CS 584: Machine Learning
Assignment-3
Generative Learning

Name: Priyank P Shah Date:04/04/2016
CWID: A20344797 Term:Spring'16

1. Problem Statement :

Objective to develop this project is to classify the datasets using the Discriminative learning approach using Logistic Regression and multilayer perceprtron. Project consist of 2 sub-problems, which are as belows.

1) Logistic Regression for 2 Class and K-Class:
   - Select Dataset from UCI data library, I have selected MNIST image dataset for classification
   - Use of non-linear combinations of input to increase the capacity of classifiers
   - Evaluation of the performance of the dataset

2) Multilayer Perceptron:
   - Implementation of two layer feed-forward network for MLP for three class classification
   - Evaluate the performance by varying the number of elements in the hidden layer and as a function to change the learning rate
   - I have used momentum in algorithm and compared it with scikit-learn implementation.

2. Proposed Solution:

- I have implemented all the code from the scratch.
- For the implementation of Logistic reasoning, I have developed code in two different files for 2-class and K-class, I have used MNIST dataset for logistic reasoning.
- For the implement of MLP, I have user iris-date set.
- During the implementation I have used different python functions of array, matrix and mathematics calculation.
- To validate my output I have used sk learn library.

Algorithm:

Logistic Regression:

Logistic regression depends upon the logistic function, which is similar to step function but instead of producing hard value as 0 or 1, it increments by the class. Logistic function depends upon learning rate and initial value of theta. Function is as below:

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

I have assumed value of learning rate as 0.001 and initial theta as 0.002, near to zero. procedural flow of program is as below:

2 – Class:

- From the dataset, extract data and place accordingly in array.
- For each Cross validation State find model parameters that is sigmoid and theta. Initially take theta near to zero so that sigmoid value set near to 0.5.
- For all the examples, Iterate over values to update value of theta which will improve the value of sigmoid.
- Multiply this sum with the learning factor, the lower the learning factor it will generate the optimum accuracy.
- Every time new theta calculated we will deduct it from the previous value, we will iterate it so better accuracy.
- Once theta is calculated we will predict the value of its class and compare it with the given value of class from test data.

In K-class, Procedure is almost similar to 2-class but here we are finding softmax function instead of sigmoid function and we are putting the hypothesis.

Multilayer Perceptron:

- MLP is a logistic regression classifier. Here, input it transformed using learned non-linear transformation **fi.** It will transform the input data into linearly separable space. It is also known as hidden layer. Each hidden layer consist of perceptron which are known as hidden units.
- Here, we are taking parameters like weight, layer1 output, theta etc. sigmoid function will be activated. If we use theta as wight than it will produce value between 0 and 1, which if we use tanh function that it will produce the weight between -1 and 1.
- This input will generate the hidden layer output, we will iterate this process to improve the weight value. Improved weight value will lead to the higher accuracy.
- At the end of iteration cycle, from the hidden layer output, we will get the final output for predicted class value.
- We will decide the class value by the the highest generated value. Further I have

used the equation provided by the professor.

3. Implementation Details:

Instructions:

- Before running program file, put the datasets into the same folder where the program file resides for MLP, for logistic regression make sure to have active internet connection to fetch the dataset..
- I have developed date-set oriented codes so It may require some modification to run different dataset, while core functionality will remain same for any dataset.
- At the end of the program user will be shown output as Confusion matrix, Accuracy, Precision, Recall and F-1 Measures.

Design Issues:

- During the development of MLP, long haul process took so much time. Reducing the time complexity was a serious issue.
- Exponent function from numpy library was generating false value sometimes which I believe lead towards the sudden drop in the accuracy.
- MNIST is a large dataset containing 70k examples, So it took to much time to fetch it than processing it.
- At beginning while fetching MLP Classifier it was enable to fetch it.

4. Results and Discussion:

1. Logistic Regression:

2-Class

Database: iris.data.txt
fold: 10
iteration: 500
Result:

	Precision	Recall	F1 Measures
Class 0	1.0	1.0	1.0
Class 1	1.0	1.0	1.0

Accuracy: 100%

```
/home/priyankpshah/anaconda2/bin/python "/home/priyankpshah/PycharmProjects/Assignment 3/Logistic_regr.py"
28 72
[-0.62244345 -2.34871851  3.4528956   1.58128925]
 Confusion Matrix:[12  0]
                  [ 0 16]
         Precesion:   Recall:    F-1 Measures
Class 0: 1.0       1.0         1.0
Class 1: 1.0        1.0        1.0

Accuracy: 100.0%

Process finished with exit code 0
```

With the smaller dataset, logistic regression produces very accurate results and maintain consistency too.

K-Class

Database: MNIST
fold: 20
iteration: 10000
Result:

	Precision	Recall	F1 Measures
Class 0	0.9547	0.9712	0.9798
Class 1	0.9337	0.9695	0.9893
Class 2	0.9439	0.9832	0.9594

Accuracy: 92.60%

Accuracy is no predictable here, due to large dataset and suffleing data, sometimes exponent goes out of range[“overflow encountered in exp”] which may produce incorrect result/null resutls. Also program takes much time to finish execution.

2. Multilayer Perceptron:

Database: `iris.data.txt`

Accuracy: 68.79 %

```
/home/priyankpshah/anaconda2/bin/python "/home/priyankpshah/PycharmProjects/Assignment 3/mlp.py"
Multilayer Perceptron:

Accuracy: 65.71%

Process finished with exit code 0
```

Q-2 (A) Derivation:

Q-2(A) Derivation:

Error function: $E(v, \{\omega\}) = 1/2 \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$

$\Rightarrow$ $\hat{y} = \text{Sigmoid}(v^T z)$

$d/dx \{ \text{Sigmoid}(x) \} = \text{Sigmoid}(x)(1 - \text{Sigmoid}(x))$

(i) $\partial E / \partial v = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v}$

(ii) $\frac{\partial E}{\partial \omega_j} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_j} \cdot \frac{\partial z_j}{\partial \omega_j}$

so (i) $\frac{\partial E}{\partial v} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v}$

$= 2 \cdot 1/2 \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})(-1) \cdot \hat{y}^{(i)}(1 - \hat{y}^{(i)}) z$

$\partial E / \partial v = - \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)}) \hat{y}^{(i)}(1 - \hat{y}^{(i)}) z$

Now, we apply gradient over above formula

$v \leftarrow v + \eta \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)}) \hat{y}^{(i)}(1 - \hat{y}^{(i)}) z^{(i)}$

(ii) $\frac{\partial E}{\partial \omega_j} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_j} \cdot \frac{\partial z_j}{\partial \omega_j}$

$= 2 \times 1/2 \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})(-1) \cdot \hat{y}^{(i)}(1 - \hat{y}^{(i)}) v_j \cdot z_j (1 - z_j) x$

$= \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)}) \cdot \hat{y}^{(i)}(1 - \hat{y}^{(i)}) v_j \cdot z_j (1 - z_j) x^{(i)}$

By here, we apply Gradient:

$\omega_j \leftarrow \omega_j + \eta \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)}) \hat{y}^{(i)}(1 - \hat{y}^{(i)}) v_j \cdot z_j (1 - z_j) x^{(i)}$

Reference:

- http://archive.ics.uci.edu/ml/datasets.html?format=&task=cla&att=&area=&numAtt=&numIns=&type=text&sort=nameUp&view=table
- http://www.text-analytics101.com/2014/10/computing-precision-and-recall-for.html
- https://www.quora.com/What-is-the-simple-explanation-of-Multilayer-perceptron-algorithm-in-Machine-learning-Data-Mining
- http://stackoverflow.com/questions/2059111/python-if-statement-with-multiple-conditions