10장 랜덤 포리스트

10. 1 서론
앙상블 모형은 여러 개의 분류모형에 의한 결과를 종합하여 분류의 정확도를 높이는 방법이다.
이는 적절한 표본 추출법으로 데이터에서 여러 개의 훈련용 데이터 셋을 만들어
각각의 데이터 셋에서 하나의 분류기를 만들어 앙상블하는 방법이다.
즉, 새로운 자료에 대해 분류기 예측값들의 가중투표를 통해 분류를 수행한다.
데이터를 조절하는 가장 대표적인 방법에는 배깅과 부스팅이 있다.
랜덤포리스트 방법은 배깅의 개념과 속성의 임의 선택을 결합한 앙상블 기법이다.

앙상블 방법은 개별 모형에 비해 다음의 장점을 지닌다.
평균을 취함으로써 편의를 제거해준다. : 치우침이 있는 여러 모형의 평균을 취하면 어느 쪽에도 치우치지 않는 평균을 얻게 된다.
분산을 감소시킨다. : 한 개 모형으로부터의 단일 의견보다 여러 모형의 의견을 결합하면 변동이 작아진다.
과적합의 가능성을 줄여준다. : 과적합이 없는 각 모형으로부터 예측을 결합하면 과적합의 여지가 줄어든다.

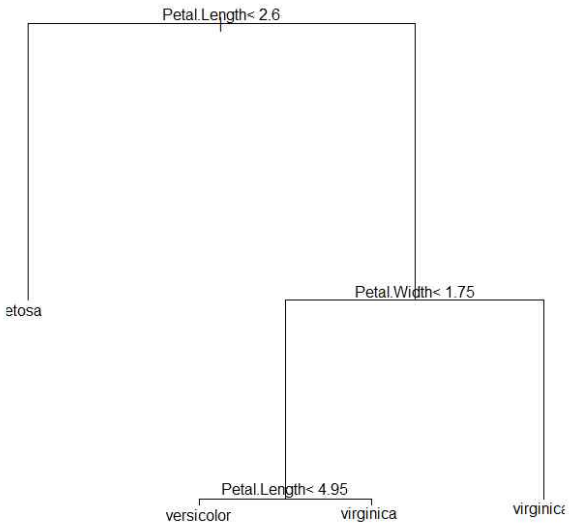
10.2 배깅
배깅은 Boostrap aggregation의 준말로 원 데이터 셋으로부터 크기가 같은 표본을 여러 번 단순임의복원추출하여
각 표본(이를 붓스트랩 표본이라고 함)에 대해 분류기를 생성한 후 그 결과를 앙상블하는 방법이다.
반복추출방법을 사용하기 때문에 같은 데이터가 한 표본에 여러 번 추출될 수도 있고, 어떤 데이터는 추출되지 않을 수도 있다.

예제 1: {adabag}bagging()함수를 이용하여 분류를 수행
데이터 불러오기기
```
data(iris)
```

{adabag}bagging()함수를 이용하여 분류를 수행
mfinal : 반복수 또는 트리수로 디폴트는 100이다. 100이면 너무 많으니 10으로 설정
```
install.packages("adabag")
library(adabag)
iris.bagging <- bagging(Species~., data=iris, mfinal=10)
```

배깅 분류에서 상대적인 중요도를 확인
각 트리변수에서 주어지는 '지니지수의 이익'(불확실성의 감소량을 의미)을 고려한 측도이다.
Petal.Length가 69.47177의 비중을 가지면서 가장 중요하다고 할 수 있다.
```
iris.bagging$importance
# 출력 Petal.Length   Petal.Width Sepal.Length   Sepal.Width
#         69.47177       30.52823        0.00000        0.00000
```

plot()함수를 이용하여 트리 형태로 출력을 진행한다.
```
plot(iris.bagging$trees[[10]]) # 트리 구조 출력
text(iris.bagging$trees[[10]]) # 트리의 텍스트 출력
```



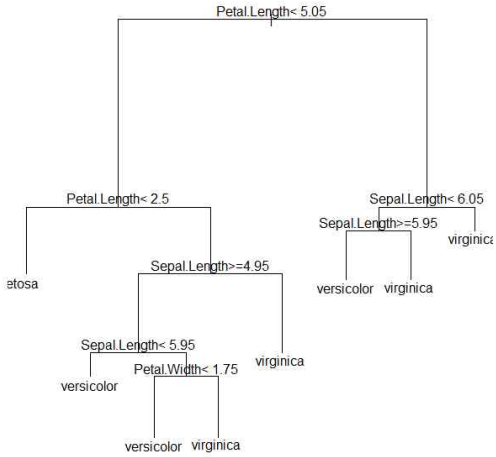
predict()하수를 이용하여 새로운 자료에 대한 예측(분류)를 수행
setosa는 50개 모두, versicolor는 47개, virginica는 49개가 제대로 분류되었다.
```
pred <- predict(iris.bagging, newdata=iris)
table(pred$class, iris[,5])
# 출력 :          setosa versicolor virginica (가로 : 실제, 세로 : 예측)
# setosa          50          0         0
# versicolor       0         47         1
# virginica        0          3        49
```

충북히 큰 자료에서 각 자료(행)가 표본에 포함되지 않을 확률
충북히 큰 자료에서, 원 자료와 동일한 크기의 부스트랩 표본을 취할 때,
특정 자료(행)가 표본에 포함되지 않을 확률은 약 36.8%이다.

10.3 부스팅
부스팅은 배깅의 과정과 유사하나 붓스트랩 표본을 구성하는 재표본 과정에서 각 자료에
동일한 확률을 부여하는 것이 아니라, 분류가 잘못된 데이터에 더 큰 가중을 주어 표본을 추출한다.
부스팅에서는 붓스트랩 표본을 추출하여 분류기를 만든 후, 그 분류결과를 이용하여 각 데이터가 추출될 확률을 조정한 후,
다음 붓스트랩 표본을 추출하는 과정을 반복한다.
각 분류기의 중요도를 계산하고(정분류율이 높을 수록 큼), 이를 가중한 결과로 분류를 수행한다.
아다 부스팅은 가장 많이 사용되는 부스팅 알고리즘이다.

예제 2 : {adabag}boosting()함수를 이용하여 분류를 수행한다.
데이터 불러오기
```
data(iris)
```

{adabag}boosting()함수를 이용하여 분류를 수행
부스팅을 사용하기 위해 boos 옵션을 T로 사용한다.
mfinal : 반복수 또는 트리수로 디폴트는 100이다. 100이면 너무 많으니 10으로 설정
```
library(adabag)
boo.adabag <- boosting(Species~., data=iris, boos=TRUE, mfinal=10)
```



부스팅 분류에서 상대적인 중요도를 확인
각 트리변수에서 주어지는 '지니지수의 이익'(불확실성의 감소량을 의미)을 고려한 측도이다.
Petal.Length가 69.192728 비중을 가지면서 가장 중요하다고 할 수 있다.
```
boo.adabag$importance
# 출력 Petal.Length   Petal.Width Sepal.Length   Sepal.Width
#        69.192728      15.343083       5.862456      9.601734
```

plot() 함수를 이용하여 트리 형태로 나타낼 수 있다.
```
plot(boo.adabag$trees[[10]]) # 트리 구조 출력
text(boo.adabag$trees[[10]]) # 트리 텍스트 출력
```

predict()함수를 이용하여 새로운 자료에 대한 예측(분류)를 수행
모형 구축에 사용된 자료를 재사용하여 분류를 수행하였다. setosa, versicolor, virginica 모두 50개 다 제대로 분류되었다.
```
pred <- predict(boo.adabag, newdata=iris)
tb <- table(pred$class, iris[,5])
tb
# 출력                setosa versicolor virginica
#      setosa         50          0         0
#      versicolor      0         50         0
#      virginica       0          0        50
```

오분류율
오분류율 = 1 - 정분류율율 = 1 - 1 = 0
```
error.rpart <- 1-(sum(diag(tb))/sum(tb)) # 1 - 150/150 = 0
error.rpart
```

예제 3 : {ada}ada()함수를 이용하여 아다부스팅을 이용한 분류를 수행한다.
데이터 불러오기
# setosa가 아닌 versicolor와 cirginica 자료만 사용하여 분석을 수행
# 60개의 훈련용 자료와 40개의 검증용 자료를 만들어서 사용
data(iris)
iris <- iris[iris$Species != "setosa",] # setosa 50
n <- dim(iris)[1] # 자료의 총 개수 저장
trind <- sample(1:n, floor(.6*n), FALSE) # (100 * 0.6 = 60) 60개의 데이터 저장
teind <- setdiff(1:n, trind) # set difference( ) # 40개의 데이터 저장
iris[,5] <- as.factor((levels(iris[, 5])[2:3])[as.numeric(iris[,5])-1]) # 실제 분류 데이터를 추가

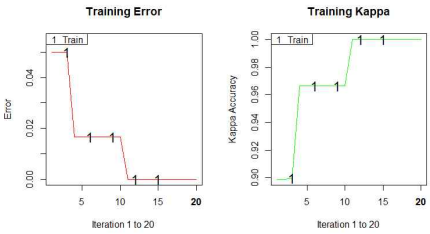
{ada}ada()함수를 이용하여 아다부스팅을 이용한 분류를 수행
# ada() 함수의 옵션
# nu= 부스팅을 위한 축소모수로 디폴트는 1이다.
# type= 부스팅 알고리즘 지정, "discrete"(디폴트), "real" , "gentle" 부스팅 지정 가능
install.packages("ada")
library(ada)
gdis<-ada(Species~., data=iris[trind,], iter=20, nu=1, type="discrete")
gdis
결과
Final Confusion Matrix for Data:
Final Prediction
True value   versicolor virginica
versicolor   28   0
virginica   0   32
#
Train Error: 0 , 오분류율 = 0, 정분류율 = 1
#
Out-Of-Bag Error: 0 iteration= 18
Additional Estimates of number of iterations:
#
train.err1 train.kap1
11   11

{ada}addtest()함수를 사용하여 검증용 자료 test데이터에 대한 분류(예측)을 실시
# addtest의 옵션
# (분류 결과, 뉴데이터의 예측변수(5번째를 뺀 데이터), 뉴데이터의 반응변수(5번째만 선택한 데이터))
gdis<-addtest(gdis, iris[teind, -5], iris[teind, 5])
gdis
출력
Final Confusion Matrix for Data:
Final Prediction
True value   versicolor virginica(가로 : 실제, 세로 : 예측)
versicolor   28   0
virginica   0   32
#
Train Error: 0 , 오분류율 = 0, 정분류율 = 1
#
Out-Of-Bag Error: 0 iteration= 12
#
Additional Estimates of number of iterations:
train.err1 train.kap1 test.errs2 test.kaps2
6   6   1   16

plot(), varplot(), pairs() 함수를 이용하여 부스팅 결과를 시각화 한 결과

plot() 함수는 오차와 일치도를 나타내는 카파계수를 그려준다. 두 True옵션은 훈련용, 검증용 자료 모두에 대해 그림을 그려준다.

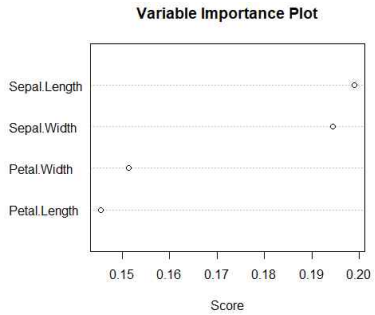
plot(gdis, TRUE, TRUE)



varplot() 함수는 변수의 중요도를 나타내는 그림 제공

Sepel.Length 변수가 분류에 가장 중요한 변수로 사용되었음을 보여준다.

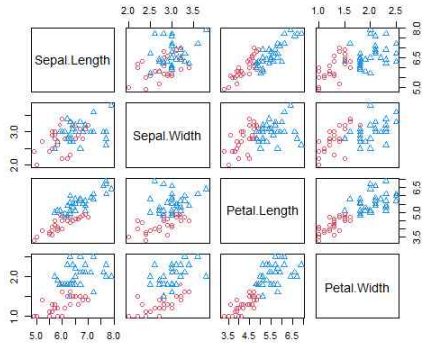
varplot(gdis)



pairs() 함수는 두 예측 변수의 조합별로 분류된 결과를 그려준다.

maxvar= 옵션을 통해 변수의 수(중요도가 높은 상위 변수의 수)를 지정할 수 있다.

pairs(gdis, iris[trind,-5], maxvar=4)

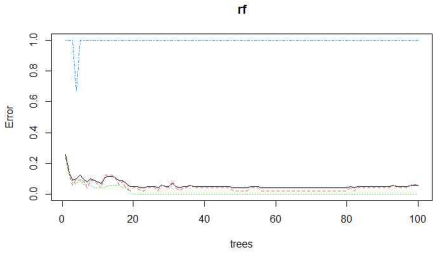


10.4 랜덤포리스트
랜덤 포리스트는 배깅에 랜덤 과정을 추가한 방법이다.
원 자료로부터 붓스트랩 샘플을 추출하고, 각 붓스트랩 샘플에 대해 트리를 형성해 나가는 과정은 배깅과 유사하나,
각 노드마다 모든 예측변수 안에서 최적의 분할을 선택하는 방법 대신
예측변수들을 임의로 추출하고, 추출된 변수 내에서 최적의 분할을 만들어 나가는 방법을 사용한다.
새로운 자료에 대한 예측을 할 때 '분류의 경우는 다수결'로 '회귀의 경우에는 평균'을 취하는 방법을 사용하며
이는 다른 앙상블 모형에서도 동일하다.

예제 4 : {randomForest}randomForest()함수를 이용하여 분석을 진행
전립선 암 환자 자료, ploidy 자료를 사용
# 데이터 불러오기
data(stagec)
# 데이터 구조 확인
str(stagec)
# 결측값 제거
stagec1<- subset(stagec, !is.na(g2))
stagec2<- subset(stagec1, !is.na(gleason))
stagec3<- subset(stagec2, !is.na(eet))
# 결측값을 제거한 데이터 구조 확인
str(stagec3)
# 시드를 설정하여 훈련용 자료와 검증용 자료를 7:3으로 나누는 과정
set.seed(1234)
ind <- sample(2, nrow(stagec3), replace=TRUE, prob=c(0.7, 0.3)); ind
trainData <- stagec3[ind==1,] # 훈련용 데이터
testData <- stagec3[ind==2,] # 검증용 데이터
str(trainData)
str(testData)

{randomForest}randomForest()함수를 이용하여 분석을 진행
# ploidy는 상동염색체수이고, 예측변수는 7개 이다.
library(randomForest)
rf <- randomForest(ploidy ~ ., data=trainData, ntree=100, proximity=TRUE) # proximity=TRUE : matirix
# 따로 테스트 만들지 않고 OOB과정에서 빠진 데이터를 가진고 분류(예측)을 진행
table(predict(rf), trainData$ploidy)
# 결과   diploid tetraploid aneuploid
# diploid   45   0   3
# tetraploid   1   51   0
# aneuploid   2   0   0
rf # 정오분류표와 함께 오류율에 대한 OOB추정치를 제공한다.
결과
randomForest(formula = ploidy ~ ., data = trainData, ntree = 100,   proximity = TRUE)
Type of random forest: classification
Number of trees: 100
No. of variables tried at each split: 2
OOB estimate of error rate: 5.88%
Confusion matrix:
diploid tetraploid aneuploid class.error
diploid   45   1   2   0.0625
tetraploid   0   51   0   0.0000
aneuploid   3   0   0   1.0000

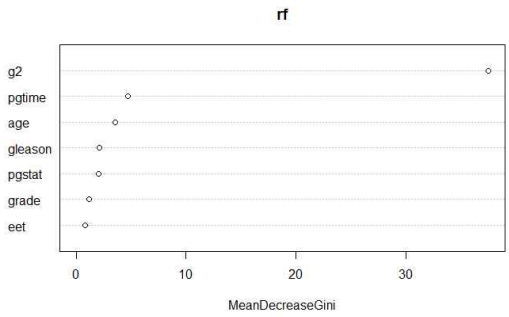
새로운 데이터에 대한 예측
# 랜덤포리스트에서는 별도의 검증용 데이터를 사용하지 않더라도 붓스트랩 샘플과정에서 제외된 out-of-bag 자료를 사용하여 검증을 실시
# plot() 함수는 트리 수에 따른 종속변수의 범주별 오분류율을 나타낸다.
# 검정색은 전체 오분류를 나타낸다.
# 오분류율이 1로(전부다 오분류 됨)나타난 범주는 aneuploid 범주로 개체수가 매우 작은 (n=3) 범주에서 발생된 결과이다.
plot(rf)



importance()함수와 varImpPlot()함수를 이용하여 변수의 중요성을 알 수 있다.
# importance() 함수 : g2가 가장 중요하고, pgtime가 그다음으로 중요한 요인이다.
importance(rf)

```
# 결과 MeanDecreaseGini
# pgtime          4.6800225
# pgstat          2.0635061
# age             3.5726107
# eet             0.7875501
# g2             37.5032896
# grade           1.2084410
# gleason         2.0820408
```

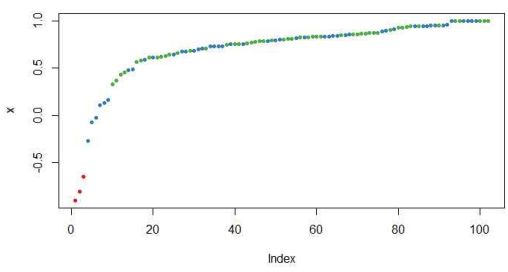
# varImpPlot()함수 : 해당 변수로부터 분할이 일어날 때 불순도의 감소가 얼마나 일어나는지를 나타내는 값이다.
# 지니 지수(gini index)는 노드의 불순도를 나타내는 값이다. 회귀의 경우에는 잔차제곱합을 통해 측정된다.
varImpPlot(rf)



앞에서 진행했던 것과 같이 predict()함수를 이용하여 뉴데이터를 지정하여 분류(예측)을 진행하는 방법
rf.pred <- predict(rf, newdata=testData)
table(rf.pred, testData$ploidy)

```
# 결과
# rf.pred       diploid tetraploid aneuploid
# diploid         17          0         1
# tetraploid       0         13         1
# aneuploid        0          0         0
```

plot() 함수를 이용하여 훈련용 자룟값(총 102개)의 마진을 나타낸다.
# 마진(margin)은 랜덤 포리스트의 분류기 가운데 정분류를 수행한 비율에서 다른 클래스로 분류한 비율의 최댓값을 뺀 값을 나타낸다.
# 즉, 양의 마진은 정확한 분류를 의미하며, 음은 그 반대이다.
plot(margin(rf))



예제 5 : {caret}를 이용하여 랜덤포리스트를 수행한다.

```
require(caret)
require(ggplot2)
require(randomForest)
# 트레이닝 셋 만들기
training_URL<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
# 테스트 셋셋
test_URL<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
# 결측값 대체
training<-read.csv(training_URL, na.strings=c("NA","") )
test<-read.csv(test_URL, na.strings=c("NA","") )
# 훈련용 셋과 검증용 셋 데이터 구조 확인하기
str(training)
str(test)
# 불필요한 열들을 제외한다.
training<-training[,7:160]
test<-test[,7:160]
# 결측값을 가지는 열은 일단 제외하고 분석에 임한다.
mostly_data <- apply(!is.na(training), 2, sum)>19621 # be careful!
training <- training[,mostly_data]
test <- test[,mostly_data]
# 현재 훈련용 데이터 수
dim(training)
# 모형 수립의 수행 속도를 높이기 위해 편의상 훈련용 자료를 더 작게 나눈다.
# 원 자료의 30%를 임의로 추출하여 새로운 훈련용 자료를 만든다.
InTrain<-createDataPartition(y=training$classe, p=0.3, list=FALSE)
training1<-training[InTrain,]

## {caret}train함수를 이용하여 램덤포리스트를 수행하되, 5중첩 교차타당도 방법을 적용한다.
rf_model<-train(classe~., data=training1, method="rf", trControl=trainControl(method="cv", number=5), prox=TRUE, allowParallel=TRUE)
print(rf_model)
# 결과 : Random Forest
#
# 5889 samples
# 53 predictor
# 5 classes: 'A', 'B', 'C', 'D', 'E'
#
# No pre-processing
# Resampling: Cross-Validated (5 fold)
# Summary of sample sizes: 4710, 4711, 4711, 4712, 4712
# Resampling results across tuning parameters:
#
#   mtry  Accuracy   Kappa
# 2     0.9809804  0.9759364
# 27    0.9876041  0.9843159
# 53    0.9867562  0.9832443
#
# Accuracy was used to select the optimal model using the largest value.
# The final value used for the model was mtry = 27.
# 오류율에 대한 OOB 추정치값으로 0.85%가 나왔으며, 이를 통해 정확도가 99.15%라는 것을 알 수 있다.
print(rf_model$finalModel)
# 결과 : Call:
# randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE,       allowParallel = TRUE)
# Type of random forest: classification
# Number of trees: 500
# No. of variables tried at each split: 27
#
# OOB estimate of  error rate: 0.76%
# Confusion matrix:
#    A    B    C    D    E class.error
# A 1674    0    0    0    0 0.000000000
# B    7 1123    9    1    0 0.014912281
# C    0   10 1015    2    0 0.011684518
# D    0    0    8 957    0 0.008290155
# E    0    0    0    8 1075 0.007386888
```

OOB(out of bag) 오차추정
랜덤 포리스트에서는 검증 자료의 오차에 대한 불편 추정치를 얻기 위해
교차 타당법을 사용하거나 별도의 검증용 자료를 만들 필요가 없다.
이 방법은 오차에 대한 추정을 내부 알고리즘에서 자동으로 제공해준다.
그방법은 다음과 같다.
각 트리는 원 자료로부터 서로 다른 붓스트랩 표본을 사용하여 구축된다. 자료의 약 1/3은 붓스트랩 표본에서 제외되고,
K-번째 트리의 형성에 사용되지 않는다.
k-번째 트리의 형성에 제외된 (out of bag, 이하 OOB) 자료를 구축된 k-번째 트리에 적용하여 분류를 수행한다.
이러한 방식을 각 트리에 대해 적용하면, n번째 자료가 OOB인 모든 트리에서, n번째로 가장 많은 표를 획득한 클래스로 분류한다.
n개의 모든 자료에 대해 그 자료가 OOB인 트리에서 오분류된 비율의 평균이 OOB 오차 추정치 이다.
이 값은 많은 검정에서 불현성을 만족하는 것으로 증명되었다.