Acknowledgment Slides were adapted from slides provided by Tom Mitchell, Carnegie-Mellon-University and Peter Geibel, University of Osnabrück
Outline

- Decision tree representation
- ID3 learning algorithm
- Which attribute is best?
- C4.5: real valued attributes
- Which hypothesis is best?
- Noise
- From Trees to Rules
- Miscellaneous
### Decision Tree Representation

<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>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>

Outlook, Temperature, etc.: attributes
PlayTennis: class
Shall I play tennis today?
Decision Tree for *PlayTennis*

- **Outlook**
  - Sunny
  - Overcast
  - Rain
    - Yes
      - Humidity
        - High
        - Normal
          - No
          - Yes
    - Weak
      - Strong
      - No
      - Yes
Alternative Decision Tree for *PlayTennis*

**Temperature**
- **hot** \{1,2,3,13\}
- **mild** \{4,8,10,11,12,14\}
- **cool** \{5,6,7,9\}

**Humidity**
- **Normal** \{1,2,3\}
- **High** \{13\}

**Wind**
- **Mild** \{1,3\}
- **Strong** \{2\}

**Outlook**
- **Sunny** \{1\}
- **Overcast** \{3\}

What is different?
Sequence of attributes influences size and shape of tree
Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\land$, $\lor$, XOR

Example XOR:

```
A
  yes  no
    B  B
     yes  no
      NO  YES
```
When to Consider Decision Trees

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data
- Interpretable result of learning is required

Examples:
- Medical diagnosis
- Text classification
- Credit risk analysis
Top-Down Induction of Decision Trees, ID3 (R. Quinlan, 1986)

ID3 operates on whole training set $S$

Algorithm:

1. create a new node

2. If current training set is sufficiently pure:
   - Label node with respective class
   - We’re done

3. Else:
   - $x \leftarrow$ the “best” decision attribute for current training set
   - Assign $x$ as decision attribute for node
   - For each value of $x$, create new descendant of node
   - Sort training examples to leaf nodes
   - Iterate over new leaf nodes and apply algorithm recursively
Example ID3

- Look at current training set $S$
  $$S = \{1, \ldots, 14\}$$

- Determine best attribute
  $$Outlook$$

- Split training set according to different values
  $$Outlook$$
  $$\begin{align*}
  \text{Sunny} & \quad \text{Overcast} \\
  \{1,2,8,9,11\} & \quad \{3,7,12,13\} \\
  \text{Rain} & \\
  \{4,5,6,10,14\}
  \end{align*}$$
Example ID3

- Tree

```
Outlook
  Sunny  Overcast  Rain
  \{1,2,8,9,11\}  \{3,7,12,13\}  \{4,5,6,10,14\}

- Apply algorithm recursively

```

Recursion  Pure -> Leaf  Recursion
Example – Resulting Tree

Outlook

Sunny
Humidity
High
No
Normal
Yes

Overcast
Yes

Rain
Wind
Strong
No
Weak
Yes

High
Normal

No
Yes
ID3 – Intermediate Summary

- Recursive splitting of the training set
- Stop, if current training set is sufficiently pure
- ... What means pure? Can we allow for errors?
- What is the best attribute?
- How can we tell that the tree is really good?
- How shall we deal with continuous values?
Which attribute is best?

- Assume a training set \{+, +, -, -, +, -, +, +, -, -\} (only classes)
- Assume binary attributes \(x_1\), \(x_2\), and \(x_3\)
- Produced splits:

<table>
<thead>
<tr>
<th></th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>{+, +, -, -, +}</td>
<td>{-, +, +, -, -}</td>
</tr>
<tr>
<td>(x_2)</td>
<td>{+}</td>
<td>{+, -, -, +, -, +, +, -, -}</td>
</tr>
<tr>
<td>(x_3)</td>
<td>{+, +, +, +, -}</td>
<td>{-, -, -, -, +}</td>
</tr>
</tbody>
</table>

- No attribute is perfect
- Which one to choose?
Entropy

- $p_\oplus$ is the proportion of positive examples
- $p_\ominus$ is the proportion of negative examples
- Entropy measures the impurity of $S$

$\text{Entropy}(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$

- Information can be seen as the negative of entropy
Examples

\[ S = \{ \text{+++++++}, \text{---} \} = \{ \text{9+}, \text{5-} \}. \text{Entropy}(S) =? \]

\[ \text{Entropy}(S) = -9/14 \log(9/14) - 5/14 \log(5/14) = 0.94 \]

\[ S = \{ \text{+++++++}, \text{---} \} = \{ \text{8+}, \text{6-} \}. \text{Entropy}(S) =? \]

(Größer oder kleiner oder gleich)?

\[ \text{Entropy}(S) = -8/14 \log(8/14) - 6/14 \log(6/14) = 0.98 \]

\[ S = \{ \text{+++++++}, \text{++++} \} = \{ \text{14+} \}. \text{Entropy}(S) =? \]

\[ \text{Entropy}(S) = 0 \]

\[ S = \{ \text{+++++++}, \text{---} \} = \{ \text{7+}, \text{7-} \}. \text{Entropy}(S) =? \]

\[ \text{Entropy}(S) = 1 \]
Information Gain

- Measuring attribute $x$ creates subsets $S_1$ and $S_2$ with different entropies

- Taking the mean of $\text{Entropy}(S_1)$ and $\text{Entropy}(S_2)$ gives conditional entropy $\text{Entropy}(S|x)$, i.e. in general:
  $$\text{Entropy}(S|x) = \sum_{v \in \text{Values}(x)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- Choose that attribute that maximizes difference:
  $$\text{Gain}(S, x) := \text{Entropy}(S) - \text{Entropy}(S|x)$$

- $\text{Gain}(S, x) =$ expected reduction in entropy due to partitioning on $x$
  $$\text{Gain}(S, x) = \text{Entropy}(S) - \sum_{v \in \text{Values}(x)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$
## Example - Training Set

<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>
Example

\[ Gain(S, x) = Entropy(S) - \sum_{v \in Values(x)} \frac{|S_v|}{|S|} Entropy(S_v) \]

For top node: \( S = \{9+, 5-\} \), \( Entropy(S) = 0.94 \) (see previous slide)

Attribute Wind:
\( S_{weak} = \{6+, 2-\}, |S_{weak}| = 8 \)
\( S_{strong} = \{3+, 3-\}, |S_{strong}| = 6 \)
\( Entropy(S_{weak}) = \)
\[ -\frac{6}{8} \log\left(\frac{6}{8}\right) - \frac{2}{8} \log\left(\frac{2}{8}\right) = 0.81 \]
\( Entropy(S_{strong}) = \)
\[ = 1 \]

Expected Entropy when assuming attribute 'Wind':
\( Entropy(S|Wind) = \)
\[ \frac{8}{14} Entropy(S_{weak}) + \frac{6}{14} Entropy(S_{strong}) = 0.89 \]
\( Gain(S, Wind) = \)
\[ 0.94 - 0.89 \approx 0.05 \]
Selecting the Next Attribute

- For whole training set:
  \[ \text{Gain}(S, \text{Outlook}) = 0.246 \]
  \[ \text{Gain}(S, \text{Humidity}) = 0.151 \]
  \[ \text{Gain}(S, \text{Wind}) = 0.048 \]
  \[ \text{Gain}(S, \text{Temperature}) = 0.029 \]
- → \textit{Outlook} should be used to split training set!
- Further down in the tree, \text{Entropy}(S) is computed locally
- Usually, the tree does not have to be minimized
- Reason of good performance of ID3!
Real-Valued Attributes

- **Temperature = 82.5**
- Create discrete attributes to test continuous:
  - \((Temperature > 54) = true\) or \(false\)
  - Sort attribute values that occur in training set:
    | Temperature | 40 | 48 | 60 | 72 | 80 | 90 |
    |-------------|----|----|----|----|----|----|
    | PlayTennis  | No | No | Yes| Yes| Yes| No |
  - Determine points where the class changes
    - Candidates are \((48 + 60)/2\) and \((80 + 90)/2\)
- Select best one using info gain
- Implemented in the system C4.5 (successor of ID3)
Hypothesis Space Search by ID3

...
Hypothesis Space Search by ID3

- Hypothesis $H$ space is complete:
  - This means that every function on the feature space can be represented
  - Target function surely in there for a given training set
- The training set is only a subset of the instance space
- Generally, several hypotheses have minimal error on training set
- Best is one that minimizes error on instance space
  - ... cannot be determined because only finite training set is available
  - Feature selection is shortsighted
  - ... and there is no back-tracking $\rightarrow$ local minima...
- ID3 outputs a single hypothesis
Inductive Bias in ID3

- **Inductive Bias** corresponds to explicit or implicit prior assumptions on the hypothesis
  - E.g. hypothesis space $H$ (language for classifiers)
  - Search bias: how to explore $H$
  - Bias here is a preference for some hypotheses, rather than a restriction of hypothesis space $H$

- Bias of ID3:
  - Preference for short trees,
  - and for those with high information gain attributes near the root

- **Occam's razor**: prefer the shortest hypothesis that fits the data

- How to justify Occam’s razor?
Occam’s Razor

• Why prefer short hypotheses?
• Argument in favor:
  – Fewer short hyps. than long hyps.
  → A short hyp that fits data unlikely to be coincidence
  → A long hyp that fits data might be coincidence
• Bayesian Approach: A probability distribution on the hypothesis space is assumed.
  – The (unknown) hypothesis \( h_{gen} \) was picked randomly
  – The finite training set was generated using \( h_{gen} \)
  – We want to find the most probable hypothesis \( h' \approx h_{gen} \) given the current observations (training set).
Consider adding noisy (=wrongly labeled) training example #15:

\[\text{Sunny, Mild, Normal, Weak, PlayTennis} = \text{No}\]

i.e. outlook = sunny, humidity = normal

What effect on earlier tree?

```
Outlook
\[\begin{array}{c}
\text{Sunny} \\
\text{Overcast} \\
\text{Rain}
\end{array}\]
```

```
\begin{array}{c}
\text{Humidity} \\
\text{Wind}
\end{array}
\begin{array}{c}
\text{High} \\
\text{Normal}
\end{array}
\begin{array}{c}
\text{Strong} \\
\text{Weak}
\end{array}
\begin{array}{c}
\text{No} \\
\text{Yes}
\end{array}
\begin{array}{c}
\text{No} \\
\text{Yes}
\end{array}
```
Overfitting in Decision Trees

- Algorithm will introduce new test
- Unnecessary, because new example was erroneous due to the presence of Noise
- → Overfitting corresponds to learning coincidental regularities
- Unfortunately, we generally don’t know which examples are noisy
- ... and also not the amount, e.g. percentage, of noisy examples
Overfitting

Consider error of hypothesis \( h \) over

- training data \((x_1, k_1), \ldots, (x_d, k_d)\): training error

\[
error_{\text{train}}(h) = \frac{1}{d} \sum_{i=1}^{d} L(h(x_i), k_i)
\]

with loss function \( L(c, k) = 0 \) if \( c = k \) and \( L(c, k) = 1 \) otherwise

- entire distribution \( D \) of data \((x, k)\): true error

\[
error_D(h) = P(h(x) \neq k)
\]

Definition Hypothesis \( h \in H \) overfits training data if there is an alternative \( h' \in H \) such that

\[
error_{\text{train}}(h) < error_{\text{train}}(h') \quad \text{and} \quad error_D(h) > error_D(h')
\]
Overfitting in Decision Tree Learning

- The accuracy is estimated on a separate test set.
- Learning produces more and more complex trees (horizontal axis).
Avoiding Overfitting

1. How can we avoid overfitting?
   • Stop growing when data split not statistically significant (pre-pruning)
     – e.g. in C4.5: Split only, if there are at least two descendant that have at least \( n \) examples, where \( n \) is a parameter
   • Grow full tree, then post-prune (post-prune)

2. How to select “best” tree:
   • Measure performance over training data
   • Measure performance over separate validation data set
   • Minimum Description Length (MDL): minimize
     
     \[
     \text{size}(\text{tree}) + \text{size}(\text{misclassifications}(\text{tree}))
     \]
Reduced-Error Pruning

1. An example for post-pruning
2. Split data into *training* and *validation* set
3. Do until further pruning is harmful:
   (a) Evaluate impact on *validation* set of pruning each possible node (plus those below it)
   (b) respective node is labeled with most frequent class
   (c) Greedily remove the one that most improves *validation* set accuracy
4. Produces smallest version of most accurate subtree
5. What if data is limited?
Rule Post-Pruning

1. **Grow tree** from given training set that fits data best, and allow overfitting
2. **Convert** tree to equivalent set of rules
3. **Prune** each rule independently of others
4. **Sort** final rules into desired sequence for use

- Perhaps most frequently used method (e.g., C4.5)
- allows more fine grained pruning
- converting to rules increases **understandability**
Converting A Tree to Rules

\[
\begin{align*}
\text{IF} & \quad (Outlook = Sunny) \land (Humidity = High) \\
\text{THEN} & \quad PlayTennis = No \\
\text{IF} & \quad (Outlook = Sunny) \land (Humidity = Normal) \\
\text{THEN} & \quad PlayTennis = Yes \\
\ldots
\end{align*}
\]
Special Aspects: Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- For example, imagine using Date as attribute (very many values!) (e.g. $Date = Day1, ...$)
- Sorting by date, the training data can be perfectly classified
- this is a general phenomenon with attributes with many values, since they split the training data in small sets.
- but: generalisation suffers!

One approach: use $GainRatio$ instead
Special Aspects: Gain Ratio

Idea: Measure how broadly and uniformly A splits the data:

\[ SplitInformation(S, A) \equiv - \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \]

where \( S_i \) is subset of \( S \) for which \( A \) has value \( v_i \) and \( c \) is the number of different values.

Example:

- Attribute 'Date': \( n \) examples are completely separated. Therefore:
  \[ SplitInformation(S, 'Date') = - \sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = - \log_2 \frac{1}{n} = \log_2 n \]

- Other extreme: binary attribute splits data set in two even parts:
  \[ SplitInformation(S, Data) = - \sum_{i=1}^{2} \frac{1}{2} \log_2 \frac{1}{2} = - \log_2 \frac{1}{2} = 1 \]

By considering as a splitting criterion the

\[ GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)} \]

one relates the Information gain to the way, the examples are split
Special Aspects: Attributes with Costs

• Consider
  – medical diagnosis, BloodTest has cost $150
  – robotics, Width_from_1ft has cost 23 sec.

• How to learn a consistent tree with low expected cost?

• One approach: replace gain by
  – Tan and Schlimmer (1990): \( \frac{\text{Gain}^2(S,A)}{\text{Cost}(A)} \)
  – Nunez (1988): \( \frac{2\text{Gain}(S,A) - 1}{(\text{Cost}(A)+1)^w} \)

• Note that not the misclassification costs are minimized, but the costs of classifying
Special Aspects: Unknown Attribute Values

• What if an example $x$ has a missing value for attribute $A$?
• To compute gain $(S, A)$ two possible strategies are:
  – Assign most common value of $A$ among other examples with same target value $c(x)$
  – Assign a probability $p_i$ to each possible value $v_i$ of $A$
• Classify new examples in same fashion
Summary

- Decision trees are a symbolic representation of knowledge
- → Understandable for humans
- Learning:
  - Incremental, e.g., CAL2
  - Batch, e.g., ID3
- Issues:
  - Pruning
  - Assessment of Attributes (Information Gain)
  - Continuous attributes
  - Noise