TreeBasedRegression

November 4, 2019

1 Tree-based regression

Tree-based regression makes predictions on a target variable y by developing rules that divide our X data set (called our feature space) into regions. Given a vector of covariates x_i^*, a TBR model looks at the training observations, picks those observations that are simialr, and averages the corresponding y values. A TBR model does this by partitioning the training observations through a series of rules. TBR models are non-linear.

As an example, a TBR model could specify the following rules: (i) if $x_1 < 4.5$ and $x_2 < 2.0$ then y falls in region 1, (ii) if $x_1 < 4.5$ and x_{2}

$$\geq$$

2.0 then y falls in region 2,if x_{1}

$$\geq$$

4.5 and $x_2 < 2.0$ then y falls in region 3, and if x_{1}

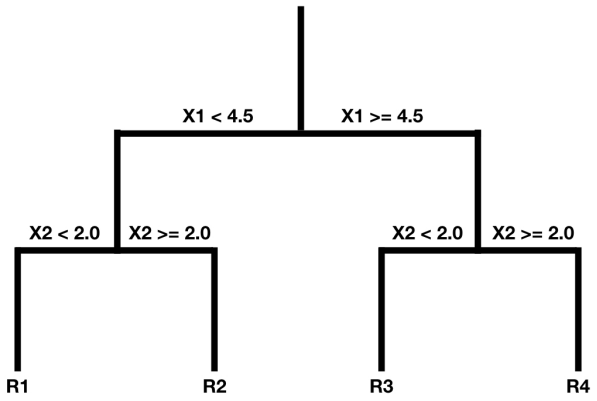
$$\geq$$

4.5 and x_{2}

$$\geq$$

2.0 then y falls in region 4.

And the above rule can be represented as a tree.



You start at the top (root) of the tree with your x covariates. Work your way down the tree, and every time you come to a split, take the path that corresponds to your x data. For example, if my x data was $x = [2, 5]$ I would move to the left at the first split because $x_1 = 2 < 4.5$. I would continue down the left side of the tree until the next split. At the next split I would move right because $x_2 = 5 > 2.0$ and the prediction my tree would make is the average of all y values in region 2.

There is a simple idea behind trees and rule-based regression methods. Observations that have similar characteristics can be predicted with a simpler model that explains $P(Y|X)$. Rather than try to model $P(Y|X)$ directly, which could have complicated nonlinearities, model a small portion of $P(Y|X^*)$ that has a simpler structure. By taking a simple average of y values we are assuming that, in any given small region of feature space, the y values can be predicted to be a simple average.

Rather than a tree or textual rules, we can graphically understand how the tree above splits our data. First, we can load the ISLR package that includes the Hitters dataset.

[1]:
```
require(ISLR)
print(head(Hitters))
```

```
Loading required package: ISLR
```

```
                  AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
-Andy Allanson      293   66     1   30  29    14     1    293    66      1
-Alan Ashby         315   81     7   24  38    39    14   3449   835     69
-Alvin Davis        479  130    18   66  72    76     3   1624   457     63
-Andre Dawson       496  141    20   65  78    37    11   5628  1575    225
-Andres Galarraga   321   87    10   39  42    30     2    396   101     12
-Alfredo Griffin    594  169     4   74  51    35    11   4408  1133     19
                  CRuns CRBI CWalks League Division PutOuts Assists Errors
```

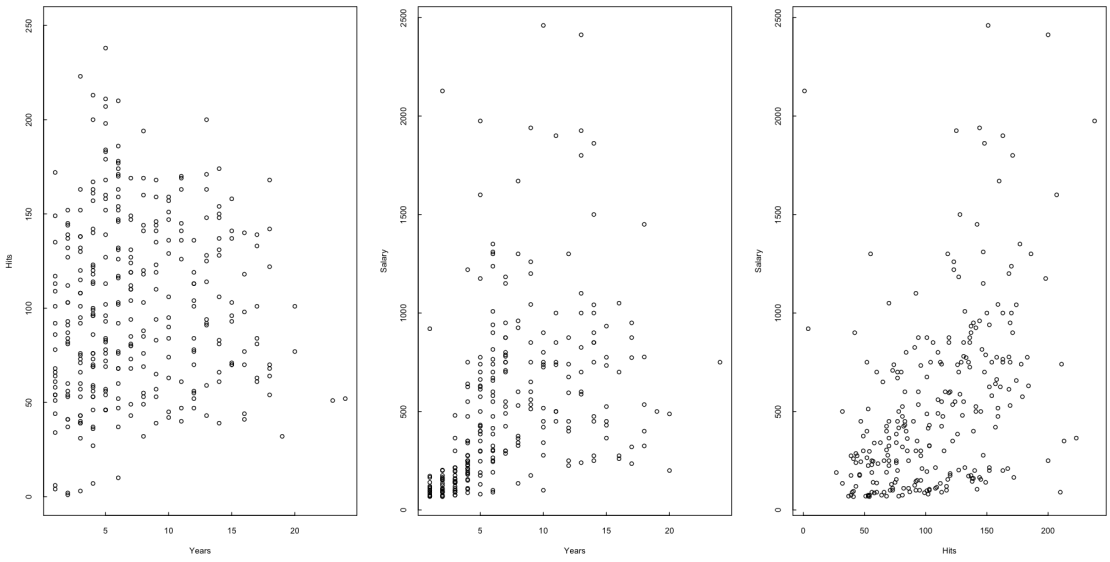
```
-Andy Allanson       30   29     14       A        E      446      33      20
-Alan Ashby         321  414    375       N        W      632      43      10
-Alvin Davis        224  266    263       A        W      880      82      14
-Andre Dawson       828  838    354       N        E      200      11       3
-Andres Galarraga    48   46     33       N        E      805      40       4
-Alfredo Griffin    501  336    194       A        W      282     421      25
                  Salary NewLeague
-Andy Allanson        NA         A
-Alan Ashby        475.0         N
-Alvin Davis       480.0         A
-Andre Dawson      500.0         N
-Andres Galarraga   91.5         N
-Alfredo Griffin   750.0         A
```

For this example, I'm using the number of years the baseball is active and the number of hits they have as my X data. I will try and predict baseball player's salary, my target (y) variable.

[6]:
```
x1 = Hitters$Years
x2 = Hitters$Hits
y =  Hitters$Salary
```

First I will plot my $x1$ and $x2$ data.

[12]:
```
options(repr.plot.width=15,repr.plot.height=8)
par(mfrow=c(1,3))
plot(x1,x2, ylim=c(0,250),ylab='Hits'  ,xlab='Years',tck=0.01)
plot(x1,y ,ylab='Salary',xlab='Years',tck=0.01)
plot(x2,y ,ylab='Salary',xlab='Hits',tck=0.01)
```



The relationship between salary, and hits and years appears non-linear. A linear model could be difficult to fit, and accurately predict salary. Assume instead we created a tree (like the above example) with two rules. The first rule partitions the data into Years less than or greater than or equal 4.5. The second rule then partitions the data by the average number of hits a player has as greater than or equal, or less than 117.5.

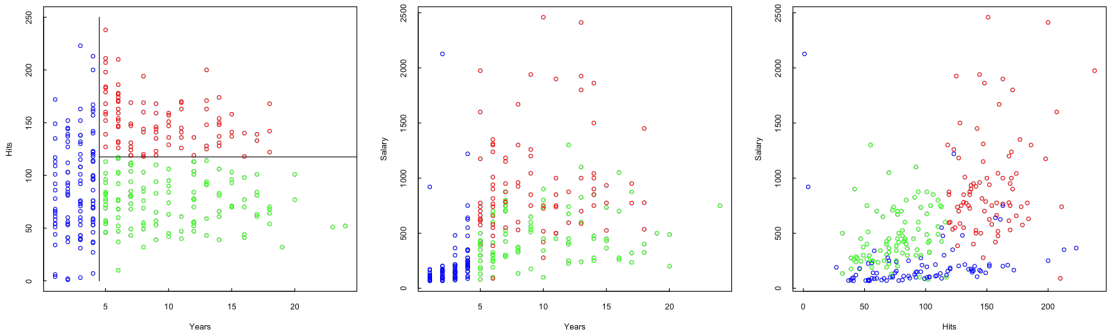
[25]:
```
options(repr.plot.width=15,repr.plot.height=5)
par(mfrow=c(1,3))

color = rep(0,length(x1))
for (i in 1:length(x1)){
    singleX1 = x1[i]
    singleX2 = x2[i]

    if (singleX1 < 4.5){
       color[i] = "blue"
    }
    else if (singleX1 >= 4.5 & singleX2 < 117.5){
       color[i] = "green"
    }
    else if (singleX1 >= 4.5 & singleX2 >= 117.5){
       color[i] = "red"
    }
}

plot(x1,x2,col=color, ylim=c(0,250),ylab='Hits',xlab='Years',tck=0.01)
lines(rep(4.5,2),c(0,250))
lines(c(4.5,25), rep(117.5,2) )

plot(x1,y,col=color ,ylab='Salary',xlab='Years',tck=0.01)
plot(x2,y,col=color ,ylab='Salary',xlab='Hits',tck=0.01)
```



The feature space (left) is partitioned into three regions. These regions correspond to predictions of salary.

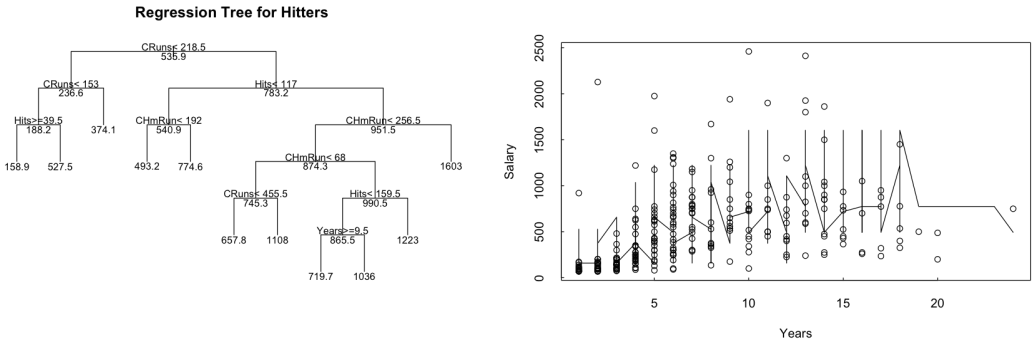
This tree is relatively simple. More complicated trees can be built from these same two X variables

[46]:
```
library(rpart)
fit <- rpart(Salary~Hits+Years+CHmRun+CRuns+Errors, method="anova", data=Hitters)

par(mfrow=c(1,2))
plot(fit, uniform=TRUE,
   main="Regression Tree for Hitters ")
text(fit, use.n=FALSE, all=TRUE, cex=0.8)

X = Hitters[,c("Hits","Years","CHmRun","CRuns","Errors")]
X = X[order(Hitters$Years),]
predictions = predict(fit,X)

plot(x1,y,xlab='Years',ylab="Salary",tck=0.01)
lines(X$Years,predictions)
```



On the left, a tree is made by splitting the feature space over and over into finer and finer sets of observations. Within each region the y values are averaged and considered the predictions for all observations in that region.

On the right, the predictions are plotted over the data "Years" by "Salary". The constant, and jagged looking, appearance of the predictions is because of this "constant within region" approach to prediction.

1.1 Prediction by splitting the feature space

To make it clear, Trees split our feature space (X data) into J different regions. Given x_{new}, a prediction is made by finding the region x_{new} belongs to and averaging all the training example y values that fall into that same region. The same prediction is made for **all** observations that fall into that region.

1.2 How to find the splits

To find a tree that makes the "best" predictions, we need to define what best means. For this class, by best we mean the tree that minimizes

$$\text{SSE}(\hat{y}_{R_j}) = \sum_{j=1}^{J} \sum_{x_i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$$

or the sum square errors. The Sum square errors is a function of j different variables, one prediction for each region (R_j). An exhaustive approach to finding the best set of $\hat{y}_{R_j}$ that minimize the SSE would consider every possible tree. But this means we would need to compute the SSE for every possible combination of every possible cut point for all variable in our feature space. This type of computation isn't feasible in a reasonable amount of time.

Instead, we will take a top-down approach called recursive binary splitting. Recursive binary splitting examines, for all variables X_k and corresponding cut points, the single X variable and cut point that minimizes the SSE. The above equation reduces to

$$\text{SSE}_{\text{RBS}}(\hat{y}_{R_1}, \hat{y}_{R_2}) = \sum_{x_i \in R_1} (y_i - \hat{y}_{R_1})^2 + \sum_{x_i \in R_2} (y_i - \hat{y}_{R_2})^2$$

where $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ correspond to the two regions generated by picking a single covariate X_j and a single cut point $X_j < s$ that minimize the SSE. This SSE_{RBS} set j equal to 2 in the more general SSE equation above.

The process is repeated, except now we find the best covariate and cutpoint that minimizes SSE_{RBS} inside R_1 and inside R_2.

[ ]:
```
library(rpart)
fit <- rpart(Salary~Hits+Years, method="anova", data=Hitters)

plot(fit, uniform=TRUE,
   main="Regression Tree for Hitters ")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```