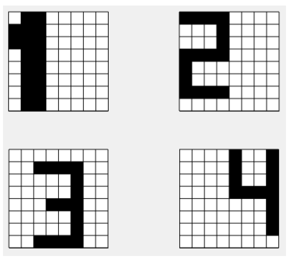
Onur Poyraz
2010401036
11 March 2015

Homework 1

In this project, I deal with the Hopfield Neural Network. It is asked from us that we should write four numbers with 8*8 blocks via Hopfield Neural Network Algorithm. I choose one, two, three and four as a numbers. I used Matlab to implement the algorithm and to take an graphical output.

In the beginning of the project, I define my numbers (which are also my stable equilibrium vectors) just writing them as a 1*64 vectors. After doing that I construct my Tij matrix by using the algorithm that is given in lectures and the sampler patterns that I defined in the beginning of the code.


The sampler patterns that I defined.

```
T=zeros(64);
for i=1:64
    for j=1:64
        if i==j
            T(i,j)=0;
        else
            T(i,j)=one(i)*one(j)+two(i)*two(j)+three(i)*three(j)+four(i)*four(j);
        end
    end
end
```

After the construction of Tij, I choose that numbers that I defined as a input vectors respectively. After doing that I put the noise on these numbers by using 0 mean and 2 variance Gaussian distribution function. I put the noise only on the bits which are equals to '1'. I do not put the noise on the bits which are equals to '-1'.

```
x = three;                    //the part that I decide the reference input
for i=1:64
    if x(i)==1
        R = normrnd(0,2);       //the part that I decide the variance and mean of the noise
        x(i) = x(i) + R;
        if x(i)>=0
            x(i)=1;
        else
            x(i)=-1;
        end
    end
end
```

In this point I have already a weight matrix and the input vector. So, the next step is to find the outputs of the neurons until the stable equilibrium point. If the input does not have too much noise on itself my stable equilibrium point will be my sampler pattern which I choose as a reference vector. However, after the increasing noise level my structure can be damaged and the output can be irrelevant to my reference vector.

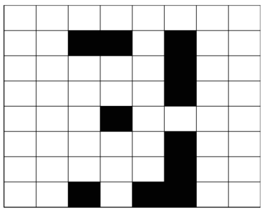
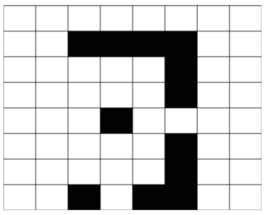
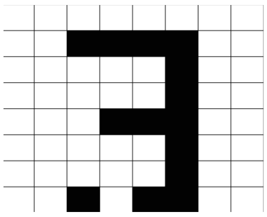
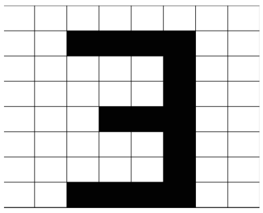
```
xplus=x;
while (1)
    for j=1:64
        sum=0;
        for i=1:64
            sum = sum + T(i,j)*x(i);
        end

        if sum>=0
            xplus(j)=1;
        else
            xplus(j)=-1;
        end
    end
    if (all(x==xplus))
        break;
    end
    x=xplus;

    k=k+1;
    figure(k);
    plothopfied(x);
end
```

From beginning to this point I simply tell about the algorithm that I wrote. This part is my algorithm and calculation part of my code. The next step is the plotting this vectors as a 8*8 blocks. To do that I create two function which I called *plotstablestate* and *plothopfield.* The first one print out my sampler pattern by using the second one. On the other hand the second one plot the 8*8 matrix from distorted signal to correct output of the neurons.

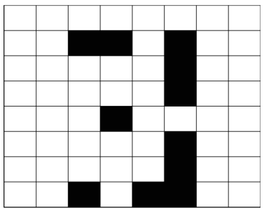
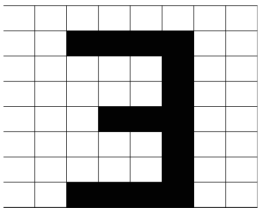
```
function plotstablestate(X)
samplerpatterns = size(X,1);
rows = sqrt(samplerpatterns);
cols = samplerpatterns/rows;
for i=1:samplerpatterns
   subplot(rows, cols, i);
   axis equal;
   plothopfied(X(i,:))
end

function plothopfied(x)
neurons = length(x);
rows = sqrt(neurons);
cols = neurons/rows;
for m=1:rows
   for n=1:cols
      neuronnumber = rows*(m-1)+n;
      if neuronnumber > neurons
         break;
      elseif x(neuronnumber)==1
         rectangle('Position', [n-1 rows-m 1 1], 'FaceColor', 'k');
      elseif x(neuronnumber)==-1
         rectangle('Position', [n-1 rows-m 1 1], 'FaceColor', 'w');
```

```
        end
    end
end
set(gca, 'XLim', [0 cols], 'XTick', []);
set(gca, 'YLim', [0 rows], 'YTick', []);
```

This is the whole of my code. Now I show some of my outputs as a figure.

This graphs show that the Hopfield Neural Network structure can give correct output although there is corruption on the signal. Each neuron gives correct output after they take distorted signal as a input.
The graph that are shown above is display the every neurons output respectively and update the graph respectively.

This two graph show the u(t) and u(t+1) vectors. The first graph is distorted signal and the second graph shows the output which are taken after one time interval. In my tests, the code rarely use the second time interval(usually it was converges wrong values) and usually the first time interval I took the output as shown in the figure.