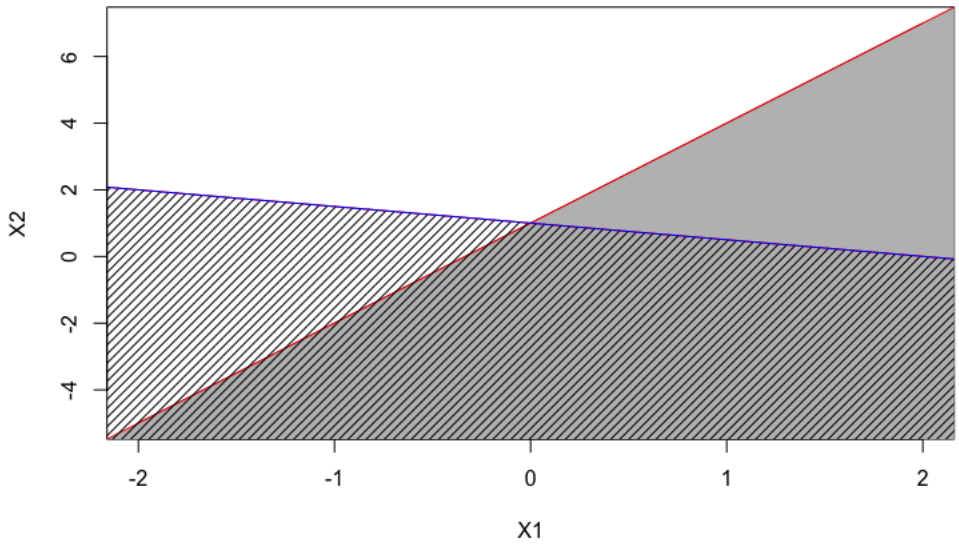
Math 189 HW7 Solutions

Q1.

(a) $1 + 3X_1 - X_2 = 0$ is the red line. $1 + 3X_1 - X_2 > 0$ is below this line and shaded in grey; $1 + 3X_1 - X_2 < 0$ is above this line and is not shaded.

(b) $-2 + X_1 + 2X_2 = 0$ is the blue line. $-2 + X_1 + 2X_2 < 0$ is below this line and shaded in slanted lines. $-2 + X_1 + 2X_2 > 0$ is above this line and is not shaded.

```
> curve(3*x + 1,-2,2,xlab = "X1",ylab = "X2")
> x = c(-3,3)
> y = 3*x + 1
> x = c(x, 3)
> y = c(y,-6)
> polygon(x,y,col = "grey",border = "red")

> abline(a = 1,b = -1/2,col = "red",lty = 1)
> x = c(-3,3)
> y = -1/2*x + 1
> x = c(-3,x,3)
> y = c(-10,y,-10)
> polygon(x,y,density = 20,border = "blue",lty = 1)
```



Q2.

(a) $(1+X_1)^2+(2-X_2)^2=4$ is a circle centered at $(-1,2)$ with radius 2.

(b) $(1+X_1)^2+(2-X_2)^2 \leq 4$ is shaded in grey. $(1+X_1)^2+(2-X_2)^2 > 4$ is not shaded.

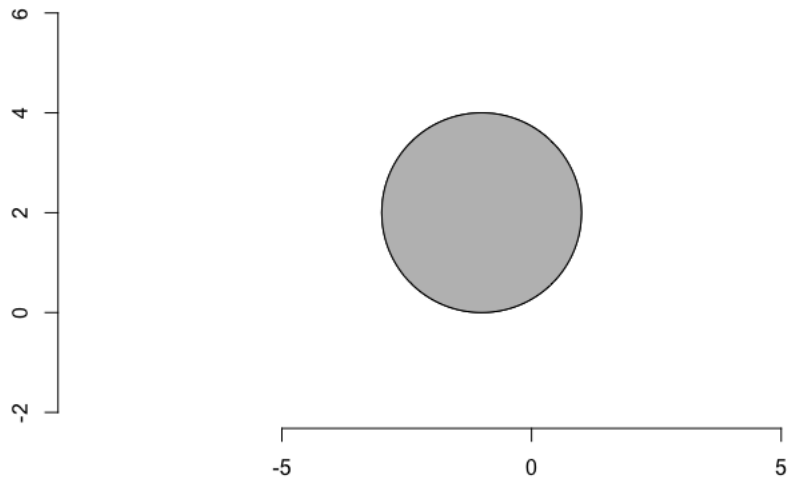
(c) blue; red; blue; blue.

(d) Clearly, the decision boundary is a circle and is not linear in X_1 and X_2. However,

$$(1+X_1)^2+(2-X_2)^2 = X_1^2+X_2^2+2X_1-4X_2+5$$

is a linear combination of X_1, X_2, X_1^2, X_2^2. Therefore, the decision boundary is linear in this higher dimensional space.

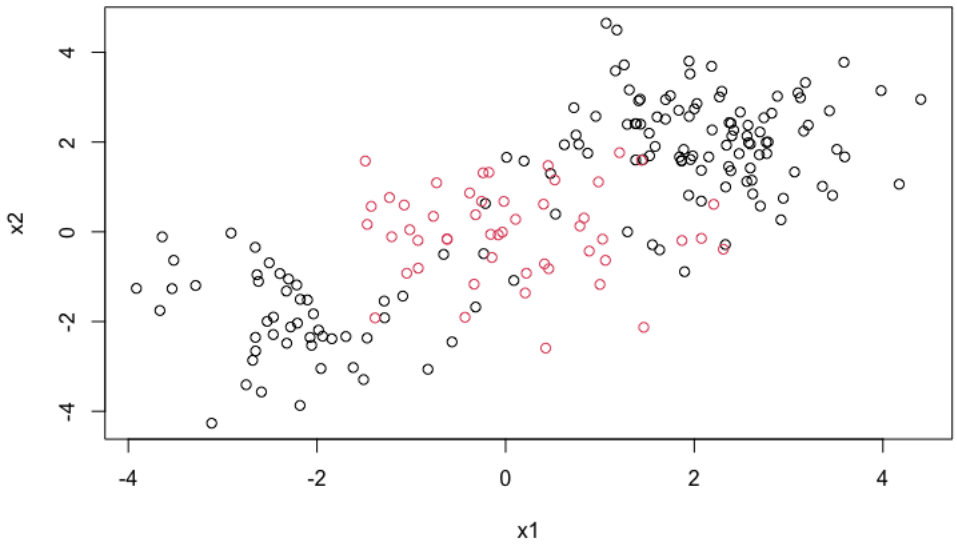
```
> library(plotrix)
> plot.new()
> plot.window(xlim = c(-4,2),ylim = c(-2,6),asp = 1)
> draw.circle(x = -1, y = 2,radius = 2)
> axis(1)
> axis(2)
> x = seq(-3,1,0.01)
> y = 2 - sqrt(4 - (1+x)^2)
> y2 = 2 + sqrt(4 - (1+x)^2)
> polygon(c(x,rev(x)),c(y,rev(y2)),col = "grey")
```



Q3.

(a)

```
> data.train <- read.csv("SVM_train.csv")
> data.test <- read.csv("SVM_test.csv")

> data.train$X = NULL
> data.test$X = NULL
> data.train$y = as.factor(data.train$y)
> data.test$y = as.factor(data.test$y)

> plot(data.train$x.2 ~ data.train$x.1, col = data.train$y, xlab
= "x1", ylab = "x2")
```



Visually separable, however it's not perfectly separable and the decision boundary is nonlinear.

(b) Choose cost C by cross validation:

```
> library(e1071)
> tune.linear = tune(svm ,y~.,data=data.train ,kernel ="linear",
ranges =list(cost= c(0.001, 0.01, 0.05, 0.1, 0.5, 1, 5, 10,
100)))
```

Check the selection results:

```
> summary(tune.linear)

Parameter tuning of ‘svm’:
```

```
- sampling method: 10-fold cross validation

- best parameters:
  cost
 0.001

- best performance: 0.25

- Detailed performance results:
   cost error dispersion
1 1e-03  0.25 0.08498366
2 1e-02  0.25 0.08498366
3 5e-02  0.25 0.08498366
4 1e-01  0.25 0.08498366
5 5e-01  0.25 0.08498366
6 1e+00  0.25 0.08498366
7 5e+00  0.25 0.08498366
8 1e+01  0.25 0.08498366
9 1e+02  0.25 0.08498366
```

Choose the best model (classifier with optimal C):

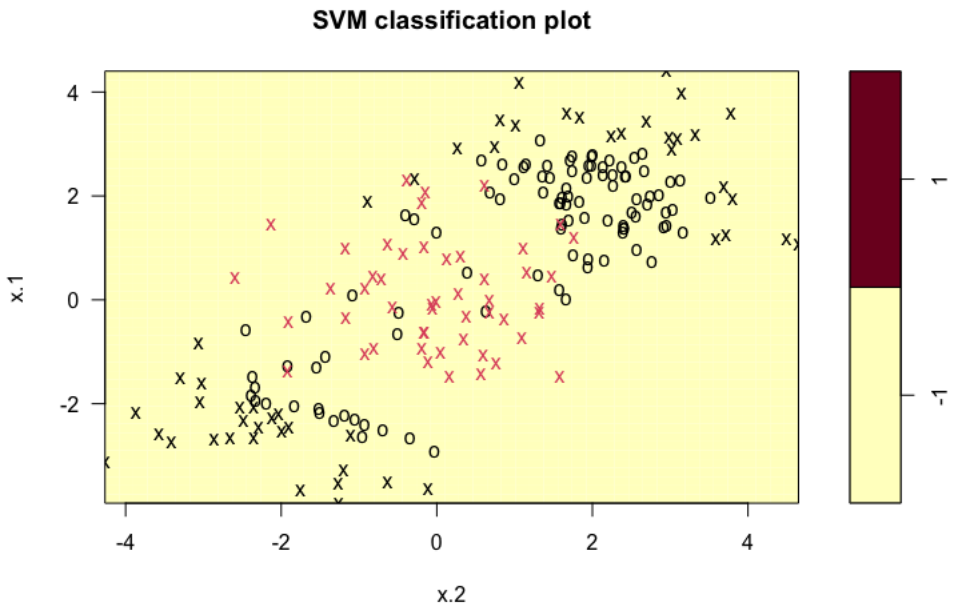
```
> bestmod.linear = tune.linear$best.model
```

Check the classifier:

```
> summary(bestmod.linear)

Call:
best.tune(method = svm, train.x = y ~ ., data = data.train,
ranges = list(cost = c(0.001,
    0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100)), kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.001

Number of Support Vectors:  101

 ( 51 50 )


Number of Classes:  2

Levels:
 -1 1
```

Plot the classifier on the training set:

```
> plot(bestmod.linear, data.train)
```



Prediction on test set:

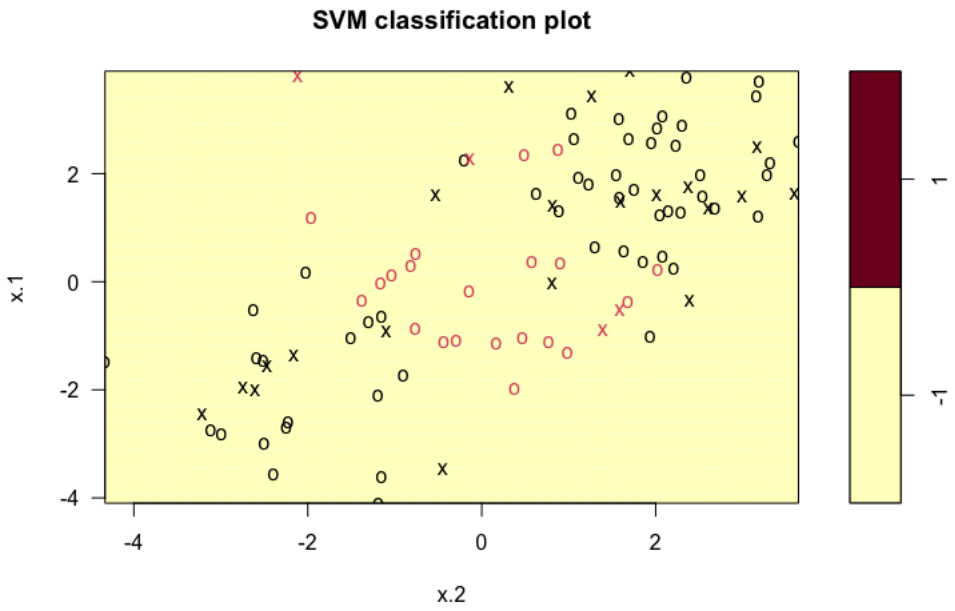
```
> ypred.linear=predict(bestmod.linear, data.test)
```

Check the prediction accuracy:

```
> table(predict = ypred.linear, truth = data.test$y)
       truth
predict -1  1
     -1 75 25
     1   0  0
> mean(ypred.linear == data.test$y)
[1] 0.75
```

Plot the classifier on the test set

```
> plot(bestmod.linear, data.test)
```



(c) Choose cost C and γ by cross validation:

```
> tune.gaussian = tune(svm, y~., data=data.train, kernel
="radial",
+                      ranges =list(cost=c(0.1,
0.5,1 ,1.5 ,2 ,5), gamma=c(0.1,0.5,1,1.5,2) ))
```

Check the selection results:

```
> summary(tune.gaussian)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  1.5   0.1

- best performance: 0.105

- Detailed performance results:
   cost gamma error dispersion
1   0.1   0.1 0.250 0.11303883
2   0.5   0.1 0.250 0.11303883
3   1.0   0.1 0.140 0.09368980
4   1.5   0.1 0.105 0.05986095
5   2.0   0.1 0.105 0.05986095
6   5.0   0.1 0.115 0.07090682
```

```
7   0.1   0.5 0.145 0.10658851
8   0.5   0.5 0.115 0.07090682
9   1.0   0.5 0.110 0.06992059
10  1.5   0.5 0.110 0.06992059
11  2.0   0.5 0.115 0.07090682
12  5.0   0.5 0.110 0.06582806
13  0.1   1.0 0.120 0.08881942
14  0.5   1.0 0.120 0.07149204
15  1.0   1.0 0.120 0.07149204
16  1.5   1.0 0.120 0.05868939
17  2.0   1.0 0.120 0.05868939
18  5.0   1.0 0.115 0.05797509
19  0.1   1.5 0.110 0.07378648
20  0.5   1.5 0.120 0.07149204
21  1.0   1.5 0.120 0.07149204
22  1.5   1.5 0.120 0.05868939
23  2.0   1.5 0.120 0.05868939
24  5.0   1.5 0.130 0.05374838
25  0.1   2.0 0.130 0.07527727
26  0.5   2.0 0.130 0.07149204
27  1.0   2.0 0.120 0.07149204
28  1.5   2.0 0.120 0.05868939
29  2.0   2.0 0.120 0.05868939
30  5.0   2.0 0.125 0.06346478
```

Choose the best model (classifier with optimal C and γ):

```
> bestmod.gaussian =tune.gaussian$best.model
```

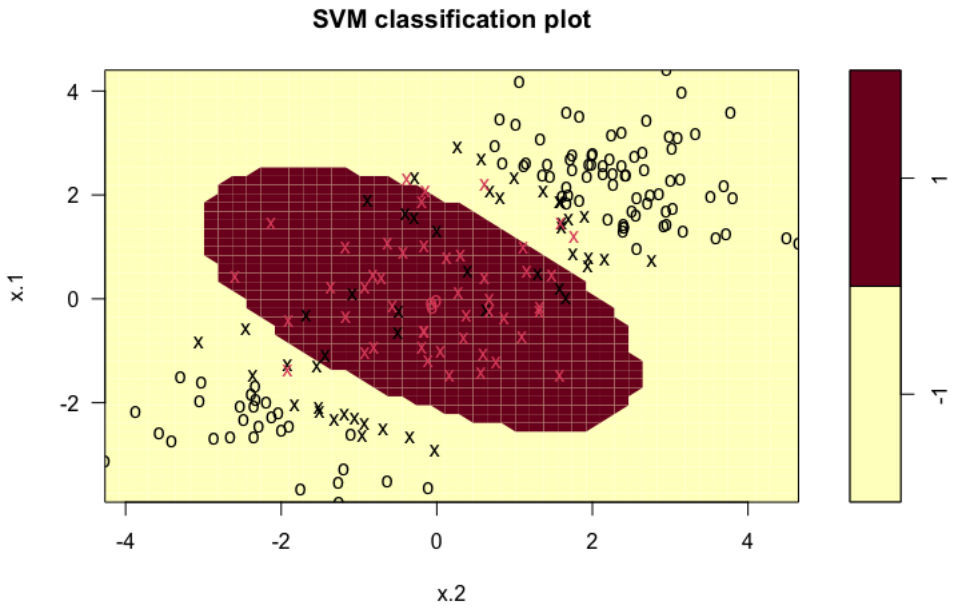
Check the classifier:

```
> summary(bestmod.gaussian)

Call:
best.tune(method = svm, train.x = y ~ ., data = data.train,
ranges = list(cost = c(0.1,
    0.5, 1, 1.5, 2, 5), gamma = c(0.1, 0.5, 1, 1.5, 2)), kernel
= "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1.5

Number of Support Vectors:  95

 ( 48 47 )
```

```
Number of Classes:  2

Levels:
 -1 1
```

Plot the classifier on the training set:

```
> plot(bestmod.gaussian, data.train)
```



Prediction on test set:

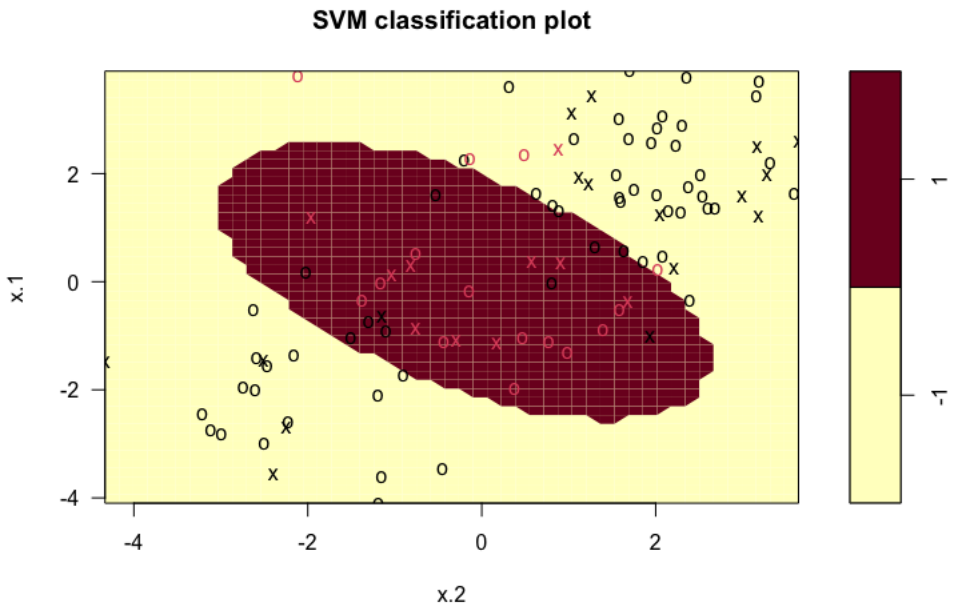
```
> ypred.gaussian=predict(bestmod.gaussian, data.test)
```

Check the prediction accuracy:

```
> table(predict=ypred.gaussian, truth= data.test$y)
       truth
predict -1  1
     -1 60  4
     1  15 21
> mean(ypred.gaussian == data.test$y)
[1] 0.81
```

Plot the classifier on the test set:

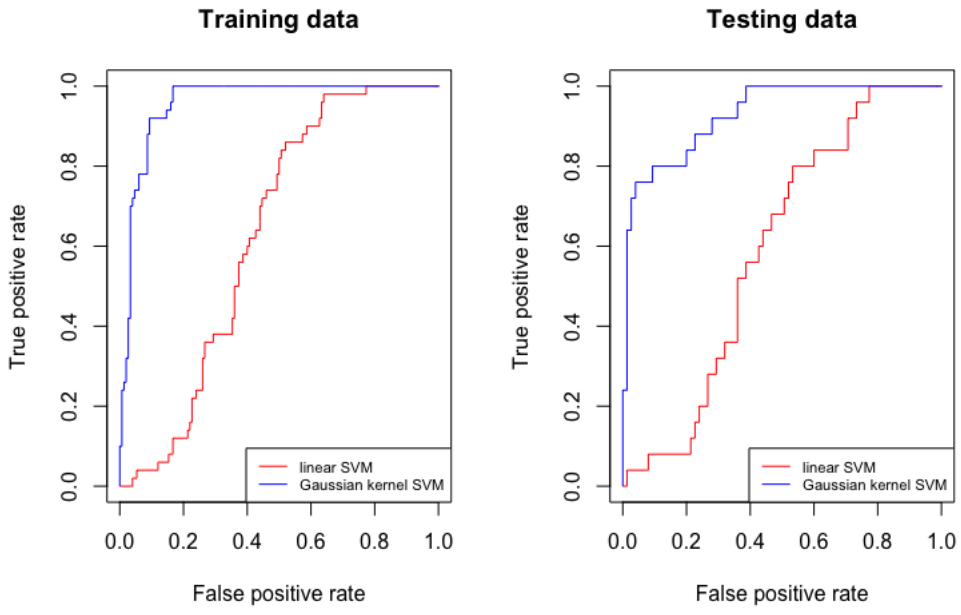
```
> plot(bestmod.gaussian, data.test)
```



(d) We make ROC plot using the following codes:

```
> library(ROCR)
> rocplot=function(pred, truth, ...){
+   predob = prediction(pred, truth)
+   perf = performance(predob, "tpr", "fpr")
+   plot(perf,...)}
> par(mfrow=c(1,2))

> fitted.linear.train = attributes(predict(bestmod.linear,
data.train, decision.values=TRUE))$decision.values
> rocplot(-fitted.linear.train, data.train$y, main="Training
data", col = "red")

> fitted.gaussian.train = attributes(predict(bestmod.gaussian,
data.train, decision.values=TRUE))$decision.values
> rocplot(-fitted.gaussian.train, data.train$y, add = TRUE, col
= "blue")

> legend("bottomright",legend = c("linear SVM","Gaussian kernel
SVM"),col = c("red","blue"),lty = c(1,1), cex  = 0.7)
>
> library(ROCR)
> rocplot=function(pred, truth, ...){
+     predob = prediction(pred, truth)
+     perf = performance(predob, "tpr", "fpr")
+     plot(perf,...)}
> par(mfrow=c(1,2))
```

```
> fitted.linear.train = attributes(predict(bestmod.linear,
data.train, decision.values=TRUE))$decision.values
> rocplot(-fitted.linear.train, data.train$y, main="Training
data", col = "red")

> fitted.gaussian.train = attributes(predict(bestmod.gaussian,
data.train, decision.values=TRUE))$decision.values
> rocplot(-fitted.gaussian.train, data.train$y, add = TRUE, col
= "blue")

> legend("bottomright",legend = c("linear SVM","Gaussian kernel
SVM"),col = c("red","blue"),lty = c(1,1), cex  = 0.7)

> fitted.linear.test = attributes(predict(bestmod.linear,
data.test, decision.values=TRUE))$decision.values
> rocplot(-fitted.linear.test, data.test$y, main="Testing data",
col = "red")

> fitted.gaussian.test = attributes(predict(bestmod.gaussian,
data.test, decision.values=TRUE))$decision.values
> rocplot(-fitted.gaussian.test, data.test$y, add = TRUE, col =
"blue")

> legend("bottomright",legend = c("linear SVM","Gaussian kernel
SVM"),col = c("red","blue"),lty = c(1,1), cex  = 0.7)
```



For the linear SVM, ROC plots on the training and test set indicate that it is slightly better than random guess. For the Gaussian kernel SVM, the classifier performs well on both the training and test set. This is expected because the two datasets are not linear separable.