Stat415-homework10

This question uses the same crabs data used in Homework 9. Use the following code to split the data into training and test sets:

```
set.seed(45678)
library(MASS)
data(crabs)
attach(crabs)
blueMale = which(sp == "B" & sex == "M")
orangeMale = which(sp == "O" & sex == "M")
blueFemale = which(sp == "B" & sex == "F")
orangeFemale = which(sp == "O" & sex == "F")
train_id = c(sample(blueMale, size = trunc(0.80 * length(blueMale))),
sample(orangeMale, size = trunc(0.80 * length(orangeMale))),
sample(blueFemale, size = trunc(0.80 * length(blueFemale))),
sample(orangeFemale, size = trunc(0.80 * length(orangeFemale))))
crabs_train = crabs[train_id, ]
crabs_test = crabs[-train_id,]
```

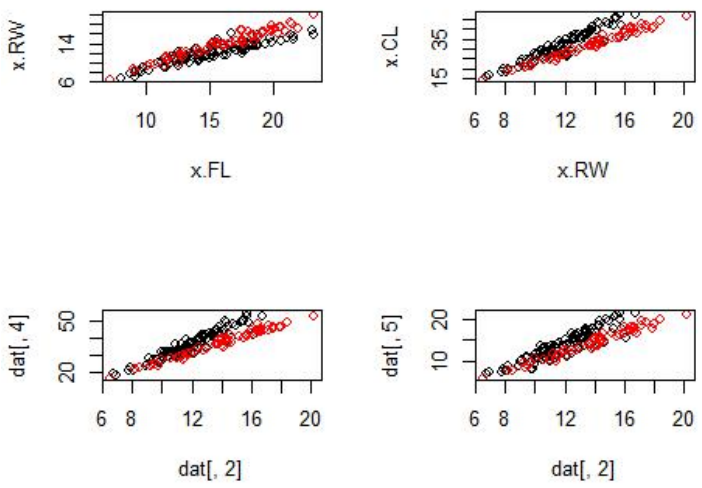
(a) Fit a linear support vector machine to the data with various values of cost, in order to predict Species from the five numerical measurements.

```
# omit the variable Sex
data_delSex = crabs[,-2]
# omit the variable index
data_delSex = data_delSex[,-2]
testdat = data_delSex[-train_id,]
traindat = data_delSex[train_id,]
# support vector machine
library(e1071)

svmfit = svm(sp ~., data = traindat, kernel = "linear", cost = 0.1, sca
le = FALSE)
summary(svmfit)
## Call:
## svm(formula = sp ~ ., data = traindat, kernel = "linear", cost = 0.1,

##     scale = FALSE)
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.2
##
## Number of Support Vectors:  29
##  ( 15 14 )
## Number of Classes:  2
## Levels:
##  B O
par(mfrow=c(2,2))
```

```
dat = data.frame(x = traindat[,-1], y = as.factor(traindat$sp))
plot(dat[,1:2],col=data_delSex$sp)
plot(dat[,2:3],col=data_delSex$sp)
plot(dat[,2],dat[,4],col=data_delSex$sp)
plot(dat[,2],dat[,5],col=data_delSex$sp)
```




```
# CV errors
set.seed(45678)
tune.out = tune(svm, sp ~ ., data = data_delSex[train_id,], kernel = "l
inear", ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)
## Parameter tuning of 'svm':
## - sampling method: 10-fold cross validation
## - best parameters:
##  cost
##     1
## - best performance: 0
## - Detailed performance results:
##    cost   error dispersion
## 1 1e-03 0.53750 0.09860133
## 2 1e-02 0.35000 0.10704360
## 3 1e-01 0.10625 0.10643366
## 4 1e+00 0.00000 0.00000000
## 5 5e+00 0.00000 0.00000000
## 6 1e+01 0.00000 0.00000000
## 7 1e+02 0.00000 0.00000000

bestmod = tune.out$best.model
summary(bestmod)
## Call:
## best.tune(method = svm, train.x = sp ~ ., data = data_delSex[train_i
d,
##     ], ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
##     kernel = "linear")
```

```
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.2
## Number of Support Vectors:  48
##  ( 24 24 )
## Number of Classes:  2
## Levels:
##  B O
```

```
ypred = predict(bestmod, testdat)
table(predict = ypred, truth = testdat$sp)
```

```
##        truth
## predict  B  O
##       B 20  0
##       O  0 20
```

Comment: According to the plot, we can notice that the linear support vector machine should be effective in this data set since the data could be divided by a line visually. Linear support vector machine with cost=0.001,0.1 and 100 has been created.Cross-validation errors associated with different values of cost have also been shown above. when cost=1e+00, the error is the smallest, zero and the corresponding gamma is 0.2. After using the best model to predict in the test data, all the predictions are correct.

(b) Fit nonlinear SVMs with radial and polynomial kernels, with different values of gamma and degree and cost.

```
set.seed(45678)
svmfit = svm(sp ~., data = traindat, kernel = "radial", ranges = list(c
ost = 1, gamma = 1))
summary(svmfit)
## Call:
## svm(formula = sp ~ ., data = traindat, kernel = "radial", ranges = l
ist(cost = 1,
##     gamma = 1))
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.2
## Number of Support Vectors:  110
##  ( 54 56 )
## Number of Classes:  2
## Levels:
##  B O
```

```
tune.out = tune(svm, sp ~ ., data = traindat, kernel = "radial", ranges
 = list(cost = c(0.1, 1, 10, 40), gamma = c(0.5, 1, 2, 3, 4)))
summary(tune.out)
## Parameter tuning of 'svm':
```

```
## - sampling method: 10-fold cross validation
## - best parameters:
##  cost gamma
##    10   0.5
## - best performance: 0.0125
##
## - Detailed performance results:
##    cost gamma   error dispersion
## 1   0.1   0.5 0.34375 0.11505584
## 2   1.0   0.5 0.03750 0.04370037
## 3  10.0   0.5 0.01250 0.02635231
## 4  40.0   0.5 0.01250 0.02635231
## 5   0.1   1.0 0.31875 0.08564502
## 6   1.0   1.0 0.05000 0.04930066
## 7  10.0   1.0 0.01875 0.04218428
## 8  40.0   1.0 0.01875 0.04218428
## 9   0.1   2.0 0.31875 0.14568445
## 10  1.0   2.0 0.04375 0.04218428
## 11 10.0   2.0 0.03125 0.04419417
## 12 40.0   2.0 0.03125 0.04419417
## 13  0.1   3.0 0.32500 0.14373490
## 14  1.0   3.0 0.04375 0.04218428
## 15 10.0   3.0 0.03750 0.04370037
## 16 40.0   3.0 0.03750 0.04370037
## 17  0.1   4.0 0.35000 0.11102427
## 18  1.0   4.0 0.05000 0.04930066
## 19 10.0   4.0 0.04375 0.05145454
## 20 40.0   4.0 0.04375 0.05145454

bestmod = tune.out$best.model
summary(bestmod)
## Call:
## best.tune(method = svm, train.x = sp ~ ., data = traindat, ranges =
list(cost = c(0.1,
##     1, 10, 40), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
## Number of Support Vectors:  38
##  ( 19 19 )
## Number of Classes:  2
## Levels:
##  B O

table(true = testdat$sp,pred = predict(bestmod, newdata = testdat))

##     pred
## true  B  O
##    B 20  0
##    O  0 20
```

Comment: The error plot with different gamma and cost value has been shown above.Non-linear support vector machine with cost=0.1, 1, 10 and 20 has been created. Cross-validation errors associated with different values of cost and gamma have also been shown above. when cost=10 and gamma=0.5, we have the best model for radial kernels since the error is the smallest, 0.01250.After using the best model to predict in the test data, all the predictions of radial kernels are correct.

```
set.seed(45678)
svmfit = svm(sp ~., data = traindat, kernel = "polynomial", ranges = li
st(cost = 1, gamma = 1, degree = 3))
summary(svmfit)
## Call:
## svm(formula = sp ~ ., data = traindat, kernel = "polynomial",
##     ranges = list(cost = 1, gamma = 1, degree = 3))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##       gamma:  0.2
##      coef.0:  0
## Number of Support Vectors:  113
##  ( 56 57 )
## Number of Classes:  2
## Levels:
##  B O

tune.out2 = tune(svm, sp ~ ., data = traindat, kernel = "polynomial", r
anges = list(cost = c(0.1, 1, 10), gamma = c(0.5, 1, 2), degree = c(1,
2, 3, 4)))
summary(tune.out2)
## Parameter tuning of 'svm':
## - sampling method: 10-fold cross validation
## - best parameters:
##  cost gamma degree
##     1   0.5      1
## - best performance: 0
## - Detailed performance results:
##    cost gamma degree   error dispersion
## 1   0.1   0.5      1 0.25625 0.11199733
## 2   1.0   0.5      1 0.00000 0.00000000
## 3  10.0   0.5      1 0.00000 0.00000000
## 4   0.1   1.0      1 0.10625 0.10643366
## 5   1.0   1.0      1 0.00000 0.00000000
## 6  10.0   1.0      1 0.00000 0.00000000
## 7   0.1   2.0      1 0.03125 0.06073908
## 8   1.0   2.0      1 0.00000 0.00000000
## 9  10.0   2.0      1 0.00000 0.00000000
## 10  0.1   0.5      2 0.50625 0.07482619
## 11  1.0   0.5      2 0.46250 0.13565684
```

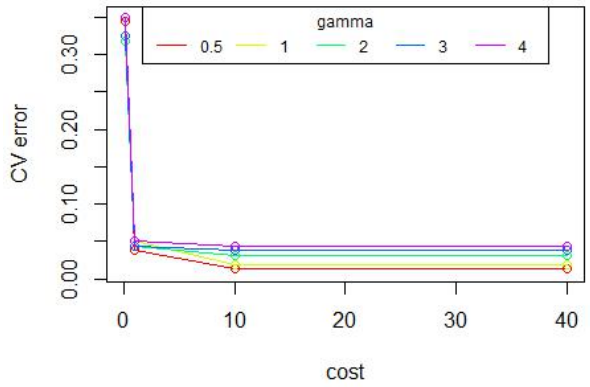
```
## 12 10.0   0.5      2 0.35000 0.12219065
## 13  0.1   1.0      2 0.48125 0.11803513
## 14  1.0   1.0      2 0.36875 0.12993722
## 15 10.0   1.0      2 0.27500 0.08936504
## 16  0.1   2.0      2 0.43125 0.10805252
## 17  1.0   2.0      2 0.32500 0.13437096
## 18 10.0   2.0      2 0.26250 0.11334559
## 19  0.1   0.5      3 0.06250 0.04166667
## 20  1.0   0.5      3 0.08750 0.07336174
## 21 10.0   0.5      3 0.01875 0.04218428
## 22  0.1   1.0      3 0.08750 0.07336174
## 23  1.0   1.0      3 0.02500 0.04370037
## 24 10.0   1.0      3 0.02500 0.04370037
## 25  0.1   2.0      3 0.03125 0.04419417
## 26  1.0   2.0      3 0.03125 0.04419417
## 27 10.0   2.0      3 0.01250 0.02635231
## 28  0.1   0.5      4 0.40625 0.10724615
## 29  1.0   0.5      4 0.37500 0.13819270
## 30 10.0   0.5      4 0.34375 0.13258252
## 31  0.1   1.0      4 0.38125 0.11580785
## 32  1.0   1.0      4 0.34375 0.14804865
## 33 10.0   1.0      4 0.31875 0.10395812
## 34  0.1   2.0      4 0.31875 0.13645436
## 35  1.0   2.0      4 0.33125 0.10643366
## 36 10.0   2.0      4 0.35625 0.08863353
```

```
bestmod2 = tune.out2$best.model
summary(bestmod2)
## Call:
## best.tune(method = svm, train.x = sp ~ ., data = traindat, ranges =
list(cost = c(0.1,
##     1, 10), gamma = c(0.5, 1, 2), degree = c(1, 2, 3, 4)), kernel =
"polynomial")
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  1
##       gamma:  0.5
##      coef.0:  0
## Number of Support Vectors:  73
##  ( 36 37 )
## Number of Classes:  2
## Levels:
##  B O
```

```
table(true = testdat$sp,pred = predict(bestmod2, newdata = testdat))
```

```
##     pred
## true  B  O
##    B 20  0
##    O  0 20
```

```
with(tune.out$performances, {
plot(error[gamma==0.5] ~ cost[gamma == 0.5], ylim = c(0.01, 0.35), type
 = "o", col = rainbow(5)[1], ylab = "CV error", xlab = "cost")
lines(error[gamma==1] ~ cost[gamma==1], type = "o", col = rainbow(5)[2])
lines(error[gamma==2] ~ cost[gamma==2], type = "o", col = rainbow(5)[3])
lines(error[gamma==3] ~ cost[gamma==3], type = "o", col = rainbow(5)[4])
lines(error[gamma==4] ~ cost[gamma==4], type = "o", col = rainbow(5)[5])
})
legend("top", horiz = T, legend = c(0.5, 1:4), col = rainbow(5), lty =
1, cex = .75, title = "gamma")
```

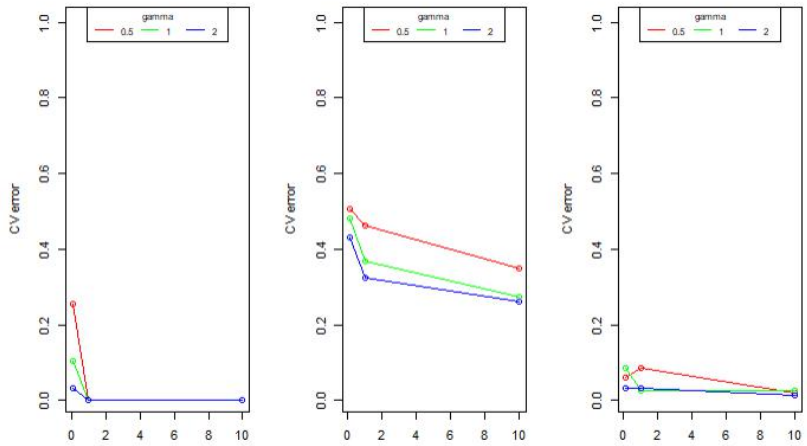



```
par(mfrow=c(1,3))
with(tune.out2$performances, {
plot(error[degree == 1][gamma == 0.5] ~ cost[degree == 1][gamma == 0.5],
 ylim = c(0.01, 1), type = "o", col = rainbow(3)[1], ylab = "CV error",
 xlab = "cost")
lines(error[degree == 1][gamma == 1] ~ cost[degree == 1][gamma == 1], t
ype = "o", col = rainbow(3)[2])
lines(error[degree == 1][gamma == 2] ~ cost[degree == 1][gamma == 2], t
ype = "o", col = rainbow(3)[3])
})
legend("top", horiz = T, legend = c(0.5, 1:2), col = rainbow(3), lty =
1, cex = .75, title = "gamma")

with(tune.out2$performances, {
plot(error[degree == 2][gamma == 0.5] ~ cost[degree == 2][gamma == 0.5],
 ylim = c(0.01, 1), type = "o", col = rainbow(3)[1], ylab = "CV error",
 xlab = "cost")
lines(error[degree == 2][gamma == 1] ~ cost[degree == 2][gamma == 1], t
ype = "o", col = rainbow(3)[2])
lines(error[degree == 2][gamma == 2] ~ cost[degree == 2][gamma == 2], t
ype = "o", col = rainbow(3)[3])
})
legend("top", horiz = T, legend = c(0.5, 1:2), col = rainbow(3), lty =
1, cex = .75, title = "gamma")

with(tune.out2$performances, {
```

```
plot(error[degree == 3][gamma == 0.5] ~ cost[degree == 3][gamma == 0.5],
 ylim = c(0.01, 1), type = "o", col = rainbow(3)[1], ylab = "CV error",
 xlab = "cost")
lines(error[degree == 3][gamma == 1] ~ cost[degree == 3][gamma == 1], t
ype = "o", col = rainbow(3)[2])
lines(error[degree == 3][gamma == 2] ~ cost[degree == 3][gamma == 2], t
ype = "o", col = rainbow(3)[3])
})
legend("top", horiz = T, legend = c(0.5, 1:2), col = rainbow(3), lty =
1, cex = .75, title = "gamma")
```



Comment: The error plot with different gamma and cost value has been shown above.Non-linear support vector machine with cost=0.1, 1, 10 has been created. Cross-validation errors associated with different values of cost and gamma and degree have also been shown above. when cost=1, degree = 1 and gamma=0.5, we have the best model for polynomial kernels. After using the best model to predict in the test data, all the predictions are correct.