Advanced Methods Recreation

This is a recreation of the methods found in the “Practical Data Science with R” book.

##Random Forests Method

Load data and create sets

spamD <- read.table('G:/Shared drives/Sale/Escuela/Elmhurst\_MSDS/MDS\_556\_analytical\_methods/spamD.tsv',header = T, sep = '\t')  
spamTrain <- subset(spamD, spamD$rgroup>=10)  
spamTest <- subset(spamD, spamD$rgroup<10)  
spamVars <- setdiff(colnames(spamD), list('rgoup','spam'))

Prep the model and train it

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

set.seed(105030)  
  
fmodel <- randomForest(x=spamTrain[, spamVars],  
 y = spamTrain$spam,  
 ntree = 100,  
 nodesize = 7,  
 importance = T  
 )

Setup accuracy measures

loglikelihood <- function(y, py) {  
 pysmooth <- ifelse(py==0, 1e-12,  
 ifelse(py==1, 1-1e-12, py))  
 sum(y \* log(pysmooth) + (1-y)\*log(1 - pysmooth))  
}  
  
accuracyMeasures <- function(pred, truth, name="model") {  
 dev.norm <- -2\*loglikelihood(as.numeric(truth), pred)/length(pred)  
 ctable <- table(truth=truth,  
 pred=(pred>0.5))  
 accuracy <- sum(diag(ctable))/sum(ctable)  
 precision <- ctable[2,2]/sum(ctable[,2])  
 recall <- ctable[2,2]/sum(ctable[2,])  
 f1 <- precision\*recall  
 data.frame(model=name, accuracy=accuracy, f1=f1, dev.norm)  
}

Test model accuracy

accuracyMeasures(predict(fmodel,  
 newdata = spamTrain[, spamVars],  
 type = 'prob')[, 'spam'],  
 spamTrain$spam=="spam",  
 name = "random forest, train")

## model accuracy f1 dev.norm  
## 1 random forest, train 0.9884142 0.9706524 0.1428859

accuracyMeasures(predict(fmodel,  
 newdata = spamTest[, spamVars],  
 type = 'prob')[, 'spam'],  
 spamTest$spam=="spam",  
 name = "random forest, train")

## model accuracy f1 dev.norm  
## 1 random forest, train 0.9563319 0.8897059 0.432223

See which variables are important

varImp <- importance(fmodel)  
varImp[1:10, ]

## non-spam spam MeanDecreaseAccuracy  
## word.freq.make 0.5187278 4.401854 4.298073  
## word.freq.address 5.2547222 3.572609 6.425185  
## word.freq.all 1.9653493 5.398803 5.299335  
## word.freq.3d 2.4638791 -1.332236 1.169645  
## word.freq.our 9.1950096 9.382239 11.493243  
## word.freq.over 5.2496441 4.727619 7.378664  
## word.freq.remove 16.3939471 17.651596 18.594709  
## word.freq.internet 7.0166696 5.246922 8.064138  
## word.freq.order 4.0057762 3.787778 5.678220  
## word.freq.mail 3.0041896 3.377981 3.961599  
## MeanDecreaseGini  
## word.freq.make 8.975427  
## word.freq.address 6.898615  
## word.freq.all 14.996153  
## word.freq.3d 1.329323  
## word.freq.our 49.249174  
## word.freq.over 9.575504  
## word.freq.remove 159.661983  
## word.freq.internet 22.348713  
## word.freq.order 6.618738  
## word.freq.mail 15.098377

##Gam methods

Prepare the example data

set.seed(405060)  
x <- rnorm(1000)  
noise <- rnorm(1000, sd = 1.5)  
y <- 3\*sin(2\*x) + cos(0.75\*x) - 1.5\*(x^2) + noise  
select <- runif(1000)  
frame <- data.frame(y=y, x = x )  
train <- frame[select > 0.1,]  
test <- frame[select <= 0.1,]

Apply GAM model

library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.

glin.model <- gam(y~s(x), data = train)  
glin.model$converged

## [1] TRUE

Summarize and view model

summary(glin.model)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## y ~ s(x)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.75367 0.04768 -15.81 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(x) 8.555 8.946 454.2 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.817 Deviance explained = 81.8%  
## GCV = 2.0999 Scale est. = 2.078 n = 914

resid.glin <- train$y-predict(glin.model)  
sqrt(mean(resid.glin^2))

## [1] 1.433962