Problem Set 1

Muhammad Hamza Ikram

April 24, 2021

Contents

In this exercise, you will implement linear regression and get to see it work on data.

All the information you need for solving this assignment is in directory.

Section	Part	Submitted Function	Points
1	Warm up exercise	warmUpExercise	10
2	Compute cost for one variable	computeCost	40
3	Gradient descent for one variable	gradientDescent	50
4	Feature normalization	featureNormalize	10
5	Compute Cost for multiple variables	computeCostMulti	20
6	Gradient descent for multiple variables	gradientDescentMulti	20
		Total Points	130

1 Simple python and numpy function

The first part of this assignment gives you practice with python and numpy syntax and the homework submission process. In the next code block, you will find the outline of a python function. Modify it to return a 5 x 5 identity matrix by filling in the following code:

A = np.eye(5)

```
def warmUpExercise(
        matrix
):
    """
    Example function in Python which computes the identity matrix.

    Returns
    -------
    matrix : array_like
        The 5x5 identity matrix.

    Instructions
    ------------
    Return the 5x5 identity matrix.
    """
    # ======== YOUR CODE HERE ======
    identity_matrix =   # modify this line

    # ==============================
    return identity
```

The previous code block only defines the function warmUpExercise. Run it by executing it in your own script. You should see output similar to the following for a 5x5 input matrix:

```
array([[ 1.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  1.]])
```

2 Linear regression with one variable

Now 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 Data/ex1data1.txt contains the dataset for our linear regression problem. The first column is the population of a city (in 10,000s) and the second column is the profit of a food truck in that city (in $10,000s). A negative value for profit indicates a loss.

You are provided with the code needed to load this data. The dataset is loaded from the data file into the variables x and y:

```
import numpy as np
import os
# Read comma separated data
data = np.loadtxt(os.path.join('Data', 'ex1data1.txt'), delimiter=',')
X, y = data[:, 0], data[:, 1]

m = y.size  # number of training examples
```

2.1 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 cannot be plotted on a 2-d plot. There are many plotting libraries in python (see this blog post for a good summary of the most popular ones).

In this course, we will be mostly be using matplotlib to do all our plotting. matplotlib is one of the most popular scientific plotting libraries in python and has extensive tools and functions to make beautiful plots. pyplot is a module within matplotlib which provides a simplified interface to matplotlib's most common plotting tasks, mimicking MATLAB's plotting interface.

```
from matplotlib.pyplot as plt
def plotData(x, y):
    """
    Plots the data points x and y into a new figure. Plots the data
    points and gives the figure axes labels of population and profit.

    Parameters
```

```
----------
x : array_like
    Data point values for x-axis.

y : array_like
    Data point values for y-axis. Note x and y should have the same size.

Instructions
------------
Plot the training data into a figure using the "figure" and "plot"
functions. Set the axes labels using the "xlabel" and "ylabel" functions.
Assume the population and revenue data have been passed in as the x
and y arguments of this function.

Hint
----
You can use the 'ro' option with plot to have the markers
appear as red circles. Furthermore, you can make the markers larger by
using plot(..., 'ro', ms=10), where `ms` refers to marker size. You
can also set the marker edge color using the `mec` property.
"""
fig = plt.figure()  # open a new figure

# ====================== YOUR CODE HERE ======================


# ============================================================
```

2.2 Gradient Descent

In this part, you will fit the linear regression parameters θ to our dataset using gradient descent.

2.2.1 Update Equations The objective of linear regression is to minimize the cost function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

where the hypothesis $h_\theta(x)$ is given by the linear model

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

Recall that the parameters of your model are the θ_j values. These are the values you will adjust to minimize cost $J(\theta)$. One way to do this is to use the batch gradient descent algorithm. In batch gradient descent, each iteration performs the update

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \qquad \text{simultaneously update } \theta_j \text{ for all } j$$

With each step of gradient descent, your parameters θ_j come closer to the optimal values that will achieve the lowest cost J(θ).