LABORATORY EXPERIENCE #9
(Support Vector Machines)

Introduction

This laboratory experience was focused on implementing binary classification through **Support Vector Machine (SVM).** This method allows to separate a dataset in two classes defining an hyperplane that divides the data space. This hyperplane is defined by two parameters w and b, that are chosen to maximize the margin of the hyperplane with respect to the closest points, called **support vectors.**
Increasing this margin it is possible to obtain a better-defined separation between the two classes.
The experiment has been proposed on the arrhythmia dataset, already treated in the previous laboratory experiences.

Data preparation

As in the previous experiments, the dataset has been normalized and the original 16 classes have been collapsed into a two-classes set.

Algorithms implementation and results

The training of the Support Vector Machine has been repeated varying the box constraint C within a set of values. The results show that increasing this value, we are able to maximize the margin and separate the dataset in a better way, but we might increase the probability of misclassification in testing phase, as we decrease the tolerance for ambiguous points.
Then, it was selected the box constraint value that allowed the best trade-off between maximizing the margin and having tolerance for wrongly-positioned points.
The probability of misclassification can be evaluated through the **k-fold cross-validation**, that divides the dataset in K groups of same dimensions and use a subset of them for training and the remaining part for testing. We can see that repeating the experiment we may obtain different behaviour of the probability of misclassification depending on how the cross-validation divides the data.

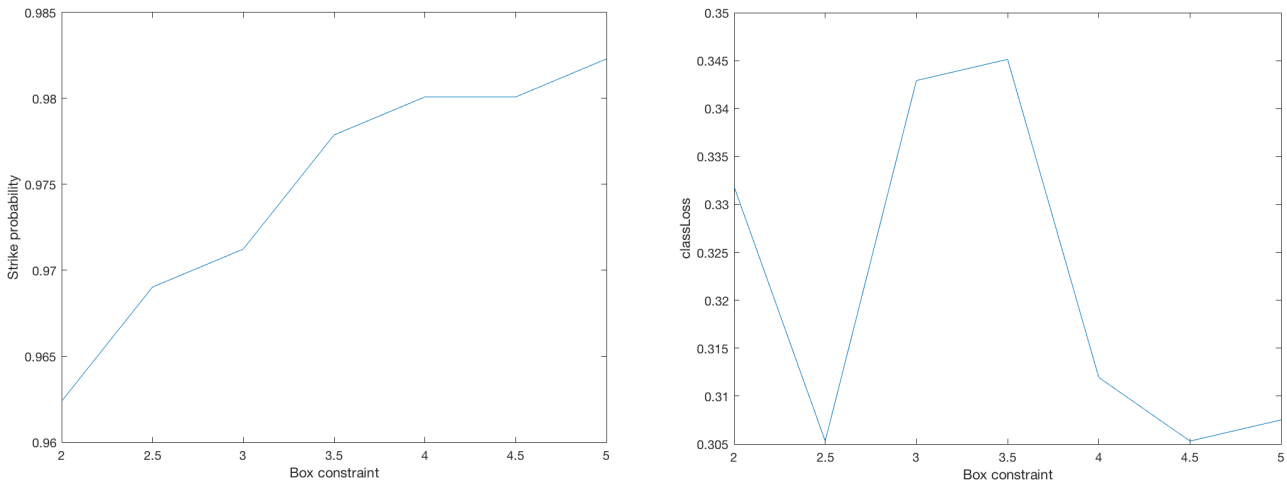

Figure 1: Trend of strike probability and misclassification probability (classLoss) with respect to box constraint value C. (SVM with linear kernel)

In this case the better value was C=2.5.

Then the Support Vector Machine has been trained with a Gaussian kernel instead of a linear kernel, that in theory should improve the classification performance as it has more freedom in defining a hyperplane that divides the dataset.
The experiment did not show any particular difference from the previous one, and it could be due to the fact that the dataset is well defined by itself.

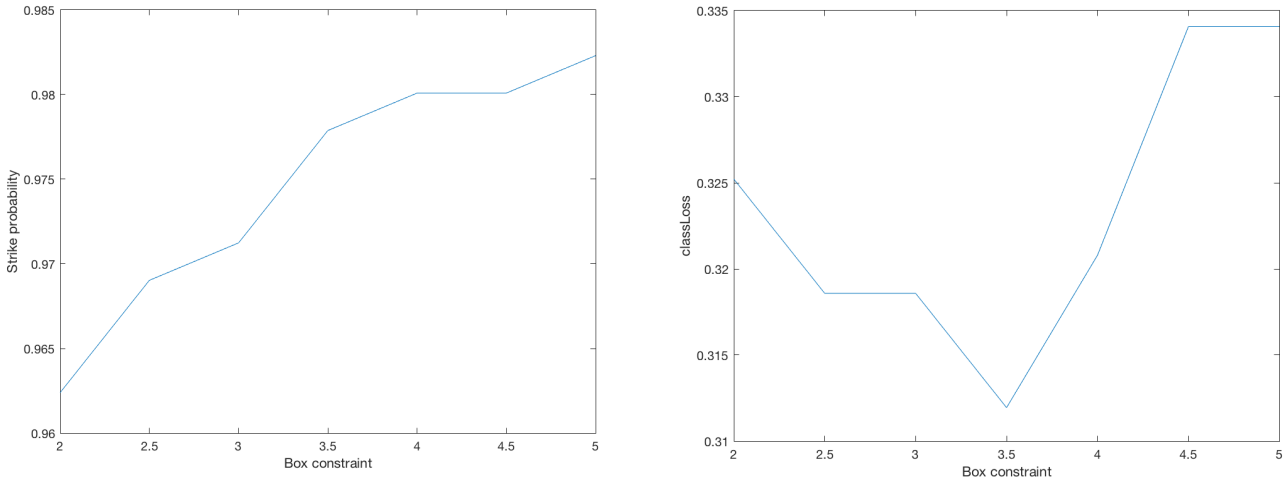

Figure 2: Trend of strike probability and misclassification probability (classLoss) with respect to box constraint value C. (SVM with Gaussian kernel)

Data preparation

```
clear all
close all
clc

load('arrhythmia.mat');

A=arrhythmia;

A(:, find(sum(abs(A)) == 0)) = []; % we erase the zero columns

class_id=A(:,end); % last vector of the matrix
class_id(find(class_id>1))=2; % all the values higher than 1 are put
 equal to 2
y=A;
y(:,end)=[]; % we put in y all the features but the last one
[N,F]=size(y);

%normalizing y
mean_y=mean(y,1);
stdv_y=std(y,1);

o=ones(N,1);% o is a column vector
y=(y-o*mean_y)./(o*stdv_y);% y is normalized

mean_y=mean(y,1); % checking that y matrix is properly normalized
var_y=var(y,1);

n_healthy=sum(class_id==1);
n_ill=sum(class_id==2);
```

SVM implementation

```
C = [2:0.5:5]

for ii=1:length(C)

    Mdl=fitcsvm(y,class_id,'BoxConstraint',C(ii));
    classhat=sign(y*Mdl.Beta+Mdl.Bias);

    n_false_negative=length(find(class_id(classhat==-1)==2));
    n_false_positive=length(find(class_id(classhat==1)==1));
    n_true_negative=length(find(class_id(classhat==-1)==1));
    n_true_positive=length(find(class_id(classhat==1)==2));

    p_true_positive=100*n_true_positive/n_ill; % 87.92
    p_true_negative=100*n_true_negative/n_healthy; % 93.87
    p_false_positive=100*n_false_positive/n_healthy; % 6.12
    p_false_negative=100*n_false_negative/n_ill; % 12.07

    p_strike(ii)=(n_true_positive+n_true_negative)/N % 91,15
```

```
    CVMdl = crossval(Mdl);
    classLoss(ii) = kfoldLoss(CVMdl);


end

figure
plot(C,p_strike)
xlabel('Box constraint');
ylabel('Strike probability');

figure
plot(C,classLoss)
xlabel('Box constraint');
ylabel('classLoss');

%
% %% visualizing support vectorz
%
% [COEFF, SCORE] = pca(y,'NumComponents',3);
% y_pca=SCORE*COEFF';
%
%
%
 Mdl=fitcsvm(y_pca,class_id,'BoxConstraint',3.0,'KernelFunction','gaussian');
%
% sv = Mdl.SupportVectors;
% figure
% gscatter(y_pca(:,1),y_pca(:,2),class_id)
% hold on
% plot(sv(:,1),sv(:,2),'ko','MarkerSize',10)
% legend('versicolor','virginica','Support Vector')
% hold off
%

% figure
% hold on
% b=bar(1,p_strike);
% b2=bar(2,p_true_positive,'r');
% b3=bar(3,p_true_negative,'g');
% b4=bar(4,p_false_positive,'y');
% b5=bar(5,p_false_negative,'m');
%
% title('Results')
%
 legend('pStrike','pTruePositive','pTrueNegative','pFalseePositive','pFalseNegativ
```

Published with MATLAB® R2016a