高级统计方法 第9次作业:

序号： 08 姓名： 潘晨楷 学号： 20212241116 班级：软2107

**作业评阅：**

**概念**

2.问题（略）

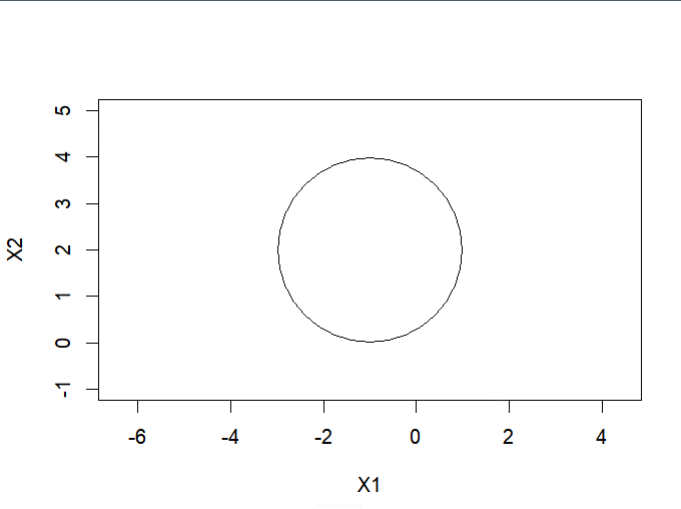
（a）问题（略）

radius = 2

plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1",

ylab = "X2")

symbols(c(-1), c(2), circles = c(radius), add = TRUE, inches = FALSE)



（b）问题（略）

radius = 2

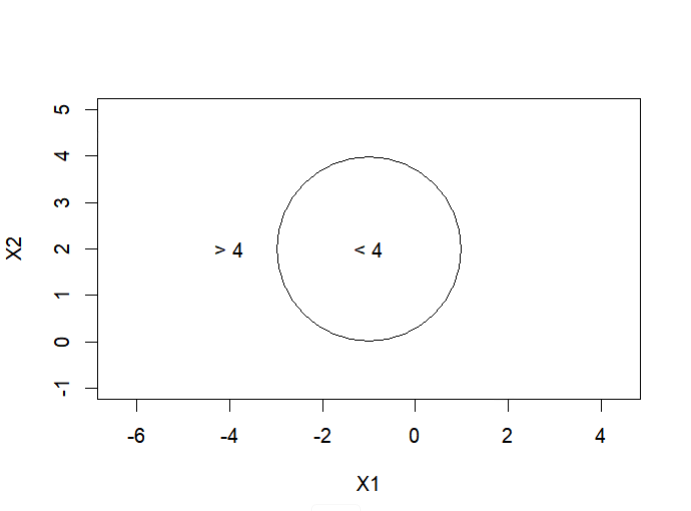
plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1",

ylab = "X2")

symbols(c(-1), c(2), circles = c(radius), add = TRUE, inches = FALSE)

text(c(-1), c(2), "< 4")

text(c(-4), c(2), "> 4")



（c）问题（略）

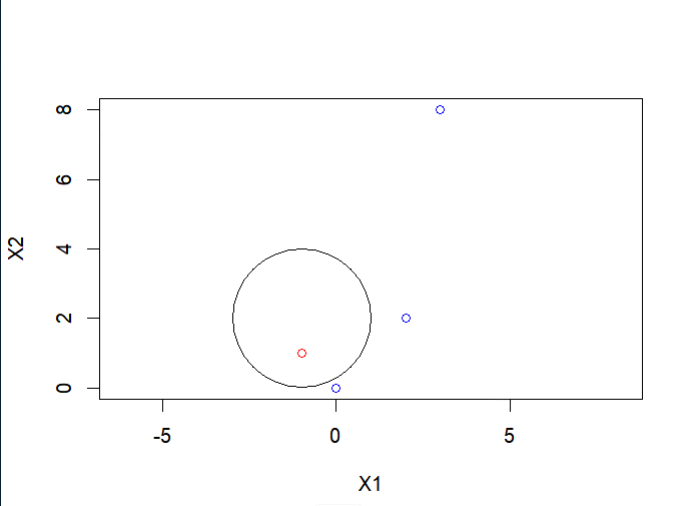
为了重申边界，圆圈外为蓝色，圆圈内和圆圈上为红色。

radius = 2

plot(c(0, -1, 2, 3), c(0, 1, 2, 8), col = c("blue", "red", "blue", "blue"),

type = "p", asp = 1, xlab = "X1", ylab = "X2")

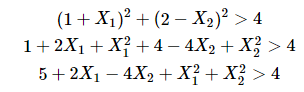
symbols(c(-1), c(2), circles = c(radius), add = TRUE, inches = FALSE)



（-1，-1）蓝色， （0，0）蓝色， （3，8）蓝色

（d）问题（略）

决策边界在展开时是二次项的和。



3.问题（略）

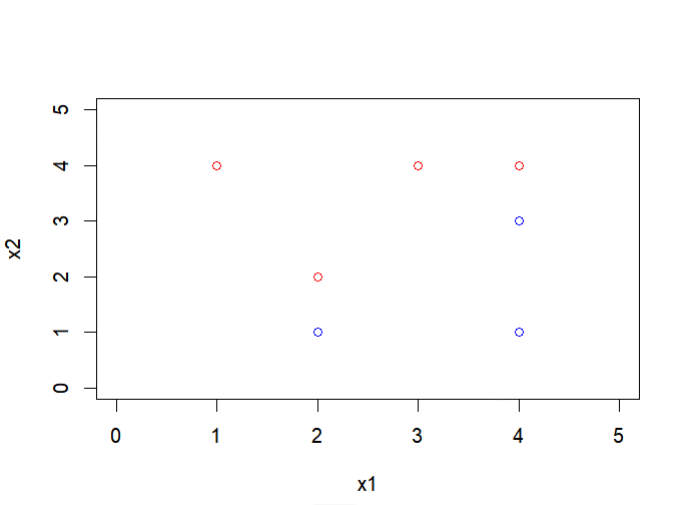
（a）问题（略）

x1 = c(3, 2, 4, 1, 2, 4, 4)

x2 = c(4, 2, 4, 4, 1, 3, 1)

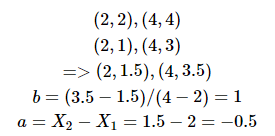
colors = c("red", "red", "red", "red", "blue", "blue", "blue")

plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))



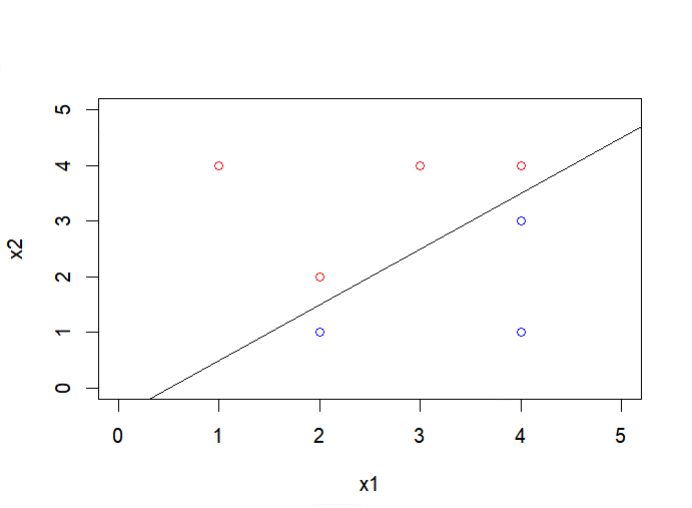
（b）问题（略）

最大边界分类器必须位于观测值#2、#3和#5、#6之间。



plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))

abline(-0.5, 1)



（c）问题（略）

0.5−X1+X2>0

β0 = 0.5，β1=-1，β2 = 1

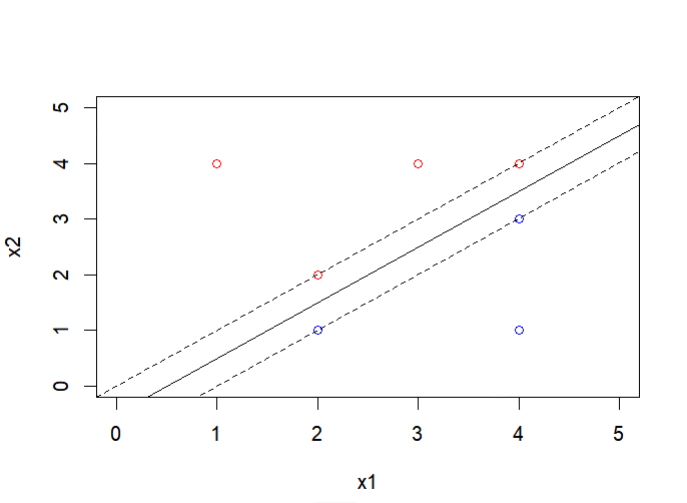
（d）问题（略）

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）问题（略）

plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))

abline(-0.5, 1)

arrows(2, 1, 2, 1.5)

arrows(2, 2, 2, 1.5)

arrows(4, 4, 4, 3.5)

arrows(4, 3, 4, 3.5)



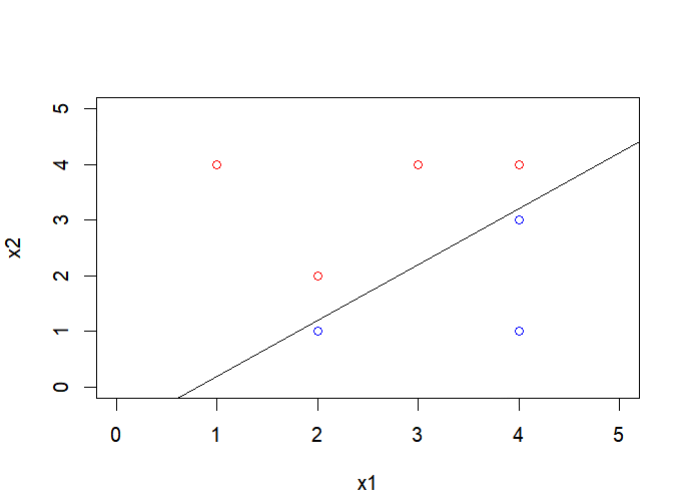
（f）问题（略）

观察#7(4,1)蓝色的轻微移动不会对最大边距超平面产生影响，因为它的移动将在边距之外。

（g）问题（略）

plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))

abline(-0.8, 1)

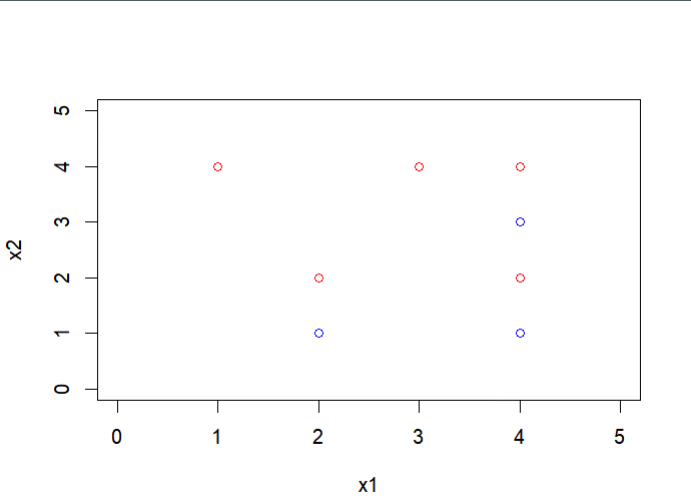


−0.8−X1+X2>0

（h）问题（略）

plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))

points(c(4), c(2), col = c("red"))



**应用**

4.问题（略）

我们创建一个随机的初始数据集，它位于抛物线y=3 \* x2+4。然后我们通过沿y轴平移来分离这两个类。

set.seed(131)

x = rnorm(100)

y = 3 \* x^2 + 4 + rnorm(100)

train = sample(100, 50)

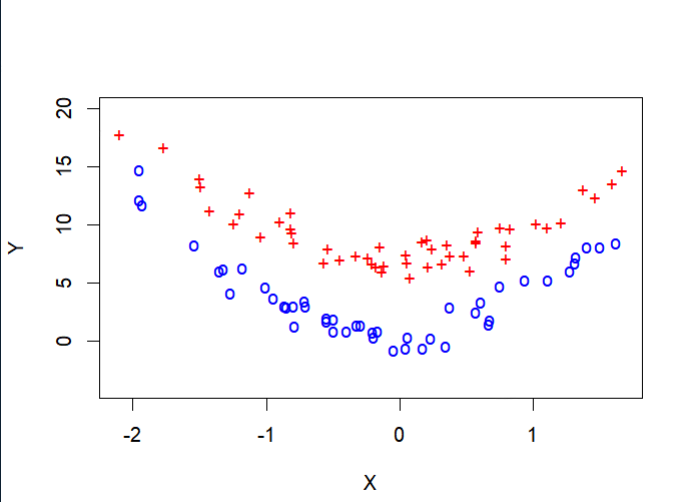
y[train] = y[train] + 3

y[-train] = y[-train] - 3

# Plot using different colors

plot(x[train], y[train], pch="+", lwd=4, col="red", ylim=c(-4, 20), xlab="X", ylab="Y")

points(x[-train], y[-train], pch="o", lwd=4, col="blue")



该图清楚地显示了非线性分离。现在我们通过取一半的正类和负类并为类创建一个新的z向量0和1来创建训练和测试数据帧。

set.seed(315)

z = rep(0, 100)

z[train] = 1

# Take 25 observations each from train and -train

final.train = c(sample(train, 25), sample(setdiff(1:100, train), 25))

data.train = data.frame(x=x[final.train], y=y[final.train], z=as.factor(z[final.train]))

data.test = data.frame(x=x[-final.train], y=y[-final.train], z=as.factor(z[-final.train]))

install.packages(“e1071”)

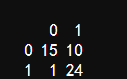
library(e1071)

svm.linear = svm(z~., data=data.train, kernel="linear", cost=10)

plot(svm.linear, data.train)



table(z[final.train], predict(svm.linear, data.train))



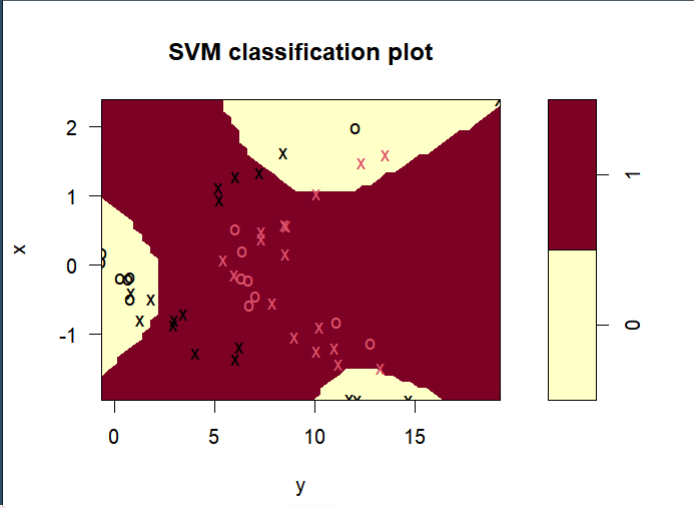
该图显示了线性边界。该分类器对列车数据进行了11次分类错误。

接下来，我们训练了一个多项式核支持向量机

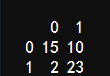
set.seed(32545)

svm.poly = svm(z~., data=data.train, kernel="polynomial", cost=10)

plot(svm.poly, data.train)



table(z[final.train], predict(svm.poly, data.train))



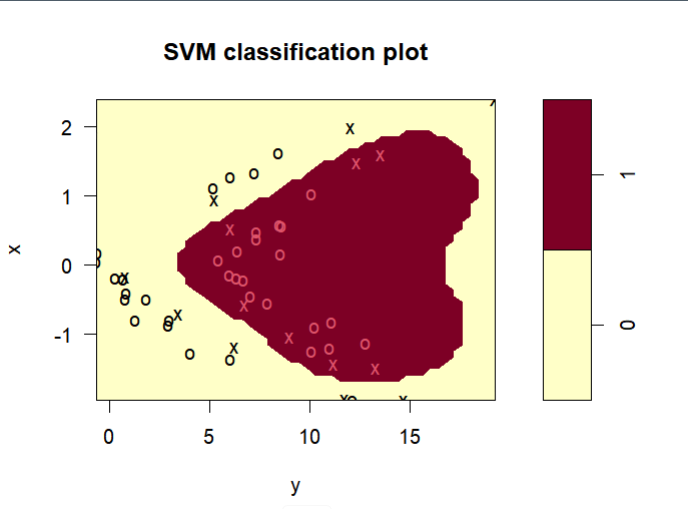
这是一个默认的3次多项式核。它在列车数据上有12个错误。

最后，我们训练了一个gamma为1的径向基核支持向量机

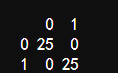
set.seed(996)

svm.radial = svm(z~., data=data.train, kernel="radial", gamma=1, cost=10)

plot(svm.radial, data.train)



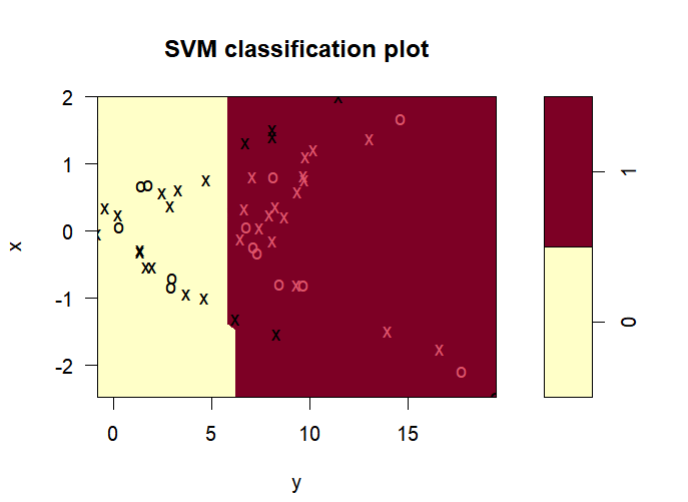
table(z[final.train], predict(svm.radial, data.train))



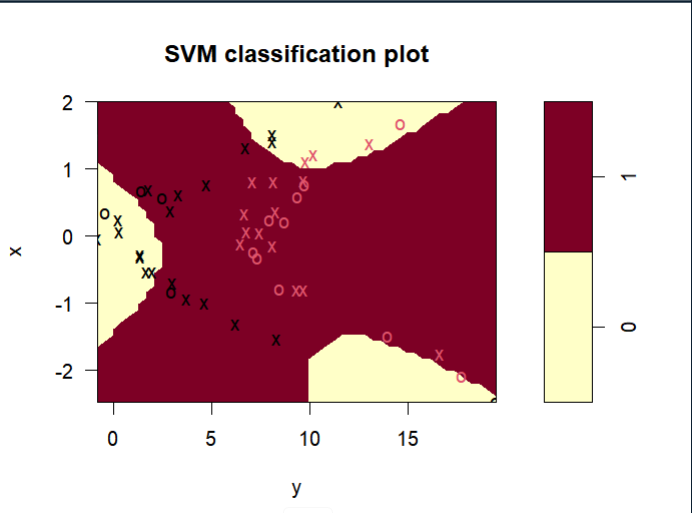
这个分类器可以很好地分类训练数据!

下面是测试错误的样子。

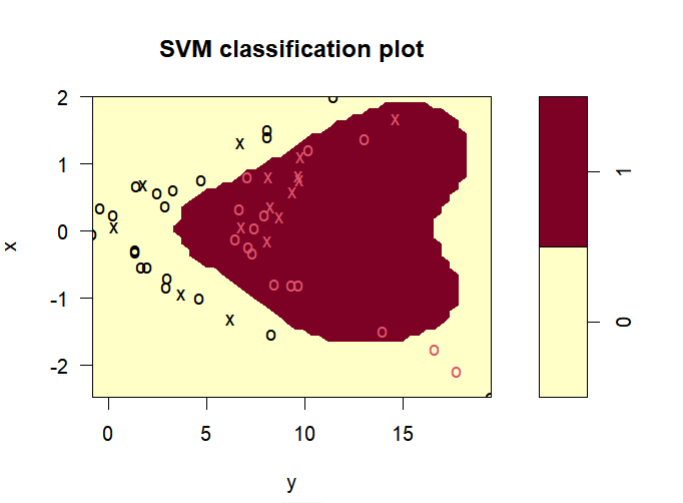
plot(svm.linear, data.test)



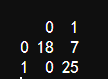
plot(svm.poly, data.test)



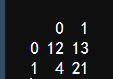
plot(svm.radial, data.test)



table(z[-final.train], predict(svm.linear, data.test))



table(z[-final.train], predict(svm.poly, data.test))



table(z[-final.train], predict(svm.radial, data.test))



从表中可以看出，线性基核、多项式基核和径向基核分别对7个、17个和2个测试点进行了错误分类。

5.问题（略）

（a）问题（略）

set.seed(421)

x1 = runif(500) - 0.5

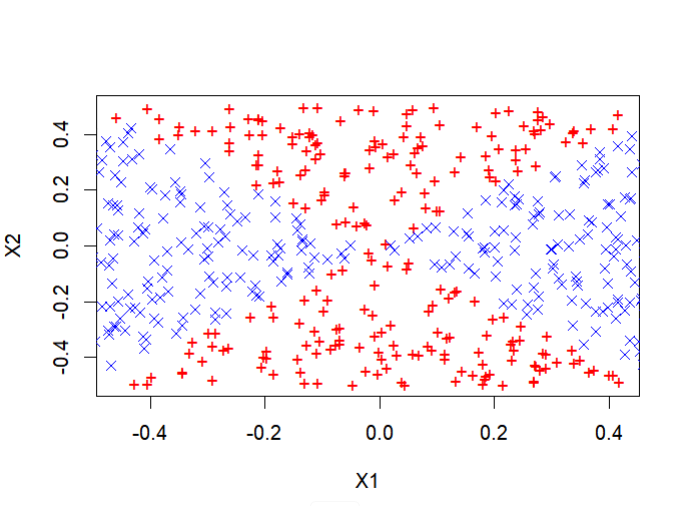
x2 = runif(500) - 0.5

y = 1 \* (x1^2 - x2^2 > 0)

（b）问题（略）

plot(x1[y == 0], x2[y == 0], col = "red", xlab = "X1", ylab = "X2", pch = "+")

points(x1[y == 1], x2[y == 1], col = "blue", pch = 4)

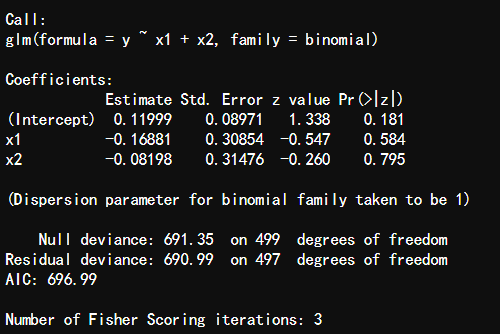


该图清晰地显示了非线性决策边界。

（c）问题（略）

lm.fit = glm(y ~ x1 + x2, family = binomial)

summary(lm.fit)



这两个变量对于预测y都是不显著的。

（d）问题（略）

data = data.frame(x1 = x1, x2 = x2, y = y)

lm.prob = predict(lm.fit, data, type = "response")

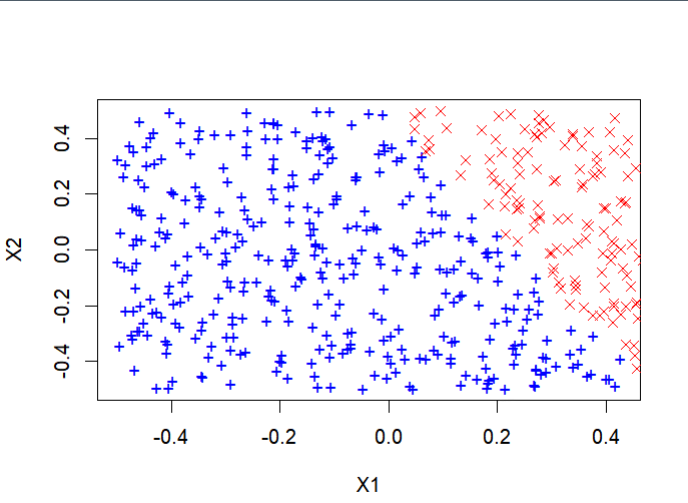
lm.pred = ifelse(lm.prob > 0.52, 1, 0)

data.pos = data[lm.pred == 1, ]

data.neg = data[lm.pred == 0, ]

plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2", pch = "+")

points(data.neg$x1, data.neg$x2, col = "red", pch = 4)



在给定的模型和概率阈值为0.5的情况下，所有点都被归为单一类，无法显示决策边界。因此，我们将概率阈值移至0.52，以显示有意义的决策边界。如图所示，这个边界是线性的。

（e）问题（略）

我们使用平方，产品交互项来拟合模型。

lm.fit = glm(y ~ poly(x1, 2) + poly(x2, 2) + I(x1 \* x2), data = data, family = binomial)

（f）问题（略）

lm.prob = predict(lm.fit, data, type = "response")

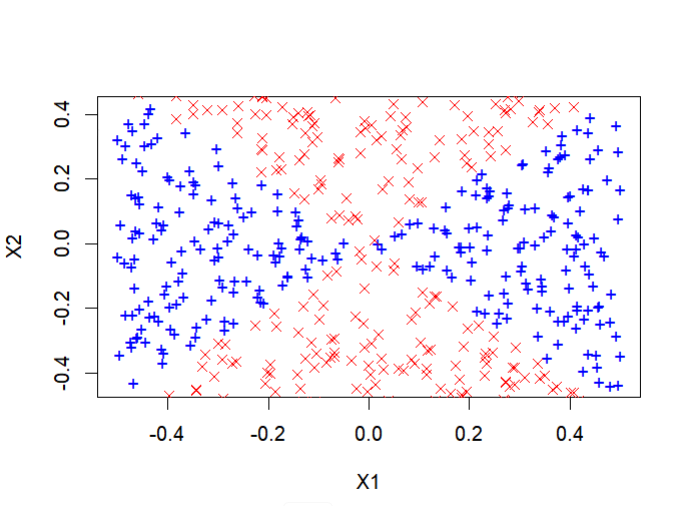
lm.pred = ifelse(lm.prob > 0.5, 1, 0)

data.pos = data[lm.pred == 1, ]

data.neg = data[lm.pred == 0, ]

plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2", pch = "+")

points(data.neg$x1, data.neg$x2, col = "red", pch = 4)



这种非线性决策边界与真正的决策边界非常相似。

（g）问题（略）

library(e1071)

svm.fit = svm(as.factor(y) ~ x1 + x2, data, kernel = "linear", cost = 0.1)

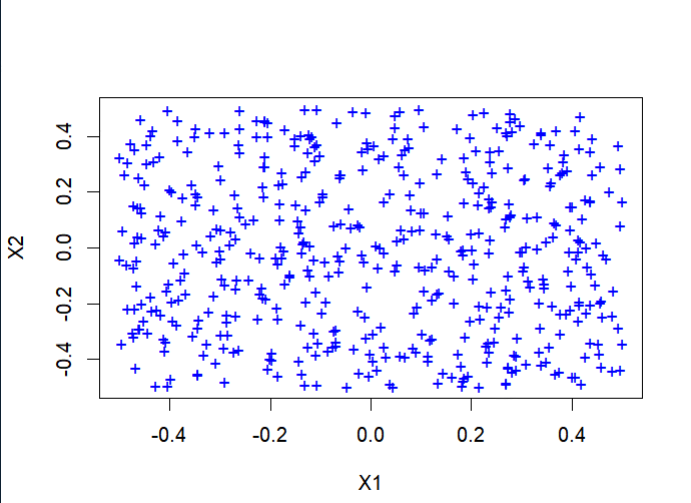
svm.pred = predict(svm.fit, data)

data.pos = data[svm.pred == 1, ]

data.neg = data[svm.pred == 0, ]

plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2", pch = "+")

points(data.neg$x1, data.neg$x2, col = "red", pch = 4)



线性核函数即使代价很低，也无法找到非线性决策边界，并将所有点归为一类。

（h）问题（略）

svm.fit = svm(as.factor(y) ~ x1 + x2, data, gamma = 1)

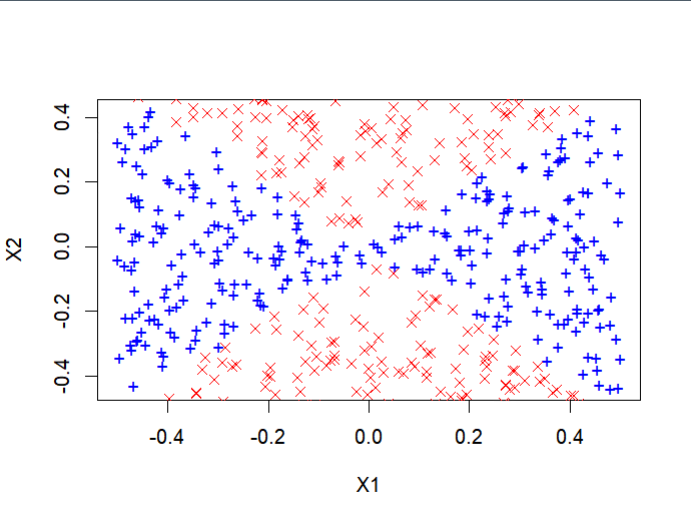
svm.pred = predict(svm.fit, data)

data.pos = data[svm.pred == 1, ]

data.neg = data[svm.pred == 0, ]

plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2", pch = "+")

points(data.neg$x1, data.neg$x2, col = "red", pch = 4)



同样，预测标签上的非线性决策边界与真实决策边界非常相似。

1. 问题（略）

这个实验强化了具有非线性核的支持向量机在寻找非线性边界方面非常强大的想法。非交互逻辑回归和线性核支持向量机都不能找到决策边界。将交互项添加到逻辑回归中似乎使它们具有与径向基核相同的功能。然而，在选择正确的交互术语时需要一些手工工作和调优。如果有大量的特性，这种努力可能会变得令人望而却步。另一方面，径向基核只需要调整一个参数-伽马-这可以很容易地通过交叉验证来完成。

6.问题（略）

（a）问题（略）

我们随机生成1000个点，并将它们分散在x=y线上。我们还沿着5x−4y−50=0这条线创建了噪声点。这些点使得类几乎不可分，也改变了最大边界分类器。

set.seed(3154)

# Class one

x.one = runif(500, 0, 90)

y.one = runif(500, x.one + 10, 100)

x.one.noise = runif(50, 20, 80)

y.one.noise = 5/4 \* (x.one.noise - 10) + 0.1

# Class zero

x.zero = runif(500, 10, 100)

y.zero = runif(500, 0, x.zero - 10)

x.zero.noise = runif(50, 20, 80)

y.zero.noise = 5/4 \* (x.zero.noise - 10) - 0.1

# Combine all

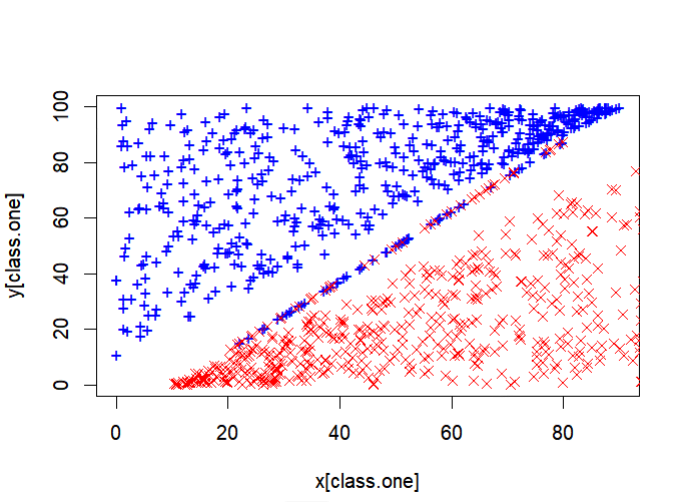
class.one = seq(1, 550)

x = c(x.one, x.one.noise, x.zero, x.zero.noise)

y = c(y.one, y.one.noise, y.zero, y.zero.noise)

plot(x[class.one], y[class.one], col = "blue", pch = "+", ylim = c(0, 100))

points(x[-class.one], y[-class.one], col = "red", pch = 4)



该图显示，类几乎是不可分的。噪声点创建一个虚构的边界5x−4y−50=0。

（b）问题（略）

根据类创建一个z变量。

library(e1071)

set.seed(555)

z = rep(0, 1100)

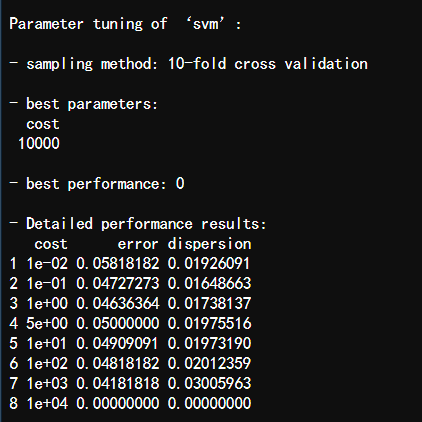
z[class.one] = 1

data = data.frame(x = x, y = y, z = z)

tune.out = tune(svm, as.factor(z) ~ ., data = data, kernel = "linear", ranges = list(cost = c(0.01,

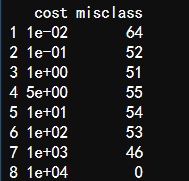
0.1, 1, 5, 10, 100, 1000, 10000)))

summary(tune.out)



data.frame(cost = tune.out$performances$cost, misclass = tune.out$performances$error \*

1100)



上表显示了所有费用的列车误分类误差。花费10000似乎可以正确地分类所有的点。这也对应于0的交叉验证误差。

（c）问题（略）

现在我们生成一个相同大小的随机测试集。这个测试集满足真决策边界x=y。

set.seed(1111)

x.test = runif(1000, 0, 100)

class.one = sample(1000, 500)

y.test = rep(NA, 1000)

# Set y > x for class.one

for (i in class.one) {

y.test[i] = runif(1, x.test[i], 100)

}

# set y < x for class.zero

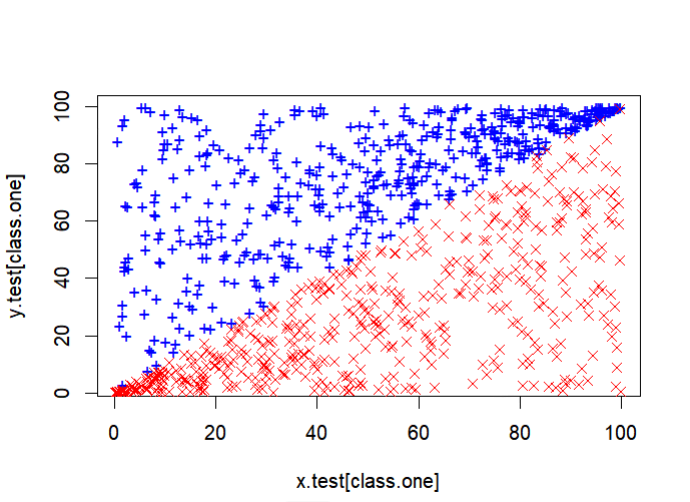
for (i in setdiff(1:1000, class.one)) {

y.test[i] = runif(1, 0, x.test[i])

}

plot(x.test[class.one], y.test[class.one], col = "blue", pch = "+")

points(x.test[-class.one], y.test[-class.one], col = "red", pch = 4)



我们现在使用所有线性支持向量机进行相同的预测，并使用前一部分中使用的所有成本

set.seed(30012)

z.test = rep(0, 1000)

z.test[class.one] = 1

all.costs = c(0.01, 0.1, 1, 5, 10, 100, 1000, 10000)

test.errors = rep(NA, 8)

data.test = data.frame(x = x.test, y = y.test, z = z.test)

for (i in 1:length(all.costs)) {

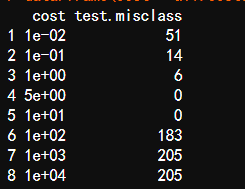
svm.fit = svm(as.factor(z) ~ ., data = data, kernel = "linear", cost = all.costs[i])

svm.predict = predict(svm.fit, data.test)

test.errors[i] = sum(svm.predict != data.test$z)

}

data.frame(cost = all.costs, `test misclass` = test.errors)



Cost =5和10似乎在测试数据上表现更好，产生的分类错误最少。这比训练数据的最优值10000小得多。

（d）问题（略）

我们再次看到线性核的过拟合现象。一个大的代价试图正确拟合分类噪声点，从而过拟合列车数据。然而，成本小，在有噪声的测试点上误差小，在测试数据上表现较好。

7.问题（略）

（a）问题（略）

library(ISLR)

gas.med = median(Auto$mpg)

new.var = ifelse(Auto$mpg > gas.med, 1, 0)

Auto$mpglevel = as.factor(new.var)

（b）问题（略）

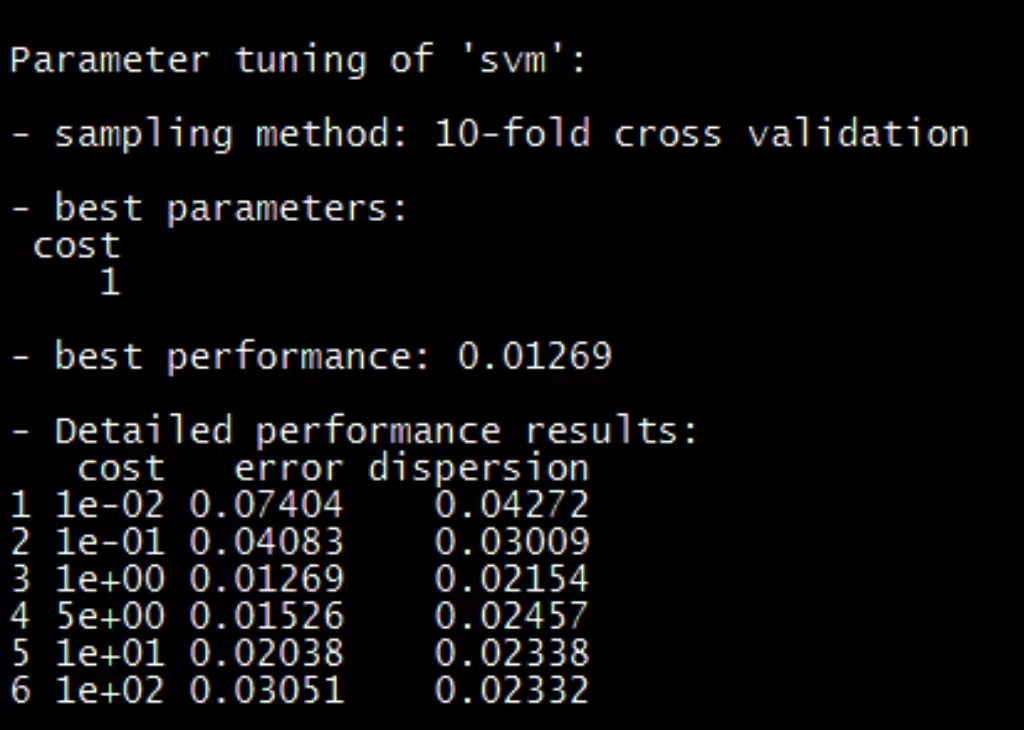
library(e1071)

set.seed(3255)

tune.out = tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges = list(cost = c(0.01,

0.1, 1, 5, 10, 100)))

summary(tune.out)



我们看到，当cost=1时，交叉验证误差最小。

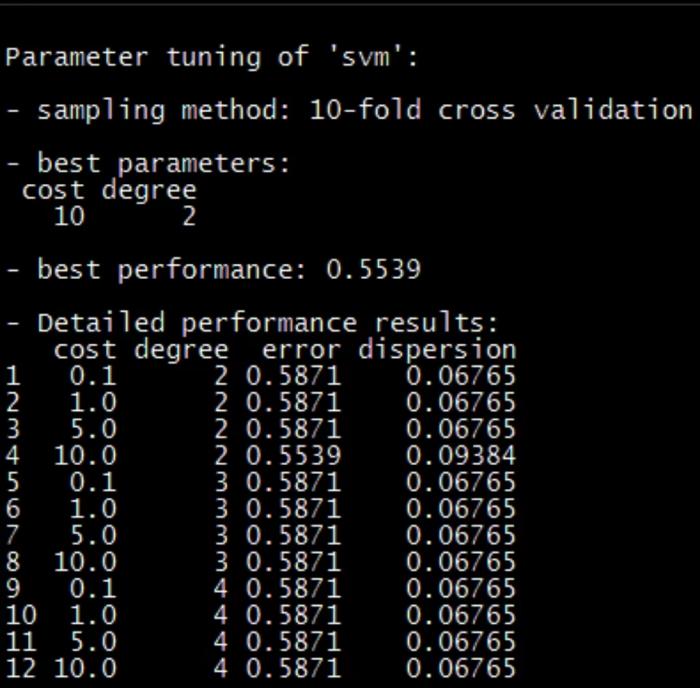
（c）问题（略）

set.seed(21)

tune.out = tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges = list(cost = c(0.1,

1, 5, 10), degree = c(2, 3, 4)))

summary(tune.out)



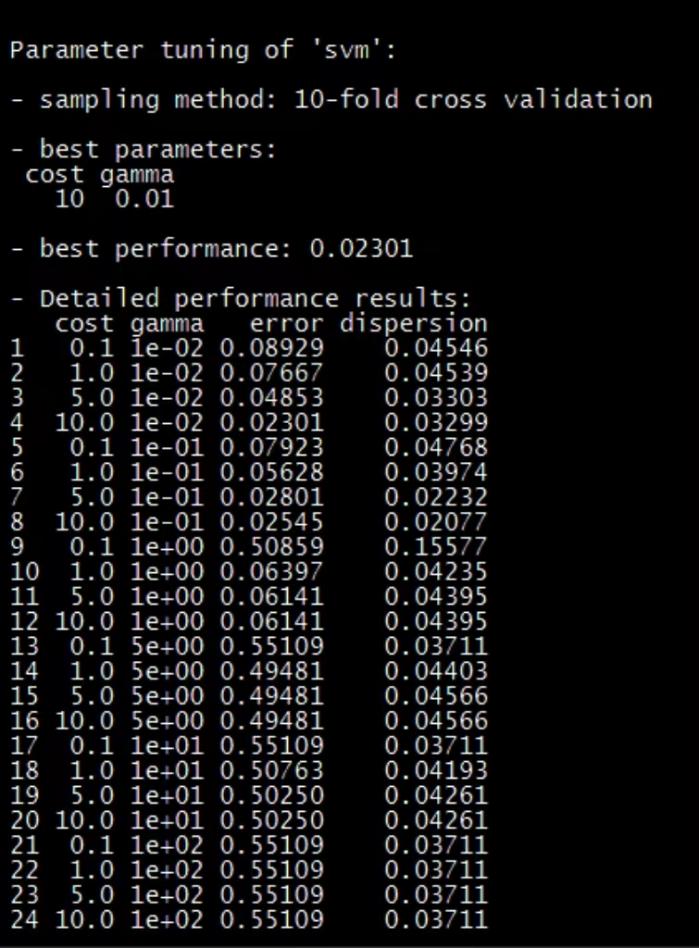
当cost=10, degree=2时，交叉验证误差最小。

set.seed(463)

tune.out = tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges = list(cost = c(0.1,

1, 5, 10), gamma = c(0.01, 0.1, 1, 5, 10, 100)))

summary(tune.out)



最后，对于径向基核，cost=10, gamma=0.01。

（d）问题（略）

svm.linear = svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 1)

svm.poly = svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 10,

degree = 2)

svm.radial = svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 10, gamma = 0.01)

plotpairs = function(fit) {

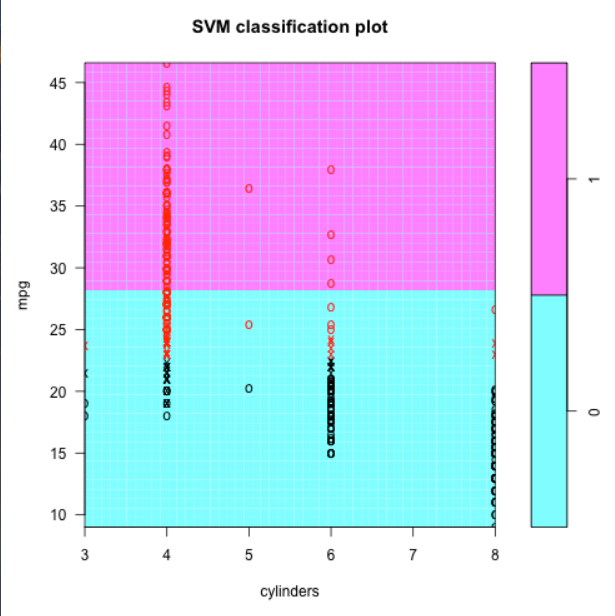
for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpglevel", "name"))]) {

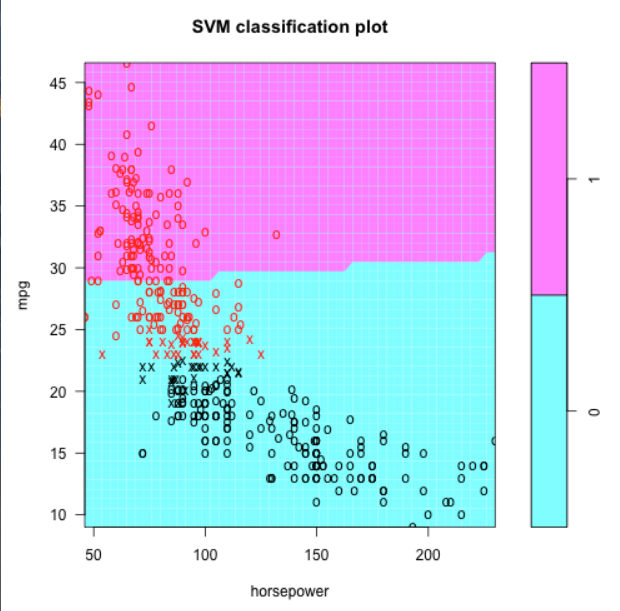
plot(fit, Auto, as.formula(paste("mpg~", name, sep = "")))

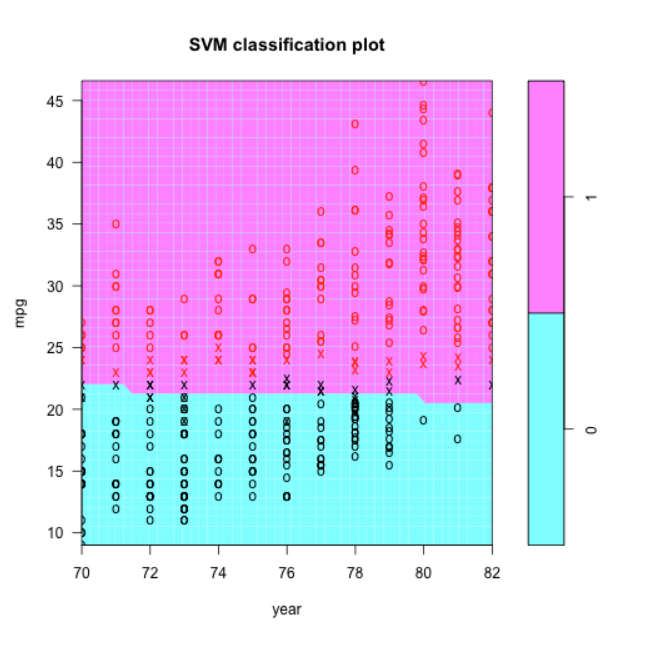
}

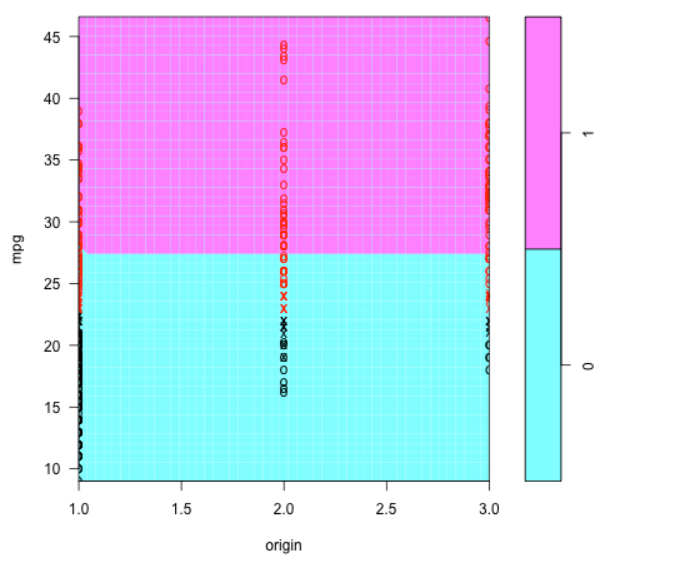
}

plotpairs(svm.linear)

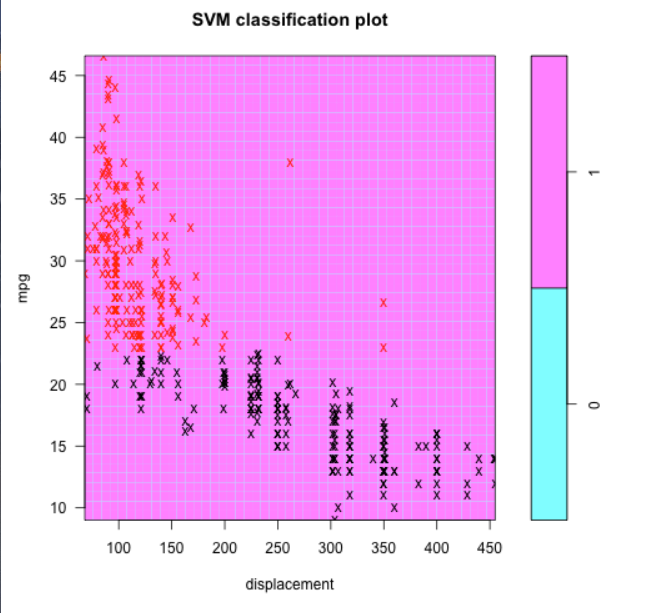
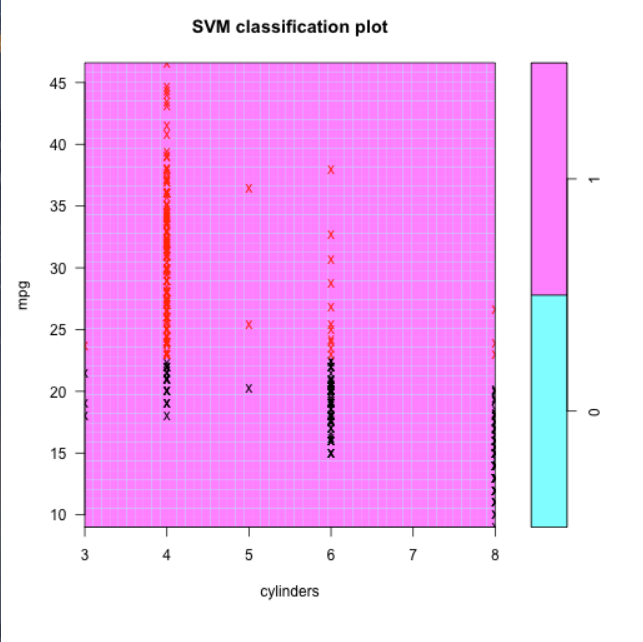


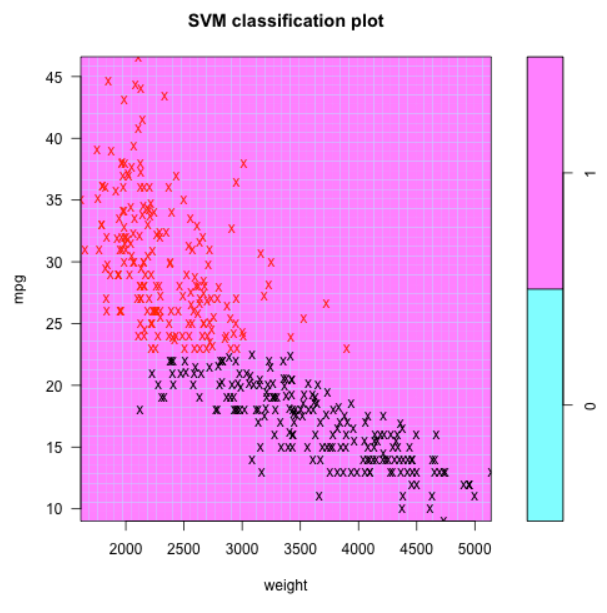
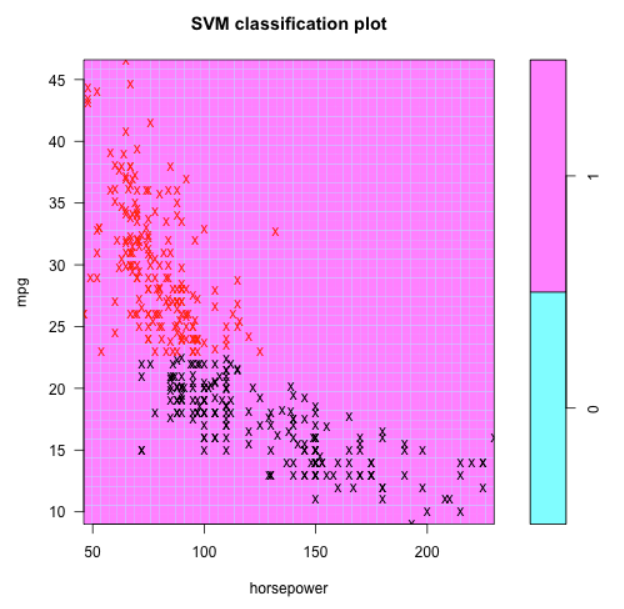


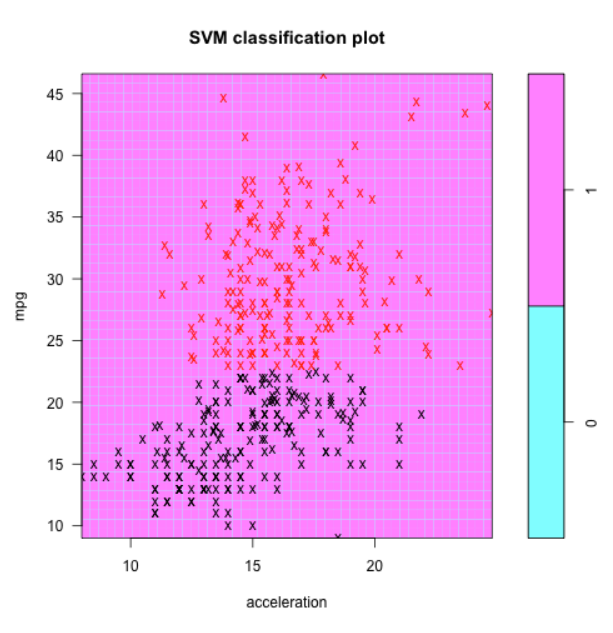


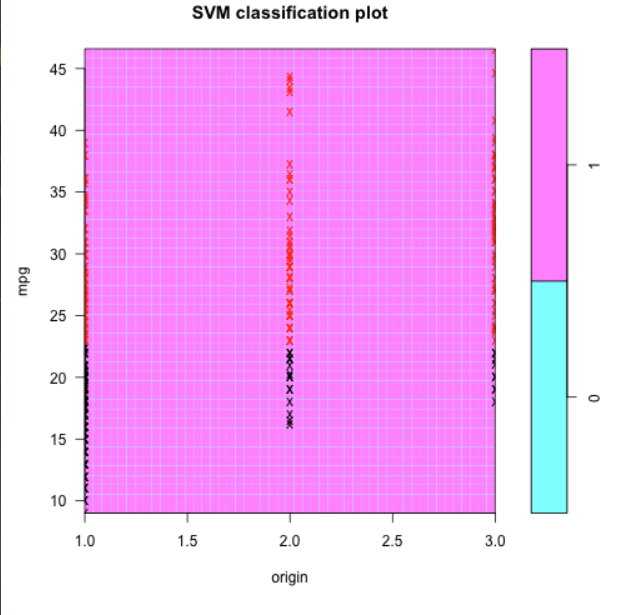


plotpairs(svm.poly)









plotpairs(svm.radial)

