Tree based methods III

Random Forest examples

STAT 32950-24620

Spring 2023 (5/16)

Random Forest for classification

Random Forest

- Random ensemble method on divide-and-conquer approach.
- The idea and tried-and-true power of assembling methods:

Group of weaker classifiers can form a stronger one!

Classification by Random Forest (RF)

- Response variable: Class labels
- Data: n observations; input variable has p features
- Important RF parameters: Fix $m < p$
  ($m << p$ for large p)
- **Bootstrap sampling**

  Select a size n random sample **with replacement**
  from the original sample data to grow a classification tree.
- At each node of the tree, **randomly select m input variables** to split the node.
- Continue to grow a full classification tree without pruning
- Yield a tree with a classification rule for each bootstrap sample
- Grow a forest of T trees.

Decision rule for each observation

Each observation will be in some, not all, bootstrap samples.

That is, each observation will be used in some, not all trees.

Each observation can be classified by each tree it's not in.

For the kth observation,

the classification rule is by the most votes of trees

or by taking the average (as in regression)

where the kth observation is not used in the tree construction.

In other words, when the observation is oob (out-of-bag).

OOB = Out-Of-Bag

Training vs testing per observation

- Each observation is OOB for about one third of the trees.
- From which a classification rule yields for the observation by majority vote.
- From the classification rule yields an error rate.
- The average of the error rate of the n observations is the **oob error rate of the forest**.
- The unused observations (oob = out-of-bag) are used to estimate classification errors for each tree (thus no validation sample is needed).
- Forecast: each new observation will be classified by the forest rule.

Fitting RandomForest

```
library(tree)
library(MASS)
library(randomForest)
```

```
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
mtry=if (!is.null(y) && !is.factor(y))
max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
replace=TRUE, classwt=NULL, cutoff, strata,
sampsize =if (replace) nrow(x) else ceiling(.632*nrow(x)),
nodesize =if (!is.null(y) && !is.factor(y)) 5 else 1,
maxnodes =NULL,
importance=FALSE, localImp=FALSE, nPerm=1,
proximity, oob.prox=proximity,
norm.votes=TRUE, do.trace=FALSE,
keep.forest=!is.null(y) && is.null(xtest),
corr.bias=FALSE, keep.inbag=FALSE, ...)
```


Bagging: Use all variables at each split

($m = p$)

```
set.seed(1)
train = sample(1:nrow(Boston), nrow(Boston)/2)
boston.test=Boston[-train,"medv"]

bag.boston=randomForest(medv~.,data=Boston,
               subset=train,mtry=13,importance=TRUE)
```

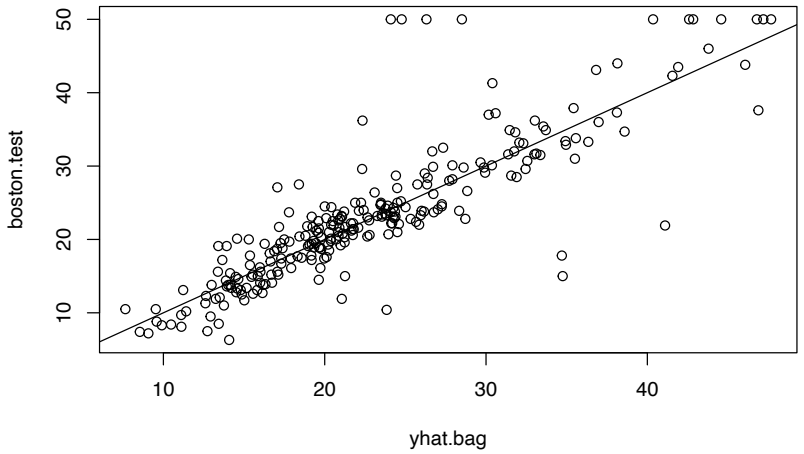


```
bag.boston
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, mtry =
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 11.33
##                     % Var explained: 85.26
```

Checking fit ($m = p$)

```
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag, boston.test)
abline(0,1)
```



Compare mean RSS

$m = p = 13$, $ntree = 500$ (number of trees to grow)

```
mean((yhat.bag-boston.test)^2)
```

```
## [1] 23.46
```


$m = p = 13$, $ntree = 25$

```
bag.boston=randomForest(medv~.,data=Boston,
                        subset=train,mtry=13,ntree=25)
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
mean((yhat.bag-boston.test)^2)
```

```
## [1] 22.99
```


$m = 6$, $ntree = 500$

```
set.seed(1)
rf.boston=randomForest(medv~.,data=Boston,
                       subset=train,mtry=6,importance=TRUE)
yhat.rf = predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)
```

```
## [1] 19.62
```

Variable Importance

OOB observations also used to evaluate variable importance.

Each variable is randomly permuted to check its impact.

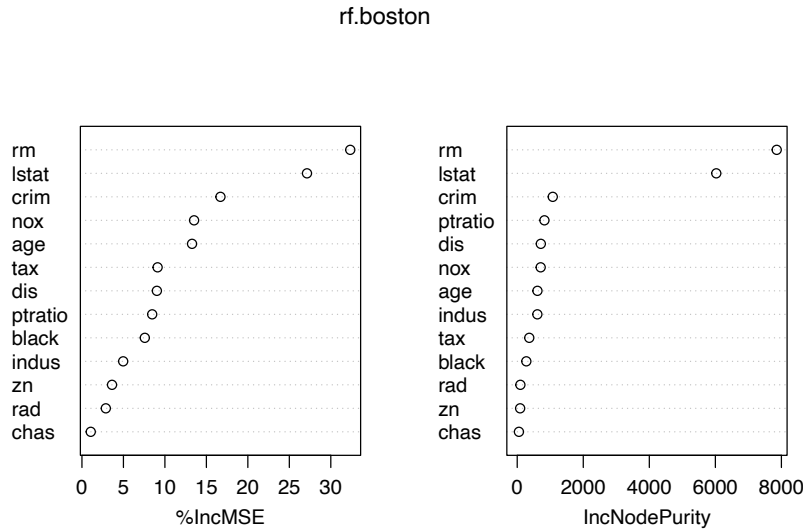
- On change of MSE
- On change of Gini Node Purity (similar to entropy)


```
importance(rf.boston)
```

```
##         %IncMSE IncNodePurity
## crim     16.697       1076.09
## zn        3.626         88.35
## indus     4.969        609.53
## chas      1.061         52.22
## nox      13.518        709.87
## rm       32.343       7857.65
## age      13.272        612.21
## dis       9.032        714.95
## rad       2.878         95.81
## tax       9.119        364.92
## ptratio   8.467        823.93
## black     7.579        275.62
## lstat    27.130       6027.64
```


Variable Importance plot

```
varImpPlot(rf.boston)
```



Remarks on the number of predictors m

- Each node has a different set of m predictors.
- m values
  - $m = 1$: random split
  - $m = p$: bagging
  - $m = \sqrt{p}/2, \sqrt{p}, 2\sqrt{p}$ suggested for Random Forest
  - Less interpretable, more blackbox like Neural Network, Support Vector Machine

Remarks on error rates

- Error rate of the forest increases with the correlation between trees.
- Then the more uncorrelated the trees, the better.
- Trade-off:

  Inter-tree correlation increases with m,

  while error rates of individual trees decreases with m

  (the strength of individual trees increases with m).

More details - variable importance (accuracy)

Accuracy-based importance

Each tree has its own out-of-bag sample of data that was not used during construction.

This sample is used to calculate importance of a specific variable.

- First, the prediction accuracy on the out-of-bag sample is measured.
- Then, the values of the variable in the out-of-bag-sample are randomly shuffled,

keeping all other variables the same.

- Finally, the decrease in prediction accuracy on the shuffled data is measured.

More on accuracy-based variable importance

- The mean decrease in accuracy across all trees is reported.
- This importance measure is also broken down by outcome class.
- Intuitively, the random shuffling means that, on average, the shuffled variable has no predictive power.
- This importance is a measure of by how much removing a variable decreases accuracy, and vice versa — by how much including a variable increases accuracy.
- Note that if a variable has very little predictive power, shuffling may lead to a slight increase in accuracy due to random noise. This in turn can give rise to small negative importance scores, which can be essentially regarded as equivalent to zero importance.

More details - variable importance (Gini)

Gini-based importance

When a tree is built, the decision about which variable to split at each node

uses a calculation of the Gini impurity.

- For each variable, the sum of the Gini decrease across every tree of the forest is accumulated every time that variable is chosen to split a node.
- The sum is divided by the number of trees in the forest to give an average.