STA 314: Statistical Methods for Machine Learning I

Lecture - Tree-based approaches: decision trees

Xin Bing

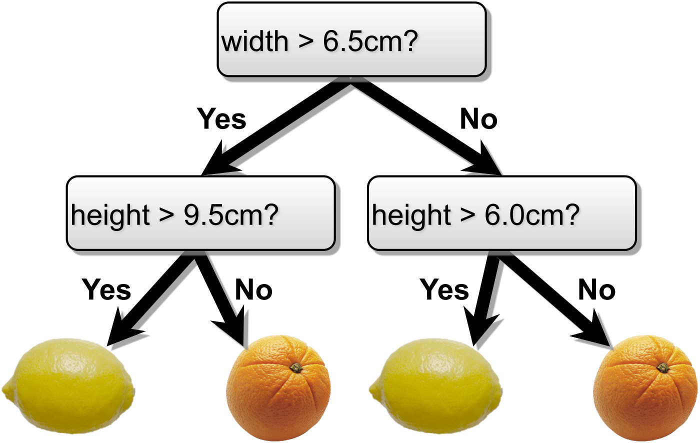
Department of Statistical Sciences
University of Toronto

Tree-Based Methods

- Tree-based method can be applied for both regression and classification.
- Since the method can be summarized in a tree structure, these types of approaches are known as **decision tree methods**.
- Tree-based methods are simple and useful for interpretation.
- However, they typically are not competitive with the best supervised learning approaches.
- We will (later) introduce **bagging**, **random forests**, and **boosting** to combine multiple decision trees to improve the performance.

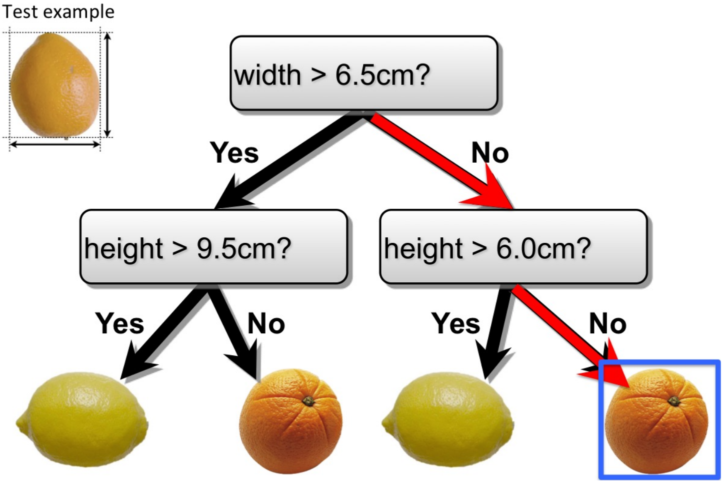
Example of Decision Trees

- Make predictions by splitting on features according to a tree structure.



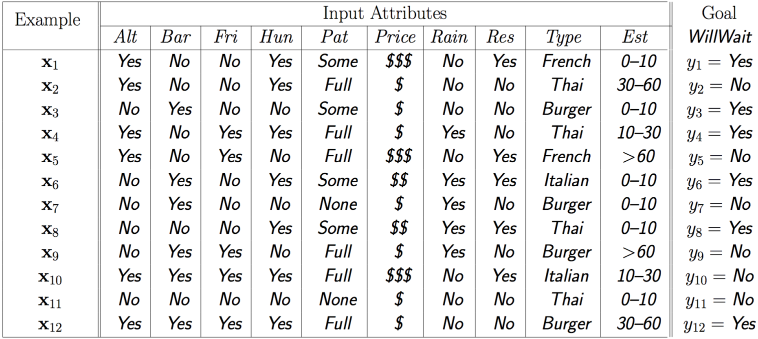
Example of Decision Trees

- Make predictions by splitting on features according to a tree structure.



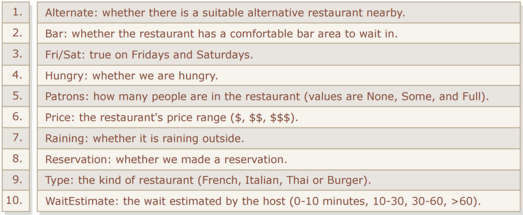
Example of Decision Trees—Discrete features

First, what if features are discrete?



Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

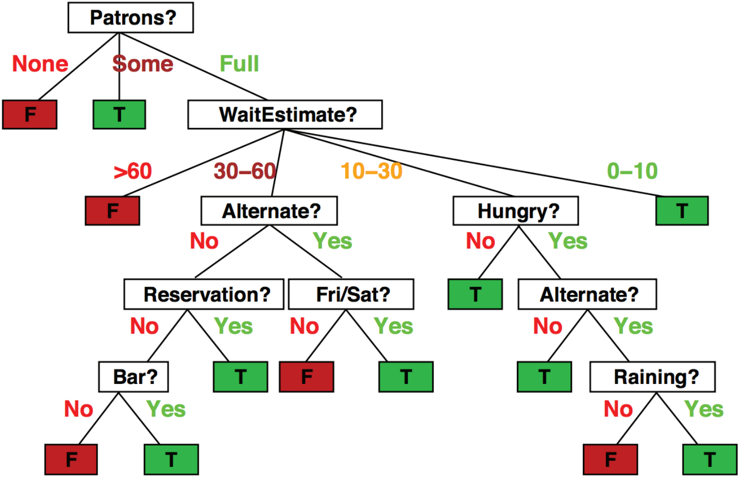
Features:



1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai or Burger).
10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

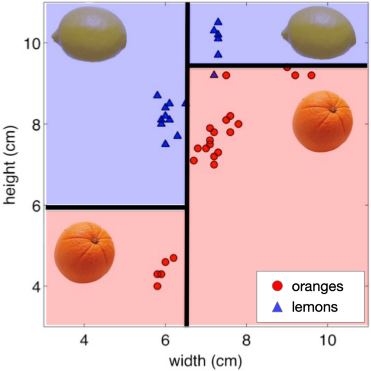
Example of Decision Trees—Discrete features

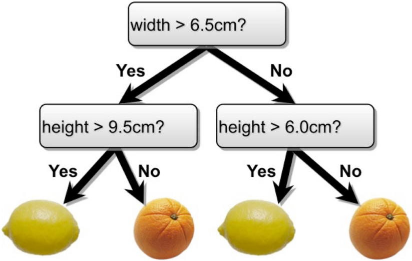
- Split *discrete features* into a partition of possible values.



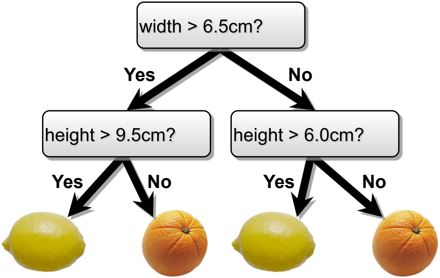
Example of Decision Trees—Continuous features

- For *continuous features*, we partition the range by checking whether that attribute is greater than or less than some threshold.
- Decision boundary is made up of rectangles.





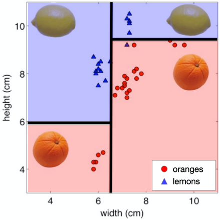
Terminology of Decision Trees



- An **internal node** means an **attribute**, i.e., a feature.
- **Branching** is determined by the **attribute value**.
- Children of a node partition the range of the attribute from the parent.
- **Leaf nodes** are **predicted outputs**.

Decision Trees—Classification and Regression

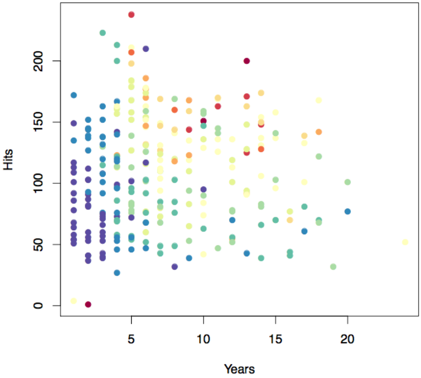
- Let J be the total number of paths from root to a leaf.
- Each path, $j \in \{1, 2, \ldots, J\}$, defines a region R_j of the feature space.
- Let $\{(x^{(j_1)}, y^{(j_1)}), \ldots, (x^{(j_k)}, y^{(j_k)})\}$ be the training data points that fall into R_j



- **Regression tree**:
  - continuous output
  - leaf value y^{R_j} typically set to the mean value in $\{y^{(j_1)}, \ldots, y^{(j_k)}\}$
- **Classification tree**:
  - discrete output
  - leaf value y^{R_j} typically set to the most common value in $\{y^{(j_1)}, \ldots, y^{(j_k)}\}$

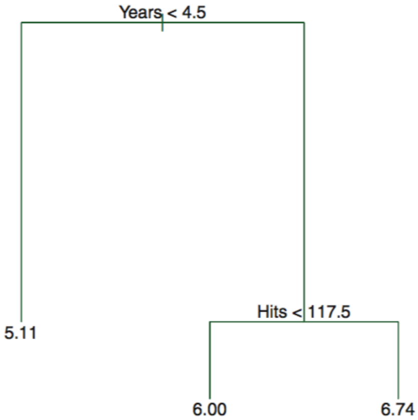
Regression Decision Trees

Consider the baseball salary data (Hitters data): We want to predict the salary of a baseball player, based on (1) the number of years that he has played in the leagues, and (2) the number of hits that he made in the previous year.



Salary is color-coded from low (blue, green) to high (yellow, red)

What does a fitted decision tree look like?



At a given internal node, the label (of the form $X_j < t$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t$. The number in each leaf (external node) is the mean of the response for the observations that fall there.

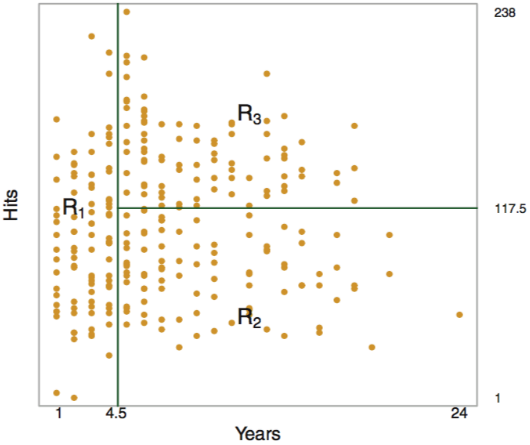
Partitioned regions for the Hitters data set

Overall, the tree stratifies or segments the players into three regions of feature space:

$$R_1 = \{X \mid Years < 4.5\}$$

$$R_2 = \{X \mid Years \geq 4.5, \\ Hits < 117.5\}$$

$$R_3 = \{X \mid Years \geq 4.5, \\ Hits \geq 117.5\}.$$



Interpretation of Results

- Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players.

- Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary.

- But among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary, and players who made more Hits last year tend to have higher salaries.

- Surely an over-simplification, but compared to a regression model, it is easier to interpret, and has a nice graphical representation.

How to Build a Regression Tree?

- Step 1: We divide the feature space, i.e., the set of possible values for $X_1, X_2, \ldots, X_p$, into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
- Step 2: For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

How to Construct Regions $R_1, R_2, \ldots, R_J$?

- In theory, the splitted regions could have any shape. However, we choose to divide the feature space into multi-dimensional rectangles for ease of interpretation of the resulting predictive model.

- The goal is to find rectangles $R_1, R_2, \ldots, R_J$ that minimize the residual sum of squares (RSS), given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2,$$

where

$$\bar{y}_{R_j} = \frac{1}{|R_j|} \sum_{i \in R_j} y_i$$

is the mean response for the training observations in R_j.

How to Construct Regions $R_1, R_2, \ldots, R_J$?

- Unfortunately, it is computationally infeasible to enumerate all possible partitions of the feature space into J boxes.
- For this reason, we take a **top-down, greedy** approach that is known as **recursive binary splitting**.
- The approach is **top-down** because it begins at the top of the tree and then successively splits the feature space; each split is indicated via two new branches further down on the tree.
- It is **greedy** because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

How to Construct Regions $R_1, R_2, \ldots, R_J$?

- We first select the feature X_j among all $X_1, \ldots, X_p$, and the cutpoint s such that splitting the feature space into the regions $\{X | X_j < s\}$ and $\{X | X_j \geq s\}$ leads to the greatest possible reduction in RSS.

- Mathematically, we seek the value of $j \in \{1, 2, \ldots, p\}$ and $s \in \mathbb{R}$ that minimize the equation

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \bar{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \bar{y}_{R_2})^2$$

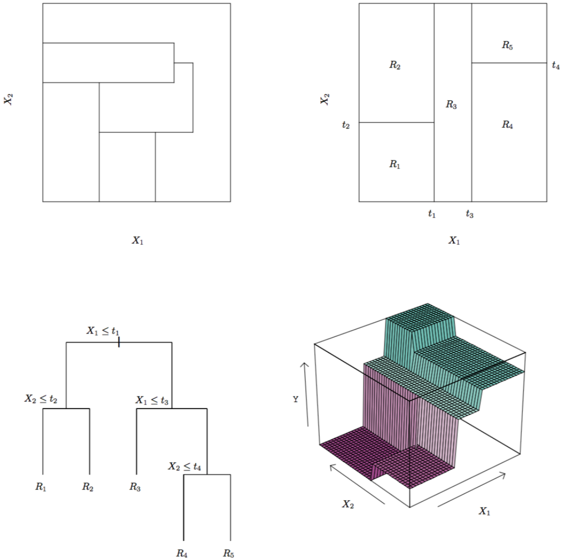
  where

  - $R_1(j, s) = \{X | X_j < s\}$ and $R_2(j, s) = \{X | X_j \geq s\}$
  - $\bar{y}_{R_1}$ and $\bar{y}_{R_2}$ are, respectively, the averaged responses of the training data in $R_1(j, s)$ and $R_2(j, s)$.

How to Construct Regions $R_1, R_2, \ldots, R_J$?

- Next, we repeat the process, looking for the best feature and the best cutpoint in order to **further split** the data, leading to the greatest reduction of RSS.
- However, this time, instead of splitting the entire feature space, we split one of the two previously identified regions. We now have three regions.
- The process continues until a stopping criterion is reached
  - we may stop when no region contains more than 5 observations
- Once $R_1, R_2, \ldots, R_J$ are identified, we predict the response for a given test point using the averaged response of the training observations in the region.

An Example with 5 Regions



An Example with 5 Regions

- Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting.
- Top Right: The output of recursive binary splitting on a two-dimensional example.
- Bottom Left: A tree corresponding to the partition in the top right panel.
- Bottom Right: A perspective plot of the prediction surface corresponding to that tree.

Tree Pruning

- The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance.
  - Consider the tree that has one observation per region.
- A smaller tree with fewer splits (that is, fewer regions $R_1, R_2, ..., R_J$) might lead to lower variance and better interpretation at the cost of a little bias.
- Intuitively, our goal is to select a subtree that leads to the lowest test error rate. However, estimating the cross-validation error for every possible subtree is not feasible, since the number of possible subtrees is large.

Tree Pruning

- A common strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree. We use the **cost complexity pruning**, a.k.a. **weakest link pruning**.

- We consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α find a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \bar{y}_{R_m})^2 + \alpha |T|,$$

  is minimized. Here $|T|$ indicates the number of terminal nodes of the tree T.
    - The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
    - $\alpha = 0$ gives T_0.

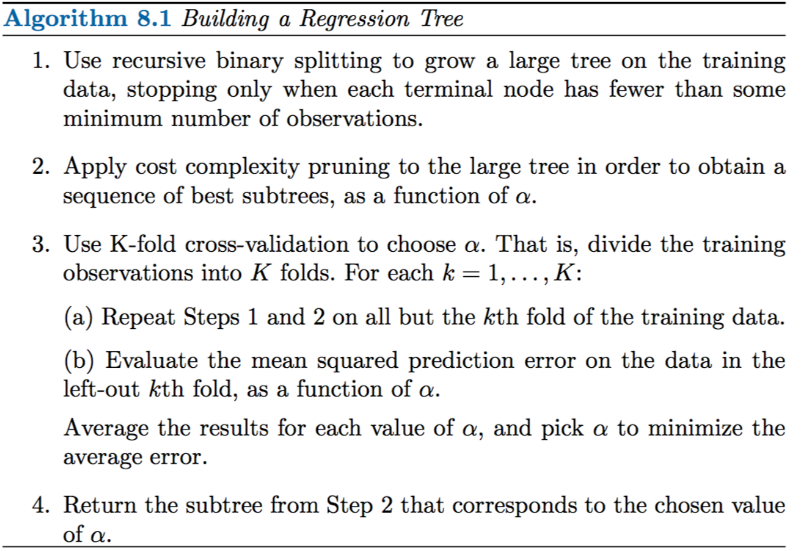
Tree Pruning

- It turns out that as we increase α from 0, branches get pruned from the tree in a nested and predictable fashion, so obtaining the whole sequence of subtrees as a function of α is easy.

- We can select a value of α using a validation set or using cross-validation.

- We then return to the full data set and obtain the subtree corresponding to α.

Algorithm for Building a Regression Tree



Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

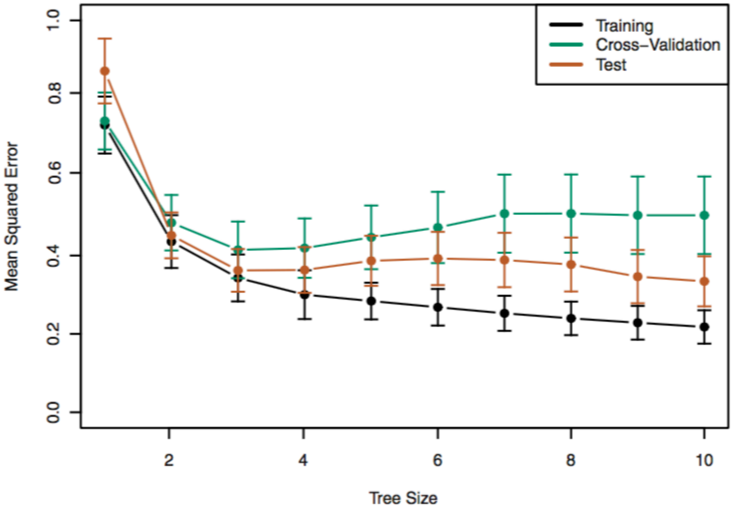
   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

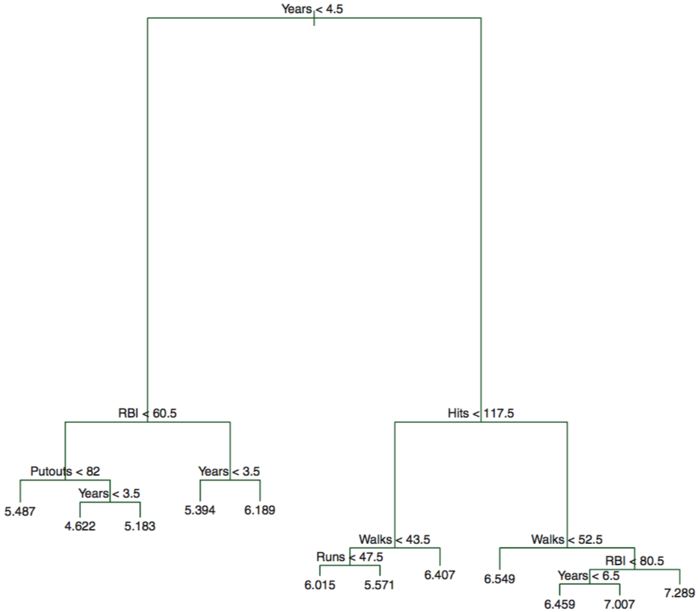
Baseball Example

- First, we randomly divided the data set in half, yielding 132 observations in the training set and 131 observations in the test set.
- We then built a large regression tree on the training data and varied α to create subtrees with different numbers of terminal nodes.
- Finally, we performed six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of α.

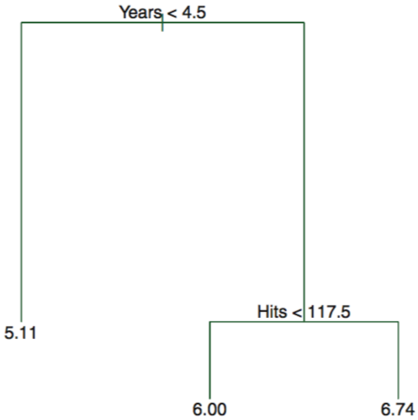
Hitters data set



Hitters data set: the unpruned tree



Hitters data set: the pruned tree



Classification Trees

- A classification tree is very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.
- Just as in the regression setting, we use **recursive binary splitting** to grow a classification tree.
- But, RSS cannot be used as a criterion for making the binary splits. What metric should we use instead?

Metrics used for classification trees

- A natural alternative to RSS is the **0-1 misclassification error rate**. For region R_m with K classes of labels: for $k \in \{1, \dots, K\}$,

$$\hat{p}_{mk} = \frac{1}{|R_m|} \sum_{i:x_i \in R_m} 1\{y_i = k\}$$

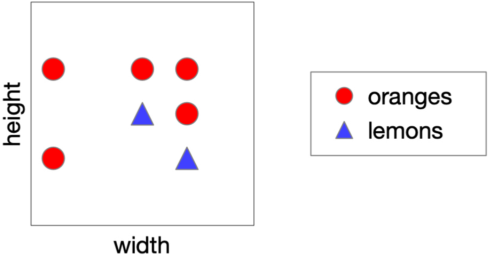
  The misclassification error rate in region R_m is

$$1 - \max_{k \in \{1,\dots,K\}} \hat{p}_{mk}.$$

- However misclassification error is not sufficiently sensitive for the change of $\hat{p}_{mk}$, hence not suitable for finding the splits for **building** classification trees.

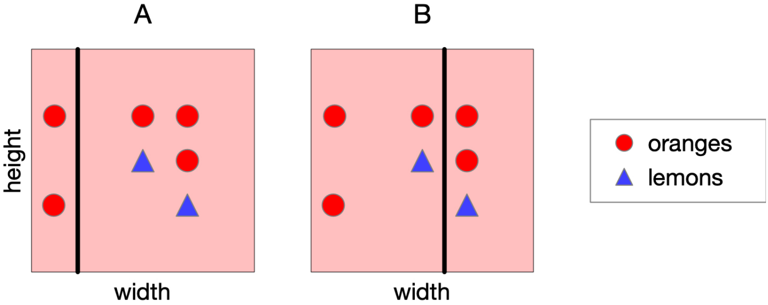
Choosing a Good Split

- Consider the following data. Let's split on width.



Choosing a Good Split

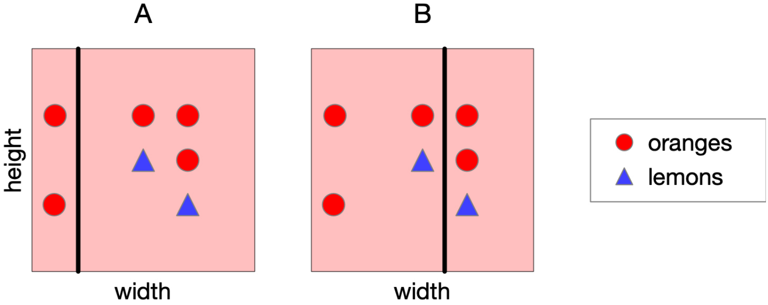
- Recall: classify by majority.



- A and B have the same misclassification rate, so which is the best split? Vote!

Choosing a Good Split

- Case A feels like a better split, because the left-hand region is very certain about whether the fruit is an orange.



- Can we quantify this?

Quantifying Uncertainty

- The **entropy** of a discrete random variable is a number that quantifies the **uncertainty** inherent in its possible outcomes.
- The mathematical definition of entropy that we give in a few slides may seem arbitrary, but it can be motivated axiomatically.
  - If you're interested, check: *Information Theory* by Robert Ash.
- To explain entropy, consider flipping two different coins...

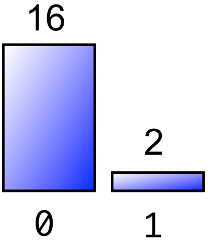
We Flip Two Different Coins

Sequence 1:

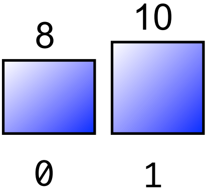
```
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?
```

Sequence 2:

```
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?
```



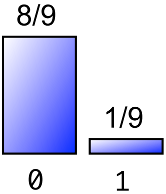
versus



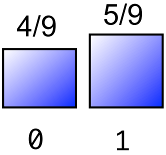
Quantifying Uncertainty

- The entropy of a loaded coin with probability p of heads is given by

$$-p\log_2(p) - (1-p)\log_2(1-p)$$



$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

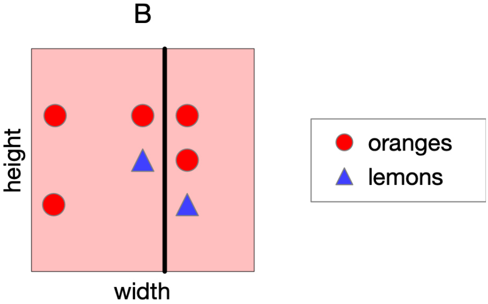


$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

- Notice: the coin whose outcomes are more certain has a lower entropy.
- In the extreme case $p = 0$ or $p = 1$, we were certain of the outcome before observing. So, we gained no certainty by observing it, i.e., entropy is 0.

Revisiting Our Original Example

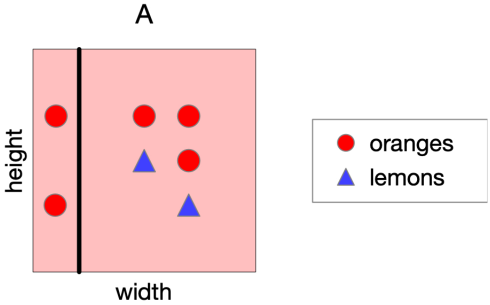
- Let's compute the entropies associated with this split



- Left branching: $\hat{p}_{orange} = 3/4$ and $\hat{p}_{lemon} = 1/4$. Its entropy is $-\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \approx 0.56$.
- Right branching: $\hat{p}_{orange} = 2/3$ and $\hat{p}_{lemon} = 1/3$. Its entropy is $-\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \approx 0.64$
- The summation of entropies is 1.2.

Revisiting Our Original Example

- What is the information gain of split A? Very informative!



- Left branching: $\hat{p}_{orange} = 1$ and $\hat{p}_{lemon} = 0$. Its entropy is 0.
- Right branching: $\hat{p}_{orange} = 3/5$ and $\hat{p}_{lemon} = 2/5$. Its entropy is $-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} \approx 0.67$
- The summation of entropies is 0.67. So we have less uncertainty for the second split. Uncertainty can also be understood as **node purity**.

Deviance and Gini index

- Entropy based metric is called **Deviance**, given by

$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk},$$

  for region R_m.

- Another popular metric is **Gini index**, defined by

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

  a measure of total variance across the K classes.

- Both Deviance and Gini index are small if all of the $\hat{p}_{mk}$'s are close to zero or one. They are measures of **node purity** – a small value indicates that a node contains observations that most of them come from a single, dominant class.

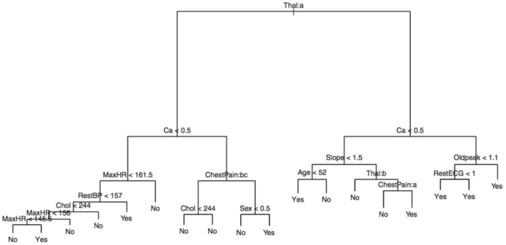
Misclassification error rate, Gini index and Deviance

- It turns out that Gini index and Deviance are quite similar numerically.
- On the one hand, for **building** a classification tree, either Gini index or Deviance can be used for finding best splits, since they are more sensitive to node purity than the misclassification error rate.
- On the other hand, all Gini index, Deviance and misclassification error can be used for **pruning** the tree, but the misclassification error is preferable if classification accuracy is the final goal.

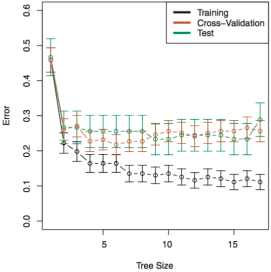
Heart Data

- These data contain a binary outcome HD for 303 patients who presented with chest pain.
- An outcome value of Yes indicates the presence of heart disease based on an angiographic test, while No means no heart disease.
- There are 13 features including Age, Sex, Chol (a cholesterol measurement), and other heart and lung function measurements.
- Cross-validation yields a tree with six terminal nodes.

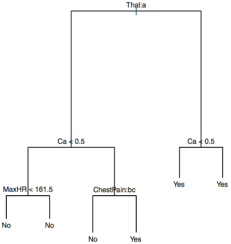
Heart Data



Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Slope < 1.5
Oldpeak < 1.1
RestBP < 157
Age < 52
Thal:b
RestECG < 1
Yes
Chol < 244
ChestPain:a
MaxHR < 156
No
Yes
Chol < 244
Sex < 0.5
Yes
No
No
Yes
Yes
MaxHR < 145.5
No
No
No
Yes
No
Yes
No
No
No
Yes



Training
Cross–Validation
Test
Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
Tree Size



Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Yes
Yes
No
No
No
Yes

Heart Data

- Top: The unpruned tree.
- Bottom Left: Cross-validation error, training, and test error, for different sizes of the pruned tree.
- Bottom Right: The pruned tree corresponding to the minimal cross-validation error.

Trees Versus Linear Models

- The linear model says

$$f(X) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j,$$

whereas the regression tree says

$$f(X) = \sum_{m=1}^{J} c_m \, 1\{X \in R_m\}.$$

- If the relationship between the features and the response is well approximated by a linear model, then an approach such as linear regression will likely work well. If instead there is a highly non-linear and complex relationship between the features and the response, then decision trees performs better.
- The relative performances of tree-based and classical approaches can be assessed by estimating the test error, using either cross-validation or the validation set approach.

Advantages and Disadvantages of Trees

- Trees are self-explanatory. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mimic human decision-making.
- Trees can easily handle qualitative features without the need to create dummy variables.
- But, trees generally do not have the same predictive power as other regression and classification approaches.
- Relatedly, trees could be very non-robust. In other words, a small change in the data can lead to a drastically different estimated tree.

Nevertheless, by aggregating many trees (via bagging, random forests, and boosting), the predictive performance can be substantially improved.