Programming Exercise 1

Linear Regression

Anísio Mendes Lacerda
Machine Learning Class – CEFET-MG

1 Introduction

A very important and general problem in Machine Learning, which has a large set of applications, is *learning* or *inferring* a functional relationship between a set of *attribute* variables and associated *response* variables [1, 3]. Remember that, in class, we learned two types of linear regression models: (i) simple linear regression (with one feature), and (ii) multiple linear regression (with more than one features). The idea of this programming exercise is to complement the concepts learned in class with important new algorithms and techniques.

In this programming exercise, you will implement linear regression and get to see it work on real data. Throughout the exercise, you will be using the files `ex1.ipynb` and `ex1-multi.ipynb`. These files set up the dataset for the problems and make calls to functions that you will write. You do not need to modify either of them. You are only required to modify functions in other files, by following the instructions in this assignment.

1.1 Files included in this exercise

- `pa1.pdf` This file.
- `ex1.ipynb` The notebook file that will help you throughout the programming assignment.
- `ex1data1.txt` Dataset for simple linear implementations.
- `ex1data2.txt` Dataset for multivariate implementations.
- `daily-deals-data.tar.gz` Dataset for real world experimentation on multivariate linear regression.

2 Linear programming with one variable

In part of the assignment, you are asked to implement linear regression with one feature to predict profits for a food truck. The idea is that you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. In other words, we are considering cities to send more food trucks. Obviously, our aim is finding a way to maximize our profit.

Since you are the CEO of a big chain to restaurants, you has trucks in various cities and you have data for profits and populations from the cities. Hence, you are able to examine the data from these and predict the profits of future trucks. Given that we have this training dataset with a label associated to each instance, this is a supervised learning rather than an unsupervised learning task. The dataset includes the population sizes of the cities where we already have food trucks, and our profits from each of these cities. This is an univariate (as opposed to multivariate) linear regression because we only have one input feature (i.e., the population size) per target variable that we're trying to predict (i.e, the profit).

2.1 Plotting the Data

Before starting to solve any problem with machine learning, it is often useful to understand the data by visualizing it. For this dataset, you can use a scatter plot to visualize the data, given that it has only two properties (population and profit). Obviously, the vast majority of problems that you will encounter in real life are multi-dimensional and can't be plotted on a 2-d plot.

In `ex1.ipynb`, the dataset is loaded from the data file into the variables X and y:

```
data = pd.read_csv('ex1data1.txt', header=None, names=['Population', 'Profit'])
data.head()
```

Next, you are asked to plot the data. Hence, execute the following code:

```
data.plot(kind='scatter', x='Population', y='Profit', figsize=(12,8),grid=True);
```

2.2 Gradient descent

In class, we learn how to use a closed pair of equations to find the parameters of our linear model (i.e., the β_is). That method can be generalized to find the parameters of a multivariate linear regression. However, this approach has very problematic issues regarding scalability, i.e., it doesn't scale very well for large data sets. In contrast, in this assignment, we are going to use an optimization algorithm called gradient descent to find the parameters β. This optimization method scale to data sets of unlimited size, so for machine learning problems this is a more practical approach.

TASK 0: Study and understand how gradient descent works. After that, document it and report your understanding. Include equations, figures, and every resource that you need to explain the algorithm.

2.2.1 Update equations

The first thing we need is a cost function. The cost function evaluates the quality of our model by computing the error between our prediction for a data point (using the model parameters) and the actual data point. For example, if the population of a given city is 4 and we predicted that it was 7, our error is $(7-4)^2 = 9$ (assuming an L2 or "*least squares*" loss function).

Formally, the objective of linear regression is to minimize the cost function:

$$J(\beta) = \frac{1}{2m}\sum_{i=1}^{n}(h_\beta(x_i - y_i))^2 \tag{1}$$

where, the model h_β is given by the linear model

$$h_\beta(x) = \beta^T x = \beta_0 + \beta_1 x_1 \tag{2}$$

You are asked to do this for each data point in X and sum the result to get the cost.

TASK 1: Implement Eq. 1 in the following function.

```
def computeCost(X, y, beta):
```

2.2.2 Computing gradient descent

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

$$\beta_j = \beta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}(h_\beta(x_i) - y_i)x_{ij} \quad \text{(simultaneously update } \beta_j \text{ for all } j\text{)} \tag{3}$$

With each step of gradient descent, your parameters β_j, come closer to the optimal values that will achieve the lowest cost $J(\beta)$. Next, you will implement gradient descent in the file `ex1.ipynb`. The loop structure has been written for you, and you only need to supply the updates to β within each iteration.

Make sure you understand what you trying to optimize and what is being updated. Keep in mind that the cost $J(\beta)$ is parameterized by the vector β, not X and y. Refer to the equations above equations in this assignment.

TASK 2: Implement Eq. 3 in the following function.

```
def gradient_descent(X, y, theta, alpha, iters):
```

2.3 Visualizing J(β)

In order to verify that the gradient descent is working correctly, you may take a look at the value of $J(\beta)$ and check that it is decreasing with each step. Assuming that you have implemented gradient descent and compute cost correctly, your value of $J(\beta)$ should never increase, and should converge to a steady value by the end of the algorithm.

3 Multivariate linear regression

Fit a linear model using the dataset of housing prices `ex1data2.txt`.

TASK 3: Repeat the analysis conducted in the previous section, but now considering the multivariate dataset..

4 A real world dataset

In [2], the authors analyze a real world daily deals dataset, specifically, using data from Groupon. They investigate relationships between deal features and deal size beyond simple measures such as the offer price.

TASK 4: Implement their proposed regression method (ref. Eq. 1) considering the Groupon dataset. The idea is that you will use your implementation of gradient descent on this dataset to fit a multivariate linear model.

You are expected to discuss any difficult that you have, and convince that you leaned the key concepts of linear regression.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] John W. Byers, Michael Mitzenmacher, and Georgios Zervas. Daily deals: prediction, social diffusion, and reputational ramifications. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 543–552, New York, NY, USA, 2012. ACM.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.