Table of Contents

```
% FML LAB_5: Autoassociative Neural Network using Hebbian Outer Product Rule
% Author: ARISH
% Stores and recalls 7x5 binary patterns using bipolar encoding

clear; clc;
```

Parameters

```
rows = 7; cols = 5;
N = rows * cols;
max_patterns = 10;
```

Define input patterns (binary: 0=black, 1=white)

Replace these with actual 7x5 binary arrays from your assignment Each pattern is a 7x5 matrix manually defined

```
% Digit 0
digit0 = [ ...
    0 1 1 1 0;
    1 0 0 0 1;
    1 0 0 0 1;
    1 0 0 0 1;
    1 0 0 0 1;
    1 0 0 0 1;
    0 1 1 1 0
];

% Digit 1
digit1 = [ ...
    0 0 1 0 0;
    0 1 1 0 0;
    0 0 1 0 0;
    0 0 1 0 0;
    0 0 1 0 0;
    0 0 1 0 0;
    0 1 1 1 0
];

% Digit 2
digit2 = [ ...
```

```
    0 1 1 1 0;
    1 0 0 0 1;
    0 0 0 0 1;
    0 0 0 1 0;
    0 0 1 0 0;
    0 1 0 0 0;
    1 1 1 1 1
];

% Digit 3
digit3 = [ ...
    0 1 1 1 0;
    1 0 0 0 1;
    0 0 0 0 1;
    0 0 1 1 0;
    0 0 0 0 1;
    1 0 0 0 1;
    0 1 1 1 0
];

% Digit 4
digit4 = [ ...
    0 0 0 0 1;
    0 0 0 1 1;
    0 0 1 0 1;
    0 1 0 0 1;
    1 1 1 1 1;
    0 0 0 0 1;
    0 0 0 0 1
];

% Digit 5
digit5 = [ ...
    1 1 1 1 1;
    1 0 0 0 0;
    1 0 0 0 0;
    1 1 1 1 0;
    0 0 0 0 1;
    0 0 0 0 1;
    1 1 1 1 0
];

% Digit 6
digit6 = [ ...
    0 1 1 1 1;
    1 0 0 0 0;
    1 0 0 0 0;
    1 1 1 1 0;
    1 0 0 0 1;
    1 0 0 0 1;
    0 1 1 1 0
];

% Digit 7
```

```
digit7 = [ ...
    1 1 1 1 1;
    0 0 0 0 1;
    0 0 0 1 0;
    0 0 0 1 0;
    0 0 1 0 0;
    0 0 1 0 0;
    0 1 0 0 0
];

% Digit 8
digit8 = [ ...
    0 1 1 1 0;
    1 0 0 0 1;
    1 0 0 0 1;
    0 1 1 1 0;
    1 0 0 0 1;
    1 0 0 0 1;
    0 1 1 1 0
];

% Digit 9
digit9 = [ ...
    0 1 1 1 0;
    1 0 0 0 1;
    1 0 0 0 1;
    0 1 1 1 1;
    0 0 0 0 1;
    0 0 0 0 1;
    1 1 1 1 0
];

patterns = cat(3, digit0, digit1, digit2, digit3, digit4, digit5, digit6,
digit7, digit8, digit9);
P = size(patterns,3);  % number of patterns
```

Convert to bipolar vectors

```
toBipolar = @(img) 2*img(:) - 1;
X = zeros(N, P);
for i = 1:P
    X(:,i) = toBipolar(patterns(:,:,i));
end
```

Train using Hebbian outer product

```
W = zeros(N,N);
for i = 1:P
    W = W + X(:,i) * X(:,i)';
end
W(1:N+1:end) = 0;  % zero diagonal
```

Test mode: recall stored patterns

```
sgn = @(v) 2*(v>=0) - 1;
fprintf('Recall results:\n');
for i = 1:P
    x_input = X(:,i);
    x_output = sgn(W * x_input);

    % Display input and output as 7x5 grids
    fprintf('\nPattern %d:\nInput:\n', i);
    disp(reshape(x_input, rows, cols));
    fprintf('Output:\n');
    disp(reshape(x_output, rows, cols));
end
```

```
Recall results:

Pattern 1:
Input:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1


Pattern 2:
Input:
    -1    -1     1    -1    -1
    -1     1     1    -1    -1
    -1    -1     1    -1    -1
    -1    -1     1    -1    -1
    -1    -1     1    -1    -1
    -1    -1     1    -1    -1
    -1     1     1     1    -1

Output:
    -1     1     1     1    -1
    -1     1     1    -1    -1
    -1    -1    -1    -1    -1
    -1    -1     1     1    -1
    -1    -1     1    -1    -1
    -1    -1     1    -1    -1
    -1     1     1     1    -1


Pattern 3:
Input:
    -1     1     1     1    -1
     1    -1    -1    -1     1
    -1    -1    -1    -1     1
    -1    -1    -1     1    -1
    -1    -1     1    -1    -1
    -1     1    -1    -1    -1
     1     1     1     1     1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
    -1    -1    -1    -1     1
    -1    -1     1     1    -1
    -1    -1    -1    -1     1
    -1    -1    -1    -1     1
    -1     1     1     1    -1


Pattern 4:
Input:
    -1     1     1     1    -1
     1    -1    -1    -1     1
    -1    -1    -1    -1     1
    -1    -1     1     1    -1
    -1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1
    -1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1


Pattern 5:
Input:
    -1    -1    -1    -1     1
    -1    -1    -1     1     1
    -1    -1     1    -1     1
    -1     1    -1    -1     1
     1     1     1     1     1
    -1    -1    -1    -1     1
    -1    -1    -1    -1     1
```

```
Output:
    -1    -1    -1    -1     1
    -1     1     1     1     1
    -1     1     1     1     1
    -1     1    -1    -1     1
     1     1     1     1     1
    -1     1     1     1     1
    -1    -1    -1    -1     1


Pattern 6:
Input:
     1     1     1     1     1
     1    -1    -1    -1    -1
     1    -1    -1    -1    -1
     1     1     1     1    -1
    -1    -1    -1    -1     1
    -1    -1    -1    -1     1
     1     1     1     1    -1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
     1     1     1     1    -1
    -1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1


Pattern 7:
Input:
    -1     1     1     1     1
     1    -1    -1    -1    -1
     1    -1    -1    -1    -1
     1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
     1     1     1     1    -1
    -1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1


Pattern 8:
Input:
     1     1     1     1     1
    -1    -1    -1    -1     1
    -1    -1    -1     1    -1
    -1    -1    -1     1    -1
    -1    -1     1    -1    -1
    -1    -1     1    -1    -1
    -1     1    -1    -1    -1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
    -1    -1    -1    -1    -1
    -1    -1    -1     1    -1
    -1    -1     1    -1    -1
    -1    -1    -1    -1    -1
    -1     1     1     1    -1


Pattern 9:
Input:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1
    -1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1


Pattern 10:
Input:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1     1
    -1    -1    -1    -1     1
    -1    -1    -1    -1     1
     1     1     1     1    -1

Output:
    -1     1     1     1    -1
     1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1     1     1     1    -1
    -1    -1    -1    -1     1
     1    -1    -1    -1     1
    -1      1      1      1     -1
```

Published with MATLAB® R2025a