LECTURE 29

`sklearn` and Multiple Linear Regression

Building models in code.

CSCI 3022, Fall 2023

Maribeth Oscamou

Content credit: Acknowledgments

Announcements

HW 11 (last HW!) due Thursday - *Make sure you are using v2*

No Quiz this week (last Quiz will be next Friday, Dec 8th)

Project Part 1 due next Thursday at 11:59pm MT (plan accordingly - no late submissions accepted)

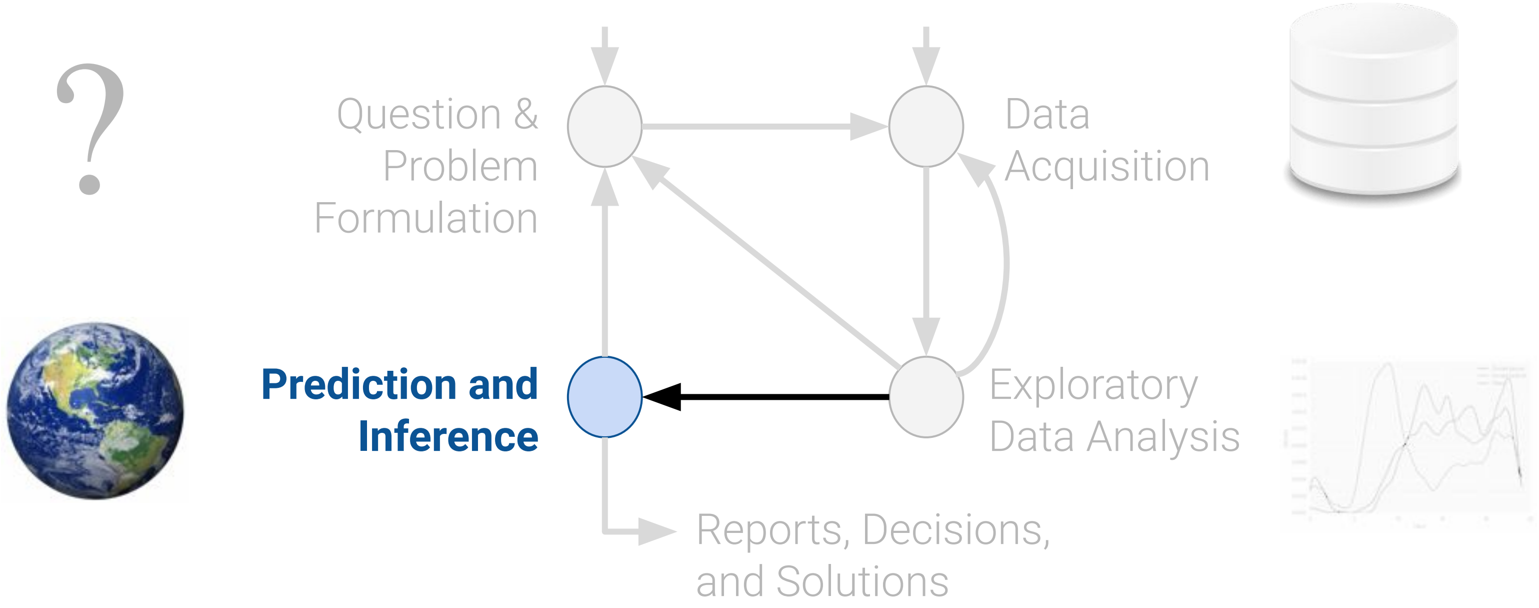
Goals for this Lecture

Last few lectures: underlying theory of modeling

This lecture: putting things into practice!

- Introducing `sklearn`, a useful Python library for building and fitting models
- Multiple Linear Regression
- Feature Engineering

Plan for Next Few Lectures: Modeling



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

(today)

Modeling I:
Intro to Modeling, Simple Linear Regression

→

Modeling II:
Different models, loss functions, linearization

→

**Modeling III:
Multiple Linear Regression**

sklearn

- Implementing Models in Code
- **sklearn**
- Multiple Linear Regression
- Feature Engineering
- One-Hot Encoding
- Polynomial Features
- Complexity and Overfitting

`sklearn`: a Standard Library for Model Creation

So far, we have been doing the “heavy lifting” of model creation ourselves – via calculus

In research and industry, it is more common to rely on data science libraries for creating and training models. In CSCI 3022 we will use Scikit-Learn, commonly called **`sklearn`**

```
import sklearn
my_model = linear_model.LinearRegression()
my_model.fit(X, y)
my_model.predict(X)
```

`sklearn`: a Standard Library for Model Creation

`sklearn` uses an object-oriented programming paradigm. Different types of models are defined as their own classes. To use a model, we initialize an instance of the model class.

- Don’t worry if you are not familiar with objects in Python. You can think of sklearn as allowing you to “copy” an existing template of a useful model.

The `sklearn` Workflow

At a high level, there are three steps to creating an **`sklearn`** model:

1 Initialize a new model instance
Make a "copy" of the model template

2 Fit the model to the training data
Save the optimal model parameters

3 Use fitted model to make predictions
Fitted model outputs predictions for y

The `sklearn` Workflow

At a high level, there are three steps to creating an **`sklearn`** model:

1	Initialize a new model instance *Make a “copy” of the model template*	`my_model = lm.LinearRegression()`
2	Fit the model to the training data *Save the optimal model parameters*	`my_model.fit(X, y)`
3	Use fitted model to make predictions *Fitted model makes predictions for y*	`my_model.predict(X)`

To extract the fitted parameters: `my_model.coef_` and `my_model.intercept_`

Multiple Linear Regression

- Implementing Models in Code
- `sklearn`
- Multiple Linear Regression

A Note on Terminology

There are several equivalent terms in the context of regression.

Feature(s)

Covariate(s)

Independent variable(s)

Explanatory variable(s)

Predictor(s)

Input(s)

Regressor(s)

Output

Outcome

Response

Dependent variable

Prediction

Predicted response

Estimated value

Weight(s)

Parameter(s)

Coefficient(s)

Estimator(s)

Optimal parameter(s)

Bolded terms are the most common in this course.

Match each column
with the appropriate term: $x, y, \hat{y}, \theta, \hat{\theta}$

🤔

A Note on Terminology

There are several equivalent terms in the context of regression.

Feature(s)

Covariate(s)

Independent variable(s)

Explanatory variable(s)

Predictor(s)

Input(s)

Regressor(s)

x

Output

Outcome

Response

Dependent variable

y

Prediction

Predicted response

Estimated value

$\hat{y}$

Weight(s)

Parameter(s)

Coefficient(s)

θ

Estimator(s)

Optimal parameter(s)

$\hat{\theta}$

Bolded terms are the most common in this course.

A datapoint (x, y) is also called an **observation**.

Multiple Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p$$

Today's Goal: Ordinary Least Squares

1. Choose a model	**Multiple Linear Regression**
2. Choose a loss function	L2 Loss **Mean Squared Error (MSE)**
3. Fit the model	Minimize average loss with linear algebra and/or geometry
4. Evaluate model performance	Visualize, ~~Root MSE~~ Multiple R^2

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p$$

In statistics, this model + loss is called **Ordinary Least Squares (OLS)**.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The solution to OLS are the minimizing loss for parameters $\hat{\theta}$, also called the **least squares estimate**.

Performance

Performance: Residuals, Multiple R^2

OLS Properties

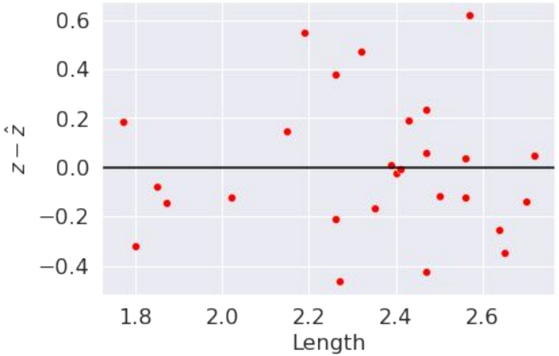
- Residuals
- Existence of a Unique Solution

[Visualization] Residual Plots

Compare

Simple linear regression

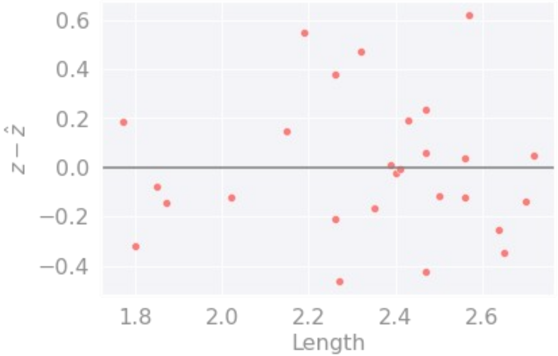
Plot residuals vs
the single feature x.



[Visualization] Residual Plots

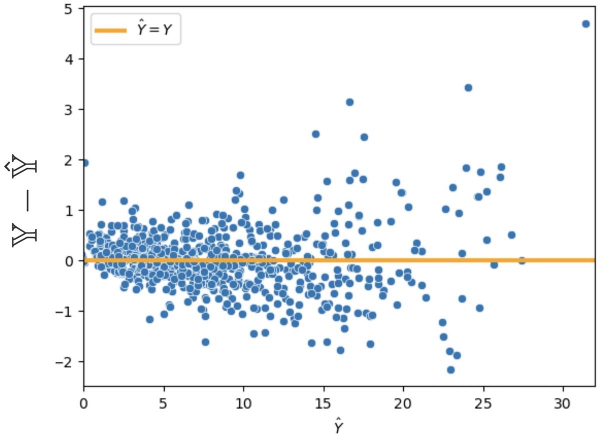
Simple linear regression

Plot residuals vs
the single feature *x*.



Multiple linear regression

Plot residuals vs
fitted (predicted) values $\hat{y}$.



Compare

See notebook

Same interpretation as before

- A good residual plot shows no pattern.
- A good residual plot also has a similar vertical spread throughout the entire plot. Else (heteroscedasticity), the accuracy of the predictions is not reliable.

[Metrics] Multiple R²

Compare

Simple linear regression

Error
RMSE

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Linearity
Correlation coefficient, ***r***

$$r = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{\sigma_x}\right)\left(\frac{y_i - \bar{y}}{\sigma_y}\right)$$

Multiple linear regression

Error
RMSE

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Linearity
Multiple R^2, also called the **coefficient of determination**

$$R^2 = \frac{\text{variance of fitted values}}{\text{variance of } y} = \frac{\sigma_{\hat{y}}^2}{\sigma_y^2}$$

Compare

[Metrics] Multiple R²

We define the **multiple R²** value as the **proportion of variance** or our **fitted values** (predictions) $\hat{y}$ to our true values y.

$$R^2 = \frac{\text{variance of fitted values}}{\text{variance of } y} = \frac{\sigma_{\hat{y}}^2}{\sigma_y^2}$$

Also called the **correlation of determination**.

R^2 ranges from 0 to 1 and is effectively
"the proportion of variance that the **model explains**."

For OLS with an intercept term (e.g. $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p$),

$R^2 = [r(y, \hat{y})]^2$ is equal to the square of correlation between y ,$\hat{y}$.

- For SLR, $R^2 = r^2$, the correlation between x, y.
- The proof of these last two properties is beyond this course.

Properties When Our Model Has an Intercept Term

For all linear models with an **intercept term**, the **sum of residuals is zero**.

$$\sum_{i=1}^{n} e_i = 0$$ (previous slide)

- This is the real reason why we don’t directly use residuals as loss. $$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n} e_i = 0$$
- This is also why positive and negative residuals will cancel out in any residual plot where the (linear) model contains an intercept term, even if the model is terrible.

It follows from the property above that for linear models with intercepts, the average predicted y value is equal to the average true y value.

$$\bar{y} = \overline{\hat{y}}$$

These properties are true when there is an intercept term, and not necessarily when there isn’t.

Does a Unique Solution Always Exist?

	Model	Estimate	Unique?
Constant Model + MSE	$\hat{y} = \theta_0$	$\hat{\theta}_0 = mean(y) = \bar{y}$	**Yes**. Any set of values has a unique mean.
Constant Model + MAE	$\hat{y} = \theta_0$	$\hat{\theta}_0 = median(y)$	**Yes**, if odd. **No**, if even. Return average of middle 2 values.
Simple Linear Regression + MSE	$\hat{y} = \theta_0 + \theta_1 x$	$\hat{\theta}_0 = \bar{y} - \hat{\theta}_1 \bar{x}$ $\hat{\theta}_1 = r\frac{\sigma_y}{\sigma_x}$	**Yes**. Any set of non-constant* values has a unique mean, SD, and correlation coefficient.
Ordinary Least Squares (Linear Model + MSE)	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	$\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$	**Yes**, if $\mathbb{X}$ is full col rank (all cols lin independent, #datapoints>> #features)

Feature Engineering

- Implementing Models in Code
- `sklearn`
- **Feature Engineering**
- Polynomial Features
- Complexity and Overfitting

Feature Engineering

Feature engineering is the process of transforming raw features into more informative features for use in modeling

Allows us to:

- Capture domain knowledge
- Express non-linear relationships using linear models
- Use non-numeric features in models

Multiple Linear Regression

Define the **multiple linear regression** model:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p$$

Parameters are $\theta = (\theta_0, \theta_1, \ldots, \theta_p)$.

Is this linear in θ ?

A. no
B. yes
C. maybe

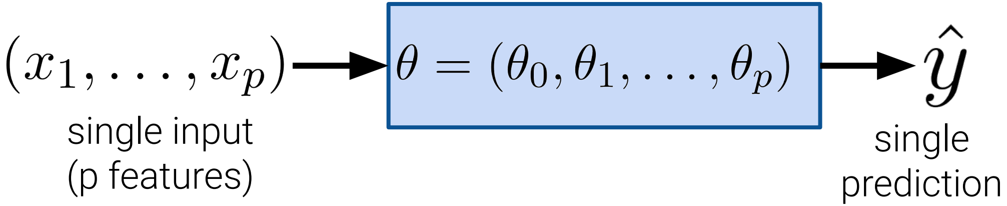
Multiple Linear Regression

Define the **multiple linear regression** model:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p$$

Parameters are $\theta = (\theta_0, \theta_1, \ldots, \theta_p)$.

Yes! This is a **linear combination** of θ_j'S, each scaled by x_j.



Example: Predict dugong ages $\hat{y}$ as a linear model of 2 features: length x_1 **and** weight x_2.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

intercept

parameter for length

parameter for weight

Linear in Theta

An expression is "**linear in theta**" if it is a **linear combination** of parameters $\theta = [\theta_0, \theta_1, \dots, \theta_p]$.

1. $\hat{y} = \theta_0 + \theta_1(2) + \theta_2(4 \cdot 8) + \theta_3(\log 42)$

2. $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 x_3 + \theta_3 . \log(x_4)$

3. $\hat{y} = \theta_0 + \theta_1 x_1 + \log(\theta_2) x_2 + \theta_3 \theta_4$

4. $$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 \\ 1 & 8 & 9 & 0 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

5. $$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} \\ 1 & x_{21} & x_{22} & x_{23} \\ 1 & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

Which of the following expressions are linear in theta?

Linear in Theta

An expression is "**linear in theta**" if it is a **linear combination** of parameters $\theta = [\theta_0, \theta_1, \dots, \theta_p]$.

1. $\hat{y} = \theta_0 + \theta_1(2) + \theta_2(4 \cdot 8) + \theta_3(\log 42)$

$$= \begin{bmatrix} 1 & 2 & 4.8 & \log(42) \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

2. $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 x_3 + \theta_3 . \log(x_4)$

$$= \begin{bmatrix} 1 & x_1 & x_2 x_3 & \log(x_4) \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

3. $\hat{y} = \theta_0 + \theta_1 x_1 + \log(\theta_2) x_2 + \theta_3 \theta_4$

❌

4. $$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 \\ 1 & 8 & 9 & 0 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

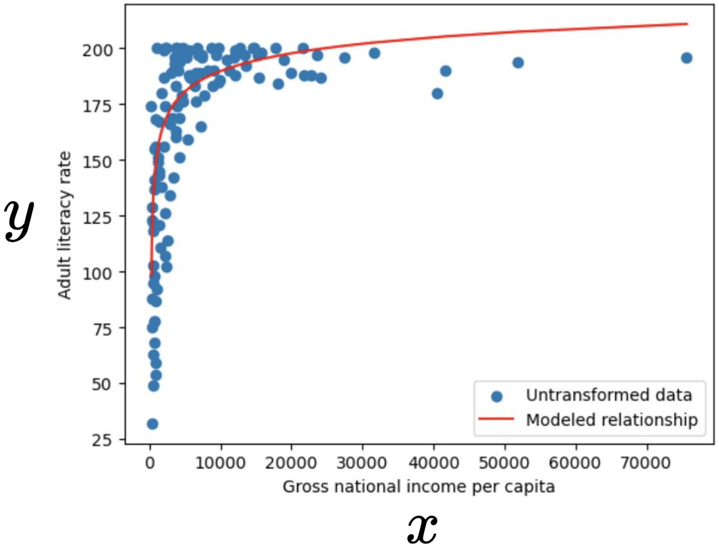
5. $$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} \\ 1 & x_{21} & x_{22} & x_{23} \\ 1 & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

"**Linear in theta**" means the expression can separate into a matrix product of two terms: **a vector of thetas**, and a matrix/vector not involving thetas.

Transforming Features

Two observations:

- We looked at transforming variables – we found that applying a transformation could help **linearize** a dataset
- In our work on SLR modeling, we saw that linear **modeling works best** when our dataset has linear relationships



$$y^4 = m(\log x) + b$$

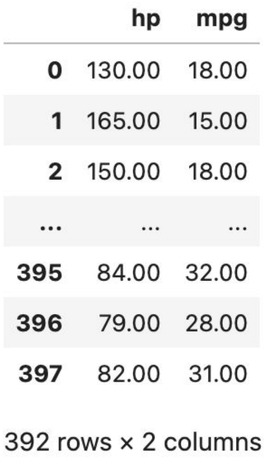
Putting ideas together:

Feature engineering = transforming features to improve model performance

Feature Functions

A **feature function** describes the transformations we apply to raw features in the dataset to create transformed features. Often, the dimension of the *featurized* dataset increases.

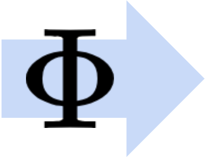
Example: a feature function that adds a squared feature to the design matrix

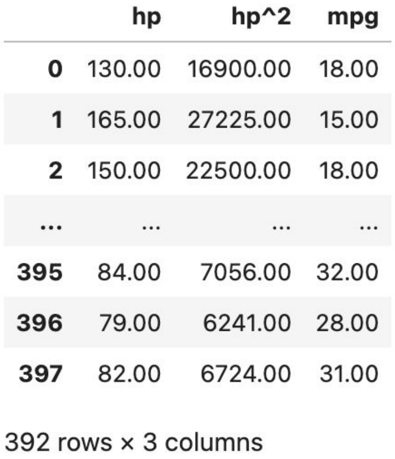


	hp	mpg
0	130.00	18.00
1	165.00	15.00
2	150.00	18.00
...	...	...
395	84.00	32.00
396	79.00	28.00
397	82.00	31.00

392 rows × 2 columns

Dataset of raw features:





	hp	hp^2	mpg
0	130.00	16900.00	18.00
1	165.00	27225.00	15.00
2	150.00	22500.00	18.00
...	...	...	...
395	84.00	7056.00	32.00
396	79.00	6241.00	28.00
397	82.00	6724.00	31.00

392 rows × 3 columns

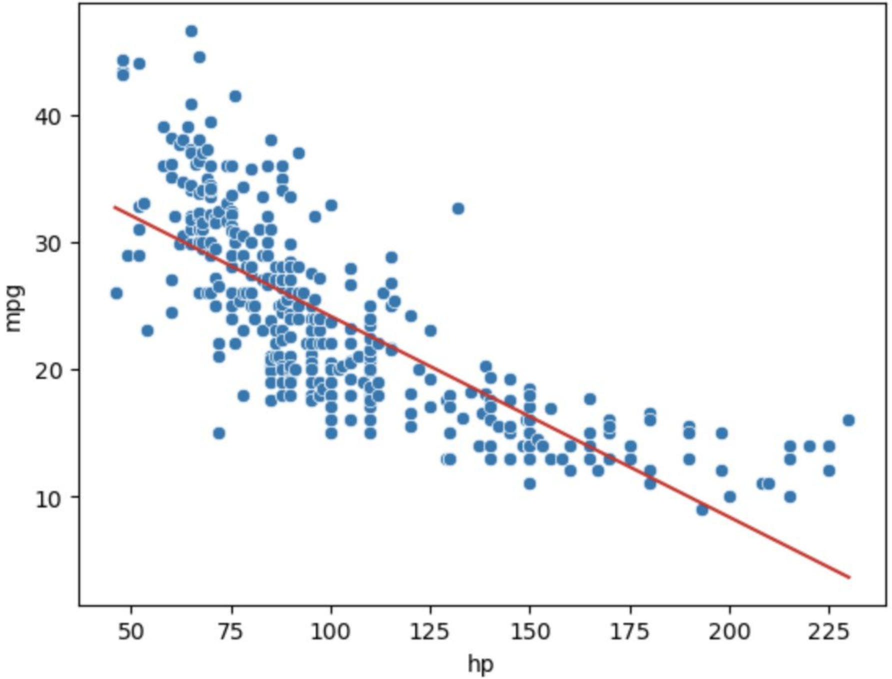
After applying the feature function Φ:

Polynomial Features

- `sklearn`
- Feature Engineering
- One-Hot Encoding
- **Polynomial Features**
- Complexity and Overfitting

Accounting for Curvature

We’ve seen a few cases now where models with linear features have performed poorly on datasets with a clear non-linear curve.



$$\hat{y} = \theta_0 + \theta_1(\text{hp})$$

MSE: 23.94

When our model uses only a single linear feature (**hp**), it cannot capture non-linearity in the relationship

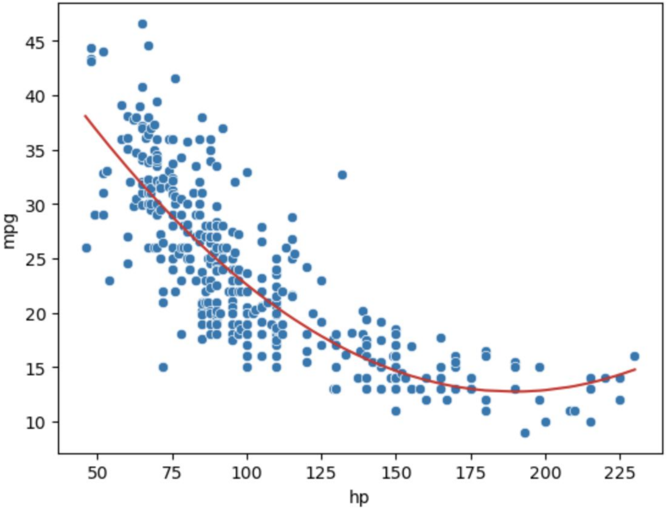
Solution: incorporate a non-linear feature!

Polynomial Features

We create a new feature: the square of the **hp**

$$\hat{y} = \theta_0 + \theta_1(\text{hp}) + \theta_2\left(\text{hp}^2\right)$$

This is still a **linear model**. Even though there are non-linear *features*, the model is linear with respect to θ

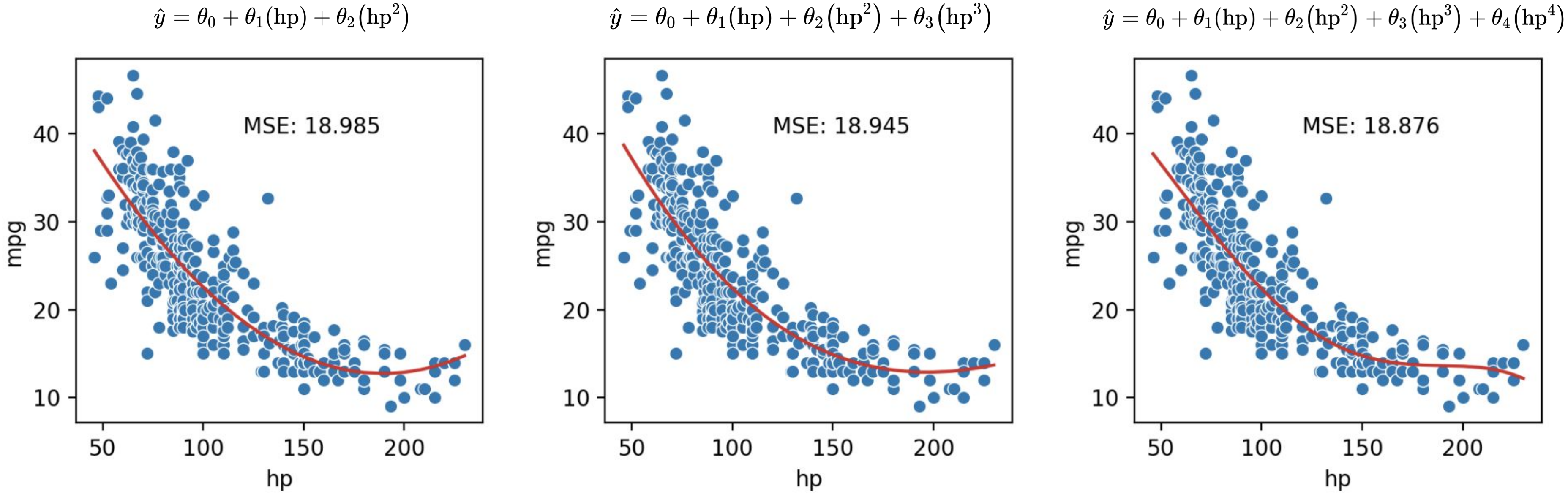


Degree of model: 2
MSE: 18.98

Looking a lot better: our predictions capture the curvature of the data.

Polynomial Features

What if we add more polynomial features?



$\hat{y} = \theta_0 + \theta_1(\text{hp}) + \theta_2\left(\text{hp}^2\right)$

$\hat{y} = \theta_0 + \theta_1(\text{hp}) + \theta_2\left(\text{hp}^2\right) + \theta_3\left(\text{hp}^3\right)$

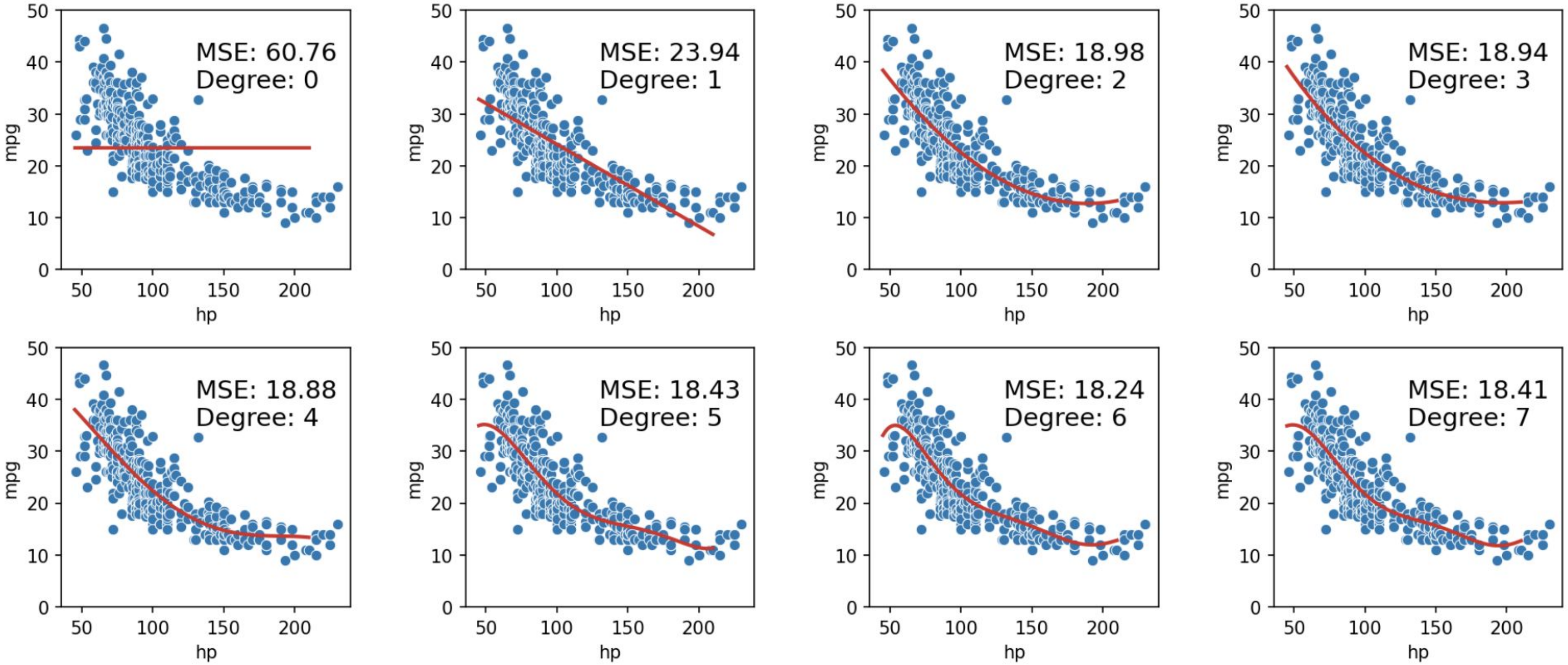
$\hat{y} = \theta_0 + \theta_1(\text{hp}) + \theta_2\left(\text{hp}^2\right) + \theta_3\left(\text{hp}^3\right) + \theta_4\left(\text{hp}^4\right)$

MSE continues to decrease with each additional polynomial term

Complexity and Overfitting

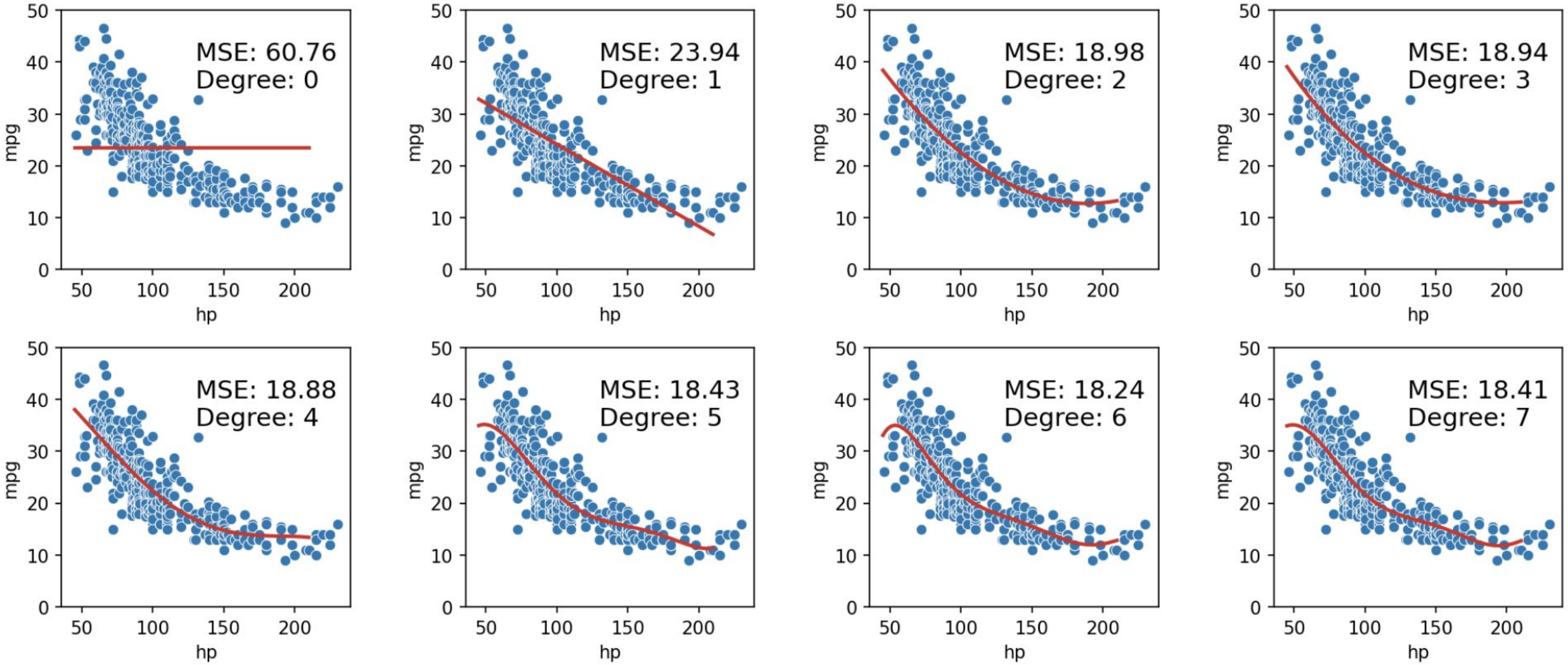
- `sklearn`
- Feature Engineering
- One-Hot Encoding
- Polynomial Features
- **Complexity and Overfitting**

How Far Can We Take This?



How Far Can We Take This?

Poll: Which higher-order polynomial model do you think fits best?

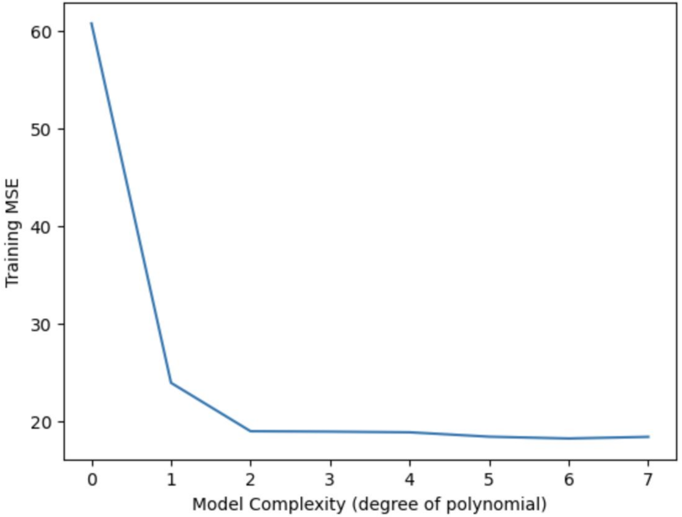


Model Complexity

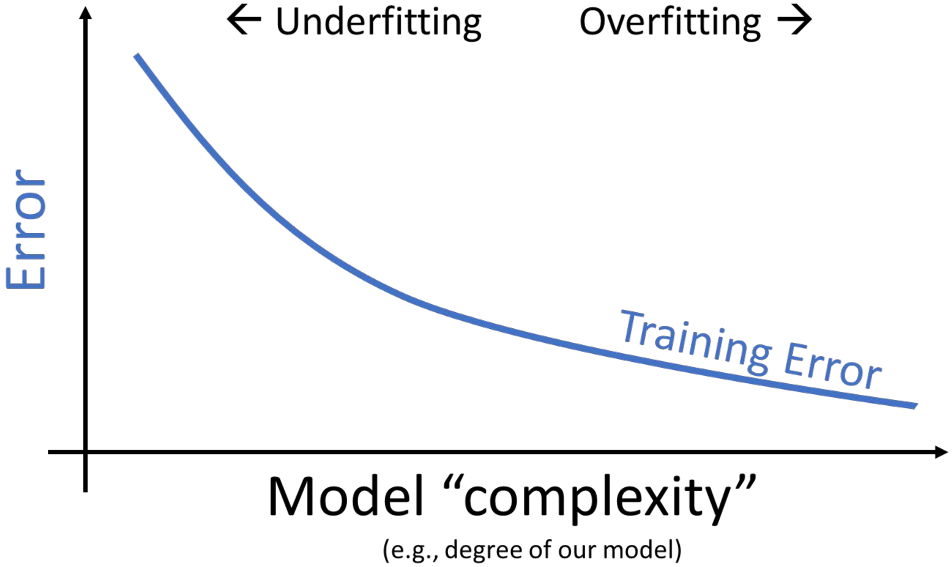
As we continue to add more and more polynomial features, the MSE continues to decrease

Equivalently: as the **model complexity** increases, its *training error* decreases

Our experiment using `vehicles`



General trend for an arbitrary dataset

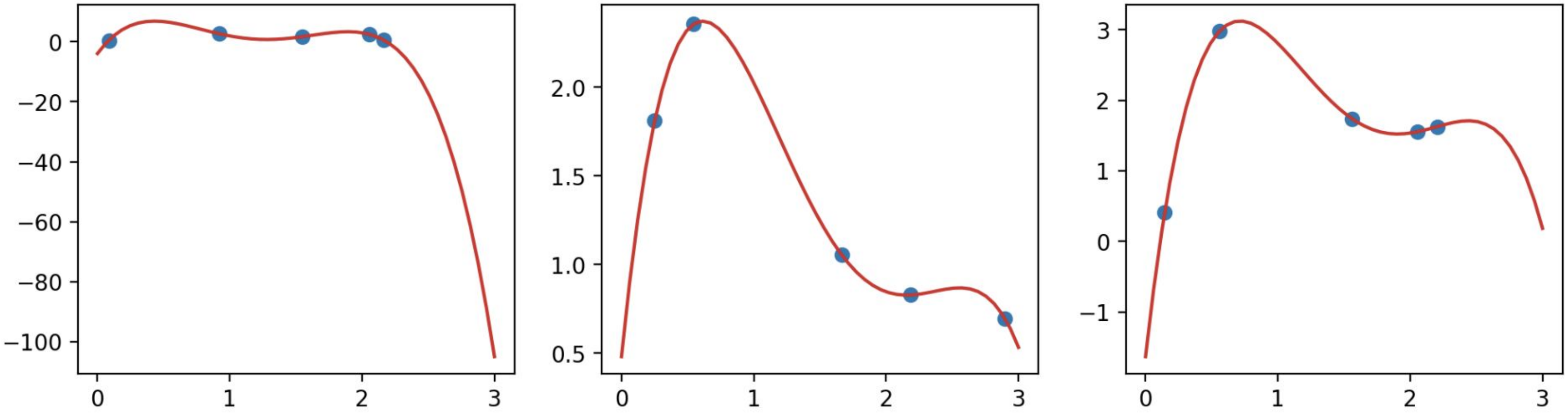


Seems like a good deal?

An Extreme Example: Perfect Polynomial Fits

Math fact: given N non-overlapping data points, we can always find a polynomial of degree N-1 that goes through all those points.

For example, there always exists a degree-4 polynomial curve that can perfectly model a dataset of 5 datapoints



Model Performance on Unseen Data

Our `vehicle` models from before considered a somewhat artificial scenario – we trained the models on the *entire* dataset, then evaluated their ability to make predictions on this same dataset

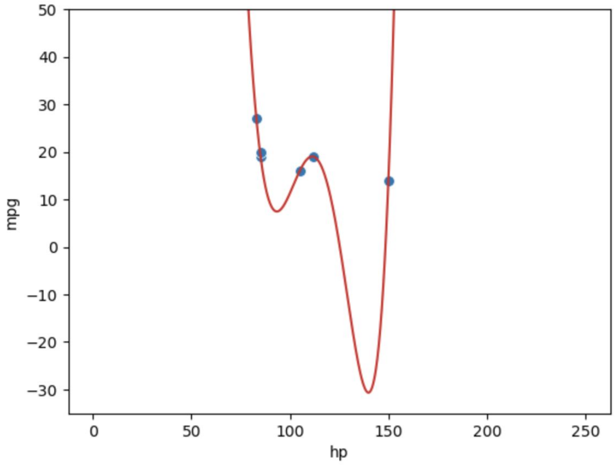
More realistic situation: we train the model on a *sample* from the population, then use it to make predictions on data it didn't encounter during training

Model Performance on Unseen Data

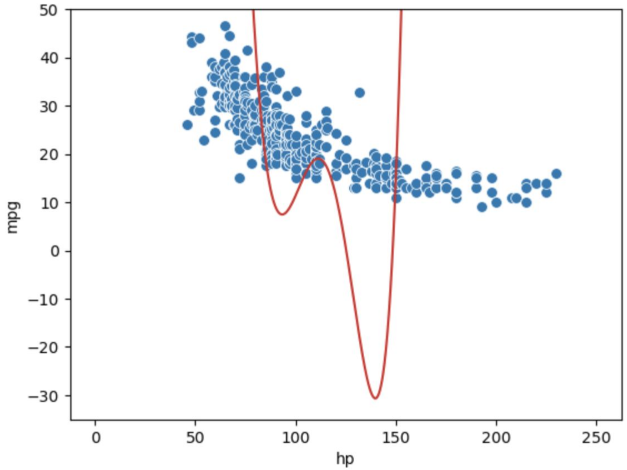
New (more realistic) example:

- We are given a training dataset of just 6 datapoints
- We want to train a model to then make predictions on a *different* set of points

We may be tempted to make a highly complex model (eg degree 5)



Complex model makes perfect predictions on the training data…



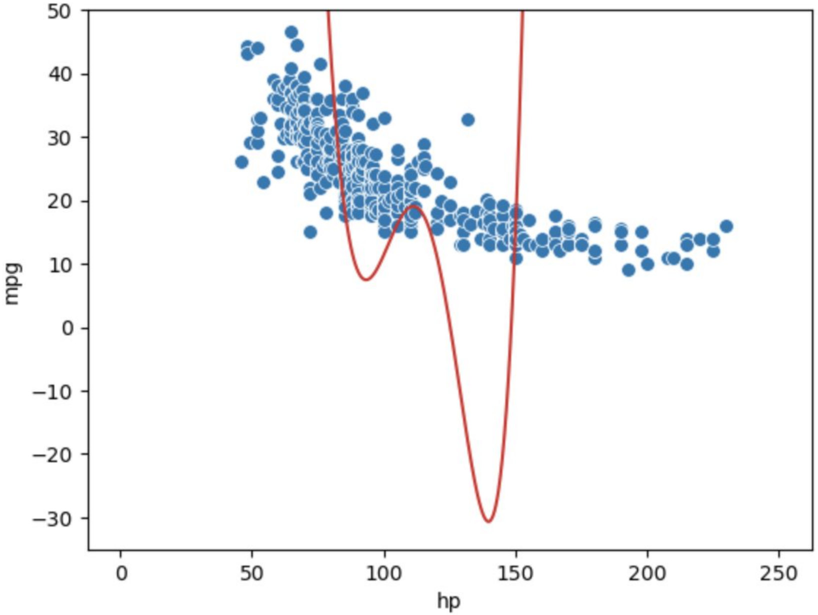
…but performs *horribly* on the rest of the population!

Model Performance on Unseen Data

What went wrong?

- The complex model **overfit** to the training data – it essentially “memorized” these 6 training points
- The overfitted model does not **generalize** well to data it did not encounter during training

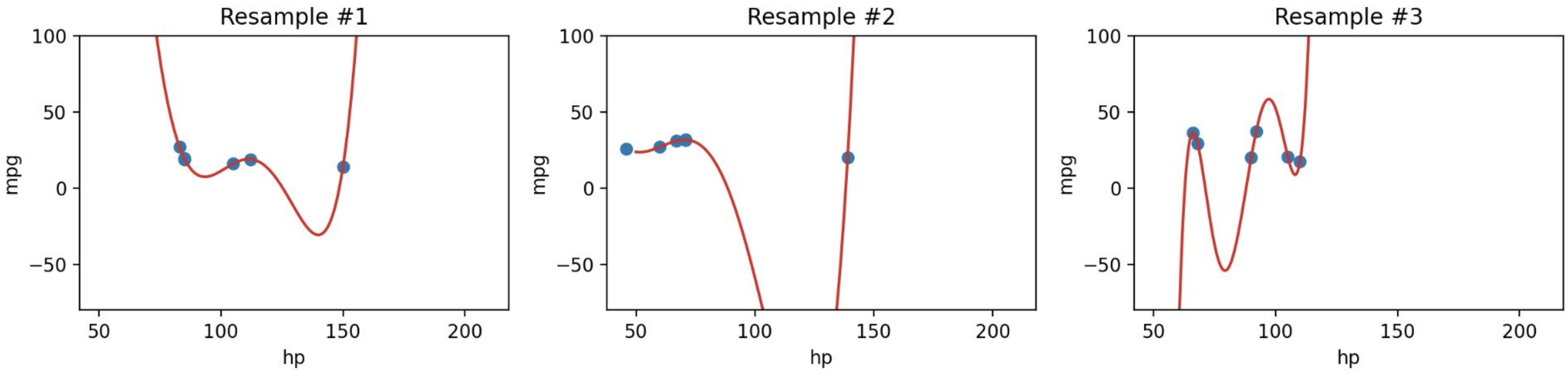
This is a problem: we want models that are generalizable to “unseen” data



Model Variance

Complex models are sensitive to the specific dataset used to train them – they have high **variance**, because they will *vary* depending on what datapoints are used for training them

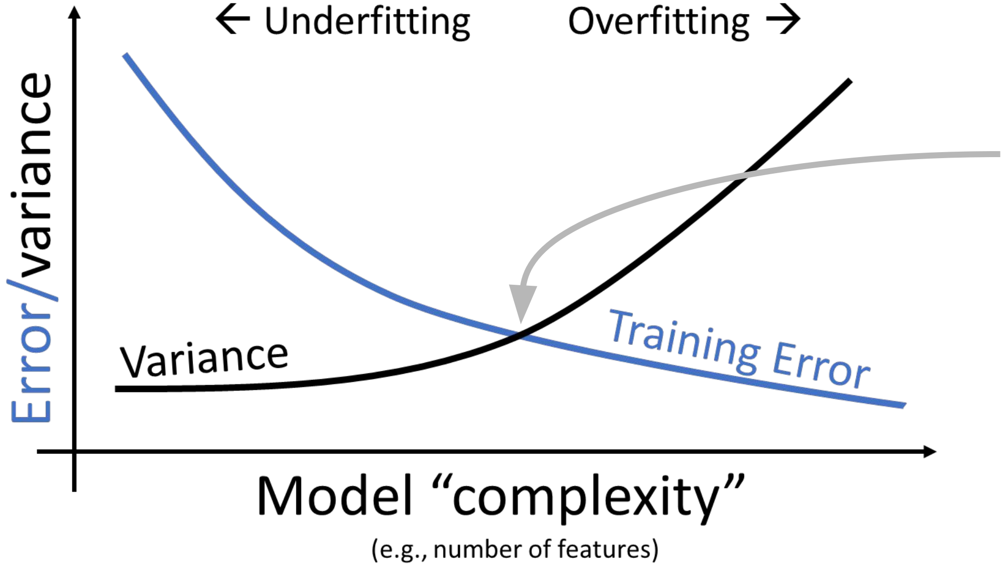
Our degree-5 model varies erratically when we fit it to different samples of 6 points from `vehicles`



Error, Variance, and Complexity

We face a dilemma:

- We know that we can **decrease training error** by increasing model complexity
- However, models that are *too* complex start to overfit and do not generalize well – their **high variance** means they can't be reapplied to new datasets



Our goal: find this "sweet spot"

Stay tuned for future lectures covering this!

Appendix: Geometric Derivation

2327602

OLS Problem Formulation

- Multiple Linear Regression Model
- Mean Squared Error

Geometric Derivation

- Lin Alg Review: Orthogonality, Span
- Least Squares Estimate Proof

[Linear Algebra] Vector Norms and the L2 Vector Norm

The **norm** of a vector is some measure of that vector's **size**.

- The two norms we need to know for Data 100 are the L_1 and L_2 norms (sound familiar?).
- Today, we focus on L_2 norm. We'll define the L_1 norm another day.

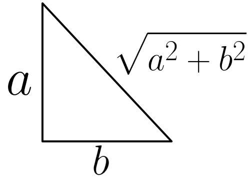
For the n-dimensional vector $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, the **L2 vector norm** is

$$||\vec{x}||_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\sum_{i=1}^{n} x_i^2}$$

[Linear Algebra] The L2 Norm Is a Measure of Distance

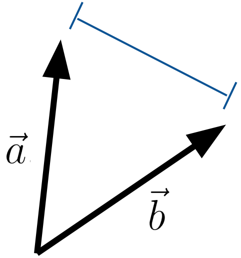
$$||\vec{x}||_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\sum_{i=1}^{n} x_i^2}$$

In $\mathbb{R}^2$



The L2 vector norm is a generalization of the Pythagorean theorem into ***n*** dimensions.

In $\mathbb{R}^n$



It can therefore be used as a measure of **distance** between two vectors.

- For n-dimensional vectors $\vec{a}, \vec{b}$, their distance is $||\vec{a} - \vec{b}||_2$.

Note: The square of the L2 norm of a vector is the sum of the squares of the vector's elements:

$$(||\vec{x}||_2)^2 = \sum_{i=1}^{n} x_i^2$$

Looks like Mean Squared Error!!

Mean Squared Error with L2 Norms

We can rewrite mean squared error as a squared L2 norm:

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{n}||\mathbb{Y} - \hat{\mathbb{Y}}||_2^2$$

With our linear model $\hat{\mathbb{Y}} = \mathbb{X}\theta$:

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

Ordinary Least Squares

The **least squares estimate** $\hat{\theta}$ is the parameter that **minimizes** the objective function $R(\theta)$:

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

How should we interpret the OLS problem?

A. Minimize the mean squared error for the linear model $\hat{\mathbb{Y}} = \mathbb{X}\theta$

B. Minimize the **distance** between true and predicted values $\mathbb{Y}$ and $\hat{\mathbb{Y}}$

C. Minimize the **length** of the residual vector, $e = \mathbb{Y} - \hat{\mathbb{Y}} = \begin{bmatrix} y_1 - \hat{y_1} \\ y_2 - \hat{y_2} \\ \vdots \\ y_n - \hat{y_n} \end{bmatrix}$

D. All of the above

E. Something else

Ordinary Least Squares

The **least squares estimate** $\hat{\theta}$ is the parameter that **minimizes** the objective function $R(\theta)$:

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

How should we interpret the OLS problem?

A. Minimize the mean squared error for the linear model $\hat{\mathbb{Y}} = \mathbb{X}\theta$

B. Minimize the **distance**
between true and predicted values $\mathbb{Y}$ and $\hat{\mathbb{Y}}$

C. Minimize the **length** of the residual vector, $e = \mathbb{Y} - \hat{\mathbb{Y}} = \begin{bmatrix} y_1 - \hat{y_1} \\ y_2 - \hat{y_2} \\ \vdots \\ y_n - \hat{y_n} \end{bmatrix}$

$$\left.\begin{bmatrix} y_1 - \hat{y_1} \\ y_2 - \hat{y_2} \\ \vdots \\ y_n - \hat{y_n} \end{bmatrix}\right\} \text{Important for today}$$

[✓ D. All of the above]

E. Something else

Today's Goal: Ordinary Least Squares

1. Choose a model ✅

Multiple Linear Regression

$$\hat{\mathbb{Y}} = \mathbb{X}\theta$$

2. Choose a loss function ✅

L2 Loss

Mean Squared Error (MSE)

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

3. Fit the model

Minimize average loss with ~~calculus~~ geometry

The calculus derivation requires matrix calculus (out of scope, but here's a link if you're interested).

Instead, we will derive $\hat{\theta}$ using a **geometric argument**.

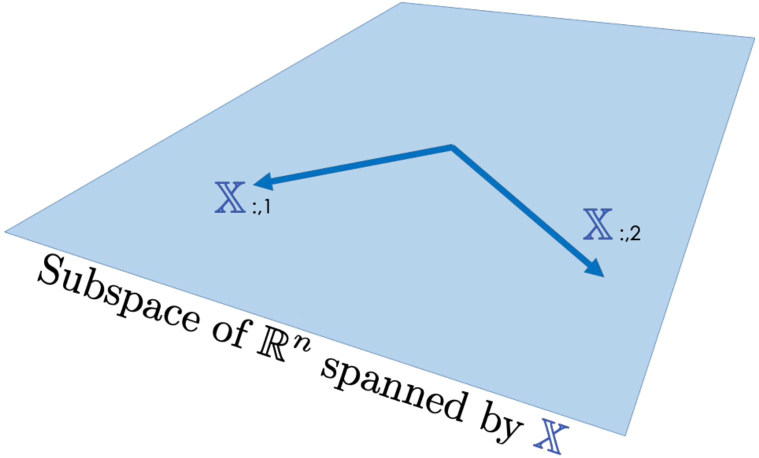
4. Evaluate model performance

Visualize,
~~Root MSE~~
Multiple R^2

[Linear Algebra] Span

The set of all possible linear combinations of the columns of $\mathbb{X}$ is called the **span** of the columns of $\mathbb{X}$ (denoted $span(\mathbb{X})$), also called the **column space**.

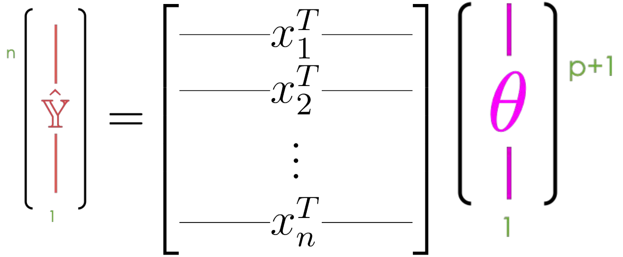
- Intuitively, this is all of the vectors you can "reach" using the columns of $\mathbb{X}$.
- If each column of $\mathbb{X}$ has length ***n***, $span(\mathbb{X})$ is a subspace of $\mathbb{R}^n$.



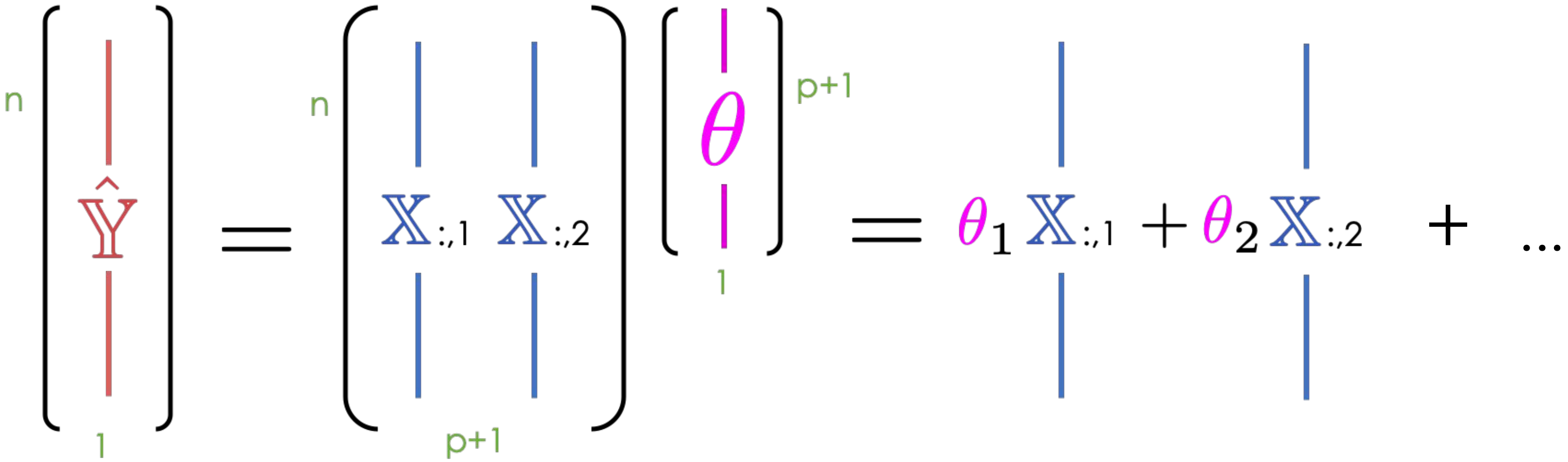
A Linear Combination of Columns

$$\hat{\mathbb{Y}} = \mathbb{X}\,\theta$$

So far, we've thought of our model as horizontally stacked predictions per datapoint:



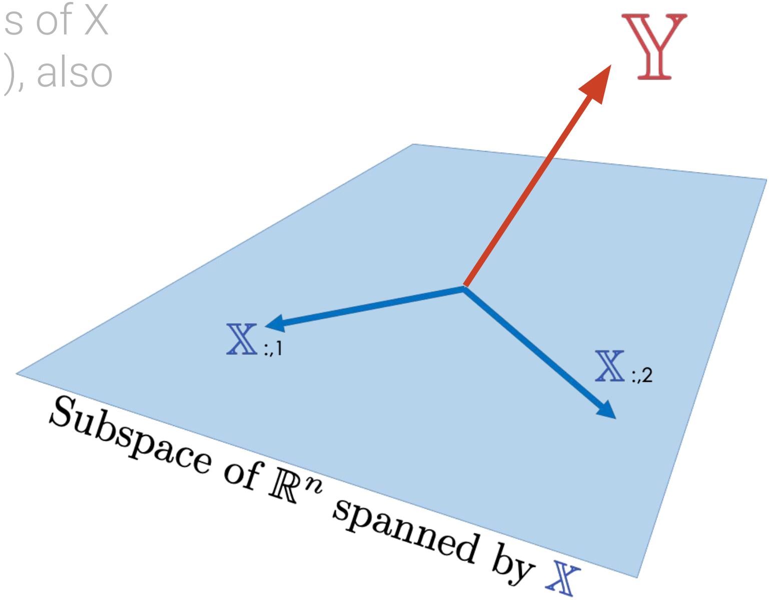
We can also think of $\hat{\mathbb{Y}}$ as a **linear combination of feature vectors**, scaled by **parameters**.



A Linear Combination of Columns

The set of all possible linear combinations of the columns of X is called the **span** of the columns of X (denoted $span(\mathbb{X})$), also called the **column space**.

- Intuitively, this is all of the vectors you can "reach" using the columns of X.
- If each column of X has length *n*, $span(\mathbb{X})$ is a subspace of $\mathbb{R}^n$.

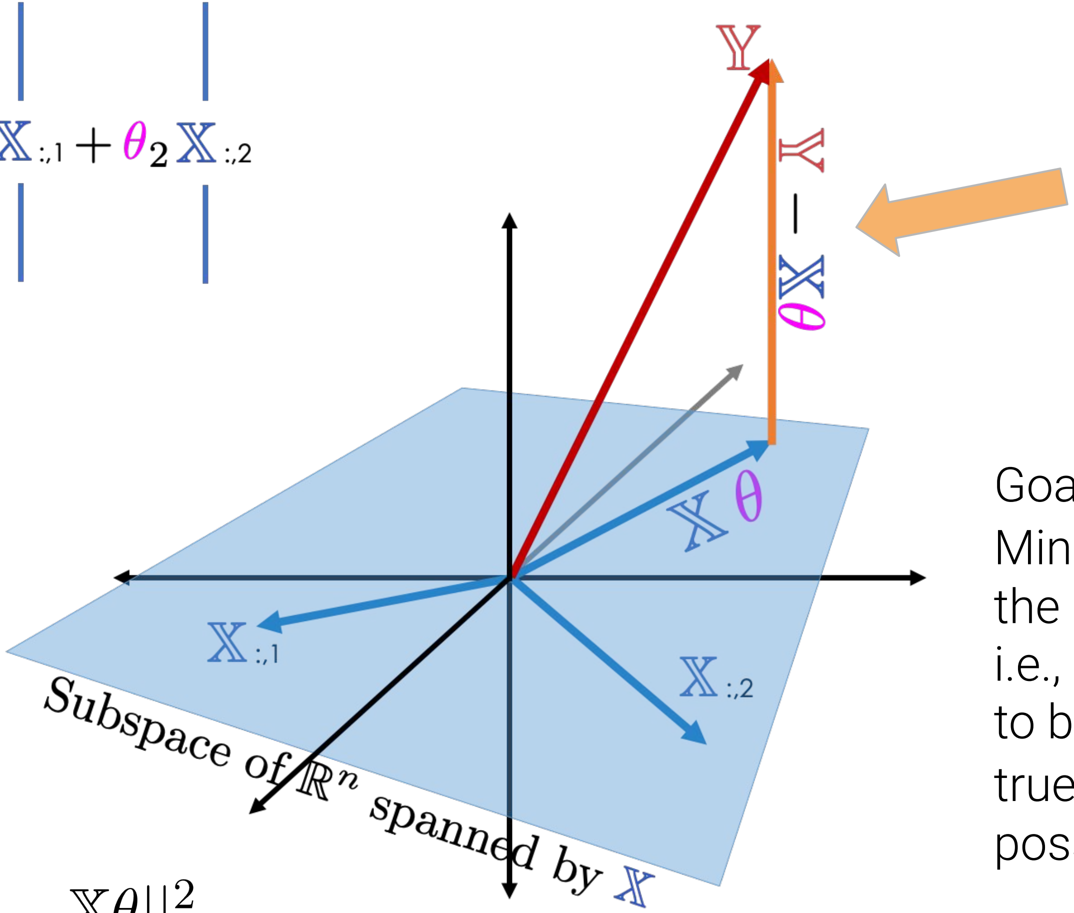


Our prediction $\hat{\mathbb{Y}} = \mathbb{X}\theta$ is a **linear combination** of the columns of $\mathbb{X}$. Therefore $\hat{\mathbb{Y}} \in span(\mathbb{X})$.

Interpret: Our linear prediction $\hat{\mathbb{Y}}$ will be in $span(\mathbb{X})$, even if the true values $\mathbb{Y}$ might not be.

Goal: Find the vector in $span(\mathbb{X})$ that is **closest** to $\mathbb{Y}$.

$$\begin{array}{c c} n & \\ & \left[\begin{array}{c} | \\ \hat{\mathbb{Y}} \\ | \end{array}\right] \\ & 1 \end{array} = \theta_1 \left.\begin{array}{c} | \\ \mathbb{X}_{:,1} \\ | \end{array}\right. + \theta_2 \left.\begin{array}{c} | \\ \mathbb{X}_{:,2} \\ | \end{array}\right.$$



$\mathbb{Y}$
$\mathbb{Y} - \mathbb{X}\theta$
$\mathbb{X}\theta$
$\mathbb{X}_{:,1}$
$\mathbb{X}_{:,2}$
Subspace of $\mathbb{R}^n$ spanned by $\mathbb{X}$

This is the residual vector, $e = \mathbb{Y} - \hat{\mathbb{Y}}$.

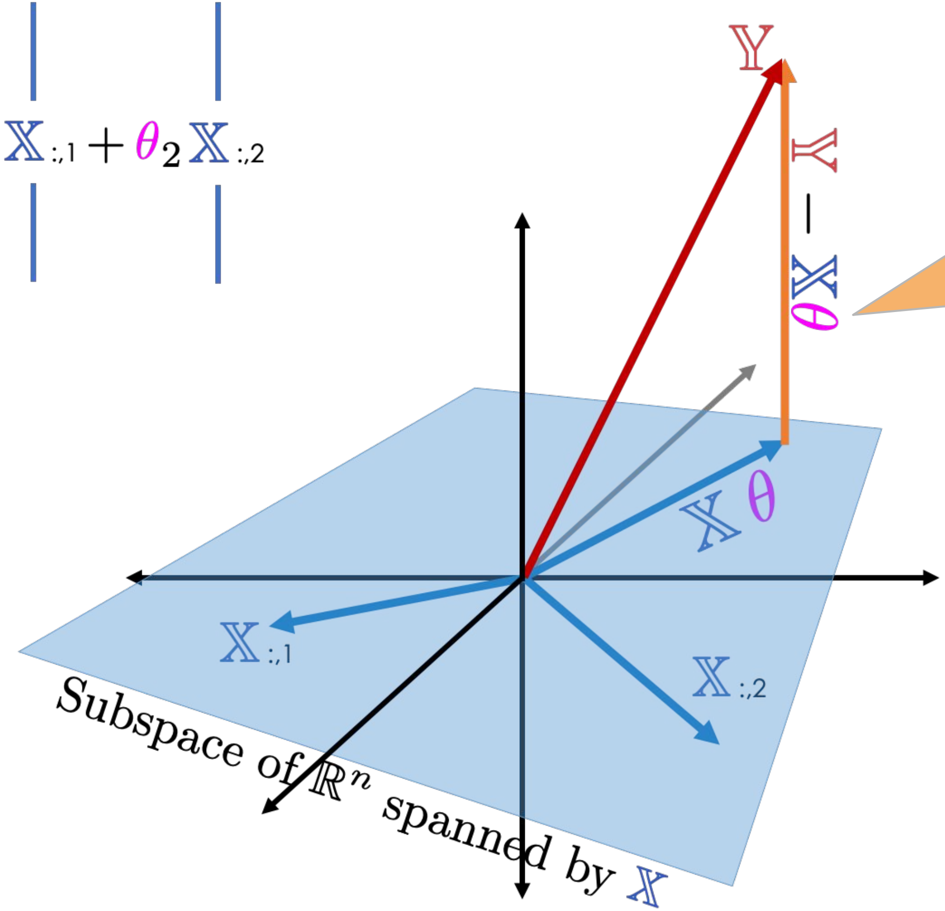
Goal:

Minimize the L_2 norm of the residual vector.

i.e., get the predictions $\hat{\mathbb{Y}}$ to be "as close" to our true $\mathbb{Y}$ values as possible.

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

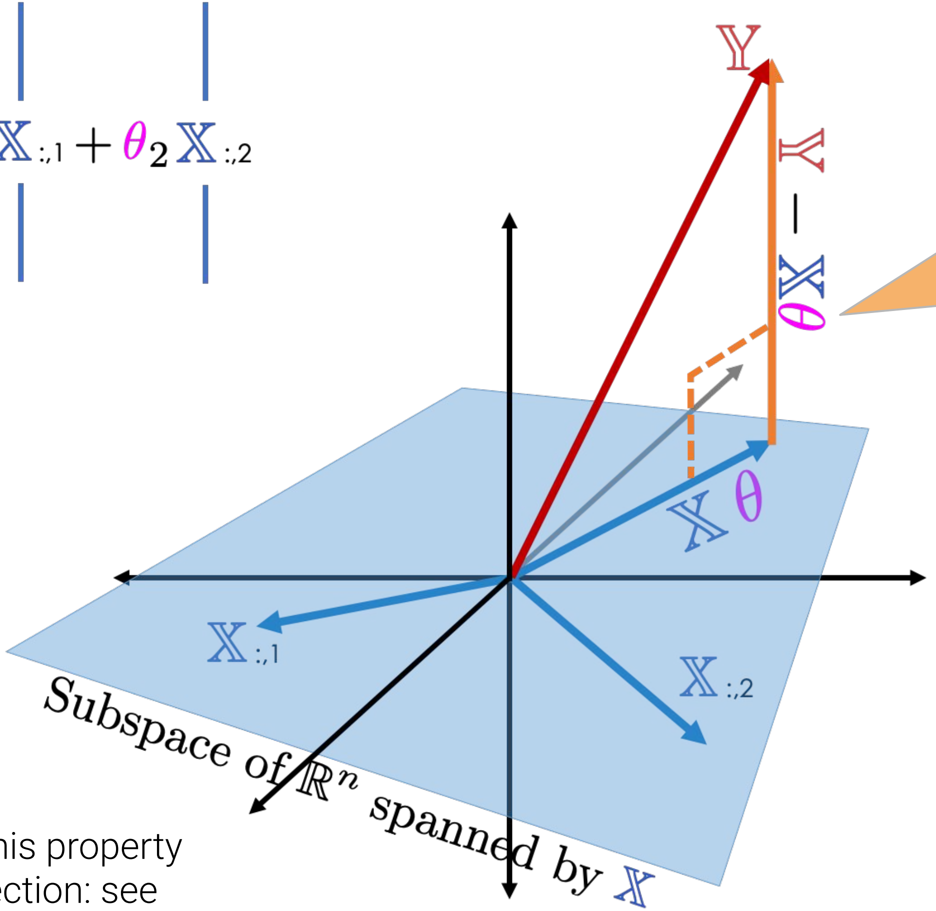
$${}^{n}\left[\hat{\mathbb{Y}}\right]_{1} = \theta_1 \mathbb{X}_{:,1} + \theta_2 \mathbb{X}_{:,2}$$


$\mathbb{Y}$
$\mathbb{Y} - \mathbb{X}\theta$
$\mathbb{X}\theta$
$\mathbb{X}_{:,1}$
$\mathbb{X}_{:,2}$
Subspace of $\mathbb{R}^n$ spanned by $\mathbb{X}$

How do we minimize this distance – the norm of the residual vector (squared)?

2327602

$$\begin{array}{cc} n & \left[\hat{\mathbb{Y}}\right] \\ & 1 \end{array} = \theta_1 \mathbb{X}_{:,1} + \theta_2 \mathbb{X}_{:,2}$$


$\mathbb{Y}$
$\mathbb{Y} - \mathbb{X}\theta$
$\mathbb{X}\theta$
$\mathbb{X}_{:,1}$
$\mathbb{X}_{:,2}$
Subspace of $\mathbb{R}^n$ spanned by $\mathbb{X}$

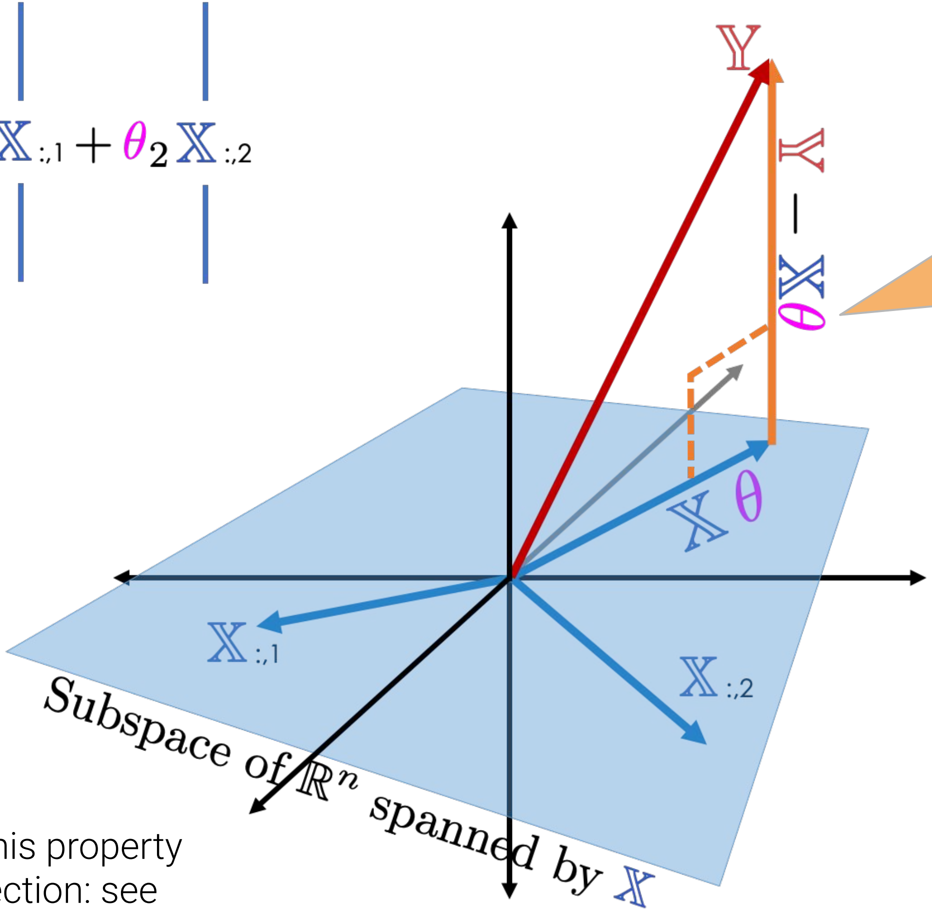
How do we minimize this distance – the norm of the residual vector (squared)?

The vector in $span(\mathbb{X})$ that is closest to $\mathbb{Y}$ is the **orthogonal projection** of $\mathbb{Y}$ onto $span(\mathbb{X})$.

We will not prove this property of orthogonal projection: see Khan Academy.

2327602

$$\underset{n \times 1}{\hat{\mathbb{Y}}} = \theta_1 \mathbb{X}_{:,1} + \theta_2 \mathbb{X}_{:,2}$$


$\mathbb{Y}$
$\mathbb{Y} - \mathbb{X}\theta$
$\mathbb{X}\theta$
$\mathbb{X}_{:,1}$
$\mathbb{X}_{:,2}$
Subspace of $\mathbb{R}^n$ spanned by $\mathbb{X}$

How do we minimize this distance – the norm of the residual vector (squared)?

The vector in $span(\mathbb{X})$ that is closest to $\mathbb{Y}$ is the **orthogonal projection** of $\mathbb{Y}$ onto $span(\mathbb{X})$.

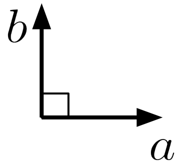
Thus, we should choose the θ that makes the residual vector **orthogonal** to $span(\mathbb{X})$.

We will not prove this property of orthogonal projection: see Khan Academy.

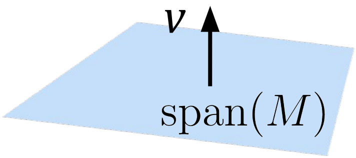
[Linear Algebra] Orthogonality

1. Vector a and Vector b are **orthogonal** if and only if their dot product is 0: $a^T b = 0$

This is a generalization of the notion of two vectors in 2D being perpendicular.



2. A vector v is **orthogonal** to $\mathrm{span}(M)$, the span of the columns of a matrix M, if and only if v is orthogonal to **each column** in M.



Let's express **2** in matrix notation. Let $v \in \mathbb{R}^{n\times 1}$, $M \in \mathbb{R}^{n\times d}$ where $M = \begin{bmatrix} | & | & & | \\ m_1 & m_2 & \dots & m_d \\ | & | & & | \end{bmatrix}$:

$$\begin{aligned} m_1^T v &= 0 \\ m_2^T v &= 0 \\ &\vdots \\ m_d^T v &= 0 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} m_1^T v \\ m_2^T v \\ \vdots \\ m_d^T v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Rightarrow \quad \underbrace{M^T}_{M^T \in \mathbb{R}^{d\times n}} v = \underbrace{\vec{0}}$$

v is orthogonal to each column of M, $m_j \in \mathbb{R}^{n\times 1}$

zero vector (*d*-length vector full of 0s).

Ordinary Least Squares Proof

The **least squares estimate** $\hat{\theta}$ is the parameter θ that minimizes the objective function $R(\theta)$:

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

Design matrix

$$M^T v = \vec{0}$$

Residual vector

Equivalently, this is the $\hat{\theta}$ such that the residual vector $\mathbb{Y} - \mathbb{X}\hat{\theta}$ is orthogonal to $span(\mathbb{X})$.

Definition of orthogonality of $\mathbb{Y} - \mathbb{X}\hat{\theta}$ to $span(\mathbb{X})$ (0 is the $\vec{0}$ vector)

$$\mathbb{X}^T\left(\mathbb{Y} - \mathbb{X}\hat{\theta}\right) = 0$$

Rearrange terms

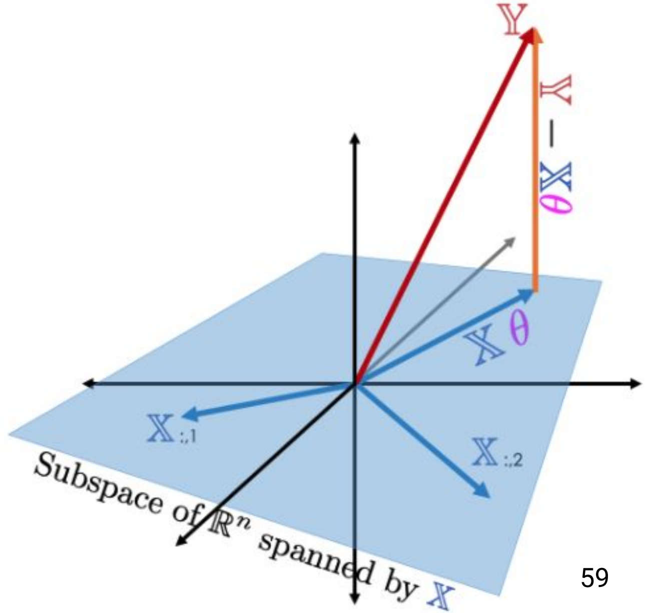
$$\mathbb{X}^T\mathbb{Y} - \mathbb{X}^T\mathbb{X}\hat{\theta} = 0$$

The **normal equation**

$$\mathbb{X}^T\mathbb{X}\hat{\theta} = \mathbb{X}^T\mathbb{Y}$$

If $\mathbb{X}^T\mathbb{X}$ is invertible

$$\boxed{\hat{\theta} = \left(\mathbb{X}^T\mathbb{X}\right)^{-1}\mathbb{X}^T\mathbb{Y}}$$



$\mathbb{Y}$
$\mathbb{Y} - \mathbb{X}\theta$
$\mathbb{X}\theta$
$\mathbb{X}_{:,1}$
$\mathbb{X}_{:,2}$
Subspace of $\mathbb{R}^n$ spanned by $\mathbb{X}$

$$\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$$

This result is so important that it deserves its own slide.

It is the **least squares estimate** and the solution to the normal equation $\mathbb{X}^T\mathbb{X}\hat{\theta} = \mathbb{X}^T\mathbb{Y}$.

2327602

🎉🎉🎉

$$\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$$

This result is so important that it deserves its own slide.

It is the **least squares estimate** and the solution to the normal equation $\mathbb{X}^T\mathbb{X}\hat{\theta} = \mathbb{X}^T\mathbb{Y}$.

Least Squares Estimate

1. Choose a model

Multiple Linear Regression

$$\hat{\mathbb{Y}} = \mathbb{X}\theta$$

2. Choose a loss function

L2 Loss

Mean Squared Error (MSE)

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

3. Fit the model ✅

Minimize average loss with ~~calculus~~ geometry

$$\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$$

4. Evaluate model performance

Visualize,
~~Root MSE~~
Multiple R^2