Semih Akbayrak
2010401168
11 March 2015

Homework1 Report

In this assignment, it is asked from us to write a code for character(I used numbers 1,5,3,4) recognition by using Hopfield algorithm. I preferred python to write my codes, because of its computational power stems from its large libraries like pynum and matplotsci.

First off all, I defined my sample patterns as string in order to design easily then I converted them to arrays by defining function "samplearray".

```
def samplearray(letter):
    return array([+1 if c=='X' else -1 for c in letter.replace('\n','')])

sample1 = samplearray(One)
sample2 = samplearray(Five)
sample3 = samplearray(Three)
sample4 = samplearray(Four)
```

Below you can find how i computed weight matrix T. This is the part where I train the system by using sample patterns which can also be considered as the equilibrium points of our hypercube.

```
for i in range(64):
    for j in range(64):
        if i==j:
            T[i,j] = 0
        else:
            T[i,j] = sample1[i]*sample1[j]+sample2[i]*sample2[j]+sample3[i]*sample3[j]+sample4[i]*sample4[j]
```

Another function I defined is "engine".It mainly takes two arguments, these are T matrix which is formed by the weights between neurons, and the other one is distorted number. Main duty of this function is determination of neurons' outputs for the next state. As

it can be seen in the code, I used another variable x, just because to make this process synchronously.

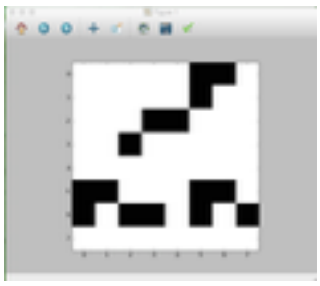
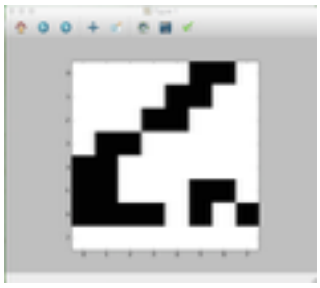
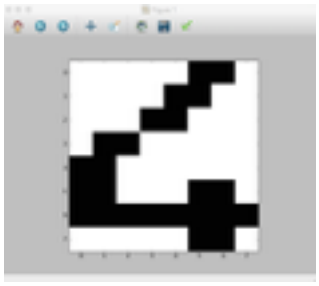
```
def engine(T, newpat):
    while 1:
        x = newpat
        from matplotlib import pyplot as plt
        plt.ion()
        for j in range(64):
            m = 0
            for i in range(64):
                m = m + T[i,j]*x[i]
            if m>=0:
                newpat[j] = 1
            else:
                newpat[j] = -1

            plt.imshow(newpat.reshape((8,8)),cmap=cm.binary, interpolation='nearest')
            pause(0.1)
        if(x.all()==newpat.all()):
            break
```

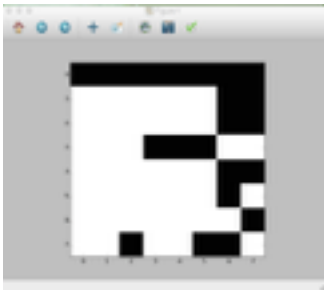
The last function I defined is “noiser”.

```
def noiser(numb,var):
    dev = math.sqrt(var)
    patnumb = samplearray(numb)
    for i in range(64):
        if patnumb[i] == 1:
            noise = np.random.normal(0,dev,1)
            patnumb[i] = patnumb[i] + noise[0]
            if patnumb[i] >= 0:
                patnumb[i] = 1
            else:
                patnumb[i] = -1
    return patnumb
```

Thanks to this function, I am able to produce distorted characters automatically. As it can be seen, noise generator in there is nothing but a random number generator from Gaussian distribution with 0 mean and selectable variance.

With a command ‘engine(T,noiser(Three,10))’, code starts to display a plot screen which is updated in every 0.1 sec. and the transformation of noised character to a sample pattern(equilibrium point) can be observed.




I just added noise to array elements which values with 1, as it was wanted. Although I increased variance to big numbers, I encountered one or two incorrect convergance in many trials. But when I applied noise to all array elements, in relatively small numbers like 2 or 3, system couldn’t find the correct equilibrium point. Another observation I’ve made is because of its relatively unique shape, algorithm can easily recognize the number 4, on the other hand, recognition of 3 is much more hard because of the overlapped shapes of 3 and 5.