ICS3102 : Foundations of Logic and Symbolic Reasoning

Programming Exercise 1: Linear Regression

0.1 Introduction

In this exercise, you will implement linear regression and get to see it work on data. Before starting on this programming exercise, we strongly recommend watching the video lectures from our class discussion. To get started with the exercise, you will need to download the starter code and unzip its contents to the directory where you wish to complete the exercise.

0.2 Files included in this exercise

1. ex1data1.txt - Dataset for linear regression with one variable
2. ex1data2.txt - Dataset for linear regression with multiple variables 3. excercise1.ipynb - A Jupyter notebook, with guiding codes and sections to be filled up by the student

0.3 Linear regression with one variable

In this part of this exercise, you will implement linear regression with one variable to predict profits for a food truck. Suppose you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. The chain already has trucks in various cities and you have data for profits and populations from the cities.
You would like to use this data to help you select which city to expand to next. The file ex1data1.txt contains the dataset for our linear regression problem. The first column is the population of a city and the second column is the profit of a food truck in that city. A negative value for profit indicates a loss.

0.3.1 Plotting the Data

Before starting on any task, it is often useful to understand the data by visualizing it. For this dataset, you can use a scatter plot to visualize the data, since it has only two properties to plot (profit and population). (Many other problems that you will encounter in real life are multi-dimensional and can't be plotted on a 2-d plot.). The dataset is loaded from the data file into the variables X and y:
Next, the script calls the plotData function to create a scatter plot of the data. Your job is to complete the part of the function that performs the actual plotting to draw the plot; modify the file and fill in with your code.

0.3.2 Gradient Descent

In this part, you will fit the linear regression parameters θ to our dataset using gradient descent. Follow through carefully including the update equations

Implementation

We have already set up the data for linear regression. In the following lines, we add another dimension to our data to accommodate the θ_0 intercept term. We also initialize the initial parameters to 0 and the learning rate alpha to 0.01.

Computing the cost $J(\theta)$

As you perform gradient descent to learn minimize the cost function $J(\theta)$, it is helpful to monitor the convergence by computing the cost. In this section, you will implement a function to calculate $J(\theta)$ so you can check the convergence of your gradient descent implementation. Your next task is to complete the code in the code which is a function that computes $J(\theta)$. As you are doing this, remember that the variables X and y are not scalar values, but matrices whose rows represent the examples from the training set. Once you have completed the function, the next step will run computeCost once using θ initialized to zeros, and you will see the cost printed to the screen.

Gradient descent

Next, you will implement gradient descent in the. The loop structure has been written for you, and you only need to supply the updates to θ within each iteration. As you program, make sure you understand what you are trying to optimize and what is being updated. Keep in mind that the cost $J(\theta)$ is parameterized by the vector θ, not X and y. That is, we minimize the value of $J(\theta)$ by changing the values of the vector θ, not by changing X or y. Refer to the equations in this handout and to the video lectures if you are uncertain. A good way to verify that gradient descent is working correctly is to look at the value of J and check that it is decreasing with each step. The starter calls computeCost on every iteration and prints the cost. Assuming you have implemented gradient descent and computeCost correctly, your value of $J(\theta)$ should never increase, and should converge to a steady value by the end of the algorithm. After you are finished, use your final parameters to plot the linear fit. The result Your final values for θ will also be used to make predictions on profits in areas of 35,000 and 70,000 people. Note the way that the following lines use matrix multiplication, rather than explicit summation or looping, to calculate the predictions.

Visualizing $J(\theta)$

To understand the cost function $J(\theta)$ better, you will now plot the cost over a 2-dimensional grid of θ_0 and θ_1 values. You will not need to code anything new for this part, but you should understand how the code you have written already is creating these images. In the next step, there is code set up to calculate $J(\theta)$ over a grid of values using the computeCost function that you wrote.
After these lines are executed, you will have a 2-D array of $J(\theta)$ values. The script will then use these values to produce surface and contour plots of $J(\theta)$.
The purpose of these graphs is to show you that how $J(\theta)$ varies with changes in θ_0 and 1. The cost function $J(\theta)$ is bowl-shaped and has a global mininum. (This is easier to see in the contour plot than in the 3D surface plot). This minimum is the optimal point for θ_0 and 1, and each step of gradient descent moves closer to this point.

0.4 Linear regression with multiple variables

Just like in the Single Regression Case, follow through the notebook for this excersice.