TEXAS McCombs

MASTER OF SCIENCE IN BUSINESS ANALYTICS

Introduction to trees

Outline

Trees

Regression trees

Classification trees

Summary

Trees

Tree based methods are a major player in data-mining.

Good:

- flexible fitters, capture non-linearity and interactions.
- do not have to think about scale of variables.
- handles categorical and numeric y and x very nicely.
- fast.
- interpretable (when small).

Bad:

Not the best in out-of-sample predictive performance
(*but not bad!*).

But,

If we **bag** or **boost** trees, we can get the best off-the-shelf prediction available.

Bagging and Boosting are *ensemble methods* that combine the fit from many (hundreds, thousands) of tree models to get an overall predictor.

1. Regression trees

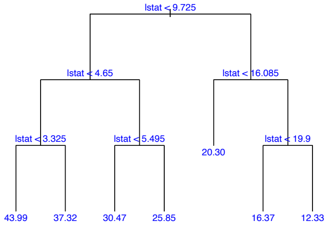
Let's look at a simple 1-dimensional example so that we can see what is going on.

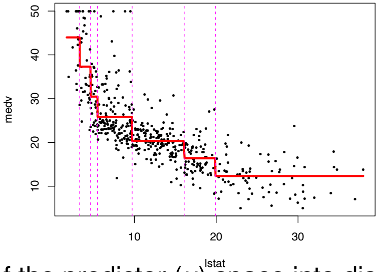
We'll use the Boston housing data and relate x=lstat to y=medval.

At left is the *tree* fit to the data.

At each *interior node* there is a decision rule of the form $\{x < c\}$.
If $x < c$ you go left, otherwise you go right.

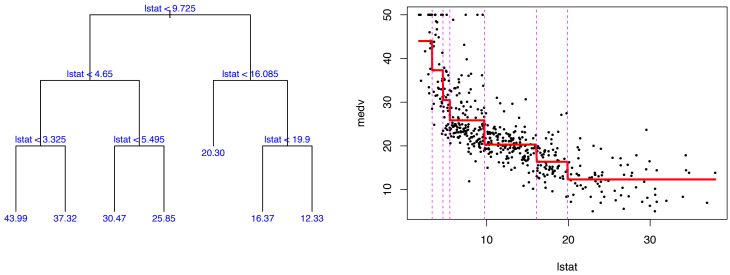
Each observation is sent down the tree until it hits a bottom node or *leaf* of the tree.





The set of bottom nodes gives us a partition of the predictor (x) space into disjoint regions. At right, the vertical lines display the partition. With just one x, this is just a set of intervals.

Within each region (interval) we compute the average of the y values for the subset of training data in the region. This gives us the step function which is our $\hat{f}$. The $\bar{y}$ values are also printed at left at the bottom nodes.

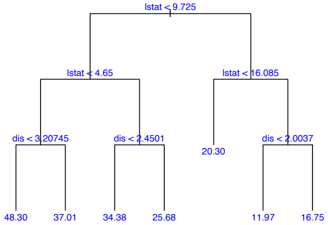


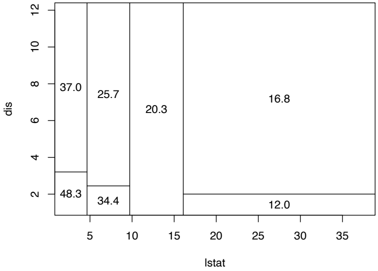
To predict, we just use our step function estimate of $f(x)$.

Equivalently, we drop x down the tree until it lands in a leaf and then predict the average of the y values for the training observations in the same leaf.

A Tree with Two Explanatory Variables

Here is a tree with $x = (x_1, x_2) =$ (lstat,dis) and y=medv. Now the decision rules can use either of the two x's.



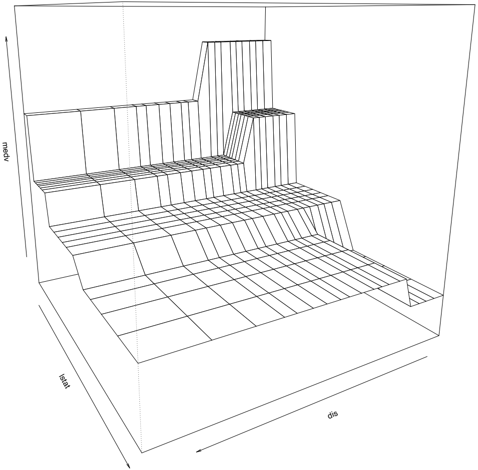


At right is the *partition* of the x space corresponding to the set of bottom nodes (leaves). The average y for training observations assigned to a region is printed in each region and at the bottom nodes.

This is the regression function given by the tree.

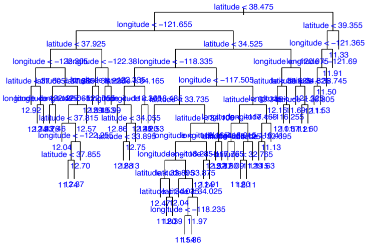
It is a step function which can seem simple, but it delivers non-linearity *and* interactions in a simple way and works with a lot of variables.

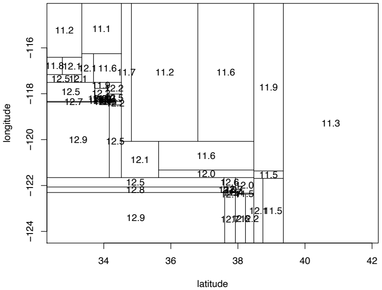
Notice the interaction.
The effect of `dis` depends on `lstat`!!



The California Housing Data

Here is a tree with 50 bottom nodes fit to the California Housing data using only `longitude` and `latitude`.

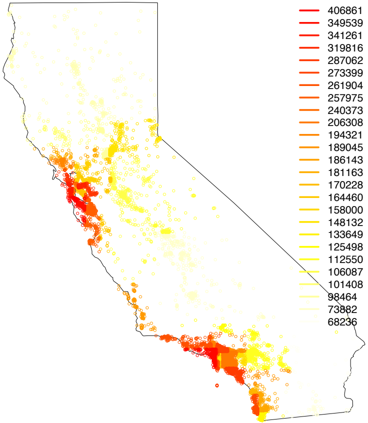




Don't extrapolate into the ocean!

Here is a view of the fit using the map of the state.

(units are dollars, the logMedVal was exponentiated for the labels).



Classification trees

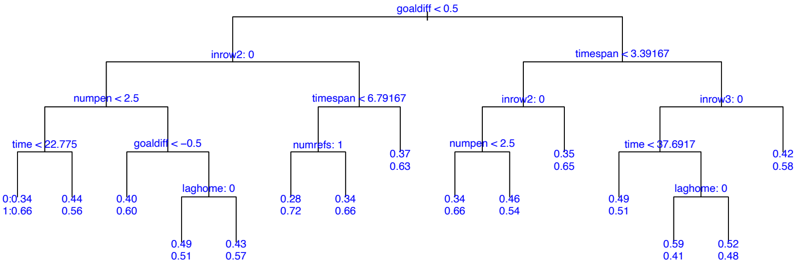
Let's build a tree for a classification problem.

We'll use the hockey penalty data.

The response is whether or not the next penalty is on the other team and x is a bunch of stuff about the game situation (the score, etc ...).

In addition, this time some of our predictors (features, x's) are categorical.

Here is the tree:



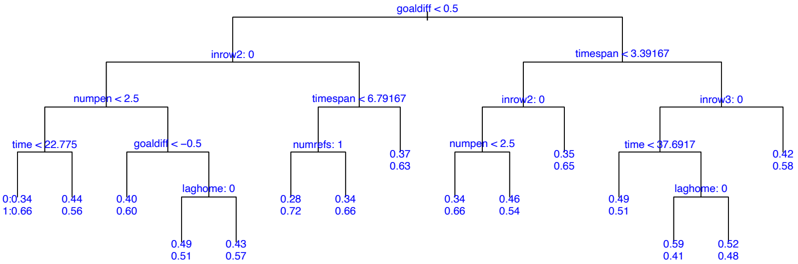
- Each bottom node gives the fraction of training data in the two outcome categories. Think of it as $\hat{p}$ for the kind of x associated with that bottom node.
- The form of the decision rule can't be $x < c$ for categorical variables.
  We pick a subset of the levels to go left. `inrow2:0` means all the observations with `inrow2` in the category labeled 0 go left.

if:

- if you are not winning
- you had the last two penalties
- it has not been long since the last call
- and there is only 1 referee

then:

there is a 72% chance the next call will be on the other team.



Summary of trees

- Trees use recursive binary splits to partition the predictor space.
- Each binary split consists of a decision rule which sends x left or right.
- For numeric x_i, the decision rule is of the form if $x_i < c$.
- For categorical x_i, the rule lists the set of categories sent left.
- The set of bottom nodes (or leaves) give a partition of the x space.
- To predict, we drop an out-of-sample x down the tree until it lands in a bottom node.
- For numeric y, we predict the average y value for the training data that ended up in the bottom node.
- For categorical y we use the category proportions for the training data that ended up in the bottom node.

Good:

- Handles categorical/numeric x and y nicely.
- Don't have to think about the scale of x's !!!
- Computationally fast ("scales").
- Small trees are interpretable.
- Variable selection.

Bad:

- Step function is crude, does not give the best predictive performance.
- Hard to assess uncertainly.
- Big trees are not interpretable.