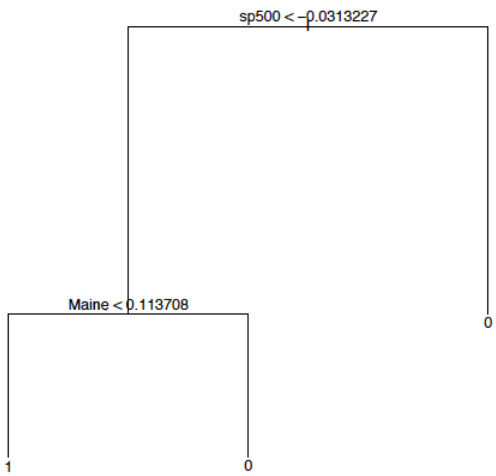
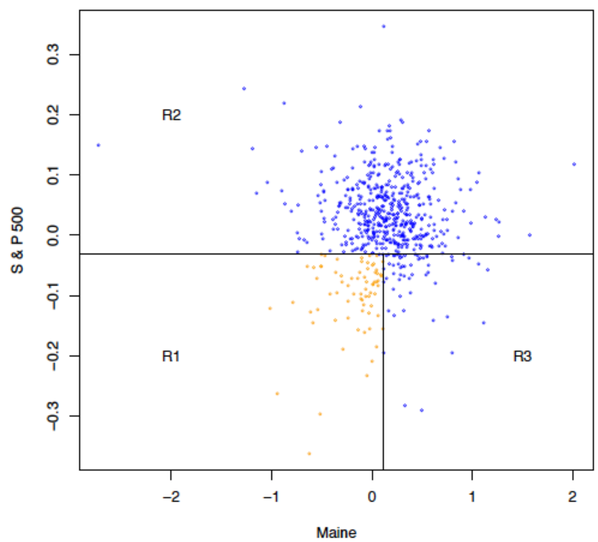
Statistical Machine Learning 4 — Lecture 13 – Decision Trees October 14th, 2014

Lecturer: Prof. Homrighausen Scribe: Chuck Vollmer

1 What is a (decision) tree?





Basically...

- Trees are models that *stratify* or *segment* the predictor space into a number of simple regions.
- We predict all observations in a region the same prediction
- The three regions R1, R2, and R3 are the *terminal nodes*

Main takeaways...

- Trees are simple and useful for interpretation.
- Basic trees are not great at prediction.
- More modern methods that use trees are much better.

2 How do we build a tree?

1. Divide the predictor space into M non-overlapping regions $R_1, \ldots, R_M$

   -this is done via greedy, recursive, binary splitting

2. Every observation that falls into a given region R_m is given the same prediction
   - **Regression:** The average of the responses for a region
   - **Classification:** Determined by majority (or plurality) vote in that region

2.1 Important:

- Trees can only make rectangular regions that are *aligned* with the coordinate axis.
- The fit is *greedy*, which means that after a split is made, all further decisions are conditional on that split.

3 Regression Trees

For a given partition $R_1, \ldots, R_M$, the model for the response is

$$f(x) = \sum_{m=1}^{M} c_m \mathbf{1}_{R_m}(x)$$

For squared error loss, if $n_m = \sum_{i=1}^{n} \mathbf{1}_{R_m}(X_i)$, then

$$\hat{c}_m = n_m^{-1} \sum_{i: x_i \in R_m} Y_i$$

As for the regions, M encodes the tree complexity

This is challenging as considering all possible regions is computationally *infeasible*

(This would involve sifting through all $M \leq n$ and all configurations for R_m.)

3.1 Model Selection for Trees

As a greedy approximation, do the following

1. **Grow a large tree:** $T_{\max}$, stopping when some minimal terminal node size requirement is met
2. **Cost-complexity pruning:** For all $\lambda \geq 0$

$$C_\lambda(T) = \sum_{m=1}^{M} \sum_{i: x_i \in R_m} (Y_i - \hat{c}_m)^2 + \lambda M$$

   (Note that often it is written that $|T|= M$)

3. **Weakest link pruning:** For each λ, there is a unique smallest T_λ that minimizes $C_\lambda(T)$. Eliminating nodes that produce the smallest increase in training error produces a sequence of solutions that must contain T_λ

   (many details ommitted)

4 Classification Trees

The only modification for *classification* is choice of *loss function*

For region m and class g, we get training proportions

$$\hat{p}_{mg}(x) = \mathbf{1}_{R_m}(x) n_m^{-1} \sum_{i:X_i \in R_m} \mathbf{1}(Y_i = g)$$

Our classification is

$$\hat{g}(x) = \max_g \hat{p}_{mg}(x)$$

4.1 And measuring quality of fit?

Different measures of *node impurity* (loss function in tree terminology)

classification error rate:	$E = 1 - \max_k(\hat{p}_{mk})$
Gini index:	$G = \sum_k \hat{p}_{mk}(1 - \hat{p}_{mk})$
cross-entropy:	$D = -\sum_k \hat{p}_{mk} \log(\hat{p}_{mk})$

Both G and D can be thought of as measuring the purity of the classifier (small if all $\hat{p}_{mk}$ are near zero or 1). These are preferred over the classification error rate.

(Also, E isn't differentiable and hence not as amenable to numerical optimization)

We build a classifier by *growing* a tree that minimizes G or D.

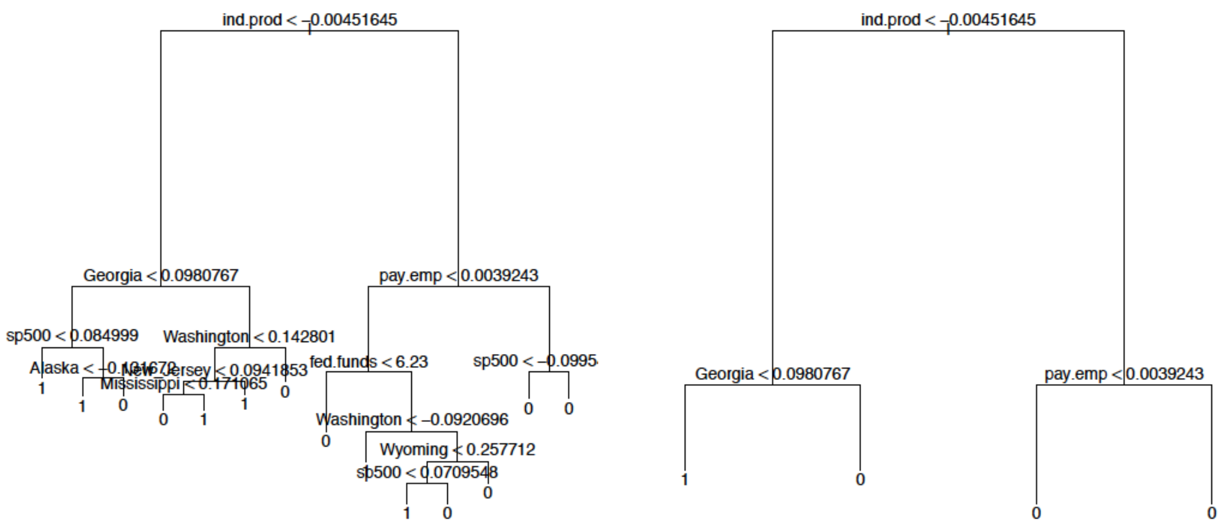
5 Advantages and Disadvantages of Trees:

+ Trees are very easy to explain (much easier than even linear regression).
+ Some people believe that decision trees mirror human decision.
+ Trees can easily be displayed graphically no matter the dimension of the data.
+ Trees can easily handle qualitative predictors without the need to create dummy variables.
− Trees aren't very good at prediction.

To fix this last one, we can try to grow many trees and average their performance. This can be done through *Bagging! Boosting* -and- *Random Forests* are variations which also address this last issue.

6 Comparing tree depths:

Trees, in any of their varied forms of aggregation as well, can overfit and thus can be "pruned." Below is an illustration of this procedure:



Unpruned tree Pruned Tree