## Table of Contents



# Project - 2

```
clear all
close all
clc
```

# Editable Values

```
windowSize=10; % Number of data points in the input window
predictionSize=1; % Number of data points in the  output window
testSize=30; % Number of data points withheld for testing
finalTestSize=30; % Size of final test Vector
Train.Algorithm='trainbr'; % Algorithm used for training {lm,br}
Train.Lrate=0.0001; % Learning rate used in training
Train.Hneurons=[10 5]; % Number of neurons used in net
Train.Niterations=1000; % Number of total epochs to run (set low for
 early stopping)
```

# Setup

```
load('data.mat')
totalSize=windowSize+predictionSize;
vecLen=length(vector);
shiftSize=predictionSize-1;
totalShift=totalSize-1;
Best.err=50000;
Best.net=fitnet(Train.Hneurons,Train.Algorithm);
Best.seed=0;
```

# Organize Data

```
trainStart=windowSize+1;
trainEnd=vecLen;
testStart= vecLen-testSize+1;
testEnd= vecLen;
```

```
for i=trainStart+shiftSize:trainEnd
    trainData(:,i-windowSize-shiftSize)=vector(i-totalShift:i);
end

testData=vector(testStart:testEnd)';
```

# Train all

```
u=25;
rng(u);
m=1;

% prep training input
order = randperm(length(trainData));
randtrainData = trainData(:,order);

% setup net
net{m} = fitnet(Train.Hneurons,Train.Algorithm);
net{m} =
 configure(net{m},randtrainData(1:windowSize,:),randtrainData(windowSize
+1:end,:));
net{m}.trainParam.mu = 0.005;
net{m}.trainParam.epochs=Train.Niterations;% Number of Iterations
net{m}.divideParam.trainRatio = 0.95;
net{m}.divideParam.valRatio = 0.05;
net{m}.divideParam.testRatio = 0.0;
net{m}.trainParam.max_fail=50;

% Training algorithm
net{m} =
 train(net{m},randtrainData(1:windowSize,:),randtrainData(windowSize
+1:end,:));
```

# Testing

```
train_result = net{m}(trainData(1:windowSize,:));
test_vec=trainData(trainEnd-windowSize+1:trainEnd);

for i=1:predictionSize:testSize
    test_result(1,(i:i-1+predictionSize)) = net{m}(test_vec(i:i
+windowSize-1)');
    test_vec(windowSize+i:windowSize+i-1+predictionSize) =
 test_result(1,(i:i-1+predictionSize));
end
```

# MSE Training Calculation

```
errTest = immse(testData,test_result);
```

# Save Best

```
if errTest<Best.err
```

```
    Best.err=errTest;
    Best.net=net{m};
    Best.seed=u
end
```

# Test Best

```
train_result = Best.net(trainData(1:windowSize,:));

test_vec=trainData(trainEnd-windowSize+1:trainEnd);

for i=1:predictionSize:testSize+29
    test_result(1,(i:i-1+predictionSize)) = Best.net(test_vec(i:i
+windowSize-1)');
    test_vec(windowSize+i:windowSize+i-1+predictionSize) =
 test_result(1,(i:i-1+predictionSize));
end
```

# Plot Best

```
fig1=figure(1);
fig1.Renderer='Painters';
set(fig1,'units','points','position',[200,450,1200,300]);
hold on;grid on;
title('Neural Net Output vs Given Data');
xlabel('Time [Year]');
ylabel('Magnitude');
xlim([0 310]);
plot(vector,'k','lineWidth',2)
for i=0:shiftSize
    plot((trainStart+i:trainEnd-shiftSize+i),train_result(i
+1,:)','r','lineWidth',1.2);
end
plot((testStart:testEnd+29),test_result,'b','lineWidth',2)
legend('Time Series Data','Neural Net Output on Training Data','Neural
 Net Prediction','Location','northwest')
print('-painters','-depsc','figure2')
print('-painters','-dpdf','figure2')
```

# Final Test Vector

```
Best.final_vector=test_result(end-30+1:end)
```

*Published with MATLAB® R2017b*