Table of Contents


```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 24-Feb-2018 12:14:15
%
% This script assumes these variables are defined:
%
%   A_human - input data.
%   A_baxter - target data.
```

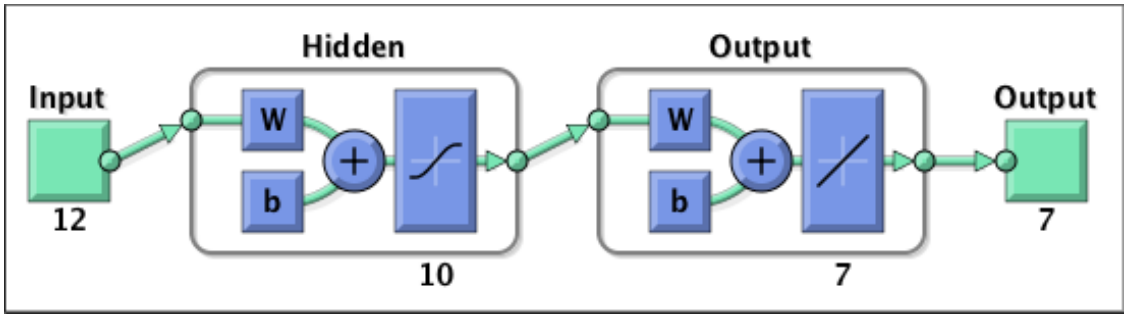
SETUP INPUT FOR NEURAL NETWORK TRAINING

X is the dataset of human arm coordinate that we use as input on Neural Network for training it t is the output of N.N., the dedired BAxter's arm position taken online with sync_node.cpp

```
x = [human_training2,human_test_dataset];
t = [baxter_training2,baxter_test_dataset];
%human_test_dataset =[human7',human8',human9'];
%baxter_test_dataset =[baxter7',baxter8',baxter9'];


% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 10;  % number of neurons used
net = fitnet(hiddenLayerSize,trainFcn);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivision
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
```

```
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean Squared Error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression', 'plotfit'};

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
testTargets = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)
```

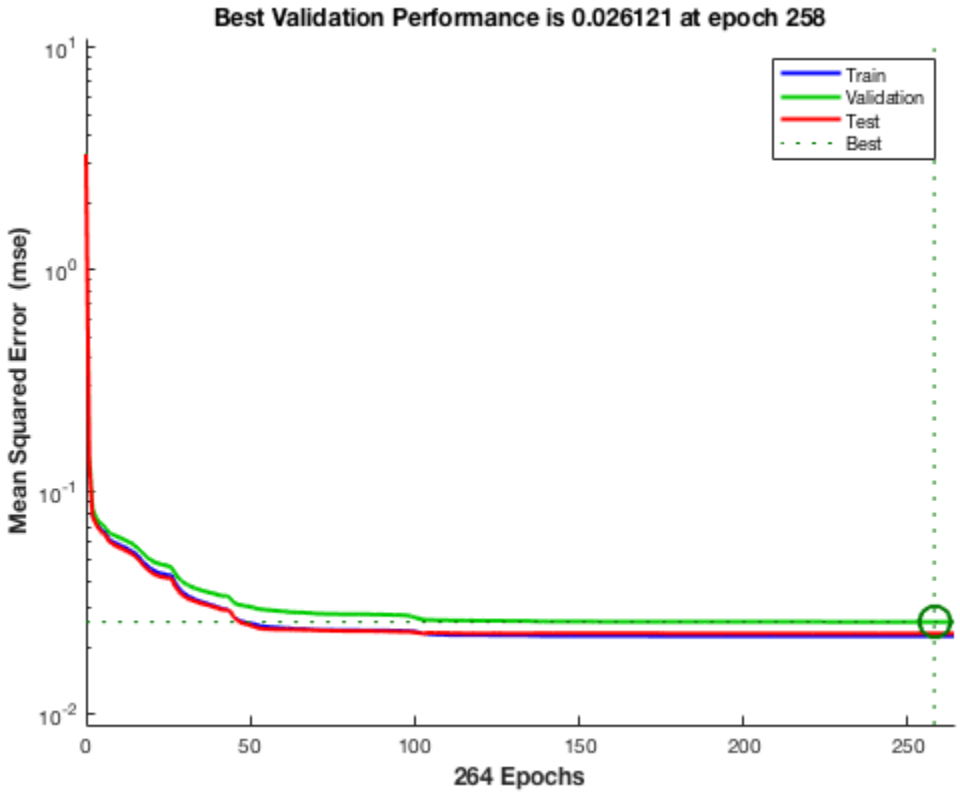
```
performance =

    0.0232


trainPerformance =

    0.0226


valPerformance =

    0.0261


testPerformance =

    0.0232
```

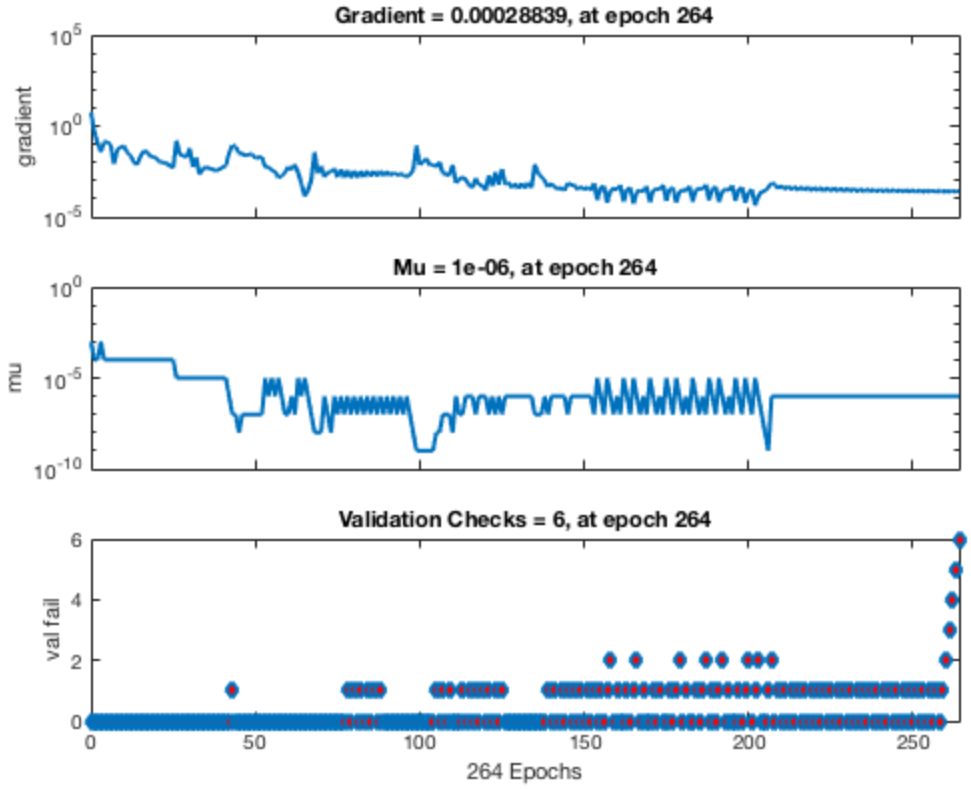


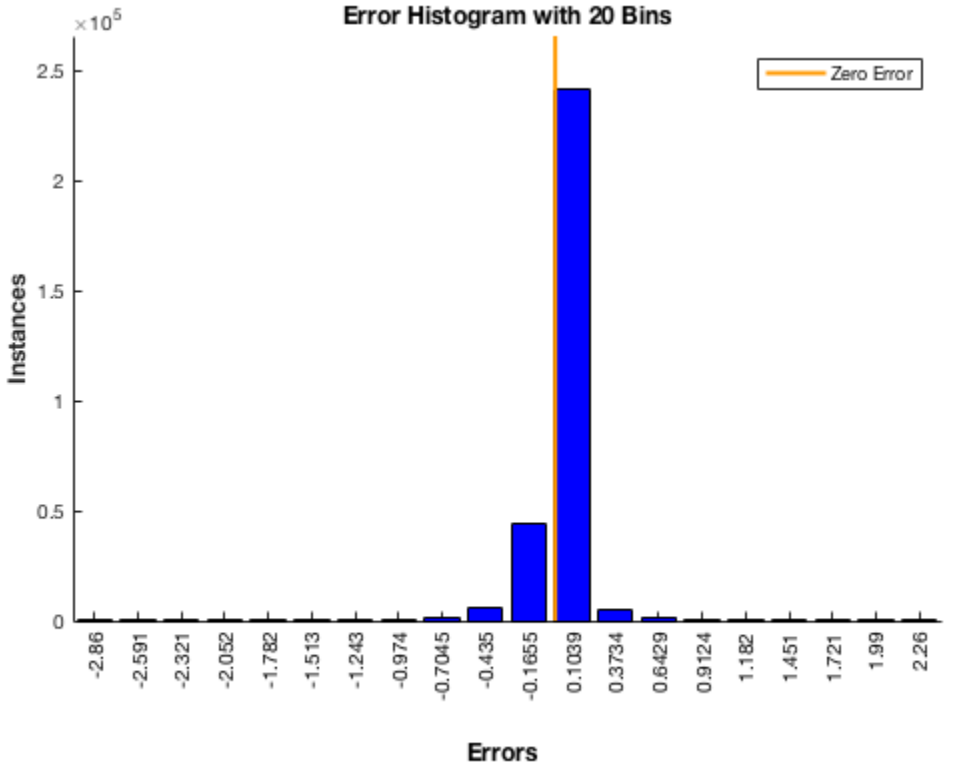
Error checking plots

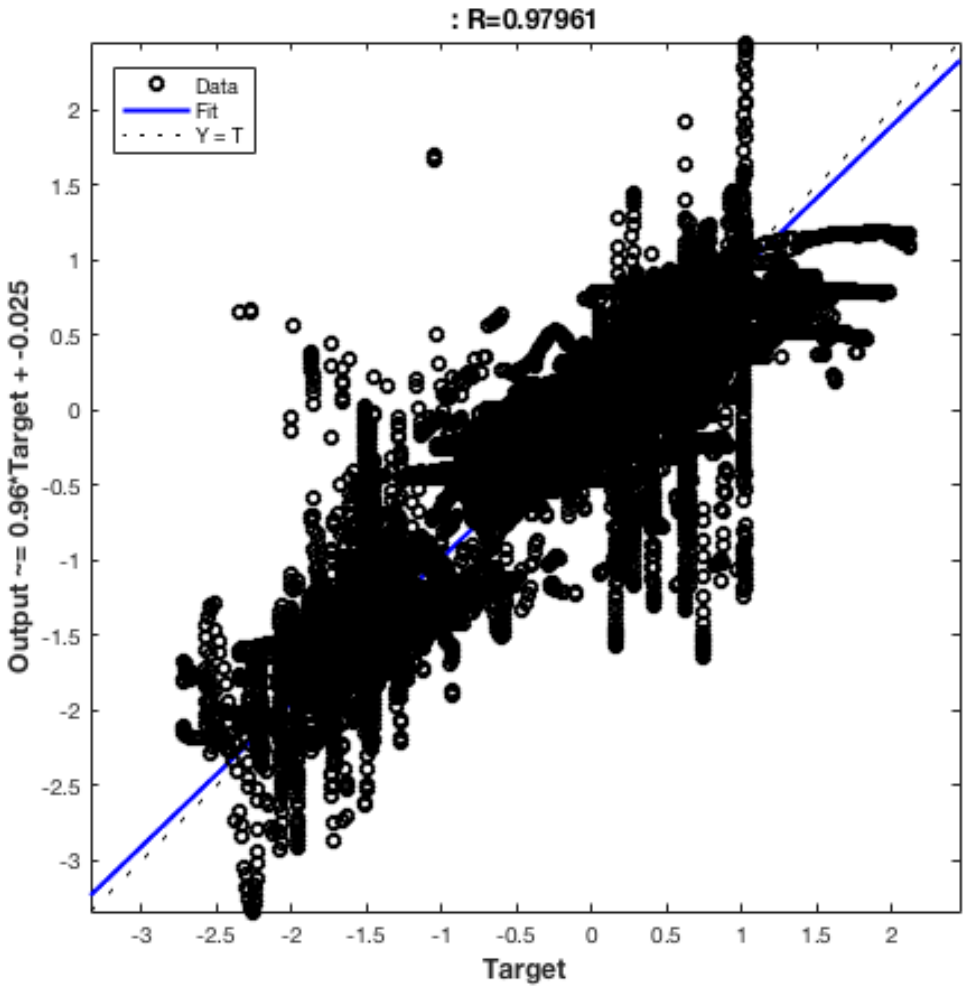
Uncomment these lines to enable various plots.

```
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, ploterrhist(e)
figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```




Gradient = 0.00028839, at epoch 264
gradient
Mu = 1e-06, at epoch 264
mu
Validation Checks = 6, at epoch 264
val fail
264 Epochs


Error Histogram with 20 Bins
Zero Error
Instances
Errors



Neuaral Network generation

```
if (false)
    % Generate MATLAB function for neural network for application
    % deployment in MATLAB scripts or with MATLAB Compiler and Builder
    % tools, or simply to examine the calculations your trained neural
    % network performs.
    genFunction(net,'myNeuralNetworkFunction');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a matrix-only MATLAB function for neural network code
    % generation with MATLAB Coder tools.
    genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
    y = myNeuralNetworkFunction(x);
end
if (false)
```

```
    % Generate a Simulink diagram for simulation or deployment with.
    % Simulink Coder tools.
    gensim(net);
end
```

Published with MATLAB® R2017b