Machine Learning 10-601 B, Fall 2016

Introduction, Admin, Course Overview

Lecture 1, 08/29/ 2016

Maria-Florina (Nina) Balcan
Machine Learning

Image Classification

Document Categorization

Speech Recognition

Protein Classification

Spam Detection

Branch Prediction

Fraud Detection

Natural Language Processing

Playing Games

Computational Advertising
Machine Learning is Changing the World

“Machine learning is the hot new thing”
(John Hennessy, President, Stanford)

“A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Microsoft)

“Web rankings today are mostly a matter of machine learning”
(Prabhakar Raghavan, VP Engineering at Google)
The COOLEST TOPIC IN SCIENCE

• “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)

• “Machine learning is the next Internet” (Tony Tether, Director, DARPA)

• Machine learning is the hot new thing” (John Hennessy, President, Stanford)

• “Web rankings today are mostly a matter of machine learning” (Prabhakar Raghavan, Dir. Research, Yahoo)

• “Machine learning is going to result in a real revolution” (Greg Papadopoulos, CTO, Sun)

• “Machine learning is today’s discontinuity” (Jerry Yang, CEO, Yahoo)
This course: introduction to machine learning.

- Cover (some of) the most commonly used machine learning paradigms and algorithms.
  - Sufficient amount of details on their mechanisms: explain why they work, not only how to use them.
  - Applications.
What is Machine Learning?

Examples of important machine learning paradigms.
Supervised Classification
from data to discrete classes
Supervised Classification. Example: Spam Detection

Decide which emails are spam and which are important.

Goal: use emails seen so far to produce good prediction rule for future data.
Supervised Classification. Example: Spam Detection

Represent each message by features. (e.g., keywords, spelling, etc.)

```
<table>
<thead>
<tr>
<th>&quot;money&quot;</th>
<th>&quot;pills&quot;</th>
<th>&quot;Mr.&quot;</th>
<th>bad spelling</th>
<th>known-sender</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>
```

**example**

```
<table>
<thead>
<tr>
<th>&quot;money&quot;</th>
<th>&quot;pills&quot;</th>
<th>&quot;Mr.&quot;</th>
<th>bad spelling</th>
<th>known-sender</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
```

**Reasonable RULES:**

Predict SPAM if unknown AND (money OR pills)

Predict SPAM if 2money + 3pills –5 known > 0

*Linearly separable*
Supervised Classification. Example: Image classification

- Handwritten digit recognition (convert hand-written digits to characters 0..9)

- Face Detection and Recognition
Supervised Classification. Many other examples

- Weather prediction

- Medicine:
  - diagnose a disease
    - input: from symptoms, lab measurements, test results, DNA tests, …
    - output: one of set of possible diseases, or “none of the above”
    - examples: audiology, thyroid cancer, diabetes, …
      - or: response to chemo drug X
      - or: will patient be re-admitted soon?

- Computational Economics:
  - predict if a stock will rise or fall
  - predict if a user will click on an ad or not
    - in order to decide which ad to show
Regression. Predicting a numeric value

Stock market

Weather prediction

Predict the temperature at any given location
Other Machine Learning Paradigm

Clustering: discovering structure in data (only unlabeled data)

- E.g., cluster users of social networks by interest (community detection).

Semi-Supervised Learning: learning with labeled & unlabeled data

Active Learning: learns pick informative examples to be labeled

Reinforcement Learning (accommodates indirect or delayed feedback)

Dimensionality Reduction

Collaborative Filtering (Matrix Completion), …
Many communities relate to ML

Economics and Organizational Behavior

Computer science

Animal learning (Cognitive science, Psychology, Neuroscience)

Machine learning

Evolution

Statistics

Adaptive Control Theory
Admin, Logistics, Grading
Brief Overview

- **Meeting Time:** Mon, Wed, GHC 4401, 12:00 – 1:20
- **Course Staff**
  - **Instructors:**
    - Maria Florina (Nina) Balcan (ninamf@cs.cmu.edu)
    - Matt Gormley (mgormley@cs.cmu.edu)
  - **TAs:**
    - Ben Cowley (bcowley@andrew.cmu.edu)
    - Pradeep Dasigi (pdasigi@andrew.cmu.edu)
    - Simon Shaolei Du (ssdu@andrew.cmu.edu)
    - Tianshu Ren (tren@andrew.cmu.edu)
    - Hsiao-Yu Fish Tung (htung@cs.cmu.edu)
    - Varshaa Naganathan (vnaganat@andrew.cmu.edu)
    - Sriram Vasudevan (svasude1@andrew.cmu.edu)
  - **Assistant Instructor:** Sarah Schultz (sschultz@cs.cmu.edu)
Brief Overview

• Course Website
  http://www.cs.cmu.edu/~mgormley/courses/10601b-f16/

• See website for:
  • Syllabus details
    • All the lecture slides and homeworks
    • Additional useful resources.
  • Office hours
  • Review sessions (before each exam and before each hwk is due)
  • Grading policy
  • Honesty policy
  • Late homework policy
  • Piazza pointers

• Will use Piazza for discussions.

• For personal questions (e.g., extensions or prerequisites), use the mailing list: 10601-instructors@cs.cmu.edu
Prerequisites. What do you need to know now?

• You should know how to do math and how to program:
  – Calculus (multivariate)
  – Probability/statistics
  – Algorithms. Big O notation.
  – Linear algebra (matrices and vectors)
  – Programming:
    • You will implement some of the algorithms and apply them to datasets
    • Assignments will be mostly in Matlab and/or Octave (play with that now if you want)
    • All CMU students can download Matlab free of charge from CMU software website. Octave is open-source software.
• We may review these things but we will not teach them
• Use HWK #1 (out on Wed) as a self-assessment.
• Strongly encourage to take 10-600 if you need help with math.
Source Materials

No textbook required. Will point to slides and freely available online material.

Useful textbooks:


*Pattern Recognition and Machine Learning*
Christopher Bishop, Springer-Verlag 2006
Grading

- 55% for homeworks. There are 7 and you can drop 1.
- 20% for midterm
- 20% for final
- 5% for class participation.
  - Piazza polls in class: bring a laptop or a phone
- 7 weekly (mostly) homeworks
  - Theory/math handouts
  - Programming exercises
  - Applying/evaluating existing learners
  - Late assignments:
    - Up to 50% credit if it’s less than 48 hrs late
    - You can drop your lowest assignment grade
Collaboration policy (see syllabus)

• Discussion of anything is ok…
• …but the goal should be to understand better, not save work.

• So:
  – no notes of the discussion are allowed…the only thing you can take away is whatever’s in your brain.
  – you should acknowledge who you got help from/did help in your homework

• This policy is stolen from Roni Rosenfeld.

• Just so you know: we will fail students, and CMU will expel them.
Learning Decision Trees.
Supervised Classification.

Useful Readings:
• Mitchell, Chapter 3
• Bishop, Chapter 14.4

DT learning: Method for learning discrete-valued target functions in which the function to be learned is represented by a decision tree.
**Supervised Classification: Decision Tree Learning**

**Example:** learn concept **PlayTennis** (i.e., decide whether our friend will play tennis or not in a given day)

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Supervised Classification: Decision Tree Learning

- Each internal node: test one (discrete-valued) attribute $X_i$
- Each branch from a node: corresponds to one possible values for $X_i$
- Each leaf node: predict $Y$ (or $P(Y=1|x \in \text{leaf})$)

Example: A Decision tree for

\[ f: \langle \text{Outlook, Temperature, Humidity, Wind} \rangle \rightarrow \text{PlayTennis?} \]

E.g., \( x=(\text{Outlook}=\text{sunny}, \ \text{Temperature}=\text{Hot}, \ \text{Humidity}=\text{Normal}, \ \text{Wind}=\text{High}), \ f(x)=\text{Yes}. \)
Supervised Classification: Problem Setting

**Input:** Training labeled examples \( \{(x^{(i)}, y^{(i)})\} \) of unknown target function \( f \)

- Examples described by their values on some set of features or attributes
  - E.g. 4 attributes: Humidity, Wind, Outlook, Temp
    - e.g., \(<\text{Humidity}=\text{High}, \text{Wind}=\text{weak}, \text{Outlook}=\text{rain}, \text{Temp}=\text{Mild}>\)  
- Set of possible instances \( X \) (a.k.a instance space)

- Unknown target function \( f : X \rightarrow Y \)
  - e.g., \( Y=\{0,1\} \) label space
  - e.g., 1 if we play tennis on this day, else 0

**Output:** Hypothesis \( h \in H \) that (best) approximates target function \( f \)

- Set of function hypotheses \( H=\{ \ h \ | \ h : X \rightarrow Y \ \} \)
  - each hypothesis \( h \) is a decision tree
Supervised Classification: Decision Trees

Suppose $X = <x_1, \ldots, x_n>$
where $x_i$ are boolean-valued variables

How would you represent the following as DTs?

- $f(x) = x_2 \text{ AND } x_5$
- $f(x) = x_2 \text{ OR } x_5$

Hwk: How would you represent $X_2 \land X_5 \lor X_3X_4(\neg X_1)$?
**Supervised Classification: Problem Setting**

**Input:** Training labeled examples \( \{(x^{(i)}, y^{(i)})\} \) of unknown target function \( f \)

- Examples described by their values on some set of features or attributes
  - E.g. 4 attributes: *Humidity, Wind, Outlook, Temp*
    - e.g., \(<\text{Humidity}=\text{High}, \text{Wind}=\text{weak}, \text{Outlook}=\text{rain}, \text{Temp}=\text{Mild}>\)

- Set of possible instances \( X \) (a.k.a instance space)
  - Unknown target function \( f : X \rightarrow Y \)
    - e.g., \( Y=\{0,1\} \) label space
    - e.g., 1 if we play tennis on this day, else 0

**Output:** Hypothesis \( h \in H \) that (best) approximates target function \( f \)

- Set of function hypotheses \( H=\{ h \mid h : X \rightarrow Y \} \)
  - each hypothesis \( h \) is a decision tree

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Core Aspects in Decision Tree & Supervised Learning

How to automatically find a good hypothesis for training data?

• This is an *algorithmic* question, the main topic of computer science

When do we generalize and do well on unseen data?

• **Learning theory** quantifies ability to **generalize** as a function of the amount of training data and the hypothesis space
• **Occam’s razor**: use the *simplest* hypothesis consistent with data!

Fewer short hypotheses than long ones
• a short hypothesis that fits the data is less likely to be a statistical coincidence
• highly probable that a sufficiently complex hypothesis will fit the data
Core Aspects in Decision Tree & Supervised Learning

How to automatically find a good hypothesis for training data?

- This is an **algorithmic** question, the main topic of computer science

When do we generalize and do well on unseen data?

- **Occam’s razor:** use the *simplest* hypothesis consistent with data!

- Decision trees: if we were able to find a **small decision tree** that explains data well, then good generalization guarantees.
  - **NP-hard** [Hyafil-Rivest’76]: unlikely to have a poly time algorithm

- Very nice practical heuristics: top down algorithms, e.g, ID3
Top-Down Induction of Decision Trees

ID3: Natural greedy approach to growing a decision tree top-down (from the root to the leaves by repeatedly replacing an existing leaf with an internal node.).

Algorithm:
- Pick “best” attribute to split at the root based on training data.
- Recurse on children that are impure (e.g., have both Yes and No).
Top-Down Induction of Decision Trees

ID3: Natural greedy approaches where we grow the tree from the root to the leaves by repeatedly replacing an existing leaf with an internal node.

\[\text{node} = \text{Root}\]

Main loop:

1. \(A \leftarrow \text{the “best” decision attribute for next node}\)
2. Assign \(A\) as decision attribute for node
3. For each value of \(A\), create new descendent of node
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes.

Key question: Which attribute is best?
Top-Down Induction of Decision Trees

ID3: Natural greedy approach to growing a decision tree top-down.

Algorithm:

- Pick “best” attribute to split at the root based on training data.
- Recurse on children that are impure (e., have both Yes and No).

Key question: Which attribute is best?

ID3 uses a statistical measure called information gain (how well a given attribute separates the training examples according to the target classification)
Top-Down Induction of Decision Trees  

Which attribute to select?

ID3: The attribute with highest information gain.

- a statistical measure of how well a given attribute separates the training examples according to the target classification

**Information Gain** of $A$ is the expected reduction in entropy of target variable $Y$ for data sample $S$, due to sorting on variable $A$

$$Gain(S, A) = H_S(Y) - H_S(Y | A)$$

Entropy information theoretic measure that characterizes the impurity of a labeled set $S$. 

[ID3, C4.5, Quinlan]
Sample Entropy of a Labeled Dataset

- $S$ is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in $S$.
- $p_\ominus$ is the proportion of negative examples in $S$.
- Entropy measures the impurity of $S$.

$$H(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

- E.g., if all negative, then entropy=0. If all positive, then entropy=0.
- If 50/50 positive and negative then entropy=1.
- If 14 examples with 9 positive and 5 negative, then entropy=.940
Sample Entropy of a Labeled Dataset

- $S$ is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in $S$.
- $p_\ominus$ is the proportion of negative examples in $S$.
- Entropy measures the impurity of $S$.

$$H(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Interpretation from information theory: expected number of bits needed to encode label of a randomly drawn example in $S$.

- If $S$ is all positive, receiver knows label will be positive, don’t need any bits.
- If $S$ is 50/50 then need 1 bit.
- If $S$ is 80/20, then in a long sequence of messages, can code with less than 1 bit on average (assigning shorter codes to positive examples and longer codes to negative examples).
Sample Entropy of a Labeled Dataset

- $S$ is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in $S$.
- $p_\ominus$ is the proportion of negative examples in $S$.
- Entropy measures the impurity of $S$.

$$H(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

If labels not Boolean, then $H(S) = \sum_{i \in Y} -p_i \log_2 p_i$

E.g., if $c$ classes, all equally likely, then $H(S) = \log_2 c$
Information Gain

Given the definition of entropy, can define a measure of effectiveness of attribute in classifying training data:

Information Gain of $A$ is the expected reduction in entropy of target variable $Y$ for data sample $S$, due to sorting on variable $A$

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- $Gain(S, A)$: entropy of original collection
- $Entropy(S)$: Expected entropy after $S$ is partitioned using attribute $A$
- $|S_v|$: sum of entropies of subsets $S_v$ weighted by the fraction of examples that belong to $S_v$. 
Selecting the Next Attribute

Which attribute is the best classifier?

Selecting Humidity:

S: \([9+, 5 -]\)
\[E = 0.940\]

Humidity

\[\text{High}\]
\[\text{Normal}\]

\[\text{[3+, 4 -]}\]
\[E = 0.985\]

\[\text{[6+, 1 -]}\]
\[E = 0.592\]

Gain(S, Humidity)

\[= .940 - \left(\frac{7}{14}\right) .985 - \left(\frac{7}{14}\right) .592\]
\[= .151\]

Selecting Wind:

S: \([9+, 5 -]\)
\[E = 0.940\]

Wind

\[\text{Weak}\]
\[\text{Strong}\]

\[\text{[6+, 2 -]}\]
\[E = 0.8111\]

\[\text{[3+, 3 -]}\]
\[E = 1.00\]

Gain(S, Wind)

\[= .940 - \left(\frac{8}{14}\right) .811 - \left(\frac{6}{14}\right) 1.0\]
\[= .048\]
Which attribute should be tested here?

\[ s_{\text{sunny}} = \{D1, D2, D8, D9, D11\} \]

\[
\text{Gain}(s_{\text{sunny}}, \text{Humidity}) = .970 - \left(\frac{3}{5}\right)0.0 - \left(\frac{2}{5}\right)0.0 = .970
\]

\[
\text{Gain}(s_{\text{sunny}}, \text{Temperature}) = .970 - \left(\frac{2}{5}\right)0.0 - \left(\frac{2}{5}\right)1.0 - \left(\frac{1}{5}\right)0.0 = .570
\]

\[
\text{Gain}(s_{\text{sunny}}, \text{Wind}) = .970 - \left(\frac{2}{5}\right)1.0 - \left(\frac{3}{5}\right)0.918 = .019
\]
Final Decision Tree for

\( f: \langle \text{Outlook, Temperature, Humidity, Wind} \rangle \rightarrow \text{PlayTennis?} \)

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

\( \text{Outlook:} \) Sunny, Overcast, Rain

\( \text{Humidity:} \) High, Normal

\( \text{Wind:} \) Strong, Weak

\( \text{Final Decision Tree:} \)

1. **Outlook**
   - Sunny
   - Overcast
   - Rain

   1. **Humidity**
      - High
      - Normal

      1. No
      2. Yes

   2. **Wind**
      - Strong
      - Weak

      1. No
      2. Yes
Properties of ID3

- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

**Occam’s razor:** prefer the simplest hypothesis that fits the data
Overfitting in Decision Trees

Consider adding noisy training example #15:  
*Sunny, Hot, Normal, Strong, PlayTennis = No*  
What effect on earlier tree?
Properties of ID3

Overfitting could occur because of noisy data and because ID3 is not guaranteed to output a small hypothesis even if one exists.

Consider a hypothesis $h$ and its

- Error rate over training data: $error_{\text{train}}(h)$
- True error rate over all data: $error_{\text{true}}(h)$

We say $h$ overfits the training data if

\[ error_{\text{true}}(h) > error_{\text{train}}(h) \]

Amount of overfitting = $error_{\text{true}}(h) - error_{\text{train}}(h)$
Overfitting in Decision Tree Learning

![Graph showing the accuracy of training and test data with increasing tree size](#)

- **On training data**
- **On test data**

**Axes:**
- x-axis: Size of tree (number of nodes)
- y-axis: Accuracy

The accuracy on training data increases rapidly with the size of the tree but plateaus at around 0.9. In contrast, the accuracy on test data increases more slowly and levels off at around 0.8, indicating overfitting as the tree grows larger.
Avoiding Overfitting

How can we avoid overfitting?

• Stop growing when data split not statistically significant

• Grow full tree, then post-prune
Key Issues in Machine Learning

• How can we gauge the accuracy of a hypothesis on unseen data?
  – **Occam’s razor**: use the *simplest* hypothesis consistent with data!
    This will help us avoid overfitting.
  – *Learning theory* will help us quantify our ability to *generalize* as a
    function of the amount of training data and the hypothesis space

• How do we find the best hypothesis?
  – This is an **algorithmic** question, the main topic of computer science

• How do we choose a hypothesis space?
  – Often we use **prior knowledge** to guide this choice

• How to model applications as machine learning problems?
  (engineering challenge)