ISLR-HW7

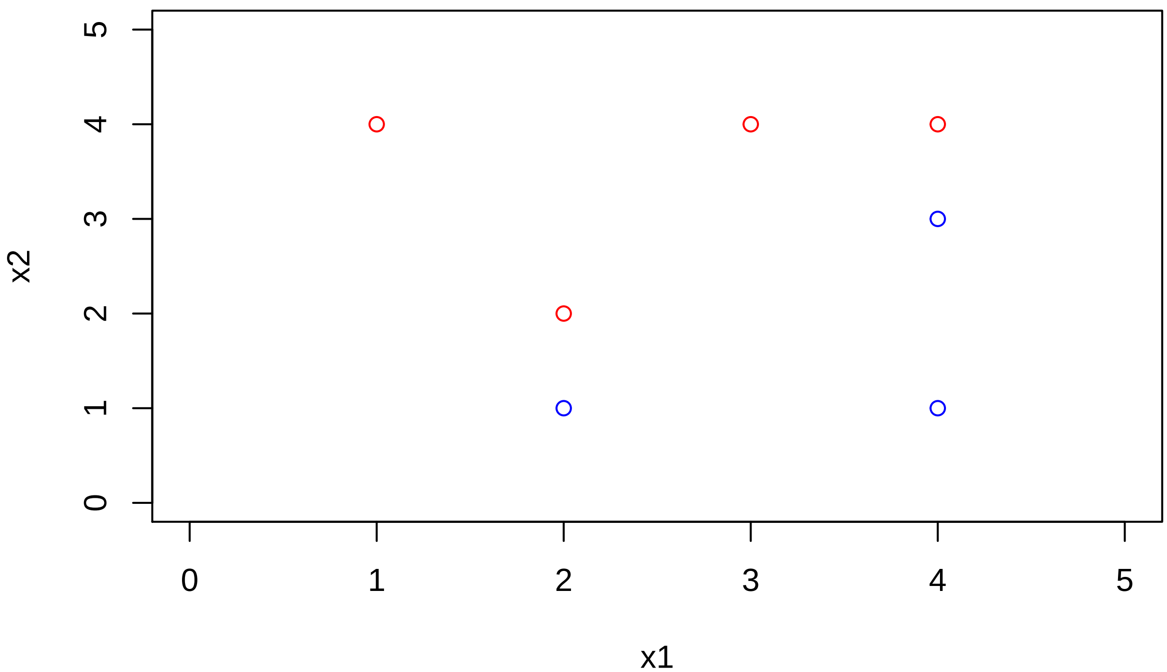
Chaoqun Yin

5/3/2019

Exercise 9.3

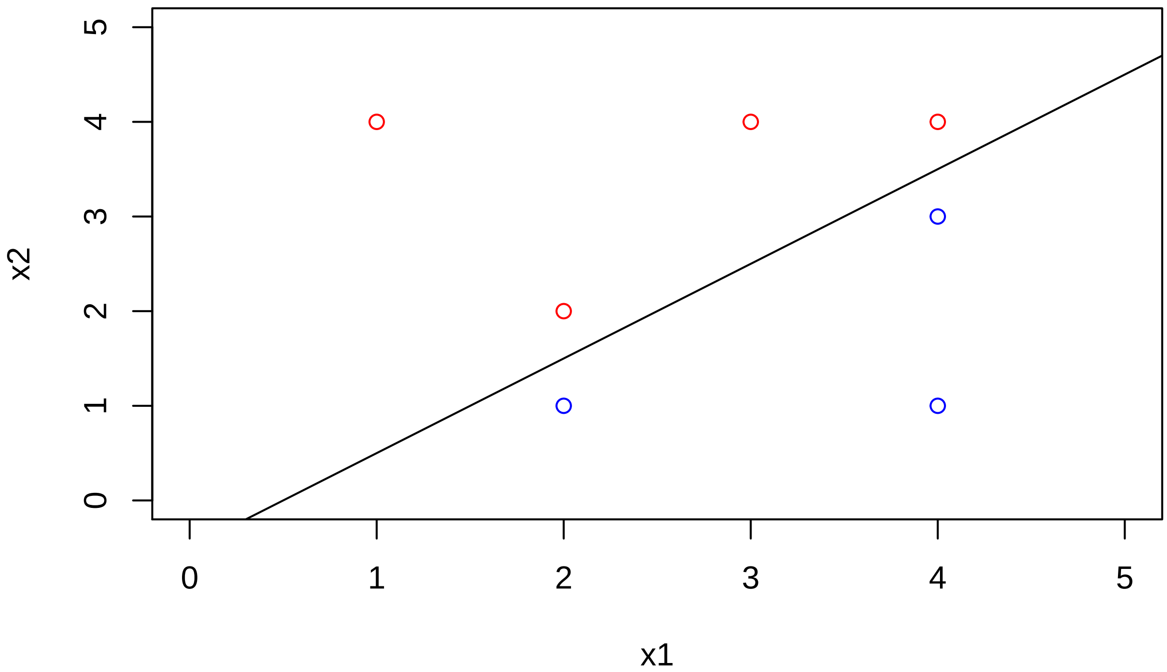
a

```
x1 = c(3, 2, 4, 1, 2, 4, 4)
x2 = c(4, 2, 4, 4, 1, 3, 1)
colors = c("red", "red", "red", "red", "blue", "blue", "blue")
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
```



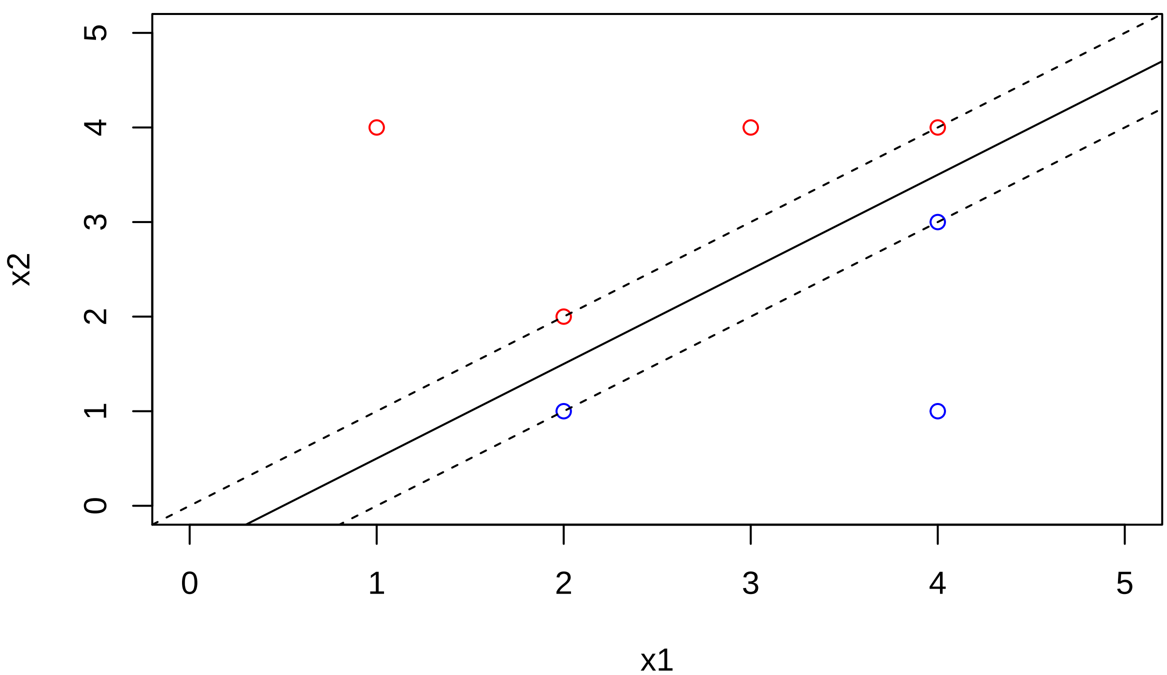
b

```
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
```



d

```
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
abline(-1, 1, lty = 2)
abline(0, 1, lty = 2)
```

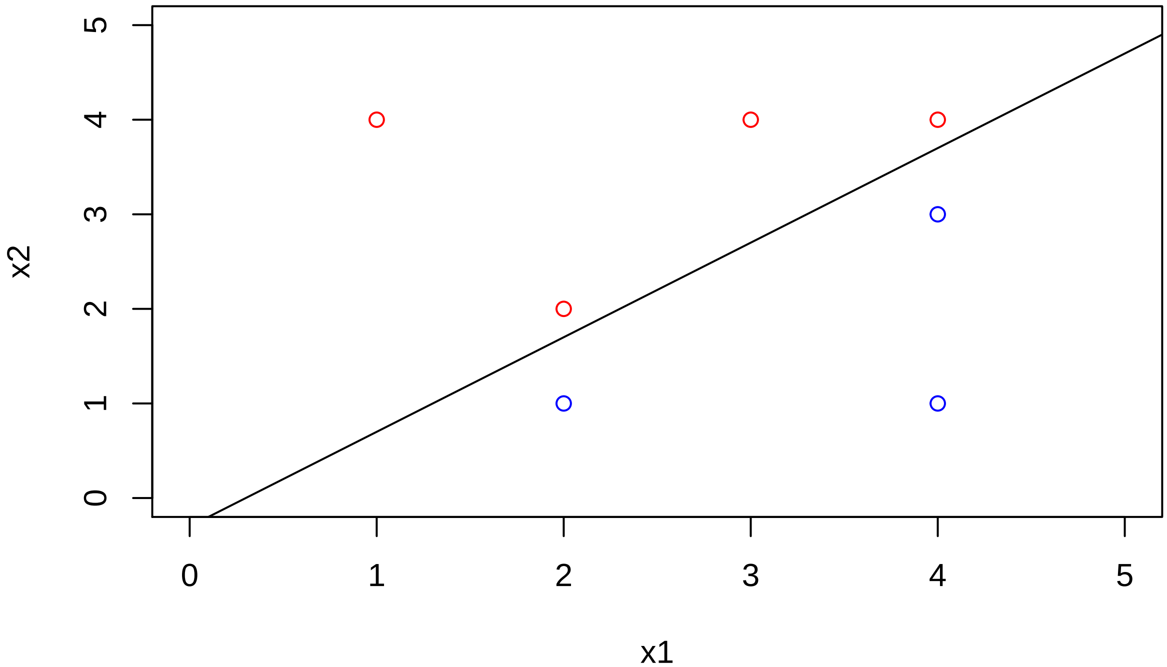


e

The support vectors are the points (2,1), (2,2), (4,3) and (4,4).

g

```
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.3, 1)
```



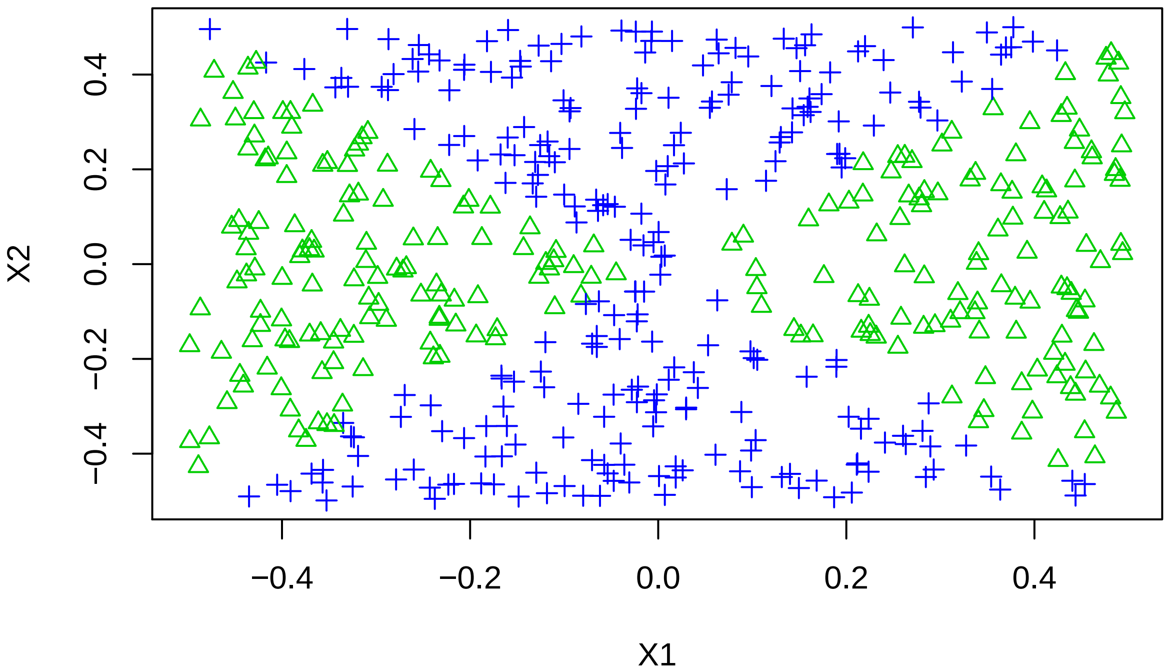
Exercise 9.5

a

```
set.seed(1)
x1 <- runif(500) - 0.5
x2 <- runif(500) - 0.5
y <- 1 * (x1^2 - x2^2 > 0)
```

b

```
plot(x1, x2, xlab = "X1", ylab = "X2", col = (4 - y), pch = (3 - y))
```



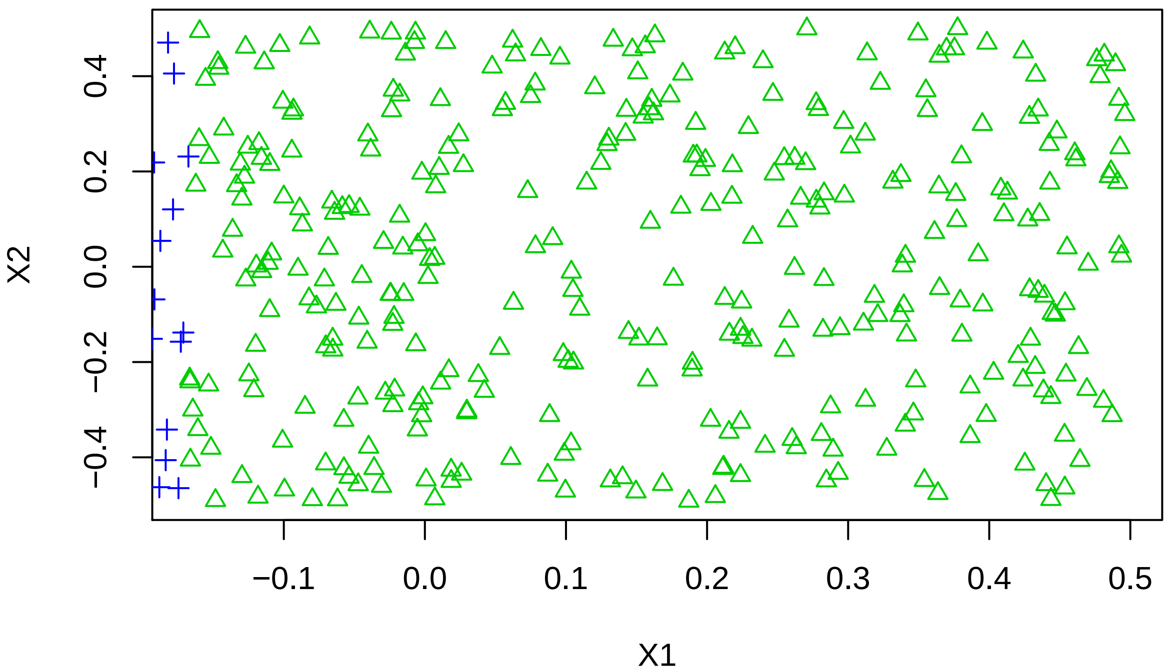
c

```r
logit.fit <- glm(y ~ x1 + x2, family = "binomial")
summary(logit.fit)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = "binomial")
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.179  -1.139  -1.112   1.206   1.257
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.087260   0.089579  -0.974    0.330
## x1           0.196199   0.316864   0.619    0.536
## x2          -0.002854   0.305712  -0.009    0.993
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 692.18  on 499  degrees of freedom
## Residual deviance: 691.79  on 497  degrees of freedom
## AIC: 697.79
##
## Number of Fisher Scoring iterations: 3
```

d

```
data <- data.frame(x1 = x1, x2 = x2, y = y)
probs <- predict(logit.fit, data, type = "response")
preds <- rep(0, 500)
preds[probs > 0.47] <- 1
plot(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1), xlab = "X1", ylab = "X2
points(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0))
```



e

```
logitnl.fit <- glm(y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial")
```

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

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

```
summary(logitnl.fit)
```

```
##
## Call:
## glm(formula = y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial")
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -8.240e-04  -2.000e-08  -2.000e-08   2.000e-08   1.163e-03
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -102.2     4302.0  -0.024    0.981
## poly(x1, 2)1    2715.3   141109.5   0.019    0.985
## poly(x1, 2)2   27218.5   842987.2   0.032    0.974
```

```
## poly(x2, 2)1   -279.7    97160.4  -0.003    0.998
## poly(x2, 2)2 -28693.0   875451.3  -0.033    0.974
## I(x1 * x2)     -206.4    41802.8  -0.005    0.996
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.9218e+02  on 499  degrees of freedom
## Residual deviance: 3.5810e-06  on 494  degrees of freedom
## AIC: 12
##
## Number of Fisher Scoring iterations: 25
```

Exercise 9.7

a

```r
library(ISLR)
var <- ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
Auto$mpglevel <- as.factor(var)
```

b

```r
library(e1071)
set.seed(1)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01275641
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07403846 0.05471525
## 2 1e-01 0.03826923 0.05148114
## 3 1e+00 0.01275641 0.01344780
## 4 5e+00 0.01782051 0.01229997
## 5 1e+01 0.02038462 0.01074682
## 6 1e+02 0.03820513 0.01773427
## 7 1e+03 0.03820513 0.01773427
```

c

```
set.seed(1)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges = list(cost = c(0.01, 0.
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      2
##
## - best performance: 0.3013462
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-02      2 0.5611538 0.04344202
## 2  1e-01      2 0.5611538 0.04344202
## 3  1e+00      2 0.5611538 0.04344202
## 4  5e+00      2 0.5611538 0.04344202
## 5  1e+01      2 0.5382051 0.05829238
## 6  1e+02      2 0.3013462 0.09040277
## 7  1e-02      3 0.5611538 0.04344202
## 8  1e-01      3 0.5611538 0.04344202
## 9  1e+00      3 0.5611538 0.04344202
## 10 5e+00      3 0.5611538 0.04344202
## 11 1e+01      3 0.5611538 0.04344202
## 12 1e+02      3 0.3322436 0.11140578
## 13 1e-02      4 0.5611538 0.04344202
## 14 1e-01      4 0.5611538 0.04344202
## 15 1e+00      4 0.5611538 0.04344202
## 16 5e+00      4 0.5611538 0.04344202
## 17 1e+01      4 0.5611538 0.04344202
## 18 1e+02      4 0.5611538 0.04344202
```

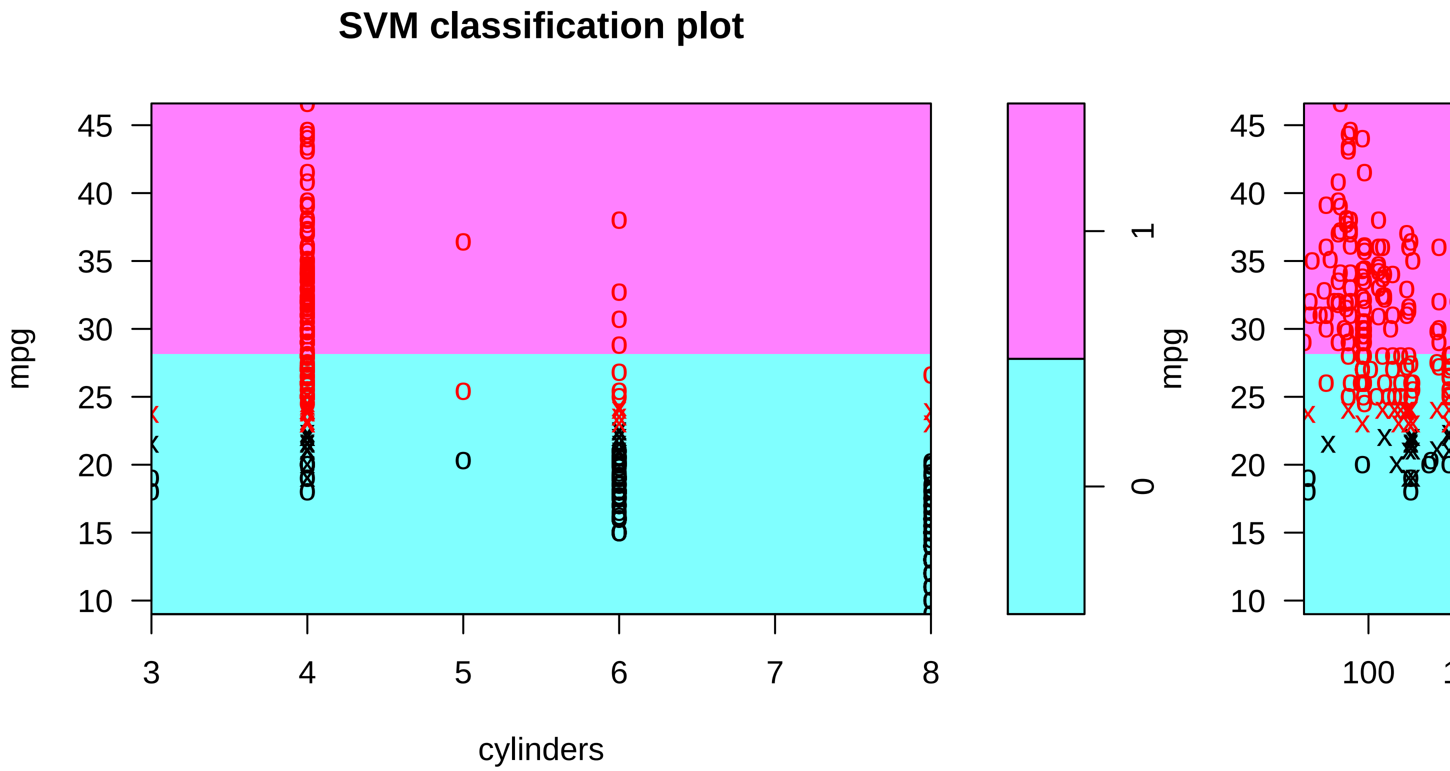
```
set.seed(1)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges = list(cost = c(0.01, 0.1, 1
summary(tune.out)
```

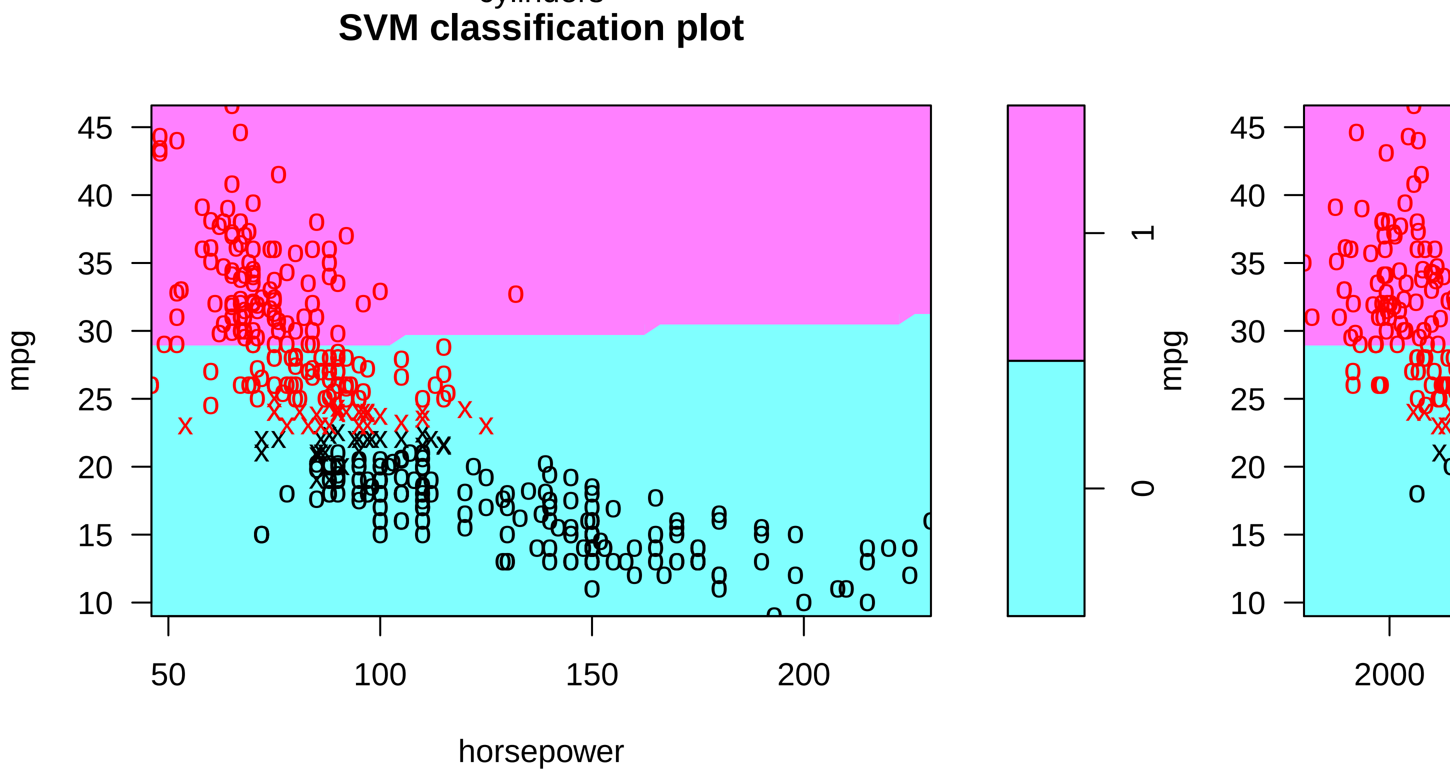
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100  0.01
##
## - best performance: 0.01532051
##
## - Detailed performance results:
```

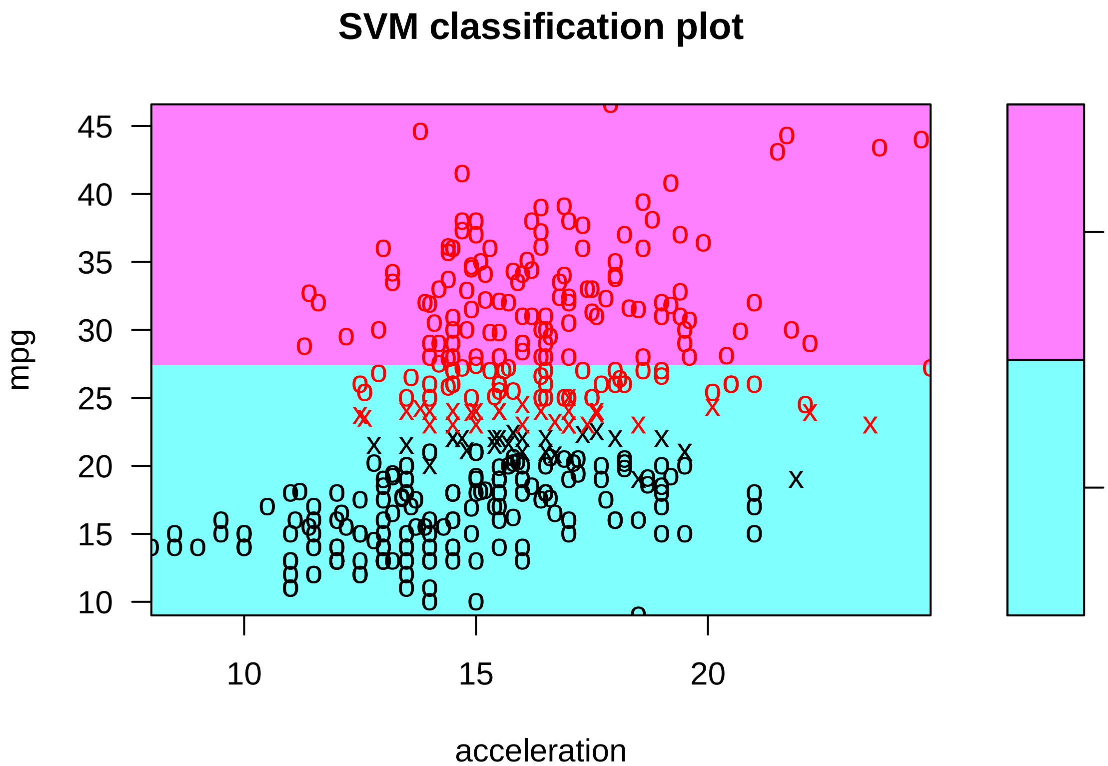
```
##     cost gamma      error dispersion
## 1  1e-02 1e-02 0.56115385 0.04344202
## 2  1e-01 1e-02 0.09185897 0.03862507
## 3  1e+00 1e-02 0.07147436 0.05103685
## 4  5e+00 1e-02 0.04326923 0.04975032
## 5  1e+01 1e-02 0.02551282 0.03812986
## 6  1e+02 1e-02 0.01532051 0.01788871
## 7  1e-02 1e-01 0.19153846 0.07612945
## 8  1e-01 1e-01 0.07916667 0.05201159
## 9  1e+00 1e-01 0.05608974 0.05092939
## 10 5e+00 1e-01 0.03064103 0.02637448
## 11 1e+01 1e-01 0.02551282 0.02076457
## 12 1e+02 1e-01 0.02807692 0.01458261
## 13 1e-02 1e+00 0.56115385 0.04344202
## 14 1e-01 1e+00 0.56115385 0.04344202
## 15 1e+00 1e+00 0.06634615 0.06187383
## 16 5e+00 1e+00 0.06128205 0.06186124
## 17 1e+01 1e+00 0.06128205 0.06186124
## 18 1e+02 1e+00 0.06128205 0.06186124
## 19 1e-02 5e+00 0.56115385 0.04344202
## 20 1e-01 5e+00 0.56115385 0.04344202
## 21 1e+00 5e+00 0.49224359 0.03806832
## 22 5e+00 5e+00 0.48967949 0.03738577
## 23 1e+01 5e+00 0.48967949 0.03738577
## 24 1e+02 5e+00 0.48967949 0.03738577
## 25 1e-02 1e+01 0.56115385 0.04344202
## 26 1e-01 1e+01 0.56115385 0.04344202
## 27 1e+00 1e+01 0.51775641 0.04471079
## 28 5e+00 1e+01 0.51012821 0.03817175
## 29 1e+01 1e+01 0.51012821 0.03817175
## 30 1e+02 1e+01 0.51012821 0.03817175
## 31 1e-02 1e+02 0.56115385 0.04344202
## 32 1e-01 1e+02 0.56115385 0.04344202
## 33 1e+00 1e+02 0.56115385 0.04344202
## 34 5e+00 1e+02 0.56115385 0.04344202
## 35 1e+01 1e+02 0.56115385 0.04344202
## 36 1e+02 1e+02 0.56115385 0.04344202
```

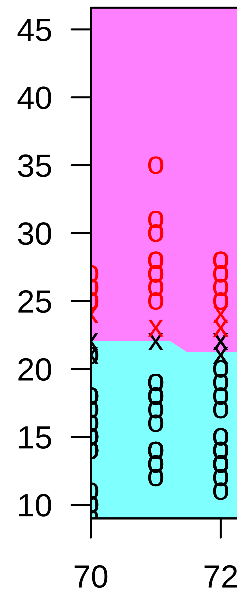
d

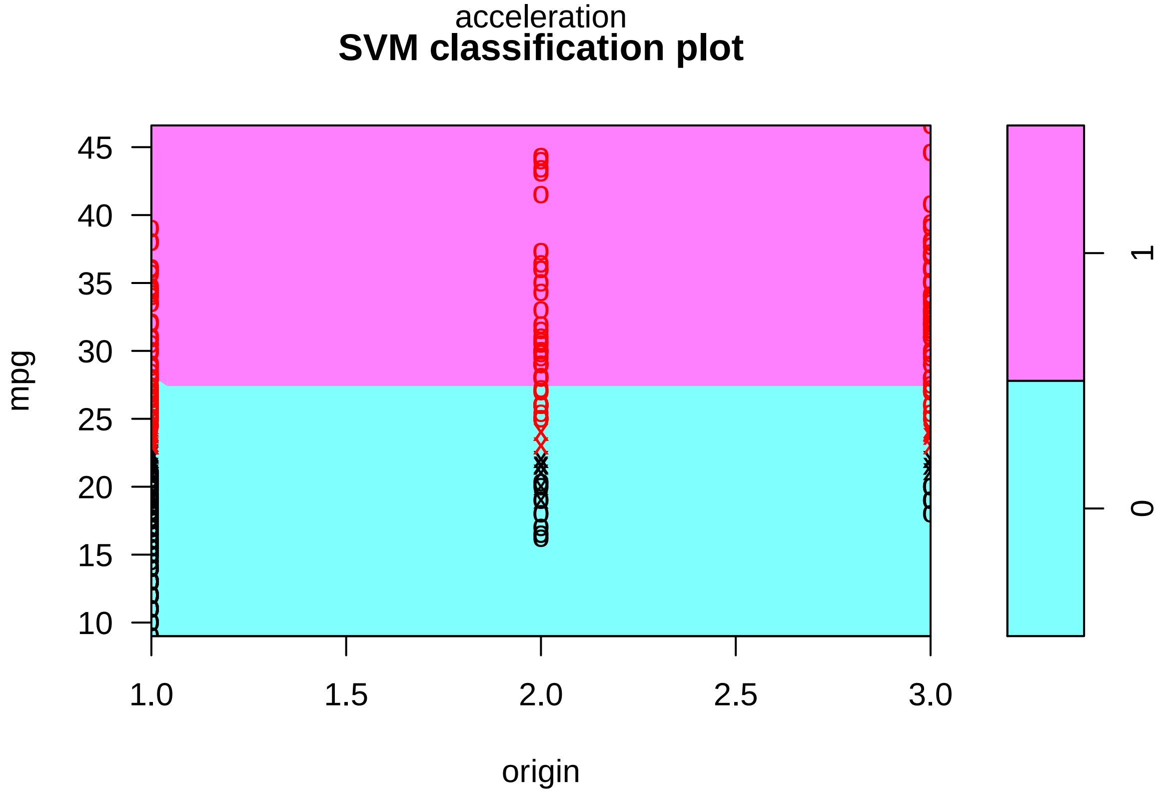
```r
svm.linear <- svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 1)
svm.poly <- svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 100, degree = 2)
svm.radial <- svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 100, gamma = 0.01)
plotpairs = function(fit) {
    for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpglevel", "name"))]) {
        plot(fit, Auto, as.formula(paste("mpg~", name, sep = "")))
    }
}
plotpairs(svm.linear)
```


SVM classification plot
mpg
45
40
35
30
25
20
15
10
3
4
5
6
7
8
cylinders
1
0
mpg
100

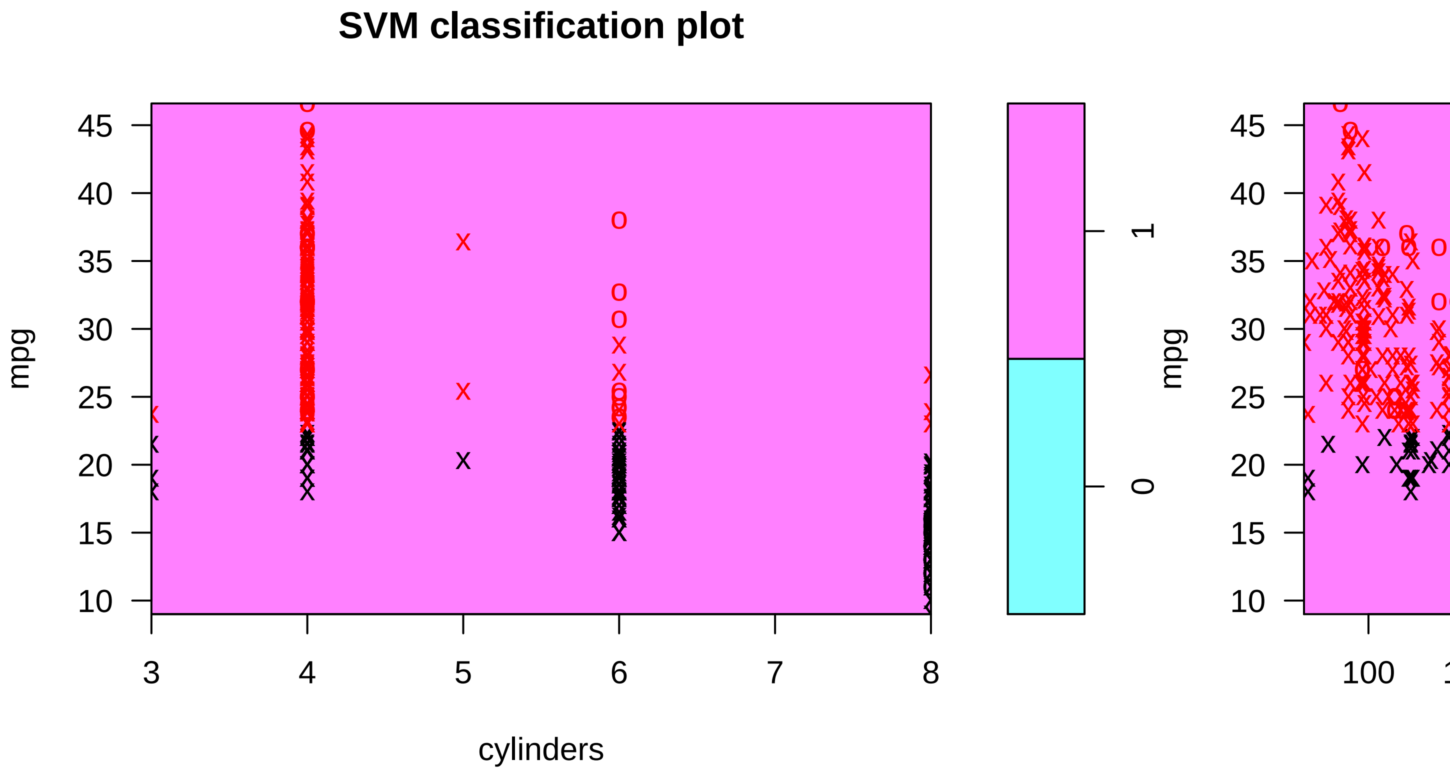

SVM classification plot
mpg
45
40
35
30
25
20
15
10
50
100
150
200
horsepower
1
0
mpg
2000

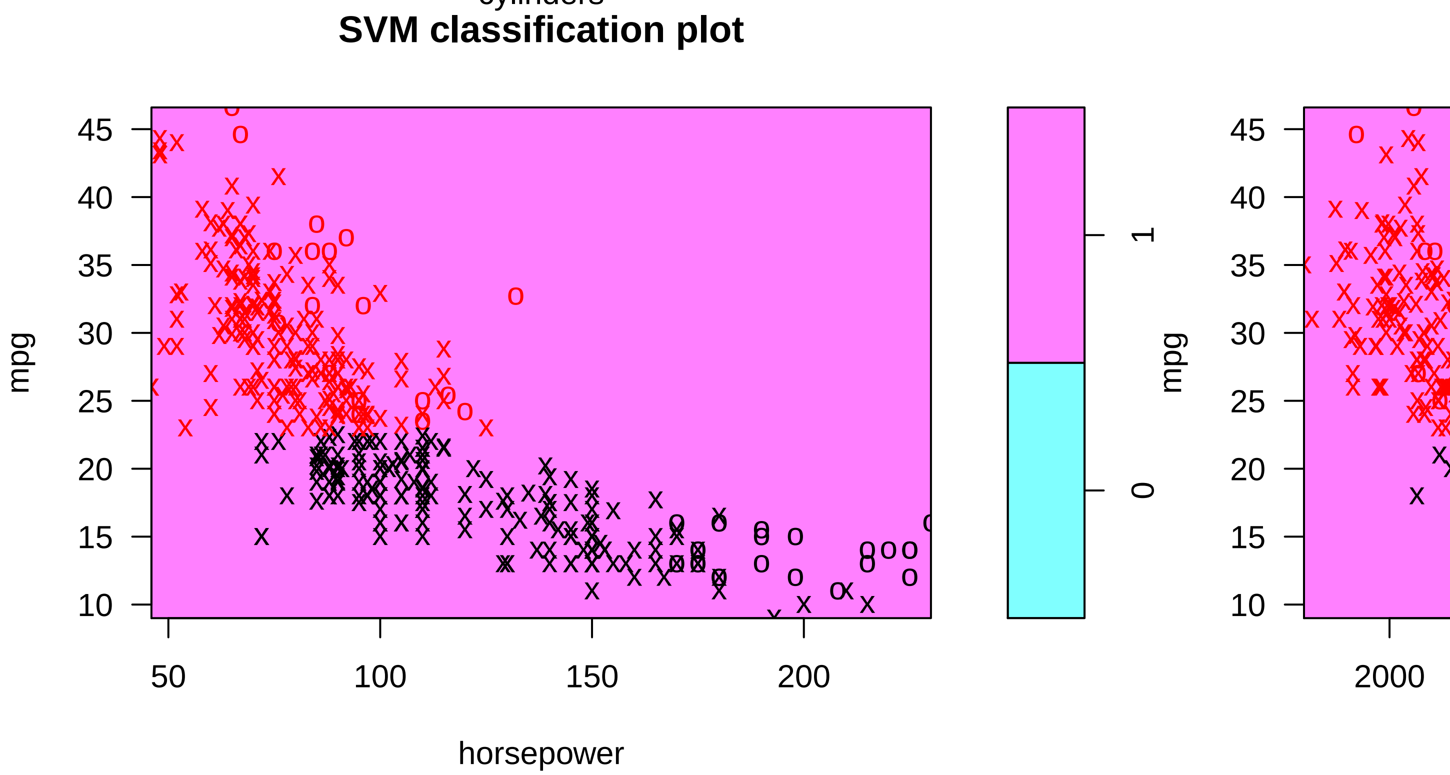


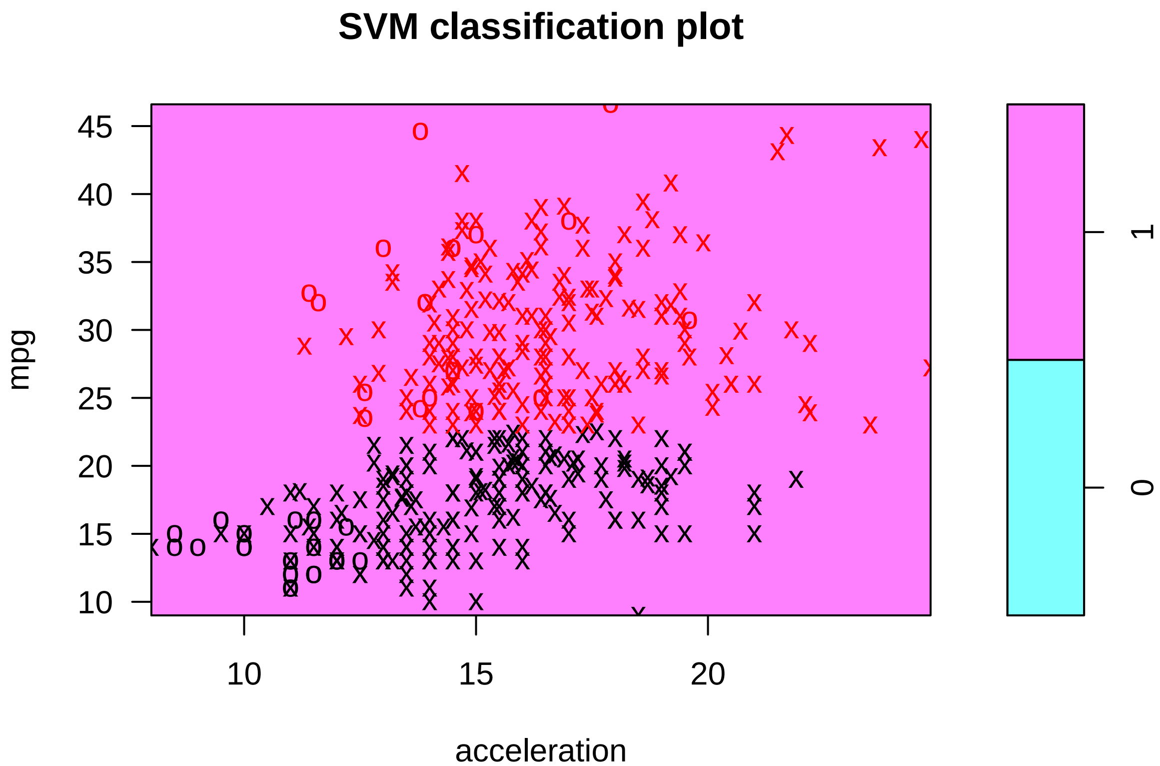


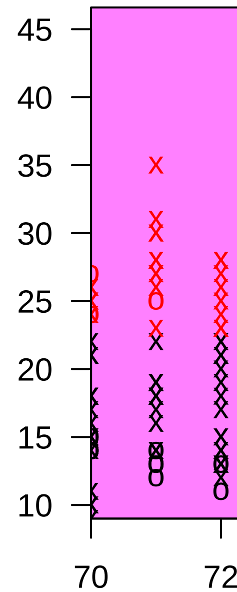


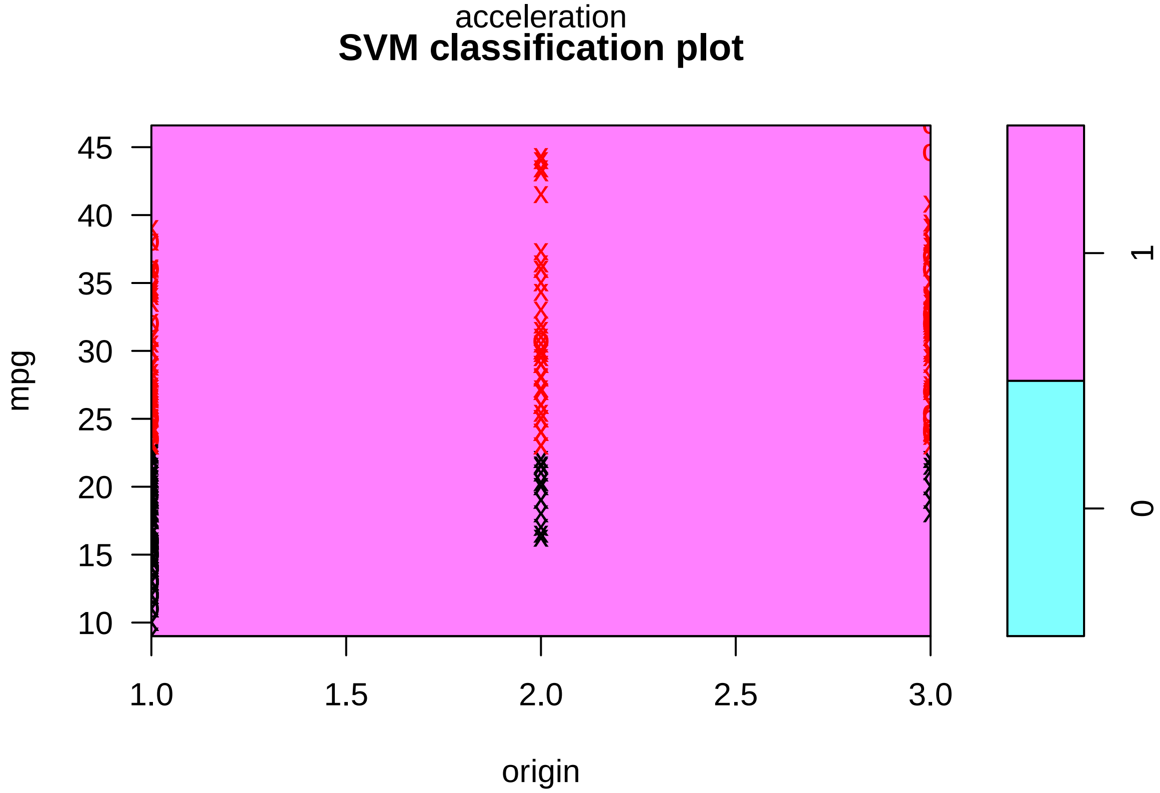

```
plotpairs(svm.poly)
```


SVM classification plot
mpg
45
40
35
30
25
20
15
10
3
4
5
6
7
8
cylinders
1
0
mpg
100

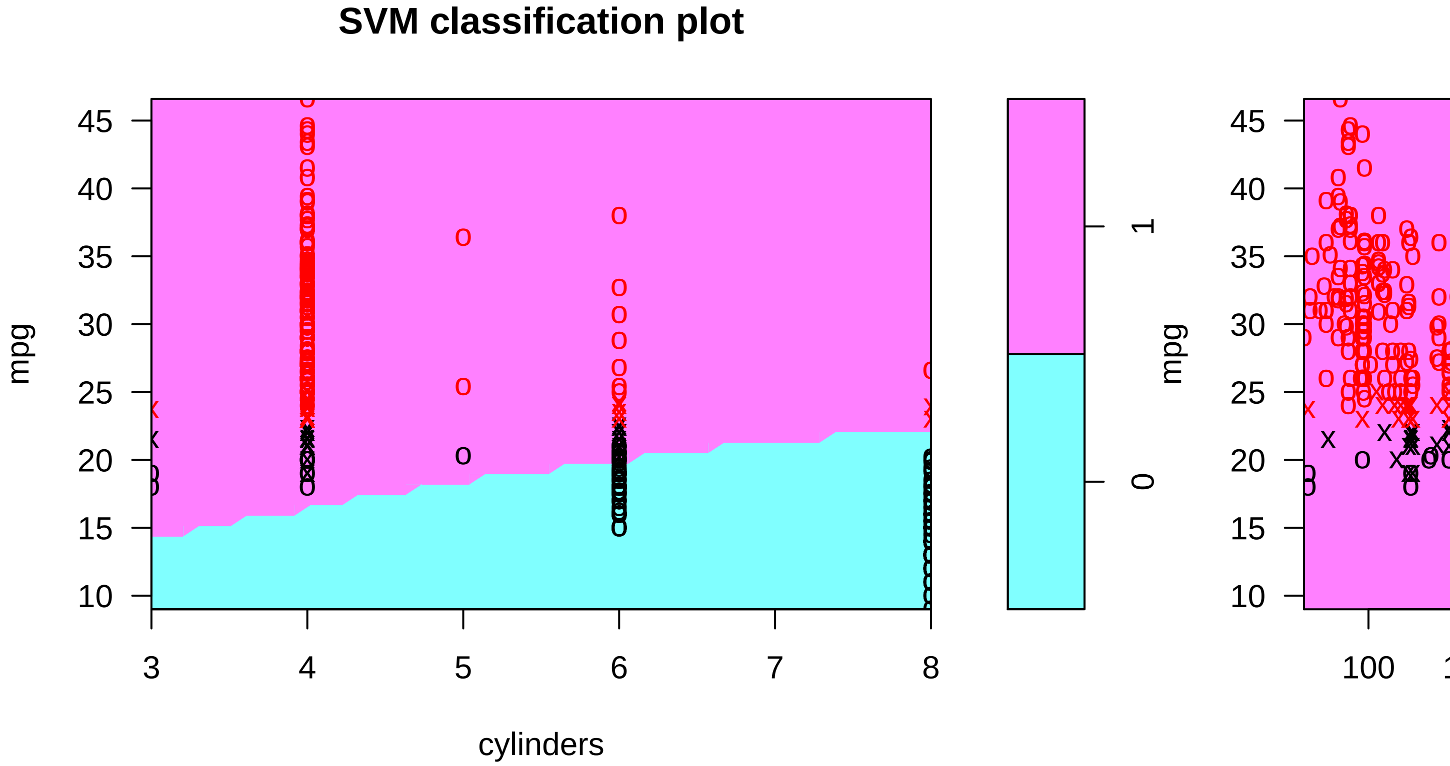

SVM classification plot
mpg
45
40
35
30
25
20
15
10
50
100
150
200
horsepower
1
0
mpg
2000

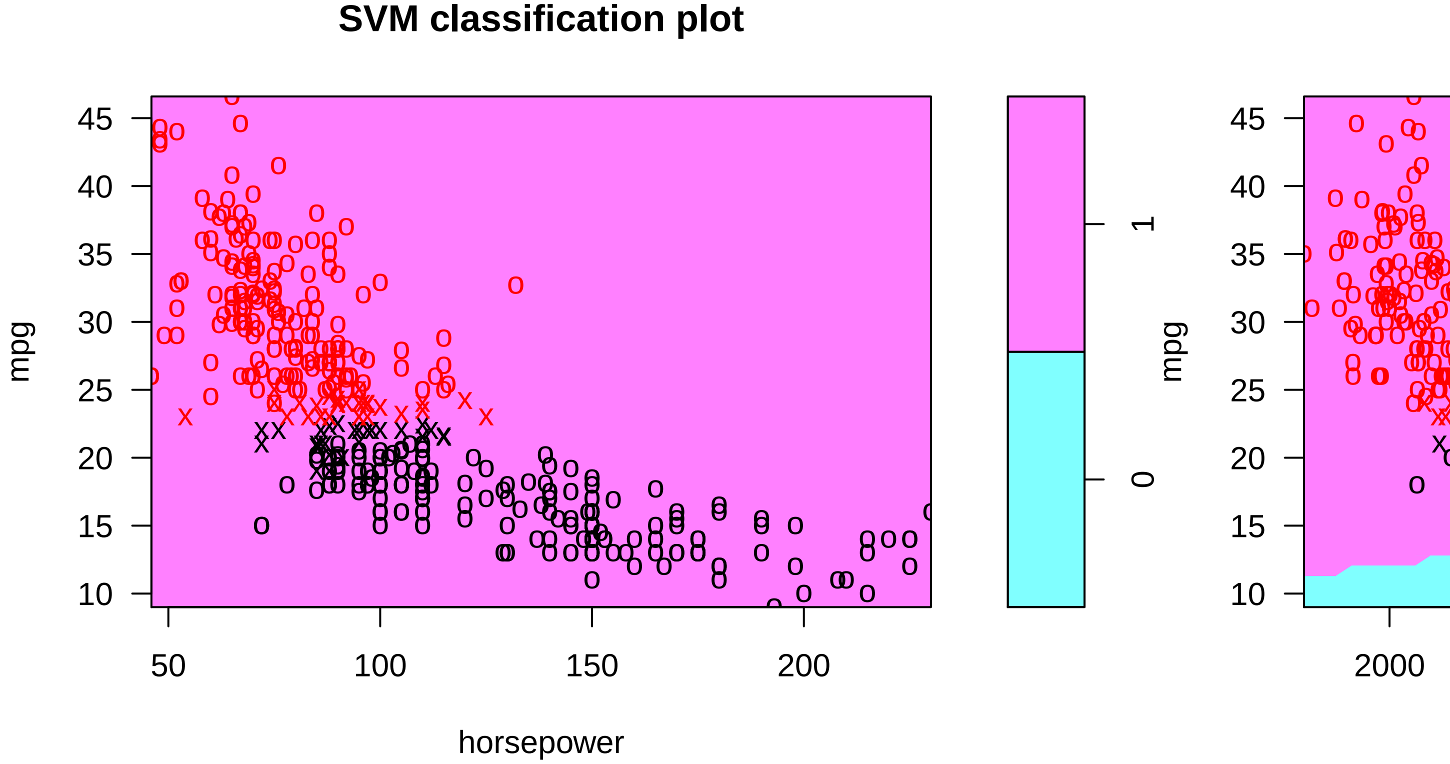




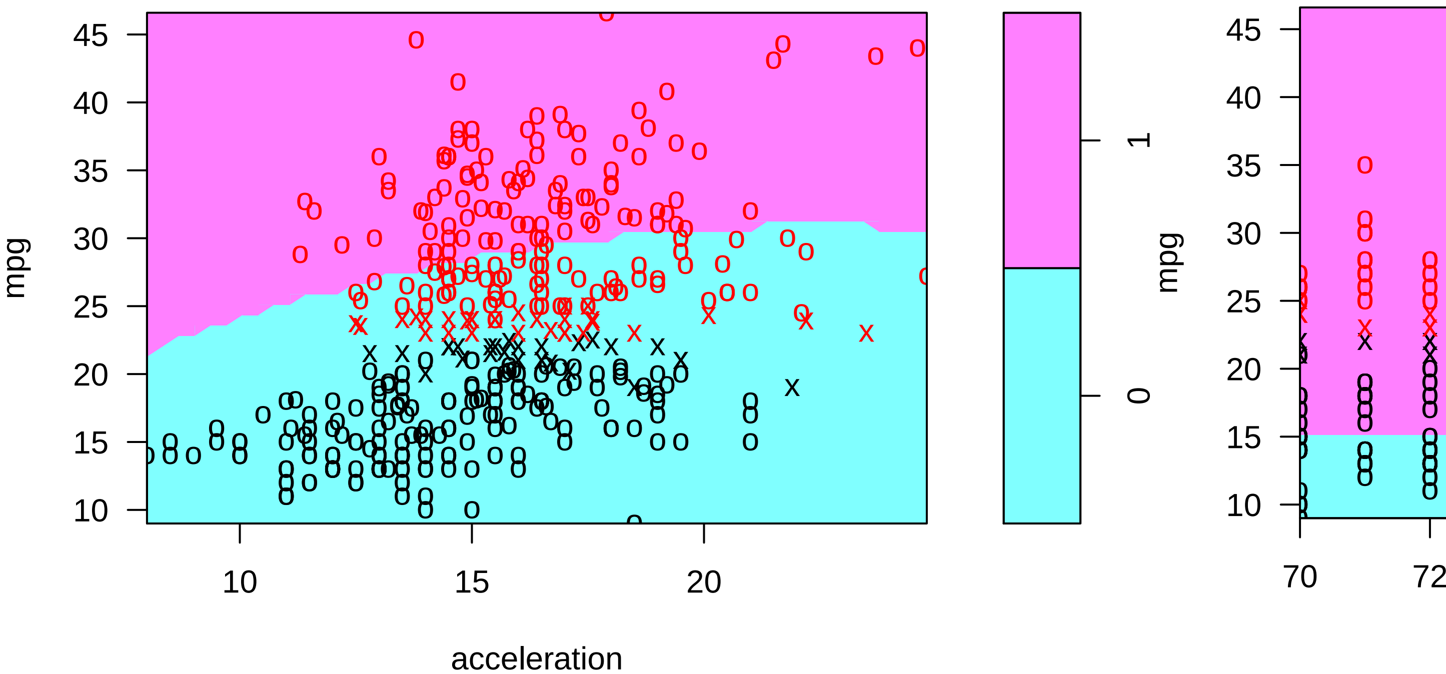



```
plotpairs(svm.radial)
```

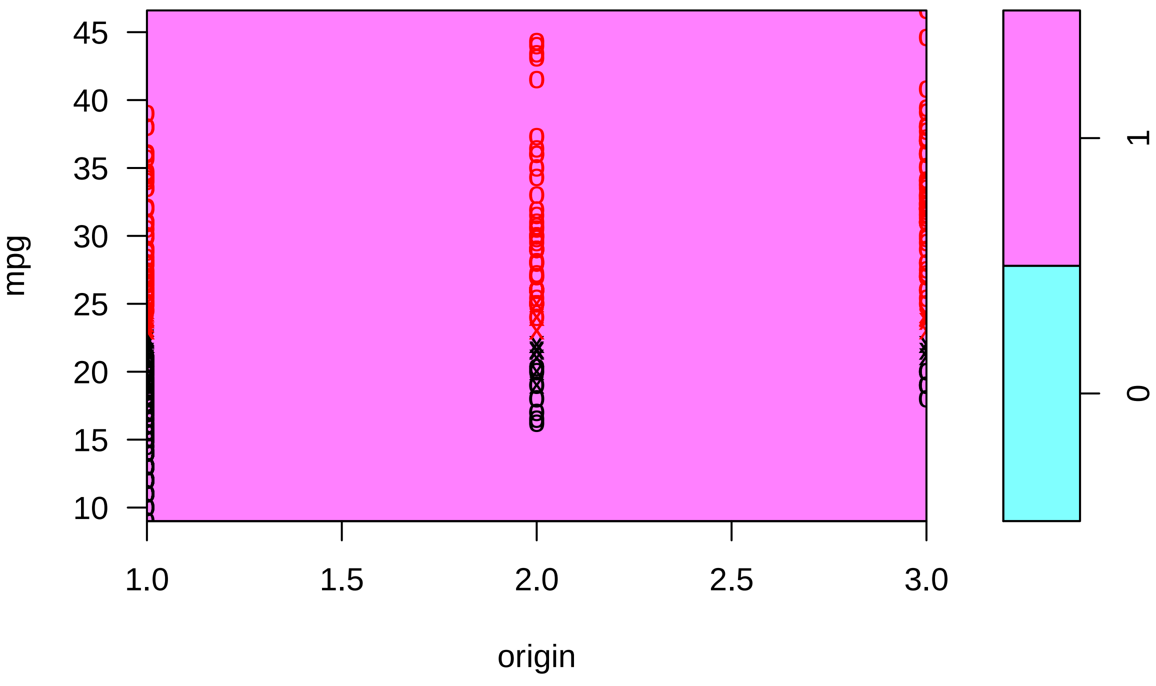

SVM classification plot
mpg
45
40
35
30
25
20
15
10
3
4
5
6
7
8
cylinders
1
0
100


SVM classification plot
mpg
45
40
35
30
25
20
15
10
50
100
150
200
horsepower
1
0
2000

SVM classification plot



SVM classification plot



Exercse 9.8

a

```r
set.seed(1)
train <- sample(nrow(OJ), 800)
OJ.train <- OJ[train, ]
OJ.test <- OJ[-train, ]
```

b

```r
svm.linear <- svm(Purchase ~ ., data = OJ.train, kernel = "linear", cost = 0.01)
summary(svm.linear)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  432
##
##  ( 215 217 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

c

```r
train.pred <- predict(svm.linear, OJ.train)
table(OJ.train$Purchase, train.pred)
```

```
##     train.pred
##       CH  MM
##   CH 439  55
##   MM  78 228
```

```r
(78 + 55) / (439 + 228 + 78 + 55)
```

```
## [1] 0.16625
```

```r
test.pred <- predict(svm.linear, OJ.test)
table(OJ.test$Purchase, test.pred)
```

```
##     test.pred
##       CH  MM
##   CH 141  18
##   MM  31  80
```

```r
(31 + 18) / (141 + 80 + 31 + 18)
```

```
## [1] 0.1814815
```