Untitled

Jingpeng

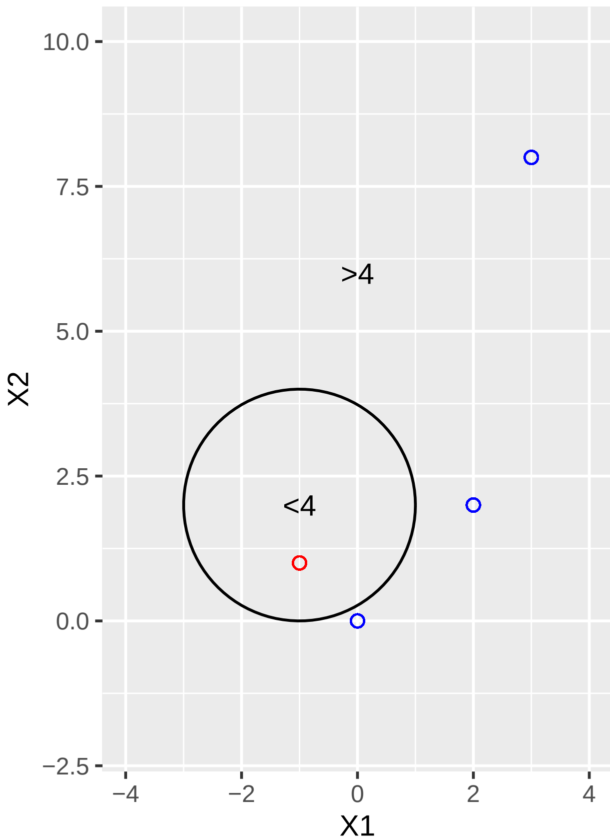
3/5/2022

Question 2

a-c

```
## create the circle function
circleFun = function(center = c(-1, 2), r = 2, npoints = 100){
    tt <- seq(0, 2*pi, length.out = npoints)
    xx <- center[1] + r * cos(tt)
    yy <- center[2] + r * sin(tt)
    return(data.frame(x = xx, y = yy))
}

## sketch the curve
data = circleFun(c(-1, 2), 2, npoints = 100)
ggplot(data,aes(x,y)) +
  geom_path() +
  xlim(-4, 4) +
  xlab("X1") +
  ylim(-2, 10) +
  ylab("X2") +
  annotate("text", x = -1, y = 2, label = "<4") +
  annotate("text", x = 0, y = 6, label= ">4") +
  geom_point(aes(x = 0, y = 0), colour = 'blue', fill = NA, size = 2, shape = 21)+
  geom_point(aes(x = -1, y = 1), colour = 'red', fill = NA, size = 2, shape = 21)+
  geom_point(aes(x = 2, y = 2), colour = 'blue', fill = NA, size = 2, shape = 21)+
  geom_point(aes(x = 3, y = 8), colour = 'blue', fill = NA, size = 2, shape = 21)+
  coord_fixed()
```



d.

We can rewrite the non-linear boundary:

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

as

$$1 + 2X_1 - 4X_2 + X_1^2 + X_2^2 = 0$$

This boundary is linear in terms of X_1, X_1^2, X_2, and X_2^2.

Question 4

```
library(ISLR)
# install.packages('e1071')
library(e1071)
Auto = Auto
```

a.

Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```
Auto$mpg2 = ifelse(Auto$mpg>median(Auto$mpg), 1, 0) %>%
  as.factor()
```

b.

```
set.seed(1)
## cross validation
tune.out=tune(svm, mpg2~., data=Auto, kernel="linear",
```

```
             ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01025641
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.09442308 0.04519425
## 2 1e-02 0.07653846 0.03617137
## 3 1e-01 0.04596154 0.03378238
## 4 1e+00 0.01025641 0.01792836
## 5 5e+00 0.02051282 0.02648194
## 6 1e+01 0.02051282 0.02648194
## 7 1e+02 0.03076923 0.03151981
```

From the results above we find that `cost=1` results in the lowest cross-validation error rate. We can see that when `cost` is large, the margins will be narrow. The classifier will fit the training data well, which may have low bias but high variance. On the contrary, when `cost` is small, the margins will be wide, which may have low variance but high bias. `cost=1` can control the bias-variance trade-off best among these values. Now we use the best model obtained through cross-validation to make predictions.

```
bestmod = tune.out$best.model
ypred = predict(bestmod, Auto)
table(predict = ypred, truth = Auto$mpg2)
```

```
##        truth
## predict   0   1
##       0 196   1
##       1   0 195
```

c