Homework 02

Linear Regression with Multiple Variables

In this part of this exercise, you will implement linear regression with multiple variables to predict the prices of houses. Suppose you are selling your house and you want to know what a good market price would be. One way to do this is to first collect information on recent houses sold and make a model of housing prices. The file **hw2data.txt** contains a training set of housing prices in Salt Lake City, Utah. The first column is the size of the house (in square feet), the second column is the number of bedrooms, and the third column is the price of the house. The **hw2.m** script has been set up to help you step through this exercise.

Part 1: Feature Normalization (10 points)

The hw2.m script will start by loading and displaying some values from this dataset. 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 featureNormalize.m 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."

About standard deviations: 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 Octave, you can use the **std** function to compute the standard deviation. For example, inside **featureNormalize.m**, the quantity **X(:,1)** contains all the values of **x1** (house sizes) in the training set, so **std(X(:,1))** computes the standard deviation of the house sizes. At the time that **featureNormalize.m** is called, the extra column of 1's corresponding to **x0 = 1** has not yet been added to **X** (see **hw2.m** for details). 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.

Part 2: Gradient Descent (30 points)

Previously, you implemented gradient descent on a univariate regression problem. The only difference now is that there is one more feature in the matrix **X**. The hypothesis function and the batch gradient descent update rule remain unchanged.

You should complete the code in **computeCostMulti.m** and **gradientDescentMulti.m** to implement the cost function and gradient descent for linear regression with multiple variables. If your code in the previous part (single variable) already supports multiple variables, you can use it here too. Make sure your code supports any number of features and is well-vectorized. You can use **size(X, 2)** to find out how many features are present in the dataset. For your final values of theta, you should have:

$$\theta_0 = 340412.66$$
$$\theta_1 = 110631.05$$
$$\theta_2 = -6649.47$$

And the predicted price of a 1650 square feet, 3 bedroom house should be **$293081.46**.

Part 3: Normal Equations (10 points)

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

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

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 **normalEqn.m** to use 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). The code in **hw2.m** will add the column of 1's to X for you. Theta computed from the normal equations should be:

$$\theta_0 = 89597.91$$
$$\theta_1 = 139.21$$
$$\theta_2 = -8738.02$$

And the predicted price of a 1650 square feet, 3 bedroom house should be **$293081.46**.

Submission:

To submit, turn in the following files on Canvas:

- **featureNormalize.m**
- **computeCostMulti.m**
- **gradientDescentMulti.m**
- **normalEqn.m**