viii. Exercise 9.7.5

Kaarthik Sundaramoorthy, Sahil Shah and Vidhi Shah

7/17/2020

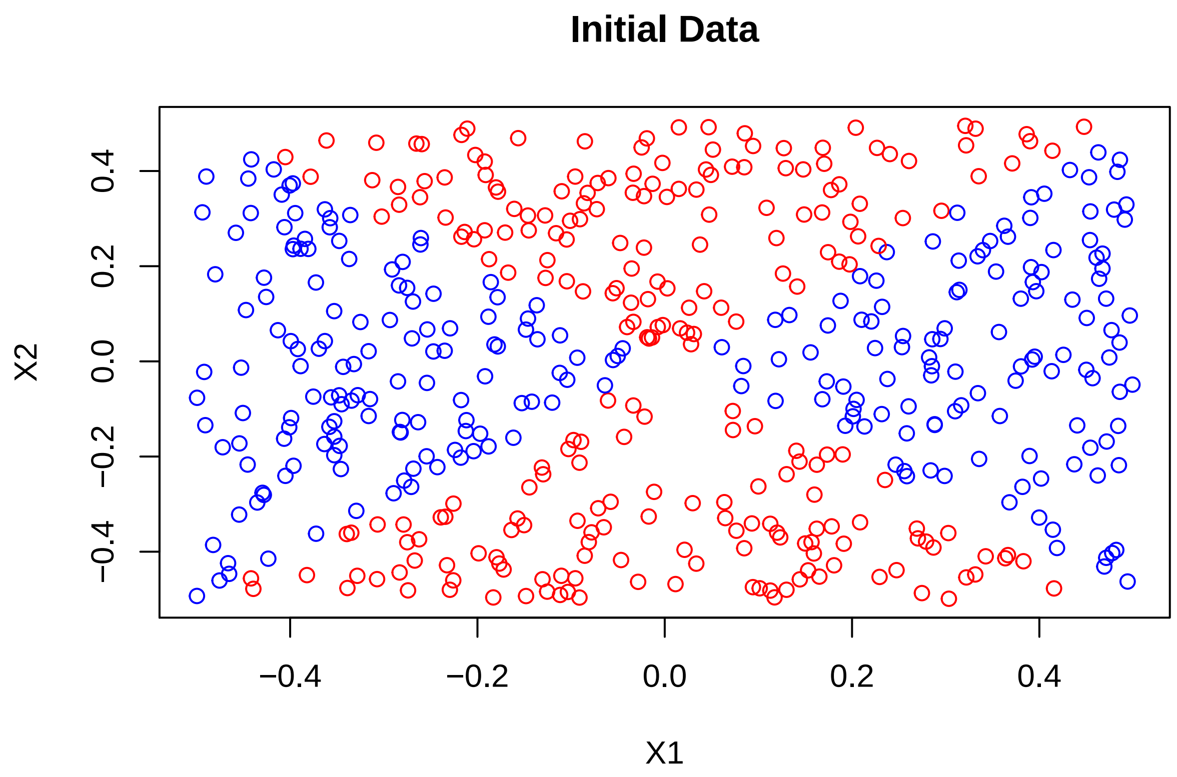
We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.370 9. Support Vector Machines

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

```r
set.seed(123)
n <- 500
p <- 2
x1 <- runif(n) - 0.5
x2 <- runif(n) - 0.5
y <- 1*(x1^2-x2^2 > 0)
```

(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.

```r
plot(x1,x2, col = ifelse(y, 'blue', 'red'), xlab = 'X1', ylab = 'X2', main = "Initial Data")
```



The data defines that these is a non-linear data.

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

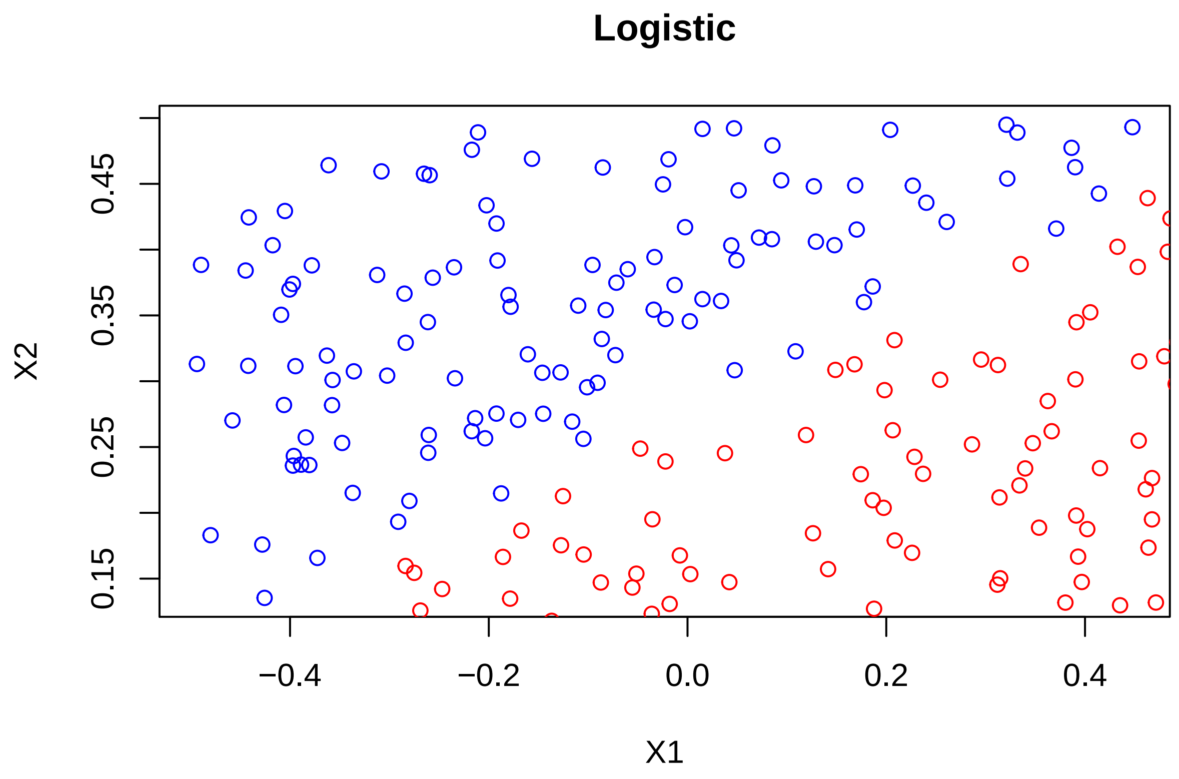
```
# require(glm)
log.fit <- glm(y ~ x1 + x2, family = binomial)
summary(log.fit)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.227  -1.200    1.133    1.157    1.188
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.04792    0.08949   0.535    0.592
## x1          -0.03999    0.31516  -0.127    0.899
## x2           0.11509    0.30829   0.373    0.709
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 692.86  on 499  degrees of freedom
## Residual deviance: 692.71  on 497  degrees of freedom
## AIC: 698.71
##
## Number of Fisher Scoring iterations: 3
```

X_1 and X_2 are insignificant for predicting the y.

(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.

```
df = data.frame(x1 = x1, x2 = x2, y = as.factor(y))
log.prob = predict(log.fit, df, type = "response")
log.pred = ifelse(log.prob>0.52, 1, 0)
df.neg = df[log.pred == 0, ]
df.pos = df[log.pred == 1, ]
plot(df.pos$x1, df.pos$x2, col = "blue", xlab = "X1", ylab = "X2", main = "Logistic")
points(df.neg$x1, df.neg$x2, col = "red")
```



The logistic model, with the probablity threshold of 0.5, The classification of the points are done in single class. We shift the probablity threshold to 0.52 to have the decision boundry. Hence, we get the linear data in the plot.

(e) Now fit a logistic regression model to the data using non-linear functions of X_1 and X_2 as predictors (e.g. X_1^2, $X_1 * X_2$, $log(X_2)$, and so forth).

```r
log.fit2 = glm(y ~ x1 + x2 + I(x1^2) + I(x2^2) + I(x1*x2), data = df, family = "binomial")
```

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

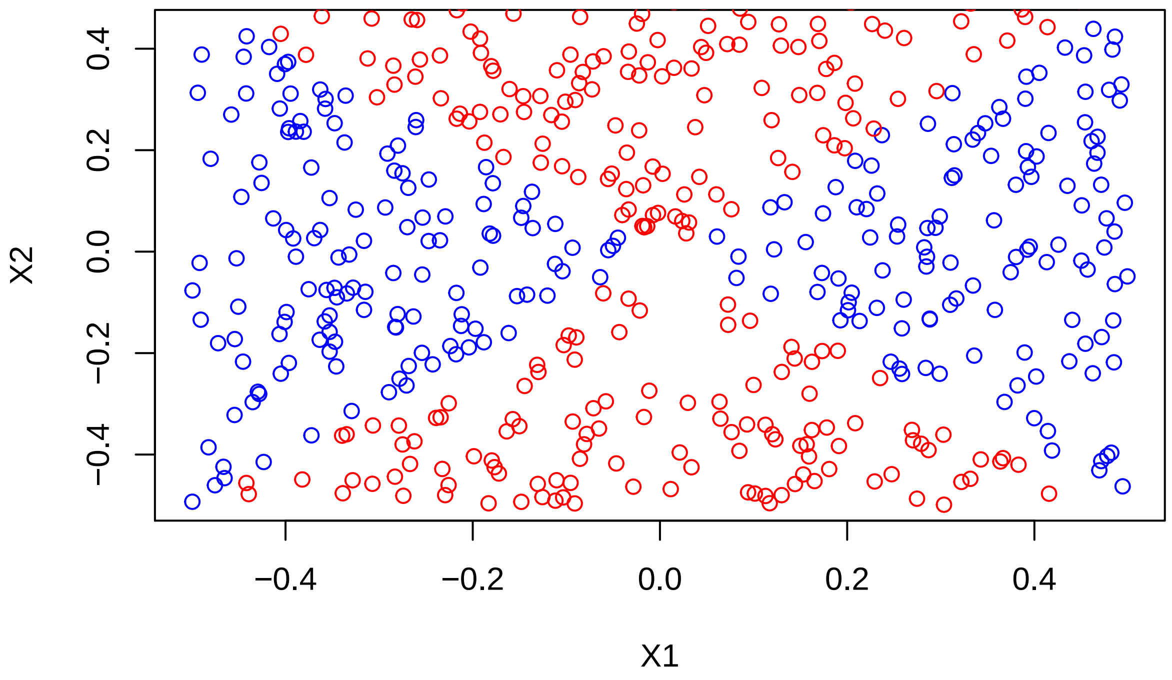
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# log.fit2 = glm(y ~ poly(x1,2) + poly(x2,2) + I(x1 * x2), data = df, family = "binomial")
summary(log.fit2)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2 + I(x1^2) + I(x2^2) + I(x1 * x2), family = "binomial",
##     data = df)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -8.625e-04  -2.000e-08   2.000e-08   2.000e-08   9.604e-04
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -7.12    1170.02  -0.006    0.995
## x1            -146.56   10983.22  -0.013    0.989
## x2              55.75   11290.45   0.005    0.996
## I(x1^2)      12828.87  479505.56   0.027    0.979
## I(x2^2)     -12842.99  481258.81  -0.027    0.979
## I(x1 * x2)     566.19   36927.06   0.015    0.988
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.9286e+02  on 499  degrees of freedom
## Residual deviance: 2.3990e-06  on 494  degrees of freedom
## AIC: 12
##
## Number of Fisher Scoring iterations: 25
```

(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.

```r
log.prob1 = predict(log.fit2, df, type = "response")
log.pred1 = ifelse(log.prob1>0.52, 1, 0)
df.neg1 = df[log.pred1 == 0, ]
df.pos1 = df[log.pred1 == 1, ]
plot(df.pos1$x1, df.pos1$x2, col = "blue", xlab = "X1", ylab = "X2")
points(df.neg1$x1, df.neg1$x2, col = "red")
```



(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*.

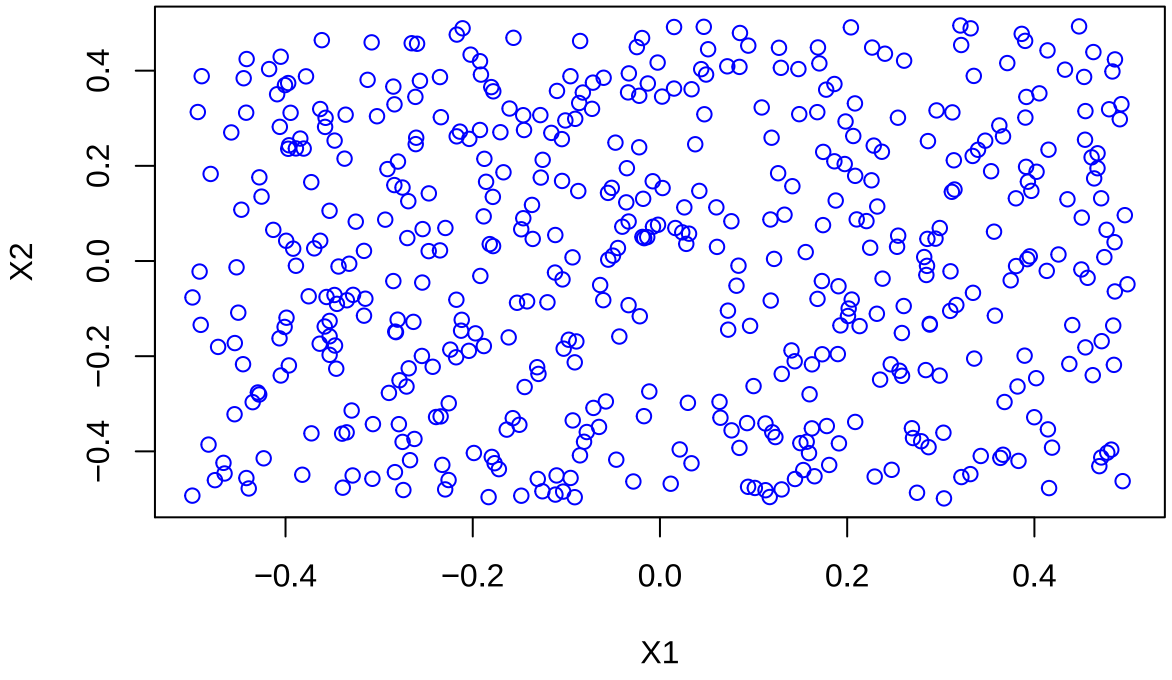
```r
library(e1071)
tune.out <- tune(svm, y ~ ., data = df, kernel = "linear",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.524
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03 0.548 0.07554248
## 2 1e-02 0.548 0.07554248
## 3 1e-01 0.524 0.06095536
## 4 1e+00 0.528 0.05266245
## 5 5e+00 0.528 0.05266245
## 6 1e+01 0.528 0.05266245
## 7 1e+02 0.528 0.05266245
## 8 1e+03 0.526 0.05337498
```

```r
names(tune.out)
```

```
## [1] "best.parameters"  "best.performance" "method"           "nparcomb"
## [5] "train.ind"        "sampling"         "performances"     "best.model"
```

```r
bestmod = tune.out$best.model
y_hat <- predict(bestmod, newdata = data.frame(x1 = x1, x2 = x2))
y_hat <- as.numeric(as.character(y_hat))
plot(x1, x2, col = ifelse(y_hat, "blue", "red"), xlab = "X1", ylab = "X2")
```



Here the linear kernel even with the cost 0.1 fails to find the linear decision boundry and classifies into the 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.*

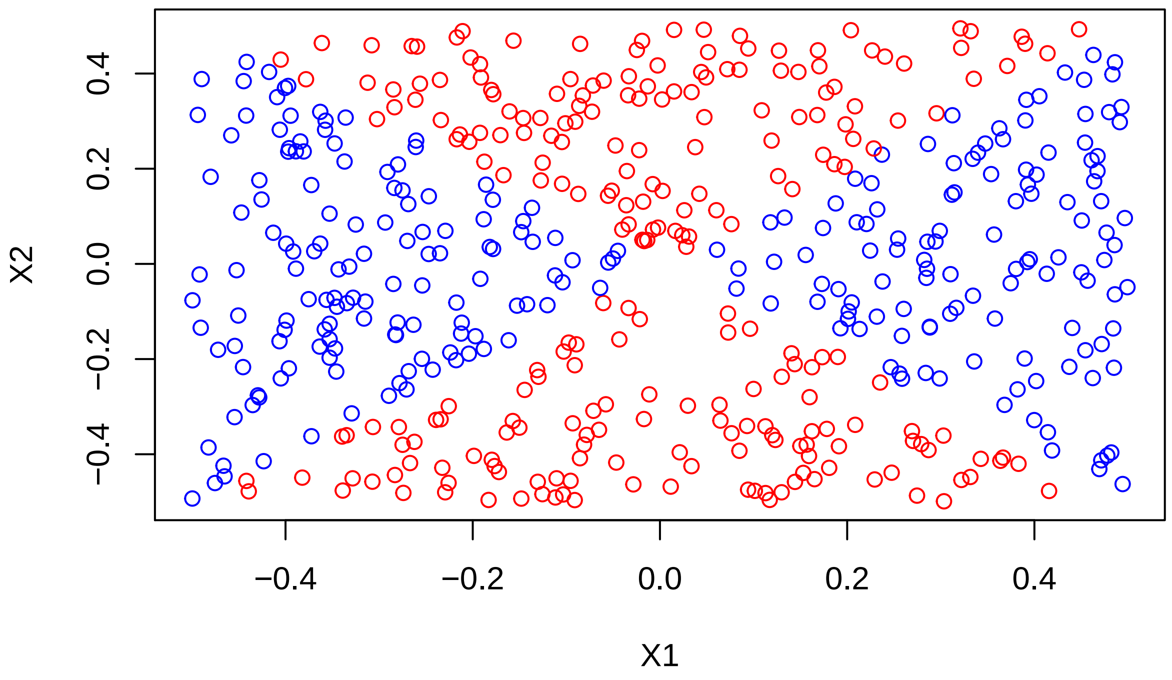
```r
tune.out1 <- tune(svm, y ~ ., data = df, kernel = "radial",
                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000),
                                gamma = c(0.5, 1, 2, 3, 4)))
summary(tune.out1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##  1000   0.5
##
## - best performance: 0.008
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-03   0.5 0.456 0.14630258
## 2  1e-02   0.5 0.456 0.14630258
## 3  1e-01   0.5 0.056 0.04880801
## 4  1e+00   0.5 0.040 0.02494438
## 5  5e+00   0.5 0.030 0.03162278
## 6  1e+01   0.5 0.020 0.02108185
## 7  1e+02   0.5 0.010 0.01414214
## 8  1e+03   0.5 0.008 0.01032796
## 9  1e-03   1.0 0.456 0.14630258
## 10 1e-02   1.0 0.456 0.14630258
## 11 1e-01   1.0 0.050 0.03299832
## 12 1e+00   1.0 0.038 0.02740641
## 13 5e+00   1.0 0.022 0.02201010
## 14 1e+01   1.0 0.024 0.02796824
## 15 1e+02   1.0 0.008 0.01032796
## 16 1e+03   1.0 0.010 0.01414214
## 17 1e-03   2.0 0.460 0.13432961
## 18 1e-02   2.0 0.460 0.13432961
## 19 1e-01   2.0 0.048 0.03552777
## 20 1e+00   2.0 0.040 0.02981424
## 21 5e+00   2.0 0.022 0.01988858
## 22 1e+01   2.0 0.022 0.02201010
## 23 1e+02   2.0 0.014 0.01349897
## 24 1e+03   2.0 0.014 0.01646545
## 25 1e-03   3.0 0.458 0.14030126
## 26 1e-02   3.0 0.458 0.14030126
## 27 1e-01   3.0 0.042 0.02898275
## 28 1e+00   3.0 0.038 0.02898275
## 29 5e+00   3.0 0.026 0.02503331
## 30 1e+01   3.0 0.026 0.02503331
## 31 1e+02   3.0 0.016 0.01577621
## 32 1e+03   3.0 0.014 0.01646545
## 33 1e-03   4.0 0.458 0.14030126
## 34 1e-02   4.0 0.458 0.14030126
## 35 1e-01   4.0 0.040 0.02981424
## 36 1e+00   4.0 0.034 0.02319004
## 37 5e+00   4.0 0.024 0.02065591
## 38 1e+01   4.0 0.022 0.01751190
## 39 1e+02   4.0 0.012 0.01398412
## 40 1e+03   4.0 0.012 0.01398412
```

```r
names(tune.out1)
```

```
## [1] "best.parameters"  "best.performance" "method"           "nparcomb"
## [5] "train.ind"        "sampling"         "performances"     "best.model"
```

```r
bestmod1 = tune.out1$best.model
y_hat1 <- predict(bestmod1, newdata = data.frame(x1 = x1, x2 = x2))
y_hat1 <- as.numeric(as.character(y_hat1))
plot(x1, x2, col = ifelse(y_hat1, "blue", "red"), xlab = "X1", ylab = "X2")
```



(i) Comment on your results.

The practical approach towards the idea of generating the SVM with non-linear kernel are having significantly better approach in finding the non-linear boundry. The logistic regression with no interactions and the SVM generated with linear kernels fails to find the decision boundry. We could get similar kind of results in logistic regression, but by adding non-linear functions, and the process is way more complicated. The kernel based on radial requires one additional tuning parameter i.e. gamma, which is done using cross-validation. Also, SVM linear kernels are acceptable when using a small cost.