Set S05 - Random Forests

STAT 401 (Engineering) - Iowa State University

April 26, 2017

Regression trees

Consider a regression model that uses a set of indicator variables to group the data, e.g.

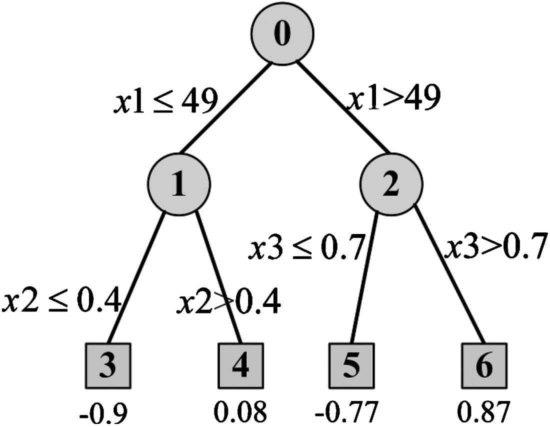
$$Y_i \stackrel{ind}{\sim} N(\mu_i, \sigma^2)$$

where

$$\begin{array}{rlr} \mu_i = & \beta_0 & \text{group 1} \\ & +\beta_1 \mathrm{I}(x_{i1} \le 49)\mathrm{I}(x_{i2} > 0.4) & \text{group 2} \\ & +\beta_2 \mathrm{I}(x_{i1} > 49)\mathrm{I}(x_{i3} \le 0.7) & \text{group 3} \\ & +\beta_3 \mathrm{I}(x_{i1} > 49)\mathrm{I}(x_{i3} > 0.7) & \text{group 4} \end{array}$$

Thus group 1 corresponds to those observations with $x_{i1} \le 49$ and $x_{i2} \le 0.4$.

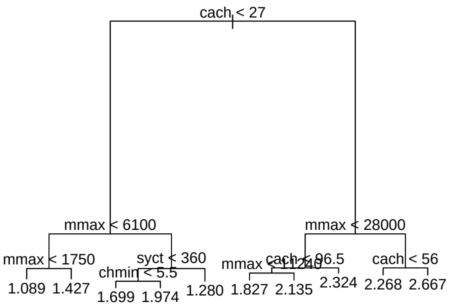
Visualization of a regression tree



Regression trees in R tree

```
library("tree")
data(cpus, package="MASS")
m_tree <- tree(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax, cpus)
summary(m_tree)


Regression tree:
tree(formula = log10(perf) ~ syct + mmin + mmax + cach + chmin +
    chmax, data = cpus)
Variables actually used in tree construction:
[1] "cach"  "mmax"  "syct"  "chmin"
Number of terminal nodes:  10
Residual mean deviance:  0.03187 = 6.342 / 199
Distribution of residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.4945000 -0.1191000  0.0003571  0.0000000  0.1141000  0.4680000
```

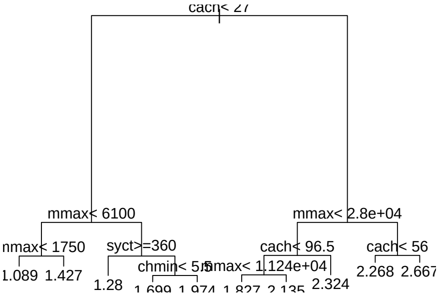
```
plot(m_tree); text(m_tree)
```



Regression trees in R rpart

```
library("rpart")
m_rpart <- rpart(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax, cpus)
summary(m_rpart)


Call:
rpart(formula = log10(perf) ~ syct + mmin + mmax + cach + chmin +
    chmax, data = cpus)
  n= 209

          CP nsplit rel error    xerror       xstd
1 0.54926971      0 1.0000000 1.0080363 0.09735912
2 0.08933901      1 0.4507303 0.4701784 0.04776144
3 0.08763324      2 0.3613913 0.4274450 0.04457527
4 0.03281589      3 0.2737580 0.3227759 0.03101707
5 0.02692205      4 0.2409421 0.3118627 0.03024666
6 0.01855609      5 0.2140201 0.2954596 0.02917108
7 0.01679918      6 0.1954640 0.2919951 0.03094696
8 0.01579084      7 0.1786648 0.2873176 0.03034303
9 0.01000000      9 0.1470831 0.2588373 0.02846206

Variable importance
 cach  mmax  mmin chmin  syct chmax
   25    20    17    15    14     9

Node number 1: 209 observations,    complexity param=0.5492697
  mean=1.753333, MSE=0.2062945
  left son=2 (143 obs) right son=3 (66 obs)
  Primary splits:
      cach  < 27    to the left,  improve=0.5492697, (0 missing)
      mmax  < 14000 to the left,  improve=0.4942141, (0 missing)
```

```r
plot(m_rpart); text(m_rpart)
```



How do this approaches decide on the splits?

From the help file for `tree`:

> *A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side. Numeric variables are divided into $X < a$ and $X > a$; the levels of an unordered factor are divided into two non-empty groups. The split which maximizes the reduction in impurity is chosen, the data set split and the process repeated.Splitting continues until the terminal nodes are too small or too few to be split.*

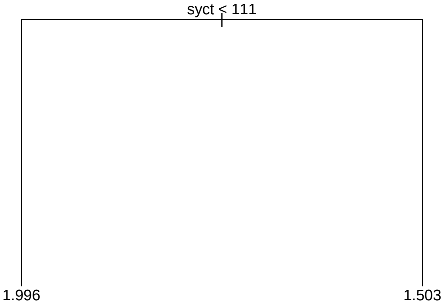
The *impurity* for a regression tree is most likely the estimate of $\hat{\sigma}^2$. Thus, the algorithm searches over all possible splits and finds the one that results in the smallest $\hat{\sigma}^2$. Then the process is repeated for each split.

To determine when to stop, the algorithm has a set of control values. For `tree` the values are

- mincut: minimum number of observations to include in either child node
- minsize: smallest allowed node size
- mindev: within-node deviance must be at least this times that of the root node for the node to be split

Little tree

```
m_tree <- tree(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax, cpus,
               control = list(mincut = 100, mindev = 0.01, minsize = 200, nmax = 90))
plot(m_tree); text(m_tree)
```



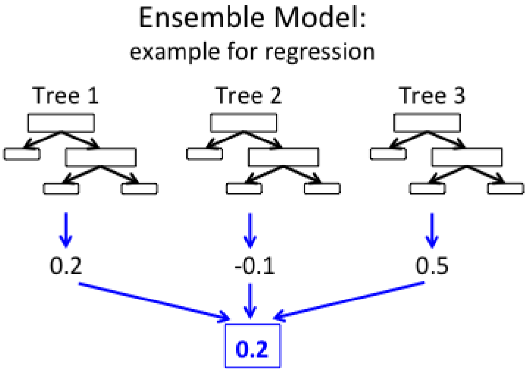
Random forests

Repeat this algorithm B times:

1. Randomly sample data with replacement from training set.
2. Train a tree on these data (randomly evaluating a subset of explanatory variables for each split).
3. Evaluate the tree based on its out of sample performance.

After training, predictions for new data are averaged across all the trees.

Visualizing

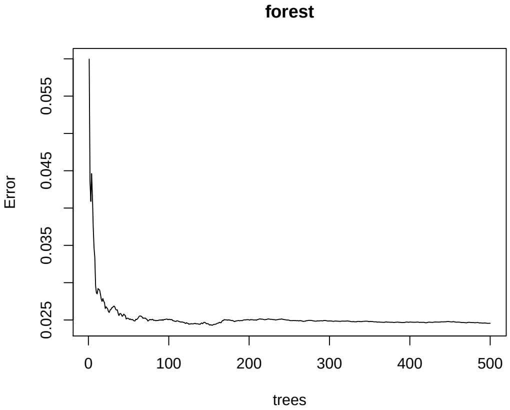


Random forests in R

```
forest <- randomForest(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax,
                       data = cpus,
                       importance = TRUE)
forest


Call:
 randomForest(formula = log10(perf) ~ syct + mmin + mmax + cach +      chmin + chmax, data = cpus, importance =
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 0.024559
                    % Var explained: 88.1
```

Out of bag error

```
plot(forest)
```



Variable importance

```r
importance(forest) %>% round(2)
```

```
      %IncMSE IncNodePurity
syct    16.94          3.79
mmin    17.84          5.43
mmax    31.60         11.08
cach    32.59         11.87
chmin   19.57          6.16
chmax   20.02          2.96
```

Prediction

```
new_cpus = cpus %>%
  sample_n(10)
10^predict(forest, new_cpus, interval="confidence") # Performance with no uncertainty
```

```
      101       134        82       116       165       155        86         6       195       153
 21.43632  33.77890  37.36409  71.12837  47.50400 184.26909  30.50168 315.67296 102.32854 318.04787
```

Classification trees