LECTURE 30

Regularization

Methods for ensuring the generalizability of our models to unseen data.

.

CSCI 3022, Fall 2023

Maribeth Oscamou

Content credit: Acknowledgments

Course Logistics: 13th and 14th Weeks At A Glance

Mon 4/15	Tues 4/16	Wed 4/17	Thurs 4/18	Fri 4/19
Attend & participate in class	TA NB Discussion 5pm-6pm via Zoom **Project Part 2 Released**	Attend & participate in class	**Project Part 1 Due: 11:59pm MT** **No Late Submissions Accepted**	Attend & participate in class Quiz 7: Scope: L24-L26, HW 10, TA Discussion NB 12
Mon 4/22	Tues 4/23	Wed 4/24	Thurs 4/25	Fri 4/26
Attend & participate in class	TA NB Discussion 5pm-6pm via Zoom	Attend & participate in class	**Project Part 2 Due: 11:59pm MT** **No Late Submissions Accepted**	Attend & participate in class Quiz 8: Scope: L26-L29, HW 10, TA Discussion NB 12

Project Part 2: Extra Credit Opportunity!

Each part of the project is 50 points each (100 pts total).
The total project grade (out of 100) is 5% of your overall class grade.

- The last question in Project Part 2 is extra credit.
- You can earn up to 20 points of extra credit on that problem (rubric below)
- Thus you can earn up to 120/100 points for your project grade!

Extra Credit: How Low Can You Go? Create Your Own Model and Check RMSE on the Test Data

For **extra credit**, you can create your own model to try to improve the RMSE and residual plots even further.

The tables below provide scoring guidelines for the extra credit opportunity in this problem. If your RMSE lies in a particular range, you will receive the number of points associated with that range.

Extra Credit Grading Scheme

Important: while your Validation RMSE can be checked at any time in this notebook, your Test RMSE can only be checked once by submitting your model's predictions to Gradescope. The thresholds are as follows:

Extra Credit Points	**+10**	**+8**	**+6**	**+4**	**+ 2**
Validation RMSE	Less than 200k	[200k, 210k)	[210k, 220k)	[220k, 230k)	[230k, 235k)

Extra Credit Points	**+10**	**+8**	**+6**	**+4**	**+ 2**
Test RMSE	Less than 200k	[200k, 210k)	[210k, 220k)	[220k, 230k)	[230k, 235k)

To receive these points, you need to show your work in the cells below AND complete the EXPLANATION STEP at the end (explaining what you did to create your model).

Project Part 2: Extra Credit Opportunity!

To receive credit on the extra credit you must upload the following 3 files by the Project 2 Deadline:

a). Your test predictions (in a .csv) to the Gradescope assignment Project 2 Extra Credit Test Predictions: Submit .csv file here

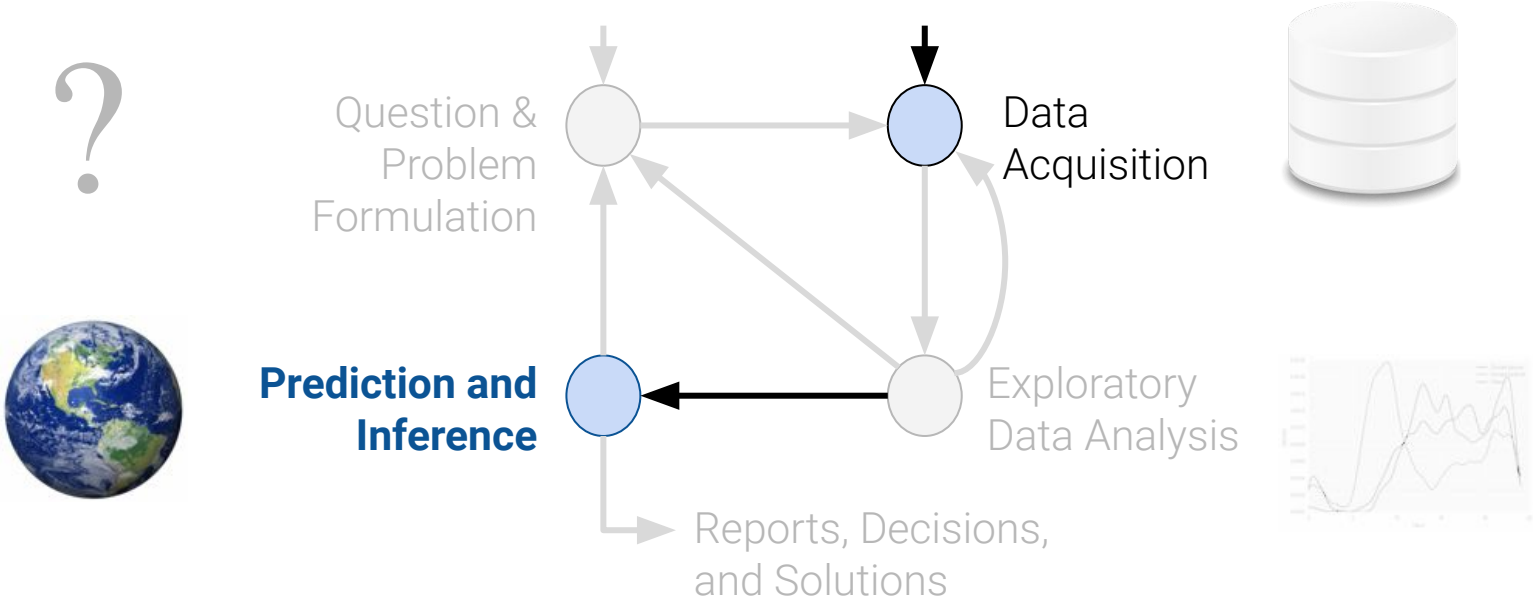
b). The PDF of your Project 2 assignment (with your answers to the Extra Credit filled out in the last section) to the Gradescope assignment Project Part 2 Manually Graded: Upload PDF here

c). The zipfile of your autograded parts of the assignment to the Gradescope assignment "Project Part 2 Autograded: Upload Zipfile here

Feel free to visit office hours and/or post on Piazza with any questions!

To receive extra credit on Project 2, when testing your model on the validation set, ***the only rows you can remove from the Validation dataset*** are the ***rows with Pure Market Filter = 0***. And recall you cannot remove any rows from the test dataset.

Plan for this week: Model Selection



(today)

Model Selection Basics:
Cross Validation

Model Selection Basics:
Regularization

Today's Roadmap

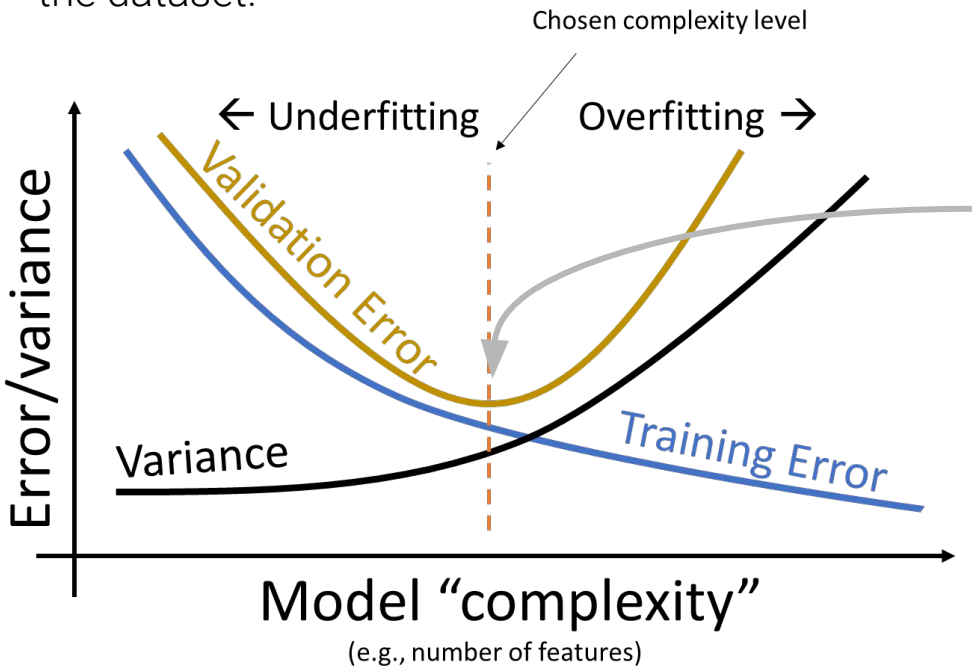
Regularization

- Constraining Model Parameters
- L1 Regularization (LASSO)
- Standard Units

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?

Hyperparameter: Terminology

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

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

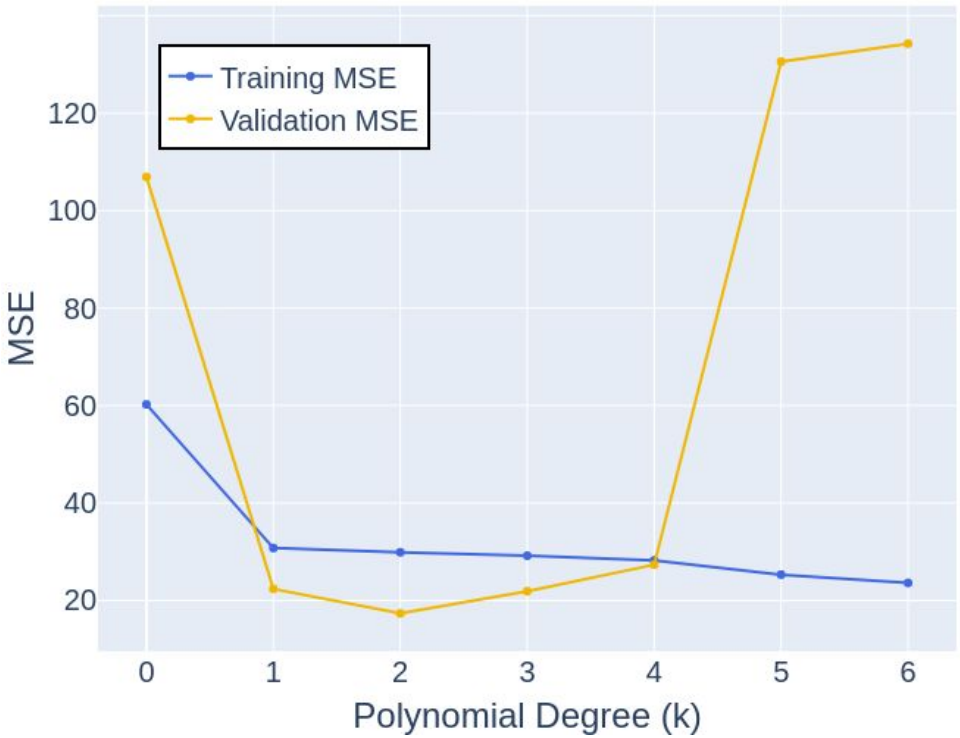
- Cannot solve for hyperparameters via calculus/numerical optimization - instead we must choose it ourselves.
- Examples
  - Last time built a series of models each with increasing orders of horsepower. In this case, the hyperparameter is the degree or k that controlled the order of our polynomial.
  - Regularization penalty,(this time)

We use:

- The **validation set** (a.k.a. development set) (a.k.a. cross validation set) to **select hyperparameters**, or more generally, between different competing models.
- The training and validation data all together to select the parameters (i.e. the thetas) once we have selected the best model.

Recap: Validation

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

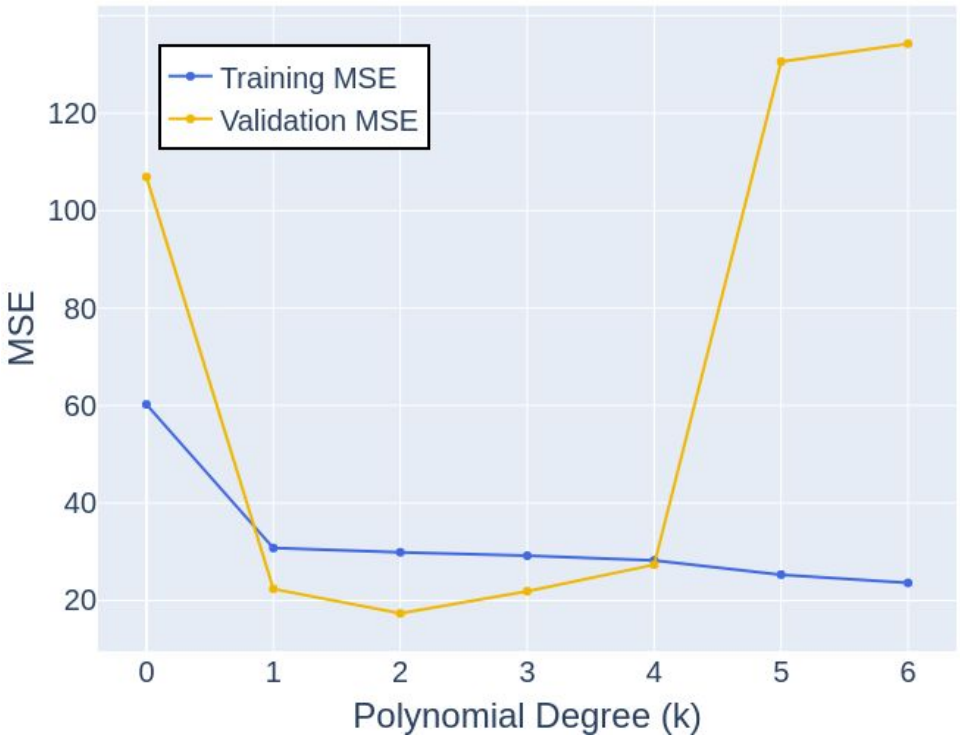


k	Training MSE	Validation MSE
0	60.235744	106.925296
1	30.756678	22.363676
2	29.875269	17.331880
3	29.180868	21.889257
4	28.214850	27.340989
5	25.290990	130.597678
6	23.679651	135.787493

Recap: Selecting Hyperparameters

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	130.597678
6	23.679651	135.787493

A More Complex Example

Suppose we have a dataset with 9 features.

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

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:

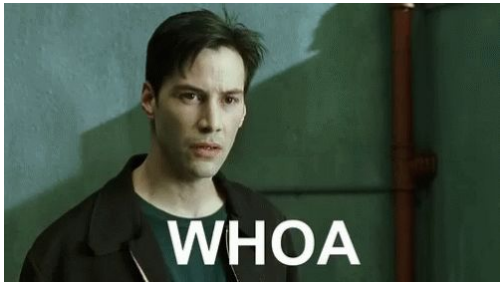
- For each of the 2^9 linear regression models, compute the **validation MSE**.
- Pick the model that has the lowest **validation MSE**.

PROBLEM! 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

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



	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

Constraining Model Parameters

Regularization

- **Constraining Model Parameters**
- L1 Regularization (LASSO)
- Standard Units

Restricting Model Complexity

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

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

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

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

Restricting Model Complexity

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

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

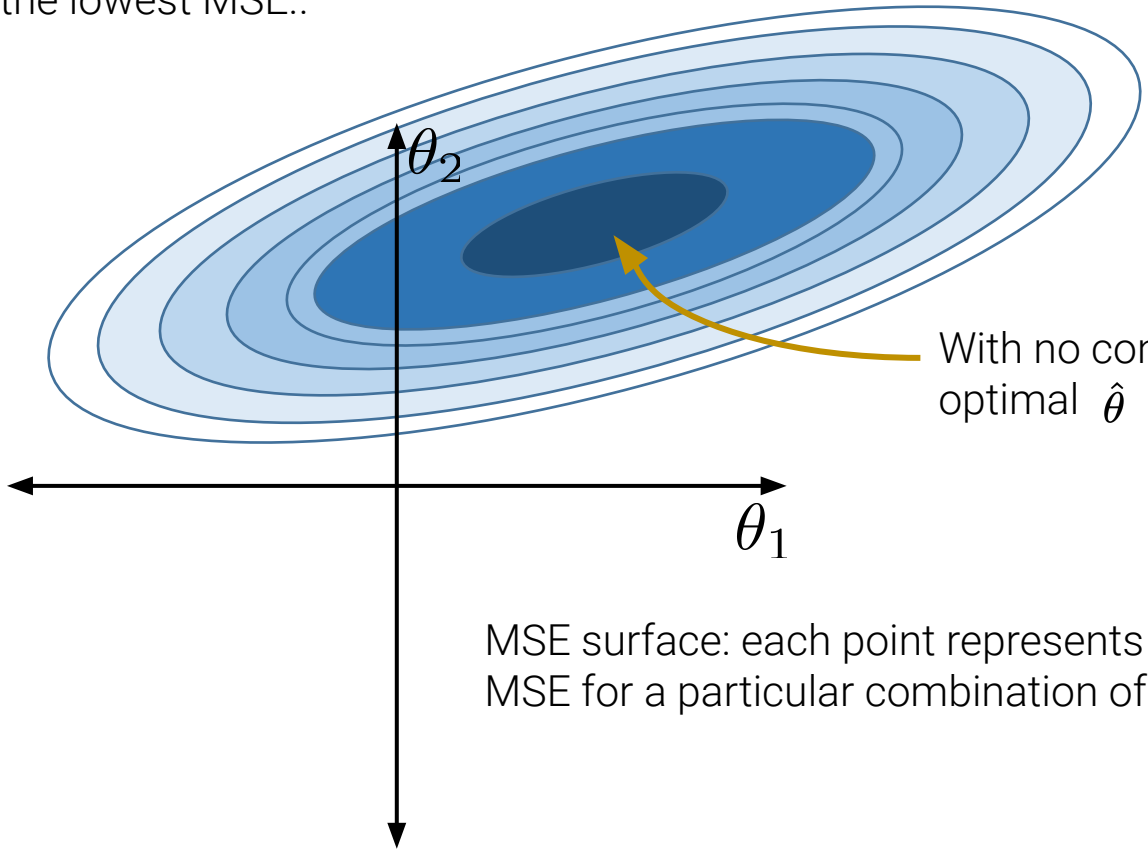
$$\sum_{i=1}^{p} |\theta_i| \leq Q$$

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

Note that the intercept term, θ_0, is typically excluded from this constraint.

Visualizing

Visualize the MSE surface as a contour map. Our goal is to find the combination of parameters that gives the lowest MSE..

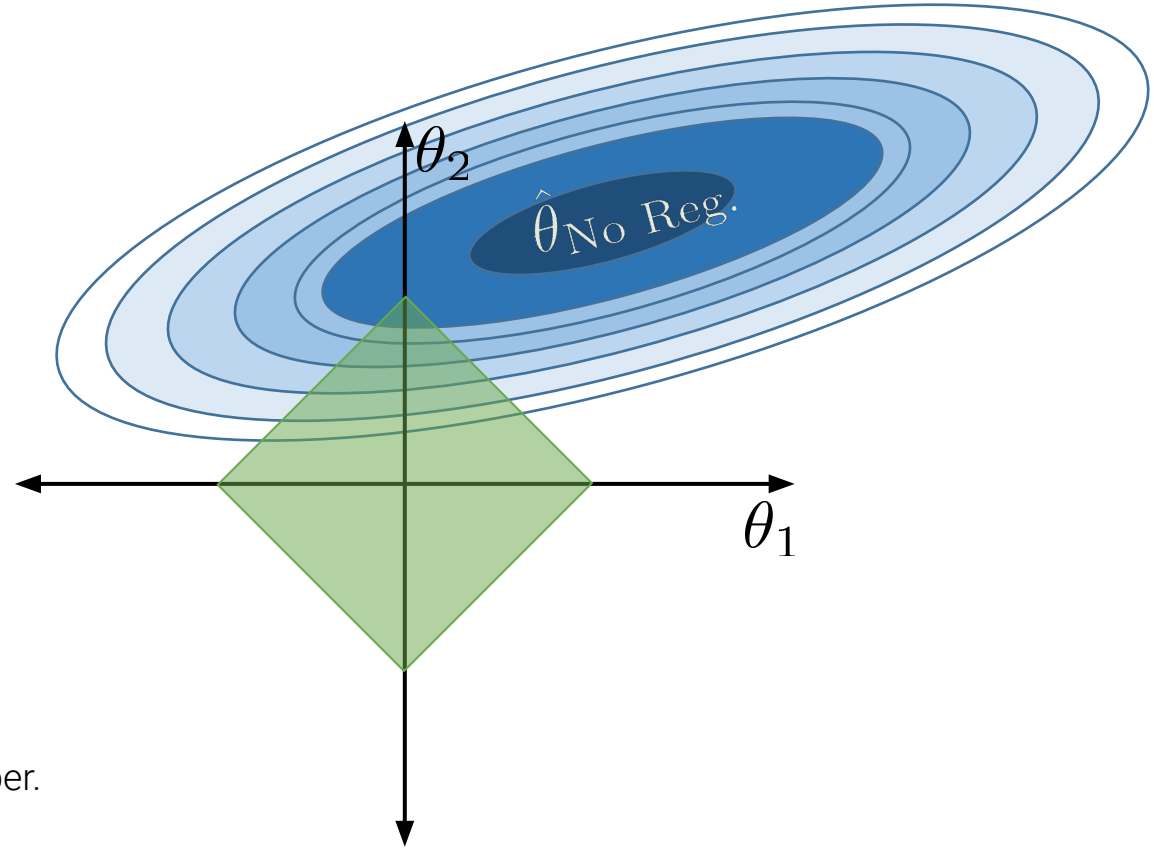


With no constraint, the optimal $\hat{\theta}$ is here.

MSE surface: each point represents the model's MSE for a particular combination of θ_1, θ_2 .

Constraining Model Parameters

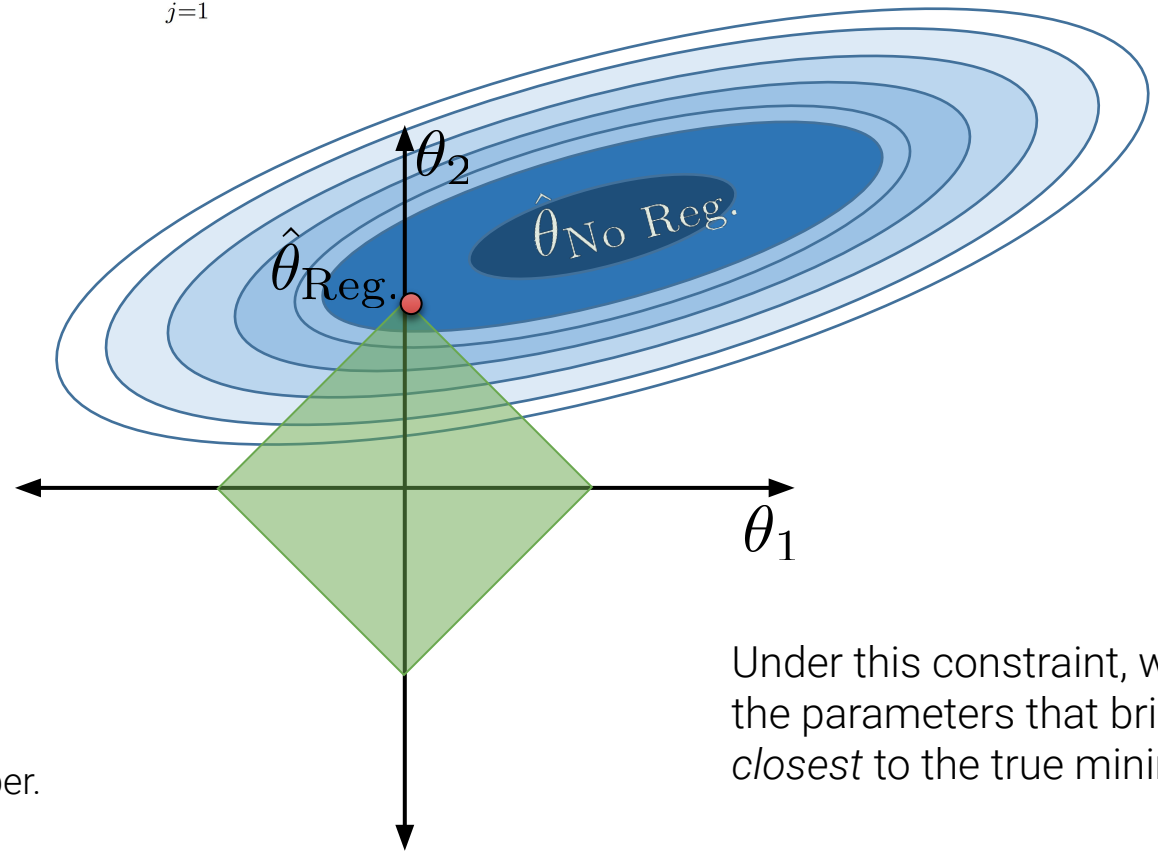
When we apply the constraint $\sum_{i=1}^{p} |\theta_i| \leq Q$, only parameter combinations inside the diamond are valid.



Q is some arbitrary number.

Constraining Model Parameters

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

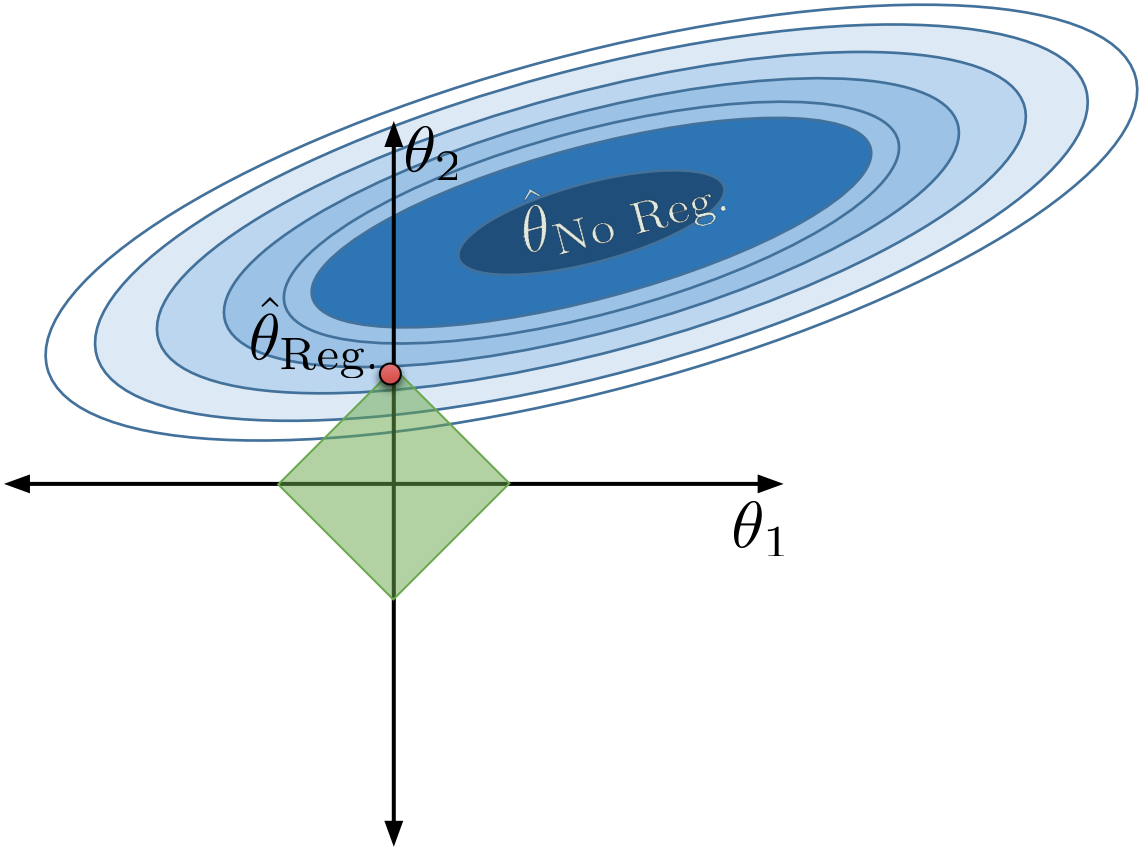


Under this constraint, we choose the parameters that bring us *closest* to the true minimum loss.

Q is some arbitrary number.

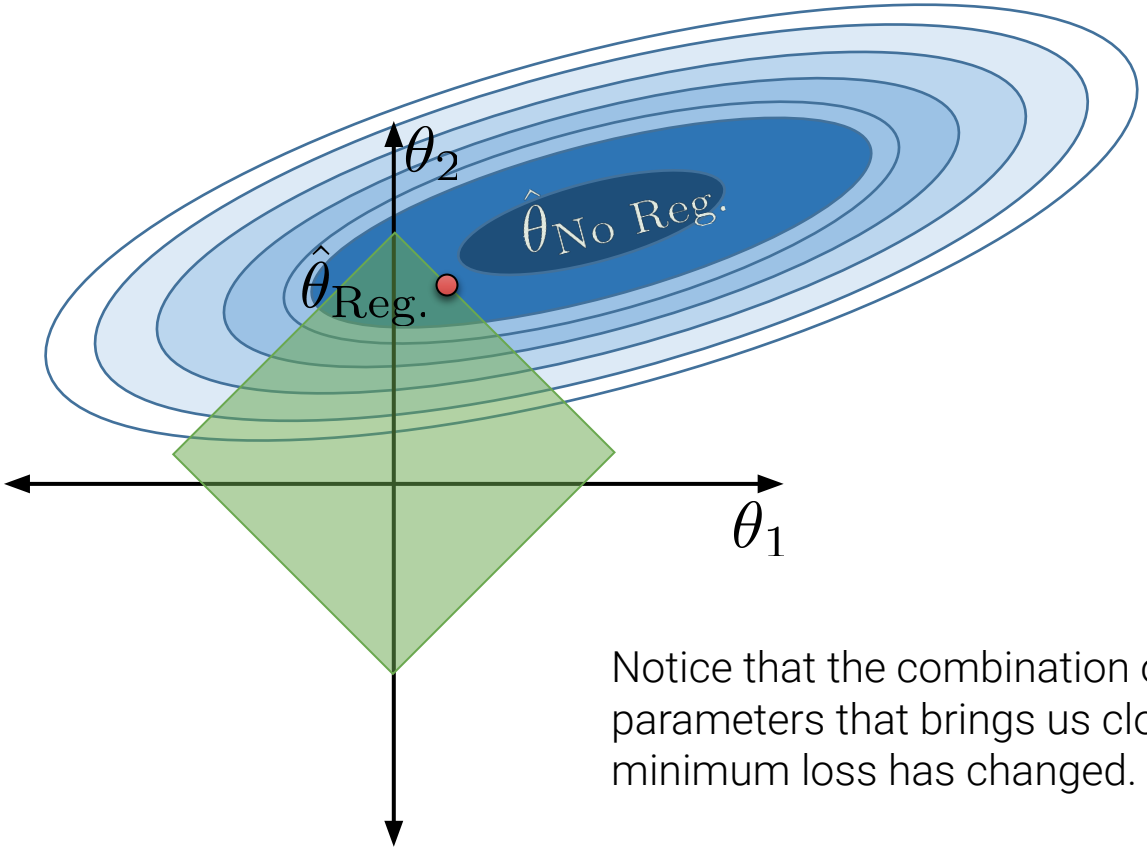
Smaller Q

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



Larger Q

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



Notice that the combination of model parameters that brings us closest to minimum loss has changed.

How does the size of Q relate to model complexity?

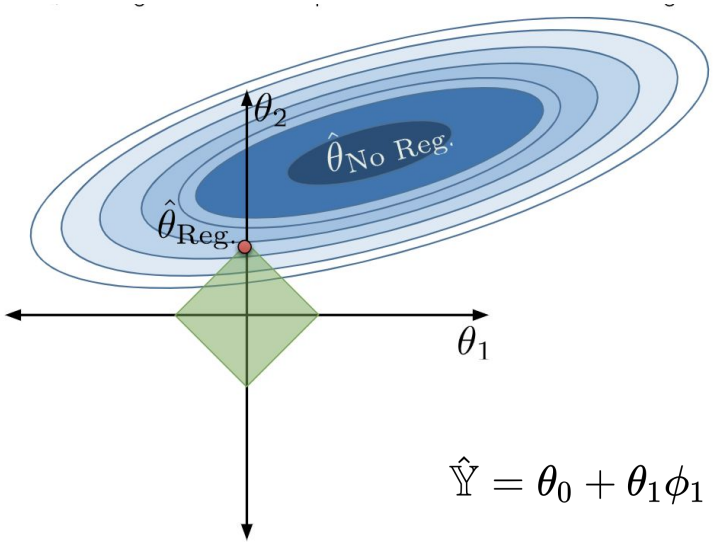
A). Small Q implies use more predictors in the model.

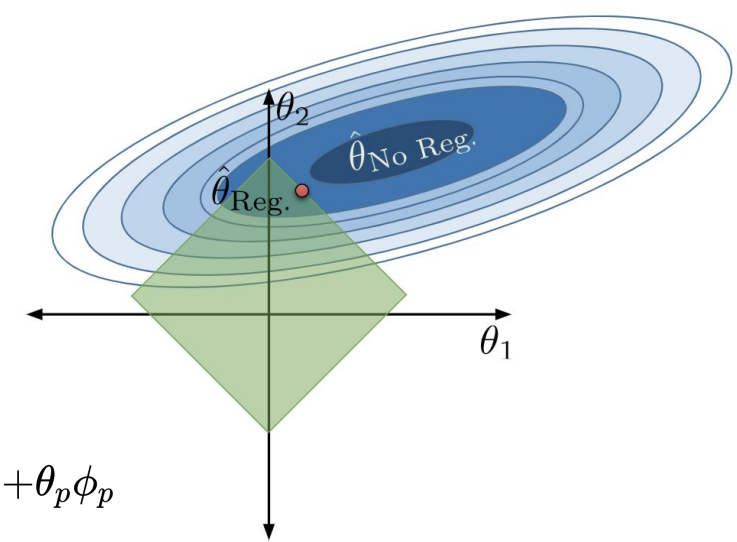
B). Small Q implies use less predictors in the model.

ⓘ

Size of Q

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





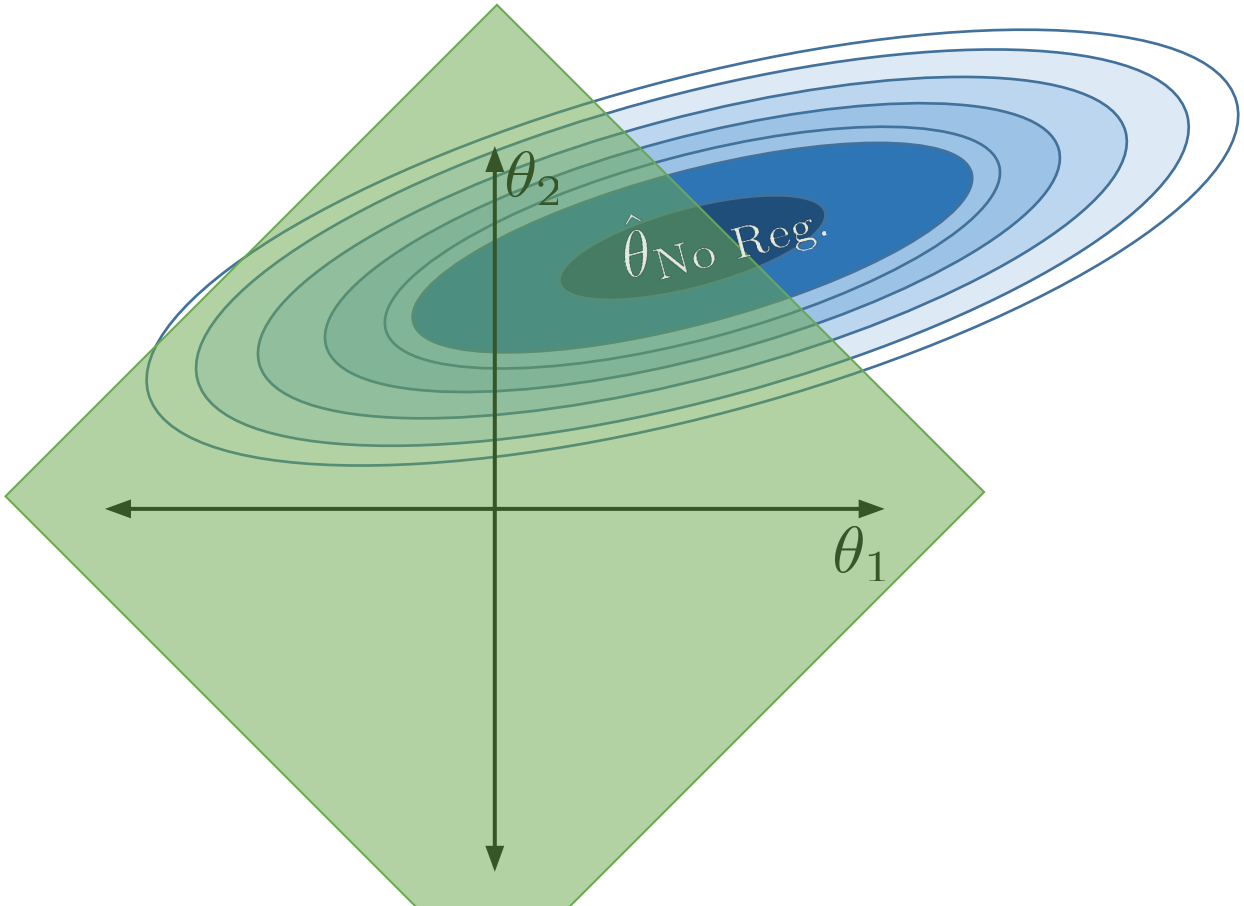
$$\hat{\mathbb{Y}} = \theta_0 + \theta_1\phi_1 + \theta_2\phi_2 \ldots + \theta_p\phi_p$$

Small Q: θ_i are small in value; feature ϕ_i only contributes a little to the model → model becomes simpler.

Large Q: θ_i are large in value; feature ϕ_i contributes more to the model → model becomes more complex.

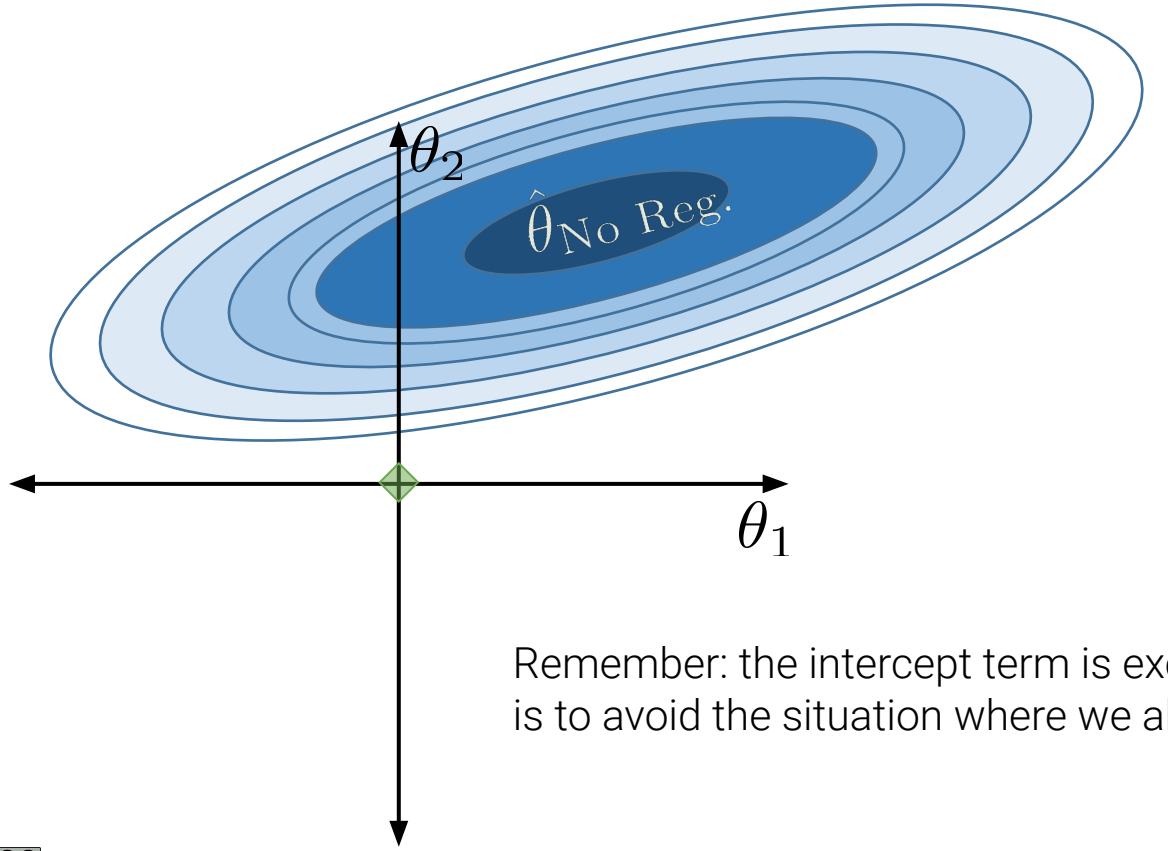
Size of Q

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



Size of Q

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



If the model has no intercept term:

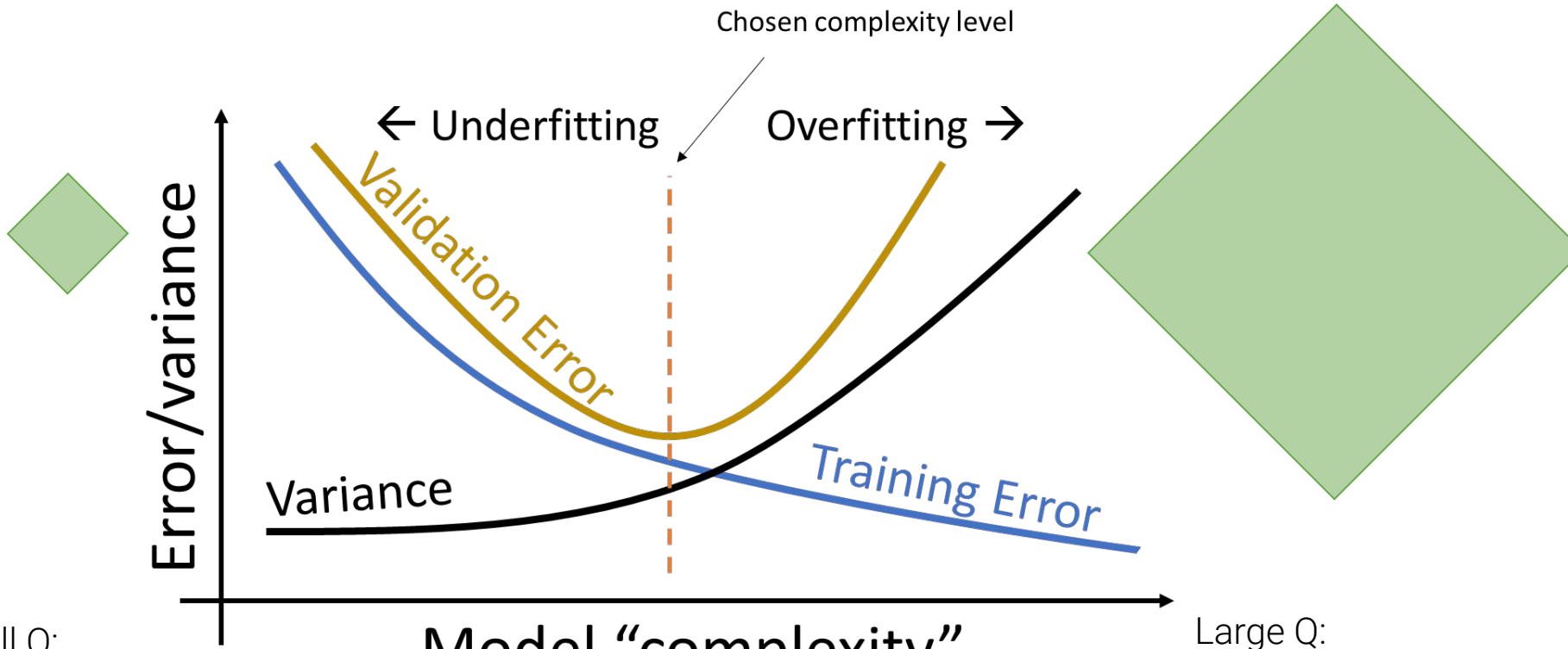
$$\hat{\mathbb{Y}} = (0)\phi_1 + (0)\phi_2 + \ldots = 0$$

If the model has an intercept term:

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

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

Complexity and Q



Small Q:

- Simple model
- Underfitting

Large Q:

- Complex model
- Overfitting

L1 Regularization (LASSO)

Regularization

- Constraining Model Parameters
- **L1 Regularization (LASSO)**
- Standard Units

L1 Regularization

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:

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

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

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

L1 Regularization

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

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

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

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

*Details out of scope for Data CSCI 3022 (By the Lagrangian Duality, these two problems are equivalent)

L1 Regularization

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

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

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

- In our earlier approach: $\boldsymbol{\alpha} \approx \frac{1}{Q}$
- How to choose the value for $\boldsymbol{\alpha}$?

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

How to Choose alpha?

Option 1: If you have domain knowledge with a specific number of predictors you'd like to keep in your model (i.e. you want a model with only 4 predictors), you can guess/check alpha values until you get to one large enough that only 4 predictors remain).

Option 2: Use cross validation to find alpha!

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

L1 Regularized OLS in `sklearn`

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

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

- Runs numerical optimization to **minimize the L1 objective function.**

```
import sklearn.linear_model as lm
lasso_model = lm.Lasso(alpha = 1)
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])
```

LASSO and Feature Selection

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

One Issue With Our Approach

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

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])
```

Pitfalls of Unscaled Data

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 y_hat 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: $$\hat{y}_i = \theta_0 + \theta_1(150) + \ldots + \theta_8\left(2.56 \times 10^{17}\right)$$

The large values of hp^8 dominate the prediction.

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

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

Making Things Fair

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

- One approach: Standardize the data, i.e., replace everything with standard units:

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

- 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.        ])
```

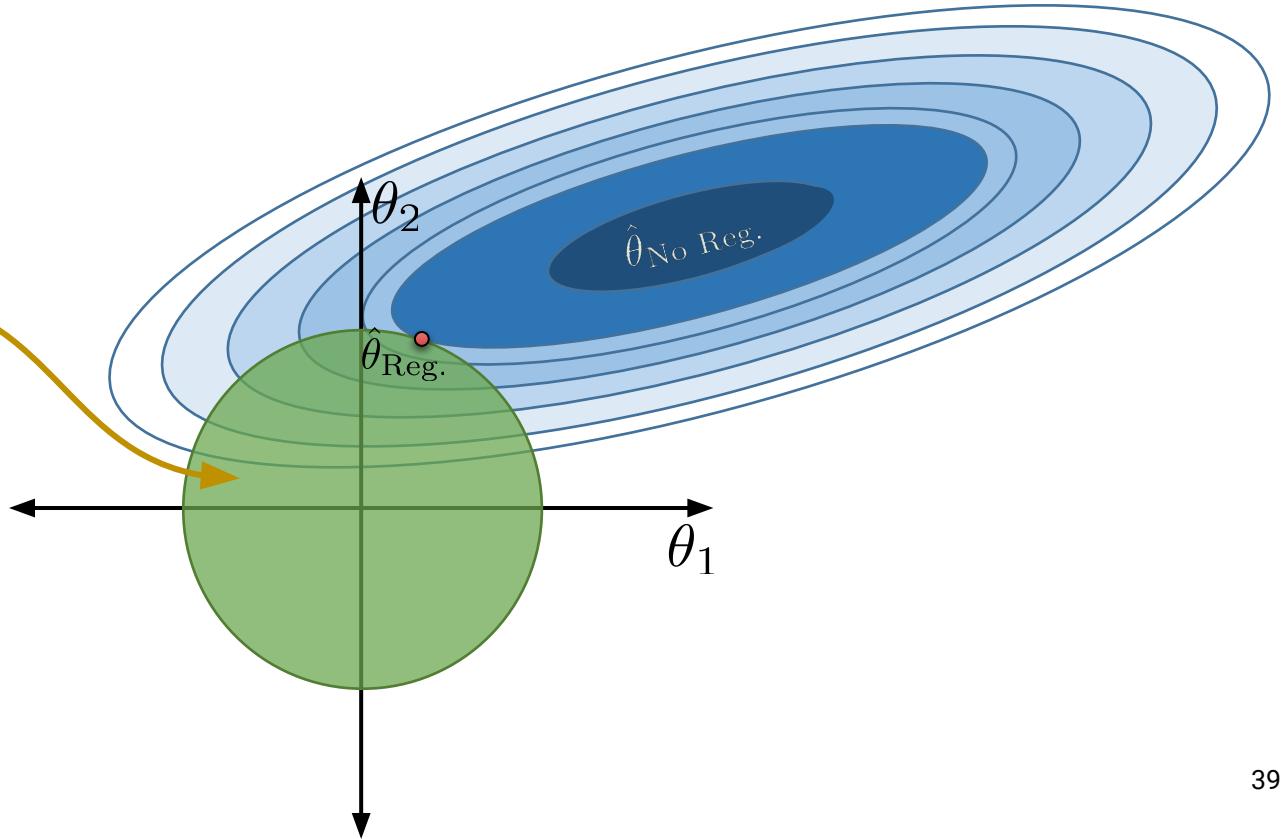
Appendix: Other types of Regularization

- **L2 Regularization (Ridge)**

Changing The Constraint

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

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



L2 Regularization

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

Original formulation:

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

L2 objective function:

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

L2 Regularization

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

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

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

L2 Regularized OLS in `sklearn`

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])
```

Closed Form Solution for L2 Regularization

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 multicolinearity next week.

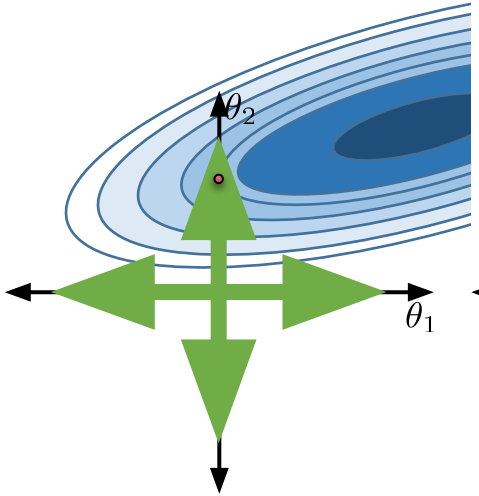
Summary of Regression Methods

Our regression models are summarized below.

- The objective function is what the built-in Sklearn optimizer minimizes.

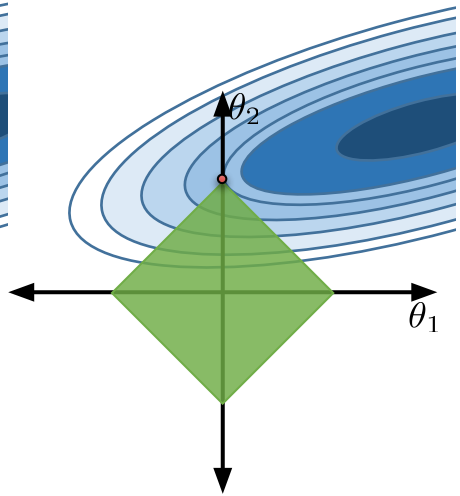
Name	Model	Loss	Reg.	Objective Function	Solution
OLS	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	Squared loss	None	$\frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2$	$\hat{\theta}_{\text{OLS}} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$ If $\mathbb{X}$ is full-column rank
Ridge Regression	$\hat{\mathbb{Y}}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

L^0 Norm Ball



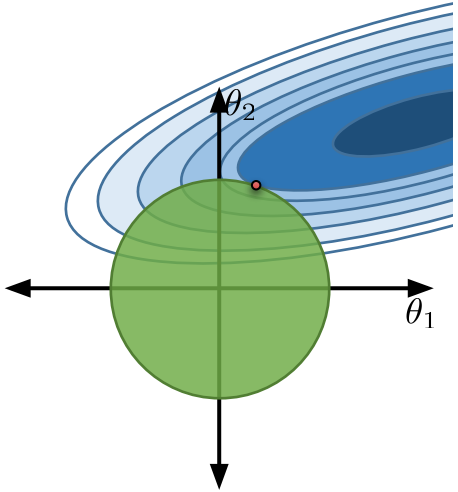
Ideal for
Feature Selection
but combinatorically difficult to optimize

L^1 Norm Ball



Encourages
Sparse Solutions
Convex!

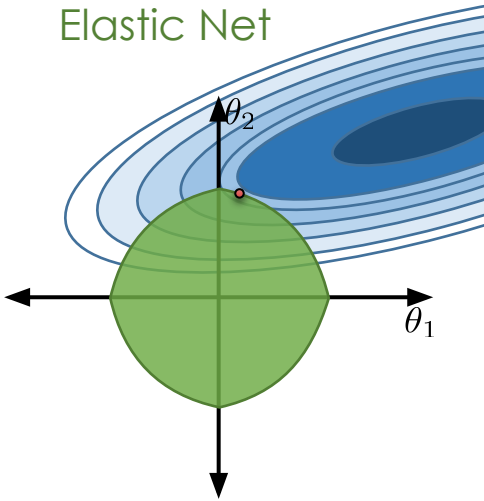
L^2 Norm Ball



Spreads weight
over features **(robust)**
does not
encourage sparsity

$L^1 + L^2$ Norm

Elastic Net



Compromise
Need to tune
two regularization
parameters