CS6301: R For Data Scientists

LECTURE 20: TREES

Decision Trees

Popular classification method

Can also be used for regression, but less popular

Works extremely well with categorical predictors, but can also handle numerical predictors

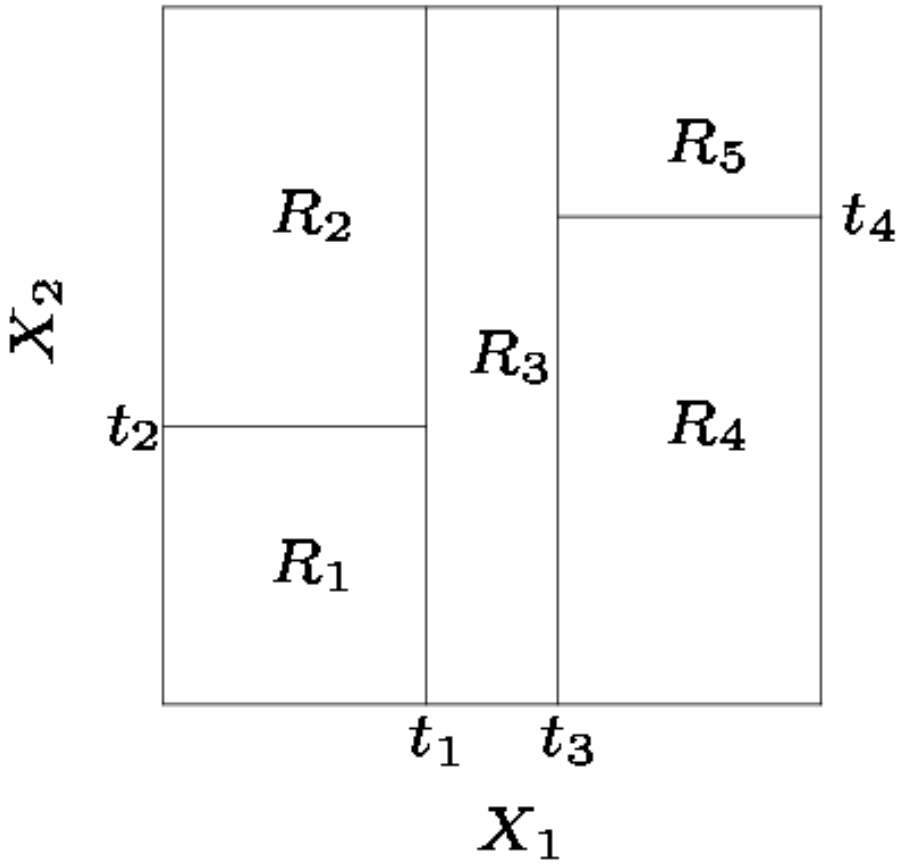
Base method is prone to overfitting, will not give good results …

There are enhancements that make the method work well

Trees

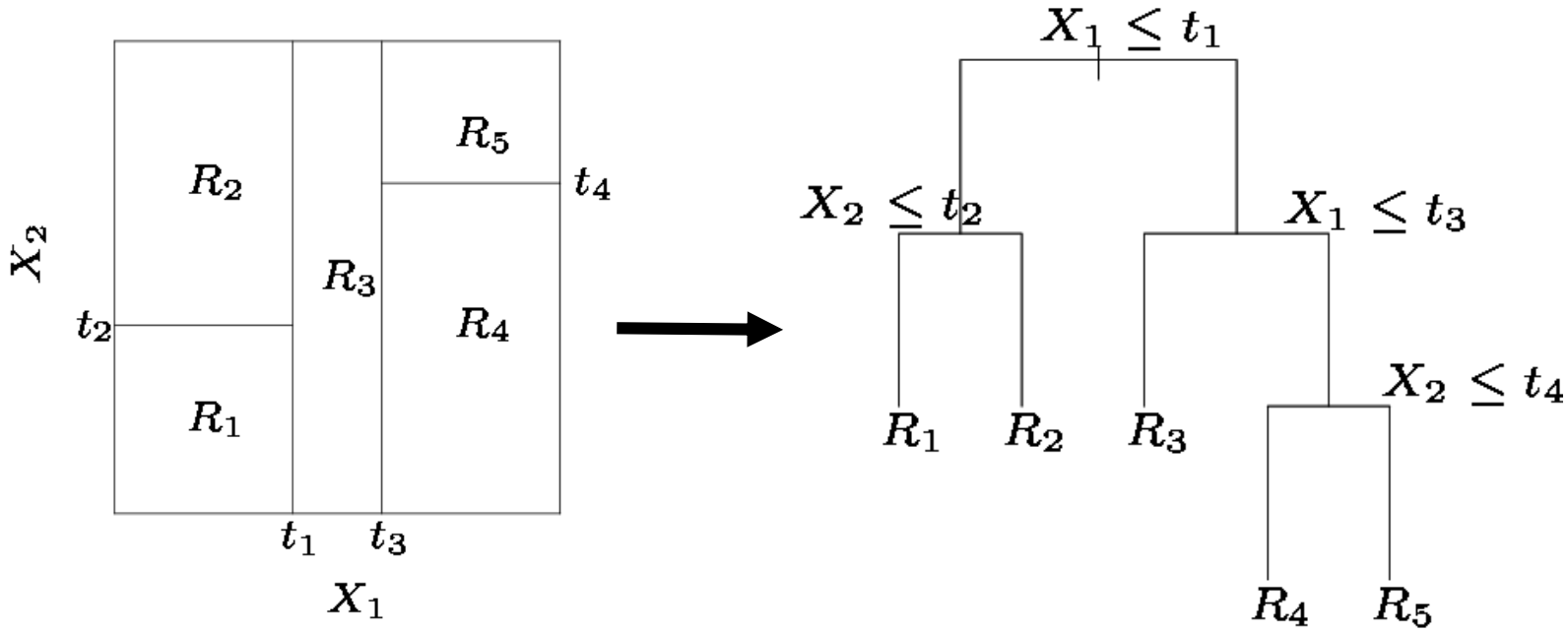
One way to make predictions in a classification problem is to divide the predictor space (i.e. all the possible values for $X_1,X_2,\ldots,X_p$) into distinct regions, say R_1, $R_2,\ldots,R_k$

Then for every X that falls in a particular region (say R_j) we make the same prediction based upon a "majority vote" of all training points in that region.



Trees - Interpretability

- When we create partitions this way we can always represent them using a tree structure.
- This provides a very simple way to explain the model to a non-expert.



Building The Tree

Suppose *y* is binary (0,1) for a moment …

Given a set *S* of training points, define

$$Entropy(S) \equiv -p_0 log_2 p_0 - p_1 log_2 p_1$$

This measures the level of "homogeneity" in a set …

Suppose all predictors are categorical for the moment

At each step of the algorithm, we split on a particular value that a predictor may take on

We choose this value to be the one that is most likely to reduce the entropy the most

Building The Tree

Let's suppose we have a predictor *x* which takes on values {a, b, c}

Let S_a denote all training points that have $x = a$ and $S_{not(a)}$ be the set of all training points where *x* is not *a*

To see how much splitting on $x = a$ would reduce entropy, we calculate the gain …

$$Gain = Entropy(S) - \frac{|S_a|}{|S|} Entropy(S_a) - \frac{|S_{not(a)}|}{|S|} Entropy(S_{not(a)})$$

This is the current entropy minus the expected value for the entropy if we split on this value for this predictor

Building The Tree

This process continues, until all values have been split

This also works for numeric predictors – treat each value as a possible split point – but can lead to complicated trees

For regression problems, make the split based upon largest decrease in RSS

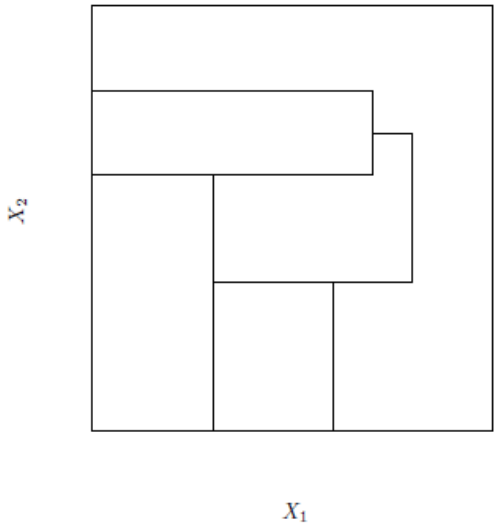
- Here, we use the average value of the training points in the region for our predicted value

Major problem – this approach will overfit the training data, leading to bad predictive models

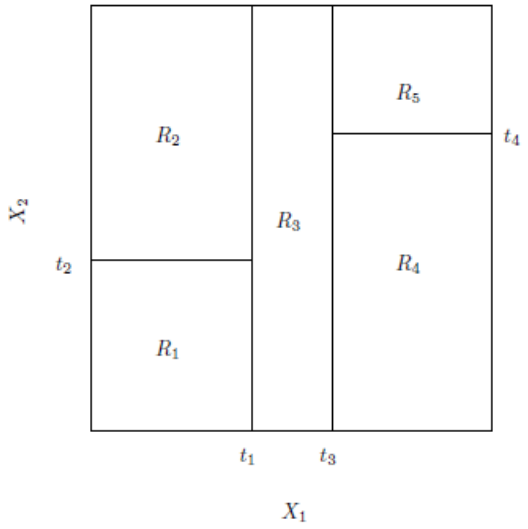
Bias in Decision Trees

Second issue: This approach builds the tree in a particular way, which may not always fit with how the response variable is related to the predictors …

This is actually a bias in the method



Not Possible



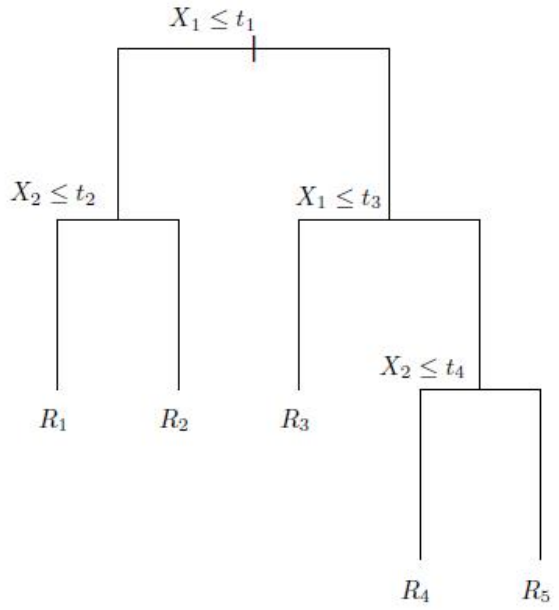
Possible

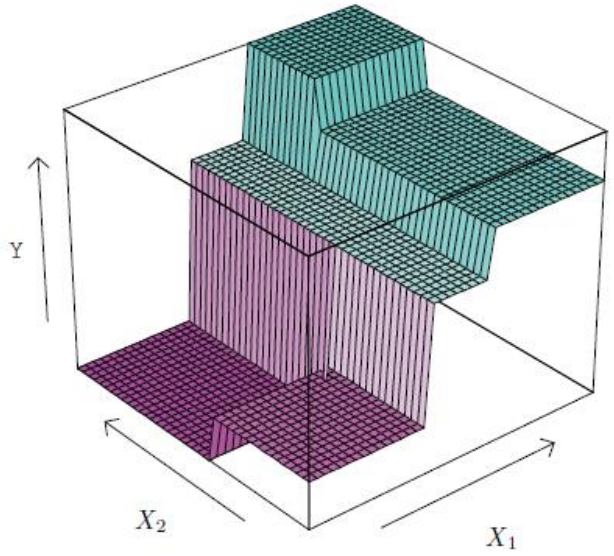
Variance in Trees

The method gives very nonlinear solutions

Note this is nonparametric – we do not need to specify a form for the solution

High variance





Trees Versus Linear Methods

If the relationship between the predictors and response is linear, then classical linear models such as linear regression would outperform regression trees

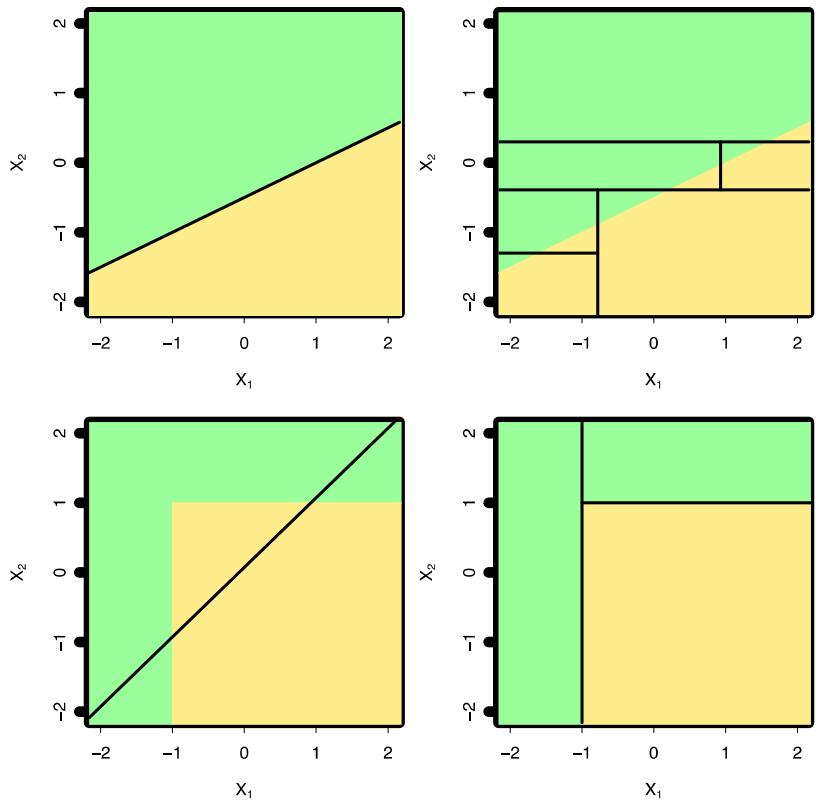
On the other hand, if the relationship between the predictors is non-linear, then decision trees would outperform classical approaches

How to avoid overfitting?

- One approach is “pruning” where we deliberating stop growing the tree at a certain point
- How deep? Cross Validation can be used

Trees Versus Linear Methods

- Top row: the true decision boundary is linear
  - Left: linear model (good)
  - Right: decision tree
- Bottom row: the true decision boundary is non-linear
  - Left: linear model
  - Right: decision tree (good)



Better Fitting Trees

Pruning:

- Build trees of different depths, and provide a measure of CV error for each depth
- Should see the typical Bias-Variance tradeoff curve

Another approach: Ensemble Method called “Bagging”, often useful for high variance models

- Basic idea: Create many models, average them together – error should cancel!
- Where do we get the models?

Bagging

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

Bagging – **Bootstrap Aggregate**
Create Bootstrap samples from original sample, calculate models on these
Average the models to get prediction at new point
Note that this is a general method – can be applied to many regression methods
For classification use majority vote, use average for regression
Lose interpretability

Out-Of-Bag Observations

For a sample of size n, the probability that x_i is not in the boot sample is $(1 - 1/n)^n$, which approaches $e^{-1} \sim .367$ as n gets large

A third of the observations are left out of the boot sample – we can use these to calculate a test error on the tree we created

- Create an prediction for x_i for each tree it is not a part of, then average these to get the prediction for this point (called an Out-Of-Bag Observation)
- Do the same for each point in sample
- Use these to compute error for each point, average to estimate test error
- Much like LOOCV

Key Points

Note that interpretation is hard for these models

Some packages produce a ranking of how important each predictor is in reducing RSS, entropy or Gini – usually shown as bar chart

Recall that the key to reducing variance was to have each estimate independent

Not necessarily true for bagging, but method still works very well

Random Forests

Random Forests improves on bagging with a tweak: for each boot sample forest, at each split, only consider a random subset of *m* predictors to split

- Recall how trees were constructed for Bagging – at each split, all values for all predictors are considered to see which split reduces RSS or entropy the most
- For RF, only consider a subset of predictors at each split

Makes the trees more independent, so when the values are averaged, the errors are de-correlated

Set *m* small for a large number of correlated predictors