HW9

Cathy Su

8/11/2019

Q2

(a)

Use a support vector machine to classify emails as spam or ham. The e1071 package on CRAN provides a svm function that will be useful. Start by using the default parameters of the svm function and report your test-set accuracy.

The test set accuracy with default parameters is about 60%.

```r
# data
spam <- read.csv("../data/gam-spam.csv", header=FALSE)
spam$V58 <- as.factor(spam$V58)
training_rows <- sample.int(nrow(spam), round(nrow(spam) / 3))

spam.train <- spam[training_rows, ]
spam.test <- spam[-training_rows, ]

# apply log transform
cols <- names(spam.train)[1:(length(spam.train) - 1)]
for (col in cols) {
  spam.train[col] <- log(0.1 + spam.train[[col]])
  spam.test[col] <- log(0.1 + spam.test[[col]])
}
#
fit <- svm(V58 ~ ., data = spam.train)
#summary(fit)
pred <- predict(fit, newdata = spam.test, probability = TRUE)
table(spam.test$V58, pred)/length(pred)
```

```
##    pred
##            0        1
##   0 0.000000 0.601239
##   1 0.000000 0.398761
```

(b)

By default, svm uses a radial kernel, but it also supports linear, polynomial, and sigmoid kernels. Each has different tuning parameters, such as the degree of the polynomial kernel or gamma. perform cross-validation to select the best tuning parameters. Build the best classifier for each kernel, tuning over reasonable ranges of the tuning pa- rameters, and report your results. Which kernel performs best on this data? Which kernel performs worst, and why might it be the worst?

These are the tuning parameters for each kernel:

- radial: cost, gamma
- linear: cost
- polynomial: cost, coef0, degree, gamma
- sigmoid: cost, coef0, gamma

I found that tuning the parameters in different function calls vs together seemed to give the same results. Therefore I tuned once, and then printed the outputs as below. The best performing kernel seems to be the radial kernel with about 94% accuracy, but sigmoid and polynomial do about as good. Linear kernel performs the worst. This result could be because the boundary between the classes is not linear.

```
type <- c("linear", "polynomial","sigmoid", "radial")

params <- tune.svm(V58 ~ ., data=spam.train,
                   cost=10^(0:4),
                   gamma=10^(-4:-1),
                   coef0=10^(-1:2),
                   degree=1:4)

####### fit the kernels
for(k in type){
  print(k)
  fit <- svm(V58 ~ ., data = spam.train,
             kernel = k,
             degree=params$best.parameter[[1]],
             cost=params$best.parameter[[4]],
             coef0=params$best.parameter[[3]],
             gamma=params$best.parameter[[2]],
             probability = TRUE)
  #summary(fit)
  pred <- predict(fit, newdata = spam.test)
  t <- table(spam.test$V58, pred)/length(pred)
  print(kable(t))
}
```

```
## [1] "linear"
##
##
##               0           1
## ---  ----------  ----------
## 0     0.5679817   0.0332573
## 1     0.0322791   0.3664819
## [1] "polynomial"
##
##
##               0           1
## ---  ----------  ----------
## 0     0.5758070   0.0254320
## 1     0.0384741   0.3602869
## [1] "sigmoid"
##
##
##               0           1
## ---  ----------  ----------
## 0     0.5735246   0.0277144
## 1     0.0388001   0.3599609
## [1] "radial"
##
##
##               0           1
## ---  ----------  ----------
## 0      0.5787414   0.0224976
## 1      0.0348875   0.3638735
```

(c)

Compare the accuracy youhaveobtainedheretotheaccuracyyouobtainedonHomework 8 while using random forests

With random forests we observed about 5% error. We see that despite small differences based upon kernel type, this is similar to what we achieve with the kernels when tuned.

Q3. An Introduction to Statistical Learning, chapter 9, exercise 5 (pp. 369–370).

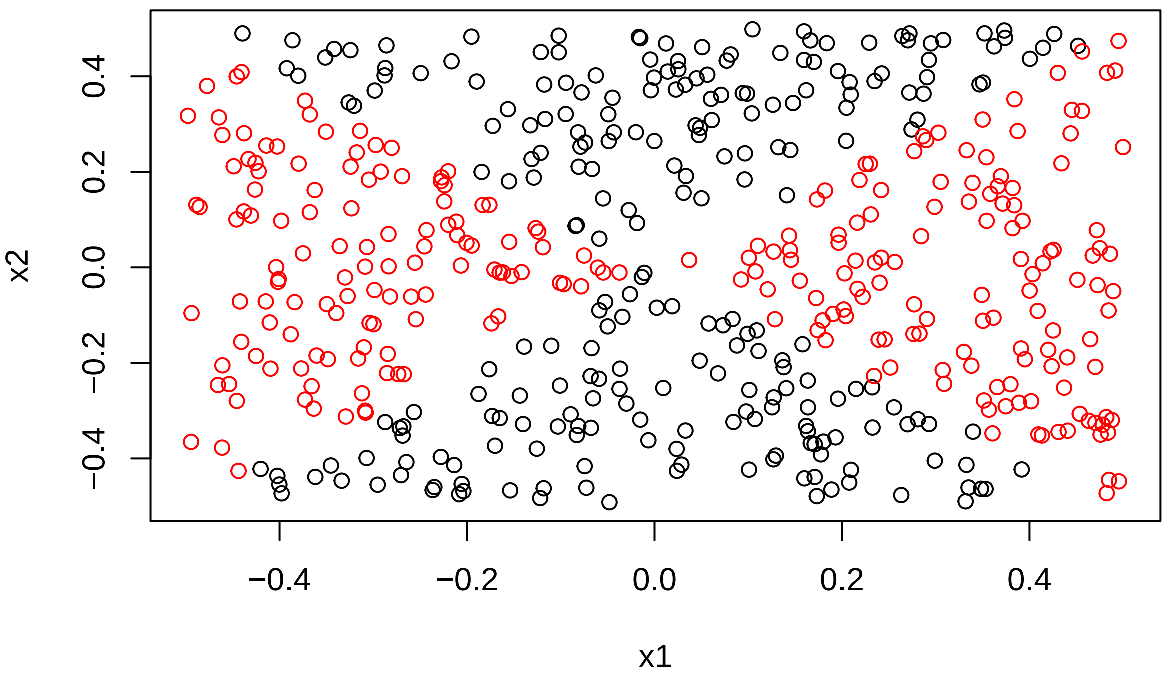
We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.

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

```r
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, col = as.factor(y>0.5))
```

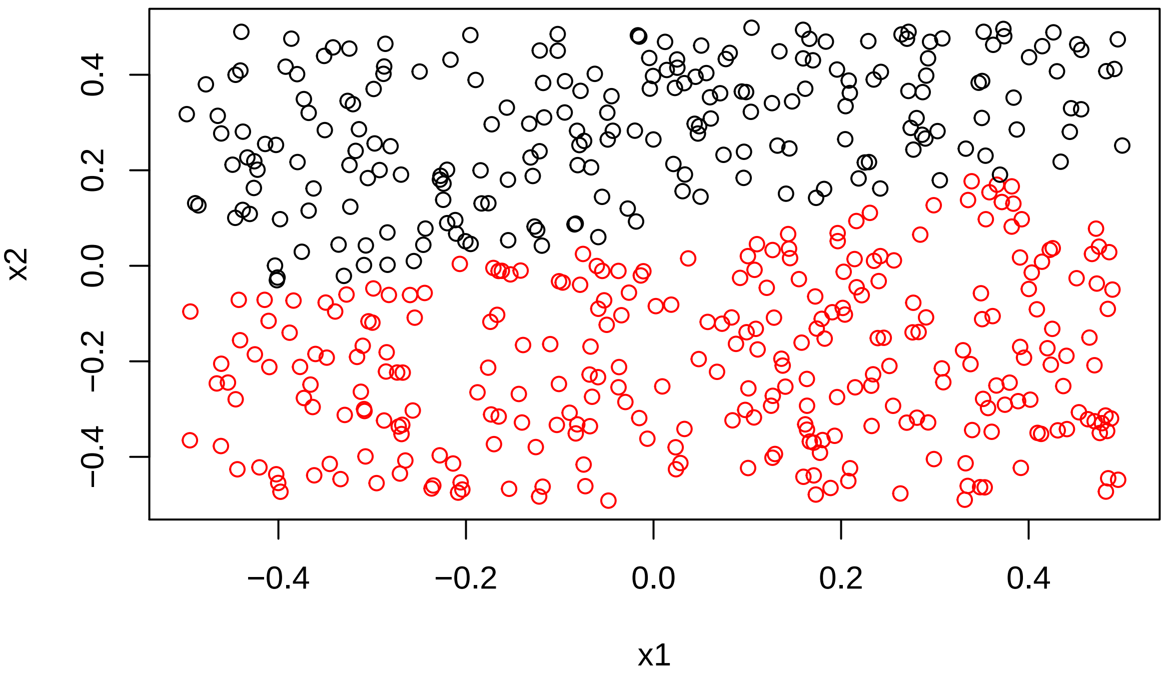


c) Fit a logistic regression model to the data, using X1 and X2 as predictors.

```r
mod <- glm(y ~x1 + x2,family=binomial(link='logit'))
#summary(model)
```

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

```r
p <- predict(mod, type="response")
plot(x1, x2, col = as.factor(p>0.5))
```

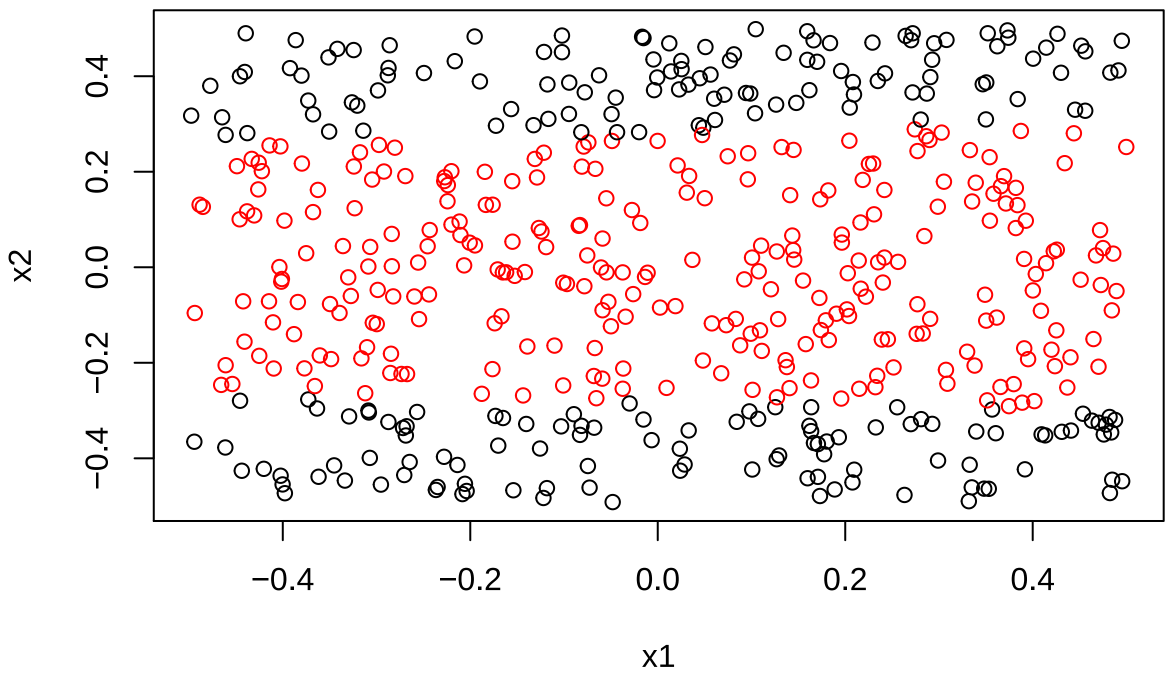


e) Now fit a logistic regression model to the data using non-linear functions of X1 and X2 as predictors (e.g. X12, X1 ×X2, log(X2), and so forth).

```r
model <- glm(y ~x1 + x2 +  poly(x2,2)+ poly(x1,1),family=binomial(link='logit'))
#summary(model)
```

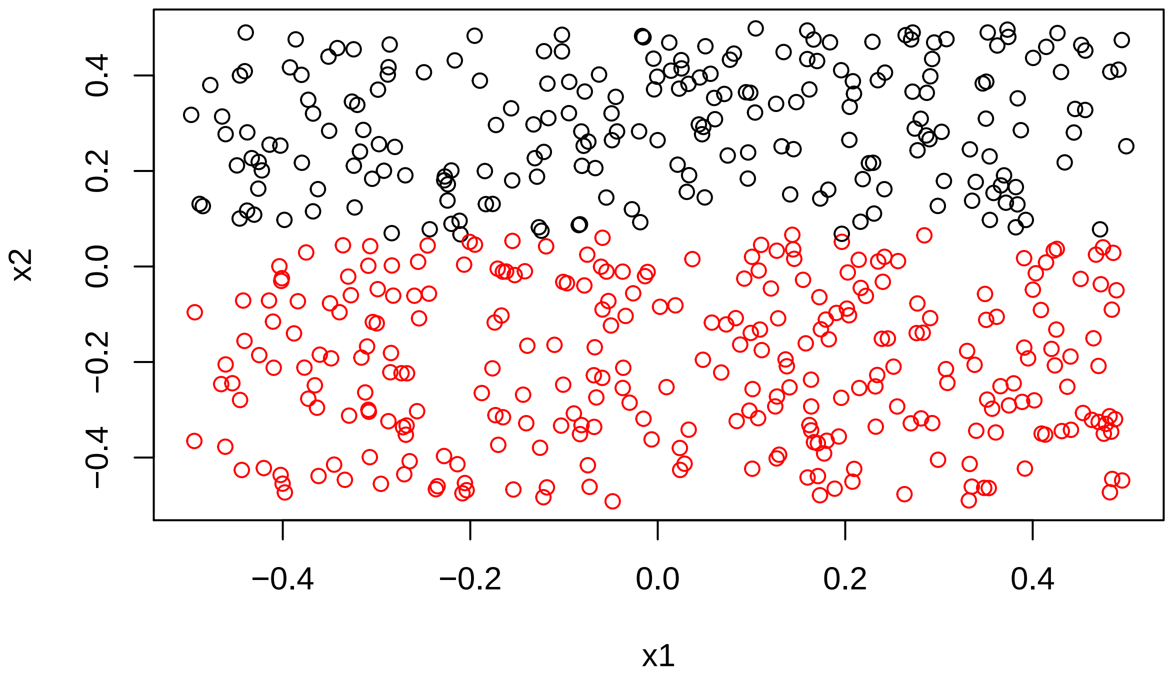
f) Apply this model to the training data in order to obtain a pre- dicted class label for each training observation. Plot the ob- servations, colored according to the predicted class labels. The decision boundary should be obviously non-linear. If it is not, then repeat (a)-(e) until you come up with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

```r
pred_y <- predict.glm(model, type="response")
plot(x1, x2, col = as.factor(pred_y>0.5))
```



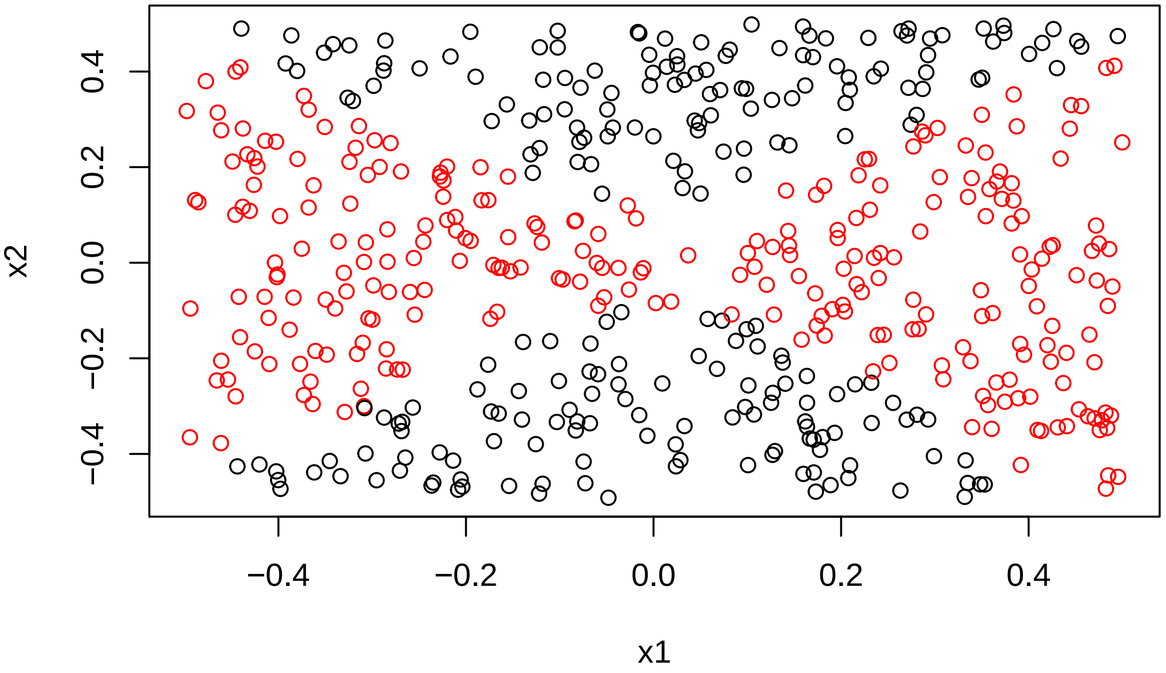
g) Fit a support vector classifier to the data with X1 and X2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

```
fit <- svm(y ~x1 + x2,kernel = "linear")
pred <- predict(fit)
plot(x1, x2, col = as.factor(pred>0.5))
```



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
fit <- svm(y ~x1 + x2, data = spam.train,kernel = "radial")
pred <- predict(fit)
plot(x1, x2, col = as.factor(pred>0.5))
```



i) Comment on your results.

- In what space is the decision boundary linear?
- If we can make logistic regression produce nonlinear decision boundaries by adding ap- propriate covariates, why is the kernel trick in SVMs useful?

We know the data was generated by a quadratic function. The decision boundary for this data becomes linear if we square both x1 and x2 since the assignments are symmetric across the x1 and x2 axis.

We can get the nonlinear decision boundary through logistic regression, but it requires us to manually decide on what the new covariates should be. Also as we see above the fit is not as good as with the SVM with radial kernel, so we need to invest additional time to tune the logistic regression fit.

Q4

Should we set C large or small if we want our estimated boundary to have the minimum possible variance?

If the parameter C is large, then any wrong classification is penalized a lot. This would lead to higher variance in general since the classifier tries to choose a more flexible decision boundary so there will be no datapoints within the margins. Thus to have smaller variance we should set C to be small.

Q5

Unbalanced classes. Simulate a dataset with two variables, X1 and X2, and an outcome variable Y. Make the dataset have a linear decision boundary between Y = 1 and Y = 0. But put the code that simulates data into a function, and make one of the parameters of that function be the fraction of cases that should have Y = 1. This way, we can simulate what happens when the classes are imbalanced.

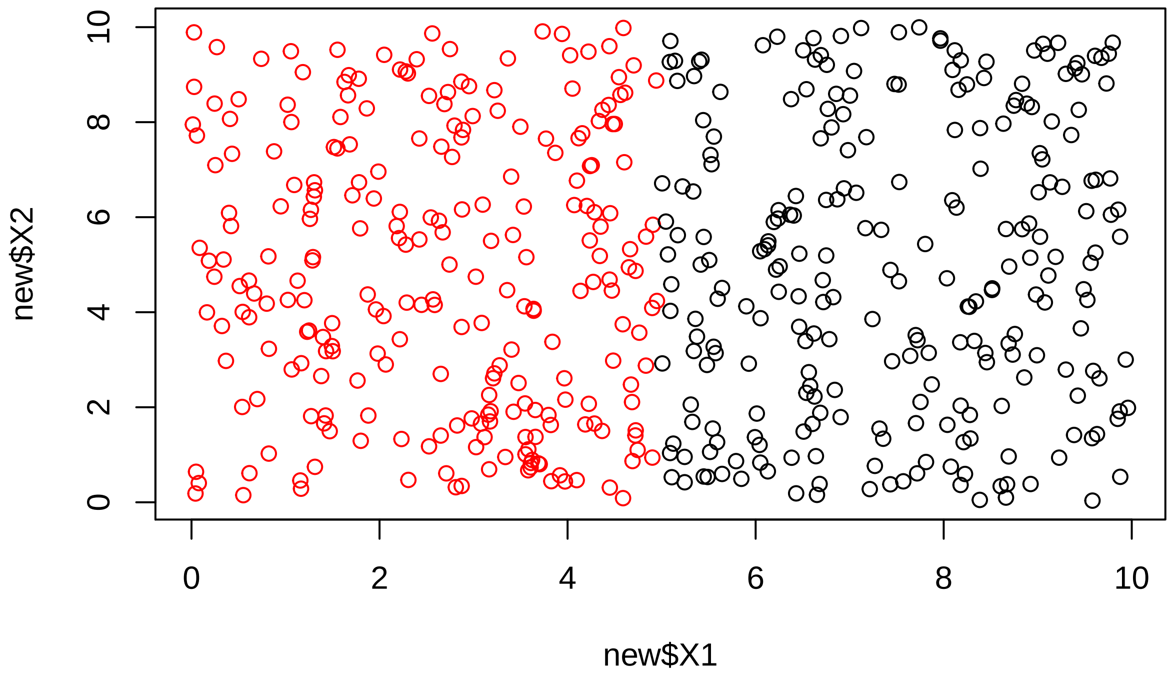
```r
simulate <- function(fraction) {
  length= 10.0
  X1  <- matrix(runif(500, min=0, max=10),ncol=1)
  X2 <- matrix(runif(500, min=0, max=10),ncol=1)
  Y <- ifelse((X1> length * fraction), 0, 1) #linear decision boundary
  # Y <- ifelse((X1 > length*fraction & X2 > length*fraction), 0, 1)

  #stopifnot(nrow(X1) == length(Y)) # sanity check!
  new <- cbind(as.data.frame(X1), as.data.frame(X2), as.data.frame(Y))
  colnames(new)  <- c("X1", "X2", "Y")
  new$Y <- as.factor(new$Y)
  #colnames(new)[-1] <- "Y"
  return(new)
}
```

a)

Set the fraction to 0.5: perfect balance. Run a logistic regression and an SVM on the data to predict Y. Compare the confusion matrices on a test set (just run your function again to get a test set!). How do the error rates compare, and do the error rates differ for points with Y = 0 vs. those with Y = 1?

The error rates for logistic and SVM classifiers are similar, though the SVM does slightly better. Furthermore with balanced classes the error rates across the classes is similar for both classifiers.

```r
new <- simulate(0.5)
plot(new$X1, new$X2, col=as.factor(new$Y))
```



```r
new_test <- simulate(0.5)
# run SVM
fit <- svm(Y ~X1+X2, data = new, probability = TRUE)
pred <- predict(fit, newdata = new_test)
table(new$Y, pred)/length(pred)
```

```
##     pred
##          0     1
##    0 0.244 0.256
##    1 0.246 0.254
```

```
# run LR
model <- glm(Y~X1+X2,  data = new,family=binomial(link='logit'))
pred <- predict(model, newdata = new_test)
table(new$Y, as.numeric(pred > 0.5))/length(pred)
```

```
##
##          0     1
##    0 0.242 0.258
##    1 0.244 0.256
```

b)

Now repeat this analysis with fractions varying from 0.6 to 0.95. Look at the confusion matrices as the classes become less balanced.What happens to the errors for logistic regression? What kinds of errors become more common?

We have shown the results for increasing fraction of Y=0 cases below for both logistic and svm classifiers. We see that as the fraction increases, the error increases overall. It seems that logistic regression does well on the dataset but missclassification of the Y=1 cases becomes more common as the dataset is unbalanced.

```
frac <- seq(0.6, 0.95, length.out= 5)
svm_error_0 <- vector(mode="numeric", length=length(frac))
svm_error_1 <- vector(mode="numeric", length=length(frac))
lr_error_0 <- vector(mode="numeric", length=length(frac))
lr_error_1 <- vector(mode="numeric", length=length(frac))

for(i in length(frac)){
  f = frac[i]
  #print(f)
  new <- simulate(fraction = f)
  new_test <- simulate(fraction = f)
  ####### logistic
  model <- glm(Y ~X1+X2,data = new,family=binomial(link='logit'))
  #  R will output probabilities in the form of P(y=1|X)
  pred <- predict(model, newdata = new_test, type = "response")
  pred <- ifelse(pred > 0.5, 1, 0)
  error <- pred[pred != new_test$Y]
  #print(kable(table(new$Y, as.numeric(pred > 0.5))/length(pred)))
  lr_error_0[i] <- length(error[error==0])/length(pred)
  lr_error_1[i] <- length(error[error==1])/length(pred)

  ####### SVM
  fit <- svm(Y ~X1+X2, kernel= "linear", data = new)
  pred <- predict(fit, newdata = new_test, probability = FALSE)
  #print(kable(table(new$Y, pred)/length(pred)))
  error <- pred[pred != new_test$Y]
  svm_error_0[i] <-  length(error[error ==0])/length(pred)
  svm_error_1[i] <- length(error[error==1])/length(pred)
}

df <- as.data.frame(cbind(frac, lr_error_0, lr_error_1,
            svm_error_0, svm_error_1))
temp <- melt(df, id.vars = 'frac', variable.name = 'prop.error')
# ggplot(temp, aes(frac,value))+ geom_line(aes(colour = prop.error))
df
```

```
##     frac lr_error_0 lr_error_1 svm_error_0 svm_error_1
## 1 0.6000      0.000          0       0.000       0.000
## 2 0.6875      0.000          0       0.000       0.000
## 3 0.7750      0.000          0       0.000       0.000
## 4 0.8625      0.000          0       0.000       0.000
## 5 0.9500      0.002          0       0.004       0.024
```

c)

What happens to the errors for SVMs in the same case? Show a graph or table summarizing the results and comparing SVMs to logistic regression.

The outputs (above) show that for the svm classifier, missclassification of the Y=1 cases becomes more common with increasing fraction just as with logistic regression. However svms do comparatively worse.

d)

Describe, in words, why the difference you observed should exist. Give your explanation in terms of how SVMs and logistic regression find the decision boundary, and what loss functions they use.

The SVM outputs predictions whereas logistic regression gives probabilities. For a highly imbalanced dataset, the logistic regression model will still give some probability for test samples to belong to the rare class. However the classification boundary of SVMs is dependent on the datapoints near the boundary only and is penalized here with a default cost. Since the rare class is not well represented, the penalty for incorrect predictions on the rare class is low so the svm model may choose a boundary that does not perform well on the test set.