Feed-forward Neural Network

A Feed-forward Neural Network (FNN) is a type of artificial neural network where the connections between the nodes do not form a cycle. It consists of an input layer, one or more hidden layers, and an output layer. Each layer contains units (neurons) that process input data through weights and biases. The primary characteristic of an FNN is that information moves in one direction—forward—from the input nodes, through the hidden nodes (if any), and to the output nodes.

Key points about Feed-forward Neural Networks:

Architecture: Comprised of layers (input, hidden, and output).

Activation Functions: Apply nonlinear transformations to inputs, allowing the network to model complex relationships.

Training: Typically done using backpropagation and gradient descent.

Application: Widely used in classification, regression, and function approximation problems.

Practical Example in Cybersecurity

Problem: Detecting whether a network connection is legitimate or part of a cyber attack.

Data: We'll use a simplified dataset with features such as connection duration, protocol type, and number of bytes transferred.

Python Code:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical

# Generating a simple dataset
data = {
   'duration': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
   'protocol_type': [1, 1, 2, 2, 1, 2, 2, 1, 1, 2],
   'bytes': [1000, 1500, 1600, 1100, 1800, 1200, 1300, 1400, 1700, 1900],
   'attack': [0, 0, 1, 0, 1, 0, 1, 1, 0, 1]
}

# Converting to DataFrame
```

```
df = pd.DataFrame(data)

# Splitting data
X = df[['duration', 'protocol_type', 'bytes']]
y = to_categorical(df['attack'])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardizing the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Building the FNN model
model = Sequential()
model.add(Dense(12, input_dim=3, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(2, activation='softmax'))

# Compiling the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Training the model
model.fit(X_train, y_train, epochs=150, batch_size=10, verbose=0)

# Evaluating the model
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Model Accuracy: {accuracy * 100:.2f}%')
```