VIT-Vellore SCOPE

CSE6037 - Deep Learning and its Applications

SUTHAR MANAN BHARATKUMAR 20MAI0016

Assessment 1

https://github.com/manansuthar55/CSE6037_20MAI0016_Assessment_1

Problem 1

Train a single preceptron to take multiple inputs and predict binary output by learning the weigts and biases form the training data.

In [1]:

```
# Make a prediction with weights
def predict(row, weights):
 activation = weights[0]
 for i in range(len(row)-1):
  activation += weights[i + 1] * row[i]
 return 1.0 if activation >= 0.0 else 0.0

# Learning Perceptron weights and bias using Stochastic Gradient Descent
def learning_parameters(train, l_rate, n_epoch):
 weights = [0.0 for i in range(len(train[0]))]
 for epoch in range(n_epoch):
  sum_error = 0.0
  for row in train:
   prediction = predict(row, weights)
   error = row[-1] - prediction
   sum_error += error**2
   weights[0] = weights[0] + l_rate * error
   for i in range(len(row)-1):
    weights[i + 1] = weights[i + 1] + l_rate * error * row[i]
  print('Epoch = %d, Sum^2_Error = %.2f' % (epoch+1, sum_error))
 return weights

# Input Samples
dataset = [[1,2,3,0],
 [4,5,6,1],
 [7,8,9,1]]
test=[4,7,8]
l_rate = 0.1    #Learning rate or Bias
n_epoch = 10    #Number of iterations for training the dataset
weights = learning_parameters(dataset, l_rate, n_epoch)
print("Learned weights :", weights)
print("Predicted output for the test sample is ", predict(test, weights))
```

```
Epoch = 1, Sum^2_Error = 2.00
Epoch = 2, Sum^2_Error = 1.00
Epoch = 3, Sum^2_Error = 2.00
Epoch = 4, Sum^2_Error = 1.00
Epoch = 5, Sum^2_Error = 2.00
Epoch = 6, Sum^2_Error = 1.00
Epoch = 7, Sum^2_Error = 1.00
Epoch = 8, Sum^2_Error = 0.00
Epoch = 9, Sum^2_Error = 0.00
Epoch = 10, Sum^2_Error = 0.00
Learned weights : [-0.4, 0.5000000000000001, 0.09999999999999992, -0.30000000000000004]
```

```
Predicted output for the test sample is  1.0
```

Problem 2

Use MLP Classifier algorithm on a classification dataset. Show the score, accuracy and confusion matrix of test samples.

In [2]:

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

#Dataset fetched directly form UCI repository.
path="https://archive.ics.uci.edu/ml/machine-learning-databases/00429/Cryotherapy.xlsx"
df=pd.read_excel(path)
```

- **Here, we will be classifying whether the result of the cryotherapy treatment was a success on a patient or not. We have a detailed info of the patient for better future predictions.**

In [3]:

```
df.head()
```

Out[3]:

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
0	1	35	12.00	5	1	100	0
1	1	29	7.00	5	1	96	1
2	1	50	8.00	1	3	132	0
3	1	32	11.75	7	3	750	0
4	1	67	9.25	1	1	42	0

- **Preparing the sperate input dataframe and output vector. Also, splitting the training and testing data keeping 30% data for testing and rest for training.**

In [4]:

```
X=df.drop('Result_of_Treatment', axis=1)
y=df['Result_of_Treatment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, stratify=y, ra
ndom_state=42, shuffle=True)
```

- **Now, we will you the inbuilt function MLPClassifier i.e. the Multilayer Perceptron Classifier for the classification problem.**
- **We measure the classifier's accuracy by score of the trained model, the confusion matrix of teh test data on the trained model and the accuracy_score of the test sample.**

In [5]:

```
#Inbult function used for MLP classifier.
clf = MLPClassifier(random_state=42, max_iter=300).fit(X_train, y_train)
sc=clf.score(X_test, y_test)
print("Score : {:.2f}%".format(sc*100))
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_pred, y_test)
print("Confussion Matrix of the predicted samples using test data:\n",cm)
ass=accuracy_score(y_test, y_pred, normalize=False)
print("Accuracy of MLPClassifier from test samples:", ass)
```

```
Score : 92.59%
Confussion Matrix of the predicted samples using test data:
 [[13  2]
 [ 0 12]]
Accuracy of MLPClassifier from test samples: 25
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```