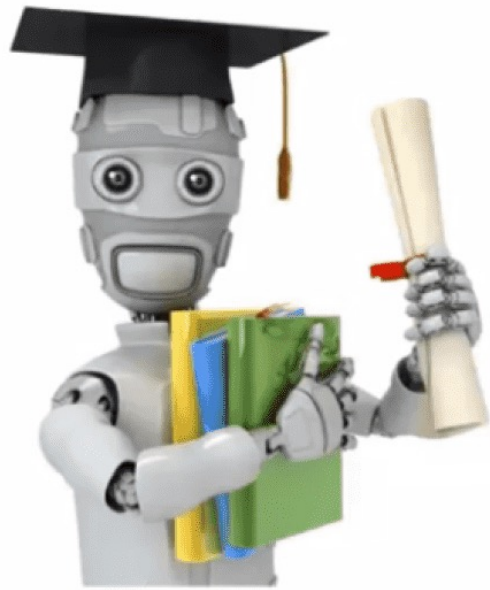


Linear Regression Multiple Variables

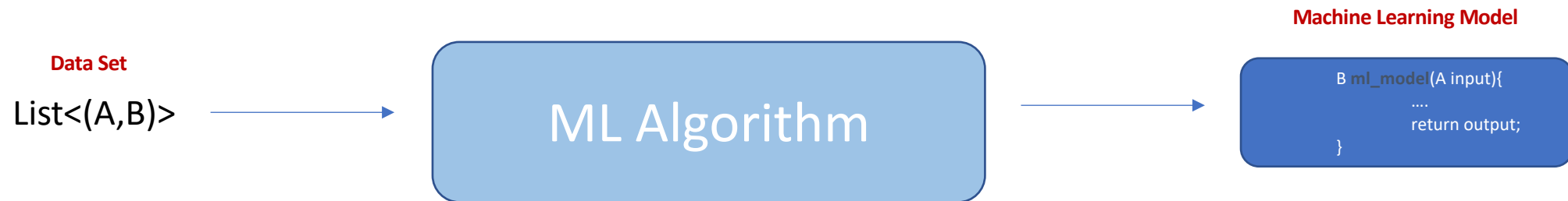
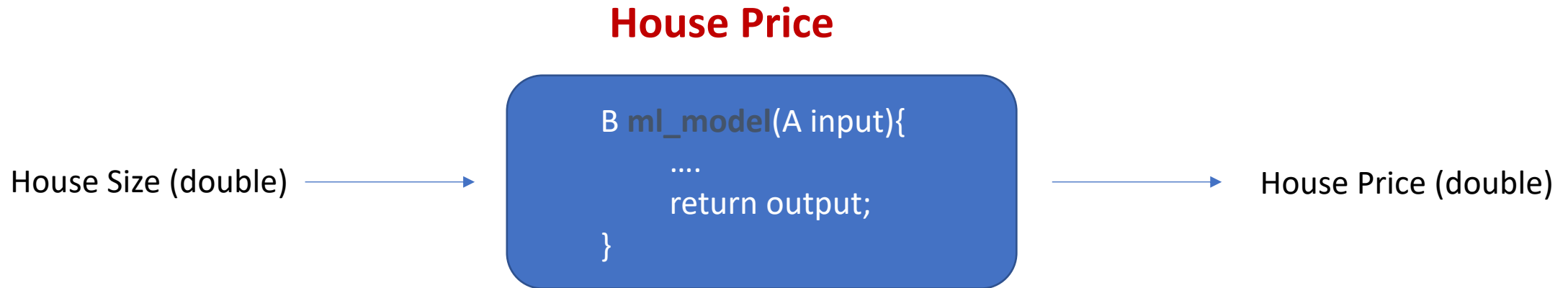
Andrew Ng's Machine Learning Course



Machine Learning



Predict House Price given House Size



Multiple Features

Size in feet ²	Number of bedrooms	Number of floors	Age of home in years	Price (\$) in \$1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

Multiple Features

Size in feet ²	Number of bedrooms	Number of floors	Age of home in years	Price (\$) in \$1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

$x_j = j^{th}$ feature

n = number of features

$\vec{x}^{(i)}$ = features of i^{th} training example

$x_j^{(i)}$ = value of feature j in i^{th} training example

Multi Linear Regression

Model:

$$f_{w,b}(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

Multi Linear Regression

Model:

$$f_{w,b}(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b =$$

Gradient Descent for Multi Linear Regression

Parameters w_1, \dots, w_n
 b

Model $f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + \dots + w_n x_n + b$

Cost function $J(w_1, \dots, w_n, b)$

Gradient descent

```
repeat {  
     $w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \dots, w_n, b)$   
     $b = b - \alpha \frac{\partial}{\partial b} J(w_1, \dots, w_n, b)$   
}
```


Gradient Descent for Multi Linear Regression

One feature

repeat {

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})$$

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$$

}

Gradient Descent for Multi Linear Regression

One feature

repeat {

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})$$

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$$

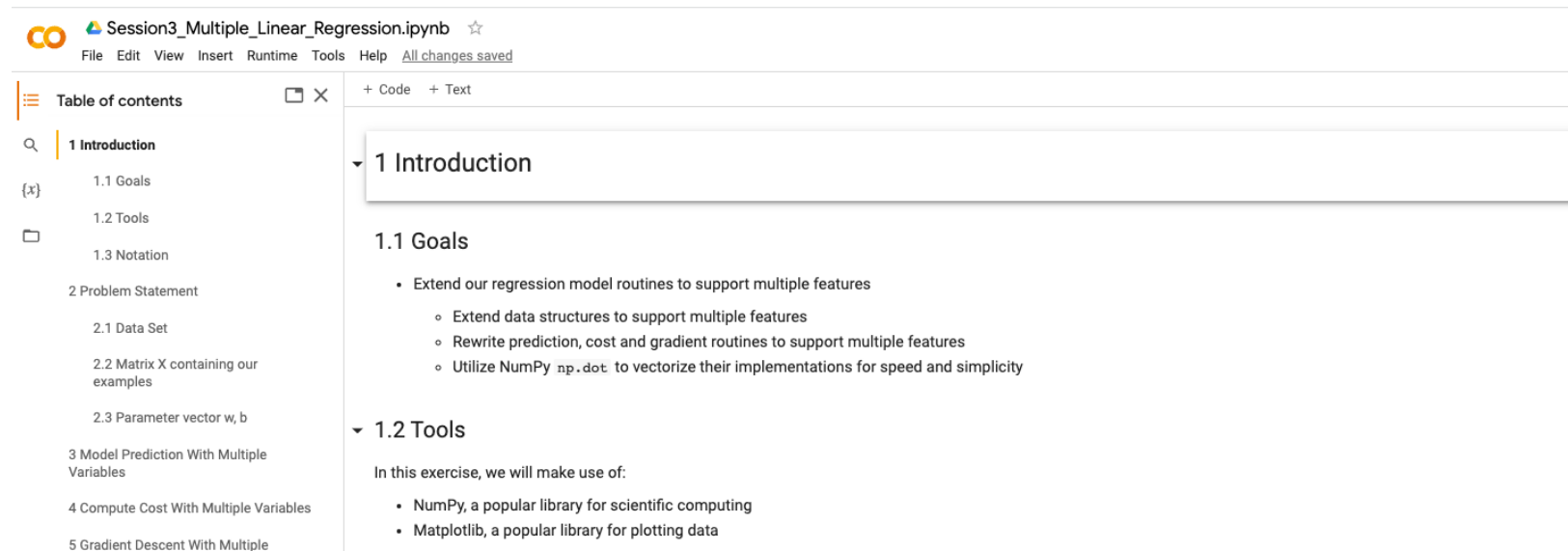
$$w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_n^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$

simultaneously update
 w_j (for $j = 1, \dots, n$) and b

}

Notebook: Multiple Linear Regression



The screenshot displays a Jupyter Notebook titled "Session3_Multiple_Linear_Regression.ipynb". The interface includes a top menu bar with options: File, Edit, View, Insert, Runtime, Tools, and Help. A "Table of contents" sidebar on the left lists the notebook's sections: 1 Introduction, 2 Problem Statement, 3 Model Prediction With Multiple Variables, 4 Compute Cost With Multiple Variables, and 5 Gradient Descent With Multiple. The main content area shows the "1 Introduction" section, which is expanded to reveal "1.1 Goals". Under "1.1 Goals", there is a bulleted list: "Extend our regression model routines to support multiple features", which is further detailed with sub-points: "Extend data structures to support multiple features", "Rewrite prediction, cost and gradient routines to support multiple features", and "Utilize NumPy `np.dot` to vectorize their implementations for speed and simplicity". Below this, the "1.2 Tools" section is partially visible, starting with the text "In this exercise, we will make use of:" followed by a bulleted list: "NumPy, a popular library for scientific computing" and "Matplotlib, a popular library for plotting data".

Session3_Multiple_Linear_Regression.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

Table of contents

- 1 Introduction
 - 1.1 Goals
 - 1.2 Tools
 - 1.3 Notation
- 2 Problem Statement
 - 2.1 Data Set
 - 2.2 Matrix X containing our examples
 - 2.3 Parameter vector w, b
- 3 Model Prediction With Multiple Variables
- 4 Compute Cost With Multiple Variables
- 5 Gradient Descent With Multiple

1 Introduction

1.1 Goals

- Extend our regression model routines to support multiple features
 - Extend data structures to support multiple features
 - Rewrite prediction, cost and gradient routines to support multiple features
 - Utilize NumPy `np.dot` to vectorize their implementations for speed and simplicity

1.2 Tools

In this exercise, we will make use of:

- NumPy, a popular library for scientific computing
- Matplotlib, a popular library for plotting data

Feature Values and Parameter Values

$$\widehat{price} = w_1 x_1 + w_2 x_2 + b$$

x_1 : size (feet²)
range: 300 – 2,000

x_2 : # bedrooms
range: 0 – 5

House: $x_1 = 2000$, $x_2 = 5$, $price = \$500k$

size of the parameters w_1, w_2 ?

$$w_1 = 50, \quad w_2 = 0.1, \quad b = 50$$

$$w_1 = 0.1, \quad w_2 = 50, \quad b = 50$$

Gradient Descent for Multi Linear Regression

n features ($n \geq 2$)

repeat {

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\bar{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\bar{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_n^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\bar{w},b}(\vec{x}^{(i)}) - y^{(i)})$$

}

Exercise 1: Feature Scaling

6 Feature Scaling

6.1 Running Gradient Descent without Feature Scaling

```
✓ ▶ # initialize parameters
initial_w = np.zeros_like(w_init)
initial_b = 0.
# some gradient descent settings
iterations = 10000
alpha = 0.0000001
# run gradient descent
w_final, b_final, J_hist = gradient_descent(X_train, y_train, initial_w, initial_b,
                                             compute_cost, compute_gradient,
                                             alpha, iterations)

print(f"b,w found by gradient descent: {b_final:0.2f},{w_final} ")
m,_ = X_train.shape
for i in range(3):
    print(f"prediction: {np.dot(X_train[i], w_final) + b_final:0.2f}, target value: {y_train[i]}")
```

Exercise 1: Explore the effect of feature scaling in Gradient Descent

Perform the following experiments:

- Observe how the learning rate has different effects before and after normalization.
- Observe how quickly gradient descent converges depending of the learning rate before and after normalization.

Feature Engineering

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$$



Feature Engineering

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$$

$$area = frontage \times depth$$

$$x_3 = x_1 x_2$$

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



Feature engineering:
Using **intuition** to design
new features, by
transforming or combining
original features.

Exercise 2: Feature Engineering

7 Feature Engineering

Let us introduce in our training data sample a new feature "size_per_bedroom":

$$\text{size_per_bedroom} = \frac{\text{size}}{\text{number of bedrooms}}$$

```
✓ 0s # Create a larger matrix
X_train_extended = np.zeros((X_train.shape[0],X_train.shape[1]+1))

# Copy the training data to the new matrix
X_train_extended[:, :-1] = X_train

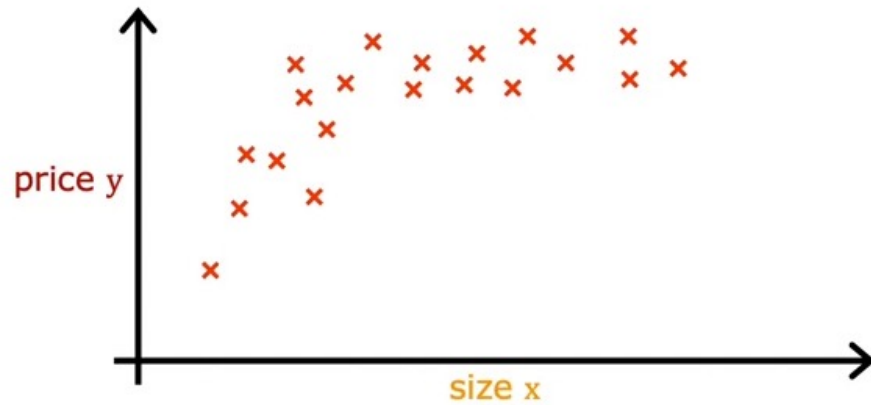
#Add the new colum for "size_per_bedroom"
X_train_extended[:, -1] = X_train[:, 0]/(X_train[:, 1])

#Create new names for the variable
X_features_extended = ['size', 'bedrooms', 'floors', 'age', 'sizer_per_bedroom']
```

Exercise 2: Add a new feature and see what happens

- Add a new feature you think it could help to better predict the price of a house, following the same approach used to add "size_per_bedroom" feature. (i) Justify why did you decided to use this feature? (ii) Explore the training cost of the new solution an compare it with the cost of the previous solutions. Is the cost smaller?

Polyomial Regression



Exercise 3: Polynomial Features

8.1 Adding Polynomial Features

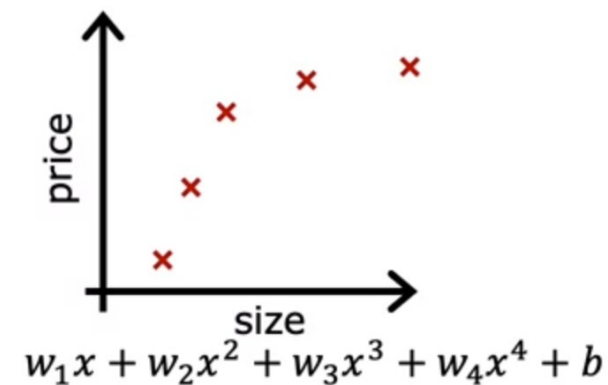
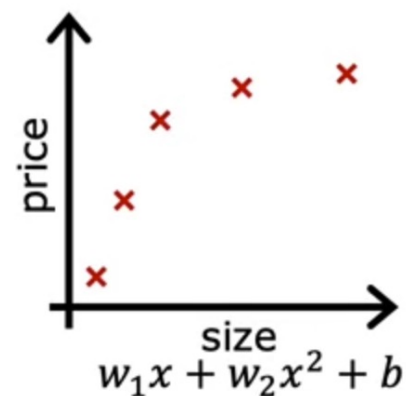
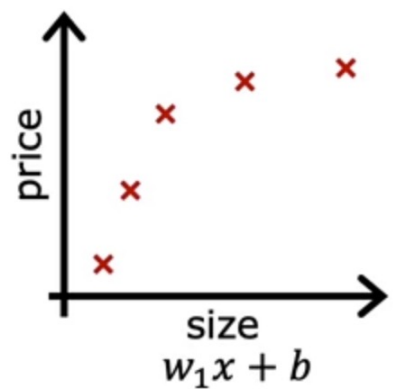
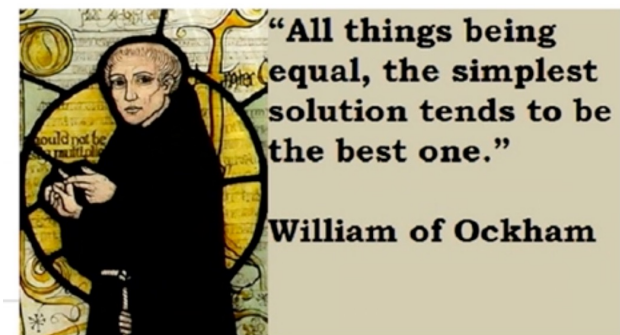
Let us add new polynomial features (i.e. $size^2$, $size^3$, etc) to our training data set and see what happens with the cost function after running gradient descent.

```
# Enter the degree of the new polynomial features.
degree = 2
# Create a larger matrix
X_train, y_train = load_house_data()
X_train, _, _ = zscore_normalize_features(X_train)
X_train_extended = np.zeros((X_train.shape[0], X_train.shape[1]+degree-1))
```

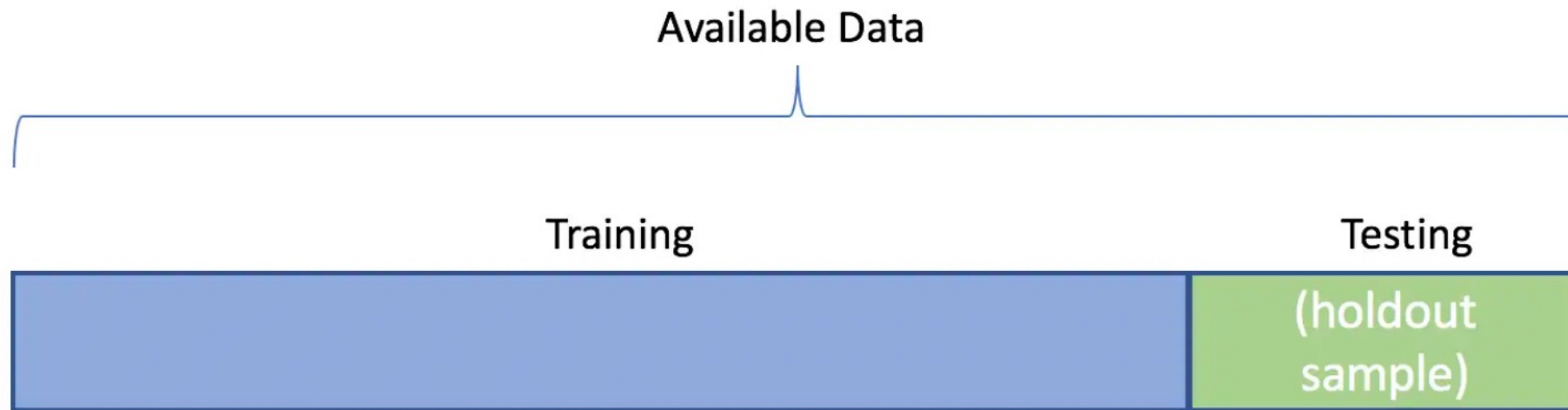
Exercise 3: Polynomial Features

- Add polynomial features using the code above. Start with degree=2 and move to degree=20.
- Look how the Cost function is reduced.
- Do you think we are doing better by having more polynomial functions?

The problem of Overfitting



Measuring Overfitting



Exercise 4: Measuring Overfitting

8.2 Measuring Overfitting

We now split the data set in training data set and validation data set.

```
#Permutate the data
np.random.seed(123)
perm = np.random.permutation(X_train_extended.shape[0])
X_train_extended = X_train_extended[perm,:]
y_train = y_train[perm]

#Split the data
size_train = 66
X_split_train, y_split_train = X_train_extended[0:size_train,:], y_train[0:size_train]
X_split_val, y_split_val = X_train_extended[size_train:,:], y_train[size_train:]
```

Exercise 4: Measuring Training and Test Error

- Add polynomial features using the code above. Start with degree=2 and move to degree=20.
- Look how the Cost function for the training and the test cost evolves.
- Increase and decrease the number of iterations gradient descent is run (move it from 10000 to 20000) and observed what happen with the the training and the test cost.
- What do you think is happening?
- Which should be the degree of the included polynomial features?

Kaggle Competition

Search

Community Prediction Competition

Bike Sharing

Let us see how good you are at making predictions

7 days to go

Overview

Data

Code

Discussion

Leaderboard

Rules

Join Competition

...

Overview

Description	<p>Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.</p>
Evaluation	<p>The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.</p>