Homework 7

Charles Hwang

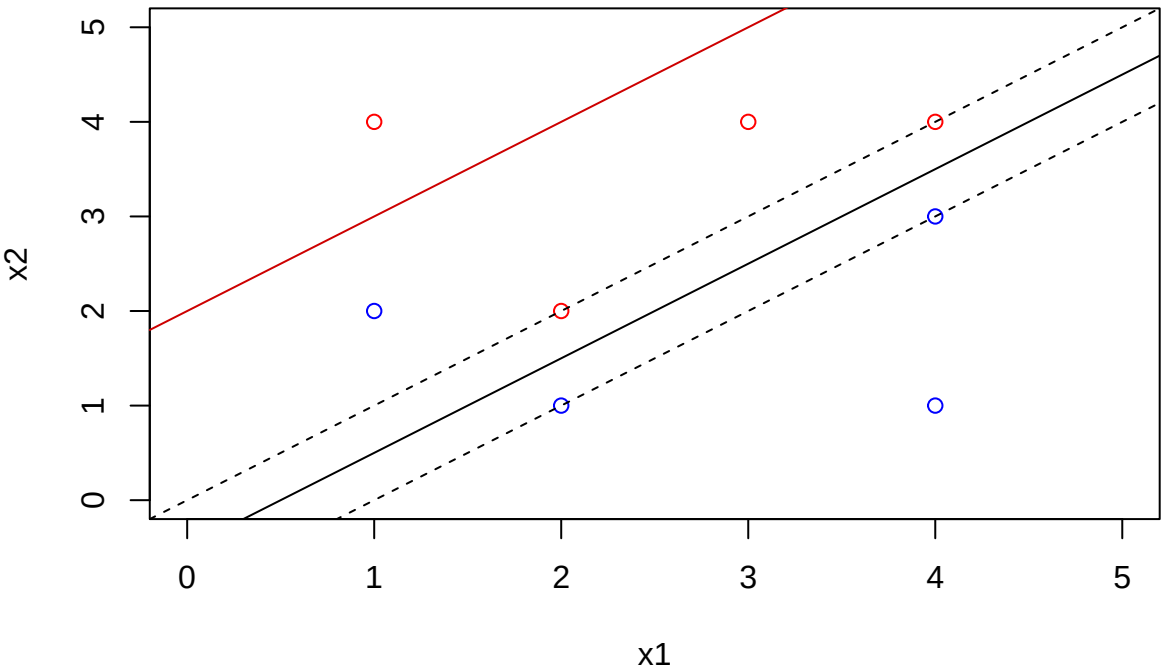
Professor Matthews

STAT 388-001

3 December 2019

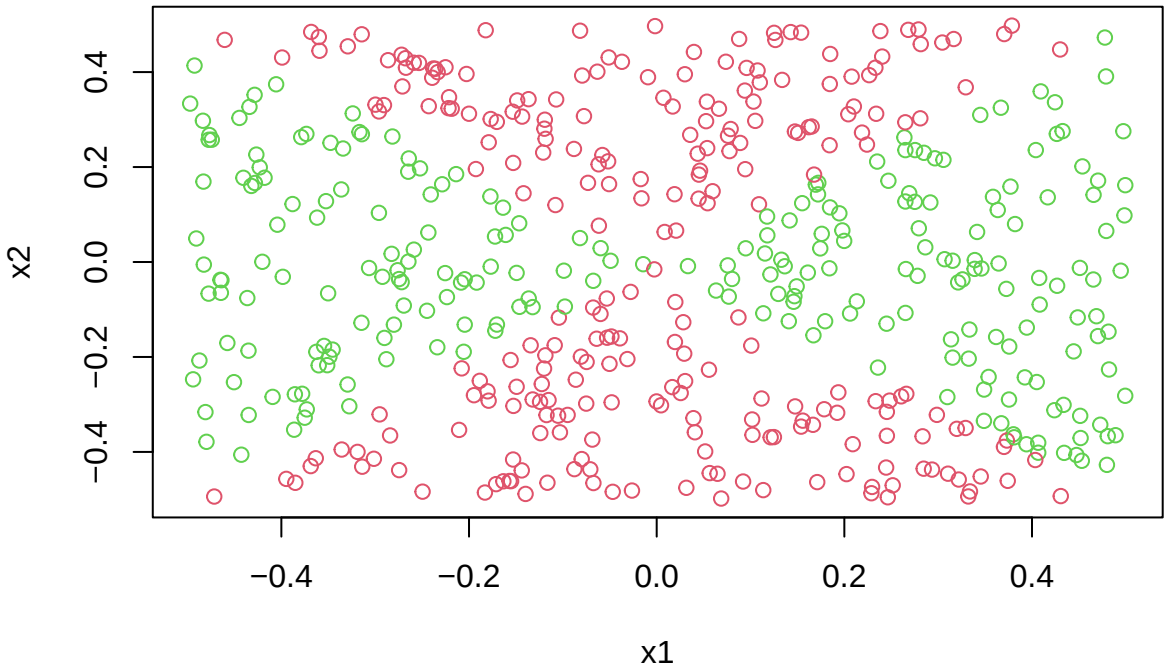
Problem 3

```
rm(list=ls())                                                    # Problem 3a
library(e1071)
x1 <- c(3,2,4,1,2,4,4)
x2 <- c(4,2,4,4,1,3,1)
y <- c("red","red","red","red","blue","blue","blue")
plot(x1,x2,col=y,xlim=c(0,5),ylim=c(0,5))
abline(-.5,1) # -0.5 + x1 - x2 = 0                               # Problem 3b
# Classify to Blue if -0.5+x1-x2>0 and classify to Red otherwise. # Problem 3c
abline(-1,1,lty=2)                                               # Problem 3d
abline(0,1,lty=2)
# The support vectors are observations 2, 3, 5, and 6.           # Problem 3e
# A slight movement of observation 7 would not affect the maximal margin hyperplane because it is not a
abline(2,1,col="red3")                                           # Problem 3g
points(1,2,col="blue")                                           # Problem 3h
```



Problem 5

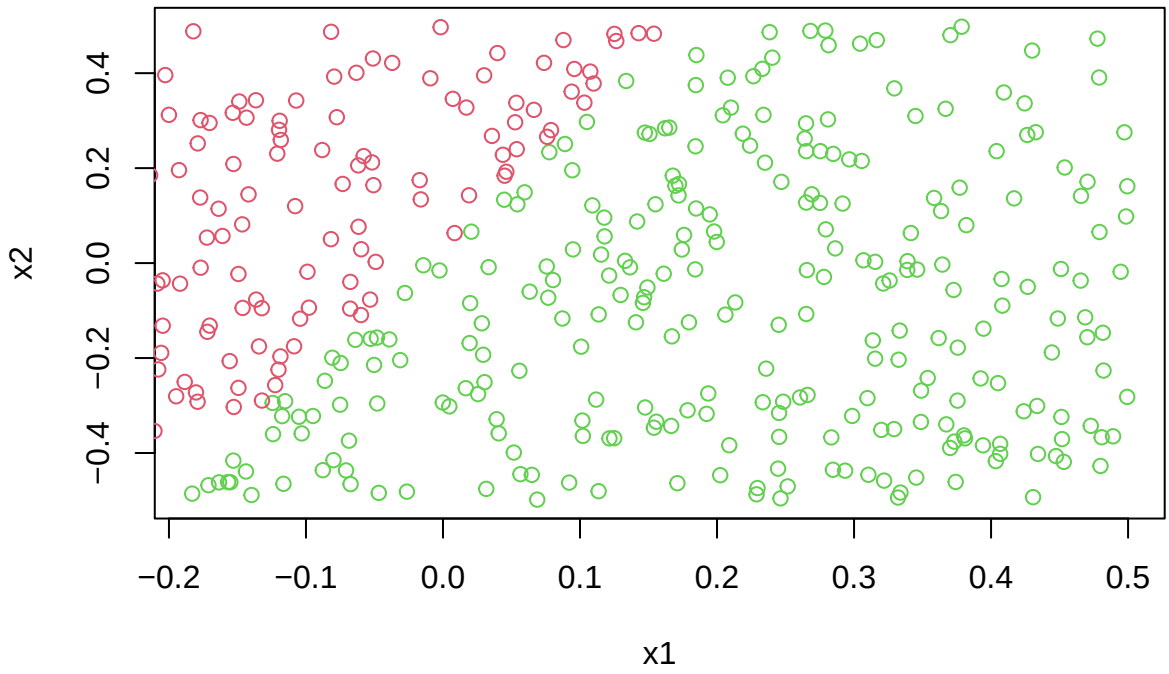
```
rm(list=ls())
set.seed(312)
x1 <- runif(500)-0.5                                    # Problem 5a
x2 <- runif(500)-0.5
y <- 1*(x1^2-x2^2>0)
plot(x1,x2,col=2+y)                                     # Problem 5b
```




```
summary(glm(y~x1+x2,family="binomial"))                 # Problem 5c
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = "binomial")
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.218  -1.146  -1.095   1.196   1.288
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.07592    0.08970  -0.846    0.397
## x1           0.27585    0.32556   0.847    0.397
## x2          -0.10499    0.31044  -0.338    0.735
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 692.50  on 499  degrees of freedom
## Residual deviance: 691.61  on 497  degrees of freedom
## AIC: 697.61
##
## Number of Fisher Scoring iterations: 3
```

```r
data <- data.frame(x1=x1,x2=x2,y=as.factor(y))                    # Problem 5d
plog <- rep(0,500)
plog[predict(glm(y~x1+x2,family="binomial"),data,type="response")>0.48] <- 1
plot(data[plog==1,]$x1,data[plog==1,]$x2,col=2+1,xlab="x1",ylab="x2")
points(data[plog==0,]$x1,data[plog==0,]$x2,col=2+0)
```




```r
summary(glm(y~x1+I(x1^2)+x2+I(x2^2)+I(x1*x2),family="binomial")) # Problem 5e
```

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

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

##
## Call:
## glm(formula = y ~ x1 + I(x1^2) + x2 + I(x2^2) + I(x1 * x2), family = "binomial")
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.003237   0.000000   0.000000   0.000000   0.003313
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)      6.347   1208.329   0.005    0.996
## x1             179.789  56604.327   0.003    0.997
## I(x1^2)      53588.055 826601.228   0.065    0.948
## x2             378.565  68549.350   0.006    0.996
## I(x2^2)     -52357.098 759231.079  -0.069    0.945
## I(x1 * x2)    1629.312  86574.152   0.019    0.985
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.9250e+02  on 499  degrees of freedom
## Residual deviance: 2.3519e-05  on 494  degrees of freedom
## AIC: 12
```

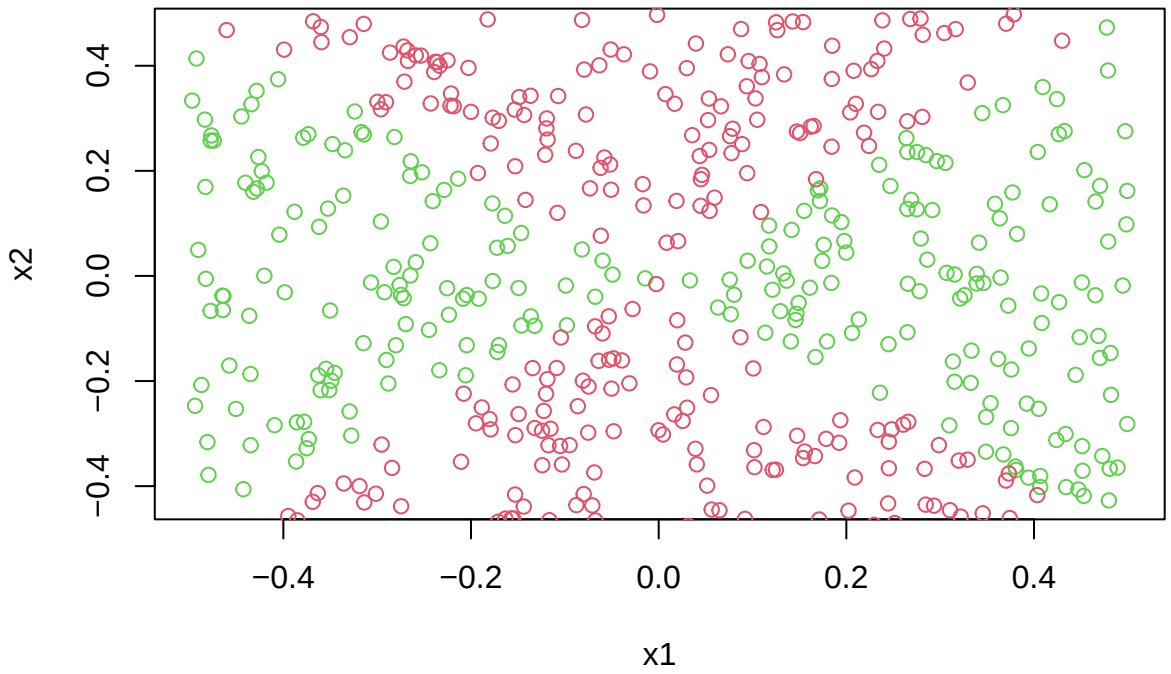
```
##
## Number of Fisher Scoring iterations: 25
```

```
pnl <- rep(0,500)                                                    # Problem 5f
pnl[predict(glm(y~x1+I(x1^2)+x2+I(x2^2)+I(x1*x2),family="binomial"),data,type="response")>0.45] <- 1
```

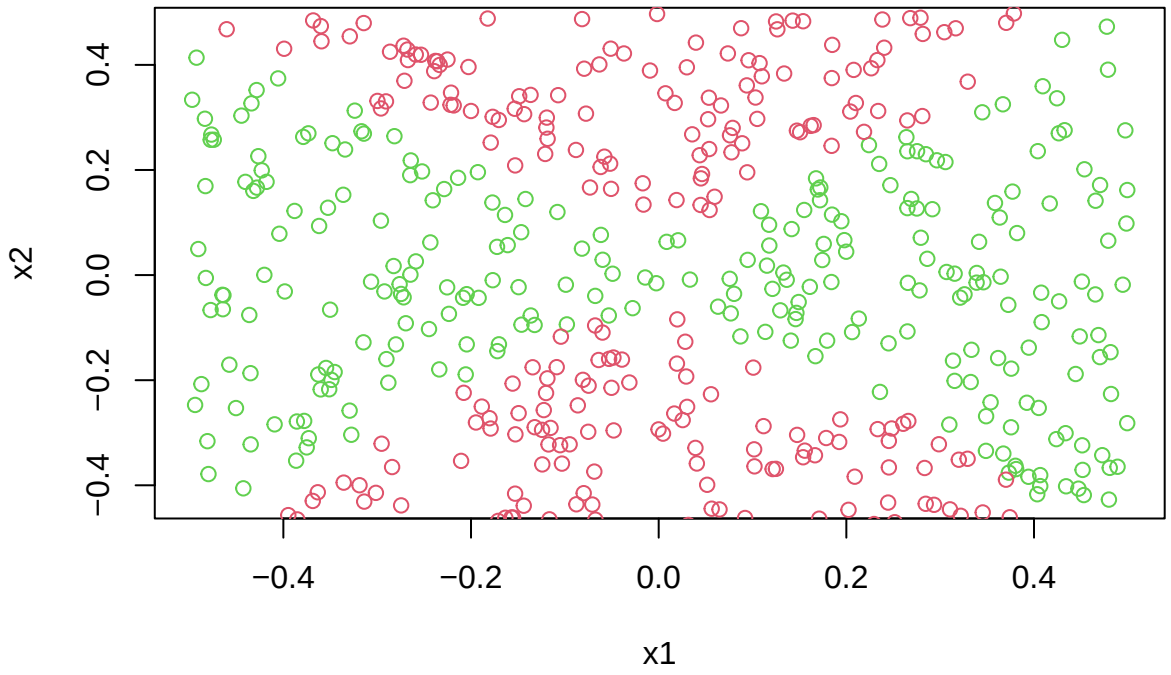
```
## Warning: glm.fit: algorithm did not converge
```

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

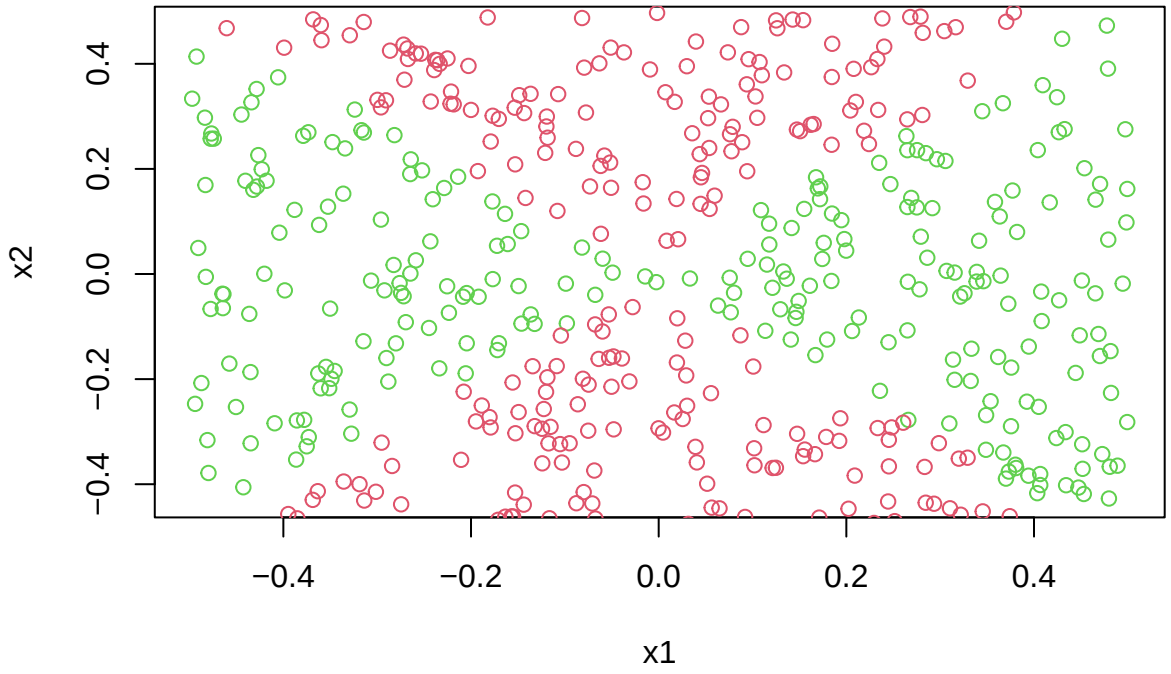
```
plot(data[pnl==1,]$x1,data[pnl==1,]$x2,col=2+1,xlab="x1",ylab="x2")
points(data[pnl==0,]$x1,data[pnl==0,]$x2,col=2+0)
```




```
psvc <- predict(svm(y~x1+x2,data),data)                              # Problem 5g
plot(data[psvc==1,]$x1,data[psvc==1,]$x2,col=2+1,xlab="x1",ylab="x2")
points(data[psvc==0,]$x1,data[psvc==0,]$x2,col=2+0)
```




```
psvm <- predict(svm(y~x1+x2,data,kernel="radial",gamma=.1),data) # Problem 5h
plot(data[psvm==1,]$x1,data[psvm==1,]$x2,col=2+1,xlab="x1",ylab="x2")
points(data[psvm==0,]$x1,data[psvm==0,]$x2,col=2+0)
```




```
# The models that included the non-linear function (5f), support vector classifier (5g), non-linear ker
```

Problem 8

```
rm(list=ls())
set.seed(312)
library(ISLR)
```

```r
s <- sample(nrow(OJ), 800)                                                    # P
train <- OJ[s,]
test <- OJ[-s,]
summary(svm(Purchase~.,train,kernel="linear",cost=.01))                       # P
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "linear", cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  433
##
##  ( 217 216 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
# There are 441 support vectors; 220 are classified to CH and 221 are classified to MM.
c(sum(table(train$Purchase,predict(svm(Purchase~.,train,kernel="linear",cost=.01),train))[2:3])/nrow(tra
```

```
## [1] 0.1700000 0.1555556
```

```r
summary(tune(svm,Purchase~.,data=train,kernel="linear",ranges=list(cost=c(seq(.01,.1,.01),.15,.2,seq(.25
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.03
##
## - best performance: 0.175
##
## - Detailed performance results:
##     cost   error dispersion
## 1   0.01 0.18250 0.04647281
## 2   0.02 0.18375 0.04168749
## 3   0.03 0.17500 0.04166667
## 4   0.04 0.17875 0.04251225
## 5   0.05 0.17750 0.03987829
## 6   0.06 0.18000 0.04133199
## 7   0.07 0.17875 0.04210189
## 8   0.08 0.18000 0.04133199
## 9   0.09 0.17875 0.04372023
## 10  0.10 0.18000 0.04133199
```

```
## 11  0.15 0.17750 0.04322101
## 12  0.20 0.17750 0.04322101
## 13  0.25 0.17750 0.04322101
## 14  0.50 0.17750 0.04401704
## 15  0.75 0.17750 0.04518481
## 16  1.00 0.17750 0.04518481
## 17  1.25 0.17625 0.04543387
## 18  1.50 0.17750 0.04556741
## 19  1.75 0.17625 0.04543387
## 20  2.00 0.17625 0.04543387
## 21  2.25 0.17625 0.04543387
## 22  2.50 0.17625 0.04543387
## 23  2.75 0.17625 0.04543387
## 24  3.00 0.17625 0.04543387
## 25  3.25 0.17625 0.04543387
## 26  3.50 0.17625 0.04543387
## 27  3.75 0.17750 0.04556741
## 28  4.00 0.17750 0.04556741
## 29  4.25 0.17750 0.04556741
## 30  4.50 0.17750 0.04556741
## 31  4.75 0.17750 0.04556741
## 32  5.00 0.17750 0.04556741
## 33  5.25 0.17750 0.04556741
## 34  5.50 0.17875 0.04788949
## 35  5.75 0.18000 0.04794383
## 36  6.00 0.18000 0.04794383
## 37  6.25 0.17875 0.04788949
## 38  6.50 0.18000 0.04794383
## 39  6.75 0.18000 0.04794383
## 40  7.00 0.18000 0.04794383
## 41  7.25 0.18000 0.04794383
## 42  7.50 0.18000 0.04794383
## 43  7.75 0.18000 0.04794383
## 44  8.00 0.18000 0.04794383
## 45  8.25 0.17875 0.04641674
## 46  8.50 0.17875 0.04641674
## 47  8.75 0.17875 0.04641674
## 48  9.00 0.17750 0.04518481
## 49  9.25 0.17750 0.04518481
## 50  9.50 0.17875 0.04372023
## 51  9.75 0.17750 0.04518481
## 52 10.00 0.17750 0.04518481
```

```
c(sum(table(train$Purchase,predict(svm(Purchase~.,train,kernel="linear",cost=7.25),train))[2:3])/nrow(t
```

```
## [1] 0.1662500 0.1333333
```

```
summary(svm(Purchase~.,train,kernel="radial",cost=.01))                    # 
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "radial", cost = 0.01)
##
##
## Parameters:
```

```
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  620
##
##  ( 308 312 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
# There are 640 support vectors; 319 are classified to CH and 321 are classified to MM.
c(sum(table(train$Purchase,predict(svm(Purchase~.,train,kernel="radial",cost=.01),train))[2:3])/nrow(tra
```

```
## [1] 0.3850000 0.4037037
```

```
summary(tune(svm,Purchase~.,data=train,kernel="radial",ranges=list(cost=c(seq(.01,.1,.01),.15,.2,seq(.25
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1725
##
## - Detailed performance results:
##     cost   error dispersion
## 1   0.01 0.38500 0.04632314
## 2   0.02 0.38500 0.04632314
## 3   0.03 0.36625 0.04931827
## 4   0.04 0.24750 0.05426274
## 5   0.05 0.20625 0.03547789
## 6   0.06 0.19625 0.04126894
## 7   0.07 0.18875 0.03793727
## 8   0.08 0.18500 0.03855011
## 9   0.09 0.18375 0.04084609
## 10  0.10 0.18250 0.03917553
## 11  0.15 0.18750 0.04289846
## 12  0.20 0.18375 0.03634805
## 13  0.25 0.18125 0.03498512
## 14  0.50 0.17750 0.03809710
## 15  0.75 0.17500 0.03679900
## 16  1.00 0.17250 0.04401704
## 17  1.25 0.17500 0.04289846
## 18  1.50 0.17875 0.04411554
## 19  1.75 0.18000 0.04090979
## 20  2.00 0.18125 0.04340139
## 21  2.25 0.18000 0.04338138
```

```
## 22  2.50 0.18250 0.04721405
## 23  2.75 0.18250 0.04721405
## 24  3.00 0.18375 0.04528076
## 25  3.25 0.18750 0.04526159
## 26  3.50 0.19000 0.04362084
## 27  3.75 0.19000 0.04362084
## 28  4.00 0.18875 0.04101575
## 29  4.25 0.19000 0.03944053
## 30  4.50 0.19125 0.03866254
## 31  4.75 0.19125 0.03866254
## 32  5.00 0.19125 0.03866254
## 33  5.25 0.19250 0.03827895
## 34  5.50 0.19250 0.03827895
## 35  5.75 0.19125 0.03586723
## 36  6.00 0.19000 0.03717451
## 37  6.25 0.19000 0.03944053
## 38  6.50 0.19000 0.03944053
## 39  6.75 0.19000 0.03944053
## 40  7.00 0.19000 0.03944053
## 41  7.25 0.19000 0.03944053
## 42  7.50 0.19000 0.03944053
## 43  7.75 0.19125 0.04126894
## 44  8.00 0.19000 0.04199868
## 45  8.25 0.19125 0.04126894
## 46  8.50 0.19250 0.04005205
## 47  8.75 0.19250 0.04005205
## 48  9.00 0.19250 0.04005205
## 49  9.25 0.19250 0.04005205
## 50  9.50 0.19250 0.04005205
## 51  9.75 0.19000 0.04158325
## 52 10.00 0.19125 0.04168749
```

```
c(sum(table(train$Purchase,predict(svm(Purchase~.,train,kernel="radial",cost=6.25),train))[2:3])/nrow(t
```

```
## [1] 0.1512500 0.1148148
```

```
summary(svm(Purchase~.,train,kernel="polynomial",degree=2,cost=.01))                    #
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "polynomial",
##     degree = 2, cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  619
##
##  ( 308 311 )
##
```

```
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
# There are 642 support vectors; 319 are classified to CH and 323 are classified to MM.
c(sum(table(train$Purchase,predict(svm(Purchase~.,train,kernel="polynomial",degree=2,cost=.01),train))[
```

```
## [1] 0.3625000 0.4037037
```

```
summary(tune(svm,Purchase~.,data=train,kernel="polynomial",degree=2,ranges=list(cost=c(seq(.01,.1,.01),
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   9.5
##
## - best performance: 0.19125
##
## - Detailed performance results:
##     cost   error dispersion
## 1   0.01 0.37750 0.05552777
## 2   0.02 0.35625 0.04419417
## 3   0.03 0.35125 0.04308019
## 4   0.04 0.35000 0.04330127
## 5   0.05 0.33375 0.04566256
## 6   0.06 0.32250 0.03987829
## 7   0.07 0.31375 0.04910660
## 8   0.08 0.31125 0.05084358
## 9   0.09 0.30625 0.04903584
## 10  0.10 0.30000 0.04487637
## 11  0.15 0.24750 0.04816061
## 12  0.20 0.21250 0.04249183
## 13  0.25 0.20250 0.04556741
## 14  0.50 0.20625 0.05376453
## 15  0.75 0.19875 0.05382908
## 16  1.00 0.20000 0.05034602
## 17  1.25 0.19875 0.05285265
## 18  1.50 0.19875 0.05285265
## 19  1.75 0.20000 0.05237419
## 20  2.00 0.20125 0.04803428
## 21  2.25 0.20500 0.04090979
## 22  2.50 0.20250 0.04362084
## 23  2.75 0.20125 0.04348132
## 24  3.00 0.20375 0.04372023
## 25  3.25 0.20125 0.04016027
## 26  3.50 0.20250 0.04281744
## 27  3.75 0.20125 0.03884174
## 28  4.00 0.20000 0.03632416
## 29  4.25 0.19750 0.04158325
```

```
## 30  4.50 0.20000 0.03952847
## 31  4.75 0.19875 0.03928617
## 32  5.00 0.19750 0.03622844
## 33  5.25 0.19750 0.03622844
## 34  5.50 0.19625 0.03682259
## 35  5.75 0.19750 0.03525699
## 36  6.00 0.19750 0.03525699
## 37  6.25 0.19750 0.03525699
## 38  6.50 0.19875 0.03839216
## 39  6.75 0.19875 0.03839216
## 40  7.00 0.19625 0.03821086
## 41  7.25 0.19500 0.03872983
## 42  7.50 0.19250 0.04048319
## 43  7.75 0.19250 0.03917553
## 44  8.00 0.19250 0.03917553
## 45  8.25 0.19250 0.03917553
## 46  8.50 0.19250 0.03917553
## 47  8.75 0.19250 0.03917553
## 48  9.00 0.19250 0.03917553
## 49  9.25 0.19375 0.03875224
## 50  9.50 0.19125 0.04041881
## 51  9.75 0.19125 0.04041881
## 52 10.00 0.19125 0.04041881
```

```
c(sum(table(train$Purchase,predict(svm(Purchase~.,train,kernel="polynomial",degree=2,cost=9.25),train))
```

```
## [1] 0.1525000 0.1074074
```

```
cat("                Train   Test \n     Linear Old",sum(table(train$Purchase,predict(svm(Purchase~.,tra
```

```
##                 Train   Test
##      Linear Old 0.17 0.1555556
##      Linear New 0.16625 0.1333333
##      Radial Old 0.385 0.4037037
##      Radial New 0.15125 0.1148148
##  Polynomial Old 0.3625 0.4037037
##  Polynomial New 0.1525 0.1074074
```