1. The data for this problem can be found in the file 'Pizza.csv' (used for HW #3, problem 2). The file contains the values of 7 features (moisture (moi), protein (prot), fat, ash, sodium, carb, and calories (cal)) for 300 selected pizzas. Each row (starting with the second row) is a sample that contains the brand of the selected pizza, its ID, and the feature values. For each classifier you are asked to find, provide the associated hyperplane.

   Ans: The Matlab code for the problems is on the next page.

   (a) Find a margin perceptron, i.e., a hyperplane, that separates the samples with brand 'A' from those with label 'B'. Repeat the same exercise to find (i) a margin perceptron that separates the samples with brand 'A' from those with label 'C' and (ii) a margin perceptron that separates the samples with brand 'B' from those with label 'C'.

   (b) Compute the margins provided by the linear classifiers you found in part (a).

   (c) Find the fusion rule for classifying brands 'A', 'B', and 'C'. Label the following instances using the fusion rule.

$$s1 = (49.29, 24.82, 21.68, 2.76, 0.52, 1.47, 3.00)$$
$$s2 = (30.95, 19.81, 42.28, 5.11, 1.67, 1.85, 4.67)$$
$$s3 = (50.33, 13.28, 28.43, 3.58, 1.03, 4.38, 3.27)$$

```
data = readtable('Pizza.csv') ;
dataTable = data(:, :) ;
brandArray = table2array(dataTable(:, 1)) ;
brandStr = string(brandArray) ; % get the brands

nFeature = 7 ;
feature = data(:, 3:9) ; % get the features
feature = table2array(feature) ;  % convert to double

brandA = find(brandStr == 'A') ;
brandB = find(brandStr == 'B') ;
brandC = find(brandStr == 'C') ;
lenA = length(brandA) ;
lenB = length(brandB) ;
lenC = length(brandC) ;

featureA = feature(brandA, :) ;
featureB = feature(brandB, :) ;
featureC = feature(brandC, :) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Part (a)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

featureAB = [featureA ; featureB] ;
featureABaug = [featureAB ones(lenA+lenB, 1)] ;
featureAC = [featureA ; featureC] ;
featureACaug = [featureAC ones(lenA+lenC, 1)] ;
featureBC = [featureB ; featureC] ;
featureBCaug = [featureBC ones(lenB+lenC, 1)] ;

labelAB = ones(size(brandA)) ;
labelAB = [labelAB ; -ones(size(brandB))] ;

SVMAB = fitclinear(featureAB, labelAB) ;
weightAB = [SVMAB.Beta ; SVMAB.Bias] ;
predictedAB = (featureABaug * weightAB)' ;
% The following provides the distance from the hyperplane
% (with + and -) and the minimum distance from each dataset
predictedABA = predictedAB(1:lenA)
predictedABB = predictedAB(lenA+1:lenA+lenB)
minmaxAB = [max(predictedABA) min(predictedABB)]

labelAC = ones(size(brandA)) ;
labelAC = [labelAC ; -ones(size(brandC))] ;

SVMAC = fitclinear(featureAC, labelAC) ;
weightAC = [SVMAC.Beta ; SVMAC.Bias] ;
predictedAC = (featureACaug * weightAC)' ;
% The following provides the distance from the hyperplane
```

```
% (with + and -) and the minimum distance from each dataset
predictedACA = predictedAC(1:lenA)
predictedACC = predictedAC(lenA+1:lenA+lenC)
minmaxAC = [max(predictedACA) min(predictedACC)]

labelBC = ones(size(brandB)) ;
labelBC = [labelBC ; -ones(size(brandC))] ;

SVMBC = fitclinear(featureBC, labelBC) ;
weightBC = [SVMBC.Beta ; SVMBC.Bias] ;
predictedBC = (featureBCaug * weightBC)' ;
% The following provides the distance from the hyperplane
% (with + and -) and the minimum distance from each dataset
predictedBCB = predictedBC(1:lenB)
predictedBCC = predictedBC(lenB+1:lenB+lenC)
minmaxBC = [max(predictedBCB) min(predictedBCC)]


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Part (b)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

marginAB = 2 / norm(weightAB(1:nFeature))
marginAC = 2 / norm(weightAC(1:nFeature))
marginBC = 2 / norm(weightBC(1:nFeature))


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Part (c)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

labelA2 = [ones(1, lenA) -ones(1, lenB) -ones(1, lenC)]' ;
labelB2 = [-ones(1, lenA) ones(1, lenB) -ones(1, lenC)]' ;
labelC2 = [-ones(1, lenA) -ones(1, lenB) ones(1, lenC)]' ;
len = lenA + lenB + lenC ;

OvA = zeros(nFeature+1, 3) ;
SVMA2 = fitclinear(feature(1:len,:), labelA2) ;
OvA(:,1) = [SVMA2.Beta ; SVMA2.Bias] ;

SVMB2 = fitclinear(feature(1:len,:), labelB2) ;
OvA(:,2) = [SVMB2.Beta ; SVMB2.Bias] ;

SVMC2 = fitclinear(feature(1:len,:), labelC2) ;
OvA(:,3) = [SVMC2.Beta ; SVMC2.Bias] ;

sample = [50.33 13.28 28.43 3.58 1.03 4.38 3.27 1 ; ...
                49.29 24.82 21.68 2.76 0.52 1.47 3.00 1 ; ...
                30.95 19.81 42.28 5.11 1.67 1.85 4.67 1 ] ;
```

```
DFvalue = sample * OvA

predictedLabel = zeros(3,1) ;
for i = 1:3,
        [M, predicedLabel(i)] = max(DFvalue(i,:))  ;
end
predicedLabel
```