Inductive Learning and Decision Trees

Doug Downey
with slides from Pedro Domingos, Bryan Pardo
Outline

- Announcements
  - Homework #1 to be assigned soon
- Inductive learning
- Decision Trees
Outline

- Announcements
  - Homework #1 to be assigned soon
- Inductive learning
- Decision Trees
Instances

- E.g. Four Days, in terms of weather:

<table>
<thead>
<tr>
<th>Sky</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>same</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>same</td>
</tr>
<tr>
<td>rainy</td>
<td>cold</td>
<td>high</td>
<td>strong</td>
<td>change</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>change</td>
</tr>
</tbody>
</table>
Functions

- “Days on which Anne agrees to get lunch with me”

<table>
<thead>
<tr>
<th>Sky</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Forecast</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>same</td>
<td>1</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>same</td>
<td>1</td>
</tr>
<tr>
<td>rainy</td>
<td>cold</td>
<td>high</td>
<td>strong</td>
<td>change</td>
<td>0</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>change</td>
<td>1</td>
</tr>
</tbody>
</table>
### Inductive Learning!

- **Predict** the output for a new instance (**generalize!**) 

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sky</td>
<td>Temp</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
</tr>
<tr>
<td>rainy</td>
<td>cold</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
</tr>
<tr>
<td>rainy</td>
<td>warm</td>
</tr>
</tbody>
</table>
General Inductive Learning Task

**DEFINE:**
- Set $X$ of Instances (of $n$-tuples $x = <x_1, ..., x_n>$)
  - E.g., days described by *attributes* (or *features*): Sky, Temp, Humidity, Wind, Forecast
- **Target function** $f : X \rightarrow Y$, e.g.:
  - GoesToLunch $X \rightarrow Y = \{0,1\}$
  - ResponseToLunch $X \rightarrow Y = \{”No,” ”Yes,” ”How about tomorrow?”\}$
  - ProbabilityOfLunch $X \rightarrow Y = [0, 1]$

**GIVEN:**
- *Training examples* $D$
  - examples of the target function: $<x, f(x)>$

**FIND:**
- A *hypothesis* $h$ such that $h(x)$ approximates $f(x)$. 
Learn function from $\mathbf{x} = (x_1, \ldots, x_d)$ to $f(\mathbf{x}) \in \{0, 1\}$ given labeled examples $(\mathbf{x}, f(\mathbf{x}))$
Hypothesis Spaces

- **Hypothesis space** $H$ is a **subset** of all $f : X \rightarrow Y$ e.g.:
  - Linear separators
  - Conjunctions of constraints on attributes (humidity must be low, and outlook != rain)
  - Etc.

- In machine learning, we restrict ourselves to $H$
Examples

- **Credit Risk Analysis**
  - \( X \): Properties of customer and proposed purchase
  - \( f(x) \): Approve (1) or Disapprove (0)

- **Disease Diagnosis**
  - \( X \): Properties of patient (symptoms, lab tests)
  - \( f(x) \): Disease (if any)

- **Face Recognition**
  - \( X \): Bitmap image
  - \( f(x) \): Name of person

- **Automatic Steering**
  - \( X \): Bitmap picture of road surface in front of car
  - \( f(x) \): Degrees to turn the steering wheel
Inductive Learning *tasks*

- Defined in terms of **inputs** and **outputs**:
  - Predicting outcomes of sporting events
    - Input: A game (two opponents, a date)
    - Output: which team will win (classification)

- On the other hand, these are *not* tasks:
  - “Studying the relationship between weather and sports game outcomes.”
  - “Applying neural networks to natural language processing.”
When to use?

- Inductive Learning is appropriate for building a face recognizer
- It is not appropriate for building a calculator
  - You’d just write a calculator program

Question:
What general characteristics make a problem suitable for inductive learning?
Think/Pair/Share

What general characteristics make a problem suitable for inductive learning?
What general characteristics make a problem suitable for inductive learning?
Think/Pair/Share

What general characteristics make a problem suitable for inductive learning?

Share
Appropriate applications

- Situations in which:
  - There is no human expert
  - Humans can perform the task but can’t describe how
  - The desired function changes frequently
  - Each user needs a customized $f$
Outline

- Announcements
  - Homework #1
- Inductive learning
- Decision Trees
Why Decision Trees?

- Simple inductive learning approach
  - Training procedure is easy to understand
  - Models are easy to understand

- Popular
  - The most popular learning method, according to surveys
  [Domingos, 2016]
Task: Will I wait for a table?

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₂</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>X₃</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₄</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>X₅</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₆</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₇</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₈</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₉</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₁₀</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>X₁₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₁₂</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

Classification of examples is positive (T) or negative (F)
A Decision Tree for “Will I Wait”
Decision Trees can represent *any* Boolean function

E.g., for two binary attributes \{A, B\}, the tree for binary function \(f(A, B) = A \text{ xor } B\):
Inductive Learning with Decision Trees

In inductive learning, our goal is to learn a decision tree from a data set, such that it can generalize to new examples.

What tree might you learn from the following three examples?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>( f(A, B) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Think/Pair/Share

What tree might you learn from the following three examples?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>( f(A, B) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Think/Start | End
What tree might you learn from the following three examples?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>( f(A, B) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Think/Pair/Share

Pair

Start

End
Think/Pair/Share

What tree might you learn from the following three examples?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$f(A, B)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Share
Inductive Bias

- To learn, we **must** prefer some functions to others
  - **Selection bias**
    - use a *restricted* hypothesis space, e.g.:
      - linear separators
      - depth-2 decision trees
  - **Preference bias**
    - use the whole function space, but state a *preference* over functions, e.g.:
      - *Lowest-degree* polynomial that separates the data
      - *shortest* decision tree that fits the data
A learned decision tree

Decision tree learned from the 12 examples:

Substantially simpler than “true” tree—a more complex hypothesis isn’t justified by small amount of data
Summary

- **Inductive Learning**
  - Given **examples** of a **target function** $f$
    - **example** = **instance** (a vector of **attributes**)
      - and its corresponding target function value
  - Learn a **hypothesis** that approximates the function

- **Decision Trees**
  - One way of **representing** a hypothesis
  - Can represent any Boolean function

- **Inductive Bias**
  - Bias in favor of some functions over others
  - Necessary for learning
Outline

- Decision Tree Learning (ID3)
Decision Tree Learning (ID3*)

Goal: Find a (small) tree consistent with examples
Function ID3(examples, default) returns a tree

if examples is empty
    return tree(default)
else if all examples have same classification or no non-trivial splits are possible:
    return tree(MODE(examples))
else:
    best \leftarrow \text{CHOOSE-ATTRIBUTE}(examples)
    t \leftarrow \text{new tree with root test } best
    \text{for each } value_i \text{ of } best:
        examples_i \leftarrow \{ \text{elements of } examples \text{ with } best = value_i \}
        subtree \leftarrow \text{ID3}(examples_i, \text{MODE}(examples))
        \text{add branch to } t \text{ with label } value_i \text{ and subtree } subtree
    return t

* Our algorithm’s termination conditions differ in small ways from the original published ID3
Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”

Patrons? is a better choice—gives information about the classification
Think/Pair/Share

How should we choose which attribute to split on next?

Think

Start

End
How should we choose which attribute to split on next?
How should we choose which attribute to split on next?

Share
Information

- Brief sojourn into information theory
  - (on board)
Entropy

The entropy $H(X)$ of a Boolean random variable $X$ as the probability of $X = 0$ varies from 0 to 1.
Using Information

- Say we have $n$ attributes $A_1, A_2, \ldots, A_n$
- The key question: how much information, on average, will I gain about the class $y = f(x)$ by doing the split?
  - Choose attribute $A_i$ that maximizes this expected value
- \[ \text{InfoGain}(A_i) = H_{prior} - \sum_v P(A_i = v) H(y|A_i = v) \]
- Since $H_{prior}$ is constant w.r.t. $A_i$, we can just choose attribute with minimum $\sum_v P(A_i = v) H(y|A_i = v)$
Evaluating Decision Trees

- **Accuracy of a tree**
  - Fraction of examples where tree output matches the output in the data set

- What is the accuracy of a tree on the examples used to train it?
  - Assuming the “noiseless” case where the same attribute vector $\mathbf{x}$ always maps to the same output $f(\mathbf{x})$.
  - …100%

- If I deployed a tree and used it to classify new examples, would I expect it to be 100% accurate?
  - No.

- How to estimate accuracy of tree on new examples?
Measuring Performance

How do we know that $h \approx f$? (Hume’s **Problem of Induction**)

1) Use theorems of computational/statistical learning theory

2) Try $h$ on a new **test set** of examples
   (use **same distribution over example space** as training set)

**Learning curve** = % correct on test set as a function of training set size
Overfitting

![Graph showing the relationship between accuracy and the size of the tree (number of nodes) on training and test data.](image)
Overfitting is due to “noise”

- Sources of noise:
  - Erroneous training data
    - concept variable incorrect (annotator error)
    - Attributes mis-measured
  - More significant:
    - Irrelevant attributes
    - Target function not realizable in attributes
Irrelevant attributes

- If many attributes are noisy, information gains can be spurious, e.g.:
  - 20 noisy attributes
  - 10 training examples
  - Expected # of different depth-3 trees that split the training data perfectly using *only* noisy attributes: \( 13.4 \)
Not realizable

In general:
- We can rarely measure well enough for **perfect** prediction
- Target function is not uniquely determined by attribute values
- Target outputs appear to be “noisy”
  - Same attribute vector may yield distinct output values
Not realizable: Example

<table>
<thead>
<tr>
<th>Humidity</th>
<th>f(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.90</td>
<td>0</td>
</tr>
<tr>
<td>0.87</td>
<td>1</td>
</tr>
<tr>
<td>0.80</td>
<td>0</td>
</tr>
<tr>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td>0.70</td>
<td>1</td>
</tr>
<tr>
<td>0.69</td>
<td>1</td>
</tr>
<tr>
<td>0.65</td>
<td>1</td>
</tr>
<tr>
<td>0.63</td>
<td>1</td>
</tr>
</tbody>
</table>

**Decent hypothesis:**
Humidity > 0.70 → No
Otherwise → Yes

**Overfit hypothesis:**
Humidity > 0.89 → No
Humidity > 0.80
^ Humidity <= 0.89 → Yes
Humidity > 0.70
^ Humidity <= 0.80 → No
Humidity <= 0.70 → Yes
Overfitting in Decision Trees

Consider adding a noisy training example:
Sunny, Hot, Normal, Strong, PlayTennis=No
What effect on tree?
Avoiding Overfitting

- **Approaches**
  - Stop splitting when information gain is low or when split is not statistically significant.
  - Grow full tree and then **prune** it when done.
Reduced-Error Pruning

Split data into training and validation set

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy
Effect of Reduced Error Pruning

![Graph showing the effect of reduced error pruning on accuracy with respect to the size of the tree (number of nodes). The graph compares accuracy on training data, test data, and test data during pruning.](image)
C4.5 Algorithm

- Builds a decision tree from labeled training data
- Generalizes simple “ID3” tree by
  - Prunes tree after building to improve generality
  - Allows missing attributes in examples
  - Allowing continuous-valued attributes
Rule post pruning

- Used in C4.5
- Steps
  1. Build the decision tree
  2. Convert it to a set of logical rules
  3. Prune each rule independently
  4. Sort rules into desired sequence for use
Converting A Tree to Rules

```
Outlook
  Sunny
    Humidity
      High
        No
      Normal
        Yes
  Overcast
    Yes
  Rain
    Wind
      Strong
        No
      Weak
        Yes
```
IF \((Outlook = Sunny) \text{ AND } (Humidity = High)\) THEN \(PlayTennis = No\)

IF \((Outlook = Sunny) \text{ AND } (Humidity = Normal)\) THEN \(PlayTennis = Yes\)

\ldots
Other Odds and Ends

• Unknown Attribute Values?
Unknown Attribute Values

What if some examples are missing values of $A$?
Use training example anyway, sort through tree

- If node $n$ tests $A$, assign most common value of $A$ among other examples sorted to node $n$
- Assign most common value of $A$ among other examples with same target value
- Assign probability $p_i$ to each possible value $v_i$ of $A$ 
  Assign fraction $p_i$ of example to each descendant in tree

Classify new examples in same fashion
Odds and Ends

- Unknown Attribute Values?
- Continuous Attributes?
Decision Tree Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.
Learning Parity with Noise

When learning exclusive-or (2-bit parity), all splits look equally good. If extra random boolean features are included, they also look equally good. Hence, decision tree algorithms cannot distinguish random noisy features from parity features.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$J=4$ $J=4$ $J=4$
Decision Trees Bias

- How to solve 2-bit parity:
  - Split on *pairs* of attributes at once

- For *k*-bit parity, why not split on *k* attribute values at once?

=> *Parity functions are among the “victims” of the decision tree’s inductive bias.*
Now we have choices

- Re-split continuous attributes?
- Handling unknown variables?
- Prune or not?
- Stopping criteria?
- Split selection criteria?
- Use look-ahead?

- In homework #1: one choice for each
- In practice, how to decide? An instance of Model Selection
  - In general, we could also select an H other than decision trees
We can do model selection using a 70% train, 30% validation split of our data. But can we do better?
We can do model selection using a 70% train, 30% validation split of our data. But can we do better?
We can do model selection using a 70% train, 30% validation split of our data. But can we do better?
10-fold Cross-Validation

- On board
Take away about decision trees

- Used as classifiers
- Supervised learning algorithms (ID3, C4.5)
- Good for situations where
  - Inputs, outputs are discrete
  - Interpretability is important
  - “We think the true function is a small tree”
Readings

- **Decision Trees:**
  - Induction of decision trees, Ross Quinlan (1986) (covers ID3)
    - [https://link.springer.com/article/10.1007%2FBF00116251](https://link.springer.com/article/10.1007%2FBF00116251) (may need to be on campus to access)
  - C4.5: Programs for Machine Learning (2014) (covers C4.5)
    - [https://books.google.com/books?hl=en&lr=&id=b3ujBQAAQBAJ&oi=fnd&pg=PP1&dq=c4.5&ots=sPanSTEtC4&sig=c2Np0fBu37b-Ie-dVUyhulPjsv4#v=onepage&q=c4.5&f=false](https://books.google.com/books?hl=en&lr=&id=b3ujBQAAQBAJ&oi=fnd&pg=PP1&dq=c4.5&ots=sPanSTEtC4&sig=c2Np0fBu37b-Ie-dVUyhulPjsv4#v=onepage&q=c4.5&f=false)

- **Overfitting in Decision Trees**

- **Cross-Validation**