Artificial Neural Network
Home Work - 2

Trilokinath Modi

October 9, 2020

1 Problem 2

```
import numpy as np
import csv

numberOfPatterns = 16
patternDimension = 4
numberOfInputNeurons = patternDimension
numberOfOutputNeurons = 1

filePath = 'D:\\Masters Program Chalmers\\Projects and Labs\\ANN\\input_data_numeric.csv'
fileHandler = open(filePath, "r")
readFile = csv.reader(fileHandler)
storedPatterns = list(readFile)
for iRow in range(len(storedPatterns)):
    for iList in range(len(storedPatterns[iRow])):
        storedPatterns[iRow][iList] = int(storedPatterns[iRow][iList])

storedPatterns = np.delete(storedPatterns, 0, 1)

targetNeurons = [[-1, -1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, 1, -1, -1, -1],
                 [1, 1, 1, -1, -1, 1, -1, -1, 1, 1, 1, 1, 1, -1, -1, 1],
                 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1],
                 [-1, 1, 1, -1, 1, 1, -1, 1, -1, 1, -1, -1, 1, -1, -1, -1],
                 [-1, 1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, 1, -1, -1],
                 [-1, -1, -1, 1, -1, 1, -1, -1, 1, -1, -1, 1, -1, -1, -1, -1]]

errorOutput = 0
learningRate = 0.02

for iFunction in range(len(targetNeurons)):
    threshold = np.random.uniform(-1, 1, 1)
    weights = np.random.uniform(-2, 2, (numberOfInputNeurons * numberOfOutputNeurons))
    outputNeurons = np.zeros(numberOfPatterns)
    flag = 0
    iter = 0
    while flag == 0 and iter <= 100000:
        randomPattern = np.random.randint(0, len(storedPatterns), 1)
        randomPattern = randomPattern.item(0)

        outputNeurons[randomPattern] = np.tanh(0.5 * (-threshold + np.dot(weights, storedPatterns[
            ↪ randomPattern])))

        errorOutput = np.dot(weights, storedPatterns[randomPattern]) - threshold
        errorOutput = (1 - (np.tanh(errorOutput)) ** 2) # g'(B_i)
        errorOutput = errorOutput * (targetNeurons[iFunction][randomPattern] - outputNeurons[
            ↪ randomPattern])
```

```
        weights = weights + learningRate * errorOutput * storedPatterns[randomPattern]

        threshold = threshold - learningRate * errorOutput
        iter += 1

        comparison = np.sign(outputNeurons) == targetNeurons[iFunction]

        if all(comparison) is True:
            print("Function", iFunction + 1, "is converged")
            flag = 1
```