M6L3 Homework Assignment

Joshua Conte

November 5, 2017

1 M6L3 Homework Assignment

R studio was configured with the following parameters before beginning the project:

```
# clears the console in RStudio
cat("\014")
```

```
# clears environment
rm(list = ls())

# Load required packages
require(ggplot2)
require(e1071)
require(kernlab)
```

1.1 Load Data.

I opened the Wholesale customers Data Set using read.csv2 and downloaded it directly from the UC Irvine Machine Learning Repository.

To format the data, the data is separated by ',', stringsAsFactors = FALSE so that the strings in a data frame will be treated as plain strings and not as factor variables. I set na strings for missing data. Once the data was loaded I added the column names and changed the data types to numeric and finally removed the text data type.

Below is my R code:

```
# Some csv files are really big and take a while to open.  This command checks to
# see if it is already opened, if it is, it does not open it again.
# I also omitted the first column
if (!exists("dfWCD")) {
dfWCD <-
  read.csv2("Wholesale customers data.csv",
    sep = ",",
    stringsAsFactors = FALSE,
    na.strings=c("","NA")
  )
}

# Download directly from site (unreliable from Ecuador)
# if (!exists("dfWCD")) {
# dfWCD <-
#   read.csv2(
#     url(
#       "https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale customers data.csv"
#     ),
#     sep = ",",
#     stringsAsFactors = FALSE,
#     na.strings=c("","NA")
#   )
# # Add a column so I know which study the data is referring to
# study <- sprintf("study_%s",seq(1:440))
# dfWCD$study<-study
# }
```

```
# change 2 to 24 to numeric
dfWCD[1:8] <- sapply(dfWCD[1:8], as.numeric)

# Print first lines
str(dfWCD)
```

```
## 'data.frame':    440 obs. of  8 variables:
##  $ Channel         : num  2 2 2 1 2 2 2 2 1 2 ...
##  $ Region          : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ Fresh           : num  12669 7057 6353 13265 22615 ...
##  $ Milk            : num  9656 9810 8808 1196 5410 ...
##  $ Grocery         : num  7561 9568 7684 4221 7198 ...
##  $ Frozen          : num  214 1762 2405 6404 3915 ...
##  $ Detergents_Paper: num  2674 3293 3516 507 1777 ...
##  $ Delicassen      : num  1338 1776 7844 1788 5185 ...
```

1.1.1 Understanding the data

The data set refers to clients of a wholesale distributor in Portugal. It includes the annual spending in monetary units (m.u.) on diverse product categories. The data has the following attribute information:

1. FRESH: annual spending (m.u.) on fresh products (Continuous);
2. MILK: annual spending (m.u.) on Fresh products (Continuous);
3. GROCERY: annual spending (m.u.)on grocery products (Continuous);
4. FROZEN: annual spending (m.u.)on frozen products (Continuous)
5. DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous)
6. DELICATESSEN: annual spending (m.u.)on and delicatessen products (Continuous);
7. CHANNEL: customer channel - 1 = Horeca (Hotel/Restaurant/Cafe) or 2 = Retail
8. REGION: Customers Region - 1= Lisnon 2 = Oporto or 3 = Other (Nominal)

2 Support Vector Machines in R

Support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Given a set of data points that belong to either of two classes, an SVM finds the hyperplane that:

Leaves the largest possible fraction of points of the same class on the same side. Maximizes the distance of either class from the hyperplane. *Find the optimal separating hyperplane that minimizes the risk of misclassifying the training samples and unseen test samples.

2.1 Understanding kernels on random data

```
# -----------Generate random Data----------

set.seed(33)
x<-matrix(rnorm(400),ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2

y=c(rep(1,150),rep(2,50))
dat= data.frame(x=x,y=as.factor(y))
```

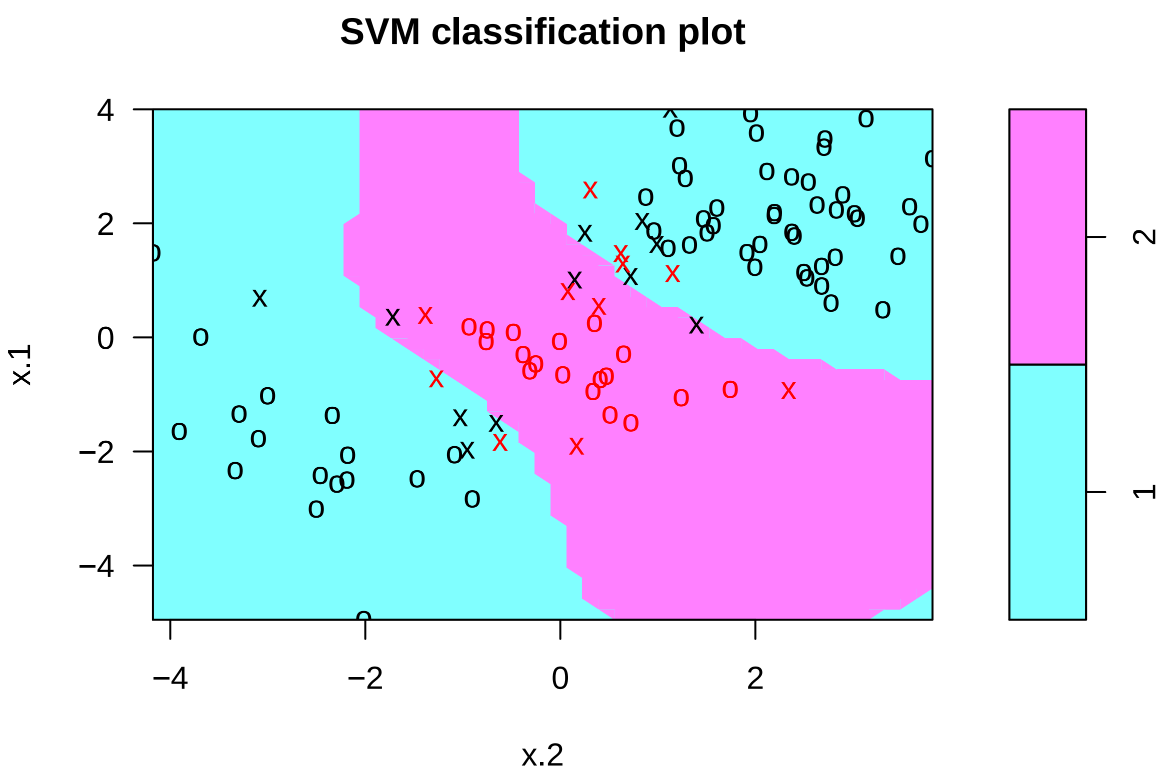
```r
# ----------- Training Model on data --------

train = sample(200,100)
svmfit1<-svm(y~.,data=dat[train,],kernel="radial",gamma=1,cost=100000)


#------------- Cross Validation to set best choice of gamma and cost
# kernel = radial
tune.out1= tune(svm,y~.,data=dat[train,],kernel="radial",ranges=list(cost=c(0.1,10,100,1000)))
summary(tune.out1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.12
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-01  0.19 0.15951315
## 2 1e+01  0.12 0.07888106
## 3 1e+02  0.14 0.10749677
## 4 1e+03  0.17 0.10593499
```

```r
bestmodel1<-tune.out1$best.model
bestmodel1
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = dat[train, ],
##     ranges = list(cost = c(0.1, 10, 100, 1000)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  23
```

```r
plot(bestmodel1,dat[train,])
```




```
# kernel = rbfdot
tune.out2= tune(svm,y~.,data=dat[train,],kernel="polynomial",ranges=list(cost=c(0.1,10,100,1000)),gamma=
summary(tune.out2)
```

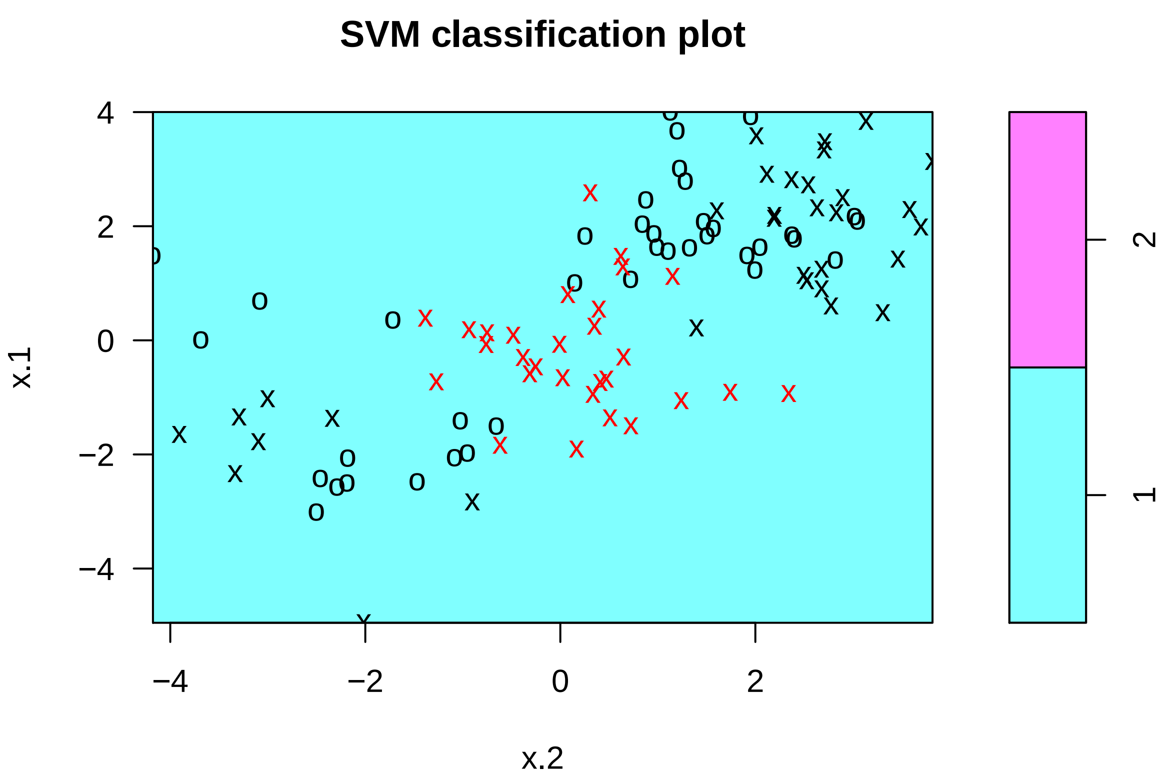
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.29
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-01  0.29  0.1791957
## 2 1e+01  0.30  0.1763834
## 3 1e+02  0.30  0.1763834
## 4 1e+03  0.29  0.1663330
```

```
bestmodel2<-tune.out2$best.model
bestmodel2
```

```
##
## Call:
```

```
## best.tune(method = svm, train.x = y ~ ., data = dat[train, ],
##     ranges = list(cost = c(0.1, 10, 100, 1000)), kernel = "polynomial",
##     gamma = c(0.5, 1, 2, 3, 4))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.1
##      degree:  3
##       gamma:  0.5 1 2 3 4
##      coef.0:  0
##
## Number of Support Vectors:  61
```

```r
plot(bestmodel2,dat[train,])
```




```r
# kernel = linear
tune.out3= tune(svm,y~.,data=dat[train,],kernel="linear",ranges=list(cost=c(0.1,10,100,1000)))
summary(tune.out3)
```

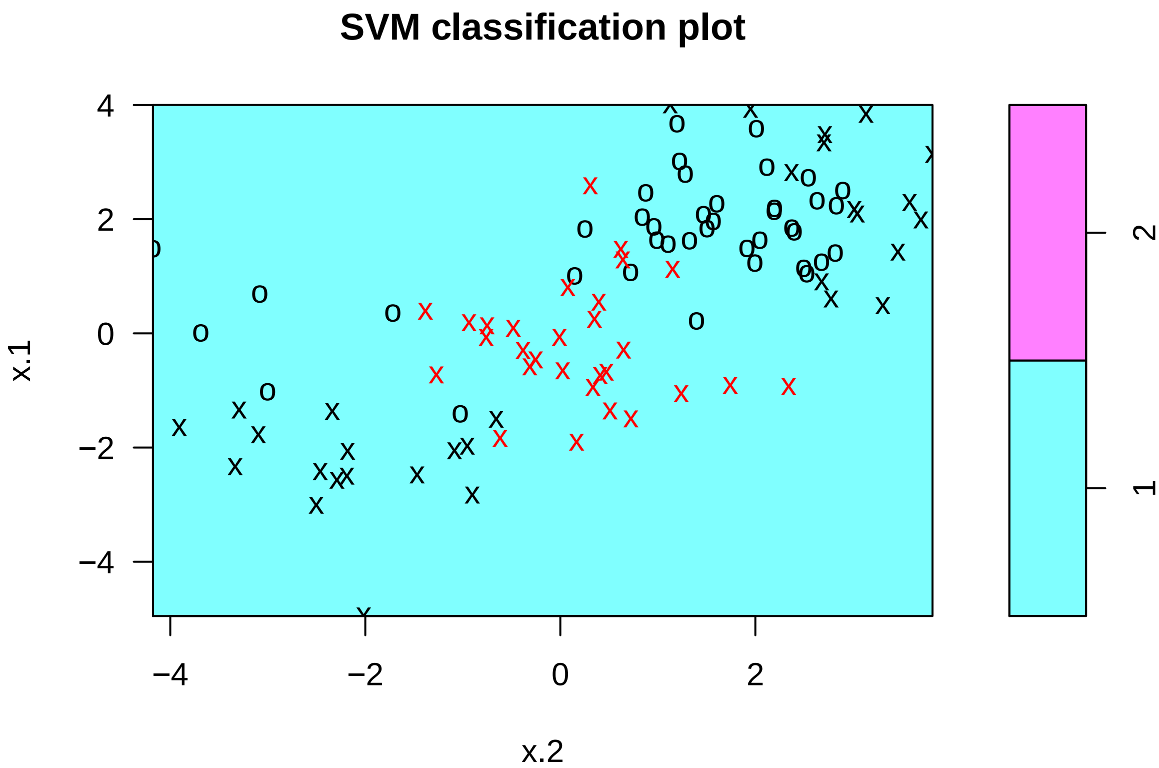
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
```

```
##  cost
##   0.1
##
## - best performance: 0.29
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-01  0.29  0.1449138
## 2 1e+01  0.29  0.1449138
## 3 1e+02  0.29  0.1449138
## 4 1e+03  0.29  0.1449138
```

```
bestmodel3<-tune.out3$best.model
bestmodel3
```

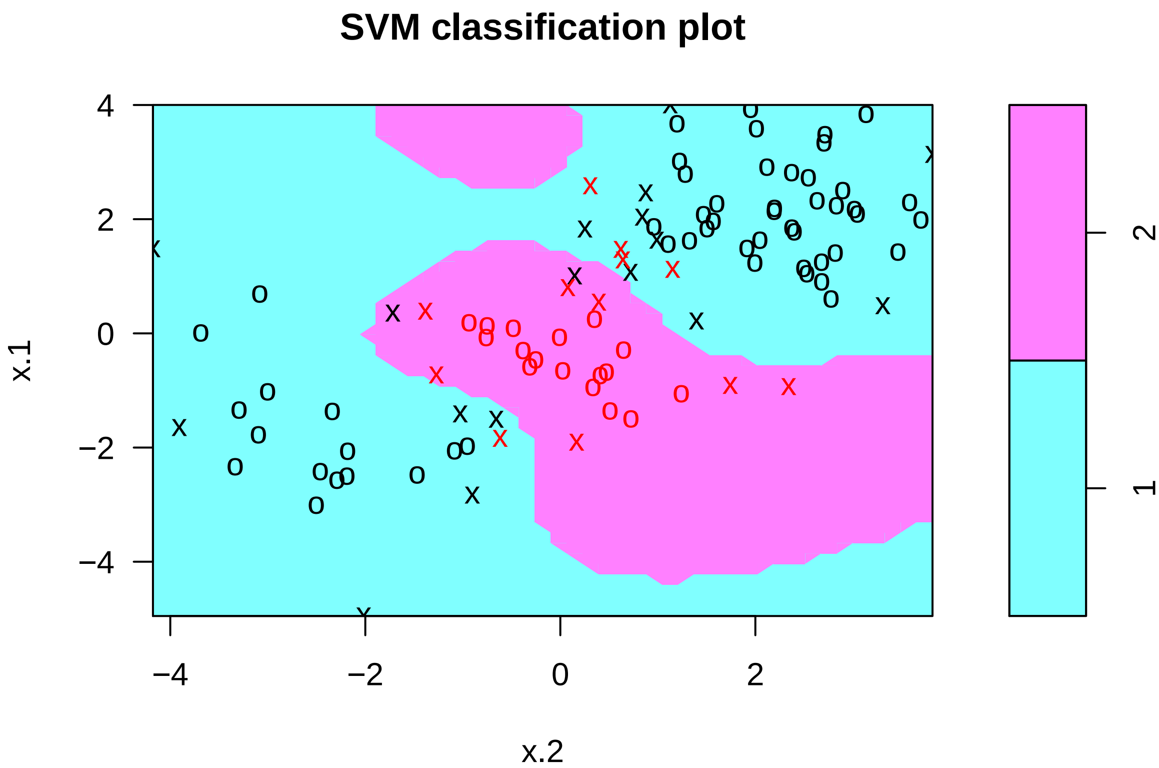
```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = dat[train, ],
##     ranges = list(cost = c(0.1, 10, 100, 1000)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.5
##
## Number of Support Vectors:  60
```

```
plot(bestmodel3,dat[train,])
```

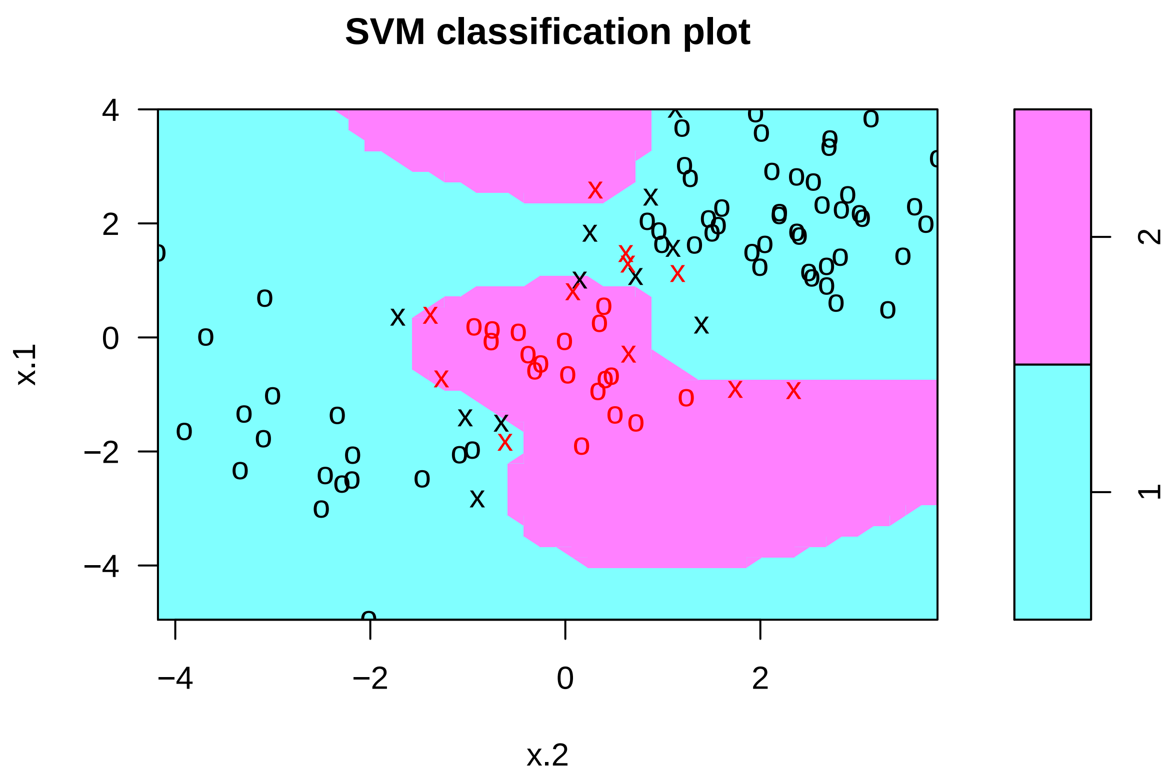


Understanding cost on random data

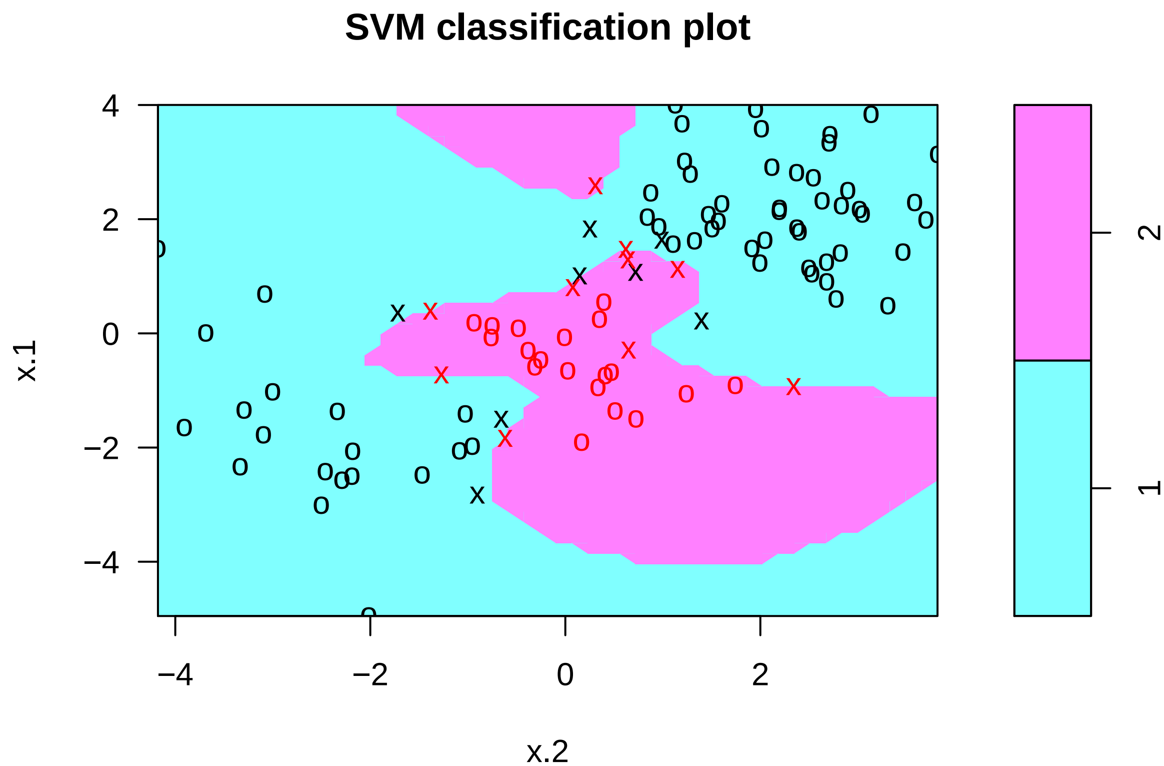
```
# ----------- Training Model on data --------
# cost=10
svmfit1<-svm(y~.,data=dat[train,],kernel="radial",gamma=1,cost=10)
plot(svmfit1,dat[train,])
```



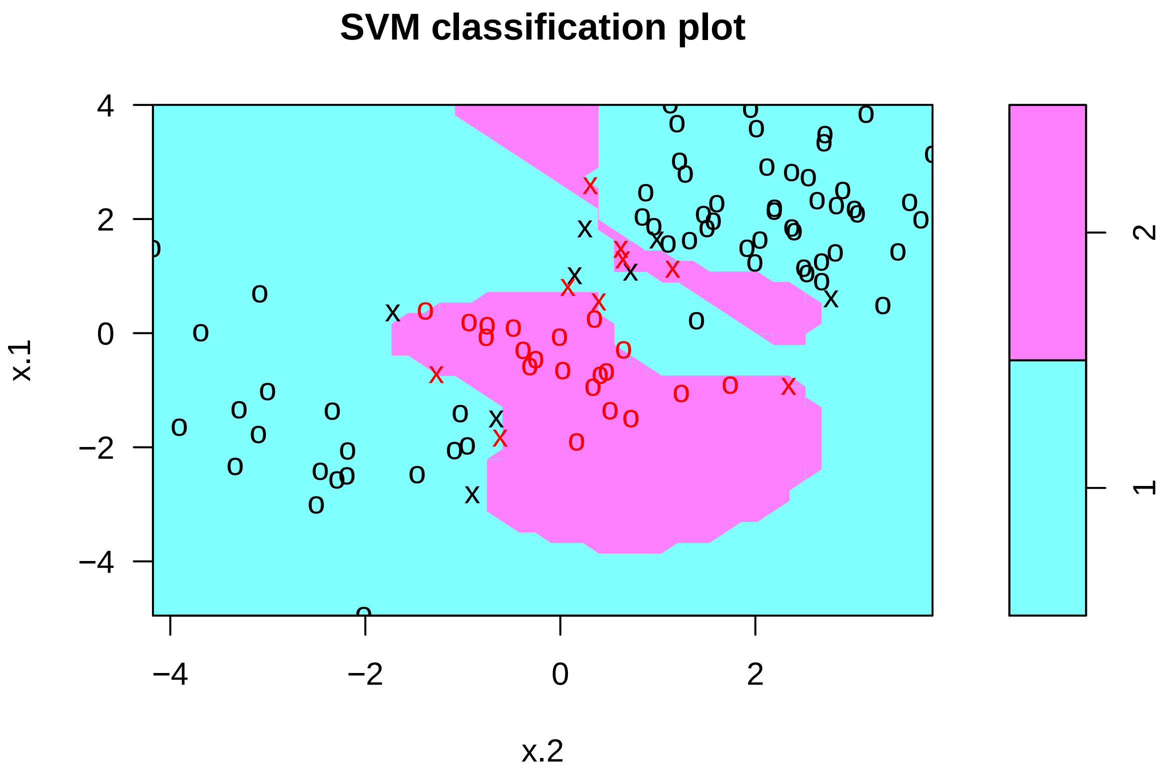

```
# cost=100
svmfit2<-svm(y~.,data=dat[train,],kernel="radial",gamma=1,cost=100)
plot(svmfit2,dat[train,])
```




```r
# cost=1000
svmfit3<-svm(y~.,data=dat[train,],kernel="radial",gamma=1,cost=1000)
plot(svmfit3,dat[train,])
```




```r
# cost=10000
svmfit3<-svm(y~.,data=dat[train,],kernel="radial",gamma=1,cost=10000)
plot(svmfit3,dat[train,])
```



2.2 Usning the dfWCD Dataset

2.2.1 Changing Kernels

This is using radial kernel:

```r
# Explore the data
str(dfWCD)
```

```
## 'data.frame':    440 obs. of  8 variables:
##  $ Channel         : num  2 2 2 1 2 2 2 2 1 2 ...
##  $ Region          : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ Fresh           : num  12669 7057 6353 13265 22615 ...
##  $ Milk            : num  9656 9810 8808 1196 5410 ...
##  $ Grocery         : num  7561 9568 7684 4221 7198 ...
##  $ Frozen          : num  214 1762 2405 6404 3915 ...
##  $ Detergents_Paper: num  2674 3293 3516 507 1777 ...
##  $ Delicassen      : num  1338 1776 7844 1788 5185 ...
```

```r
# create a random sample for training and test data
set.seed(12345)
dfWCD_rand <- dfWCD[order(runif(440)), ]

# normalize
normalize<- function(x) {
  return((x-min(x))/(max(x)-min(x)))
```

```
}
dfWCD_rand.normalized<-as.data.frame(lapply(dfWCD_rand,normalize))


dfWCD_sub<-subset(dfWCD_rand.normalized, select = c("Channel", "Detergents_Paper"))

# ----------- Training Model on data --------

svmfit4<-svm(Channel~.,data=dfWCD_sub,kernel="radial",gamma=1,cost=100000)
svmfit4
```

```
##
## Call:
## svm(formula = Channel ~ ., data = dfWCD_sub, kernel = "radial",
##     gamma = 1, cost = 1e+05)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1e+05
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  102
```

```
plot(svmfit4,dfWCD_sub)

#------------- Cross Validation to set best choice of gamma and cost

tune.out4= tune(svm,Channel~.,data=dfWCD_sub,kernel="radial",ranges=list(cost=c(0.1,10,100,1000)),gamma=
summary(tune.out4)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.07590569
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-01 0.07639292 0.02892524
## 2 1e+01 0.07590569 0.03510598
## 3 1e+02 0.07803692 0.03701741
## 4 1e+03 0.07840297 0.03797046
```

```
bestmodel4<-tune.out4$best.model
bestmodel4
```

```
##
## Call:
## best.tune(method = svm, train.x = Channel ~ ., data = dfWCD_sub,
##     ranges = list(cost = c(0.1, 10, 100, 1000)), kernel = "radial",
##     gamma = c(0.5, 1, 2, 3, 4))
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5 1 2 3 4
##     epsilon:  0.1
##
##
## Number of Support Vectors:  154
```

```r
plot(bestmodel4,dfWCD_sub)
```

This is using a linear kernel:

```r
# ----------- Training Model on data --------
svmfit5<-svm(Channel~.,data=dfWCD_sub,kernel="linear",gamma=1,cost=100000)
svmfit5
```

```
##
## Call:
## svm(formula = Channel ~ ., data = dfWCD_sub, kernel = "linear",
##     gamma = 1, cost = 1e+05)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  1e+05
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  211
```

```r
plot(svmfit5,dfWCD_sub)

#------------- Cross Validation to set best choice of gamma and cost

tune.out5= tune(svm,Channel~.,data=dfWCD_sub,kernel="linear",ranges=list(cost=c(0.1,10,100,1000)),gamma=
summary(tune.out5)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
```

```
##
## - best performance: 0.1549362
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-01 0.1549362 0.07365779
## 2 1e+01 0.1648681 0.09358278
## 3 1e+02 0.1646225 0.09357888
## 4 1e+03 0.1646116 0.09356729
```

```
bestmodel5<-tune.out5$best.model
bestmodel5
```

```
##
## Call:
## best.tune(method = svm, train.x = Channel ~ ., data = dfWCD_sub,
##     ranges = list(cost = c(0.1, 10, 100, 1000)), kernel = "linear",
##     gamma = c(0.5, 1, 2, 3, 4))
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.5 1 2 3 4
##     epsilon:  0.1
##
##
## Number of Support Vectors:  235
```

```
plot(bestmodel5,dfWCD_sub)
```

This is using a sigmoid kernel:

```
# ----------- Training Model on data --------


svmfit6<-svm(Channel~.,data=dfWCD_sub,kernel = "sigmoid",gamma=1,cost=100000)
svmfit6
```

```
##
## Call:
## svm(formula = Channel ~ ., data = dfWCD_sub, kernel = "sigmoid",
##     gamma = 1, cost = 1e+05)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  sigmoid
##        cost:  1e+05
##       gamma:  1
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  440
```

```
plot(svmfit6,dfWCD_sub)

#------------- Cross Validation to set best choice of gamma and cost

tune.out6= tune(svm,Channel~.,data=dfWCD_sub,kernel="sigmoid",ranges=list(cost=c(0.1,10,100,1000)),gamma
summary(tune.out6)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.2662274
##
## - Detailed performance results:
##    cost        error   dispersion
## 1 1e-01 2.662274e-01 7.695164e-02
## 2 1e+01 2.353865e+03 2.347325e+03
## 3 1e+02 2.350820e+05 2.325832e+05
## 4 1e+03 2.352996e+07 2.327727e+07
```

```
bestmodel6<-tune.out6$best.model
bestmodel6
```

```
##
## Call:
## best.tune(method = svm, train.x = Channel ~ ., data = dfWCD_sub,
##     ranges = list(cost = c(0.1, 10, 100, 1000)), kernel = "sigmoid",
##     gamma = c(0.5, 1, 2, 3, 4))
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  sigmoid
##        cost:  0.1
##       gamma:  0.5 1 2 3 4
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  227
```

```
plot(bestmodel6,dfWCD_sub)
```

Understanding cost

```
# ----------- Training Model on data --------
# cost=10
svmfit1<-svm(Channel~.,data=dfWCD_sub,kernel="radial",gamma=1,cost=10)
plot(svmfit1,dat[train,])

# cost=100
```

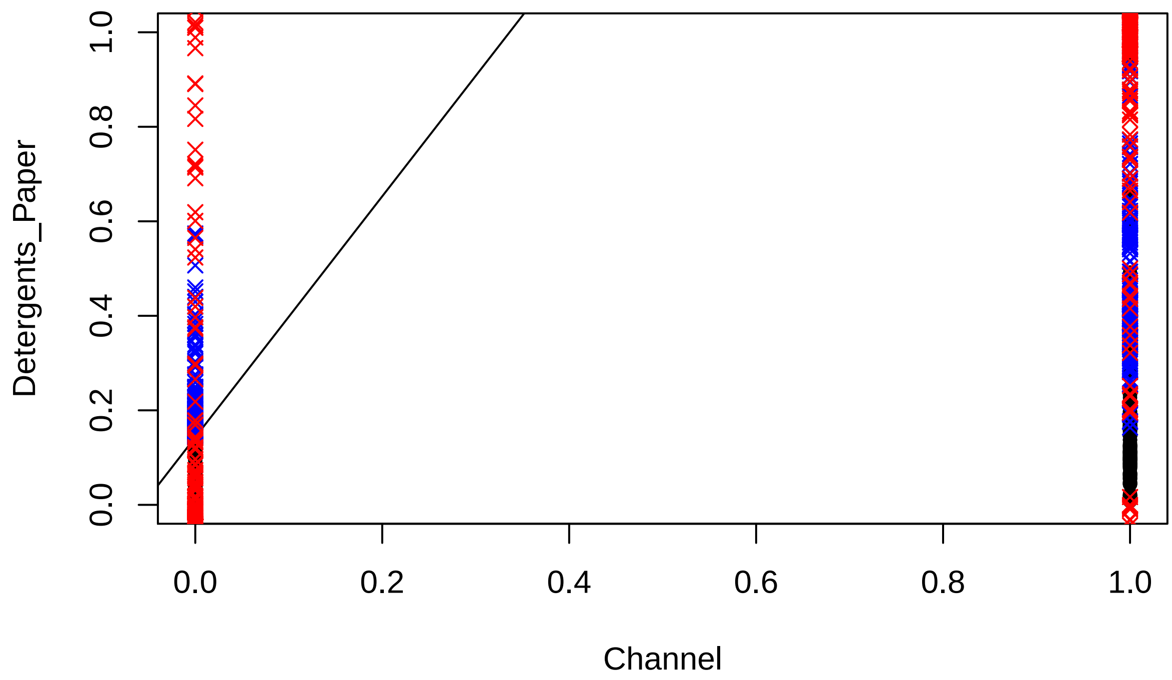
```
svmfit2<-svm(Channel~.,data=dfWCD_sub,kernel="radial",gamma=1,cost=100)
plot(svmfit2,dat[train,])

# cost=1000
svmfit3<-svm(Channel~.,data=dfWCD_sub,kernel="radial",gamma=1,cost=1000)
plot(svmfit3,dat[train,])

# cost=10000
svmfit4<-svm(Channel~.,data=dfWCD_sub,kernel="radial",gamma=1,cost=10000)
plot(svmfit4,dat[train,])
```

Prediction Error Comparison between Linear Regression and SVM:

```
 #  Prediction Error Comparison between Linear Regression and SVM  #


# Plot the data
plot(dfWCD_sub, pch=16)

# Create a linear regression model
lr_model <- lm(Channel~., dfWCD_sub)

# Add the fitted line
abline(lr_model)

# make a prediction for each X
predictedY <- predict(lr_model, dfWCD_sub)

# display the predictions
points(dfWCD_sub$Channel, predictedY, col = "blue", pch=4)

# Function to find the Error
rmse <- function(error)
{
  sqrt(mean(error^2))
}
error <- lr_model$residuals         # same as data$Region - predictedY
predictionRMSE <- rmse(error)
predictionRMSE
```

```
## [1] 0.3607694
```

```
# Support Vector Machine(Finding root mean square error)

# Create Support Vector Model
svm_model <- svm(Channel~. , dfWCD_sub)

# make a prediction for each X
predictedY1 <- predict(svm_model, dfWCD_sub)

# display the predictions
points(dfWCD_sub$Channel, predictedY1, col = "red", pch=4)
```




```
# Function to find the Error
error <- dfWCD_sub$Channel - predictedY1
svrPredictionRMSE <- rmse(error)

tuneResult <- tune(svm, Channel~.,  data = dfWCD_sub,ranges = list(epsilon = seq(0,1,0.1), cost = 2^(2:9
print(tuneResult)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##        0    4
##
## - best performance: 0.07515077
```

```
svrPredictionRMSE
```

```
## [1] 0.2749311
```

2.3 Questions

1. How well does the classifier perform?
   - Its hard to tell, I cannot get my data to plot very well. The final plot for the predictions is very hard to read.

2. Try different kernels. How do they effect its performce?
    - On the random data, the best kernel looks like the radial, it is the only one that seperated 1 and 2, polynomial and lenear only had it as all blue.
3. What might improve its performce?
    - I am not sure, I could not get my data to graph. After reading some info on the internet, I tried normalizing the data (which did not work), because the data should be scaled. If the data is properly scaled then grid-search the hyper-parameter "C" and if you use a non-linear kernel the kernel-specific parameters too.