ISLR | Chapter 9 Exercises

Marshall McQuillen

11/17/2018

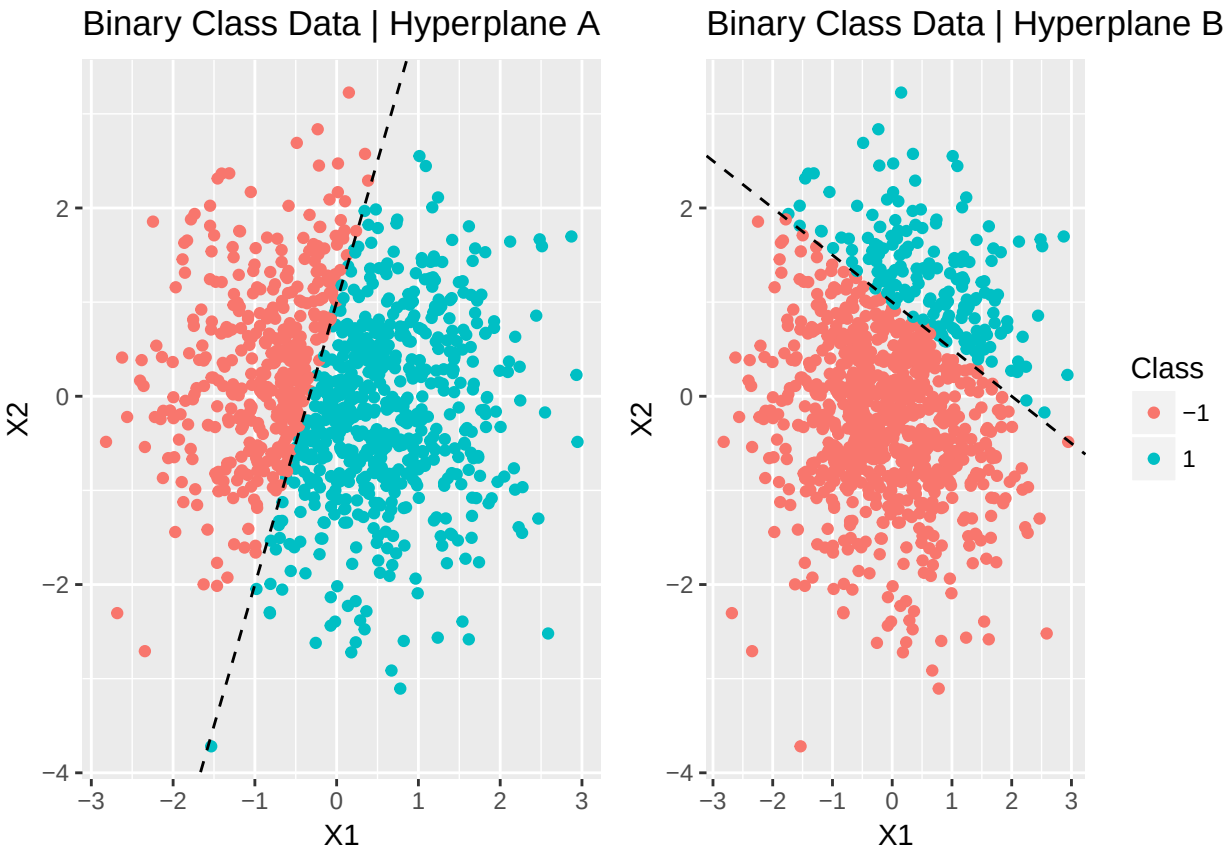
Conceptual

1

- **A & B.**

```
suppressPackageStartupMessages(library(gridExtra))
suppressPackageStartupMessages(library(ggplot2))

# data setup
x <- matrix(rnorm(1000*2), ncol = 2)

# define hyperplane 1 (part A)
hyperplane1 <- 1 + 3*x[, 1] - x[, 2]
y <- ifelse(hyperplane1 > 0, 1, -1)

# define hyperplane 2 (part b)
hyperplane2 <- -2 + x[, 1] + 2*x[, 2]
y2 <- ifelse(hyperplane2 > 0, 1, -1)

par(mfrow = c(1, 2))

# plot hyperplanes
plot1 <- ggplot(data.frame(x = x,
                  y = factor(y, levels = c(-1, 1))), aes(x.1, x.2, color=y)) +
            geom_point(show.legend = FALSE) +
            geom_abline(intercept = 1,
                        slope = 3,
                        linetype = 'dashed') +
            ggtitle("Binary Class Data | Hyperplane A") +
            xlab("X1") +
            ylab("X2") +
            labs(color = "Class")

plot2 <- ggplot(data.frame(x = x,
                  y = factor(y2, levels = c(-1, 1))), aes(x.1, x.2, color=y)) +
            geom_point() +
            geom_abline(intercept = 1,
                        slope = -0.5,
                        linetype = 'dashed') +
            ggtitle("Binary Class Data | Hyperplane B") +
            xlab("X1") +
            ylab("X2") +
            labs(color = "Class")
```

```
grid.arrange(plot1, plot2, ncol = 2)
```

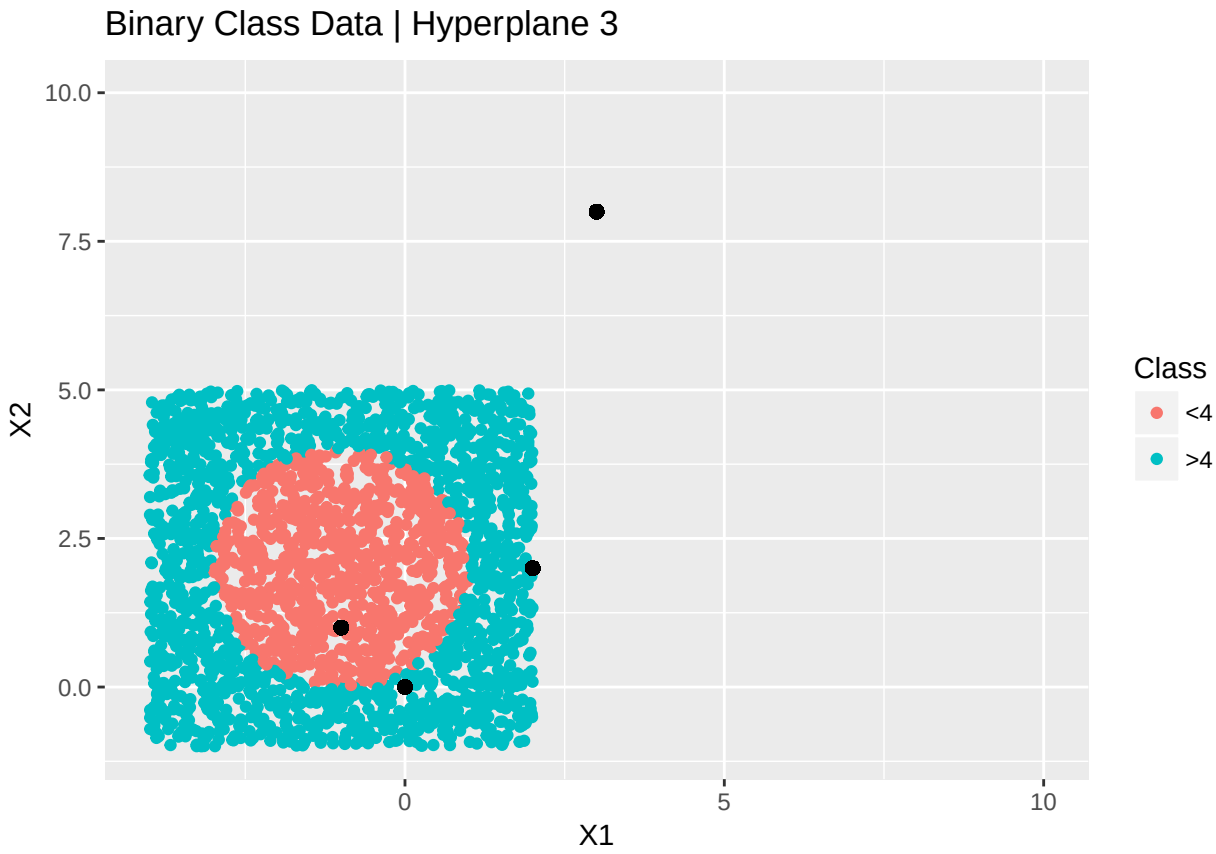


2

- **A, B & C**. The hyperplane is the circle encompassing the pink/orange data points below, where those data points are the ones who's value, when plugged into the equation $f(X_1, X_2) = (1 + X_1)^2 + (2 - X_2)^2 - 4$, will be negative. The blue data points output value of the above equation would be positive. The 4 data points plotted in black are those requested in part **C**, and it is clear which class they would fall into.

```
# data setup
x_1 <- runif(2500, min = -4, max = 2)
x_2 <- runif(2500, min = -1, max = 5)

# define hyperplane
hyperplane3 <- (1 + x_1)^2 + (2 - x_2)^2 - 4
y3 <- factor(ifelse(hyperplane3 > 0, ">4", "<4"))

df <- data.frame(x1 = x_1,
                 x2 = x_2,
                 y = y3)

# plot hyperplane with additional points in black
ggplot(df, aes(x1, x2, color=y)) +
```

```
    geom_point() +
    geom_point(x = 0,
               y = 0,
               col = 'black',
               cex = 2) +
    geom_point(x = -1,
               y = 1,
               col = 'black',
               cex = 2) +
    geom_point(x = 2,
               y = 2,
               col = 'black',
               cex = 2) +
    geom_point(x = 3,
               y = 8,
               col = 'black',
               cex = 2) +
    ggtitle("Binary Class Data | Hyperplane 3") +
    scale_y_continuous(limits = c(-1, 10)) +
    scale_x_continuous(limits = c(-4, 10)) +
    xlab("X1") +
    ylab("X2") +
    labs(color = "Class")
```



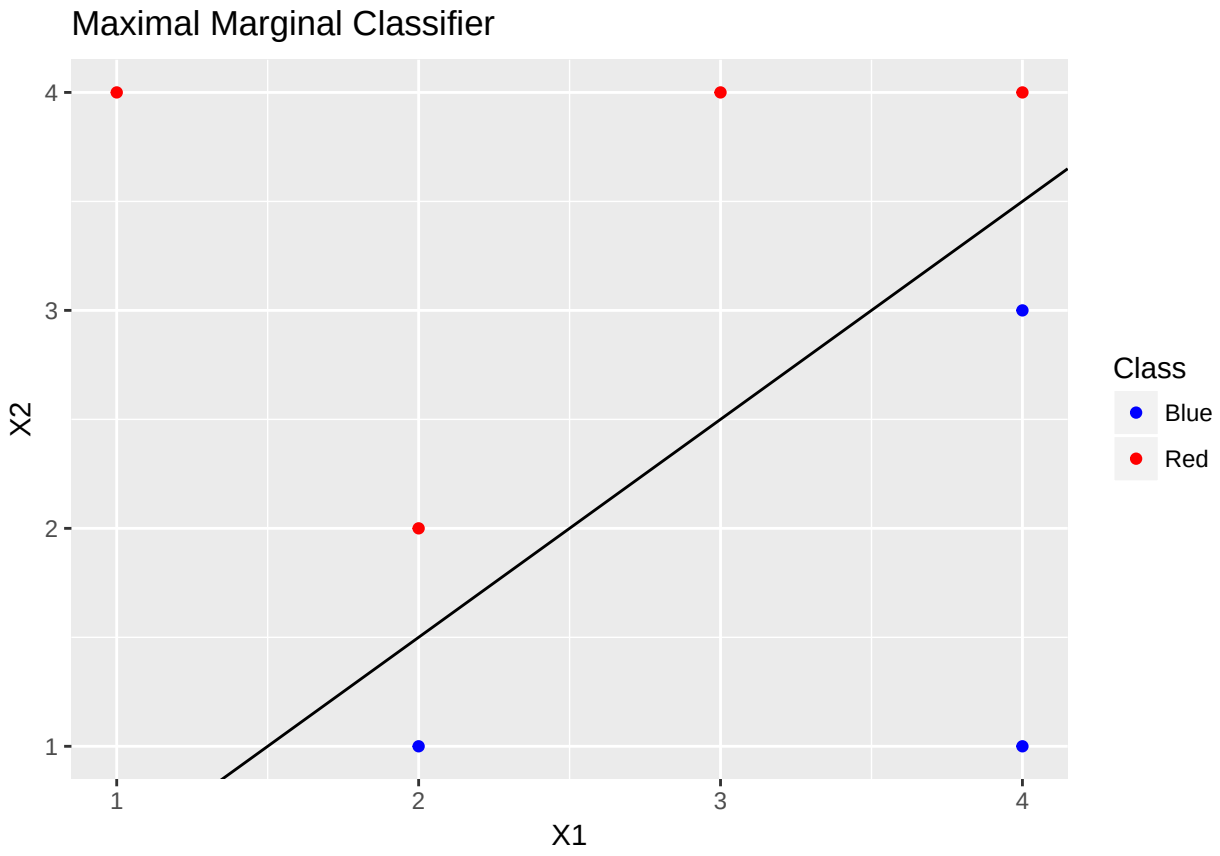
- **D**. One can see that the equation given in the text is non-linear with regard to X_1 and X_2. However, when expanded and refactored, it is clear that the hyperplane is linear with regard to to X_1, X_2, X_1^2 and X_2^2.

$$(1 + X_1)^2 + (2 - X_2)^2 = 4 \tag{1}$$
$$(1 + X_1)^2 + (2 - X_2)^2 - 4 = 0 \tag{2}$$
$$(1 + 2X_1 + X_1^2) + (4 - 4X_2 + X_2^2) - 4 = 0 \tag{3}$$
$$1 + 2X_1 + X_1^2 - 4X_2 + X_2^2 = 0 \tag{4}$$
$$\tag{5}$$

3

- **A & B**. The separating hyperplane has the equation $X_1 - X_2 - 0.5 = 0$.

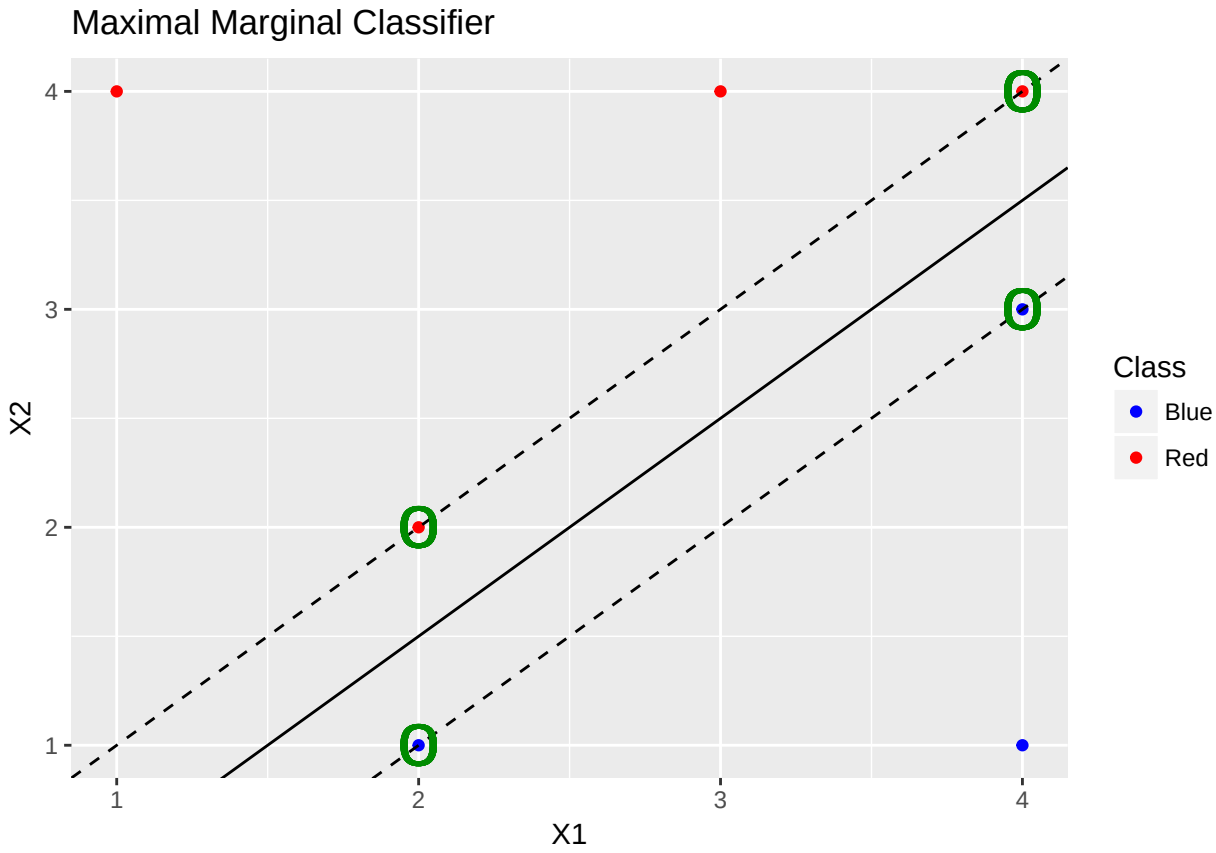
```
# data setup
df <- data.frame(x_1 = c(3,2,4,1,2,4,4),
                 x_2 = c(4,2,4,4,1,3,1),
                 y = c(rep("Red", 4), rep("Blue", 3)))

# plot MMC
ggplot(df, aes(x_1, x_2, color = y)) +
    geom_point() +
    geom_abline(intercept = -0.5,
                slope = 1,
                linetype = 'solid') +
    scale_color_manual(values = c("Red" = 'red', "Blue" = 'blue')) +
    ggtitle("Maximal Marginal Classifier") +
    xlab("X1") +
    ylab("X2") +
    labs(color = 'Class')
```



- **C**. Classify a test observation x_t to Red if $(1)X_1 + (-1)X_2 - 0.5 < 0$, otherwise classify to Blue.
- **D**. The support vectors are those data points circled in green, the edge of the margins are the dashed lines.

```
# illustrate support vectors and margins
ggplot(df, aes(x_1, x_2, color = y)) +
    geom_point() +
    geom_abline(intercept = -0.5,
                slope = 1,
                linetype = 'solid') +
    geom_abline(intercept = -0,
                slope = 1,
                linetype = 'dashed') +
    geom_abline(intercept = -1,
                slope = 1,
                linetype = 'dashed') +
    geom_point(x = 2,
               y = 2,
               color = 'green4',
               size = 10,
               shape = "o") +
    geom_point(x = 2,
               y = 1,
               color = 'green4',
```

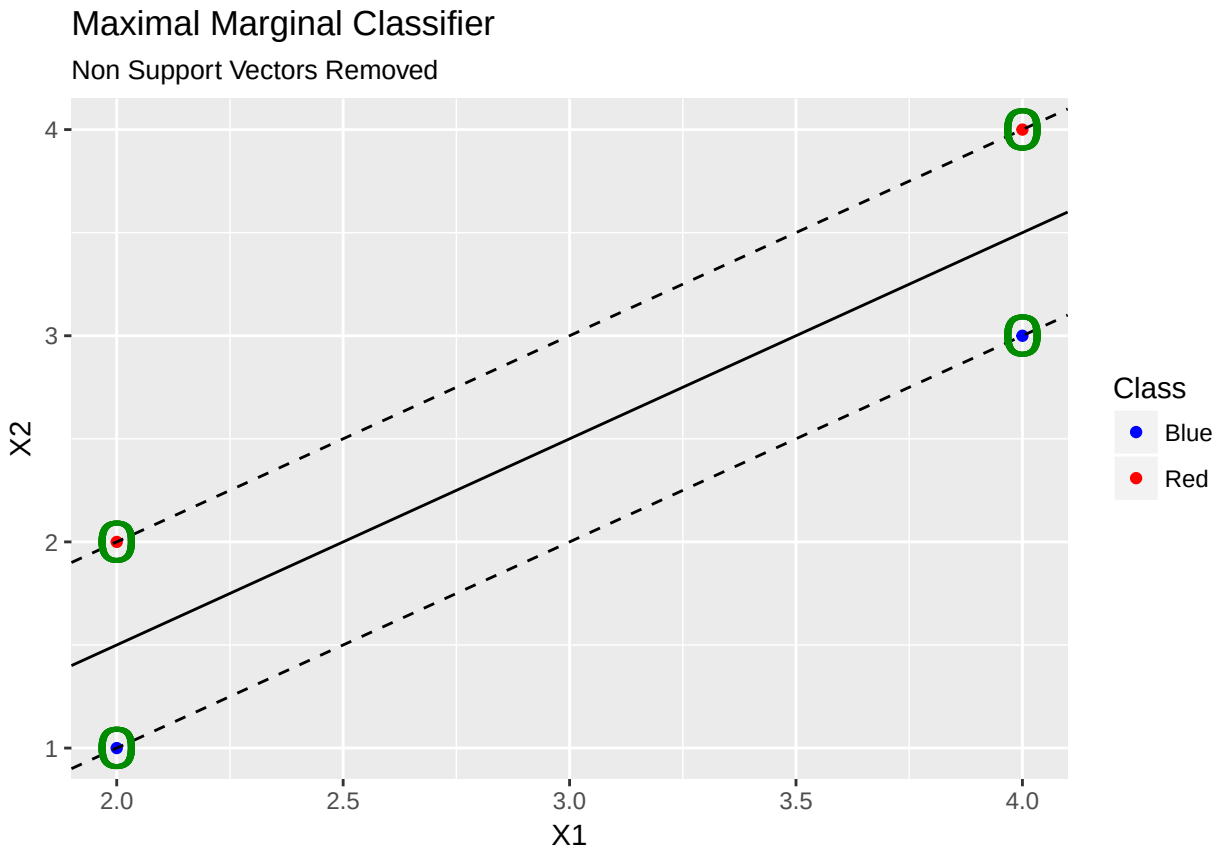
```
              size = 10,
              shape = "o") +
  geom_point(x = 4,
              y = 3,
              color = 'green4',
              size = 10,
              shape = "o") +
  geom_point(x = 4,
              y = 4,
              color = 'green4',
              size = 10,
              shape = "o") +
  scale_color_manual(values = c("Red" = 'red', "Blue" = 'blue')) +
  ggtitle("Maximal Marginal Classifier") +
  xlab("X1") +
  ylab("X2") +
  labs(color = 'Class')
```



- **F**. Not only can the seventh observation be removed from the data set without affecting the hyperplane, **any observation that is not a support vector can be removed without distorting the hyperplane,** as shown in the plot below.

```
# remove non support vectors
sub_df <- df[-c(1, 4, 7), ]
```

```
# remove non support vectors
ggplot(sub_df, aes(x_1, x_2, color = y)) +
    geom_point() +
    geom_abline(intercept = -0.5,
                slope = 1,
                linetype = 'solid') +
    geom_abline(intercept = -0,
                slope = 1,
                linetype = 'dashed') +
    geom_abline(intercept = -1,
                slope = 1,
                linetype = 'dashed') +
    geom_point(x = 2,
               y = 2,
               color = 'green4',
               size = 10,
               shape = "o") +
    geom_point(x = 2,
               y = 1,
               color = 'green4',
               size = 10,
               shape = "o") +
    geom_point(x = 4,
               y = 3,
               color = 'green4',
               size = 10,
               shape = "o") +
    geom_point(x = 4,
               y = 4,
               color = 'green4',
               size = 10,
               shape = "o") +
    scale_color_manual(values = c("Red" = 'red', "Blue" = 'blue')) +
    ggtitle("Maximal Marginal Classifier",
            "Non Support Vectors Removed") +
    xlab("X1") +
    ylab("X2") +
    labs(color = 'Class')
```

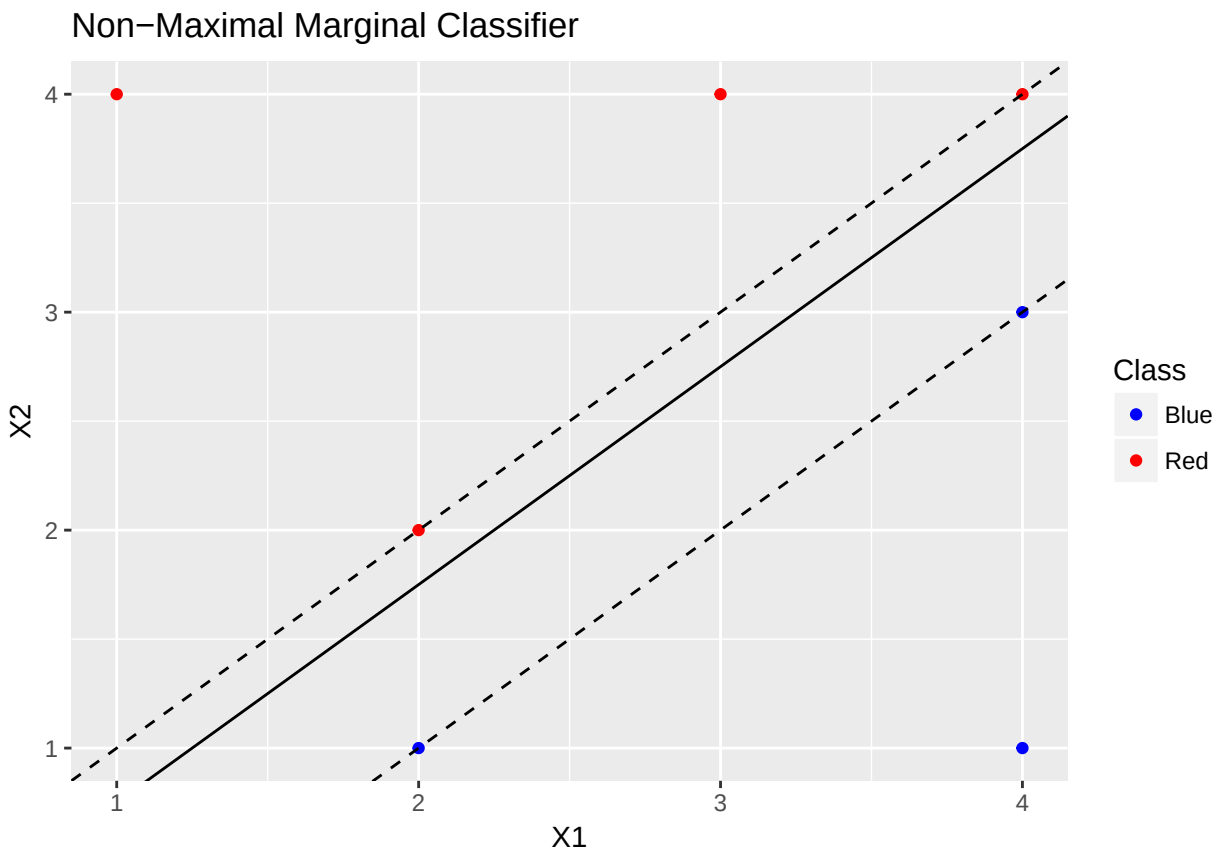


- **G**. The below hyperplane, with the equation $(1)X_1 + (-1)X_2 - 0.25$ would not be the **maximal** separating hyperplane because the margin between the red support vectors and the hyperplane is smaller than the margin between the blue support vectors and the machine.

  The reason that this is less desirable than the Maximal Marginal Classifier is because the hyperplane, *in this scenario*, is favoring the blue data points, **with no "supporting" evidence (pun intended)**. It is giving them (the blue support vectors) a wider berth than is necessary, at the cost of the berth to the red support vectors.

```
# illustrate non-maximal margin classifier
ggplot(df, aes(x_1, x_2, color = y)) +
    geom_point() +
    geom_abline(intercept = -0.25,
                slope = 1,
                linetype = 'solid') +
    geom_abline(intercept = -0,
                slope = 1,
                linetype = 'dashed') +
    geom_abline(intercept = -1,
                slope = 1,
                linetype = 'dashed') +
    scale_color_manual(values = c("Red" = 'red', "Blue" = 'blue')) +
    ggtitle("Non-Maximal Marginal Classifier") +
    xlab("X1") +
    ylab("X2") +
```

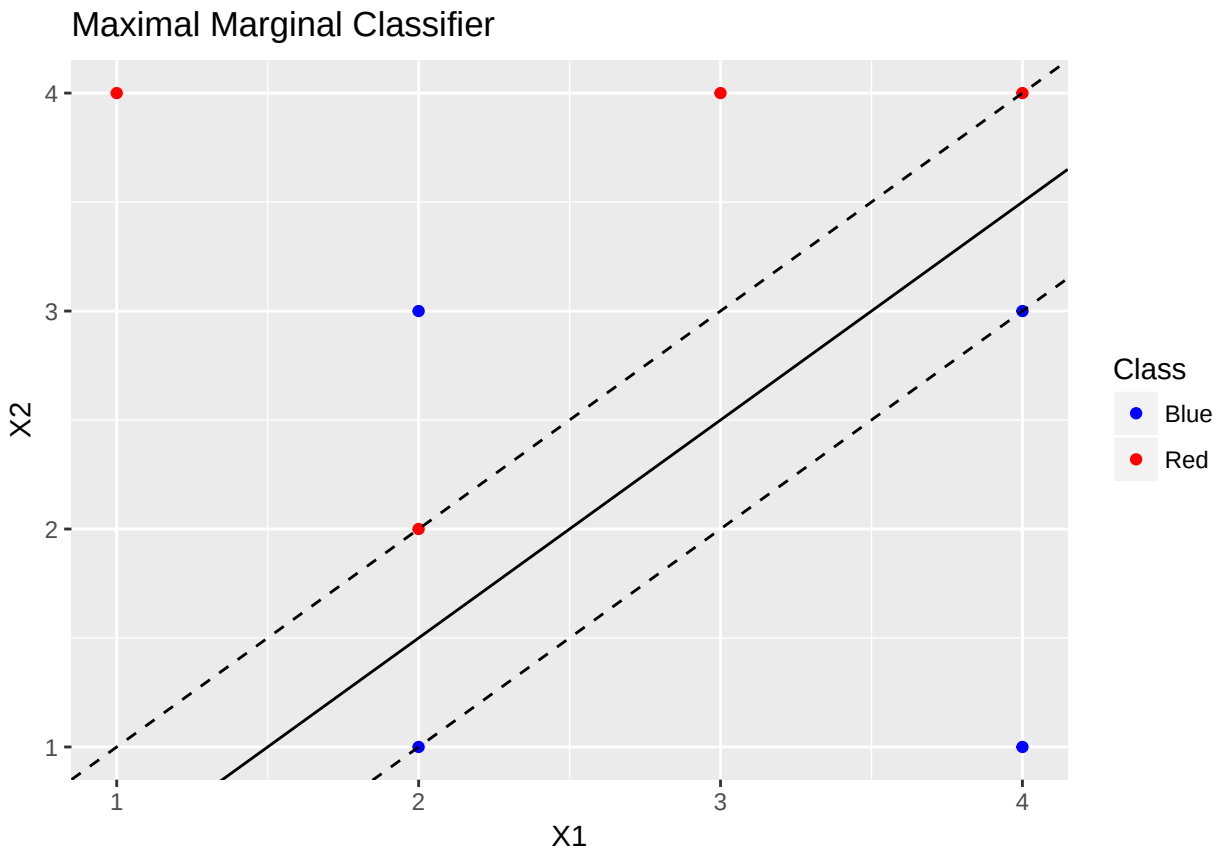
```
    labs(color = 'Class')
```



- **H**. A data set where the Maximal Marginal Classifier does not exist.

```
new.point <- data.frame(x_1 = 2,
                        x_2 = 3,
                        y = "Blue")
df <- rbind(df, new.point)

# illustrate tha data set where a non-maximal is not possible
ggplot(df, aes(x_1, x_2, color = y)) +
    geom_point() +
    geom_abline(intercept = -0.5,
                slope = 1,
                linetype = 'solid') +
    geom_abline(intercept = -0,
                slope = 1,
                linetype = 'dashed') +
    geom_abline(intercept = -1,
                slope = 1,
                linetype = 'dashed') +
    scale_color_manual(values = c("Red" = 'red', "Blue" = 'blue')) +
    ggtitle("Maximal Marginal Classifier") +
    xlab("X1") +
```

```
    ylab("X2") +
    labs(color = 'Class')
```



Applied

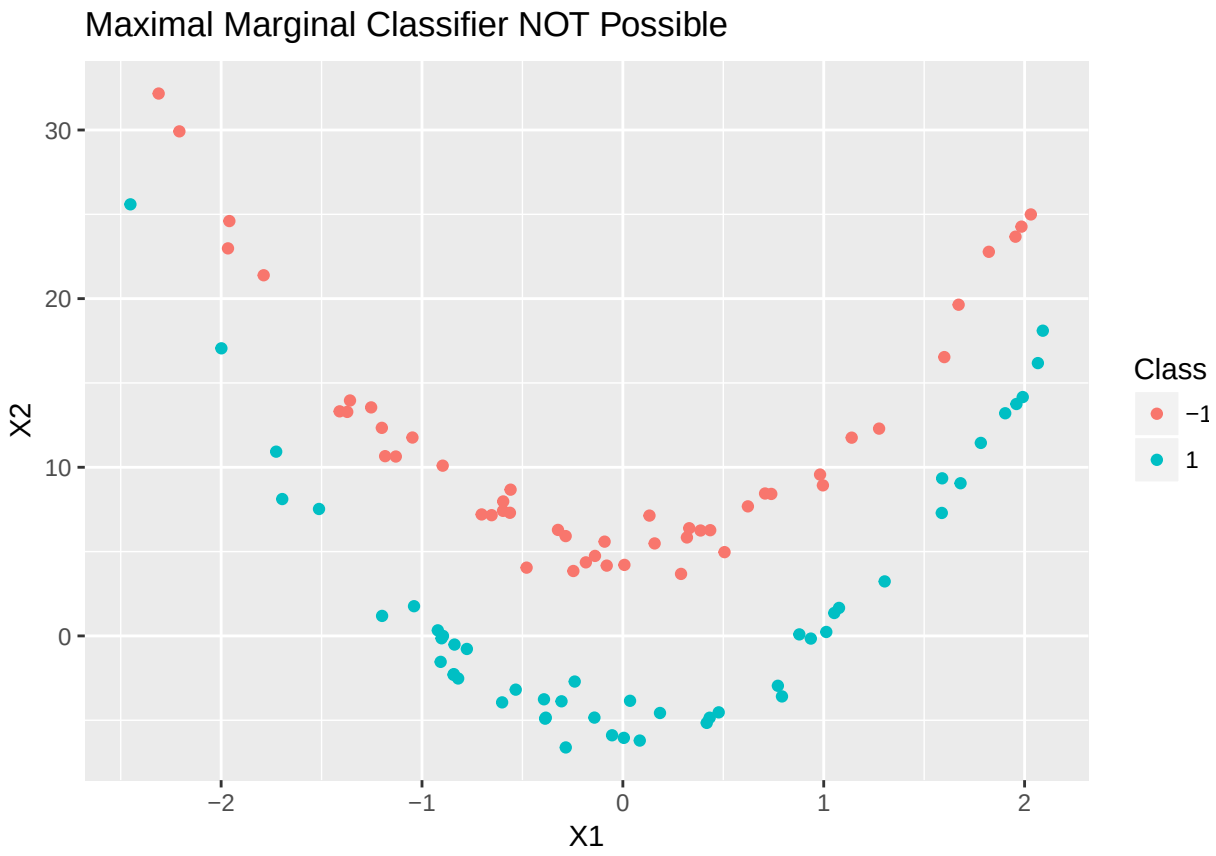
4

As shown below, a linear decision boundary is not possible with the data provided. Knowing this, it is unlikely that a linear SVM would outperform a more complex model. This intuition is confirmed in the following three plots, showing that the radial SVM (with $\gamma = 2$) is the model that fits the training data the best.

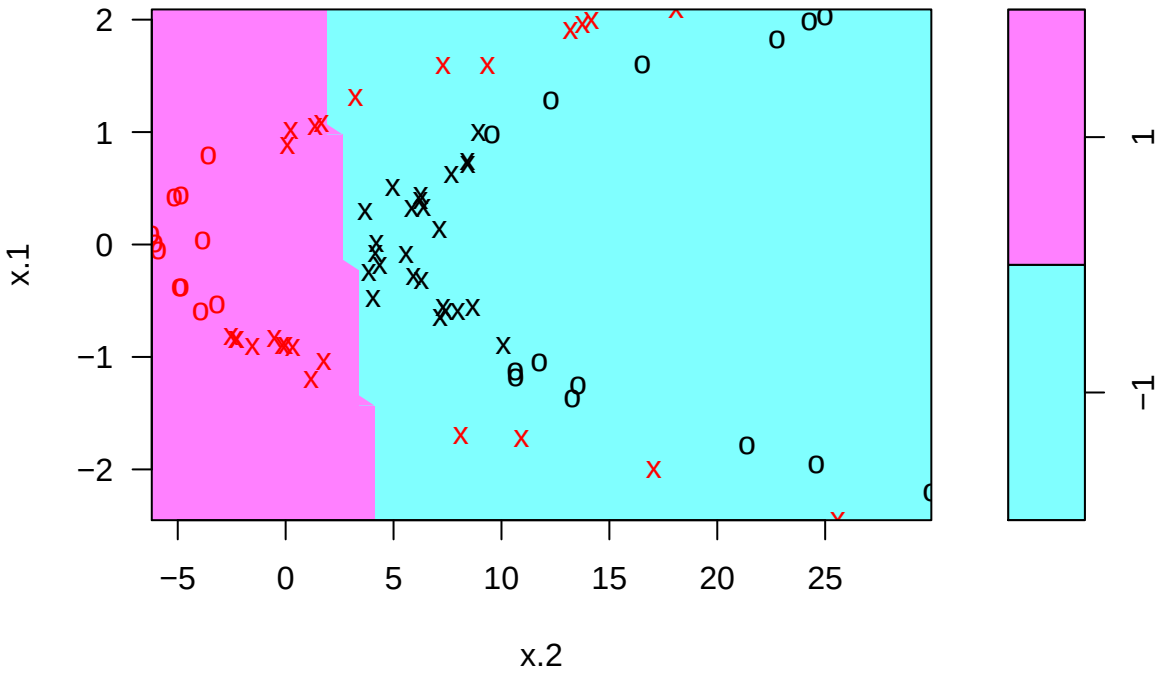
Not surprisingly, the radial SVM outperforms the linear and polynomial SVM's on the test set as well. Interestingly, the radial SVM also has fewer support vectors, therefore making the model more computationally efficient to store.

```
suppressPackageStartupMessages(library(e1071))
suppressPackageStartupMessages(library(caret))
set.seed(2)

# data setup
x_1 <- rnorm(100)
x_2 <- 5*(x_1^2) + rnorm(100)
idx <- sample(100, 50)
```

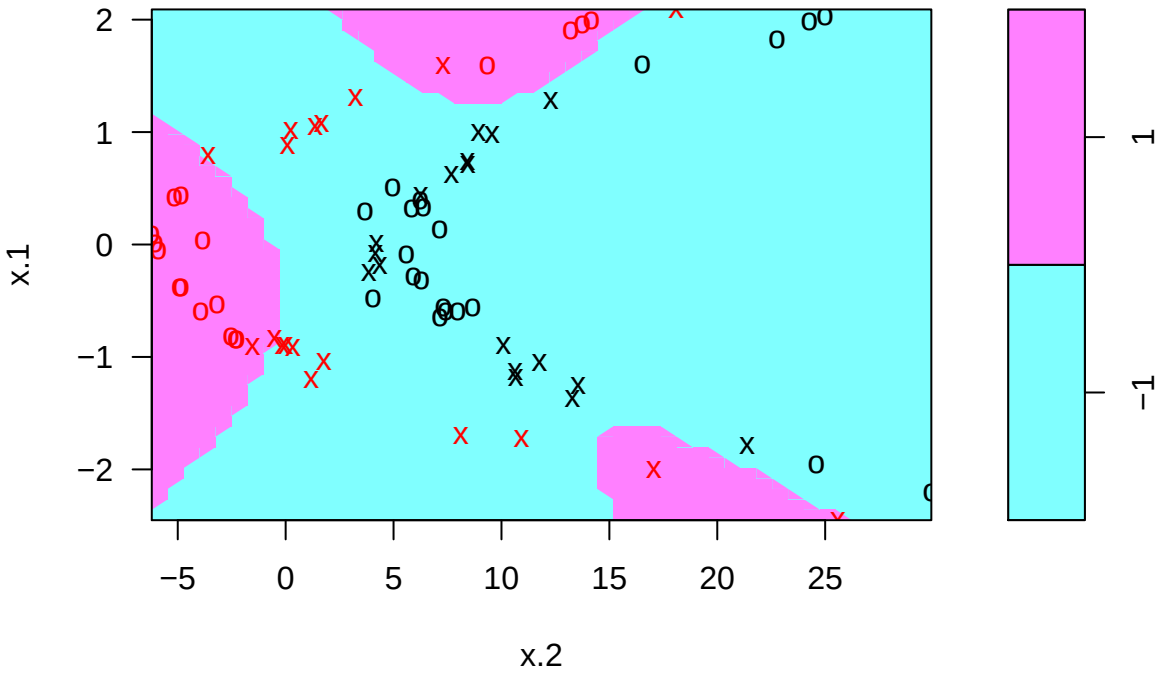
```r
x_2[idx] <- x_2[idx] + 5
x_2[-idx] <- x_2[-idx] - 5
y <- factor(ifelse(x_2 > 5*(x_1^2) + rnorm(100), -1, 1), levels = c(-1, 1))

df <- data.frame(x.1 = x_1,
                 x.2 = x_2,
                 y = y)

# plot data
ggplot(df, aes(x.1, x.2, color = y)) +
    geom_point() +
    ggtitle("Maximal Marginal Classifier NOT Possible") +
    xlab("X1") +
    ylab("X2") +
    labs(color = 'Class')
```



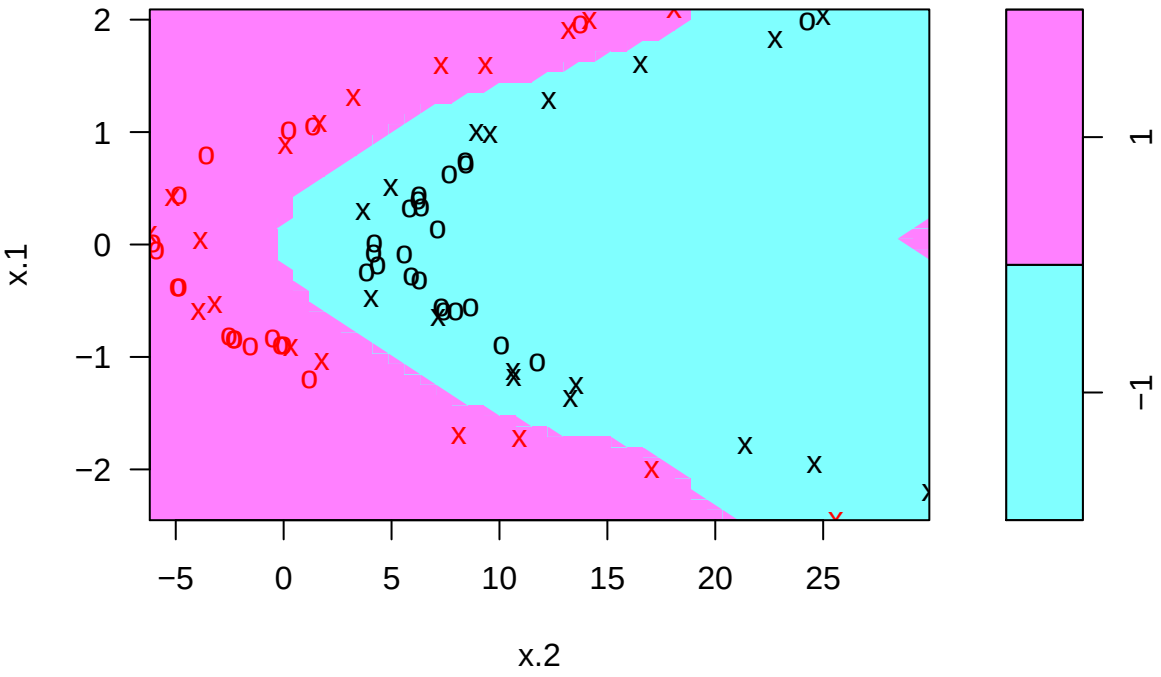

```r
set.seed(5)
train <- sample(100, 75)

# linear support vector machine
linear.svm <- svm(y ~ .,
                  data = df[train,],
                  kernel = 'linear')
plot(linear.svm, df[train,])
```




```
# polynomial support vector machine
poly.svm <- svm(y ~ .,
                data = df[train,],
                kernel = 'polynomial',
                gamma = 2)
plot(poly.svm, df[train,])
```




```
# radial support vector machine
radial.svm <- svm(y ~ .,
                  data = df[train,],
                  kernel = 'radial',
                  gamma = 2)
plot(radial.svm, df[train,])
```




```
# prediction
linear.svm.yhat <- predict(linear.svm, newdata = df[-train, ])
poly.svm.yhat <- predict(poly.svm, newdata = df[-train, ])
radial.svm.yhat <- predict(radial.svm, newdata = df[-train, ])

linear.svm.acc <- confusionMatrix(df[-train, 'y'], linear.svm.yhat)$overall[1]
poly.svm.acc <- confusionMatrix(df[-train, 'y'], poly.svm.yhat)$overall[1]
radial.svm.acc <- confusionMatrix(df[-train, 'y'], radial.svm.yhat)$overall[1]

print(paste("Linear SVM test accuracy =",
            linear.svm.acc,
            "with",
            linear.svm$tot.nSV,
            "support vectors."))
```

```
## [1] "Linear SVM test accuracy = 0.84 with 50 support vectors."
```

```
print(paste("Polynomial SVM test accuracy =",
            poly.svm.acc,
            "with",
            poly.svm$tot.nSV,
            "support vectors."))
```

```
## [1] "Polynomial SVM test accuracy = 0.88 with 37 support vectors."
```

```
print(paste("Radial SVM test accuracy =",
            radial.svm.acc,
            "with",
            radial.svm$tot.nSV,
            "support vectors."))
```

```
## [1] "Radial SVM test accuracy = 1 with 36 support vectors."
```