Colton Cunov
Artificial Neural Networks, B. Mehlig

One-step error probability (2020)

Write a computer program implementing asynchronous deterministic updates for a Hopfield network.

Functions Used:

```
function vector = GeneratePattern(rows,cols)
% Generates a matrix of 1s and -1s, each with probability 1/2, of size
% rows x cols
    vector = randi([0 1],rows,cols);
    vector(vector==0) = -1;
end


function out = OneStepError(pattern,W,N,i)
% Outputs 1 if a single updated bit on input pattern matches old bit,
0 otherwise,
%    according to inputs weighted matrix W, bit length N, and index i
    sum = W(i,:)*pattern';
    if sgn(sum) ~= pattern(i)
        out = 0;
    else
        out = 1;
    end
end


function out = sgn(num)
% Outputs 1 if input >=0 and -1 if <0
    if num >= 0
        out = 1;
    else
        out = -1;
    end
end
```

Scripts Used:

```
N = 120;
probs = zeros(1,6);
c = 0;
numTrials = 10^5;
for p = [12,24,48,70,100,120]
    matches = zeros(1,10^5);
    X = GeneratePattern(p,N);
    W = (X'*X - p*eye(N))/N;
    % W = (X'*X)/N;
    for i=1:numTrials
        iRand = randi(p,1);
        test_pattern = X(iRand,:);
        iRand2 = randi(N,1);
        matches(i) = OneStepError(test_pattern,W,N,iRand2);
    end
    c = c + 1;
    probs(c) = 1 - sum(matches)/numTrials;
end
probs
```

With diagonals set to zero

```
>> Main
probs =
    0.0008   0.0082   0.0590   0.0962   0.1381   0.1607
```

With diagonals NOT set to zero

```
>> Main
probs =
  0.0006   0.0018   0.0128   0.0181   0.0195   0.0222
```

Colton Cunov
Artificial Neural Networks, B. Mehlig

Recognising digits (2020)

For each of the three experiments you are asked two questions: (A) To which pattern does your network converge? (B) Classify this pattern using the following scheme: if the pattern you obtain correspods to any of the stored patterns $x^{(\mu)}$, enter the pattern index μ. If your network retrieves an inverted stored pattern, then enter $-\mu$. If you get anything else, enter 6.

Functions Used:

```
function [new_pattern, isSame] = aSynchronousUpdate(s,W,N)
%Outputs [new_pattern, isSame] where new_pattern is an asynchronously
% updated pattern s according to matrix W and bit-length N and
isSame=1 if
% steady state is reached, 0 otherwise
    new_pattern = s;
    neuronsChecked = zeros(1,N); % 1 if neuron at index i has been
checked, 0 otherwise

    while ismember(0,neuronsChecked)
        i = randi(N);
        if neuronsChecked(i) == 0
            neuronsChecked(i) = 1;
        end
        b = W(i,:)*new_pattern';
        new_pattern(i) = sgn(b);
    end

    isSame = isequal(new_pattern,s);
end
```

```
function out = sgn(num)
%Outputs 1 if input >=0 and -1 if <0
    if num >= 0
        out = 1;
    else
        out = -1;
    end
end
```

Scripts Used:

```
X = readmatrix('X.txt'); % A matrix (csv format) file where each row
is a pattern i.e. 1st row is pattern "0", 2nd row is pattern "1", ...

% These are in csv format, typewriter (one line)
% test_pattern = readmatrix('test_pattern1.txt');
% test_pattern = readmatrix('test_pattern2.txt');
test_pattern = readmatrix('test_pattern3.txt');

sizeX = size(X);
p = sizeX(1);
N = sizeX(2);
W = (X'*X - p*eye(N))/N;

converged = 0;
cnt = 0;
while converged == 0
    [test_pattern, converged] = aSynchronousUpdate(test_pattern,W,N);
end

state = 6;
digit = NaN;
for i=1:p
    if isequal(X(i,:),test_pattern)
        formatted_pattern = reshape(test_pattern,10,16)';
        state = i;
        digit = i - 1;
        writematrix(formatted_pattern,'formatted_pattern.csv');
        break
    elseif  isequal(-1*X(i,:),test_pattern)
        formatted_pattern = reshape(test_pattern,10,16)';
        state = -i;
        digit = i - 1;
        writematrix(formatted_pattern,'formatted_pattern.csv');
        break
    end
end

disp('The pattern is classified as state:')
disp(state)
if ~isnan(digit)
    if state > 0
        disp('The pattern converged to the digit:')
    else
        disp('The pattern converged to the INVERSE of digit:')
    end
    disp(digit)
else
    disp('The pattern did not converge to any stored pattern or its
inverse')
end
```

Colton Cunov
Artificial Neural Networks, B. Mehlig

Stochastic Hopfield network (2020)

Write a computer program implementing a Hopfield network using Hebb's rule with wii=0, and asynchronous stochastic updating with $p(b)=1/1+exp(-2\beta b)$ with the noise parameter $\beta=2$. Use your computer program to answer the questions below.

Functions Used:

```
function vector = GeneratePattern(rows,cols)
% Generates a matrix of 1s and -1s, each with probability 1/2, of size
% rows x cols
    vector = randi([0 1],rows,cols);
    vector(vector==0) = -1;
end
```

```
function s = aSynchronousStochasticUpdate(s,W,N,beta)
%Outputs new_pattern after asynchronously updating input
%pattern s according to weight matrix W, bit-length N, and noise
parameter beta
    i = randi(N);

    b = W(i,:)*s';
    prob_b = 1/(1+exp(-2*b*beta));
    s(i) = sgn(prob_b);
end
```

```
function m = Calculate_m(s,x,N)
%Calculates m(t) given test pattern (s), original test pattern (x),
and
%bit-length N
    m = (1/N) * s * x';
end
```

```
function out = sgn(p_of_b)
%Outputs 1 with probability p_of_b, and -1 with probability 1-p_of_b
    if rand() <= p_of_b
        out = 1;
    else
        out = -1;
    end
end
```

Scripts Used:

```
N = 200;
p = 7;
% p = 45;
T = 2*10^5;
beta = 2;

avg_ms = zeros(1,100);

for i = 1:100
    X = GeneratePattern(p,N);
    W = (X'*X - p*eye(N))/N; % Wii = 0

    pattern_original = X(1,:);
    pattern = pattern_original;
    m = zeros(T,1);
    for t = 1:T
        pattern = aSynchronousStochasticUpdate(pattern,W,N,beta);
        m(t) = Calculate_m(pattern,pattern_original,N);
    end
    avg_ms(i) = mean(m);
end
disp('< m1(T) >')
disp(mean(avg_ms))
```