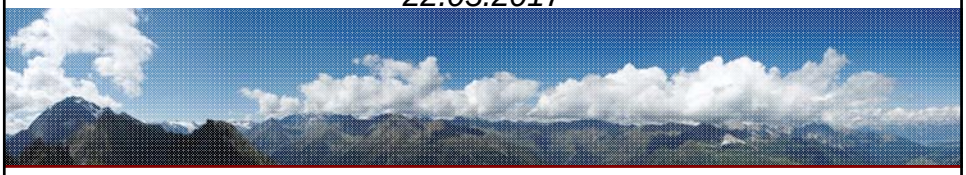
Artificial Intelligence
SS 2017

Rule Learning

Anna Fensel

22.05.2017



Where are we?

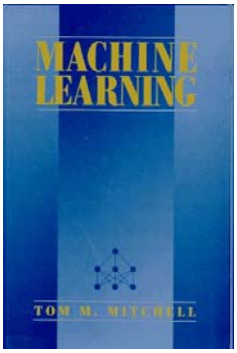
STI · INNSBRUCK

#	Title
0.1	Propositional Logic
0.2	Predicate Logic
1	Introduction
2	Reasoning
3	Search Methods
4	CommonKADS
5	Problem-Solving Methods
6	Planning
7	Software Agents
8	**Rule Learning**
9	Inductive Logic Programming
10	Neural Networks
11	Semantic Web and Services
12	Exam

Tom Mitchell "Machine Learning"

STI · INNSBRUCK

- Slides in this lecture are partially based on [1], Section 3 "Decision Tree Learning".



3

Outline

STI · INNSBRUCK

- 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

4

STI · INNSBRUCK

MOTIVATION

Motivation: Definition of Learning in AI

STI · INNSBRUCK

- 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

STI · INNSBRUCK

- 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

7

Rule learning

STI · INNSBRUCK

- 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 aquisition) – problem in expert systems [4].

8

Simple Motivating Example

STI · INNSBRUCK

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

ID	Milk	Bread	Butter	Beer
1	1	1	1	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

- A possible rule would be {milk, bread} → {butter}

Source: Wikipedia

9

STI · INNSBRUCK

TECHNICAL SOLUTIONS

10

Decision Trees

STI · INNSBRUCK

- 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.

11

Decision Tree: Example

STI · INNSBRUCK

{Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong}: **Negative Instance**

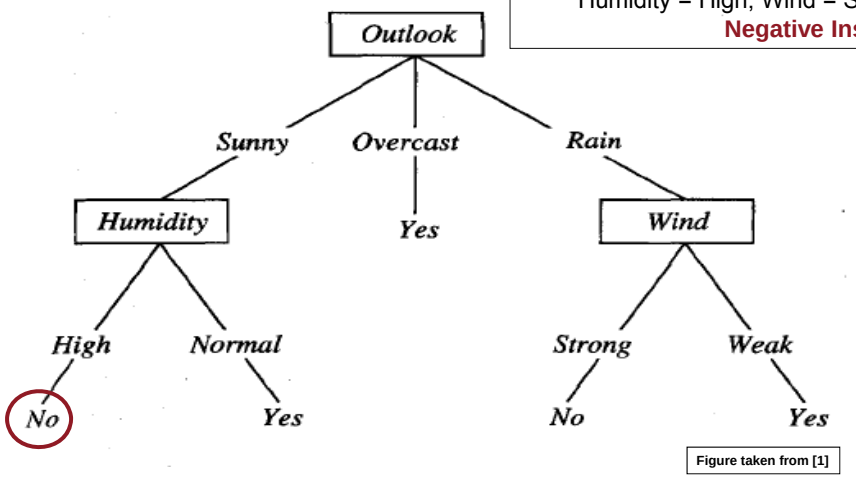


Figure taken from [1]

12

Decision Trees and Rules

STI · INNSBRUCK

- Rules can represent a decision tree:
  if item1 then subtree1
  elseif item2 then subtree2
  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.

13

Advantages of Decision Trees

STI · INNSBRUCK

- **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.

14

Advantages of Decision Trees (1)

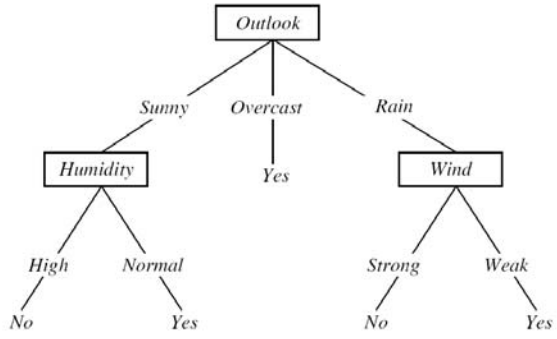
STI · INNSBRUCK

- **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.

15

Converting a Decision Tree to Rules

STI · INNSBRUCK



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

16

STI · INNSBRUCK

RULE LEARNING TASKS

STI · INNSBRUCK

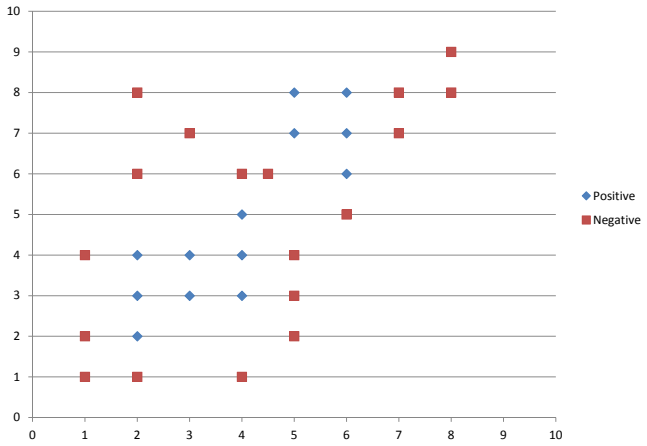
There are two major rule learning tasks:

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

STI · INNSBRUCK

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

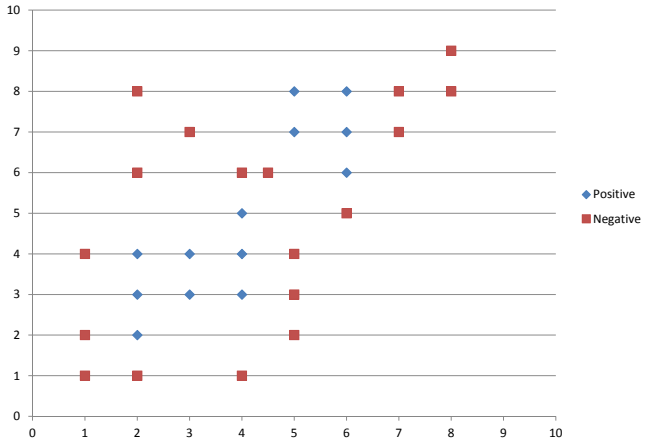


19

Maximal requirement for each rule

STI · INNSBRUCK

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



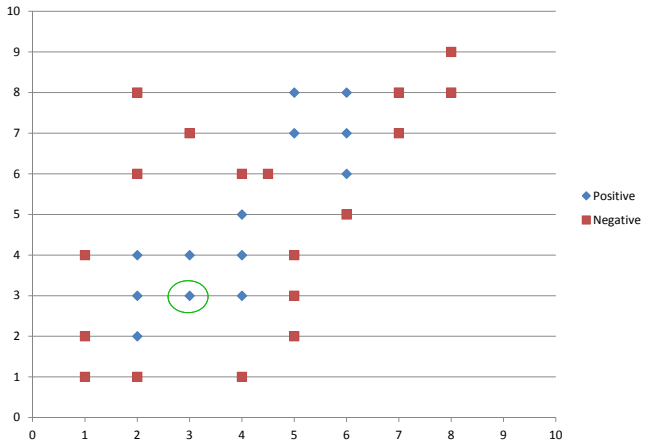
20

Maximal requirement for each rule

STI · INNSBRUCK

We start with deriving a rule that cover the positive example (x=3;y=3).

If x=3 and y=3 then class = positive



21

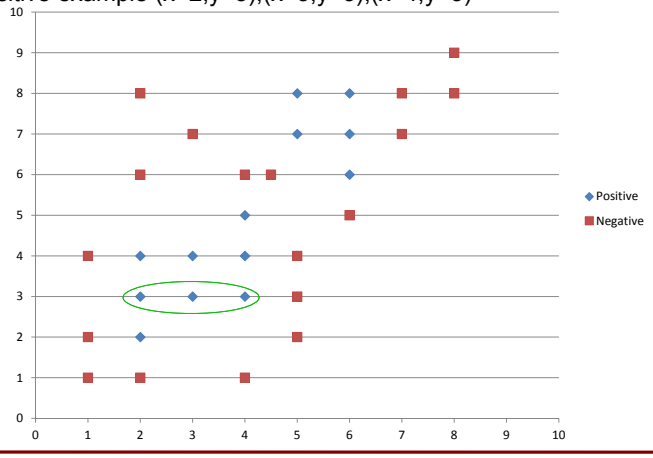
Maximal requirement for each rule

STI · INNSBRUCK

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>=2 and x<=4 and y=3 then class = positive

covers positive example (x=2;y=3),(x=3;y=3),(x=4;y=3)



22

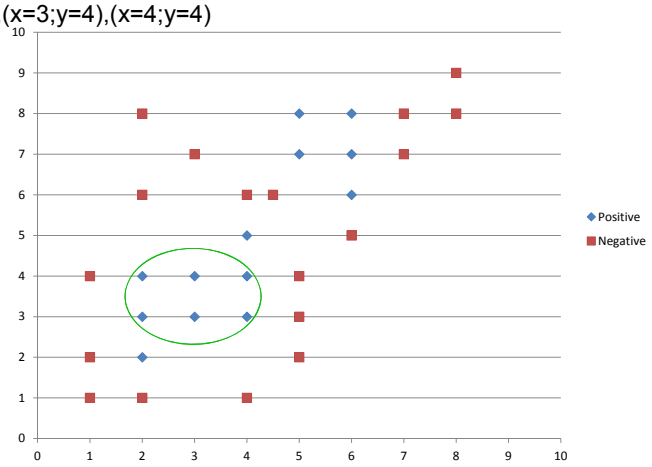
Maximal requirement for each rule

STI · INNSBRUCK

We try to cover more positive examples by further refining the rule. The new rule

if x>=2 and x<=4 and y>=3 and y<=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)



23

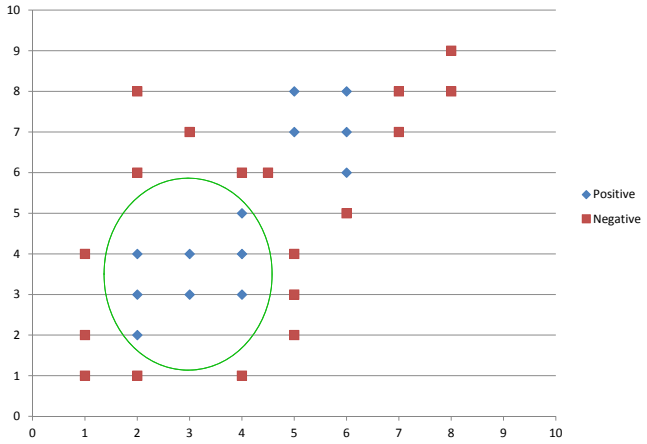
Maximal requirement for each rule

STI · INNSBRUCK

We try to cover more positive examples by further refining the rule. The new rule

if x>=2 and x<=4 and y>=2 and y<=5 then class = positive

covers all positive example in the lower part of the space



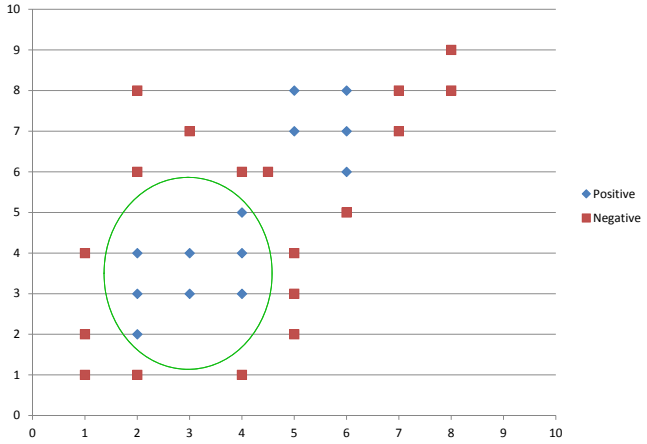
24

Maximal requirement for each rule

STI · INNSBRUCK

We can not refine anymore the rule

if x>=2 and x<=4 and y>=2 and y<=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.



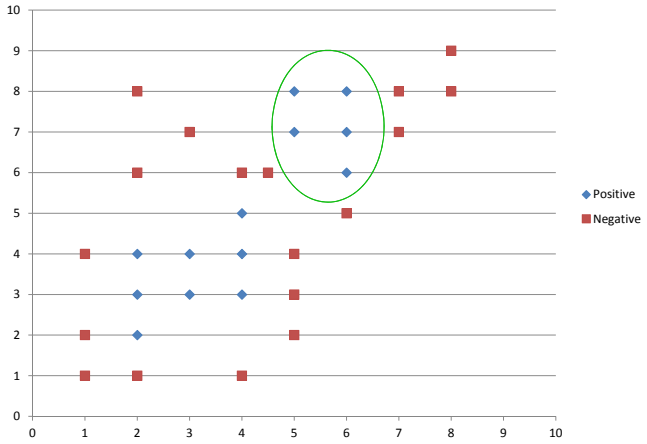
25

Maximal requirement for each rule

STI · INNSBRUCK

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

if x>=5 and x<=6 and y>=6 and y<=8 then class = positive



26

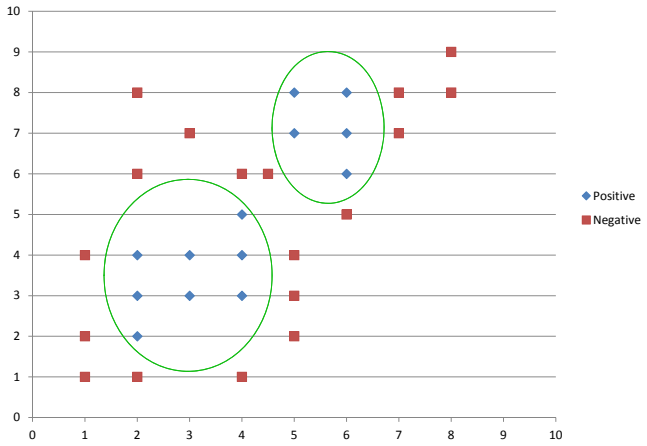
Maximal requirement for each rule

STI · INNSBRUCK

The full set of positive examples is covered by the following two rules:

if x>=2 and x<=4 and y>=2 and y<=5 then class = positive

if x>=5 and x<=6 and y>=6 and y<=8 then class = positive



27

Minimal requirement for the rule set

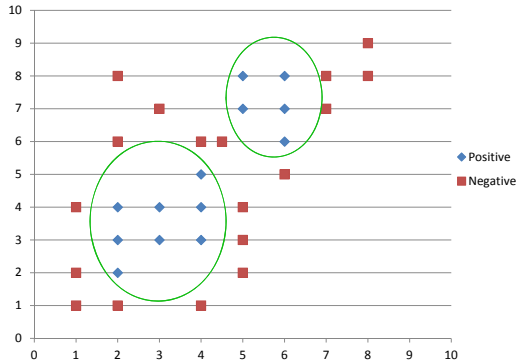
STI · INNSBRUCK

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>=2 and x<=4 and y>=2 and y<=5 then class = positive

if x>=5 and x<=6 and y>=6 and y<=8 then class = positive



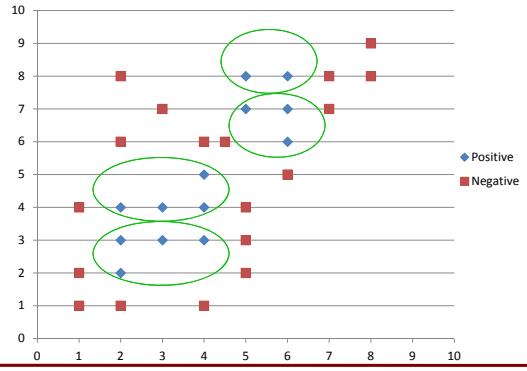
28

Minimal requirement for the rule set

STI · INNSBRUCK

Solution 2:

if x>=2 and x<=4 and y>=2 and y<=3 then class = positive
if x>=2 and x<=4 and y>3 and y<=5 then class = positive
if x>=5 and x<=6 and y>=6 and y<=7 then class = positive
if x>=5 and x<=6 and y>7 and y<=8 then class = positive



29

Minimal requirement for the rule set

STI · INNSBRUCK

- 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].

30

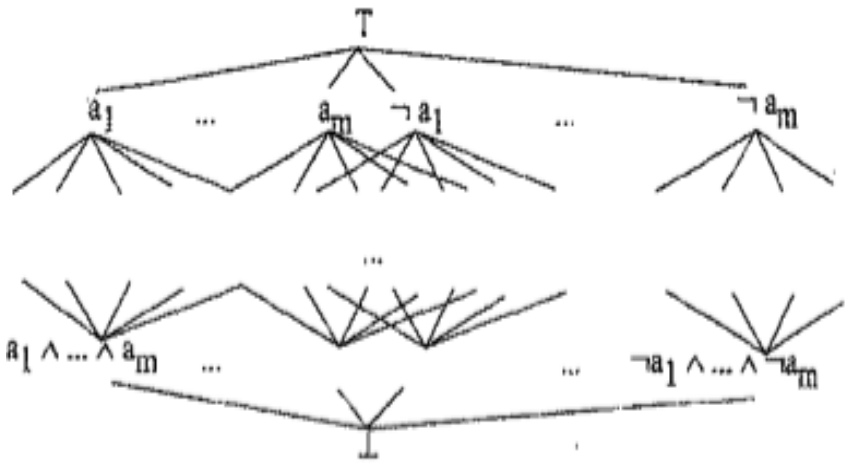
STI · INNSBRUCK

RULE LEARNING APPROACHES

31

Latice of possible rules

STI · INNSBRUCK



32

Rule learning approaches

STI · INNSBRUCK

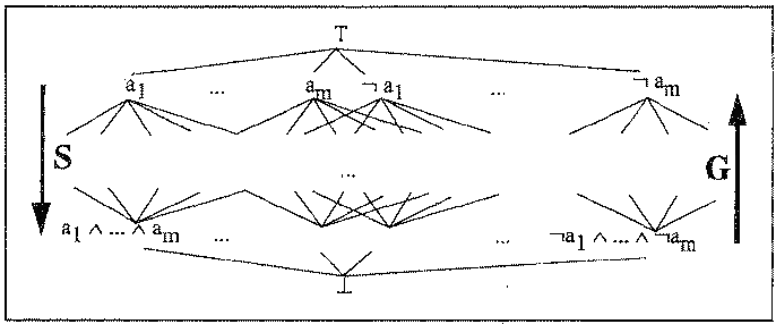
- Rule learning can be seen as a search problem in the latice of possible rules
- Depending on the starting point and on the direction of search we can distinguished three classes of rule learning approaches:
  1. **Specialization** – the search procedure starts at the top of the lattice and searches towards the bottom of the latice, 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 t**o**p 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.

33

Rule learning approaches

STI · INNSBRUCK

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



34

STI · INNSBRUCK

SPECIALIZATION

35

Specialization

STI · INNSBRUCK

- 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.

36

STI · INNSBRUCK

THE ID3 ALGORITHM

37

ID3

STI · INNSBRUCK

- 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.

38

The Basic ID3 Algorithm

STI · INNSBRUCK

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]

39

Selecting the Best Classifier

STI · INNSBRUCK

- 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.

40

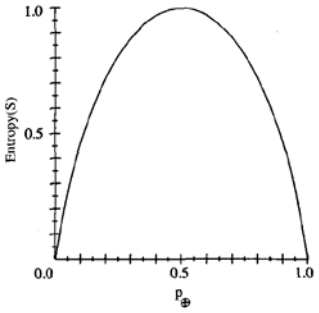
Entropy

STI · INNSBRUCK

- 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

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

- $\mathbf{p_+}/\mathbf{p_-}$ 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: $Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$



41

Information Gain

STI · INNSBRUCK

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

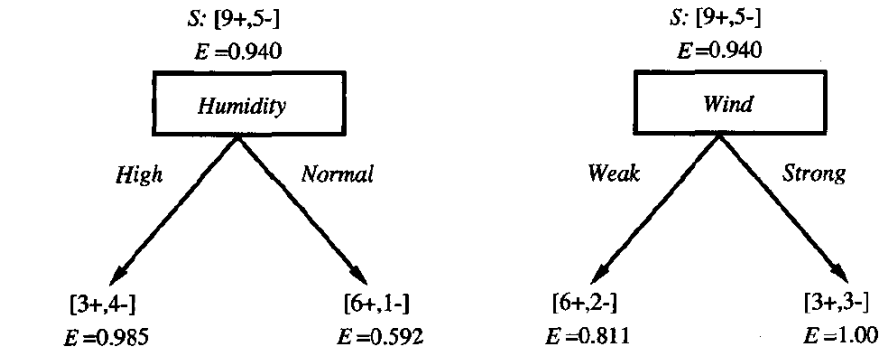
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- 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.

42

Example: Which attribute is the best classifier?

STI · INNSBRUCK



Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

Figure taken from [1]

Hypothesis Space Search in Decision Tree Learning

STI · INNSBRUCK

- 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.

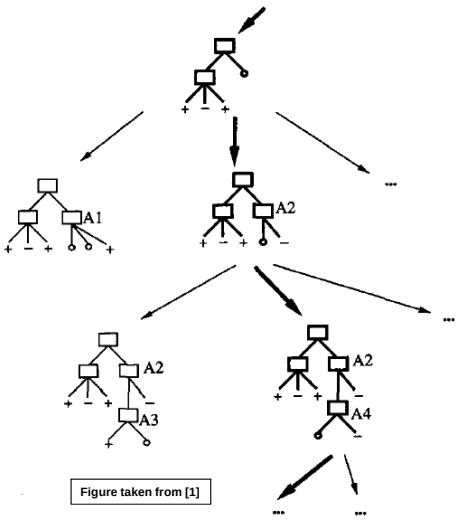


Figure taken from [1]

Capabilities and Limitations of ID3

STI · INNSBRUCK

- 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.

45

Inductive Bias in Decision Tree Learning

STI · INNSBRUCK

- 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.

46

ID3's Inductive Bias

STI · INNSBRUCK

- 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.

47

Occam's Razor

STI · INNSBRUCK

- …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.

48

ID3 by Example ([1])

STI · INNSBRUCK

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

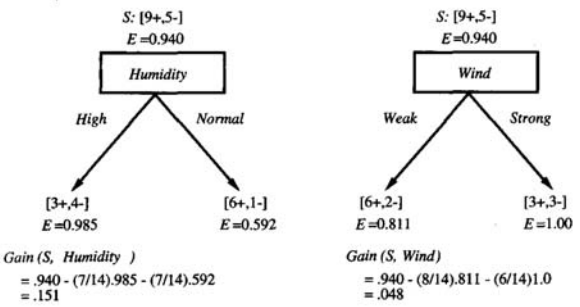
– Target attribute: PlayTennis (values: yes / no)

49

Example: Which attribute should be tested first in the tree?

STI · INNSBRUCK

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

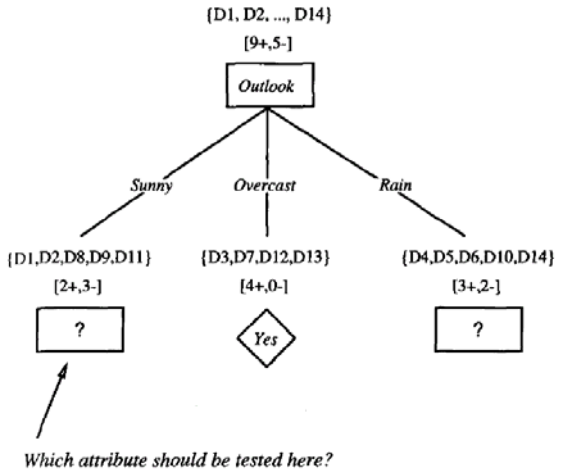


- **Gain(S, Outlook) = 0.246**; Gain(S, Humidity) = 0.151; Gain(S, Wind) = 0.048; Gain(S, Temperature)=0.029

50

Example: Resulting Tree

STI · INNSBRUCK



51

Example: Continue Selecting New Attributes

STI · INNSBRUCK

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

S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

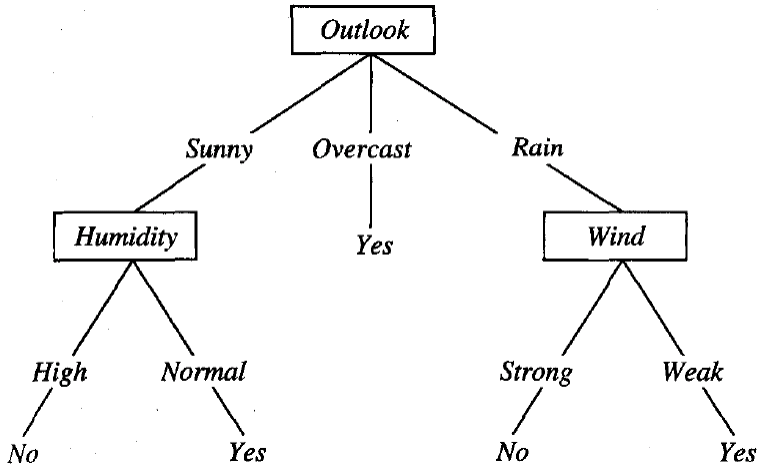
$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

- 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).

52

Example: Final Decision Tree

STI · INNSBRUCK



53

STI · INNSBRUCK

GENERALIZATION

54

Generalization

STI · INNSBRUCK

- 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.

55

STI · INNSBRUCK

THE RELAX ALGORITHM

56

RELAX

STI · INNSBRUCK

- 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.

57

RELAX: Ilustration by an example

STI · INNSBRUCK

- 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) $x \cap \neg y \cap z \rightarrow C$	5) $x \rightarrow C$
2) $\neg y \cap z \rightarrow C$	6) $\neg y \rightarrow C$
3) $x \cap z \rightarrow C$	7) $z \rightarrow C$
4) $x \cap \neg y \rightarrow C$	8) $\rightarrow C$

58

STI · INNSBRUCK

COMBINING SPECIALIZATION AND GENERALIZATION

59

STI · INNSBRUCK

THE JOJO ALGORITHM

60

JoJo

STI · INNSBRUCK

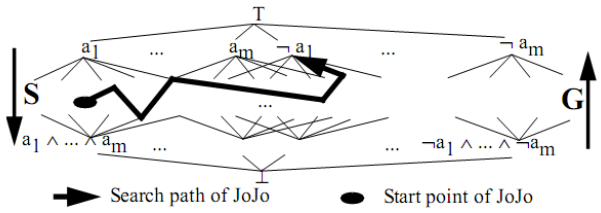
- In general, it cannot be determined which search direction is the better one.
- JoJo is an algorithm that combines both search directions in one heuristic search procedure.
- JoJo can start at an arbitrary point in the lattice of complexes and generalizes and specializes as long as the quality and correctness can be improved, i.e. until a local optimum can be found, or no more search resources are available (e.g., time, memory).

61

JoJo (1)

STI · INNSBRUCK

- 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).

62

JoJo (2)

STI · INNSBRUCK

- 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.

63

JoJo (4)

STI · INNSBRUCK

- Criteria for choosing a vertical position:
  1. Approximation of possible lenght 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.

64

JoJo Principle Components

STI · INNSBRUCK

- 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 criteria.
- 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.

65

STI · INNSBRUCK

REFINEMENT OF RULES WITH JOJO

66

Refinement of rules with JoJo

STI · INNSBRUCK

- Refinement of rules refers to the modification of a given rule set based on additonal 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.

67

Refinement of rules with JoJo (1)

STI · INNSBRUCK

- There is a four step procedure for the refinment 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.

68

Refinement of rules with JoJo (3)

STI · INNSBRUCK

- 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).

69

STI · INNSBRUCK

ILLUSTRATION BY A LARGER EXAMPLE: ID3

70

ID3 Example: The Simpsons

STI · INNSBRUCK

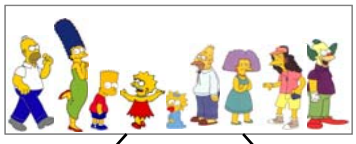
Person	Hair Length	Weight	Age	Class
Homer	0”	250	36	M
Marge	10”	150	34	F
Bart	2”	90	10	M
Lisa	6”	78	8	F
Maggie	4”	20	1	F
Abe	1”	170	70	M
Selma	8”	160	41	F
Otto	10”	180	38	M
Krusty	6”	200	45	M
Comic	8”	290	38	?

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

71

ID3 Example: The Simpsons

STI · INNSBRUCK



yes — Hair Length <= 5? — no

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

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

Let us try splitting on *Hair length*

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

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

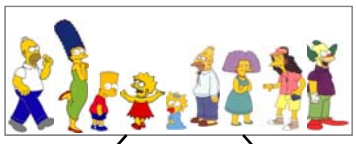
$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Hair Length <= 5) = **0.9911** – (4/9 * **0.8113** + 5/9 * **0.9710**) = **0.0911**

72

ID3 Example: The Simpsons

STI · INNSBRUCK



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

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

yes — Weight <= 160? — no

Let us try splitting on *Weight*

Entropy(0**F**,4**M**) = -(0/4)$\log_2$(0/4) - (4/4)$\log_2$(4/4)
= **0**

Entropy(4**F**,1**M**) = -(4/5)$\log_2$(4/5) - (1/5)$\log_2$(1/5)
= **0.7219**

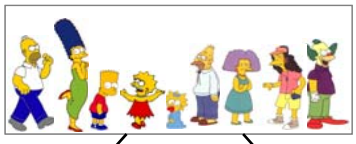
$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Weight <= 160) = **0.9911** – (5/9 * **0.7219** + 4/9 * **0**) = **0.5900**

73

ID3 Example: The Simpsons

STI · INNSBRUCK



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

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

yes — age <= 40? — no

Let us try splitting on *Age*

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

Entropy(3**F**,3**M**) = -(3/6)$\log_2$(3/6) - (3/6)$\log_2$(3/6)
= **1**

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

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

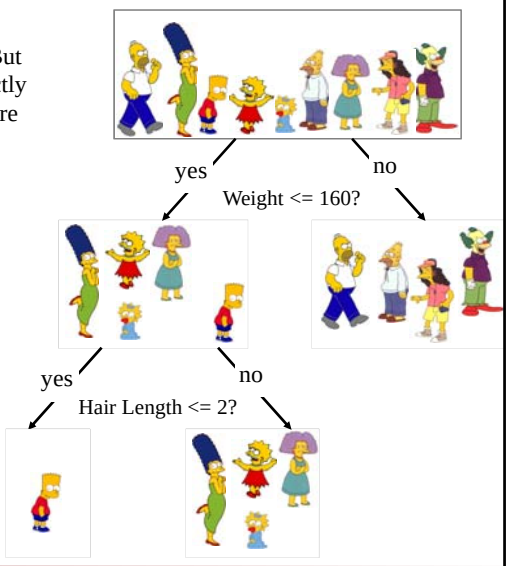
74

ID3 Example: The Simpsons

STI · INNSBRUCK

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!



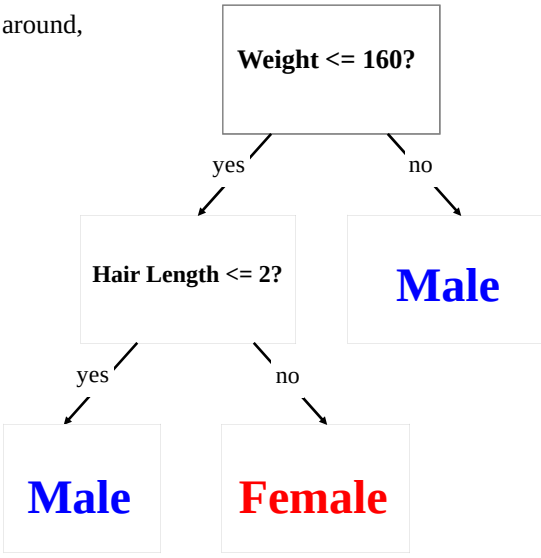
75

ID3 Example: The Simpsons

STI · INNSBRUCK

We need don't need to keep the data around, just the test conditions.

How would these people be classified?

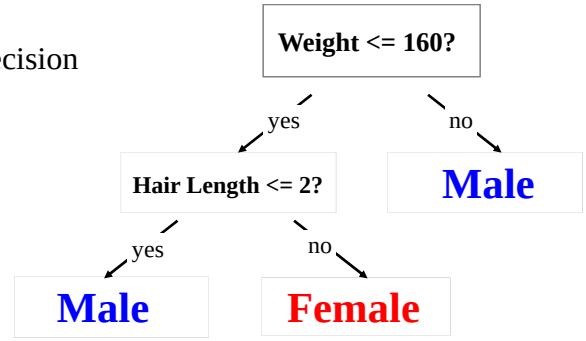


76

ID3 Example: The Simpsons

STI · INNSBRUCK

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**

77

STI · INNSBRUCK

EXTENSIONS

78

Issues in Decision Tree Learning

STI · INNSBRUCK

- 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]).

79

STI · INNSBRUCK

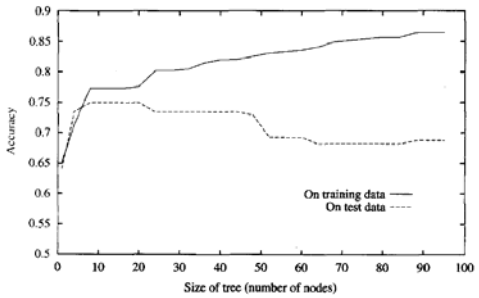
THE C4.5 ALGORITHM

80

Avoiding Overfitting the Data

STI · INNSBRUCK

- **Definition:** Given a hypothesis space H, a hypothesis h ϵ H is said to **overfit** the training data if there exists some alternative hypothesis h' ϵ 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]

81

Example: Overfitting due to Random Noise

STI · INNSBRUCK

- Original decision tree based on correct data:

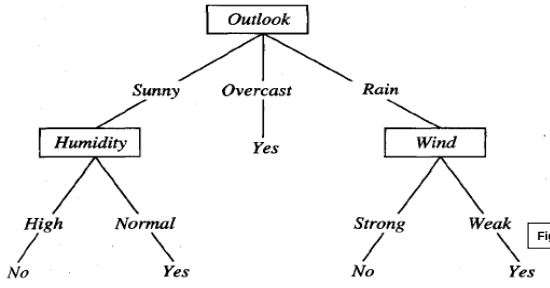


Figure taken from [1]

- 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.

82

Strategies to Avoid Overfitting

STI · INNSBRUCK

- 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 Quinlain: "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)

83

Rule Post-Pruning

STI · INNSBRUCK

- 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.

84

Incorporating Continuous-Valued Attributes

STI · INNSBRUCK

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

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- 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?

85

Computing a Threshold

STI · INNSBRUCK

- 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.

86

Alternative Measures for Selecting Attributes

STI · INNSBRUCK

- 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]).

87

Gain Ratio

STI · INNSBRUCK

- 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:

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

88

Handling Training Examples with Missing Attribute Values

STI · INNSBRUCK

- 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).

89

STI · INNSBRUCK

SUMMARY

90

Summary

STI · INNSBRUCK

- 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.

91

Summary (2)

STI · INNSBRUCK

- 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.

92

STI · INNSBRUCK

REFERENCES

References

STI · INNSBRUCK

- Mandatory reading:
  - [1] Mitchell, T. "Machine Learning", McGraw Hill, 1997. (Section 3)
  - [2] Mitchell, T. "The Discipline of Machine Learning" Technical Report CMU-ML-06-108, July 2006. Available online: http://www.cs.cmu.edu/~tom/pubs/MachineLearning.pdf
  - [3] Fensel, D. “Ein integriertes System zum maschinellen Lernen aus Beispielen” Künstliche Intelligenz (3). 1993.
- Further reading:
  - [4] Feigenbaum, E. A. “Expert systems in the 1980s” In: A. Bond (Ed.), State of the art report on machine intelligence. Maidenhea: Pergamon-infotech.
  - [5] Quinlan, J. R. “Induction of decision trees”. Machine Learning 1 (1), pp. 81-106, 1986.

References

- [6] Quinlan, J. R. “C4.5: Programs for Machine Learning” Morgan Kaufmann, 1993.
- [7] Fayyad, U. M. “On the induction of decision trees for multiple concept learning” PhD dissertation, University of Michigan, 1991.
- [8] Agrawal, Imielinsky and Swami: Mining Association Rules between Sets of Items in Large Databases. ACM SIGMOD Conference, 1993, pp. 207-216.
- [9] Quinlan: Generating Production Rules From Decision Trees. 10th Int’l Joint Conference on Artificial Intelligence, 1987, pp. 304-307.
- [10] Fensel and Wiese: Refinement of Rule Sets with JoJo. European Conference on Machine Learning, 1993, pp. 378-383.

References

- [11] AQ: Michalski, Mozetic, Hong and Lavrac: The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. AAAI-86, pp. 1041-1045.
- [12] C4: Quinlan: Learning Logical Definitions from Relations: Machine Learning 5(3), 1990, pp. 239-266.
- [13] CN2: Clark and Boswell: Rule Induction with CN2: Some recent Improvement. EWSL-91, pp. 151-163.
- [14] CABRO: Huyen and Bao: A method for generating rules from examples and its application. CSNDAL-91, pp. 493-504.
- [15] FOIL: Quinlan: Learning Logical Definitions from Relations: Machine Learning 5(3), 1990, pp. 239-266.
- [16] PRISM: Cendrowska: PRISM: An algorithm for inducing modular rules. Journal Man-Machine Studies 27, 1987, pp. 349-370.
- [17] RELAX: Fensel and Klein: A new approach to rule induction and pruning. ICTAI-91.

References

– [18] http://www.ensta.fr/~diam/ro/online/viggo_wwwcompendium/node146.html.
– [19] Goodrich, M.T., Mirelli, V. Orletsky, M., and Salowe, J. Decision tree conctruction in fixed dimensions: Being global is hard but local greed is good. Technical Report TR-95-1, Johns Hopkins University, Department of Computer Science, Baltimore, MD 21218, May 1995

97

References

- Wikipedia links:
  - Machine Learning, http://en.wikipedia.org/wiki/Machine_learning
  - Rule learning, http://en.wikipedia.org/wiki/Rule_learning
  - Association rule learning, http://en.wikipedia.org/wiki/Association_rule_learning
  - ID3 algorithm, http://en.wikipedia.org/wiki/ID3_algorithm
  - C4.5 algorithm, http://en.wikipedia.org/wiki/C4.5_algorithm

98

Next Lecture

STI · INNSBRUCK

#	Title
0.1	Propositional Logic
0.2	Predicate Logic
1	Introduction
2	Reasoning
3	Search Methods
4	CommonKADS
5	Problem-Solving Methods
6	Planning
7	Software Agents
8	Rule Learning
9	**Inductive Logic Programming**
10	Neural Networks
11	Semantic Web and Services
12	Exam

99

Questions?

STI · INNSBRUCK

Email: anna.fensel@uibk.ac.at
Web: http://bit.ly/2mwtf1h
Twitter: @alphaverda
Facebook: https://www.facebook.com/STIInnsbruck/

100