```
nbrOfBits = 100;
nbrOfTrials = 10^5;

N = zeros(nbrOfBits, 1);
W = zeros(nbrOfBits);
pVec = [12, 20, 40, 60, 80, 100];
pError = zeros(length(pVec), 1);

w_ii_zero = false; % Toggle whether w_ii should be equal to zero or not

for p = pVec
    nbrOfPErrors = 0;
    for trial = 1:nbrOfTrials
        randomPatternsVec = randi(2, nbrOfBits, p);
        randomPatternsVec(randomPatternsVec == 2) = -1;
        feedPatternInd = randi(p, 1);
        feedPattern = randomPatternsVec(:, feedPatternInd);

        cInd = randi(nbrOfBits, 1, 1); % pick a random i
        sum_cInd = 0;
        for j=1:nbrOfBits
            if j ~= cInd || ((j == cInd) && (~w_ii_zero))
                for mu = [1:feedPatternInd-1, feedPatternInd+1:p]
                    sum_cInd = sum_cInd + randomPatternsVec(cInd, mu) *...
                        randomPatternsVec(j, mu) *...
                        randomPatternsVec(j, feedPatternInd);
                end
            end
        end
        C_i_nu = -randomPatternsVec(cInd, feedPatternInd) * 1 / nbrOfBits * sum_cInd;
        nbrOfPErrors = nbrOfPErrors + double(C_i_nu > 1);
    end
    pError(pVec == p) = nbrOfPErrors / nbrOfTrials;
end
```

```
nbrOfBits = 160;
nbrOfStoredPatterns = 5;

xVec = AssignDigitPatterns();
patterns = AssignFeedPatterns();
W = AssignWeights(nbrOfBits, nbrOfStoredPatterns, xVec);

lenPatterns = length(patterns(:,1));
s_new = patterns;
s_old = ones(lenPatterns, nbrOfBits);
for p = 1:lenPatterns
    while ~isequal(s_new(p, :), s_old(p, :))
        s_old(p,:) = s_new(p,:);
        for i = 1:nbrOfBits
            sum_i = 0;
            for j = 1:nbrOfBits
                sum_i = sum_i + W(i,j) * s_new(p,j);
            end
            if s_new(p, i) ~= 0
                s_new(p, i) = sign(sum_i);
            else
                s_new(p, i) = 1;
            end
        end
    end

    figure(p)
    subplot(1,2,1)
    imshow(-reshape(patterns(p, :), [10, 16])', 'InitialMagnification', 'fit')
    subplot(1,2,2)
    imshow(-reshape(s_new(p,:), [10, 16])', 'InitialMagnification', 'fit')
    s_str = '';
    for i = 1:16
        for j = 1:10
            ind = 10*(i-1)+j;
            s_str = [s_str, strcat(num2str(s_new(p, ind)), ', ')];
        end
        s_str = strcat(s_str(1:end-1), '], [');
    end
    s_str = strcat('[[', s_str(1:end-3), ']');
    disp(['Pattern ', num2str(p), ':'])
    disp(s_str)

    classification6 = true;
    for xInd = 1:nbrOfStoredPatterns
        if isequal(xVec(xInd, :), s_new(p, :))
            disp(['Classification: ', num2str(xInd)])
            classification6 = false;
        elseif isequal(-xVec(xInd, :), s_new(p, :))
            disp(['Classification: ', num2str(-xInd)])
            classification6 = false;
        end
    end
    if classification6
        disp('Classification: 6')
    end
end

function W = AssignWeights(nbrOfBits, nbrOfStoredPatterns, xVec)
    W = zeros(nbrOfBits);
    for mu = 1:nbrOfStoredPatterns
        W = W + xVec(mu, :)' * xVec(mu, :);
    end
```

```
    W = 1/nbrOfBits * W .*~eye(nbrOfBits);
end

function xVec = AssignDigitPatterns()
    x1 = [-1*ones(1,13), ones(1,4), -1*ones(1,5), ones(1,6), -1*ones(1,2),...
    repmat([-1, ones(1,3), -1], 1, 20), -1*ones(1,2), ones(1,6), -1*ones(1,5),...
    ones(1,4), -1*ones(1,13)];
    x2 = repmat([-1*ones(1,3), ones(1,4), -1*ones(1,3)], 1, 16);
    x3 = [repmat([ones(1,8), -1*ones(1,2)], 1, 2),...
        repmat([-1*ones(1,5), ones(1,3), -1*ones(1,2)], 1, 5),...
        repmat([ones(1,8), -1*ones(1,2)], 1, 2),...
        repmat([ones(1,3), -1*ones(1,7)], 1, 5),...
        repmat([ones(1,8), -1*ones(1,2)], 1, 2)];
    x4 = [-1*ones(1,2), ones(1,6), -1*ones(1,2), -1*ones(1,2), ones(1,7), -1,...
        repmat([-1*ones(1,6), ones(1,3), -1], 1, 5),...
        repmat([-1*ones(1,2), ones(1,6), -1*ones(1,2)], 1, 2),...
        repmat([-1*ones(1,6), ones(1,3), -1], 1, 5),...
        -1*ones(1,2), ones(1,7), -1, -1*ones(1,2), ones(1,6), -1*ones(1,2)];
    x5 = [repmat([-1, ones(1,2), -1*ones(1,4), ones(1,2), -1], 1, 7),...
        repmat([-1, ones(1,8), -1], 1, 2), repmat([-1*ones(1,7), ones(1,2), -1], 1, 7)];

    xVec = [x1; x2; x3; x4; x5];
end

function patterns = AssignFeedPatterns()
    pattern1 = [[-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,...
    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], [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, -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], [1, 1, 1, 1, 1, 1, 1, 1, -1, -1], [1, 1, 1,...
    1, 1, 1, 1, 1, -1, -1]];

    pattern2 = [[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,...
        -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], [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, -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], [1, 1, -1, 1, 1, -1,...
        -1, -1, 1, 1], [-1, 1, 1, -1, -1, -1, -1, -1, 1, -1]];

    pattern3 = [[-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, -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], [-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, -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, 1, 1, 1, 1, 1, -1, -1, 1], [1,...
        1, 1, 1, 1, 1, -1, -1, 1]];

    patterns = [pattern1; pattern2; pattern3];
end
```

```
nbrOfBits = 200;
nbrOfTrials = 100;
nbrOfUpdates = 10^5;
beta = 2;

nbrOfStoredPatterns = 40; %5;
m_1 = zeros(nbrOfTrials, 1);

for trial = 1:nbrOfTrials
    % Assign random patterns
    randomPatternsVec = randi(2, nbrOfBits, nbrOfStoredPatterns);
    randomPatternsVec(randomPatternsVec == 2) = -1;
    feedPatternInd = 1;
    feedPattern = randomPatternsVec(:, feedPatternInd);

    % Calculate the weight matrix
    W = AssignWeights(nbrOfBits, randomPatternsVec);

    % Update first pattern asynchronously <nbrOfUpdates> times
    s_new = feedPattern;
    for t = 1:nbrOfUpdates
        i_rndm = randi(nbrOfBits);
        b = W(i_rndm, :) * s_new;
        g_b = 1 / (1 + exp(-2 * beta * b));
        if rand(1) <= g_b
            s_new(i_rndm) = 1;
        else
            s_new(i_rndm) = -1;
        end
        m_1(trial) = m_1(trial) + s_new' * feedPattern;
    end

    m_1(trial) = m_1(trial) / (nbrOfBits * nbrOfUpdates);
    disp(trial)
end
mEnsembleAvg = sum(m_1) / nbrOfTrials;

function W = AssignWeights(nbrOfBits, randomPatternsVec)
    W = randomPatternsVec * randomPatternsVec' ./ nbrOfBits;
    W = W.*~eye(nbrOfBits);
end
```