old_exam2

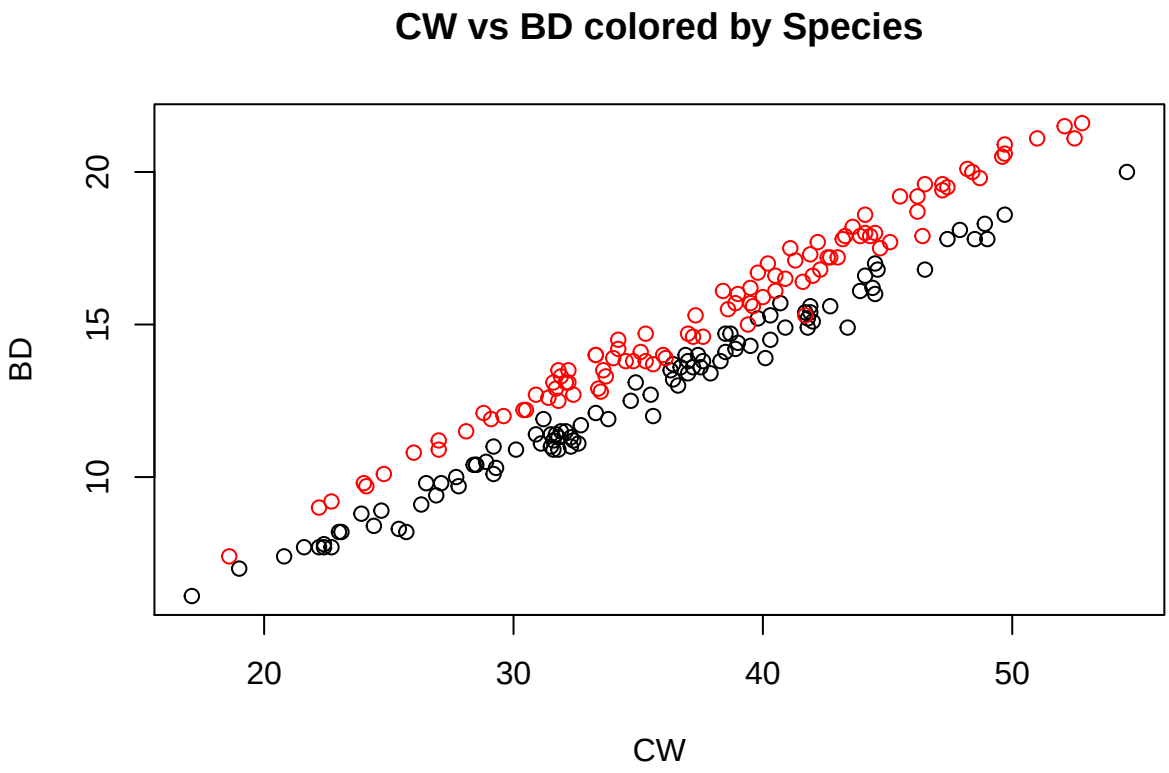
Andreas

12 Jan 2019

Contents

Assignment 1

1



As we can see from the plot there seems to be a linear boundary between CW and BD but some points are very difficult to classified properly because they are to close to each other and a linear line can't seperate them.

2

```
## ================================================
##  The confusion matrix for naive bayes model is :

##             Predicted species
```

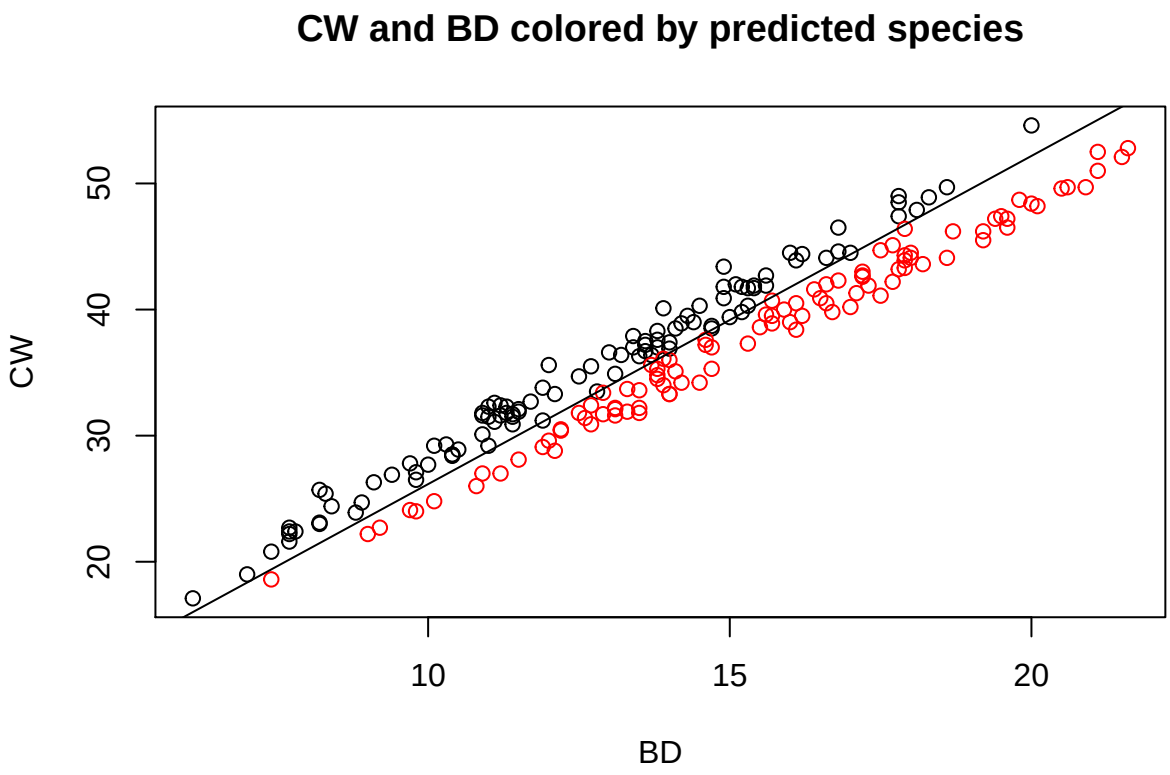
```
## True Species Blue Orange
##       Blue     62     38
##       Orange   41     59
```

```
## ========================================
##  The misclassification error is : 0.395
```

The naive Bayes model makes assumptions of conditional independence of the features $P(CW, BD|species) = P(CW|species) * P(BD|species)$ something that in our case is not valid since CW and BD are not independent and both are used to specify the class species of a crab.

3

```
## ============================================================
##  The misclassification error for logistic regression is: 0.02
```



Imagine the logistic regression line $p(y) = \frac{e(b0+b1*x1+b2*x2)}{1+\exp(b0+b1*x1+b2*x2}$) Suppose if $p(y) > 0.5$ then class-1 or else class-0

$\log(\frac{y}{1-y}) = b0 + b1 * x1 + b2 * x2$

$\log(\frac{0.5}{0.5}) = b0 + b1 * x1 + b2 * x2$

$0 = b0 + b1 * x1 + b2 * x2$

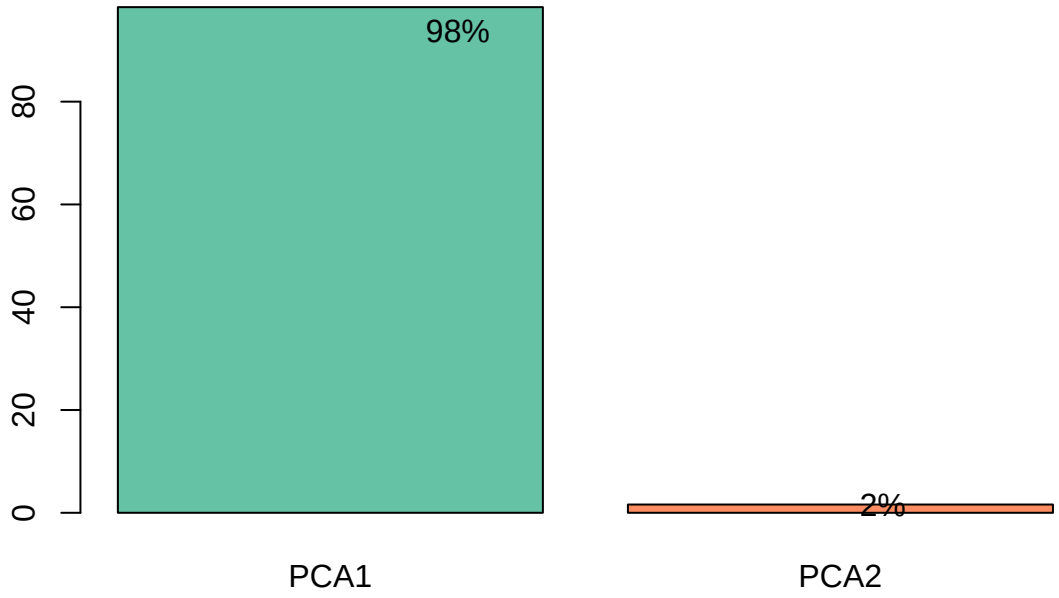
$b0 + b1 * x1 + b2 * x2 = 0$ is the line

Rewriting it in $m * x + c$ form

$X2 = (-\frac{b1}{b2}) * X1 - (\frac{b0}{b2})$

$CW = (\frac{(-0.484810)}{(-3.624760)}) + (\frac{(-9.432817)}{(-3.624760)})$ is the equation of decision boundary

4



We can see clearly that one component is dominating the variance of the data between the 2 features

Assignment 2

1

```
## (Intercept)        Time
##   0.1742155   0.4017126
```

The estimated model is: $\log(Visitors) = 0.1742155 + 0.4017126 * Time$

2

Assignment 3

1

```
## ============================================================
##  The table below summarizes the results of svm with diffrent values of C :
```

Table 1: Errors svm table

Error	C-value
0.0804171	1
0.0765055	10

Error	C-value
0.0849823	100

C is a regularization parameter that controls the trade off between the achieving a low training error and a low testing error that is the ability to generalize classifier to unseen data.Consider the objective function of a linear SVM . If your C is too large the optimization algorithm will try to reduce $|w|$ as much as possible leading to a hyperplane which tries to classify each training example correctly. Doing this will lead to loss in generalization properties of the classifier. On the other hand if your C is too small then you give your objective function a certain freedom to increase $|w|$ a lot, which will lead to large training error.So there is a bias-variance trade of as we increase the values of the C parameter.

2

```
## Warning: package 'neuralnet' was built under R version 3.5.2
```

```
## [1] 0.001368483974
```

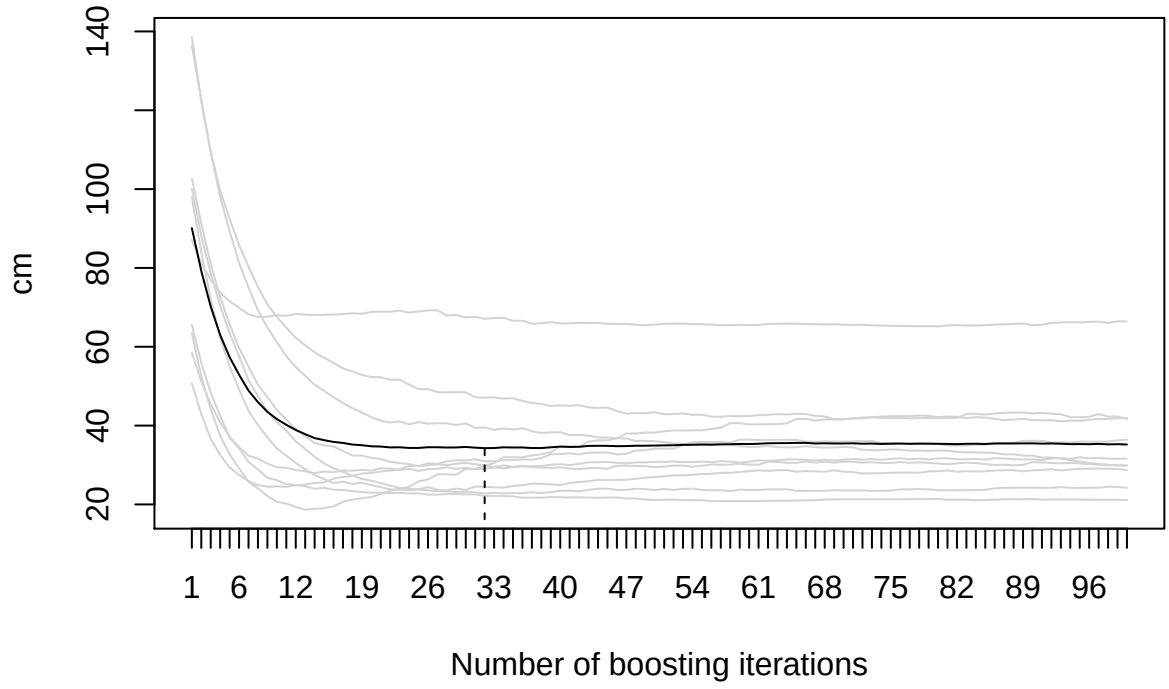
3

```
## Loading required package: parallel
```

```
## Loading required package: stabs
```

```
## This is mboost 2.9-1. See 'package?mboost' and 'news(package  = "mboost")'
## for a complete list of changes.
```

```
## [1] 100
```



```
## [1] 32
```

The above plot is showing the cross-validated squared error with respect to the number of iterations.We can see that the optimal number of iterations achieved is 32.

```
## ============================================================
##  The mean squared error for train data is :  23.06043272
##  The mean squared error for  test data is :  19.84227792
```

Appendix

```
#read csv data
crabs<-read.csv("australian-crabs.csv")
#make plot
plot(BD~CW,data=crabs,col=species,
     main="CW vs BD colored by Species")

library(e1071)
#fit naive bayes model
naive_fit<-naiveBayes(species~CW+BD,data=crabs)
#make predicitons
pr_naive<-predict(naive_fit,crabs,type="class")
#print table
cat("================================================\n",
    "The confusion matrix for naive bayes model is : \n")
table(crabs$species,pr_naive,dnn = c("True Species","Predicted species"))

#calculate misclassificaiton error
cat("========================================\n",
    "The misclassification error is :",mean(crabs$species!=pr_naive))
#fit logistic regression
logit_fit<-glm(species~CW+BD,data=crabs,family = binomial(link = "logit"))
#make predictions and use 0.5 threshold
pr_logit<-predict(logit_fit,crabs,type="response")
pr_logit_fit<-ifelse(pr_logit>0.5,"Orange","Blue")

cat("============================================================\n",
    "The misclassification error for logistic regression is:",
    mean(crabs$species!=pr_logit_fit))

cat("\n")
#calculate slope intercept for decision boundary
slope <- coef(logit_fit)[3]/(-coef(logit_fit)[2])
intercept <- coef(logit_fit)[1]/(-coef(logit_fit)[2])

plot(CW~BD,data=crabs,col=as.factor(pr_logit_fit),
     main="CW and BD colored by predicted species")
abline(intercept,slope)

library(RColorBrewer)
#calculate scaled pca for 2 components
pca_fit<-prcomp(crabs[,names(crabs)%in%c("CW","BD")],scale=T)
pca_var<-pca_fit$sdev^2
# calculate percentage of variance of PCAs
pca_var_per<-round(pca_var/sum(pca_var)*100,3)
```

```r
barplot(pca_var_per,names.arg=c("PCA1","PCA2"),col=brewer.pal(8, "Set2"))
text((pca_var_per/1.05),labels=paste0(round(pca_var_per),"%"))

bank<-read.csv2("bank.csv")
#fit poisson model
poisson_fit<-glm(Visitors~.,data=bank,family = poisson(link = "log"))
#coeficients of model
coef(poisson_fit)

library(boot)
# rng=function(data,mle) {
# data1=data.frame(Time=data$Time, Visitors=data$Visitors)
# n=length(data$Time)
# #generate new Price
# data1$Visitors=rnorm(n,predict(mle, newdata=data1,type="response"),sd(mle$residuals))
# return(data1)
# }
# f1=function(data1){
# res=glm(Visitors~.,family = poisson(link = "log"),data=data1) #fit linear model
# #predict values for all Area values from the original data
# VisitorsP=predict(res,newdata=bank,type="response")
# n=length(bank$Time)
# predictedV=rnorm(n,VisitorsP,sd(mle$residuals))
# return(predictedV)
# }
# res=boot(bank, statistic=f1, R=1000,mle=poisson_fit,
#          ran.gen=rng, sim="parametric")

library(kernlab)
library(knitr)

data("spam")
set.seed(1234567890)
#fit svm models with 2 folds cross validation(-cross=)
svm1<-ksvm(type~.,data=spam,
kernel = "rbfdot", kpar =list(sigma = 0.05),C = 1,cross=2)

svm2<-ksvm(type~.,data=spam,
kernel = "rbfdot", kpar =list(sigma = 0.05),C = 10,cross=2)

svm3<-ksvm(type~.,data=spam,
kernel = "rbfdot", kpar =list(sigma = 0.05),C = 100,cross=2)

table1<-data.frame(c(cross(svm1),cross(svm2),cross(svm3)),c("1","10","100"))
colnames(table1)<-c("Error","C-value")
cat("============================================================\n",
"The table below summarizes the results of svm with diffrent values of C : \n")
kable(table1, caption = "Errors svm table")

library(neuralnet)
set.seed(1234567890)
Var <- runif(50, 0, 10)
tr <- data.frame(Var, Sin=sin(Var))
```

```r
tr1 <- tr[1:25,] # Fold 1
tr2 <- tr[26:50,] # Fold 2
w_init<-runif(41,-1,1)
#fit nn for the first fold
nn_fit1<-neuralnet(formula =Sin~Var,
                   data = tr1,hidden = 10, startweights = w_init,
threshold = 0.001, lifesign = "none")
#make predictions for the second fold
pr_nn1<-compute(nn_fit1,tr2[,1])$net.result

#fit nn for the second fold
nn_fit2<-neuralnet(formula =Sin~Var,
                   data = tr2,hidden = 10, startweights = w_init,
threshold = 0.001, lifesign = "none")
#make predictions for the first fold
pr_nn2<-compute(nn_fit2,tr1[,1])$net.result
#calclate mean squared errors
m1<-mean((tr2[,2]-pr_nn1)^2)
m2<-mean((tr1[,2]-pr_nn2)^2)
#calculate cv error
cv<-(m1+m2)/2
cv

library(mboost)
bf <- read.csv2("bodyfatregression.csv")
set.seed(1234567890)
m <- blackboost(Bodyfat_percent~Waist_cm+Weight_kg, data=bf)
mstop(m)
cvf <- cv(model.weights(m),type="kfold")
cvm <- cvrisk(m, folds=cvf, grid=1:100)
plot(cvm)
mstop(cvm)
#split data to 2/3 train and 1/3 test
n=dim(bf)[1]
set.seed(1234567890)
id=sample(1:n, floor(n*2/3))
train_bf=bf[id,]
test_bf=bf[-id,]
#fit boost model with mstop=cvm
boost_fit<-blackboost(Bodyfat_percent~Waist_cm+Weight_kg, data=bf,
control=boost_control(mstop =mstop(cvm)))#mstop
#make predictions
pr_tr_boost<-predict(boost_fit,train_bf,type="response")
pr_tes_boost<-predict(boost_fit,test_bf,type="response")

cat("============================================================\n",
    "The mean squared error for train data is : " ,mean((train_bf$Bodyfat_percent-pr_tr_boost)^2),
    "\n The mean squared error for  test data is : ", mean((test_bf$Bodyfat_percent-pr_tes_boost)^2))
```