Table of Contents


```
clear;
clc;

close all;     %closes all figures
```

data formatting

fer2013.csv - training data test.csv - test data for submission

```
disp('loading data ..............');

% The training set is  28709 samples
% testing set is  7178  samples
fer = fopen('fer2013.csv', 'r');                 % read fer2013.csv
tr = 28709;
testSize = 7178;
fullSize = tr + testSize;

%get headers and data from dataset
headers = textscan(fer, '%s %s %s', 1, 'delimiter', ', ');
data = textscan(fer, '%d %s %s', fullSize, 'delimiter', ',');
```

```
loading data ..............
```

all data for training

```
%defining all arrays for the grabbing the data
pixels = [];
emotions =[];
trainingPixels = [];
testPixels = [];
testEmotions =[];
testingPixels = [];

%parse out training values for emotions and pixels
```

```
for i=1:15000
    pixels = [pixels; data{1,2}(i)];                %parsing all pixel
 values
    emotions = [emotions; data{1,1}(i)];            %parsing all
 emotion values
    stringPix = char(pixels{i,1});                  %convert into
 string
    parsePix = str2double(strsplit(stringPix));     %seperate each
 pixel value
    trainingPixels = [trainingPixels; uint8(emotions(i,1)),
 uint8(parsePix)]; %put all into new training array
end
```

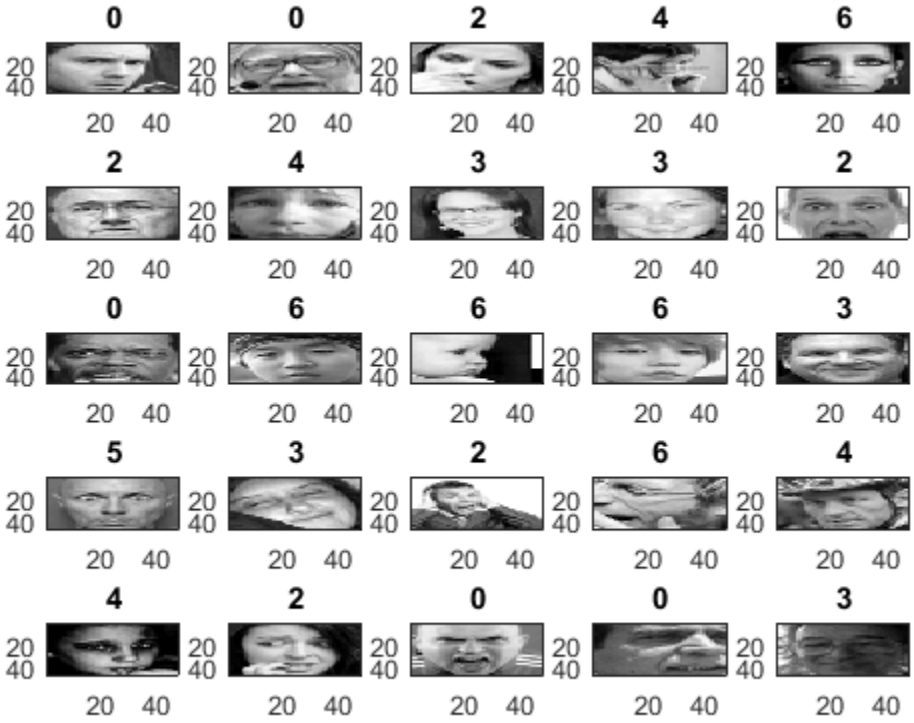
all data for testing

```
%parse out testing data
for j=tr+1:fullSize
    testPixels = [testPixels; data{1,2}(j)];            %parsing all
 pixel values
    testEmotions = [testEmotions; data{1,1}(j)];        %parsing all
 emotion values
end
%parse out each testing pixel we need
for k=1:testSize
    stringTestPix = char(testPixels{k,1});              %convert into
 string
    parseTestPix = str2double(strsplit(stringTestPix)); %seperate each
 pixel value
    testingPixels = [testingPixels; uint8(testEmotions(k,1)),
 uint8(parseTestPix)]; %put all new values into a new testing array
end



disp('Loaded ....');
```

Reshape the data to Visualize example for the digits sample

```
figure    ;                                                 % plot
 images
colormap(gray)                                              % set to
 grayscale
for i = 1:25                                                % preview
 first 25 samples
    subplot(5,5,i)                                          % plot
 them in 6 x 6 grid
    digit = reshape(trainingPixels(i, 2:end), [48,48])';    % row = 48
 x 48 image
    imagesc(digit)                                          % show the
 image
```

```
    title(num2str(trainingPixels(i, 1)))                    % show the
 label
end
```



The dataset stores samples in rows rather than in columns, so you need to

transpose it. Then you will partition the data so that you hold out 1/3 of the data for model evaluation, and you will only use 2/3 for training our artificial neural network model.

```
% n = size(trainingPixels, 1);                    % number of samples
 in the dataset
n = 9000;
targets  = double(trainingPixels(:,1));                 % 1st column
 is |label|
targets(targets == 0) = 7;                      % use '7' to present
 '0'
targetsd = dummyvar(targets);                   % convert label into a
 dummy variable

% No need for the first column in the (trainingPixels) set any longer
inputs = double(trainingPixels(:,2:end));               % the rest of
 columns are predictors; have to double so all inputs are the same

inputs = inputs';                   % transpose input
```

```
targets = targets';                 % transpose target
targetsd = targetsd';               % transpose dummy variable
```

partitioning the dataset based on random selection of indices

```
rng(1);                                                 % for
 reproducibility
patitionObject = cvpartition(n,'Holdout', uint8(n/3));   % hold out
 1/3 of the dataset

Xtrain = inputs(:, training(patitionObject));    % 2/3 of the input
 for training
Ytrain = targetsd(:, training(patitionObject));  % 2/3 of the target
 for training

Xtest = inputs(:, test(patitionObject));         % 1/3 of the input
 for testing
Ytest = targets(test(patitionObject));           % 1/3 of the target
 for testing
Ytestd = targetsd(:, test(patitionObject));      % 1/3 of the dummy
 variable for testing


disp('Ready for NNstart...');
```

```
Ready for NNstart...
```

Time to Run the Neural Network GUI Application

```
% type NNstart on the command prompt
```

Computing the Categorization Accuracy

```
Ypred = myNNfun(Xtest);             % predicts probability for each
 label
Ypred(:, 1:5)                       % display the first 5 columns
[~, Ypred] = max(Ypred);            % find the indices of max
 probabilities
sum(Ytest == Ypred) / length(Ytest); % compare the predicted vs.
 actual
```

```
ans =

    0.0042    0.0008    0.0000    0.0001    0.0000
    0.0013    0.2113    0.9811    0.5026    0.0001
    0.0008    0.3507    0.0189    0.0056    0.0001
    0.9607    0.0002    0.0000    0.1572    0.0000
```

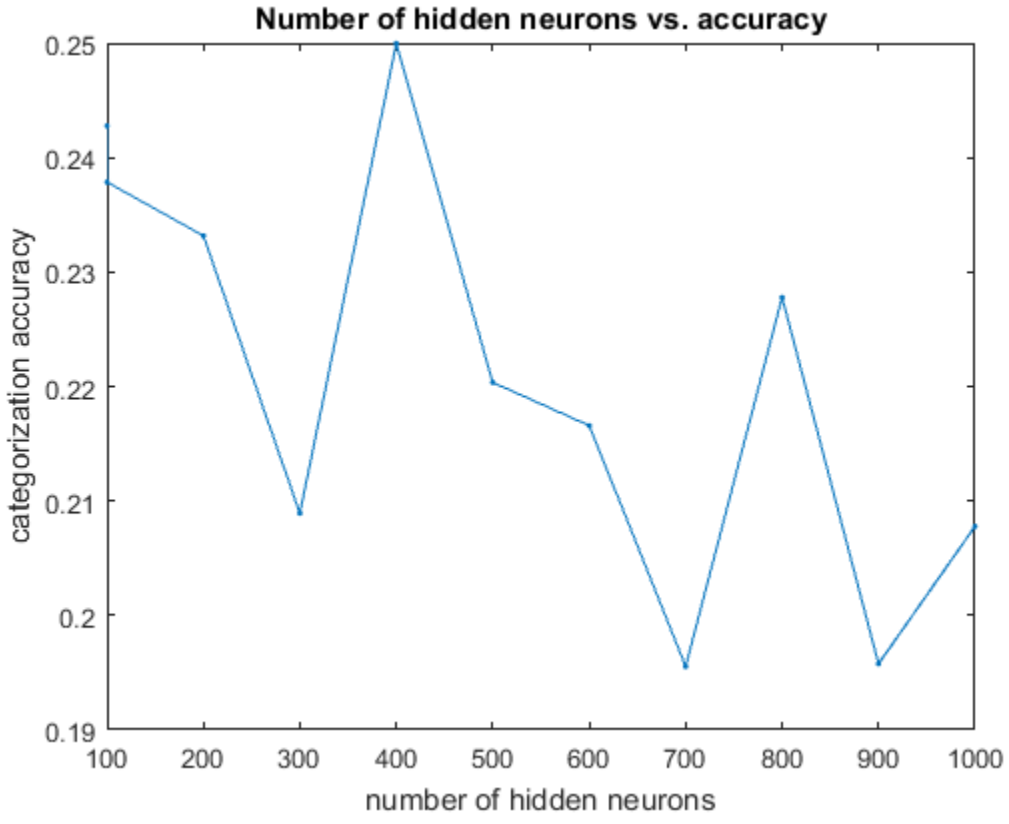
```
    0.0023    0.0410    0.0001    0.0007    0.0208
    0.0268    0.0001    0.0000    0.3337    0.9790
    0.0039    0.3958    0.0000    0.0001    0.0000
```

Sweep Code Block

Sweeping to choose different sizes for the hidden layer

```
sweep = [100,100:100:1000];                 % parameter values to test
scores = zeros(length(sweep), 1);          % pre-allocation
% we will use models to save the several neural network result from
 this
% sweep and run loop
models = cell(length(sweep), 1);        % pre-allocation
x = Xtrain;                             % inputs
t = Ytrain;                             % targets
trainFcn = 'trainscg';                  % scaled conjugate gradient

%    figure
for i = 1:length(sweep)
    hiddenLayerSize = sweep(i);         % number of hidden layer
 neurons
    net = patternnet(hiddenLayerSize);  % pattern recognition network
    net.divideParam.trainRatio = 70/100;% 70% of data for training
    net.divideParam.valRatio = 15/100;  % 15% of data for validation
    net.divideParam.testRatio = 15/100; % 15% of data for testing
    net = train(net, x, t);             % train the network
%     net = train(net, x, t,'useParallel','yes');
 % ,'useGPU','yes','showResources','yes'train the network
%
%
%     simpleclusterOutputs = sim(net,x);
%
%     % Ploting the ROC
%     plotroc(t,simpleclusterOutputs,sprintf('%d Neurons' ,sweep(i)));
%
%     formatSpec = "./Q5figSaves/N%dRoc";
%     savefigpath = sprintf(formatSpec,sweep(i));
%   pause();


    models{i} = net;                    % store the trained network
    p = net(Xtest);                     % predictions
    [~, p] = max(p);                    % predicted labels
    scores(i) = sum(Ytest == p) /length(Ytest);  % categorization
 accuracy
```

```
%     plot(sweep, scores, '.-')
% xlabel('number of hidden neurons')
% ylabel('categorization accuracy')
% title('Number of hidden neurons vs. accuracy')
%     pause();



end
% Let's now plot how the categorization accuracy changes versus number
 of
% neurons in the hidden layer.

figure
plot(sweep, scores, '.-')
xlabel('number of hidden neurons')
ylabel('categorization accuracy')
title('Number of hidden neurons vs. accuracy')
```



Published with MATLAB® R2017a