

## CS5486 Intelligent Systems Assignment 2

Name: WANG Yue  
Student ID: 56359462  
EID: ywang2554

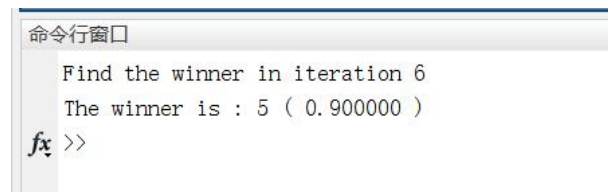
### QUESTION ONE:

**【1】** Simulate a 5-neuron MAXNET with lateral connection weight of -0.15, and external input vector of (0.1, 0.3, 0.5, 0.7, 0.9).

#### ◆ Code

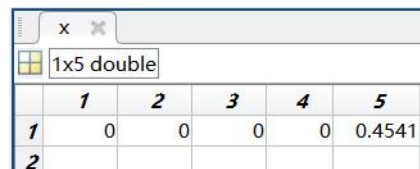
```
clear; clc;
close all
input_vector = [0.1, 0.3, 0.5, 0.7, 0.9];
x=input_vector;
w_ilc = -0.15; % weight of inhibitory lateral connections
[m,n] = size(input_vector); % n is number of nodes
% build weight matrix
%(the weight of self excitatory connections is 1)->eye(n)
%(the weight of inhibitory lateral connections is -w)->ones(n,n)*w_ilc - eye(n)*w_ilc
w = eye(n) + ones(n,n)*w_ilc - eye(n)*w_ilc;
t=0;
flag=0;
while (flag==0)
    t=t+1;
    for i=1:n
        u(i) = x * w(i,:);
        v(i) = max (0, u(i)); %activation function
    end
    x=v;
    count = sum(v(:)== 0); % the number of zeros in v
    if count==n-1
        winner = find(v~=0);
        flag = 1;
        break;
    end
end
v = input_vector(winner);
fprintf('Find the winner in iteration %g \n',t);
fprintf('The winner is : %g ( %f ) \n',winner,v);
```

## ◆ Result



```
命令行窗口
Find the winner in iteration 6
The winner is : 5 ( 0.900000 )
fx >>
```

Figure 1.1 – The screenshot of the result in command line window



	1	2	3	4	5
1	0	0	0	0	0.4541
2					

Figure 1.2 – activation in the last iteration

## ◆ Conclusion

From figure1.2, we can see that node5 is the only non-zero activation in the last iteration, so it is the winner.

## 【2】 Do the same for the kWTA network with k = 1.

## ◆ Code

```
clear; clc;
close all
input_vector = [0.1, 0.3, 0.5, 0.7, 0.9];
% u - input
% x - output
% y(t) - state variable
% beta - step size (the step size  $\beta$  is no bigger than the minimum difference of inputs)
u = input_vector;
[m,n] = size(u); % n is number of nodes
beta = 0.05; % beta=0.01/0.02/0.05/0.08
k = 1;
t = 1;
y(t) = 0;
%initialize x -----
for i= 1:n
    if u(i)-y(t) >= 0 %activation function
        x(i) = 1;
    else
        x(i) = 0;
    end
end
end
%-----
```

```

flag=0;
while (flag==0)
    y(t+1) = y(t) + beta * (sum(x) - k);
    for i= 1:n
        if u(i)-y(t+1) >= 0 %activation function
            x(i) = 1;
        else
            x(i) = 0;
        end
    end
    count = sum(x(:)== 0); % the number of zeros in x
    if count==n-k
        winner=find(x~=0);
        flag=1;
        break;
    end
    t = t+1;
end
w = input_vector(winner);
fprintf('Find the winner in iteration %g \n',t);
fprintf('The winner is : %g ( %f ) \n',winner,w);

```

### ◆ Result

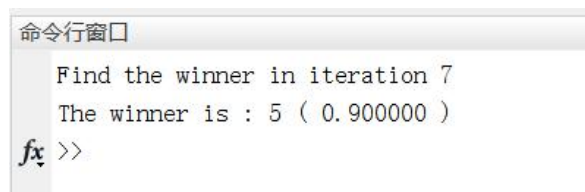


Figure 1.3 – The screenshot of the result in command line window

	1	2	3	4	5
1	0	0	0	0	1
2					

Figure 1.4 – output vector

### ◆ Conclusion

From figure1.4, we can see that node5 is the only non-zero activation in the last iteration, so it is the winner.

### ◆ The influence of different step size on the number of iterations

In this experiment, the step size (beta) is set to different value --> 0.1, 0.2, 0.5, 0.8.

```

命令行窗口
Find the winner in iteration 39
The winner is : 5 ( 0.900000 )
fx >>

```

Figure 1.5 – when step size is 0.01

```

命令行窗口
Find the winner in iteration 18
The winner is : 5 ( 0.900000 )
fx >>

```

Figure 1.6 – when step size is 0.02

```

命令行窗口
Find the winner in iteration 7
The winner is : 5 ( 0.900000 )
fx >>

```

Figure 1.7 – when step size is 0.05

```

命令行窗口
Find the winner in iteration 4
The winner is : 5 ( 0.900000 )
fx >>

```

Figure 1.8 – when step size is 0.08

From figure 1.5 to figure 1.8, we can see that the number of iteration is affected by the value of step size. Within the bounds of tolerance, the larger step size is, the less iteration is.

## QUESTION ONE:

### ◆ Code

```

clear; clc;
close all
% Original patterns to be stored
x1 = [-1;+1;+1; -1;-1;-1; -1;+1;+1];
x2 = [-1;-1;-1; -1;+1;-1; -1;+1;-1];
% caculate W-----
W1=x1*x1'-eye(9);
W2=x2*x2'-eye(9);
W=W1+W2;
% Noisy patterns to be used for retrieval // probe / key / cue
input1 = [-1;+1;-1; -1;-1;-1; -1;+1;+1];
input2 = [-1;-1;-1; -1;+1;-1; -1;+1;+1];
% Desired output / corresponding prototype patterns
output1 = W*input1;
output2 = W*input2;
% sgn -----
for i = 1:9
    if output1(i)>=0
        output1(i) = 1;
    else
        output1(i) = -1;
    end
end

```

```

end
for i = 1:9
    if output2(i)>=0
        output2(i) = 1;
    else
        output2(i) = -1;
    end
end

% plot -----
%Original patterns figures
fig11=reshape(x1, 3,3);
figure(1),subplot(1,2,1),imshow(255*uint8(fig11))
title('Original patterns 1');
fig12=reshape(x2, 3,3);
figure(1),subplot(1,2,2),imshow(255*uint8(fig12));
title('Original patterns 2');
% Noisy patterns to be used for retrieval
fig21=reshape(input1, 3,3);
figure(2),subplot(1,2,1),imshow(255*uint8(fig21));
title('Noisy patterns 1');
fig22=reshape(input2, 3,3);
figure(2),subplot(1,2,2),imshow(255*uint8(fig22));
title('Noisy patterns 2');
% Corresponding prototype patterns
fig31=reshape(output1, 3,3);
figure(3),subplot(1,2,1),imshow(255*uint8(fig31));
title('Corresponding prototype patterns 1');
fig32=reshape(output2, 3,3);
figure(3),subplot(1,2,2),imshow(255*uint8(fig32));
title('Corresponding prototype patterns 2');

```

## ◆ Result



Figure 2.1 – Two original patterns



Figure 2.2 – Two noisy patterns to be used for retrieval

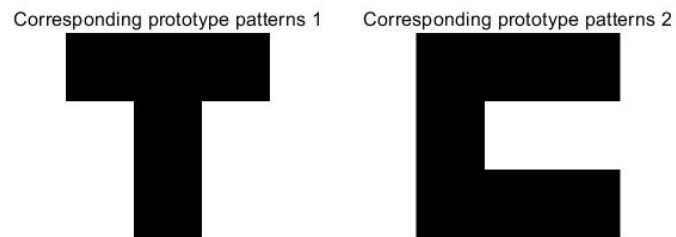


Figure 2.3 – Result of retrieval

### ◆ Conclusion

From figure2.2 and figure2.3, we can see that the two original patterns could be retrieved from the two noisy patterns.