Ömer Mesud TOKER
21302479
CS 464 – 2
HW 2 – Report

1 - a)

If we split the data with a boundary parallel to X1 axis, information gain will be 0 since parent's and children's entropies will be 1. So we split the data with a boundary parallel to X2 axis.

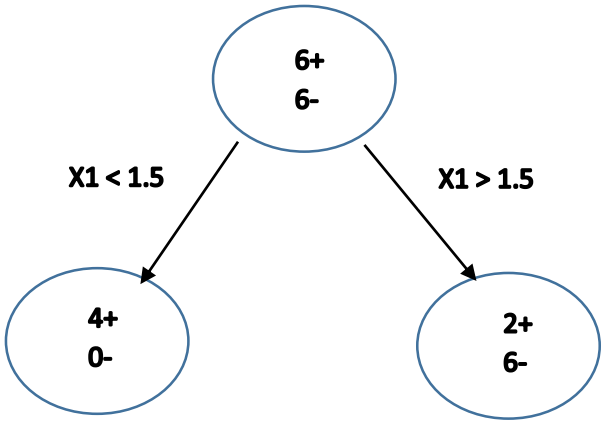
Since figure is symmetric and in an interval like (2, 6), place of the boundary will not affect the information gain, there are 2 ways to split the data.

i) boundary is X1 = 1.5 -> IG = 1 – 4/12*0 – 8/12 (-2/8 log(2/8) -6/8 log(6/8)) = 0.459147917

ii) boundary is X1 = 4 -> IG = 1 – (-2/6 log(2/6) - 4/6 log(4/6)) = 0.0817041659

Actually, without any calculation, we can infer that information gain of X1 = 1.5 is obviously higher than the information gain of X1 = 4.

Therefore, for root node, we split the node as X1 < 1.5 and X1 > 1.5

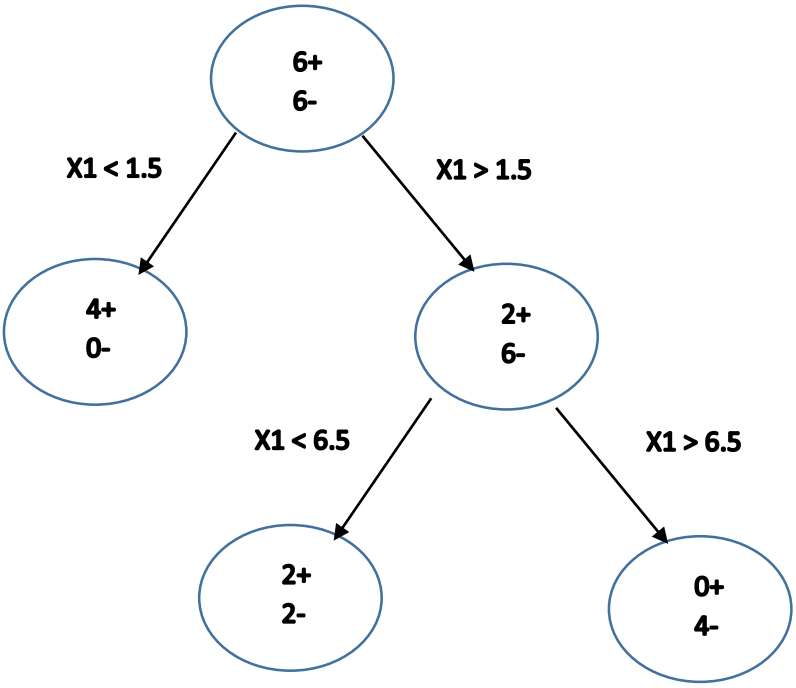


Since left node is pure we will not continue to split it. For right node, again there are 2 ways to split. (Splitting with a boundary parallel to X1 is obviously not a good idea.) I am going to compare that ways by looking the weighted average entropy (WAE) of the children.

i) boundary is X1 = 4 -> WAE = 2/8*0 + 6/8 (-2/6*log(2/6) - 4/6*log(4/6)) = 0.688721876

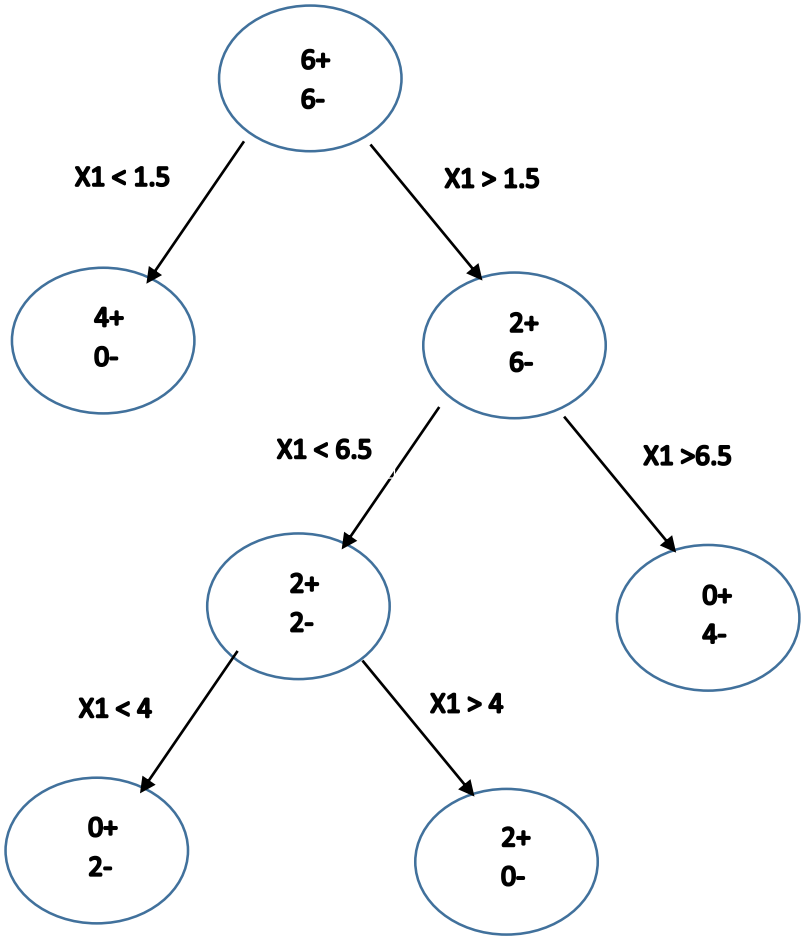
ii) boundary is X1 = 6.5 -> WAE = 4/8*0 + 4/8*1 = 0.5

Since WAE of X1 = 6.5 is lower, we split the node as X1 < 6.5 and X1 > 6.5.



Since right node is pure we will not continue to split it. For the left node, there is only 1 way to split. (Splitting with a boundary parallel to X1 is obviously not a good idea.)

We will split it as X1 < 4 and X1 > 4



Since all leaves are pure we will not continue to split.

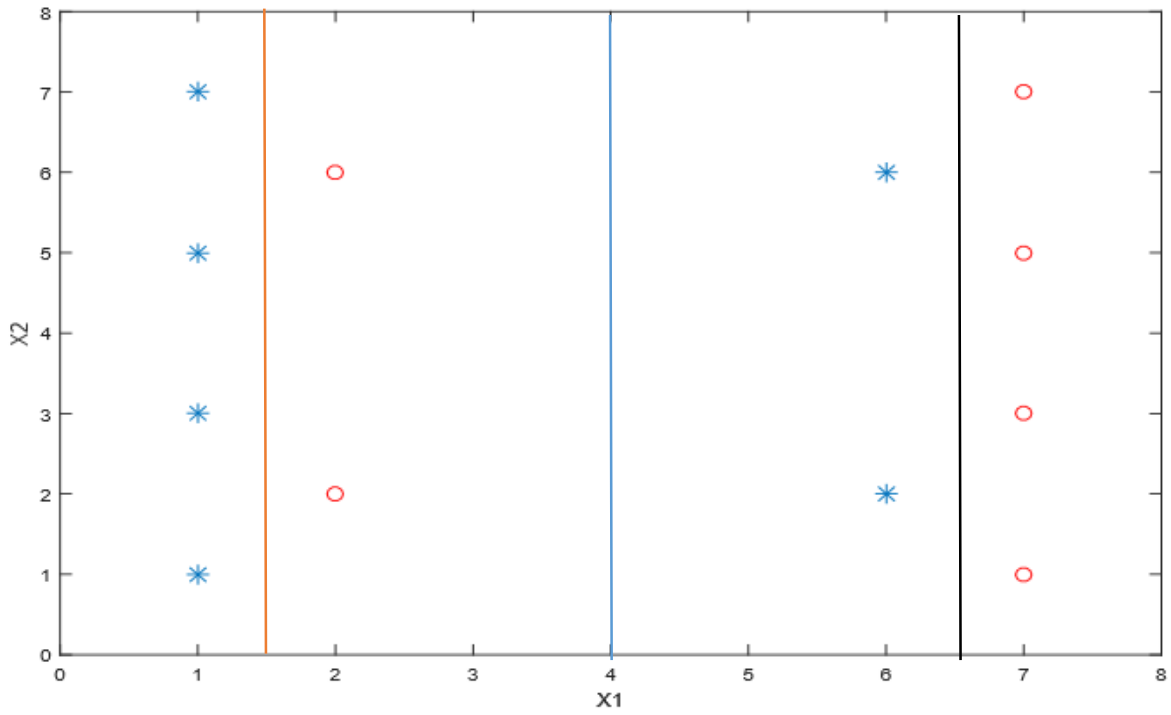
1-b)

According to 1-a this is the decision boundaries.

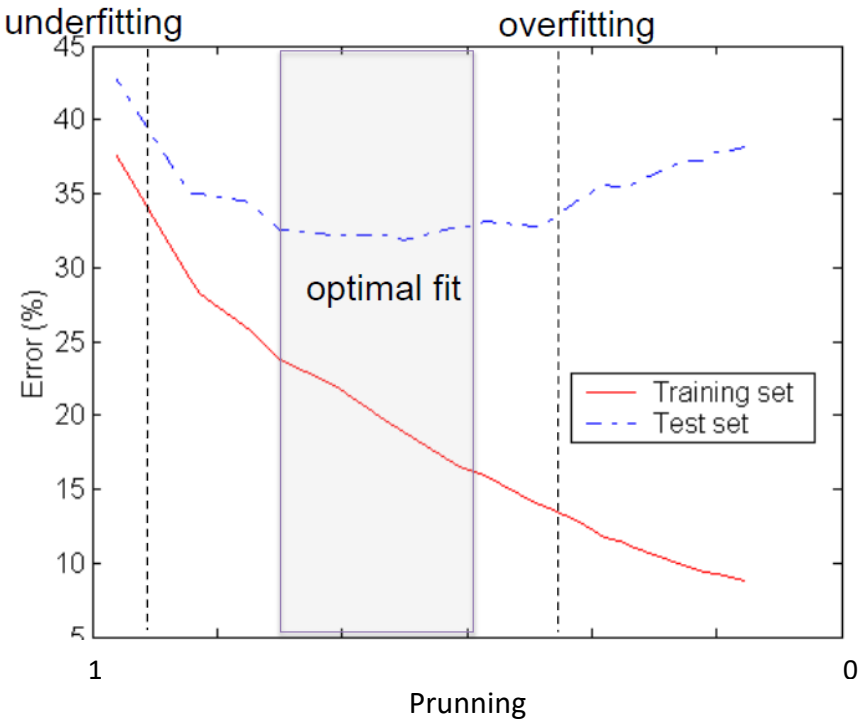
Red is for first split

Black is for second split

Blue is for third and last split.

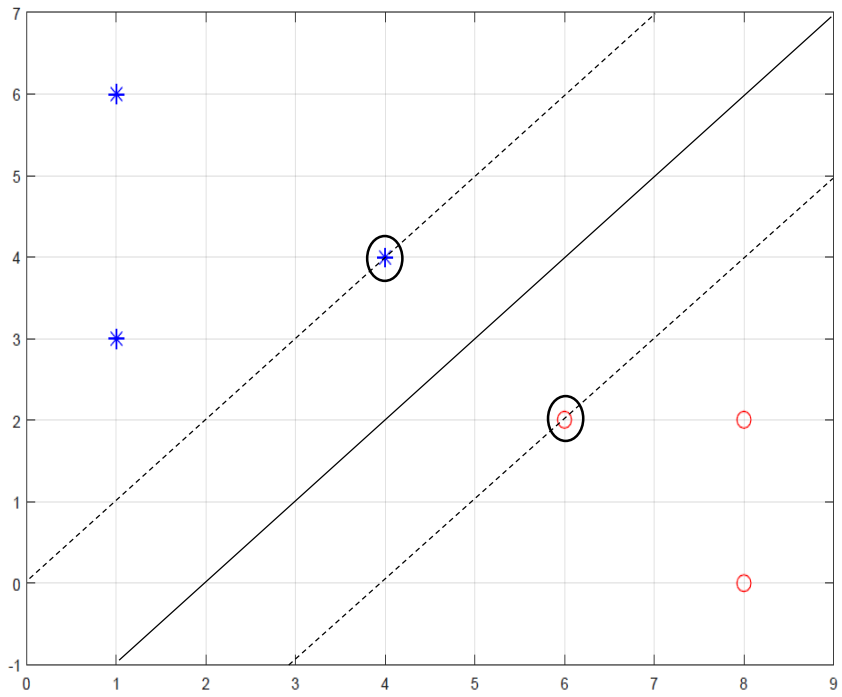


1-c)



When T goes to 0, which means all leaf nodes will be approximately pure, then number of nodes will increase to satisfy the purity. It will cause overfitting to training data so that test accuracy will decrease while training accuracy will increase. (Error of test will increase while error training will increase.) If T is 1, there will be no need for splitting which means all data will be considered as a class and error rate will be high (accuracy will be low.) for training and testing data and it will cause under fitting for both sets.

2-a)



2-b)

$\gamma = \sqrt{2} \rightarrow ||w|| = \frac{1}{\sqrt{2}} \rightarrow w = \left(\frac{1}{2}, -\frac{1}{2}\right) or\ w = \left(-\frac{1}{2}, \frac{1}{2}\right)$ since it is perpendicular the decision boundary. If $w = \left(\frac{1}{2}, -\frac{1}{2}\right)$, then w.(2,0) + b = 0 since (2,0) is on the decision boundary.

$$\left(\frac{1}{2}, -\frac{1}{2}\right)(2,0) = 1 \rightarrow b = -1$$

Then, w.(4,0) = 1 since it is on the margin of positive class.
$\left(\frac{1}{2}, -\frac{1}{2}\right)(4,0) = 2 \rightarrow 2 - b = 1$ so it is true. So there is no need for any calculation since we check the correctness of w and b.

Therefore, $w = \left(\frac{1}{2}, -\frac{1}{2}\right)$ and b = -1.

2-c)

Since $w = \sum_{i=1}^{N} \alpha_i y_i x_i\ and\ 0 = \sum_{i=1}^{N} \alpha_i y_i$ and for the points which are not support vectors, $\alpha_i = 0$, $\left(\frac{1}{2}, -\frac{1}{2}\right) = \alpha_+(6,2) - \alpha_-(4,4)$. So, $\alpha_+ = \alpha_- = \frac{1}{4}$ and if we put it $0 = \sum_{i=1}^{N} \alpha_i y_i$ it satisfies.

$\alpha_+\ and\ \alpha_-$ show the weights of support vectors according to their classes. (Since for both classes we have 1 support vectors.)

2-d)

If we moved the point at (4,4) to (5,3), then γ will decrease so ||w|| will increase. By $w = \sum_{i=1}^{N} \alpha_i y_i x_i$ since x_i and y_i do not change, α_i will increase.

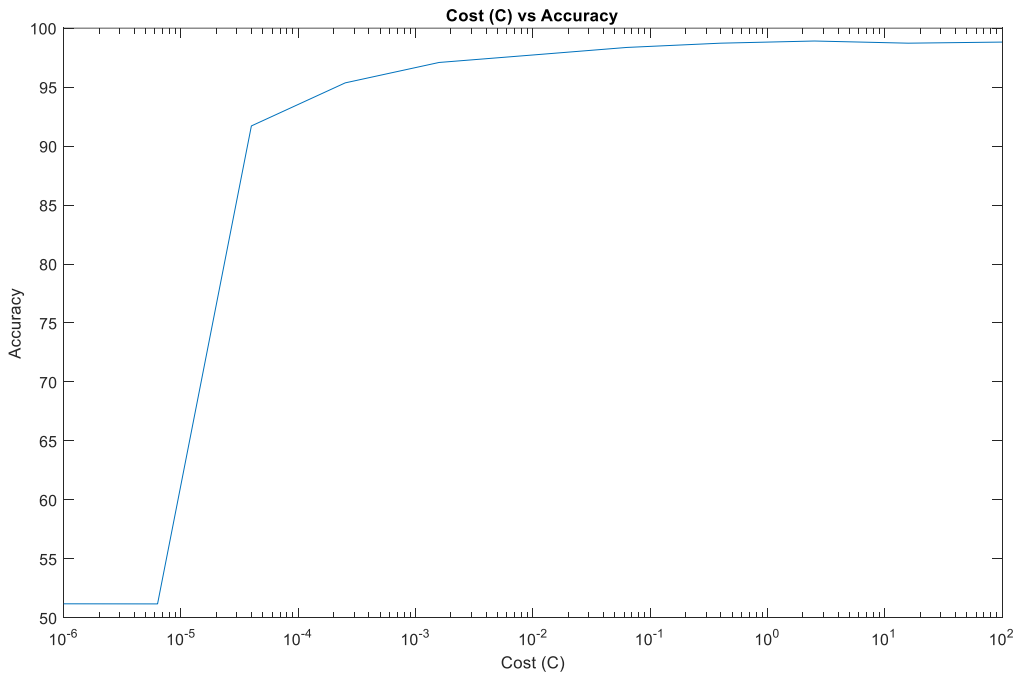
3-a)

See the appendix for the code and output the decision values of test set. [hw2_q3a.m, crossValidation_3a.m, hw2_q3a_PredictLabels.txt]

Highest accuracy for cross validation is 98.9074

Best cost parameter C corresponding to highest cross validation accuracy is 2.5119

Accuracy for the test data is 98.5138



3-b)

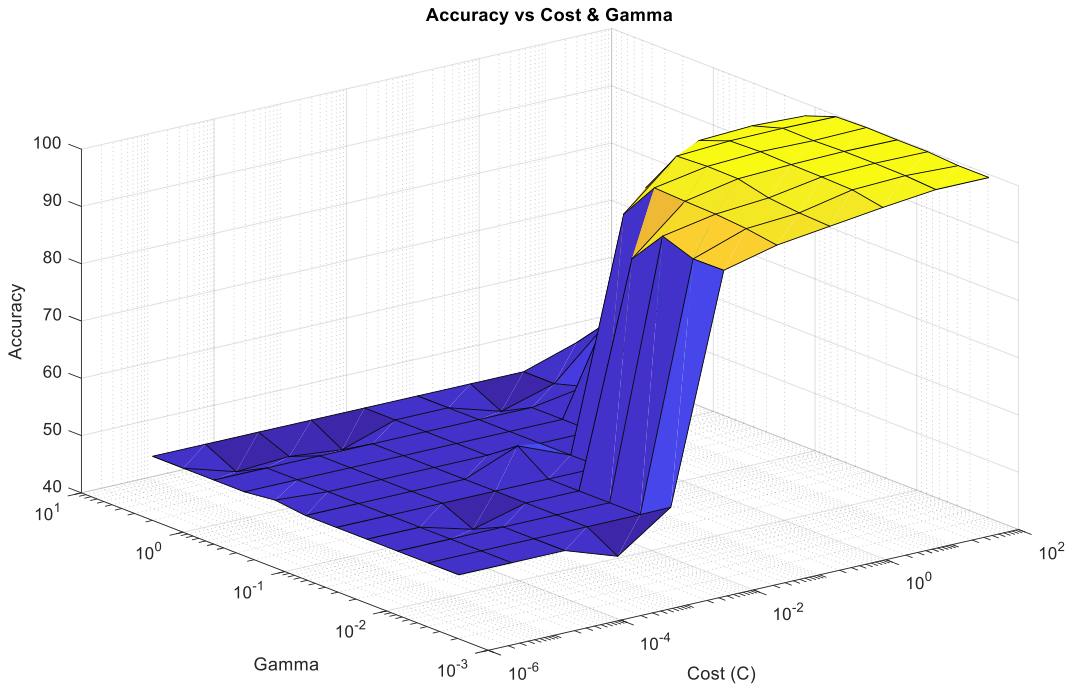
See the appendix for the code and output the decision values of test set [hw2_q3b.m, crossValidation_3b.m, hw2_q3b_PredictLabels.txt]

Highest accuracy for cross validation is 99.9091

Best cost parameter C corresponding to highest cross validation accuracy is 15.8489

Best gamma corresponding to highest cross validation accuracy is 0.0156

Accuracy for the test data is 99.7877



RBF kernel gives better result than linear kernel in terms of accuracies for this case. The data set is mostly linearly separable, which means we can find linear decision boundary by assigning a cost parameter so that linear kernel gives acceptable accuracy. However, that does not mean that it is completely linearly separable. Transforming features with RBF kernel leads to an infinite dimensional representation for the data and it gives better result than liner kernel due to nan-linearly separable points of the data.

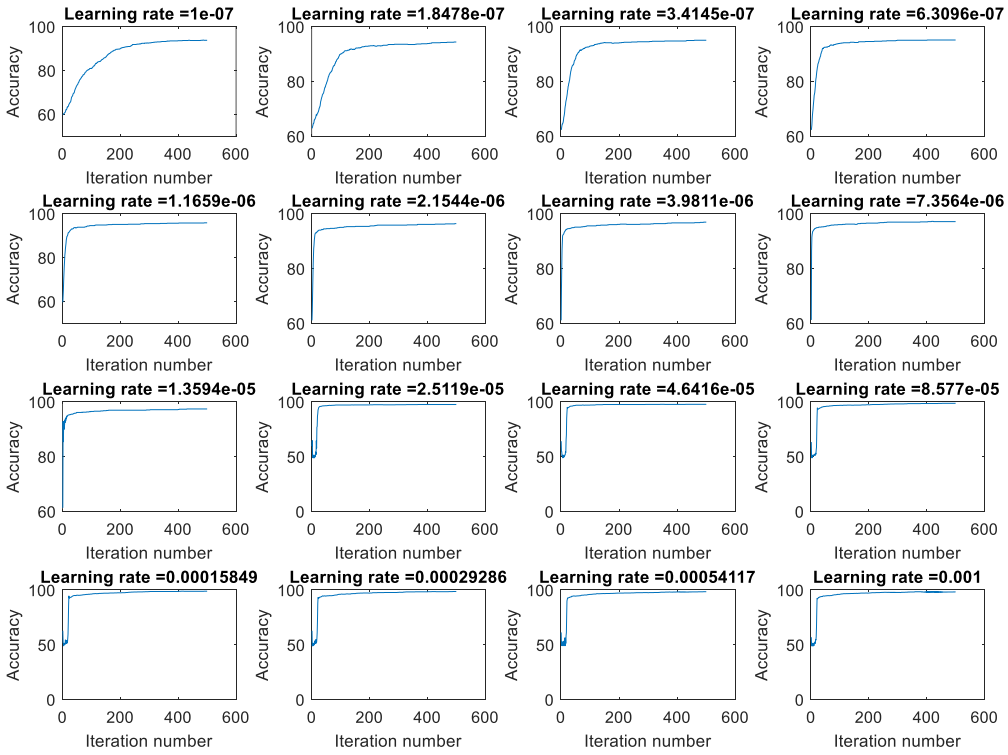
4)

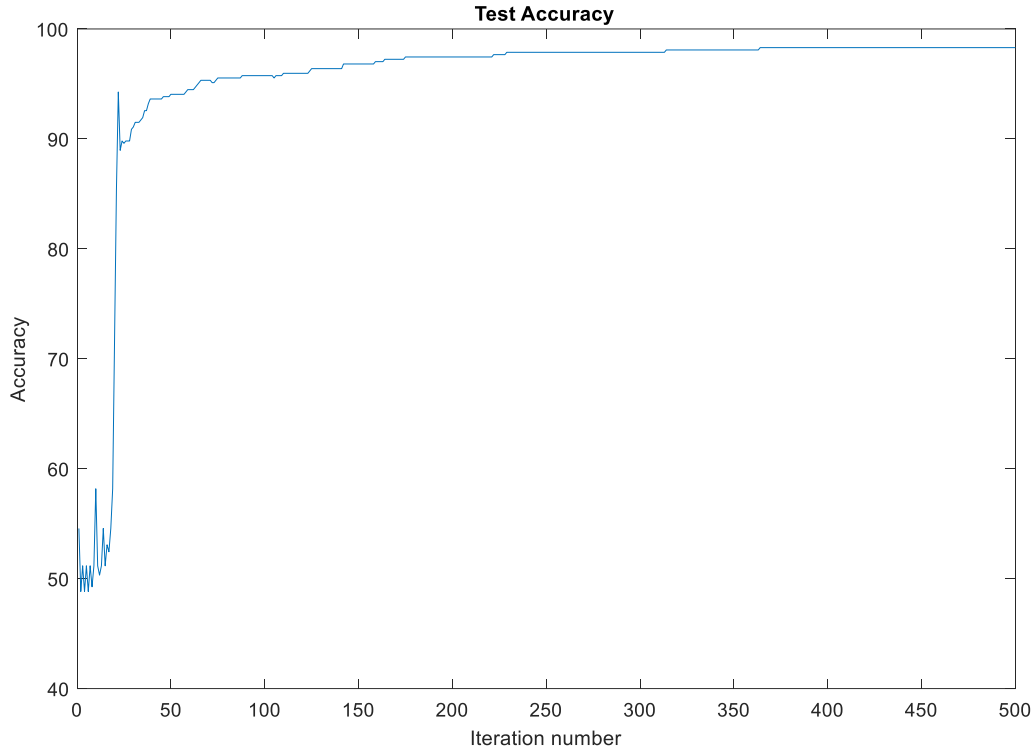
See the appendix for the code [hw2_q4.m, crossValidation_4.m]

Highest accuracy for cross validation is 98.7248

Best learning rate value corresponding to highest cross validation accuracy is 0.0002

Best accuracy for the test data is 98.3015




Test Accuracy
Accuracy
Iteration number
100
90
80
70
60
50
40
0
50
100
150
200
250
300
350
400
450
500

APPENDIX

```
% Function for Question 3a
% hw2_q3a function

% It takes the dataPath parameter
% dataPath is the path of the folder in which HW2data.mat placed

% It gives bestAccuracy, bestC, testAccuracy array as output
% output = [bestAccuracy bestC testAccuracy];
% bestAccuracy shows highest cross validation accuracy
% bestC shows the corresponding C value of bestAccuracy
% testAccuracy shows the accuracy of the test data

function [output] = hw2_q3a(dataPath)
% Example dataPath =
% 'C:\Users\lenovo\Desktop\MESUD\6. Semester\CS 464 Machine
Learning\HW 2'

% Getting data from data folder
currPath = cd(dataPath);
data = load('HW2data.mat');
cd(currPath);

Bs = data.Bs;
Ps = data.Ps;

% To shuffle the data set
Bs = Bs(randperm(size(Bs, 1)), :);
Ps = Ps(randperm(size(Ps, 1)), :);

% Save shuffled versions of data set to workspace
save('shuffledB.mat', 'Bs');
save('shuffledP.mat', 'Ps');

% 70% of the data is train data
sizeB = int32( size(Bs,1)*0.7 );
sizeP = int32( size(Ps,1)*0.7 );
train = Bs(1:sizeB,:);
train = [train;Ps(1:sizeP,:)];

% Class B is 0, Class P is 1
truetrainlabel = zeros(sizeB+sizeP,1);
truetrainlabel(sizeB+1:end,1) = ones;

% Remainder (30%) of the data is test data
test = Bs(sizeB+1:end,:);
test = [test;Ps(sizeP+1:end,:)];

truetestlabel = zeros(size(Bs,1)-sizeB+size(Ps,1)-sizeP,1);
truetestlabel(size(Bs,1) - sizeB + 1 : end, 1) = ones;

% Cost parameter C
noofC = 11;
C = logspace(-6,2,noofC);
successOfC = zeros(1,noofC);
```

```
kfold = 10;

% Cross Validation and Accuracy of C Calculation
for i=1:noofC
    [ avg ] = crossValidation_3a(C(i), kfold, train,
truetrainlabel);
    successOfC(1,i) = avg;
end

successOfC = successOfC * 100;

% Plot for accuracy vs cost parameter C
figure;
semilogx(C, successOfC);
title('Cost (C) vs Accuracy');
xlabel('Cost (C)');
ylabel('Accuracy');

% Best accuracy of all C's
[bestAccuracy, indexC] = max(successOfC);
bestC = C(1,indexC);

% Train SVM with best C
bestSVMMdl = fitcsvm(train, truetrainlabel, 'KernelFunction',
'linear', 'BoxConstraint', bestC);
% Prediction for test data
predictlabel = predict(bestSVMMdl, test);
% Accuracy of SVM model with test data
CP = classperf(truetestlabel, predictlabel);
testAccuracy = CP.CorrectRate*100;

% output of the function
output = [bestAccuracy bestC testAccuracy];

% Output of predict label of test data
file = fopen('hw2_q3a_PredictLabels.txt','w');
fprintf(file,'%d\n',predictlabel);
fclose(file);

end
```

```
% crossValidation_3a function
% Function for Cross Validation and Accuracy of C Calculations

% It takes C, K, traindata, label as input
% C = cost parameter
% K shows the fold parameter
% traindata is train data for k fold cross validation
% label is the true label for the train data

% It gives avg as output
% avg is the average of the accuracy of k folds for that C

function [avg] = crossValidation_3a(C, K, traindata, label)
```

```
% divide train data into K group
indices = crossvalind('Kfold', size(traindata, 1), K);

success = zeros(1,K);

% In each turn, a group is test and other is train
% Train a SVM model and calculate the accuracy
for index = 1:K
    test = (indices == index);
    train = ~test;
    svmMdl = fitcsvm(traindata(train,:), label(train,1),
'KernelFunction', 'linear', 'BoxConstraint', C);

    predictlabel = predict(svmMdl, traindata(test,:));

    CP = classperf(label(test,1), predictlabel);

    success(1,index) = CP.CorrectRate;
end

% mean of the accuracies of K folds for given C
avg = mean(success);

end
```

```
% Function for Question 3b
% hw2_q3b function


% It gives bestAccuracy, bestC, bestGamma, testAccuracy array as
output
% output = [bestAccuracy bestC bestGamma testAccuracy];
% bestAccuracy shows highest cross validation accuracy
% bestC shows the corresponding C value of bestAccuracy
% bestGamma shows the corresponding Gamma value of bestAccuracy
% testAccuracy shows the accuracy of the test data

function [output] = hw2_q3b()
% Getting data from the folder
sBs = load('shuffledB.mat');
sPs = load('shuffledP.mat');

Bs = sBs.Bs;
Ps = sPs.Ps;

% 70% of the data is train data
sizeB = int32( size(Bs,1)*0.7 );
sizeP = int32( size(Ps,1)*0.7 );
train = Bs(1:sizeB,:);
train = [train;Ps(1:sizeP,:)];

% Class B is 0, Class P is 1
truetrainlabel = zeros(sizeB+sizeP,1);
truetrainlabel(sizeB+1:end,1) = ones;
```

```
% Remainder (30%) of the data is test data
test = Bs(sizeB+1:end,:);
test = [test;Ps(sizeP+1:end,:)];

truetestlabel = zeros(size(Bs,1)-sizeB+size(Ps,1)-sizeP,1);
truetestlabel(size(Bs,1) - sizeB + 1 : end, 1) = ones;

% Cost parameter C and Gamma
noofC = 11;
C = logspace(-6,2,noofC);
gamma = [1/512 1/256 1/128 1/64 1/32 1/16 1/8 1/4 1/2 1 2];

accuracy = zeros(noofC, noofC);

kfold = 10;

% Cross Validation and Accuracy of C and Gamma Calculations
for i=1:noofC
    for k=1:noofC
        [ avg ] = crossValidation_3b(C(i), gamma(k), kfold, train,
truetrainlabel);
        accuracy(i,k) = avg;
    end
end

accuracy = accuracy * 100;

% Surface Plot for accuracy vs parameters C and Gamma
surf(C, gamma, accuracy.');
title('Accuracy vs Cost & Gamma');
xlabel('Cost (C)');
ylabel('Gamma');
zlabel('Accuracy');
set(gca,'xScale','log','yScale','log');

% Best accuracy among all C's and Gamma's
[bestAccuracies, index] = max(accuracy);
[bestAccuracy, indexGamma] = max(bestAccuracies);

indexC = index(indexGamma);
bestC = C(indexC);
bestGamma = gamma(indexGamma);

% Train SVM with best C and best Gamma
bestSVMMdl = fitcsvm(train, truetrainlabel, 'KernelFunction', 'RBF',
'BoxConstraint', bestC, 'KernelScale', 1/(2*bestGamma)^(1/2));
% Prediction for test data
predictlabel = predict(bestSVMMdl, test);
% Accuracy of SVM model with test data
CP = classperf(truetestlabel, predictlabel);
testAccuracy = CP.CorrectRate * 100;

% output of the function
output = [bestAccuracy bestC bestGamma testAccuracy];
```

```
% Output of predict label of test data
file = fopen('hw2_q3b_PredictLabels.txt','w');
fprintf(file,'%d\n',predictlabel);
fclose(file);

end
```

```
% crossValidation_3b function
% Function for Cross Validation and Accuracy of C and Gamma
Calculations

% It takes C, Gamma K, traindata, label as input
% C = cost parameter
% Gamma = gamma parameter
% K shows the fold parameter
% traindata is train data for k fold cross validation
% label is the true label for the train data

% It gives avg as output
% avg is the average of the accuracy of k folds for that C and Gamma

function [avg] = crossValidation_3b(C, Gamma, K, traindata, label)

% divide train data into K group
indices = crossvalind('Kfold', size(traindata, 1), K);

success = zeros(1,K);

% In each turn, a group is test and other is train
% Train a SVM model and calculate the accuracy
for index = 1:K
    test = (indices == index);
    train = ~test;
    svmMdl = fitcsvm(traindata(train,:), label(train,1),
'KernelFunction', 'RBF', 'BoxConstraint', C, 'KernelScale',
1/(2*Gamma)^(1/2));

    predictlabel = predict(svmMdl, traindata(test,:));

    CP = classperf(label(test,1), predictlabel);

    success(1,index) = CP.CorrectRate;
end

% mean of the accuracies of K folds for given C
avg = mean(success);

end
```

```
% Function for Question 4
% hw2_q4 function

% It gives bestAccuracy, bestRate, bestTestAccuracy array as output
```

```
% output = [bestAccuracy bestRate bestTestAccuracy];
% bestAccuracy shows highest cross validation accuracy
% bestRate shows the corresponding learning rate value for
bestAccuracy
% bestTestAccuracy shows the accuracy of the test data

function [output] = hw2_q4()
% Getting data from data folder
sBs = load('shuffledB.mat');
sPs = load('shuffledP.mat');

Bs = sBs.Bs;
Ps = sPs.Ps;

% 70% of the data is train data
sizeB = int32( size(Bs,1)*0.7 );
sizeP = int32( size(Ps,1)*0.7 );
train = Bs(1:sizeB,:);
train = [train;Ps(1:sizeP,:)];

% A column of ones are added to the train matrix
% That column shows the wi parameters
train = [ones(size(train,1),1) train];

% Class B is 0, Class P is 1
truetrainlabel = zeros(sizeB+sizeP,1);
truetrainlabel(sizeB+1:end,1) = ones;

% Remainder (30%) of the data is test data
test = Bs(sizeB+1:end,:);
test = [test;Ps(sizeP+1:end,:)];

% A column of ones are added to the test matrix
% That column shows the wi parameters
test = [ones(size(test,1),1) test];

truetestlabel = zeros(size(Bs,1)-sizeB+size(Ps,1)-sizeP,1);
truetestlabel(size(Bs,1) - sizeB + 1 : end, 1) = ones;

% Learning Rate
% I take the square of aNumber for the plot
% The plot will be aNumber by aNumber
aNumber = 4;
% Learning Rates
noofrate = aNumber*aNumber;
lRate = logspace(-7,-3,noofrate);

% No of Iterations
noofIter = 500;
% K fold number
kfold = 10;

accuracy = zeros(noofIter, noofrate);

% Cross Validation and Accuracy Calculations with different learning
rates
```

```
% Also this for loop is for plot the ploting di?erent accuracies
over number of iterations
for i=1:noofrate
    [ accuracy(:,i)] = crossValidation_4(lRate(i), kfold, train,
truetrainlabel, noofIter);   %calculate the cross validation
accuracy
    subplot(aNumber,aNumber,i);
    plot(accuracy(:,i));
    title(strcat('Learning rate = ', num2str(lRate(i))));
    xlabel('Iteration number');
    ylabel('Accuracy');
end

% Take the last iterations accuracy for each rate
% It is for finding the best learning rate
success = accuracy(500,:);
[bestAccuracy,index] = max(success);
bestRate = lRate(index);

w = zeros(1, size(train,2));
testAccuracies = zeros(1, noofIter);

for j=1:noofIter
    % Calculation for change in w
    deltaW = (truetrainlabel - (exp(train*w')./(1 + exp(train*w')))
)' * train;
    w = w + bestRate*deltaW;

    predictlabel = test * w';

    % predictlabel is the boundary line.
    % if predictlabel > 0 assign its label as 1
    % ow. assign its label as 0
    predictlabel (predictlabel > 0) = 1;
    predictlabel (predictlabel <= 0) = 0;

    %Calculation for Accuracy
    testAccuracy = ( 1-sum( abs(predictlabel - truetestlabel) ) /
size(test,1) ) * 100;

    testAccuracies(1,j) = testAccuracy;
end

[bestTestAccuracy,~] = max(testAccuracies);

% Plot for the test accuracy over number of iterations
% the best accuracy obtained over the test dataset
figure;
plot(testAccuracies)
title('Test Accuracy');
xlabel('Iteration number');
ylabel('Accuracy');

% output of the function
output = [bestAccuracy bestRate bestTestAccuracy];
```

```
end
```

```
% crossValidation_4 function
% Function for Cross Validation and Accuracy Calculations with
different learning rates

% It takes rate, K, traindata, label, noofIter as input
% rate is learning rate
% K shows the fold parameter
% traindata is train data for k fold cross validation
% label is the true label for the train data
% noofIter shows the number of iterations

% It gives avg as output
% avg is the average of the accuracy of k folds for that learning
rate

function [avg] = crossValidation_4(rate, K, traindata, label,
noofIter)

% divide train data into K group
indices = crossvalind('Kfold', size(traindata, 1), K);

success = zeros(noofIter, K);

% In each turn, a group is test and other is train
% Train a SVM model and calculate the accuracy
for index = 1:K
    test = (indices == index);
    train = ~test;

    w = zeros(1, size(traindata,2));

    for j=1:noofIter
        % Calculation for change in w
        deltaW = ( label(train,1) - (exp(traindata(train,:)*w')./(1
+ exp(traindata(train,:)*w'))) )' * traindata(train,:);
        w = w + rate*deltaW;

        predictlabel = traindata(test,:)*w';

        % predictlabel is the boundary line.
        % if predictlabel > 0 assign its label as 1
        % ow. assign its label as 0
        predictlabel (predictlabel > 0) = 1;
        predictlabel (predictlabel <= 0) = 0;

        %Calculation for Accuracy
        accuracy = ( 1-sum( abs(predictlabel - label(test,1)) ) /
size(traindata(test,:),1) ) * 100;

        success(j,index) = accuracy;
    end
end
```

```
% mean of the accuracies of K folds for given learning rate
avg = mean(success, 2);

end
```

---

hw2_q3a_PredictLabels.txt and hw2_q3b_PredictLabels.txt are in the zip file. They show the output of the decision values of test set for questions 3a and 3b respectively.