CS542 HW2 Programming assignment report

CS 542: Machine Learning **Spring 2019**

Problem Set 2

Student: Haoying Zhou *Due: Mar 5, 2019*

2. **(120 points) Programming Report**
   (a) (60 points) Linear Regression
   (b) (60 points) Nearest Neighbors
      i. solution
      ii. solution
      iii. solution
      iv. solution

Files in the zip file:
Report

Five files are for problem (a):
detroit.mat : data source file
hw2_pa.m : main program for running
Normalize.m : function program for normalizing
GDMulti.m : function program for gradient descent
computeCostMulti.m : function program for calculating the cost

Two python files are for problem (b):
hw2_2_1.py : for the first problem of (b)
Be in root folder and run: *python hw2_2_1.py*
hw2_2_2.py: for the rest problems of (b)
Be in root folder and run: *python hw2_2_2.py*
All corresponding data files are in zip file as well, they are very easy to be found out

Most algorithms' meanings are presented in the comment lines, I'll just explain the main idea in this report

(a)
I. Problem

We are given data used in a study of the homicide rate (HOM) in Detroit, over the years 1961-1973. The following data were collected by J.C. Fisher, and used in his paper "Homicide in Detroit: The Role of Firearms," Criminology, vol. 14, pp. 387-400, 1976. Each row is for a year, and each column are values of a variable.

FTP	UEMP	MAN	LIC	GR	NMAN	GOV	HE	WE	HOM
260.35	11.0	455.5	178.15	215.98	538.1	133.9	2.98	117.18	8.60
269.80	7.0	480.2	156.41	180.48	547.6	137.6	3.09	134.02	8.90
272.04	5.2	506.1	198.02	209.57	562.8	143.6	3.23	141.68	8.52
272.96	4.3	535.8	222.10	231.67	591.0	150.3	3.33	147.98	8.89
272.51	3.5	576.0	301.92	297.65	626.1	164.3	3.46	159.85	13.07
261.34	3.2	601.7	391.22	367.62	659.8	179.5	3.60	157.19	14.57
268.89	4.1	577.3	665.56	616.54	686.2	187.5	3.73	155.29	21.36
295.99	3.9	596.9	1131.21	1029.75	699.6	195.4	2.91	131.75	28.03
319.87	3.6	613.5	837.60	786.23	729.9	210.3	4.25	178.74	31.49
341.43	7.1	569.3	794.90	713.77	757.8	223.8	4.47	178.30	37.39
356.59	8.4	548.8	817.74	750.43	755.3	227.7	5.04	209.54	46.26
376.69	7.7	563.4	583.17	1027.38	787.0	230.9	5.47	240.05	47.24
390.19	6.3	609.3	709.59	666.50	819.8	230.2	5.76	258.05	52.33

It turns out that three of the variables together are good predictors of the homicide rate: FTP, WE, and one more variable.

Use methods described in Chapter 3 of the textbook to devise a mathematical formulation to determine the third variable. Implement your formulation and then conduct experiments to determine the third variable. In your report, be sure to provide the step-by-step mathematical formulation (citing Chapter 3 as needed) that corresponds to the implementation you turn in. Also give plots and a rigorous argument to justify the scheme you use and your conclusions.

Note: the file `detroit.mat` containing the data is given on the resources section of our course piazza. To load the data into matlab workspace, use `load()` command. Least-squares linear regression in Matlab can be done with the help of the backslash (\) command.

II. Theory

I will use gradient descent to get the parameters for different features. This has several reasons: firstly, if I use least-square linear regression with some zeroes in denominator, it may come across zero-division error. Secondly, if two features' difference is too large, it's too hard to accomplish

normalization and inverse-normalization when solving normal equation . Finally, gradient descent can solve for the whole set of parameters clearly after some iterations in one time.

The basic function we need to find is:

$$y = w_0 + w_1 x_1 + \cdots + w_9 x_9$$

The above equation is citing chapter 3.

To simplify the problem, assume that $x_0 = z_0 = 1$, so we write the first parameter as itself directly.

To handle the data without influence on numerical difference, the first step is to normalize the data:

$$z_i = \frac{x_i - \mu_i}{\sigma_i}, i = 1,2,\cdots,9$$

Where μ_i, σ_i are means and standard deviations of each feature.
And the above equation becomes:

$$y = \theta_0 + \theta_1 z_1 + \cdots + \theta_9 z_9 = \Theta^T Z$$

Where $\Theta = [\theta_0 \quad \cdots \quad \theta_9], Z = [z_0 \quad \cdots \quad z_9]$

Maybe $\theta s \; and \; ws$ are not identical but they have the same tendency.

To measure whether all the points are in the same lane or all the points are most close to one lane. It is necessary to define a function named as cost function:

$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(\theta_i z_i)^2$$

Where m is the number of features, in the specific example, is equal to 9.
Then, we need to find Θ by iterations. The change rate is proportional to the derivative of cost function.
The gradient descent algorithm is:
Repeat

$$\{\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\Theta)\} (simultaneously\ update\ for\ every\ j = 0, \cdots, i)$$

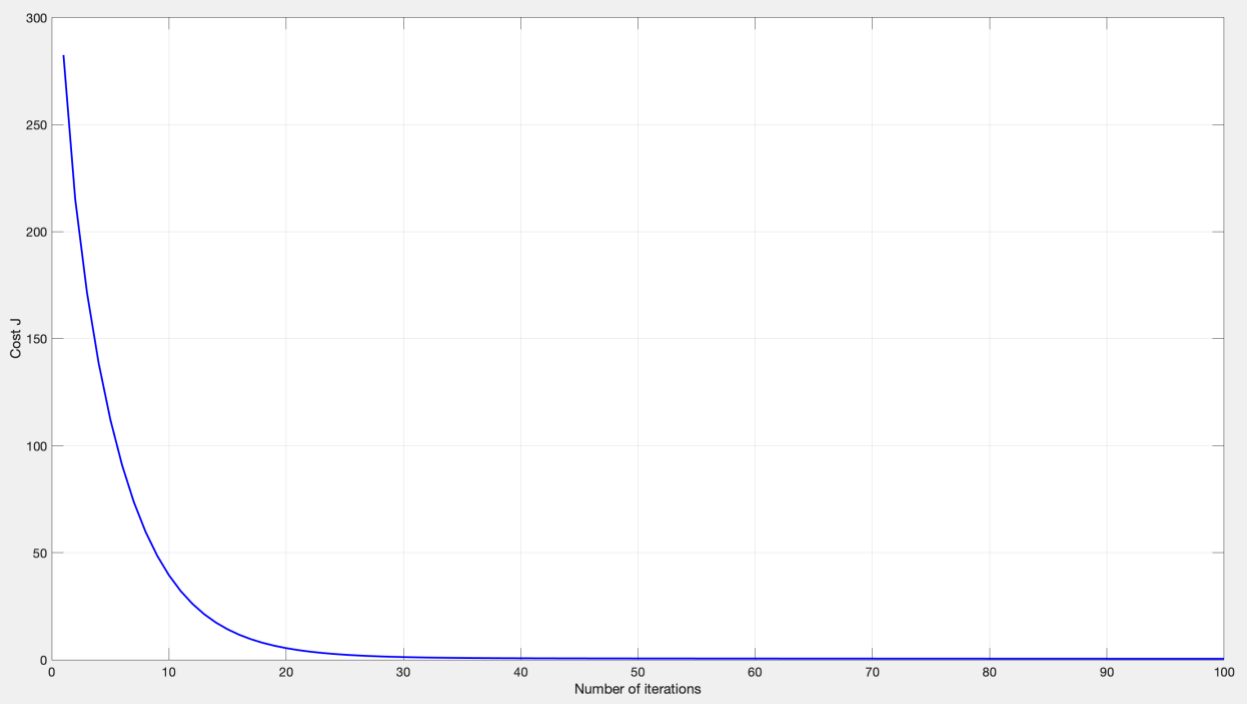
Until θ_js don't change.

α is named as learning rate. Since it is a not a large data set, we could choose learning rate as 0.1 to avoid local maximum. And iteration number

is 100. And we can choose the three largest parameters then we could find the corresponding three features.

III. Results

The code is in MATLAB, I write some basic comments in it.
This is the plot of cost function:



It can see that after 100 times iterations the cost is almost zero.

The main programming file is hw2_pa.m

And Θ is:
Theta computed from gradient descent:
25.126256
3.624453
2.201921
0.094063
3.239165
1.322575
2.308523
2.124974
2.390091
2.745974

IV. Conclusion

Since the first term is constant term, we'll skip the first term. Then the second, last and fifth is the three largest term is corresponded to the first, fourth and last features. The features' names are: *FTP, WE, LIC*

(b)

Problem:

i. Write down (on the written part of your solution) exactly how you imputed missing values for each feature (i.e., replaced all missing values of feature 1 with b) for the CA dataset. Note that you are free to impute the values using statistics over the entire dataset (training and testing combined) or just training, but please state your method. Create a program that can be run on the BU computing environment with the command

ii. Write a k-NN algorithm with L2 distance, $\mathcal{D}_{L2}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_i (a_i - b_i)^2}$, program that can be run with the commands

respectively where k is an integer indicating the k in your k-NN algorithm. The output of your program should be the testing file with an additional comma-separated field indicating the prediction of your classifier.

iii. Generate a table that reports the accuracy on both data sets for at least two different values of k in each case. I have included a small script that you can use to calculate the accuracy and check if your code works (the numbers are made up just for instructional purposes).

i. For all string-typed missing elements, take the median of the feature's dataset as backup because if I have to choose from median and mean, the mean will never make sense. For all int-typed or float -typed elements, take the mean of the feature's dataset as backup because it won't change the mean of the dataset. For all missing elements, the positive and negative are recovered separately: positive to positive, negative to negative.

ii. The basic algorithm of k-NN is that find k nearest neighbor for a specific data point then decide its label which is + or - . For my voting strategy, take the label which has larger amount; If there is a tie, then return the label of the nearest one. And the distance of string-typed feature is that if they

are the same the value is 0, otherwise it's 1. Normalize the data at the very beginning.

iii.

The value of k	Accuracy in %
1	82.6087
2	82.6087
3	85.5072
4	86.9565
5	84.0580
6	86.2319
7	84.7826
8	85.5072
9	85.5072
10	87.6812

Actually I run k from 1 to 30, the highest one is k = 28/30, accuracy = 89.1304%