CS 584

MACHINE LEARNING

Assignment 3:

Discriminative Learning

Nivedita Kalele

A20329966

Logistic Regression:

Introduction:

Logistic regression is a classification model which classifies data according to the probability of occurrence and this classification takes parameters generated by sigmoid function or logistic function.

Iris Dataset (continuous dataset)

This is the multivariate dataset with 4 features and 150 samples in which 3 classes are there. Each class has 50 samples. The classification using logistic regression model is done for these 3 classes. This dataset is used for first question of the assignment.

1] 2 class Logistic Regression

a) Loading the dataset:

The dataset is taken from the sklearn datasets. The features are taken in the x as input and the labels are taken in the y. As 2 class dataset was required and iris is 3 class dataset so, only first 100 samples are considered as input.

b) Estimating the model parameters:

Theta is the parameter we need to estimate for classification.

Assume theta initially.

Sigmoid function:

$$\text{Sigmoid}(x) = \frac{1}{1+\exp(-x)}$$

For prediction of y values:

$$\text{Sigmoid}(\theta^T x) = \frac{1}{1+\exp(-(\theta^T x))}$$

Using this function h($\theta^T x$) is calculated which is then used in gradient descent to update the value of theta. Then if h($\theta^T x$) is greater than 0.5 then it is considered as class 1 otherwise 0.

Gradient Descent to update theta:

$$\theta = \theta - Learning\ rate * \sum_{i=1}^{m}(\mathrm{h}\big(\theta x^{(i)}\big) - y^{(i)})X^{(i)}$$

In this way classification is done using Logistic regression.

c) Performance:

Performance with 10 fold (For Some folds)

```
LOGISTIC REGRESSION FOR 2 CLASS
('FOR FOLD', 1)
   y_pred  y_test
0       1       1
1       1       1
2       1       1
3       1       1
4       1       1
5       1       1
6       1       1
7       1       1
8       0       0
9       0       0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 2)
   y_pred  y_test
0       1       1
1       1       1
2       1       1
3       1       1
4       0       0
5       0       0
6       0       0
7       0       0
8       0       0
9       0       0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 3)
   y_pred  y_test
0       1       1
1       1       1
2       1       1
```

```
3      1      1
4      1      1
5      1      1
6      0      0
7      0      0
8      0      0
9      0      0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 4)
   y_pred  y_test
0      1      1
1      1      1
2      1      1
3      0      0
4      0      0
5      0      0
6      0      0
7      0      0
8      0      0
9      0      0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 5)
   y_pred  y_test
0      1      1
1      1      1
2      1      1
3      1      1
4      1      1
5      0      0
6      0      0
7      0      0
8      0      0
9      0      0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 6)
   y_pred  y_test
0      1      1
1      1      1
2      1      1
3      1      1
4      1      1
5      1      1
6      0      0
```

```
7       0       0
8       0       0
9       0       0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 7)
   y_pred  y_test
0       1       1
1       1       1
2       1       1
3       1       1
4       1       1
5       0       0
6       0       0
7       0       0
8       0       0
9       0       0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 8)
   y_pred  y_test
0       1       1
1       1       1
2       1       1
3       0       0
4       0       0
5       0       0
6       0       0
7       0       0
8       0       0
9       0       0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 9)
   y_pred  y_test
0       1       1
1       1       1
2       1       1
3       1       1
4       0       0
5       0       0
6       0       0
7       0       0
8       0       0
9       0       0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
('FOR FOLD', 10)
   y_pred  y_test
0       1       1
1       1       1
2       1       1
3       1       1
4       1       1
5       1       1
6       0       0
7       0       0
8       0       0
9       0       0
ACCURACY BY MY MODEL 1.0
ACCURACY BY FUNCTION 1.0
```

The Accuracy helps in understanding the performance of the classification. Hence, we can get know what percent of input is misclassified.

By looking at the accuracies and the predicted values above we can say that the data was classified accurately for all folds in 10 fold cross validation.

Using Non-linear combinations of inputs to increase the capacity of the classifier:

For this, we take multivariate data to get more features. By increasing number of features we map data from low dimension feature space to high dimension feature space.

2] 3 class Logistic Regression

a) Loading the dataset: (4 features)

The dataset is taken from the sklearn datasets. All the features are taken in the x as input and the labels are taken in the y. All the samples are considered as input.

b) Estimating the model parameters:

Theta is the parameter we need to estimate for classification.

Assume theta initially.

Sigmoid function:

$$\text{Sigmoid}(x) = \frac{1}{1+\exp(-x)}$$

For prediction of y values:

$$\text{Sigmoid}\left(\theta_j^T x\right) = \frac{\exp(\theta_j^T x)}{\sum \exp(\theta_j^T x)}$$

Using this function h($\theta^T x$) is calculated which is then used in gradient descent to update the value of theta. Then if h($\theta^T x$) is greater than 0 then it is considered as classes j otherwise k.

Stochastic Gradient Descent to update theta:

$$\theta_j = \theta_j - Learning\ rate * (h\theta_j(x) - Indicator(y = j))X^i$$

In this way classification is done using Logistic regression.

c) Performance:

Performance with 10 fold (For one of the folds)

```
LOGISTIC REGRESSION FOR K CLASS
FOR FOLD: 1
Predicted Y :  [0 0 1 1 0 2 0 0 2 0 2 0 0 0 0]
Actual Y :  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
ACCURACY BY MY MODEL 0.666666666667
ACCURACY BY FUNCTION 1.0
FOR FOLD: 2
Predicted Y :  [0 0 0 1 1 0 0 0 0 0 0 1 0 0 1]
Actual Y :  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
ACCURACY BY MY MODEL 0.733333333333
ACCURACY BY FUNCTION 1.0
FOR FOLD: 3
Predicted Y :  [0 0 1 0 0 0 2 0 1 0 0 0 1 0 0]
Actual Y :  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
ACCURACY BY MY MODEL 0.733333333333
ACCURACY BY FUNCTION 1.0
FOR FOLD: 6
Predicted Y :  [1 1 0 0 0 1 0 0 1 0 0 1 0 2 1]
Actual Y :  [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
ACCURACY BY MY MODEL 0.4
ACCURACY BY FUNCTION 0.666666666667
```

The Accuracy helps in understanding the performance of the classification. Hence, we can get know what percent of input is misclassified.

By looking at the accuracies and the predicted values above we can say that the data was classified accurately for some folds in 10 fold cross validation.

3] Two layer feedforward MLP with error function:

2.1) Multilayer perception:

Assumed training dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$ where $x^{(i)} \in R^{n}$ & $y^{(i)} \in \{0,1\}$

Error function = $E(\{w_i\}, v)$

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

According to Backpropagation algorithm, we update (i-1)th layer parameter after updating the parameter of ith layer.

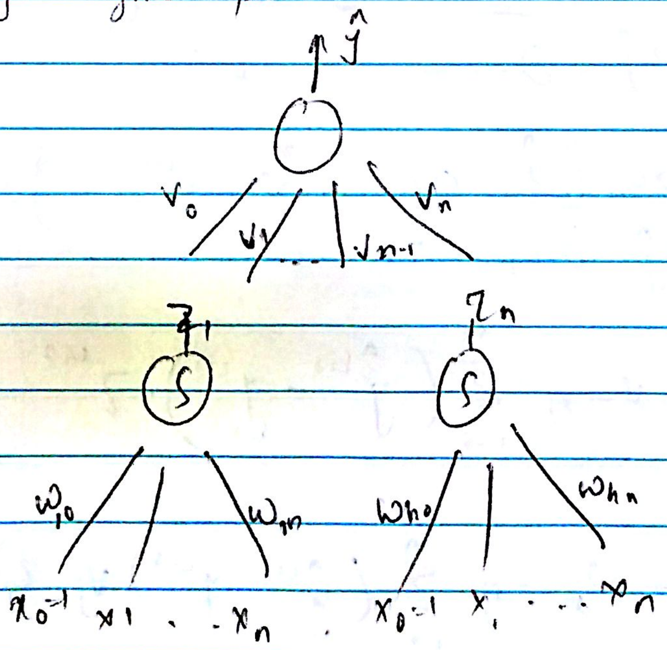
from assumption,

x → Original input

w - parameter of input layer

v - Parameter of hidden

$\hat{y}$ - final prediction.



Minimizing the error:

$$V \leftarrow V - \eta \frac{\partial E}{\partial V}$$

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

$$\therefore V \leftarrow V - \eta \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v}$$

$$w_j = w_j - \eta \frac{\partial E}{\partial w_j}$$

Chain rule, $\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_j} \frac{\partial z_j}{\partial w_j}$

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

$$z_j = h(w_j^T x)$$

$$\frac{\partial z_j}{\partial w_j} = z_j (1 - z_j) x$$

$$\hat{y} = V^T z$$

$$\frac{\partial \hat{y}}{\partial v} = z$$

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

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_j} \frac{\partial z_j}{\partial w_j} = \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) v_j z_j^{(i)} (1 - z_j^{(i)}) \} x^{(i)}$$

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

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

Update equations are same as that of calculated using maximum likelihood function in class.

4] MLP

a) Loading the dataset: (Multi-feature digit dataset)

Input Layer:

This dataset consist of features of handwritten numerals (0-9). 2000 patterns are digitized in binary in which 200 patterns are for each class. As we need 3-class dataset we will be taking first 600 patterns.

Hidden Layer:

The output of input layer is given to the hidden layer as input. 3 activation units are used for hidden layer.

Output Layer:

This gives the final prediction.

The feed-forward algorithm:

1) Start with initial guess of parameters: w_j
2) w_j is used to calculate z^i (output of hidden layer)
3) Update the parameters using z^i

Equation for calculating z^i

$$\text{Z} = \frac{1}{1+\exp(-(W^T x))}$$

Equation for updating w_j

$$w_j = w_j - Learning\ rate * \sum(\text{h}(\theta x^{(j)}) - y^{(j)})X^{(j)}$$