Introduction to Machine Learning - Lab 3

Gustav Sternelöv

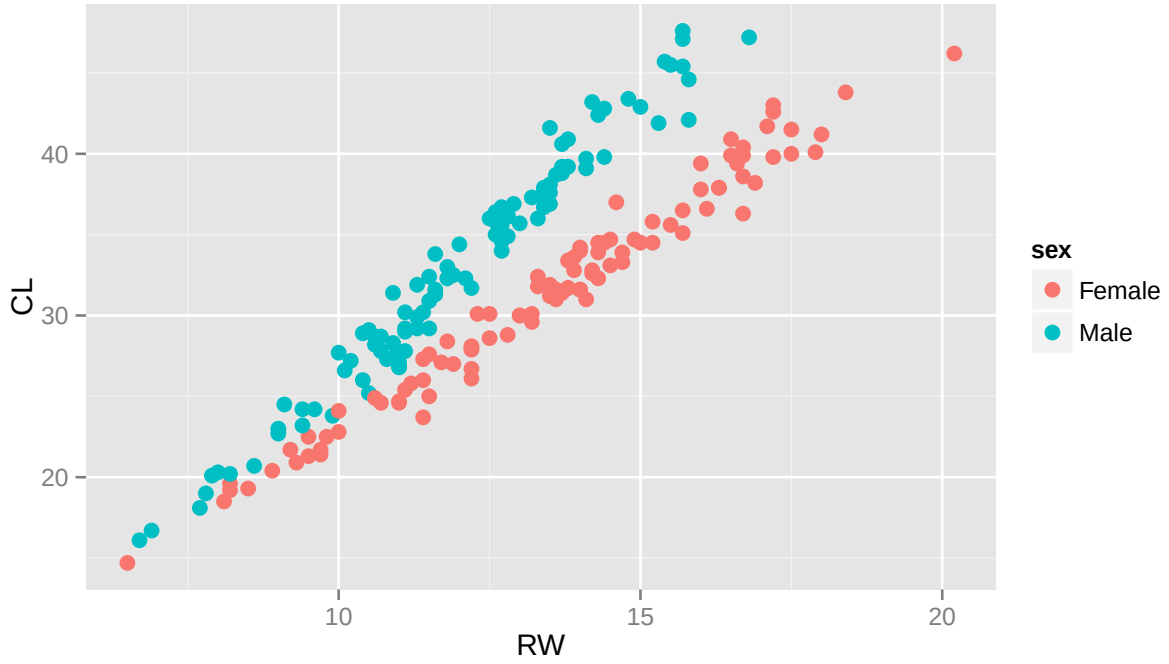
Tuesday, November 10, 2015

Assignment 1

In the first assignment a data material named australian crabs is analysed. It contanis information about measurements of the frontal lobe, rear witdh etc. for 200 crabs.

1.1

A scatterplot visualizing the carapace length versus rear width.



For values of RW higher than 12.5 and values of CL higher than 30 this data should be easy to classify by linear discriminant analysis. For lower values the obseervations lies closer and it will problaby be harder to find a linear decision boundary that classifies all those observations correctly.

1.2

The discriminant function can be written as:

$$\delta(x)_k = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k = x^T w_i - w_{0k}$$

And the respective discriminant function for males and females looks as follows:

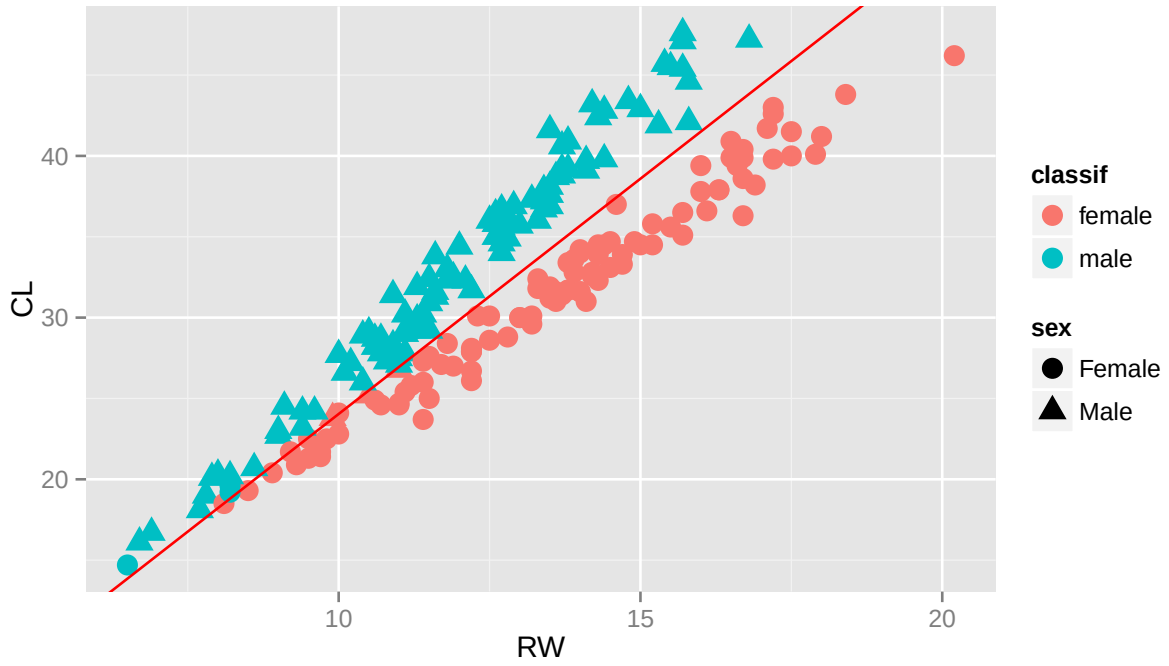
$$\delta(x)_{male} = x^T 0.029410201\begin{pmatrix} 11.990 \\ 32.851 \end{pmatrix} - \frac{1}{2}(11.990 \quad 32.851)0.029410201\begin{pmatrix} 11.990 \\ 32.851 \end{pmatrix} + log(0.50)$$

$$\delta(x)_{female} = x^T 0.029410201 \begin{pmatrix} 13.487 \\ 31.360 \end{pmatrix} - \frac{1}{2}(13.487 \quad 31.360) 0.029410201 \begin{pmatrix} 13.487 \\ 31.360 \end{pmatrix} + log(0.50)$$

And the decision boundary is given by the this expression : CL = 2.91RW - 5.06

1.3

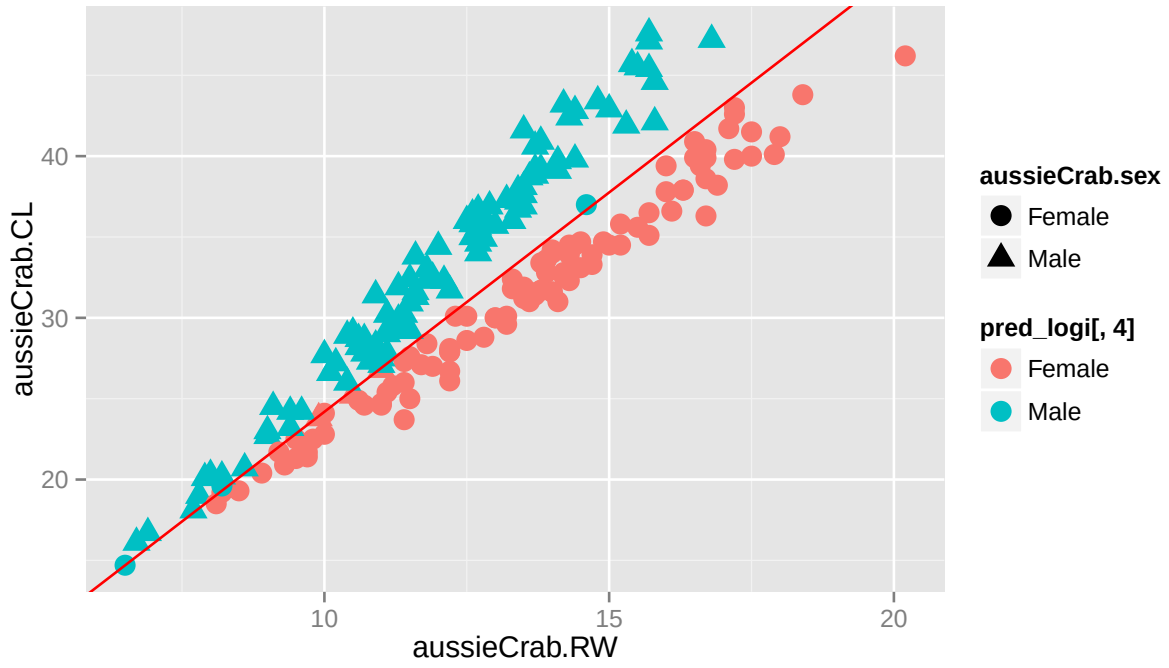
In the following plot the observed values of RW and CL are visualized. Colours are given after the classification label and shape after the true class for the observations. The red line gives the decision boundary.



The fit is considered to be pretty god. Only 7 of 200 observations is classified wrongly, and as expected most of these observations are find at the bottom left of the plot.

1.4

Another method to classify the observations are by a logistic regression model. In the graph below the classifications are plotted in the same way as in 1.3.



The decision boundary for the logistic regression model: CL = 2.713RW - 2.94
The obtained result with logistic regression is very similar to the result given with the *lda* model. For both models seven observations has been misclassified, and even though the decision boundaries are different it is first and foremost at the bottom left of the graph the misclassified observations can be found. A comparsion of the lines for the two models decision boundaries gives that the line for the logistic model is slightly shifted because .

Assignment 2

For the second assignment the analyzed data contains information about how well costumers have managed their loans. The costumers are divided into two classes, good or bad borrowers, and there is 19 different potential features that can be used for classifying.

2.1 - 2.2

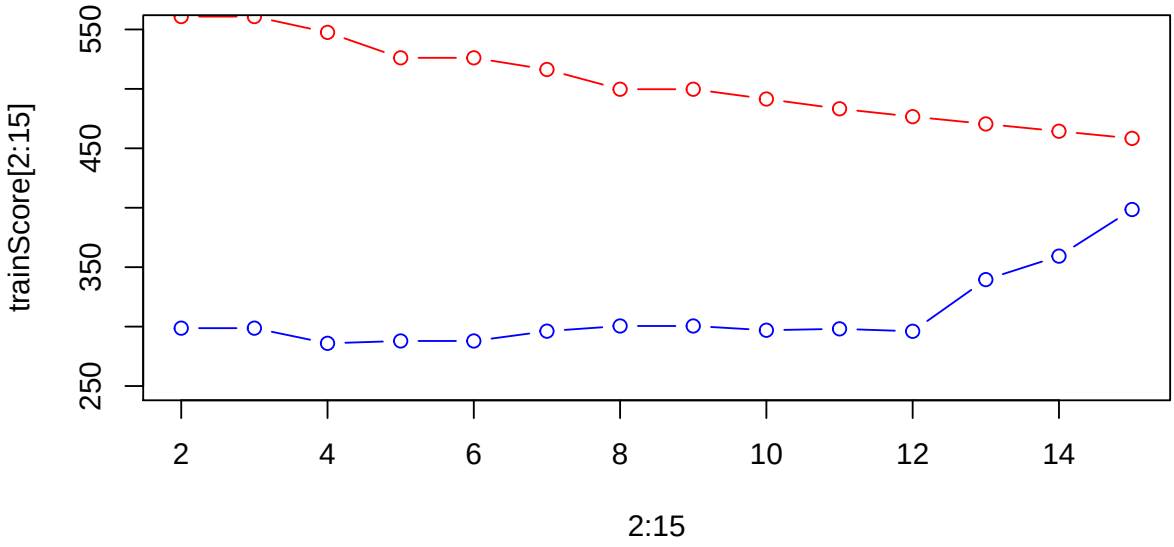
To start with, data is splitted into a training, validation and test data set.

Decision tree is the first classification method used to classify the costumers. When a decision tree is fitted one of the choices is which impurity measure to use. Here two different models, one with deviance and one with gini as measure, is compared.

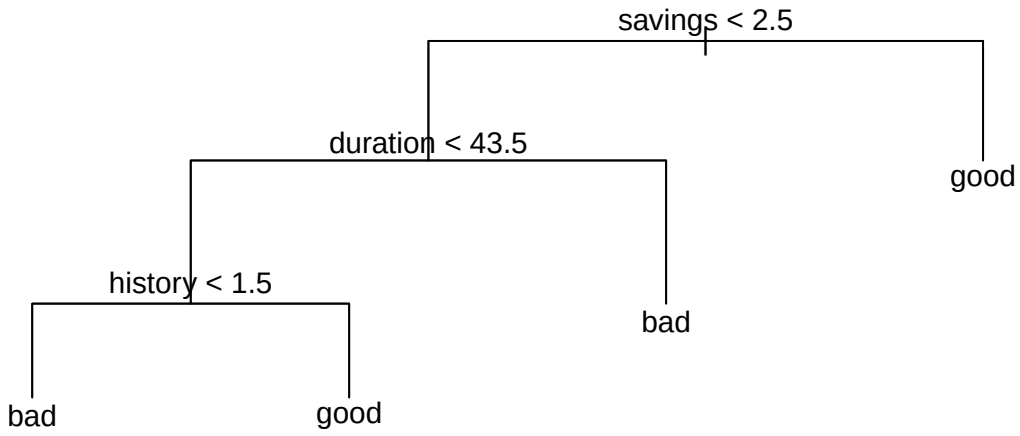
For training data with deviance as measure of impurity the misclassification rate is 0.2105263 and for test data the misclassification rate is 0.268. With gini as impurity measure the rate is 0.2368421 for training data and 0.368 for test data. Since the misclassification rate is lower in the first case deviance is the chosen measure for the upcoming steps.

2.3

The chosen tree is pruned by looking on the dependence of deviances for the training and validation set. This is shown by the plot below where values for training data are visualized by the red line and values for the validation set by the blue line.



When the number of leaves is equal to four the lowest score is obtained. This implies that the optimal number of leaves is four.



The tree presented above has a depth of 3 and uses 3 variables.
We see from the tree that the most important factor is that a borrower should have a high amount of savings. Further a loan of shorter duration are better credit risks than loans of longer duration and a good history is preferable.

The misclassification rate for test data is 0.256.

2.4

In 2.4 the classification of the costumers is performed with the Naive Bayes classifier. The first of the tables below gives the confusion matrix for training data and the second for test data. For training data the misclassification rate is 0.3 and for test data it is 0.316. Compared to the results in 2.3 these rates are higher.

```
##       bayesPredTrain
##        bad good
##   bad   95   52
##   good  98  255
```

```
##       bayesPredTest
##        bad good
##   bad   46   30
##   good  49  125
```

2.5

Naive Bayes classification is again used, but now with a loss function defiend as following:

$$L = \begin{matrix} \\ good \\ bad \end{matrix} \begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}$$

With this function the confusion matrices seen below is obtained for train and test data.

```
##       preds
##        bad good
##   bad  137   10
##   good 263   90
```

```
##       preds
##        bad good
##   bad   71    5
##   good 122   52
```

The misclassification rate for training data is0.546 and for test data 0.508. This implies that the Naive Bayes classification under the assumption of the loss function stated above is worse in terms of misclassification rate than the classifier used in 2.4. The loss function changes the decision boundary for the classification and thereby also the probability for different types of errors. This can be seen in the results for test data were the amount of costumers that is wrongly classified as good is low and the amount of costumers wrongly classified as bad is much higher now.

Appendix

R-code

```
aussieCrab <- read.csv("australian-crabs.csv", sep=",", header = TRUE)

library(ggplot2)
ggplot(aussieCrab, aes(y=CL, x=RW)) + geom_point(aes(color=sex), size=3)
```

```r
# 1.2

male_data <- subset(aussieCrab, sex=="Male")[5:6]
female_data <- subset(aussieCrab, sex=="Female")[5:6]

male_vec <- c(mean(male_data[,1]), mean(male_data[,2]))
female_vec <- c(mean(female_data[,1]), mean(female_data[,2]))

male_cov <- cov(male_data)
female_cov <- cov(female_data)

covHat <- matrix(100/200 * male_cov + 100/200 * female_cov, ncol=2)

# prop priors
prior_male <- 100/200
prior_female <- 100/200

W_male <- (as.matrix(aussieCrab[, 5:6]))%*% solve(covHat) %*% as.matrix(male_vec)
Wo_male <-  (0.5 *  t(as.matrix(male_vec)) %*% solve(covHat) %*% as.matrix(male_vec) +
  log(prior_male))
disc_male <- W_male - as.vector(Wo_male)

W_female <- (as.matrix(aussieCrab[, 5:6]))%*% solve(covHat)%*% as.matrix(female_vec)
Wo_female <- (0.5 *  t(as.matrix(female_vec)) %*% solve(covHat) %*% as.matrix(female_vec) +
   log(prior_female))
disc_female <-W_female - as.vector(Wo_female)

classif <- 0
for(i in 1:200){
  if(disc_male[i] > disc_female[i]){
    classif[i] = "male"
  }else{
    classif[i] = "female"
  }
}

#table(classif, aussieCrab$sex)
lda_class <- data.frame(aussieCrab, classif)

ggplot(lda_class, aes(y=CL, x=RW)) +
  geom_point(aes(color=classif,shape=sex), size=4) +
  geom_abline(intercept = -5.06, slope=2.91, colour="red")

logi_class <- glm(sex~RW+CL,data=aussieCrab, family=binomial())
#summary(logi_class)
#exp(coef(logi_class))
pred_logi <- data.frame(aussieCrab$RW,aussieCrab$CL,aussieCrab$sex , predict(logi_class, type="response

for (i in 1:length(pred_logi[,4])){
if(pred_logi[,4][i] <0.5){
  pred_logi[,4][i] = 0
}else{
  pred_logi[,4][i] = 1
```

```
} }

for (i in 1:length(pred_logi[,4])){
  if(pred_logi[,4][i] == 0){
    pred_logi[,4][i] = "Female"
  }else{
    pred_logi[,4][i] = "Male"
  } }

ggplot(pred_logi, aes(y=aussieCrab.CL, x=aussieCrab.RW)) +
  geom_point(aes(color=pred_logi[,4],shape=aussieCrab.sex), size=4) +
  geom_abline(intercept = -2.94, slope=2.713, colour="red")


#table(pred_logi[,4], pred_logi$aussieCrab.sex)
creditScore <- read.csv("creditscoring.csv", sep=";", header = TRUE)

# create train set
n=dim(creditScore)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=creditScore[id,]
test=creditScore[-id,]

# create valid and test
n=dim(test)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
valid=test[id,]
test=test[-id,]
library(tree)
# a) - deviance
# train
deviFit <- tree(good_bad~., data=train, split="deviance")
#plot(deviFit)
#text(deviFit, pretty=0)
#deviFit
devi_summary <- summary(deviFit)
deviPred <- predict(deviFit, newdata=test, type="class")
deviTable <- table(test$good_bad, deviPred)

# b) gini index
# train
giniFit <- tree(good_bad~., data=train, split="gini")
gini_summary <- summary(giniFit)
giniPred <- predict(giniFit, newdata=test, type="class")
giniTable <- table(test$good_bad, giniPred)

# Lower misclassification for deviance
trainScore <- rep(0,15)
testScore <- rep(0,15)

for (i in 2:15){
```

```
  prunedTree <- prune.tree(deviFit, best=i)
  pred <- predict(prunedTree, newdata=valid,
                  type="tree")
  trainScore[i] <- deviance(prunedTree)
  testScore[i] <- deviance(pred)
}

plot(2:15, trainScore[2:15], type="b", col="red", ylim=c(250,550))
points(2:15, testScore[2:15], type="b", col="blue")
finalTree <- prune.tree(deviFit, best=4)
# The variables used
plot(finalTree)
text(finalTree, pretty=0)
# Model tested on test data
NewFit <- predict(finalTree, newdata=test, type="class")
NewFitTable <- table(test$good_bad, NewFit)
library(e1071)
# bayes classifier based on training data
bayesFit <- naiveBayes(good_bad~., data=train)
# Fitted valuse for training and test based on classifier
bayesPredTrain <- predict(bayesFit, newdata=train)
bayesPredTest <- predict(bayesFit, newdata=test)
# confusion matrices for train and test
bayesTrain <-table(train$good_bad, bayesPredTrain)
bayesTest <- table(test$good_bad, bayesPredTest)
bayesTrain
bayesTest
# Repeating 2.4 but with a defined loss matrix.
raws <- data.frame(predict(bayesFit, newdata=train, type="raw"))

preds <- 0
for (i in 1:500){
  if((raws[i,1]/raws[i,2]) > 0.1){
    preds[i] = "bad"
  }else{
    preds[i] ="good"
  }
}

loss_train <- table(train$good_bad, preds)
loss_train

rawTest <- predict(bayesFit, newdata=test, type="raw")
preds <- 0
for (i in 1:250){
  if((rawTest[i,1]/rawTest[i,2]) > 0.1){
    preds[i] = "bad"
  }else{
    preds[i] ="good"
  }
}

loss_test <- table(test$good_bad, preds)
```

```
loss_test
##
```