Name of Student : AHMED ALI ANSARI **ID No : 1402-2020**

Task :

1.Build the above code?

ANSWER:

In [1]:
```
# Import python libraries required in this example:
import numpy as np
from scipy.special import expit as activation_function
from scipy.stats import truncnorm

# DEFINE THE NETWORK

# Generate random numbers within a truncated (bounded)
# normal distribution:
def truncated_normal(mean=0, sd=1, low=0, upp=10):
    return truncnorm(
        (low - mean) / sd, (upp - mean) / sd, loc=mean, scale=sd)

# Create the 'Nnetwork' class and define its arguments:
# Set the number of neurons/nodes for each layer
# and initialize the weight matrices:
class Nnetwork:

    def __init__(self,
                 no_of_in_nodes,
                 no_of_out_nodes,
                 no_of_hidden_nodes,
                 learning_rate):
        self.no_of_in_nodes = no_of_in_nodes
        self.no_of_out_nodes = no_of_out_nodes
        self.no_of_hidden_nodes = no_of_hidden_nodes
        self.learning_rate = learning_rate
        self.create_weight_matrices()

    def create_weight_matrices(self):
        """ A method to initialize the weight matrices of the neural network"""
        rad = 1 / np.sqrt(self.no_of_in_nodes)
```

Name of Student : AHMED ALI ANSARI **ID No : 1402-2020**

```
        X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
        self.weights_in_hidden = X.rvs((self.no_of_hidden_nodes,
                                        self.no_of_in_nodes))
        rad = 1 / np.sqrt(self.no_of_hidden_nodes)
        X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
        self.weights_hidden_out = X.rvs((self.no_of_out_nodes,
                                         self.no_of_hidden_nodes))

    def train(self, input_vector, target_vector):
        pass  # More work is needed to train the network

    def run(self, input_vector):
        """
        Running the network with an input vector 'input_vector'.
        'input_vector' can be tuple, list, or ndarray
        """
        # Turn the input vector into a column vector:
        input_vector = np.array(input_vector, ndmin=2).T
        # activation_function() implements the expit function,
        # which is an implementation of the sigmoid function:
        input_hidden = activation_function(self.weights_in_hidden @ input_vector)
        output_vector = activation_function(self.weights_hidden_out @ input_hidden)
        return output_vector

# RUN THE NETWORK AND GET A RESULT

# Initialize an instance of the class:
simple_network = Nnetwork(no_of_in_nodes=2,
                          no_of_out_nodes=2,
                          no_of_hidden_nodes=4,
                          learning_rate=0.6)

# Run simple_network for arrays, lists, and tuples with shape (2):
# and get a result:
result = simple_network.run([(3, 4)])
print(result)
```

```
[[0.38656355]
 [0.45347203]]
```