Lecture 07 of 42

Decision Trees, Occam’s Razor, and Overfitting

Wednesday, 31 January 2007

William H. Hsu
Department of Computing and Information Sciences, KSU
http://www.cis.ksu.edu/~bhsu

Readings:
Chapter 3.6-3.8, Mitchell

Lecture Outline

• Read Sections 3.6-3.8, Mitchell
• Occam’s Razor and Decision Trees
  - Preference biases versus language biases
  - Two issues regarding Occam algorithms
    • Is Occam’s Razor well defined?
    • Why prefer smaller trees?
• Overfitting (aka Overtraining)
  - Problem: fitting training data too closely
    • Small-sample statistics
    • General definition of overfitting
  - Overfitting prevention, avoidance, and recovery techniques
    • Prevention: attribute subset selection
    • Avoidance: cross-validation
    • Detection and recovery: post-pruning
• Other Ways to Make Decision Tree Induction More Robust
Decision Tree Learning: Top-Down Induction (ID3)

- Algorithm Build-DT (Examples, Attributes)
  
  IF all examples have the same label THEN RETURN (leaf node with label)
  ELSE
  IF set of attributes is empty THEN RETURN (leaf with majority label)
  ELSE
    Choose best attribute A as root
    FOR each value v of A
      Create a branch out of the root for the condition A = v
      IF \{x ∈ Examples: x.A = v\} = Ø THEN RETURN (leaf with majority label)
      ELSE Build-DT ((x ∈ Examples: x.A = v), Attributes ~ (A))
  
  But Which Attribute Is Best?

Broadening the Applicability of Decision Trees

- Assumptions in Previous Algorithm
  - Discrete output
    - Real-valued outputs are possible
    - Regression trees [Breiman et al., 1984]
  - Discrete input
    - Quantization methods
    - Inequalities at nodes instead of equality tests (see rectangle example)

- Scaling Up
  - Critical in knowledge discovery and database mining (KDD) from very large databases (VLDB)
  - Good news: efficient algorithms exist for processing many examples
  - Bad news: much harder when there are too many attributes

- Other Desired Tolerances
  - Noisy data (classification noise ≡ incorrect labels; attribute noise ≡ inaccurate or imprecise data)
  - Missing attribute values
Choosing the “Best” Root Attribute

• Objective
  – Construct a decision tree that is as small as possible (Occam’s Razor)
  – Subject to: consistency with labels on training data

• Obstacles
  – Finding the minimal consistent hypothesis (i.e., decision tree) is NP-hard (D’oh!)
  – Recursive algorithm (Build-DT)
    • A greedy heuristic search for a simple tree
    • Cannot guarantee optimality (D’oh!)

• Main Decision: Next Attribute to Condition On
  – Want: attributes that split examples into sets that are relatively pure in one label
  – Result: closer to a leaf node
  – Most popular heuristic
    • Developed by J. R. Quinlan
    • Based on information gain
    • Used in ID3 algorithm

A Measure of Uncertainty

– The Quantity
  • Purity: how close a set of instances is to having just one label
  • Impurity (disorder): how close it is to total uncertainty over labels

– The Measure: Entropy
  • Directly proportional to impurity, uncertainty, irregularity, surprise
  • Inversely proportional to purity, certainty, regularity, redundancy

Example

– For simplicity, assume \( H = \{0, 1\} \), distributed according to \( Pr(y) \)
  • Can have (more than 2) discrete class labels
  • Continuous random variables: differential entropy

– Optimal purity for \( y \): either
  • \( Pr(y = 0) = 1, Pr(y = 1) = 0 \)
  • \( Pr(y = 1) = 1, Pr(y = 0) = 0 \)

– What is the least pure probability distribution?
  • \( Pr(y = 0) = 0.5, Pr(y = 1) = 0.5 \)
  • Corresponds to maximum impurity/uncertainty/irregularity/surprise
  • Property of entropy: concave function (“concave downward”)
Entropy: Information Theoretic Definition

- **Components**
  - $D$: a set of examples $\{<x_1, c(x_1)>, <x_2, c(x_2)>, \ldots, <x_m, c(x_m)>\}$
  - $p_+ = Pr(c(x) = +), p_- = Pr(c(x) = -)$

- **Definition**
  - $H$ is defined over a probability density function $p$
  - $D$ contains examples whose frequency of + and - labels indicates $p_+$ and $p_-$ for the observed data
  - The entropy of $D$ relative to $c$ is:
    $$H(D) = -p_+ \log_b (p_+) - p_- \log_b (p_-)$$

- **What Units is $H$ Measured In?**
  - Depends on the base $b$ of the log (bits for $b = 2$, nats for $b = e$, etc.)
  - A single bit is required to encode each example in the worst case ($p_+ = 0.5$)
  - If there is less uncertainty (e.g., $p_+ = 0.8$), we can use less than 1 bit each

---

Information Gain: Information Theoretic Definition

- **Partitioning on Attribute Values**
  - Recall: a partition of $D$ is a collection of disjoint subsets whose union is $D$
  - Goal: measure the uncertainty removed by splitting on the value of attribute $A$

- **Definition**
  - The information gain of $D$ relative to attribute $A$ is the expected reduction in entropy due to splitting ("sorting") on $A$:
    $$\text{Gain}(D, A) = -H(D) - \sum_{v\in\text{values}(A)} \frac{|D_v|}{|D|} H(D_v)$$

  - Idea: partition on $A$; scale entropy to the size of each subset $D_v$

- **Which Attribute Is Best?**
  - $A_1$
    - True: $[29+, 35-]$
    - False: $[8+, 30-]$
  - $A_2$
    - True: $[18+, 33-]$
    - False: $[11+, 2-]$

KSU
An Illustrative Example

- Training Examples for Concept PlayTennis

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

- ID3 = Build-DT using Gain(*)
- How Will ID3 Construct A Decision Tree?

Constructing A Decision Tree for PlayTennis using ID3 [1]

- Selecting The Root Attribute

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

- Prior (unconditioned) distribution: 9+, 5-
  - $H(D) = -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) = 0.94$ bits
  - $H(D, Humidity = High) = -(3/7) \log_2 (3/7) - (4/7) \log_2 (4/7) = 0.985$ bits
  - $H(D, Humidity = Normal) = -(6/7) \log_2 (6/7) - (1/7) \log_2 (1/7) = 0.592$ bits
  - $Gain(D, Humidity) = 0.94 - (7/14) \times 0.985 - (7/14) \times 0.592 = 0.151$ bits
  - Similarly, $Gain(D, Wind) = 0.94 - (8/14) \times 0.811 - (6/14) \times 1.0 = 0.048$ bits

$$Gain(D,A) = -H(D) - \sum_{v \in \text{values}(A)} \left( \frac{P_v}{P} \right) H(D_v)$$
Constructing A Decision Tree for PlayTennis using ID3 [2]

- Selecting The Root Attribute
  - \( \text{Gain}(D, \text{Humidity}) = 0.151 \) bits
  - \( \text{Gain}(D, \text{Wind}) = 0.048 \) bits
  - \( \text{Gain}(D, \text{Temperature}) = 0.029 \) bits
  - \( \text{Gain}(D, \text{Outlook}) = 0.246 \) bits

- Selecting The Next Attribute (Root of Subtree)
  - Continue until every example is included in path or purity = 100%
  - What does purity = 100% mean?
  - Can \( \text{Gain}(D, A) < 0 \)?

Constructing A Decision Tree for PlayTennis using ID3 [3]

- Selecting The Next Attribute (Root of Subtree)
  - Convention: \( \lg (0/a) = 0 \)
  - \( \text{Gain}(D_{\text{Sunny}}, \text{Humidity}) = 0.97 - (3/5) \times 0 - (2/5) \times 0 = 0.97 \) bits
  - \( \text{Gain}(D_{\text{Sunny}}, \text{Wind}) = 0.97 - (2/5) \times 1 - (3/5) \times 0.92 = 0.02 \) bits
  - \( \text{Gain}(D_{\text{Sunny}}, \text{Temperature}) = 0.57 \) bits

- Top-Down Induction
  - For discrete-valued attributes, terminates in \( O(n) \) splits
  - Makes at most one pass through data set at each level (why?)
Constructing A Decision Tree for PlayTennis using ID3 [4]

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Outlook?
1,2,3,4,5,6,7,8,9,10,11,12,13,14
[9+,5-]
Sunny
Overcast
Rain

Humidity?
High
Normal
1,2,8,9,11
[2+,3-]
3,7,12,13
[4+,0-]
Strong
Light

Wind?
Yes
No
4,5,6,10,14
[3+,2-]

1,2,3,4,5,6,7,8,9,10,11,12,13,14
[9+,5-]

Hypothesis Space Search by ID3

- **Search Problem**
  - Conduct a search of the space of decision trees, which can represent all possible discrete functions
    - Pros: expressiveness; flexibility
    - Cons: computational complexity; large, incomprehensible trees (next time)
  - Objective: to find the best decision tree (minimal consistent tree)
  - Obstacle: finding this tree is \( \text{NP} \)-hard
  - Tradeoff
    - Use heuristic (figure of merit that guides search)
    - Use greedy algorithm
      - Aka hill-climbing (gradient “descent”) without backtracking

- **Statistical Learning**
  - Decisions based on statistical descriptors \( p_+, p_- \) for subsamples \( D_v \)
  - In ID3, all data used
  - Robust to noisy data
Inductive Bias in ID3

- Heuristic : Search :: Inductive Bias : Inductive Generalization
  - \( H \) is the power set of instances in \( X \)
  - \( \Rightarrow \) Unbiased?  Not really…
    - Preference for short trees (termination condition)
    - Preference for trees with high information gain attributes near the root
    - \( \text{Gain}(\cdot) \): a heuristic function that captures the inductive bias of ID3
  - Bias in ID3
    - Preference for some hypotheses is encoded in heuristic function
    - Compare: a restriction of hypothesis space \( H \) (previous discussion of propositional normal forms: \( k\)-CNF, etc.)

- Preference for Shortest Tree
  - Prefer shortest tree that fits the data
  - An Occam’s Razor bias: shortest hypothesis that explains the observations

MLC++:
A Machine Learning Library

- \textit{MLC++}
  - An object-oriented machine learning library
  - Contains a suite of inductive learning algorithms (including \textit{ID3})
  - Supports incorporation, reuse of other DT algorithms (\textit{C4.5}, etc.)
  - Automation of statistical evaluation, cross-validation

- Wrappers
  - Optimization loops that iterate over inductive learning functions (\textit{inducers})
  - Used for performance tuning (finding subset of \textit{relevant} attributes, etc.)

- Combiners
  - Optimization loops that iterate over or interleave inductive learning functions
  - Used for performance tuning (finding subset of \textit{relevant} attributes, etc.)
  - Examples: bagging, boosting (later in this course) of \textit{ID3}, \textit{C4.5}

- Graphical Display of Structures
  - Visualization of DTs (AT&T \textit{dotty}, SGI \textit{MineSet TreeViz})
  - General logic diagrams (projection visualization)
Occam’s Razor and Decision Trees: A Preference Bias

- Preference Biases versus Language Biases
  - Preference bias
    - Captured (“encoded”) in learning algorithm
    - Compare: search heuristic
  - Language bias
    - Captured (“encoded”) in knowledge (hypothesis) representation
    - Compare: restriction of search space
    - aka restriction bias

- Occam’s Razor: Argument in Favor
  - Fewer short hypotheses than long hypotheses
    - e.g., half as many bit strings of length \(n\) as of length \(n + 1\), \(n \geq 0\)
    - Short hypothesis that fits data less likely to be coincidence
    - Long hypothesis (e.g., tree with 200 nodes, \(|D| = 100\)) could be coincidence
  - Resulting justification / tradeoff
    - All other things being equal, complex models tend not to generalize as well
    - Assume more model flexibility (specificity) won’t be needed later

Occam’s Razor and Decision Trees: Two Issues

- Occam’s Razor: Arguments Opposed
  - \(size(h)\) based on \(H\) - circular definition?
  - Objections to the preference bias: “fewer” not a justification

- Is Occam’s Razor Well Defined?
  - Internal knowledge representation (KR) defines which \(h\) are “short” - arbitrary?
    - e.g., single “(Sunny ∧ Normal-Humidity) ∨ Overcast ∨ (Rain ∧ Light-Wind)” test
    - Answer: \(L\) fixed; imagine that biases tend to evolve quickly, algorithms slowly

- Why Short Hypotheses Rather Than Any Other Small \(H\)?
  - There are many ways to define small sets of hypotheses
  - For any size limit expressed by preference bias, some specification \(S\) restricts \(size(h)\) to that limit (i.e., “accept trees that meet criterion \(S\)”)  
    - e.g., trees with a prime number of nodes that use attributes starting with “Z”
    - Why small trees and not trees that (for example) test \(A_1, A_1, \ldots, A_{11}\) in order?
    - What’s so special about small \(H\) based on \(size(h)\)?
      - Answer: stay tuned, more on this in Chapter 6, Mitchell
Overfitting in Decision Trees: An Example

- **Recall: Induced Tree**

  ![Boolean Decision Tree for Concept PlayTennis](image)

  - **Noisy Training Example**
    - Example 15: `<Sunny, Hot, Normal, Strong, ->`
      - Example is noisy because the correct label is +
      - Previously constructed tree misclassifies it
    - How shall the DT be revised (incremental learning)?
      - New hypothesis $h' = T'$ is expected to perform worse than $h = T$

- **Overfitting in Inductive Learning**

  - **Definition**
    - Hypothesis $h$ overfits training data set $D$ if \exists an alternative hypothesis $h'$ such that $\text{error}_D(h) < \text{error}_D(h')$ but $\text{error}_{\text{test}}(h) > \text{error}_{\text{test}}(h')$
    - Causes: sample too small (decisions based on too little data); noise; coincidence

  - **How Can We Combat Overfitting?**
    - Analogy with computer virus infection, process deadlock
    - **Prevention**
      - Addressing the problem “before it happens”
      - Select attributes that are relevant (i.e., will be useful in the model)
    - **Avoidance**
      - Sidestepping the problem just when it is about to happen
      - Holding out a test set, stopping when $h$ starts to do worse on it
    - **Detection and Recovery**
      - Letting the problem happen, detecting when it does, recovering afterward
      - Build model, remove (prune) elements that contribute to overfitting
How Can We Combat Overfitting?

- **Prevention** (more on this later)
  - Select attributes that are relevant (i.e., will be useful in the DT)
  - Predictive measure of relevance: attribute filter or subset selection wrapper

- **Avoidance**
  - Holding out a validation set, stopping when \( h \equiv T \) starts to do worse on it

How to Select “Best” Model (Tree)

- Measure performance over training data and separate validation set
- **Minimum Description Length (MDL):**
  - minimize \( \text{size}(h \equiv T) + \text{size}(\text{misclassifications}(h \equiv T)) \)

Today: Two Basic Approaches

- **Pre-pruning** (avoidance): stop growing tree at some point during construction when it is determined that there is not enough data to make reliable choices
- **Post-pruning** (recovery): grow the full tree and then remove nodes that seem not to have sufficient evidence

Methods for Evaluating Subtrees to Prune

- **Cross-validation:** reserve hold-out set to evaluate utility of \( T \) (more in Chapter 4)
- Statistical testing: test whether observed regularity can be dismissed as likely to have occurred by chance (more in Chapter 5)
- **Minimum Description Length (MDL):**
  - Additional complexity of hypothesis \( T \) greater than that of remembering exceptions?
  - Tradeoff: coding model versus coding residual error
Reduced-Error Pruning

• Post-Pruning, Cross-Validation Approach
• Split Data into Training and Validation Sets
• Function Prune(T, node)
  – Remove the subtree rooted at node
  – Make node a leaf (with majority label of associated examples)
• Algorithm Reduced-Error-Pruning (D)
  – Partition D into D_{train} (training / “growing”), D_{validation} (validation / “pruning”)
  – Build complete tree T using ID3 on D_{train}
  – UNTIL accuracy on D_{validation} decreases DO
    FOR each non-leaf node candidate in T
      Temp[candidate] ← Prune (T, candidate)
      Accuracy[candidate] ← Test (Temp[candidate], D_{validation})
    T ← T’ ∈ Temp with best value of Accuracy (best increase; greedy)
  – RETURN (pruned) T

Effect of Reduced-Error Pruning

• Reduction of Test Error by Reduced-Error Pruning
  – Test error reduction achieved by pruning nodes
  – NB: here, D_{validation} is different from both D_{train} and D_{test}
• Pros and Cons
  – Pro: Produces smallest version of most accurate T’ (subtree of T)
  – Con: Uses less data to construct T
    • Can afford to hold out D_{validation}?
    • If not (data is too limited), may make error worse (insufficient D_{train})
Rule Post-Pruning

- **Frequently Used Method**
  - Popular anti-overfitting method; perhaps most popular pruning method
  - Variant used in C4.5, an outgrowth of ID3

- **Algorithm Rule-Post-Pruning (D)**
  - Infer $T$ from $D$ (using ID3) - grow until $D$ is fit as well as possible (allow overfitting)
  - Convert $T$ into equivalent set of rules (one for each root-to-leaf path)
  - Prune (generalize) each rule independently by deleting any preconditions whose deletion improves its estimated accuracy
  - Sort the pruned rules
    - Sort by their estimated accuracy
    - Apply them in sequence on $D_{test}$

Converting a Decision Tree into Rules

- **Rule Syntax**
  - LHS: precondition (conjunctive formula over attribute equality tests)
  - RHS: class label

```
Outlook?
   Sunny Overcast Rain
humidity?
      High Normal
          Yes No Strong Light
    Wind?
        Yes No
```

- **Example**
  - IF (Outlook = Sunny) $\land$ (Humidity = High) THEN PlayTennis = No
  - IF (Outlook = Sunny) $\land$ (Humidity = Normal) THEN PlayTennis = Yes
  - ...

Boolean Decision Tree for Concept PlayTennis
Continuous Valued Attributes

- Two Methods for Handling Continuous Attributes
  - Discretization (e.g., histogramming)
    - Break real-valued attributes into ranges in advance
    - e.g., \{high = Temp > 35º C, med = 10º C < Temp ≤ 35º C, low = Temp ≤ 10º C\}
  - Using thresholds for splitting nodes
    - e.g., \( A ≤ a \) produces subsets \( A ≤ a \) and \( A > a \)
    - Information gain is calculated the same way as for discrete splits

- How to Find the Split with Highest Gain?
  - FOR each continuous attribute \( A \)
    - Divide examples \( \{x \in D\} \) according to \( x.A \)
      - FOR each ordered pair of values \((l, u)\) of \( A \) with different labels
        - Evaluate gain of mid-point as a possible threshold, i.e., \( D_A ≤ (l+u)/2 \), \( D_A > (l+u)/2 \)
  - Example
    - \( A \equiv \text{Length}: 10 \ 15 \ 21 \ 28 \ 32 \ 40 \ 50 \)
    - Class: \(+\ +\ +\ -\ +\ -\ -\)
    - Check thresholds: \( \text{Length} ≤ 12.5? \ 24.5? \ 30? \ 45? \)

Attributes with Many Values

- Problem
  - If attribute has many values, \( \text{Gain}(\cdot) \) will select it (why?)
  - Imagine using \( \text{Date} = 06/03/1996 \) as an attribute!
- One Approach: Use \( \text{GainRatio} \) instead of \( \text{Gain} \)
  \[
  \text{Gain}(D, A) = -H(D) - \sum_{v \in \text{values}(A)} \left[ \frac{P_v}{|P|} \cdot H(D_v) \right]
  \]
  \[
  \text{GainRatio}(D, A) = \frac{\text{Gain}(D, A)}{\text{SplitInformation}(D, A)}
  \]
  \[
  \text{SplitInformation}(D, A) = -\sum_{v \in \text{values}(A)} \left[ \frac{P_v}{|P|} \cdot \frac{P_v}{|P|} \log \left( \frac{|P_v|}{|P|} \right) \right]
  \]
- \( \text{SplitInformation} \): directly proportional to \( c = |\text{values}(A)| \)
- i.e., penalizes attributes with more values
  - e.g., suppose \( c_1 = c_{\text{date}} = n \) and \( c_2 = 2 \)
  - \( \text{SplitInformation} (A_1) = \log(n), \text{SplitInformation} (A_2) = 1 \)
  - If \( \text{Gain}(D, A_1) = \text{Gain}(D, A_2), \text{GainRatio} (D, A_1) \ll \text{GainRatio} (D, A_2) \)
  - Thus, preference bias (for lower branch factor) expressed via \( \text{GainRatio}(\cdot) \)
Attributes with Costs

- **Application Domains**
  - **Medical:** *Temperature* has cost $10; *BloodTestResult*, $150; *Biopsy*, $300
  - Also need to take into account *invasiveness* of the procedure (patient utility)
  - Risk to patient (e.g., amniocentesis)
  - Other units of cost
    - *Sampling time:* e.g., robot sonar (range finding, etc.)
    - Risk to artifacts, organisms (about which information is being gathered)
    - Related domains (e.g., tomography): *nondestructive evaluation*

- **How to Learn A Consistent Tree with Low Expected Cost?**
  - One approach: replace gain by *Cost-Normalized-Gain*
  - Examples of normalization functions
    - [Nunez, 1988]:
      \[ \text{Cost-Normalized-Gain}(D,A) = \frac{\text{Gain}(D,A)}{\text{Cost}(D,A)} \]
    - [Tan and Schlimmer, 1990]:
      \[ \text{Cost-Normalized-Gain}(D,A) = \frac{\text{Gain}(D,A)}{\text{Cost}(D,A)} - 1 \]
      where \( w \) determines importance of cost

---

Missing Data: Unknown Attribute Values

- **Problem:** What If Some Examples Missing Values of *A*?
  - Often, values not available for all attributes during training or testing
  - Example: medical diagnosis
    - *<Fever = true, Blood-Pressure = normal, ..., Blood-Test = ?, ...>*
    - Sometimes values truly unknown, sometimes low priority (or cost too high)
  - Missing values in learning versus classification
    - **Training:** evaluate \( \text{Gain}(D, A) \) where for some \( x \in D \), a value for *A* is not given
    - **Testing:** classify a new example \( x \) without knowing the value of *A*

- **Solutions:** Incorporating a *Guess* into Calculation of \( \text{Gain}(D, A) \)

---

<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>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cold</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cold</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cold</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

---

KSU

CIS 732: Machine Learning and Pattern Recognition
Terminology

- **Occam’s Razor and Decision Trees**
  - Preference biases: captured by hypothesis space search algorithm
  - Language biases: captured by hypothesis language (search space definition)

- **Overfitting**
  - Overfitting: \( h \) does better than \( h' \) on training data and worse on test data
  - Prevention, avoidance, and recovery techniques
    - Prevention: attribute subset selection
    - Avoidance: stopping (termination) criteria, cross-validation, pre-pruning
    - Detection and recovery: post-pruning (reduced-error, rule)

- **Other Ways to Make Decision Tree Induction More Robust**
  - Inequality DTs (decision surfaces): a way to deal with continuous attributes
  - Information gain ratio: a way to normalize against many-valued attributes
  - Cost-normalized gain: a way to account for attribute costs (utilities)
  - Missing data: unknown attribute values or values not yet collected
  - Feature construction: form of constructive induction; produces new attributes
  - Replication: repeated attributes in DTs

Summary Points

- **Occam’s Razor and Decision Trees**
  - Preference biases versus language biases
  - Two issues regarding Occam algorithms
    - Why prefer smaller trees? (less chance of “coincidence”)
    - Is Occam’s Razor well defined? (yes, under certain assumptions)
  - MDL principle and Occam’s Razor: more to come

- **Overfitting**
  - Problem: fitting training data too closely
    - General definition of overfitting
    - Why it happens
    - Overfitting prevention, avoidance, and recovery techniques

- **Other Ways to Make Decision Tree Induction More Robust**
  - Next Week: Perceptrons, Neural Nets (Multi-Layer Perceptrons), Winnow