FFR135, Artificial Neural Networks

Home Problem 1

23 september 2019

Ella Guiladi
930509-0822
guiladi@student.chalmers.se

1 One-step error probability

1.1 One-step error probability

```
clear
clc
numberOfTrials = 1000;
numberBits = 120;
p=[12,24,48,70,100,120];
%diagElements=0; %if diagonal = 0
diagElements=1; %if diagonal is not = 0
for j=1:6
    errorOccured = 0;
    for n=1:numberOfTrials
    numberPatterns = p(j);
    patterns=GeneratePatterns(numberBits,numberPatterns);
    w=WeightMatrix(patterns,diagElements);

    chosenPattern=datasample(patterns,1,2);

    randPattern=randi([1 120],1);

    newState=sign(sum(w.*chosenPattern));
    newState=transpose(newState);

    if newState(randPattern) ~= chosenPattern(randPattern)
        errorOccured = errorOccured+1;

    end
    end
    errorProbability (j) = errorOccured/numberOfTrials
end
```

1.2 Weight Matrix

```
function weightMatrix=WeightMatrix(patterns,diagElements)

sizePattern=size(patterns);
numberBits = sizePattern(1);
weightMatrix=0;

for j=1:numberBits
    for i=1:numberBits
    weightMatrix(i,j)=sum(patterns(i,:).*patterns(j,:));
    end
end
if diagElements==0
  weightMatrix = (1/(numberBits).*(weightMatrix-eye(numberBits).*↩
      weightMatrix(1,1)));
else
    weightMatrix = 1/(numberBits).*weightMatrix;
end

end
```

1.3 Generate Patterns

```
function randPatterns=GeneratePatterns(numberBits,numberPatterns)
randPatterns=rand(numberBits,numberPatterns);
randPatterns=sign(randPatterns-0.5*ones(numberBits,numberPatterns));

end
```

2 Recognising digits

```
clear
clc
numberOfPatterns=5;
numberOfBits=160;
diagElements=0;
numberTrial=100;
hammingDistance=50;

x1=[ [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1,↩
      -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, 1, 1, 1, -1, -1, 1,↩
      1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, ↩
     1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, ↩
     -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, ↩
     -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, ↩
     1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, -1, ↩
     1, 1, 1, 1, 1, 1, -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, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, ↩
     1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, ↩
     1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, ↩
     -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ ↩
     -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, ↩
     -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, ↩
     -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -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, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, ↩
     -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, ↩
     1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1, ↩
     1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1, -1, ↩
     -1, -1, -1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1, -1],[ 1, 1,↩
      1, -1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1, ↩
     -1],[ 1, 1, 1, -1, -1, -1, -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, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, ↩
     1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, ↩
     -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, ↩
     1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, ↩
     -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, ↩
     -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, ↩
     1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 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, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, ↩
     -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, ↩
     -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, ↩
     1, 1, 1, 1, 1, 1, 1, -1],[ -1, 1, 1, 1, 1, 1, 1, 1, 1, -1],[ -1, -1, ↩
     -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, ↩
     -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, ↩
     -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, ↩
     -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1] ];

savedPatterns=[x1
    x2
    x3
    x4
    x5]'; %storing the patterns in a matrix
```

```
weightMatrix=WeightMatrix(savedPatterns,diagElements);

%feeding patterns that we want to recognize
pattern1A = [[-1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1,↩
      -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, 1, -1, -1], [-1, -1, -1, 1, -1,↩
      1, 1, 1, -1, -1], [-1, -1, -1, 1, -1, 1, 1, 1, -1, -1], [-1, -1, -1, ↩
     1, -1, 1, 1, 1, -1, -1], [-1, -1, -1, 1, -1, 1, 1, 1, -1, -1], [-1, 1,↩
      1, 1, 1, 1, 1, -1, -1], [-1, 1, 1, 1, 1, 1, 1, 1, -1, -1], [-1, 1,↩
      1, 1, -1, -1, 1, -1, -1, -1], [-1, 1, 1, 1, -1, -1, 1, -1, -1, -1], ↩
     [-1, 1, 1, 1, -1, -1, 1, -1, -1, -1], [-1, 1, 1, 1, -1, -1, 1, -1, -1,↩
      -1], [-1, -1, 1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, ↩
     -1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1, -1]]';
pattern2A = [[1, 1, -1, -1, 1, -1, 1, 1, -1, -1], [1, 1, -1, -1, 1, -1, 1, ↩
     1, -1, -1], [1, 1, -1, -1, 1, -1, 1, 1, -1, -1], [1, 1, -1, -1, 1, -1,↩
      1, 1, -1, -1], [1, 1, -1, -1, 1, -1, 1, 1, -1, -1], [1, 1, -1, -1, 1,↩
      -1, 1, 1, -1, -1], [1, 1, -1, -1, 1, -1, 1, 1, -1, -1], [1, 1, -1, 1,↩
      -1, 1, -1, 1, -1, -1], [1, 1, -1, 1, -1, 1, -1, 1, -1, -1], [1, -1, ↩
     1, -1, 1, -1, 1, 1, -1, -1], [1, -1, 1, -1, 1, -1, 1, 1, -1, -1], [1, ↩
     -1, 1, -1, 1, -1, 1, 1, -1, -1], [1, -1, 1, -1, 1, -1, 1, 1, -1, -1], ↩
     [1, -1, 1, -1, 1, -1, 1, 1, -1, -1], [1, -1, 1, -1, 1, -1, 1, 1, -1, ↩
     -1], [1, -1, 1, -1, 1, -1, 1, 1, -1, -1]]';
pattern3A = [[1, -1, -1, 1, -1, 1, -1, 1, 1, -1], [1, -1, -1, 1, -1, 1, -1,↩
      1, -1, -1], [1, -1, 1, -1, 1, -1, 1, -1, -1, -1], [1, -1, 1, -1, 1, ↩
     -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, ↩
     -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, ↩
     -1, -1, 1, -1, 1, -1, 1, 1, -1], [1, -1, -1, 1, -1, 1, -1, 1, 1, -1], ↩
     [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1, 1, -1,↩
      -1], [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1, ↩
     1, -1, -1], [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, -1, 1, -1, ↩
     1, -1, 1, -1, -1], [1, -1, -1, 1, -1, 1, -1, 1, 1, -1]]';
chosenPattern=pattern2A; %change feeding pattern here

for j=1:numberTrial
    for i=1:numberOfBits
    patternState(i)=sign(sum(weightMatrix(:,i).*chosenPattern));
    chosenPattern(i) = patternState(i)
    end

    patternState=chosenPattern';

    if patternState==x1
    disp('Feeded pattern converges to x1')
    break

    elseif patternState==x2
    disp('Feeded pattern converges to x2')
    break

    elseif patternState==-x2
    disp('Feeded pattern converges to -x2')
    break

    elseif patternState==x3
    disp('Feeded pattern converges to x3')
    break

    elseif patternState==-x3
    disp('Feeded pattern converges to -x3')
    break

    elseif patternState==x4
    disp('Feeded pattern converges to x4')
    break

    elseif patternState==-x4
    disp('Feeded pattern converges to -x4')
    break
```

```
    elseif patternState==x5
    disp('Feeded pattern converges to x5')
    break

    elseif patternState==-x5
    disp('Feeded pattern converges to -x5')
    break

    end
    if j==numberTrial
        disp('Pattern does not converge')
    end
end
```

2.1 Stochastic Hopfield network

```
clear
clc
rng(123)
numberBits=200;
numberPatterns=7; %Change to 45 here
numberUpdates=2*10^5; %T
noiseParameter=2;
diagElements=0;
numberExperiment=100;

orderParameterSum=0;
for anExperiment=1:numberExperiment

patterns=GeneratePatterns(numberBits,numberPatterns);
patternOne=patterns(:,1);
updatedPatternOne=patternOne;
weightMatrix=WeightMatrix(patterns,diagElements);

orderParameter=0;
for anUpdate=1:numberUpdates

   bit=randi([1 numberBits],1);
   meanField=sum(weightMatrix(:,bit).*updatedPatternOne);


   g=1/(1+exp(-2*noiseParameter*meanField));
   probabilityOfg=rand;

   if probabilityOfg < g
       updatedPatternOne(bit)=1;

   else
       updatedPatternOne(bit)=-1;
   end

   my=0;
   for aBit=1:numberBits
       my = my + patternOne(aBit)*updatedPatternOne(aBit);

   end

   orderParameter = orderParameter + my/numberBits;
end

orderParameterSum = orderParameterSum + orderParameter/numberUpdates;

end

orderParameterAverage=orderParameterSum/numberExperiment
```