Discriminative Learning

Karthik Bangalore Mani
Department Of Computer Science
Illinois Institute of Technology

April 6, 2016

Abstract

The goal of this assignment is to fit a Logistic Regression Classifer for k-classes and to implement a Multi-Layer Perceptron to recognize digits in the MNIST dataset.

Problem Statement

- **Logistic Regression** : Given a n Dimensional, k-class Dataset, fit a Logistic Classifier to the train data. The examples must be classified and error should be measured. The confusion matrix should be constructed and the performance measures such as precision,recall,F-Measure and accuracy should be determined from the confusion matrix.

- **Multi-Layer Perceptron :** Train Neural Networks for the iris dataset with 3 classes and MNIST dataset with 10 classes . The number of units in the hidden layer should varied and its performance should be noted . The confusion matrix should be constructed and the performance measures such as precision,recall,F-Measure and accuracy should be determined from the confusion matrix.

Proposed Solution

- **Logistic Regression:** Perform Gradient Descent until the theta vectors doesn't change much.The equation for gradient descent is :

  $$\theta = \theta - \eta \frac{1}{m} \sum_{i=1}^{m} (h_\theta(X^i) - y^i).X^i$$

  The equation to determine the hypothesis is :

  $$h_\theta(X^i) = \frac{1}{1+e^{-\theta^T.X^i}}$$

  Once the theta vectors are determined for all k –classes, compute the hypothesis. The one with the highest value of hypothesis is the predicted class.

  $$h_\theta^{(j)}(X^i) = P(y = j|x;\theta) \text{ and } j \in \{1,2,3...\}$$

  $$\hat{y} = \underset{j}{\operatorname{argmax}}\, h_\theta^{(j)}(X)$$

The orders of polynomials will be varied and its performance will be noted.

- **Multi-Layer Perceptron :**
  - **Architecture** : Create a Neural network with 3 layers. Input layer will have nodes = the number of features, output layer will have nodes = the number of different classes, and nodes in the hidden layer will be varied to determine best performance.

  - **Training** : Train the created neural network by varying the learning rate and the number of iterations. Choose the learning rate and number of iterations which gives the least testing error. Training will be done using backpropagation Algorithm as follows :

    Repeat until convergence
    {
    for i in range(1,m):
    1.Forward Propogate to get the activations vectors a^x for x = {2,3,...L}, where x is the layer no
    2.Compute $\delta^L = a^L - y^i$
    3.Compute $\delta^{L-1}, \delta^{L-2} \ldots . \delta^2$ by back propogating
    4.Crank up the θ_{ij}^l for all layers l = {1,2,3..} , all previous layer nodes i , all current layer nodes j
    }
  - **Testing :** Use the parameters computed in the previous step, forward propagate against each example, and find values of activation functions at the output layer. The one with the highest value is the predicted class.

Implementation Details

1. **Program Design Issues :**
   Memory got crashed when I tried training the ANN on the MNIST dataset with all the 50,000 examples. Hence the number of training examples were reduced to 2,000.

2. **Problems faced :**
   Training the neural network on the MNIST dataset was super slow because the program had several nested for loops. Later, most of them were removed by taking dot product of vectors, individually multiplying each entry and adding them. Once the nested loops, were removed, there was significant improvement in the performance of the code.

3. **Instructions to use the program**
   Open the Discriminative Learning.ipynb file in iPython notebook, and execute each cell to get the desired output. The datasets must be placed in the same folder as that of the Discriminative Learning.ipynb file.

Results and discussion

1. **Logistic Regression on 2 class discrimination :**

The iris.data dataset was used from the UCI website. The original dataset contained 3 target classes, out of which 2 were converted to the same number in order to get a 2-class dataset.

Experiment 1 : Iterate 10 times, with a learning rate of 0.001 and poly order 1

Theta Vector:[0.0004 0.00026 0.0022237 -0.00424296 -0.0018363]

Confusion Matrix:

```
[[ 99.   0.]
 [  1.  50.]]
```

Accuracy:

0.993333333333

Class	Precision	Recall	F-Measure
1	1.0	0.99	0.994974874372
2	0.98039215686	1.0	0.990099009901

Experiment 2 : Iterate 10 times, with a learning rate of 0.001 and poly order **2**

Theta Vector [0.00048148 0.00164815 0.00201037 -0.00140963 -0.00073259 0.00335859
0.00792852 -0.01101859 -0.00497059 0.00799393 -0.00313563 -0.00195378
-0.01268511 -0.00461422 -0.00161904]

Confusion Matrix :

```
[[ 100.    0.]
 [   0.   50.]]
```

Accuracy:

1.0

Class	Precision	Recall	F-Measure
1	1.0	1.0	1.0
2	1.0	1.0	1.0

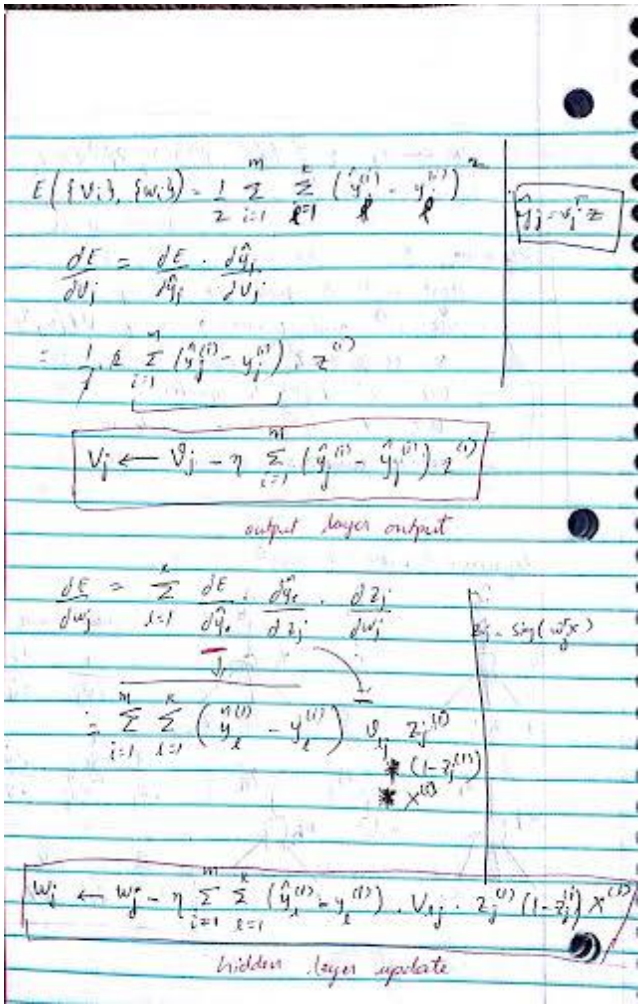
2. Logistic Regression on k-class discrimination :

The original iris.data dataset was used from UC-Irvine website.
After Iterating 10 times, with a learning rate of 0.001 and poly order 2, the 3 theta vectors were found to be :

```
[array([ 0.0062    ,  0.02256667,  0.02461333, -0.01378667, -0.00802
667,
        0.05862467,  0.0992    , -0.11303067, -0.055218  ,  0.094814
67,
       -0.030364  , -0.02177467, -0.13677333, -0.05202467, -0.019004
]),

array([ 0.06273333,  0.23431333,  0.02153333,  0.45573333,  0.131906
67,
       -0.12045133, -0.20581333,  0.58842867, -0.01254867, -0.444258
67,
        0.89982933,  0.27564933, -0.73844267, -0.583638  , -0.361414
67]),

array([-0.06553333, -0.22743333, -0.12871333, -0.12131333, -0.00158
,
       -0.25513667, -0.23756333,  0.148236  ,  0.27943733, -0.201842
,
       -0.00867267,  0.10623733,  0.306582  ,  0.27280067,  0.160711
33])]

Confusion Matrix :
[[ 50.   0.   0.]
 [  0.  49.   2.]
 [  0.   1.  48.]]

Accuracy :
0.98
```

Class	Precision	Recall	F-Measure
1	1.0	1.0	1.0
2	0.960784313725	0.98	0.970297029703
3	0.979591836735	0.96	0.969696969697

3. Multi-Layer Perceptron on iris dataset with 3 classes :



The iris.data dataset from UCI website was used.

- **Neural net with 4 input neurons, 5 hidden neurons and 3 output neurons :**

Confusion Matrix :

```
[[ 50.   0.   0.]
 [  0.  46.   0.]
 [  0.   4.  50.]]
```

Accuracy :

0.973333333333

Class	Precision	Recall	F-Measure
1	1.0	1.0	1.0
2	1.0	0.92	0.958333333333
3	0.925925925926	1.0	0.961538461538

- **Neural net with 4 input neurons, 8 hidden neurons and 3 output neurons :**

- Interestingly, the performance was worse when hidden neurons were increased from 5 to 8.

Confusion Matrix:

```
[[ 50.   0.   0.]
 [  0.  42.   0.]
 [  0.   8.  50.]]
```

Accuracy:

0.946666666667

Class	Precision	Recall	F-Measure
1	1.0	1.0	1.0
2	1.0	0.84	0.913043478261
3	0.862068965517	1.0	0.925925925926

4. Multi-Layer Perceptron on MNIST dataset with 10 classes:

The MNIST images dataset was used.

- **Neural net with 784 input neurons, 785 hidden neurons and 10 output neurons :**

Confusion Matrix :

```
[[  84.    0.    0.    1.    0.    1.    3.    0.    1.    0.]
 [   0.  125.    0.    0.    2.    0.    0.    5.    0.    1.]
 [   0.    0.  108.    6.    1.    0.    0.    4.    1.    0.]
 [   0.    0.    0.   82.    0.    0.    0.    0.    2.    0.]
 [   0.    0.    0.    1.  101.    2.    1.    3.    3.    3.]
 [   0.    0.    1.    5.    0.   77.    1.    0.    2.    2.]
 [   1.    1.    0.    2.    1.    3.   80.    0.    0.    0.]
 [   0.    0.    3.    5.    0.    2.    2.   84.    1.    3.]
 [   0.    0.    2.    4.    0.    2.    0.    1.   78.    2.]
 [   0.    0.    2.    1.    5.    0.    0.    2.    1.   83.]]
```

Accuracy:

0.902

Digit	Precision	Recall	F-Measure
0	0.933333333333	0.988235294118	0.96

1	0.93984962406	0.992063492063	0.965250965251
2	0.9	0.931034482759	0.915254237288
3	0.97619047619	0.766355140187	0.858638743455
4	0.885964912281	0.918181818182	0.901785714286
5	0.875	0.885057471264	0.88
6	0.909090909091	0.919540229885	0.914285714286
7	0.84	0.848484848485	0.844221105528
8	0.876404494382	0.876404494382	0.876404494382
9	0.882978723404	0.882978723404	0.882978723404

Weakness :

- The one v/s all approach in logistic regression works well when there are few distinct target labels. However, when the number target classes increase, it tends to run slow
- The Neural Network takes almost 40 minutes to get trained on the MNIST dataset with 5,000 examples.

References

1. **Stackoverflow.com**
2. **Wikipedia.org**
3. **Coursera.org**