ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΕΡΓΑΣΙΑ 6

ΑΛΒΑΝΑΚΗ ΠΑΡΑΣΚΕΥΗ

Α.Μ 57286

6.1 XOR

Στόχος ενός SVM είναι μεγιστοποίηση, αρχικά, του περιθωρίου margin και η εύρεση ενός βέλτιστου υπερεπιπέδου το οποίο διαχωρίζει γραμμικά τα δείγματα του dataset σε 2 κατηγορίες. Η δημιουργία του παραπάνω υπερεπιπέδου γίνεται με τη χρήση μη γραμμικών kernels με τα οποία αυξάνουμε τις διαστάσεις του προβλήματος εισάγοντας νέα χαρακτηριστικά από τα υπάρχοντα.

Η συνάρτηση XOR δεν είναι διαχωρίσιμη γραμμικά συνεπώς χρειαζόμαστε ένα μη γραμμικό kernel για το διαχωρισμό. Τα χαρακτηριστικά του SVM είναι τα εξής:

$$\omega_1 = [(0,0),(1,1)] \quad \omega_2 = [(0,1),(1,0)] \quad \Phi(x) = x_1^2 + \sqrt{2}x_1x_2 + x_2^2$$
$$K(x,y) = \Phi(x)\Phi(y) = (x_1y_1 + x_2y_2)^2 = (xy)^2$$

Για να επιτύχω τον παραπάνω διαχωρισμό χρησιμοποιώ την συνάρτηση SMO2 του matlab με παραμέτρους kernel=poly kpar1=0, kpar2=2 και C=4.Ουσιαστικά με τις παραπάνω παραμέτρους ορίζω το Kernel του SVM που αναφέρθηκε παραπάνω καθώς και το ανώτατο όριο C για τις τιμές τον Lagrange multipliers και ουσιαστικά ορίζει την ανοχή σε λάθη που θα υπάρξουν σε σχέση με το margin.

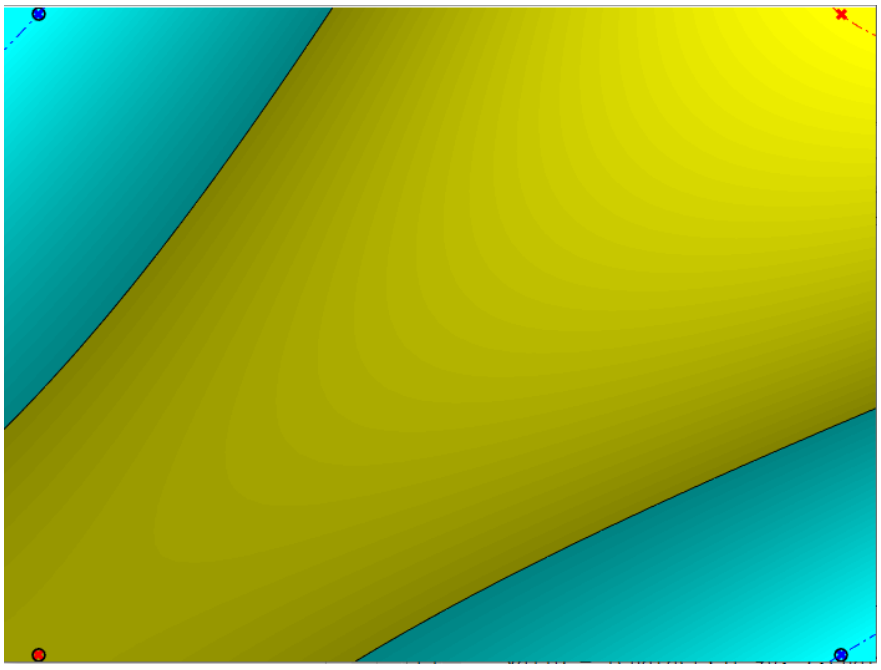
Ο κώδικας που χρησιμοποιήθηκε είναι ο εξής:

```
clear; clc;
close all
global figt4

X1 = [0 0; 0 1; 1 0; 1 1]';
y1 = [1 -1 -1 1];
l=2; % dimensionality
N=4; % number of vectors
figt4=1;
% Plot the training set X1
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'r.',X1(1,y1==-
1),X1(2,y1==-1),'bo')
figure(1), axis equal

kernel='poly'
kpar1=0;
kpar2=2;
C=4;
tol=0.001;
steps=100000;
eps=10^(-10);
method=1;
[alpha, b, w, evals, stp, glob] = SMO2(X1', y1', kernel, kpar1,
kpar2, C, tol, steps, eps, method);

mag=0.1;
xaxis=1;
yaxis=2;
input = zeros(1,size(X1',2));
bias=-b;
aspect=0;
svcplot_book(X1',y1',kernel,kpar1,kpar2,alpha,bias,aspect,mag,xaxis
,yaxis,input);
```

```
figure(figt4), axis equal

X_sup=X1(:,alpha'~=0);
alpha_sup=alpha(alpha~=0)';
y_sup=y1(alpha~=0);

% Classification of the training set
for i=1:N
t=sum((alpha_sup.*y_sup).*CalcKernel(X_sup',X1(:,i)',kernel,kpar1,k
par2)')-b;
    if(t>0)
        out_train(i)=1;
    else
        out_train(i)=-1;
    end
end

% Computing the training error
Pe_train=sum(out_train.*y1<0)/length(y1)

%Counting the number of support vectors
sup_vec=sum(alpha>0)

x1=sym('x1','real');
x2=sym('x2','real');
func=sum((alpha_sup.*y_sup).*(x1*X_sup(1,:)+x2*X_sup(2,:)).^2)-b;
f_func=simplify(func);
ff_func=vpa(f_func,10);
disp("The non linear function is:");
disp(ff_func);
```

Αποτελέσματα

Προκύπτει ο παρακάτω μη γραμμικός διαχωρισμός:



Για τη μη γραμμική εξίσωση ουσιαστικά χρησιμοποιούμε τους Lagrange multipliers που δημιουργήθηκαν. Χρησιμοποιώντας τη σχέση :

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

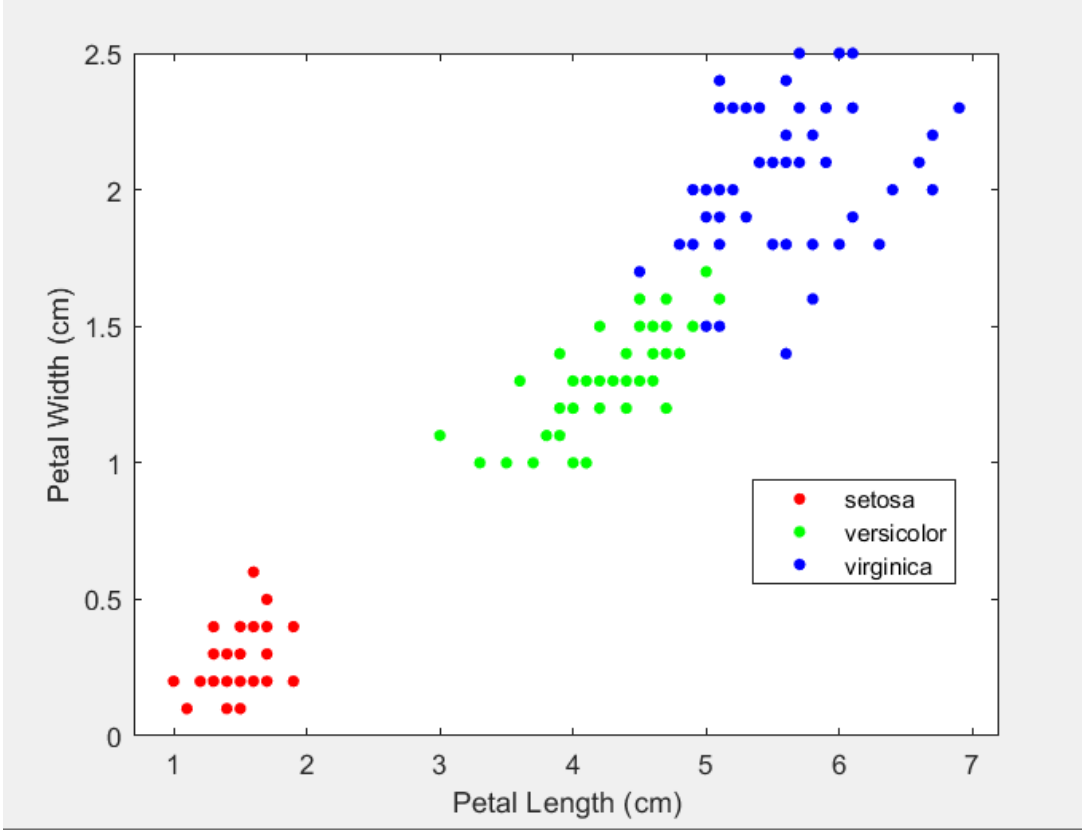
Προκύπτει η ακόλουθη εξίσωση:

- 1.200731596*x1^2 + 3.200731596*x1*x2 - 1.198902606*x2^2 + 0.1998171011

6.2 Iris Dataset

Α Ερώτημα:

Σε αυτό το ερώτημα διαχωρίζουμε την 2η κλάση από την 1η και 3η με ένα γραμμικό kernel στο SVM.Για το ερώτημα αυτό επιλέχθηκε η συνάρτηση fitcsvm καθώς μπορούμε απευθείας να βγάλουμε το error για κάθε set χωρίς να το υπολογίζουμε χειροκίνητα. Κάνοντας plot καποια χαρακτηριστικά βλέπουμε ότι η 2η κλάση δεν είναι γραμμικά διαχωρίσιμη από τις άλλες 2 (που εμείς σε αυτό το ερώτημα έχουμε ενώσει σε 1 non-versicolor).



Για τα 3 χαρακτηριστικά (1,2 και 4) παίρνουμε ως error στο training set 27.78%, 30% για την ταξινόμηση του validation set και 33.33% για την ταξινόμηση του test set.Τα αποτελέσματα αυτά είναι αναμενόμενα καθώς τα 2 σετ δεν είναι γραμμικά διαχωρίσιμα.

Ομοίως και στα 4 χαρακτηριστικά παίρνουμε 28.89% για την ταξινόμηση του training set. 26.67% για την ταξινόμηση του validation set και 33.33% για την ταξινόμηση του test set.

Στην εργασία 5 δεν έγινε ακρι΄βως ο ιδιος διαχωρισμός του Iris-versicolor από τις άλλες 2 κλάσεις. Ωστόσο αν τρέξουμε τους αλγορίθμους της άσκησης 5 για το ζητούμενο διαχωρισμό με τη μέθοδος των ελαχίστων τετραγώνων με χρήση του ψευδοαντιστρόφου (LS)(error= 26%)και με την επαναληπτική μέθοδο του Ho-Kashyap(error=26%) παίρνουμε για το LS error= 26,2% και για την Ho-Kashyap error=26%. Τα αποτελέσματα εκεί ήταν πολύ καλύτερα, αφού καταφέραμε να επιτύχουμε μικρότερο σφάλμα στην ταξινόμηση του training set με τη χρήση και των τεσσάρων χαρακτηριστικών.

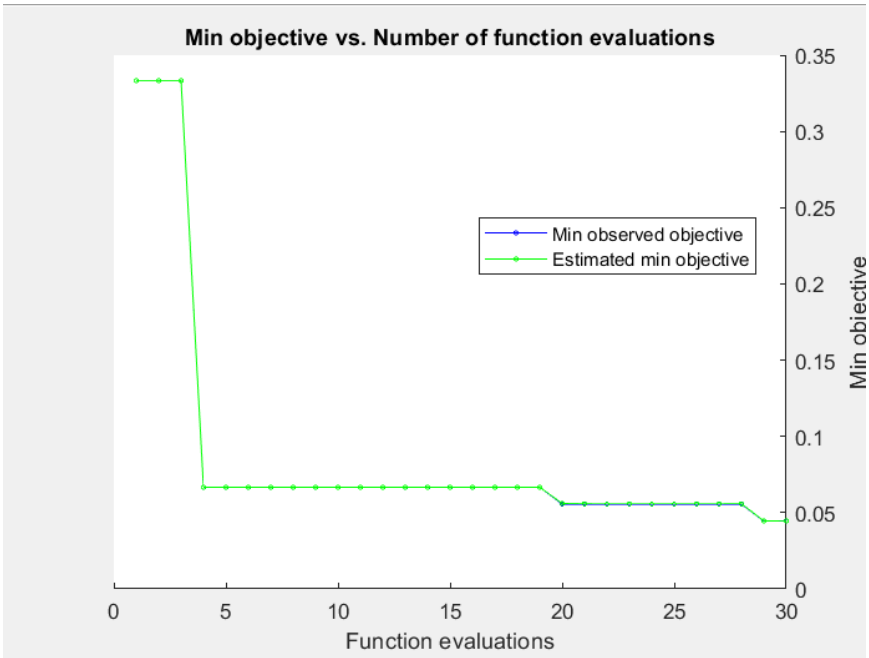
Ο κώδικας που χρησιμοποιήθηκε για τα 4 χαρακτηριστικά είναι ο εξής:

```
clear;
clc;
close all;

data = readtable('iris.txt');

x1 = table2array(data(:,1));
x2 = table2array(data(:,2));
x3 = table2array(data(:,3));
x4 = table2array(data(:,4));
charact=([x1 x2 x3 x4]);
% Training Set
train = [charact(1:30,:);charact(51:80,:);charact(101:130,:)];
% Validation Set
valid = [charact(31:40,:);charact(81:90,:);charact(131:140,:)];
% Test Set
test = [charact(41:50,:);charact(91:100,:);charact(141:150,:)];
y_train = [ones(1,30) (-1)*ones(1,30) ones(1,30)];
y_valid = [ones(1,10) (-1)*ones(1,10) ones(1,10)];
y_test = [ones(1,10) (-1)*ones(1,10) ones(1,10)];
kernel = 'linear';
%train the model
SVMModel =
fitcsvm(train,y_train,'Standardize',false,'KernelFunction',kernel);
classOrder = SVMModel.ClassNames;
sv = SVMModel.SupportVectors;
%error for each set
Train_Error       = loss(SVMModel,train,y_train)
Validation_Error  = loss(SVMModel,valid,y_valid)
Test_Error        = loss(SVMModel,test,y_test)
```

Β Ερώτημα:

Σε αυτό το ερώτημα χρησιμοποιούμε μη γραμμικά kernel.Η fitsvm μας δίνει τη δυνατότητα να «τρέξουμε» το πρόγραμμα με διάφορα kernel αυτόματα και να επιλέξει το kernel που μας δίνει την καλύτερη ταξινόμηση και συνεπώς το μικρότερο σφάλμα.

Έτσι παίρνουμε την παρακάτω γραφική για τη μεταβολή του σφάλματος.



Μετα από δοκιμές διάφορων kernel παίρνουμε ως βέλτιστο το εξής:

BoxConstraint	KernelScale	KernelFunction	PolynomialOrder
3.3591	NaN	polynomial	2

Για τα 3 χαρακτηριστικά ως error στο training set έχουμε 3.3%, 4.44% για την ταξινόμηση του validation set και 3.3% για την ταξινόμηση του test set.Τα αποτελέσματα αυτά είναι αναμενόμενα καθώς τα 2 σετ είναι μη γραμμικά διαχωρίσιμα και μπορούν να ταξινομηθούν με πολυωνυμικό kernel και συνεπώς μη γραμμικό ταξινομητή ικανοποιητικά.
Για τέσσερα χαρακτηριστικά η βέλτιστη απόδοση έχει σφάλμα 1,11% για την ταξινόμηση του training set, 6.67% για την ταξινόμηση του validation set και 3,33% για την ταξινόμηση του test set.

Σε σύγκριση με την 5η εργασία τα αποτελέσματα είναι σαφώς καλύτερα το οποίο είναι αναμενόμενο καθώς μία μη γραμμική μέθοδος ταξινόμησης μπορεί να πετύχει μικρότερο σφάλμα από μία γραμμική. Όπως περιμέναμε, τα αποτελέσματα της non-linear ταξινόμησης είναι πολύ καλύτερα απ' ότι αυτά της linear.Επίσης χρησιμοποιώντας όλα τα διαθέσιμα χαρακτηριστικά το σφάλμα μειώνεται αφού χρησιμοποιείται περισσότερη πληροφορία.

Κώδικας για τα 4 χαρακτηριστικά(non-linear)

```
clear;
clc;
close all;

data = readtable('iris.txt');
```

```
x1 = table2array(data(:,1));
x2 = table2array(data(:,2));
x3 = table2array(data(:,3));
x4 = table2array(data(:,4));
charact=([x1 x2 x3 x4]);
% Training Set
train = [charact(1:30,:);charact(51:80,:);charact(101:130,:)];
% Validation Set
valid = [charact(31:40,:);charact(81:90,:);charact(131:140,:)];
% Test Set
test = [charact(41:50,:);charact(91:100,:);charact(141:150,:)];
y_train = [ones(1,30) (-1)*ones(1,30) ones(1,30)];

y_valid = [ones(1,10) (-1)*ones(1,10) ones(1,10)];
y_test = [ones(1,10) (-1)*ones(1,10) ones(1,10)];
SVMModel =
fitcsvm(train,y_train,'Standardize',false,'OptimizeHyperparameters'
,{'BoxConstraint','KernelScale','KernelFunction','PolynomialOrder'}
);
classOrder = SVMModel.ClassNames;
sv = SVMModel.SupportVectors;
Train_Error       = loss(SVMModel,train,y_train)
Validation_Error  = loss(SVMModel,valid,y_valid)
Test_Error        = loss(SVMModel,test,y_test)
```

Γ Ερώτημα:

One vs all: Βρίσκουμε τους γραμμικούς ταξινομητές που ταξινομούν τα στοιχεία ανάμεσα σε κάθε κλάση ενάντια σε όλες τις άλλες. Επιλέγουμε και πάλι την κλάση της οποίας η συνάρτηση gk(x) είναι μέγιστη.

Για το ερώτημα αυτό δεν μπορούμε απλά να βάλουμε διαφορετικά labels σε κάθε κλάση και να χρησιμοποιήσουμε της fitsvm καθώς θα χρησιμοποιήσει one vs one ταξινόμηση. Συνεπώς κάνουμε 3 one vs all ταξινομήσεις(1 για κάθε κλάση)και επιλέγουμε να ταξινομήσουμε κάθε sample στην κλάση με τη μεγαλύτερη πιθανότητα. Για την εκπαίδευση των svm χρησιμοποιούμε το training set και γραμμικά kernel.

Έτσι παίρνουμε τα παρακάτω σφάλματα:

```
The error for the characteristics 1,2 and 4  with the one versus all method for the training set is:
    0.0667

The error for the characteristics 1,2 and 4  with the one versus all method for the validation set is:
    0.2000

The error for the characteristics 1,2 and 4  with the one versus all method for the test set is:
    0.0333
```

Συγκρίνοντας τα σφάλματα για κάθε μια κλάση one vs all στην 5η εργασία και στο ερώτημα αυτό (στα training sets)παρατηρούμε πως η χρήση γραμμικού SVM οδηγεί σε καλύτερα αποτελέσματα καθώς στην 5η εργασία είχα σφάλμα 0,2867 για την κλάση 2 vs 1-3 ,0,26 με χαρακτηριστικά 1,2,3, 0.26 για την ίδια κλαση με χαρακτηριστικά 2,3,4 και 0,1 και 0,06 για την κλάση 3 vs 1-2 με τα αντίστοιχα χαρακτηριστικα.Αυτό το γεγονός πιθανόν

να οφείλεται στη χρήση των χαρακτηριστικών 1,2,4 κατά τον σχηματισμό και την ταξινόμηση των στοιχείων με τον γραμμικό SVM.

Κώδικας για one-vs all για τα 4 χαρακτηριστικά

```
clear;
clc;
close all;
data = readtable('iris.txt');

x1 = table2array(data(:,1));
x2 = table2array(data(:,2));
x3 = table2array(data(:,3));
x4 = table2array(data(:,4));
charact=([x1 x2 x3 x4]);

% Training Set
train = [charact(21:50,:);charact(51:80,:);charact(101:130,:)];
% Validation Set
valid = [charact(1:10,:);charact(91:100,:);charact(141:150,:)];
% Test Set
test = [charact(11:20,:);charact(81:90,:);charact(131:140,:)];
train_Scores=zeros(90,3);
val_Scores=zeros(30,3);
test_Scores=zeros(30,3);
N=90;
kernel='linear'
kpar1=0;
kpar2=2;
C=15;
tol=0.001;
steps=100000;
eps=10^(-10);
method=1;
%124 linear 1rst one vs all
y_train = [ones(1,30) (-1)*ones(1,30) (-1)*ones(1,30)];

y_valid = [ones(1,10) (-1)*ones(1,10) (-1)*ones(1,10)];
y_test = [ones(1,10) (-1)*ones(1,10) (-1)*ones(1,10)];
SVMModel_1 =
fitcsvm(train,y_train,'Standardize',false,'KernelFunction',kernel);
%training score
[~,score1]=predict(SVMModel_1,train);
train_Scores(:,1)=score1(:,2);
%validation score
[~,val_score1]=predict(SVMModel_1,valid);
val_Scores(:,1)=val_score1(:,2);
%test score
[~,test_score1]=predict(SVMModel_1,test);
test_Scores(:,1)=test_score1(:,2);

%124 linear 2nd class one vs all
y_train = [(-1)*ones(1,30) ones(1,30) (-1)*ones(1,30)];

y_valid = [(-1)*ones(1,10) ones(1,10) (-1)*ones(1,10)];
y_test = [(-1)*ones(1,10)  ones(1,10) (-1)*ones(1,10)];
SVMModel_2 =
fitcsvm(train,y_train,'Standardize',false,'KernelFunction',kernel);
```

```
%train score
[~,score2]=predict(SVMModel_2,train);
train_Scores(:,2)=score2(:,2);% Second column contains positive-
class scores
%validation score
[~,val_score2]=predict(SVMModel_2,valid);
val_Scores(:,2)=val_score2(:,2);
%test score
[~,test_score2]=predict(SVMModel_2,test);
test_Scores(:,2)=test_score2(:,2);

%124 linear 3rd class one vs all
y_train = [(-1)*ones(1,30) (-1)*ones(1,30) ones(1,30)];

y_valid = [(-1)*ones(1,10) (-1)*ones(1,10) ones(1,10)];
y_test = [(-1)*ones(1,10)  (-1)*ones(1,10) ones(1,10)];
SVMModel_3 =
fitcsvm(train,y_train,'Standardize',false,'KernelFunction',kernel);
%train scores
[~,score3]=predict(SVMModel_3,train);
train_Scores(:,3)=score3(:,2);% Second column contains positive-
class scores
%validation score
[~,val_score3]=predict(SVMModel_3,valid);
val_Scores(:,3)=val_score3(:,2);
%test score
[~,test_score3]=predict(SVMModel_3,test);
test_Scores(:,3)=test_score3(:,2);

%find the maximum probability and assign each sample to its maximum
%probability for all the sets
[~,train_maxScore]=max(train_Scores,[],2);
[~,val_maxScore]=max(val_Scores,[],2);
[~,test_maxScore]=max(test_Scores,[],2);
%assign true labels 1 2 3 for each class
y_train = [ones(1,30) (2)*ones(1,30) 3*ones(1,30)];
y_valid = [ones(1,10) 2*ones(1,10) 3*ones(1,10)];
y_test = [ones(1,10) 2*ones(1,10) 3*ones(1,10)];

tr_error =sum((train_maxScore'~=y_train))/length(y_train);
disp("The error for all the characteristics with the one versus all
method for the training set is:");
disp(tr_error);
val_error =sum((test_maxScore'~=y_valid))/length(y_valid);
disp("The error for all the characteristics with the one versus all
method for the validation set is:");
disp(val_error);
test_error =sum((test_maxScore'~=y_test))/length(y_test);
disp("The error for all the characteristics  with the one versus
all method for the test set is:");
disp(test_error);
```

Δ Ερώτημα:

Όμοια με το προηγούμενο ερώτημα κάνω one vs all classification για κάθε κλάση και επιλέγω αυτή που δίνει μέγιστη τιμή. Χρησιμοποιώντας την επιλογή

OptimizeHyperparameters μπορώ να ελέγξω όλους τους δυνατούς συνδυασμούς που μπορώ να κάνω και να επιλέξω τον βέλτιστο όπως έκανα και στο ερώτημα β.

Έτσι παίρνουμε τα παρακάτω σφάλματα:

```
The error for all the characteristics  with the one versus all method for the training set is:
     0

The error for all the characteristics with the one versus all method for the validation set is:
    0.0667

The error for all the characteristics with the one versus all method for the test set is:
     0
```

Συγκρίνοντας τα σφάλματα για κάθε μια κλάση στα ερωτήματα Α, Β, Γ και στο ερώτημα αυτό παρατηρούμε πως εδώ έχουμε τα καλύτερα αποτελέσματα. Αυτό είναι αναμενόμενο καθώς ταξινομείται κάθε sample σε μια κλάση 3 φορές(3 one vs all ταξινομιτές) και επιλέγεται ως τελική κλάση αυτή με τη μεγαλύτερη τιμή σε αντίθεση με τις προηγούμενες που γινόταν 1 one-vs all classification και η ταξινόμιση γινεται με non linear ταξινομιτή που στη συγκεκριμένη περίπτωση που έχουμε μη γραμμικά διαχωρίσιμες κλάσεις παίζει ρόλο.Έτσι έχουμε καλύτερη ακρίβεια στα αποτελέσματα.

Κώδικας για one-vs all για τα 4 χαρακτηριστικά(non linear)

```
clear;
clc;
close all;
data = readtable('iris.txt');
x1 = table2array(data(:,1));
x2 = table2array(data(:,2));
x3 = table2array(data(:,3));
x4 = table2array(data(:,4));
charact=([x1 x2 x3 x4]);
% Training Set
train = [charact(21:50,:);charact(51:80,:);charact(101:130,:)];
% Validation Set
valid = [charact(1:10,:);charact(81:90,:);charact(131:140,:)];
% Test Set
test = [charact(11:20,:);charact(91:100,:);charact(141:150,:)];
train_Scores=zeros(90,3);
val_Scores=zeros(30,3);
test_Scores=zeros(30,3);N=90;
kpar1=0; kpar2=2; C=15;tol=0.001;steps=100000;eps=10^(-10);
method=1;
%124 linear 1rst one vs all
y_train = [ones(1,30) (-1)*ones(1,30) (-1)*ones(1,30)];
y_valid = [ones(1,10) (-1)*ones(1,10) (-1)*ones(1,10)];
y_test = [ones(1,10) (-1)*ones(1,10) (-1)*ones(1,10)];
SVMModel_1 =
fitcsvm(train,y_train,'Standardize',false,'OptimizeHyperparameters'
,{'BoxConstraint','KernelScale','KernelFunction','PolynomialOrder';
%training score
[~,score1]=predict(SVMModel_1,train);
train_Scores(:,1)=score1(:,2);
%validation score
[~,val_score1]=predict(SVMModel_1,valid);
```

```
val_Scores(:,1)=val_score1(:,2);
%test score
[~,test_score1]=predict(SVMModel_1,test);
test_Scores(:,1)=test_score1(:,2);

%124 linear 2nd class one vs all
y_train = [(-1)*ones(1,30) ones(1,30) (-1)*ones(1,30)];

y_valid = [(-1)*ones(1,10) ones(1,10) (-1)*ones(1,10)];
y_test = [(-1)*ones(1,10)  ones(1,10) (-1)*ones(1,10)];
SVMModel_2 =
fitcsvm(train,y_train,'Standardize',false,'OptimizeHyperparameters'
,{'BoxConstraint','KernelScale','KernelFunction','PolynomialOrder';
%train score
[~,score2]=predict(SVMModel_2,train);
train_Scores(:,2)=score2(:,2);% Second column contains positive-
class scores
%validation score
[~,val_score2]=predict(SVMModel_2,valid);
val_Scores(:,2)=val_score2(:,2);
%test score
[~,test_score2]=predict(SVMModel_2,test);
test_Scores(:,2)=test_score2(:,2);
%124 linear 3rd class one vs all
y_train = [(-1)*ones(1,30) (-1)*ones(1,30) ones(1,30)];

y_valid = [(-1)*ones(1,10) (-1)*ones(1,10) ones(1,10)];
y_test = [(-1)*ones(1,10)  (-1)*ones(1,10) ones(1,10)];
SVMModel_3 =
fitcsvm(train,y_train,'Standardize',false,'OptimizeHyperparameters'
,{'BoxConstraint','KernelScale','KernelFunction','PolynomialOrder';
%train scores
[~,score3]=predict(SVMModel_3,train);
train_Scores(:,3)=score3(:,2);% Second column contains positive-
class scores
%validation score
[~,val_score3]=predict(SVMModel_3,valid);
val_Scores(:,3)=val_score3(:,2);
%test score
[~,test_score3]=predict(SVMModel_3,test);
test_Scores(:,3)=test_score3(:,2);
%find the maximum probability and assign each sample to its maximum
%probability for all the sets
[~,train_maxScore]=max(train_Scores,[],2);
[~,val_maxScore]=max(val_Scores,[],2);
[~,test_maxScore]=max(test_Scores,[],2);
%assign true labels 1 2 3 for each class
y_train = [ones(1,30) (2)*ones(1,30) 3*ones(1,30)];
y_valid = [ones(1,10) 2*ones(1,10) 3*ones(1,10)];
y_test = [ones(1,10) 2*ones(1,10) 3*ones(1,10)];
tr_error =sum((train_maxScore'~=y_train))/length(y_train);
disp("The error for all the characteristics  with the one versus
all method for the training set is:");
disp(tr_error);
val_error =sum((val_maxScore'~=y_valid))/length(y_valid);
disp("The error for all the characteristics with the one versus all
method for the validation set is:");
disp(val_error);
test_error =sum((test_maxScore'~=y_test))/length(y_test);
```

```
disp("The error for all the characteristics with the one versus all
method for the test set is:");
disp(test_error);
```