Lab 4 report: Echo state network Lissajous curves

Dekel Viner(S2612925),Luis Knigge(S2560224)

January 2015

1 Introduction

1.1 problem analysis

The problem is to design a neural network that would be able to learn and generate two different predetermined Lissajous curves based on input.

1.2 Tackling the problem using an ESN

To tackle the problem an echo state network will be used. The output will be generated by teacher forcing the desired output into the output nodes and then using back propagation the weights between the nodes in the dynamic reservoir and the output nodes will be adjusted. Two output nodes will be used, one for x axis and one for the y axis of the Lissajous curves.

2 Methods

2.1 Data Generation

The data has been generated by running the network 11 times per setting of the parameters. The data then has been sorted to make it easier to read. The values you see in the table are the MSE values to be able to compare the different settings of parameters.

2.2 How do you address the performance of the network.

The output of the network will be compared with the training set. Only the test set is important. The network will be considered working and generally manages to produce the curves required.

2.3 Which experiments are you going to conduct and what are the parameter settings?

The data will be trained on two Lissajous curves with the sin(a) sin(b) parameters being 1,2,1,3 respectively. We test the parameters of the number of nodes, the connectivity of the input nodes to the dynamic reservoir. The connectivity of the nodes in the dynamic reservoir and the Alpha parameter. we test them by running them 11 times and storing the MSE results for each in an array and sorting the array. We then graph it. The reason we sorted the array is because we wanted to compare curves instead of an average of the mean square errors as the networks are highly unpredictable and contain a lot of extreme outliers which bias the results by pulling the average of the MSE up, this was certainly visible when we attempted to get an average of the MSE as the results we have obtained were complete varied and unpredictable.

2.4 Are you going to optimize your parameter settings? How?

- Number of nodes=100, 200, 300, 1000
- alpha=0.1, 0.3, 0.5, 0.7, 0.9
- C=0.1, 0.3, 0.5, 0.7, 0.9
- Cin=0.1, 0.3, 0.5, 0.7, 0.9

We've looked at our results and came the following parameters as an optimization, its based on both results from comparing mean square errors and from results from feel by running the network several times on each parameter level and seeing where the network tends to work best in.

3 results

Look at the results pdf for full results.

4 Discussion and Conclusion

4.1 Performance Evaluation

The network works fairly well when run on the optimal settings that we have found. It finds a solution we would consider sufficient by comparing the desired output to the actual output figures over 50 percent of the time.

4.2 problems that should be addressed

The networks is very unpredictable the mean square error can change dramatically between 2 tests. A correct solution is found most of the time but still there are many failed attempts.

4.3 Other suitable networks

We believe that a hopfield network could have also being used similar to the one in the letter recognition lab.

5 Appendix

The code is filled with comments explaining the decisions we made in the process. We both worked on the network together. Luis did most of the coding but both of us have tried to figure out how the network should function and what needs to be changed and implemented from the given controlled sine wave function. Luis then experimented and provided the data for the model and Dekel conduction the data analysis. Trying to understand the basic principles that make a ESN function was certainly the most difficult part, once we figured out the theory behind the network coding and implementing was a fairly straight forward task.

5.1 Code

LissajousCurveFinished.m

```
close all
clear all

%input
K=2;                  % Number of input units
Cin=0.3;              % Connection probability from input to internal
    layer



%Dynamic reservoir
N=200;                % Number of inetrnal nodes
C=0.25;               % Connection probability within internal layer
alpha=0.7;            % Spectral radius

%output
L=2;                  % Number of output units
Cback=0.65;            % Connection probability from output to
   internal layer
amount_noise=0.0005; %amount of noice in the driver

% Length of different phases
Transientlength=1200;           %lenght of transient time
Traininglength=2400;            %training time
Testlength=2400;                %Test time

% From here network code
% Do not change anything.
% ofcourse you could add some plot functions for added clarity

% Initialisation internal network
% here a sparce connected Dynamic reservoir is created using the
% steps on blz 30
```

```
Wpre=sprand(N,N,C);              %Sparse uniformly distributed random matrix
W0=2*(Wpre-0.5).*(Wpre>0);       %All non zero values minus 0.5 times 2
lambda_max=abs(eigs(W0,1));      %Calculate labda max
W1=W0/lambda_max;                %Normalize matrix to unit spectral radius
W=alpha*W1;                      %scale the matrix

% Initialisation input weights
Win0=sprand(N,K,Cin);            %sparce randin uniform distr random matrix
Win=2*(Win0-0.5).*(Win0>0);      %al zero minus 0.5 times 2


% Initialisation feedback weights
Wback0=sprand(N,L,Cback);        %sparce randin uniform distr random matrix
Wback=2*(Wback0-0.5).*(Wback0>0);%al zero minus 0.5 times 2


% Training set length
TotalLength=Transientlength+Traininglength+Testlength;  %total time


% create the time lines for al time blocks
TTransient=1:Transientlength;
TTraining=Transientlength+1:Transientlength+Traininglength;
TTest=Transientlength+Traininglength+1:Transientlength+Traininglength+Testlength;
TTotal=[TTransient, TTraining,TTest];


% create the signal
slowperiod=300;                  % length of the steps for a piecewise constant input signal
fastperiod=150;                  % the fast sinus of the desired output signal
a =2;                            % These values are the parametes of the Lissajous curves
b =1;                            % a and b are for the first curve and c and d are for the second curve
c =1;
d =2;
upre(1)=1;                       % Start points if the input signal

t(1)=0;
```

```
for i=2:TotalLength                    % create the input signal for the whole time lenght
    upre(i)=0.5*mod(floor(i/slowperiod),2)+0.05;
    t(i)=t(i-1)+10*upre(i);
end
u = [upre;upre];                       % we create an input that has two values because we first though we would
                                       % need two inputs and build the whole
                                       % program based on that. Later we switched to a
                                       % single input and couldn't change the
                                       % program according to that.


% create two drives for the two curves the network has to learn
drive_cleanA1=0.5*sin(a*2*pi*t([TTransient, TTraining,TTest])/fastperiod);
drive_cleanA2=0.5*sin(b*2*pi*t([TTransient, TTraining,TTest])/fastperiod);
drive_cleanA=[drive_cleanA1;drive_cleanA2];

drive_cleanB1=0.5*sin(c*2*pi*t([TTransient, TTraining,TTest])/fastperiod);
drive_cleanB2=0.5*sin(d*2*pi*t([TTransient, TTraining,TTest])/fastperiod);
drive_cleanB=[drive_cleanB1;drive_cleanB2];


% combine the two drives to form a single drive that switches between the
% inputs
drive_clean=drive_cleanA;
for i=301:600:TotalLength
     drive_clean(:,i:i+299)=drive_cleanB(:,i:i+299);
end


noise=[randn(1,Transientlength+Traininglength)*amount_noise ,0.*TTest;randn(1,Transientlength+Traininglength)*amount_noise ,0.*TTest];
drive=drive_clean+noise;               % add noise
T=drive(:,TTraining)';                 % T represents the activation of the output units during training
M=zeros(Traininglength,N);             % create matrix to store train data: network state
Mpre=zeros(Transientlength,N);  % create matrix to store transient data: network state
```

```


% Driver forcing the network with the desired output
for i=TTransient    % for the whole transient time
    if(i==1)
        x=rand(N,1);        %start network in random state
    else
        x=tanh(Win*u(:,i) + W*x + Wback*drive(:,i-1));  %use update rule and input to
        % we canged this function a little bit so that the network can handle two inputs and outputs
        % We did the same changes for the ttraining and the testing (u(i) -->
        % u(:,i))
    end                     % calculate next network activation
    Mpre(i,:)=x';           % save the network state
end

for i=TTraining             % get the training data
    x=tanh(Win*u(:,i) + W*x + Wback*drive(:,i-1));
    M(i-Transientlength,:)=x';
end



% Batch Training
% atanh = tanh^-1
% atanh(T) = Wout*M, so: M\atanh(T) = Wout
Wout = (M\atanh(T))';



y=[drive(1,[TTransient, TTraining]), zeros(1,Testlength);drive(2,[TTransient, TTraining]), zeros(1,Testlength)]; % output of network (2x6000)
MPost=zeros(Testlength,N);            %create matrix to store network actiovation in test


% test the network
for i=TTest                                                      %for the test time
    x=tanh(Win*u(:,i) +W*x+Wback*y(:,i-1));                          % calculate network state
    y(:,i)=tanh(Wout*x);                                           % calculate output
    MPost(i-Transientlength-Traininglength,:)=x';       %  save the network state
end
```

```
% Errors berekenen
YTraining=Wout*M';

% to calculate the MSE for the training and the test phase we take the
% 2x6000 matrix of the outputs shorten them to the appropriate length of
% the phase and put them into a single vector.
YTraining_1line= [YTraining(1,:),YTraining(2,:)];
drive_short1 = drive(:,TTraining);
drive1_1line= [drive_short1(1,:),drive_short1(2,:)];
difference1 = YTraining_1line - drive1_1line;

MSE_Training = 1/Traininglength*sum((difference1).^2);


YTest=Wout*MPost';

YTest_short = YTest(:,(TTest-Transientlength-Traininglength));
YTest_1line= [YTest_short(1,:),YTest_short(2,:)];
drive_short2 = drive(:,TTest);
drive2_1line= [drive_short2(1,:),drive_short2(2,:)];
difference2 = YTest_1line - drive2_1line;

MSE_Test = 1/Testlength*sum(difference2.^2);


plot_handlers = zeros(5,1);
fig_handler = figure(1);
plot_handlers(1) = subplot(5,1,1);
plot(plot_handlers(1), TTest, upre(TTest))
title('Input signal during test phase', 'Color', 'White')

% the next two plots have been changed to display the Lissajous curves
% instead of the sin waves
g=1;
xplot(Testlength)=0;
yplot(Testlength)=0;
for h=(Transientlength + Traininglength +1):300:TotalLength
    k=h-(Transientlength + Traininglength);
    plot_handlers(2) = subplot(5,1,2);
    xplot(k:k+299)=drive_clean(1,h:h+299)+g;
    yplot(k:k+299)=drive_clean(2,h:h+299);
    plot(plot_handlers(2), xplot, yplot,'-g')
    g=g+1;
    title('Goal during test phase', 'Color', 'White')
    ylim([-.6,.6])
```

```
end

g=1;
xplot(Testlength)=0;
yplot(Testlength)=0;
for h=(Transientlength + Traininglength +1):300:TotalLength
    k=h-(Transientlength + Traininglength);
    plot_handlers(3) = subplot(5,1,3);
    xplot(k:k+299)=y(1,h:h+299)+g;
    yplot(k:k+299)=y(2,h:h+299);
    plot(plot_handlers(3), xplot, yplot,'-g')
    g=g+1;
    title('Actual output during test phase', 'Color', 'White')
    ylim([-.6,.6])
end

drive_short=drive(:,TTest);
y_short=y(:,TTest);

plot_handlers(4) = subplot(5,1,4);
plot(plot_handlers(4), TTest, (drive_short(1,:)-y_short(1,:)).^2, '-r')
title(['Squared Error for Output Horizontal (MSE = ' num2str(MSE_Test) ')'], 'Color', 'White')

% we added another plot to show the errors in the horizontal as well as the
% vertical output
plot_handlers(5) = subplot(5,1,5);
plot(plot_handlers(5), TTest, (drive_short(2,:)-y_short(2,:)).^2, '-r')
title(['Squared Error for Output Vertical (MSE = ' num2str(MSE_Test) ')'], 'Color', 'White')

set(plot_handlers,'Color','Black')
set(plot_handlers, 'XColor', 'White', 'YColor', 'White')

fig_pos = [1 1 27 27];

set(fig_handler, 'Color', 'Black')
set(fig_handler, 'Name', 'Testresultaten')
set(fig_handler, 'Units', 'centimeters');
set(fig_handler, 'Position', fig_pos);
set(fig_handler, 'PaperPositionMode', 'auto');
```

5.2 Data