Artificial Neural Network
Home Work - 1

Trilokinath Modi

September 18, 2020

1 Problem 2

```
from typing import List
import numpy as np
from numpy.core._multiarray_umath import ndarray

numOfPatterns = 5  # Assume this as a set of patterns
numOfFedPatterns = 3
numOfBits = 160
rawStoredPattern = [
    np.array([[-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, 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]]),
    np.array([[-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, 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]]),
    np.array([[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, -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]]),
    np.array([[-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, 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]]),
    np.array([[-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, -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]])
]

rawFedPattern = [
    np.array([[-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, -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]]),
    np.array([[-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, 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]]),
    np.array([[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, -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]])
]

storedPattern = list()
fedPattern = list()
weightMatrix: List[ndarray] = list()
for iPattern in range(numOfPatterns):
    storedPattern.append(np.array(rawStoredPattern[iPattern].flatten()))

for iPattern in range(numOfFedPatterns):
    fedPattern.append(np.array(rawFedPattern[iPattern].flatten()))

for iPattern in range(numOfPatterns):
    storedPattern.append(-1 * storedPattern[iPattern])

patternId = 2
flag = 0
iIter = 0
newStateForNeuron = list(np.zeros(numOfBits))
```

```
# Weight Matrix
weightMatrix.append(np.zeros([numOfBits, numOfBits]))
for iPattern in range(numOfPatterns):
    weightMatrix += np.outer(storedPattern[iPattern], storedPattern[iPattern])
np.fill_diagonal(weightMatrix[0], 0)
weightMatrix = (1 / numOfBits) * weightMatrix

while flag == 0 and iIter < 1000000:
    if iIter % 1000 == 0:
        print(iIter)

    # Asynchronous update
    for iNeuron in range(numOfBits):
        newStateForNeuron[iNeuron] = np.dot(weightMatrix[0][iNeuron],
                                            fedPattern[patternId])
        if newStateForNeuron[iNeuron] == 0:
            newStateForNeuron[iNeuron] = 1
        newStateForNeuron[iNeuron] = np.sign(newStateForNeuron[iNeuron])
        fedPattern[patternId][iNeuron] = newStateForNeuron[iNeuron].copy()

    fedPattern[patternId] = np.array(fedPattern[patternId], int)

    for iPattern in range(2 * numOfPatterns):
        comparison = fedPattern[patternId] == storedPattern[iPattern]
        if all(comparison) is True:
            print("Converged to pattern", iPattern + 1)
            flag = 1
    iIter += 1

openTaArray = np.reshape(fedPattern[patternId], [16, 10])
print(openTaArray)
```