Contents


```
clc
clear all
close all
```

Part 1.1 Getting Data via Python

loading data

```
load('sonar5841-1.mat')
```

Part 1.3 Partitioning the data

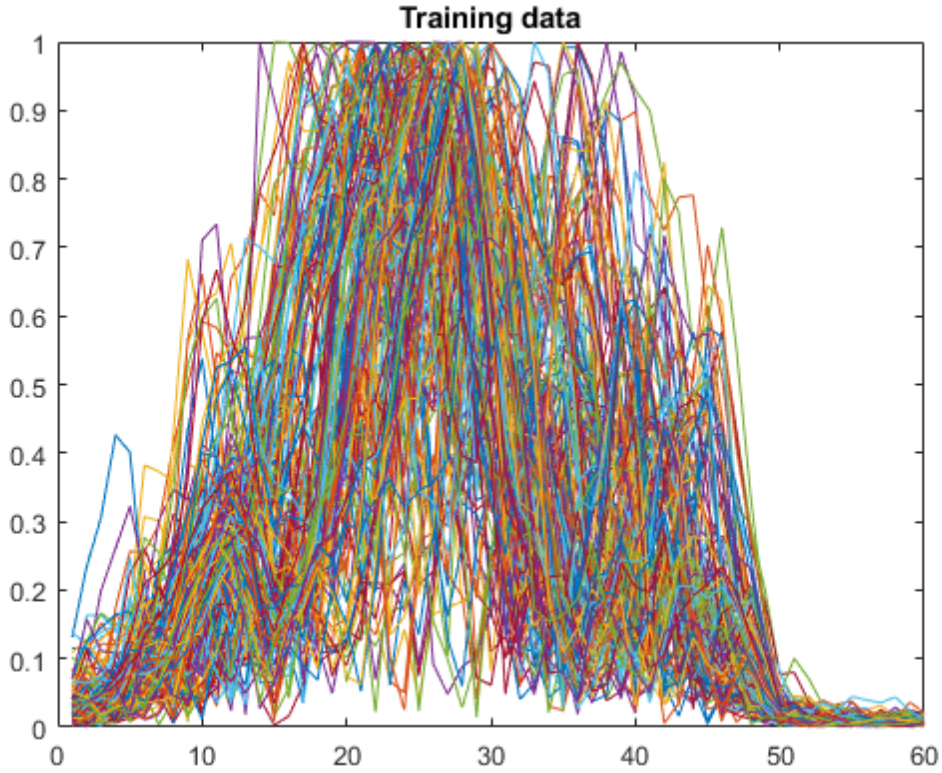
```
data = [X Y];
seed = 7;
rng(seed)
idx = randperm(208,42);
idx = sort(idx);
testdata = data(idx,:);
traindata = data;
traindata(idx,:) = [];
trainfeatures = traindata(:,1:end-1);
trainlabels = traindata(:,end);
testfeatures = testdata(:,1:end-1);
testlabels = testdata(:,end);
```

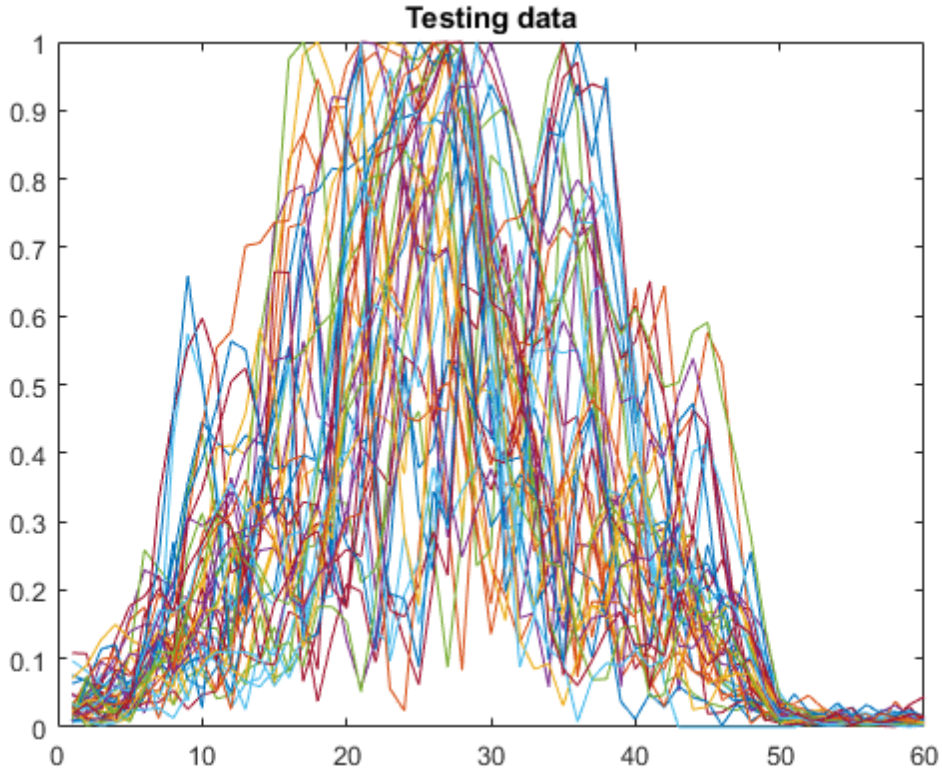
Part 1.4 Part 1 Deliverables

Plot for training data

```
figure
for i = 1:size(traindata,1)
    plot(traindata(i,1:end-1));
    title('Training data');
    hold on
```

```
end
hold off

% Plot for testing data
figure
for i = 1:size(testdata,1)
    plot(testdata(i,1:end-1));
    title('Testing data');
    hold on
end
hold off
```





Part 2.1 Logistic Regression

```
x = trainfeatures;
y = trainlabels;
```

```
for i = 1:size(y,1)
    if y(i,1) == -1
        y(i,1) = 0;
    end
end

[m, n] = size(x);
x = [ones(m, 1) x];
initial_theta = zeros(n + 1, 1);

options = optimset('GradObj', 'on', 'MaxIter', 1000);
[theta, cost] = fminunc(@(t)(costFunction(t, x, y)), initial_theta, options);

x = testfeatures;
y = testlabels;
for i = 1:size(y,1)
    if y(i,1) == -1
        y(i,1) = 0;
    end
end

[m, n] = size(x);
x = [ones(m, 1) x];
p = pred(theta, x);
fprintf('Accuracy on test data using Logistic Regression: %f\n', mean(double(p == y)) * 100)
lracc = mean(double(p == y)) * 100;
```

Part 2.2 Linear SVM (Hard Margin)

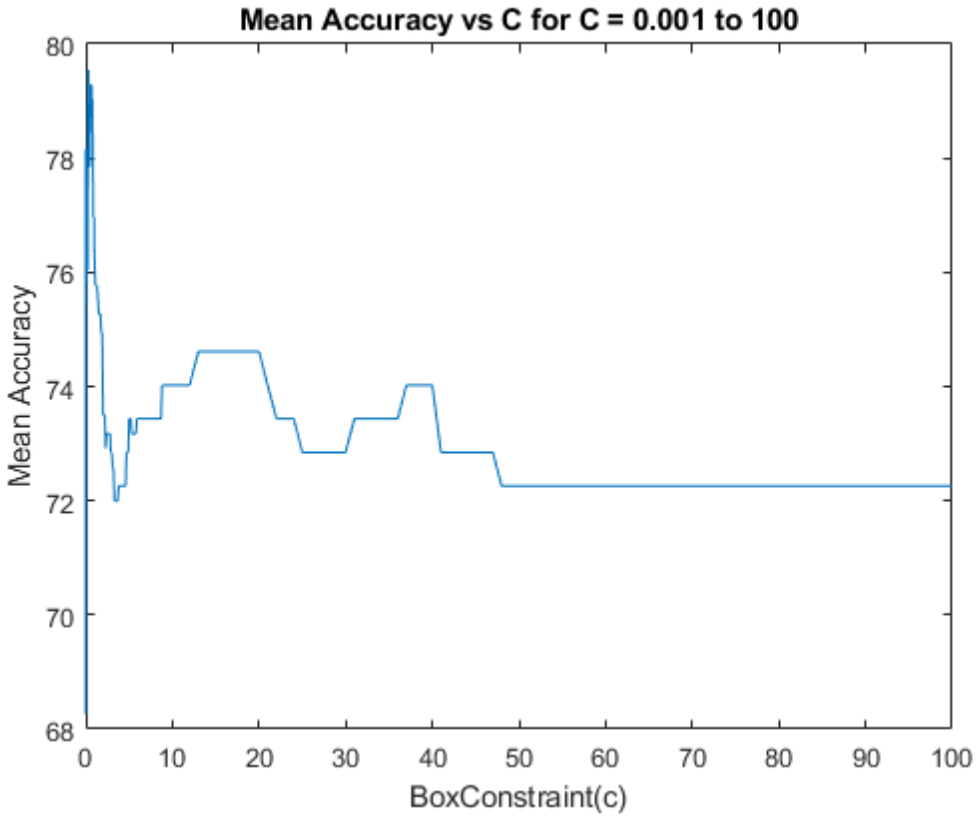
```
x = trainfeatures;
y = trainlabels;

% C: Box constraint
LinearSVMmodel = fitcsvm(x, y,'standardize',true);

x = testfeatures;
y = testlabels;

[label, ~] = predict(LinearSVMmodel,x);

fprintf('Accuracy on test data using Linear SVM (Hard Margin): %f\n', mean(double(label == y)) * 100)
hmacc = mean(double(label == y)) * 100;
```

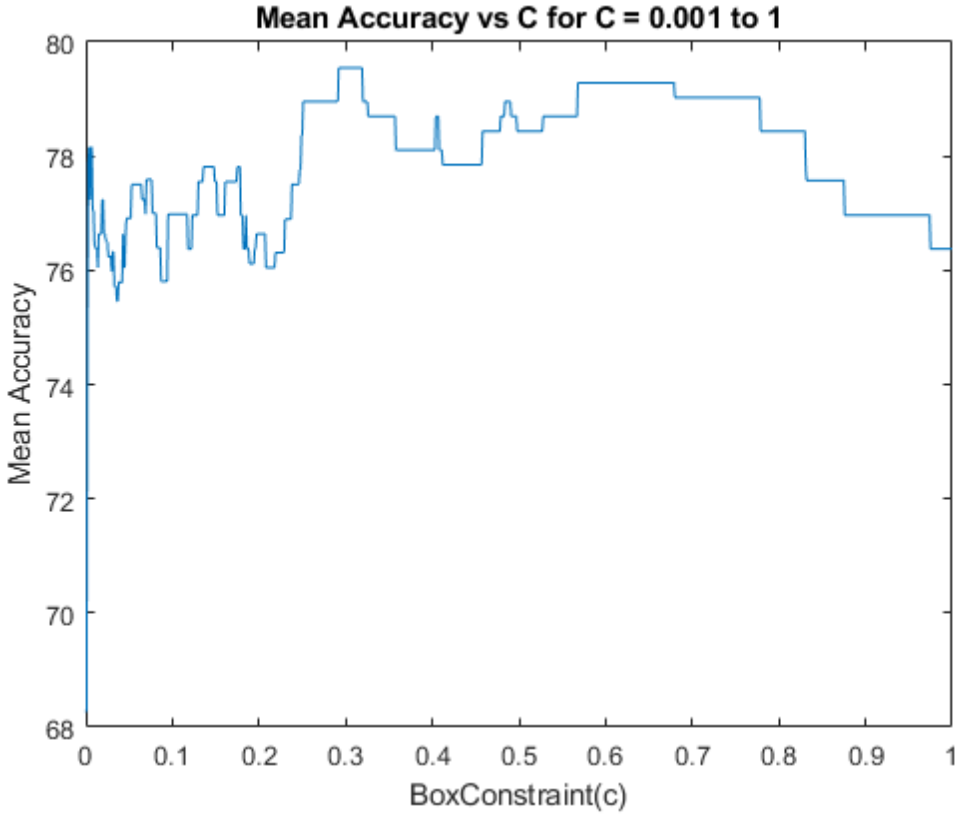
```
Accuracy on test data using Linear SVM (Hard Margin): 71.428571
```

Implementing 5-fold Cross Validation to find out C

```
interval = round(size(traindata,1)/5);
parts = [1; interval; (2*interval); (3*interval); (4*interval); 208];

rng(seed)
data = data(randperm(size(data,1)),:);

for j = 1:1180
    if j <= 1000
        c = 0.001 * (j);
    elseif j <= 1090
```

```
        c = 1+ 0.1 * (j-1000);
    else
        c = 10 + 1 * (j-1090);
    end
    Acc = zeros(5,1);
    for i = 1:5
        testfeatures1 = data(parts(i,1):parts(i+1,1),1:end-1);
        testlabels1 = data(parts(i,1):parts(i+1,1),end);
        trainfeatures1 = data(:,1:end-1);
        trainlabels1 = data(:,end);
        trainfeatures1(parts(i,1):parts(i+1,1),:) = [];
        trainlabels1(parts(i,1):parts(i+1,1),:) = [];
        x = trainfeatures1;
        y = trainlabels1;
        LinearSVMmodel = fitcsvm(x, y,'standardize',true,'BoxConstraint',c);
        x = testfeatures1;
        y = testlabels1;
        [label, ~] = predict(LinearSVMmodel,x);
        Acc(i,1) = mean(double(label == y)) * 100;
    end
    BoxConstraint(j,1) = c;
    MeanAcc(j,1) =  mean(Acc);

end

figure
plot(BoxConstraint(:,1), MeanAcc(:,1));
xlabel('BoxConstraint(c)');
ylabel('Mean Accuracy');
title('Mean Accuracy vs C for C = 0.001 to 100')

figure
plot(BoxConstraint(1:1000,1), MeanAcc(1:1000,1));
xlabel('BoxConstraint(c)');
xlim([0 1]);
ylabel('Mean Accuracy');
title('Mean Accuracy vs C for C = 0.001 to 1')
```





Implementation of SVM using value of C obtained from cross validation

```
[~,idx] = max(MeanAcc);
c = BoxConstraint(idx,1);
```

```
x = trainfeatures;
y = trainlabels;

LinearSoftSVMmodel = fitcsvm(x, y,'standardize',true,'BoxConstraint',c);

x = testfeatures;
y = testlabels;

[label, ~] = predict(LinearSoftSVMmodel,x);

fprintf('Accuracy on test data using SVM (c = %f) : %f\n',c, mean(double(label == y)) * 100);
smacc = mean(double(label == y)) * 100;
```

```
Accuracy on test data using SVM (c = 0.292000) : 78.571429
```

Part 2.3 Kernel SVM (trained using default width)

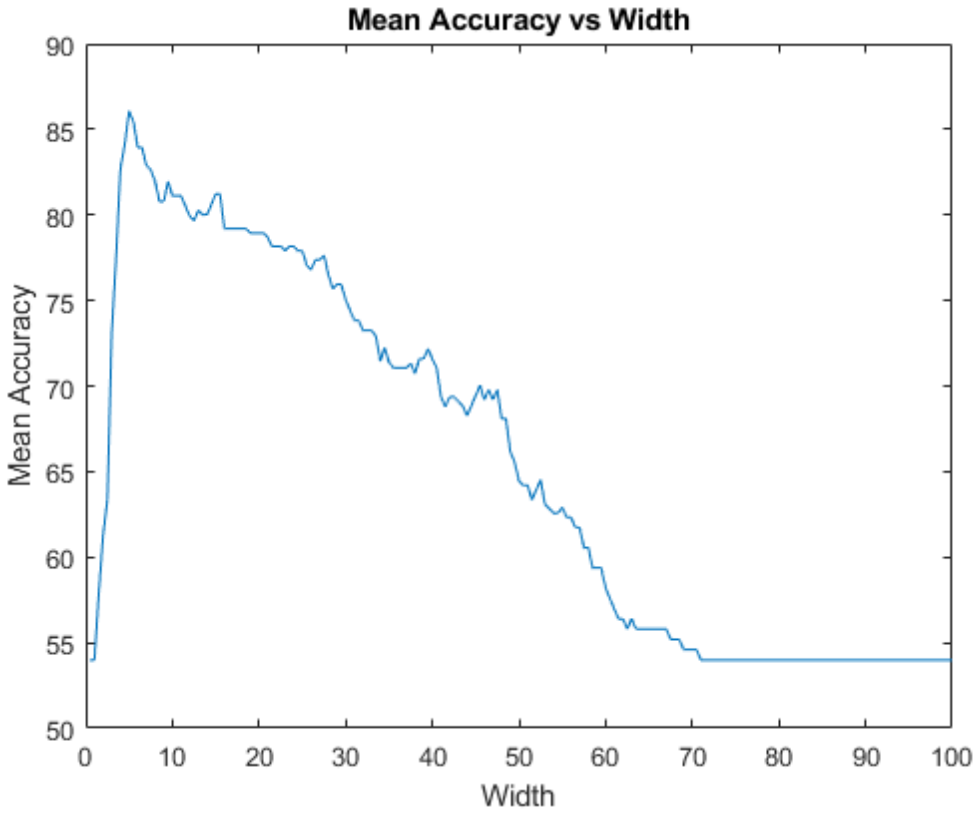
```
x = trainfeatures;
y = trainlabels;

KernelSVMmodel = fitcsvm(x, y,'standardize',true,'KernelFunction','rbf');

x = testfeatures;
y = testlabels;

[label, ~] = predict(KernelSVMmodel,x);

fprintf('Accuracy on test data using Kernel SVM (width = 1): %f\n', mean(double(label == y)) * 100)
```

```
Accuracy on test data using Kernel SVM (width = 1): 52.380952
```

Implementing 5 fold cross validation to find width parameter in Kernel SVM

```
for j = 1:200
    width = 0.5 * (j) ;
    Acc = zeros(5,1);
    for i = 1:5
        testfeatures1 = data(parts(i,1):parts(i+1,1),1:end-1);
        testlabels1 = data(parts(i,1):parts(i+1,1),end);
        trainfeatures1 = data(:,1:end-1);
        trainlabels1 = data(:,end);
        trainfeatures1(parts(i,1):parts(i+1,1),:) = [];
        trainlabels1(parts(i,1):parts(i+1,1),:) = [];

        x = trainfeatures1;
        y = trainlabels1;

        KernelSVMmodel = fitcsvm(x, y,'standardize',true,'KernelFunction','rbf','KernelScale',width);

        x = testfeatures1;
        y = testlabels1;

        [label, ~] = predict(KernelSVMmodel,x);

        Acc(i,1) = mean(double(label == y)) * 100;
```

```
    end

    gamma(j,1) = width;
    MeanAcc1(j,1) =  mean(Acc);

end
figure
plot(gamma(:,1), MeanAcc1(:,1));
xlabel('Width');
ylabel('Mean Accuracy');
title('Mean Accuracy vs Width');
```



Implementation of kernel SVM using value of width obtained from cross validation

```
[~,idx] = max(MeanAcc1);
width = gamma(idx,1);

x = trainfeatures;
y = trainlabels;

LinearSoftSVMmodel = fitcsvm(x, y,'standardize',true,'KernelFunction','rbf','KernelScale',width);

x = testfeatures;
y = testlabels;

[label, ~] = predict(LinearSoftSVMmodel,x);

fprintf('Accuracy on test data using SVM (width = %f) : %f\n',width, mean(double(label == y)) * 100);
ksvmacc = mean(double(label == y)) * 100;
```

```
Accuracy on test data using SVM (width = 5.000000) : 92.857143
```

Table of Accuracies

```
Method = {'Logistic Regression'; 'Linear SVM (Hard Margin)'; 'Linear SVM (Soft Margin)' ; 'Kernel SVM'};
Accuracy = {lracc; hmacc; smacc; ksvmacc};
cc = sprintf('c = %f',c);
dd = sprintf('width = %f',width);
Parameters = {'Null'; 'Null'; cc ; dd};
T = table(Method, Accuracy, Parameters);
disp(T)
```

```
            Method              Accuracy         Parameters
    ____________________________    _________    ___________________

    'Logistic Regression'           [69.0476]    'Null'
    'Linear SVM (Hard Margin)'      [71.4286]    'Null'
    'Linear SVM (Soft Margin)'      [78.5714]    'c = 0.292000'
    'Kernel SVM'                    [92.8571]    'width = 5.000000'
```

Functions

```
function [J, grad] = costFunction(theta, X, y)
m = length(y);
J = 0;
grad = zeros(size(theta));
J = (-1/m) * sum(y'*log (sigmoid(X*theta)) + (1-y)'* log(1 - sigmoid(X*theta)));
grad = (1/m) * sum( (sigmoid(X*theta) - y) .* X );
end

function g = sigmoid(z)
g = zeros(size(z));
g = 1./(1+exp(-z));
end

function p = pred(theta, X)
m = size(X, 1);
p = zeros(m, 1);
p = sigmoid(X*theta);
s=0.5*ones(size(p));
p = (p>=s);
end
```

```
Local minimum possible.

fminunc stopped because it cannot decrease the objective function
along the current search direction.

Accuracy on test data using Logistic Regression: 69.047619
```

Published with MATLAB® R2019a