%%data prep

features = csvread('CleanedTrackDataSet2.csv')

%%

Classes = csvread('PlaylistDataSet2.csv')

%%

class = Classes(:,1:1995);

%%

classsize = size(class,2);

%%

Laccruacy = []

%%

%%

for j = 1:100

i = round(classsize\*rand);

%%

%% number of classes to select

numClass = 1;

position = i;

numClass = numClass+position-1;

netclass = class(:,position : numClass);

%%

%% summing all rows where songs appeared

classSum = sum(netclass,2);

% location of where songs exists

classLoc = classSum > 0;

% recuperating the location of the songs

lclass2 = find(classLoc==1);

% location of where songs are not

classLocNot = classSum < 1;

% number of songs

numsongs = size(lclass2,1);

% selecting a number of songs that do not exist in the playlist at random

lclassNot = find(classLocNot==1);

classLocNotRed = randsample(lclassNot,numsongs,false);

%%

% joining the matrix of the sets of songs that exitst in the playlist with

% the ones that do not

ListSongs = cat(1,classLocNotRed,lclass2);

%%

netclass2 = netclass(ListSongs,:);

netfeatures = features(ListSongs,:);

%%

x = rot90(netfeatures);

t = rot90(netclass2);

%%

% clear class numClass classSum classLoc lclass2 netclass netclass2 netfeatures Sclass position numsongs ListSongs lclassNot lclass classLocNotRed classLocNot

% x = features

% t = Classes

%%

% 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 for low memory situations.

% the scaled conjugate gradient has been chosen over gradient descent as it

% has good performances over multiple types of datasets, although trainlm

% has been considered, 154 features are high for it. for us to really know

% which is better, we should test the principal ones and compare them as we

% do not know the structure of our dataset.

% https://au.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html

% https://arxiv.org/ftp/arxiv/papers/1409/1409.4727.pdf

trainFcn = 'trainscg'; % Scaled conjugate gradient backpropagation.

% trainFcn = 'traingd'; % standard gradient descent backpropagation.

% trainFcn = 'trainlm' % Levenberg-Marquardt

% trainFcn = 'trainrp' % Resilient Backpropagation

% Create a Pattern Recognition Network

%it seems that having multiple hidden layers does not make the equation

%more accurate, having a high value for the hidden layer, however, does help

%obtaining a high accuracy, but at the cost of some overfitting. having tried a few variations of the number of neurons, it seems that the

%increase of accuracy levels off at 200

% reducing the number of neurons per layer, however increasing the number

% of hidden layers also gives an accurate value but since it does not give

% a more accurate result it seems that there is no real need for multiple

% layers.

% hiddenLayerSize = [50 50 50];

learningRate = 0.02;

hiddenLayerSize = [20,10,10,10,10,20];

net = patternnet(hiddenLayerSize, trainFcn);

% net=feedforwardnet([200 200 200]);

% Setup Division of Data for Training, Validation, Testing, because we have

% 70 000 datapoints, it would be hard for us to choose wrong parameters, I

% prioritized validation ratio over test as test is only used once and 10%

% of 70 000 is more than enough for testing.

net.divideParam.trainRatio = 70/100;

net.divideParam.valRatio = 20/100;

net.divideParam.testRatio = 10/100;

%although the epochs is extreemly high, and could take an hour if it were

%to reach 2000, it is likely to stop first as there will be too many

%validation fails before reaching a stop.

net.trainParam.epochs = 2000;

%non concequencial param

net.trainParam.showWindow = true;

% this parameter, the learning rate, is purely an accuracy vs processing power, the lower the

% training parameter the smaller the adjustments and the more accurate the

% results. however, the lower the learning rate, the higher computation

% needed. the default value of 0.01 is low enough

net.trainParam.lr = learningRate;

% this is a good default parameter.

net.trainParam.min\_grad = 1e-5;

% number of validation fails before stopping

%by default the paramter is 6, but because we're dealing at max 2000 epochs

%we can increase this to 10 without much cost in time.

net.trainParam.max\_fail = 20;

% Train the Network

[net,tr] = train(net,x,t);

% Test the Network

y = net(x);

e = gsubtract(t,y);

performance = perform(net,t,y);

tind = vec2ind(t);

yind = vec2ind(y);

percentErrors = sum(tind ~= yind)/numel(tind);

epochs = tr.num\_epochs;

%%

% %View the Network

view(net)

%%

% %%Plots

% %%Uncomment these lines to enable various plots.

%

% figure, plotperform(tr)

% %%

% figure, plottrainstate(tr)

% %%

% figure, ploterrhist(e)

% %%

% figure, plotconfusion(t,y)

% %%

% t = mean(t)

% %%

% y = mean(y)

% figure, plotroc(t,y)

% %%axis([0 .1 0.9 1])

%

% %%

y2 = round(y);

k = t==y2;

%%

r = sum(sum(k));

tot = (size(y,2));

accuracy = r/tot;

Laccruacy = [Laccruacy,accuracy];

end

%%

%%

acc\_MEAN = mean(Laccruacy)

acc\_Median = median(Laccruacy)

acc\_MODE = mode(Laccruacy)

acc\_STD = std(Laccruacy)

acc\_VAR = var(Laccruacy)

acc\_QRT = quantile(Laccruacy,[0.25 0.75])

acc\_MIN = min(Laccruacy)

acc\_MAX = max(Laccruacy)

%%

hold off

histogram(Laccruacy)

%%

hold off

plot(Laccruacy(1:100))

hold on

line([1,100],[acc\_QRT(1),acc\_QRT(1)])

line([1,100],[acc\_QRT(2),acc\_QRT(2)])

line([1,100],[acc\_Median,acc\_Median])