S. B. JAIN INSTITUTE OF TECHNOLOGY,

MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 05

Aim: Apply the knowledge and implement feedforward Neural Network.

Name of Student : _____

Roll No : _____

Semester/Year : VIIth Sem / IVth Year

Academic Session : 2025-2026 [ODD]

Date of Performance : _____

Date of Submission : _____

AIM: Apply the knowledge and implement feedforward Neural Network.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To develop a learning algorithm for feedforward neural networks, empowering the networks to be trained to capture the mapping implicitly.
- To develop learning algorithms for feedforward neural networks.
- Efficiently compute the gradient of the loss function with respect to the network weights.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement: Computer System with high configurations

Software Requirement: Google Colab

THEORY:

Feed forward neural network

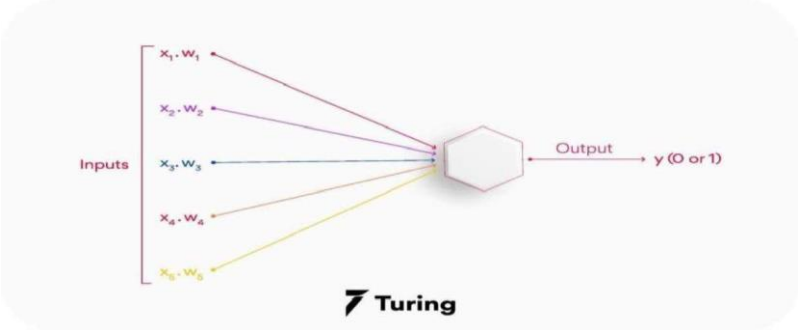
Feed forward neural networks are artificial neural networks in which nodes do not form loops. This type of neural network is also known as a multi-layer neural network as all information is only passed forward.

During data flow, input nodes receive data, which travel through hidden layers, and exit output nodes. No links exist in the network that could get used to by sending information back from the output node.

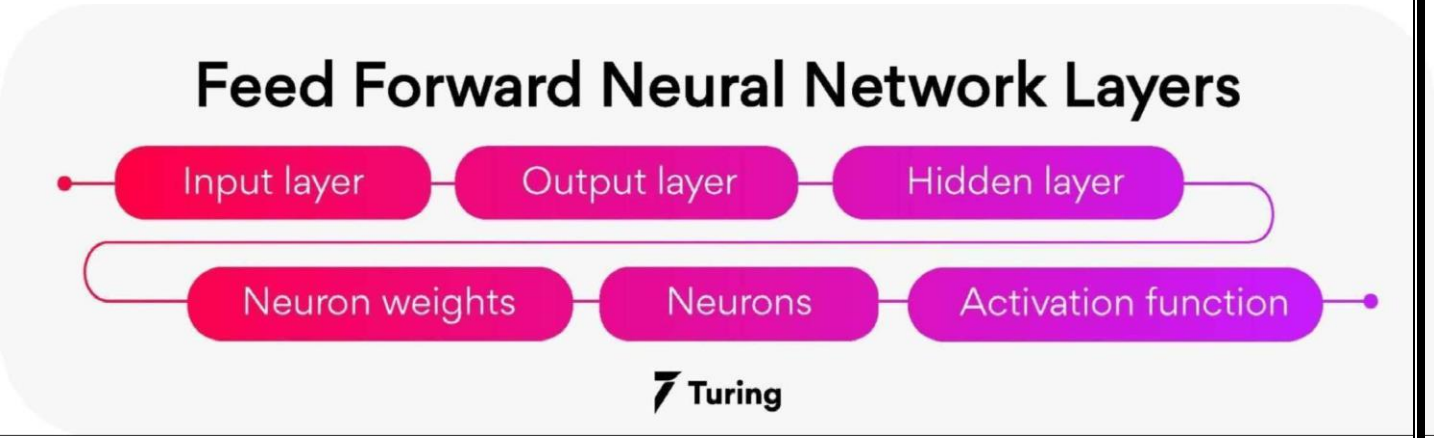
A feed forward neural network approximates functions in the following way:

- An algorithm calculates classifiers by using the formula $y = f^*(x)$.
- Input x is therefore assigned to category y.
- According to the feed forward model, $y = f(x; \theta)$. This value determines the closest approximation of the function.

Working principle of a feed forward neural network



Layers of feed forward neural network



- Input layer: The neurons of this layer receive input and pass it on to the other layers of the network. Feature or attribute numbers in the dataset must match the number of neurons in the input layer.
- Output layer: According to the type of model getting built, this layer represents the forecasted feature.
- Hidden layer: Input and output layers get separated by hidden layers. Depending on the type of model, there may be several hidden layers.

There are several neurons in hidden layers that transform the input before actually transferring it to the next layer. This network gets constantly updated with weights in order to make it easier to predict.

- Neuron weights: Neurons get connected by a weight, which measures their strength or magnitude. Similar to linear regression coefficients, input weights can also get compared.

Weight is normally between 0 and 1, with a value between 0 and 1.

- Neurons: Artificial neurons get used in feed forward networks, which later get adapted from biological neurons. A neural network consists of artificial neurons.

Neurons function in two ways:

First, they create weighted input sums, and

Second, they activate the sums to make them normal.

Activation functions can either be linear or nonlinear. Neurons have weights based on their inputs. During the learning phase, the network studies these weights.

- Activation Function: Neurons are responsible for making decisions in this area.

According to the activation function, the neurons determine whether to make a linear or nonlinear decision. Since it passes through so many layers, it prevents the cascading effect from increasing neuron outputs.

An activation function can be classified into three major categories: sigmoid, Tanh, and Rectified Linear Unit (ReLu).

- Sigmoid: Input values between 0 and 1 get mapped to the output values.
- Tanh: A value between -1 and 1 gets mapped to the input values.
- Rectified linear Unit: Only positive values are allowed to flow through this function. Negative values get mapped to 0.

Advantages of feed forward Neural Networks

- Machine learning can be boosted with feed forward neural networks' simplified architecture.
- Multi-network in the feed forward networks operate independently, with a moderated intermediary.
- Complex tasks need several neurons in the network.
- Neural networks can handle and process nonlinear data easily compared to perceptrons and sigmoid neurons, which are otherwise complex.
- A neural network deals with the complicated problem of decision boundaries.
- Depending on the data, the neural network architecture can vary. For example, convolutional neural networks (CNNs) perform exceptionally well in image processing, whereas recurrent neural networks (RNNs) perform well in text and voice processing.
- Neural networks need graphics processing units (GPUs) to handle large datasets for massive computational and hardware performance. Several GPUs get used widely in the market, including Kaggle Notebooks and Google Collab Notebooks.

Applications of feed forward neural networks

PROGRAM CODE:

```
from google.colab import drive
drive.mount('/content/drive')
path="/content/drive/MyDrive/Deep Learning/Practical 5/diabetes - diabetes.csv"
import numpy as np
import pandas as pd
df=pd.read_csv(path)
df.head(5)
df.shape
df.isnull().sum()
df.info()
df.describe()
df.tail()
X = df.drop('Outcome', axis=1)
y = df['Outcome']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(12, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
epochs = 50
history = model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_split=0.2)
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
```

OUTPUT (SCREENSHOT):

```
[ ] #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
    path="/content/drive/MyDrive/Deep Learning/Practical 5/diabetes - diabetes.csv"
```

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

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

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

```
(768, 9)
```

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

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    X = df.drop('Outcome', axis=1)
    y = df['Outcome']
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(12, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
#Name: Shrutika Pradeep Bagdi_CS22130
epochs = 50
history = model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_split=0.2)
```

```
Epoch 1/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.7214 - loss: 0.5489 - val_accuracy: 0.6585 - val_loss: 0.5977
Epoch 2/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7508 - loss: 0.5012 - val_accuracy: 0.6911 - val_loss: 0.6049
Epoch 3/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7200 - loss: 0.5541 - val_accuracy: 0.6016 - val_loss: 0.6662
Epoch 4/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7271 - loss: 0.5293 - val_accuracy: 0.6992 - val_loss: 0.5702
Epoch 5/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7667 - loss: 0.5138 - val_accuracy: 0.6992 - val_loss: 0.5841
Epoch 6/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7783 - loss: 0.4976 - val_accuracy: 0.6585 - val_loss: 0.6016
Epoch 7/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7711 - loss: 0.5103 - val_accuracy: 0.7154 - val_loss: 0.5805
Epoch 8/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7706 - loss: 0.5117 - val_accuracy: 0.7073 - val_loss: 0.5650
Epoch 9/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7649 - loss: 0.4980 - val_accuracy: 0.7073 - val_loss: 0.5839
Epoch 10/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7891 - loss: 0.4979 - val_accuracy: 0.6748 - val_loss: 0.5926
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    history = model.fit(X_train, y_train, epochs=100, validation_split=0.2, verbose=0)
```

```
[ ] #Name: Shrutika Pradeep Bagdi_CS22130
    loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
    print(f'Test Loss: {loss:.4f}')
    print(f'Test Accuracy: {accuracy:.4f}')
```

```
Test Loss: 0.7857
Test Accuracy: 0.7013
```

CONCLUSION:

Thus successfully apply and implement the knowledge of feedforward neural network.

DISCUSSION AND VIVA VOCE:

- What is Feedforward Network?
- Why do we need Feedforward in a neural network?
- What is the use of the Activation function?
- How Feedforward can use for diabetics disease prediction?

REFERENCE:

- https://medium.com/hackernoon/building-a-feedforward-neural-network-from-scratch-in-pythond3526457156b
- https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-feed-forward-network-indeep-
- learning/
- https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network
- https://www.turing.com/kb/mathematical-formulation-of-feed-forward-neural-network
- https://www.javatpoint.com/pytorch-feed-forward-process-in-deep-neural-network
- https://builtin.com/data-science/feedforward-neural-network-intro