EE583 Pattern Recognition HW3

Kutay Uğurlu 2232841

November 28, 2021

1 Question 1

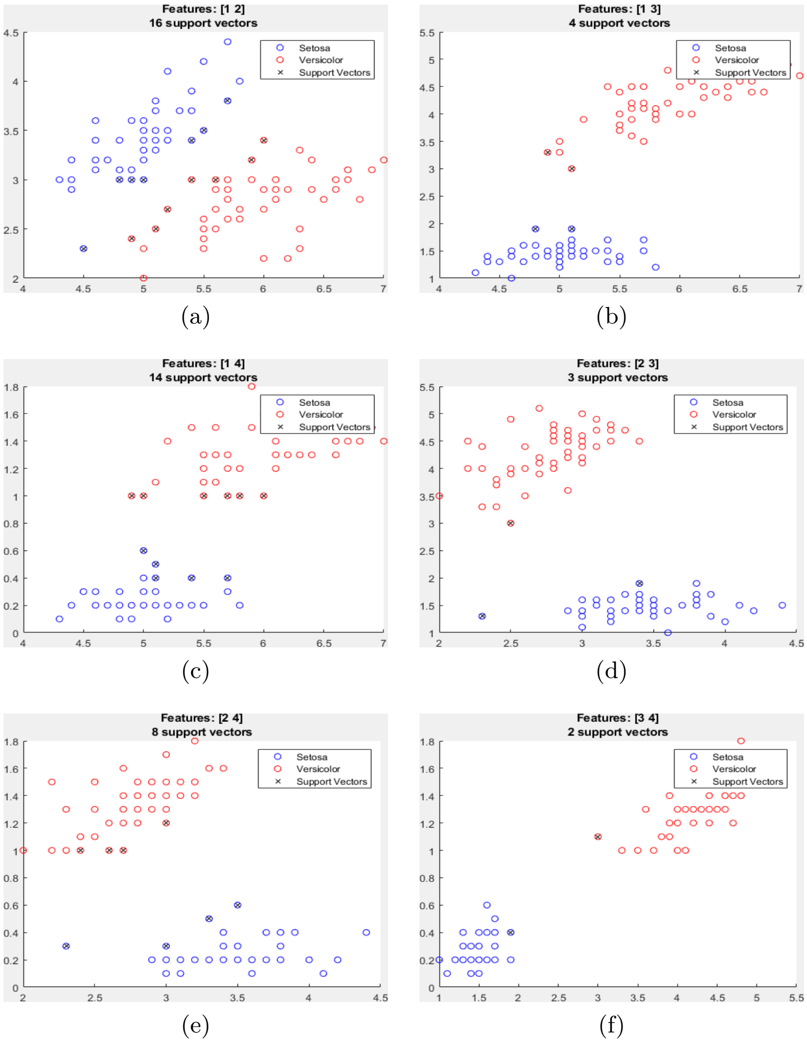


Figure 1: Support Vectors for different feature pairs

The number of support vectors was minimum for the features 3 & 4. One can notice the opposite relationship between the linear separability of the feature space and the number of support vectors. However, it is important to note that one can achieve the same number of minimum vectors for other features by setting the BoxConstraint parameter manually. I observed that setting it to 15 resulted in 2 support vectors for two cases.

2 Question 2

4 Features

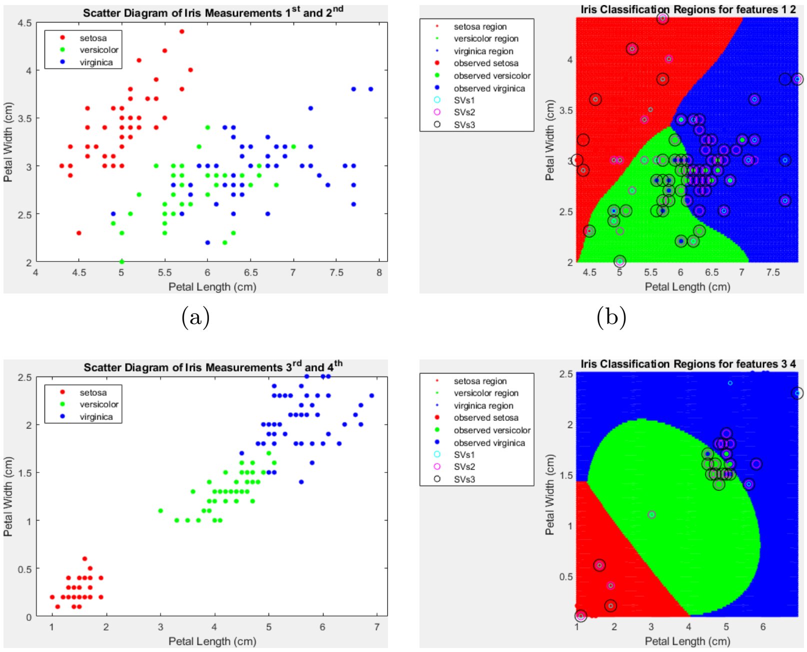
$kfold\ loss = 0$ $LOOCV\ loss = 0$

1 Feature

$kfold\ loss = 0.1900$ $LOOCV\ loss = 0.1700$

Using four features, having higher dimensions, have resulted in SVM Model linearly separating the feature space. Hence, for both Cross Validation Experiments, the loss rate was 0. On the other hand, when I decreased the number of utilized dimensions to 1, LOOCV loss was slightly less than the 10-Fold CV loss. This may be attributed to the fact that LOOCV utilizes more training data than kfold, resulting in higher more accurate test set predictions, considering that the number of training data is not so large that it causes overfitting.

3 Question 3



(a) (b) (c) (d)

Figure 2: Support Vectors for different feature pairs

Changing feature pairs that are utilized in SVM classification resulted in different distributions. Features 3 and 4 provides a more linearly separable feature space than 1 and 2. The resultant numbers of support vectors 196 and 58 also indicates the same result. In addition, this is also observable in scatter plots.

4 Question 4

The objective function, as a function of BoxConstraint and KernelScale, is optimized using Bayesian optimization. Hence, these parameters determines the cross validation loss and selecting them is a crucial part of optimizing an SVM classifier, keeping in mind that rbf as kernel function is fixed. In addition, I have tried polynomial kernel function, and observed that optimized cross validation loss is different from the one that rbf kernel function is able to achieve. Therefore,

- Box Constraint
- Kernel Scale
- Kernel Function

are the hyperparameters that can be optimized in an SVM to achieve better cross validation success, in terms of loss.

5 APPENDIX

The code given in this section is shared @.

5.1 Q1

```
%% HW3_Q1
load fisheriris
inds = ~strcmp(species,'virginica');

features = 1:4;
all_feature_pairs = nchoosek(features,2);
counter = zeros(size(all_feature_pairs,1),1);

for i = 1:size(all_feature_pairs,1)
    temp_features = all_feature_pairs(i,:);
    X = meas(inds,temp_features);
    y = species(inds);
    setosa_idx = strcmp(y,'setosa');
    versicolor_idx = strcmp(y,'versicolor');
    SVMModel = fitcsvm(X,y);
    % n_support_vectors = sum(SVMModel.IsSupportVector);
    n_support_vectors = size(SVMModel.SupportVectors,1);
    idx = SVMModel.IsSupportVector;
    figure
    scatter(X(setosa_idx,1),X(setosa_idx,2),'bo')
    hold on
    scatter(X(versicolor_idx,1),X(versicolor_idx,2),'ro')
    hold on
    scatter(X(idx,1),X(idx,2),'kx')
    title({['Features: ',mat2str(temp_features)],[num2str(n_support_vectors),' support vectors']})
    counter(i) = n_support_vectors;
    legend('Setosa','Versicolor','Support Vectors')
end

min_support_vectors = min(counter);
min_SV_features = all_feature_pairs(find(counter == min_support_vectors),:);


%% Save all
chdir('..')
addpath('export_fig')
chdir('HW3')
```

```
figHandles = findall(0,'Type','figure');

for i = 1:numel(figHandles)
    export_fig(['Q1_',num2str(i)], '-png', figHandles(i), '-append')
end

close all
```

5.2 Q2

```
%% HW2 Q2
%% All features
load fisheriris
inds = ~strcmp(species,'virginica');
features = 1:4;
X = meas(inds,features);
Y = species(inds);
SVMModel = fitcsvm(X,Y,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');

CVSVMModel = crossval(SVMModel,'Kfold',10,'Leaveout','off');
kfold_loss = kfoldLoss(CVSVMModel)

SVMModel = fitcsvm(X,Y,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');

CVSVMModel_xval = crossval(SVMModel,'Leaveout','on');
leave_one_out_loss = kfoldLoss(CVSVMModel_xval)

%% Just 2nd Feature
inds = ~strcmp(species,'virginica');
features = 2;
X = meas(inds,features);
Y = species(inds);
SVMModel = fitcsvm(X,Y,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');

CVSVMModel = crossval(SVMModel,'Kfold',10,'Leaveout','off');
kfold_loss = kfoldLoss(CVSVMModel)

SVMModel = fitcsvm(X,Y,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');

CVSVMModel_xval = crossval(SVMModel,'Leaveout','on');
leave_one_out_loss = kfoldLoss(CVSVMModel_xval)
```

5.3 Q3

```matlab
%% Same features

load fisheriris
X = meas(:,1:2);
Y = species;

figure('Position',[250 250 600 400]);
gscatter(X(:,1),X(:,2),Y);
h = gca;
lims = [h.XLim h.YLim]; % Extract the x and y axis limits
title('{\bf Scatter Diagram of Iris Measurements 1^{st} and 2^{nd}}');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');
legend('Location','Northwest');

SVMModels = cell(3,1);
classes = unique(Y);
rng(1); % For reproducibility

for j = 1:numel(classes)
    indx = strcmp(Y,classes(j)); % Create binary classes for each classifier
    SVMModels{j} = fitcsvm(X,indx,'ClassNames',[false true],'Standardize',true,...
        'KernelFunction','rbf','BoxConstraint',1);
end

d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
    min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
N = size(xGrid,1);
Scores = zeros(N,numel(classes));

for j = 1:numel(classes)
    [~,score] = predict(SVMModels{j},xGrid);
    Scores(:,j) = score(:,2); % Second column contains positive-class scores
end

[~,maxScore] = max(Scores,[],2);

```

```
figure('Position',[250 250 600 400]);
h(1:3) = gscatter(xGrid(:,1),xGrid(:,2),maxScore,...
    [1 0 0; 0 1 0; 0 0 1]);
hold on
h(4:6) = gscatter(X(:,1),X(:,2),Y);

title('{\bf Iris Classification Regions for features 1 2}');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');

hold on

colors = [0 1 1; 1 0 1; 0 0 0];
sizes = [15,60,150];
for j = 1:3
SVM_SV = X(SVMModels{j}.IsSupportVector,:);
h(j+6) = scatter(SVM_SV(:,1),SVM_SV(:,2),'CData',repmat(colors(j,:),size(SVM_SV,1),1),'SizeData',sizes(j)*ones(size(SVM_SV,1),1));
hold on
sum(SVMModels{j}.IsSupportVector)
end

legend(h,{'setosa region','versicolor region','virginica region',...
    'observed setosa','observed versicolor','observed virginica',...
    'SVs1','SVs2','SVs3'},...
    'Location','NorthwestOutside')

axis tight
hold on

%% Other features

load fisheriris
X = meas(:,3:4);
Y = species;

figure('Position',[250 250 600 400]);
gscatter(X(:,1),X(:,2),Y);
h = gca;
lims = [h.XLim h.YLim]; % Extract the x and y axis limits
title('{\bf Scatter Diagram of Iris Measurements 3^{rd} and 4^{th}}');
```

```
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');
legend('Location','Northwest');

SVMModels = cell(3,1);
classes = unique(Y);
rng(1); % For reproducibility

for j = 1:numel(classes)
    indx = strcmp(Y,classes(j)); % Create binary classes for each classifier
    SVMModels{j} = fitcsvm(X,indx,'ClassNames',[false true],'Standardize',true,...
        'KernelFunction','rbf','BoxConstraint',1);
    sum(SVMModels{j}.IsSupportVector)
end

d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
    min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
N = size(xGrid,1);
Scores = zeros(N,numel(classes));

for j = 1:numel(classes)
    [~,score] = predict(SVMModels{j},xGrid);
    Scores(:,j) = score(:,2); % Second column contains positive-class scores
end

[~,maxScore] = max(Scores,[],2);

figure('Position',[250 250 600 400]);

h(1:3) = gscatter(xGrid(:,1),xGrid(:,2),maxScore,...
    [1 0 0; 0 1 0; 0 0 1]);
hold on
h(4:6) = gscatter(X(:,1),X(:,2),Y);

title('{\bf Iris Classification Regions for features 3 4}');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');

colors = [0 1 1; 1 0 1; 0 0 0];
sizes = [15,60,150];
```

```
for j = 1:3
SVM_SV = X(SVMModels{j}.IsSupportVector,:);
h(j+6) = scatter(SVM_SV(:,1),SVM_SV(:,2),'CData',repmat(colors(j,:),size(SVM_SV,1),1),'SizeData',sizes(j)*ones(size(SVM_SV,1),1));
hold on
end

legend(h,{'setosa region','versicolor region','virginica region',...
    'observed setosa','observed versicolor','observed virginica',...
    'SVs1','SVs2','SVs3'},...
    'Location','NorthwestOutside')

axis tight

%% Save all
chdir('..')
addpath('export_fig')
chdir('HW3')
figHandles = findall(0,'Type','figure');

for i = 1:numel(figHandles)
    export_fig(['Q3_',num2str(i)], '-png', figHandles(i), '-append')
end

close all
```

5.4 Q4

```
rng default % For reproducibility
grnpop = mvnrnd([1,0],eye(2),10);
redpop = mvnrnd([0,1],eye(2),10);

plot(grnpop(:,1),grnpop(:,2),'go')
hold on
plot(redpop(:,1),redpop(:,2),'ro')
hold off

redpts = zeros(100,2);grnpts = redpts;
for i = 1:100
    grnpts(i,:) = mvnrnd(grnpop(randi(10),:),eye(2)*0.02);
    redpts(i,:) = mvnrnd(redpop(randi(10),:),eye(2)*0.02);
end

figure
plot(grnpts(:,1),grnpts(:,2),'go')
hold on
plot(redpts(:,1),redpts(:,2),'ro')
hold off

cdata = [grnpts;redpts];
grp = ones(200,1);
% Green label 1, red label -1
grp(101:200) = -1;

c = cvpartition(200,'KFold',10);


opts = struct('Optimizer','bayesopt','ShowPlots',true,'CVPartition',c,...
    'AcquisitionFunctionName','expected-improvement-plus');
svmmod = fitcsvm(cdata,grp,'KernelFunction','rbf',...
    'OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',opts)

lossnew = kfoldLoss(fitcsvm(cdata,grp,'CVPartition',c,'KernelFunction','rbf',...
    'BoxConstraint',svmmod.HyperparameterOptimizationResults.XAtMinObjective.BoxConstraint,...
    'KernelScale',svmmod.HyperparameterOptimizationResults.XAtMinObjective.KernelScale))
```

```
d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(cdata(:,1)):d:max(cdata(:,1)),...
    min(cdata(:,2)):d:max(cdata(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(svmmod,xGrid);
figure;
h = nan(3,1); % Preallocation
h(1:2) = gscatter(cdata(:,1),cdata(:,2),grp,'rg','+*');
hold on
h(3) = plot(cdata(svmmod.IsSupportVector,1),...
    cdata(svmmod.IsSupportVector,2),'ko');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 0],'k'
    );
legend(h,{'-1','+1','Support Vectors'},'Location','Southeast');
axis equal
hold off
```