MACHINE LEARNING ASSIGNMENT 3

prateek agrawal

A20358932

Table of Contents

Problem Statement

We are trying to solve the problem of classification using Logistic Regression and Neural Networks. Classification is the problem of identifying a class for a given feature vector. We use different dataset to train our machine for this task. The basic approach that we follow here is drawing a line between different data points discriminating one class from other.

Following are the data set we use:

- Banknote data set for two class logistic regression classification.
- Iris Data set for K class logistic regression classification.
- MNIST data set from Sklearn package for using digit recognition using neural network.

Proposed Solution

Instead of computing the membership function as done in generative learning we implement a decision boundary and use it for classification. We call this decision boundary as Discriminant function. If the value of Discriminant function is greater than zero, we classify it as class 1 and class 2 otherwise.

The model we use here is equation of the line X.N = d, where N is the normal to the line and d is the distance of the line from the origin. We train our model to fit the optimal value of d and N.

The hypothesis we use here is to estimate the parameter Θs and check the product of Θ with the feature vector. If resulting in greater then 0 we classify as class 1 otherwise class 0.

$$h\Theta(X) = 1 \text{ if transpose}(\Theta)*X > 0 \text{ else } 0.$$

2 Class Logistic Regression

Logistic regression uses a better way of classification using a different hypothesis function called Logistic or sigmoid function.

$$h\Theta(X) = 1 / (1 + \exp(-\Theta.X))$$

This always gives a value between zero and one. For classification we say that if $h\Theta(X) > 0.5$ we classify as class 1 else class 0.

We maximize the log likelihood and use iterative approach to find the optimal value the the parameter Θ.

$$l(\theta) = \log\left(\prod_{X=c1} P(y = 1|X) \prod_{X=c2} P(y = 0|X)\right)$$

To maximize this, we take the derivative of the function and equate it to zero.

We get he derivative as:

$$\sum_{i=1}^{m} \big(y(i) - h\Phi(Xi)\big).X(i)$$

Using this derivative in the iterative approach we get the optimal parameter.

$$\emptyset = \emptyset - \eta \sum_{i=1}^{m} \big(h\Phi(Xi) - y(i)\big).X(i)$$

Stopping condition for this approach would be until the likelihood function stops changing.

K Class Logistic Regression

The approach for K class is same as that of 2 class with just some differences. Instead of using one unit we use K unit and each unit has it own vector of parameters. Every unit takes the feature vector as input and uses an activation function to compute a value between zero and one. Activation function that we use here is called Softmax function.

$$Y_{hat(j)} = P(y = j|X) = h\theta(X) = \frac{\exp(\theta j.transpose * X)}{\sum_{i=1}^{k} \exp(\theta i.transpose * X)}$$

The output from each unit is used for classification using the following formulae.

$$y_{hat} = argmax\ h\theta j(X)$$

The iterative equation used is.

$$\emptyset j = \emptyset j - \eta \sum_{i=1}^{m} (h\Phi j(Xi) - indicator(y = j)).X(i)$$

K Class Neural Network Using 2 Layer perceptron

Neural networks are more successful as a classifier. We use feed forward approach this problem. There are two layers in the model. Hidden layer has H unit and output layer has K class. Hidden layer has a weight of W for each unit and output layer has weight V for each unit.

We use the same approach to maximizing the log likelihood to update the parameters in the iterative formulae.

Following is the algorithm followed.

- Start by guessing V_j and W_j.
- Compute Z (output of hidden layer) and Y (output of output layer) using V_j and W_j.
- Update V and W using the computer value in the above step.
- Follow the above steps until the objective (log likelihood) stops changing.

The activation function used in the output function is the softmax and sigmoid in the hidden layer.

Following are the equation we used for updating V and W.

$$Vj = Vj - \eta \sum_{i=1}^{m} (h\Phi j(Zi) - indicator(y = j)).Z(i)$$

$$Wj = Wj - \eta \sum_{i=1}^{m} \left(\sum_{l=1}^{k} (h\Phi j(Zi) - ndicator(y = j)) * Zij * (1 - Zij) \right) . X(i)$$

Implementation Details

Algorithm discussed above is implemented for three different data set. Several functions are build and used according to their functionality.

- **Indicator function** returns 1 if value of the label and the class passed are same else returns 0.
- **Sigmoid** functions are used to calculate hypothesis logistic value for a given parameter and input to the unit.
- **Softmax** functions are used to calculate hypothesis for K class predictions for given parameters and input to the unit.
- **Training** functions are used to estimate the best value of the parameter using an iterative approach. Stopping condition used is if the iteration reaches 100 or the likelihood starts changing less then 0.1
- **Likelihood** functions are used to calculate the log likelihood for the predicted value from the estimated parameters.
- **Ten cross fold** is used to divide the data into training and testing values compute the error for each fold.
- **Compute_Z_Y** is used to calculate the value of Z and Y for an estimated parameter value in the case of K class multi layer perceptron.

Data set used are as follows:

- For N dimension two class we use the Bank Note Authentication available at UCI data repository (http://archive.ics.uci.edu/ml/machine-learning-databases/00267/data_banknote_authentication.txt).

- For N dimension and N class we use the Iris data set ("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data").

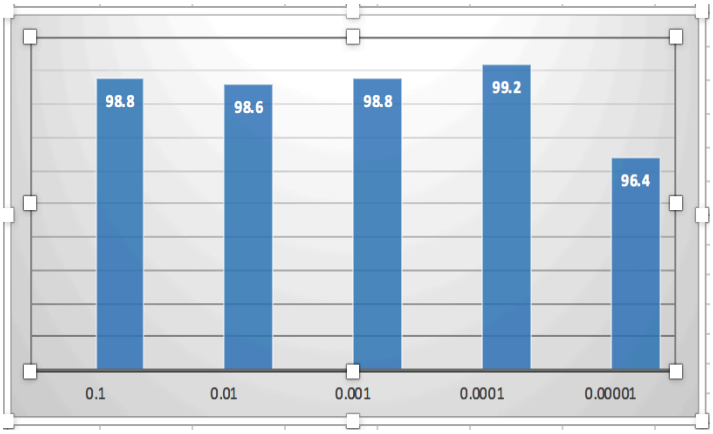
- For K class neural network, we use the MNIST image data set from Sklearn package and choose only digits representing 0, 1 and 2. For training we use 1000 random rows and for testing we use 500 random rows. As the running time for code is high.

Result and Discussion

Logistic Regression for 2 class

- Data set used for this is the banknote authentication from UCI data repository.
- This data set has 1372 row.
- We used ten cross fold to compute the error on the testing set.
- Relation between the learning rate and error is given in the table below.

Learning Rate	Accuracy
0.1	98.8
0.01	98.6
0.001	98.8
0.0001	99.2
0.00001	96.4



Best accuracy comes at 0.001 learning rate.

Logistic Regression for k class

- Data set used for this is the Iris data set from UCI data set
- There are 150 rows with labels 0,1 and 2.
- We used ten cross fold to compute the error on the testing set.
- Relation between the learning rate and error is given in the table below.

Learning Rate	Accuracy
0.1	33.3
0.01	33.3
0.001	33.3

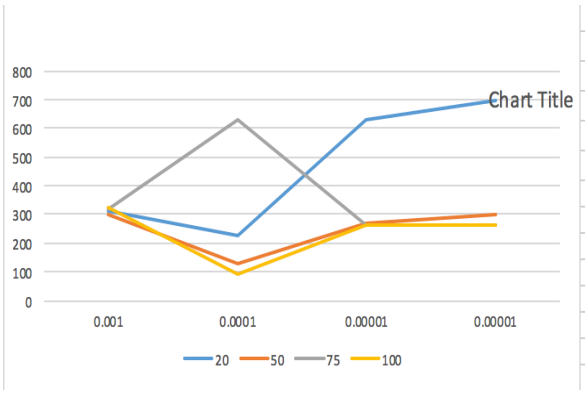
0.0001	83.33
0.00001	96.66

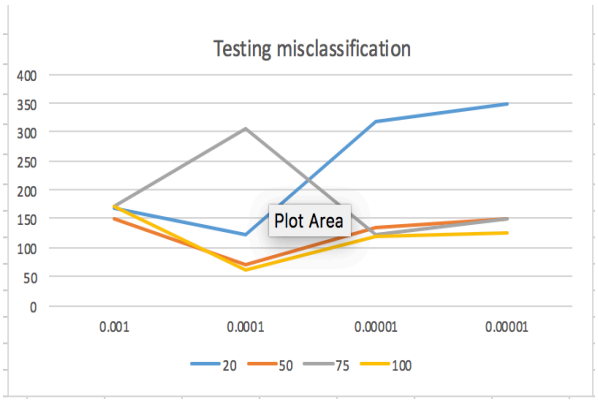
Best accuracy comes at 0.00001 learning rate. Decreasing the learning rate further stops the change in the likelihood and the algorithm converges giving very less accuracy.

Multilayer perceptron for K class classification

- Data set used for this the MNIST image data set from the Sklearn package.
- Rows only with digits representing 0,1 and 2 are taken.
- Out of the 21000 rows only 1000 random rows are selected for the purpose of training and 500 for testing.
- Relation between different parameters such as H and the learning rate is shown in the table below.

Number of hidden units	**Learning rate**	**Training Error(misclassification in 1000 training examples)**	**Testing Error(misclassification in 500 testing examples)**
20	0.01	683	317
20	0.001	315	169
20	0.0001	226	123
20	0.00001	630	317
30	0.01	321	158
30	0.001	318	169
30	0.0001	192	95
50	0.001	299	150
50	0.0001	129	70
50	0.00001	268	134
50	0.000001	-	-
75	0.001	316	172
75	0.0001	629	306
75	0.00001	262	122
75	0.000001	-	-
100	0.001	324	172
100	0.0001	95	60
100	0.00001	266	120
100	0.000001	-	-



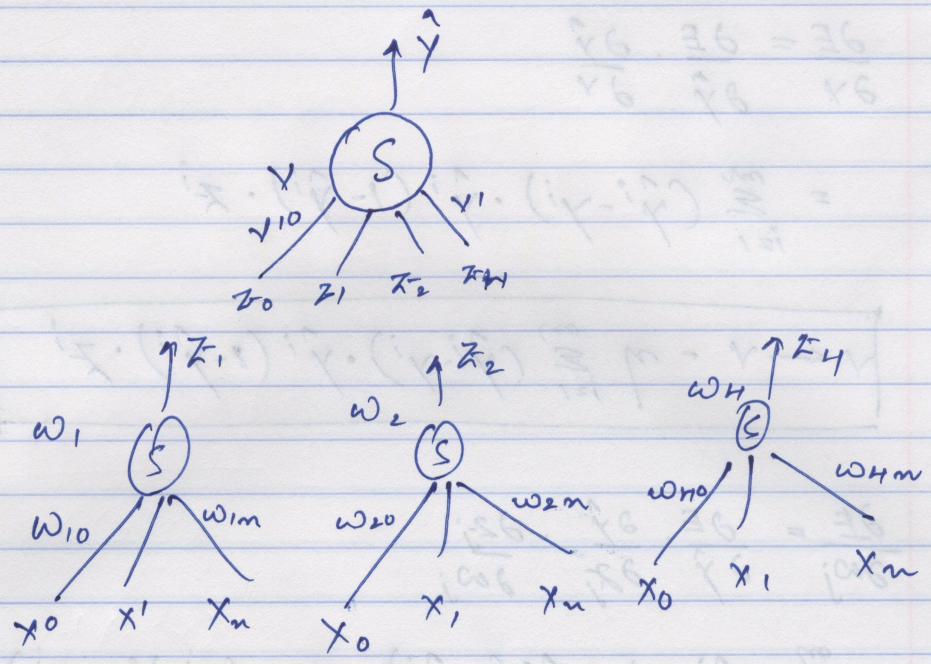


- From the above table we can see that the best error occurs at 100 and 0.0001.
- Keeping the Learning rate less increases the oscillation the increases the misclassification.
- As the value of H increases we need to decrease learning rate in order to get better accuracy.
- Comparison with the actual implementation of MLP classifier takes less time but does not give better accuracy then what I get here.

References

- Stackoverflow.com
- Pythonprogramming.net
- Coursera.com
- Element of statistical learning.

2.
a.



All the Unit- have sigmoid as Activation function.

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

$$z_j = \text{Sigmoid}(w_j^T \cdot X)$$

We use Error funcn as sum of squared Error

$$E = \frac{1}{2} \sum_{i=1}^{m} (y^i - \hat{y}^i)^2$$

To minimize the error we use chain Rule derivation:-

$$\frac{\partial E}{\partial v} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v}$$

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_j}$$

$$\frac{\partial E}{\partial v} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v}$$

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

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

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_j}$$

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

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

These results differ from the one obtained with likelihood as error function.

As, we use sigmoid in all the units and derivative of sigmoid. i.e. sigmoid(x)(1-sigmoidx) is add everytime in the equation.