**20CS2032L – MACHINE LEARNING TECHNIQUES**

**URK22AI1016**

|  |  |
| --- | --- |
| **EXERCISE** | **5. MULTIPLE LAYER PRCEPTRON** |
| **DATE** | **30.08.2024** |

# AIM:

To design and implement feed forward multiple layer perceptron (MLP) trained using backpropagation algorithm to classify the given dataset.

# DESCRIPTION:

A MLP is a class of feedforward artificial neural network trained using backpropagation algorithms. The architecture uses, at least, three layers of nodes: an input layer, a hidden layer and an output layer.

# Algorithm:

Step 1: Select the number of input, hidden and output nodes

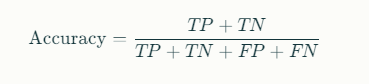
Step 2: Initialize the network and set the required hyperparameters like initial weights, hidden nodes, learning rate, etc.

Step 3: Read a pattern, calculate the output of nodes in the successive layers Step 4: Calculate the error E, at the output node.

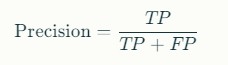
Step 5: Propagate the error and adapt weights based on error derivatives Step 6: Continue from step 3 for the required number of epochs.

Step 7: Evaluate the performance of the model using metrics and plot the hyperplane.

# Calculation metrics:

Accuracy: Measures the proportion of correct predictions among the total predictions.

TP = True Positives TN = True Negatives FP = False Positives FN = False Negatives

Precision: Measures the proportion of true positive predictions among all positive predictions.

Recall (Sensitivity): Measures the proportion of true positives among all actual positives.



F1 Score: The harmonic mean of precision and recall, providing a balance between the two.

Ex. 05 | Multiple Layer Analysis

1

**20CS2032L – MACHINE LEARNING TECHNIQUES**

**URK22AI1016**

# SOURCE CODE:

1. **Implement MLP to classify the given data set and analyse the performance of the classifier.**

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import classification\_report, accuracy\_score data = load\_iris()

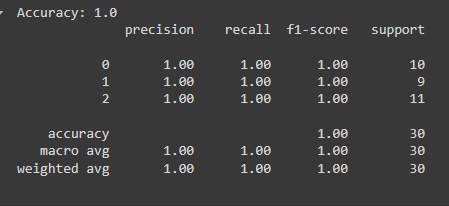
X = data.data y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

mlp = MLPClassifier(hidden\_layer\_sizes=(10, 10), max\_iter=1000, learning\_rate\_init=0.01) mlp.fit(X\_train, y\_train)

y\_pred = mlp.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred)) print(classification\_report(y\_test, y\_pred)) **Output:**



# Implement MLP for regression task.

import numpy as np

from sklearn.datasets import fetch\_california\_housing from sklearn.model\_selection import train\_test\_split from sklearn.neural\_network import MLPRegressor from sklearn.metrics import mean\_squared\_error

data = fetch\_california\_housing()

X = data.data y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

mlp\_regressor = MLPRegressor(hidden\_layer\_sizes=(10, 10), max\_iter=1000, learning\_rate\_init=0.01) mlp\_regressor.fit(X\_train, y\_train)

y\_pred = mlp\_regressor.predict(X\_test) # Evaluate performance

mse = mean\_squared\_error(y\_test, y\_pred) print("Mean Squared Error:", mse)

Ex. 05 | Multiple Layer Analysis

2

**20CS2032L – MACHINE LEARNING TECHNIQUES**

**URK22AI1067**

# Output:

1. **Improve the performance of the model by adjusting the hyperparameters such as number of hidden nodes, learning rate parameter, momentum etc.**

from sklearn.model\_selection import GridSearchCV param\_grid = {

'hidden\_layer\_sizes': [(10,), (20,), (10, 10)],

'learning\_rate\_init': [0.001, 0.01, 0.1],

'momentum': [0.9, 0.95, 0.99]

}

grid\_search = GridSearchCV(MLPRegressor(max\_iter=1000), param\_grid, cv=5) grid\_search.fit(X\_train, y\_train)

print("Best parameters:", grid\_search.best\_params\_) best\_model = grid\_search.best\_estimator\_ y\_pred\_best = best\_model.predict(X\_test)

mse\_best = mean\_squared\_error(y\_test, y\_pred\_best) print("Mean Squared Error with best parameters:", mse\_best) **Output:**



# Implement SVM to classify the given data set and analyse the performance of the classifier.

from sklearn import svm

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, accuracy\_score data = load\_iris()

X = data.data y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) svm\_classifier = svm.SVC(kernel='rbf', C=1.0, gamma='scale') svm\_classifier.fit(X\_train, y\_train)

y\_pred = svm\_classifier.predict(X\_test) print("Accuracy:", accuracy\_score(y\_test, y\_pred)) print(classification\_report(y\_test, y\_pred))

Ex. 05 | Multiple Layer Analysis

3

**20CS2032L – MACHINE LEARNING TECHNIQUES**

**URK22AI1067**

# Output:

1. **Implement SVM for regression task.**

from sklearn import svm

from sklearn.datasets import fetch\_california\_housing from sklearn.model\_selection import train\_test\_split from sklearn.metrics import mean\_squared\_error

data = fetch\_california\_housing()

X = data.data y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) svm\_regressor = svm.SVR(kernel='rbf', C=1.0, gamma='scale') svm\_regressor.fit(X\_train, y\_train)

y\_pred = svm\_regressor.predict(X\_test) mse\_svm = mean\_squared\_error(y\_test, y\_pred) print("Mean Squared Error for SVM:", mse\_svm) **Output:**



# Improve the performance of the model by adjusting the hyperparameters such as number of hidden nodes, learning rate parameter, momentum etc.

from sklearn import svm

from sklearn.datasets import fetch\_california\_housing

from sklearn.model\_selection import train\_test\_split, GridSearchCV from sklearn.metrics import mean\_squared\_error

data = fetch\_california\_housing()

X = data.data y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) param\_grid = {

'C': [0.1, 1, 10],

'gamma': ['scale', 'auto'],

'kernel': ['linear', 'rbf', 'poly']

Ex. 05 | Multiple Layer Analysis

4

**20CS2032L – MACHINE LEARNING TECHNIQUES**

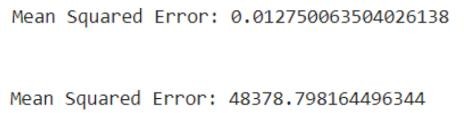
**URK22AI1067**

}

grid\_search = GridSearchCV(svm.SVR(), param\_grid, cv=5) grid\_search.fit(X\_train, y\_train)

print("Best parameters:", grid\_search.best\_params\_) best\_model = grid\_search.best\_estimator\_ y\_pred\_best = best\_model.predict(X\_test)

mse\_best = mean\_squared\_error(y\_test, y\_pred\_best) print("Mean Squared Error with best parameters:", mse\_best) **Output:**



# RESULT:

The above multiple layer perceptron is successfully executed.

Ex. 05 | Multiple Layer Analysis

5