Andrew Rothman
anr248@stanford.edu

December 7th, 2018
SUID: 06325118

Problem #1:

Exercise 2 from section 9.7 of ITSL textbook

We have seen that in $p = 2$ dimensions, a linear decision boundary takes the form

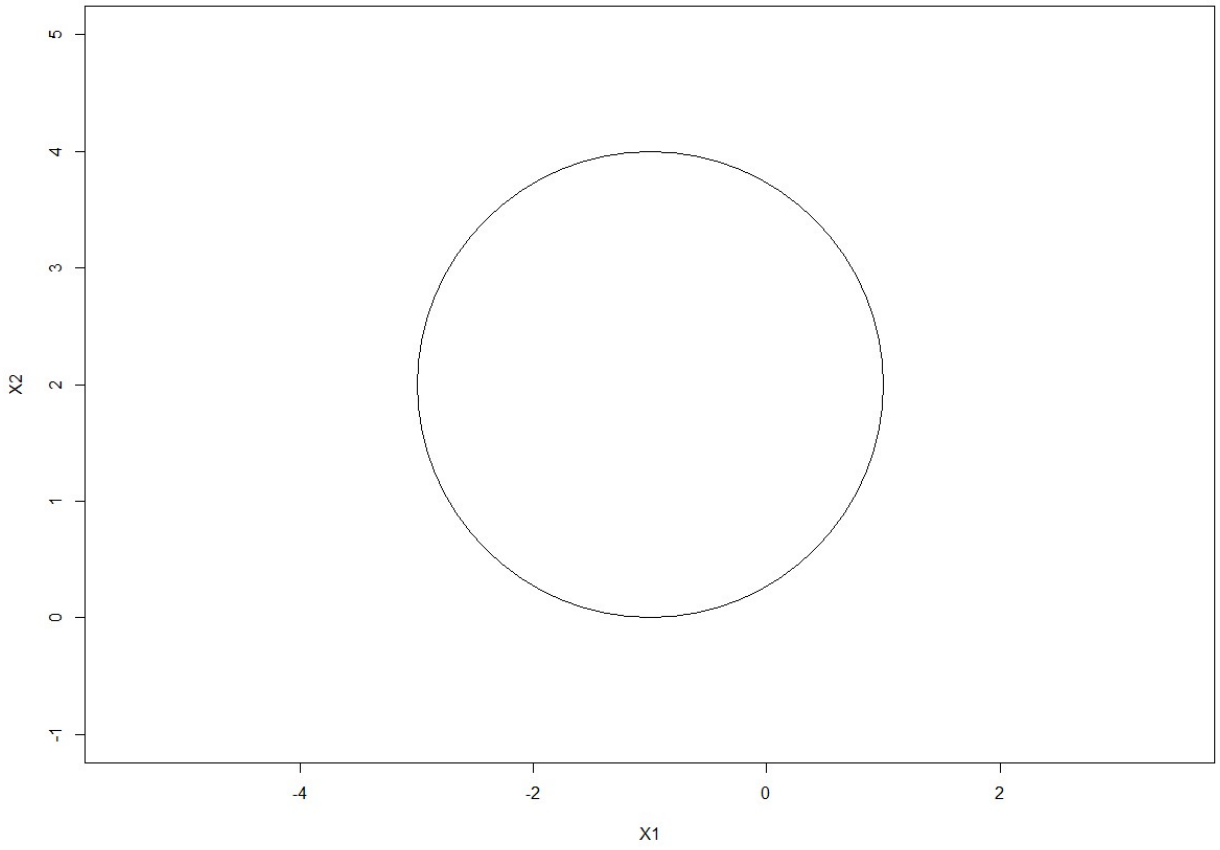
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

We now investigate a non-linear decision boundary.

(a) Sketch the curve

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

Answer:



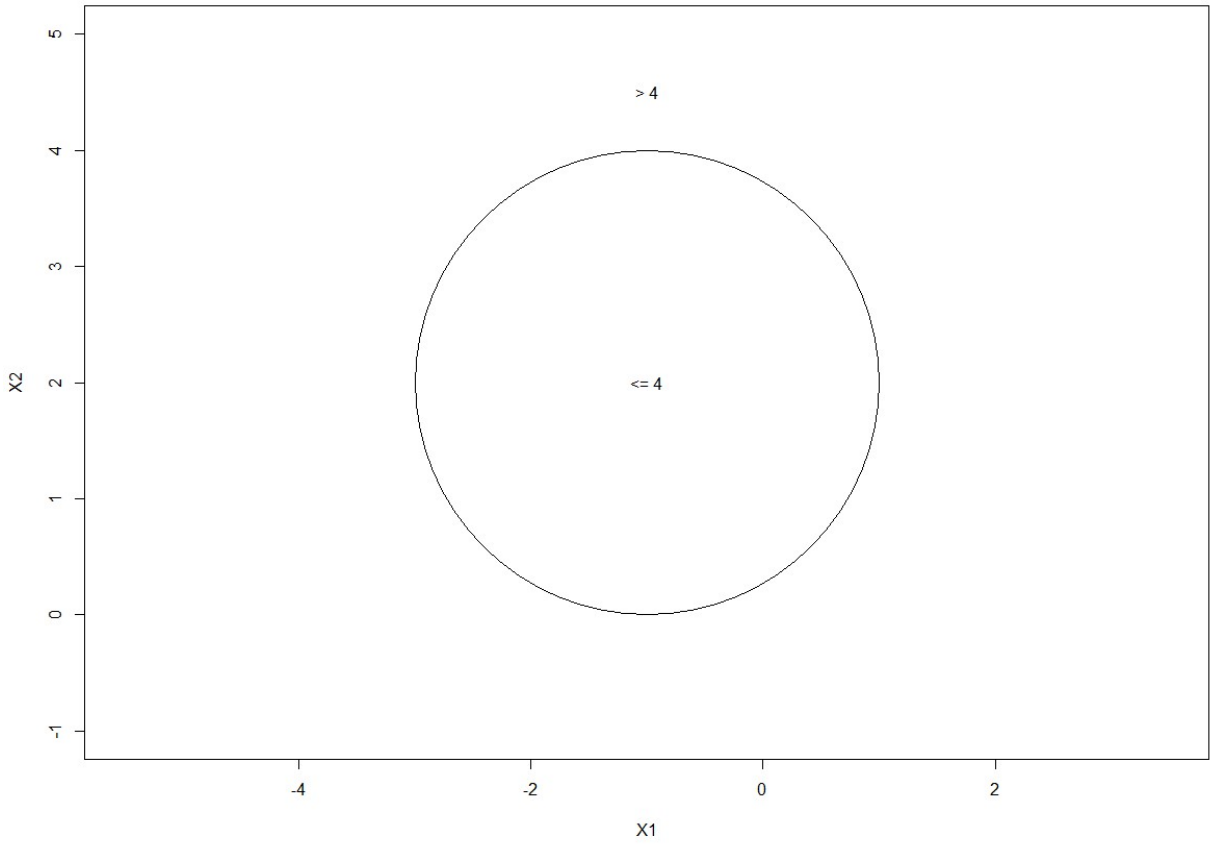
(b) On your sketch, indicate the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 > 4$$

as well as the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 \leq 4$$

Answer:

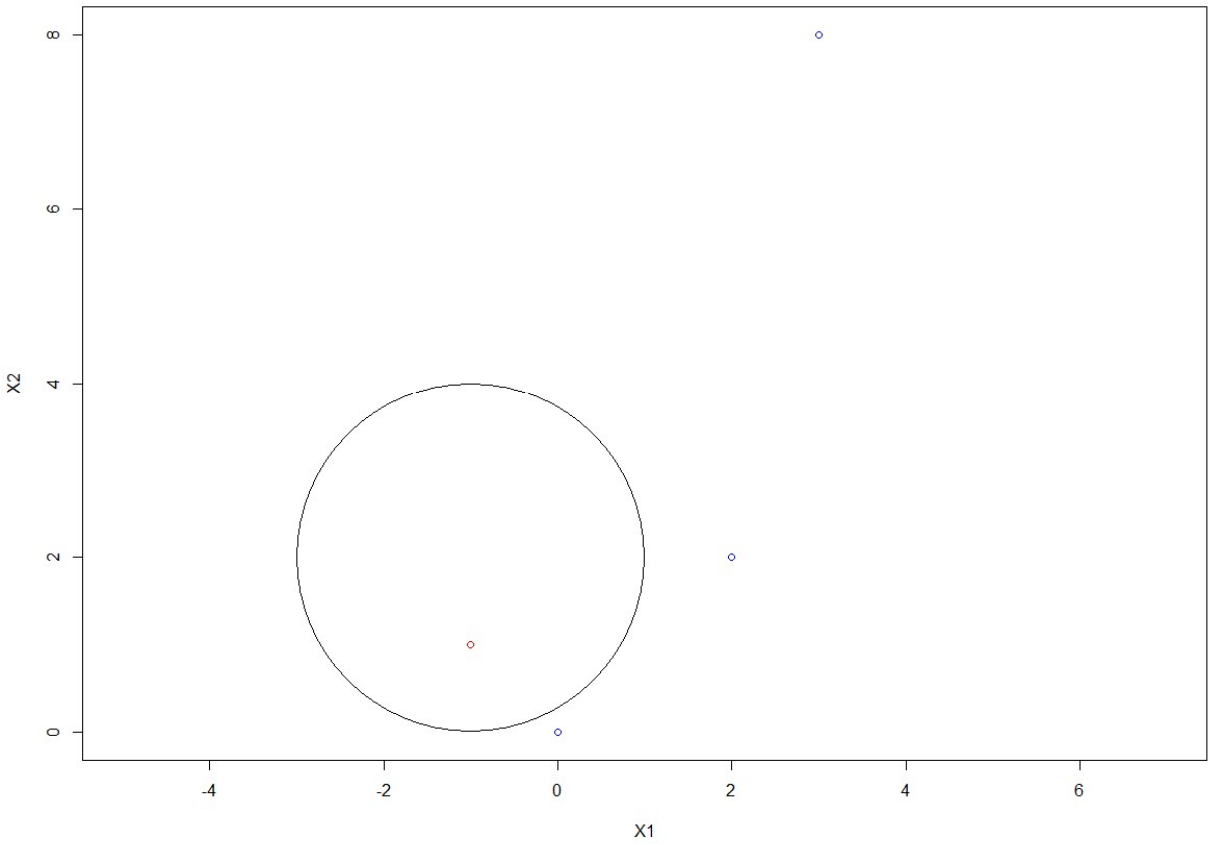


(c) Suppose that a classifier assigns an observation to the blue class if:

$$(1 + X_1)^2 + (2 - X_2)^2 > 4$$

and to the red class otherwise. To what class is the observation (0, 0) classified? (-1, 1)? (2, 2)? (3, 8)?

Answer:



Points (0,0), (2,2), and (3,8) are all in the blue class. The point (-1,1) is in the red class.

(d) Argue that while the decision boundary in (c) is not linear in terms of X_1 and X_2, it is linear in terms of X_1, ${X_1}^2$, X_2, and ${X_2}^2$

Answer:

If we expand the decision boundary specified in part (c), we can see that it is a linear function of X_1, ${X_1}^2$, X_2, and ${X_2}^2$:

$$(1 + X_1)^2 + (2 - X_2)^2 > 4$$
$$1 + 2X_1 + {X_1}^2 + 4 - 4X_2 + {X_2}^2 > 4$$
$$2X_1 + {X_1}^2 - 4X_2 + {X_2}^2 > -1$$

This means the decision boundary is therefore linear in terms of X_1, ${X_1}^2$, X_2, and ${X_2}^2$

R-code:

```
#################
## Problem #1 ##
#################

#############
## Part A ##
#############
radius = 2
plot(NA, NA, type="n", xlim=c(-4,2), ylim=c(-1,5), asp=1, xlab="X1", ylab="X2")
symbols(c(-1), c(2), circles=c(radius), add=TRUE, inches=FALSE)

#############
## Part B ##
#############
radius = 2
plot(NA, NA, type="n", xlim=c(-4,2), ylim=c(-1,5), asp=1, xlab="X1", ylab="X2")
symbols(c(-1), c(2), circles=c(radius), add=TRUE, inches=FALSE)
text(c(-1), c(2), "<= 4")
text(c(-1), c(4.5), "> 4")

#############
## Part C ##
#############
radius = 2
plot(c(0,-1,2,3), c(0,1,2,8), col=c("blue","red","blue","blue"), type="p", asp=1, xlab="X1",
ylab="X2")
symbols(c(-1), c(2), circles=c(radius), add=TRUE, inches=FALSE)
```

Problem #2:

Exercise 4 from section 9.7 of ITSL textbook

Generate a simulated two-class data set with 100 observations and two features in which there is a visible but non-linear separation between the two classes. Show that in this setting, a support vector machine with a polynomial kernel (with degree great than 1) or a radial kernel will outperform a support vector classifier on the training data. Which technique performs best on the test data? Make plots and report training and test error rates in order to back up your assertions.

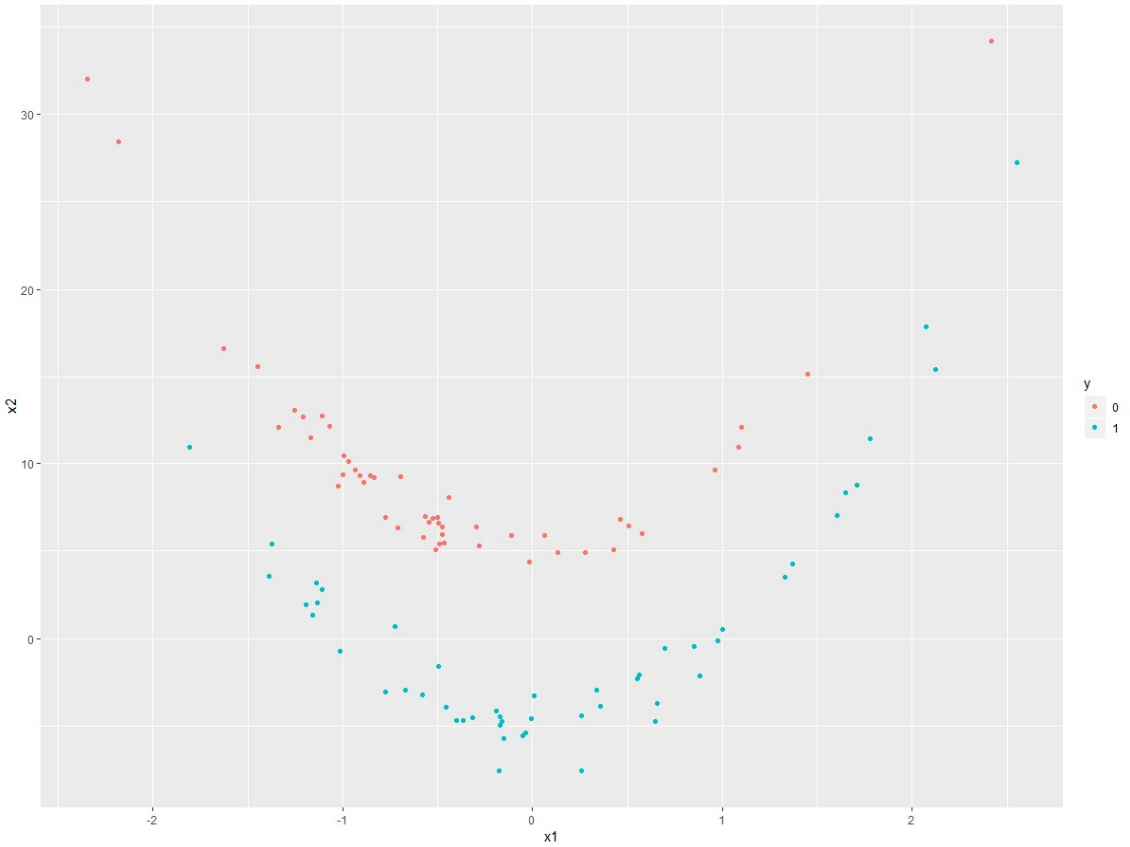
Answer:

Let us begin by creating a simulated dataset of 100 observations. I specified two continuous features (X_1 and X_2) and binary outcome Y. I specified the following non-linear relationship between X_1 and X_2 conditional on the value of Y:

$$X_2 = 5{X_1}^2 + 5 + \epsilon \quad \text{, for Y=0}$$
$$X_2 = 5{X_1}^2 - 5 + \epsilon \quad \text{, for Y=1}$$

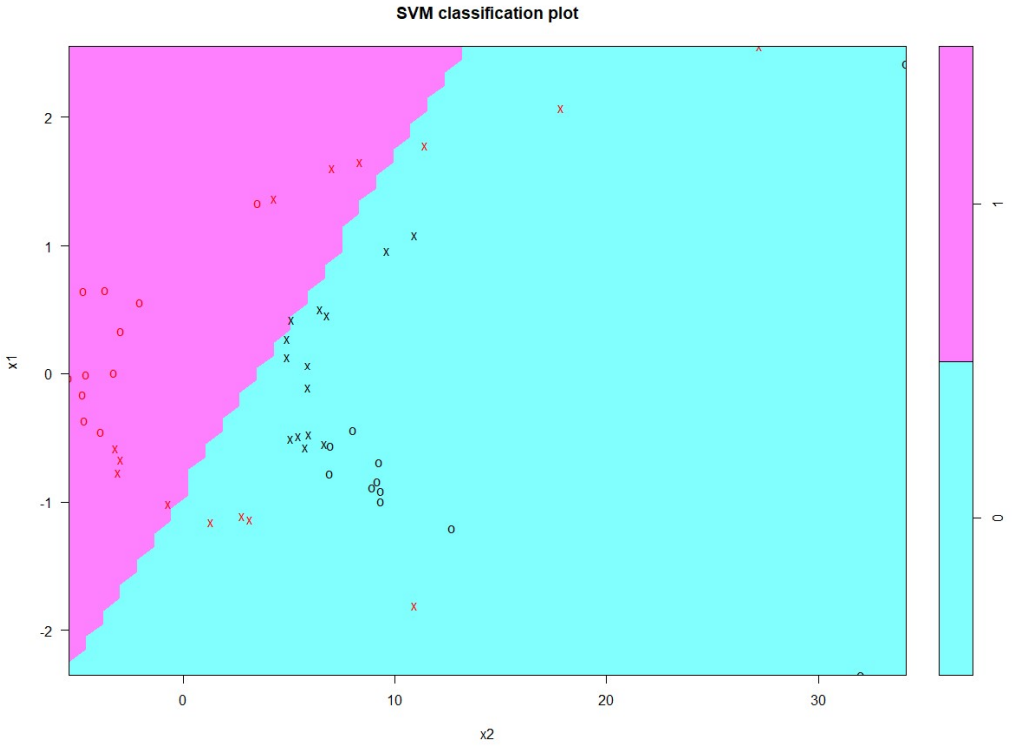
with error term $\epsilon \sim N(0,1)$

A plot of the datapoints (colored by the Y class) in the X_1, X_2 plane is shown below:



We then split the simulated data set into separate but equal sized training and testing sets (n=50 each). In each of the training and testing sets, 25 of the observations were drawn from the Y=0 class, and the remaining 25 observations drawn from the Y=1 class.

Let us now fit a linear support vector classifier to the 50 observations in the training set. A plot of the decision boundary estimated by the linear support vector classifier on the training set is shown below:



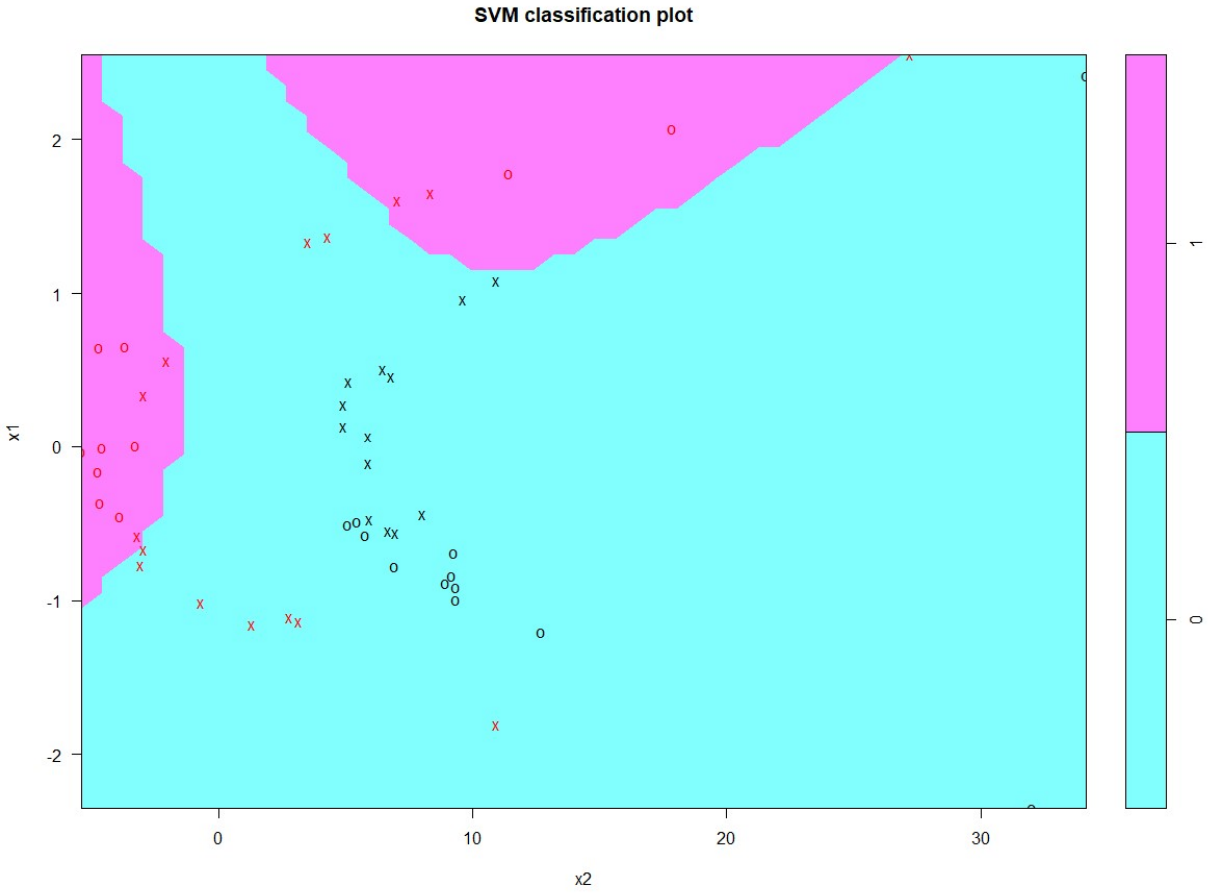
As we can see above, the resulting decision boundary estimated by the linear support vector classifier is linear in the X_1,X_2 plane. The confusion matrix for the plot above is shown below:

Confusion Matrix: Linear Support Vector Classifier on Training Set

		Predicted Class	
		Y=0	Y=1
Actual Class	Y=0	23	2
	Y=1	7	18

As per the confusion matrix above, the accuracy of the linear support vector classifier on the training set is (41/50)=82%

Let us now try fitting a support vector machine with a polynomial kernel of degree 3 and gamma parameter=0.5 on the training set. A plot of the estimated polynomial kernel SVM on the training data is shown below, along with the confusion matrix:

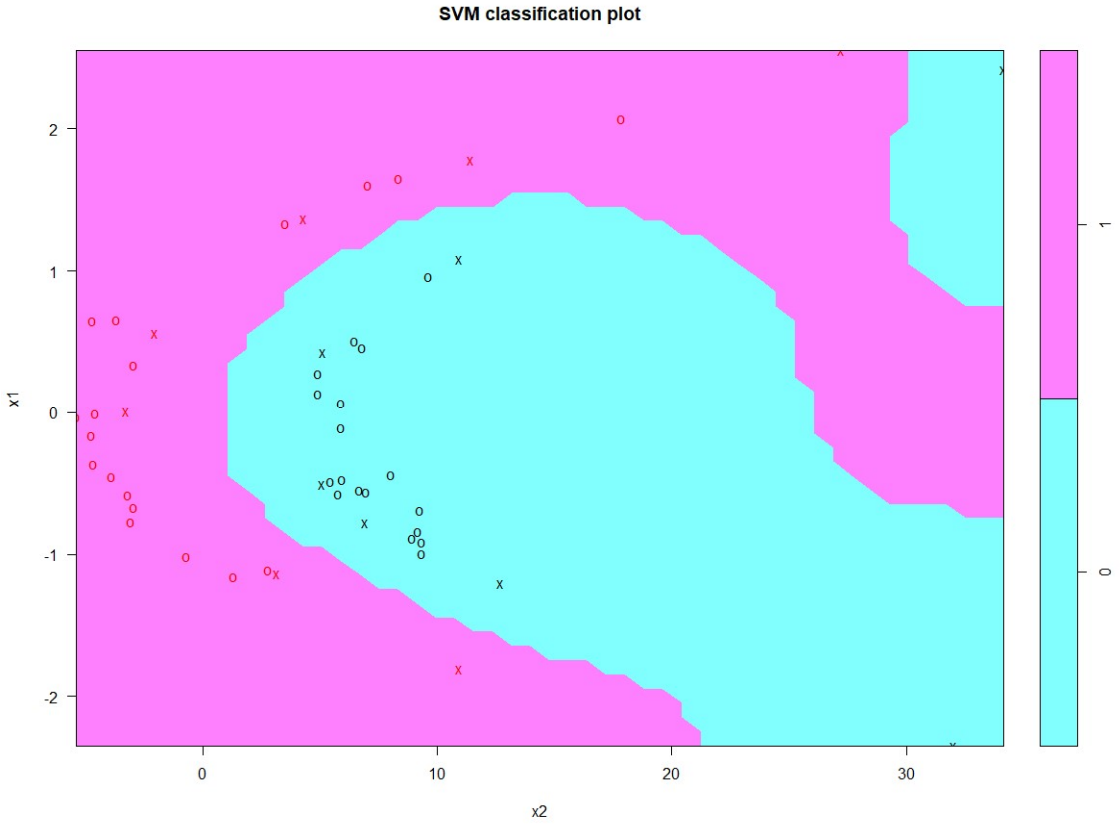


Confusion Matrix: SVM with polynomial kernel degree 3 on Training Set

		Predicted Class: Y=0	Predicted Class: Y=1
Actual Class	Y=0	25	0
	Y=1	9	16

as per the confusion matrix above, the accuracy of the SVM with a polynomial kernel on the training set is (41/50)=82%. This is the same as the accuracy of the linear support vector classifier on the training set. Let's see if we can specify a SVM with a radial kernel to improve the accuracy on the training set:

Next a support vector machine with a radial kernel and gamma parameter=01 was fit assessed on the training set. A plot of the estimated radial kernel SVM on the training data is shown below, along with the confusion matrix:



Confusion Matrix: SVM with radial kernel on Training Set

		Predicted Class: Y=0	Predicted Class: Y=1
Actual Class	Y=0	25	0
	Y=1	0	25

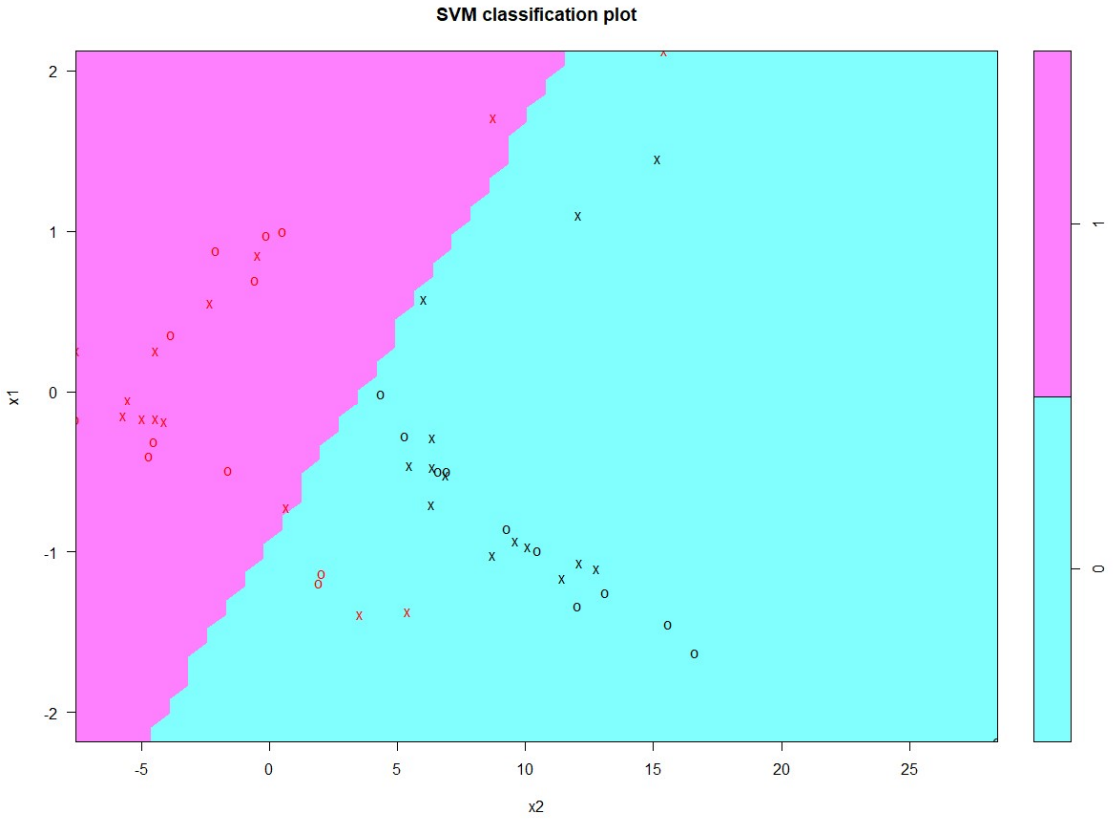
From the confusion matrix above, we see the accuracy on the training set is 100% (there is no misclassification of the Y class by the SVM with radial kernel). Therefore we have shown that a SVM with radial kernel outperformed a linear support vector classifier on the training data.

We fit three different models so far:

- Linear Support Vector Classifier
- SVM with polynomial kernel of degree 3
- SVM with radial kernel

Let us now assess how each of these models (that were estimated using the training data) perform on the testing set.

Let us start with the Linear Support Vector Classifier:

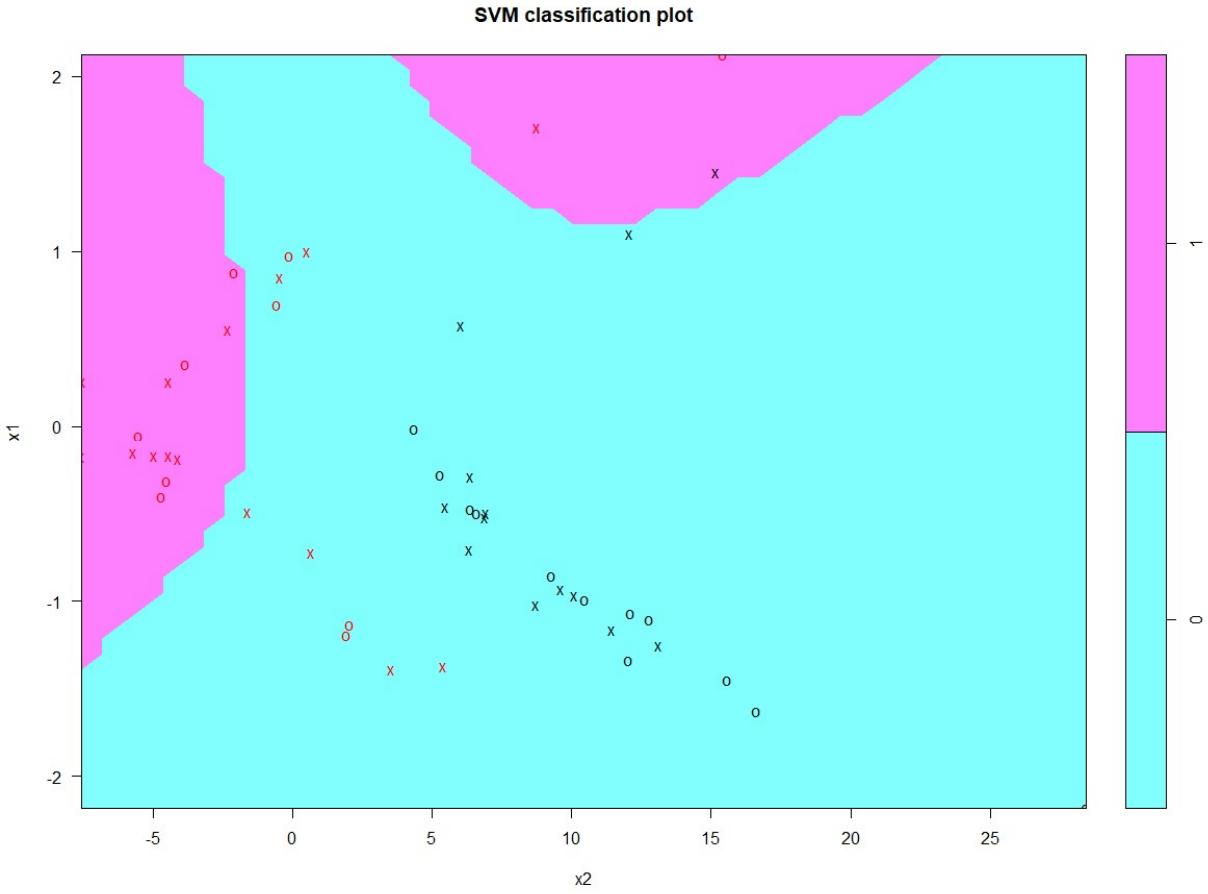


Confusion Matrix: Linear Support Vector Classifier on Testing Set

		Predicted Class	
		Y=0	Y=1
Actual Class	Y=0	24	1
	Y=1	5	20

From the confusion matrix above, the accuracy of the linear support vector classifier on the testing set is 88%, which is actually an improvement over the accuracy of this model on the training set (training set accuracy=82%).

The SVM with polynomial kernel degree 3:

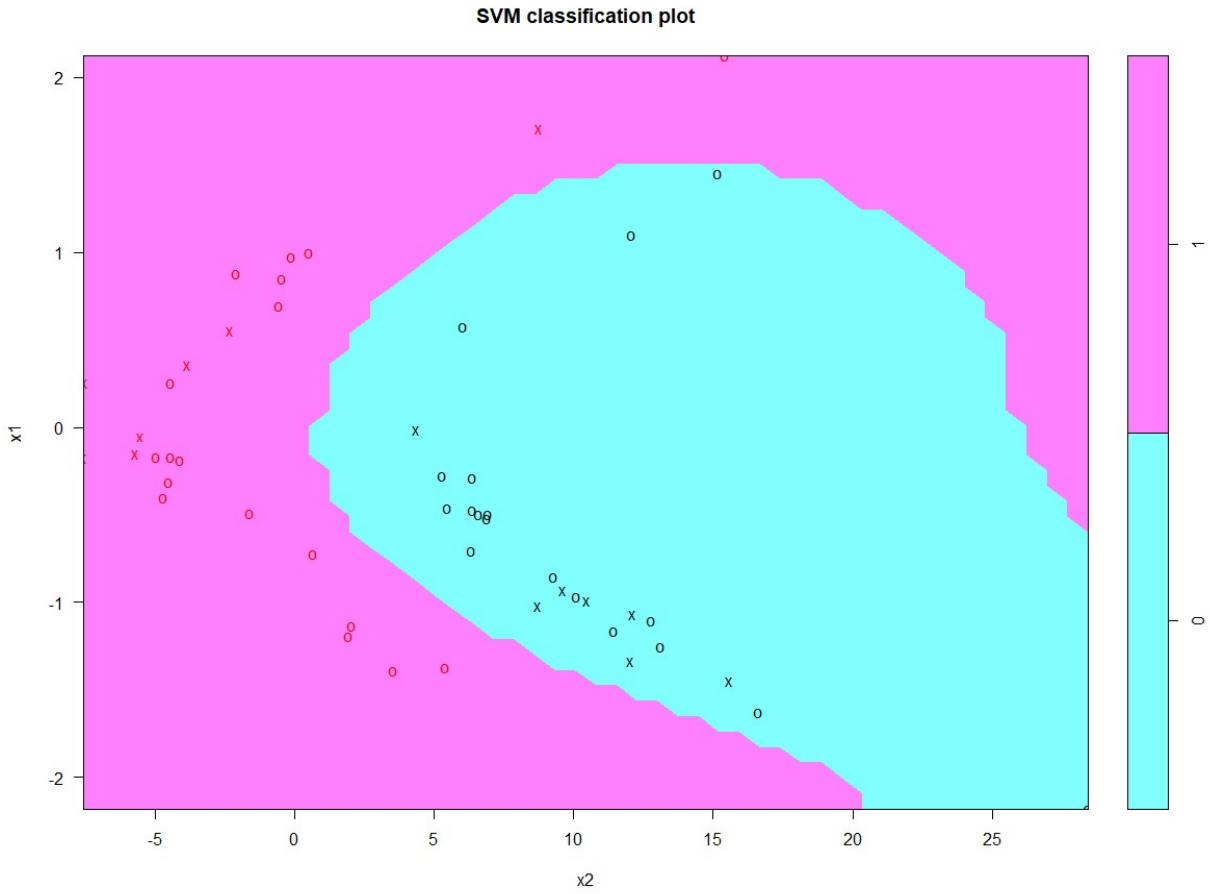


Confusion Matrix: SVM with polynomial kernel degree 3 on Testing Set

		Predicted Class: Y=0	Predicted Class: Y=1
Actual Class	Y=0	24	1
	Y=1	10	15

The accuracy of the SVM with polynomial kernel degree 3 on the testing data is 78%, which is lower than when assessed on the training data (training set accuracy=82%).

And lastly we have the SVM with radial kernel:



Confusion Matrix: SVM with radial kernel on Testing Set

		Predicted Class: Y=0	Predicted Class: Y=1
Actual Class	Y=0	25	0
	Y=1	0	25

From the confusion matrix above, the accuracy of the SVM with radial kernel on the testing set is 100%, the same perfect accuracy as when assessing on the training set.

From the three models fit (linear support vector classifier, SVM with polynomial kernel, SVM with radial kernel), we can see the highest testing accuracy of 100% was attained with the SVM with radial kernel. We can therefore conclude that the SVM with radial kernel performed best at estimated the non-linear decision boundary between the two Y-classes in the X_1,X_2 plane in our simulated data set.

R-code:

```
#################
## Problem #2 ##
#################
library(dplyr)
library(ggplot2)
set.seed(1234)
id <- c(1:100)
df <- data.frame(id)
df$x1 <- rnorm(100)
df$error <- rnorm(100)
df$x2 <- 5*((df$x1)^2) + df$error
df$x2[df$id<=50] <- df$x2[df$id<=50]+5
df$x2[df$id>50] <- df$x2[df$id>50]-5
df$y <- 0
df$y[df$id>50] <- 1
df$y <- as.factor(df$y)

##########################
## Plot the data points ##
##########################
ggplot(df, aes(x=x1, y=x2, color=y)) + geom_point()

#####################################
## create training and testing sets ##
#####################################
train_df <- filter(df, (id>=1 & id<=25)|(id>50 & id<=75))
test_df <- filter(df, (id>25 & id<=50)|(id>75))

###############################
## Suuport Vector Classifier ##
###############################
library(e1071)
svc_linear <- svm(y~., data=select(train_df, y, x1, x2), kernel="linear", cost=10)

## training set
plot(svc_linear, select(train_df, y, x1, x2))
train_df$svc_linear_p <- predict(svc_linear, select(train_df, y, x1, x2))
table(train_df$y, train_df$svc_linear_p)

## testing set
plot(svc_linear, select(test_df, y, x1, x2))
test_df$svc_linear_p <- predict(svc_linear, select(test_df, y, x1, x2))
table(test_df$y, test_df$svc_linear_p)


###########################
## SVM polynomial kernel ##
###########################
set.seed(1)
svm_poly <- svm(y~., data=select(train_df, y, x1, x2), kernel="polynomial", cost=10)

## training set
plot(svm_poly, select(train_df, y, x1, x2))
train_df$svm_poly_p <- predict(svm_poly, select(train_df, y, x1, x2))
table(train_df$y, train_df$svm_poly_p)

## testing set
plot(svm_poly, select(test_df, y, x1, x2))
test_df$svm_poly_p <- predict(svm_poly, select(test_df, y, x1, x2))
table(test_df$y, test_df$svm_poly_p)
```

```
######################
## SVM radial kernel ##
######################
set.seed(1)
svm_radial <- svm(y~., data=select(train_df, y, x1, x2), kernel="radial", gamma=1, cost=10)

## training set
plot(svm_radial, select(train_df, y, x1, x2))
train_df$svm_radial_p <- predict(svm_radial, select(train_df, y, x1, x2))
table(train_df$y, train_df$svm_radial_p)

## testing set
plot(svm_radial, select(test_df, y, x1, x2))
test_df$svm_radial_p <- predict(svm_radial, select(test_df, y, x1, x2))
table(test_df$y, test_df$svm_radial_p)
```

Problem #3:

Exercise 5 from section 9.7 of ITSL textbook

We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.

(a) Generate a data set with n=500 and p=2, such that observations belong to two classes with a quadratic decision boundary between them.

Answer:

Let us begin by creating a simulated dataset of 500 observations. I specified two continuous features (X_1 and X_2) and binary outcome Y. I specified the following quadratic relationship between X_1 and X_2 conditional on the value of Y:

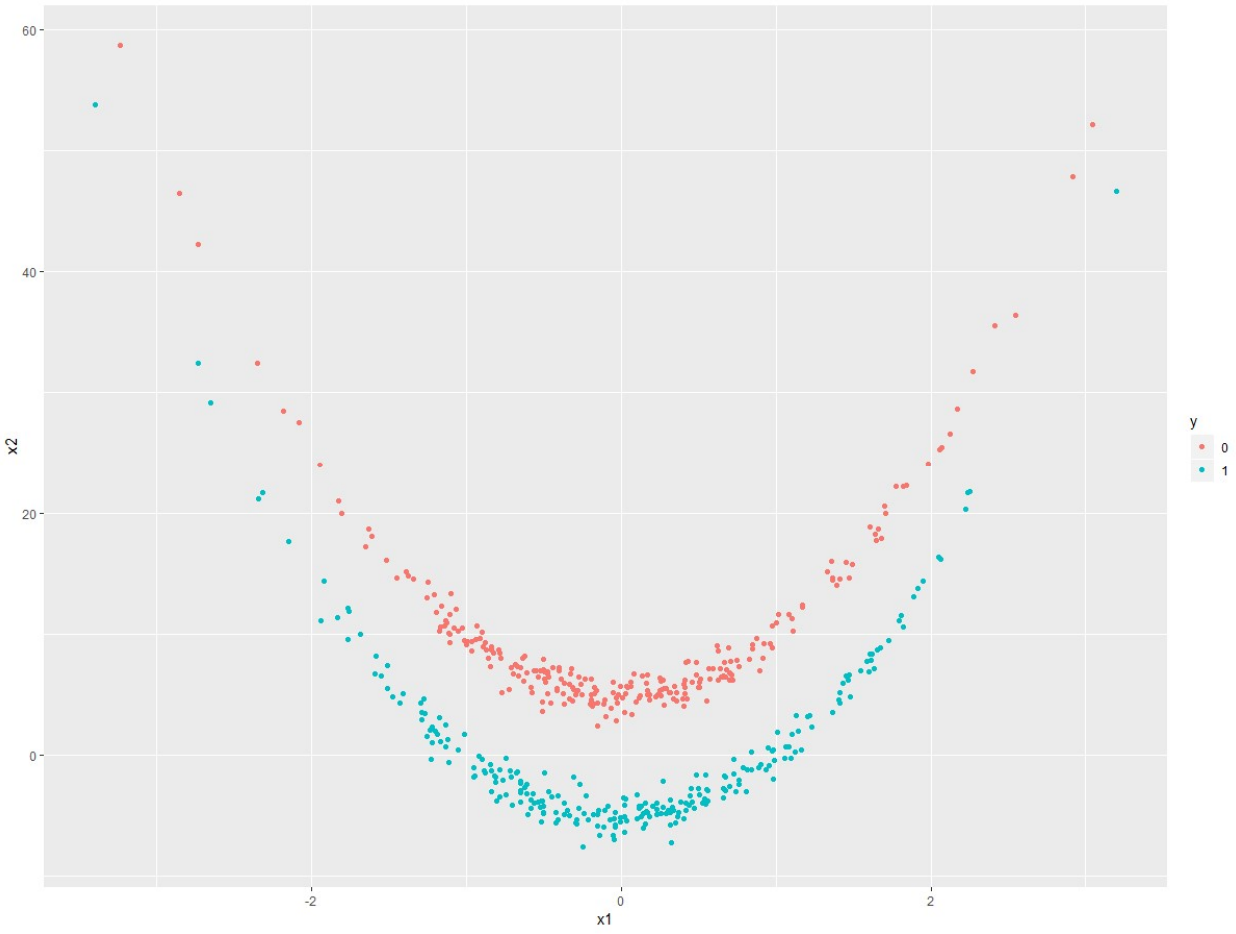
$$X_2 = 5{X_1}^2 + 5 + \epsilon \quad \text{, for Y=0}$$
$$X_2 = 5{X_1}^2 - 5 + \epsilon \quad \text{, for Y=1}$$

with error term $\epsilon \sim N(0,1)$

Of the 500 observations simulated for this dataset, 250 observations are in class Y=0, and the remaining 250 observations in class Y=1.

(b) Plot the observations, colored according to their class labels. Your plot should display X_1 on the x-axis, and X_2 on the y-axis.

Answer:



(c) Fit a logistic regression model to the data, using X_1 and X_2 as predictors.

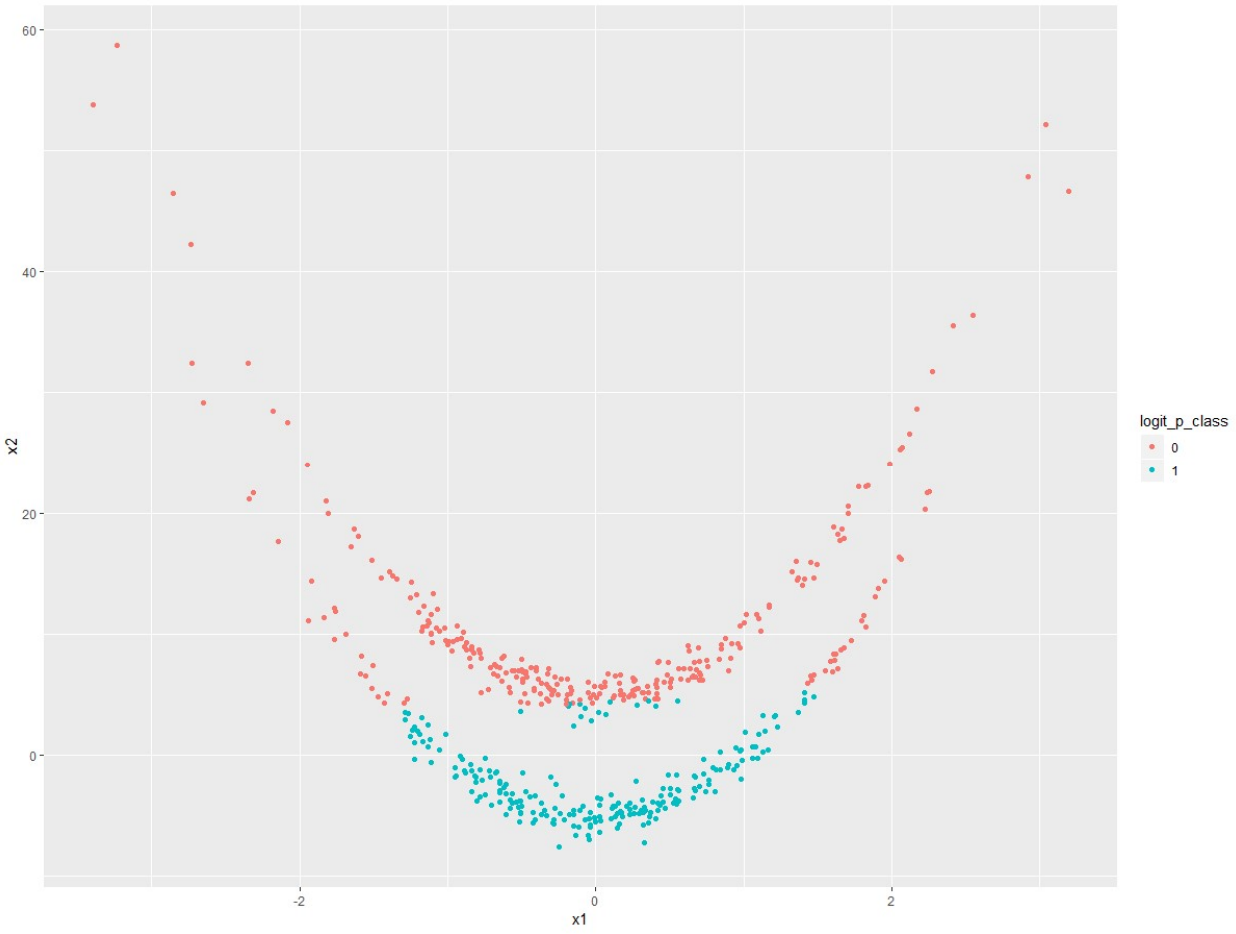
Answer:
The summary output of the fit logistic regression model to the data is shown below:

```
Call:
glm(formula = y ~ ., family = "binomial", data = select(df, y,
    x1, x2))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3508  -0.9215   0.1864   0.6519   4.7836

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.9649     0.1398   6.901 5.17e-12 ***
x1            0.1433     0.1300   1.102     0.27
x2           -0.2218     0.0214 -10.368  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 693.15  on 499  degrees of freedom
Residual deviance: 501.66  on 497  degrees of freedom
AIC: 507.66

Number of Fisher Scoring iterations: 5
```

(d) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be linear.

Answer:

The resulting plot is shown below. As we can see, the estimated decision boundary from the linear regression model appears linear in the X_1,X_2 plane (looking at the colored classes, the transition from the red to blue points is from a linear decision boundary at approximately $X_2 = 5$).



(e) Now fit a logistic regression model to the data using non-linear functions of X_1 and X_2 as predictors.

Answer:

The following logistic regression model was fit:

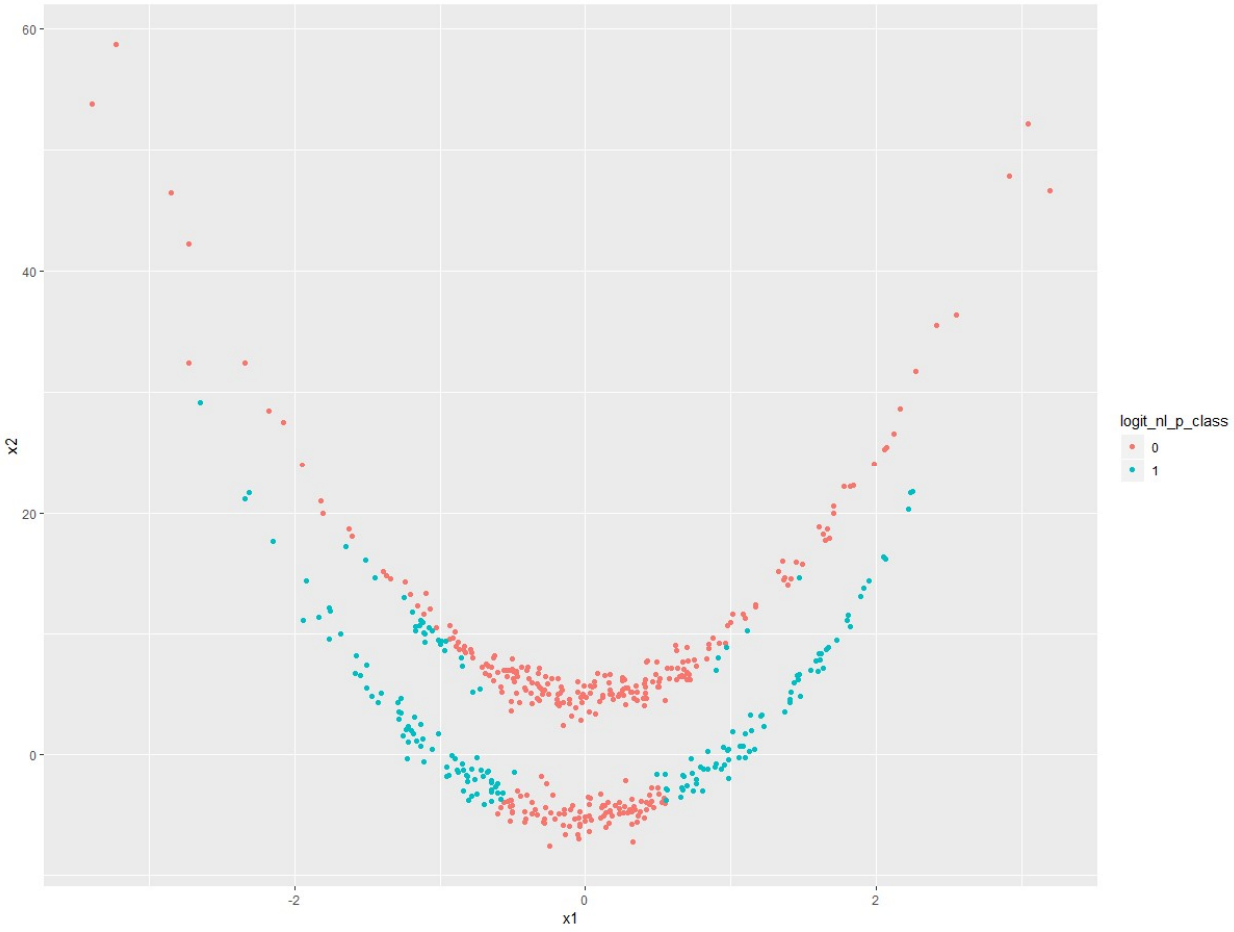
$$ln\left(\frac{\widehat{Pr(Y=1)}}{1-\widehat{Pr(Y=1)}}\right) = \beta_0 + \beta_1 {X_1}^2 + +\beta_2 {X_2}^2 + +\beta_3 X_1 X_2$$

The summary output from estimating the specified logistic regression model above is:

```
Call:
glm(formula = y ~ ., family = "binomial", data = select(df, y,
    x1_squared, x2_squared, x1_x2))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4212  -1.0073   0.0965   1.0399   5.8841

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.301449   0.121325  -2.485    0.013 *
x1_squared   1.490861   0.199473   7.474 7.78e-14 ***
x2_squared  -0.012382   0.001749  -7.079 1.45e-12 ***
x1_x2       -0.008461   0.008677  -0.975    0.329
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 693.15  on 499  degrees of freedom
Residual deviance: 587.70  on 496  degrees of freedom
AIC: 595.7

Number of Fisher Scoring iterations: 6
```

(f) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be obviously non-linear. If it is not, then repeat (a)-(e) until you come up with an example in which the predicted class labels are obviously non-linear.

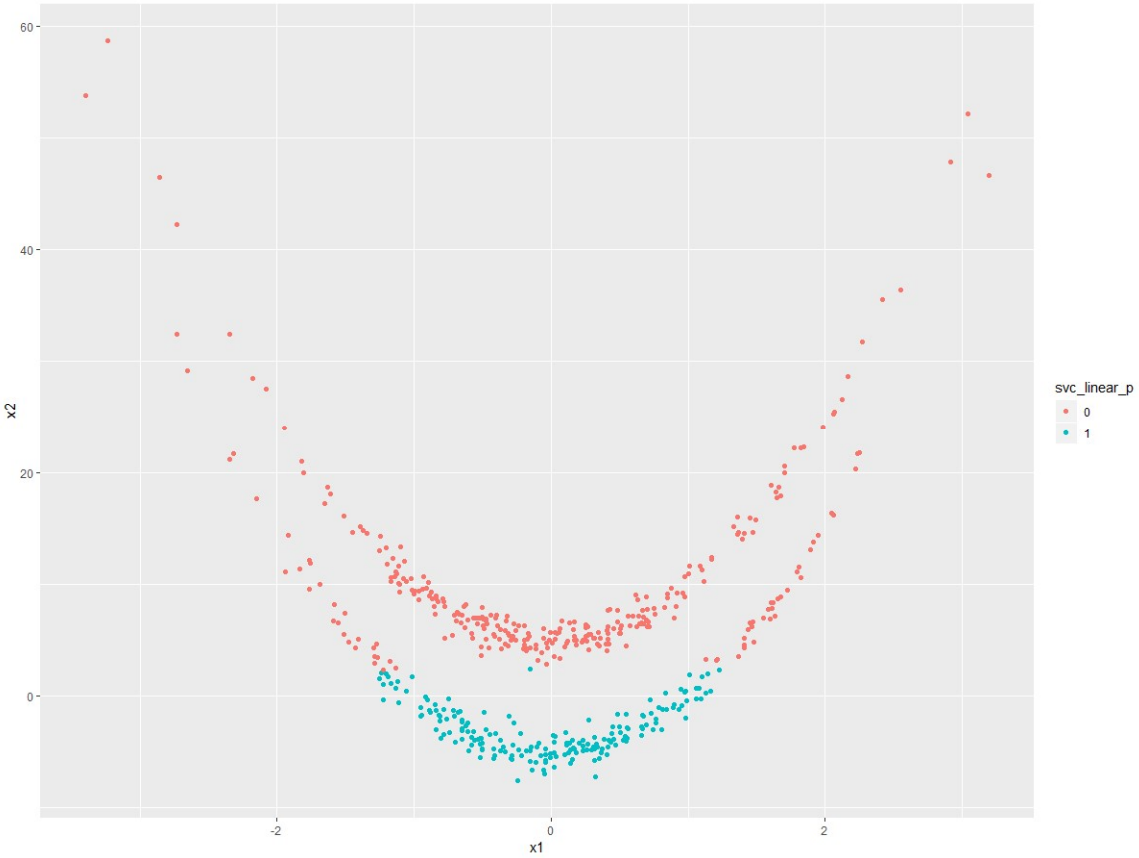
Answer:



As we can see from the plot above, the boundary that separates the colored classes in Y is clearly not linear in the X_1,X_2 plane.

(g) Fit a support vector classifier to the data with X_1 and X_2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

Answer:

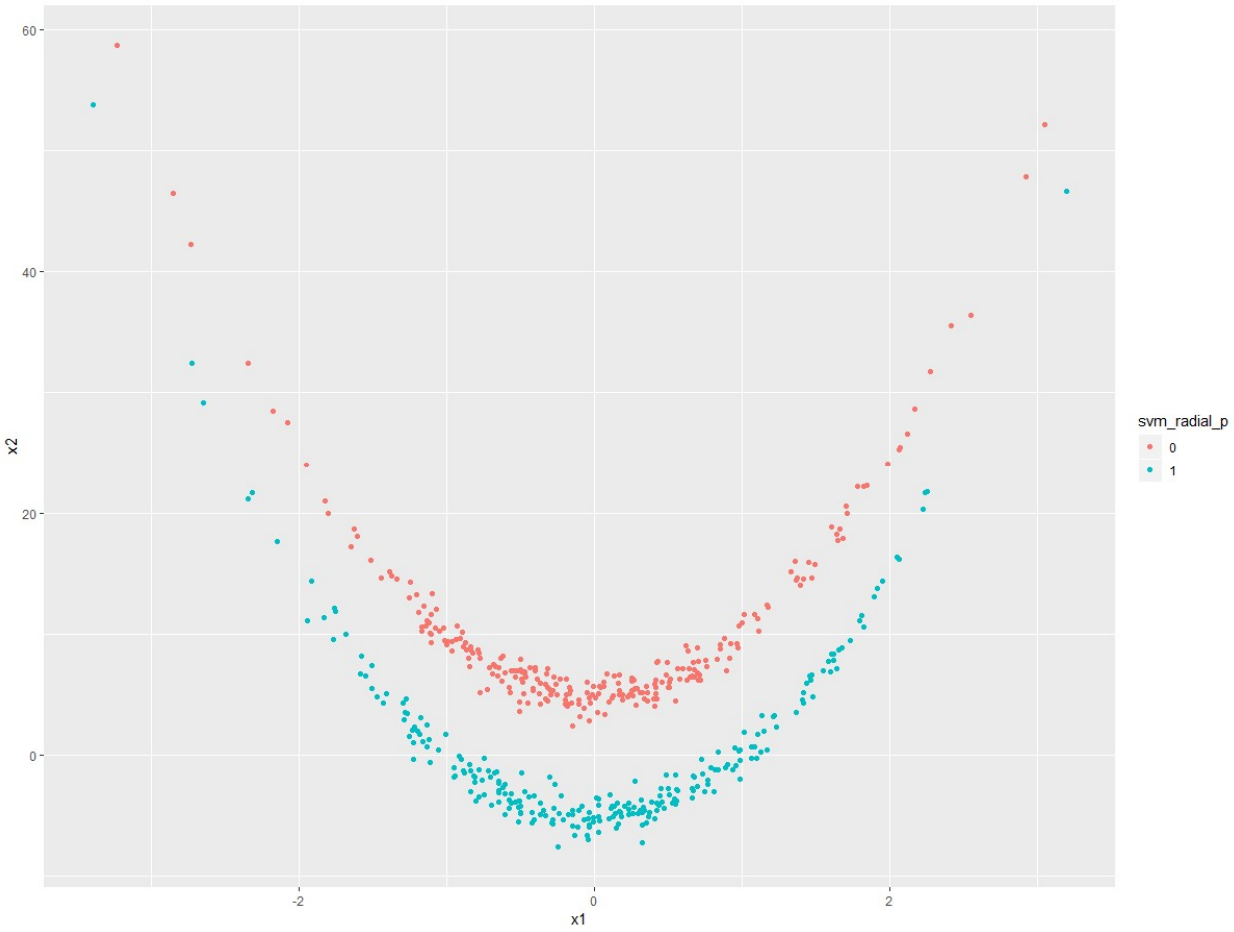


For the linear support vector classifier, we can see the separation of the colored classes in Y is by a linear decision boundary in the X_1,X_2 plane.

(h) Fit a SVM using a non-linear kernel to the data. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

Answer:

A SVM with a radial kernel (with gamma parameter=1) was fit to the training data. The resulting plot is shown below:



As we can see from the plot above, the SVM with radial kernel perfectly separated the data points into their appropriate and true Y classes. This SVM has accuracy 100% on separating the training data into the appropriate Y-classes.

(i) Comment on your results.

Answer:

In the beginning of this problem, we simulated a dataset that head clear separation between Y-classes that had a decision boundary that was clearly non-linear (quadratic) in the X_1,X_2 plane. Of all of the models we fit in this problem, only the logistic regression with non-linear X_1 and X_2 terms, and the SVM with radial kernel were capable of producing an estimated non-linear decision boundary in the X_1,X_2 plane with respect to Y. The logistic regression with non-linear terms for X_1 and X_2, although it can produce a non-linear decision boundary, we are assuming we have specified the model correctly parametrically. This can be a large assumption. If we look at the plot in part (f), the plot shows a decision boundary that is clearly non-linear in the X_1,X_2 plane, but still produces a lot of misclassification in the predicted Y-class. The SVM with radial kernel however does not require nearly the level of a-priori specification by the analyst analyzing the data of the non-linear terms comprising the model as compared to logistic regression. It is therefore little surprise the SVM with radial kernel performed with 100% accuracy on the training data (as shown in part (h)), and performed best compared to all other models considered in this problem.

R-code:

```
#################
## Problem #3 ##
#################
library(dplyr)
library(ggplot2)
set.seed(1234)

#############
## Part A ##
#############
id <- c(1:500)
df <- data.frame(id)
df$x1 <- rnorm(500)
df$error <- rnorm(500)
df$x2 <- 5*((df$x1)^2) + df$error
df$x2[df$id<=250] <- df$x2[df$id<=250]+5
df$x2[df$id>250] <- df$x2[df$id>250]-5
df$y <- 0
df$y[df$id>250] <- 1
df$y <- as.factor(df$y)

#############
## Part B ##
#############
ggplot(df, aes(x=x1, y=x2, color=y)) + geom_point()

#############
## Part C ##
#############
logit_model <- glm(y~., data=select(df, y, x1, x2), family="binomial")
summary(logit_model)

#############
## Part D ##
#############
df$logit_p <- predict(logit_model, df, type="response")
df$logit_p_class <- 0
df$logit_p_class[df$logit_p>=0.5] <- 1
```

```
df$logit_p_class <- as.factor(df$logit_p_class)
ggplot(df, aes(x=x1, y=x2, color=logit_p_class)) + geom_point()

#############
## Part E ##
#############
df$x1_squared <- df$x1*df$x1
df$x2_squared <- df$x2*df$x2
df$x1_x2 <- df$x1*df$x2

logit_model_nl <- glm(y~., data=select(df, y, x1_squared, x2_squared, x1_x2), family="binomial")
summary(logit_model_nl)

#############
## Part F ##
#############
df$logit_nl_p <- predict(logit_model_nl, df, type="response")
df$logit_nl_p_class <- 0
df$logit_nl_p_class[df$logit_nl_p>=0.5] <- 1
df$logit_nl_p_class <- as.factor(df$logit_nl_p_class)
ggplot(df, aes(x=x1, y=x2, color=logit_nl_p_class)) + geom_point()

#############
## Part G ##
#############
library(e1071)
svc_linear <- svm(y~., data=select(df, y, x1, x2), kernel="linear", cost=10)
df$svc_linear_p <- predict(svc_linear, select(df, y, x1, x2))
df$svc_linear_p <- as.factor(df$svc_linear_p)
ggplot(df, aes(x=x1, y=x2, color=svc_linear_p)) + geom_point()

#############
## Part H ##
#############
set.seed(1)
svm_radial <- svm(y~., data=select(df, y, x1, x2), kernel="radial", gamma=1, cost=10)
df$svm_radial_p <- predict(svm_radial, select(df, y, x1, x2))
df$svm_radial_p <- as.factor(df$svm_radial_p)
ggplot(df, aes(x=x1, y=x2, color=svm_radial_p)) + geom_point()
```

Problem #4:

Exercise 8 from section 9.7 of ITSL textbook

This problem involves the "OJ" data set which is part of the ISLR package.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

Answer:

R-code:

```
df <- OJ
train_id <- sample(dim(df)[1], 800)
train_df <- df[train_id,]
test_df <- df[-train_id,]
```

(b) Fit a support vector classifier to the training data using cost=0.01, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics, and describe the results obtained.

Answer:

The summary output is shown below:

```
Call:
svm(formula = Purchase ~ ., data = train_df, kernel = "linear", cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  445

 ( 223 222 )


Number of Classes:  2

Levels:
 CH MM
```

From the summary output above, the linear support vector classifier created 445 support vectors from 800 training points. From these 445 support vectors, 223 belong to the "CH" class and 222 belong to the "MM" class.

(c) What are the training and test error rates?

Answer:

Confusion Matrix: Linear Support Vector Classifier on Training Set

		Predicted Purchase: CH	Predicted Purchase: MM
Actual Purchase	CH	432	57
	MM	76	235

Training error rate=16.625%

Confusion Matrix: Linear Support Vector Classifier on Testing Set

		Predicted Purchase: CH	Predicted Purchase: MM
Actual Purchase	CH	151	13
	MM	32	74

Testing error rate=16.667%

(d) Use the tune() function to select an optimal cost. Consider values in the range 0.01 to 10.

Answer:

From the tune() function, the optimal cost value was estimated to be 9.71

(e) Compute the training and test error rates using this new value for cost.

Answer:

Confusion Matrix: Linear Support Vector Classifier (tuned cost) on Training Set

		Predicted Purchase: CH	Predicted Purchase: MM
Actual Purchase	CH	431	58
	MM	71	240

Tuned training error rate=16.125%

Confusion Matrix: Linear Support Vector Classifier (tuned cost) on Testing Set

		Predicted Purchase: CH	Predicted Purchase: MM
Actual Purchase	CH	149	15
	MM	27	79

Tuned testing error rate=15.556%

(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for gamma.

Answer:

Summary Output:

```
Call:
svm(formula = Purchase ~ ., data = train_df, kernel = "radial", cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  626

 ( 315 311 )


Number of Classes:  2

Levels:
 CH MM
```

From the summary output above, the SVM with radial kernel created 626 support vectors from 800 training points. From these 626 support vectors, 315 belong to the "CH" class and 311 belong to the "MM" class.

Confusion Matrix: SVM with radial kernel on Training Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	489	0
	MM	311	0

Training error rate=38.875%

Confusion Matrix: SVM with radial kernel on Testing Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	164	0
	MM	106	0

Testing error rate=39.259%

From the tune() function, the optimal cost value was estimated to be 1.81

Confusion Matrix: SVM with radial kernel (tuned cost) on Training Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	446	43
	MM	77	234

Tuned training error rate=15%

Confusion Matrix: SVM with radial kernel (tuned cost) on Testing Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	147	17
	MM	33	73

Tuned testing error rate=18.519%

(g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2.

Answer:

Summary Output:

```
Call:
svm(formula = Purchase ~ ., data = train_df, kernel = "polynomial", degree = 2, cost = 0.
01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  0.01
     degree:  2
      gamma:  0.05555556
     coef.0:  0

Number of Support Vectors:  629

 ( 318 311 )


Number of Classes:  2

Levels:
 CH MM
```

From the summary output above, the SVM with polynomial degree 2 kernel created 629 support vectors from 800 training points. From these 629 support vectors, 318 belong to the "CH" class and 311 belong to the "MM" class.

Confusion Matrix: SVM with polynomial kernel on Training Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	488	1
	MM	296	15

Training error rate=37.125%

Confusion Matrix: SVM with polynomial kernel on Testing Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	164	0
	MM	104	2

Testing error rate=38.519%

From the tune() function, the optimal cost value was estimated to be 8.71

Confusion Matrix: SVM with polynomial kernel (tuned cost) on Training Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	445	44
	MM	82	229

Tuned training error rate=15.75%

Confusion Matrix: SVM with polynomial kernel (tuned cost) on Testing Set

		Predicted Purchase	
		CH	MM
Actual Purchase	CH	152	12
	MM	36	70

Tuned testing error rate=17.778%

(h) Overall, which approach seems to give the best results on this data?

Answer:

Comparing all of the results, and concentrating particularly on the error rates of the models with respect to the testing set (as opposed to the training set). The model that produced the lowest testing error was the linear support vector classifier with the tuned cost parameters (cost=9.71). This produced the lowest testing error of 15.556%.

R-code:

```
#################
## Problem #4 ##
#################
library(ISLR)
library(dplyr)
library(ggplot2)
library(e1071)

#############
## Part A ##
#############
set.seed(1234)
df <- OJ
train_id <- sample(dim(df)[1], 800)
train_df <- df[train_id,]
test_df <- df[-train_id,]

#############
## Part B ##
#############
svc_linear <- svm(Purchase~., data=train_df, kernel="linear", cost=0.01)
summary(svc_linear)

#############
## Part C ##
#############
train_df$svc_linear_p <- predict(svc_linear, train_df)
table(train_df$Purchase, train_df$svc_linear_p)

test_df$svc_linear_p <- predict(svc_linear, test_df)
table(test_df$Purchase, test_df$svc_linear_p)

#############
## Part D ##
#############
set.seed(1234)
tune_cost_svc_linear <- tune(svm, Purchase~., data=select(train_df, -svc_linear_p),
kernel="linear", ranges=list(cost=seq(0.01,10,0.1)))
summary(tune_cost_svc_linear)

#############
## Part E ##
#############
svc_linear_tuned <- svm(Purchase~., data=select(train_df,-svc_linear_p), kernel="linear",
cost=tune_cost_svc_linear$best.parameters$cost)
train_df$svc_linear_tuned_p <- predict(svc_linear_tuned, select(train_df,-svc_linear_p))
table(train_df$Purchase, train_df$svc_linear_tuned_p)

test_df$svc_linear_tuned_p <- predict(svc_linear_tuned, select(test_df,-svc_linear_p))
table(test_df$Purchase, test_df$svc_linear_tuned_p)
```

```
############
## Part F ##
############
set.seed(1)
svm_radial <- svm(Purchase~., data=train_df, kernel="radial", cost=0.01)
summary(svm_radial)

train_df$svm_radial_p <- predict(svm_radial, train_df)
table(train_df$Purchase, train_df$svm_radial_p)

test_df$svm_radial_p <- predict(svm_radial, test_df)
table(test_df$Purchase, test_df$svm_radial_p)

set.seed(1234)
tune_cost_svm_radial <- tune(svm, Purchase~., data=select(train_df, -svm_radial_p),
kernel="radial", ranges=list(cost=seq(0.01,10,0.1)))
summary(tune_cost_svm_radial)

svm_kernel_tuned <- svm(Purchase~., data=select(train_df,-svm_radial_p), kernel="radial",
cost=tune_cost_svm_radial$best.parameters$cost)
train_df$svm_kernel_tuned_p <- predict(svm_kernel_tuned, select(train_df,-svm_radial_p))
table(train_df$Purchase, train_df$svm_kernel_tuned_p)

test_df$svm_kernel_tuned_p <- predict(svm_kernel_tuned, select(test_df,-svm_radial_p))
table(test_df$Purchase, test_df$svm_kernel_tuned_p)

############
## Part G ##
############
set.seed(1)
svm_poly <- svm(Purchase~., data=train_df, kernel="polynomial", degree=2, cost=0.01)
summary(svm_poly)

train_df$svm_poly_p <- predict(svm_poly, train_df)
table(train_df$Purchase, train_df$svm_poly_p)

test_df$svm_poly_p <- predict(svm_poly, test_df)
table(test_df$Purchase, test_df$svm_poly_p)

set.seed(1234)
tune_cost_svm_poly <- tune(svm, Purchase~., data=select(train_df, -svm_poly_p),
kernel="polynomial", degree=2, ranges=list(cost=seq(0.01,10,0.1)))
summary(tune_cost_svm_poly)

svm_poly_tuned <- svm(Purchase~., data=select(train_df,-svm_poly_p), kernel="polynomial",
degree=2, cost=tune_cost_svm_poly$best.parameters$cost)
train_df$svm_poly_tuned_p <- predict(svm_poly_tuned, select(train_df,-svm_poly_p))
table(train_df$Purchase, train_df$svm_poly_tuned_p)

test_df$svm_poly_tuned_p <- predict(svm_poly_tuned, select(test_df,-svm_poly_p))
table(test_df$Purchase, test_df$svm_poly_tuned_p)
```