

Machine learning - Linear regression model

Background:

The science which focuses on teaching machines or computers to perform certain tasks without giving specific instructions. We are equipping machines to learn to do something themselves without explaining it to them.

Types of Machine learning:

- Supervised
- Unsupervised
- Semi-supervised
- Reinforcement

Choosing the right type of algorithm depends on the data we want to predict.

This report covers the various supervised machine learning models that can be implemented.

In supervised learning, we provide the algorithm with the labelled training data and define variables we want the algorithm to assess for correlations.

Working:

Algorithm is trained with labelled inputs and desired outputs. Useful for following:

- Binary Classification: division of data into two categories
- Multi-class Classification: Choosing between more than 2 types of answers
- Ensembling: Combining the predictions of multiple ML models to produce an accurate prediction
- Regression modeling: Predicting values

Here, we will be discussing about the regression modeling linear regression:

Q1: For task 1, we need to compute the hypothesis function and gradient descent. In the calculate_hypothesis function, following lines of codes is added to compute it:

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1$$

```
hypothesis = X[i, 0]*theta[0] + X[i, 1]*theta[1]
```

With this we are multiplying the $\theta[0]$ with $X[0]$ and similarly for $X[1]$ and $\theta[1]$.

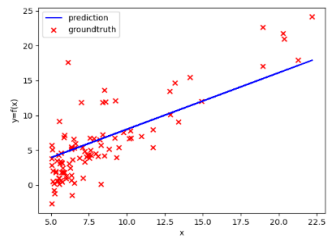
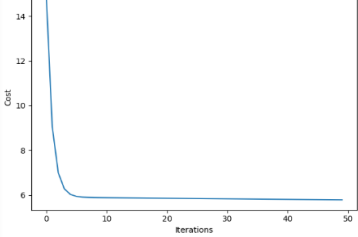
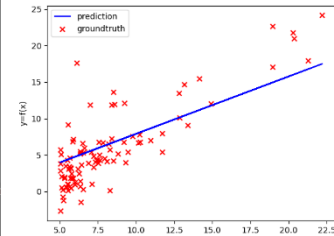
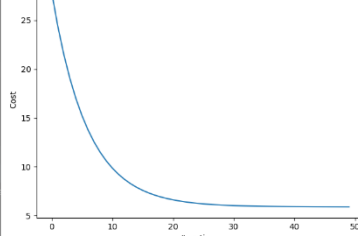
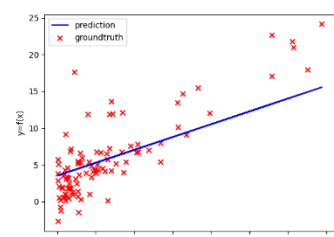
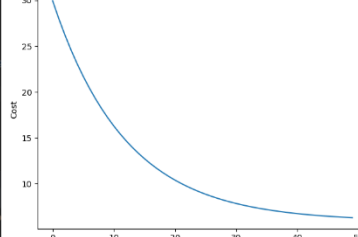
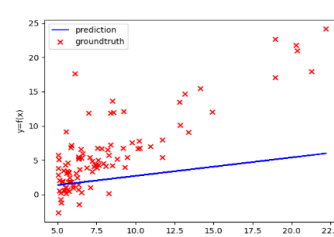
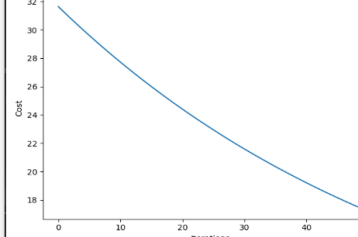
After the calculation of hypothesis function, we need to compute the gradient descent for which the function: gradient_descent should be altered. The following lines of code are added to the function:

- Here we import the function calculate_hypothesis as shown below.

```
hypothesis = calculate_hypothesis(X, theta, i)
```

After computing the gradient descent, we check the outputs – Y and cost predictions for different learning rates, i.e.: alpha in the code. We check the outputs for following learning rates: 1.0, 0.001, 0.005, 0.0005, 0.0001

Learning rate (α)	Output	Image
1.0	Minimum cost: 172570.09522, on iteration #1	

0.005	Minimum cost: 5.79176, on iteration #50	 
0.001	Minimum cost: 5.89503, on iteration #50	 
0.0005	Minimum cost: 6.29590, on iteration #50	 
0.0001	Minimum cost: 17.36882, on iteration #50	 

From the above graphs, there have been variations in the graph plots. It can be noticed that for $\alpha = 1.0$, the groundtruth = 0.0 and predictions showed a decreasing line. While for the other values of learning rates the linear straight line had a positive intercept.

Q2: For task 2, we need to compute the hypothesis function and gradient descent. In the calculate_hypothesis function, following lines of codes is added to compute it:

```
hypothesis = sum(X[i][j]*theta[j] for j in range(len(theta)))
```

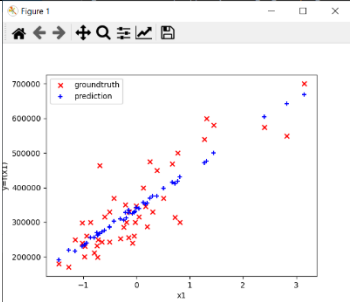
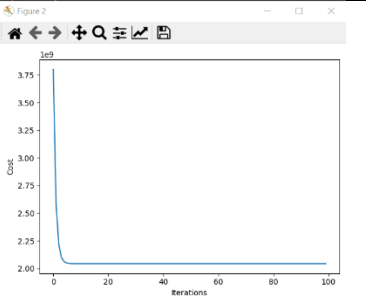
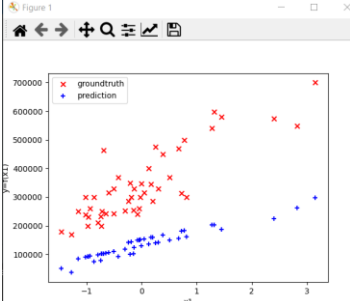
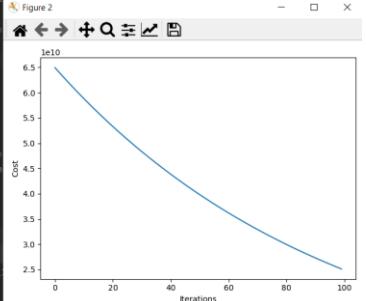
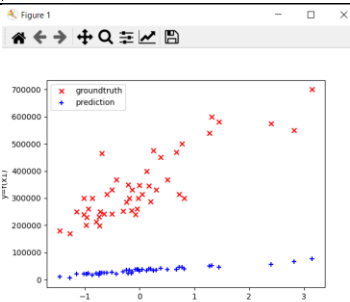
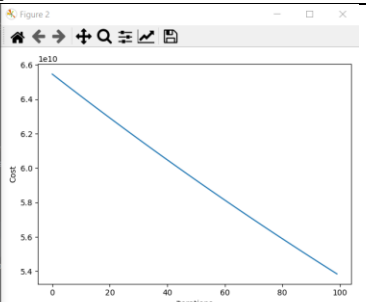
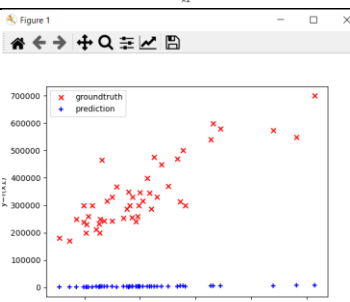
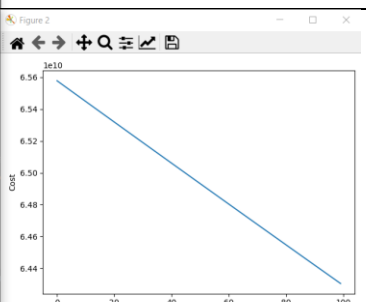
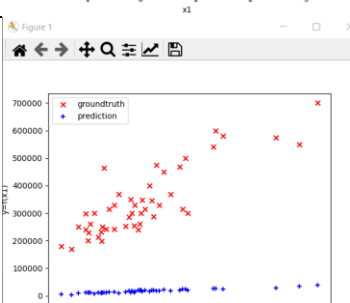
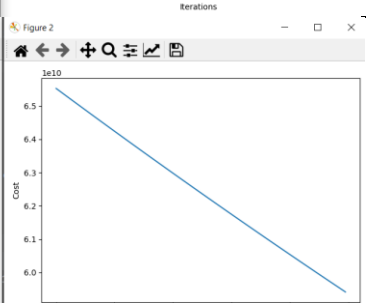
With this we are multiplying the $\theta[0]$ with $X[0]$ and similarly for $X[1]$, $X[2]$ and $\theta[1]$, $\theta[2]$.

After the calculation of hypothesis function, we need to compute the gradient descent for which the function: gradient_descent should be altered. The following lines of code are added to the function:

- Here we import the function calculate_hypothesis as shown below.

```
hypothesis = calculate_hypothesis(X, theta, i)
```

After computing the gradient descent, we check the outputs – Y and cost predictions for different learning rates, i.e.: α in the code. We check the outputs for following learning rates: 1.0, 0.001, 0.0005, 0.0001

Learning rate(α)	Output	Image
1.0	Minimum cost: 2043280050.60283, on iteration #48 0 : -102869.44161548054 1 : 102869.44161548054	 
0.005	Minimum cost: 25120203555.65904, on iteration #100 0 : -55856.67277413688 1 : 55856.67277413688	 
0.001	Minimum cost: 53852390240.30811, on iteration #100 0 : -14868.97604277176 1 : 14868.97604277176	 
0.0001	Minimum cost: 64304438675.50513, on iteration #100 0 : -1592.4009987892669 1 : 1592.4009987892669	 
0.0005	Minimum cost: 59416417935.17419, on iteration #100 0 : -7721.607341692736 1 : 7721.607341692736	 

From the above graphs, there have been variations in the graph plots. It can be noticed that for $\alpha = 1.0$, 0.005 show a cost function graph with curve while the other values have a steeper negative line. It can be seen from the graphs that with the increase in values of learning rates the predictions moved closer to the groundtruth values.

Q3: For task 3, we need to compute the hypothesis function and gradient descent. In the calculate_hypothesis function, following lines of codes is added to compute it:

```
hypothesis = X[i, 0] * theta[0] + X[i, 1] * theta[1] + (X[i, 2]**2) *
theta[2] + (X[i, 3]**3) * theta[3] + (X[i, 4]**4) * theta[4] + (X[i, 5]**5)
* theta[5]
```

With this we are multiplying the $\theta[0]$ with $X[0]$ and similarly for $X[1]$, $X[2]$ and $\theta[1]$, $\theta[2]$.

After the calculation of hypothesis function, we need to compute the gradient descent for which the function: `gradient_descent` should be altered. The following lines of code are added to the function:

- Here we import the function `calculate_hypothesis` as shown below.

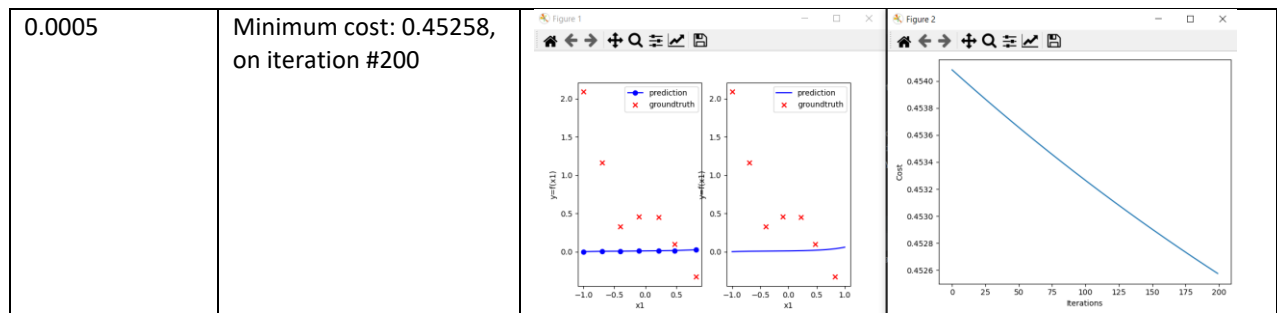
```
hypothesis = calculate_hypothesis(X, theta, i)
```

- Computation of sigma also changes:

```
sigma = sigma + ((hypothesis - output) * (X[i, 0] + X[i, 1] + X[i, 2] +
X[i, 3] + X[i, 4] + X[i, 5]))
```

After computing the gradient descent, we check the outputs – Y and cost predictions for different learning rates, i.e.: α in the code. We check the outputs for following learning rates: 1.0, 0.001, 0.005, 0.0005, 0.0001

Learning rate(α)	Output	Image
1.0	Minimum cost: 0.44596, on iteration #1	
0.005	Minimum cost: 0.44854, on iteration #200	
0.001	Minimum cost: 0.45148, on iteration #200	
0.0001	Minimum cost: 0.45375, on iteration #200	



From the above graphs, there have been variations in the graph plots. It can be noticed that for $\alpha = 1.0$, 0.005 show a cost function graph with curve while the other values have a steeper negative line.