(Topics Covered: Perceptron, MLP, RBFN and deep Neural Network)

Kriti Bapnad

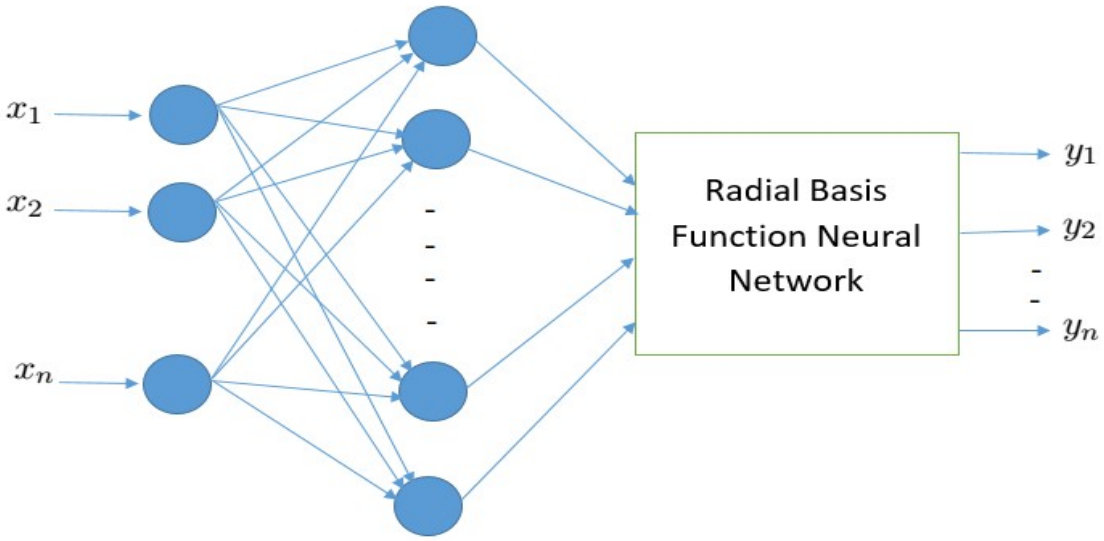
1. Implement AND, OR, NOT, ANDNOT, NAND, NOR, XOR and XNOR gates using Hebbian learning rule based perceptron algorithm. Evaluate the optimal weight parameters for each logic gate. Plot the cost function vs number of iterations for each logic gate.

2. Implement the multilayer feed forward neural network (MFNN) (three layers: input layer, hidden layer and output layer) for the classification using the loop based implementation algorithm. (Please use dataset.xlsx and hold out cross validation with 70% as training and 30 % as testing instances selection). Plot the cost function vs number of iterations. Evaluate the optimal number of hidden layer neurons based on grid search approach with the maximum accuracy for the test instances as the objective function.)

3. Implement the MFNN (4 layers: input, hidden layer 1, hidden layer 2 and output layer) for the classification using the vectorization based algorithm. (Please use dataset.xlsx and hold out cross validation with 70% as training and 30 % as testing instances selection). Plot the cost function vs number of iterations. Evaluate the optimal number of hidden layer neurons based on grid search approach with the maximum accuracy for the test instances as the objective function.)

4. Implement the radial basis function neural network (RBFN) for the classification. (Please use dataset.xlsx and hold out cross validation with 70% as training and 30 % as testing instances selection). Evaluate the optimal number of hidden layer neurons based on grid search approach with the maximum accuracy for the test instances as the objective function.)

5. Implement a deep neural network as shown in the following figure. You can use back propagation algorithm, non-linear basis function evaluation and RBFNN weight extraction approach for evaluating optimal weights from input to hidden layers. (Please use dataset.xlsx and hold out cross validation with 70% as training and 30 % as testing instances selection).


x_1
x_2
x_n
Radial Basis Function Neural Network
y_1
y_2
y_n