Experiment No 7

Aim: Use FCNN with only one neuron and plotting FCNN with one hidden layer and plotting.

Theory:

A Fully Connected Neural Network (FCNN), also known as a Feedforward Neural Network or Multilayer Perceptron (MLP), is an artificial neural network made of layers where each neuron in one layer is connected to every neuron in the next.

Key components include:

- Neurons/Nodes: Fully connected across layers.
- Layers: Comprising input, hidden, and output layers.
- Weights and Biases: Each connection has weights, and neurons have biases that are optimized during training.
- Activation Functions: Apply non-linearity to neuron outputs, with common functions like Sigmoid, ReLU, and Tanh.
- Forward Propagation: The process of passing input through layers to produce an output.
- Loss/Cost Function: Measures prediction error, with the goal to minimize it.
- Backpropagation: Updates weights and biases using gradients to reduce the loss, typically with optimization algorithms like gradient descent.
- Training: Iterative process of feeding data, computing loss, and adjusting weights using backpropagation.

Architecture:

- The input layer receives data, hidden layers transform it, and the output layer provides predictions.
- Weights and biases are adjusted during training

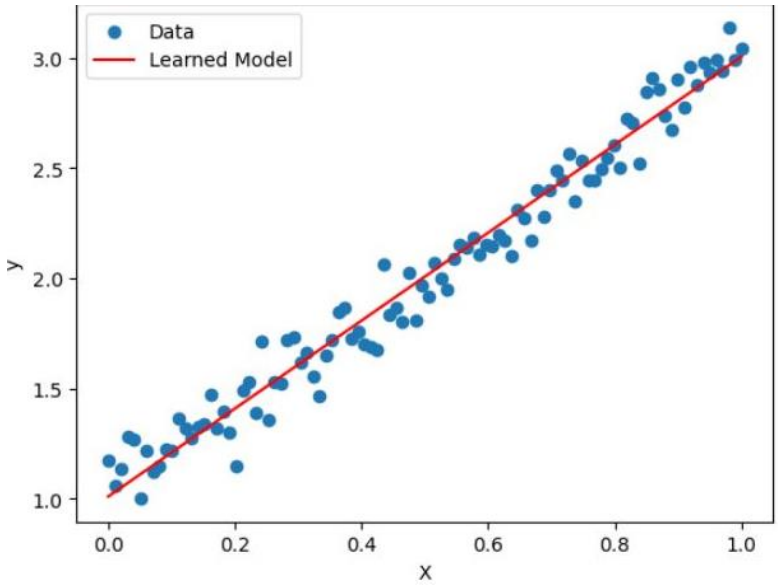
Regularization and Optimization:

- Techniques like dropout and weight decay prevent overfitting, and optimizers like Adam and SGD adjust the parameters effectively during training.

FCNNs are foundational in solving diverse machine learning tasks, with performance dependent on architecture, hyperparameters, and data quality.

Fully Connected Neural Network (FCNN)

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate some example data
np.random.seed(0)
X = np.linspace(0, 1, 100)
y = 2 * X + 1 + 0.1 * np.random.randn(100)  # Linear function with noise

# Define the FCNN model
class SimpleFCNN:
    def __init__(self):
        self.weights = np.random.randn(2)  # Weights for the input and bias
        self.learning_rate = 0.01

    def predict(self, x):
        return np.dot(x, self.weights)

    def train(self, x, y):
        y_pred = self.predict(x)
        error = y_pred - y
        gradient = np.dot(x, error)
        self.weights -= self.learning_rate * gradient

# Training the model
model = SimpleFCNN()
for epoch in range(1000):
    for xi, yi in zip(X, y):
        model.train(np.array([xi, 1]), yi)

# Making predictions
y_pred = [model.predict(np.array([xi, 1])) for xi in X]

# Plot the data and the learned model
plt.scatter(X, y, label='Data')
plt.plot(X, y_pred, color='red', label='Learned Model')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```



Conclusion: A Fully Connected Neural Network (FCNN) with only one neuron is essentially a linear regression model, while an FCNN with one hidden layer can capture more complex patterns.