## Table of Contents

```
% FML_LAB_Q7: Auto-Associative Memory for Character Recognition (A-E)
% Author: ARISH
% Stores 5 characters as 10x7 binary images, trains using Hebbian rule,
% and tests recognition under noise levels from 5% to 50%.

clear; clc;
```

# Create 5 characters as 10x7 binary images

Each character is manually defined as a 10x7 matrix (0/1) You can replace these with actual font bitmaps or use image processing

```
A = [0 0 1 1 1 0 0;
     0 1 0 0 0 1 0;
     1 0 0 0 0 0 1;
     1 0 0 0 0 0 1;
     1 1 1 1 1 1 1;
     1 0 0 0 0 0 1;
     1 0 0 0 0 0 1;
     1 0 0 0 0 0 1;
     1 0 0 0 0 0 1;
     1 0 0 0 0 0 1];

B = [1 1 1 1 1 0 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 1 1 1 1 0 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 1 1 1 1 0 0;
     0 0 0 0 0 0 0];

C = [0 0 1 1 1 1 0;
     0 1 0 0 0 0 1;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     0 1 0 0 0 0 1;
     0 0 1 1 1 1 0];

D = [1 1 1 1 0 0 0;
     1 0 0 0 1 0 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 0 0 0 0 1 0;
     1 0 0 0 1 0 0;
     1 1 1 1 0 0 0];

E = [1 1 1 1 1 1 1;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 1 1 1 1 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 0 0 0 0 0 0;
     1 1 1 1 1 1 1];

chars = cat(3, A, B, C, D, E);
labels = 'ABCDE';
```

## Convert to bipolar vectors

```
N = size(chars,1) * size(chars,2);  % 70 pixels
P = size(chars,3);                  % 5 characters
X = zeros(N, P);                    % each column is a pattern

for i = 1:P
    bin = chars(:,:,i);
    X(:,i) = 2*bin(:) - 1;   % convert 0/1 to -1/+1
end
```

## Train auto-associative memory (Hebbian)

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

## Recognition test with noise

```
noise_levels = [0.05 0.1 0.2 0.3 0.4 0.5];  % 5% to 50%
sgn = @(v) 2*(v>=0) - 1;
```

```
hamming = @(a,b) sum(a~=b);

fprintf('Recognition under noise:\n');
for nl = noise_levels
    fprintf('\nNoise level: %.0f%%\n', nl*100);
    for i = 1:P
        x_orig = X(:,i);
        x_noised = x_orig;
        flip_idx = randperm(N, round(nl*N));
        x_noised(flip_idx) = -x_noised(flip_idx);  % flip bits

        % Retrieval
        x_retrieved = sgn(W * x_noised);

        % Compare with all stored patterns
        dists = zeros(1,P);
        for j = 1:P
            dists(j) = hamming(x_retrieved, X(:,j));
        end
        [~, recog] = min(dists);
        fprintf('Input: %c → Recognized: %c (Hamming: %d)\n', ...
            labels(i), labels(recog), dists(recog));
    end
end
```

```
Recognition under noise:

Noise level: 5%
Input: A → Recognized: A (Hamming: 9)
Input: B → Recognized: B (Hamming: 6)
Input: C → Recognized: C (Hamming: 4)
Input: D → Recognized: D (Hamming: 3)
Input: E → Recognized: E (Hamming: 1)

Noise level: 10%
Input: A → Recognized: A (Hamming: 6)
Input: B → Recognized: B (Hamming: 6)
Input: C → Recognized: C (Hamming: 4)
Input: D → Recognized: D (Hamming: 3)
Input: E → Recognized: E (Hamming: 1)

Noise level: 20%
Input: A → Recognized: A (Hamming: 0)
Input: B → Recognized: B (Hamming: 6)
Input: C → Recognized: C (Hamming: 4)
Input: D → Recognized: D (Hamming: 7)
Input: E → Recognized: E (Hamming: 6)

Noise level: 30%
Input: A → Recognized: A (Hamming: 0)
Input: B → Recognized: D (Hamming: 6)
Input: C → Recognized: C (Hamming: 0)
Input: D → Recognized: D (Hamming: 3)
Input: E → Recognized: E (Hamming: 1)
```

```
Noise level: 40%
Input: A → Recognized: A (Hamming: 8)
Input: B → Recognized: D (Hamming: 9)
Input: C → Recognized: D (Hamming: 7)
Input: D → Recognized: D (Hamming: 3)
Input: E → Recognized: E (Hamming: 8)

Noise level: 50%
Input: A → Recognized: A (Hamming: 39)
Input: B → Recognized: E (Hamming: 8)
Input: C → Recognized: B (Hamming: 2)
Input: D → Recognized: B (Hamming: 8)
Input: E → Recognized: A (Hamming: 41)
```

*Published with MATLAB® R2025a*