

Name-Yash Gandhi TE IT Batch A 2017140014

Title : Error backPropagation using MLPClassifier

Theory-

Multilayer perceptrons train on a set of input-output pairs and learn to model the correlation (or dependencies) between those inputs and outputs. Training involves adjusting the parameters, or the weights and biases, of the model in order to minimize error. Backpropagation is used to make those weight and bias adjustments relative to the error, and the error itself can be measured in a variety of ways, including by root mean squared error.

Multi-layer Perceptron classifier:-

hidden_layer_sizes tuple, length = n_layers - 2, default=(100,)--The ith element represents the number of neurons in the ith hidden layer.

Activation {'identity', 'logistic', 'tanh', 'relu'}, default='relu'--Activation function for the hidden layer.

max_iter int default=200---Maximum number of iterations.

Implementation -

Steps:

1. Import the data and perform preprocessing
2. Removed the null values and split the data frame into 2 dataframe one containing all features other containing output coloumn
3. Perform the train test split.
4. Apply MLPclassifier and initialize a model using relevant parameters.
5. Trained the model using (100) epochs depending on your dataset
6. Evaluated the model using test data.
7. Performed predictions on test data.

Code-

```
import pandas as pd
import numpy as np

from sklearn.metrics import accuracy_score

df=pd.read_csv('glass.csv')

for col in df.columns:
    if df[col].dtype==object:
        df[col]=df[col].astype('category')
        df[col]=df[col].cat.codes
```

```
X = df.loc[:,df.columns != 'Type']
y = df['Type']
```

```
from sklearn.model_selection import train_test_split
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)
X_train = X_train.values
X_test = X_test.values
```

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=( 100, 100, 100, 100, 100, 100, 100, 100),activation='tanh', max_iter=1000)
mlp.fit(X_train, y_train.values.ravel())
```

```
mlp.classes_
```

```
print(mlp.coefs_)
```

```
predictions = mlp.predict(X_test)
from sklearn.metrics import accuracy_score
X_test
predictions
```

```
import matplotlib.pyplot as plt
```

```
print("The accuracy is %s" %(accuracy_score(y_test,predictions)))
```

```
plt.ylabel('LOSS_FUNCTION')
plt.xlabel('ITERATIONS')
```

```
plt.plot(mlp.loss_curve_)
plt.show()
```

Output- Dataset-to predict glass type

```
In [1]: import pandas as pd
import numpy as np

In [2]: from sklearn.metrics import accuracy_score

In [3]: df=pd.read_csv('glass.csv')

In [4]: for col in df.columns:
        if df[col].dtype==object:
            df[col]=df[col].astype('category')
            df[col]=df[col].cat.codes

In [5]: X = df.loc[:,df.columns != 'Type']
        y = df['Type']

In [7]: from sklearn.model_selection import train_test_split
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)
X_train = X_train.values
X_test = X_test.values
```

Training model-

Firefox Web Browser

lab1-Copy - Mozilla Firefox

Inbox (92) | Sardar Patel | ml-lab (yash) | lab1-Copy X | Visualization | Machine Learning | scikit learn | matplotlib | python - In | sklearn.neu |

https://mlab-yash4gandhi.notebooks.azure.com/j/notebook/... Search

Microsoft Azure Notebooks Preview My Projects Help yash4gandhi

Powered by Jupyter lab1-Copy Last Checkpoint: 23 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.6

```
In [8]: from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=( 100, 100, 100, 100, 100, 100, 100),activation='tanh', max_iter=1000)
mlp.fit(X_train, y_train.values.ravel())

mlp.classes_

Out[8]: array([1, 2, 3, 5, 6, 7])

In [14]: print(mlp.coefs_)

-8.28391580e-03, -1.85726253e-02, 9.38099057e-02,
8.15899061e-02, -6.80677234e-02, 2.75692182e-01,
-8.77848489e-02, -1.31681928e-01, -5.89340275e-02,
6.43194146e-02, -2.36796790e-01, 1.71258110e-01,
1.16157922e-01, 1.99878304e-01, -3.25946967e-02,
-7.78515805e-02, 8.26561739e-03, 1.26529122e-01,
2.42281327e-02, 2.42284434e-01, -1.02126294e-02,
7.81202197e-02, 9.45434690e-02, 6.99675036e-02,
1.61256173e-02, -3.48864884e-02, -2.84615925e-02,
-7.11768414e-02, -1.10548706e-02, 8.97203010e-03,
-3.12528614e-01, -1.19511419e-01, -1.06808802e-01,
-1.04748123e-01, 1.69439709e-01, -9.38075650e-03,
-1.78810685e-01, -1.15380331e-01, -6.20052695e-03,
-2.10561703e-02, 2.30597932e-01, -2.56823260e-02,
3.87298096e-02, 6.21851697e-02, 1.05318064e-01,
-1.46992643e-01, 1.04967543e-01, 1.89118911e-01,
-9.71563416e-02, 1.12035021e-01, 1.65482770e-01
```

Page 1 of 1 | 15 words, 130 characters | Default Style | English (India) | I | 90%

Firefox Web Browser

lab1-Copy - Mozilla Firefox

Inbox (92) | Sardar Patel | ml-lab (yash) | lab1-Copy X | Visualization | Machine Learning | scikit learn | matplotlib | python - In | sklearn.neu |

https://mlab-yash4gandhi.notebooks.azure.com/j/notebook/... Search

Microsoft Azure Notebooks Preview My Projects Help yash4gandhi

Powered by Jupyter lab1-Copy Last Checkpoint: 26 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.6

```
In [15]: predictions = mlp.predict(X_test)
from sklearn.metrics import accuracy_score

In [24]: predictions

Out[24]: array([2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 5, 1, 6, 1, 2,
2, 7, 2, 6, 7, 1, 1, 2, 7, 1, 1, 6, 1, 7, 1, 1, 1, 1, 2, 7, 5, 2,
1, 7, 1, 2, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 6, 2, 2, 1, 1, 7, 2, 1,
2, 1, 1, 5, 6])

In [21]: import matplotlib.pyplot as plt

In [22]: print("The accuracy is %s" %(accuracy_score(y_test,predictions)))

The accuracy is 0.6056338028169014

In [23]: plt.ylabel('cost')
plt.xlabel('iterations')
plt.title("Learning rate =" + str(0.001))
plt.plot(mlp.loss_curve_)
plt.show()
```

Learning rate =0.001

Page 1 of 1 | 15 words, 130 characters | Default Style | English (India) | I | 90%

Predictions-

Firefox Web Browser

lab1-Copy - Mozilla Firefox

Inbox (92) | In Sardar Patel | ml-lab (yash) | lab1-Copy X | Visualization | Machine Learning | scikit learn | matplotlib | python - Inve | sklearn.neu | +

https://mllab-yash4gandhi.notebooks.azure.com/j/notebook

Microsoft Azure Notebooks Preview My Projects Help yash4gandhi

Powered by jupyter lab1-Copy Last Checkpoint: 36 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.6

In [31]: X_test

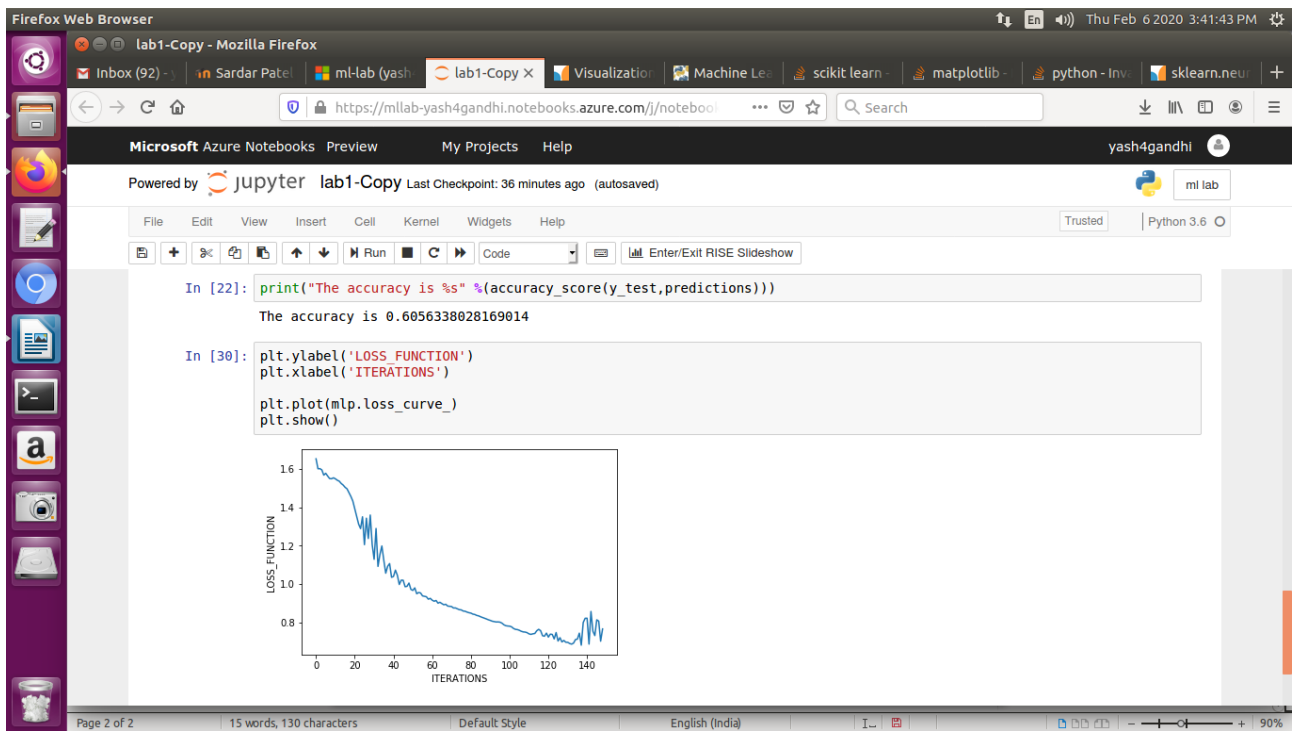
```
Out[31]: array([[1.51674e+00, 1.27900e+01, 3.52000e+00, 1.54000e+00, 7.33600e+01,
6.60000e-01, 7.90000e+00, 0.00000e+00, 0.00000e+00],
[1.52099e+00, 1.36900e+01, 3.59000e+00, 1.12000e+00, 7.19600e+01,
9.00000e-02, 9.40000e+00, 0.00000e+00, 0.00000e+00],
[1.51860e+00, 1.33600e+01, 3.43000e+00, 1.43000e+00, 7.22600e+01,
5.10000e-01, 8.60000e+00, 0.00000e+00, 0.00000e+00],
[1.51736e+00, 1.27800e+01, 3.62000e+00, 1.29000e+00, 7.27900e+01,
5.90000e-01, 8.70000e+00, 0.00000e+00, 0.00000e+00],
[1.52227e+00, 1.41700e+01, 3.81000e+00, 7.80000e-01, 7.13500e+01,
0.00000e+00, 9.69000e+00, 0.00000e+00, 0.00000e+00],
[1.51869e+00, 1.31900e+01, 3.37000e+00, 1.18000e+00, 7.27200e+01,
5.70000e-01, 8.83000e+00, 0.00000e+00, 1.60000e-01],
[1.51743e+00, 1.22000e+01, 3.25000e+00, 1.16000e+00, 7.35500e+01,
6.20000e-01, 8.90000e+00, 0.00000e+00, 2.40000e-01],
[1.51763e+00, 1.28000e+01, 3.66000e+00, 1.27000e+00, 7.30100e+01,
6.00000e-01, 8.56000e+00, 0.00000e+00, 0.00000e+00],
[1.51618e+00, 1.35300e+01, 3.55000e+00, 1.54000e+00, 7.29900e+01,
3.90000e-01, 7.78000e+00, 0.00000e+00, 0.00000e+00],
[1.51646e+00, 1.34100e+01, 3.55000e+00, 1.25000e+00, 7.28100e+01,
```

In [24]: predictions

```
Out[24]: array([2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 5, 1, 6, 1, 2,
2, 7, 2, 6, 7, 1, 1, 2, 7, 1, 1, 6, 1, 7, 1, 1, 1, 1, 2, 7, 5, 2,
```

SRCNN, EE "glass.csv" selected (10.1 kB)

Loss function curve



Conclusion-

Using Backpropagation technique in MLPClassifier hidden layers were set appropriately to increase the accuracy.

8 hidden layers with each layer having 100 nodes were used .

100 iterations were used for the process .

1. On any other number of hidden layers the accuracy was less than 60%
2. Activation function tanh was used in this model.