Classification via Trees

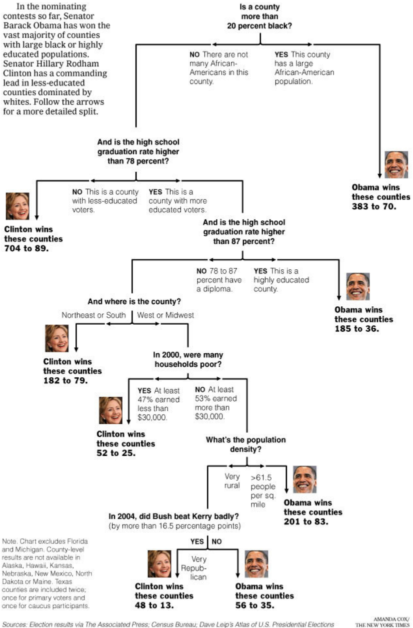
-Statistical Machine Learning-

Lecturer: Darren Homrighausen, PhD

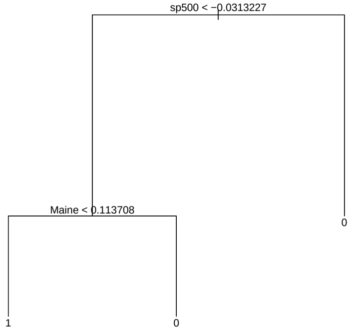
What is a (decision) tree?

- Trees involve stratifying or segmenting the predictor 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.

Example tree



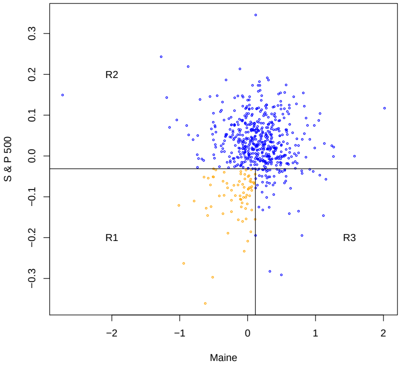
Dendrogram view



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).

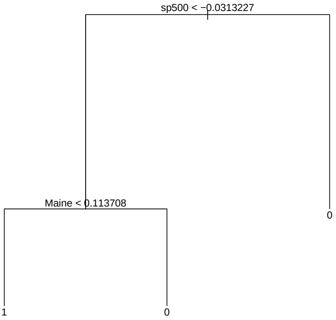
Partitioning view

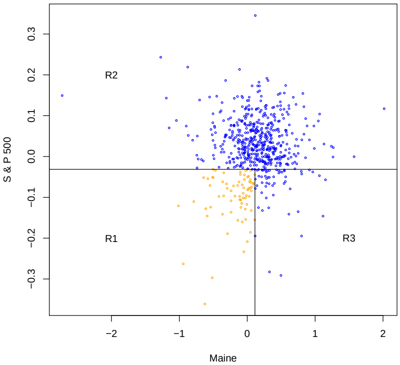


Notes

- 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 change in the employment level of Maine.

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

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

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

We need to estimate both (R_m) and (c_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 $n_k = \sum_{i=1}^{n} \mathbf{1}_{r_k}(X_i)$,

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

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

Classification trees

Classification trees

For a given partition R_m and class g, define 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

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

This presumes a given partition (R_m). This must be estimated

For this, we need a loss function

How do we measure quality of fit?

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

There are many possibilities:

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

(Cross-entropy is also known as deviance)

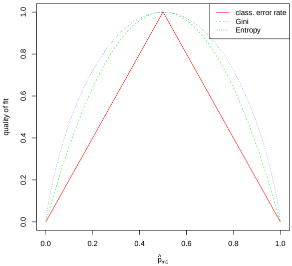
We build a classifier by growing a tree that greedily minimizes one of these criteria

How do we measure quality of fit?

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

The m^{th} node is made by minimizing E, G, or D over all

- Features
- split points of that feature



Generally, Gini index or Cross-entropy is preferred

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

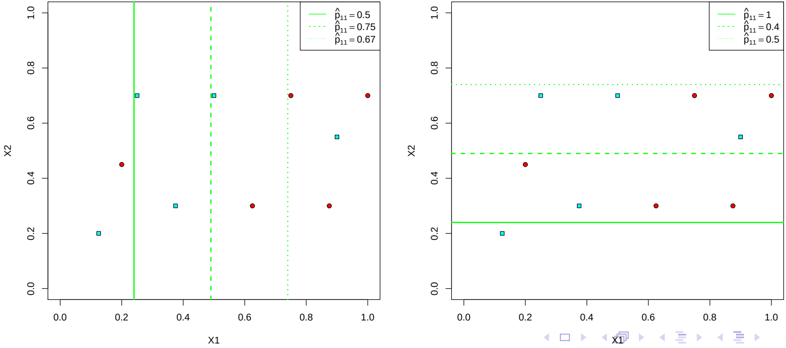
How do we measure quality of fit?

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

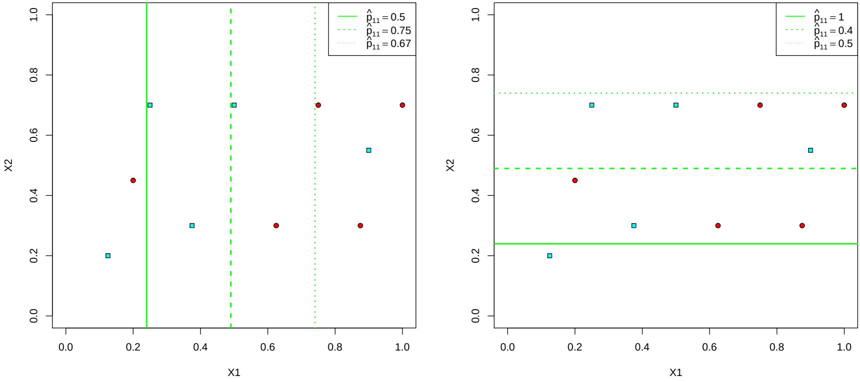
Then $\hat{p}_{11} = 1 - \hat{p}_{12}$

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

Let's look at some possible splits:

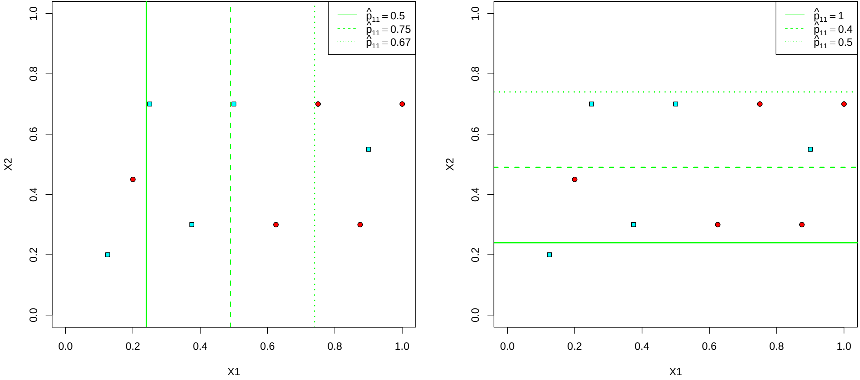


How do we measure quality of fit?



Where would we split?

How do we measure quality of fit?

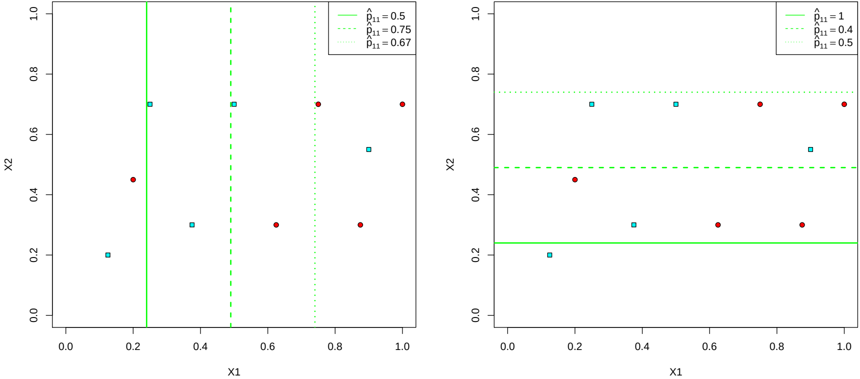


Where would we split?

For E and G, at the solid, horizontal line

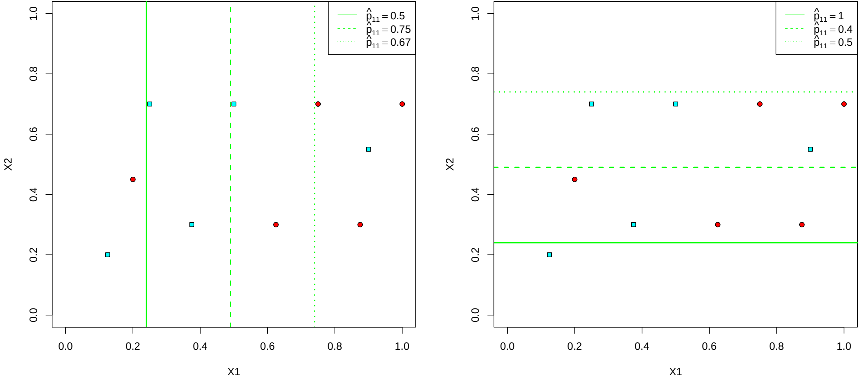
$(\hat{p}_{11} = 1 \Rightarrow E = 0, G = 20/81)$

How do we measure quality of fit?



Where would we split if we required ≥ 2 observations in a node?

How do we measure quality of fit?



Where would we split if we required ≥ 2 observations in a node?

(At the dashed, vertical line for E. At either dashed or dotted vertical line for G)

There's a problem

Following this procedure overfits!

- The process described so far will fit overly complex trees, leading to poor predictive performance.
- Overfit trees mean they have too many leaves.
- To stretch the analogy further, trees with too many leaves must be pruned.

Pruning the tree

- Cross-validation can be used to directly prune the tree, but it is far too expensive (computationally) to use in practice (combinatorial complexity)
- Instead, we use weakest link pruning

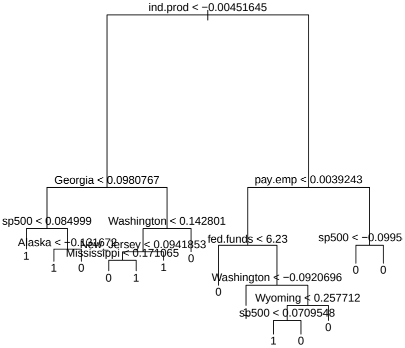
$$\sum_{m=1}^{|T|} \sum_{i \in R_m} \mathbf{1}(Y_i \neq \hat{Y}_{R_m}) + \lambda |T|$$

  where $|T|$ is the number of terminal nodes.

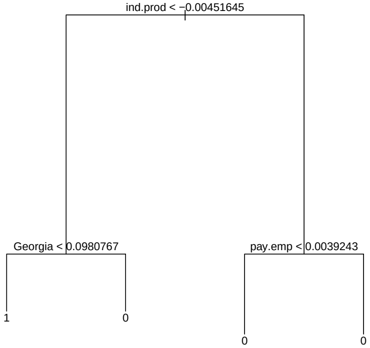
  Essentially, we are trading training fit (first term) with model complexity (second term)

  (compare to lasso)
- Now, cross-validation can be used to pick λ.

Results of trees on recession data

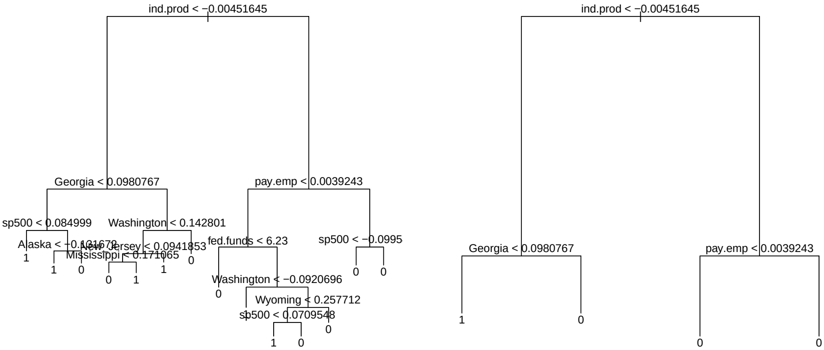


Unpruned tree



Pruned Tree

Results of trees on recession data

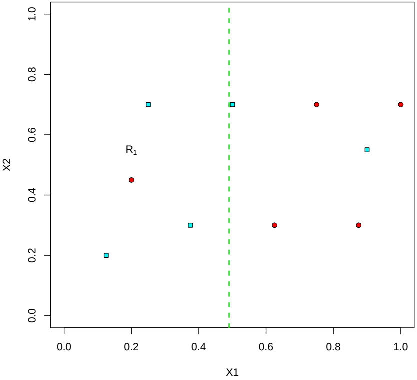


Unpruned tree | Pruned Tree

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

There are splits that result in having 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)
```

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

```
> out.tree.cv
$size
[1] 14 13 11  9  3  2  1

$dev
[1] 45 45 44 44 44 64 67

$k
[1] -Inf  0.0  2.0  2.5  3.0 15.0 20.0

$method
[1] "misclass"
```

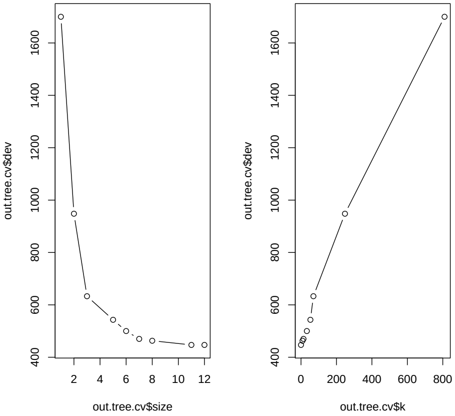
Note:

k corresponds to λ in weakest-link pruning.

dev means missclassifications in cv.tree

Cross validation plots

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



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

Use macroeconomic data to predict recessions

Use handful of national-level variables – Federal Funds Rate, Term Spread, Industrial Production, Payroll Employment, S&P500

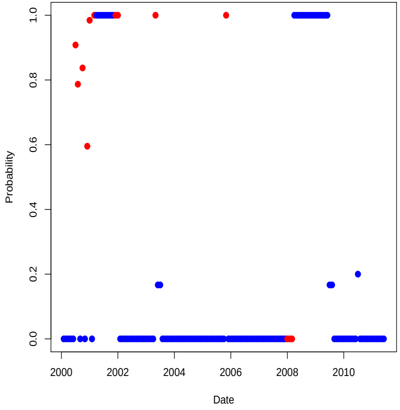
Also include state-level Payroll Employment

In this example, we code $Y = 1$ as a recession and $Y = 0$ as growth.

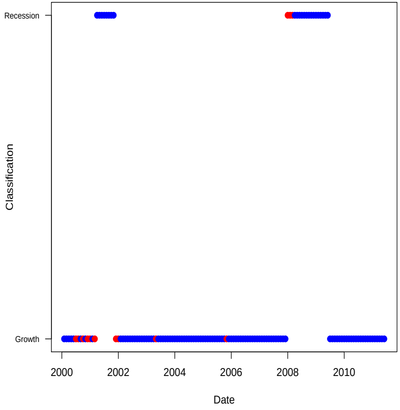
We will use data from 1960 through 1999 as training data

We will use data from 2000 through 2011 as testing data

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 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.