Decision Trees

-Applied Multivariate Analysis & Statistical Learning-

ISL 8.1

Lecturer: Darren Homrighausen, PhD

Preamble:

- Trees involve stratifying or segmenting the feature space into a number of simple regions
- Trees are simple and useful for interpretation
- Basic trees are not great at prediction
- More modern methods that use trees are much better, though not as interpretable

A motivating example: Reminder

Suppose we are interested in predicting whether or not the economy will be in a recession

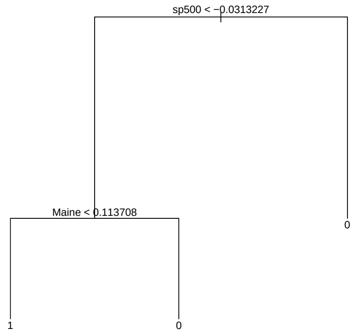
We have quarterly measurements of

- State level economic growth
  (Larger number is better)
- Federal level variables such as GDP, interest rates, employment, S&P 500, ...

Here, we will code the supervisor as

$$Y = \begin{cases} 1 & \text{if recession} \\ 0 & \text{if growth} \end{cases}$$

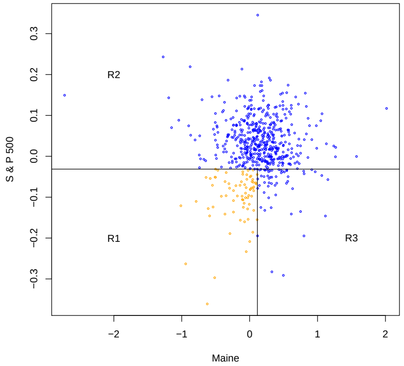
Dendrogram view



Terminology

- We call each split or end point a node. Each terminal node is referred to as a leaf
    - This tree has 2 interior nodes and 3 terminal nodes.
- The interior nodes lead to branches.
    - This graph has two main branches (the S&P 500 split).
- Interpret the plot as "Left if true" and "right if false"

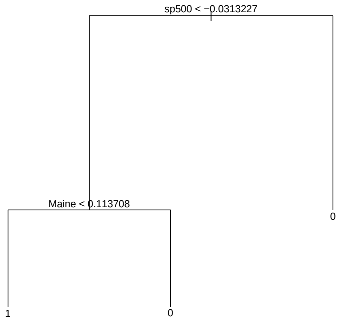
Partitioning view

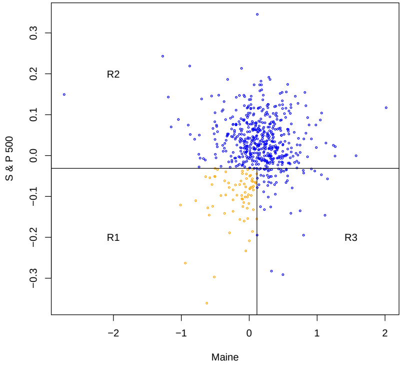


Notes

- We classify all observations in a region the same.
- The three regions R1, R2, and R3 are the leaves of the tree.

Tree





We can interpret this as

- S&P 500 is the most important variable.
- If S&P 500 is large enough, then we predict no recession.
- If S&P 500 is small enough, then we need to know the economic growth of Maine.

How do we build a tree?

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

   (this is done via greedy, binary splitting)
2. Every observation that falls into a given region R_m is given the same prediction
   - Regression: The average of the supervisors for a region
   - Classification: Determined by majority (or plurality) vote in that region

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.
- The tree stops splitting when there are too few observations in a terminal node

Regression trees

Implicit model

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

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

We need to find good values for M, $(R_m)_{m=1}^M$, and $(c_m)_{m=1}^M$

Generally, searching over all possible regions is infeasible

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

So we use a greedy approach instead

Regression trees

Define the two half-planes

$$r_1(j,s) = \{X | x_j \leq s\} \quad \text{and} \quad r_2(j,s) = \{X | x_j > s\}$$

For squared error loss, we solve

$$\min_{j,s} \left[\min_{c_1} \sum_{X_i \in r_1(j,s)} (Y_i - c_1)^2 + \min_{c_2} \sum_{X_i \in r_2(j,s)} (Y_i - c_2)^2 \right]$$

This generates, for $n_k = \sum_{i=1}^n \mathbf{1}(X_i \in r_k)$,

$$\hat{c}_k = n_k^{-1} \sum_{i:X_i \in r_k} Y_i$$

The next splits will be conditional on the minimizing $\hat{s}$

Classification trees

Classification trees

For a given region R_m and class g, define training proportions

$$\begin{aligned}\hat{p}_{mg}(X) &= \mathbf{1}(X \in R_m) n_m^{-1} \sum_{i:X_i \in R_m} \mathbf{1}(Y_i = g) \\ &= \begin{cases} \frac{1}{n_m} \sum_{i:X_i \in R_m} \mathbf{1}(Y_i = g) & \text{if } X \text{ is in } R_m \\ 0 & \text{if } X \text{ is } \textbf{not} \text{ in } R_m \end{cases}\end{aligned}$$

Our classification is

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

This presumes a given partition (R_m)

To estimate the partition we need a loss function

How do we measure quality of fit?

There are many possibilities:

Classification error rate:	$ER = 1 - \max_g(\hat{p}_{mg})$
Gini index:	$GI = \sum_g \hat{p}_{mg}(1 - \hat{p}_{mg})$
cross-entropy:	$CE = -\sum_g \hat{p}_{mg} \log(\hat{p}_{mg})$

(Cross-entropy is the multinomial log likelihood)

We build a classifier by growing a tree that greedily minimizes one of these criteria

We would like the tree to have pure nodes, in the sense that

$$\max_g \hat{p}_{mg} \approx 1$$

How do we measure quality of fit?

The m^{th} node is split by minimizing ER, GI, or CE over all

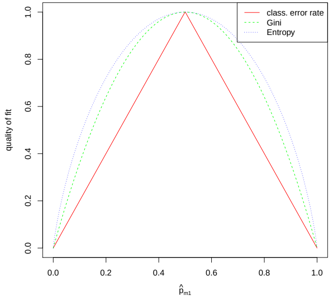
- features
- split points of those feature

such that the loss function is minimized over the 2 new regions

What do these loss functions look like?

How do we measure quality of fit?

Example: Suppose $G = 2$. Then $\hat{p} = \hat{p}_{m1} = 1 - \hat{p}_{m2}$



Generally, Gini index or Cross-entropy are preferred

(Note: they penalize values of $\hat{p}$ far from 0 or 1 more severely)

How do we measure quality of fit?

Additionally, Gini index or Cross-entropy are preferred as..

- They are differentiable everywhere
- They are amenable to treating qualitative features with a large number of levels in a computationally efficient way

  (There are $2^{L-1} - 1$ possibilities for L levels)

  (See ESL 9.2.4)
- The Gini index can be interpreted in suggestive ways

  (See ESL 9.2.3)

How do we measure quality of fit?

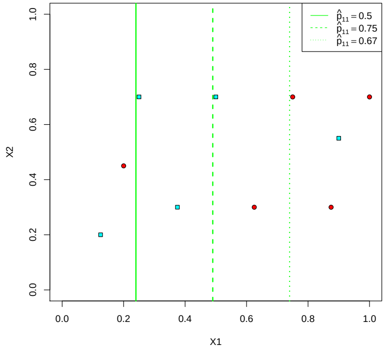
Example: $G = p = 2$ and we want to make the first split

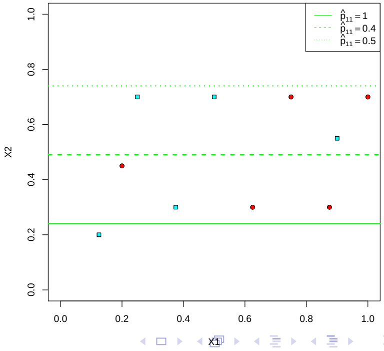
(For simplicity, we only look at the loss function at one side of the split)

Then $\hat{p}_{11} = 1 - \hat{p}_{12}$

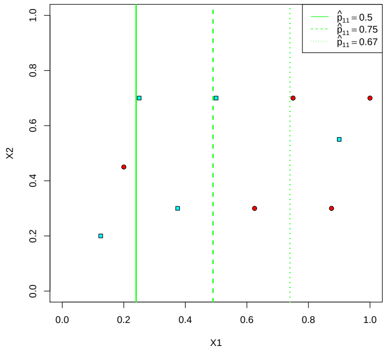
(Define the 'left' or 'bottom' region as R_1)

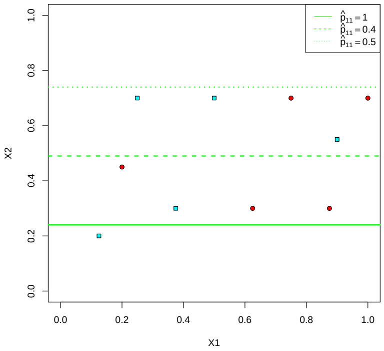
Let's look at some possible splits:





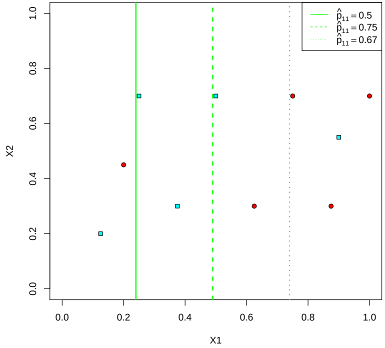
How do we measure quality of fit?

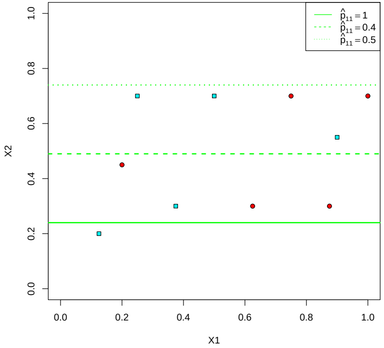




Where would we split?

How do we measure quality of fit?





Where would we split if we required ≥ 2 observations in a node for *GI* loss?

Overfitting

We have defined a loss function and have iteratively minimized the training error with respect to that loss function

It should be no surprise that following this procedure indefinitely will overfit

Overfit trees mean they have too many leaves

To stretch the analogy further, trees with too many leaves must be pruned

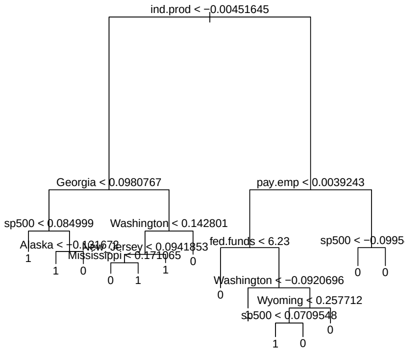
Pruning the tree

- Using weakest link pruning we can obtain a sequence of tree solutions ranging from a tree with no splits to the maximally complex tree T_0.
- This sequence is a function of a tuning parameter $\lambda \geq 0$
  (The book uses α, but I use λ to connect it to previous tuning parameters)
- Weakest link pruning: For some loss function ℓ

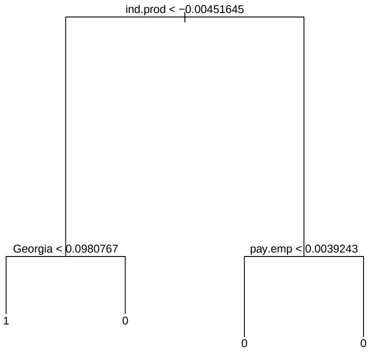
$$\sum_{i=1}^{n} \ell(Y_i, \hat{g}(X_i)) + \lambda |T|$$

  ($|T|$ is the number of terminal nodes or complexity of the tree T)
- Essentially, we are trading training fit (first term) with model complexity (second term)
- Now, cross-validation can be used to pick λ.

Results of trees on recession data

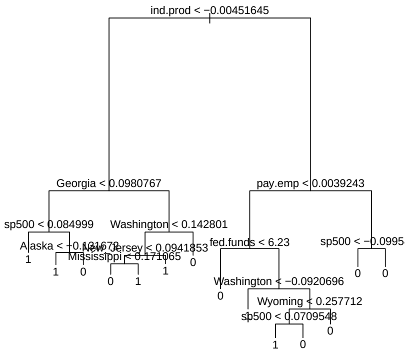


Unpruned tree

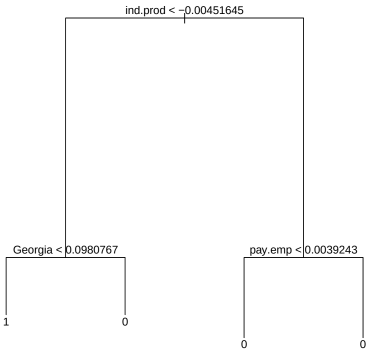


Pruned Tree

Results of trees on recession data



Unpruned tree

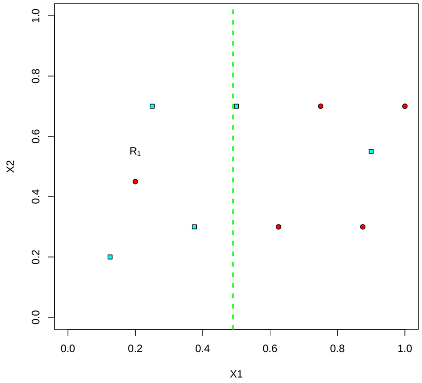


Pruned Tree

The pruned tree is a subset of the unpruned tree (nested)

There are splits that result the same prediction. Why?

Splits with same prediction



Suppose we split at vertical, dashed line. Then $\hat{p}_{11} = 0.75$.

What happens if we were to now split R_1 at $X2 = 0.5$?

Trees in R

Create a basic, unpruned tree:

```
require(tree)
out.tree = tree(Y~.,data=X,split='gini')
plot(out.tree)
text(out.tree)
```

(There is also the rpart package as well)

Trees in R

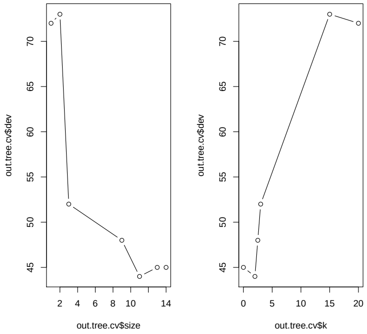
Prune the tree via cross-validation

```
out.tree.orig = tree(Y~.,data=X)
out.tree.cv   = cv.tree(out.tree.orig,FUN=prune.misclass)
> names(out.tree.cv)
[1] "size"   "dev"    "k"      "method"
```

Trees in R

Prune the tree via cross-validation

```
plot(out.tree.cv$size,out.tree.cv$dev,type="b")
plot(out.tree.cv$k,out.tree.cv$dev,type="b")
```



Note:

k corresponds to λ in weakest-link pruning.

dev means missclassifications in cv.tree

Trees in R

Prune the tree via cross-validation

```
best.size  = out.tree.cv$size[which.min(out.tree.cv$dev)]
> best.size
[1] 11
out.tree   = prune.misclass(out.tree.orig,best=best.size)
class.tree = predict(out.tree,X_0,type='class')
```

An Introductory Example

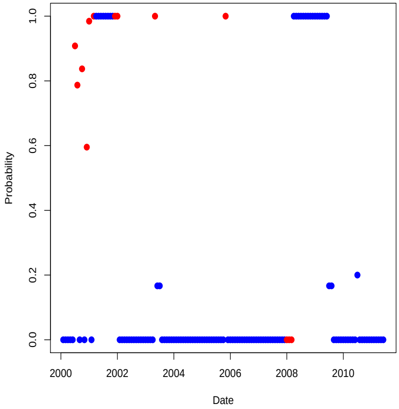
Example: Use macroeconomic data to predict recessions

We will use data from 1960 through 1999 as training data

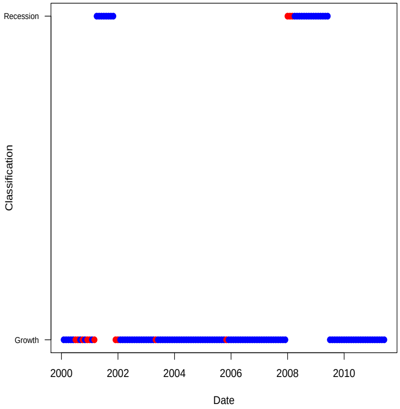
We will use data from 2000 through 2011 as testing data

(In the following plots: correct vs. incorrect classifications)

Results of trees on recession data



Posterior probability of prediction



Predictions

Advantages and disadvantages of trees

+ Trees are very easy to explain (much easier than even linear regression).
+ Some people believe that decision trees mirror the human decision-making process.
+ Trees can easily be displayed graphically no matter the dimension of the data.
+ Trees can easily handle qualitative features 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.