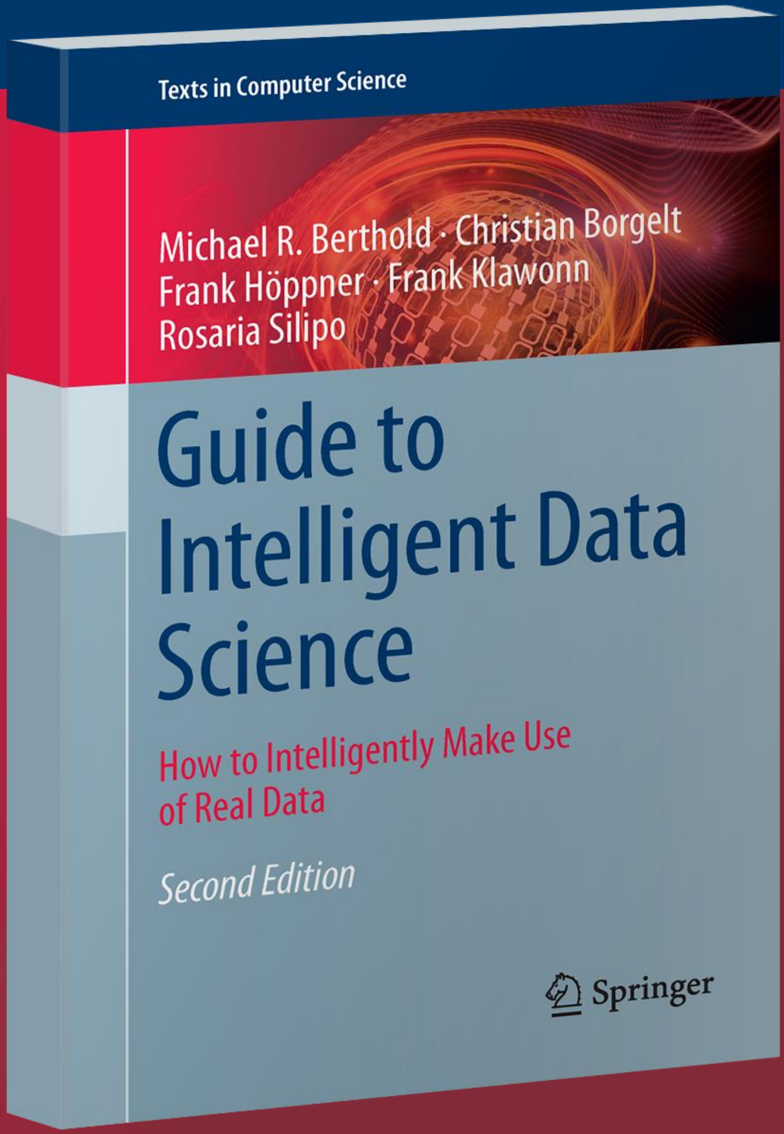
Rule Learning



Summary of this lesson

“All models are wrong but some are useful.”
-George Box

Can we use *rules* as models?

**This lesson refers to chapter 8 of the GIDS book*

Content of this lesson

- Propositional Rules
- Rule Learners
- Geometrical Rule Learners
- Heuristic Rule Learners

Propositional Rules

- Rules consisting of atomic facts and their combinations using logical operators

$$IF\ \underline{x_1 \leq 10\ AND\ x_3 = red}\ THEN\ \underline{class\ A}$$

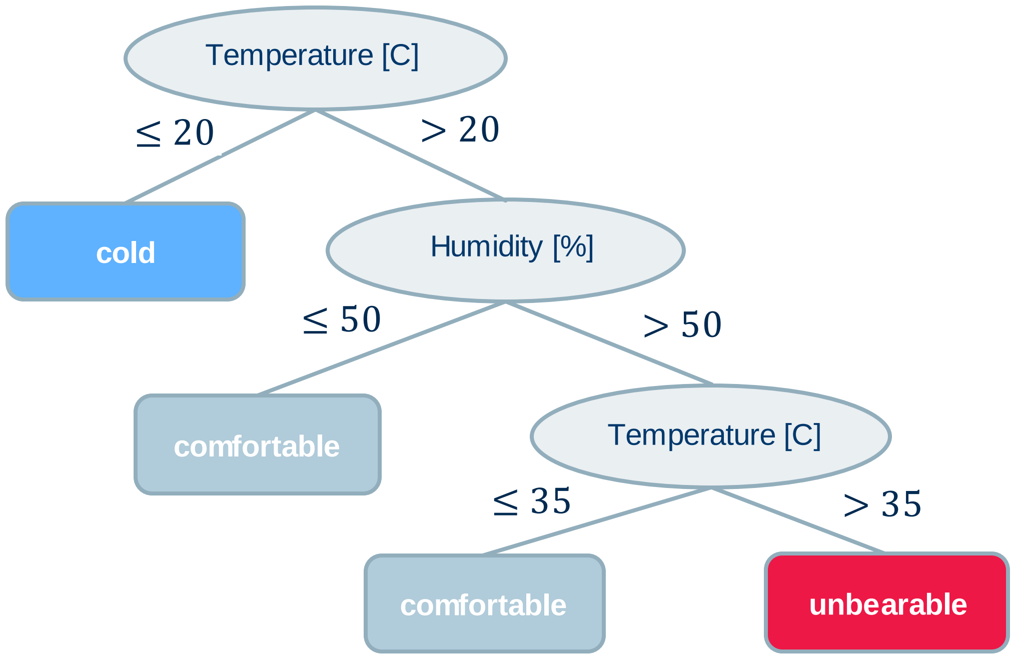
Antecedent
→ Indicating conditions to be fulfilled

Consequent
→ True when conditions are met

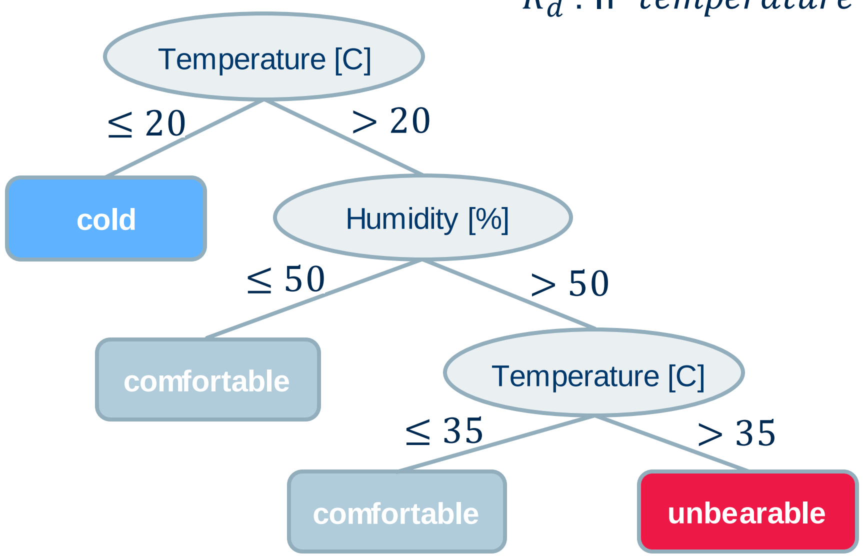
Atomic facts

- Numeric attributes: e.g., <, >, =, etc.
- Nominal attributes: e.g., =, $\in \{set\}$, etc.
- Ordinal attributes: e.g., <, >, =, $\in \{set\}$, $\in [interval]$, etc.

- Consider a decision tree:

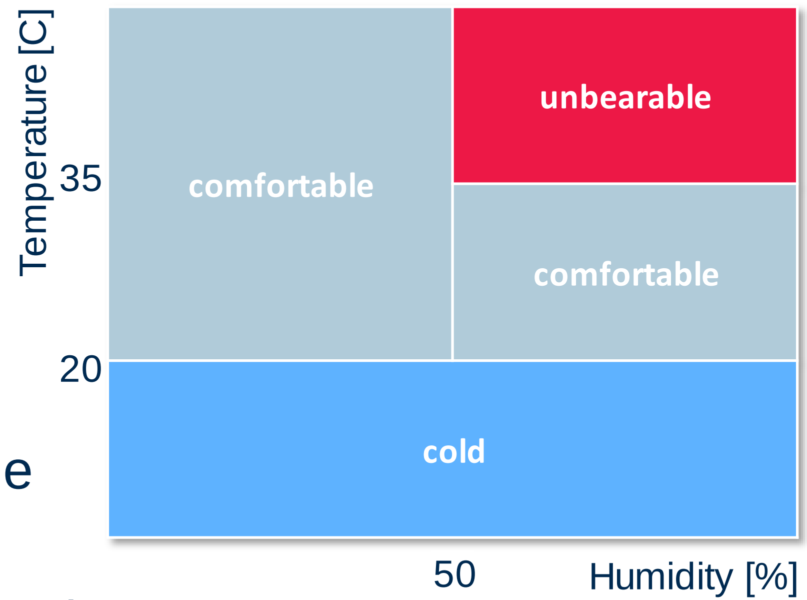


- Rules can be extracted from a decision tree
  - R_a : IF $temperature \leq 20$ THEN class "cold"
  - R_b : IF $temperature > 20$ AND $\text{humidity} \leq 50$ THEN class "comf"
  - R_c : IF $temperature \in (20{,}35]$ AND $\text{humidity} > 50$ THEN class "comf"
  - R_d : IF $temperature > 35$ AND $\text{humidity} > 50$ THEN class "unbearable"



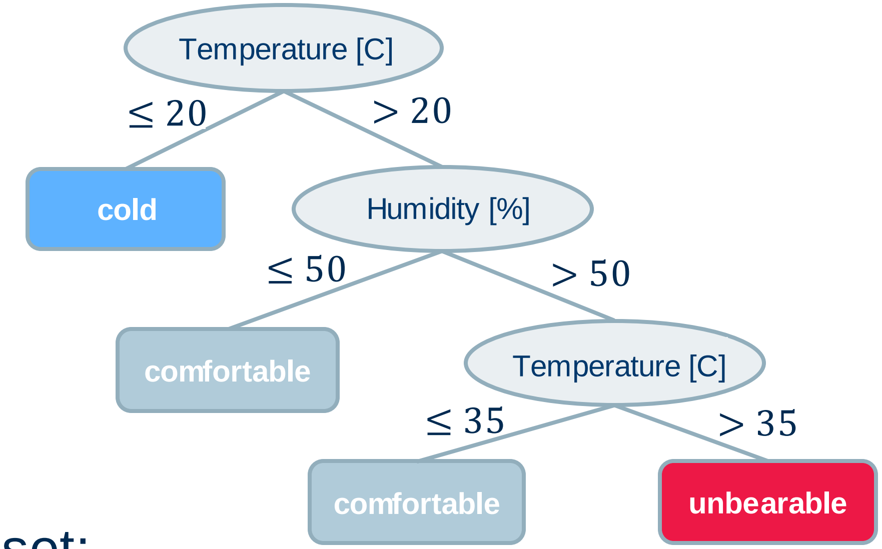
Rules from a decision tree are:

- Mutually exclusive (no overlap)
- Unordered
- Complete (covers the entire data)



Problems with rules from a decision tree:

- Instability (due to recursive nature of the trees)
- Redundancy (splitting constraints appear in multiple rules)



- Non-redundant and ordered rule set:
  - R_1 : IF $temperature \leq 20$ THEN class "cold"
  - R_2 : IF $\mathrm{humidity} \leq 50$ THEN class "comfortable"
  - R_3 : IF $temperature \leq 35$ THEN class "comfortable"
  - R_4 : class "unbearable"
- Rules have to be examined in the order

Rule Learners

Categorization of propositional rule learners:

- Supported attribute types
  - Nominal only ➔ relatively small hypothesis space
  - Numerical only ➔ geometrical rule learners
  - Mixed attributes ➔ more complex heuristics needed

- Learning strategies
  - Specializing
  - Generalizing

- Example
- Given a training instance $(\boldsymbol{x}, k)$ with $\boldsymbol{x} = (12, 3.5, red)$, an initial special rule looks like:

$$IF\ x_1 = 12\ AND\ x_2 = 3.5\ AND\ x_3 = \ red \qquad THEN\ class\ k$$

- With a second sample $(\boldsymbol{x}, k)$ with $\boldsymbol{x} = (12,.3\ 3.5, blue)$, the rule is generalized as:

$$IF x_1 \in [12,12.3]\ AND\ x_2 = 3.5\ AND \quad x_3 \in \{red, blue\} \quad THEN\ class\ k$$

Two main options for generalization exist:

- Generalize existing rule to cover one more pattern
- Merge two existing rules

The resulting training algorithms generally are:

- Greedy
  - Complete search of merge tree is infeasible
- Differ in
  - The choice of rules / patterns to merge
  - The used stopping criteria

Specialization follows the same principle

- Start with very general rules

IF true THEN class k

- Iteratively specialize the rule

Finding a Set of Rules

So far we only generalized/specialized one rule.

- Most real world data sets are too complex to be explained by one rule only.

- Many rule learning algorithms wrap the learning of one rule into an outer loop based on set covering strategy (sequential covering):
  - attempts to build most important rules first
  - iteratively adds smaller / less important rules

Geometrical Rule Learners

- Limited to numerical attributes (of comparable magnitudes)

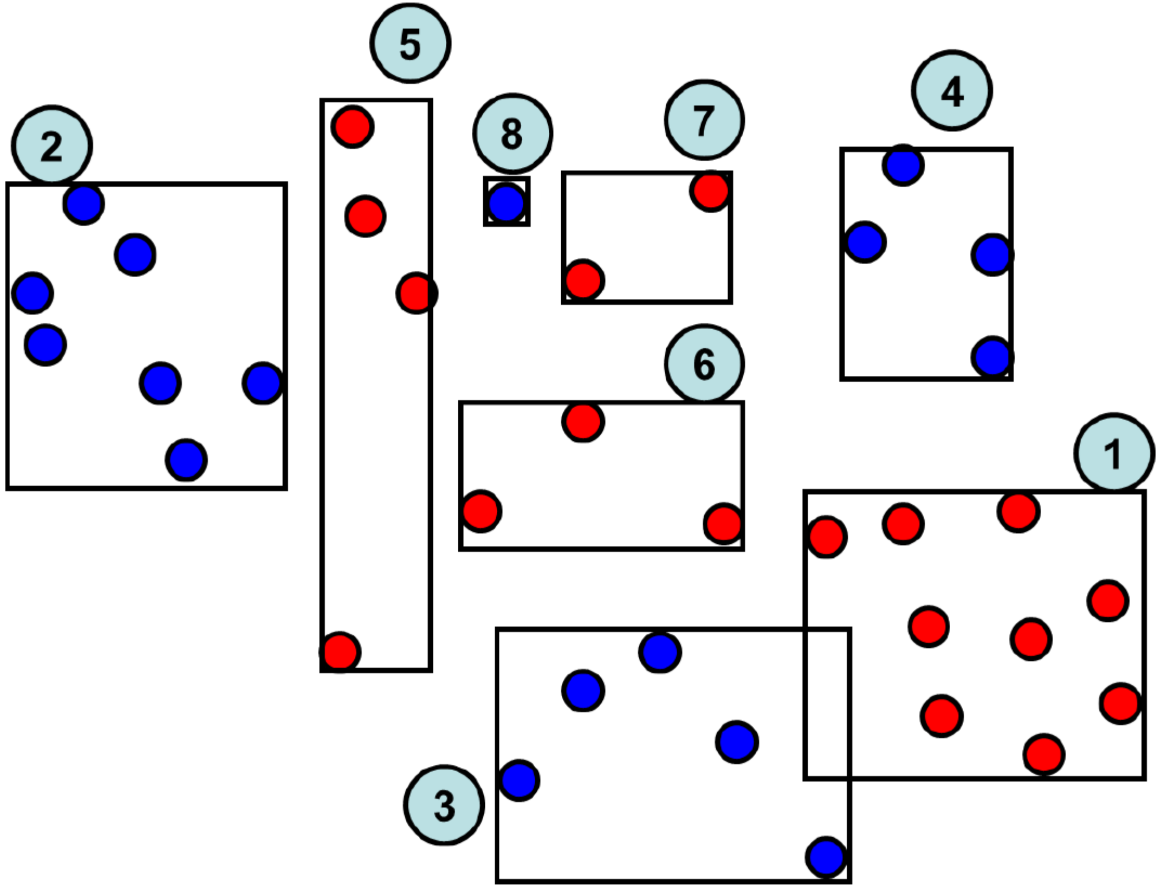
Goal:

- Find rectangular area(s) that are occupied only by patterns for one class
- Such areas represent a rule:

$$IF\ x_1 \in [a_1, b_1] \wedge \cdots \wedge \cdots \wedge x_n \in [a_n, b_n] \quad THEN\ class\ k$$

- Keep creating rules until no more useful rule can be found

Example – Geometric Rule Learners



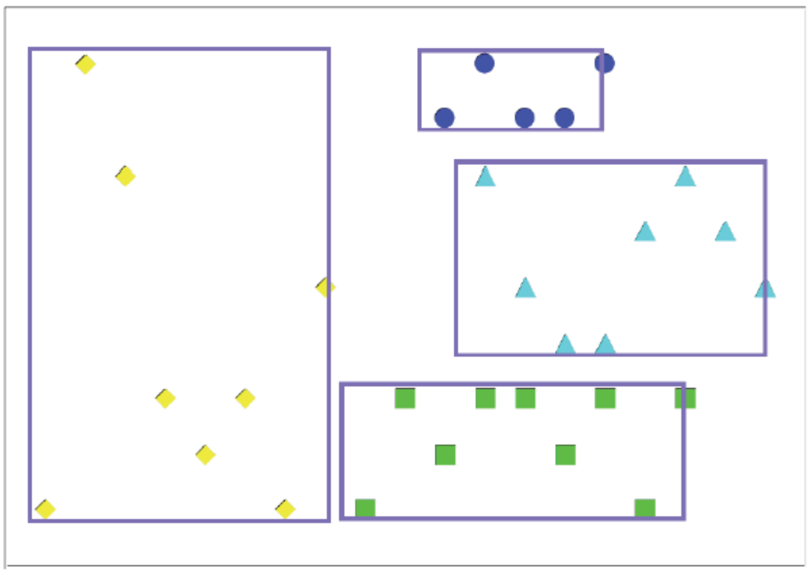
To find a rule:

- Draw a random starting point
- Find a rectangular area around the point, with points belonging to the same class

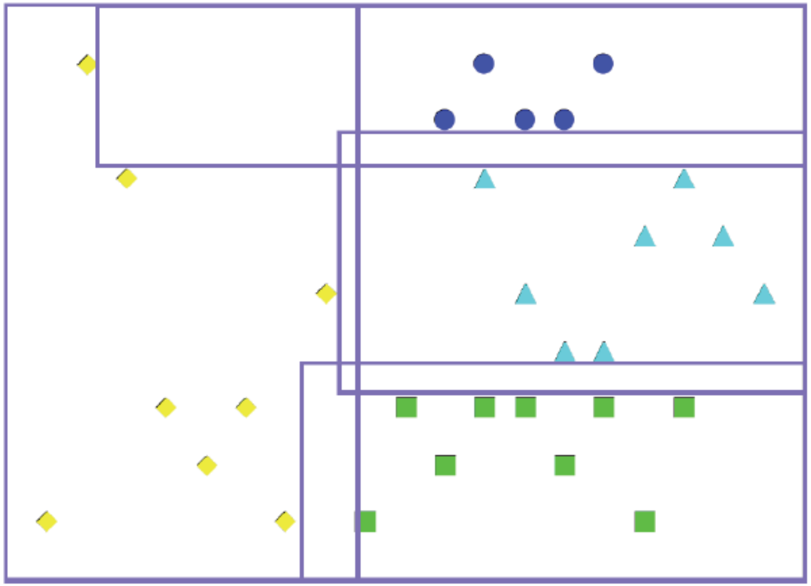
When possible

- Find nearest neighbors of the same class
- Generalize rectangles to includes this point

Specialized



Generalized



CN2 Rule Learning Algorithm

- Prominent, early example of rule learning algorithm
- Set covering approach
- Greedy algorithm rule specialization
- Simple heuristic for most important rule selection

Algorithm BuildRuleSet($D, p_{\min}$)

input:	training data D
parameter:	performance threshold $p_{\min}$
output:	a rule set R matching D with performance $\geq p_{\min}$

$R = \emptyset$
$D_{\text{rest}} = D$
while $(\text{Performance}(R, D_{\text{rest}}) < p_{\min})$
    $r = \text{FindOneGoodRule}(D_{\text{rest}})$
    $R = R \cup \{r\}$
    $D_{\text{rest}} = D_{\text{rest}} - \text{covered}(r, D_{\text{rest}})$
endwhile
return R

Heuristic Rule Learners

How do we evaluate the accuracy A of a rule?

- Base assumption:

$$A(IF\ Conditions\ THEN\ class\ k) = p(k/Conditions)$$

- Estimating the probability using relative frequencies

$$p(k/Conditions) = \frac{\#\ covered\ correct}{\#\ covered\ total}$$

- Relative frequency of covered correctly:

$$p(k/R) = \frac{\#\ covered\ correct}{\#\ covered\ total}$$

➔ Problems with small samples

- Laplace estimate

$$p(k/R) = \frac{\#\ covered\ correct + 1}{\#\ covered\ total + \#\ classes}$$

➔ Assumes uniform prior distribution of classes

- m-estimate:

$$p(k/R) = \frac{\#\ covered\ correct + m \cdot p(k)}{\#\ covered\ total + m}$$

- Where:

$$p(k) = \frac{1}{\#\ classes} \quad and \quad m = \#\ classes$$

- Special case:
- Takes into account prior class probabilities
- Independent of number of classes
- m is domain dependent (more noise, larger m)

FindOneGoodRule

Algorithm FindOneGoodRule(D_{rest})

---

input: (subset of) training data D_{rest}

output: one good rule r explaining some instances of the training data

---

$h_{\text{best}} = \text{true}$ // most general hypothesis

$H_{\text{candidates}} = \{h_{\text{best}}\}$

while $H_{\text{candidates}} \neq \emptyset$

  $H_{\text{candidates}} = \text{specialize}\,(H_{\text{candidates}})$

  $h_{\text{best}} = \arg\max_{h \in H_{\text{candidates}} \cup \{h_{\text{best}}\}} \{\text{Performance}(h, D_{\text{rest}})\}$

  $\text{update}(H_{\text{candidates}})$ // clean up

endwhile

return $'\text{IF } h_{\text{best}} \text{ THEN } \arg\max_k \{|\text{covered}_k(h_{\text{best}}, D_{\text{rest}})|\}'$

Limitations of Propositional Rules

- Propositional rule learners cannot express rules such as:

$$IF\ x\ \ is\ father\ of\ y\ \ AND\ y\ \ is\ female\ \ \ THEN\ y\ \ is\ daughter\ of\ \ x$$

- They would need to cover training examples for all possible (x,y) combinations

➔ For this, other types of rules are more appropriate

Thank you

For any questions please contact: education@knime.com