Recognising digits

Erik Norlin

All equations are taken from the course book *Machine Learning With Neural Networks.*

```
x1=[ [ -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1,
x2=[ [ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1,
x3=[ [ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1,
x4=[ [ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, 1, -1],[ -1, -1, -1, -1, -
x5=[ [ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1

sInputPattern1 = [[1, -1, -1, 1, 1, 1, 1, -1, -1, 1], [-1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [-1
sInputPattern2 = [[1, -1, -1, 1, -1, 1, -1, 1, 1, -1], [1, -1, -1, 1, -1, 1, -1, 1, -1, -1], [1
sInputPattern3 = [[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1

storedPatterns = [x1;x2;x3;x4;x5];
sInputPatterns = [sInputPattern1;sInputPattern2;sInputPattern3];

nNeurons = length(storedPatterns);
tempWeightMatrix = zeros(nNeurons); % Temporary weight matrix for each pattern
weightMatrix = zeros(nNeurons); % Weight matrix based on all patterns
nPatterns = height(storedPatterns);
```

Hebb's rule to compute the weight matrix:

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} x_i^{(\mu)} x_j^{(\mu)} \quad \text{for} \quad i \neq j, \quad w_{ii} = 0, \quad \text{and} \quad \theta_i = 0. \qquad (2.26)$$

```
% Creating weighted matrix "W"
for muPattern = 1:nPatterns
    for iNeuron = 1:nNeurons
        for jNeuron = 1:nNeurons
            tempWeightMatrix(iNeuron,jNeuron) = storedPatterns(muPattern,iNeuron)...
            * storedPatterns(muPattern,jNeuron);
            if iNeuron == jNeuron % wii = 0
                tempWeightMatrix(iNeuron,jNeuron) = 0;
            end
        end
    end
    weightMatrix = weightMatrix + tempWeightMatrix;
end
weightMatrix = (1/nNeurons) * weightMatrix;

% Looping through the three questions
for iQuestion = 1:height(sInputPatterns)
    counter = 0;
    valid = false;

    % Looping until a pattern is recognised or number of iteration exceeds 1000
    while ~valid
```

```
% Inverting a copy of the pattern we're feeding
invertedInputPattern = sInputPatterns(iQuestion,:);
for iRow = 1:length(invertedInputPattern)
    if invertedInputPattern(iRow) == 1
        invertedInputPattern(iRow) = -1;
    else % invertedInputPattern(i) == -1;
        invertedInputPattern(iRow) = 1;
    end
end

% Checking if the network is stable i.e. if the feeding pattern
% is equal to any of the stored patterns
for muPattern = 1:nPatterns
    inputPattern = sInputPatterns(iQuestion,:);
    identifyPattern = storedPatterns(muPattern,:);

    if isequal(inputPattern, identifyPattern) || ...
        isequal(invertedInputPattern, identifyPattern)

        if isequal(inputPattern, identifyPattern)
            pattern = muPattern;
        elseif isequal(invertedInputPattern, identifyPattern)
            pattern = -muPattern;
        end

        % Converting the steady pattern to OpenTA format, starting
        % with creating a matrix for the steady state pattern
        steadyStatePatternArr = zeros(16,10);
        nRows = height(steadyStatePatternArr);
        nCols = width(steadyStatePatternArr);
        iCounter = 1;

        for iRow = 1:nRows
            for jCol = 1:nCols
                steadyStatePatternArr(iRow,jCol) = inputPattern(iCounter);
                iCounter = iCounter + 1;
            end
        end

        % Converting the matrix to a string
        steadyStatePatternString = "";
        commaCounter = 1;

        for iRow = 1:nRows
            rowString = join(string(steadyStatePatternArr(iRow,:)),", ");
            steadyStatePatternString = steadyStatePatternString + "["...
            + rowString + "]";
            if commaCounter < nRows
                steadyStatePatternString = steadyStatePatternString + ", ";
            end
            commaCounter = commaCounter + 1;
        end
        steadyStatePatternString = "[" + steadyStatePatternString + "]";
```

```
                % Printing final result
                fprintf("In Q%d, the network recognized pattern %d as digit %d\n", ...
                    iQuestion, pattern, muPattern-1);
                fprintf("Steady pattern for Q%d:", iQuestion);
                fprintf("%s", steadyStatePatternString);
                valid = true;
                break
            end
        end
```

Updating the network:

$$s_i(t+1) = \begin{cases} g\left(\sum_j w_{mj} s_j(t) - \theta_m\right) & \text{for } i = m, \\ s_i(t) & \text{otherwise.} \end{cases} \qquad (1.9)$$

If the ouput of the activation function is < 0, the state is -1 else; the state is +1.

$$\operatorname{sgn}(b) = \begin{cases} -1, & b < 0, \\ +1, & b \geq 0. \end{cases} \qquad (1.3)$$

$$b_i(t) = \sum_{j=1}^{N} w_{ij} s_j(t) - \theta_i, \qquad (1.4)$$

$\theta = 0$ *(as stated in 2.26)*

```
        % Updating the network using typewriter scheme if the pattern wasn't recognised
        if valid == false
            for iNeuron = 1:nNeurons
                sOutput = weightMatrix(iNeuron,:) * sInputPatterns(iQuestion,:)';
                if sOutput < 0
                    sOutput = -1;
                else % sOutput >= 0
                    sOutput = 1;
                end

                if sOutput ~= sInputPatterns(iQuestion, iNeuron)
                    sInputPatterns(iQuestion, iNeuron) = sOutput;
                end
            end
        end

        % Breaking the while loop if the no patterns are recognized
        counter = counter + 1;
        if counter > 1000
            fprintf("No patterns were recognised in Q%d within 1000 iterations.", iQuestion)
            break
        end
    end
end
```

```
In Q1, the network recognized pattern 5 as digit 4
```

```
Steady pattern for Q1:
[[-1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [-1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [-1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [
In Q2, the network recognized pattern 5 as digit 4
Steady pattern for Q2:
[[-1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [-1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [-1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [
In Q3, the network recognized pattern 1 as digit 0
Steady pattern for Q3:
[[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, 1, 1, 1, 1, 1, 1, -1, -1],
```