TRIBHUVAN UNIVERSITY

Sagarmatha College of Science & Technology

Lab-Report On: Neural Network

Lab-Report No.: 06

Date: 2077-12-01

SUBMITTED BY	**SUBMITTED TO**
Name: Ramesh Neupane	*CSIT Department*
Roll no.: 37	

Qeustion 01

Write a Python program to achieve XOR function using RBFNN. Use two RBF centers.

Source Code

```
import numpy as np
import math

b = 0
alpha = 1

def train_perceptron(x, t, w):
    for i in range(len(x)):
        global b
        v = sum(x[i] * w) + b
        y = hard_limiter(v)
        dw = alpha * (t[i] - y) * x[i]
        w = np.add(w, dw)
        db = alpha * (t[i] - y)
        b = b + db
    return w

def predict_perceptron(x, w):
    z = x * w
    tx = sum(z) + b
    y = hard_limiter(tx)
    return y

def hard_limiter(x):
    if x > 0:
        return 1
    elif x < 0:
        return -1
    else:
        return 0

def RBF(t):
    tx = []
    for x in t:
        r = []
        d1 = np.sum(np.square(c1 - x))
        d2 = np.sum(np.square(c2 - x))
        phi1 = math.exp(d1 * (-1))
        phi2 = math.exp(d2 * (-1))
        r.append(phi1)
        r.append(phi2)
        tx.append(r)
    return tx

trainx = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
trainy = np.array([-1, 1, 1, -1])
c1 = np.array([0, 0])
c2 = np.array([1, 1])

tx = RBF(trainx)
phi = np.array(tx)
print("PHI matrix: ", *phi)

wt = np.array([0, 0])
print("\n***Training***")
print("--------------")
for i in range(50):
    wt = train_perceptron(phi, trainy, wt)
print("Final weights: ", *wt)
print("Bias: ", b)

print("\n***Prediction***")
print("----------------")
for x in phi:
    y = predict_perceptron(x, wt)
    print("Output: ", y)
```

Output

```
PHI matrix:  [1.         0.13533528] [0.36787944 0.36787944] [0.36787944 0.36787944] [0.13533528
1.        ]

***Training***
--------------
Final weights:  -3.5242462442515112 -2.6595815274881245
Bias:  3

***Prediction***
----------------
Output:  -1
```

```
Output:  1
Output:  1
Output:  -1
```

Question 02

Write the Python program to achieve XOR function using RBFNN. Use four RBF centers.

Source Code

```
import numpy as np
import math

b = 0
alpha = 1

def train_perceptron(x, t, w):
    for i in range(len(x)):
        global b
        v = sum(x[i] * w) + b
        y = hard_limiter(v)
        dw = alpha * (t[i] - y) * x[i]
        w = np.add(w, dw)
        db = alpha * (t[i] - y)
        b = b + db
    return w

def predict_perceptron(x, w):
    z = x * w
    tx = sum(z) + b
    y = hard_limiter(tx)
    return y

def hard_limiter(x):
    if x > 0:
        return 1
    elif x < 0:
        return -1
    else:
        return 0
```

```
def RBF(t):
    tx = []
    for x in t:
        r = []
        d1 = np.sum(np.square(c1 - x))
        d2 = np.sum(np.square(c2 - x))
        d3 = np.sum(np.square(c3 - x))
        d4 = np.sum(np.square(c4 - x))
        phi1 = math.exp(d1 * (-1))
        phi2 = math.exp(d2 * (-1))
        phi3 = math.exp(d3 * (-1))
        phi4 = math.exp(d4 * (-1))
        r.append(phi1)
        r.append(phi2)
        r.append(phi3)
        r.append(phi4)
        tx.append(r)
    return tx

trainx = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
trainy = np.array([-1, 1, 1, -1])
c1 = np.array([0, 0])
c2 = np.array([0, 1])
c3 = np.array([1, 0])
c4 = np.array([1, 1])

tx = RBF(trainx)
phi = np.array(tx)
print("PHI matrix: ", *phi)

wt = np.array([0, 0, 0, 0])
print("\n***Training***")
print("--------------")
for i in range(50):
    wt = train_perceptron(phi, trainy, wt)
print("Final weights: ", *wt)
print("Bias: ", b)

print("\n***Prediction***")
print("----------------")
for x in phi:
    y = predict_perceptron(x, wt)
    print("Output: ", y)
```

Ouptut

```
PHI matrix:  [1.         0.36787944 0.36787944 0.13533528] [0.36787944 1.         0.13533528
0.36787944] [0.36787944 0.13533528 1.         0.36787944] [0.13533528 0.36787944 0.36787944
1.        ]

***Training***
--------------
Final weights:  -5.794916810724737 4.427037369553295 4.427037369553294 -
4.93025209396135
Bias:  -1

***Prediction***
----------------
Output:  -1
Output:  1
Output:  1
Output:  -1
```

Conclusion

Hence, we are able to achieve XOR function using RBFNN (Radial Basis Function Neural Network) with two and four RBF centers respectively.