Cara Evangeline Chandra Mohan
190539421

MACHINE LEARNING ASSIGNMENT - 1 [Part -1]

1.)Modify the function calculate_hypothesis.py to return the predicted value for a single specified training example. Include in the report the corresponding lines from your code.

calculate_hpothesis.py

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

The above is the code included in calculate_hypothesis.py
ie. $y = x_0w_0 + x_1w_1$ where $x_0 = 1$

2.)The hypothesis function is not being used in the gradient_descent function. Modify it to use the calculate_hypothesis function. Include the corresponding lines of the code in your report.

gradient_descent.py

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

A call to calculate_hypothesis() is directly made from gradient_descent.py using above code.

3.)Observe what happens when you use a very high or very low learning rate. Document and comment on your findings in the report.

Using a very high learning rate is increasing the cost drastically
A very low learning rate like alpha = 0.001 is giving a less cost, so choosing a lesser value of learning rate is better.
When learning rate = 0.023 there is a very less error rate as shown below in the figure.
The minimum cost is 5.42692 for alpha = 0.023

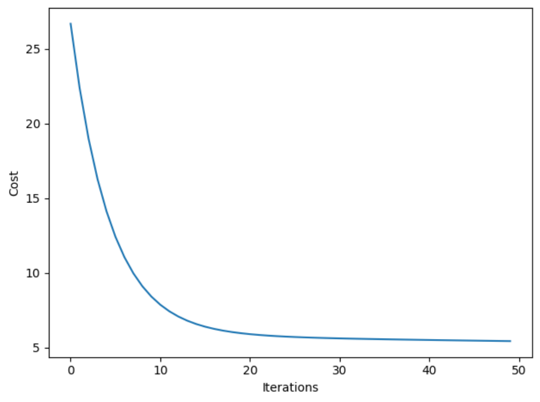


Fig 1:- Cost Graph

Cara Evangeline Chandra Mohan
190539421

4.)Modify the functions calculate_hypothesis and gradient_descent to support the new hypothesis function. Your new hypothesis function's code should be sufficiently general so that we can have any number of extra variables. Include the relevant lines of the code in your report.

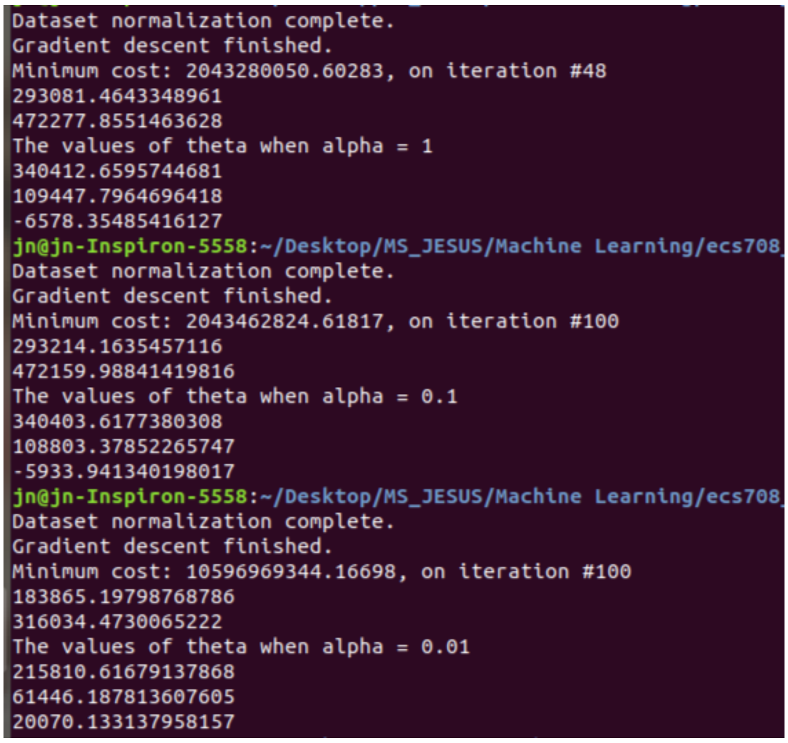
calculate_hypothesis.py

```
hypothesis = 0.0
l = len(theta)
for j in range(l):
    hypothesis = hypothesis + X[i, j]
```

gradient_descent.py

```
theta_temp = theta.copy()
sigma = np.zeros((len(theta)))
for j in range(len(theta)):
    for i in range(m):
        hypothesis = calculate_hypothesis(X, theta, i)
         output = y[i]
     sigma[j] = sigma[j] + (hypothesis - output) * X[i, j]
theta_temp[j] = theta_temp[j] - (alpha/m) * sigma[j]
theta = theta_temp.copy()
```

5.)Run ml_assgn1_2.py and see how different values of alpha affect the convergence of the algorithm. Print the theta values found at the end of the optimization. Does anything surprise you? Include the values of theta and your observations in your report.



```
Dataset normalization complete.
Gradient descent finished.
Minimum cost: 2043280050.60283, on iteration #48
293081.4643348961
472277.8551463628
The values of theta when alpha = 1
340412.6595744681
109447.7964696418
-6578.35485416127
jn@jn-Inspiron-5558:~/Desktop/MS_JESUS/Machine Learning/ecs708
Dataset normalization complete.
Gradient descent finished.
Minimum cost: 2043462824.61817, on iteration #100
293214.1635457116
472159.98841419816
The values of theta when alpha = 0.1
340403.6177380308
108803.37852265747
-5933.941340198017
jn@jn-Inspiron-5558:~/Desktop/MS_JESUS/Machine Learning/ecs708
Dataset normalization complete.
Gradient descent finished.
Minimum cost: 10596969344.16698, on iteration #100
183865.19798768786
316034.4730065222
The values of theta when alpha = 0.01
215810.61679137868
61446.187813607605
20070.133137958157
```

Fig 2:- Values of theta for different values of alpha

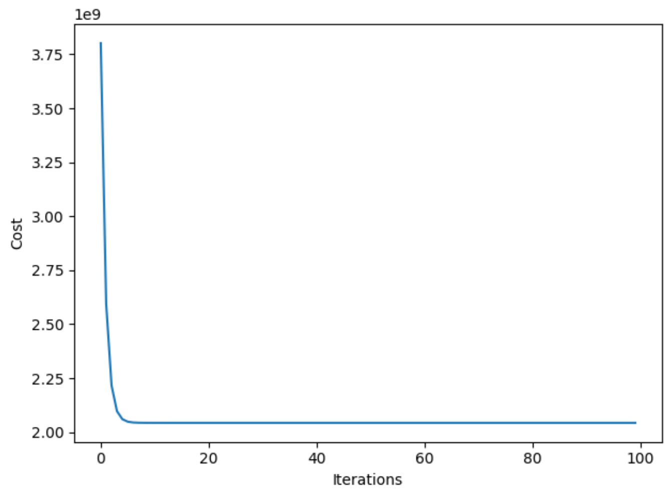


Fig 3:- Cost graph for alpha = 1

From Fig 2, it is clear that when the value of alpha is decreasing, w_2 is becoming positive (ie.increasing) and w_0 and w_1 are decreasing. The value of minimum cost is also increasing as alpha is decreasing.

6.)Add some lines of code in ml_assgn1_2.py to make predictions of house prices. Add the lines of the code that you wrote in your report. Include as well the predictions that you make for the prices of the houses above.

ml_assgn1_2.py

```
alpha = 1
x1 = [1650, 3]
x2 = [3000, 4]
x1 = (x1 - mean_vec)/std_vec
x2 = (x2 - mean_vec)/std_vec
y1 = theta_final[0] + theta_final[1] * x1[0,0] + theta_final[2] * x1[0,1]
y2 = theta_final[0] + theta_final[1] * x2[0,0] + theta_final[2] * x2[0,1]
print(y1)
print(y2)
```

For 1650 sq. ft. and 3 bedrooms the cost is : 293081.4643348961
For 3000 sq. ft. and 4 bedrooms the cost is : 472277.8551463628

7.) Note that the punishment for having more terms is not applied to the bias. This cost function has been implemented already in the function compute_cost_regularised.
Modify gradient_descent to use the compute_cost_regularised method instead of compute_cost. Include the relevant lines of the code in your report and a brief explanation.

gradient_descent.py

```
iteration_cost = compute_cost_regularised(X, y, theta, l)
```

compute_cost_regularised.py

```
for i in range(1,len(theta)):
    total_regularised_error += theta[i]**2
```

Regularization is not applied to theta[0] (ie.bias term) as seen from the above code where 'i' starts from 1 and not 0 .
Regularization is not necessary for bias term because it doesn't contribute any to the curvature of the model and hence neglected.

8.)Modify gradient_descent to incorporate the new cost function. Again, we do not want to punish the bias term. Include the relavant lines of your code in your report.

gradient_descent.py

```
theta_temp = theta.copy()
sigma = np.zeros((len(theta)))
 for i in range(m):
    hypothesis = calculate_hypothesis(X, theta, i)
    output = y[i]
    sigma[0] = sigma[0] + (hypothesis - output)
theta_temp[0] = theta_temp[0] - (alpha/m) * sigma[0]
for j in range(1,len(theta)):
   for i in range(m):
      hypothesis = calculate_hypothesis(X, theta, i)
      output = y[i]
      sigma[j] = sigma[j] + (hypothesis - output) * X[i, j]
theta_temp[j] = (theta_temp[j] * (1 - ((alpha *l)/m))) - (alpha/m) * sigma[j]
theta = theta_temp.copy()
```

9.)After gradient_descent has been updated, run ml_assgn1_3.py. This will plot the hypothesis function found at the end of the optimization. First of all, find the best value of alpha to use in order to optimize best. Report the value of alpha that you found in your report.

The best value of alpha is 1 for which the minimum cost is 0.00780. When alpha is increased to 2 or more than 2, the model is worst in predicting values. Decreasing alpha to 0.1 gives 0.00957 minimum cost and alpha = 1 is considered to be the best because of the cost graph that we get.

Cara Evangeline Chandra Mohan
190539421

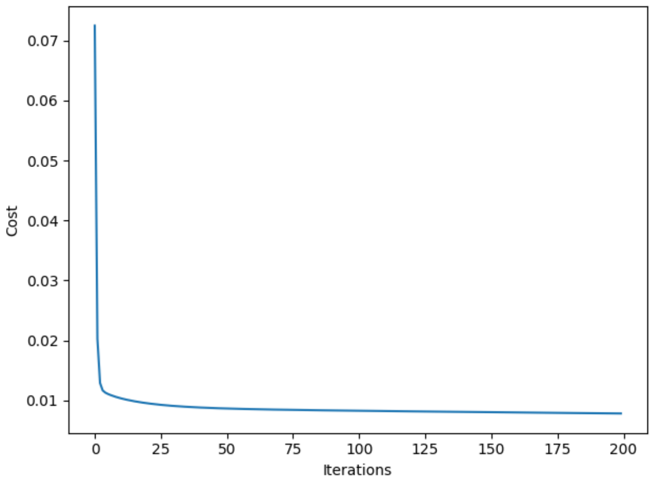


Fig 4:- Cost graph when alpha = 1

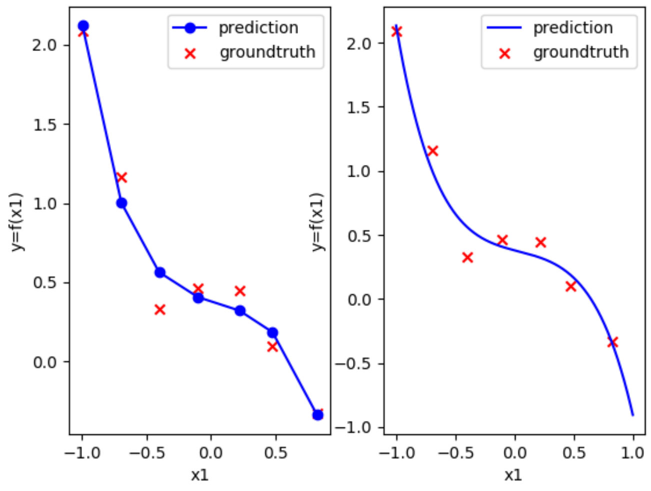


Fig 5:- Hypothesis function graph

10)Next, experiment with different values of λ and see how this affects the shape of the hypothesis. Note that gradient_descent will have to be modified to take an extra parameter, l (which represents λ, used for regularization). Include in your report the plots for a few different values of λ and comment.

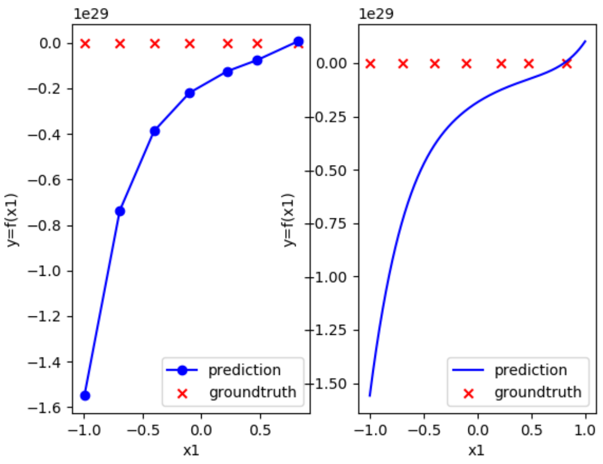


Fig 6:- Hypothesis graph when λ = 10

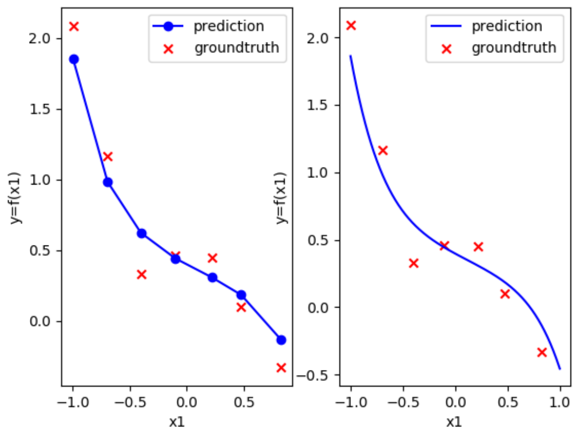


Fig 7:- Hypothesis graph when λ = 1

Cara Evangeline Chandra Mohan
190539421

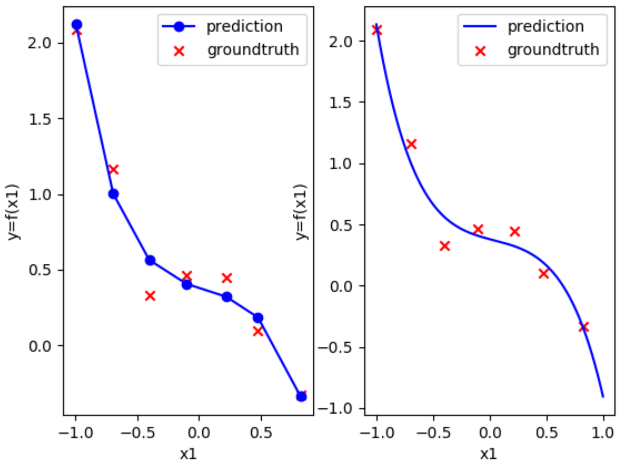


Fig 8:- Hypothesis graph when λ = 0.0001

	Minimum Cost
λ = 10	0.57065
λ = 2	0.08338
λ = 1	0.05295
λ = 0.1	0.01437
λ = 0.0001	0.00781
λ = 0.00001	0.00780
λ = 0.000001	0.00780

ml_assgn1_3.py

```
...........................
theta_final = gradient_descent(X, y, theta, alpha, iterations, do_plot, l)
...........................
```

In the above code gradient_descent is modified to take an extra parameter lambda l.
For a larger value of lambda the cost is very high compared to lesser value of lambda. As we tend lambda --> 0, the minimum cost remains constant.