Queen Mary, University of London
School of Electronic Engineering and Computer Science

ECS708U: Machine Learning

Assignment 1 (Part 1) - Linear Regression

Mughees Asif | 180288337

Due: 12 October 2021

1 Linear Regression with One Variable

Calculate the hypothesis for the i^{th} sample of X, given X, θ and i:

```python
# The hypothesis for the i-th sample of X, given X, theta and i, is the dot product of for each sample with theta can be calculated as:

hypothesis = np.dot(X[i], theta)

```

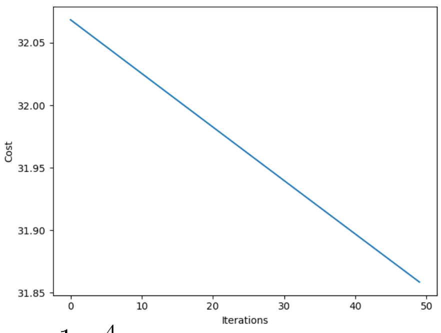
Modified `calculate_hypothesis` **function**:

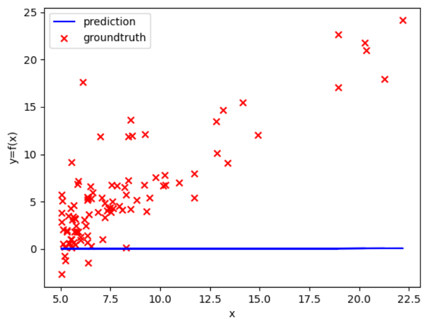
```python
# The `calculate_hypothesis` method can be called with X, theta, and i:
hypothesis = calculate_hypothesis(X, theta, i)

```

Effect of the learning rate α:

- $\alpha = 1\mathrm{e}^{-6}$

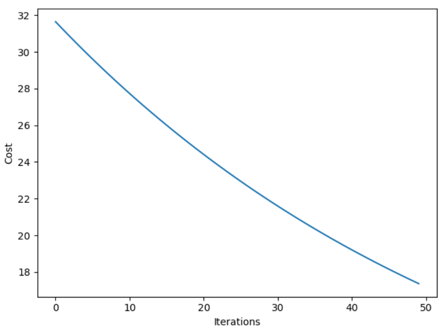
Minimum cost: 31.85851. on iteration #50

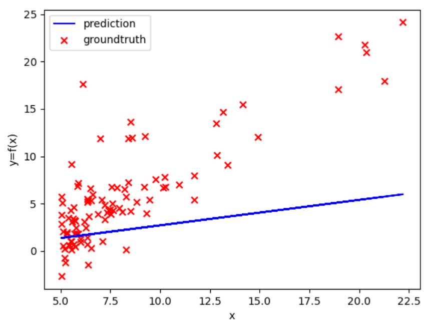




- $\alpha = 1\mathrm{e}^{-4}$

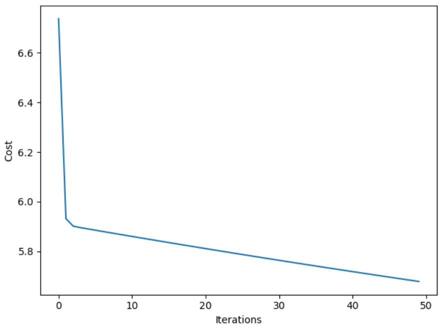
Minimum cost: 17.36882, on iteration #50

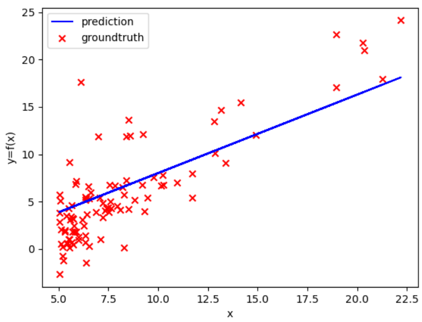




- $\alpha = 1e^{-2}$

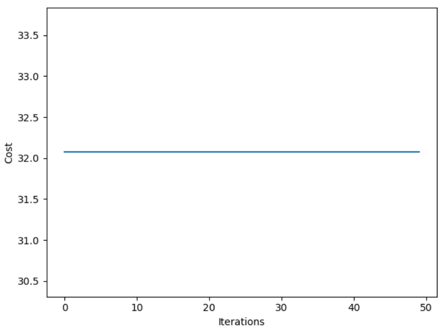
```
Minimum cost: 5.67829, on iteration #50
```

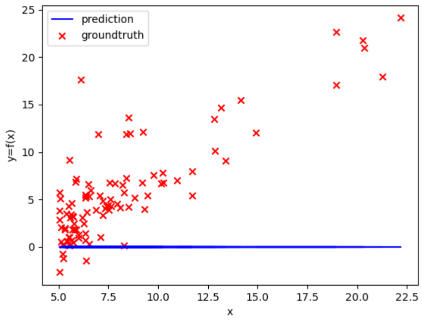




- $\alpha = 0$

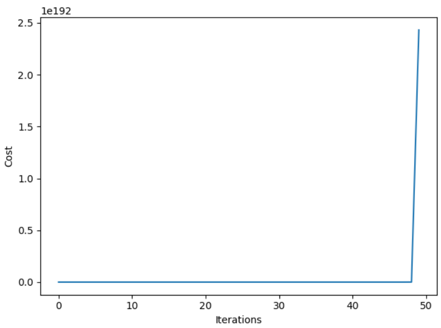
```
Minimum cost: 32.07273, on iteration #1
```

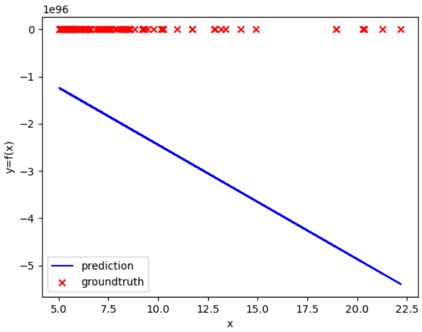




- $\alpha = 1$

```
Minimum cost: 172570.09522, on iteration #1
```





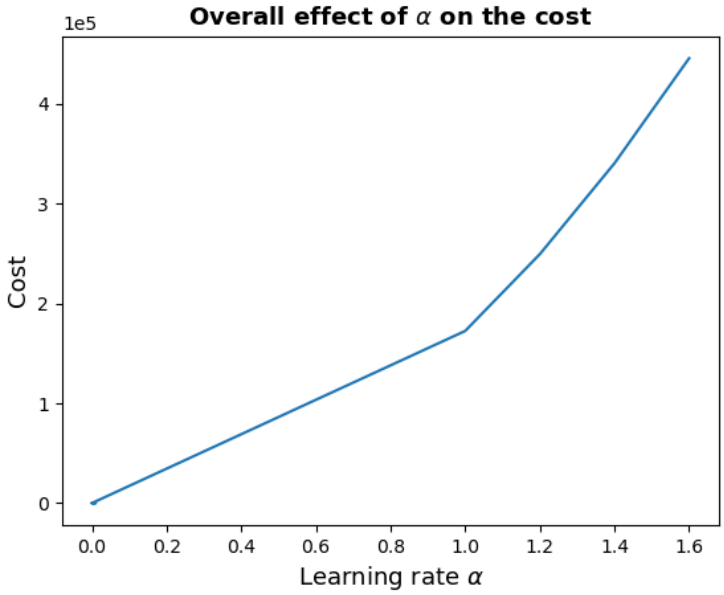


Figure 6: Learning rate and the cost

The learning rate hyperparameter α determines the amount to update the weights during gradient descent to minimise the back-propagation error loss. When α is too small ($1e^{-6}$), the `prediction` does not correlate with the `groundtruth` as the model did not have enough iterations to complete the minimisation or only reached the local minima. However, when $\alpha = 1.6$, the model overshoots (effectively missing the global maxima) and thereby, a vast increase in the cost is also incurred.

From Fig. 6, it can be observed that as α increases from a very small value to a larger value, the gradient is stable when $0 < \alpha < 1$, where a sudden increase in the cost is observable when $\alpha > 1$. The `prediction` is closer to the `groundtruth` when $\alpha = 1e^{-2}$ with a minimal cost value. Therefore, an optimum α for this model exists between the interval $[0, 1]$.

2 Linear Regression with Multiple Variables

Supporting the new hypothesis function:

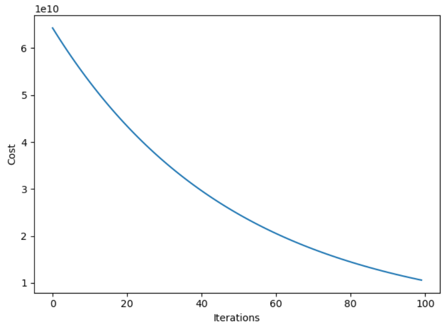
```
    # The hypothesis for the i-th sample of X, given X, theta and i, is the dot product of for each sample with theta can be calculated as:
    hypothesis = np.dot(X[i], theta)

```

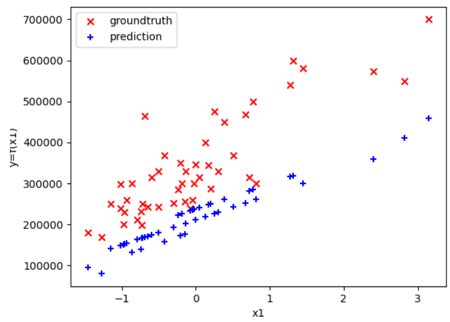
```
        # The `calculate_hypothesis` method can be called with X, theta, and i:
        hypothesis = calculate_hypothesis(X, theta, i)

        output = y[i]

        # Update `sigma`: keep the summation for the derivative term
        sigma = sigma + (hypothesis - output) * X[i]

    # Update `theta_temp`
    theta_temp = theta_temp - (alpha / m) * sigma

```

Effect of the learning rate α:

- $\alpha = 1e^{-2}$

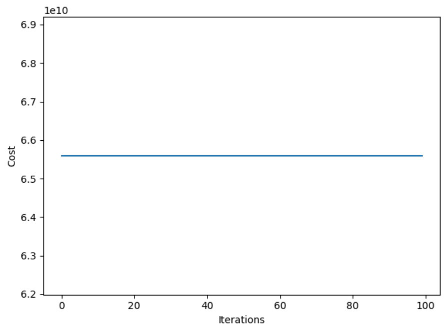
```
Minimum cost: 10596969344.16698, on iteration #100
Theta: [215810.61679138  61446.18781361  20070.13313796]
```

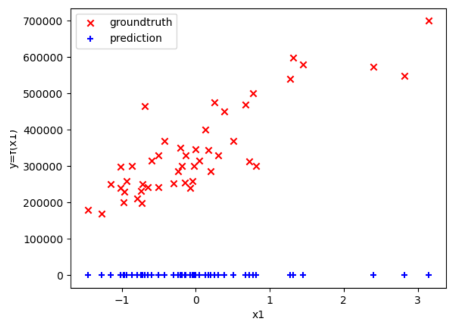




- $\alpha = 0$

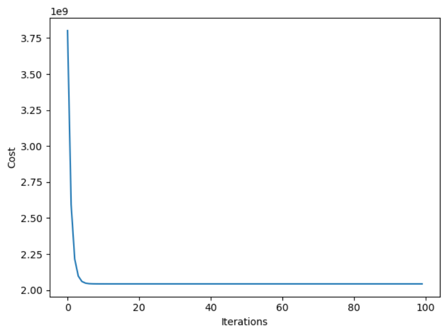
```
Minimum cost: 65591548106.45744, on iteration #1
Theta: [0. 0. 0.]
```

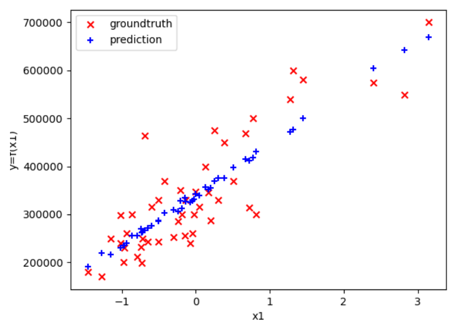




- $\alpha = 1$

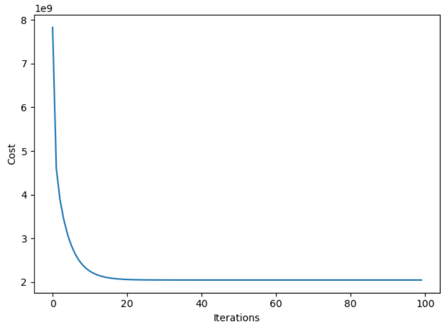
```
Minimum cost: 2043280050.60283, on iteration #48
Theta: [340412.65957447 109447.79646964  -6578.35485416]
```

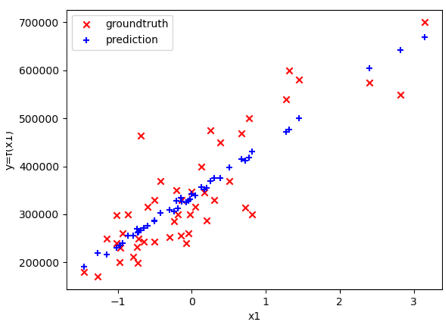




- $\alpha = 1.2$

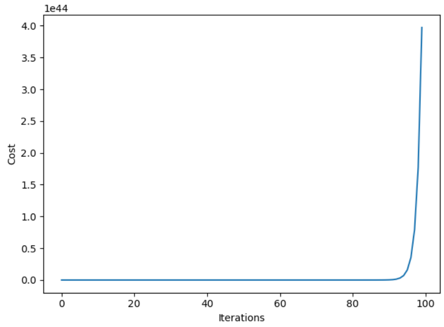
```
Minimum cost: 2043280050.60802, on iteration #100
Theta: [340412.65957447 109447.73880033  -6578.41252348]
```

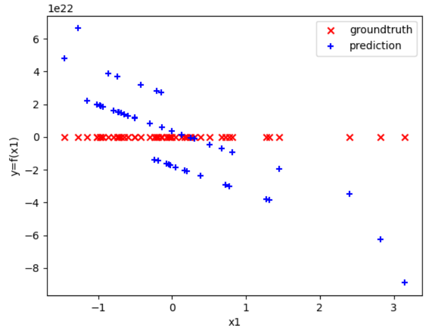




- $\alpha = 1.4$

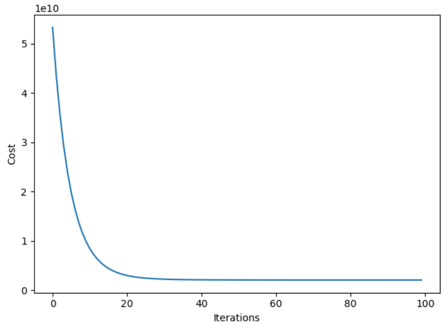
```
Minimum cost: 11667746227.66517, on iteration #2
Theta: [3.40412659e+05 -1.10851714e+12 -1.10851726e+12]
```

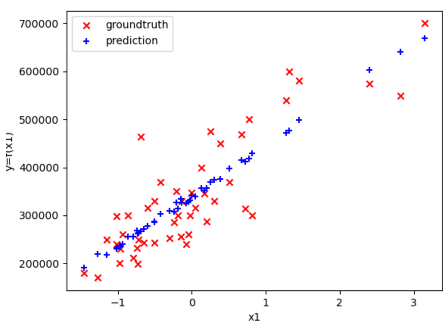




Evidently, the optimum value for α exists when $0 < \alpha < 1$, therefore, after experimenting with several values in the interval $[0, 1]$, 0.1 was identified as a suitable trade-off between convergence and overshooting:

```
Minimum cost: 2043462824.61817, on iteration #100
Theta: [340403.61773803 108803.37852266  -5933.9413402]
```





Making a prediction:

```python
# Create two new samples: (1650, 3) and (3000, 4)
houses = np.array([[1650, 3],
                   [3000, 4]])
# Make sure to apply the same preprocessing that was applied to the training data
houses, mean_vec_new, std_vec_new = normalize_features(houses)
# After normalizing, we append a column of ones to X, as the bias term
column_of_ones = np.ones((houses.shape[0], 1))
houses = np.append(column_of_ones, houses, axis=1)
# Calculate the hypothesis for each sample, using the trained parameters theta_final
prices = []
for i in range(len(houses)):
    prices.append(calculate_hypothesis(houses, theta_final, i))

# Print the predicted prices for the two samples
print("A house with 1650 sq. ft. and 3 bedrooms will cost: {:.5f}".format(prices[0]))
print("A house with 3000 sq. ft. and 4 bedrooms will cost: {:.5f}".format(prices[1]))

```

```
Minimum cost: 2043462824.61817, on iteration #100
Theta: [340403.61773803 108803.37852266  -5933.9413402 ]
A house with 1650 sq. ft. and 3 bedrooms will cost: 237534.18056
A house with 3000 sq. ft. and 4 bedrooms will cost: 443273.05492
```

3 Regularized Linear Regression

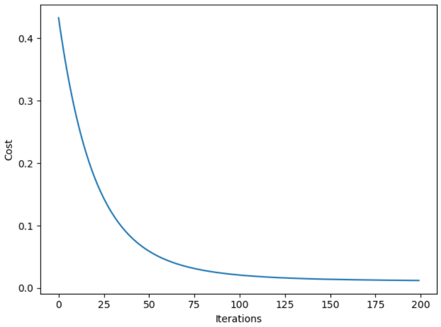
`gradient_descent` **method modifications**:

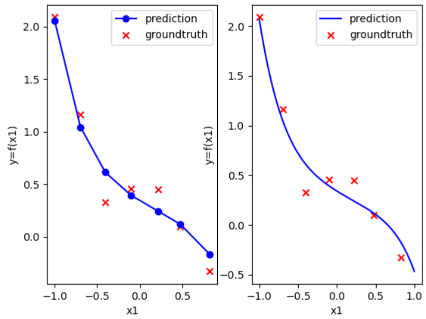
```
        # The `calculate_hypothesis` method can be called with X, theta, and i:
        hypothesis = calculate_hypothesis(X, theta, i)
        output = y[i]
        sigma = sigma + (hypothesis - output) * X[i]

    # update theta_temp
    bias = theta_temp[0]
    theta_temp = theta_temp * (1 - (alpha * (l / m))) - (alpha / m) * sigma
    theta_temp[0] = bias - (alpha / m) * sigma[0]

    # copy theta_temp to theta
    theta = theta_temp.copy()

    # append current iteration's cost to cost_vector
    iteration_cost = compute_cost_regularised(X, y, theta, l)

```

Effect of the learning rate α, and the regularization parameter λ:

- The optimum value for α was found to be **0.02**.
- $\alpha = 0.02, \quad \lambda = 0.0$

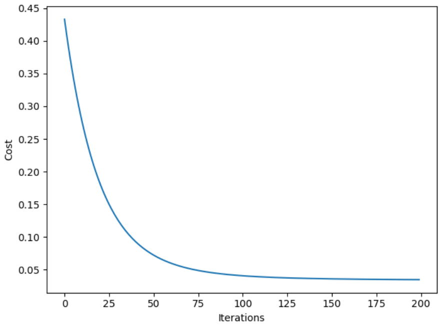
```
Minimum cost: 0.01215, on iteration #200
```

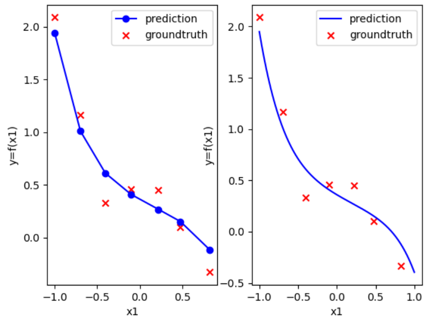




- $\alpha = 0.02, \quad \lambda = 0.5$

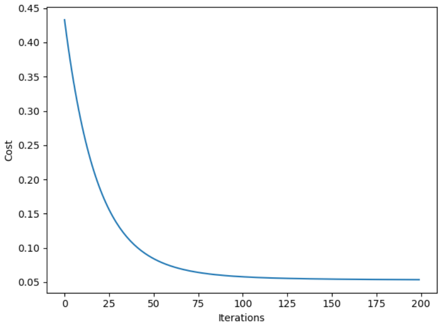
```
Minimum cost: 0.03472, on iteration #200
```

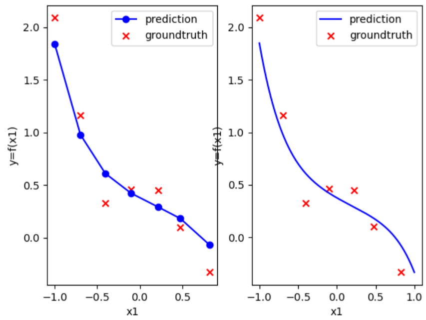




- $\alpha = 0.02, \quad \lambda = 1$

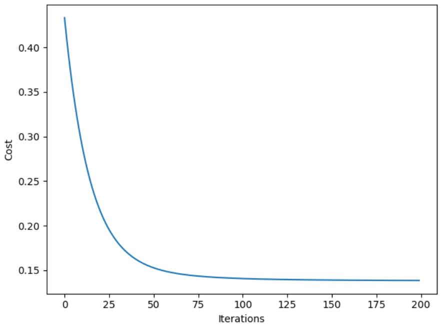
```
Minimum cost: 0.05370, on iteration #200
```

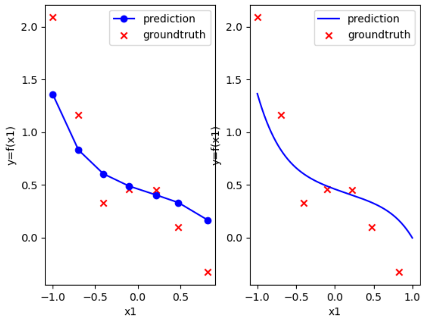




- $\alpha = 0.02, \quad \lambda = 5$

```
Minimum cost: 0.13857, on iteration #200
```





The regularization parameter λ is a scalar value that reduces overfitting. From the above plots, when $0 < \lambda < 1$, similar to α, the `prediction` is closest to the `groundtruth`. As λ is increased, the rate of the gradient descent also increases. Consequently, the minimum cost reduces and the model fits the data suitably. However, increasing the parameter beyond a threshold (≈ 1) results in underfitting.