EE 550 Homework 1 Report

Taha Küçükkatırcı - 2013400213

In this assignment, the task was to implement the binary Hopfield model. We were supposed to choose 4 numbers or letters as exampler patterns and train the model with these patterns accordingly. I chose A, X, I and 7 as exampler patterns.

I used Matlab to implement the model because of its efficiency in matrix and vector calculations. It also makes easier to visualize patterns.

Pattern generation

`The chosen numerals are A,X,I and 7`

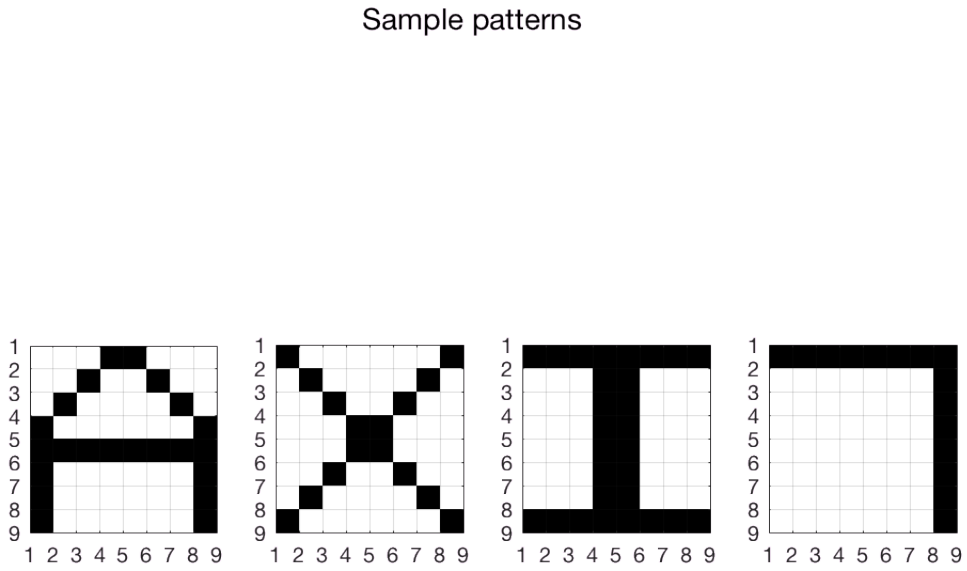
`Patterns are represented in 8x8 matrix`

```
pattern_A = -ones(8,8);
pattern_A(5,1:end) = 1;
pattern_A(5:end,1) = 1;
pattern_A(5:end,end) = 1;

left=4;
right=5;
for i=1:1:4
    pattern_A(i,right)=1;
    pattern_A(i,left)=1;
    left = left-1;
    right= right+1;
end

pattern_X = -ones(8,8);
for i=1:1:8
    pattern_X(i,i) = 1;
    pattern_X(i,9-i) = 1;
end

pattern_I = -ones(8,8);
pattern_I(1,:) = 1;
pattern_I(end,:) = 1;
pattern_I(:,4) = 1;
pattern_I(:,5) = 1;

pattern_7 = -ones(8,8);
pattern_7(1,:) = 1;
pattern_7(:,end) = 1;
```

Plot of sampler patterns

```
figure();
plotting(pattern_A, 1,4,1);
plotting(pattern_X, 1,4,2);
plotting(pattern_I, 1,4,3);
plotting(pattern_7, 1,4,4);
```

```
suptitle('Sample patterns')
```



Vectorizing the patterns and generation of weight matrix

In this part, I am converting 8x8 pattern matrices into vector form, i.e into 64x1 vector.

Then I am generating the weight matrix in accordance with these sample vectors.

```
vector_A = pattern_A'; vector_A = vector_A(:);
vector_X = pattern_X'; vector_X = vector_X(:);
vector_I = pattern_I'; vector_I = vector_I(:);
vector_7 = pattern_7'; vector_7 = vector_7(:);

w_size = size(vector_A,1);
weight_matrix = zeros(w_size,w_size);

I = eye(w_size);
weight_matrix = weight_matrix + (vector_A*vector_A'-I);
weight_matrix = weight_matrix + (vector_X*vector_X'-I);
weight_matrix = weight_matrix + (vector_I*vector_I'-I);
weight_matrix = weight_matrix + (vector_7*vector_7'-I);
```

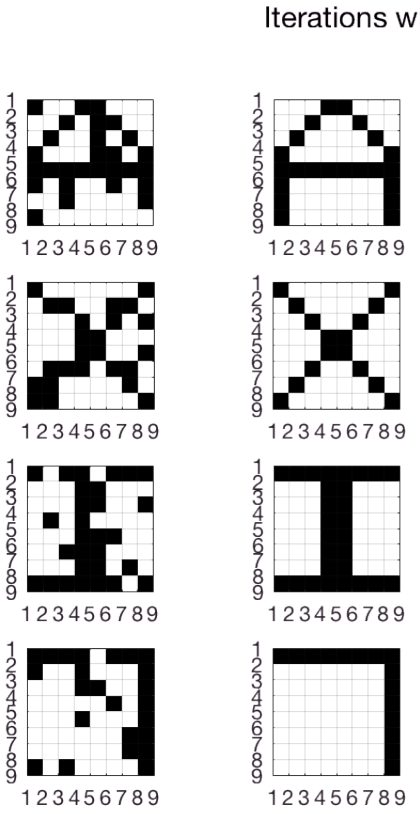
Noising samples and convergence

In this part of the code, I am adding noise to sample vectors and then run the algorithm to see the convergence.

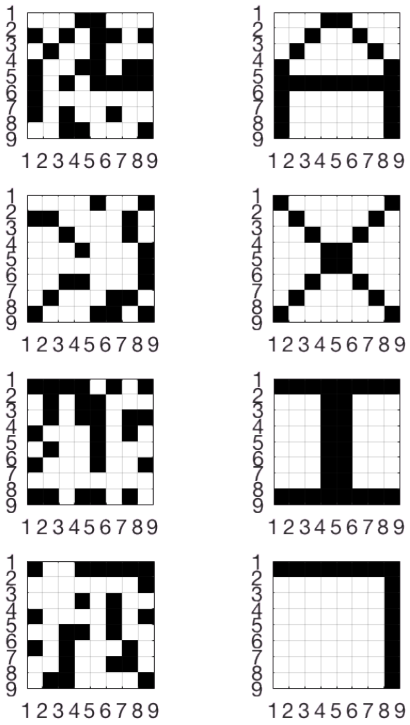
For noising, we were supposed to choose three variance values for Gaussian distribution with mean zero. Below, you may see my choices. You can change these values n the code and simulate the model again.

As one can expect, larger values of variance leads to noisier input and makes it harder to converge to the original pattern. You can see in some plots that the convergence did not happen and the simulation ended in a spurious state. This situation is of course more likely with larger variance values.

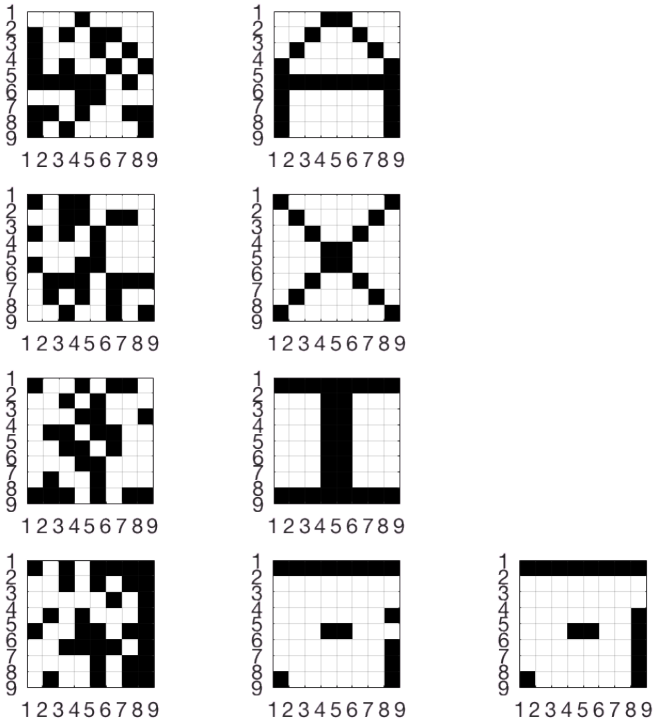
```
sigma_values = [1;2;4];
for i=1:1:size(sigma_values,1)
    var = sigma_values(i);
    figure();

    noisy_A = noise_pattern(vector_A, var); %plotting(vec2mat(noisy_A,8), 4,4,1);
    noisy_X = noise_pattern(vector_X, var); %plotting(vec2mat(noisy_X,8), 4,4,5);
    noisy_I = noise_pattern(vector_I, var); %plotting(vec2mat(noisy_I,8), 4,4,9);
    noisy_7 = noise_pattern(vector_7, var); %plotting(vec2mat(noisy_7,8), 4,4,13);

    converged_A = run(noisy_A, weight_matrix, 1); %plotting(vec2mat(converged_A,8), 4,4,4);
    converged_X = run(noisy_X, weight_matrix, 5); %plotting(vec2mat(converged_X,8), 4,4,8);
    converged_I = run(noisy_I, weight_matrix, 9); %plotting(vec2mat(converged_I,8), 4,4,12);
    converged_7 = run(noisy_7, weight_matrix, 13); %plotting(vec2mat(converged_7,8), 4,4,16);
    suptitle(strcat('Iterations with variance = ', num2str(var)));
end
```



Iterations with variance =2



Iterations with variance =4



Functions

Here are the funcitions I implemented.

- **noise_pattern:** takes an input vector and add noise to input from Gaussian distribution with zero mean and given variance.
- **plotting:** plots the values of neurons in 8x8 grid.
- **run:** simulates the Hopfield model.

```
function noisy = noise_pattern(pattern, sigma)
    noise = sqrt(sigma)*randn(size(pattern,1),1);
    temp = pattern + noise;
    greater_idx = temp >=0;
    less_idx = temp <=0;
    temp(greater_idx) = 1;
    temp(less_idx) = -1;
    noisy = temp;
end

function plotting(input_matrix, x,y,k)
    subplot(x,y,k);
    [r, c] = size(input_matrix);
    imagesc((1:c)+0.5, (1:r)+0.5, input_matrix);
    colormap([1 1 1; 0 0 0]);
    axis equal
    set(gca, 'XTick', 1:(c+1), 'YTick', 1:(r+1), ...
             'XLim', [1 c+1], 'YLim', [1 r+1], ...
             'GridLineStyle', '-', 'XGrid', 'on', 'YGrid', 'on');
end

function result = run(input, weight_matrix, type)
    w_size = size(weight_matrix,1);
    while 1
        plotting(vec2mat(input,8),4,4,type);
        indices = 1:1:w_size;
        indices = indices(randperm(length(indices)));
        x = input;
        for i=1:1:w_size
            idx = indices(i);
            temp = sum(weight_matrix(:,idx).*input);
            if temp>=0
                temp=1;
            else
                temp=-1;
            end
            input(idx)=temp;
        end
        type = type+1;
        if isequal(x,input)
            break
        end

    end
    result = input;
end
```