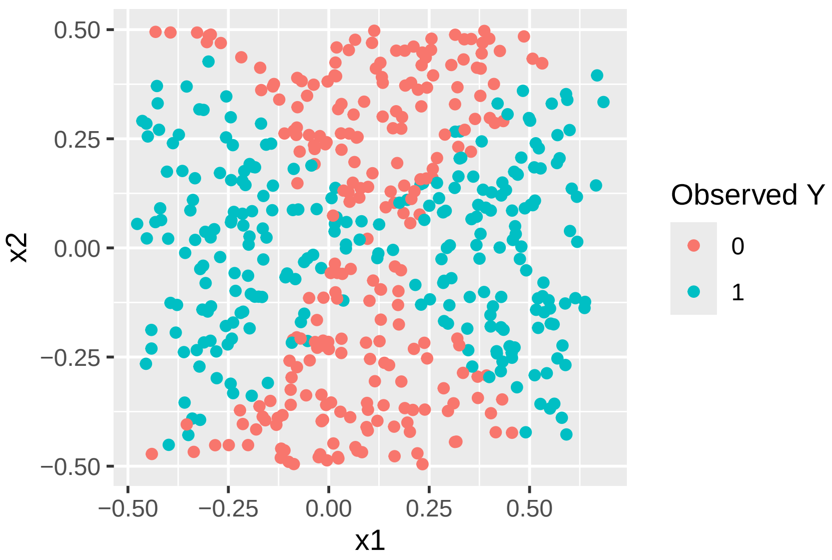
STAA 577: HW8

Your Name

Problem 1a

Problem 1b



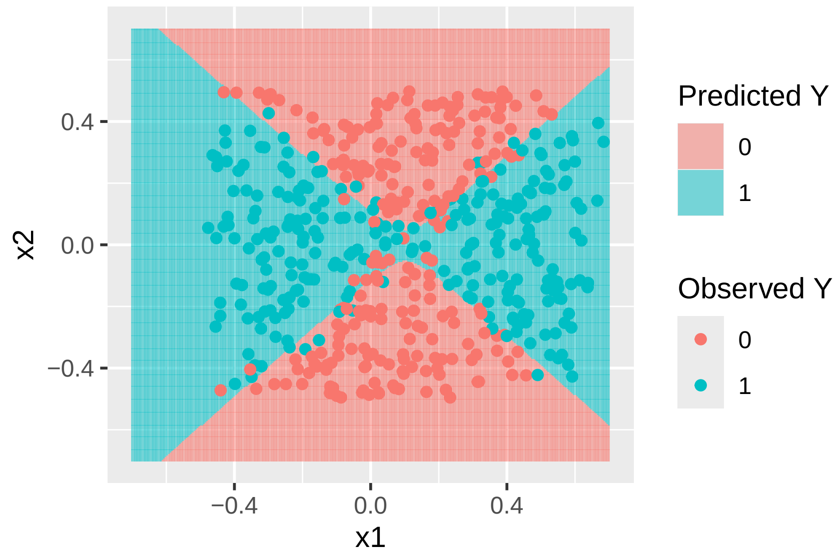
Problem 1c

Problem 1d

The decision boundary is like the two mountains kissing.

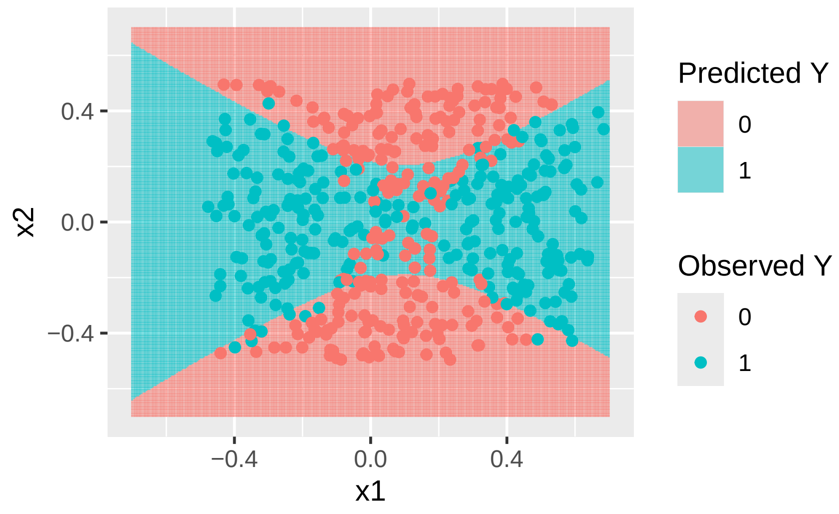
Problem 1e

Problem 1f

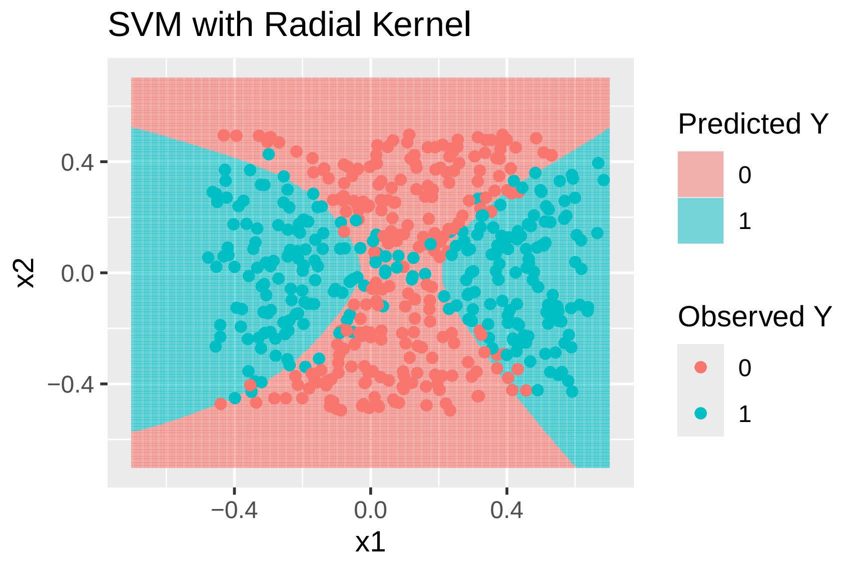


Problem 1g

SVM Decision Regions



Problem 1h



Problem 1i

Both SVMs and the polynomial logistic regression seem to provide reasonable fit to the data. That said, the radial kernal svm model appears to do the best job as it uncovers the difference in shape betwen the left and right side of the graph.

Problem 2a

Problem 2b

- optimal \$C =\$1.715
- optimal γ =.001

Problem 2c

estimated error of 0.1682243

Problem 3a

Problem 3b

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.09361123
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.10387087 0.03195142
## 2 1e-01 0.10048264 0.03290610
## 3 1e+00 0.09361123 0.02590190
## 4 1e+01 0.10596267 0.02558499
## 5 1e+02 0.12352271 0.02409274
```

It looks like a cost of 1 performs the best. We tested at orders of magnitude so we can now zoom our search in on this part of the parameters space

Problem 3c

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.1
##
## - best performance: 0.06356712
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-02 0.001 0.48990144 0.04225311
## 2  1e-01 0.001 0.26571183 0.04670781
## 3  1e+00 0.001 0.10383169 0.03908420
## 4  1e+01 0.001 0.10012350 0.04493836
## 5  1e+02 0.001 0.08996923 0.04449298
## 6  1e-02 0.010 0.30766109 0.04292230
## 7  1e-01 0.010 0.09924957 0.03834705
## 8  1e+00 0.010 0.08743307 0.04116740
## 9  1e+01 0.010 0.08379120 0.04595083
## 10 1e+02 0.010 0.07600181 0.03654256
## 11 1e-02 0.100 0.15307012 0.03585776
## 12 1e-01 0.100 0.07308524 0.04077768
## 13 1e+00 0.100 0.07304506 0.04566156
## 14 1e+01 0.100 0.06356712 0.03346940
## 15 1e+02 0.100 0.08101275 0.03456475
## 16 1e-02 1.000 0.49454504 0.04226970
## 17 1e-01 1.000 0.29466110 0.04192120
## 18 1e+00 1.000 0.09788962 0.02316286
## 19 1e+01 1.000 0.10333516 0.02501322
## 20 1e+02 1.000 0.10333365 0.02502742
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      3
##
## - best performance: 0.2463491
##
## - Detailed performance results:
##      cost degree     error dispersion
## 1  1e-02      2 0.5072215 0.04369350
## 2  1e-01      2 0.5048997 0.04446428
## 3  1e+00      2 0.4823814 0.05299827
## 4  1e+01      2 0.3384820 0.09585926
## 5  1e+02      2 0.2525576 0.06848493
## 6  1e-02      3 0.5074118 0.04362167
## 7  1e-01      3 0.5068510 0.04374719
## 8  1e+00      3 0.5011169 0.04489715
## 9  1e+01      3 0.4476439 0.05867673
## 10 1e+02      3 0.2463491 0.04661940
```

See output above

Problem 4

```
## [1] "SVM Linear Kernel Accuracy: 0.803"
```

```
## [1] "SVM Polynomial Kernel (Degree 2) Accuracy: 0.743"
```

```
## [1] "SVM Radial Kernel Accuracy: 0.836"
```

```
## [1] "Logistic Regression Accuracy: 0.783"
```

Using the boston data set it appears that the polynomial kernal SVM performs the best.This is actually not that far from the logisict regression accuracy which is surprising.

Problem 5

You can include the answer as a picture or generate it using R.

Appendix

```
library(knitr)
# install the tidyverse library (do this once) install.packages('tidyverse')
library(tidyverse)
# set chunk and figure default options
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, fig.width = 4.5,
    fig.height = 3, tidy = TRUE)
library(tidyverse)  # data manip
library(ISLR)  # data
library(GGally)  # pairs plots
library(e1071)  #svm
set.seed(577)
n <- 500

# simulate x's
x1 <- runif(n) - 0.5
x2 <- runif(n) - 0.5

# simulate deterministic y
y <- as.numeric(x1^2 - x2^2 > 0)

# add some randomness so that y is not deterministic
x1 <- x1 + runif(n, min = 0, max = 0.2)

# put results into a data frame
mydf <- data.frame(x1, x2, y)
# Problem 1b
library(ggplot2)
ggplot(mydf, aes(x = x1, y = x2, color = as.factor(y))) + geom_point() + labs(color = "Observed Y")

# Problem 1c
glmOut <- glm(y ~ x1 + x2, family = "binomial", data = mydf)

# Problem 1d
xgrid <- expand.grid(x1 = seq(-0.7, 0.7, length.out = 300), x2 = seq(-0.7, 0.7, length.out = 300))

yhat_grid <- predict(glmOut, newdata = xgrid, type = "response")
yhat_grid <- (yhat_grid > 0.5) * 1
```

```r
xgrid$yhat <- yhat_grid

gfig <- ggplot() + geom_tile(aes(x = x1, y = x2, fill = as.factor(yhat)), data = xgrid,
    alpha = 0.5) + geom_point(aes(x = x1, y = x2, col = as.factor(y)), data = mydf) +
    labs(fill = "Predicted Y") + labs(col = "Observed Y")

gfig
# Problem 1e
glmOut <- glm(y ~ poly(x1, 2) + poly(x2, 2), family = "binomial", data = mydf)
# Problem 1f

xgrid <- expand.grid(x1 = seq(-0.7, 0.7, length.out = 300), x2 = seq(-0.7, 0.7, length.out = 300))

yhat_grid <- predict(glmOut, newdata = xgrid, type = "response")
xgrid$yhat <- as.numeric(yhat_grid > 0.5)

ggplot() + geom_tile(aes(x = x1, y = x2, fill = as.factor(yhat)), data = xgrid, alpha = 0.5) +
    geom_point(aes(x = x1, y = x2, color = as.factor(y)), data = mydf) + labs(fill = "Predicted Y",
    color = "Observed Y")

library(ggplot2)
svmOut <- svm(y ~ x1 + x2, data = mydf, kernel = "polynomial", degree = 2)

yhat_grid <- predict(svmOut, newdata = xgrid)

xgrid$yhat <- as.numeric(yhat_grid > 0.5)

p <- ggplot() + geom_tile(aes(x = x1, y = x2, fill = factor(yhat)), data = xgrid,
    alpha = 0.5) + geom_point(aes(x = x1, y = x2, color = factor(y)), data = mydf) +
    labs(fill = "Predicted Y", color = "Observed Y") + ggtitle("SVM Decision Regions")

print(p)



# Problem 1h
svmOut_radial <- svm(y ~ x1 + x2, data = mydf, kernel = "radial")

yhat_svm_radial <- predict(svmOut_radial, newdata = xgrid)
xgrid$yhat_svm_radial <- as.numeric(yhat_svm_radial > 0.5)

gfig_svm_radial <- ggplot() + geom_tile(data = xgrid, aes(x = x1, y = x2, fill = as.factor(yhat_svm_radi
    alpha = 0.5) + geom_point(data = mydf, aes(x = x1, y = x2, color = as.factor(y))) +
    labs(fill = "Predicted Y", color = "Observed Y") + ggtitle("SVM with Radial Kernel")

gfig_svm_radial

# Problem 2a
library(ISLR)
data(OJ)

OJ$StoreID <- as.factor(OJ$StoreID)
```

```r
OJ <- dplyr::select(OJ, -STORE)

set.seed(577)

tune_out <- tune(svm, kernel = "radial", scale = T, Purchase ~ ., data = OJ, ranges = list(cost = seq(0
    3, length = 8), gamma = seq(0.001, 0.5, length = 8)))


# Problem 2b This is throwing an error on knit but works when im not knitting

# tune_out$best.parameters Problem 2c

# This is throwing an error at knit but not when i am not knitting.
# tune_out$best.performance Problem 3a
median_mpg <- median(Auto$mpg)
Auto$mpg01 <- ifelse(Auto$mpg > median_mpg, 1, 0)
# Problem 3b
set.seed(577)

svmTune <- tune(svm, mpg01 ~ . - mpg, data = Auto, kernel = "linear", ranges = list(cost = c(0.01,
    0.1, 1, 10, 100)))
summary(svmTune)

# Problem 3c Radial kernel tuning
svmTuneRadial <- tune(svm, mpg01 ~ . - mpg, data = Auto, kernel = "radial", ranges = list(cost = c(0.01
    0.1, 1, 10, 100), gamma = c(0.001, 0.01, 0.1, 1)))
summary(svmTuneRadial)

# Polynomial kernel tuning (e.g., degree 2)
svmTunePoly <- tune(svm, mpg01 ~ . - mpg, data = Auto, kernel = "polynomial", ranges = list(cost = c(0.
    0.1, 1, 10, 100), degree = c(2, 3)))
summary(svmTunePoly)

library(MASS)

data(Boston)

median_medv <- median(Boston$medv)
Boston$medv01 <- as.factor(ifelse(Boston$medv > median(Boston$medv), 1, 0))

# Remove the original continuous variable (to avoid leakage)
boston_data <- Boston %>%
    dplyr::select(-medv)

set.seed(577)
train_idx <- sample(1:nrow(boston_data), size = floor(0.7 * nrow(boston_data)))
train_data <- boston_data[train_idx, ]
test_data <- boston_data[-train_idx, ]

# SVM with Linear Kernel
svm_linear <- svm(medv01 ~ ., data = train_data, kernel = "linear")
pred_linear <- predict(svm_linear, newdata = test_data)
acc_linear <- mean(pred_linear == test_data$medv01)
```

```r
print(paste("SVM Linear Kernel Accuracy:", round(acc_linear, 3)))

# SVM with Polynomial Kernel (degree 2)
svm_poly <- svm(medv01 ~ ., data = train_data, kernel = "polynomial", degree = 2)
pred_poly <- predict(svm_poly, newdata = test_data)
acc_poly <- mean(pred_poly == test_data$medv01)
print(paste("SVM Polynomial Kernel (Degree 2) Accuracy:", round(acc_poly, 3)))

# SVM with Radial Kernel
svm_radial <- svm(medv01 ~ ., data = train_data, kernel = "radial")
pred_radial <- predict(svm_radial, newdata = test_data)
acc_radial <- mean(pred_radial == test_data$medv01)
print(paste("SVM Radial Kernel Accuracy:", round(acc_radial, 3)))

# Logistic Regression
glm_boston <- glm(medv01 ~ ., data = train_data, family = "binomial")
pred_glm_prob <- predict(glm_boston, newdata = test_data, type = "response")
pred_glm <- ifelse(pred_glm_prob > 0.5, 1, 0)
acc_glm <- mean(pred_glm == test_data$medv01)
print(paste("Logistic Regression Accuracy:", round(acc_glm, 3)))
```