Decision Trees II

Dr. Alex Williams

August 26, 2020

THE UNIVERSITY OF TENNESSEE KNOXVILLE

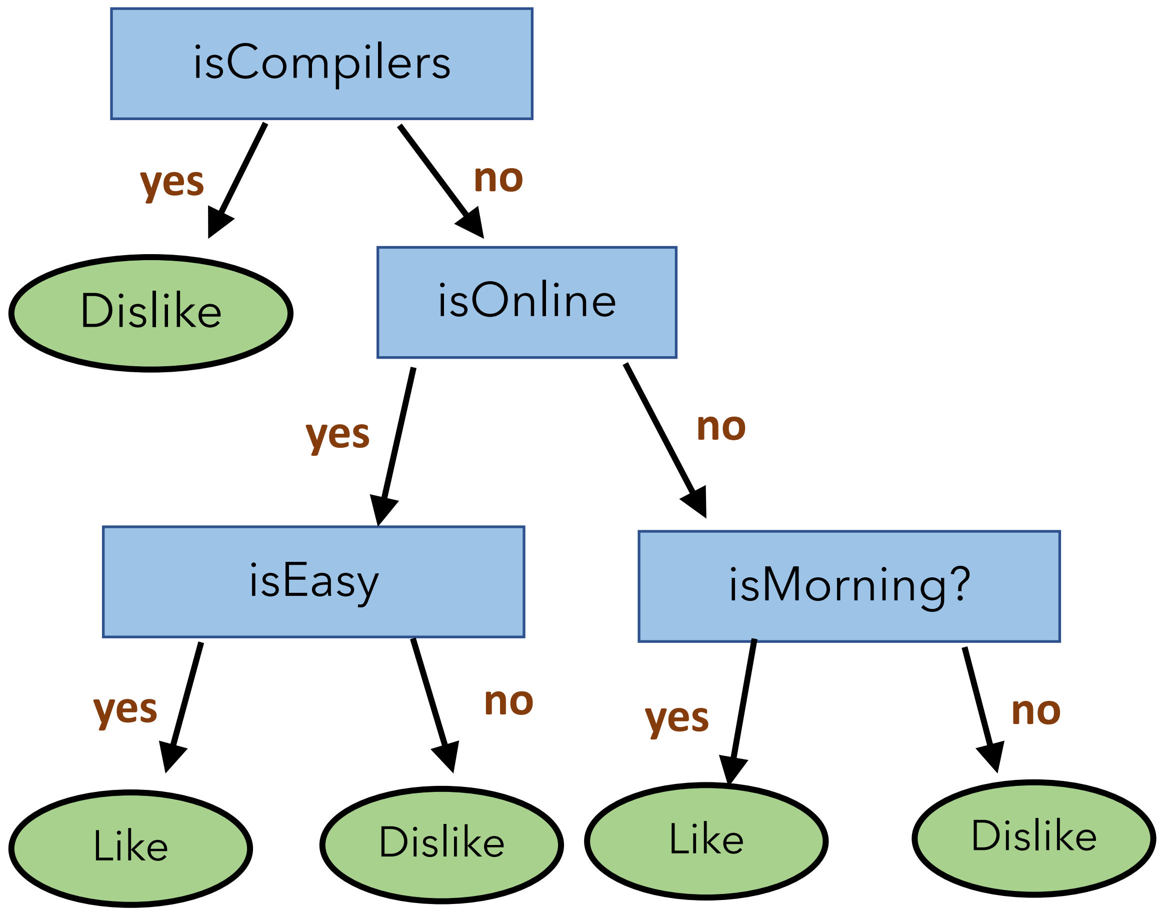
COSC 425: Introduction to Machine Learning
Fall 2020 (CRN: 44874)

Today's Agenda

We will address:

1. How do you train and test decision trees?
2. How can decision trees generalize?
3. What is the inductive bias of decision trees?

Refresher



Decision Tree Overview

1. **Questions → Trees**

   Problem: Asking the right questions.

2. **Terminology**

   Instance, Question, Answer, Label

3. **Finding the "Right" Tree**

   Informative / Uninformative Questions

4. **Boolean Functions**

   Trees ←→ If-Then Rules

5. **Decision Boundaries**

   Plotting trees in 2D space

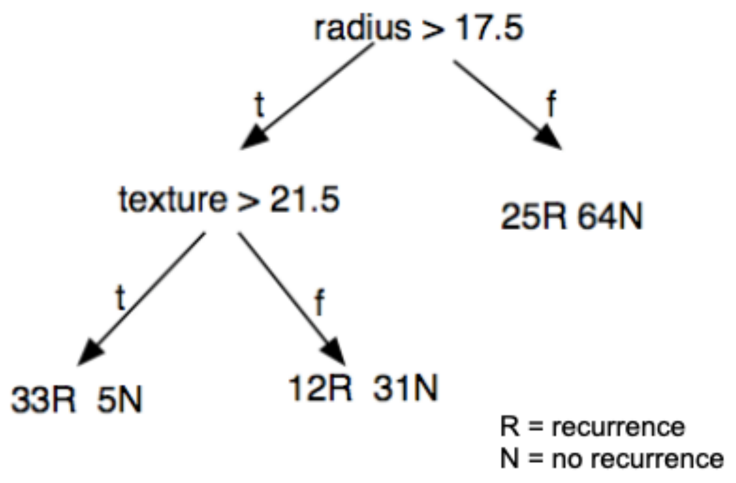
1. How do you train / test Decision Trees?

Decision Tree: Usage

Suppose we get a new instance …

radius = 16, texture = 12

How do we classify it?

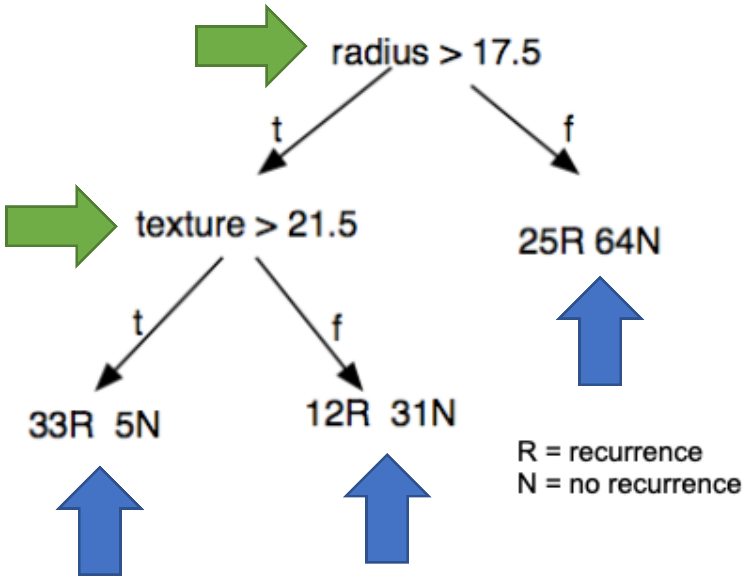


Procedure

- At every node, test the corresponding attribute.
- Follow the branch based on the test.
- When you reach a leaf, you have two options:
  1. Predict the class of the majority of examples at that test; or
  2. Sample from the probabilities of the two classes.

Decision Tree: Usage

```
DecisionTreeTest(tree, testPoint)
if tree IS leaf(guess) then:
        return guess
else if tree IS node(f, left, right) then:
    if f IS no in testPoint then:
        return DecisionTreeTest(tree, testPoint)
    else
        return DecisionTreeTest(tree, testPoint)
    end if
end if
```

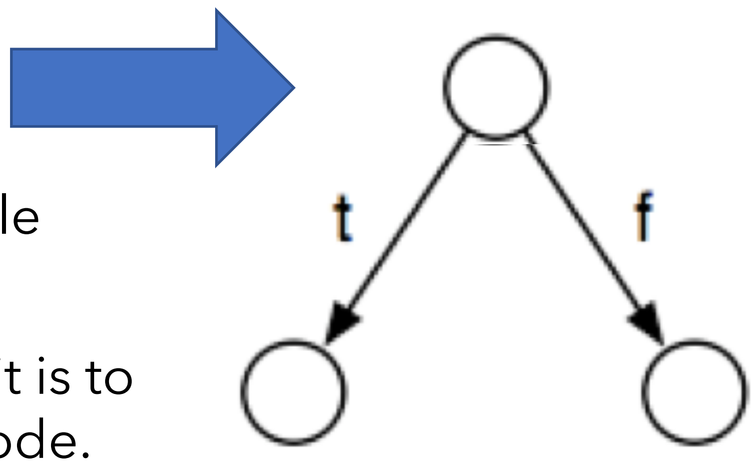


Note: Decision tree algorithms are generally variations of core top-down algos.
(See Quinlan's Programs for Machine Learning. 1993.)

Decision Tree: Training

Given a set of training instances (i.e. $<x_i, y_i>$)**, we build a tree.**

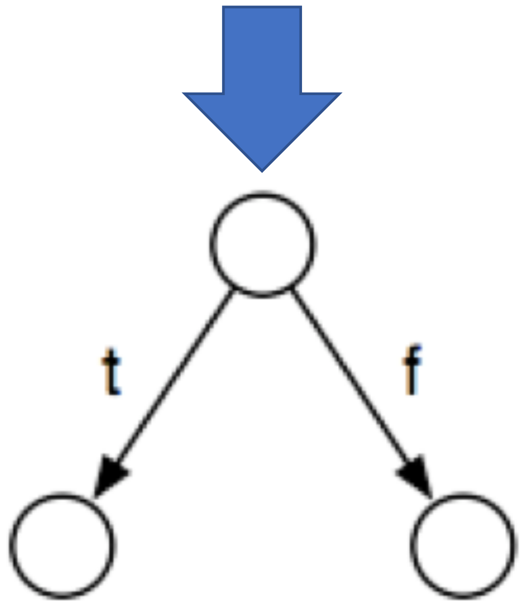
Let's say that this is our current node.



1. We iterate over all the features available in our current node. (Blue arrow)
2. For each feature, we test how "useful" it is to split on this feature from the current node.
   - This ***always*** produces two child nodes.

Decision Tree: Training

Let's say that this is our current node.



Given a set of training instances (i.e. $<x_i,y_i>$)**, we build a tree.**

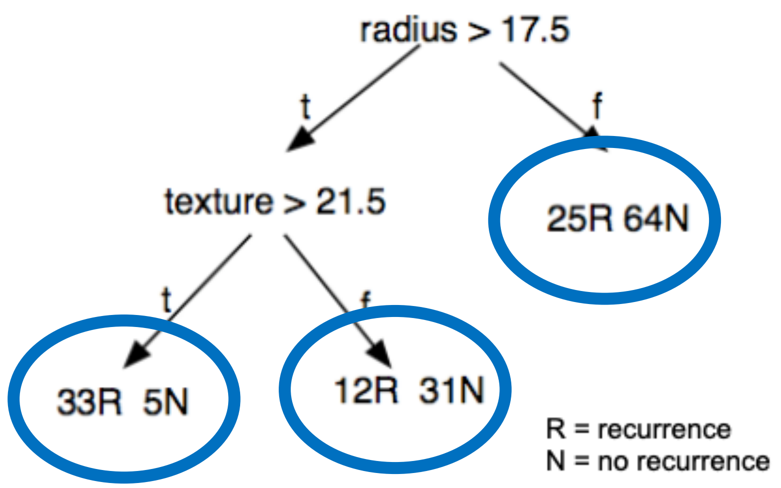
1. **Exit Condition:** If all training instances have the same class label (y_i), create a leaf with that class label and exit.
2. **Test Selection:** Pick the best test to split the data on.
3. **Splitting**: Split the training set according to the value of the outcome of the selected test from #2.
4. **Recurse:** Recursively repeat steps 1-3 on each subset of the training data.

Decision Tree: Training

1 Leaf Creation

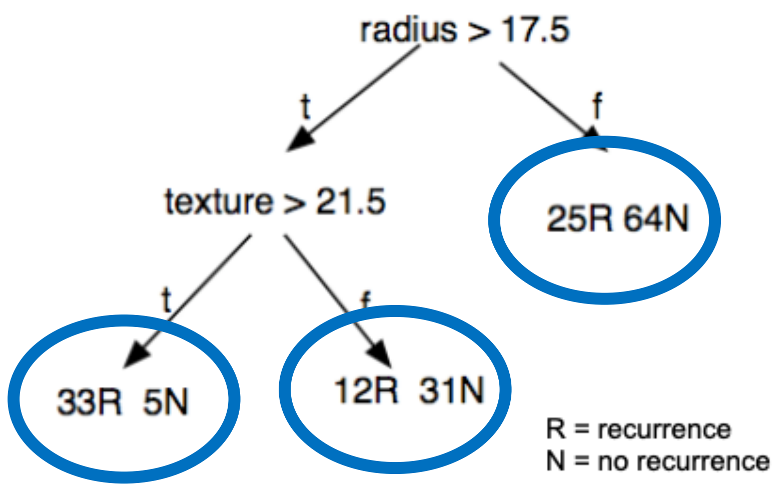
2 Splitting Criterion

```
DecisionTreeTrain(data, remainingFeatures)
    guess ← most frequent answer in data
    if labels in data IS ambiguous then:
        return LEAF(guess)
    else if remaining features IS empty then:
        return LEAF(guess)
    else:
        for all f IN remaining features do:
            NO ← the subset of data which f = no
            YES ← the subset of data which f = yes
            score ←  # of majority-vote answers in NO +
                     # of majority-vote answers in YES
        end for
    [ … Continued in Next Slide … ]
    end if
```



Decision Tree: Training

```
DecisionTreeTrain(data, remainingFeatures)
    guess ← most frequent answer in data
    [ … STEP 1 in Prior Slide… ]
    else:
        for all f IN remaining features do:                          2  Splitting Criterion
            NO ← the subset of data which f = no
            YES ← the subset of data which f = yes
            score ← # of majority-vote answers in NO +
                    # of majority-vote answers in YES
        end for
        f ← the feature with the maximal score (f)                   3  Split Selection
        NO ← the subset of the data on which f = no
        YES ← the subset of the data on which f = yes.
        left ← DecisionTreeTrain(NO, remaining features / { f })
        right ← DecisionTreeTrain(YES, remaining features / { f })
        Return NODE(f, left, right)
    end if
```



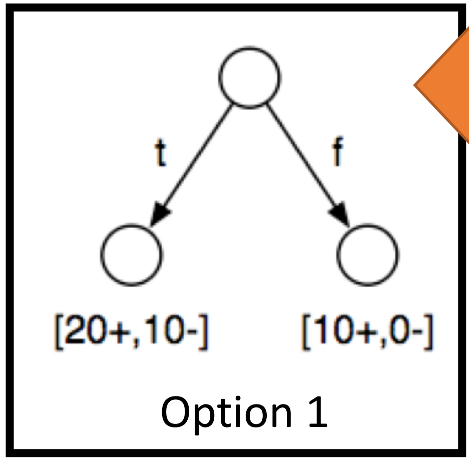
Decision Tree: Training

What makes a good test?

A "good" test provides information about the class label.

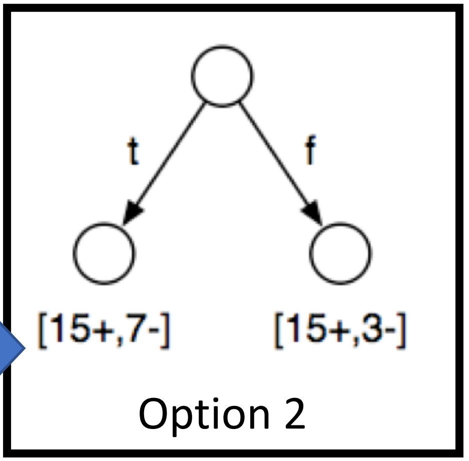
Example: Say that you're given 40 examples. (30 Positive & 10 Negative)

Consider two tests that would split the examples as follows:



All negatives bucketed to t, with some division for positives.

Positives split evenly with negatives being more even, too.



Decision Tree: Training

What makes a good test?

A “good” test provides information about the class label.

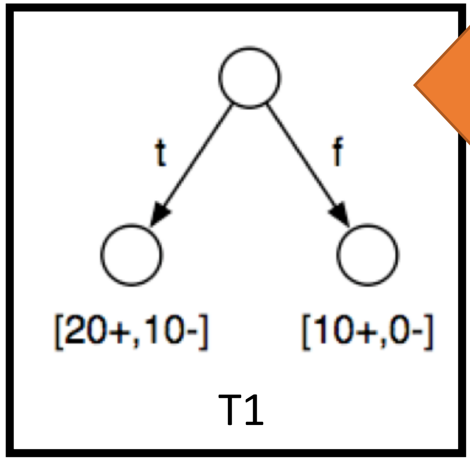
Example: Say that you’re given 40 examples. (30 Positive & 10 N

Consider two tests that would split the examples as follows:

Which is best?

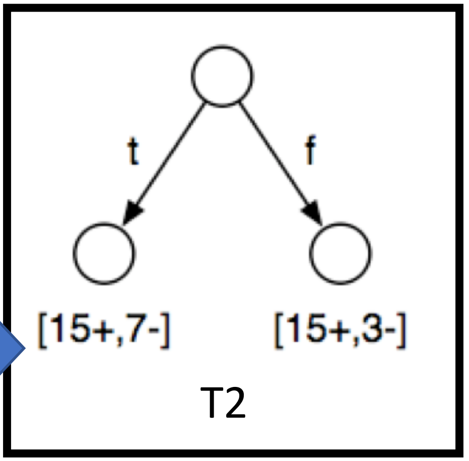
We prefer attributes that separate.

Problem: How can we quantify this?

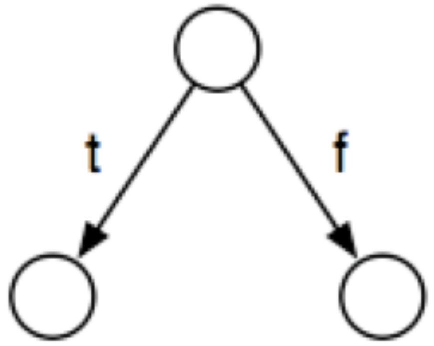


All negatives bucketed to t, with some division for positives.

Positives split evenly with negatives being more even, too.



Splitting Mechanisms



Quantifying Prospective Splits

1. **Information Gain**
   → Measure the entropy of a node's information.

Information Content as a Metric

Consider three cases:

Dice

Two-sided Coin

Biased Coin

Each case yields a different amount of **uncertainty** to their observed outcome.

Information Content as a Metric

Let E be an event that occurs with probability $P(E)$. If we are told that E has occurred with certainty, then we receive $I(E)$ bits of information.

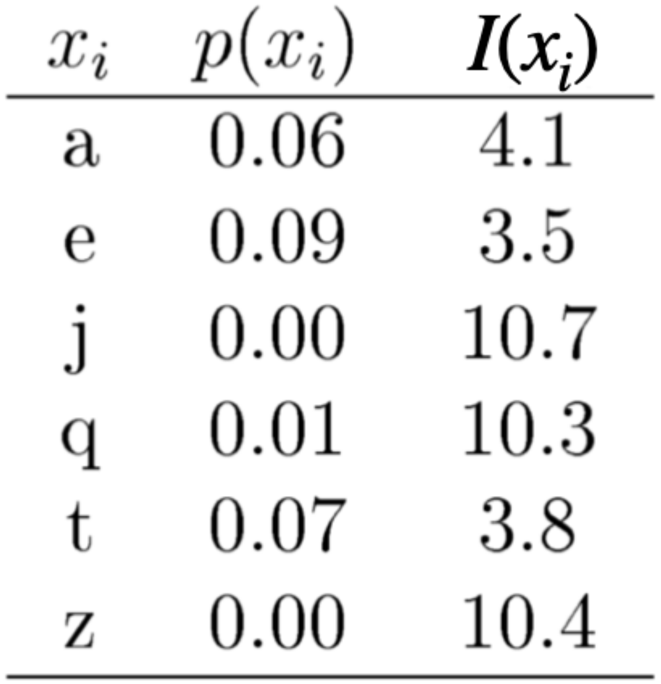
$$I(E) = \log_2 \frac{1}{P(E)}$$

Alternative Perspective: Think of information as "surprise" in the outcome. For example, if $P(E) = 1$, then $I(E) = 0$.

- Fair Coin Flip → $\log_2 2 = 1$ bit of information
- Fair Dice Roll→ $\log_2 6 = 2.58$ bits of information

Information Content as a Metric

An example is the English alphabet. Consider all the letters within it. The lower their probability, the higher their information content / surprise.



x_i	$p(x_i)$	$I(x_i)$
a	0.06	4.1
e	0.09	3.5
j	0.00	10.7
q	0.01	10.3
t	0.07	3.8
z	0.00	10.4

Information Entropy

Given an information source S which yields k symbols from an alphabet $\{s_1, \ldots, s_k\}$ with probabilities $\{p_1, \ldots, p_k\}$ where each yield is independent of the others.

$$
\begin{aligned}
H(S) &= \sum_{i}^{k} p_i \, I(s_i) \\
&= \sum_{i}^{k} p_i \log \frac{1}{p_i} \\
&= -\sum_{i}^{k} p_i \log p_i
\end{aligned}
$$

In other words …

Calculating Entropy

1. Take the log of 1 / p_i
2. Multiply the value from Step 1 by p_i.
3. Rinse and repeat for all "symbols".

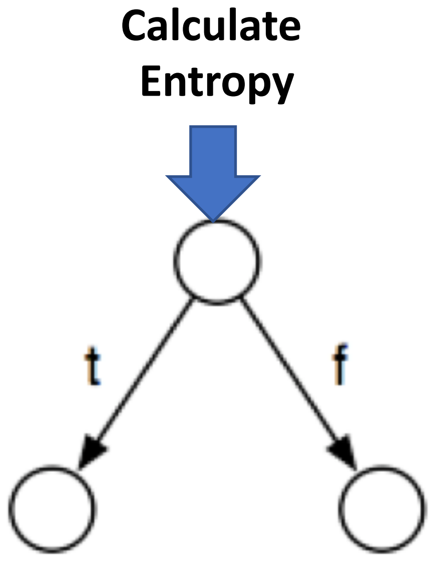
$H(S)$ is the entropy of the information source.

Information Entropy

$$H(S) = \sum_i p_i \log \frac{1}{p_i}$$

Several ways to think about Entropy:

- Average amount of information per symbol.
- Average amount of surprise when observing the symbol.
- Uncertainty the observer has before seeing the symbol.
- Average number of bits needed to communicate the symbol.



Binary Classification

Let's now try to classify a sample of the data S using a decision tree.

Suppose we have p positive samples and n negative samples.

What's the entropy of the dataset?

$$H(S) = -p_{\oplus} \log p_{\oplus} - p_{\ominus} \log p_{\ominus}$$

$$= -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Binary Classification

Example: Say that you're given 40 examples. (30 Positive & 10 Negative)

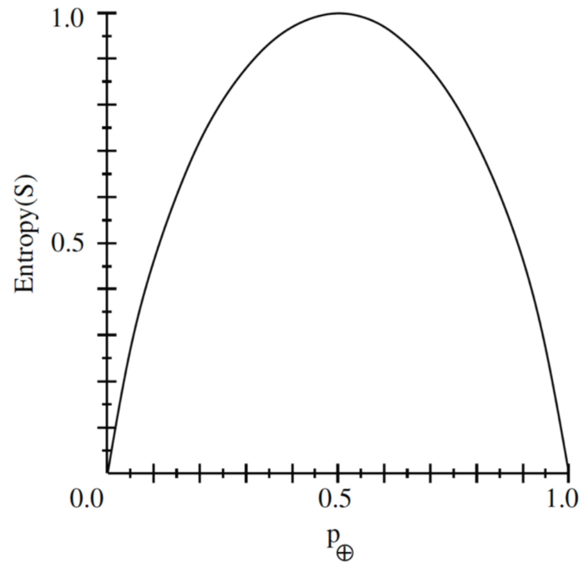
$$H(S) = -p_{\oplus}\log p_{\oplus} - p_{\ominus}\log p_{\ominus}$$

$$= -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

$$= -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$$

Binary Classification

$$H(S) = -p_{\oplus} \log p_{\oplus} - p_{\ominus} \log p_{\ominus}$$



Interpreting Entropy:

Entropy = 0 when all members of S belong to the same class.

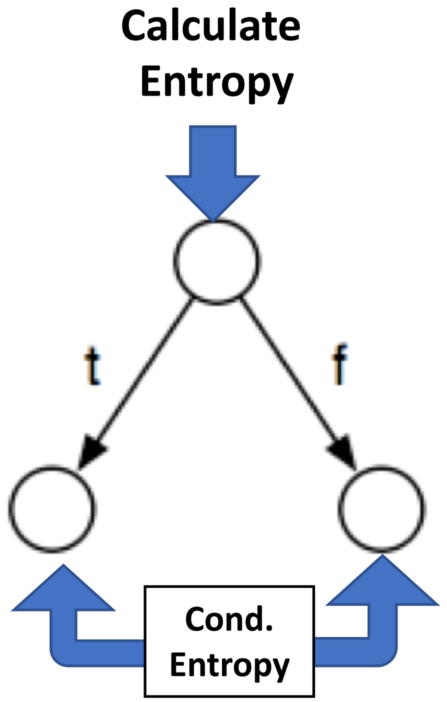
Entropy = 1 when classes in S are represented equally (i.e., Num of $\mathbf{p}$ == Num of $\mathbf{n}$)

Problem: Raw entropy only works for the current node.
→ Because child nodes have access to smaller subsets of the data.

Conditional Entropy

The conditional entropy $\mathbf{H}(\,\mathbf{y} \mid \mathbf{x}\,)$ is the average specific conditional entropy of $\mathbf{y}$ given the values of $\mathbf{x}$.

$$H(y \mid x) = \sum_{v} P(x = v) H(y \mid x = v)$$

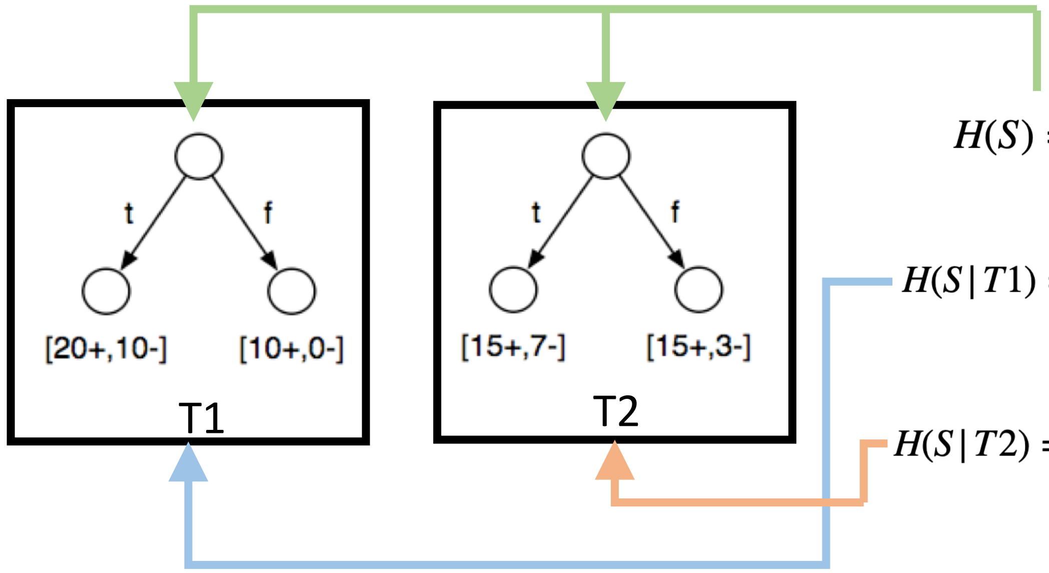


Plain English: When we evaluate a prospective child node, we need to evaluate how a node's information changes probabilistically.

Decision Tree: Training

What makes a good test? A "good" test provides information about the class label.

Example: Say that you're given 40 examples. (30 Positive & 10 Negative)



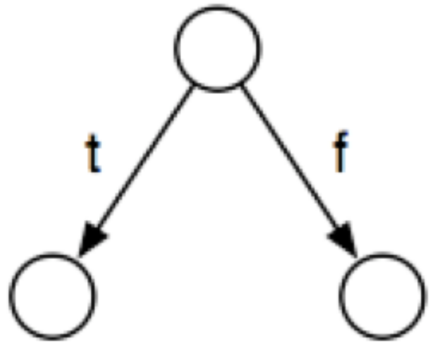
$$H(S) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$$

$$H(S\,|\,T1) = \frac{30}{40}\left[-\frac{20}{30}\log_2\frac{20}{30} - \frac{10}{30}\log_2\frac{10}{30}\right] + \frac{10}{40}[0] = 0.688$$

$$H(S\,|\,T2) = \frac{22}{40}\left[-\frac{15}{22}\log_2\frac{15}{22} - \frac{7}{22}\log_2\frac{7}{22}\right] + \frac{18}{40}\left[-\frac{15}{18}\log_2\frac{15}{18} - \frac{3}{18}\log_2\frac{3}{18}\right] = 0.788$$

Now, you split on the feature that gives you the highest information gain: **H(S) - H(S | x)**

Splitting Mechanisms



Quantifying Prospective Splits

1. **Information Gain**
   → Measure the entropy of a node's information.

2. **Gini Impurity**
   → Measure the "impurity" of a node's information..

Note: A 4th measure is variance. (Continuously targets only.)

Gini Impurity

Given an information source S which yields k symbols from an alphabet $\{s_1, \ldots, s_k\}$ with probabilities $\{p_1, \ldots, p_k\}$ where each yield is independent of the others.

$$H(S) = \sum_{i}^{k} p_i \log \frac{1}{p_i} \quad \Longrightarrow \quad G(S) = \sum_{i}^{k} p_i^{\,2}$$

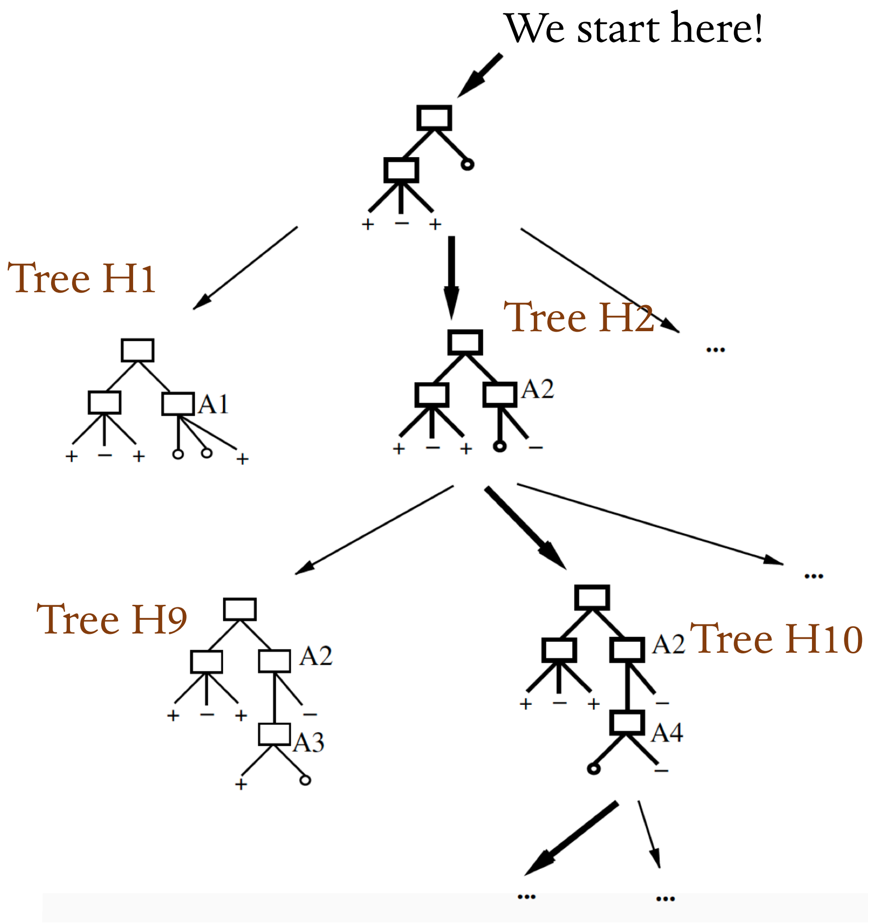
So, what? Gini outperforms computationally. (No $\log$ calls.)

Decision Trees as Search Problems

We can think about decision tree learning as **searching in a space of hypotheses** that fit our training examples.

The hypothesis space searched is the set of possible trees.

It begins with an empty tree. We move forward by considering progressively more elaborate trees.



Considerations

What if we have more than two targets?

Information Gain changes with target space.
→ More targets, more sensitivity. (Less accuracy?)
→ Features can still be relevant.

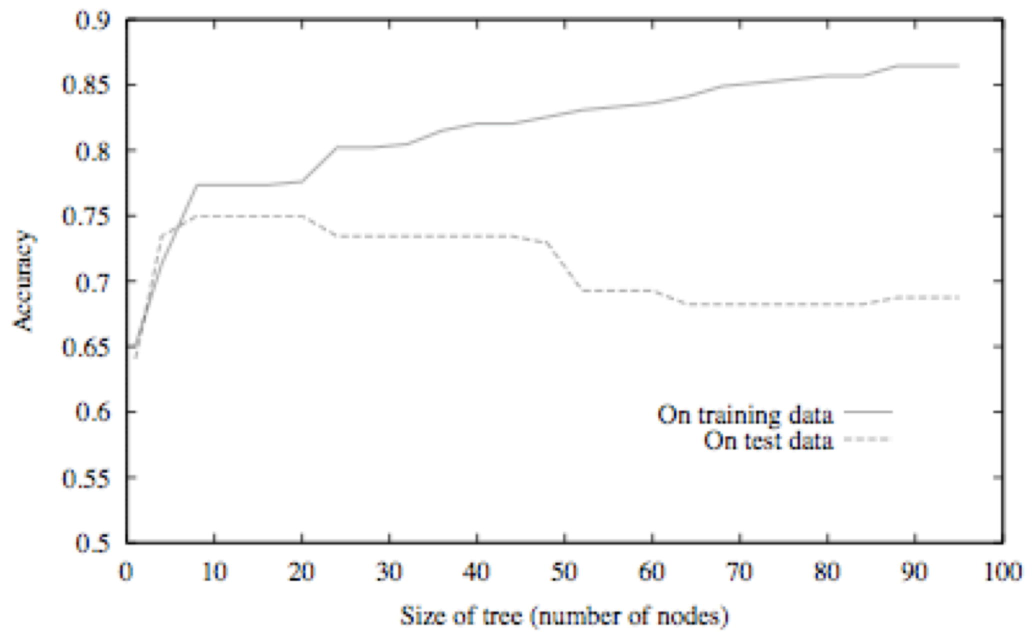
Import Consideration for Designing Tests
→ You can always make binary tests for features.
(Often multiple possibilities!)

Real-World: C4.5 uses *only* binary tests.

2. How can decision trees generalize?

Not All Trees are Fruitful

Decision tree construction continues until a node reaches purity (i.e., it contains information of only one class).



As a decision tree grows, performance can wane.

Overfitting

Definition: Given a hypothesis space $\mathbf{H}$, a hypothesis $\mathbf{h}$ in $\mathbf{H}$ is said to overfit the training data if there exist some alternative hypothesis $\mathbf{h}'$ in $\mathbf{H}$, such that h has a smaller error than $\mathbf{h}'$ over the training examples, but $\mathbf{h}'$ has a smaller error than $\mathbf{h}$ over the entire distribution of instances.

In Plain English: A learning algorithm has mapped to its training data <u>too well</u>.

Consider the Following: A new racecar driver spends a year's time learning to race professionally. Their training sessions have been conducted in sunny conditions with the same race car each session. On their first race day, it rains. Further, they discover they've entered a motorcycle race, not a "racecar" race.

How do they perform? **… Probably pretty poorly.**

Overfitting: How to Avoid

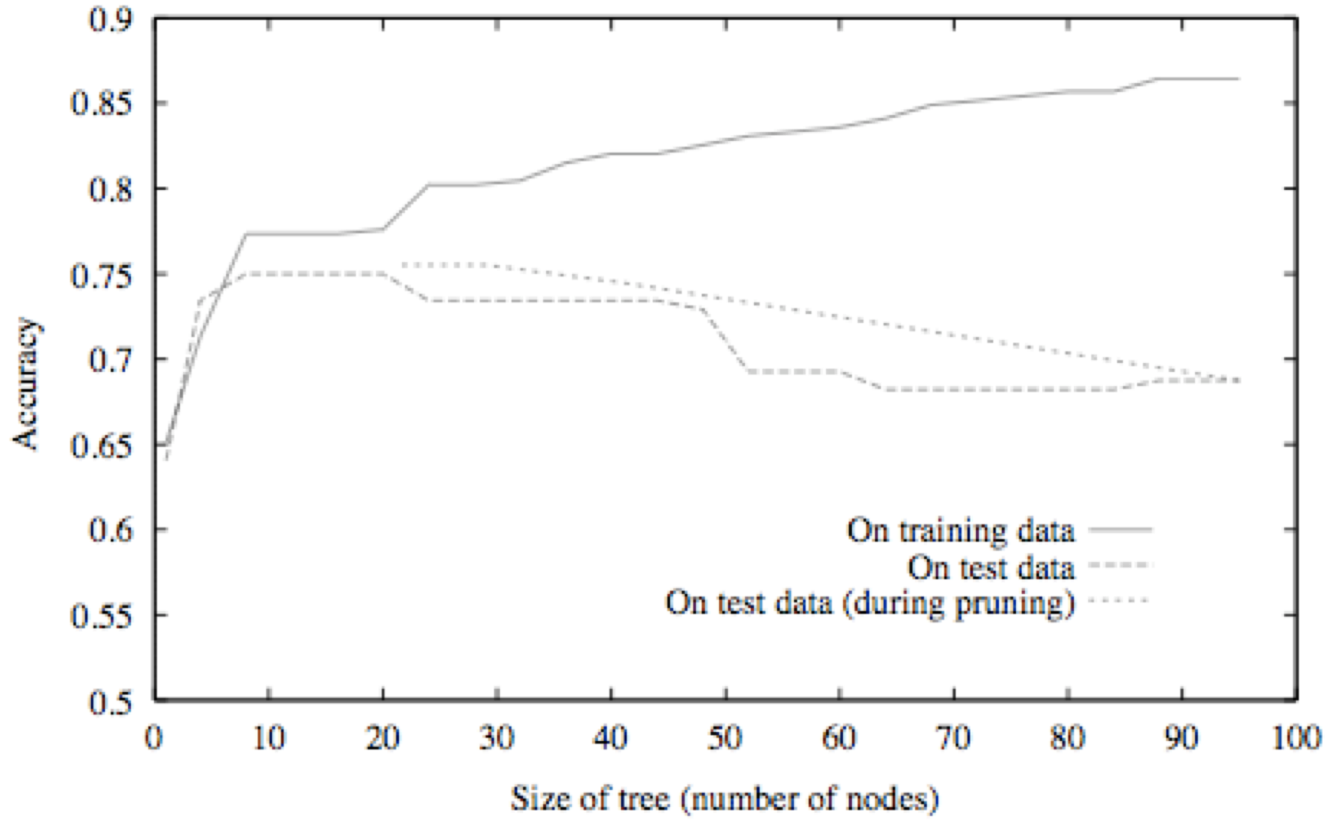
General Idea: Remove your nodes to better generalize.

1. **Early Stopping:** Stop growing the tree when further splitting the tree does not improve information gain in the dataset you're using for validation.

2. **Post-Pruning:** Build the complete tree, then revisit / prune trees that have low information gain in the dataset you're using for validation.

Preferred Solution: Post-Pruning is generally recognized as more beneficial as it allows you to account for setting where combinations of features are useful (instead of being forced to evaluate the utility of individual features at construction time).

Overfitting: Post-Pruning



Pruning isn't a panacea, … but it helps!

3. What is the inductive bias of decision trees?

Inductive Biases

All learning algorithms operate on assumptions.

- Provided x, y, z → This learning method will generalize.

  - We **want** our learned function to generalize beyond the data we have at hand!

What are Decision Trees' Inductive Biases?

- Shorter trees are better than longer trees.
- Trees that place high information gain features closer to the root are more preferred than those that do not.

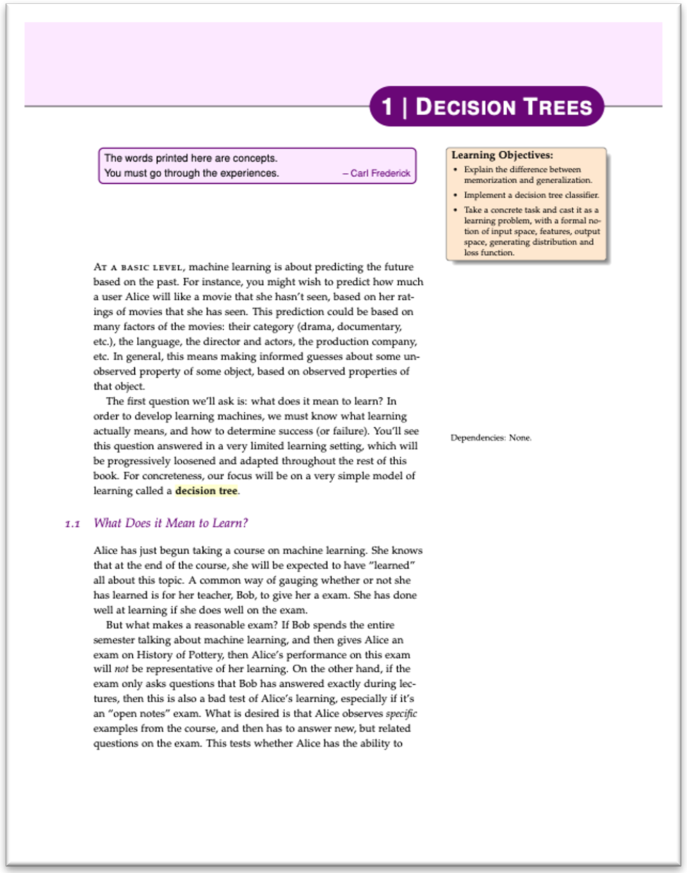
→ We aid these with (1) "good" metrics and (2) overfitting techniques.

Today's Agenda

We have addressed:

1. How do you train and test decision trees?
2. How can decision trees generalize?
3. What is the inductive bias of decision trees?

Reading



1 | Decision Trees

The words printed here are concepts.
You must go through the experiences. – Carl Frederick

Learning Objectives:
- Explain the difference between memorization and generalization.
- Implement a decision tree classifier.
- Take a concrete task and cast it as a learning problem, with a formal notion of input space, features, output space, generating distribution and loss function.

At a basic level, machine learning is about predicting the future based on the past. For instance, you might wish to predict how much a user Alice will like a movie that she hasn't seen, based on her ratings of movies that she has seen. This prediction could be based on many factors of the movies: their category (drama, documentary, etc.), the language, the director and actors, the production company, etc. In general, this means making informed guesses about some unobserved property of some object, based on observed properties of that object.

The first question we'll ask is: what does it mean to learn? In order to develop learning machines, we must know what learning actually means, and how to determine success (or failure). You'll see this question answered in a very limited learning setting, which will be progressively loosened and adapted throughout the rest of this book. For concreteness, our focus will be on a very simple model of learning called a **decision tree**.

Dependencies: None.

1.1 *What Does it Mean to Learn?*

Alice has just begun taking a course on machine learning. She knows that at the end of the course, she will be expected to have "learned" all about this topic. A common way of gauging whether or not she has learned is for her teacher, Bob, to give her a exam. She has done well at learning if she does well on the exam.

But what makes a reasonable exam? If Bob spends the entire semester talking about machine learning, and then gives Alice an exam on History of Pottery, then Alice's performance on this exam will *not* be representative of her learning. On the other hand, if the exam only asks questions that Bob has answered exactly during lectures, then this is also a bad test of Alice's learning, especially if it's an "open notes" exam. What is desired is that Alice observes *specific* examples from the course, and then has to answer new, but related questions on the exam. This tests whether Alice has the ability to

Daume. Chapter 1

Next Time

We will address:

1. How do you define “performance”?
2. How well can we generalize?