Assignment-2

March 13, 2023

Name	Department	ID
Ahmed Ashraf Mohamed	AI	2103134

Importing the libraries that are going to be used

[101]:
```
from sklearn.model_selection import train_test_split # used for easier␣
 ↪splitting of the data and also stratifying the data
from sklearn.metrics import accuracy_score # used for accuracy score function␣
 ↪which return the accuracy of our model
# importing numpy and pandas libraries
import numpy as np
import pandas as pd
```

1 Reading data's file

[102]:
```
dataframe = pd.read_csv('iris.csv',header = None) # droping the headear
dataframe.head()
```

[102]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

1.1 Separating the data's feature and lable

[103]:
```
X = df.iloc[:,:-1]
Y = df.iloc[:,-1]
print("feature Vector")
print(X.head())
print("-"*10)
print("Label Vector")
print(Y.head())
```

```
feature Vector
     0    1    2    3
0  5.1  3.5  1.4  0.2
1  4.9  3.0  1.4  0.2
2  4.7  3.2  1.3  0.2
3  4.6  3.1  1.5  0.2
4  5.0  3.6  1.4  0.2
----------
Label Vector
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: 4, dtype: object
```

1.2 Transform the class from Strings to values

[104]:
```
Y = np.where(Y == "Iris-setosa",0,1) # replace every Iris-setosa with a 0 and␣
↪everything else with a 1
Y
```

[104]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

1.3 splitting the data

Splitting the data using the sklearn's built-in function and show the count of each class

[105]:
```
X_trian,X_test,y_trian,y_test = train_test_split(X,Y,test_size = 0.3,stratify =␣
↪Y) # splitting the data using the sklearn's built-in train test split␣
↪function
```

[106]:
```
pd.DataFrame(y_trian).value_counts() # showing the count of each class in our␣
↪data
```

[106]:
```
0    35
1    35
dtype: int64
```

[107]:
```
pd.DataFrame(y_test).value_counts() # showing the count of each class in our␣
↪data.
```

[107]:
```
0    15
1    15
```

```
dtype: int64
```

1.4 Perceptron algorithm

Defining the activiation function that is going to be used for the perceptron algorithm.

[110]:
```
def binary_step_func(net,threshold):
    return np.where(net >= threshold, 1, 0)
```

Defining a class for the percertron algorithm to make it easier to call later.

[111]:
```
class Perceptron:

    def __init__(self, learning_rate=0.01, n_iteration=1000, threshold = 0.5):
        self.lr = learning_rate
        self.n_iteration = n_iteration
        self.activation_func = binary_step_func
        self.threshold = threshold
        self.weights = None
        self.bias = None

      # fitting the data on the perceptron
    def fit(self, X, y):
        n_samples, n_features = X.shape  #n_samples=rows    n_features=columns

        # initial weights and bias
        self.weights = np.random.random(n_features)
        self.bias = random.randint(0,1)

        # applying Perceptron algorithm
        for iteration in range(self.n_iteration):
            # calculating the net using the dot prodcut property
            net = np.dot(X, self.weights) + self.bias
            # applying  activation function on net
            y_predicted =self.activation_func(net,self.threshold)
            MSE=(1/n_samples)*np.sum((y - y_predicted)**2)

            for i in range(len(X)):
             #  Updating weights and bias
                delta = self.lr * MSE
                #print(X[i]) for tracing
                #print(MSE)

                self.weights += delta * X[i]
                self.bias += delta

    def predict(self,X,threshold):
        net = np.dot(X, self.weights) + self.bias
```

```
        y_predicted = self.activation_func(net,threshold)
        return y_predicted
```

1.5 fitting the algotrithm on the data

[112]:
```
q = Perceptron(0.01,10000,0)
q.fit(np.array(X_trian),np.array(y_trian))
```

1.6 Testing

[113]:
```
result = q.predict(X_test,y_test)
```

1.7 Accuracy checking

[116]:
```
print(f'Accuracy: {accuracy_score(y_test,result) * 100}%')
```

```
Accuracy: 50.0%
```

Whether the threshold change or not the accuracy of the of perceptron will remain the same.