Colton Cunov
Artificial Neural Networks, B. Mehlig

Stochastic Hopfield network (2020)

Write a computer program implementing a Hopfield network using Hebb's rule with w_{ii}=0w_{ii}=0, and asynchronous stochastic updating with $p(b)=1/_{1+\exp(-2\beta b)}$ with the noise parameter β=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))
```