Data Mining
Assignment 2: Multivariate Linear Regression and Polynomial Regression

Tauseef Iftikhar

November 20, 2017

1 Introduction

This assignment consists of following 4 files:

1. `assign2data1.txt`
2. `assign2data2.txt`
3. `feature_normalization.py`
4. `multivariate_linear_regression.py`
5. `polynomial_regression.py`
6. `multi_cost_function.py`
7. `multi_compute_gradient.py`
8. `numerical_gradient`
9. `predict.py`

There are two text file `assign2data1.txt` and `assign2data2.txt` which contains comma separated values. For multivariate regression loads data from the first one and polynomial regression uses second onc. You need to run `Multivariate_linear_regression.py` at your command prompt for multivariate regression. You have to provide code in `feature_normalization.py`, `multi_cost_function.py`, `multi_compute_gradient.py`, `numerical_gradient` and `predict.py` to run the program successfully.

For the second part of the assignment, polynomial regression, all the files completed in multivarite regression will be used. You have to run `polynomial_regression.py` at command prompt and provide the missing code in the same file. The file executes without providing the missing code put it performs linear regression and you can see the output after running the file `polynomial_regression.py`.

2 Multivariate Linear Regression

2.1 Loading the Data

In `Multivarite_linear_regression.py`, the dataset is loaded from the data file into the variables x and y:

```
df = pd.read_csv('assign2data1.txt')
# following command converts a dataframe into an numpy
    array.
data = df.as_matrix()
x = data[:,:-1]

# moving first column lableled 'y' in variable y as a
    Series
y = data[:,-1]
```

Now $\mathbf{x} \in \mathcal{R}^{m\times 2}$ and $\mathbf{y} \in \mathcal{R}^{m}$

Often with multiple input features it is not possible to draw the input for visualization.

2.2 Feature Normalization

The `multivariate_linear_regression.py` script will start by loading and displaying some values from this dataset assign2data1.txt. By looking at the values, note that house sizes are about 1000 times the number of bedrooms. When features differ by orders of magnitude, first performing feature scaling can make gradient descent converge much more quickly. Your task here is to complete the code in `feature_normalization.py` to

- Subtract the mean value of each feature from the dataset.
- After subtracting the mean, additionally scale (divide) the feature values by their respective ?standard deviations.?

You should keep it vectorized for efficient implementation.

The standard deviation is a way of measuring how much variation there is in the range of values of a particular feature (most data points will lie within ± 2 standard deviations of the mean); this is an alternative to taking the range of values (max-min). In python/MATLAB, you can use the ?std? function to compute the standard deviation. For example, inside `feature_normalization.py`, the quantity x[:,1] contains all the values of x_1 (house sizes) in the training set, so std(x[:,1]) computes the standard deviation of the house sizes. At the time that `feature_normalization.py` is called, the extra column of 1's corresponding to $x_0 = 1$ has not yet been added to X

You will do this for all the features and your code should work with datasets of all sizes (any number of features / examples). Note that each column of the matrix X corresponds to one feature. You should explore what are the results when mean and std functions are applied to a matrix.

Hint: Broadcasting feature in python is helpful.

Implementation Note: When normalizing the features, it is important to store the values used for normalization - the mean value and the standard deviation used for the computations. After learning the parameters from the model, we often want to predict the prices of houses we have not seen before. Given a new x value (living room area and number of bed- rooms), we must first normalize x using the mean and standard deviation that we had previously computed from the training set.

2.3 Implementing Cost Function and Gradient Descent

You have implemented cost function and gradient descent in your previous assignment. If it is generalized enough to be used with any regression problem then you may use your previous implementation here. You may use x.shape[1] function to know then number of features in x.

You will have to choose value for `alpha` and `iterations` in file `multivariate_linear_regression.py`. After successful implementation of above functions and proper values for `alpha` and `iterations` you could see in Figure 1 a debugging graph converging after certain iterations.

2.4 Prediction

After computing θ you may use these parameters to predict house price for a new unseen house. Predict the price of a house with 1650 square feet and 3 bedrooms. You will use value later to check your implementation of the normal equations. Don?t forget to normalize your features when you make this prediction!

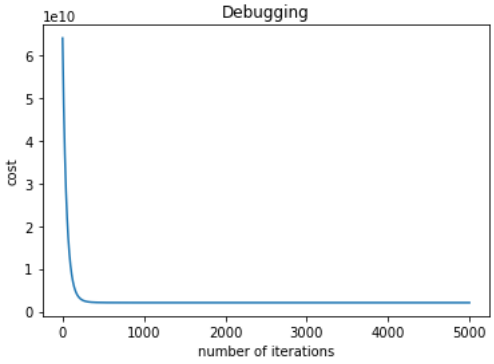

Figure 1: Debugging

2.5 Normal Equation

In the lecture, you learned that the closed-form solution to linear regression is

$$\theta = (X^T X)^{-1} X^T y \tag{1}$$

Using this formula does not require any feature scaling, and you will get an exact solution in one calculation: there is no "loop until convergence" like in gradient descent.

Complete the code in `numerical_gradient.py` by using the formula above to calculate θ. Remember that while you don't need to scale your features, we still need to add a column of 1's to the X matrix to have an intercept term θ_0

Use the θ to predict the value of a house with 1650 square feet and 3 bedrooms it should be same as you computed earlier.

3 Polynomial Regression

You have to run file `polynomial_regression.py`. This will load data from file `assign2data2.txt`. The file plots data for visualization. You can see in Figure 2it is non linear data.

However we have just one feature available in input data. So we have applied univariate linear regression and you can see that a straight line is fitted on the data, which is clear underfitting see Figure 3. Now your task is to create new features so that the model could have enough strength to fit the data reasonably.

```
"""Creating new features
you have to provide code in following line
```

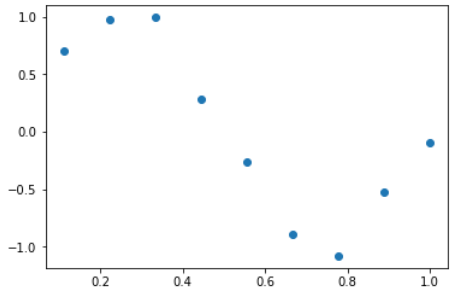

Figure 2: Polynomial regression

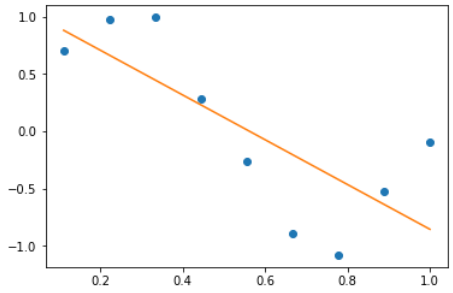

Figure 3: Underfitting

```
"""
#new_x = np.column_stack(YOUR CODE HERE))
```

After choosing appropriate value for alpha, iterations you will get a curve fitting as shown in Figure 4.

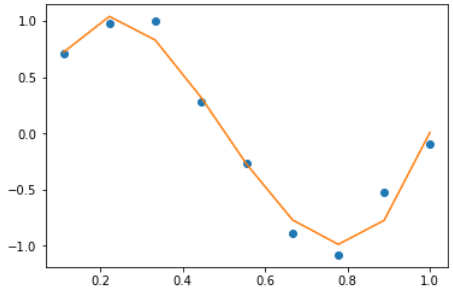

Figure 4: Polynomial regression

4 Submission

Submit a zip folder named according to your roll number at the link given for submission on course page. Folder must contain only following files:

1. `feature_normalization.py`
2. `multivariate_linear_regression.py`
3. `polynomial_regression.py`
4. `multi_cost_function.py`
5. `multi_compute_gradient.py`
6. `numerical_gradient`
7. `predict.py`