Intelligent Systems

Rule Learning
**Where are we?**

<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Propositional Logic</td>
</tr>
<tr>
<td>3</td>
<td>Predicate Logic</td>
</tr>
<tr>
<td>4</td>
<td>Reasoning</td>
</tr>
<tr>
<td>5</td>
<td>Search Methods</td>
</tr>
<tr>
<td>6</td>
<td>CommonKADS</td>
</tr>
<tr>
<td>7</td>
<td>Problem-Solving Methods</td>
</tr>
<tr>
<td>8</td>
<td>Planning</td>
</tr>
<tr>
<td>9</td>
<td>Software Agents</td>
</tr>
<tr>
<td><strong>10</strong></td>
<td><strong>Rule Learning</strong></td>
</tr>
<tr>
<td>11</td>
<td>Inductive Logic Programming</td>
</tr>
<tr>
<td>12</td>
<td>Formal Concept Analysis</td>
</tr>
<tr>
<td>13</td>
<td>Neural Networks</td>
</tr>
<tr>
<td>14</td>
<td>Semantic Web and Services</td>
</tr>
</tbody>
</table>
• Slides in this lecture are partially based on [1], Section 3 “Decision Tree Learning”.

Tom Mitchell “Machine Learning”
Outline

• Motivation
• Technical Solution
  – Rule learning tasks
  – Rule learning approaches
    • Specialization
      – The ID3 Algorithm
    • Generalization
      – The RELAX Algorithm
    • Combining specialization and generalization
      – The JoJo Algorithm
  – Refinement of rule sets with JoJo
• Illustration by a Larger Example
• Extensions
  – The C4.5 Algorithm
• Summary
MOTIVATION
Motivation: Definition of Learning in AI

- Research on learning in AI is made up of diverse subfields (cf. [5])
  - “Learning as adaptation”: Adaptive systems monitor their own performance and attempt to improve it by adjusting internal parameters, e.g.,
    - Self-improving programs for playing games
    - Pole balancing
    - Problem solving methods
  - “Learning as the acquisition of structured knowledge” in the form of
    - Concepts
    - Production rules
    - Discrimination nets
Machine learning (ML) is concerned with the design and development of algorithms that allow computers to change behavior based on data. A major focus is to automatically learn to recognize complex patterns and make intelligent decisions based on data (Source: Wikipedia).

Driving question: “How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?” (cf. [2])

Three niches for machine learning [1]:
- Data mining: using historical data to improve decisions, e.g. from medical records to medical knowledge
- Software applications hard to program by hand, e.g. autonomous driving or speech recognition
- Self-customizing programs, e.g., a newsreader that learns users interest

Practical success of ML:
- Speech recognition
- Computer vision, i.e. to recognize faces, to automatically classify microscope images of cells, or for recognition of hand writings
- Bio surveillance, i.e. to detect and track disease outbreaks
- Robot control
• Machine Learning is a central research area in AI to acquire knowledge [1].

• Rule learning
  – is a popular approach for discovering interesting relations in data sets and to acquire structured knowledge.
  – is a means to alleviate the „bottleneck“ (knowledge acquisition) – problem in expert systems [4].
Simple Motivating Example

- $I = \{\text{milk, bread, butter, beer}\}$
- Sample database

<table>
<thead>
<tr>
<th>ID</th>
<th>Milk</th>
<th>Bread</th>
<th>Butter</th>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- A possible rule would be $\{\text{milk, bread}\} \rightarrow \{\text{butter}\}$

TECHNICAL SOLUTIONS
Many inductive knowledge acquisition algorithms generate ("induce") classifiers in form of decision trees.

A decision tree is a simple recursive structure for expressing a sequential classification process.
- Leaf nodes denote classes
- Intermediate nodes represent tests

Decision trees classify instances by sorting them down the tree from the root to some leaf node which provides the classification of the instance.
Decision Tree: Example

{Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong}: Negative Instance
• Rules can represent a decision tree:
  
  \[
  \text{if item1 then subtree1} \\
  \text{elseif item2 then subtree2} \\
  \text{elseif...}
  \]

• There are as many rules as there are leaf nodes in the decision tree.

• Advantage of rules over decision trees:
  – Rules are a widely-used and well-understood representation formalism for knowledge in expert systems;
  – Rules are easier to understand, modify and combine; and
  – Rules can significantly improve classification performance by eliminating unnecessary tests.
  – Rules make it possible to combine different decision trees for the same task.
Advantages of Decision Trees

• **Decision trees are simple to understand.**
  People are able to understand decision trees model after a brief explanation.

• **Decision trees have a clear evaluation strategy**
  Decision trees are easily interpreted

• **Decision trees are able to handle both nominal and categorical data.**
  Other techniques are usually specialised in analysing datasets that have only one type of variable. Ex: neural networks can be used only with numerical variables
Advantages of Decision Trees (1)

• **Decision trees are a white box model.**
  If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. An example of a black box model is an artificial neural network since the explanation for the results is excessively complex to be comprehended.

• **Decision trees are robust, perform well with large data in a short time.**
  Large amounts of data can be analysed using personal computers in a time short enough to enable stakeholders to take decisions based on its analysis.
Converting a Decision Tree to Rules

IF \((\text{Outlook} = \text{Sunny}) \land (\text{Humidity} = \text{High})\) THEN \(\text{PlayTennis} = \text{No}\)

IF \((\text{Outlook} = \text{Sunny}) \land (\text{Humidity} = \text{Normal})\) THEN \(\text{PlayTennis} = \text{Yes}\)
RULE LEARNING TASKS
There are two major rule learning tasks:

1. Learning of correct rule that cover a maximal number of positive examples but no negative ones. We call this *maximal requirement for each rule*.

2. Learning of a minimal rule set that covers all positive example. We call this *minimal requirement for the rule set*.
Rule learning tasks

To illustrate the two rule learning tasks let’s consider a data set, containing both positive and negative examples, as depicted in the following chart.
Maximal requirement for each rule

We want to learn a correct rule that covers a maximal number of positive examples but no negative examples (maximal requirement for each rule).
Maximal requirement for each rule

We start with deriving a rule that cover the positive example \((x=3;y=3)\).
If \(x=3\) and \(y=3\) then class = positive
Maximal requirement for each rule

We try to cover more positive examples by refining the rule. We need to make sure that we don’t cover negative examples. The new rule:

if \( x \geq 2 \) and \( x \leq 4 \) and \( y = 3 \) then class = positive

covers positive example \((x=2; y=3), (x=3; y=3), (x=4; y=3)\)
We try to cover more positive examples by further refining the rule. The new rule if \( x \geq 2 \) and \( x \leq 4 \) and \( y \geq 3 \) and \( y \leq 4 \) then class = positive covers positive example \((x=2;y=3),(x=3;y=3),(x=4;y=3), (x=2;y=4),(x=3;y=4), (x=4;y=4)\)
Maximal requirement for each rule

We try to cover more positive examples by further refining the rule. The new rule if $x \geq 2$ and $x \leq 4$ and $y \geq 2$ and $y \leq 5$ then class = positive covers all positive example in the lower part of the space.
We can not refine anymore the rule

if \( x \geq 2 \) and \( x \leq 4 \) and \( y \geq 2 \) and \( y \leq 5 \) then class = positive

to cover more positive without covering negative examples as well. To cover the rest of the positive examples we need to learn a new rule.
Maximal requirement for each rule

Following the same approach as illustrated before we can learn a second rule that covers the rest of the positive examples:

if $x \geq 5$ and $x \leq 6$ and $y \geq 6$ and $y \leq 8$ then class = positive
The full set of positive examples is covered by the following two rules:

if $x \geq 2$ and $x \leq 4$ and $y \geq 2$ and $y \leq 5$ then class = positive

if $x \geq 5$ and $x \leq 6$ and $y \geq 6$ and $y \leq 8$ then class = positive
In general, for a decision problem, there are many possible sets of rules that cover all positive examples without covering negative examples. Possible sets of rules for our previous example are:

**Solution 1:**

if $x\geq 2$ and $x\leq 4$ and $y\geq 2$ and $y\leq 5$ then class = positive
if $x\geq 5$ and $x\leq 6$ and $y\geq 6$ and $y\leq 8$ then class = positive
Solution 2:

if $x \geq 2$ and $x \leq 4$ and $y \geq 2$ and $y \leq 3$ then class = positive
if $x \geq 2$ and $x \leq 4$ and $y > 3$ and $y \leq 5$ then class = positive
if $x \geq 5$ and $x \leq 6$ and $y \geq 6$ and $y \leq 7$ then class = positive
if $x \geq 5$ and $x \leq 6$ and $y > 7$ and $y \leq 8$ then class = positive
Minimal requirement for the rule set

- The minimal requirement for the rule set is about learning of a minimal rule set that covers all positive examples.
- The minimal requirement for the rule set is similar with the *minimal set cover problem* [18].
- The minimal requirement for the rule set, which basically translates to building the optimal linear decision tree, is NP-hard [19].
RULE LEARNING APPROACHES
Lattice of possible rules
Rule learning approaches

- Rule learning can be seen as a search problem in the lattice of possible rules
- Depending on the starting point and on the direction of search we can distinguished three classes of rule learning approaches:
  1. **Specialization** – the search procedure starts at the top of the lattice and searches towards the bottom of the lattice, towards the concrete descriptions. Specialization is **top-down**.
  2. **Generalization** – the search procedure starts at the bottom of the lattice and advances up the lattice towards the top element. Generalization is **bottom-up**.
  3. **Combining of specialization and generalization** – the search procedure can start at any arbitrary point in the lattice and can move freely up or down as needed.
Rule learning approaches

- Specialization and Generalization are dual search directions in a given rule set.
SPECIALIZATION
Specialization

- Specialization algorithms start from very general descriptions and specializes those until they are correct.
- This is done by adding additional premises to the antecedent of a rule, or by restricting the range of an attribute which is used in an antecedent.

- Algorithms relying on specialization generally have the problem of overspecialization: previous specialization steps could become unnecessary due to subsequent specialization steps.
- This brings along the risk for ending up with results that are not maximal-general.

- Some examples of (heuristic) specialization algorithms are the following: ID3, AQ, C4, CN2, CABRO, FOIL, or PRISM; references at the end of the lecture.
THE ID3 ALGORITHM
Most algorithms apply a top-down, greedy search through the space of possible trees.

- e.g., ID3 [5] or its successor C4.5 [6]

**ID3**

- Learns trees by constructing them top down.
- Initial question: “Which attribute should be tested at the root of the tree?” -> each attribute is evaluated using a statistical test to see how well it classifies.
- A descendant of the root node is created for each possible value of this attribute.
- Entire process is repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.
The Basic ID3 Algorithm

ID3(Examples, Target_attribute, Attributes)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a Root node for the tree
- If all Examples are positive, Return the single-node tree Root, with label = +
- If all Examples are negative, Return the single-node tree Root, with label = −
- If Attributes is empty, Return the single-node tree Root, with label = most common value of Target_attribute in Examples
- Otherwise Begin
  - \( A \leftarrow \) the attribute from Attributes that best* classifies Examples
  - The decision attribute for Root \( \leftarrow A \)
  - For each possible value, \( v_i \), of A,
    - Add a new tree branch below Root, corresponding to the test \( A = v_i \)
    - Let Examples\(_{v_i}\) be the subset of Examples that have value \( v_i \) for A
    - If Examples\(_{v_i}\) is empty
      - Then below this new branch add a leaf node with label = most common value of Target_attribute in Examples
      - Else below this new branch add the subtree
        ID3(Examples\(_{v_i}\), Target_attribute, Attributes − \{A\})
  - End
- Return Root

*Code taken from [1]*
Selecting the Best Classifier

- Core question: Which attribute is the best classifier?
- ID3 uses a statistical measure called information gain that measures how well a given example separates the training examples according to their target classification.
Entropy

- **Entropy** characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection $S$ (containing positive and negative examples), the entropy of $S$ relative to this boolean classification is

$$
Entropy(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-
$$

- $p_+$/$p_-$ is the proportion of positive/negative examples in $S$.
- Entropy is 0 if all members of $S$ belong to the same class; entropy is 1 when the collection contains an equal number of positive and negative examples.
- More general: 

$$
Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i
$$
Information Gain

- **Information gain** measures the expected reduction in entropy caused by partitioning the examples according to an attribute:

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

- First term: entropy of the original collection \(S\); second term: expected value of entropy after \(S\) is partitioned using attribute \(A\) (\(S_v\) subset of \(S\)).

- \(Gain(S,A)\): The expected reduction in entropy caused by knowing the value of attribute \(A\).

- ID3 uses information gain to select the best attribute at each step in growing the tree.
Example: Which attribute is the best classifier?

$S$: [9+, 5-]
$E = 0.940$

Gain ($S, \text{Humidity}$)

$= 0.940 - (7/14) \cdot 0.985 - (7/14) \cdot 0.592$

$= 0.151$

$S$: [9+, 5-]
$E = 0.940$

$S$: [6+, 3-, 1-]
$E = 0.592$

$S$: [3+, 4-]
$E = 0.985$

$S$: [6+, 2-]
$E = 0.811$

$S$: [3+, 3-]
$E = 1.00$

Gain ($S, \text{Wind}$)

$= 0.940 - (8/14) \cdot 0.811 - (6/14) \cdot 1.0$

$= 0.048$

Figure taken from [1]
Hypothesis Space Search in Decision Tree Learning

- ID3 performs a simple-to-complex hill-climbing search through the space of possible decision trees (the hypothesis search).
- Start: empty tree; progressively more elaborate hypotheses in search of a decision tree are tested.

Figure taken from [1]
Capabilities and Limitations of ID3

• ID3’s hypothesis space is the complete space of finite discrete-valued functions, relative to the available attributes.
• ID3 maintains only a single current hypothesis, as it searches through the space of trees (earlier (perhaps better) versions are eliminated).
• ID3 performs no backtracking in search; it converges to locally optimal solutions that are not globally optimal.
• ID3 uses all training data at each step to make statistically based decisions regarding how to refine its current hypothesis.
Inductive Bias in Decision Tree Learning

• Definition: Inductive bias is the set of assumptions that, together with the training data, deductively justifies the classifications assigned by the learner to future instances [1].

• Central question: How does ID3 generalize from observed training examples to classify unseen instances? What is its inductive bias?

• ID3 search strategy:
  – ID3 chooses the first acceptable tree it encounters.
  – ID3 selects in favour of shorter trees over longer ones.
  – ID3 selects trees that place the attributes with the highest information gain closest to the root.

• Approximate inductive bias of ID3: Shorter trees are preferred over larger trees. Trees that place high information gain attributes close to the root are preferred over those that do not.
ID3’s Inductive Bias

• ID3 searches a complete hypothesis search.
• It searches incompletely through this space, from simple to complex hypotheses, until its termination condition is met.
  – Termination condition: Discovery of a hypothesis consistent with the data

• ID3’s inductive bias is a preference bias (it prefers certain hypotheses over others).
• Alternatively: A restriction bias categorically restricts considered hypotheses.
Occam’s Razor

• …or why prefer short hypotheses?

• William Occam was one of the first to discuss this question around year 1320.
• Several arguments have been discussed since then.

• Arguments:
  – Based on combinatorial arguments there are fewer short hypothesis that coincidentally fit the training data.
  – Very complex hypotheses fail to generalize correctly to subsequent data.
ID3 by Example ([1])

<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>

- Target attribute: PlayTennis (values: yes / no)
Example: Which attribute should be tested first in the tree?

- ID3 determines information gain for each candidate attribute (e.g., Outlook, Temperature, Humidity, and Wind) and selects the one with the highest information gain.

  \[
  \begin{align*}
  \text{Gain}(S, \text{Humidity}) &= 0.246; \quad \text{Gain}(S, \text{Humidity}) = 0.151; \quad \text{Gain}(S, \text{Wind}) = 0.048; \quad \text{Gain}(S, \text{Temperature}) = 0.029
  \end{align*}
  \]
Example: Resulting Tree

\[
\begin{align*}
&\{D_1, D_2, \ldots, D_{14}\} \\
&[9+, 5-] \\
&\text{Outlook} \\
&\text{Sunny} \\
&\{D_1, D_2, D_8, D_9, D_{11}\} \\
&[2+, 3-] \\
&? \\
&\text{Overcast} \\
&\{D_3, D_7, D_{12}, D_{13}\} \\
&[4+, 0-] \\
&\text{Yes} \\
&\text{Rain} \\
&\{D_4, D_5, D_6, D_{10}, D_{14}\} \\
&[3+, 2-] \\
&? \\
\end{align*}
\]

Which attribute should be tested here?
Example: Continue Selecting New Attributes

- Determining new attributes at the „Sunny“ – branch only using the examples classified there:

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

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

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

\[
\text{Gain} (S_{\text{sunny}}, \text{Wind}) = .970 - \frac{2}{5} \times 1.0 - \frac{3}{5} \times .918 = .019
\]

- Process continues for each new leaf node until either of two conditions are met:
  1. Every attribute has already been included along this path through the tree, or
  2. The training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).
Example: Final Decision Tree

- **Outlook**
  - Sunny
  - Overcast
  - Rain
    - **Humidity**
      - High
        - No
      - Normal
        - Yes
    - **Wind**
      - Strong
        - No
      - Weak
        - Yes
GENERALIZATION
Generalization

- Generalization starts from very special descriptions and generalizes them as long as they are not incorrect, i.e. in every step some unnecessary premises are deleted from the antecedent.
- The generalization procedure stops if no more premises to remove exist.

- Generalization avoids the maximal-general issue of specialization, in fact it guarantees most-general descriptions.
- However, generalization of course risks to derive final results that are not most-specific.

- RELAX is an example of a generalization-based algorithm; references at the end of the lecture.
THE RELAX ALGORITHM
• RELAX is a generalization algorithm, and proceeds as long as the resulting rule set is not incorrect.

• The RELAX algorithm:
  – RELAX considers every example as a specific rule that is generalized.
  – The algorithm then starts from a first rule and relaxes the first premise.
  – The resulting rule is tested against the negative examples.
  – If the new (generalized) rule covers negative examples, the premise is added again, and the next premise is relaxed.
  – A rule is considered minimal, if any further relaxation would destroy the correctness of the rule.

• The search for minimal rules starts from any not yet considered example, i.e. examples that are not covered by already discovered minimal rules.
• Consider the following positive example for a consequent C: 
  (pos, (x=1, y=0, z=1))
• This example is represented as a rule: $x \cap \neg y \cap z \rightarrow C$
• In case of no negative examples, RELAX constructs and tests the following set of rules:
  1) $x \cap \neg y \cap z \rightarrow C$
  2) $\neg y \cap z \rightarrow C$
  3) $x \cap z \rightarrow C$
  4) $x \cap \neg y \rightarrow C$
  5) $x \rightarrow C$
  6) $\neg y \rightarrow C$
  7) $z \rightarrow C$
  8) $\rightarrow C$
COMBINING SPECIALIZATION AND GENERALIZATION
THE JOJO ALGORITHM
• In general, it cannot be determined which search direction is the better one.

• JoJo is an algorithm that combines both search directions in one heuristic search procedure.

• JoJo can start at an arbitrary point in the lattice of complexes and generalizes and specializes as long as the quality and correctness can be improved, i.e. until a local optimum can be found, or no more search resources are available (e.g., time, memory).
While specialization moves solely from ‘Top’ towards ‘Bottom’ and generalization from ‘Bottom’ towards ‘Top’, JoJo is able to move freely in the search space.

Either of the two strategies can be used interchangeable, which makes JoJo more expressive than comparable algorithms that apply the two in sequential order (e.g. ID3).
A starting point in JoJo is described by two parameters:
- Vertical position (length of the description)
- Horizontal position (chosen premises)

Reminder: JoJo can start at any arbitrary point, while specialization requires a highly general point and generalization requires a most specific point.

In general, it is possible to carry out several runs of JoJo with different starting points. Rules that were already produced can be used as subsequent starting points.
• Criteria for choosing a vertical position:
  1. Approximation of possible length or experience from earlier runs.
  2. Random production of rules; distribution by means of the average correctness of the rules with the same length (so-called quality criterion).
  3. Start with a small sample or very limited resources to discover a real starting point from an arbitrary one.
  4. Randomly chosen starting point (same average expectation of success as starting with ‘Top‘ or ‘Bottom‘).
  5. Heuristic: few positive examples and maximal-specific descriptions suggest long rules, few negative examples and maximal-generic descriptions rather short rules.
JoJo Principle Components

- JoJo consists of three components: The **Specializer**, **Generalizer**, and **Scheduler**
- The former two can be provided by any such components depending on the chosen strategies and preference criterias.
- The Scheduler is responsible for selecting the next description out of all possible generalizations and specializations available (by means of a t-preference, total preference).
- Simple example scheduler:
  - Specialize, if the error rate is above threshold;
  - Otherwise, choose the best generalization with allowable error rate;
  - Otherwise stop.
REFINEMENT OF RULES WITH JOJO
Refinement of rules with JoJo

- Refinement of rules refers to the modification of a given rule set based on additional examples.
- The input to the task is a so-called hypothesis (a set of rules) and a set of old and new positive and negative examples.
- The output of the algorithm are a refined set of rules and the total set of examples.
- The new set of rules is correct, complete, non-redundant and (if necessary) minimal.
• There is a four step procedure for the refinement of rules:
  1. Rules that become incorrect because of new examples are refined: incorrect rules are replaced by new rules that cover the positive examples, but not the new negative ones.
  2. Complete the rule set to cover new positive examples.
  3. Redundant rules are deleted to correct the rule set.
  4. Minimize the rule set.

• Steps 1 and 2 are subject to the algorithm JoJo that integrates generalization and specification via a heuristic search procedure.
Refinement of rules with JoJo (3)

- Correctness:
  - Modify overly general rules that cover too many negative examples.
  - Replace a rule by a new set of rules that cover the positive examples, but not the negative ones.
- Completeness:
  - Compute new correct rules that cover the not yet considered positive examples (up to a threshold).
- Non-redundancy:
  - Remove rules that are more specific than other rules (i.e. rules that have premises that are a superset of the premises of another rule).
ILLUSTRATION BY A LARGER EXAMPLE: ID3
## ID3 Example: The Simpsons

<table>
<thead>
<tr>
<th>Person</th>
<th>Hair Length</th>
<th>Weight</th>
<th>Age</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homer</td>
<td>0”</td>
<td>250</td>
<td>36</td>
<td>M</td>
</tr>
<tr>
<td>Marge</td>
<td>10”</td>
<td>150</td>
<td>34</td>
<td>F</td>
</tr>
<tr>
<td>Bart</td>
<td>2”</td>
<td>90</td>
<td>10</td>
<td>M</td>
</tr>
<tr>
<td>Lisa</td>
<td>6”</td>
<td>78</td>
<td>8</td>
<td>F</td>
</tr>
<tr>
<td>Maggie</td>
<td>4”</td>
<td>20</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>Abe</td>
<td>1”</td>
<td>170</td>
<td>70</td>
<td>M</td>
</tr>
<tr>
<td>Selma</td>
<td>8”</td>
<td>160</td>
<td>41</td>
<td>F</td>
</tr>
<tr>
<td>Otto</td>
<td>10”</td>
<td>180</td>
<td>38</td>
<td>M</td>
</tr>
<tr>
<td>Krusty</td>
<td>6”</td>
<td>200</td>
<td>45</td>
<td>M</td>
</tr>
</tbody>
</table>

| Comic  | 8”          | 290    | 38  | ?     |

Example from: www.cs.sjsu.edu/~lee/cs157b/ID3-AllanNeymark.ppt
ID3 Example: The Simpsons

Entropy($S$) = $- \frac{p}{p+n} \log_2 \left( \frac{p}{p+n} \right) - \frac{n}{p+n} \log_2 \left( \frac{n}{p+n} \right)$

Entropy(4F,5M) = -(4/9)log$_2$(4/9) - (5/9)log$_2$(5/9) = 0.9911

Entropy(1F,3M) = -(1/4)log$_2$(1/4) - (3/4)log$_2$(3/4) = 0.8113

Entropy(3F,2M) = -(3/5)log$_2$(3/5) - (2/5)log$_2$(2/5) = 0.9710

Gain($A$) = $E(\text{Current set}) - \sum E(\text{all child sets})$

Gain(Hair Length <= 5) = 0.9911 – (4/9 * 0.8113 + 5/9 * 0.9710 ) = 0.0911
Gain(A) = E(Current set) − ∑E(all child sets)

Gain(Weight <= 160) = 0.9911 − (5/9 * 0.7219 + 4/9 * 0) = 0.5900
### ID3 Example: The Simpsons

**Entropy**

\[
Entropy(S) = - \frac{p}{p+n} \log_2 \left( \frac{p}{p+n} \right) - \frac{n}{p+n} \log_2 \left( \frac{n}{p+n} \right)
\]

**Entropy(4F,5M)** = \(-(4/9)\log_2(4/9) - (5/9)\log_2(5/9)\)  
= \textbf{0.9911}

**Entropy(3F,3M)** = \(-(3/6)\log_2(3/6) - (3/6)\log_2(3/6)\)  
= \textbf{1}

**Entropy(1F,2M)** = \(-(1/3)\log_2(1/3) - (2/3)\log_2(2/3)\)  
= \textbf{0.9183}

**Gain(A)** = \(E(\text{Current set}) - \sum E(\text{all child sets})\)

**Gain(Age <= 40)** = \textbf{0.9911} – (6/9 * 1 + 3/9 * 0.9183) = \textbf{0.0183}

Let us try splitting on **Age**
Of the 3 features we had, *Weight* was best. But while people who weigh over 160 are perfectly classified (as males), the under 160 people are not perfectly classified… So we simply recurse!

This time we find that we can split on *Hair length*, and we are done!
We need don’t need to keep the data around, just the test conditions.

How would these people be classified?

ID3 Example: The Simpsons

- **Weight <= 160?**
  - **yes**
  - **no**

- **Hair Length <= 2?**
  - **yes**
  - **no**

- **Male**
- **Female**
It is trivial to convert Decision Trees to rules…

Rules to Classify Males/Females

If Weight greater than 160, classify as **Male**
Elseif Hair Length less than or equal to 2, classify as **Male**
Else classify as **Female**
EXTENSIONS
Issues in Decision Tree Learning

• How deeply to grow the decision tree?
• Handling continuous attributes.
• Choosing an appropriate attribute selection measure.
• Handling training data with missing attribute values.
• Handling attributes with differing costs.
• Improving computational efficiency.

• Successor of ID3, named **C4.5**, addresses most of these issues (see [6]).
THE C4.5 ALGORITHM
Avoiding Overfitting the Data

- **Definition:** Given a hypothesis space \( H \), a hypothesis \( h \in H \) is said to **overfit** the training data if there exists some alternative hypothesis \( h' \in H \), such that \( h' \) has smaller error than \( h' \) over the training examples, but \( h' \) has a smaller error than \( h \) over the entire distribution of instances. [1]

- ID3 grows each branch of the tree just deeply enough to perfectly classify the training examples.
  - This can lead to difficulties when there is noise in the data or when the number of training examples is too small to produce a representative sample of the true target function.
  - Overfitting trees could be produced!

Impact of overfitting in a typical application of decision tree learning [1]
Example: Overfitting due to Random Noise

- Original decision tree based on correct data:

  ![Decision Tree Diagram]

  - Incorrectly labelled data leads to construction of a more complex tree, e.g., {Outlook=Sunny, Temperature=Hot, Humidity=Normal, Wind=Strong, PlayTennis=No}
    - ID3 would search for new refinements at the bottom of the left tree.
Strategies to Avoid Overfitting

• Overfitting might occur based on erroneous input data or based on coincidental regularities.
• Different types of strategies:
  – Stopping to grow the tree earlier, before it reaches the point where it perfectly classifies the training data.
  – Allowing the tree to overfit the data, and then post-prune the tree (more successful approach).

• Key question: How to determine the correct final tree size?
  – Use of a separate set of examples to evaluate the utility of post-pruning nodes from the tree (“Training and Validation Set” – approach); two approaches applied by Quinlair: “Reduced Error Pruning” and “Rule-Post Pruning”
  – Use all available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set.
  – Use an explicit measure of the complexity for encoding the training examples and the decision trees, halting growth of the tree when this encoding size is minimized (“Minimum Decision Length” – principle, see [1] Chapter 6)
Rule Post-Pruning

- Applied in C4.5.
- Steps ([1])
  1. Infer the decision tree from the training set, growing the set until the training data is fit as well as possible and allowing overfitting to occur.
  2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
  3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
  4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.
- Rule accuracy estimation based on the training set using a pessimistic estimate: C4.5 calculates standard deviation and takes the lower bound as estimate for rule performance.
Incorporating Continuous-Valued Attributes

- ID3 is restricted to attributes with discrete values.
- Continuous-valued attributes - example:

<table>
<thead>
<tr>
<th>Temperature</th>
<th>40</th>
<th>48</th>
<th>60</th>
<th>72</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

- C4.5 allows continuous-valued attributes in the decision nodes inside the tree.
- Approach: Dynamic definition of new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals (e.g. \( A_c \) is true if \( A < c \), false otherwise).
- Central question: How to select the best value for a threshold \( c \)?
Computing a Threshold

- **Goal:** Determining a threshold \( c \) that produces the greatest information gain.
- **Approach:**
  1. Sort examples according to the continuous attribute \( A \).
  2. Identify adjacent examples that differ in their target classification.
  3. Generate a set of candidate thresholds midway between the corresponding values of \( A \) (\( c \) must always lie at such a boundary, see [5]).
  4. Evaluate candidate thresholds by computing the information gain associated with each.
- **Example:**
  - Two candidate thresholds at which the value of PlayTennis changes: \((48+60)/2, (80+90)/2\)
  - Information gain for the first threshold is higher.
Alternative Measures for Selecting Attributes

- Natural bias in the information gain measure favours attributes with many values over those with few values (e.g. a “Date” – field)
- Example:
  - Date – attributes has (too) many possible values; thus it separates the training examples into very small subsets.
  - It would have a very high information gain.
  - But it would be a very poor predictor of the target function over unseen instances
- Alternative selection measure: gain ratio (see [5]).
Gain Ratio

- Gain ratio penalizes attributes such as Date by incorporating **split information** (entropy of S with respect to A) that is sensitive to how broadly and uniformly the attribute splits the data:

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

- Definition GainRatio:

\[
GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}
\]
Handling Training Examples with Missing Attribute Values

- If data with missing attribute values exist, it is common to estimate the missing attribute value based on other examples for which the attribute has a known value.
- Common strategy: Assign value that is most common among training examples at a certain node \( n \) that have the same classification as the current example.
- More complex strategy used in C4.5: Assign a probability to each of the possible values.
  - Probabilities can be estimated based on the observed frequencies of various values of a certain attribute among examples at a certain node \( n \), e.g., \( A(x) = 1: 0.6 \)
  - Fractional of probability 0.6 is propagated down the branch of the tree for \( A=1 \).
  - Fractional examples are used to compute information gain.
  - Classification of a new instance is simply the most probable classification (computed by summing weights of instance fragments classified in different ways at leaf nodes of the tree).
SUMMARY
• Machine learning is a prominent topic in the field of AI.
• Rule learning is a means to learn rules from instance data to classify unseen instances.
• Decision tree learning can be used for concept learning, rule learning, or for learning of other discrete valued functions.
• The ID3 family of algorithms infers decision trees by growing them from the root downward in a greedy manner.
• ID3 searches a complete hypothesis space.
• ID3’s inductive bias includes a preference for smaller trees; it grows trees only as large as needed.
• A variety of extensions to basic ID3 have been developed; extensions include: methods for post-pruning trees, handling real-valued attributes, accommodating training examples with missing attribute values, or using alternative selection measures.
Summary (2)

• Rules cover positive examples and should not cover negative examples.
• There are two main approaches for determining rules:
  – Generalization
  – Specification
• RELAX is a presented example of a generalization algorithm
• JoJo combines generalization and specialization and allows the algorithm to traverse the entire search space by either generalizing or specializing rules inter-changeably.
• JoJo can also be applied to incrementally refine rule sets.
REFERENCES
• Mandatory reading:

• Further reading:
References

References


References

- **Wikipedia links:**
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Propositional Logic</td>
</tr>
<tr>
<td>3</td>
<td>Predicate Logic</td>
</tr>
<tr>
<td>4</td>
<td>Reasoning</td>
</tr>
<tr>
<td>5</td>
<td>Search Methods</td>
</tr>
<tr>
<td>6</td>
<td>CommonKADS</td>
</tr>
<tr>
<td>7</td>
<td>Problem-Solving Methods</td>
</tr>
<tr>
<td>8</td>
<td>Planning</td>
</tr>
<tr>
<td>9</td>
<td>Software Agents</td>
</tr>
<tr>
<td>10</td>
<td>Rule Learning</td>
</tr>
<tr>
<td>11</td>
<td><strong>Inductive Logic Programming</strong></td>
</tr>
<tr>
<td>12</td>
<td>Formal Concept Analysis</td>
</tr>
<tr>
<td>13</td>
<td>Neural Networks</td>
</tr>
<tr>
<td>14</td>
<td>Semantic Web and Services</td>
</tr>
</tbody>
</table>
Questions?