Lab 1

Rolf Sievert (rolsi701)

2018-12-10

Contents

Appendix

Assignment 1

```
# Search for function: RSiteSearch("expression")
library("kknn")
# 1.1

# Set working directory
setwd("~/courses/tdde01/lab1/")

# Read data
data = read.csv("spambase.csv")

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

# 1.2
# Create a General Linear Model
model = glm(formula = Spam~., data = train, family = 'binomial')
```

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

```
# Make prediction on test
predict_test = predict(model, test, 'response')
# Make prediction of train
predict_train = predict(model, train, 'response')
# Show result, match actual answers to predicted ones
table_test_0.5 = table(test$Spam, predict_test > 0.5)
table_train_0.5 = table(train$Spam, predict_train > 0.5)
cat("\nPrediction with GML model, test data. 0.5")
```

```
##
## Prediction with GML model, test data. 0.5
```

```
print(table_test_0.5)
```

```
##
```

```
##     FALSE TRUE
##   0   791  146
##   1    97  336
```

```
cat("Missclassification rate: ")
```

```
## Missclassification rate:
```

```
cat((sum(table_test_0.5[2]) + sum(table_test_0.5[3]))/sum(table_test_0.5[1:4]))
```

```
## 0.1773723
```

```
cat("\n\n");
```

```
cat("Prediction with GML model, train data. 0.5")
```

```
## Prediction with GML model, train data. 0.5
```

```
print(table_train_0.5)
```

```
##
##     FALSE TRUE
##   0   803  142
##   1    81  344
```

```
cat("Missclassification rate:")
```

```
## Missclassification rate:
```

```
cat((sum(table_train_0.5[2]) + sum(table_train_0.5[3]))/sum(table_train_0.5[1:4]))
```

```
## 0.1627737
```

```
cat("\n\n");
```

```
# 1.3
table_test_0.9 = table(test$Spam, predict_test > 0.9)
table_train_0.9 = table(train$Spam, predict_train > 0.9)
cat("Prediction with GML model, test data. 0.9")
```

```
## Prediction with GML model, test data. 0.9
```

```
print(table_test_0.9)
```

```
##
##     FALSE TRUE
##   0   936    1
##   1   427    6
```

```
cat("Missclassification rate:")
```

```
## Missclassification rate:
```

```
cat((sum(table_test_0.9[2]) + sum(table_test_0.9[3]))/sum(table_test_0.9[1:4]))
```

```
## 0.3124088
```

```
cat("\n\n");
```

```
cat("Prediction with GML model, train data. 0.9")
```

```
## Prediction with GML model, train data. 0.9
```

```r
print(table_train_0.9)
```

```
##
##     FALSE TRUE
##   0   944    1
##   1   419    6
```

```r
cat("Missclassification rate:")
```

```
## Missclassification rate:
```

```r
cat((sum(table_train_0.9[2]) + sum(table_train_0.9[3]))/sum(table_train_0.9[1:4]))
```

```
## 0.3065693
```

```r
cat("\n\n");
```

```r
# 1.4
K = 30
# kknn is classifier
kknn_test = kknn(formula = Spam~., train = train, test = test, k = K)
kknn_train = kknn(formula = Spam~., train = train, test = train, k = K)
table_kknn_test = table(test$Spam, kknn_test$fitted.values > 0.5)
table_kknn_train = table(train$Spam, kknn_train$fitted.values > 0.5)
cat("KKNN result using test data test for K=30")
```

```
## KKNN result using test data test for K=30
```

```r
print(table_kknn_test)
```

```
##
##     FALSE TRUE
##   0   672  265
##   1   187  246
```

```r
cat("Missclassification rate:")
```

```
## Missclassification rate:
```

```r
cat((sum(table_kknn_test[2]) + sum(table_kknn_test[3]))/sum(table_kknn_test[1:4]))
```

```
## 0.329927
```

```r
cat("\n\n");
```

```r
cat("KKNN result using test data train for K=30")
```

```
## KKNN result using test data train for K=30
```

```r
print(table_kknn_train)
```

```
##
##     FALSE TRUE
##   0   807  138
##   1    98  327
```

```r
cat("Missclassification rate:")
```

```
## Missclassification rate:
```

```r
cat((sum(table_kknn_train[2]) + sum(table_kknn_train[3]))/sum(table_kknn_train[1:4]))
```

```
## 0.1722628
```

```
cat("\n\n");
```

```
# 1.5
K = 1
# kknn is classifier
kknn_test = kknn(formula = Spam~., train = train, test = test, k = K)
kknn_train = kknn(formula = Spam~., train = train, test = train, k = K)
table_kknn_test = table(test$Spam, kknn_test$fitted.values > 0.5)
table_kknn_train = table(train$Spam, kknn_train$fitted.values > 0.5)
cat("KKNN result using test data test for K=1")
```

```
## KKNN result using test data test for K=1
```

```
print(table_kknn_test)
```

```
##
##     FALSE TRUE
##   0   640  297
##   1   177  256
```

```
cat("Missclassification rate:")
```

```
## Missclassification rate:
```

```
cat((sum(table_kknn_test[2]) + sum(table_kknn_test[3]))/sum(table_kknn_test[1:4]))
```

```
## 0.3459854
```

```
cat("\n\n");
```

```
cat("KKNN result using test data train for K=1")
```

```
## KKNN result using test data train for K=1
```

```
print(table_kknn_train)
```

```
##
##     FALSE TRUE
##   0   945    0
##   1     0  425
```

```
cat("Missclassification rate:")
```

```
## Missclassification rate:
```

```
cat((sum(table_kknn_train[2]) + sum(table_kknn_train[3]))/sum(table_kknn_train[1:4]))
```

```
## 0
```

```
cat("\n\n");
```

Assignment 2

```
# 2.1
# Set working directory
setwd("~/courses/tdde01")
# Read data
data = read.csv("machines.csv")
```

```r
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]

# 2.2 Exponential distribution
# PxTheta
pxt = function(x, theta){
    # Prod returns product of all arguments
    return (log(prod(theta*exp(-theta*x))))
}
thetas = seq(from=0, to=2, by=0.005)
theta_fn = sapply(thetas, function(theta) pxt(data$Length, theta))
#par(mar=c(1,1,1,1))
plot(thetas, theta_fn, xlabel="X", ylabel="Y", type="p")
theta_max = thetas[which.max(theta_fn)]
cat("Max theta value: ")
cat(theta_max)
cat("\n\n")

# 2.3
short_data = data$Length[1:6]
short_theta_fn = sapply(thetas, function(theta) pxt(short_data, theta))
par(mar=c(4,4,2,1))
plot(thetas, theta_fn, xlab="Theta", ylab="", type="l", ylim=c(-100, 0))
lines(thetas, short_theta_fn, col="blue")
short_theta_max = thetas[which.max(short_theta_fn)]
cat("Max theta value of short dataset: ")
cat(short_theta_max)
cat("\n\n")

# 2.4
pt = function(theta){
   return (10*exp(-10*theta))
}
pxt = function(x, theta) {
    return (prod(theta*exp(-theta*x)))
}
l = function(x, theta){
    return (log(pxt(x, theta)*pt(theta)))
}
theta_fn = sapply(thetas, function(theta) l(data$Length, theta))
l_max = thetas[which.max(theta_fn)]
lines(thetas, theta_fn, col="red")
cat("Max theta value: ")
cat(l_max)
cat("\n\n")

# 2.5
new_data = rexp(50, theta_max)
hist(new_data, main="Random values, exponential distribution", xlab="")
hist(data$Length, main="Original data", xlab="")
```

Assignment 4

```
# Assignment 4
# Set working directory
setwd("~/courses/tdde01")

# Read data
data = read.csv("tecator.csv")

# 4.1 No?
plot(data$Protein, data$Moisture, xlab="Protein", ylab="Moisture", type = "p")

# 4.2 TODO:

# 4.3
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
eval=data[-id,]

mse_eval_list = c()
mse_train_list = c()
for (x in c(1, 2, 3, 4, 5, 6)){
    model=lm(formula = Moisture ~ poly(Protein, x, raw = TRUE), data = train)
    # Eval data
    mse = mean((eval$Moisture - predict(model, eval))^2)
    mse_eval_list=c(mse_eval_list, mse)

    # Train data2
    mse = mean((train$Moisture - predict(model, train))^2)
    mse_train_list=c(mse_train_list, mse)

}
counts <- table(mse_eval_list, mse_train_list)
barplot(c(mse_eval_list, mse_train_list),
        main="Prediction on evaluation and training data",
        ylab="MSE",
        xlab="Blue: eval, Red: train",
        col=c("blue","blue","blue","blue","blue","blue","red", "red", "red", "red", "red", "red"),
        ylim=c(32, 35))
stop()

# 4.4
channels = data[,2:102]
model = lm(formula = Fat~. , data = channels)
step_alg = stepAIC(model, direction="both", trace = FALSE)
cat("Length: ", length(coef(step_alg)) -1, "\nCoefficients: ", coef(step_alg))

# 4.5
covariates=channels[,1:100]
response=channels[, 101]
model0=glmnet(as.matrix(covariates),
              response,
```

```
              alpha=0,
              family="gaussian")
plot(model0, xvar="lambda", label=TRUE)
# The coefficients shrink when lambda increases

# 4.6
# LASSO, alpha = 1
model1=glmnet(as.matrix(covariates),
              response,
              alpha=1,
              family="gaussian")
plot(model1, xvar="lambda", label=TRUE)

# 4.7
model=cv.glmnet(as.matrix(covariates),
                response,
                lambda = seq(0, 5, 0.005),
                alpha=1,
                family="gaussian")
cat("\n\n\nMin lambda: ", model$lambda.min)
plot(model)
cat("\nNumber of coefficients of lambda min: ")
coeffs = coef(model, s=model$lambda.min)
coeffs
cat(sum (coeffs[2:101,] != 0))

# 4.8 Compare 4 and 7
```