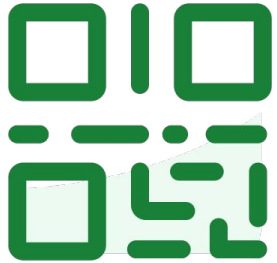




slido



Join at [slido.com](https://slido.com)  
#4531760

① Click **Present with Slido** or install our [Chrome extension](#) to display joining instructions for participants while presenting.



4531760

## Lecture 16

# Cross-Validation, Regularization

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

**Data 100/Data 200, Spring 2025 @ UC Berkeley**

Narges Norouzi and Josh Grossman

Content credit: [Acknowledgments](#)



# Goals for this Lecture

---

Lecture 16, Data 100 Spring 2025

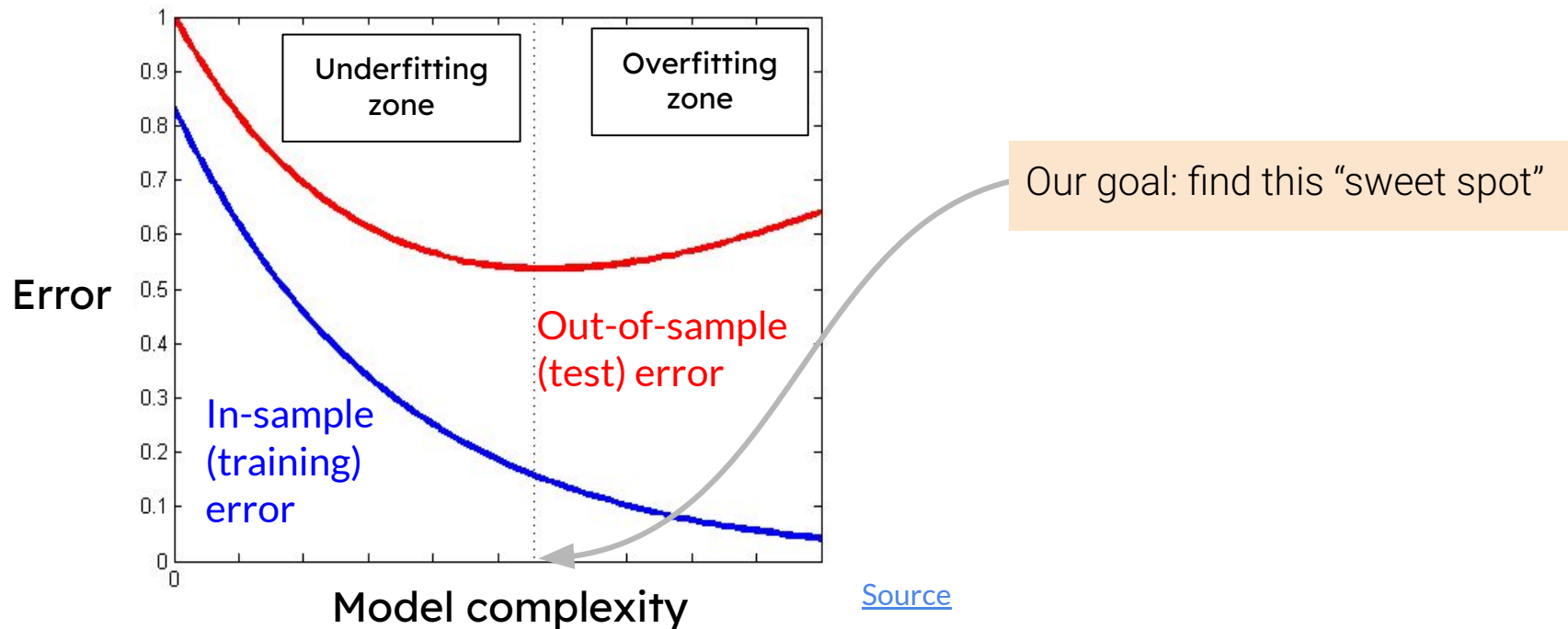
Circling back to two questions raised in the modeling lectures:

- How does model performance change on "seen" vs. "unseen" data?
- How do we control model complexity?



At the end of Lecture 14, we had two lingering thoughts:

- The idea of "unseen" data – data that the model did not encounter during training
- The idea of model complexity – a model's complexity influences if it under- or overfits

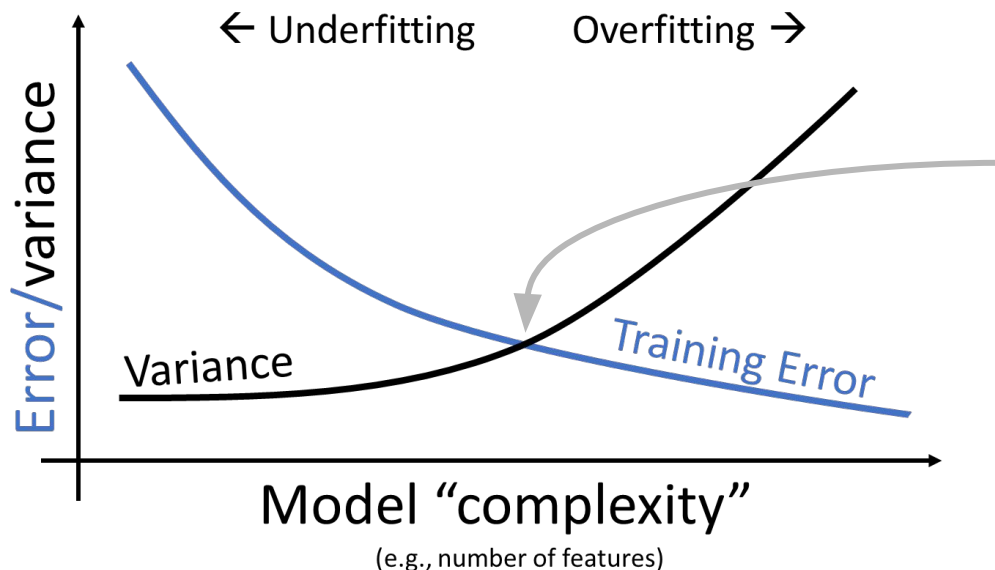


[Source](#)



At the end of Lecture 14, we had two lingering thoughts:

- The idea of "unseen" data – data that the model did not encounter during training
- The idea of model complexity – a model's complexity influences if it under- or overfits



Our goal: find this "sweet spot"

Stay tuned for ~30 min from now



4531760

At the end of Lecture 14, we had two lingering thoughts:

- The idea of "unseen" data – data that the model did not encounter during training.
- The idea of model complexity – a model's complexity influences if it under- or overfits.

In today's lecture, we will address both topics in detail.

First half of lecture: **Cross-Validation**

- Formalize the idea of training and test data.
- Introduce a method to "preview" how a model will perform on unseen data.

Second half of lecture: **Regularization**

- Introduce a method to finetune our model's complexity.



4531760

# Agenda

---

Lecture 16, Data 100 Spring 2025

## Cross-Validation

- Training, Test, and Validation Sets
- K-Fold Cross-Validation

## Regularization

- Constraining Model Parameters
- L2 Regularization (Ridge)
- L1 Regularization (LASSO)



4531760

# Training, Test, and Validation Sets

---

Lecture 16, Data 100 Spring 2025

## Cross-Validation

- **Training, Test, and Validation Sets**
- K-Fold Cross-Validation

## Regularization

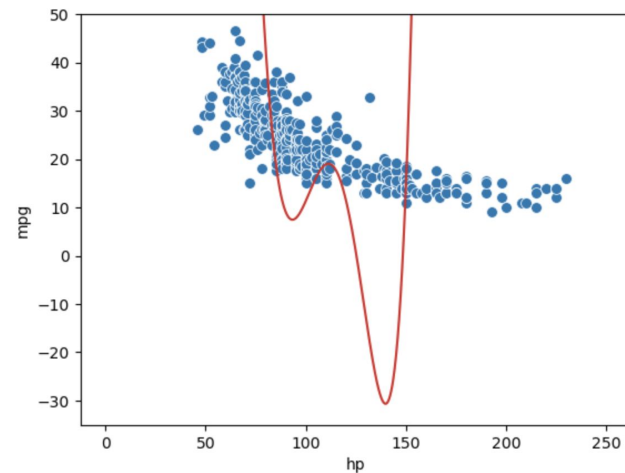
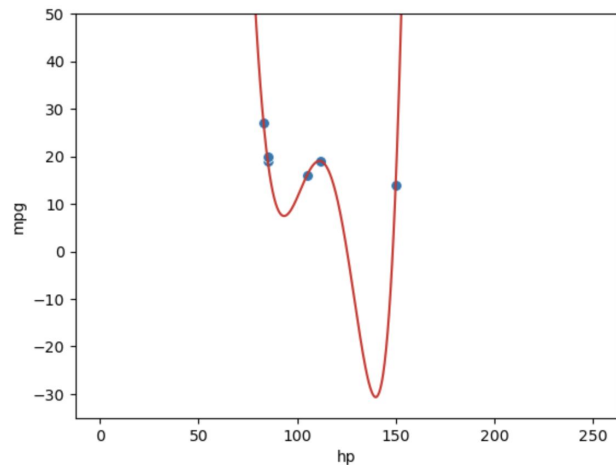
- Constraining Model Parameters
- L2 Regularization (Ridge)
- L1 Regularization (LASSO)





4531760

A complex model may not perform well on data it did not encounter during training.



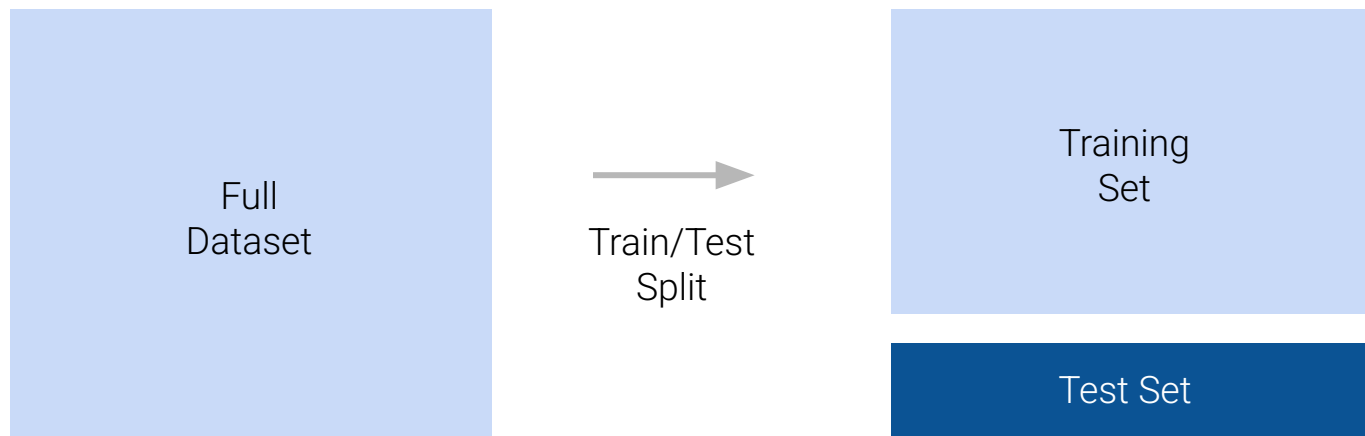
How to quantify performance on this "unseen" data? Introduce a **test set**.



A **test set** is a portion of our dataset that we set aside for testing purposes.

- We do *not* consider the test set when fitting/training the model.
- The test set is only ever touched once: to compute the performance (MSE, RMSE, etc) of the model *after* all fine-tuning has been completed.

**Our new workflow for modeling:** First, perform a **train-test split** (see [documentation](#)). Consider only the training set when designing the model. Then, evaluate on the test set.





What if we were dissatisfied with our test set performance?

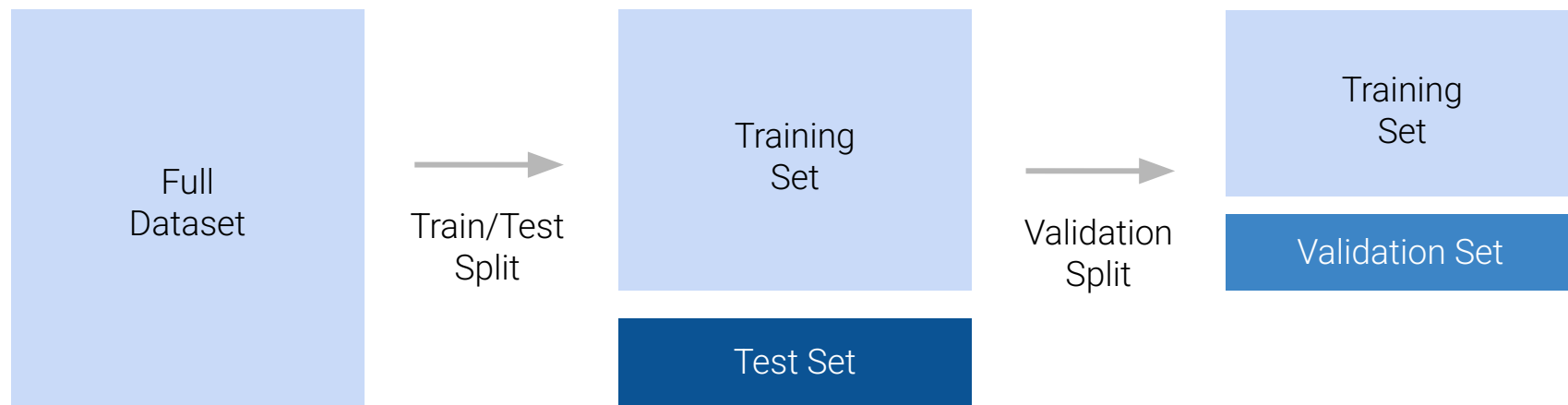
In our current framework, we'd be stuck – we can't then go back and adjust our model, because we'd be *factoring in information from the test set* to design our model. The test set would no longer represent performance on unseen data.

**Solution:** introduce a **validation set**.



A **validation set** is a portion of our *training set* that we set aside for assessing model performance while it is *still being developed*.

- Train model on the training set. Assess performance on the validation set. Adjust the model, then repeat.
- After *all* model development is complete, assess final performance on the test set.

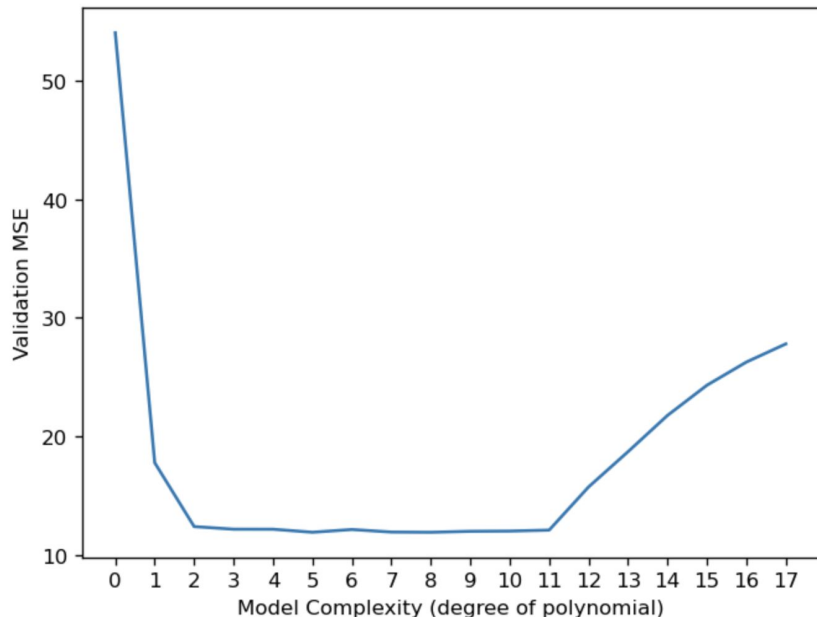


# Updating Our Understanding of Model Complexity

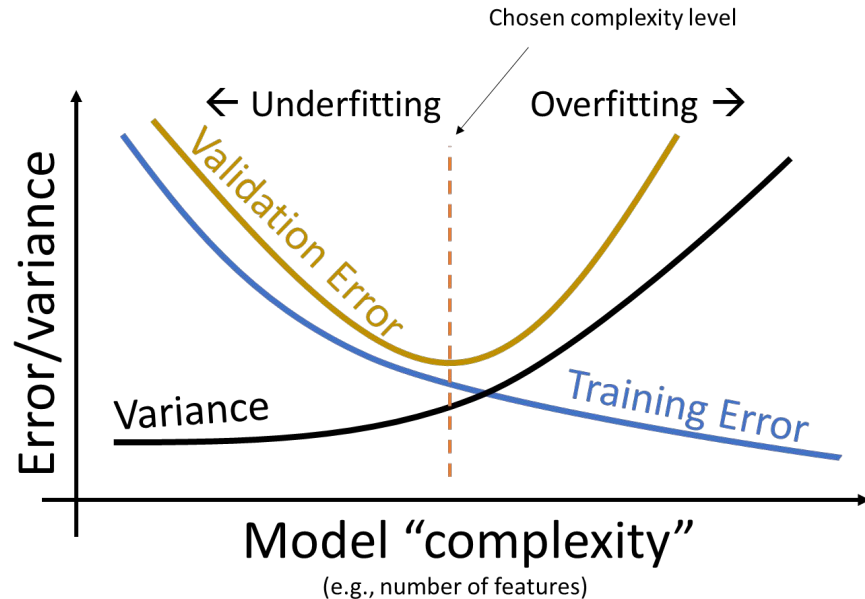


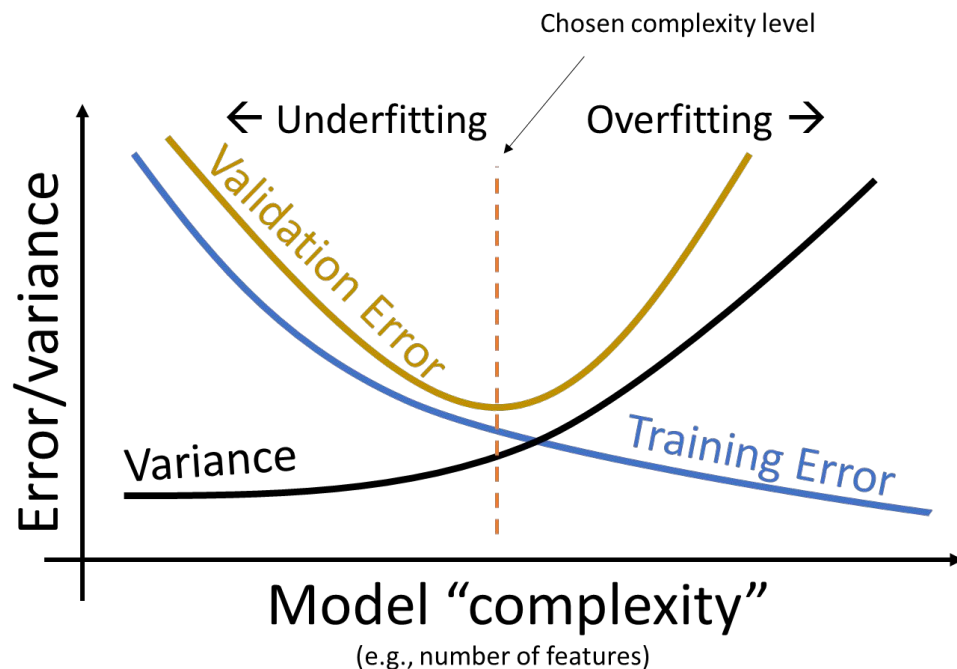
Computing the validation error allows us to visualize under- and overfitting.

Our experiment using **vehicles**



General trend for an arbitrary dataset





Typically, as model complexity increases:

- Training error decreases
- Variance increases
- Error on validation set decreases, then increases

**Our goal: Choose the model complexity that minimizes validation error.**

We will discuss how in the second half of lecture.

In Lectures 17 and 18, we'll learn the mathematical origins of these relationships



4531760

# K-Fold Cross-Validation

---

Lecture 16, Data 100 Spring 2025

## Cross-Validation

- Training, Test, and Validation Sets
- **K-Fold Cross-Validation**

## Regularization

- Constraining Model Parameters
- L1 Regularization (LASSO)
- L2 Regularization (Ridge)



Introducing a validation set gave us one "extra" chance to assess model performance.

Specifically, now we understand how the model performs on *one* particular set of unseen data.

- It's possible that we may have, by random chance, selected a set of validation points that was *not* representative of other unseen data that the model might encounter.

```
Val error from train/validation split #1: 14.6104005581132
```

```
Val error from train/validation split #2: 24.755706579814404
```

```
Val error from train/validation split #3: 22.23208329959848
```

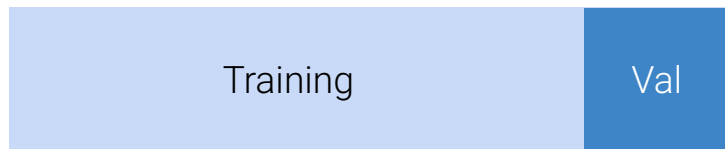
Ideally: Assess model performance on *several* different validation sets before touching the test set.





In our original validation split, we set aside  $x\%$  of the training data to use for validation.

- For example, 20% of the training data is used for validation



We could have selected *any* 20% portion of the training data for validation.

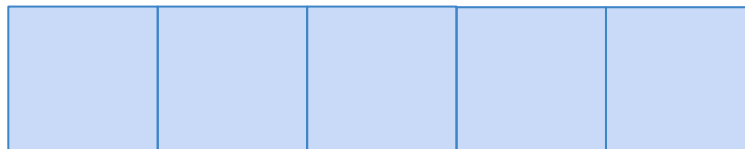


In total, there are 5 non-overlapping “chunks” of datapoints we could set aside for validation.



The common term for one of these chunks is a "fold".

- Our training data has 5 folds, each containing 20% of the datapoints.



Another perspective: we actually have 5 validation sets "hidden" in our training set.

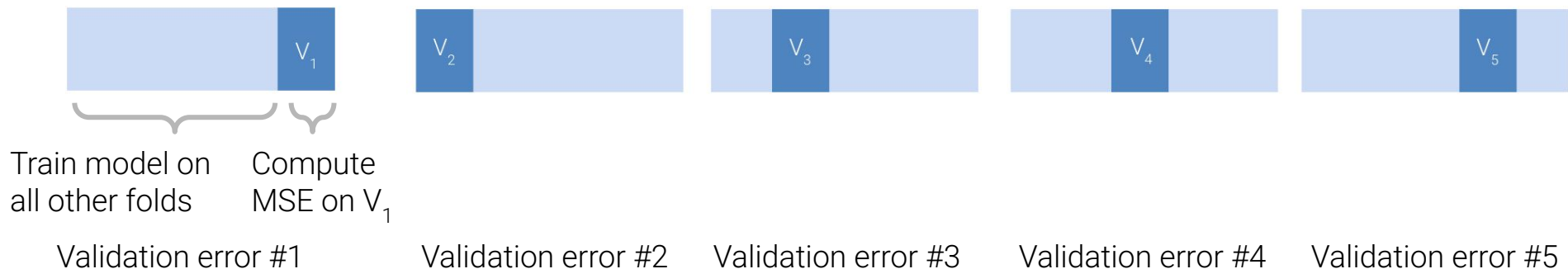
In **cross-validation**, we perform validation splits for *each* of these folds.



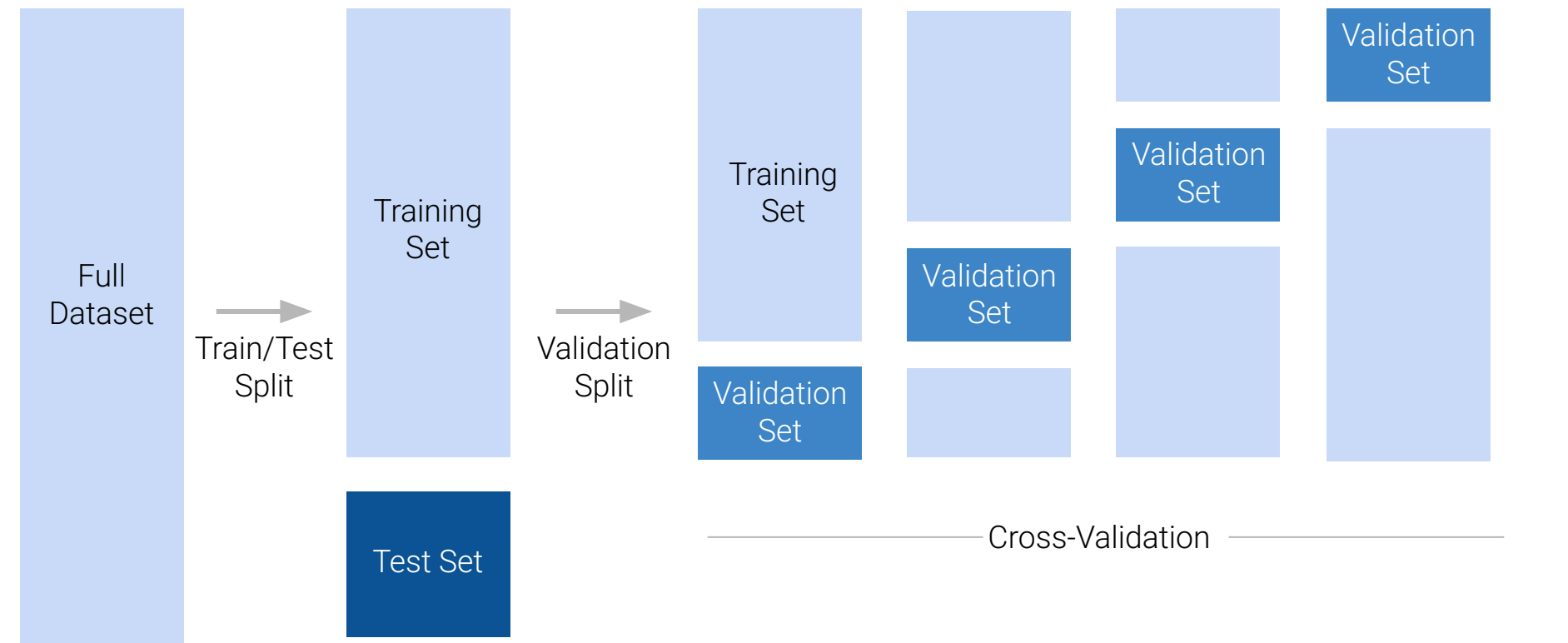
For a dataset with K folds:

- Pick one fold to be the validation fold.
- Train model on data from every fold *other* than the validation fold.
- Compute the model's error on the validation fold and record it.
- Repeat for all K folds.

The **cross-validation error** is the average error across all K validation folds.



Cross-validation error = mean of validation errors #1 to #5





Cross-validation is often used for **hyperparameter** selection.

**Hyperparameter:** Value in a model chosen *before* the model is fit to data.

- Cannot solve for hyperparameters via calculus, OLS, gradient descent, etc – we must choose it ourselves.
- Examples
  - Degree of polynomial model
  - Gradient descent learning rate,  $\alpha$
  - Regularization penalty,  $\lambda$  (to be introduced later this lecture)

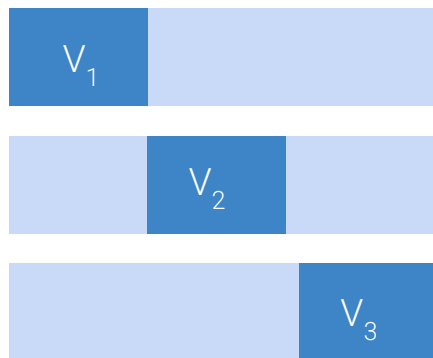


To select a hyperparameter value via cross-validation:

- List out several different “guesses” for the best hyperparameter.
- For each guess, run cross-validation to compute the CV error for that choice of hyperparameter value.
- Select the hyperparameter value with lowest CV error.

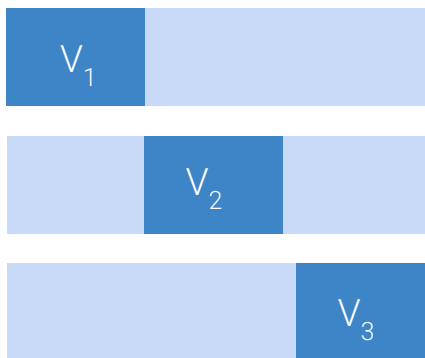
Example: Guesses for learning rate are 0.1, 1, and 10. We decide to apply 3-fold cross-validation.

$\alpha = 0.1$



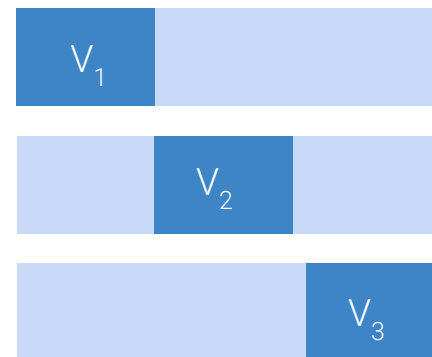
CV error: 4.67

$\alpha = 1$



CV error: 7.01

$\alpha = 10$

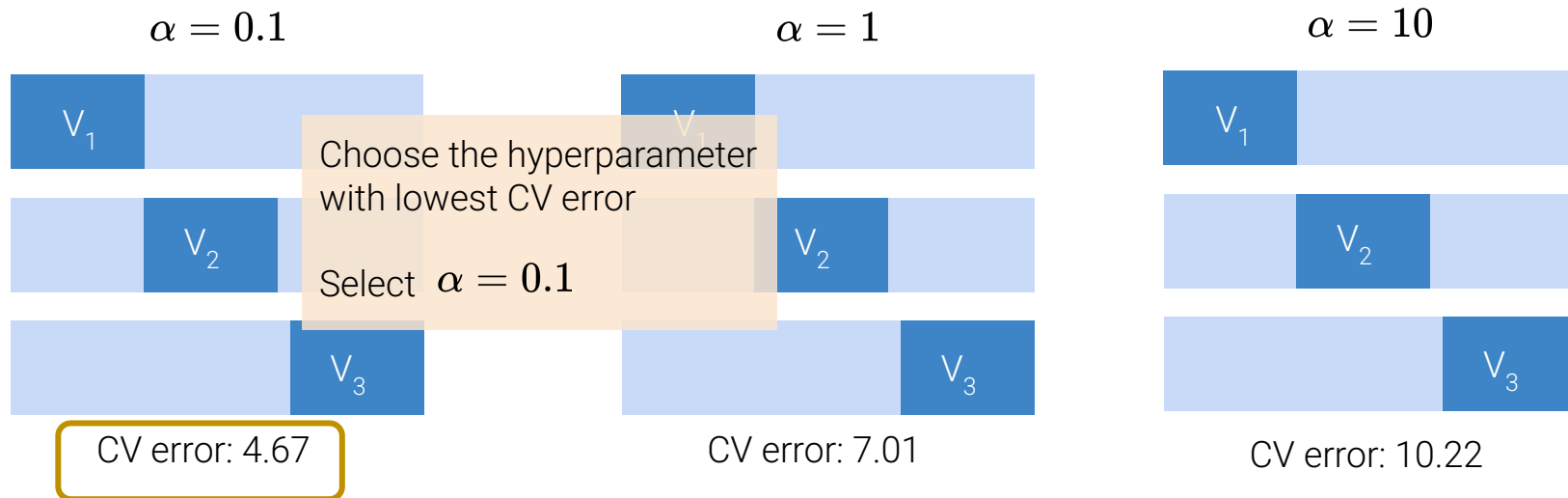


CV error: 10.22

To select a hyperparameter value via cross-validation:

- List out several different “guesses” for the best hyperparameter
- For each guess, run cross-validation to compute the CV error for that choice of hyperparameter value
- Select the hyperparameter value with lowest CV error

Example: Guesses for learning rate are 0.1, 1, and 10. We decide to apply 3-fold cross-validation.



slido



To select the best polynomial degree for our model out of 5 possible options, how many times will we compute the MSE on a validation fold during 4-fold cross-validation?

① Click **Present with Slido** or install our [Chrome extension](#) to activate this poll while presenting.





slido



# How many times would we train the model?

① Click **Present with Slido** or install our [Chrome extension](#) to activate this poll while presenting.



# Interlude

---





4531760

# Constraining Model Parameters

---

Lecture 16, Data 100 Spring 2025

## Cross-Validation

- Training, Test, and Validation Sets
- K-Fold Cross-Validation

## Regularization

- **Constraining Model Parameters**
- L2 Regularization (Ridge)
- L1 Regularization (LASSO)

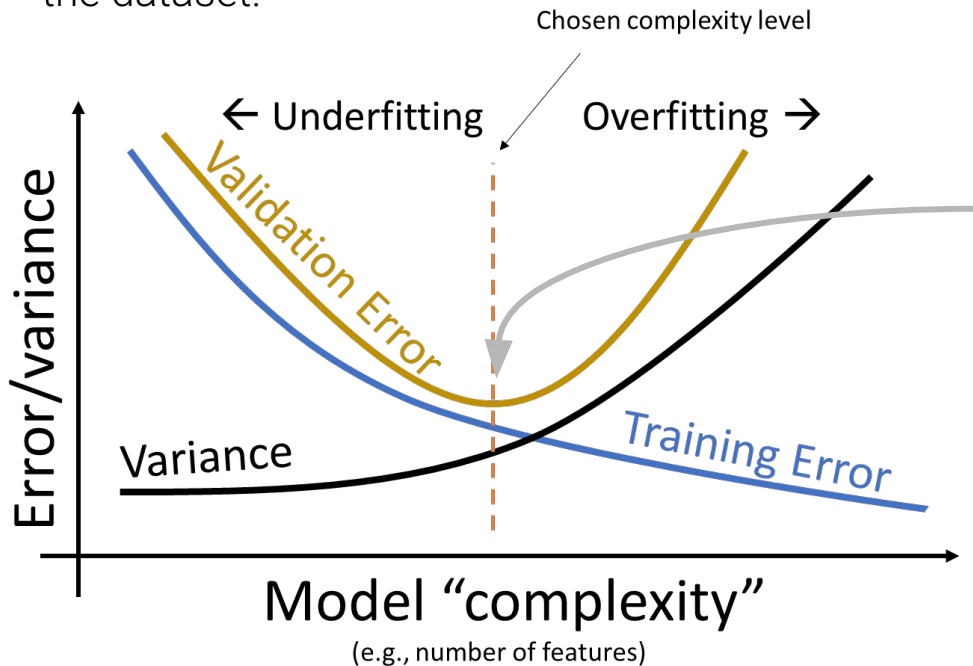


4531760

## Restricting Model Complexity

We've seen now that model complexity needs to be chosen carefully:

- Too complex: model overfits – "memorizes" training data too closely.
- Too simple: model underfits – does not take full advantage of the features available in the dataset.



Our goal: find this "sweet spot"

How do we control complexity?



**Idea:** Only use each feature "a little" in the model.

$$\hat{Y} = \theta_0 + \theta_1\phi_1 + \theta_2\phi_2 \dots + \theta_p\phi_p$$

- If we restrict how large each parameter  $\theta_i$  can be, we restrict how much each feature contributes to the model.
- When  $\theta_i$  is close to or equal to 0, the model decreases in complexity because feature  $\phi_i$  barely impacts the prediction.

In **regularization**, we restrict complexity by *putting a limit* on the magnitudes of the model parameters  $\theta_i$ .



In **regularization**, we restrict complexity by *putting a limit* on the magnitudes of the model parameters  $\theta_i$ .

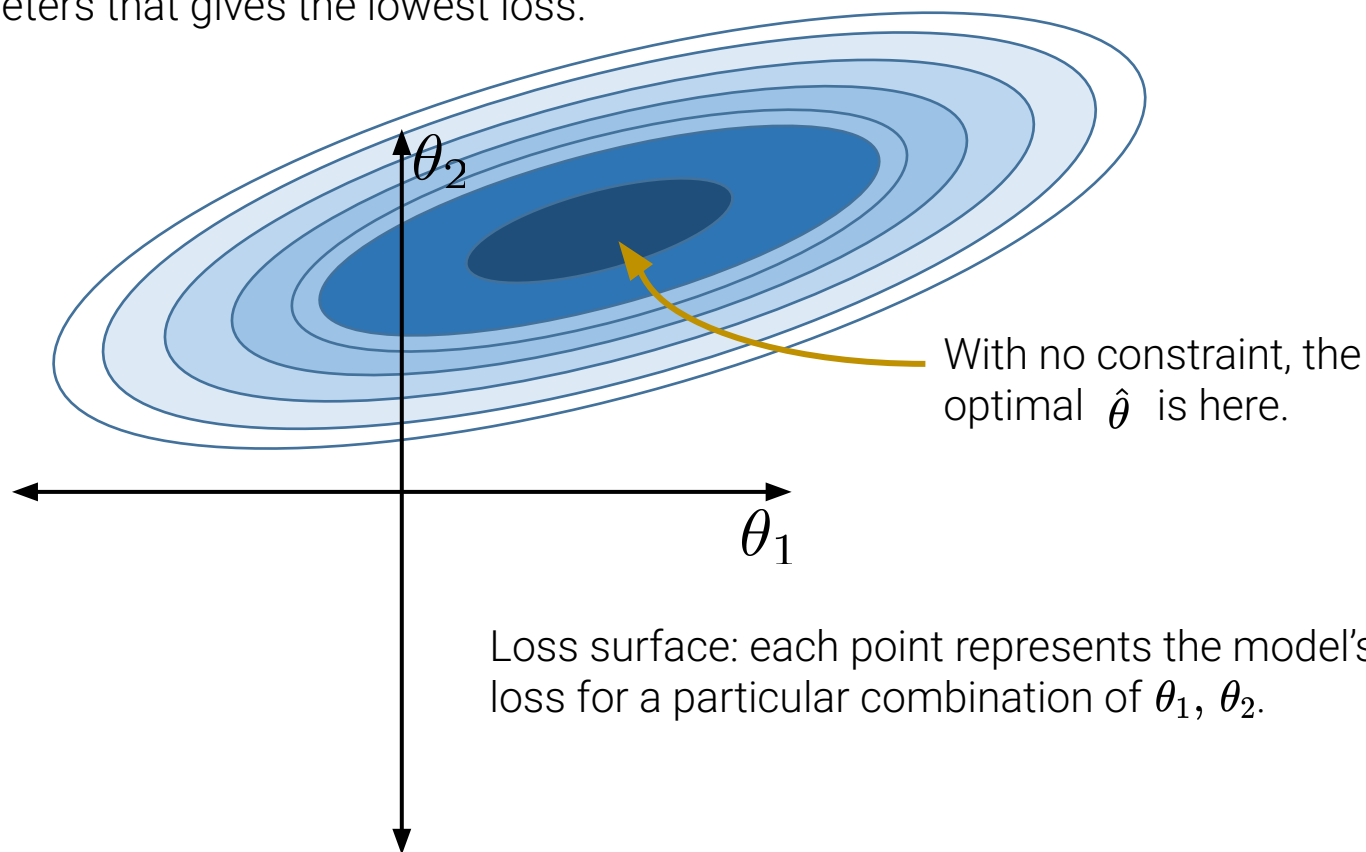
Example: Suppose we specify that the sum of all absolute parameters can be no larger than some number  $Q$

$$\sum_{i=1}^p |\theta_i| \leq Q$$

We've given the model a “budget” for how much weight to assign to each feature. Some parameters  $\theta_i$  will need to be small in value so the sum remains below  $Q$ .

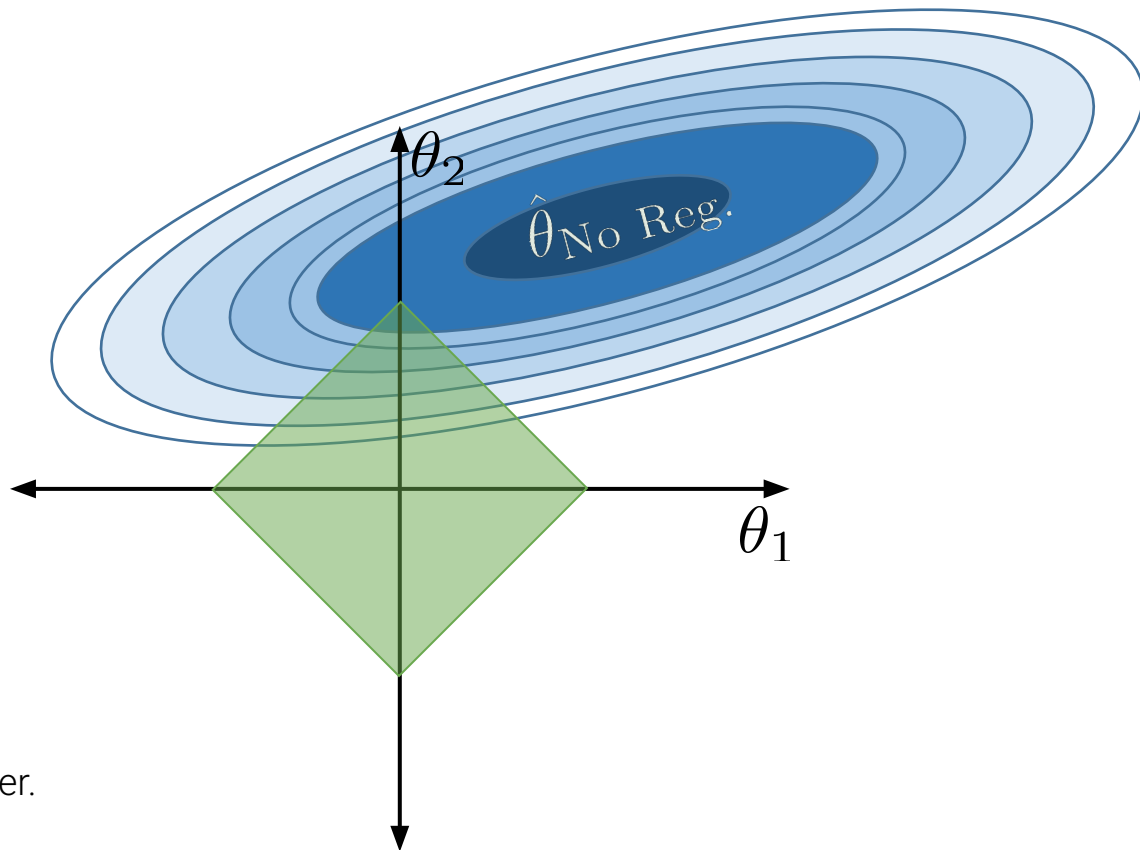
Note that the intercept term,  $\theta_0$ , is typically excluded from this constraint.

In gradient descent: visualize the loss surface as a contour map. Our goal is to find the combination of parameters that gives the lowest loss.





When we apply the constraint  $\sum_{i=1}^p |\theta_i| \leq Q$ , only parameter combinations inside the diamond are valid.



$Q$  is some arbitrary number.

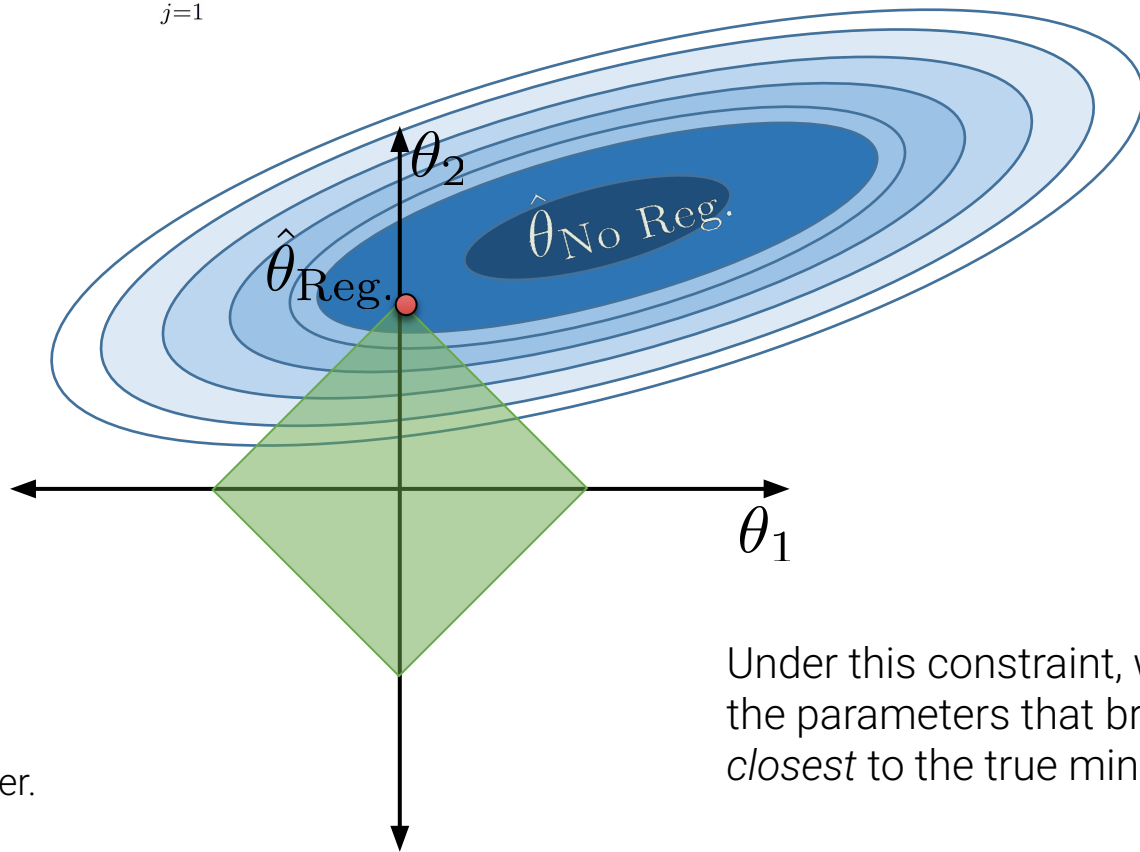




4531760

## Constraining Model Parameters

When we apply the constraint  $\sum_{j=1}^d |\theta_j| \leq Q$ , only parameter combinations inside the diamond are allowed.

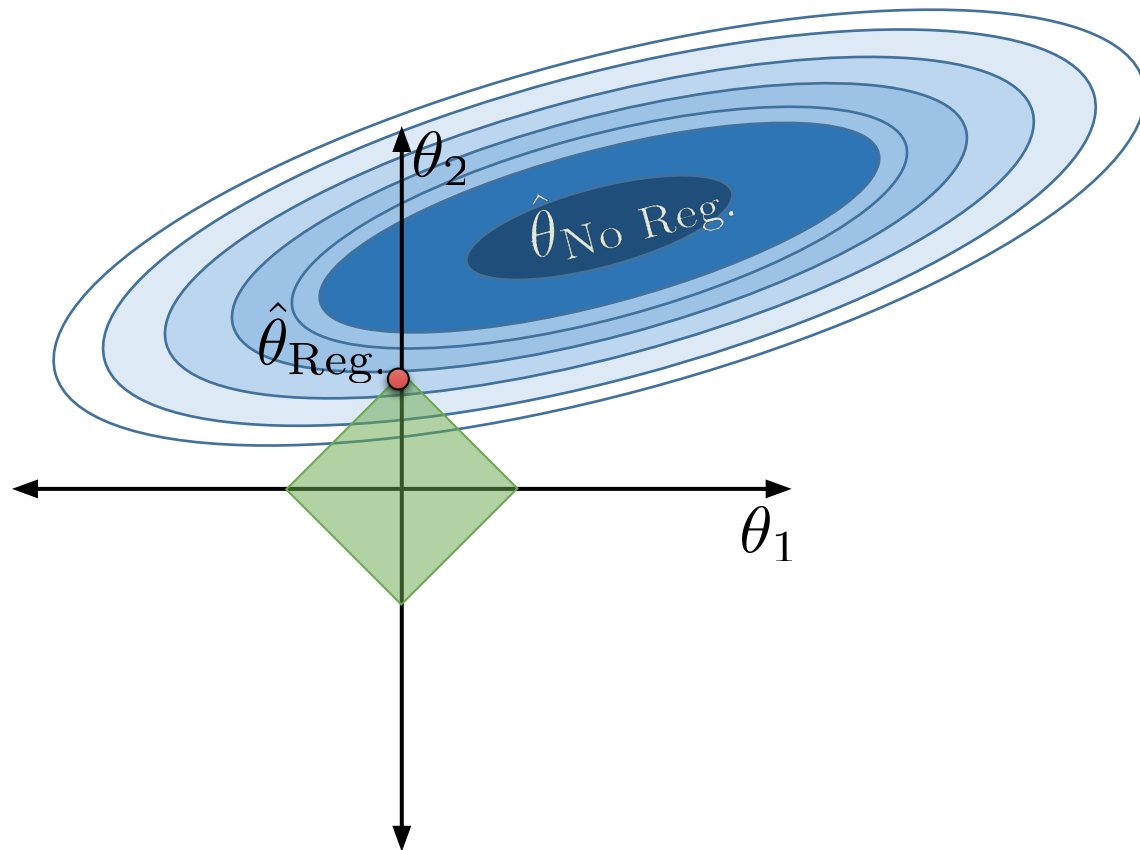


$Q$  is some arbitrary number.

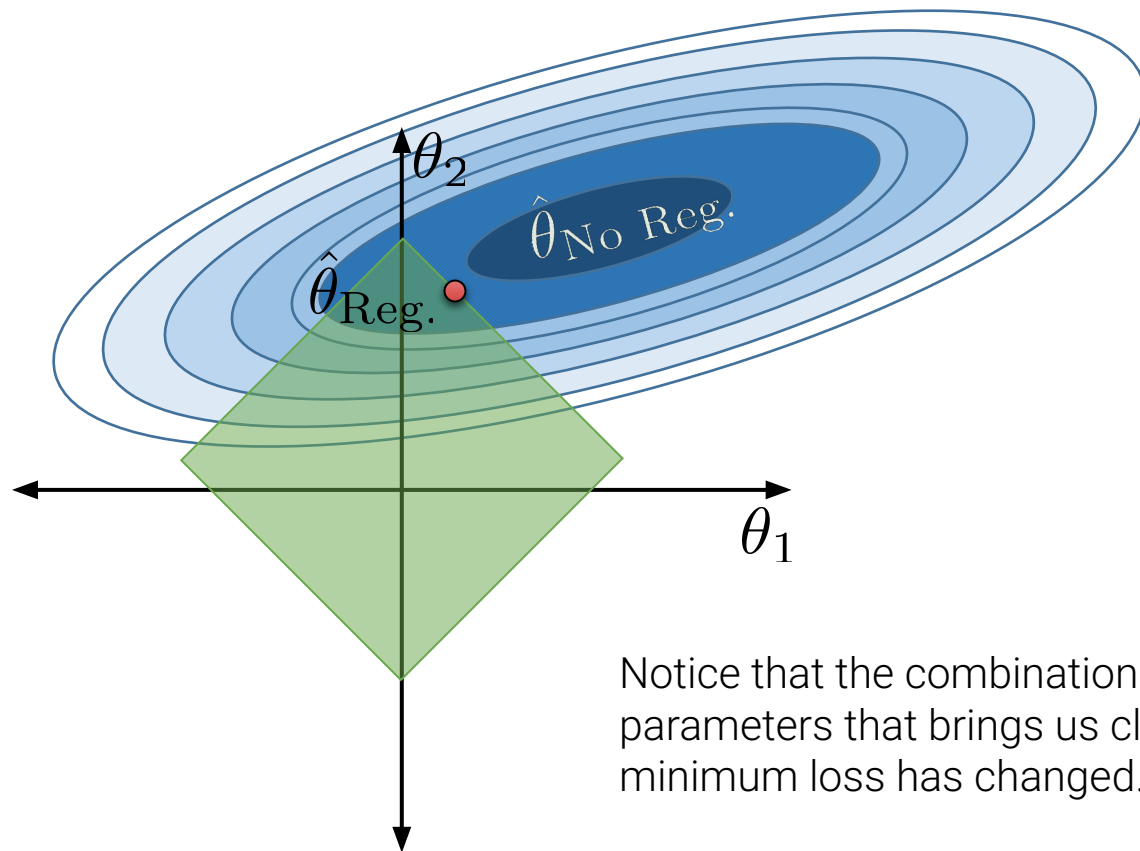
Under this constraint, we choose the parameters that bring us *closest* to the true minimum loss.



If we change the value of  $Q$ , the region of allowed parameter combinations changes.



If we change the value of  $Q$ , the region of allowed parameter combinations changes.





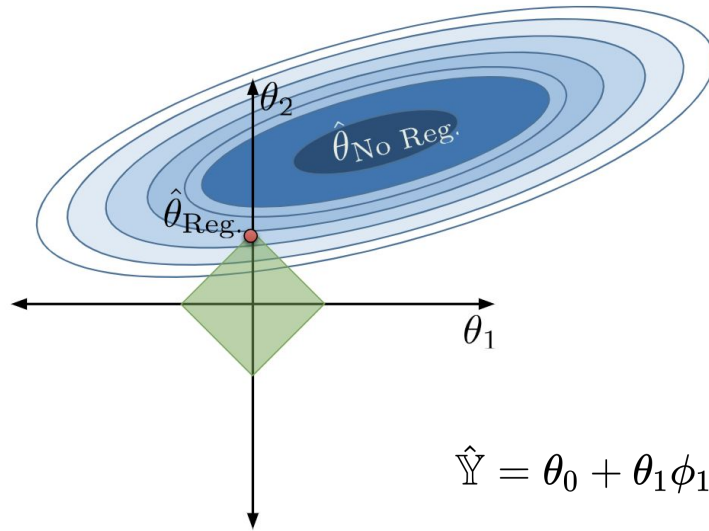
slido



# How does the size of $Q$ relate to model complexity?

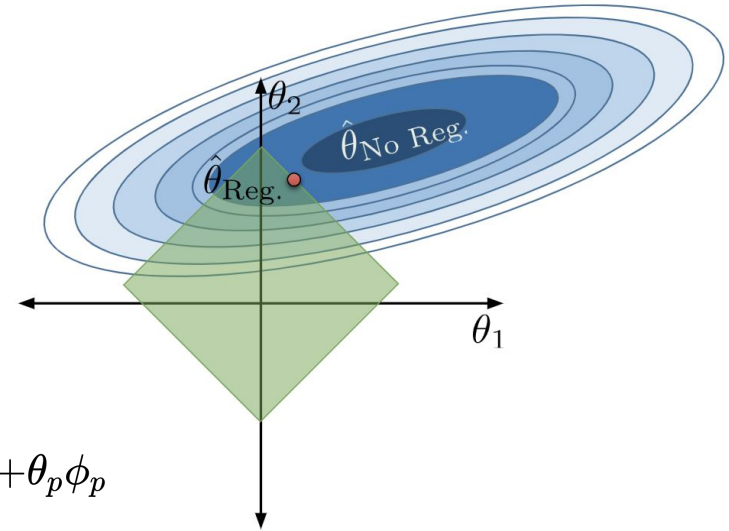
① Click **Present with Slido** or install our [Chrome extension](#) to activate this poll while presenting.

If we change the value of Q, the region of allowed parameter combinations changes.



$$\hat{Y} = \theta_0 + \theta_1\phi_1 + \theta_2\phi_2 \dots + \theta_p\phi_p$$

Small Q:  $\theta_i$  are small in value; feature  $\phi_i$  only contributes a little to the model  $\rightarrow$  model becomes simpler.

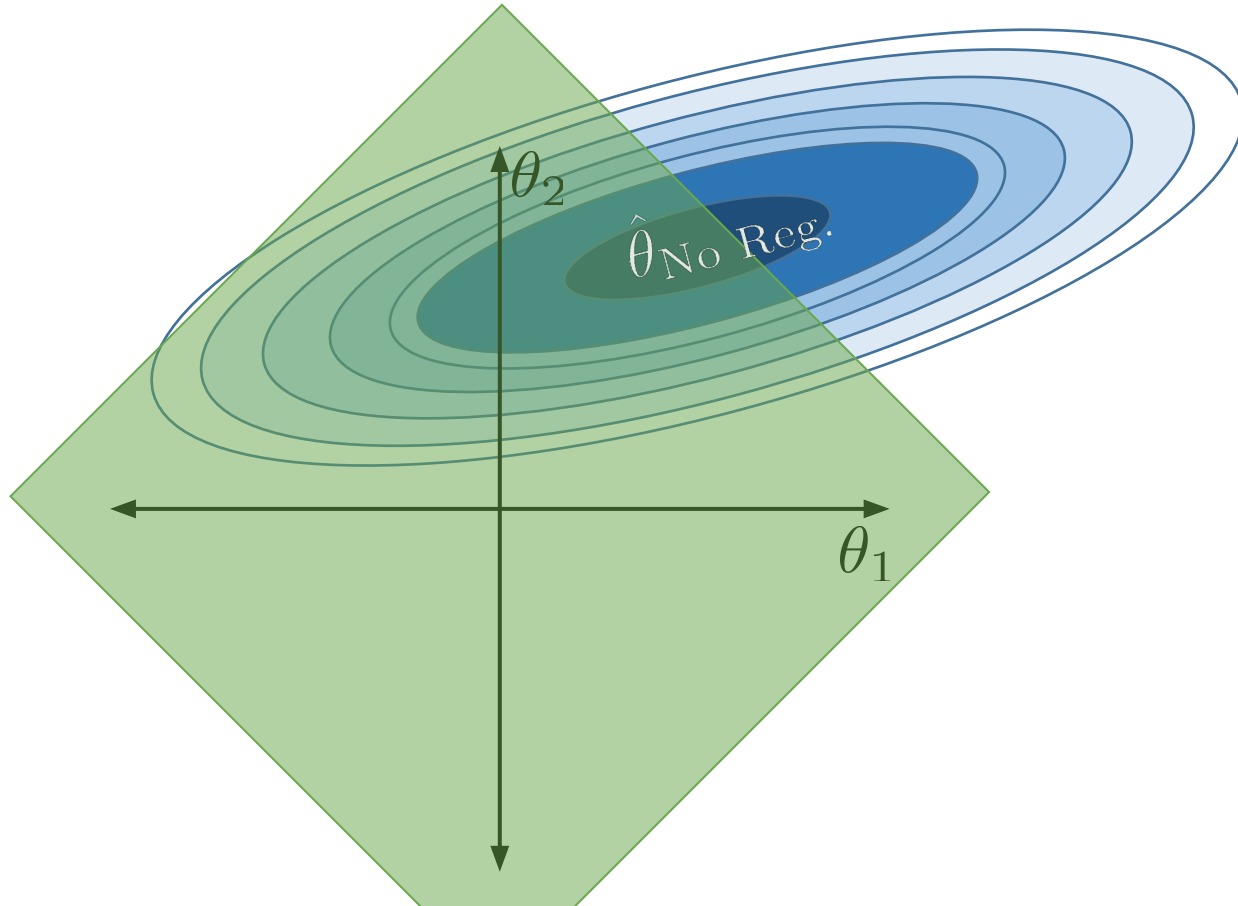


Large Q:  $\theta_i$  are large in value; feature  $\phi_i$  contributes more to the model  $\rightarrow$  model becomes more complex.



4531760

When  $Q$  is very large, our restriction essentially has no effect. The allowed region includes the OLS solution!



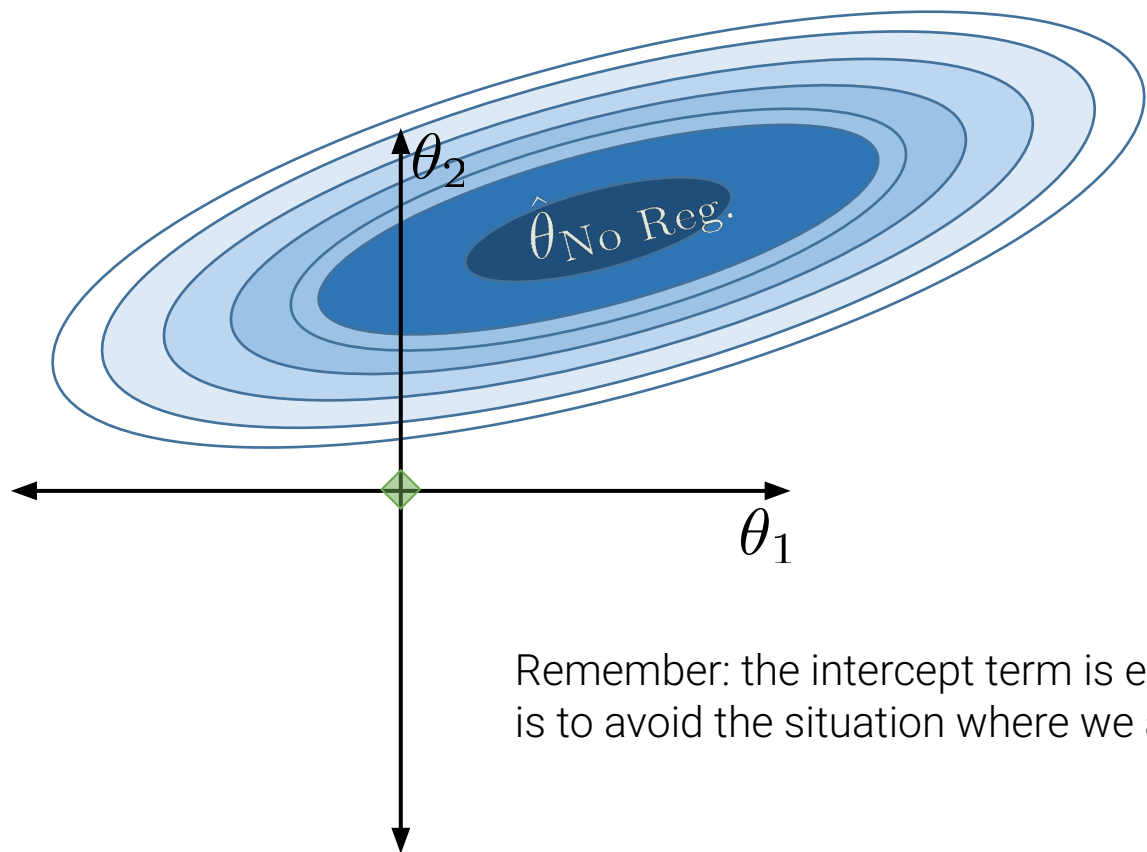
slido



If we pick a very small value of  $Q$ , what kind of model will we have?

① Click **Present with Slido** or install our [Chrome extension](#) to activate this poll while presenting.

When  $Q$  is very small, parameters are set to (essentially) 0.



If the model has no intercept term:

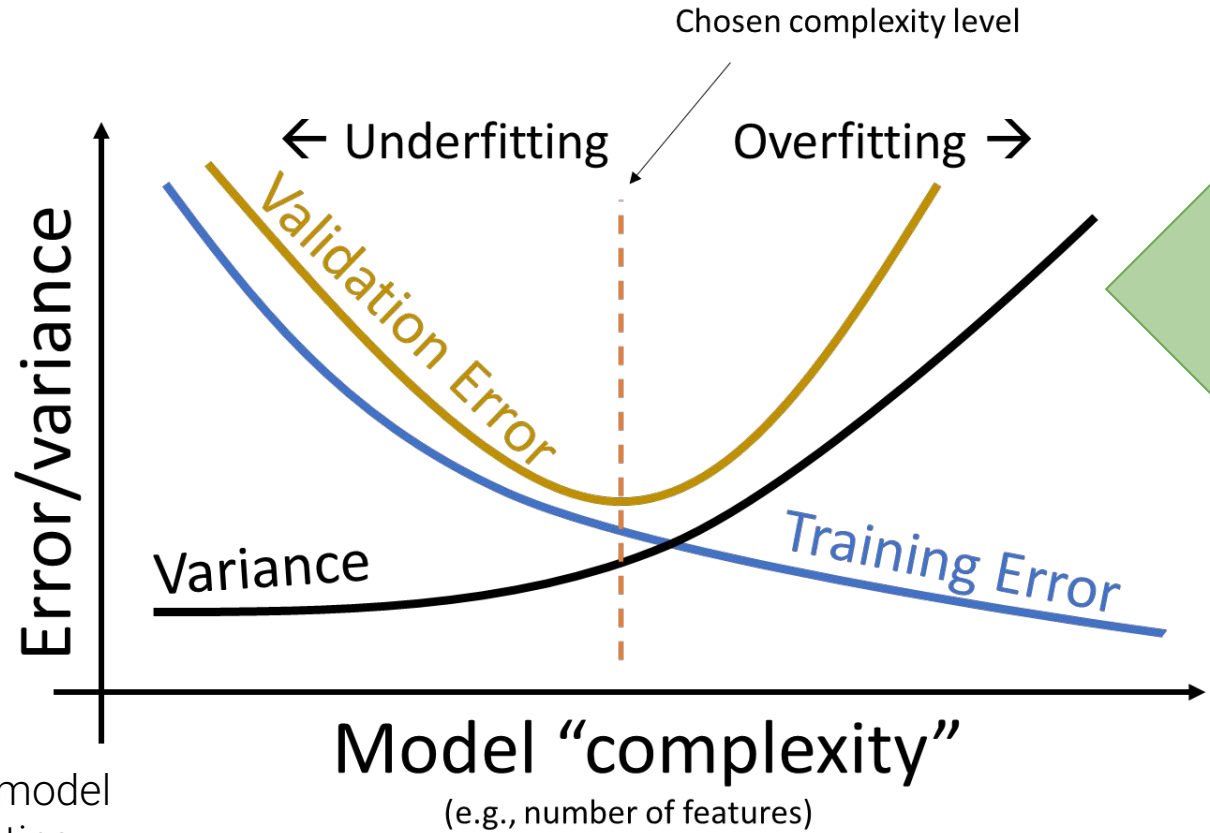
$$\hat{Y} = (0)\phi_1 + (0)\phi_2 + \dots = 0$$

If the model has an intercept term:

$$\hat{Y} = \theta_0 + (0)\phi_1 + (0)\phi_2 + \dots = \theta_0$$

Remember: the intercept term is excluded from the constraint – this is to avoid the situation where we always predict 0.





- Small Q:
- Simple model
  - Underfitting

- Large Q:
- Complex model
  - Overfitting



4531760

# L1 Regularization (LASSO)

---

Lecture 16, Data 100 Spring 2025

## Cross-Validation

- Training, Test, and Validation Sets
- K-Fold Cross-Validation

## Regularization

- Constraining Model Parameters
- **L1 Regularization (LASSO)**
- L2 Regularization (Ridge)



How do we actually apply our constraint  $\sum_{i=1}^p |\theta_i| \leq Q$ ?

Recall our OLS framework: Find thetas that minimize the objective function:

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2$$

In **L1 regularization**: Find thetas that minimize the objective function:

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2 \quad \text{such that} \quad \sum_{i=1}^p |\theta_i| \leq Q$$



By the Lagrangian Duality\*, these two problems are equivalent.

Our original problem: find thetas that minimize the objective function:

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2 \quad \text{such that} \quad \sum_{i=1}^p |\theta_i| \leq Q$$

Equivalent problem: find thetas that minimize the **augmented objective function**:

$$\frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2 + \lambda \sum_{i=1}^p |\theta_i|$$

\*Details out of scope for Data 100; take EECS 127 to learn more

In L1 regularization, we find thetas that minimize our **new objective function**:

$$\underbrace{\frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2}_{\text{Keep MSE on the data low...}} + \underbrace{\lambda \sum_{i=1}^p |\theta_i|}_{\text{...while also keeping the size of parameters small}}$$

$\lambda$  is the **regularization penalty hyperparameter**. When  $\lambda$  is large, our objective function is penalized more for choosing larger thetas  $\rightarrow$  model will adjust by reducing thetas and decreasing complexity.

- In our earlier approach:  $\lambda \approx \frac{1}{Q}$
- How to choose the value for  $\lambda$ ? Cross-validation!

L1 regularization is also called LASSO: "least absolute shrinkage and selection operator".



There is no closed form solution for the optimal LASSO theta. Instead, use `sklearn`.

In `sklearn`, we use the `Lasso` model class.

- Runs gradient descent to **minimize the L1 objective function**.

```
import sklearn.linear_model as lm
lasso_model = lm.Lasso(alpha = 1) # alpha represents the hyperparameter lambda
lasso_model.fit(X_train, Y_train)
lasso_model.coef_
```

```
lasso_model.coef_
```

```
array([-5.14100640e-01,  1.16422594e-03,  2.70209864e-06, -8.05153574e-10,
       -2.78280269e-11, -1.02040718e-13, -5.44295812e-17,  1.83589942e-18])
```



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.

- We often use L1 regularization for **feature selection** – the features with non-zero parameters are more informative for modeling than those with parameters set to zero.
- Intuition: We can get closer to the lowest loss contour at a corner of our constraint diamond.

```
lasso_model_large_lambda.coef_
```

```
array([-0.00000000e+00, -3.37446532e-03,  1.31817186e-05,  1.71062658e-08,  
       -2.44893438e-11, -2.11314339e-13, -5.38994214e-16,  7.05457777e-19])
```

LASSO: “least absolute **shrinkage** and **selection** operator”

 Shrink parameter sizes

 Select important features



Our dataset has features with wildly different numerical scales!

	hp	hp^2	hp^3	hp^4	hp^5	hp^6	hp^7	hp^8
72	150.0	22500.0	3375000.0	5.062500e+08	7.593750e+10	1.139062e+13	1.708594e+15	2.562891e+17
89	150.0	22500.0	3375000.0	5.062500e+08	7.593750e+10	1.139062e+13	1.708594e+15	2.562891e+17
92	158.0	24964.0	3944312.0	6.232013e+08	9.846580e+10	1.555760e+13	2.458100e+15	3.883799e+17
124	180.0	32400.0	5832000.0	1.049760e+09	1.889568e+11	3.401222e+13	6.122200e+15	1.101996e+18
88	137.0	18769.0	2571353.0	3.522754e+08	4.826172e+10	6.611856e+12	9.058243e+14	1.240979e+17
...	...	...	...	...	...	...	...	...
2	150.0	22500.0	3375000.0	5.062500e+08	7.593750e+10	1.139062e+13	1.708594e+15	2.562891e+17
104	167.0	27889.0	4657463.0	7.777963e+08	1.298920e+11	2.169196e+13	3.622558e+15	6.049671e+17
159	148.0	21904.0	3241792.0	4.797852e+08	7.100821e+10	1.050922e+13	1.555364e+15	2.301939e+17
180	115.0	13225.0	1520875.0	1.749006e+08	2.011357e+10	2.313061e+12	2.660020e+14	3.059023e+16
394	52.0	2704.0	140608.0	7.311616e+06	3.802040e+08	1.977061e+10	1.028072e+12	5.345973e+13





4531760

## Coefficients From Earlier

	hp	hp^2	hp^3	hp^4	hp^5	hp^6	hp^7	hp^8
72	150.0	22500.0	3375000.0	5.062500e+08	7.593750e+10	1.139062e+13	1.708594e+15	2.562891e+17
89	150.0	22500.0	3375000.0	5.062500e+08	7.593750e+10	1.139062e+13	1.708594e+15	2.562891e+17
92	158.0	24964.0	3944312.0	6.232013e+08	9.846580e+10	1.555760e+13	2.458100e+15	3.883799e+17
124	180.0	32400.0	5832000.0	1.049760e+09	1.889568e+11	3.401222e+13	6.122200e+15	1.101996e+18
88	137.0	18769.0	2571353.0	3.522754e+08	4.826172e+10	6.611856e+12	9.058243e+14	1.240979e+17
...	...	...	...	...	...	...	...	...
2	150.0	22500.0	3375000.0	5.062500e+08	7.593750e+10	1.139062e+13	1.708594e+15	2.562891e+17
104	167.0	27889.0	4657463.0	7.777963e+08	1.298920e+11	2.169196e+13	3.622558e+15	6.049671e+17
159	148.0	21904.0	3241792.0	4.797852e+08	7.100821e+10	1.050922e+13	1.555364e+15	2.301939e+17
180	115.0	13225.0	1520875.0	1.749006e+08	2.011357e+10	2.313061e+12	2.660020e+14	3.059023e+16
394	52.0	2704.0	140608.0	7.311616e+06	3.802040e+08	1.977061e+10	1.028072e+12	5.345973e+13

lasso\_model.coef\_

```
array([-5.14100640e-01,  1.16422594e-03,  2.70209864e-06, -8.05153574e-10,  
       -2.78280269e-11, -1.02040718e-13, -5.44295812e-17,  1.83589942e-18])
```



Our model parameter for a feature with small numeric values (hp) is much, much larger than the parameter for a feature with large numeric values (hp^8).

- The feature with larger values will naturally contribute more to the predicted  $\hat{y}$  for each observation.
- The LASSO model needs to “spend” more of its parameter budget to allow hp to have much of an impact on each prediction.

First datapoint:  $\hat{y}_i = \theta_0 + \theta_1(150) + \dots + \theta_8(2.56 \times 10^{17})$

The large values of hp^8 dominate the prediction.

$$\hat{y}_i = \theta_0 - 0.51(150) + \dots + 1.84 \times 10^{-18}(2.56 \times 10^{17})$$

The parameter for hp must be very large for hp to influence the prediction.



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 features will be all on the same scale with mean 0 and SD 1.

```
lasso_model_scaled = lm.Lasso(alpha=1)
lasso_model_scaled.fit(X_train_standardized, Y_train)
lasso_model_scaled.coef_

array([ -9.31789105,  0.          ,  0.          ,  2.89288682,  0.65909948,
         0.          ,  0.          ,  0.          ])
```



4531760

# L2 Regularization (Ridge)

---

Lecture 16, Data 100 Spring 2025

## Cross-Validation

- Training, Test, and Validation Sets
- K-Fold Cross-Validation

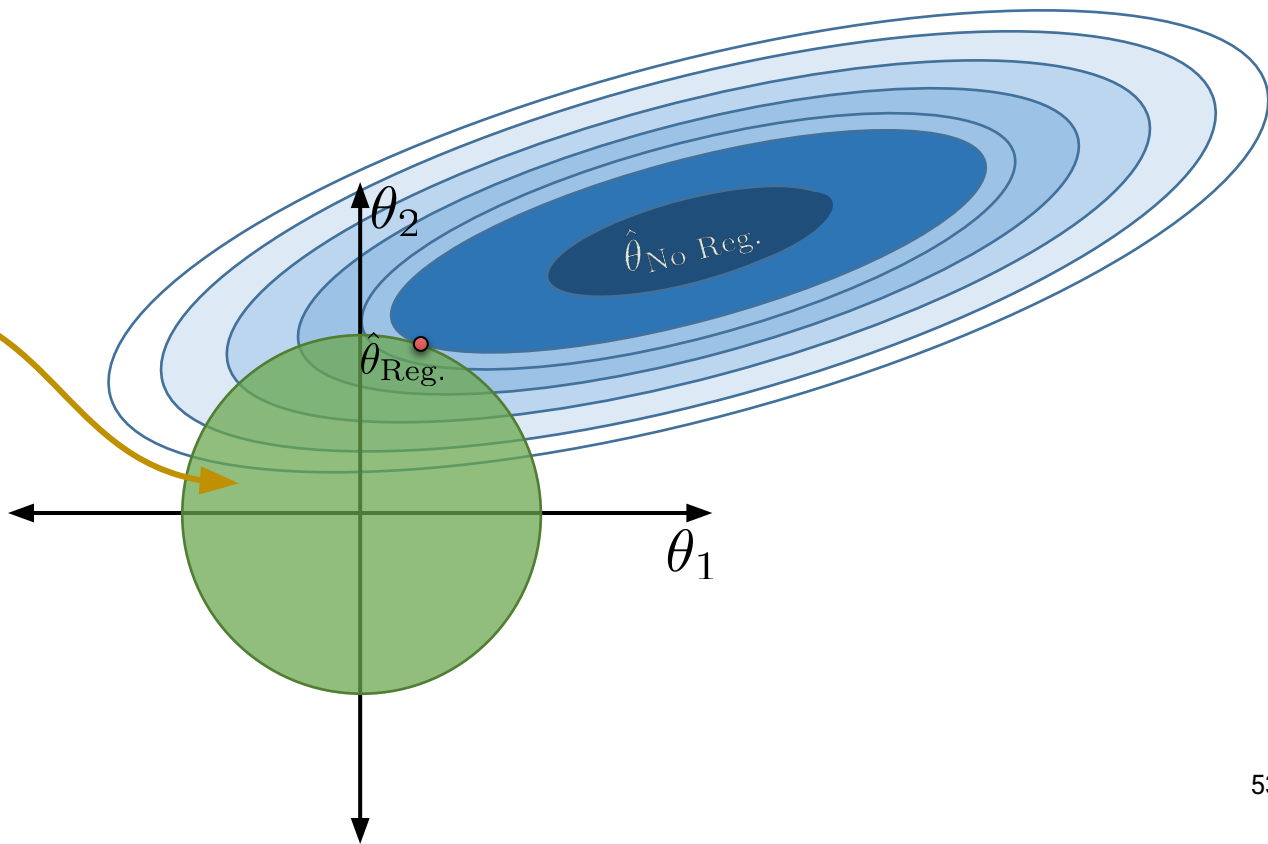
## Regularization

- Constraining Model Parameters
- L1 Regularization (LASSO)
- **L2 Regularization (Ridge)**



We could have applied a different constraint to our parameters: the sum of their *squares* must be less than some number  $Q$ .

$$\sum_{i=1}^p \theta_i^2 \leq Q$$





As with L1 regularization, we can express this constraint in two forms:

Original formulation:

$$\frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2 \quad \text{such that} \quad \sum_{i=1}^p \theta_i^2 \leq Q$$

L2 objective function:

$$\frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2 + \lambda \sum_{i=1}^p \theta_i^2$$



In L2 regularization, we find thetas that minimize our **new objective function**:

$$\underbrace{\frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \phi_{i,1} + \dots + \theta_p \phi_{i,p}))^2}_{\text{Keep MSE on the data low...}} + \underbrace{\lambda \sum_{i=1}^p \theta_i^2}_{\text{...while also keeping the size of parameters small}}$$

L2 regularization is commonly called **ridge regression**.



In `sklearn`, we use the `Ridge` model class.

- Runs gradient descent to **minimize the L2 objective function**

```
import sklearn.linear_model as lm
ridge_model = lm.Ridge(alpha = 1) # alpha represents the hyperparameter lambda
ridge_model.fit(X_train, Y_train)
ridge_model.coef_
```

```
ridge_model.coef_
```

```
array([-16.85961652,  3.26398097,  9.1167183 ,  4.53790201,
        -2.32110639, -5.6066523 , -3.15831859,  4.75104822])
```





Applying vector calculus (out of scope) allows us to find a closed-form solution for L2 regularization!

- Recall that L1 regularization has no closed-form solution:

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

This solution exists **even if**  $\mathbb{X}$  is not full rank – an important reason why we often prefer L2 regularization. This will be important once we discuss multicollinearity next week.



Our regression models are summarized below.

- The objective function is what the gradient descent optimizer minimizes.

Name	Model	Loss	Reg.	Objective Function	Solution
OLS	$\hat{\mathbf{Y}} = \mathbb{X}\boldsymbol{\theta}$	Squared loss	None	$\frac{1}{n} \ \mathbf{Y} - \mathbb{X}\boldsymbol{\theta}\ _2^2$	$\hat{\boldsymbol{\theta}}_{\text{OLS}} = (\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T \mathbf{Y}$ If $\mathbb{X}$ is full-column rank
Ridge Regression	$\hat{\mathbf{Y}} = \mathbb{X}\boldsymbol{\theta}$	Squared loss	L2	$\frac{1}{n} \ \mathbf{Y} - \mathbb{X}\boldsymbol{\theta}\ _2^2 + \lambda \sum_{i=1}^p \theta_i^2$	$\hat{\boldsymbol{\theta}}_{\text{ridge}} = (\mathbb{X}^T \mathbb{X} + n\lambda \mathbf{I})^{-1} \mathbb{X}^T \mathbf{Y}$
LASSO	$\hat{\mathbf{Y}} = \mathbb{X}\boldsymbol{\theta}$	Squared loss	L1	$\frac{1}{n} \ \mathbf{Y} - \mathbb{X}\boldsymbol{\theta}\ _2^2 + \lambda \sum_{i=1}^p  \theta_i $	No closed form



4531760

## Lecture 16

# Cross Validation, Regularization

Content credit: [Acknowledgments](#)