clc

clear all

close all

seq1 = fastaread('E:\ML\datasets\Satapathi\breastcan\NM\_005732.fasta');

seq2 = fastaread('E:\ML\datasets\Satapathi\breastnon\AF007546.fasta');

seqint1=nt2int(seq1);

seqint2=nt2int(seq2);

%seqint=seqint2(226:2526);

lseq1=length(seqint1);

lseq2=length(seqint2);

%DNA MAPPING

for i=1:lseq1

if seqint1(1,i)==1 %A

seqint11(1,i)=2;

end

if seqint1(1,i)==2 %C

seqint11(1,i)=1;

end

if seqint1(1,i)==3%G

seqint11(1,i)=3;

end

if seqint1(1,i)==4%T

seqint11(1,i)=0;

end

end

%DNA MAPPING

for i=1:lseq2

if seqint2(1,i)==1 %A

seqint21(1,i)=2;

end

if seqint2(1,i)==2 %C

seqint21(1,i)=1;

end

if seqint2(1,i)==3%G

seqint21(1,i)=3;

end

if seqint2(1,i)==4%T

seqint21(1,i)=0;

end

end

no\_of\_independent\_trials = 100;

%% Ergodic Process

for itr=1:no\_of\_independent\_trials

%% Displaying the number of independent trials

clc;

disp(['Independent Trial No: ',num2str(itr)])

%% Defining Input (Due to Large Number of inputs, this code will take a very long time to Run!! Please be patient)

no\_of\_inputs = 3380; % Total basepairs

% random signal unifodesired\_outputrmly distributed in the range [?0.5, 0.5]

input=seqint11;

%% input buffer and FLN order

%length of input buffer

N=15;

%FLN order

fln\_order =2;

% input buffer with initial condition

x\_buffer=zeros(1,N);

%length of inputs after trigonometric functional expansion

M = (2\*fln\_order+1)\*N + 1;

% FLN\_weights

fln\_weights=zeros(1,M);

%mu value

mu=0.0002;

%setting a 30 dB noise floor

noise = awgn(input,30)-input;

% FLN Begins!!!

for i=1:length(input)

% tap value generation with each input

x\_buffer=[input(i) x\_buffer(1:end-1)];

%% system output with noise (See the initial comments of this code)

q = 1.5 \* input(i) - 0.3\*input(i)^2 ;

if q>0

rho = 4;

else

rho=0.5;

end

%desired\_output(i) = 2 \* ((1/(1+exp(-rho\*q)))-0.5) + noise(i);

desired\_output=[seqint21,zeros(1,4469)];

%% Generation of Functional Expansion Block (FEB)

FEB=[];

for k =1:fln\_order

FEB=[FEB, sin(pi\*k\*x\_buffer), cos(pi\*k\*x\_buffer)];

end

% Final Contents of FEB

fln\_input= [1,x\_buffer,FEB];

%% FLN output

fln\_output= fln\_weights \* fln\_input';

%finding the error

error(i)= desired\_output(i) - fln\_output;

%FLN weight-update rule

fln\_weights=fln\_weights + 2 \* mu \* error(i) \* fln\_input;

end

err(itr,:)=error.^2;

end

%% Smoothing operation using a moving average filter of length 200 (As reported in our Paper)

disp(['Please Wait! Smoothing Operation is Going On...'])

length\_of\_smoothing\_filter = 200;

% Coefficients of Smoothing Filter

smoothing\_filter\_coeff = (1/length\_of\_smoothing\_filter)\*ones(1,length\_of\_smoothing\_filter);

for i=1:itr

err\_smooth(i,:) = filter(smoothing\_filter\_coeff,1,err(i,:));

end

%% Ploting the Learning Curve (You can verify the shape from Fig. 4(a) of our paper!)

% figure;

% plot(10\*log10(mean(err\_smooth))); xlabel('Iterations');ylabel('MSE (dB)'); grid on;

%% Average MSE Value over the last 1000 iterations (As reported in our paper)

fln\_mse=(10\*log10(mean(mean(err(end-1000:end)))));

fprintf('Average MSE Value over the last 1000 iterations is %f', fln\_mse);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Thats it! %%%%%%%%%%%%%%%%%%%%%%%%%%%%