MACHINE LEARNING FROM DATA
Fall 2018

Report: Lab Session 4 – Support Vector Machines

Names: Santagiustina Francesco, Simonetto Adriano

Instructions

- Answer the questions
- Save the report and upload the file

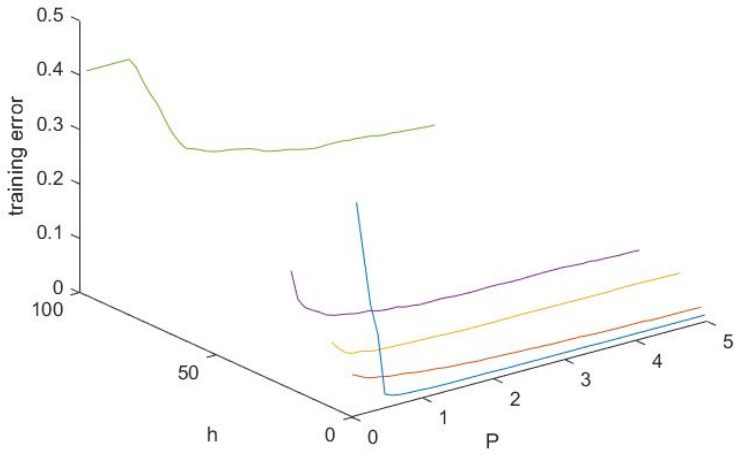
Questions

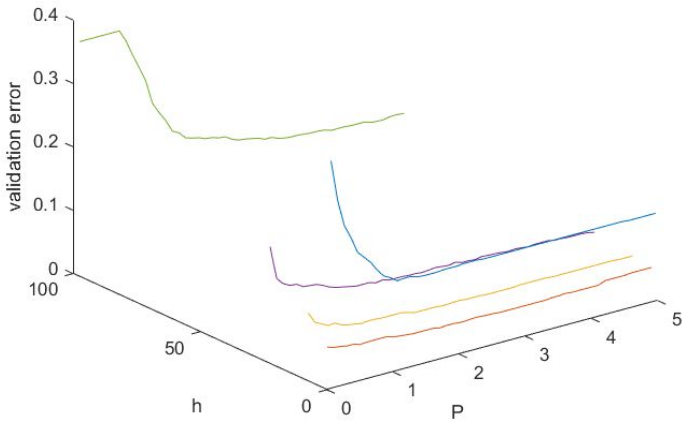
Q1: Complete a table with the training, validation and test errors for the linear and Gaussian SVMs. Which are the values of P and h?

P_e	Training set	Test set
Linear	0.0659	0.0728
Gaussian	0.3913	0.3489

The validation set has not been employed in this first part.
The values of P and h used in the code are P = 0.1, h = 1.

Q2: Plot the training error and the validation error for the Gaussian SVM as a function of h and P (two 3D plots).





Q3: Find the optimal values of P and h.
The optimal values are:
P = 1.5
H = 2.5

By running the code many time we noticed that we always get an optimal H value of 2.5 and the validation error is almost constant for H=2.5 and values of P superior to 1 so little fluctuations due to random partition of the dataset (into training, validation and test sets) can lead to different optimal P values (probably in a range of 1 to 5) . Anyway the hyperparameter selection has to be done considering only one execution, otherwise the independence of the model from the test set would be broken.

Q4: Copy the code.

```
%% ML
% LAB 4, BD: SPAM, Classifier: SVM
% April 2016, MC
clear
close all
clc

%% Loading SPAM Database
% load dataspam.txt -ascii
load dataspam
Labs=dataspam(:,end);
N_feat=size(dataspam,2)-1;
X=dataspam(:,1:57);
N_datos=length(Labs);


%% Binary quantization of features
X=X(:,1:54);
A=find(X>0);
```

```
X(A)=ones(size(A));

%% Generation of Training (60 %), Validation (20%) and Test (20%) sets
% Randomize vectors
indexperm=randperm(N_datos);
X=X(indexperm,:);
Labs=Labs(indexperm);

% Identify one vector to compute probabilities (section 4.3)
V_analisis=X(N_datos,:);
Lab_analisis=Labs(N_datos)
N_datos=N_datos-1;
X=X(1:N_datos,:);
Labs=Labs(1:N_datos);

% Generation of Train, Validation and Test sets
N_train=round(0.6*N_datos);
N_val=round(0.8*N_datos)-N_train;
N_test=N_datos-N_train-N_val;

% Train
X_train=X(1:N_train,:);
Labs_train=Labs(1:N_train);

% Validation
X_val=X(N_train+1:N_train+N_val,:);
Labs_val=Labs(N_train+1:N_train+N_val);

% Test
X_test=X(N_train+N_val+1:N_datos,:);
Labs_test=Labs(N_train+N_val+1:N_datos);

clear indexperm

%% Non-linear classifier, gaussian kernel
P1 = 0.1:0.1:5;
h1 = [1,2.5,10,25,100];
err_train = zeros(length(P1),length(h1));
err_val = zeros(length(P1),length(h1));

best_P = 0;
best_h = 0;
min_val = realmax;

for i = 1:length(P1)
    for j = 1:length(h1)
        P = P1(i);
        h=h1(j);
        Gauss_model = fitcsvm(X_train, Labs_train, 'BoxConstraint',P,...
            'KernelFunction','RBF','KernelScale',h);
        Gauss_out = predict(Gauss_model, X_train);
```

```
        err_train(i,j)=sum(Gauss_out~=Labs_train)/length(Labs_train);
        Gauss_out = predict(Gauss_model, X_val);
        err_val(i,j)=sum(Gauss_out~=Labs_val)/length(Labs_val);
        % Test confusion matrix
        %CM_Gauss_val=confusionmat(Labs_val,Gauss_out)
        if err_val(i,j)<min_val
           min_val = err_val(i,j);
           best_P = P1(i);
           best_h = h1(j);
        end
    end
end

P2 = repmat(P1,length(h1),1);
h2 = repmat(h1',1,length(P1));
figure
plot3(P2',h2',err_train)
xlabel('P');
ylabel('h');
zlabel('training error')

figure
plot3(P2',h2',err_val)
xlabel('P');
ylabel('h');
zlabel('validation error')
```

Q5: For the best classifier found in the previous step, compute classification error on the test set, compare with the error obtained for the non-optimized Gaussian classifier (Q1).

The classification error for the test set is 0.0554. As we can see the validation step majorly improved the performance of our classifier (the starting error was 0.35, almost seven times larger)

Q6: Compute the predictions for the test dataset. Compute the confusion matrix for the test set.

The classifier identifies 585 mails and 335 spam. Respectively, 556 mails and 312 spams are correctly classified.

$$CM = \begin{bmatrix} 556 & 23 \\ 29 & 312 \end{bmatrix}$$

Q7: Compute the six metrics (*error*, *accuracy*, *precision*, *recall*, *specificity* and *f-score*). Include the code.

Error = 0.0565
Accuracy = 0.9435
Precision = 0.9313
Recall = 0.9150
Specificity = 0.9603

f-score = 0.9231

```
%% Metrics of interest
CM = CM_Gauss_test;
error = (CM(1,2)+CM(2,1))/(CM(1,1)+CM(1,2)+CM(2,1)+CM(2,2));
accuracy = (CM(1,1)+CM(2,2))/(CM(1,1)+CM(1,2)+CM(2,1)+CM(2,2));
precision = CM(2,2)/(CM(2,2)+CM(1,2));
recall = CM(2,2)/(CM(2,2)+CM(2,1));
specificity = CM(1,1)/(CM(1,1)+CM(1,2));
Fscore = 2*precision*recall/(precision+recall);
```

Q8: Using as example a dataset containing 400 SPAM and 4600 MAIL samples, explain why *precision*, *recall*, *specificity* and *f-score* are more appropriate than *accuracy* and *error* for evaluating the classifier performance.

Accuracy and error are metrics that do not make any differentiation between the two classes, expecting some sort of symmetry between the two. This is not an issue if the error rates of the two different classes are more or less the same, but in case of a major difference, error and accuracy may give an evaluation of the classifier that does not reflect some important underlying factors.
As an example let's suppose that all the 4600 MAIL are correctly identified, while half of the 400 SPAM end up in the wrong class. In this framework, the error and the accuracy of our classifier are respectively 0.04 and 0.96, which seem to suggest a pretty good classifier. In practice however, the classifier has a really good performance only for one of the two classes, while for the other proceeds at random and this cannot be detected by the two metrics due to the asymmetry of the problem and due to the size disparity between the two classes.
On the other hand the other metrics clearly show this problem as the precision is 1 (all the elements classified as SPAM are actually SPAM), the recall is 0.5 (it's actually showing the critical behavior that accuracy and error could not catch, half of the SPAM is actually wrongly classified), specificity is 0.958 and f-score is 0.33 (which again shows the issue).

Q9: Compute the prior probabilities:

$$P(\text{class=SPAM}) = \frac{\#\text{SPAM samples in the test set}}{\#\text{samples in the test set}}$$

$$P(class = SPAM) = 0.3707$$

$$P(\text{class=MAIL}) = \frac{\#\text{MAIL samples in the test set}}{\#\text{samples in the test set}}$$

$$P(class = MAIL) = 0.6293$$

Q10: Classify the vector *V_analysis*. Is it correctly classified? Does the corresponding email contain the word 'make'? Does it contain the word 'address'?

V_analysis is correctly classified as SPAM. The corresponding email contains the word “address” but not the word “make”.

Q11: Analyze the reliability of the decision. That is, if the vector *V_analysis* is classified as SPAM, compute the probability that it is really a SPAM vector:

$$P(\text{V\_analysis is SPAM} \mid \text{classified as SPAM})$$

From the Bayes formula we know that:

$$P(V\ is\ SPAM|classified\ as\ SPAM) = P(classified\ as\ SPAM|V\ is\ SPAM)\ \frac{P(V\ is\ SPAM)}{P(V\ classified\ as\ SPAM)}$$

$$= recall\ \times\ \frac{P(class = SPAM)}{P(V\ classified\ as\ SPAM)} = 0.9313\ .$$

Use the test dataset and the results provided by the classifier at your convenience in order to compute these probabilities (P(class = SPAM) or P(class =MAIL)), and explain the procedure.

Q12: Copy the code used to compute the probabilities in Q11.

```
Prior_SPAM = (CM(2,1)+CM(2,2))/(CM(1,1)+CM(1,2)+CM(2,1)+CM(2,2))
P_classifyAsSPAM = (CM(1,2)+CM(2,2))/(CM(1,1)+CM(1,2)+CM(2,1)+CM(2,2));
P_really_SPAM = recall*Prior_SPAM/P_classifyAsSPAM
```