**Q 4 Pseudo Code**

1. Initialize w (R73x1) randomly between [-1,1]
2. Initialize the learning rate randomly between (0,1]
3. Set epoch = 0
4. Initialize mse[epoch] = MSE of data with weights initialized in the step 1 and store it in an array “cf”
5. We will be using online learning i.e. weights are updated for every input
6. Feed first input and find the local field values v(R24x1) of the 24 neurons of the hidden layer (the values that are applied to the activation functions of the 24 neurons) and store them in an array.
7. Use tanh(v) as the activation function for the 24 neurons and no activation function for the output neuron.
8. Apply the output values of the 24 neurons and store the output in a variable y

Back Propagation:

1. Find the difference in the actual and predicted output I.e. d[0] – y
2. Multiply the above value with each of the weights between the 24 neurons and the output and save the product vector as p
3. Find the values of the derivatives of the activation functions at their corresponding local fields i.e d(tanh(v)/dv at **Vi** for i in {1,2,3,…..24} and save it as m
4. Find the matrix product of m and p and save it as c
5. The partial derivatives with respect to weights between input layer and the hidden layer = -(c[i]) for i in {1,2,3,… 24}.
6. And the partial derivatives w.r.t weights between the hidden layer and the output layer = -(d[0] – y)
7. Save all the partial derivatives as g vector
8. Update the weights as follows:

W = W - (learning rate)\* g

1. Repeat the steps from 6 to 16 for all the inputs (use the updated weights for next input)
2. Repeat the steps from 4 to 17 until some stopping criterion is reached. The stopping criterion I used here is number of epochs.
3. For each epoch, find the mse using the updated weights and add it to the array “cf”.
4. If the mse with the updated weights in any epoch is greater than that of previous epoch, start from the beginning with new learning rate = (old learning rate) \* 0.9