COMP3003 MACHINE LEARNING
LAB PRACTICAL P7: NEURAL NETWORKS - 2

Develop a neural network model to estimate bodyfat.

This task is to develop a neural network model to estimate a person's bodyfat. The dataset (bodyfat_dataset) is a sample dataset provided by Matlab in the neural network toolbox (nnet). The dataset consists of 252 samples (i.e. from physical measurements of 252 people assuming no repetitive samples). Each sample contains 13 physical features (e.g. age, weight, height, neck circumference, chesk circumference) and one target output of bodyfat. A neural network model can be developed to learn linear or non-linear relationships between 13 physical features and bodyfat from a training data set. The trained model can be used to predict bodyfat for new measurements with 13 physical features. More details regarding the dataset can be found at the following link. https://uk.mathworks.com/help/deeplearning/ug/body-fat-estimation.html

If you have installed matlab in your own machine, you can locate the file (bodyfat_dataset.mat) at C:\Program Files\MATLAB\R2020a\toolbox\nnet\nndemos\nndatasets (the location may vary depends on your matlab installation).

- Load the dataset in matlab's command window 'load bodyfat_dataset.mat' or just 'load bodyfat_dataset', and observe the dataset in Matlab's WORKSPACE window (see Figure 1). After load the dataset, it produces two variables, bodyfatInputs (13x252) matrix for 13 input features (row) and 252 samples (column), and bodyfatTargets (1x252) matrix for 1 output (row) and 252 samples (column).

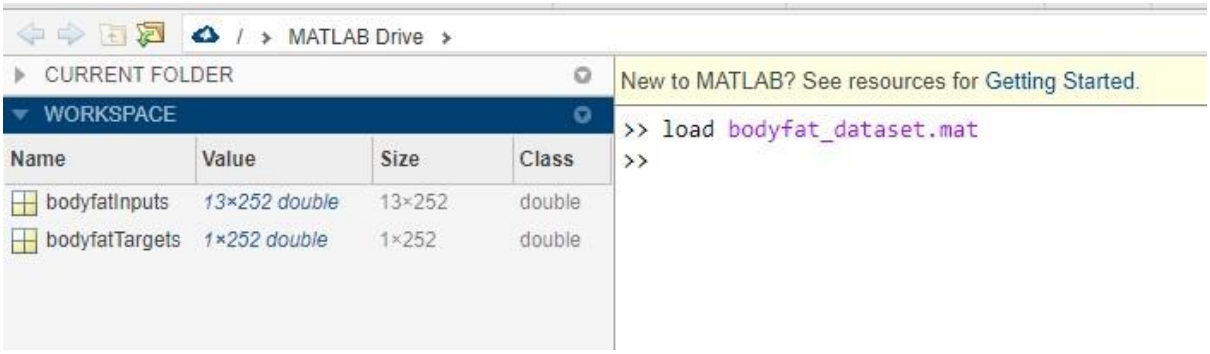

Figure 1. Load the dataset

- You can type *bodyfatInputs* or *bodyfatTargets* in command window to view the contents of these two variables. You can also double click bodyfatInputs (or bodyfatTargets) in the WORKSPACE window to view its contents.

- After understanding of the dataset, you can write your matlab code to develop neural network models in the following steps.

  (1) Load dataset
  load bodyfat_dataset
  inputData = bodyfatInputs;
  targetData = bodyfatTargets;

  (2) Train the neural network using the dataset

net = feedforward(12); %select feedforward network with 12 hidden neurons
[net, tr] = train(net, inputData, targetData); %train the neural network using the dataset
plotperform(tr); %plot performance graph

After running the above code, you should see the matlab nntraintool and performance plot, similar as Fig. 2 and Fig. 3. What is the stopping criterion used by the Matlab train function?

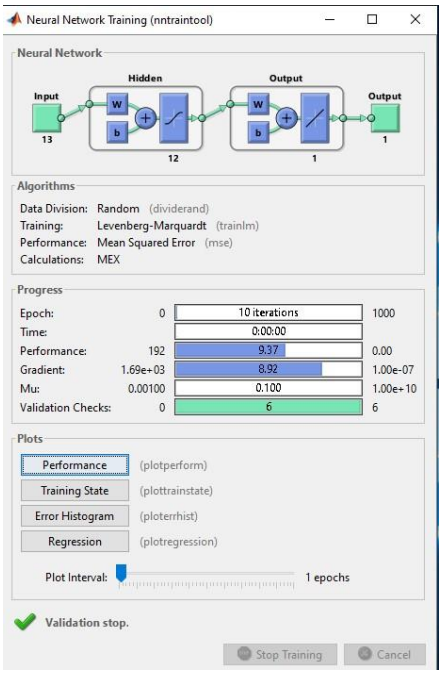

Fig 2. Matlab nntraintool

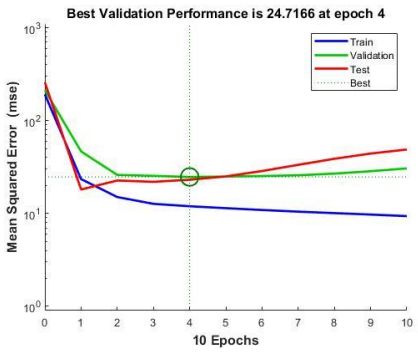

Fig 3. Performance plot

(3). When you call "train" function in matlab, it uses the *dividerand* function to divide randomly the dataset to training set (70%), validation set (15%) and testing set (15%).
After training, it also returns the indices of the dataset used for training (tr.trainInd), validation (tr.valInd) and testing (tr.testInd)

You can evaluate performance for the testing dataset as below.
% Testing the neural network for the testing dataset
testX = inputData(:, tr.testInd); %testing data inputs
testT = targetData(:, tr.testInd); %testing data Target output

testY = net(testX); % predicted testing output from the trained net.
perf = mse(net, testT, testY); %calcualte MSE between testing Target data (testT) and testing Predicted data (testY).

Similarly, you can evaluate performance for training and validation dataset.

(4) Testing the performance of neural networks by calculating correlation coefficients (R-value) between the predicted and target output, and draw a regression plot (or scatter plot between the target and predicted output).
corrcoef(testT, testY); %calculate correlation coefficients for the testing dataset
scatter(testT, testY); % draw a scatter plot between the target and predicted output)

Similarly, you can calculate correlation coefficients for training and validation datasets.

plotregression(testT, testY); %this will use plotregression function to draw a regression plot with R value displayed in the title of the graph for the testing dataset.

Y = net(inputData);
plotregression(targetData, Y); % this will draw a regression plot for all samples (similar as in Figure 4).

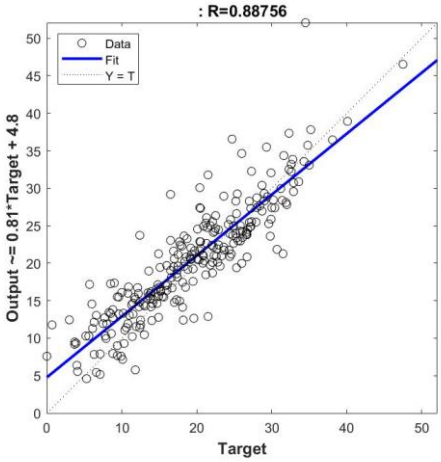

Figure 4. Regression plot for all samples

You can also see the regression plots from nntraintool (Plots => Regression), similar as in Figure 5.

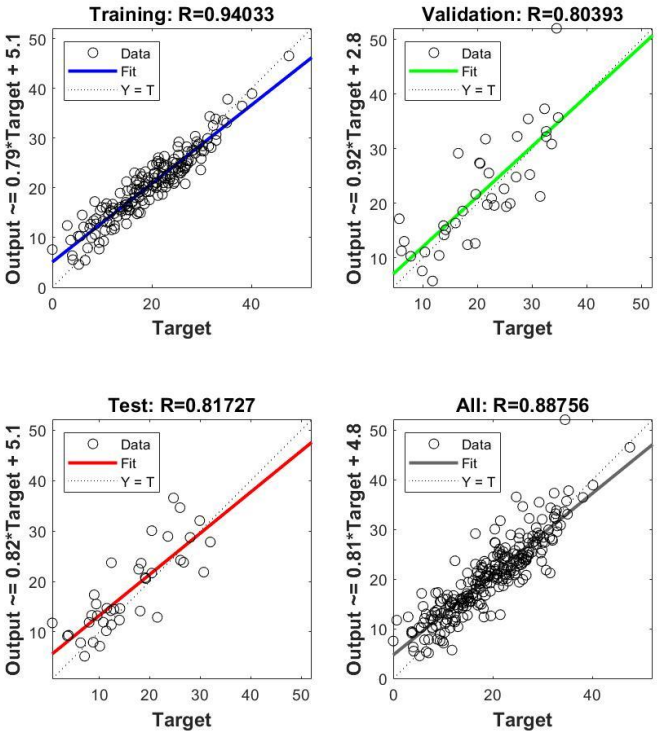

Figure 5. Regression plots for all datasets

- Now write your matlab code to do several runs (e.g. 5) for neural network training and observe how random initials affect the training results (you can calculate mean and standard deviation of MSE for the tests).

  Note: you may want to disable nntraintool display for each run by setting the showWindow parameter to false as below.
  net.trainParam.showWindow = false;

- Change the number of hidden nodes and evaluate the performance of neural network models for the training and testing dataset.

- Select/Define a different training function (the default is ‘trainlm’), re-run the test (e.g. for 20 hidden nodes), evaluate the performance of neural network models, and compare the performance with that of ‘trainlm’.