Math642_HW7_FyonaSun

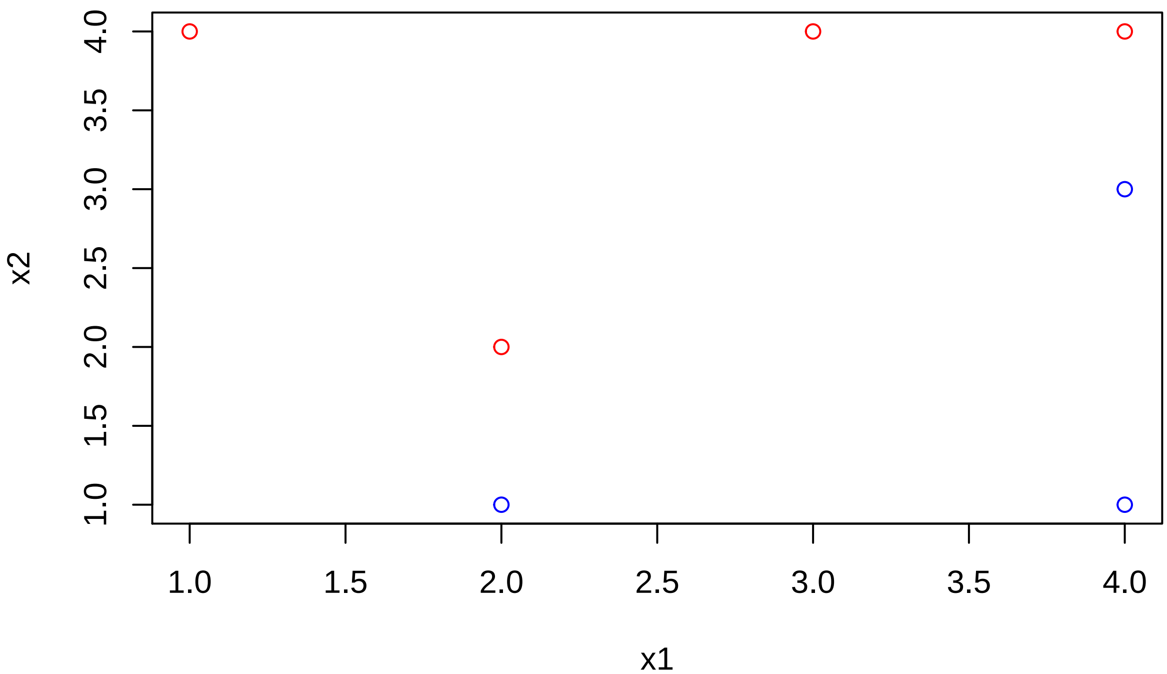
Fyona Sun

3/12/2020

9.3

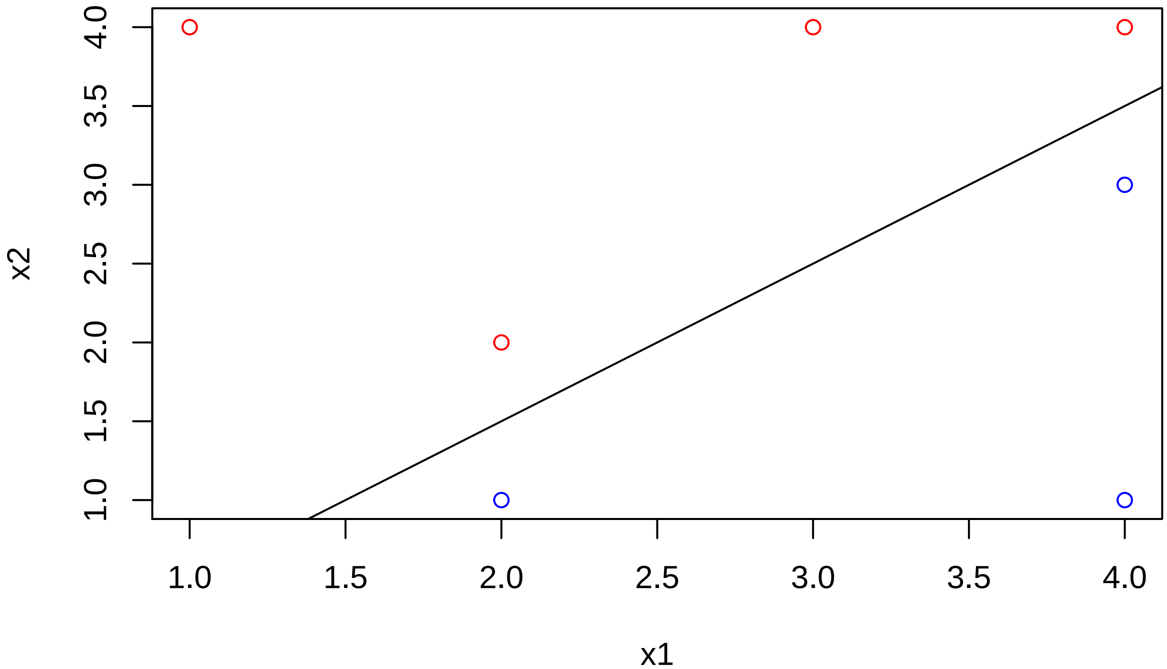
(a) We are given n = 7 observations in p = 2 dimensions. For each observation, there is an associated class label. Sketch the observations.

```
x1 = c(3, 2, 4, 1, 2, 4, 4)
x2 = c(4, 2, 4, 4, 1, 3, 1)
colors = c("red", "red", "red", "red", "blue", "blue", "blue")
plot(x1, x2, col = colors)
```



(b) Sketch the optimal separating hyperplane, and provide the equa- tion for this hyperplane (of the form (9.1)). From the plot, we can see that the optimal separating hyperplane has to be between (2,1), (2,2) and (4,3), (4,4). Thus the equation for hyperplane should be $X_1 - X_2 - 0.5 = 0$

```
plot(x1, x2, col = colors)
abline(-0.5, 1)
```

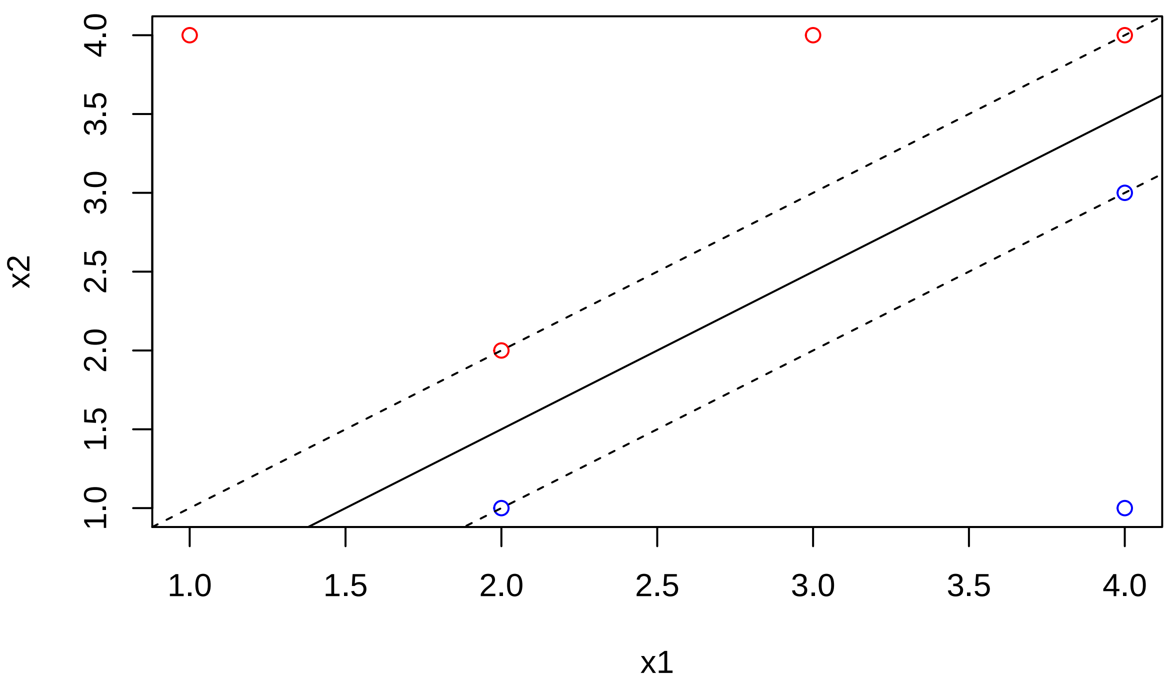


(c) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$, and classify to Blue otherwise." Provide the values for β_0, β_1, and β_2.

The classification rule is "Classify to Red if 0.5-X1+X2>0, and classify to Blue otherwise."

(d) On your sketch, indicate the margin for the maximal margin hyperplane.

```
plot(x1, x2, col = colors)
abline(-0.5, 1)
abline(-1, 1, lty = 2)
abline(0, 1, lty = 2)
```



The

margin is 1/4.

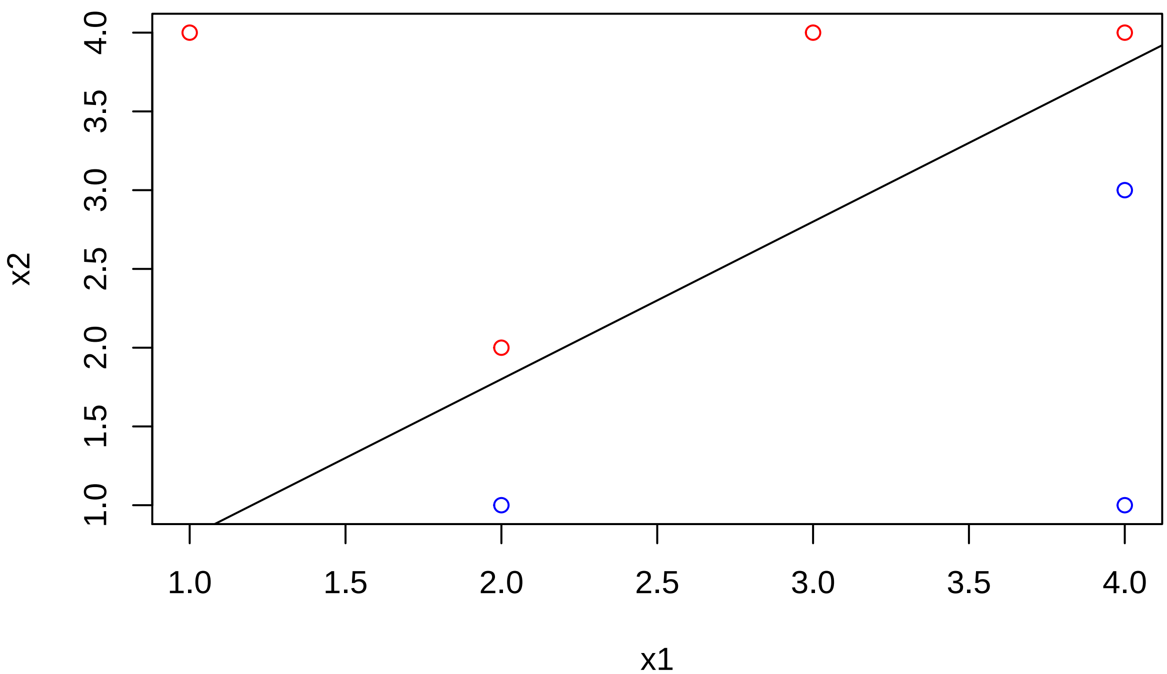
(e) Indicate the support vectors for the maximal margin classifier. The support vectors are the points (2,1), (2,2) and (4,3), (4,4).

(f) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

The seventh observation is (4,1) in color blue. If we move the observation (4,1), we would not change the maximal margin hyperplane since it is not a support vector.

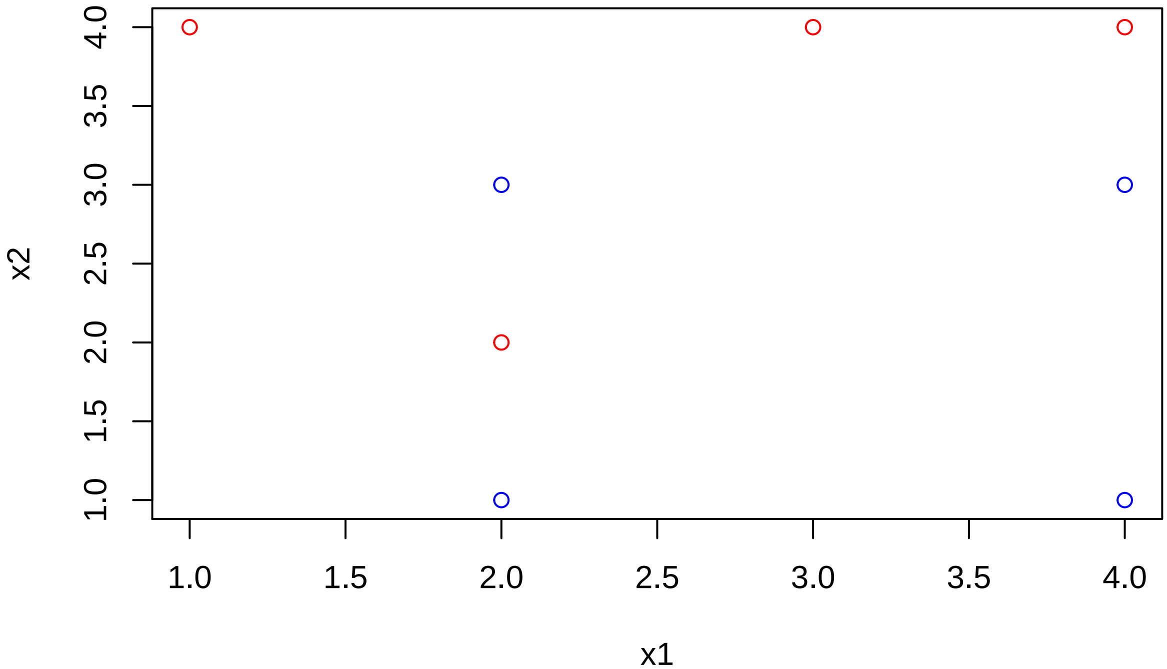
(g) Sketch a hyperplane that is not the optimal separating hyper- plane, and provide the equation for this hyperplane.

```r
plot(x1, x2, col = colors)
abline(-0.2, 1)
```



(h) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

```r
plot(x1, x2, col = colors)
points(2,3, col = "blue")
```



9.5

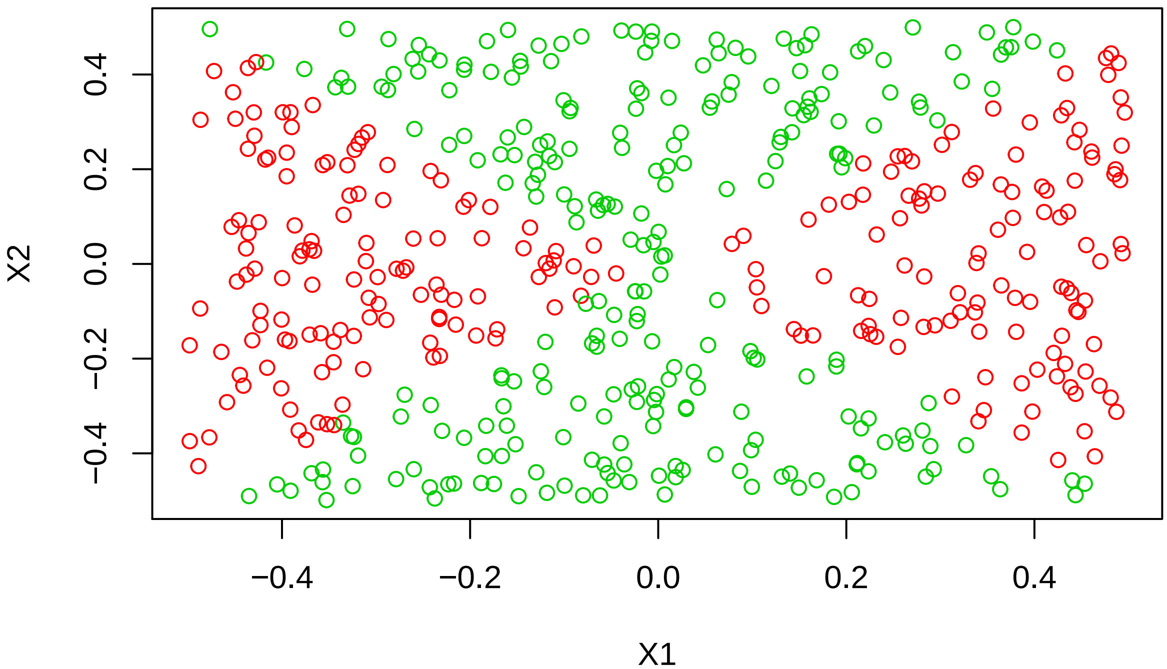
We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.

(a) Generate a data set with n = 500 and p = 2, such that the obser- vations belong to two classes with a quadratic decision boundary between them. For instance, you can do this as follows:

```r
set.seed(1)
x1=runif(500)-0.5
x2=runif(500)-0.5
y=1*(x1^2-x2^2 > 0)
```

(b) Plot the observations, colored according to their class labels. Your plot should display X1 on the x-axis, and X2 on the y- axis.

```r
plot(x1, x2, xlab = "X1", ylab = "X2", col = (11-y))
```



(c) Fit a logistic regression model to the data, using X1 and X2 as predictors.

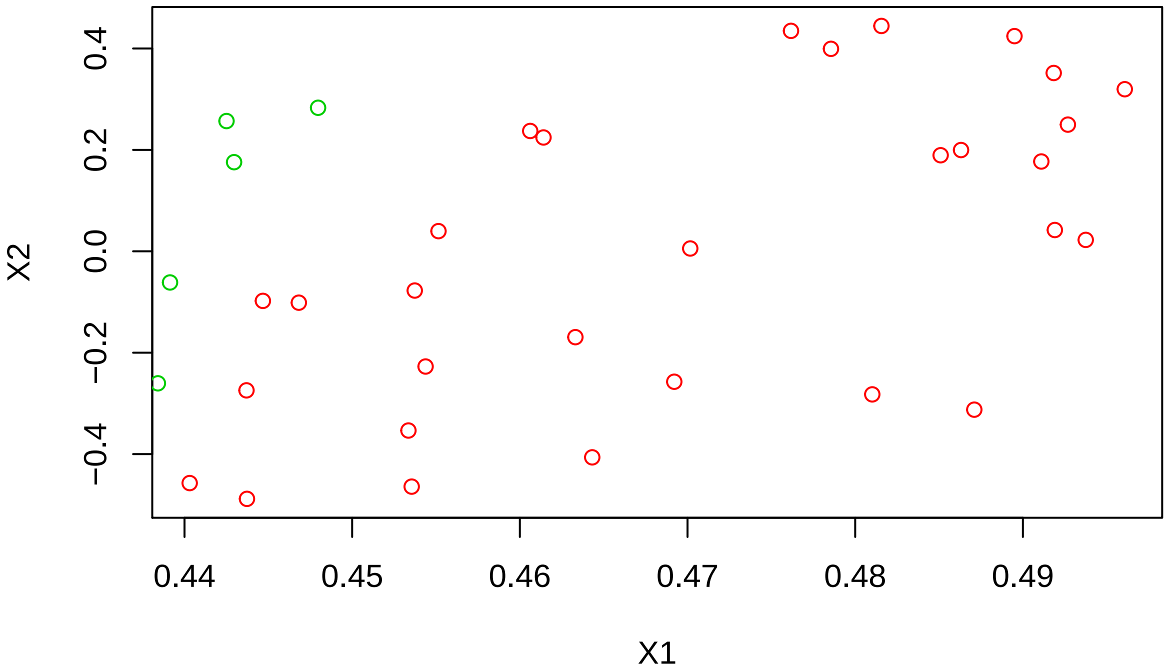
```
logit.fit <- glm(y ~ x1 + x2, family = "binomial")
summary(logit.fit)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = "binomial")
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.179  -1.139  -1.112   1.206   1.257
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.087260   0.089579  -0.974    0.330
## x1           0.196199   0.316864   0.619    0.536
## x2          -0.002854   0.305712  -0.009    0.993
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 692.18  on 499  degrees of freedom
## Residual deviance: 691.79  on 497  degrees of freedom
## AIC: 697.79
##
## Number of Fisher Scoring iterations: 3
```

(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.

```
data <- data.frame(x1 = x1, x2 = x2, y = y)
probs <- predict(logit.fit, data, type = "response")
preds <- rep(0, 500)
```

```r
preds[probs > 0.5] <- 1
plot(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (11 - 1), xlab = "X1", ylab = "X2")
points(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (11 - 0))
```



The boundary is linear.

(e) Now fit a logistic regression model to the data using non-linear functions of X1 and X2 as predictors (e.g. X12, X1 ×X2, log(X2), and so forth).

```r
nl.fit <- glm(y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2)+log(x2), family = "binomial")
```

```
## Warning in log(x2): NaNs produced

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(nl.fit)
```

```
##
## Call:
## glm(formula = y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2) + log(x2),
##     family = "binomial")
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.435e-04  -2.000e-08  -2.000e-08   2.000e-08   4.440e-04
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      249.58   68662.07   0.004    0.997
## poly(x1, 2)1    1882.02  352999.19   0.005    0.996
## poly(x1, 2)2   13795.85 1078776.27   0.013    0.990
## poly(x2, 2)1   -5876.21 1225757.81  -0.005    0.996
## poly(x2, 2)2  -11525.62  946765.65  -0.012    0.990
```

```
## I(x1 * x2)      -122.44  358430.63   0.000    1.000
## log(x2)           37.21    7099.23   0.005    0.996
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3.4497e+02  on 249  degrees of freedom
## Residual deviance: 5.4390e-07  on 243  degrees of freedom
##   (250 observations deleted due to missingness)
## AIC: 14
##
## Number of Fisher Scoring iterations: 25
```

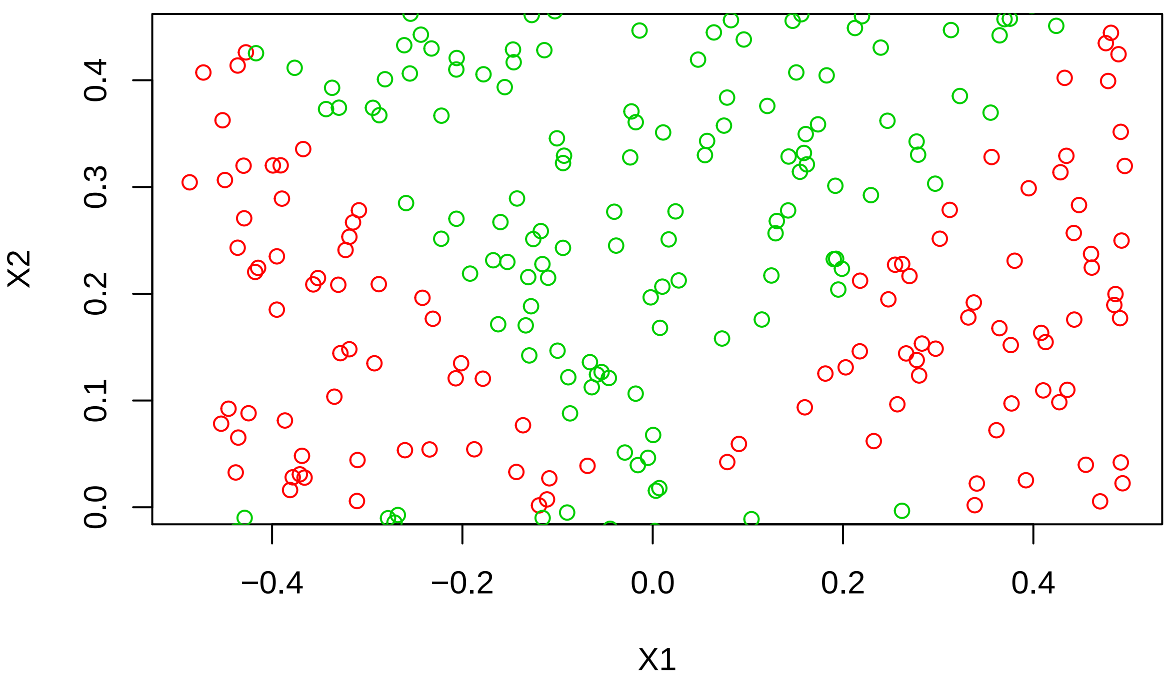
None of the variables are statistically significants.

(f) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the ob- servations, 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.

```
probs <- predict(nl.fit, data, type = "response")
```

```
## Warning in log(x2): NaNs produced
```

```
preds <- rep(0, 500)
preds[probs > 0.5] <- 1
plot(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (11 - 1), xlab = "X1", ylab = "X2")
points(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (11 - 0),)
```



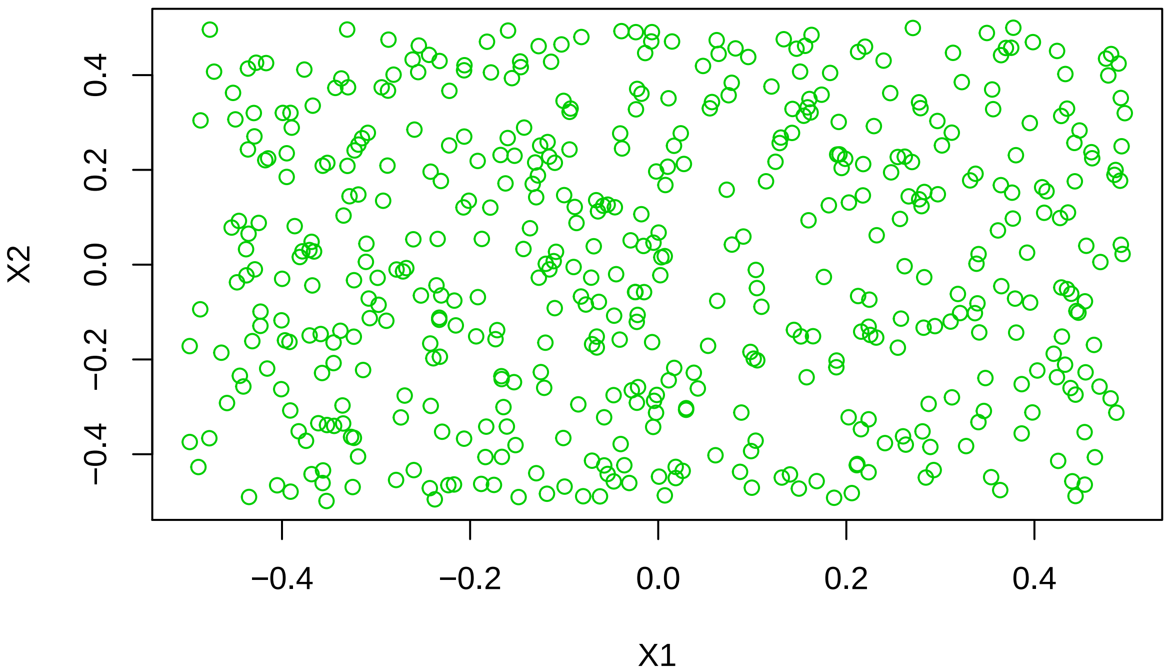
The decision boundary is non-linear.

(g) Fit a support vector classifier to the data with X1 and X2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.2
```

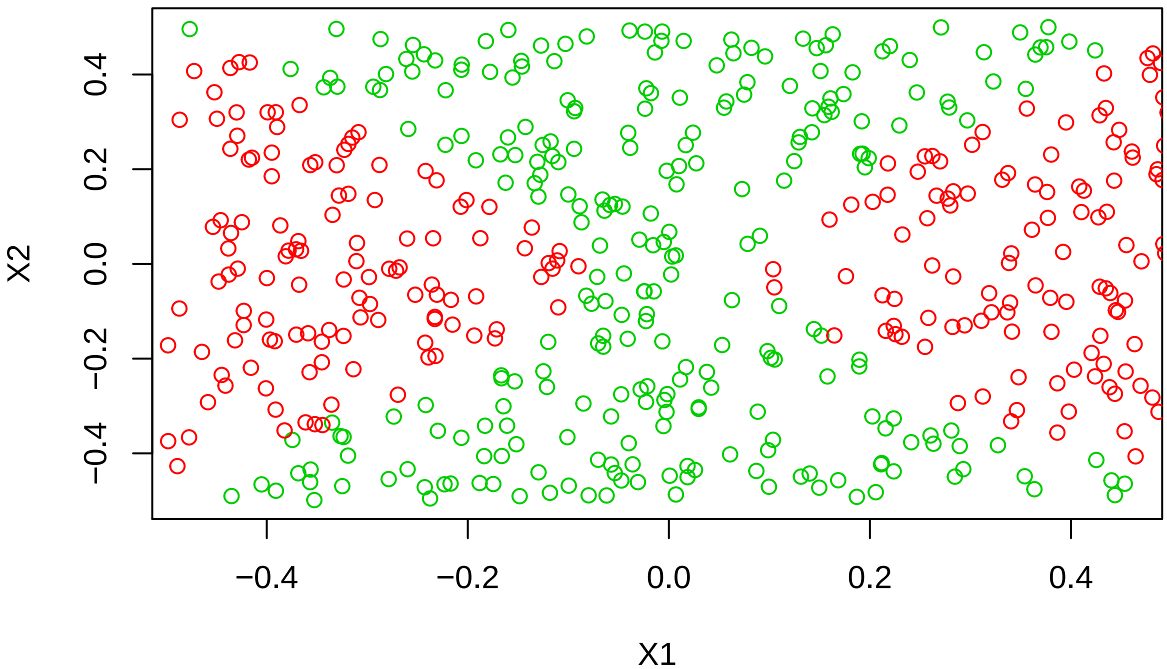
```
data$y <- as.factor(data$y)
svm.fit <- svm(y ~ x1 + x2, data, kernel = "linear", cost = 0.01)
preds <- predict(svm.fit, data)
plot(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (11 - 0),xlab = "X1", ylab = "X2")
points(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (11 - 1))
```



The support vector machine classifies all observation into one class.

(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.

```
svmnl.fit <- svm(y ~ x1 + x2, data, kernel = "radial", gamma = 1)
preds <- predict(svmnl.fit, data)
plot(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (11 - 0), xlab = "X1", ylab = "X2")
points(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (11 - 1))
```



The non-linear decision boundary provided by svm is surprisingly very similar to the true decision boundary (i) Comment on your results.

SVM with linear kernel and logistic regression without any interaction term are not very useful when finding non-linear decision boundaries. The logistic regression with non-linear functions can produce result of non-linear decision boundaries but it requires a lot of work to find the best fit. However, with suportive vector machine, we only need to adjust gamma and get decision boundaries that is very close to the true deision boundary.