Queen Mary
University of London

School of Electronic Engineering and Computer Science

ECS708 Machine Learning

Assignment 1: Part 1 - Linear Regression

Aim

The aim of this lab is to become familiar with linear regression.

1. Linear Regression with One Variable

In this exercise, we have one variable, X, and we are building a model to predict one output, Y.

From the root directory of the assignment's code, navigate to the folder "assgn_1_part_1/1_one_variable".

The main program will be run from *ml_assgn1_1.py*. Your first task is to run the main file, by executing the command "python ml_ assgn1_1.py" in a terminal.

You will see a graph open displaying the changing gradient of your hypothesis. When gradient descent has finished, you will see the cost obtained with the theta values used at each iteration. Both the hypothesis and the cost graph will be flat because we need to calculate our hypothesis. For one variable linear regression, the hypothesis function is:

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

where x_0 the the bias term and is set to 1 for all training examples.

Task 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. [5 points]

For example, if the first training example is [1,5] corresponding to [x_0, x_1], return $\theta_0 1 + \theta_1 5$

When this is completed, run *ml_assgn1_1.py* again and you should see the gradient of the hypothesis better fit the model and the cost going down over time. Notice that 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. [5 points]

Now modify the values for the learning rate, alpha in *ml_assgn1_1.py*.

Observe what happens when you use a very high or very low learning rate. Document and comment on your findings in the report. [5 points]

2. Linear Regression with Multiple Variables

From the root directory of the assignment's code, navigate to the folder "assgn_1_part_1/2_multiple_variables". In this part, we will be using the script ml_assgn1_2.py.

We will now look at linear regression with two variables (three including the bias). The hypothesis function now looks like this:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$$

In this exercise, we are looking at house price data. x_0 corresponds to the bias, x_1 is the area of the house in square feet and x_2 is the number of bedrooms. If you open ex1data2.txt you will see there is a large difference between the values of x_1 and x_2.

In order to bring the two features into a similar range, X is normalized using the *normalize_features* function.

Task 2 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. [5 points]

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. [5 points]

Finally, we would like to use our trained theta values to make a prediction. Add some lines of code in *ml_assgn1_2.py* to make predictions of house prices.

How much does your algorithm predicts that a house with 1650 sq. ft. and 3 bedrooms cost?

How about 3000 sq. ft. and 4 bedrooms?

To make a prediction you will need to normalize the two variables using the saved values for mean and standard deviation.

Remember that these are different for the two variables, x_1 and x_2. The formula used for normalization is:

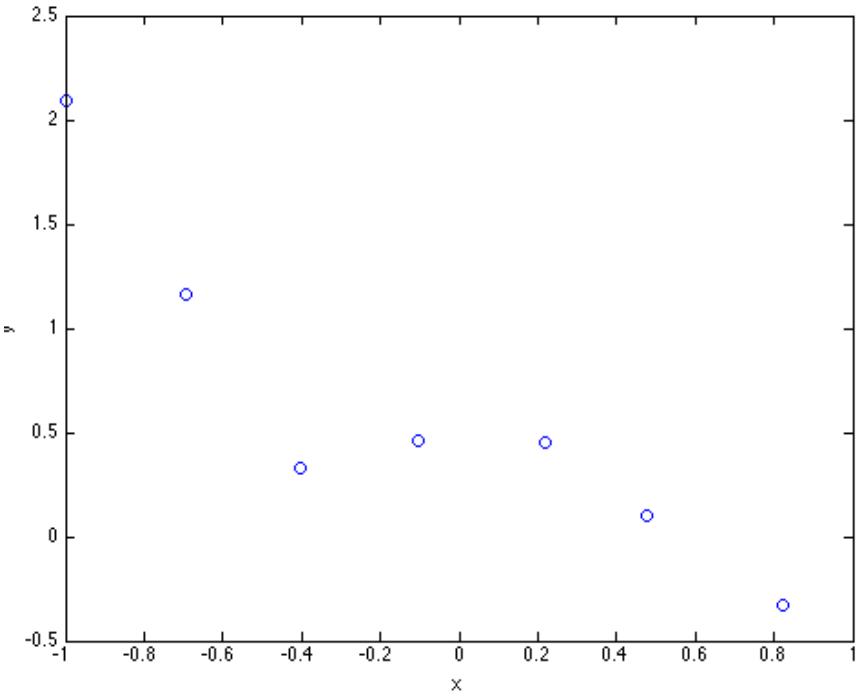
$$X_{norm} = \frac{X - mean}{standard\ deviation}$$

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. [3 points]

3. Regularized Linear Regression

From the root directory of the assignment's code, navigate to the folder "assgn_1_part_1/3_regularized_linear_regression". In this part, we will be using the script ml_assgn1_3.py.

In this exercise, we will be trying to create a model that fits data that is clearly not linear. We will be attempting to fit the data points seen in the graph below:



In order to fit this data we will create a new hypothesis function, which uses a fifth-order polynomial:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 + \theta_4 x_1^4 + \theta_5 x_1^5$$

The values of the data shown in the graph above, are the following:

X''	Y''
-0.99768	2.0885
-0.69574	1.1646
-0.40373	0.3287
-0.10236	0.46013
0.22024	0.44808
0.47742	0.10013
0.82229	-0.32952

As we are fitting a small number of points with a high order model, there is a danger of overfitting.

To attempt to avoid this we will use regularization. Our cost function becomes:

$$J_\theta = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

Task 3 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. [5 points]

Next, modify *gradient_descent* to incorporate the new cost function. Again, we do not want to punish the bias term. This means that we use a different update technique for the partial derivative of θ_c, and add the regularization to all of the others:

$$\theta_0 = \theta_0 - a\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_0^{(i)}$$

$$\theta_j = \theta_j\left(1 - a\frac{\lambda}{m}\right) - a\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

Include the relevant lines of the code in your report. [5 points]

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. [5 points]

Next, experiment with different values of 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 and comment. [5 points]

Write a report about what you have done, along with relevant plots. Save the solution in a folder with your ID. Create and submit a .zip that contains:

1) **all of your code and**

2) a copy of your report. The report should be in .pdf format names as ml_assgn1_part1_StudentID.pdf