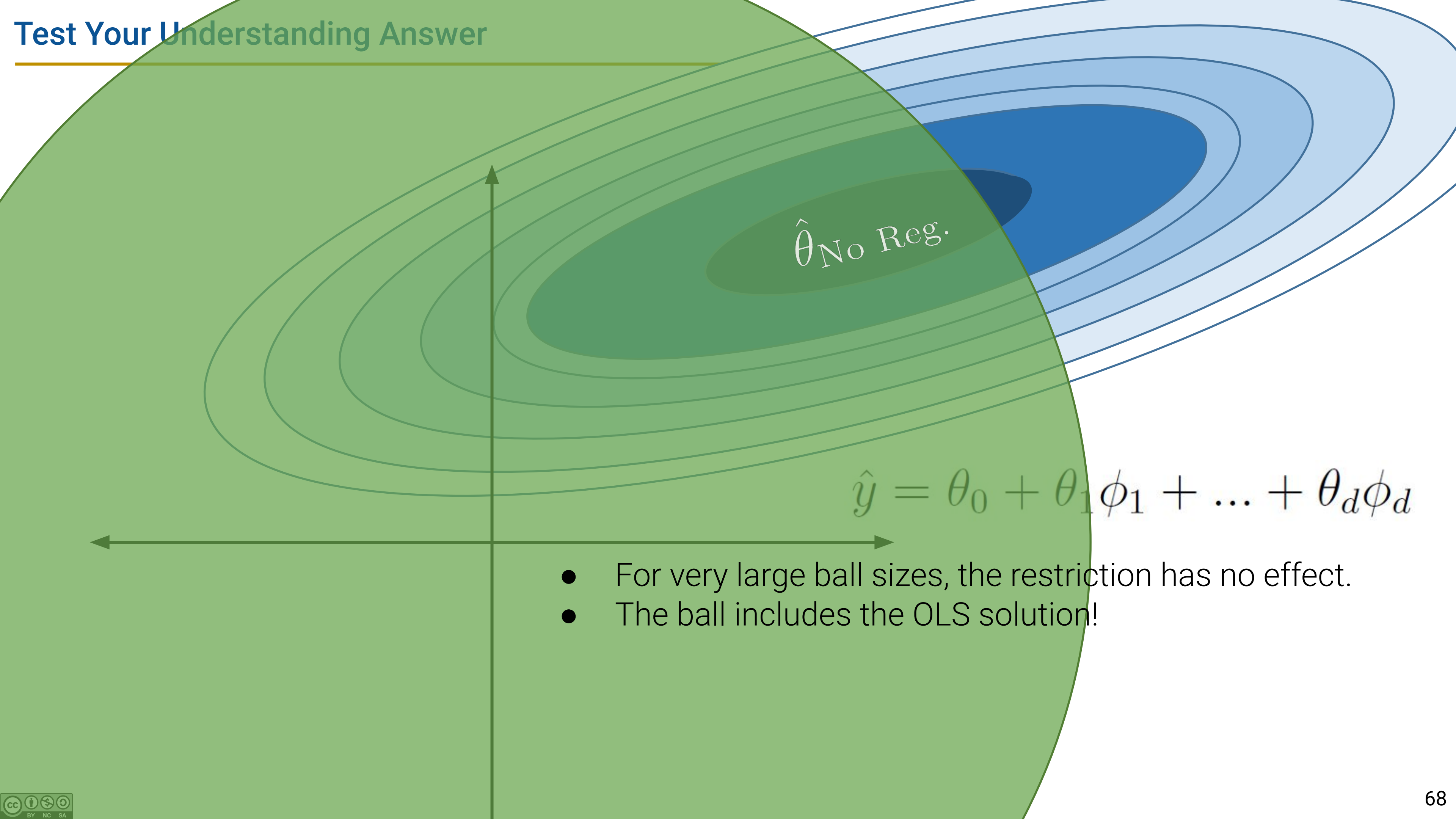
a. A model that only returns zero.
b. A constant model.
c. Ordinary least squares.
d. Something else.

$$\hat{y} = \theta_0 + \theta_1 \phi_1 + \ldots + \theta_d \phi_d$$



hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0

Test Your Understanding Answer



$$\hat{y} = \theta_0 + \theta_1\phi_1 + \ldots + \theta_d\phi_d$$

- For very large ball sizes, the restriction has no effect.
- The ball includes the OLS solution!

Training and Validation Errors vs. Ball Size for Our 9D Model

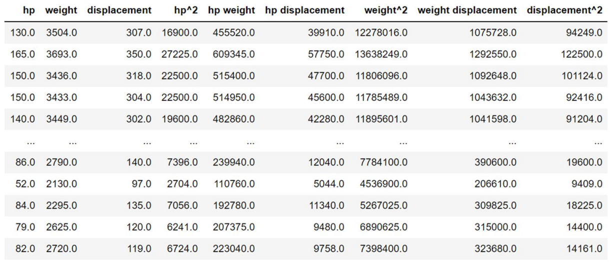
3726997

For very small ball size:

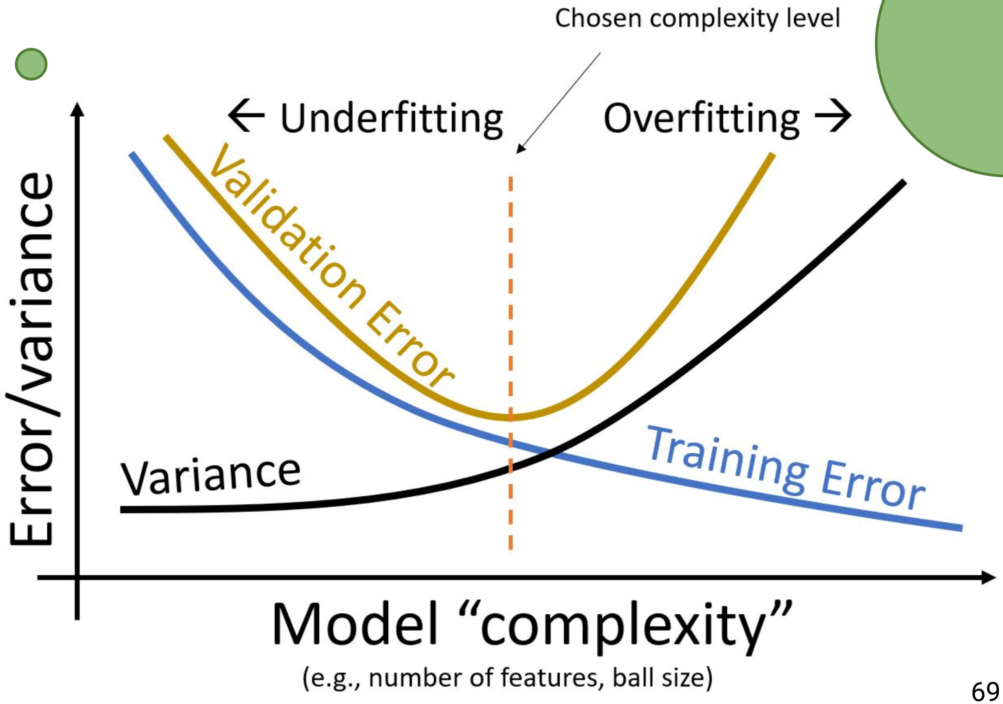
- Model behaves like a constant model. Can’t actually use our 9 features!
- High **training error**, low variance, high **validation error**.

For very large ball size:

- Model behaves like OLS.
- **If we have tons of features**, results in overfitting. Low **training error**, high variance, high **validation error**.



hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0



L2 Regularization

3726997

Constraining our model's parameters to a ball around the origin is called **L2 Regularization**.

- The smaller the ball, the simpler the model.

Ordinary least squares. Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \ldots + \theta_d\phi_{i,d}))^2$$

Ordinary least squares with **L2 regularization**. Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \ldots + \theta_d\phi_{i,d}))^2$$

Such that θ_1 through θ_d live inside a ball of radius Q.

L2 Regularization

3726997

Constraining our model's parameters to a ball around the origin is called **L2 Regularization**.

- The smaller the ball, the simpler the model.

Ordinary least squares. Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \ldots + \theta_d\phi_{i,d}))^2$$

Ordinary least squares with **L2 regularization**. Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \ldots + \theta_d\phi_{i,d}))^2 \qquad \textbf{such that} \quad \sum_{j=1}^{d}\theta_j^2 \leq Q$$

Note, intercept term not included!

Quick Detour into EECS127 (not tested in this class but worth mentioning)

In 127, you'll learn (through the magic of Lagrangian Duality) that the two problems below are equivalent:

Problem 1: Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \cdots + \theta_d\phi_{i,d}))^2 \quad \textbf{such that} \quad \sum_{j=1}^{d}\theta_j^2 \leq Q$$

Problem 2: Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \cdots + \theta_d\phi_{i,d}))^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

The "objective function" that gradient descent is minimizing now has an extra term.

Intuitively, this extra **right term penalizes large thetas**.

L2 Regularized Least Squares in sklearn

3726997

We can run least squares with an **L2 regularization term** by using the “Ridge” class.

```
from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha=10000)
ridge_model.fit(vehicle_data_with_squared_features, vehicle_data["mpg"])
```

Coefficients we get back:

```
ridge_model.coef_
```

```
array([-5.56414449e-02, -7.93804083e-03, -8.22081425e-02, -6.18785466e-04,
       -2.55492157e-05,  9.47353944e-04,  7.58061062e-07,  1.07439477e-05,
       -1.64344898e-04])
```

Note: sklearn’s “alpha” parameter is equivalent to λ in the linear regression with L2 regularizer equation

- Alpha is inversely related to the ball radius! Large alpha means small ball.

L2 Regularized Least Squares in sklearn

3726997

We can run least squares with an **L2 regularization term** by using the “Ridge” class.

```
from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha=10**-5)
ridge_model.fit(vehicle_data_with_squared_features, vehicle_data["mpg"])
```

For a tiny alpha, the coefficients are larger:

```
ridge_model.coef_
```

```
array([-1.35872588e-01, -1.46864458e-04, -1.18230336e-01, -4.03590098e-04,
       -1.12862371e-05,  8.25179864e-04, -1.17645881e-06,  2.69757832e-05,
       -1.72888463e-04])
```

Note: sklearn’s “alpha” parameter is equivalent to λ in the linear regression with L2 regularizer equation

- Alpha is inversely related to the ball radius! Large alpha means small ball.

L2 Regularized Least Squares in sklearn

We can run least squares with an **L2 regularization term** by using the “Ridge” class. For a tiny alpha, the coefficients are also about the same as a standard OLS model’s coefficients!

```
ridge_model.coef_
```

```
array([-1.35872588e-01, -1.46864458e-04, -1.18230336e-01, -4.03590098e-04,
       -1.12862371e-05,  8.25179864e-04, -1.17645881e-06,  2.69757832e-05,
       -1.72888463e-04])
```

```
from sklearn.linear_model import LinearRegression
linear_model = LinearRegression()
linear_model.fit(vehicle_data_with_squared_features, vehicle_data["mpg"])
```

```
linear_model.coef_
```

```
array([-1.35872588e-01, -1.46864447e-04, -1.18230336e-01, -4.03590097e-04,
       -1.12862370e-05,  8.25179863e-04, -1.17645882e-06,  2.69757832e-05,
       -1.72888463e-04])
```

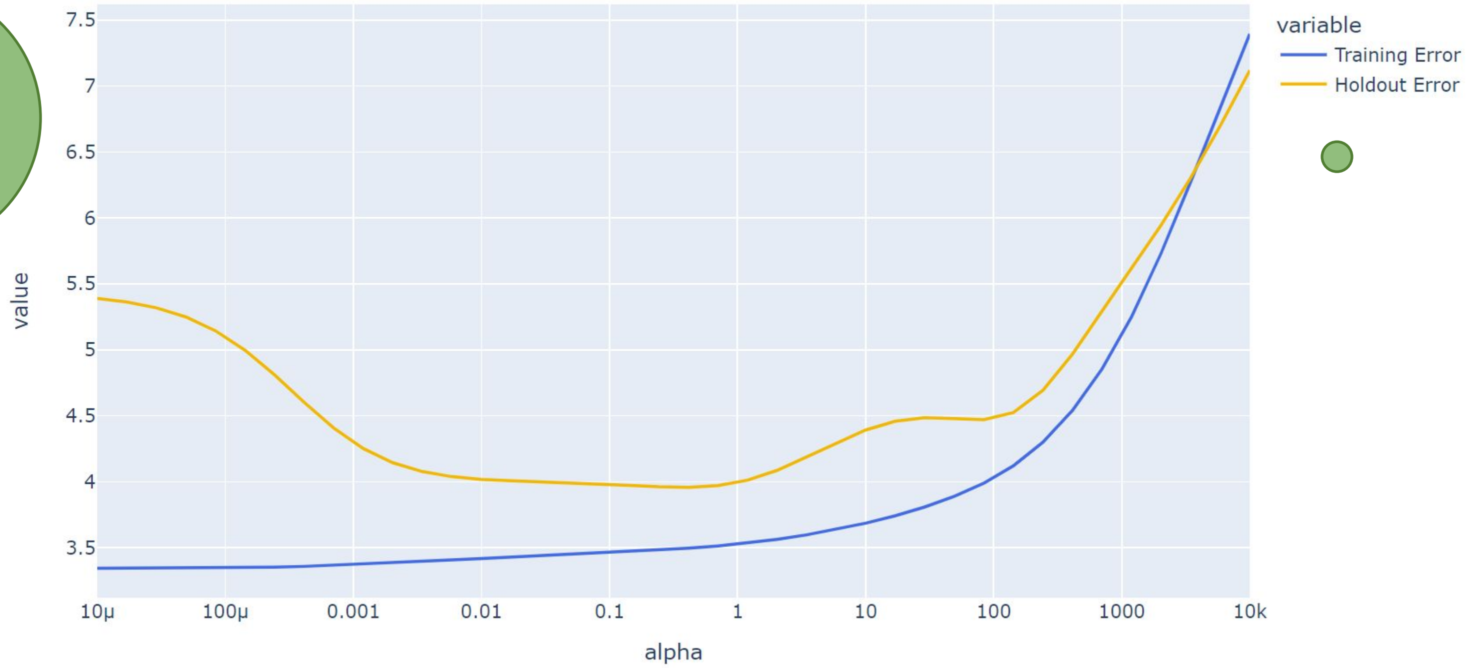
Green ball includes the OLS solution!

Figure (from lab 8)

3726997

In lab8, you'll run an experiment for different values of alpha. The resulting plot is shown below.

- Note: Since alpha is the inverse of the ball radius, the complexity is higher on the left!



Terminology Note

3726997

Why does sklearn use the word “Ridge”?

Because least squares with an **L2 regularization term** is also called “**Ridge Regression**”.

- Term is historical. Doesn’t really matter.

Why does sklearn use a hyperparameter which is the inverse of the ball radius?

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \cdots + \theta_d\phi_{i,d}))^2 + \lambda\sum_{j=1}^{d}\theta_j^2$$

Mathematical Note

3726997

Ridge Regression has a closed form solution which we will not derive.

- Note: The solution exists even if the feature matrix has collinearity between its columns.

$$\hat{\theta}_{ridge} = (\mathbb{X}^T\mathbb{X} + n\lambda I)^{-1}\mathbb{X}^T\mathbb{Y}$$

Identity matrix

3726997

Scaling Data for Regularization

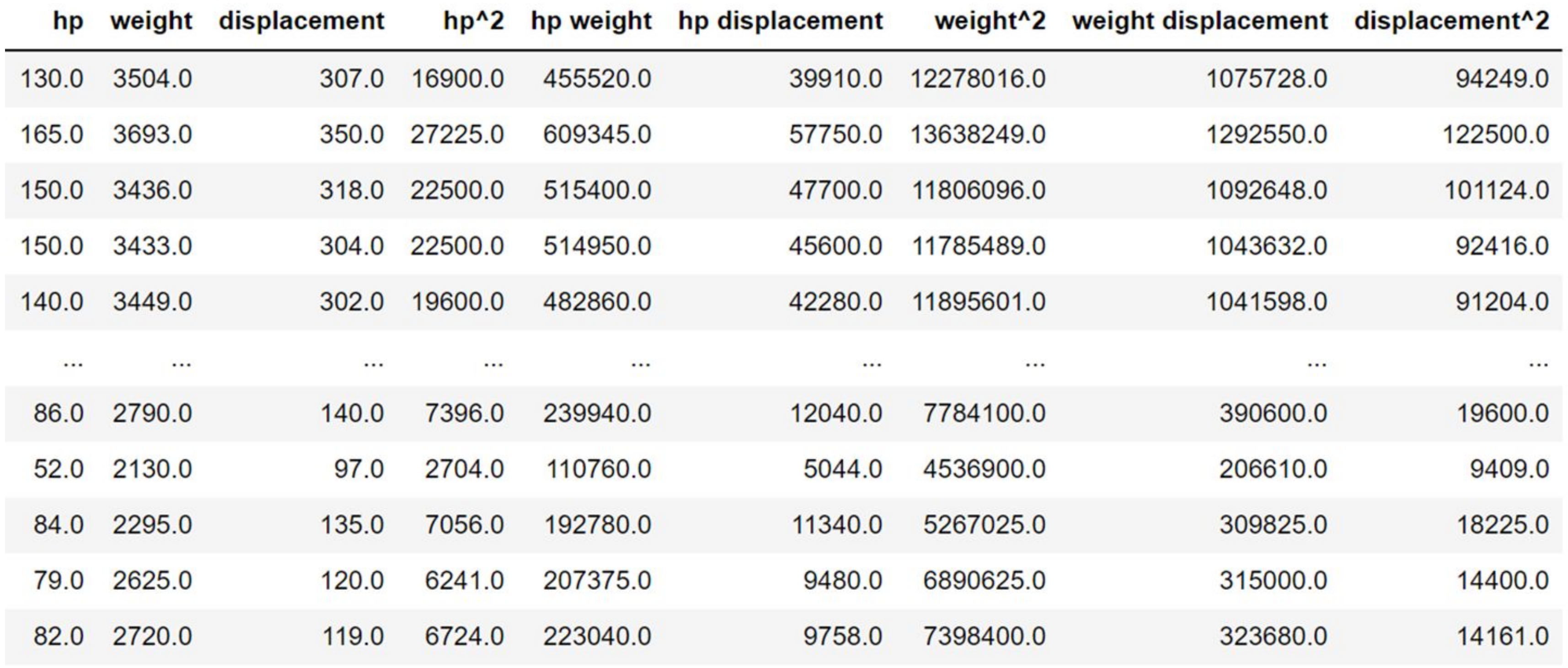
Lecture 15, Data 100 Spring 2023

One Issue With Our Approach

3726997

Our data from before has features of quite different numerical scale!

- Optimal theta for hp will probably be much further from origin than theta for weight^2.

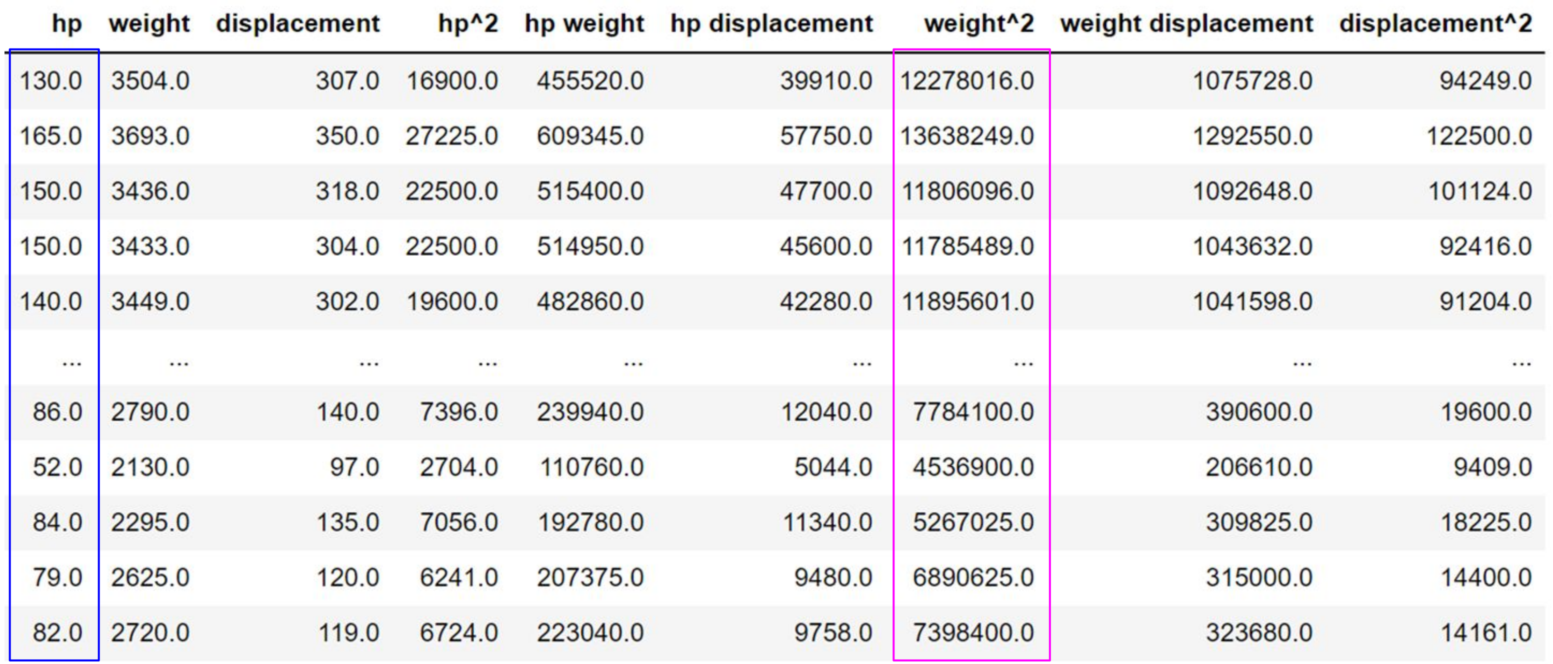


hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0

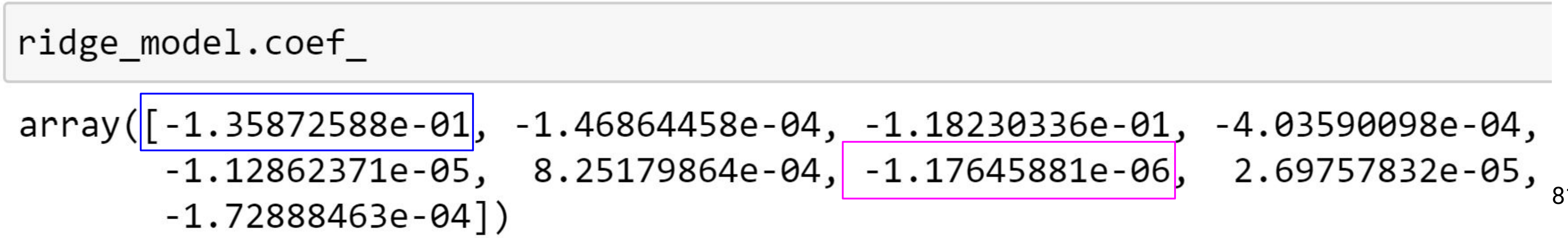
Theta will tend to be smaller for weight^2 than other parameters

Coefficients from Earlier

3726997



hp	weight	displacement	hp^2	hp weight	hp displacement	weight^2	weight displacement	displacement^2
130.0	3504.0	307.0	16900.0	455520.0	39910.0	12278016.0	1075728.0	94249.0
165.0	3693.0	350.0	27225.0	609345.0	57750.0	13638249.0	1292550.0	122500.0
150.0	3436.0	318.0	22500.0	515400.0	47700.0	11806096.0	1092648.0	101124.0
150.0	3433.0	304.0	22500.0	514950.0	45600.0	11785489.0	1043632.0	92416.0
140.0	3449.0	302.0	19600.0	482860.0	42280.0	11895601.0	1041598.0	91204.0
...	...	...	...	...	...	...	...	...
86.0	2790.0	140.0	7396.0	239940.0	12040.0	7784100.0	390600.0	19600.0
52.0	2130.0	97.0	2704.0	110760.0	5044.0	4536900.0	206610.0	9409.0
84.0	2295.0	135.0	7056.0	192780.0	11340.0	5267025.0	309825.0	18225.0
79.0	2625.0	120.0	6241.0	207375.0	9480.0	6890625.0	315000.0	14400.0
82.0	2720.0	119.0	6724.0	223040.0	9758.0	7398400.0	323680.0	14161.0



```
ridge_model.coef_
```

```
array([-1.35872588e-01, -1.46864458e-04, -1.18230336e-01, -4.03590098e-04,
       -1.12862371e-05,  8.25179864e-04, -1.17645881e-06,  2.69757832e-05,
       -1.72888463e-04])
```

Making Things Fair

Ideally, our data should all be on the same scale.

- One approach: Standardize the data, i.e. replace everything with its Z-score.

$$z_k = \frac{x_k - \mu_k}{\sigma_k}$$

- Resulting model coefficients will be all on the same scale.

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
rescaled_df = pd.DataFrame(ss.fit_transform(vehicle_data_with_squared_features),
                           columns = ss.get_feature_names_out())


ridge_model = Ridge(alpha=10000)
ridge_model.fit(rescaled_df, vehicle_data["mpg"])
ridge_model.coef_
```

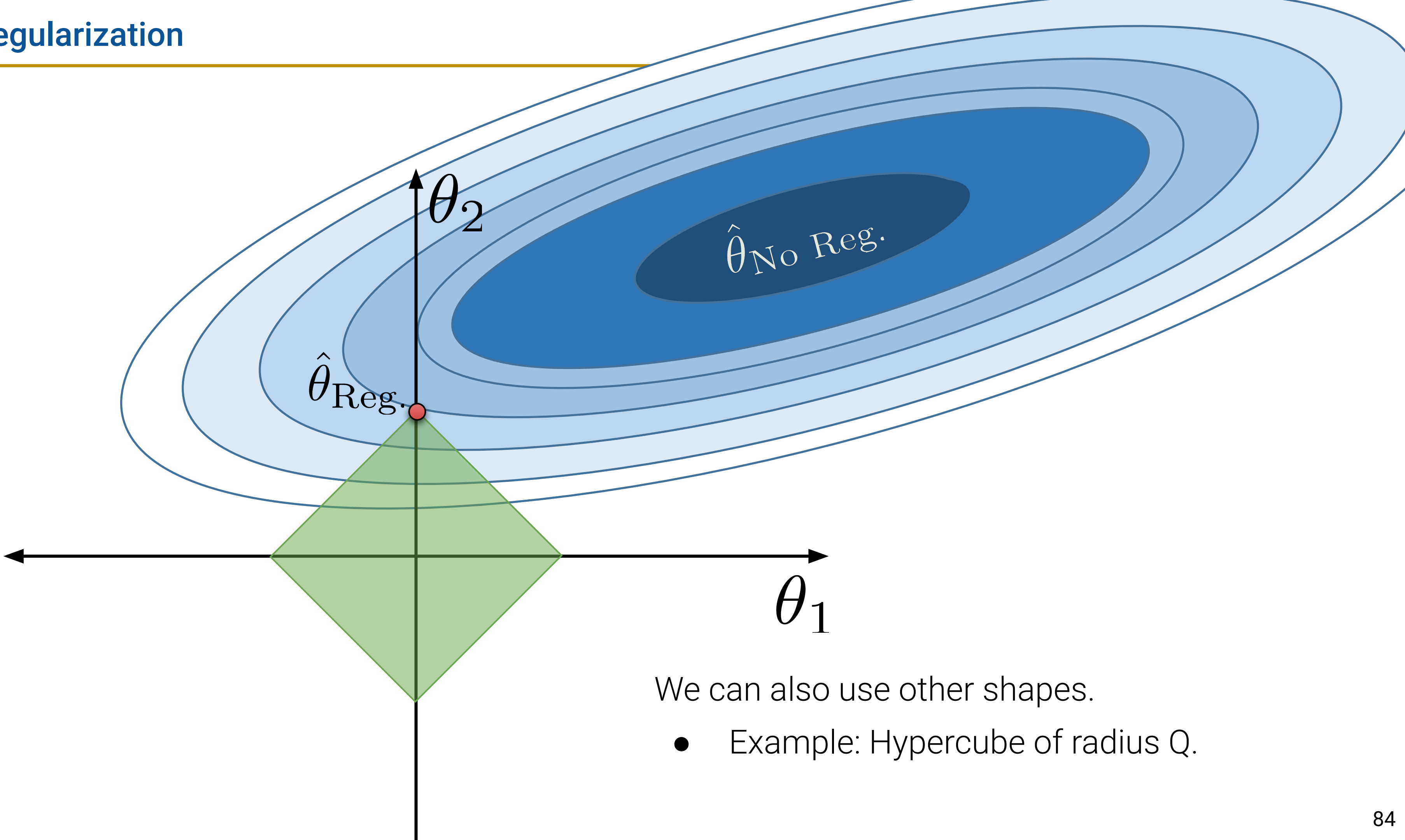
```
array([-0.1792743 , -0.19610513, -0.18648617, -0.1601219 , -0.18015125,
       -0.16858023, -0.18779478, -0.18176294, -0.17021841])
```

3726997

L1 Regularization (LASSO)

Lecture 15, Data 100 Spring 2023

L1 Regularization



We can also use other shapes.

- Example: Hypercube of radius Q.

L1 Regularization in Equation Form

3726997

Using a hypercube is known as **L1 regularization**. Expressed mathematically in the two equivalent forms below:

Problem 1: Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \cdots + \theta_d\phi_{i,d}))^2 \quad \textbf{such that} \quad \sum_{j=1}^{d}|\theta_j| \le Q$$

Problem 2: Find thetas that minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - (\theta_0 + \theta_1\phi_{i,1} + \cdots + \theta_d\phi_{i,d}))^2 + \lambda\sum_{j=1}^{d}|\theta_j|$$

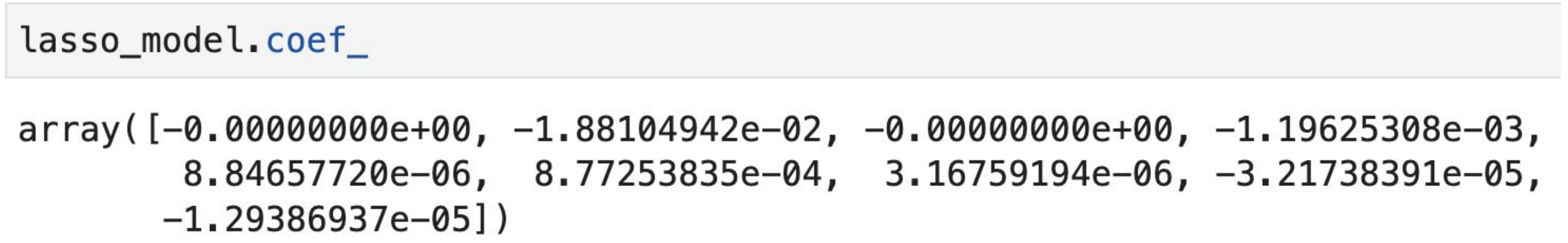
L1 Regularized OLS in sklearn

3726997

In sklearn, we use the Lasso module.

- Note: Performing OLS with L1 regularization is also called **LASSO regression**.

```
from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha = 10)
lasso_model.fit(vehicle_data_with_squared_features, vehicle_data["mpg"])
lasso_model.coef_
```



```
lasso_model.coef_
```

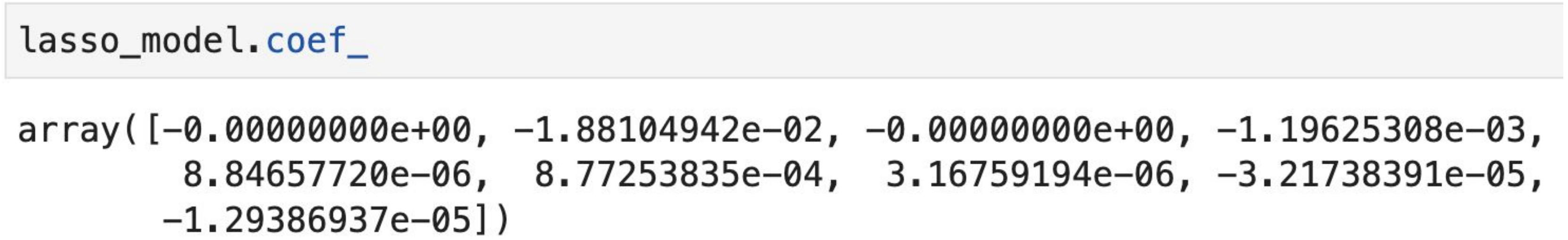
```
array([-0.00000000e+00, -1.88104942e-02, -0.00000000e+00, -1.19625308e-03,
        8.84657720e-06,  8.77253835e-04,  3.16759194e-06, -3.21738391e-05,
       -1.29386937e-05])
```

LASSO and “Feature Selection”

3726997

The optimal parameters for a LASSO model tend to include a lot of zeroes! In other words, LASSO effectively selects only a subset of the features.

```
from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha = 10)
lasso_model.fit(vehicle_data_with_squared_features, vehicle_data["mpg"])
lasso_model.coef_
```



```
lasso_model.coef_
```

```
array([-0.00000000e+00, -1.88104942e-02, -0.00000000e+00, -1.19625308e-03,
        8.84657720e-06,  8.77253835e-04,  3.16759194e-06, -3.21738391e-05,
       -1.29386937e-05])
```

Intuitive reason:

- Imagine expanding a 3D cube until it intersects a balloon. More likely to intersect at a corner or edge than a face (especially in high dimensions)

Summary of Regression Methods

3726997

Our regression models are summarized below.

- The "Objective" column gives the function that our gradient descent optimizer minimizes.
- Note that this table uses lambda instead of alpha for regularization strength. Both are common.

Name	Model	Loss	Reg.	Objective	Solution
OLS	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	Squared loss	None	$\frac{1}{n}\lvert\lvert\mathbb{Y} - \mathbb{X}\theta\rvert\rvert_2^2$	$\hat{\theta}_{\text{OLS}} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$
Ridge Regression	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	Squared loss	L2	$\frac{1}{n}\lvert\lvert\mathbb{Y} - \mathbb{X}\theta\rvert\rvert_2^2 + \lambda\sum_{j=1}^{d}\theta_i^2$	$\hat{\theta}_{\text{ridge}} = (\mathbb{X}^T\mathbb{X} + n\lambda I)^{-1}\mathbb{X}^T\mathbb{Y}$
LASSO	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	Squared loss	L1	$\frac{1}{n}\lvert\lvert\mathbb{Y} - \mathbb{X}\theta\rvert\rvert_2^2 + \lambda\sum_{j=1}^{d}\lvert\theta_i\rvert$	**No closed form**

Lecture 15

Cross Validation, Regularization