Artificial Intelligence
Training a Neural Network

Table of Contents

Load dataset 1
Define Y 1
Convert Y to 0 and 1 1
Convert Y to Dummy datatype 1
Define and Standardize X 2
Define NN Architecture 2
Train Model 3
Predict Model 4
Accuracy, Precision, Recall, FScore 4
Plot ROC curve 5

This code is used to build, train and test a neural network.

SEIS 764-02

Garth Mortensen, mort0052@stthomas.edu

2018.01.10

Load dataset

```
CellDNA = csvread('C:\tmp\CellDNA.csv');
```

Define Y

```
Y = CellDNA(:,[14]);
```

Convert Y to 0 and 1

Convert the Y target to binary values, where Y = 0 means bacterium is not interesting to study

```
Y(Y ~= 0) = 1;
Y(Y == 0) = 0;
```

Convert Y to Dummy datatype

Y should be a dummy variable, but dummyvar function cannot accept a vector of 0s and 1s. Instead, all values must be positive.

```
Y = Y + 1;
Y = dummyvar(Y);
```

```
% Then transpose. NN requires Y be transposed
Y_t = Y.';
```

Define and Standardize X

```
% Define X
XAll = CellDNA(:,1:13);

% Standardize X
XAll = zscore(XAll);

% Then transpose. NN requires Y be transposed
X_t = XAll.';
```

Define NN Architecture

We're going with a neuron structure of 2 hidden layers, with the below named neuron count in each

```
NN = patternnet([15 5])

% Define the two hidden layers using function poslin
% poslin = Positive linear transfer function = ReLu
% <https://www.mathworks.com/help/deeplearning/ref/poslin.html poslin>
NN.layers{1}.transferFcn = 'poslin';
NN.layers{2}.transferFcn = 'poslin';
```

```
NN =

    Neural Network

              name: 'Pattern Recognition Neural Network'
          userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 3
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 95
        sampleTime: 1

    connections:

       biasConnect: [1; 1; 1]
      inputConnect: [1; 0; 0]
      layerConnect: [0 0 0; 1 0 0; 0 1 0]
     outputConnect: [0 0 1]
```

```
subobjects:

          input: Equivalent to inputs{1}
         output: Equivalent to outputs{3}

         inputs: {1x1 cell array of 1 input}
         layers: {3x1 cell array of 3 layers}
        outputs: {1x3 cell array of 1 output}
         biases: {3x1 cell array of 3 biases}
   inputWeights: {3x1 cell array of 1 weight}
   layerWeights: {3x3 cell array of 2 weights}

functions:

       adaptFcn: 'adaptwb'
     adaptParam: (none)
       derivFcn: 'defaultderiv'
      divideFcn: 'dividerand'
    divideParam: .trainRatio, .valRatio, .testRatio
     divideMode: 'sample'
        initFcn: 'initlay'
     performFcn: 'crossentropy'
   performParam: .regularization, .normalization
       plotFcns: {'plotperform', plottrainstate, ploterrhist,
                 plotconfusion, plotroc}
     plotParams: {1x5 cell array of 5 params}
       trainFcn: 'trainscg'
     trainParam: .showWindow, .showCommandLine, .show, .epochs,
                 .time, .goal, .min_grad, .max_fail, .sigma,
                 .lambda

weight and bias values:

             IW: {3x1 cell} containing 1 input weight matrix
             LW: {3x3 cell} containing 2 layer weight matrices
              b: {3x1 cell} containing 3 bias vectors

methods:

          adapt: Learn while in continuous use
      configure: Configure inputs & outputs
         gensim: Generate Simulink model
           init: Initialize weights & biases
        perform: Calculate performance
            sim: Evaluate network outputs given inputs
          train: Train network with examples
           view: View diagram
    unconfigure: Unconfigure inputs & outputs
```

Train Model

Train a NN on X data using Y target This is a classification model, so use cross-entropy as loss function.

```
NN = train(NN, X_t, Y_t);
```

Predict Model

With the model trained, we now predict

```
[Y_hat] = NN(X_t);
```

Accuracy, Precision, Recall, FScore

First compute the confusion matrix, cfs

```
[conf, cfs] = confusion(Y_t, Y_hat)

% Class 0 (not flagged for study)
TP0 = cfs(1,1);
TN0 = cfs(2,2);
FP0 = cfs(1,2);
FN0 = cfs(2,1);
precision0 = TP0/(TP0 + FP0)
recall0 = TP0/(TP0 + FN0)
fscore0 = 2*((precision0 * recall0)/(precision0 + recall0));

% Class 1 (flagged for study)
TP1 = cfs(2:2,2);
TN1 = cfs(2,2);
FP1 = cfs(1:1,2);
FN1 = cfs(2:2,1);
precision1 = TP1/(TP1 + FP1)
recall1 = TP1/(TP1 + FN1)
fscore1 = 2*((precision1 * recall1)/(precision1 + recall1));

% Calculate accuracy
accuracy = sum(diag(cfs))/sum(cfs(:))
```

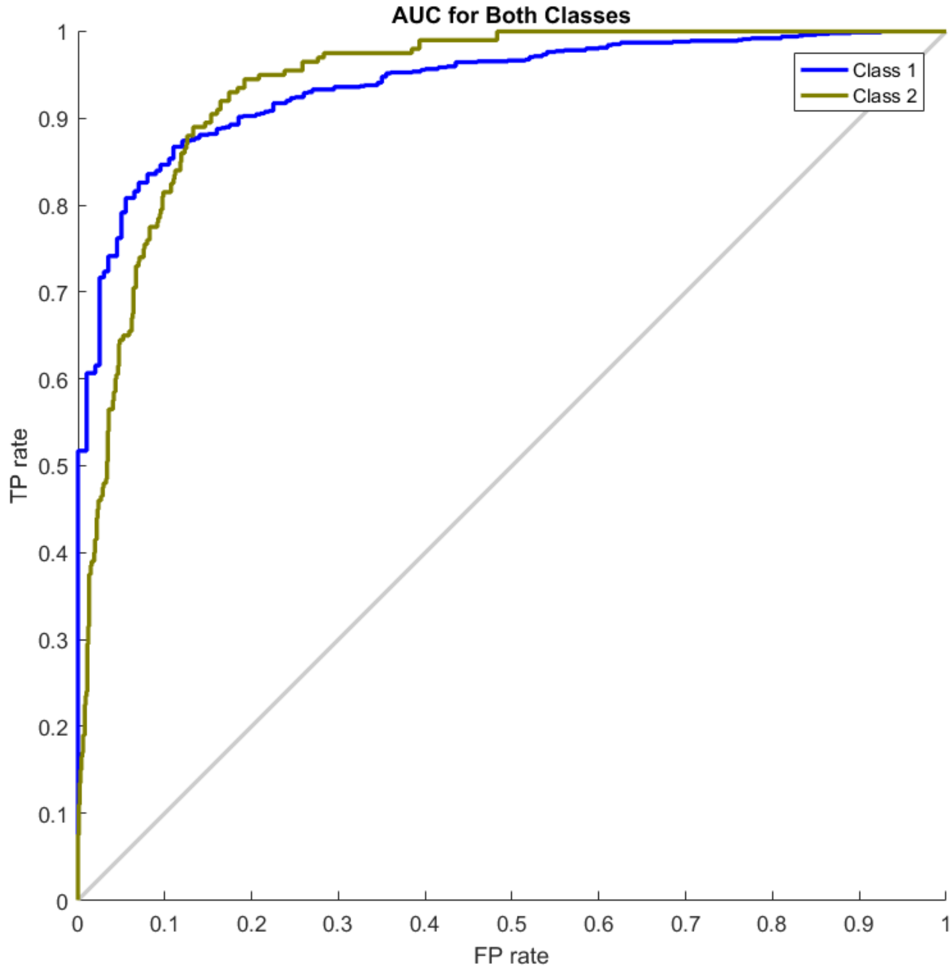
```
conf =

    0.1027


cfs =

   972    45
    80   120


precision0 =

    0.9558
```

```
recall0 =

    0.9240


precision1 =

    0.7273


recall1 =

    0.6000


accuracy =

    0.8973
```

Plot ROC curve

Place both lines on plot

```
figure(1)
plotroc(Y_t, Y_hat)
xlabel('FP rate'), ylabel('TP rate')
title('AUC for Both Classes')
```



Published with MATLAB® R2016a