Κεφάλαιο 8

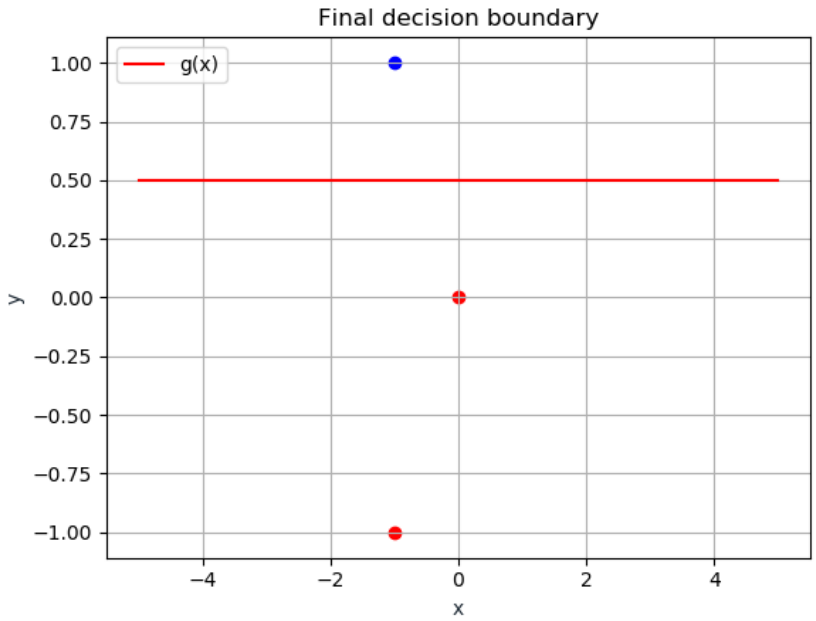
Πρόβλημα 8

8.1 A. Use MATLAB (or Python) to initialize and train the network using the perceptron learning rule

Refer to Python Code .

8.2 B. Final decision boundary and graphical classication

As it is demonstrated graphically all patterns are correctly classified.



Εικόνα 8.1: *Final decision boundary*

8.3 C. Perceptron rule classification dependency on initial weights

The perceptron rule (given enough iterations) will always learn to correctly classify the patterns in this training set, no matter what initial weights we use, because the perceptron learning algorithm is a linear classifier. If our data is separable by a hyperplane, then the perceptron will always converge. It will never converge if the data is not linearly separable. In practice, the perceptron learning algorithm can be used on data that is not linearly separable, but some extra parameter must be defined in order to determine under what conditions the algorithm should stop trying to fit the data. For example, we could set a maximum number of iterations (or epochs) for the algorithm to run, or we could set a threshold for the maximum number of allowed misclassifications.

8.4 Python Code

```
# CE418: Neuro-fuzzy Computing
#
#    Evangelos Stamos
#    02338
#    estamos@e-ce.uth.gr

# Problem-08
#
# suppose : linearly seperable classes, M = 2, labeled input training set data
#

import numpy as np

import matplotlib.pyplot as plt

lr = 1 #learning rate

bias = 0.5 #value of bias

weights = [1, 0, bias] #weights generated in a list (3 weights in total for 2 neurons and the bias)

def Perceptron(input1, input2, output) :
    outputP = input1*weights[0] + input2*weights[1] + bias
    if outputP > 0 : #activation function (here Heaviside)
        outputP = 1
    else :
```

```
27          outputP = 0
28      error = output - outputP
29      weights[0] += error * input1 * lr
30      weights[1] += error * input2 * lr
31      weights[2] += error
32
33
34
35
36  for i in range(2) :
37          Perceptron(-1,-1,0)#p_1
38          Perceptron(0,0,0)  #p_2
39          Perceptron(-1,1,1) #p_3
40
41  print('Final weights : ', weights, '\n')
42
43  if weights[1] == 0:
44      y = np.linspace(-2,2,10)
45      x = (-weights[1]*y - weights[2]) / weights[0]
46
47  else :
48          x = np.linspace(-5,5,10)
49          y = (-weights[0]*x - weights[2]) / weights[1]
50
51  plt.scatter(-1, -1, color='red')
52  plt.scatter(0, 0, color='red')
53  plt.scatter(-1, 1, color='blue')
54
55  plt.plot(x, y, '-r', label='g(x)')
56  plt.title('Final decision boundary')
57  plt.xlabel('x', color='#1C2833')
58  plt.ylabel('y', color='#1C2833')
59  plt.legend(loc='upper left')
60  plt.grid()
61  plt.show()
```