Question-1

Since the one tries to model an algorithm via supervised learning, the knowledge of prior information of class probabilities is lethal, and can be calculated via given information;

$$P(C_i) = \frac{\sum\limits_{k=1}^{K} z_{ik}}{K} \tag{1}$$
$$z_{ik} = 1 \text{ if } x_k \in C_i$$

1 leads to the result; P(C_0)=0.5056 , P(C_1)=0.4944

The number of features is 65. However, the last feature represents the class label, so the number of the features is reduced to 64. In addition some of the features are not proper for model construction since they make no difference.

Since some of the features may be damaged or contain no value, they should be deleted from the data. Using MATLAB the damaged features are found and deleted from the data matrix. Their indices are; 1-8-9-16-24-32-33-40-41-48-49-57. They are removed for classification and left 53 features.

Question-2

As the one is required to calculate K-fold cross validation, it is essential to divide the data set into equally spaced parts. As the problem wants the one to divide into 10 sets they should be in size of 36, as there are 360 samples in the data.

Essential information about the process can be found in MATLAB code with comment lines for clarification.

Question-3

The question wants the one construct a mapping, named discriminant function, for given feature values. The features are given independent. One can calculate the discriminant function;

$$\begin{gathered} g_i(x) = \sum_j \sum_k z_{jk} log p_{ijk} + log P(C_i) \\ \text{where } \overset{*}{p}_{ijk} = \frac{\sum_t z_{jk}^t r_i^t}{\sum_t r_i^t} \end{gathered} \tag{2}$$

2 says one should calculate the relative probabilities of the values of features for each class then construct a matrix that contains all the information. This is required because the prior information of probabilities are not available and should be estimated. Once this is done, the validation sample, named the x vector in notation, should be multiplied wrt. the feature values with the probabilities. The number obtained represent the frequency of the data belonging to the class 'i'. In this data set with 53 features the matrix for each class should be 17x53, where 17 is for [0,16] data range. These matrices can be checked from MATLAB calling 'p' variable for each iteration, which is a 2 indexed cell. Also sum of each column of the matrices should be equal to the zero which is satisfied in the process.

The process should be repeated for the C_1 in order to compute the separation function. Where the function is;

$$\begin{gathered} dis = g_0(x) - g_1(x) \\ x \in C_0 \text{ if } dis \geq 0 \end{gathered} \tag{3}$$

This classification is done for each validation sample (36 samples), for K folds (10 times) and misclassifications are visualized. The knowledge of errors will be used in further questions. The label data is stored in 'label1' cell (label2 cell is reserved for second algorithm) and can be checked for each iteration from MATLAB.

Question-4

In this question, the assumption is probability of variables are independent of class. Using the prior information the one should calculate a new p matrix for modelling. This time the matrix should only include probabilities of variables given that the x is in the class C_i. The calculation of p matrix is basically averaging the value probabilities for any feature. For example, summing all probabilities for 'value' over 'k' features and dividing it to 'k'. This yields a vector of length 17. Where each index represent the probability of a value to be seen in C_i.

In that model features does not mean anything and the discriminant functions are calculated just by number appearances. This can be considered that just a feature is observed multiple times and its values are being compared. Computation is analogous to Question-3.

Question-5

Confusion matrices are calculated;

$$Confusion = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$$

$$\begin{bmatrix} TP = label == 1 \ \& \ actual = 1 \\ FN = label == 0 \ \& \ actual = 1 \\ TP = label == 1 \ \& \ actual = 0 \\ TP = label == 0 \ \& \ actual = 0 \end{bmatrix}$$

$$Confusion1 = \begin{bmatrix} 182 & 0 \\ 3 & 175 \end{bmatrix}$$

$$Confusion2 = \begin{bmatrix} 179 & 3 \\ 9 & 169 \end{bmatrix}$$

Question also wants the one to determine which algorithm is better, and prove the decision mathematically. Because of lack of prior information, one can use t-distribution for confidence interval where the variance is calculated form samples.

For comparison, one should calculate the difference of error for each fold;

$$\begin{gathered} p_i^{nk} = \frac{\sum_{t=1}^{N} |actual_t^{knt} - found_i^{knt}|}{N} \\ p_i^k = p_i^{0k} - p_i^{1k} \\ \text{where } i = \text{fold } t = \text{sample } k = \text{class} \end{gathered} \tag{4}$$

4 can be used for computing confidence interval;

$$\begin{aligned} m &= \frac{\sum_i p_i}{K} \\ s^2 &= \frac{\sum_i (p_i - m)^2}{K-1} \\ z &= \frac{\sqrt{K}m}{s} \end{aligned} \tag{5}$$

Computing 5 results in z=-2.7066. The absolute value of z refers to %99 confidence interval in the t-table. The null hypothesis is accepted. The algorithm in Q3 is more efficient compared to the algorithm in Q4.

Appendix

```
%Machine Learning Homework2 Demo
%Aziz Kocanaogullari 504141303
close all;
clear all;

%Initialize data
Dat=textread('optdigits01.txt');

%%
%Number of features
d=size(Dat,2);

%Prob. of each class
%Since the last column of the feature set determines whether the data
%belongs to set 1 or 2 then we can use '0' as C_0 '1' as C_1
%Also number of '1's correspond to number of C_1s overall
PC{1}=sum(Dat(:,65))/size(Dat,1);
PC{2}=1-PC{1};

%Detect improper features
%Since entire data matrix has >0 values one can use the summation of each
%column. If the sum. =0 that shows entire column is 0.
sumDat=sum(Dat,1);
%Indices of improper features
impind=find(sumDat<=0);

%Reconstruct the data matrix with proper values
DatRe=Dat;
DatRe(:,impind)=[];

%Number of usable features
d=size(DatRe,2);

%%
%This part is not required since the operation is done by cross validation
%but not randomly selected validation sets.
%Division of reconstructed data into validation and training sets
%Generate random numbers for randomly chosen observations
%ranob=randi([1,360],1,10);
%DatVa=DatRe(ranob,:); %Validation set
%DatTra=DatRe; %Training set
%DatTra(ranob,:)=[];

%%
for x=1:10,

%Division of reconstructed data into validation and training sets
DatVa=DatRe((x-1)*36+1:x*36,:); %Validation set
DatTra=DatRe; %Training set
DatTra((x-1)*36+1:x*36,:)=[];

%Calculation of pij s
%Num of features
for j=1:d-1,
    %Num of data vals.
    for k=1:17,

        % 1 corresponds to class 0 / 2 corresponds to class 1
        p{2}(j,k)=sum((DatTra(:,j)==k-1).*DatTra(:,end))/sum(DatTra(:,end));
        p{1}(j,k)=sum((DatTra(:,j)==k-1).*(DatTra(:,end)==0))/(size(DatTra,1)-sum(DatTra(:,end)));

    end
end

%The aim is to discriminate class 0 from 1 in other words g_0-g_1>0
%g_0-g_1=sum(zjk+log(p0jk)-log(p1jk))+log(PC0)-log(PC1)
for n=1:size(DatVa,1),

    dis=0;
    for j=1:d-1,

        %Exponential functions are used to avoid log(0)
        dis=log(p{1}(j,DatVa(n,j)+1)+10^-8)-log(p{2}(j,DatVa(n,j)+1)+10^-8)+dis;

    end

    dis=dis+log(PC{1}/PC{2});
    label{1}(n)=abs((dis>0)-1);

end

%%
%Calculation with same probabilities
%Since all numbers' probs. are free of feature, one can calculate the
%values of probabilities by ignoring the feature number and taking the mean
%of total appearance of the number.
pp{1}=sum(p{1},1);
pp{2}=sum(p{2},1);0
pp{1}=pp{1}/sum(pp{1});
pp{2}=pp{2}/sum(pp{2});

for n=1:size(DatVa,1),

    dis=0;
    for j=1:d-1,

        dis=log(pp{1}(DatVa(n,j)+1)+10^-8)-log(pp{2}(DatVa(n,j)+1)+10^-8)+dis;

    end

    dis=dis+log(PC{1}/PC{2});
    label{2}(n)=abs((dis>0)-1);

end

%Variables to check correct label rate
label1(x,:)=label{1};
label2(x,:)=label{2};
actuallabels(x,:)=DatRe((x-1)*36+1:x*36 ,end).';

end

%%
%Determining better model

%Confusion matrices
Con1=[sum(sum((label1==1).*(actuallabels==1))) sum(sum((label1==0).*(actuallabels==1)));
    sum(sum((label1==1).*(actuallabels==0))) sum(sum((label1==0).*(actuallabels==0)))];

Con2=[sum(sum((label2==1).*(actuallabels==1))) sum(sum((label2==0).*(actuallabels==1)));
```

```
    sum(sum((label2==1).*(actuallabels==0))) sum(sum((label2==0).*(actuallabels==0)))];

%Calculating error rates
%Calculating t-distribution vales
p1=sum(abs(label1-actuallabels),1)/size(label1,1);
p2=sum(abs(label2-actuallabels),1)/size(label2,1);
pi=p1-p2;

m=sum(pi)/size(label2,2);

var=sum((pi-m).^2)/(size(label2,2)-1);

%Hypothesis testing via t distribution since we have calculated the
%variance value from the samples

%If the condition satisfies the %95 confidence interval that shows
%algorithm 1 yields better results than the second algorithm

z=sqrt(size(label2,2))*m/sqrt(var);

%%
%Display Results
%Probabilities of the classes (supervised)
PC{1}
PC{2}

%Confusion matrices
Con1
Con2

%Hypothesis testing
disp(strcat('Calculated z value:',num2str(abs(z))));
```