앙상블방법(Ensemble)
bagging, boosting, Random Forest

Jinseog Kim
Dongguk University
jskim1986@gmail.com

2018-03-20

Contents

1 Ensemble model

 Combine many base models(or weak learners) $\rightarrow$ strong model

$$f(x) = \sum_{i=1}^{M} w_i f_i(x) \tag{1}$$

1. Bagging: generate weak models (decision trees) from random samples and agregate them simply
2. Boosting: sequentially generate weak models from previous residuals & combime them with diferent weights
3. Random Forest: inject more randomness compared with Bagging

1.1 Ensemble R 패키지

R패키지	function	설명
ipred	bagging()	Bagging with rpart tree
adabag	bagging()	Bagging (rpart)
	boosting()	AdaBoost.M1
ada	Culp et al.	AdaBoost + Friedman's mods
randomForest	Breiman et al.	Random Forest

R패키지	function	설명
gbm	Ridgeway et al.	Stochastic Gradient Boosting
party	Hothorn	RF with faster tree growing
mboost	Hothorn	Boosting appl to glm, gam

2 배깅 (bagging)

- fitting

```
library(ipred); data(spam, package = "ElemStatLearn")
# sampling index of train data
tr.idx <- sample(nrow(spam), 0.7*nrow(spam))
# fit bagging model with stump(2-nodes trees)
m.bag <- bagging(spam~., data=spam[tr.idx, ], nbagg=50, control=list(maxdepth=1))
names(m.bag)
```

```
## [1] "y"      "X"      "mtrees" "OOB"    "comb"   "call"
```

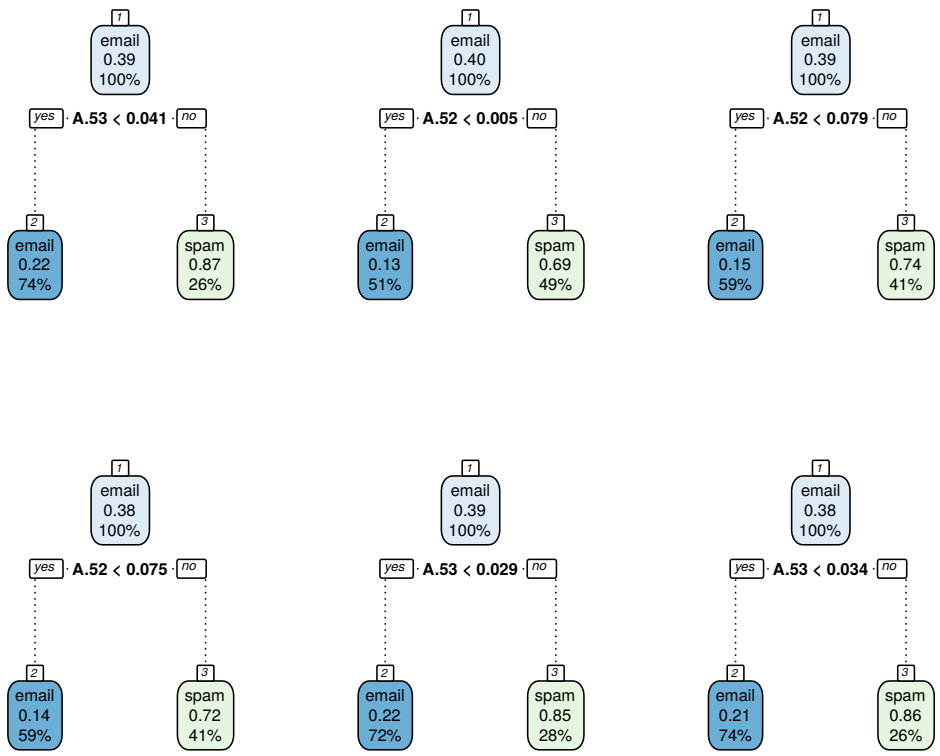
```
m.bag$mtrees[[1]]$btree
```

```
## n= 3220
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 3220 1264 email (0.6074534 0.3925466)
```

```
##   2) A.53< 0.0405 2370  527 email (0.7776371 0.2223629) *
##   3) A.53>=0.0405 850  113 spam (0.1329412 0.8670588) *
```

- plot bagging trees

```
library(rpart.plot); par(mfrow=c(2,3))
for(i in 1:6){
  rpart.plot(m.bag$mtrees[[i]]$btree, branch.lty=3, nn=TRUE)
}
```



- prediction & performance

```
pred <- predict(m.bag, newdata=spam[-tr.idx, -58], type="class") #prob
kable(table(pred, spam$spam[-tr.idx]))
```

	email	spam
email	761	252

	email	spam
spam	48	320

3 boosting

- fit: Freund & Shapire (1996)

```
library(adabag); #require(rpart)
# fit boosting model with stump(2-nodes trees)
m.boo <- boosting(spam~., data=spam[tr.idx, ], mfinal=50, control=list(maxdepth=2))
names(m.boo)
```

```
## [1] "formula"    "trees"      "weights"    "votes"      "prob"
## [6] "class"      "importance" "terms"      "call"
```

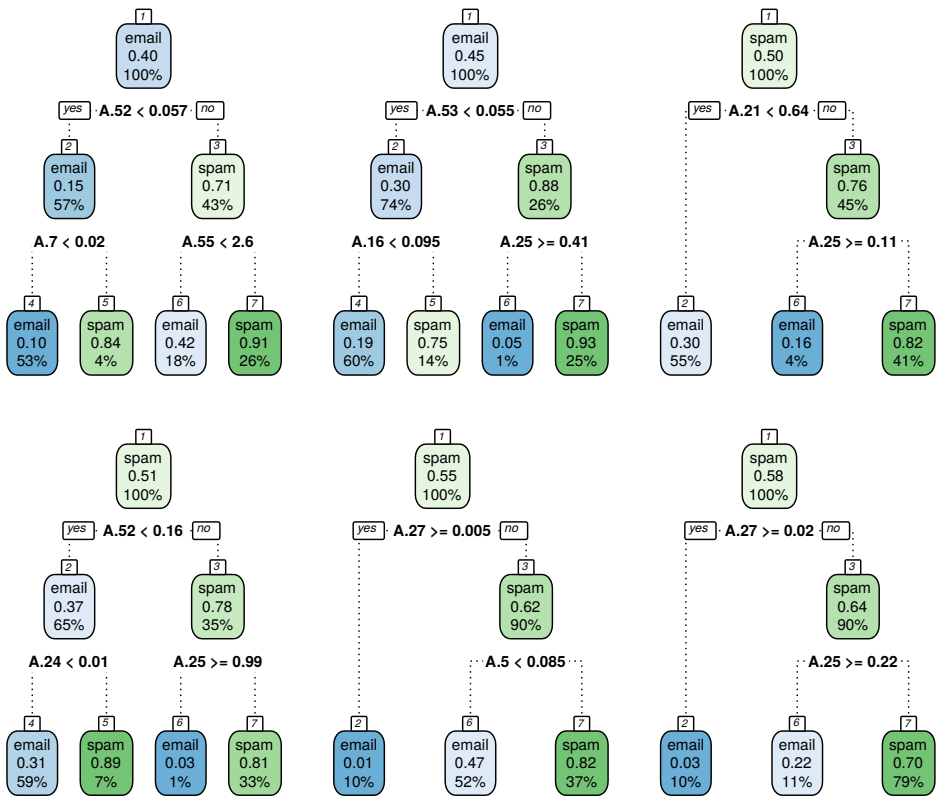
```
m.boo$trees[[1]]
```

```
## n= 3220
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 3220 1272 email (0.60496894 0.39503106)
##   2) A.52< 0.0565 1827  278 email (0.84783799 0.15216201)
```

```
##     4) A.7< 0.02 1692  164 email (0.90307329 0.09692671) *
##     5) A.7>=0.02 135   21 spam (0.15555556 0.84444444) *
##   3) A.52>=0.0565 1393  399 spam (0.28643216 0.71356784)
##     6) A.55< 2.609 565  240 email (0.57522124 0.42477876) *
##     7) A.55>=2.609 828   74 spam (0.08937198 0.91062802) *
```

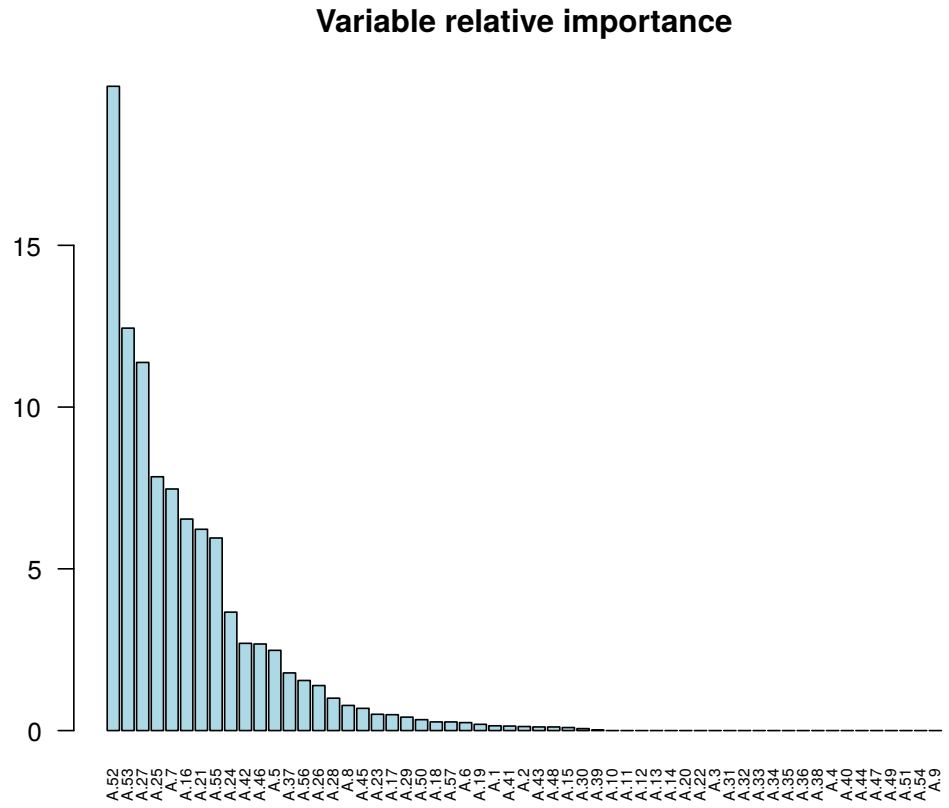
- plot boosting trees

```
library(rpart.plot); par(mfrow=c(2,3))
for(i in 1:6){
  rpart.plot(m.boo$trees[[i]], branch.lty=3, nn=TRUE)
}
```



- Variables Importance plot
  - Gain of the Gini index of the variables

```
importanceplot(m.boo, las=2, cex.names=.6)
```



- boosting: prediction & performance

```
pred.boo <- predict(m.boo, newdata=spam[-tr.idx, -58], type="class")$class #prob
kable(table(pred, spam$spam[-tr.idx]))
```

	email	spam
email	761	252
spam	48	320

```
kable(table(pred.boo, spam$spam[-tr.idx]))
```

	email	spam
email	769	68
spam	40	504

4 Random Forest (Breiman, 2001)

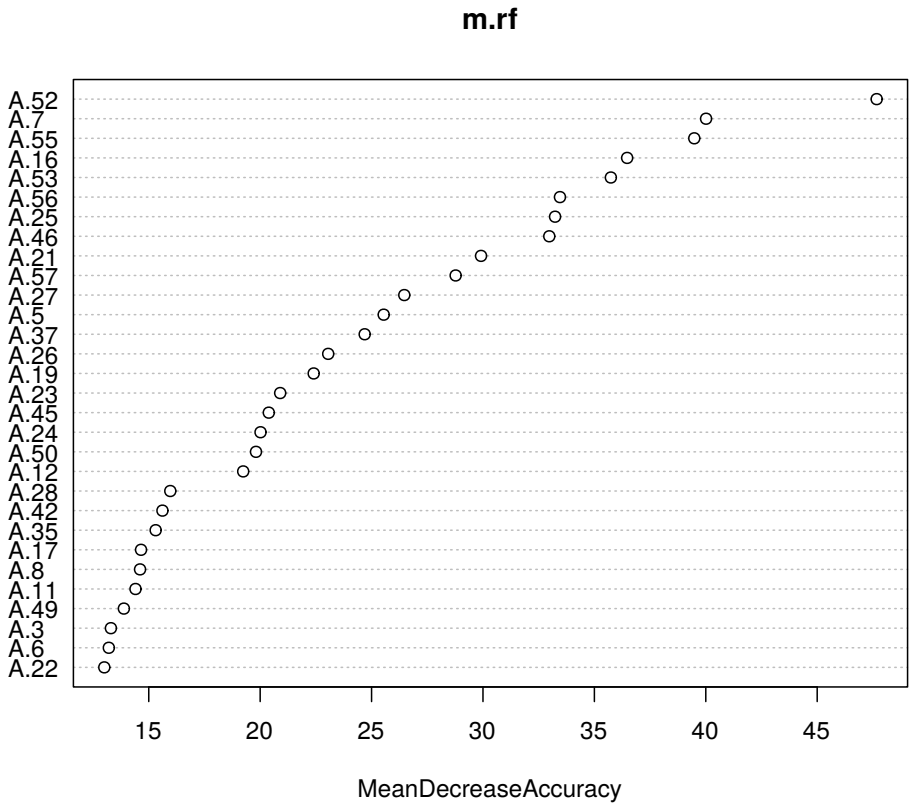
- Node size in tree: 2 (stump)
- Number of trees: 500
- No. of candidate split variables: 2-5
- fitting

```
library(randomForest); #require(rpart)
# fit RF model with stump(2-nodes trees)
m.rf <- randomForest(x=spam[tr.idx, -58], y=spam[tr.idx, 58],
                     ntree=500, mtry=7,
                     importance=TRUE)
names(m.rf)
```

```
##  [1] "call"           "type"           "predicted"
##  [4] "err.rate"       "confusion"      "votes"
##  [7] "oob.times"      "classes"        "importance"
## [10] "importanceSD"   "localImportance" "proximity"
## [13] "ntree"          "mtry"           "forest"
## [16] "y"              "test"           "inbag"
```

 Variables Importance plot

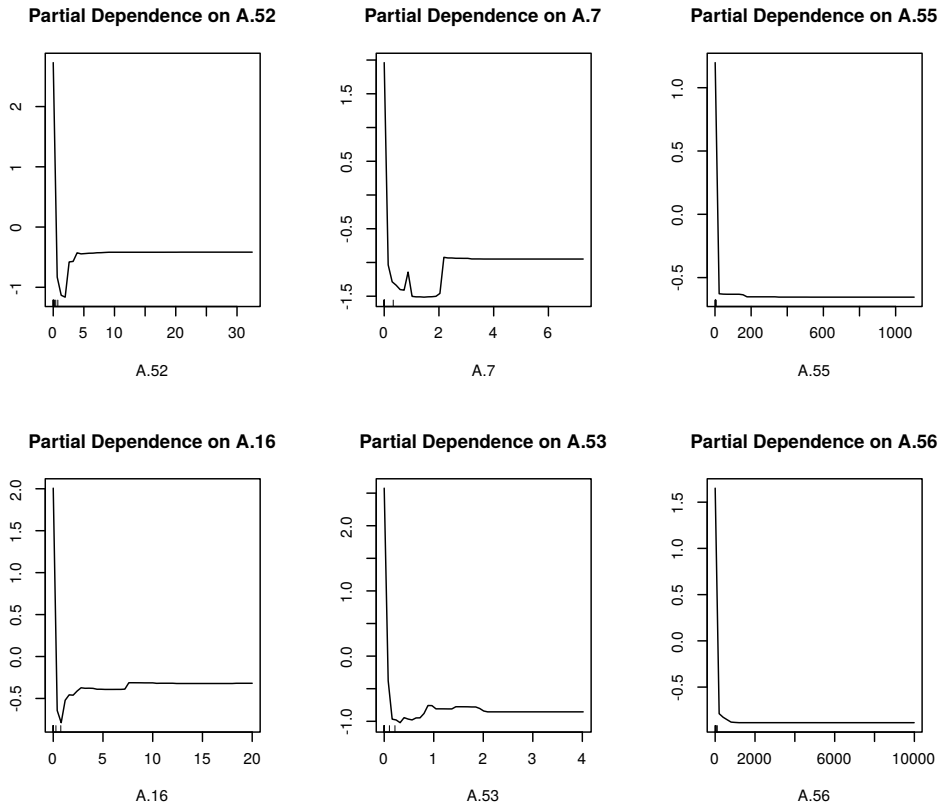
```r
# type=1: mean decrease in accuracy, 2:mean decrease in node impurity
varImpPlot(m.rf, type=1)
```



 Partial dependence plot

- partialPlot(x, pred.data, x.var)
  - x: rf model
  - pred.data : training data
  - x.var : variable name to be examined

```
imp <- importance(m.rf, type=1) # Mean Decrease Accuracy
impvar <- rownames(imp)[order(imp[, 1], decreasing=TRUE)]
par(mfrow=c(2, 3))
for (i in 1:6){
    partialPlot(m.rf, pred.data=spam[tr.idx,], impvar[i], xlab=impvar[i],
                main=paste("Partial Dependence on", impvar[i]))
}
```



- prediction & performance

```
pred.rf <- predict(m.rf, newdata=spam[-tr.idx, -58], type="response") #response¦prob
table(pred.rf, spam$spam[-tr.idx])
```

##

```
## pred.rf email spam
##   email   781   51
##   spam     28  521
```

5 Comparison: bagging, boosting, RF

```
# bagging
mean(pred != spam$spam[-tr.idx])
```

```
## [1] 0.2172339
```

```
# boosting
mean(pred.boo != spam$spam[-tr.idx])
```

```
## [1] 0.0782042
```

```
# RF
mean(pred.rf != spam$spam[-tr.idx])
```

```
## [1] 0.05720492
```