Linear Regression

APT 3025: APPLIED MACHINE LEARNING

Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

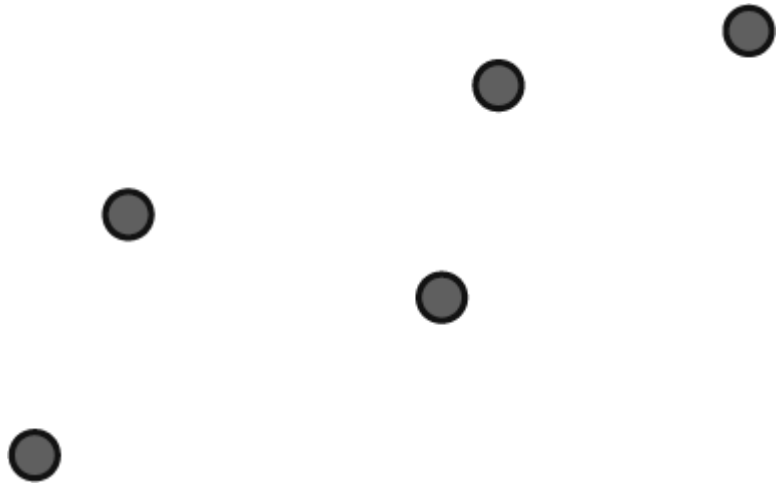
Lecture Overview

- What is linear regression?
- Fitting a line through a set of data points
- Error functions
- Gradient descent
- Using Scikit Learn and a real dataset to build a linear regression model to predict housing prices

Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

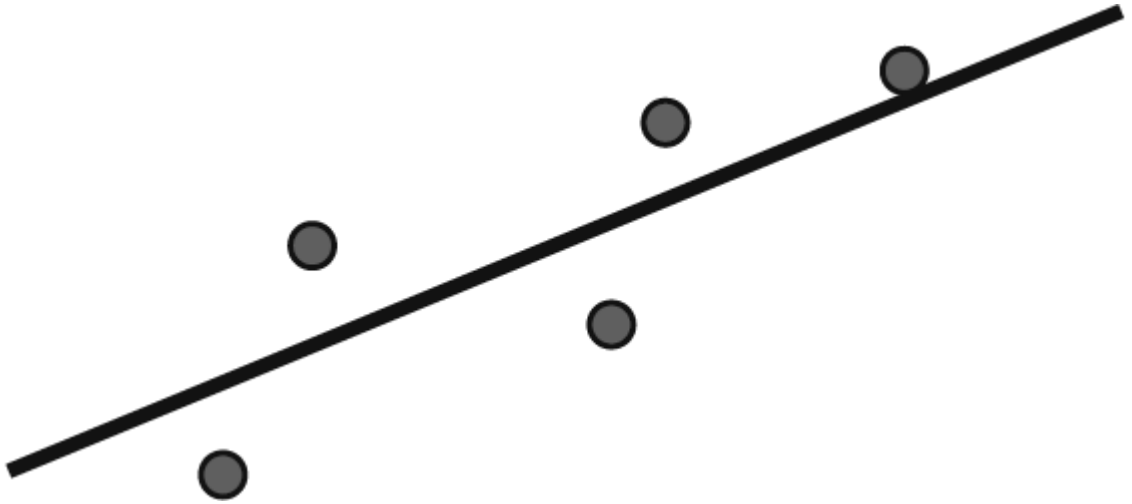
What is linear regression?

- Suppose we have some points that look like the form a line such as these:



What is linear regression?

- Here we see a line drawn through the points so that it’s as close to each point as possible.



Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Building a regression model for housing prices

- In a real housing price prediction problem, features might include size, number of rooms, location, crime rate, school quality, and distance to commerce.
- We will look at a much simpler example that uses only one input feature, namely, the number of rooms.

Toy housing price dataset

Number of rooms	Price
1	150
2	200
3	250
4	?
5	350
6	400
7	450

Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Deriving a model

- We see a pattern in the dataset and can predict a price of $300 for the 4-room house.
- We can think of the price of a house as a combination of two things: a base price of $100, and an extra charge of $50 for each of the rooms.
- This can be summarized in a simple formula:
  - Price = 100 + 50(Number of rooms)
- This formula is a model that gives us a prediction of the price of the house, based on the feature (the number of rooms).

Weights and bias

- The price per room is called the *weight* of that corresponding feature, and the base price is called the *bias* of the model.
- The weight associate with a feature indicates the importance of that feature in predicting the label.
- The bias is a component in the model that is not associated with any feature.

Slope and y-intercept

- The *slope* of a line is a measure of how steep it is. It is calculated by dividing the rise over the run (change in y over change in x)
- In a machine learning model, *the slope is the weight*.
- The y-intercept of a line is the height at which the line crosses the y-axis.
- In a machine learning model, the y-intercept is the bias; it tells us what the label would be in a data point where all the features are zero.

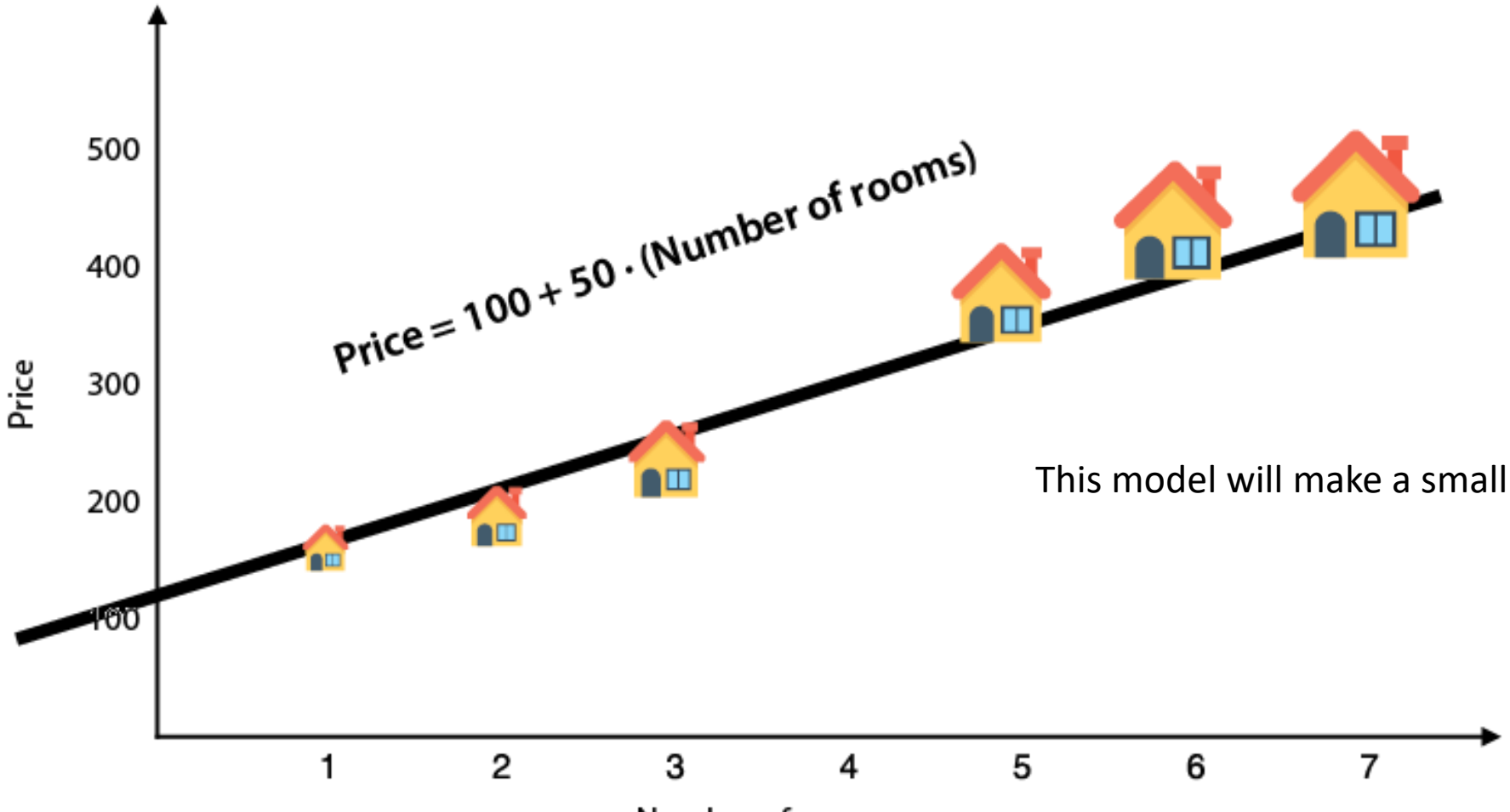
Equation of a line

- The equation of a line is given by two parameters: the slope and the y-intercept.
- If the slope is m and the y-intercept is b, then the equation of the line is $y = mx + b$.
- In machine learning (specifically linear regression), x is the feature and y is the prediction.
- In linear regression, the equation $y = mx + b$ is the model that is learned from data.

A more realistic dataset

Number of rooms	Price
1	155
2	197
3	244
4	?
5	356
6	407
7	448

The model



This model will make a small error

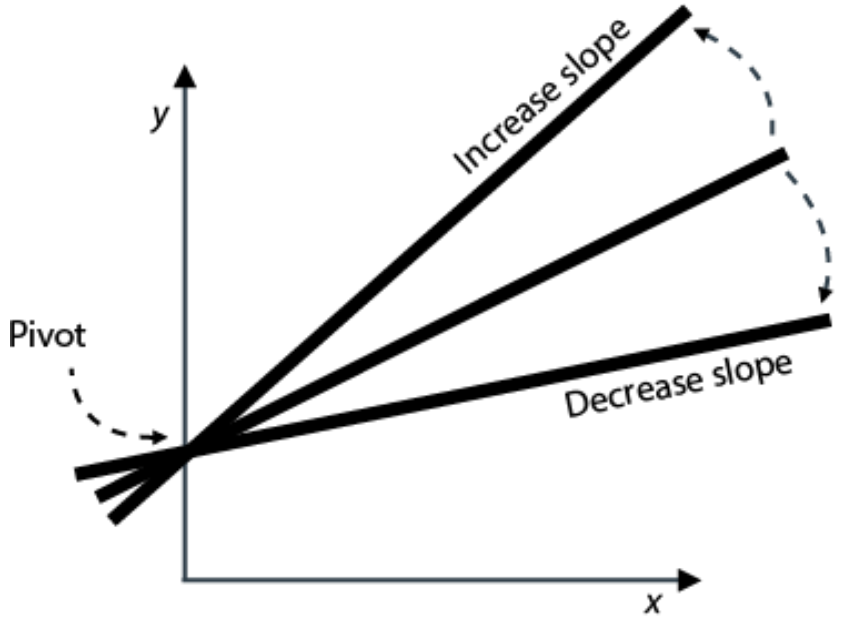
A model using many features

- If we have more features, all we need to do is multiply them by their corresponding weights and add them to the predicted price.
- The model could look like this:
- Price = 30(number of rooms) + 1.5(size) + 10(quality of the schools) – 2(age of the house) + 50
- Notice age is negatively correlated while the other three features are positively correlated to the price.
- If the weight of a feature is 0, it means that feature is irrelevant.

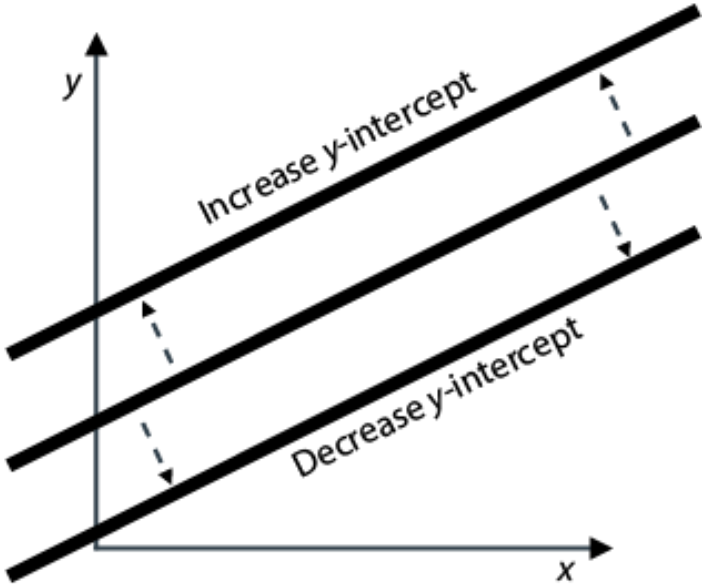
The linear regression algorithm

- The linear regression algorithm searches for the line that best fits the data
- It starts out with a random line (defined by random values for the weight and the bias).
- This line is then repeatedly adjusted by slightly adding to or subtracting from the wight and the bias.
- The prediction error guides the algorithm in adjusting the line.

Adjusting the line



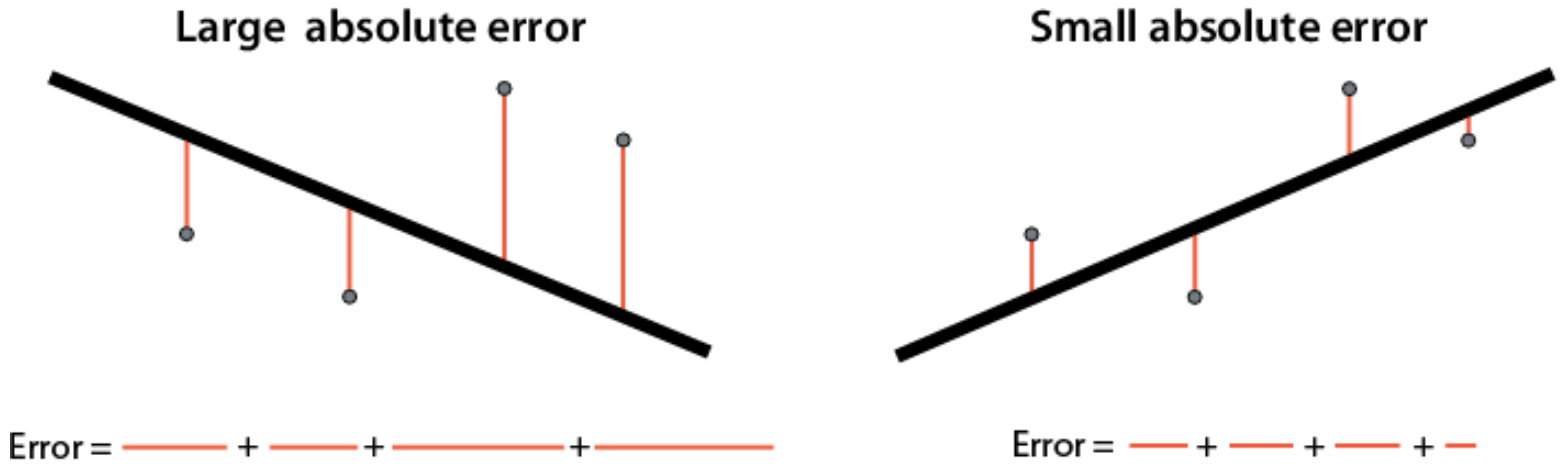
Rotate clockwise and counterclockwise



Translate up and down

Absolute error

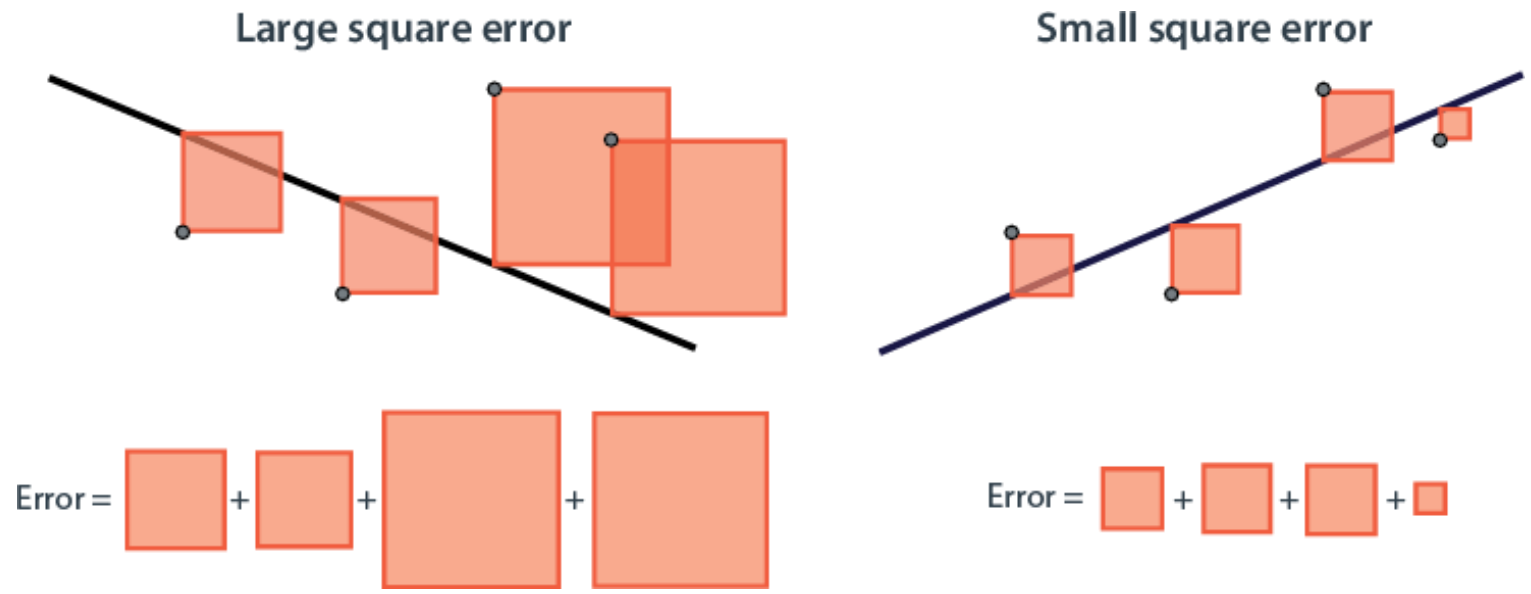
- An error function is a metric that tells us how our model is doing.
- The *absolute error* is the sum of the distances between the data points and the line.
- Linear regression looks for the line that minimizes the error function.



Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Square error

- The square error is very similar to the absolute error, except instead of taking the absolute value of the difference between the label and the predicted label, we take the square.



Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Mean absolute and mean square error

- The mean absolute error and the mean square error much more commonly used in practice.
- These are defined in a similar way, except instead of calculating sums, we calculate averages.

Root mean square error

- Another error commonly used is the root mean square error, or RMSE for short.
- This is defined as the root of the mean square error.
- It is used to match the units in the problem and also to give us a better idea of how much error the model makes in a prediction.

Root mean square error

- Imagine the following scenario: if we are trying to predict house prices, then the units of the price and the predicted price are, for example, dollars ($).
- The units of the square error and the mean square error are dollars squared, which is not a common unit.
- If we take the square root, then not only do we get the correct unit, but we also get a more accurate idea of roughly by how many dollars the model is off per house.
- Say, if the root mean square error is $10,000, then we can expect the model to make an error of around $10,000 for any prediction we make.

Gradient descent

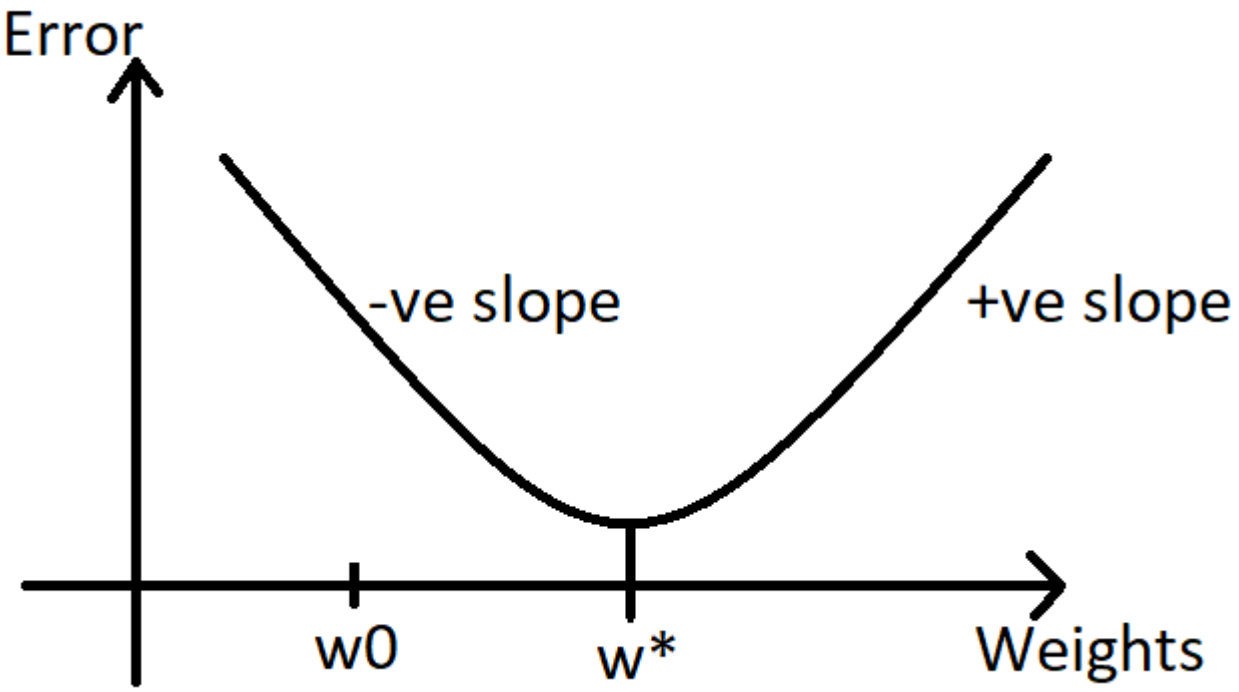
- Gradient descent is the method used by linear regression to find the model that minimises the prediction error.
- This is done by adjusting the weights and bias so that the error function is minimized.
- This is what training is in linear regression: adding something small to or subtracting something small from the weights and bias in each iteration.
- Gradient descent makes sure that we dd when the weight is too small and we subtract when the weight is too big.

Gradient descent

Gradient descent defines the change in weight as

$$\Delta w = -\eta(\text{slope}),$$

where η is a small number known as the learning rate.



Stopping conditions for gradient descent

- The time and the computational power available
- When the loss function reaches a certain value that we have predetermined
- When the loss function doesn’t decrease by a significant amount during several epochs

Stochastic and batch gradient descent

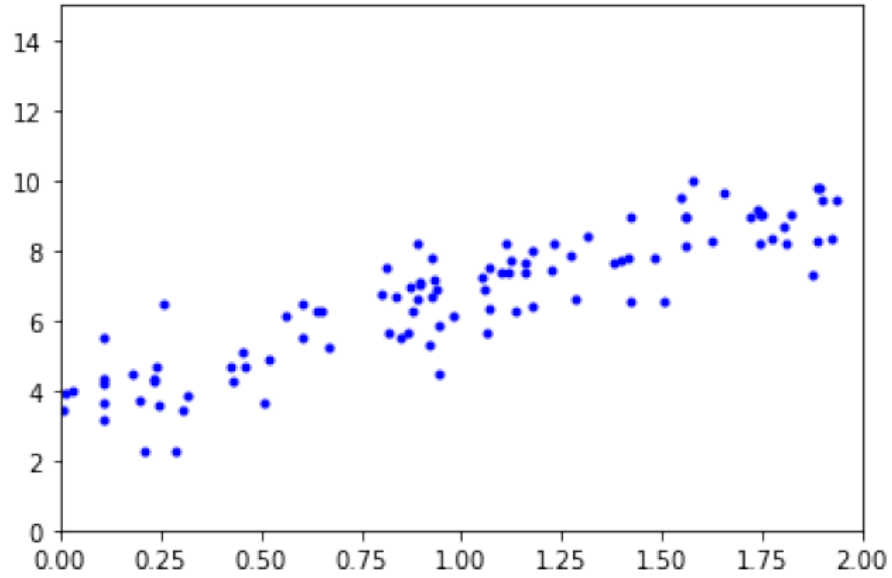
- In *stochastic gradient descent*, we pick a point at random and use it to adjust the model. Then we pick another and repeat. This continues until a stopping condition is reached.
- In *batch gradient descent*, we train the model based on the entire dataset in each iteration.
- Both the above options can be computationally expensive. A useful approach is mini-batch learning.
- In *mini-batch gradient descent*, we split the dataset into subsets (mini-batches) and train on these subsets in each iteration.

Linear Regression Example with Synthetic Data

- First, let's generate a fictitious dataset and plot it.

```
[1]: import numpy as np
     X = 2 * np.random.rand(100, 1)
     y = 4 + 3 * X + np.random.randn(100, 1)
```

```
[2]: import matplotlib.pyplot as plt
     plt.plot(X, y, "b.")
     plt.axis([0, 2, 0, 15])
     plt.show()
```



Notes taken from Introduction to Machine Learning with Python by Andreas C. Müller & Sarah Guido, O'Reilley Media, 2016

Fitting a Line to the Data

- Scikit learn defines the class LinearRegression in its linear_model module.
- We create an object of this class and then call its fit method, passing in the inputs X and the outputs y from out dataset.
- The predict method generates the estimated values of y (i.e. the points on the estimated line).

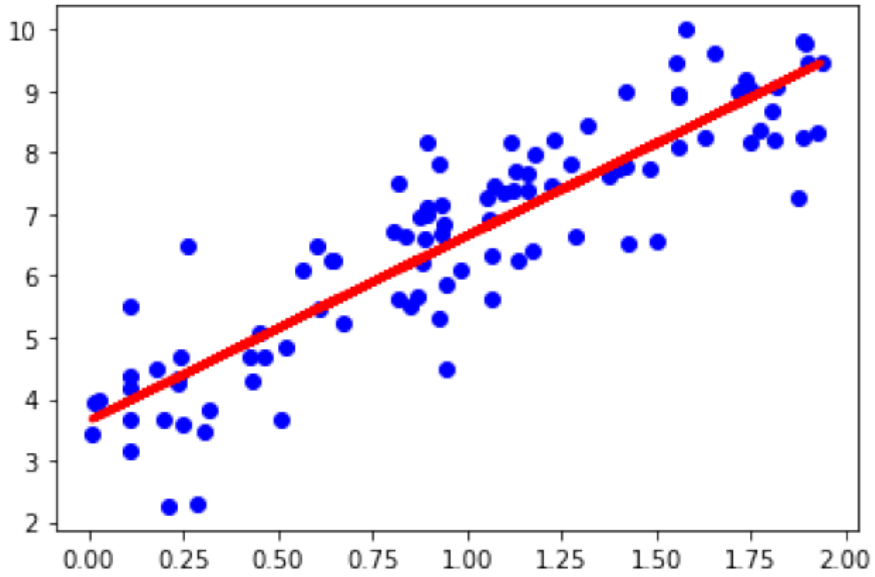
```
[4]: from sklearn.linear_model import LinearRegression
     lin_reg = LinearRegression()
     lin_reg.fit(X, y)
     y_pred = lin_reg.predict(X)
```

Notes taken from Introduction to Machine Learning with Python by Andreas C. Müller & Sarah Guido, O'Reilley Media, 2016

Plotting the Estimated Line

- We can plot the line that has been found and verify that indeed it is a good enough fit.

```
plt.scatter(X, y, color="blue")
plt.plot(X, y_pred, color="red", linewidth=3)
```



Notes taken from Introduction to Machine Learning with Python by Andreas C. Müller & Sarah Guido, O'Reilley Media, 2016

Predicting for a New Input

```
[14]: X_new = np.array([[0], [1.5], [2]])
      lin_reg.predict(X_new)
```

```
[14]: array([[3.64698866],
             [8.14771687],
             [9.6479596 ]])
```

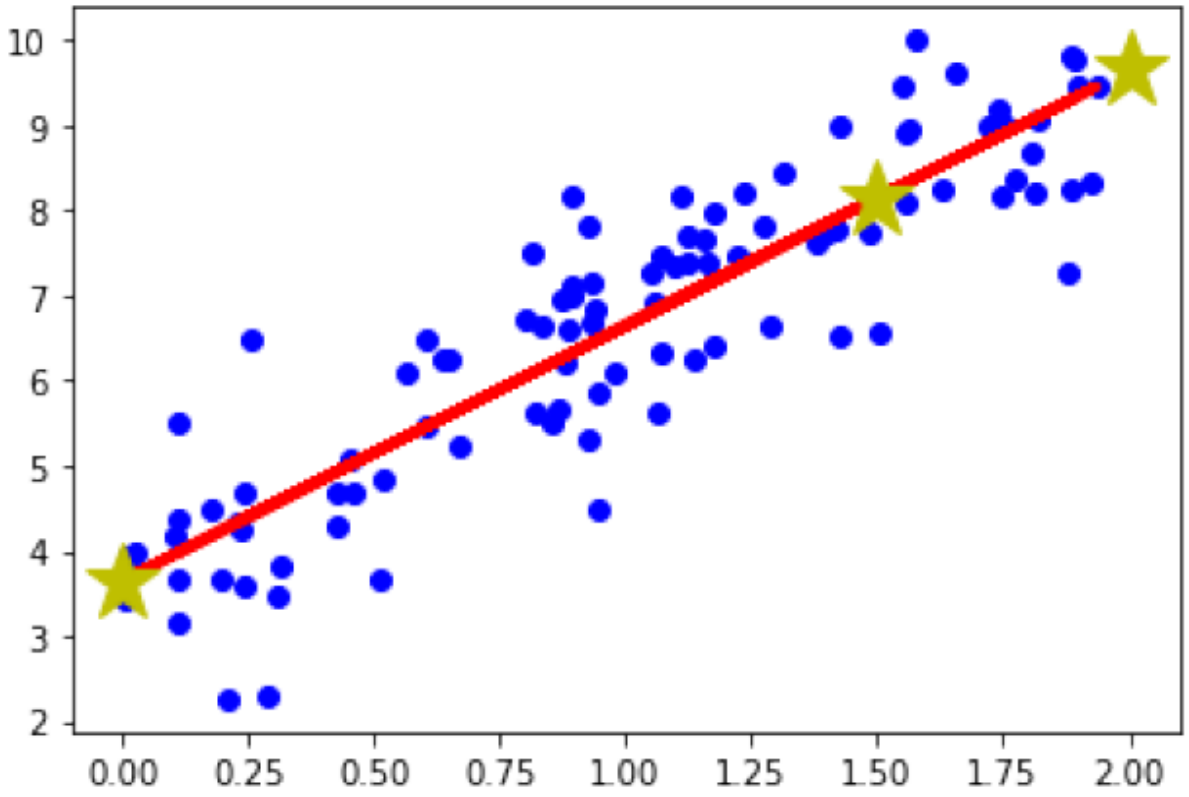
```
[16]: plt.scatter(X, y, color="blue")
      plt.plot(X, y_pred, color="red", linewidth=3)
      plt.plot(X_new, lin_reg.predict(X_new), "y*", markersize=24)
      plt.xticks()
      plt.yticks()
      plt.show()
```

Notes taken from Introduction to Machine Learning with Python by Andreas C. Müller & Sarah Guido, O'Reilley Media, 2016

Plotting Predictions for New Input

- The yellow stars show the y values predicted for three points: x = 0, x = 1.5 and x = 2.

```
[14]: array([[3.64698866],
             [8.14771687],
             [9.6479596 ]])
```



Notes taken from Introduction to Machine Learning with Python by Andreas C. Müller & Sarah Guido, O'Reilley Media, 2016

Linear Regression Example with Real Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('../data/Hyderabad.csv')
data.head()
```

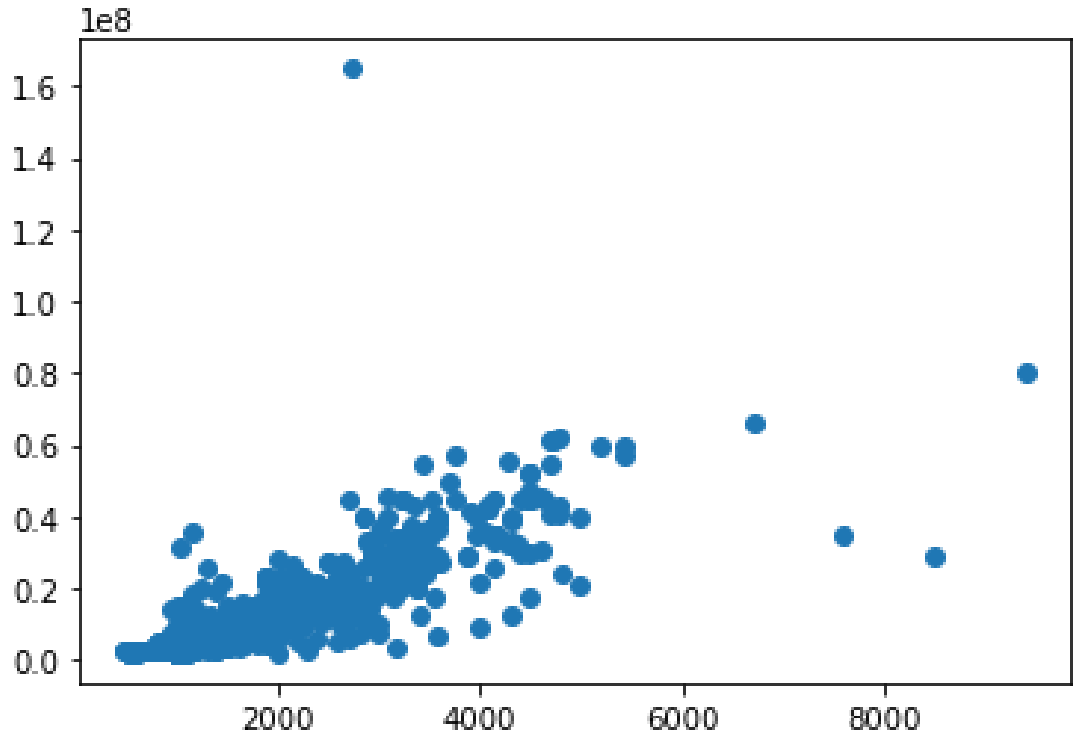
	Price	Area	Location	No. of Bedrooms	Resale	MaintenanceStaff	Gymnasium	SwimmingPool
0	6968000	1340	Nizampet	2	0	0	1	1
1	29000000	3498	Hitech City	4	0	0	1	1
2	6590000	1318	Manikonda	2	0	0	1	0
3	5739000	1295	Alwal	3	1	0	0	0
4	5679000	1145	Kukatpally	2	0	0	0	0

5 rows × 40 columns

Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Plot Area against Price

```
plt.scatter(data.Area, data.Price)
plt.show()
```



Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Split and Train

```
X = data[['Area']] # We are using the 'Area' feature only
y = data['Price']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

```
from sklearn.linear_model import LinearRegression
simple_model = LinearRegression()
simple_model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
print("Weights:", simple_model.coef_)
print("Bias:", simple_model.intercept_)
```

```
Weights: [9667.24659095]
Bias: -6103109.844015583
```

Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Evaluate using test data

```
from sklearn.metrics import mean_squared_error
test_pred = simple_model.predict(X_test)
mean_squared_error(y_test, test_pred, squared=False)
```

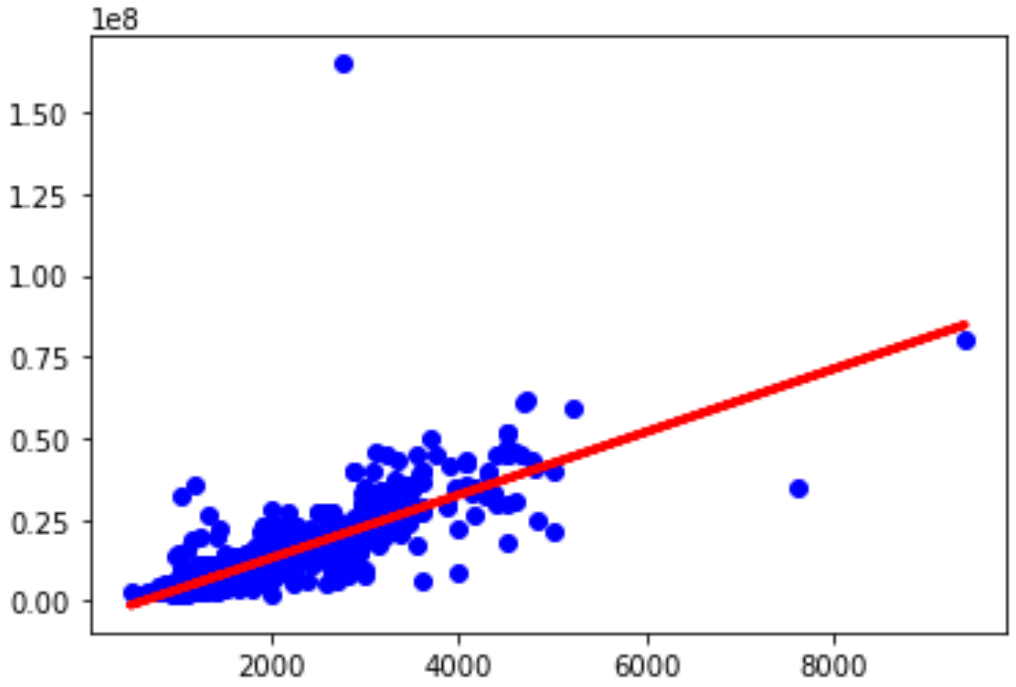
```
4518306.06476008
```

```
train_pred = simple_model.predict(X_train)
```

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, train_pred, color='red', linewidth=3)
plt.show()
```

Plot predicted model (line)

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, train_pred, color='red', linewidth=3)
plt.show()
```



Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Predict price for a new house

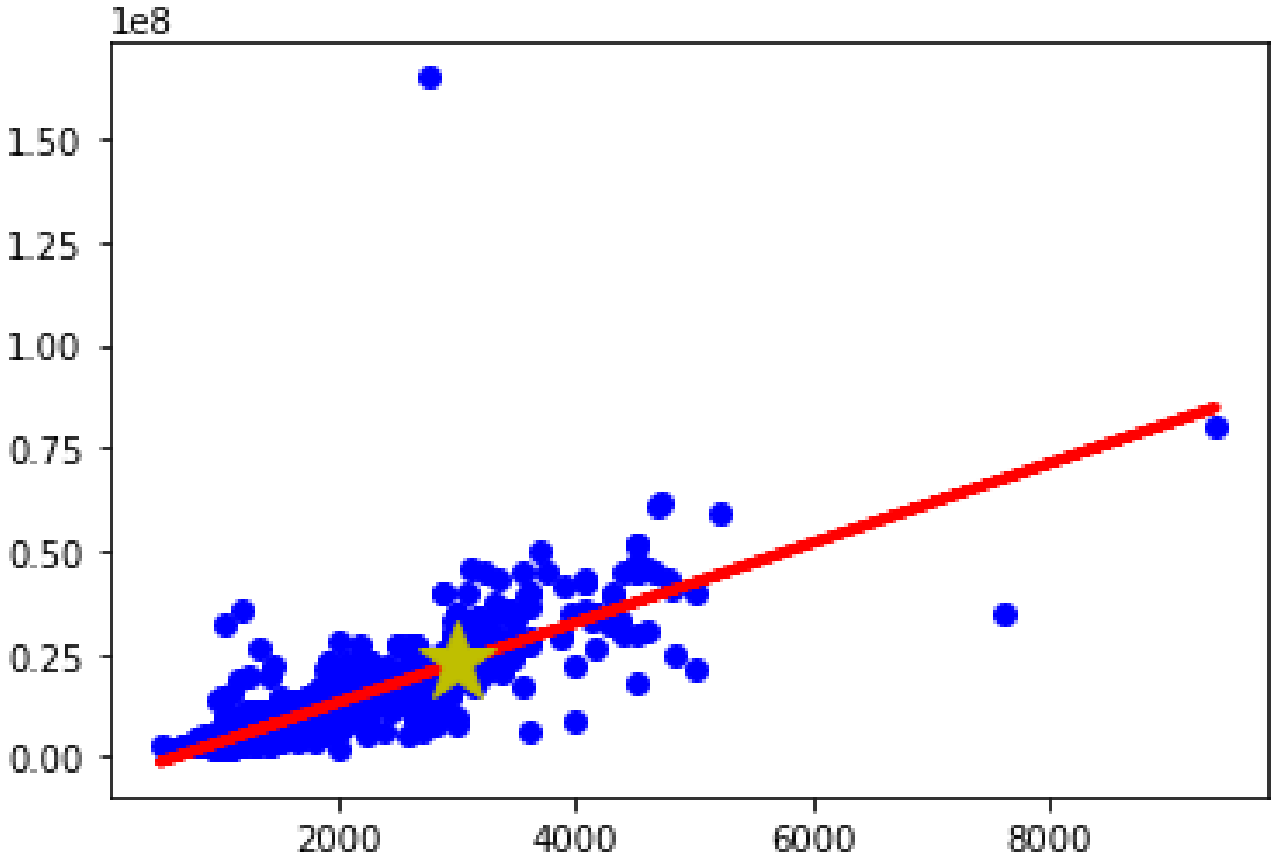
```
X_new = np.array([[3000]])
pred_new = simple_model.predict(X_new)
print(pred_new)
```

```
[22898629.92884066]
```

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, train_pred, color='red', linewidth=3)
plt.plot(X_new, pred_new, 'y*', markersize=24)
plt.show()
```

Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning

Plot predicted point



Notes taken from Serrano, L.G. (2021). Grokking Machine Learning. Manning