S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 04

Aim: Implement Multilayer perceptron by providing different weights.

Name of Student : Shrutika Pradeep Bagdi

Roll No : CS22130

Semester/Year : VII Sem / IV Year

Academic Session : 2025-2026 [ODD]

Date of Performance :

Date of Submission :

AIM: Implement Multilayer perceptron by providing different weights.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- Develop a deeper understanding of the activation functions and its limitations;
- Know how to diagnose and apply corrections to some problems with the Activation functions found in real data;
- Use and understand applications of Activation functions in Neural Network;
- Develop a greater familiarity with a range of techniques and methods through a diverse set of theoretical and applied readings;
- Know where to go to learn more about the techniques in this class and those called for that were not covered in this class.

HARDWARE AND SOFTWARE REQUIREMENTS:

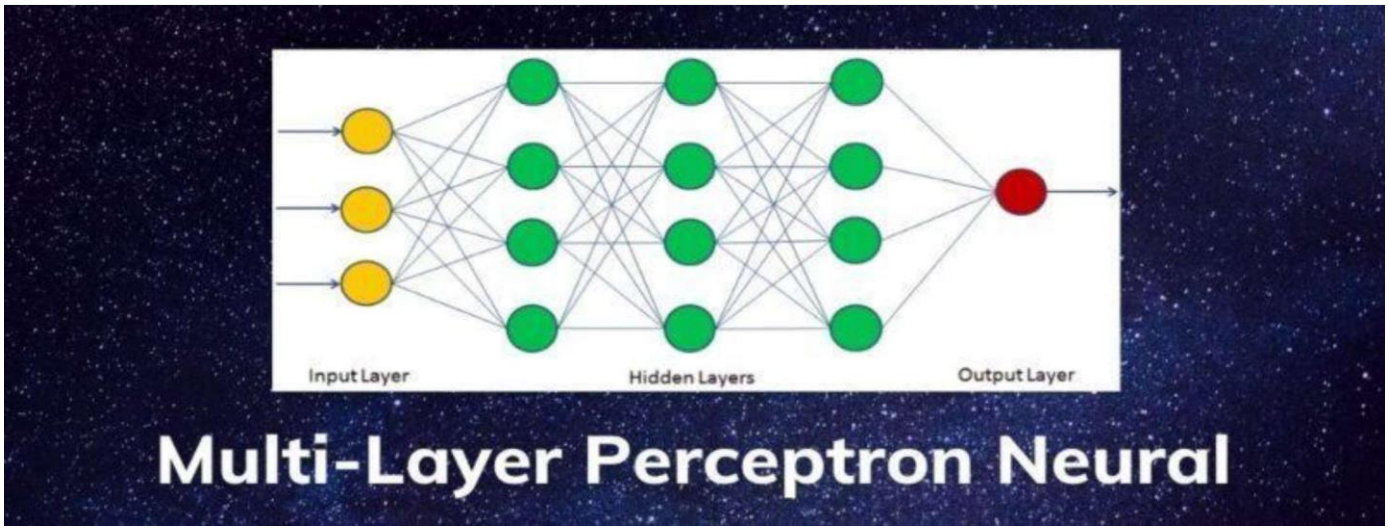
Hardware Requirement: Computer system with high configuration

Software Requirement: Google Colab, python 3

THEORY:

Multilayer Perceptron

The Multilayer Perceptron was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear.

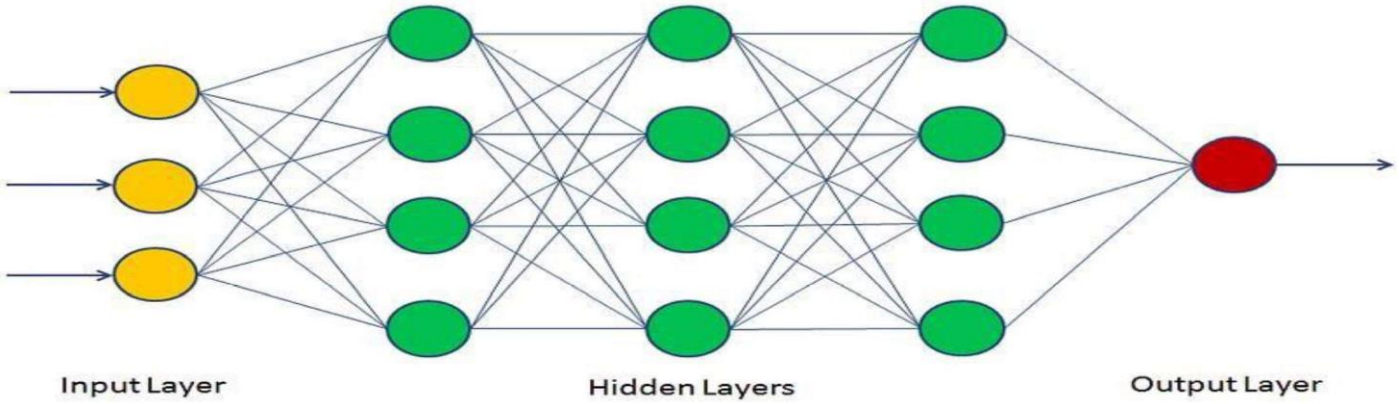


A Multilayer Perceptron has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function. Multi-Layer Perceptron(MLP) is the simplest type of artificial neural network. It is a combination of multiple perceptron models. Perceptrons are inspired by the human brain and try to simulate its functionality to solve

problems. In MLP, these perceptrons are highly interconnected and parallel in nature. This parallelization helpful in faster computation.

Perceptron

Perceptron was introduced by Frank Rosenblatt in 1950. It has the capability to learn complex things just like the human brain. Perceptron network consists of three units: Sensory Unit (Input Unit), Associator Unit (Hidden Unit), and Response Unit (Output Unit).



Multilayer Perceptron falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer.

Each layer isfeeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

But it has more to it.

If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to *learn* the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning.

How does MLP works?

The Perceptron consists of an input layer and an output layer which are fully connected. MLPs have the same input and output layers but may have multiple hidden layers in between as mentioned in the previous section figure.

If(y is not equal to t)

$W_i(new)=w_i(old) + \alpha t x_i;$

$b(new)=b(old) + \alpha t;$

else

$w_i(new)=w_i(old);$

$b(new)=b(old);$

Multi-Layer Perceptron trains model in an iterative manner. In each iteration, partial derivatives of the loss function used to update the parameters. We can also use regularization of the loss function to prevent overfitting in the model.

Let's start the model building hands-on in python.

Load dataset

Let's first load the required HR dataset using pandas' read CSV function. You can download data from the following link:

```
import numpy as np
import pandas as pd

data=pd read  csv('HR_                    csv') data head ()
```

Preprocessing: Label Encoding

Lots of machine learning algorithms require numerical input data, so you need to represent categorical columns in a numerical column. In order to encode this data, you could map each value to a number. e.g. Salary column's value can be represented as low:0, medium:1, and high:2. This process is known as label encoding. In sklearn, we can do this using LabelEncoder.

```
from sklearn import preprocessing

le = preprocessing LabelEncoder()

data[' ary']=le fit_transform( data[' ])
data['D ']=le fit_transform( data['D ])
```

Here, we imported the preprocessing module and created the Label Encoder object. Using this LabelEncoder object you fit and transform the "salary" and "Departments " column into the numeric column.

Split the dataset

In order to assess the model performance, we need to divide the dataset into a training set and a test set. Let's split dataset by using function train_test_split(). you need to pass basically 3 parameters features, target, and test_set size.

```
# Spliting data into Feature and
X=data[['satisfaction_level', 'last_evaluation', 'number roject', 'average_montly_hours', 'time_spend_company', 'Work_accident', 'promotion_last_5years', 'Departments ', 'salary']] y=data['left']
# Import train_test_split function
from sklearn.model_selection import train_test_split
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)  # 70% training and 30% test
```

Build Classification Model

Let's build an employee chum prediction model. Here, our objective is to predict chum using MLPClassifier.

First, import the MLPClassifier module and create MLP Classifier object using MLPClassifier() function. Then, fit your model on the train set using fit() and perform prediction on the test set using predict().

```
from sklearn.neural_network import MLPClassifier # Create model object
elf= MLPClassifier(hidden_layer_sizes=(6,5),
random_state=5,                verbose=True,
            leaming_rate_init=0.01)
# Fit data onto the model
elf.fit(X_train,y_train)
```

Parameters:

- **hidden_layer_sizes:** it is a tuple where each element represents one layer and its value represents the number of neurons on each hidden layer.
  learnin rate_init: It used to controls the step-size in updating the weights.
  activation: Activation function for the hidden layer. Examples, identity, logistic, tanh, and relu. by default, relu is used as an activation function.
- **random_state:** It defines the random number for weights and bias initialization.
- **verbose:** It used to print progress messages to standard output.

Make Prediction and Evaluate the Model

In this section, we will make predictions on the test dataset and assess model accuracy based on available actual labels of the test dataset.

PROGRAM CODE:

```
from google.colab import drive
drive.mount('/content/drive')
import numpy as np
import pandas as pd
path="/content/HR_comma_sep - HR_comma_sep.csv"
df=pd.read_csv(path)
df.head(5)
df.shape
df.isnull().sum()
df.info()
df.describe()
df.tail()
x = df.drop('left', axis=1)
y = df['left']
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
x_encoded = x.copy()
x_encoded['Department'] = le.fit_transform(x_encoded['Department'])
x_encoded['salary'] = le.fit_transform(x_encoded['salary'])
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_encoded, y, test_size=0.2, random_state=42)
```

```
from sklearn.neural_network import MLPClassifier
mlp=MLPClassifier(hidden_layer_sizes=(6,5),random_state=5,verbose=True,learning_rate_init=0.01)
mlp.fit(x_train, y_train)
from sklearn.metrics import accuracy_score
y_pred = mlp.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

OUTPUT (SCREENSHOT):

Aim: Implement Multilayer perceptron by providing different weights.

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
import numpy as np
import pandas as pd
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    path="/content/HR_comma_sep - HR_comma_sep.csv"
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
df=pd.read_csv(path)
df.head(5)
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    df.shape
```

```
(14999, 10)
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
df.isnull().sum()
```

	0
satisfaction_level	0
last_evaluation	0
number_project	0
average_montly_hours	0
time_spend_company	0
Work_accident	0
left	0
promotion_last_5years	0
Department	0
salary	0

dtype: int64

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
df.describe()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    #  o/p->left i/p->all other & 4. x= i/p feature y= o/p feature
    x = df.drop('left', axis=1)
    y = df['left']
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
#  Label encoding -> department, salary
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
x_encoded = x.copy()
x_encoded['Department'] = le.fit_transform(x_encoded['Department'])
x_encoded['salary'] = le.fit_transform(x_encoded['salary'])
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    #  Train test split
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x_encoded, y, test_size=0.2, random_state=42)
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    from sklearn.neural_network import MLPClassifier
    mlp=MLPClassifier(hidden_layer_sizes=(6,5),random_state=5,verbose=True,learning_rate_init=0.01)
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    mlp.fit(x_train, y_train)
```

```
Iteration 115, loss = 0.18366288
Iteration 116, loss = 0.16686251
Iteration 117, loss = 0.16162752
```

```
Iteration 160, loss = 0.15343184
Iteration 161, loss = 0.15183405
Iteration 162, loss = 0.16081026
Iteration 163, loss = 0.16869073
Iteration 164, loss = 0.17018256
Iteration 165, loss = 0.15595259
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

```
MLPClassifier(hidden_layer_sizes=(6, 5), learning_rate_init=0.01,
              random_state=5, verbose=True)
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
from sklearn.metrics import accuracy_score
y_pred = mlp.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.938
```

CONCLUSION:

Thus, successfully implement Multilayer perceptron by providing different weights.

DISCUSSION AND VIVA VOCE:

- What is the Activation Function?
- How many times of Activation Functions?
- Can you briefly talk about activation function?
- What are the applications of Activation function?
- Why do we use Activation functions in neural network?

REFERENCE:

- https://www.analvticsvidhva.com/blog/2022/10/multi-laver-perceptrons-notations-and-trainable- parameters/
- https://scikit-learn.org/stable/modules/neural_networks_supervised.html
- https://wandb.ai/wandbVmlldzo0NTIwNjQ1
- https://d2l.ai/chapter_multilayer-perceptrons/mlp.html