# SE4060 – Machine Learning

# Lab 2 – Linear Regression with multiple variables

## Exercise for Linear regression with one variable

1. Open the lab2 directory. There you will find some octave scripts.

   ex2.m - Code to run multivariate linear regression. This calls the appropriate Octave functions in other files to do univariate linear regression on the ex1data2.txt dataset. It now contains house data with varying floor areas and no. of bedrooms. So there are two input features and you have to predict the price of the house. So, it's a multi variate regression problem.

2. Open the ex2.m file. You should get some errors. Now we'll try to modify the code to do multi variate regression.

3. Since now we are dealing with multiple variables, we have to normalize them first to get them to a common scale. Go to the featureNormalize.m octave file and paste the following code under the "Your code here" section. This code is doing mean normalization. Refer the lecture note and see whether you understand what's happening here.

```
for i = 1:size(X, 2)

    mu(:,i) = mean(X_norm(:,i));

    sigma(:,i) = std(X_norm(:,i));

    X_norm(:,i) = (X_norm(:,i) - mu(:,i)) / sigma(:,i);

end
```

Note: A(:, i) selects all items in a column of matrix A in Octave. A(i, : ) will select all items in a row of matrix A.

4. Then, the cost function for multi variate linear regression has to be included to the computeCostMulti.m, under the comment "Your code here".

```
difference = X * theta - y;

vector = difference' * difference;

J = (1 / (2 * m)) * vector;
```

5. Next the Gradient descent algorithm needs to be included for multivariate regression. Go to the GradientDescentMulti.m file and add the following code under “Your code here”. Refer to your lecture note and see whether you understand how this code operates. All operations are matrix operations. A

```
difference  = X * theta - y;

summe  = difference' * X;

theta = theta - (alpha / m) * summe';
```

Note: A’ is the transpose matrix of matrix A.

6. Next, the Gradient descent algorithm will run and give the weights. After getting the theta values, we’ll try to predict the price of a house with floor area of 1650 and 3 bedrooms. For that, in the ex1_multi.m file, add the following lines under “Your code here” in the comment “Estimate the price of a 1650 sq-ft, 3 br house”.

```
prediction = [1 ((1650 - mu(:,1))/sigma(:,1)) ((3 - mu(:,2))/sigma(:,2))];

price = prediction * theta;
```

It’s a matrix with one row and three columns. First value is 1 (since it’s the weight of the intercept). The next values are normalized.

Note: mu – Mean, sigma – Sum

7. Now we’ll do the same with the Normal equation method. Go to the normalEqn.m file and add the following line under the “Your code here”, section.

```
theta = pinv(X' * X) * X' * y;
```

Refer the lecture note to see whether you understand what’s happening here.

Note: pinv method computes the inverse of a matrix in matlab.

8. Now run the ex1_multi.m file again. It will run the gradient descent for multiple variations. The plot show the error against the number of iterations. When the iterations increase, the error converges.

   Then the code is predicting the price of a house with 1650 size and 3 bedrooms. The program will pause here and press enter to continue.

   Then the normal method is used to estimate the weights and the same prediction is done. Compare the normal equation prediction with the prediction given by gradient descent.

9. In the ex1_multi.m, the learning rate is given by alpha. Currently it is assigned to 0.15. Make the alpha values smaller and larger and see how the learning rate and the theta values change.

   Note: When alpha is too small, it will take too many iterations to converge (you can vary the no. of iterations through the iterations variable) and when it's too large it will overshoot and will not converge.

**Overfitting and Under-fitting exercise**

1. Go to the following link and download the Jupyter notebook for the given example. [http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html](http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html)

2. Run the downloaded Jupyer notebook.

3. The function cross_val_score does the cross validation for the three different function degrees provided (1, 4 and 15). Search for the documentation of the function cross_val_score to see the details and the parameters of that function.

4. Change the input degree into 2, 5 and 16 and see how the polynomial functions fit into the data.

5. Export the notebook output to an html file.

**Submission:**

Upload the modified Octave code and the html file exported by Jupyter notebook as a single zip file to the courseweb link. The file name should be your registration number.