Table of Contents

loading data

```
%loading the features we have computed
load features;
%loading data sets
load dataset_BCIcomp1;
```

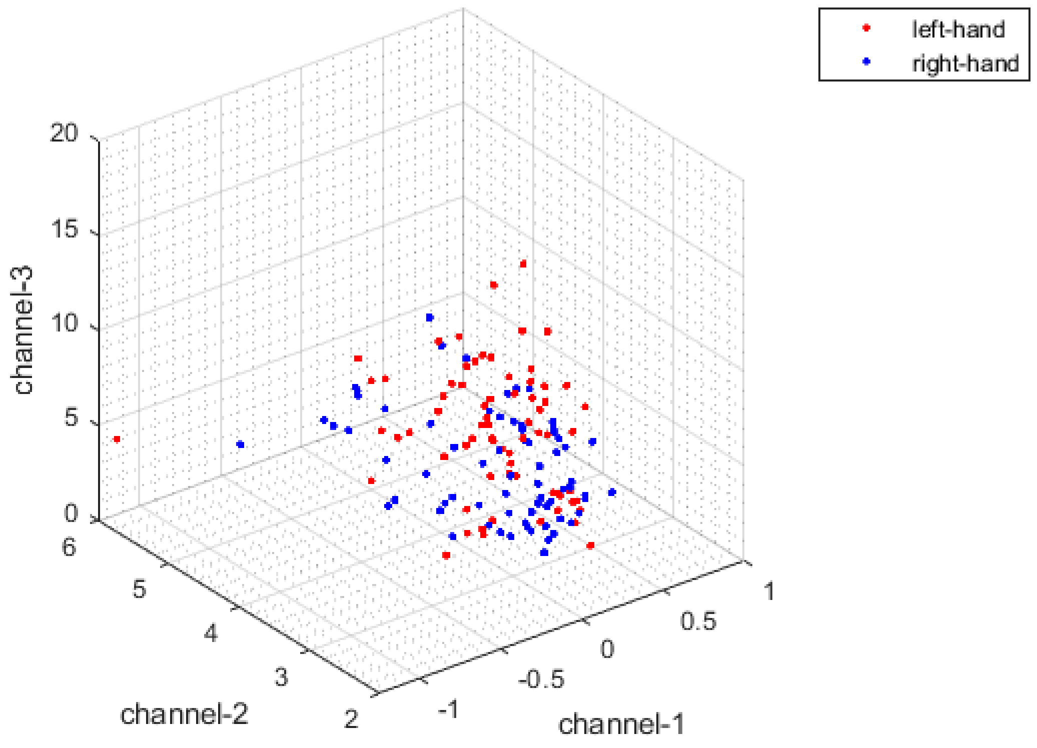
defining the parameters

```
fs=128;
n_features=10; %number of PCA features
n_trials=size(x_train,3); %140
n_channels=size(x_train,2); %3
n=size(x_train,1); %1152
kernel_size=256; %as mentioned in the essay
s_point=n - kernel_size + 1; %897, which also describes number of
 different sets of features we can have
```

showing labels for a specific start point

```
%features when the start point is j and we choose the i'th feature
j=431;%start point
i=2;%number of the feature
sample1=features(i:n_features,:,j);
%finding the datas with 1 lable
leftindex1=find(y_train==1);
%finding the datas with 2 lable
rightindex1=find(y_train==2);
plot3(sample1(1,leftindex1),sample1(2,leftindex1),...
    sample1(3,leftindex1),'.','color','r');
xlabel('channel-1');
ylabel('channel-2');
zlabel('channel-3');
hold on
plot3(sample1(1,rightindex1),sample1(2,rightindex1),...
    sample1(3,rightindex1),'.','color','b');
```

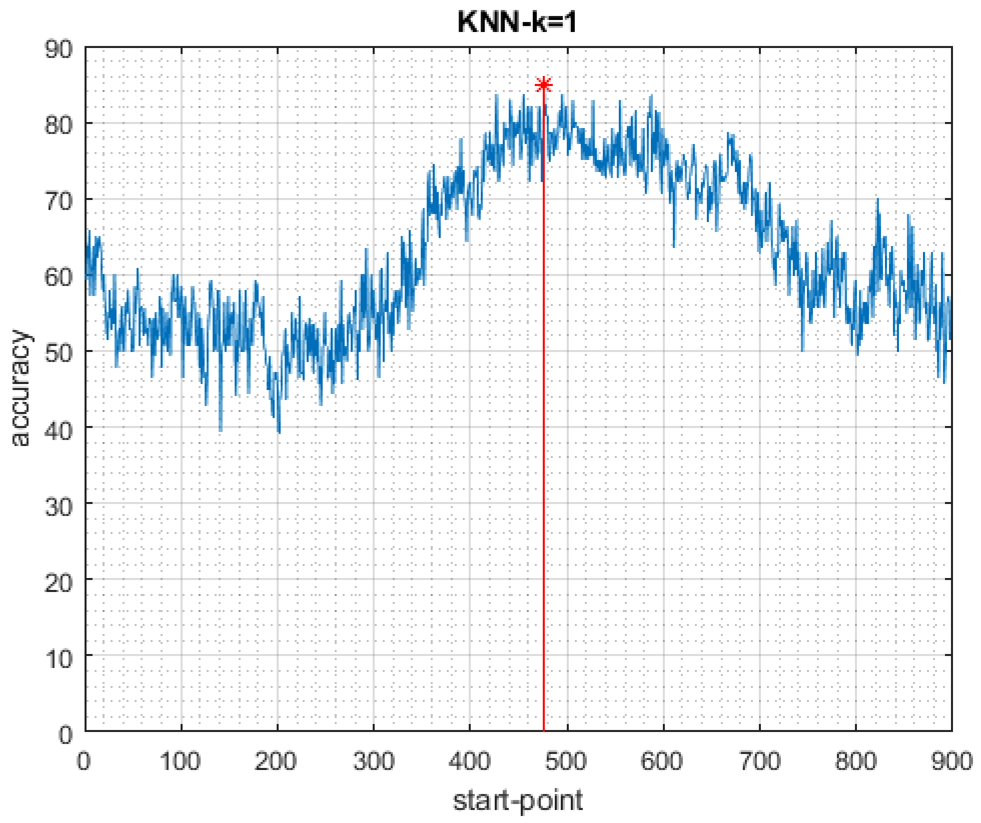
```
legend('left-hand','right-hand');
grid on
grid minor
hold off
```

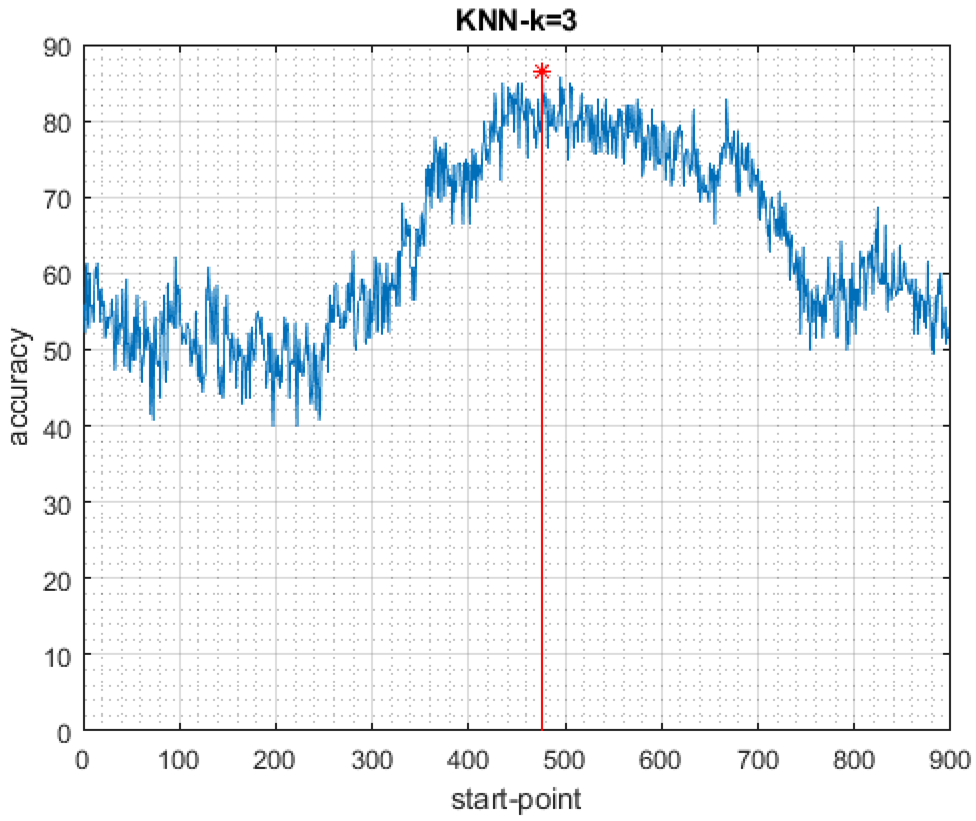


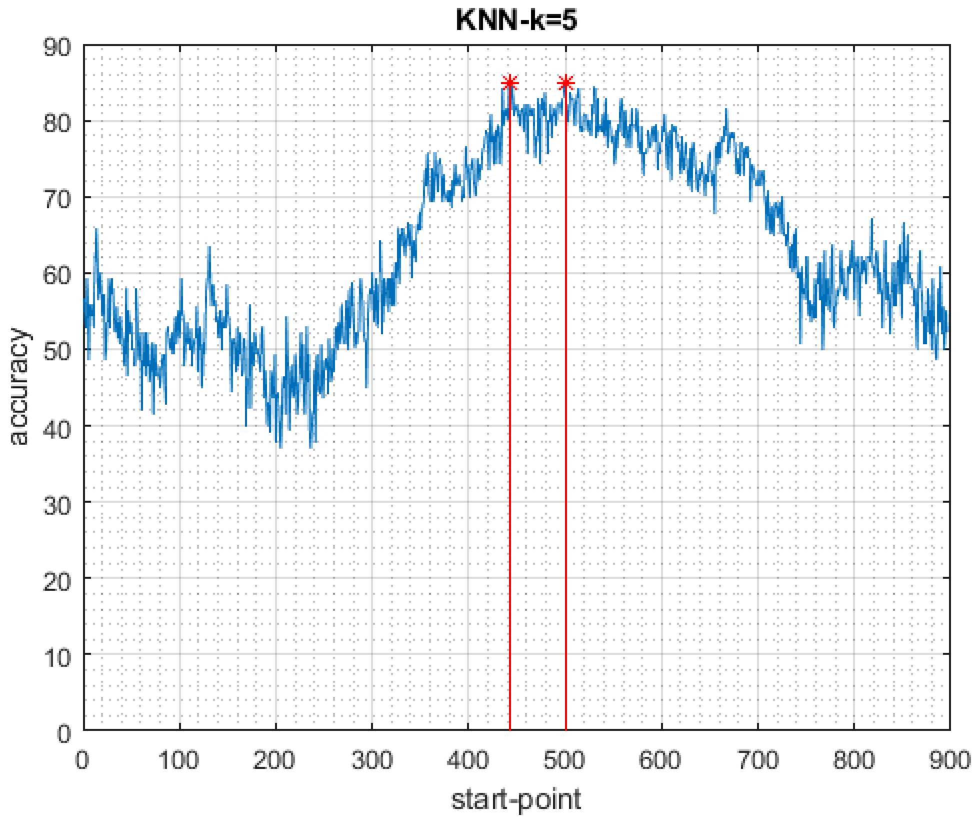
KNN

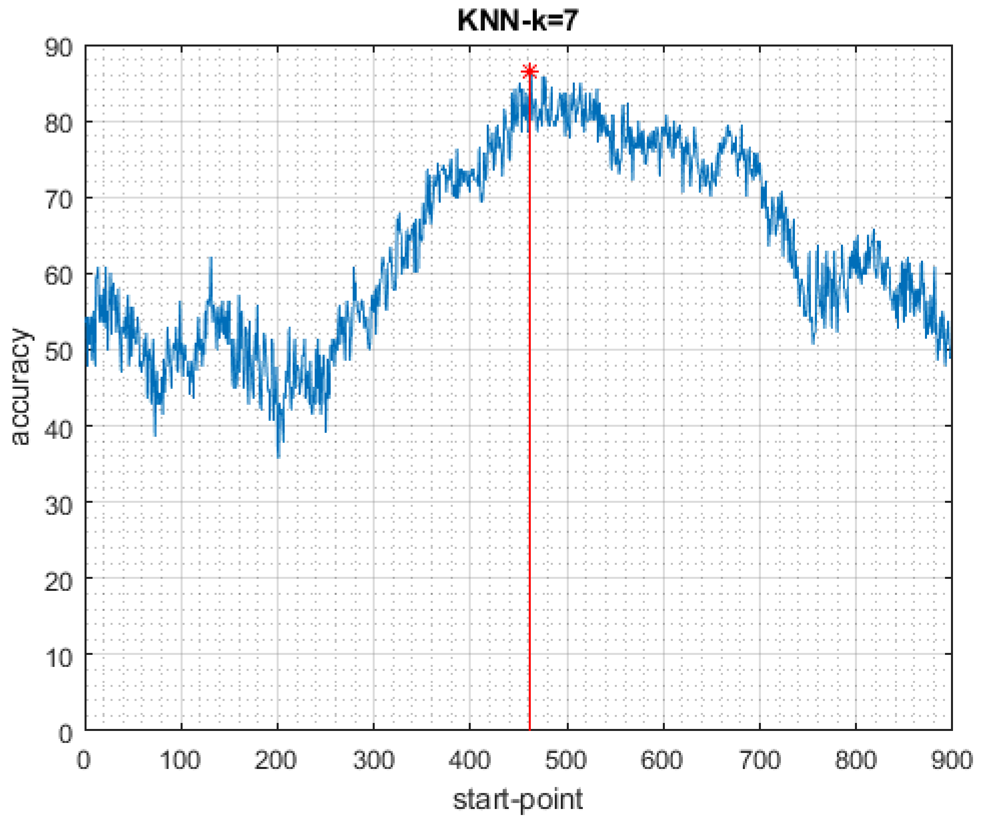
```
tp_accuracy=zeros(1,n_trials);
accuracy=zeros(1,s_point);
for k=[1 3 5 7]
    figure;
    for i=1:s_point
        %the dataset with the "i"th start point
        sample=features(:,:,i);
        % dividing data into test and split with the function we
 defined
        [Xs_train,Xs_test,ys_train,ys_test] =
 leave_one_out(sample,y_train);
        for g=1:n_trials
            sampleX_train=Xs_train(:,:,g);
            sampley_train=ys_train(:,g);
            %training the model for KNN with k neighbours

 model=fitcknn(sampleX_train',sampley_train,'NumNeighbors',k);
            %testing the model
            sampleX_test=Xs_test(:,:,g);
```

```
                op=predict(model,sampleX_test');
                %computing the accuracy using leave one out method
                sampley_test=ys_test(g,:);
                tp_accuracy(g)=sum(op==sampley_test)/length(sampley_test)
 *100;
            end
            accuracy(i)=sum(tp_accuracy)/length(tp_accuracy);
        end
        best_point=find(max(accuracy)==accuracy);
        plot(accuracy);
        hold on
        stem(best_point,accuracy(best_point),'*','r');
        title(['KNN-k=',num2str(k)]);
        xlabel("start-point");
        ylabel("accuracy");
        grid on;
        grid minor;
        hold off;
    end
```




KNN-k=3
accuracy
start-point
0
10
20
30
40
50
60
70
80
90
0
100
200
300
400
500
600
700
800
900

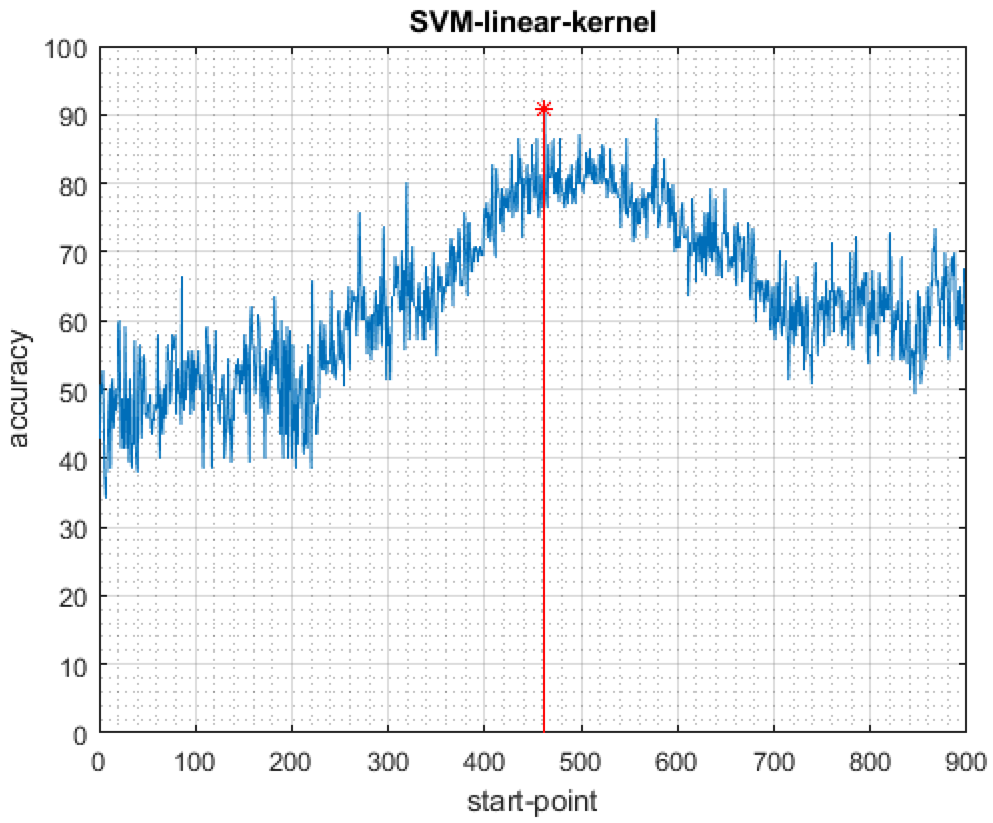

KNN-k=5
accuracy
start-point
0
10
20
30
40
50
60
70
80
90
0
100
200
300
400
500
600
700
800
900



SVM

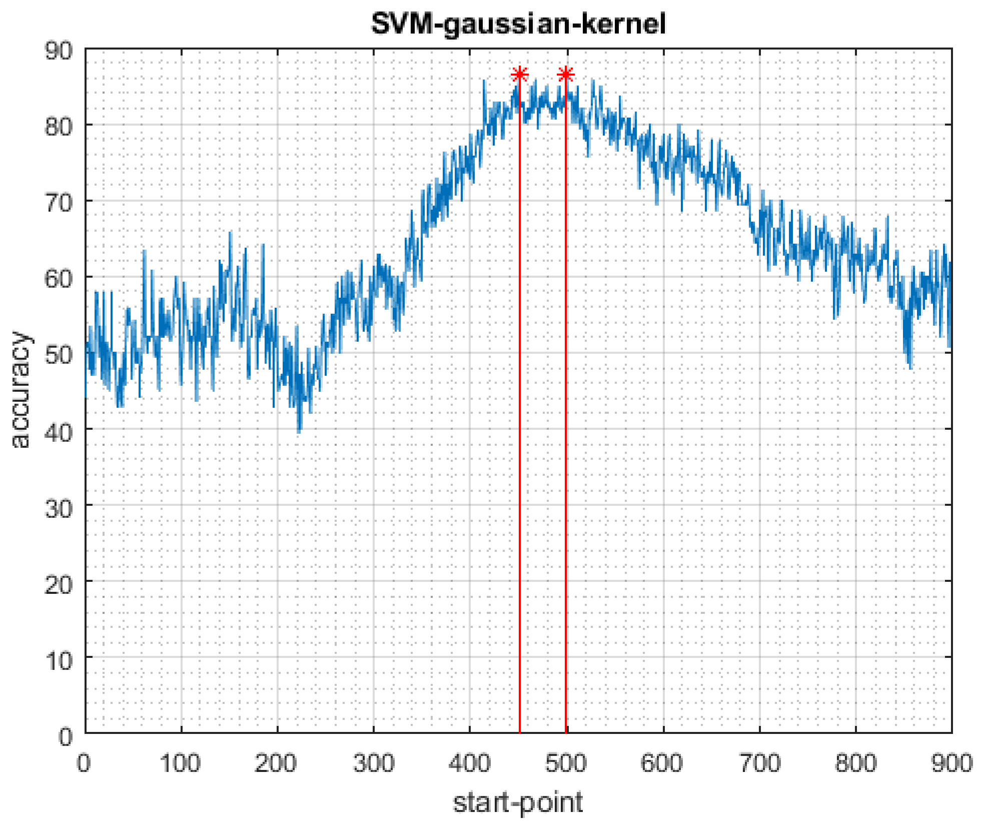
linear

```
accuracy=zeros(1,s_point);
for i=1:s_point
    %the dataset with the "i"th start point
    sample=features(:,:,i);
    % dividing data into test and split with the function we defined
    [Xs_train,Xs_test,ys_train,ys_test] =
 leave_one_out(sample,y_train);
    for g=1:n_trials
        sampleX_train=Xs_train(:,:,g);
        sampley_train=ys_train(:,g);
        %training the model for KNN with k neighbours
        model=fitcsvm(sampleX_train',sampley_train,'Standardize',1,...
        'KernelFunction','linear');
        %testing the model
        sampleX_test=Xs_test(:,:,g);
        op=predict(model,sampleX_test');
        %computing the accuracy using leave one out method
        sampley_test=ys_test(g,:);
        tp_accuracy(g)=sum(op==sampley_test)/length(sampley_test)
 *100;
    end
    accuracy(i)=sum(tp_accuracy)/length(tp_accuracy);
end
best_point=find(max(accuracy)==accuracy);
figure;
plot(accuracy);
hold on
stem(best_point,accuracy(best_point),'*','r');
title('SVM-linear-kernel');
xlabel("start-point");
ylabel("accuracy");
grid on;
grid minor;
hold off;
```



gaussian

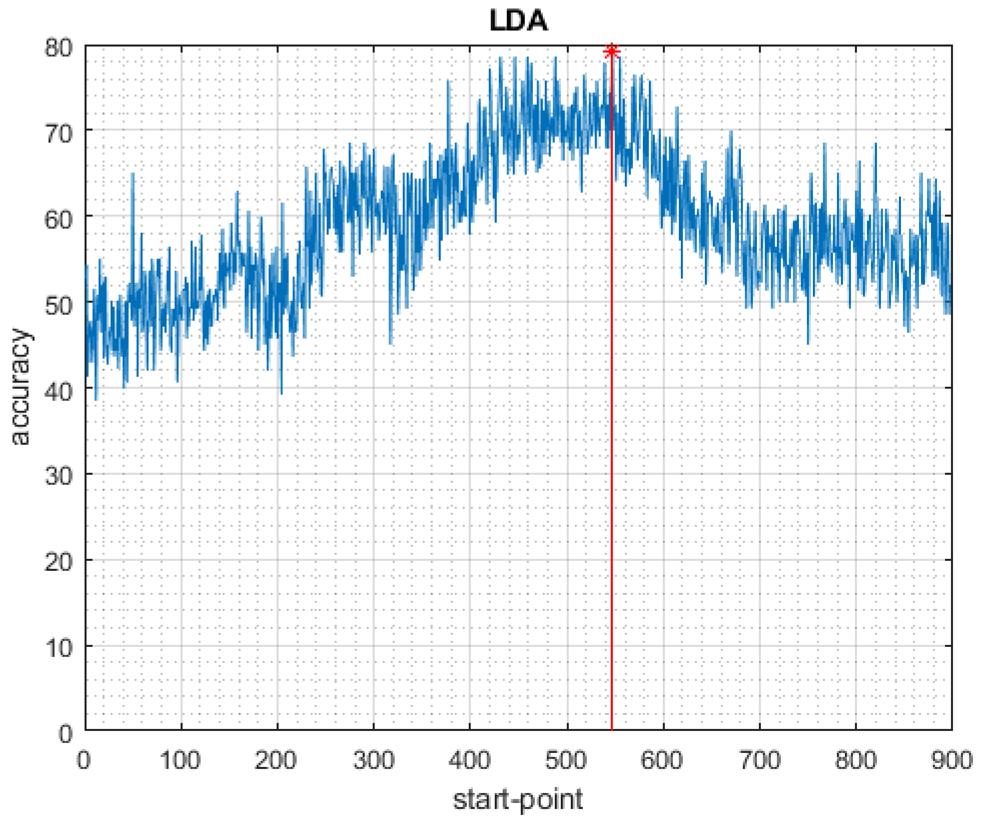
```
accuracy=zeros(1,s_point);
for i=1:s_point
    %the dataset with the "i"th start point
    sample=features(:,:,i);
    % dividing data into test and split with the function we defined
    [Xs_train,Xs_test,ys_train,ys_test] =
 leave_one_out(sample,y_train);
    for g=1:n_trials
        sampleX_train=Xs_train(:,:,g);
        sampley_train=ys_train(:,g);
        %training the model for KNN with k neighbours
        model=fitcsvm(sampleX_train',sampley_train,'Standardize',1,...
        'KernelFunction','RBF','KernelScale','auto');
        %testing the model
        sampleX_test=Xs_test(:,:,g);
        op=predict(model,sampleX_test');
        %computing the accuracy using leave one out method
        sampley_test=ys_test(g,:);
        tp_accuracy(g)=sum(op==sampley_test)/length(sampley_test)
 *100;
    end
    accuracy(i)=sum(tp_accuracy)/length(tp_accuracy);
end
best_point=find(max(accuracy)==accuracy);
figure;
plot(accuracy);
hold on
stem(best_point,accuracy(best_point),'*','r');
title('SVM-gaussian-kernel');
xlabel("start-point");
ylabel("accuracy");
grid on;
grid minor;
hold off;
```



LDA

```
accuracy=zeros(1,s_point);
for i=1:s_point
    %the dataset with the "i"th start point
    sample=features(:,:,i);
    % dividing data into test and split with the function we defined
    [Xs_train,Xs_test,ys_train,ys_test] =
 leave_one_out(sample,y_train);
    for g=1:n_trials
        sampleX_train=Xs_train(:,:,g);
        sampley_train=ys_train(:,g);
        %training the model for KNN with k neighbours
        model=fitcdiscr(sampleX_train',sampley_train);
        %testing the model
        sampleX_test=Xs_test(:,:,g);
        op=predict(model,sampleX_test');
        %computing the accuracy using leave one out method
        sampley_test=ys_test(g,:);
        tp_accuracy(g)=sum(op==sampley_test)/length(sampley_test)
 *100;
    end
    accuracy(i)=sum(tp_accuracy)/length(tp_accuracy) ;
end
best_point=find(max(accuracy)==accuracy);
```

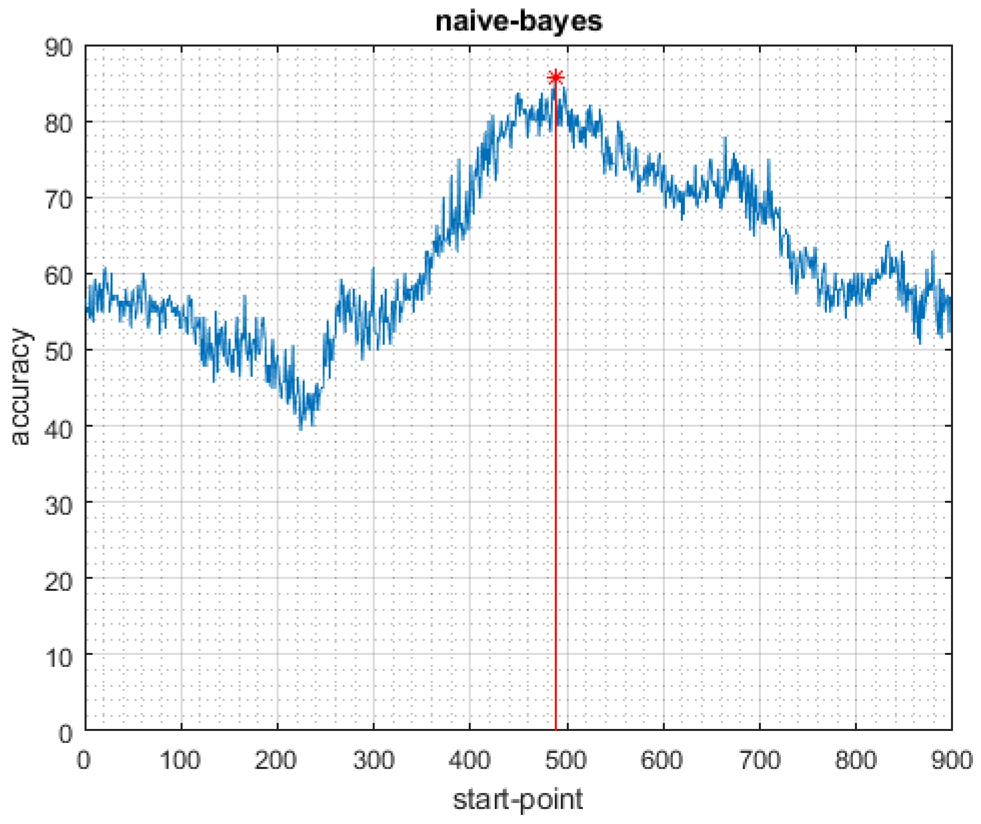
```
figure;
plot(accuracy);
hold on
stem(best_point,accuracy(best_point),'*','r');
title('LDA');
xlabel("start-point");
ylabel("accuracy");
grid on;
grid minor;
hold off
```



Naive bayes

```
accuracy=zeros(1,s_point);
for i=1:s_point
    %the dataset with the "i"th start point
    sample=features(:,:,i);
    % dividing data into test and split with the function we defined
    [Xs_train,Xs_test,ys_train,ys_test] =
 leave_one_out(sample,y_train);
    for g=1:n_trials
        sampleX_train=Xs_train(:,:,g);
        sampley_train=ys_train(:,g);
        %training the model for KNN with k neighbours
        model=fitcnb(sampleX_train',sampley_train);
        %testing the model
        sampleX_test=Xs_test(:,:,g);
        op=predict(model,sampleX_test');
        %computing the accuracy using leave one out method
        sampley_test=ys_test(g,:);
        tp_accuracy(g)=sum(op==sampley_test)/length(sampley_test)
 *100;
    end
    accuracy(i)=sum(tp_accuracy)/length(tp_accuracy);
end
best_point=find(max(accuracy)==accuracy);
```

```
figure;
plot(accuracy);
hold on
stem(best_point,accuracy(best_point),'*','r');
title('naive-bayes');
xlabel("start-point");
ylabel("accuracy");
grid on;
grid minor;
hold off
```



Published with MATLAB® R2020a