Experiment No. 9

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

Theory:

FCNN Overview: Fully Connected Neural Network (FCNN) is a type of artificial neural network characterized by dense or fully connected layers. FCNNs are also known as feedforward neural networks or multilayer perceptrons (MLPs).

Key Components of FCNNs:

- **Neurons/Nodes:** FCNNs consist of layers of interconnected nodes, where each node is often called a "neuron." Neurons in a layer are fully connected to neurons in the adjacent layers.
- **Layers:** An FCNN typically consists of an input layer, one or more hidden layers, and an output layer. The input layer takes raw input data, and the output layer produces predictions. Hidden layers transform the input data as it passes through the network.
- **Weights and Biases:** Each connection between neurons has an associated weight, representing the strength of the connection. Neurons also have biases that can be adjusted during training.
- **Activation Functions:** Each neuron applies an activation function to the weighted sum of its inputs. Common activation functions include sigmoid, ReLU (Rectified Linear Unit), and tanh.
- **Forward Propagation:** Data is passed through the network layer by layer, with each neuron applying its activation function and passing the result to the next layer until the output layer produces the final prediction.
- **Loss/Cost Function:** A loss or cost function measures how well the network's predictions match the target values, quantifying the error between predictions and actual values.
- **Backpropagation:** This process updates the weights and biases in the network to reduce the loss. It calculates the gradients of the loss with respect to each weight and bias.
- **Training:** Involves presenting a dataset, forwarding the data through the network, computing the loss, and using backpropagation to update parameters, typically repeated for multiple epochs.

Architecture:

- **Input Layer:** Receives raw data, with each neuron corresponding to a feature.
- **Hidden Layers:** Intermediate layers situated between the input and output layers. The number of hidden layers and neurons in each layer are configurable hyperparameters.
- **Output Layer:** Produces final predictions. The number of neurons depends on the problem (e.g., one neuron for binary classification, multiple for multiclass).

Weights and Biases:

- **Weights:** Adjusted during training to optimize performance.
- **Biases:** Allow the network to shift the activation function, learned during training.

Activation Functions: Introduce non-linearity to the network. Common functions include:

- **Sigmoid:** Outputs values in the range (0, 1).
- **Hyperbolic Tangent (tanh):** Outputs values in the range (-1, 1).
- **ReLU:** Outputs input for positive values and zero for negative values, widely used in hidden layers.

Forward Propagation: Data passes through the network layer by layer, with each neuron computing a weighted sum of its inputs and applying an activation function.

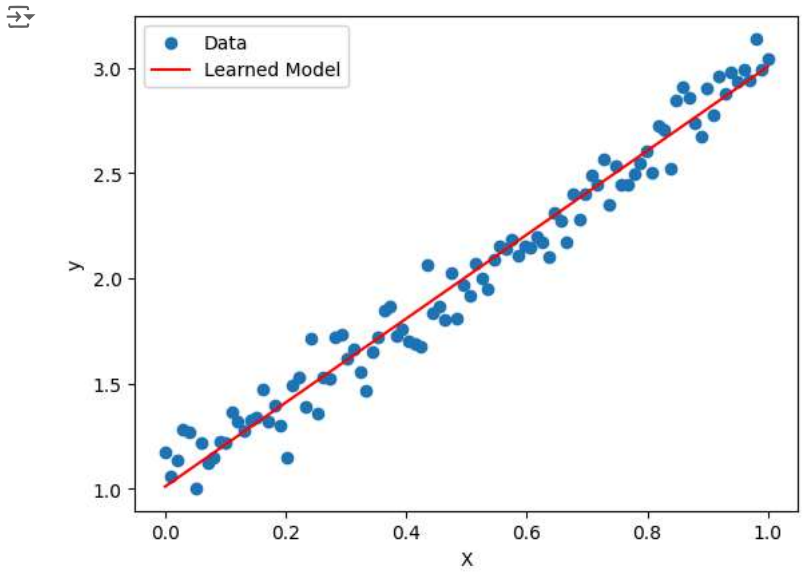
Loss Function: Quantifies the error between the model's predictions and actual target values, with the goal of minimizing this loss during training.

Backpropagation: Updates weights and biases to minimize loss, computing gradients and using optimization algorithms like gradient descent.

Training: Involves presenting a labeled dataset, performing forward and backward passes, and iteratively updating parameters for multiple epochs until convergence.

Regularization and Optimization: To prevent overfitting, techniques like dropout, weight decay (L1 and L2 regularization), and early stopping can be applied. Optimization algorithms can include variations of gradient descent like stochastic gradient descent (SGD), Adam, RMSprop, etc. FCNNs are powerful and flexible, but their performance depends on factors like architecture, hyperparameters, and the quality and quantity of training data. They have been instrumental in solving a wide range of machine learning and deep learning tasks, making them a fundamental tool in the field of artificial intelligence.

```
Pimport 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. In a simple example, the one-neuron FCNN is limited to modeling linear relationships, whereas a one-hidden-layer FCNN can capture non-linear patterns, making it more versatile for various tasks.