Department of Computer Engineering
University of Peradeniya

CO 544 Machine Learning and Data Mining

Lab 04

6^{th} August 2021

1. Objective

Implementing and training single layer and multi layer perceptron using Python modules.

2. Introduction

Single Layer Perceptron
The simplest neural network model that can classify linear separable cases with binary values. Single Layer Perceptron has only two layers (input layer and output layer).

Multi Layer Perceptron
Multi Layer Perceptron is a feedforward neural network which contains one or more hidden layers apart from the input layer and output layer. Multi Layer Perceptron can learn non-linear functions as well.

Perceptrons can be modeled using **scikit-learn** module.

3. Modeling and Training Single Layer Perceptron

```
model=Perceptron()   #Create the model
model.fit() #Train the model
model.predict() #Predict using the trained model
```

4. Single Layer Perceptron: AND Gate

```
import numpy as np
import pandas as pd
from sklearn.linear_model import Perceptron
import matplotlib.pyplot as plt
from itertools import product

#Create four possible inputs to AND gate
data = [[0,0], [0,1], [1,0], [1,1]]
labels = [0,0,0,1]

plt.scatter([point[0] for point in data], [point[1] for point in data], c = labels)
plt.show()

#Build a perceptron to learn AND gate
classifier = Perceptron(max_iter=50)
classifier.fit(data, labels)
print(classifier.score(data, labels))
```

TODO 1: Implement and train a single-layer perceptron model to OR and NOT logic gates.

TODO 2: Single Layer Perceptron using Iris data set

```
from  sklearn import  datasets
iris=datasets.load_iris()
```

Data preprocessing

(a) Extract the first 100 class labels which has information on only two labels.

(b) Convert the class labels into the two integer class labels 1 (Versicolor) and -1 (Setosa)

(c) Define feature matrix X by sepal length (first column) and petal length (third column)

(d) Visualize data by a scatter plot

Training the perceptron model

(a) Train the perceptron model

(b) Check the convergence of error by plotting a graph

(c) Plot two decision regions.

5. Modeling and Training Multiple Layer Perceptron

MLPClassifier function can be easily used to multiple layer perceptron modeling

```
model = MLPClassifier(hidden_layer_sizes=(128,64,32), max_iter=100, verbose=True)
```

Here, we have used three hidden layers with different neurons in each layer. Including input and output layer, we have 5 layers in the model. *verbose* implies whether to print progress messages or not. Default is False.

6. Example: Multiple Layer Perceptron

(a) Step 1: Import required modules

```
from sklearn.datasets import load_iris;
from sklearn.model_selection import train_test_split;
from sklearn.preprocessing import StandardScaler;
from sklearn.neural_network import MLPClassifier;
from sklearn.metrics import accuracy_score;
import numpy as np;
```

(b) Step 2: Load data

```
# load iris dataset form sklearndata
iris_data = load_iris()
```

(c) Step 3: Split training and testing data

```
# split in to train and test X, y
train_data, test_data, train_labels, test_labels = train_test_split(iris_data.data, iris_data.

n_trainx = len(train_data)
n_trainy = len(train_labels)

n_testx = len(test_data)
n_testy = len(test_labels)

# asserting sizes
assert(n_trainx == n_trainy)
assert(n_testx == n_testy)

print('train & test data x, y sizes are matched')
```

(d) Step 4: Standardize data

```
standardScaler = StandardScaler()

standardScaler.fit(train_data)
s_train_data=standardScaler.transform(train_data)

s_test_data=standardScaler.transform(test_data)
```

(e) Step 5: Model multiple layer perceptron model

```
model = MLPClassifier(hidden_layer_sizes=(64,32), max_iter=50, verbose=True);
```

(f) Step 6: Train the model

```
clf=model.fit(s_train_data, train_labels)
```

(g) Step 7: Make predictions

```
predicted_data = model.predict(s_test_data)

print(predicted_data)
```

(h) Step 8: Evaluate the accuracy of the classifier.

```
print(clf.score(s_test_data, test_labels))
```

TODO 3: Train a multiple layer perceptron model to XOR gate. (2 neurons in the input layer, 5 neurons in the hidden layer and 1 output neuron)

7. Submission

(a) In Lab Submission
Please submit whatever you have done within the lab. Rename your file as 16xxxlab04.txt where xxx is your registration number.

(b) Complete Submission
Submit three different text files three TODO sections. Your files should contain all your commands you have tried out in the lab with the answers for TODO sections (as comments in the text). Rename the files as 16xxxlab04_TD1.txt , 16xxxlab43_TD2.txt and 16xxxlab04_TD3.txt where xxx is your registration number.

8. Deadline

The deadline: August 13, 2021, by 11.59 p.m
If you do not understand any concepts, make sure you get some help from instructors.