```
function proj08
% proj08
% proj08 is a driver that does the following:
%       (i)   graphs the distances of the randomized graphs to each
 other
%             in a color plot
%       (ii)  reports the prediction accuracy of shape matching for
 open
%             curve data
%       (iii) graphs 20 curves from DataClassification.mat that are
%             representative of the curve clusters
%       (iv)  reports the prediction accuracy of shape matching for
 closed
%             curve data
% Inputs: None
% Outputs: None
% Quan Le, CAAM 210, Fall 2019, Project 08
% Last Modified: November 10, 2019

close all

% open curves
[M,len] = opencurves(10); % generates master matrix
grapher(M,len) % color graph
classifier(M) % classification accuracy

% closed curves
[DC] = load("DataClassification.mat");
trainD = DC.trainingdata;
testD = DC.testdata;
plotting_clusters(trainD) % plots
closed_shapes(trainD,testD) % classification accuracy
end

function [M,len] = opencurves(iter)
% [M,len] = opencurves(iter)
% opencurves generates a master matrix of an iter amount of
 preprocessed
% randomized curves
% Inputs:
%           iter: the number of randomized curves desired
% Outputs:
%           M: the matrix containing all randomized curves
%           len: the length of one curve

% initialize variables
x = [0:0.02:pi];
A = [cos(x)];
B = [sin(x)];
C = [x.^2];
D = [x];
len = length(x);
```

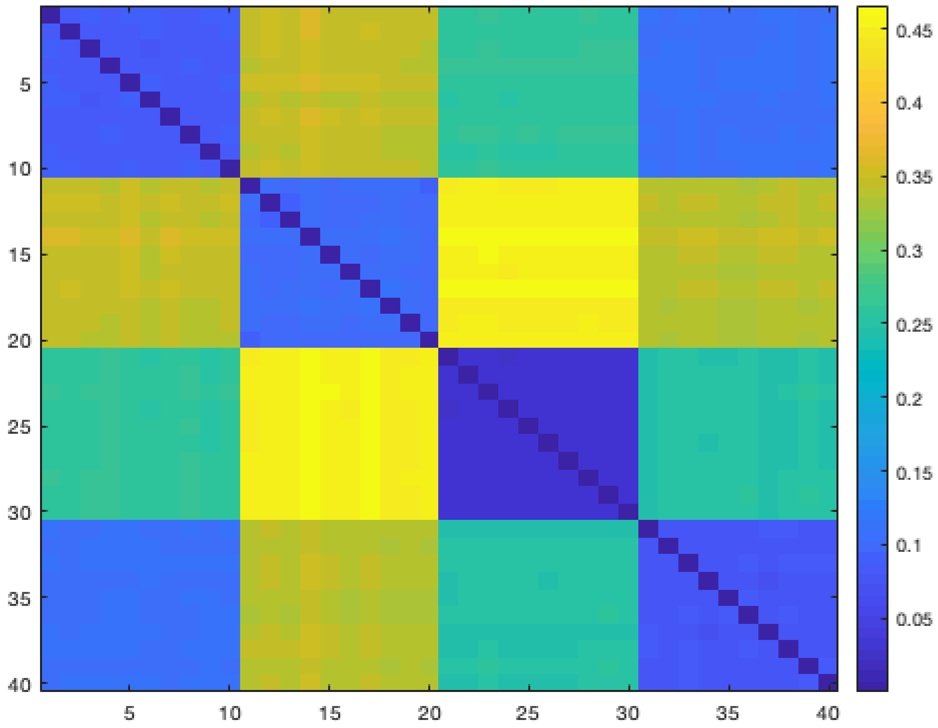
```
% create an iter-amount of randomized versions of initials
x1 = random_concat(x,iter);
A = random_concat(A,iter);
x2 = random_concat(x,iter);
B = random_concat(B,iter);
x3 = random_concat(x,iter);
C = random_concat(C,iter);
x4 = random_concat(x,iter);
D = random_concat(D,iter);

% master concatenation
M = [x1 x2 x3 x4; A B C D];

% preprocessing
for i = 1:40
    M(:, ((i-1)*len+1):(i*len) ) = preprocessing(M(:, ((i-1)*len+1):
(i*len) ));
end
end

function grapher(M,len)
% grapher(M,len)
% graphs the color coded distances between randomizd curves
% Inputs:
%           M, master matrix of all curves
%           len: length of one curve
% Outputs: none

% initialize distance matrix
P = [];

% find the distances from one curve to all the others
for j = 1:40
    for k =1:40
        C1 =  M(:, (j*len-len+1):(j*len) ); % grabs one curve from
 master matrix
        C2 =  M(:, (k*len-len+1):(k*len) ); % grabs another
        P(j,k) = distance(C1,C2);
    end
end

% plot color graph
imagesc(P)
colorbar
end

function [aconcat] = random_concat(a,iter)
% [aconcat] = random_concat(a,iter)
% creates an iter-amount of randomized versions of a
% Inputs:
%           a: the vector to be randomized
%           iter: number of times to concatenate said vector
% Outputs:
```

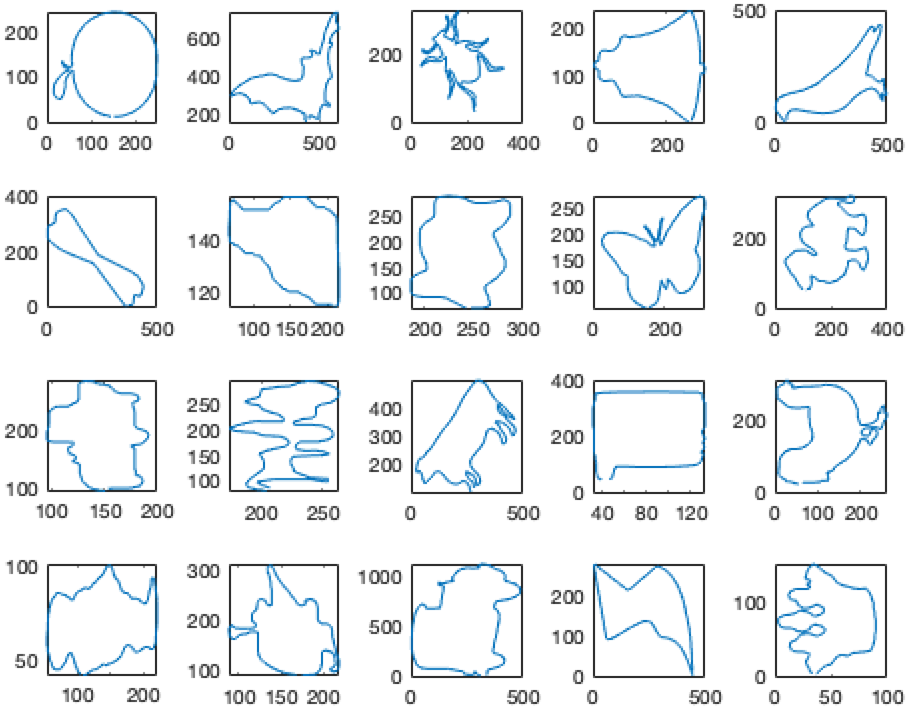
```
%           aconcat: concatenated verion of randomized a

% get length
len = length(a);

% randomize a
for i = 1:iter
    a_add = a;
    for ii = 1:len
        a_add(ii) = a(ii) + randn()*0.05;
    end

    % make sure you dont add an extra one, concatenate a
    if i ~= 1
        aconcat = [aconcat a_add];
    else
        aconcat = [a_add];
    end
end
end

function [Cf] = preprocessing(Ci)
% [Cf] = preprocessing(Ci)
% preprocesses a 2xn matrix, centers it and makes it's frobenious norm
 = 1
% Inputs:
%           Ci: the 2xn matrix to be preprocessed
% Outputs:
%           Cf: the normalized and centered version of Ci

% get length
len = size(Ci,2);

% get center
x_center = mean(Ci(1,:));
y_center = mean(Ci(2,:));
C_center = [ones(1,len).*x_center;ones(1,len).*y_center];

% center and normalize
C_hat = Ci - C_center;
Cf = C_hat./norm(C_hat,'fro');
end

function [dist] = distance(C1,C2)
% [dist] = distance(C1,C2)
% finds the distance between two curves
% Inputs:
%           C1: a curve represented by a 2xn matrix, [inputs; outputs]
%           C2: a curve of similar form
% Outputs:
%           dist: the distance between the two curves

% following the algorithm provided in the assignment:
A = C1*C2';
```

```
[U, S, V] = svd(A);
deter = det(U*V');
if deter > 0
    O_prime = U*V';
else
    O_prime = U*[1 0; 0 -1 ]*V';
end
dist = norm(C1-O_prime*C2, 'fro');
end

function classifier(M)
% classifier(M)
% deterines the accuracy of the classifier
% Inputs:
%            M: master matrix of all curves
% Outputs:   none

% create randomized data
[N,len] = opencurves(25);

% initialize count
count = 0;
for j = 1:100
    C1 =  N(:, (j*len-len+1):(j*len) ); % C1 from testing data
    P = [];
    for k =1:40
        C2 =  M(:, (k*len-len+1):(k*len) ); % C2 from training data
        P(k) = distance(C1,C2);
    end
    idx = find(P == min(P)); % closest match
    class = idx/10;
    up_bound = ceil(j/25);

    % determine if the closest match matches the original curve
    if (class <= up_bound) & (class > up_bound-1)
        count = count +1;
    end
end

disp("percent correct: " + string(count) + "%")
end

function plotting_clusters(trainD)
% plotting_clusters(trainD)
% plots a representative curve from each of 20 curve clusters in the
% training data
% Inputs:
%            trainD: training data
% Outputs:   none

figure
for idx = 1:20
    x=trainD(1,:,idx*15); % takes x and y from ONE of the 15 curves in
 a cluster
```

```
    y=trainD(2,:,idx*15);
    subplot(4,5,idx)
    plot(x,y)
    axis square
end
end

function [dist] = distance2(C1,C2)
% [dist] = distance2(C1,C2)
% finds the distance between two curves, factoring in different
 staring
% points
% Inputs:
%           C1: a curve represented by a 2xn matrix, [inputs; outputs]
%           C2: a curve of similar form
% Outputs:
%           dist: the distance between the two curves

% initialize variables
n = size(C1,2);
dist = distance(C1,C2);

% find min dist w all starting points
for i = 1:(n-1)
    Cnew = [C1(:, i+1 : n), C1(:, 1 : i)];
    dist_new = distance(Cnew,C2);
    if dist_new < dist
        dist = dist_new;
    end
end
end

function closed_shapes(trainD,testD)
% closed_shapes(trainD,testD)
% classifies and determines accuracy of classifier
% Inputs:
%           trainD: training data
%           testD: testing data
% Outputs:  none

% preprocess all training data
for idx = 1:300
    trainD(:,:,idx) = preprocessing(trainD(:,:,idx));
end

% initialize count
count = 0;

% classify each curve in testing data
for ii = 1:100
    C1 = testD(:,:,ii); % test curve
    P = [];
    for k =1:300 % iterate ove training data
        C2 =  trainD(:,:,k);
        P(k) = distance2(C1,C2); % distance vactor
    end
    idx = find(P == min(P));

    % find cluster
    class = ii/5;
    up_bound = ceil(idx/15);
    if (class <= up_bound) & (class > up_bound-1)
        count = count +1; % increment classifier
    end
end

disp("percent correct: " + string(count) + "%")

end
```

```
percent correct: 100%
percent correct: 98%
```





Published with MATLAB® R2019a