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:**

**Main\_1 (Hebb’s Rule, diagonals set to 0)**

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;

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

>> Main\_1

probs =

0.0008 0.0082 0.0590 0.0962 0.1381 0.1607

**Main\_2 (diagonals not set to 0)**

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

>> Main\_2

probs =

0.0006 0.0018 0.0128 0.0181 0.0195 0.0222