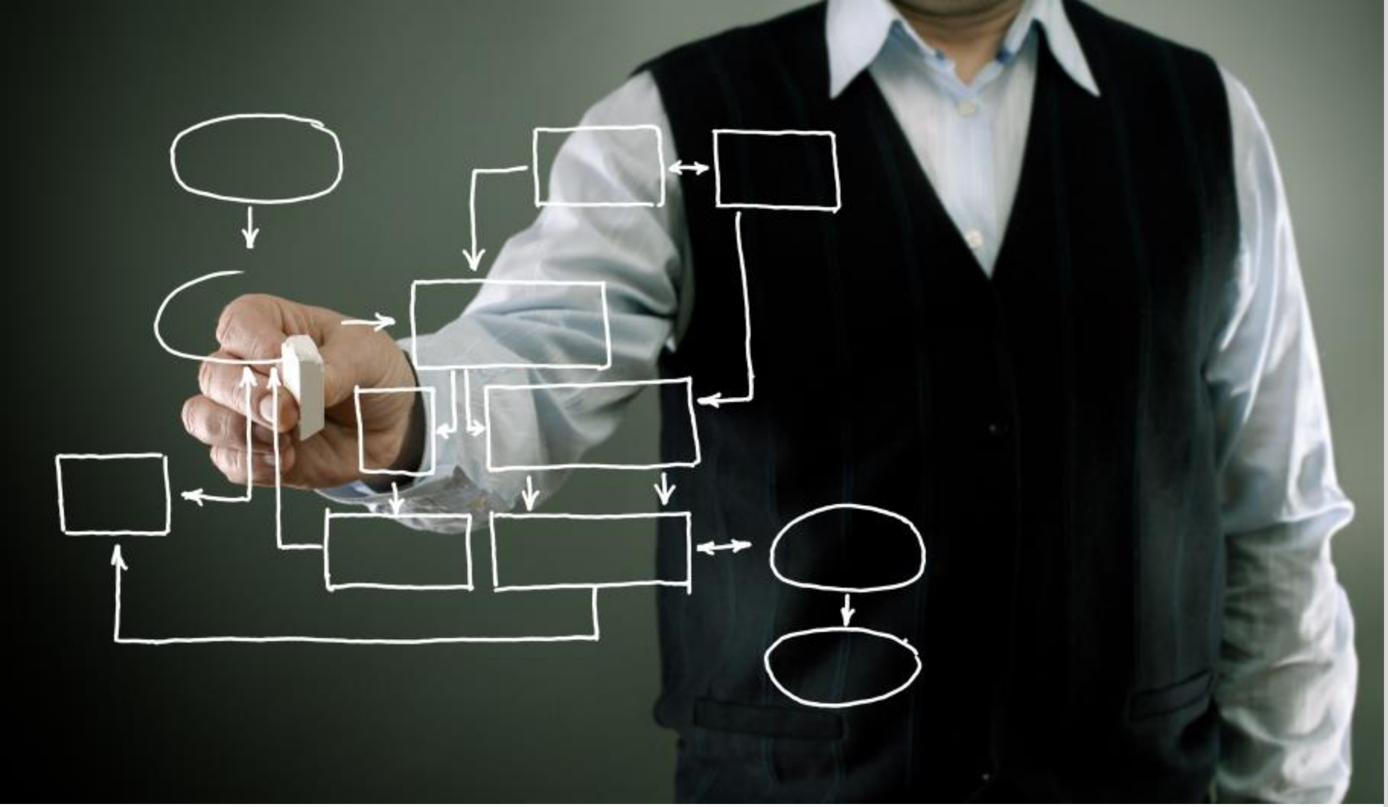


Decision trees

Contact:
Dr. Michael Affenzeller
FH OOE - School of Informatics, Communications and Media
Heuristic and Evolutionary Algorithms Lab (HEAL)
Softwarepark 11, A-4232 Hagenberg

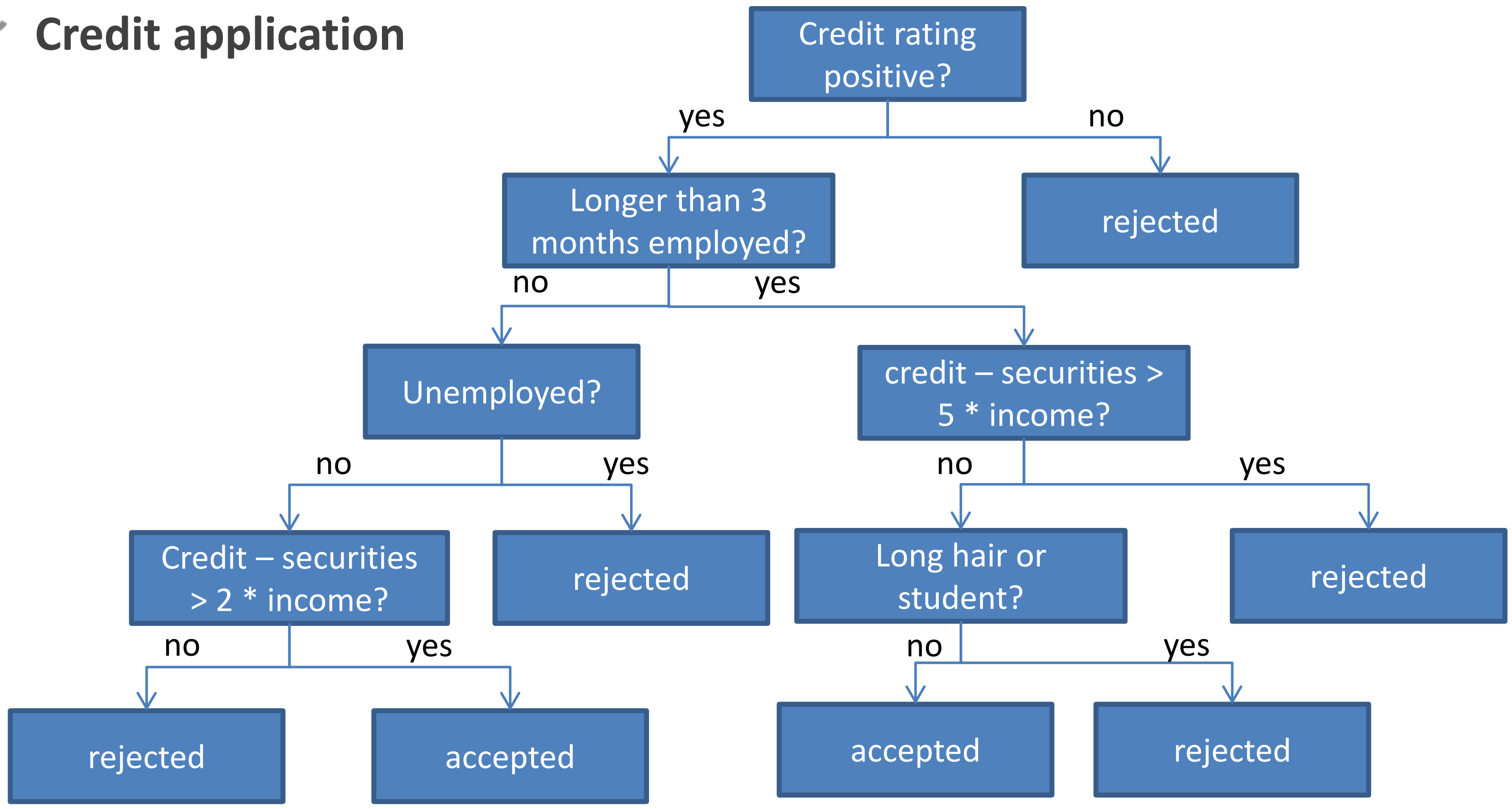
e-mail:
michael.affenzeller@fh-hagenberg.at
Web:
http://heal.heuristiclab.com
http://heureka.heuristiclab.com

HEAL
HEURISTIC AND EVOLUTIONARY ALGORITHMS LABORATORY

Decision trees

- Can be learnt from examples
- Easy to interpret
- Easy learning algorithm
- Can be transformed to rules

Credit application



Basic terms

- Population, instance space (e.g. set of all possible clients)
- Attribute a_i with discrete value range $\{v_{i,1},...v_{i,k}\}$. Instances are described with attributes (e.g. income)
- Instance: An instance $\{(Attribute_i, Value_i), ...\}$ is a configuration of the attributes with values
- Target attribute (e.g. credit rating). The target has to be predicted
- Classification: Ein Entscheidungsbaum bildet eine Instanz (Attributwerte) auf einen vorhergesagten Wert des Zielattributes ab
- Terminal node : Contains value of target attribute
- Test node (Attribute, $(Value_1,Tree_1)$,...., $(Value_n,Tree_n)$): Contains an attribute and a decision tree for every element of the value range of that attribute
- A decision tree is either a terminal node or a test node

Example:

- (rejected)
- (credit rating positive, (yes, (accepted)), (no, (rejected)))
- (credit rating positive, (yes, (Long hair or student, (yes,(rejected)),(no,(accepted)))), (no, (rejected)))

Application of decision trees

- Input: decision tree + instance
- Output: Value of target attribute
- Algorithm:
  - If node is terminal node, then return value of target attribute
  - If node test node, then test the value of the attribute with the instance and call the algorithm for the suitable sub tree recursively

Application of decision trees

- Application ((Value),Instance) = Value
- Application $((a,((v_1,Tree_1),\ldots\ldots,(v_k,Tree_k),Instance)=$
  - Application ($Tree_i$, Instance)
  - whereby $(a,v_i) \in$ Instance
- The algorithm always returns a value for the target attribue, if a value is defined for all attribues of an instance.

Decision Tree Learning

- Input: Table with instances

Attribute 1	Attribute 2	…	Attribute n	Target Attribute
Value 1, 1	Value 1, 2		Value 1, n	Value 1, z
Value 2, 1	Value 2, 2		Value 2, n	Value 2, z
Value m, 1	Value m, 2		Value m, n	Value m, z

- Output: decision tree, which returns for every instance the concrete value of the target attribute
- The tree should have as few nodes as possible

Complete search to find the smallest tree

- How many decision trees exist?
- Complexity of complete search to find the smallest tree
- Approaches……

Simple Example

- Input: Set of observations:

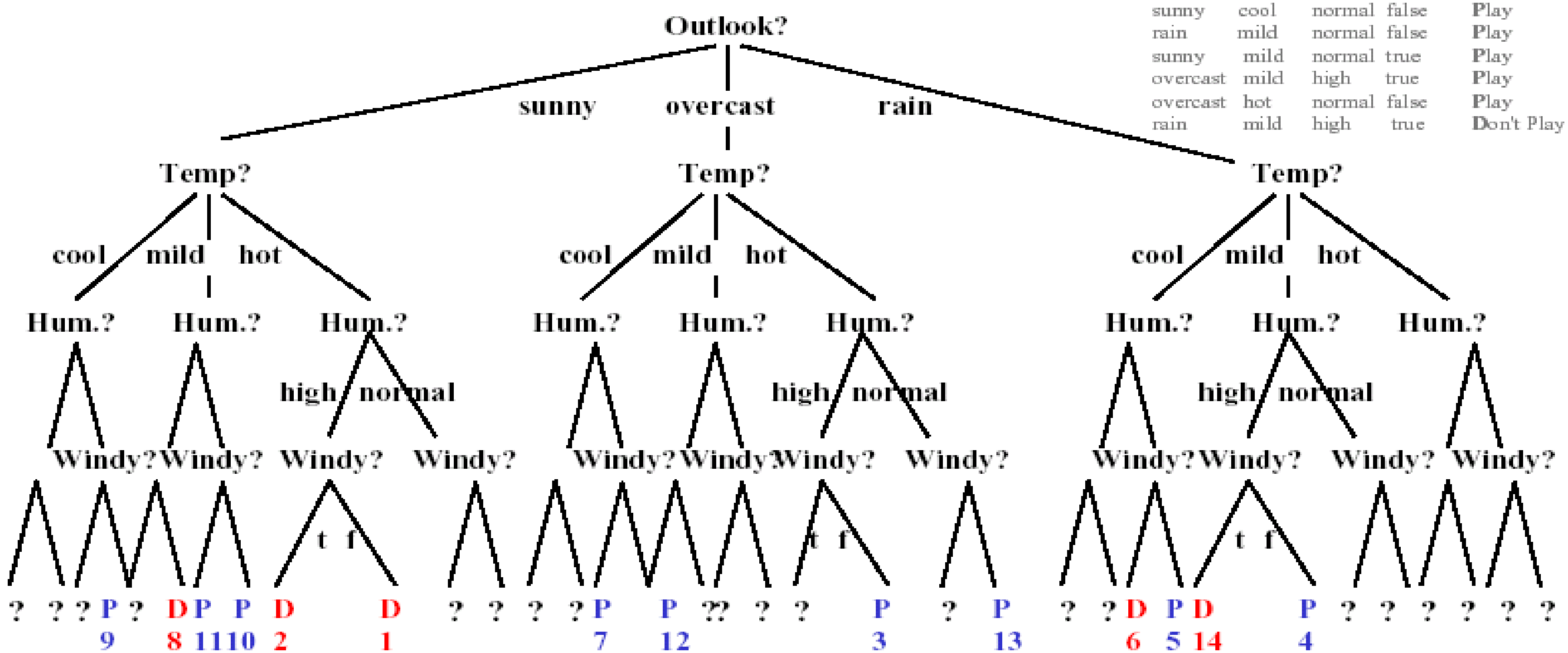
Day	Outlook	Temp.	Humidity	Windy	CLASS
day1	sunny	hot	high	false	Don't Play
day2	sunny	hot	high	true	Don't Play
day3	overcast	hot	high	false	Play
day4	rain	mild	high	false	Play
day5	rain	cool	normal	false	Play
day6	rain	cool	normal	true	Don't Play
day7	overcast	cool	normal	true	Play
day8	sunny	mild	high	false	Don't Play
day9	sunny	cool	normal	false	Play
day10	rain	mild	normal	false	Play
day11	sunny	mild	normal	true	Play
day12	overcast	mild	high	true	Play
day13	overcast	hot	normal	false	Play
day14	rain	mild	high	true	Don't Play

- Output: Decision Tree, which predicts, if it is reasonable to play golf

How to learn a decision tree?

Naive idea 1: use all attributes to construct a complete tree; assign class to those leaves that correspond to an example

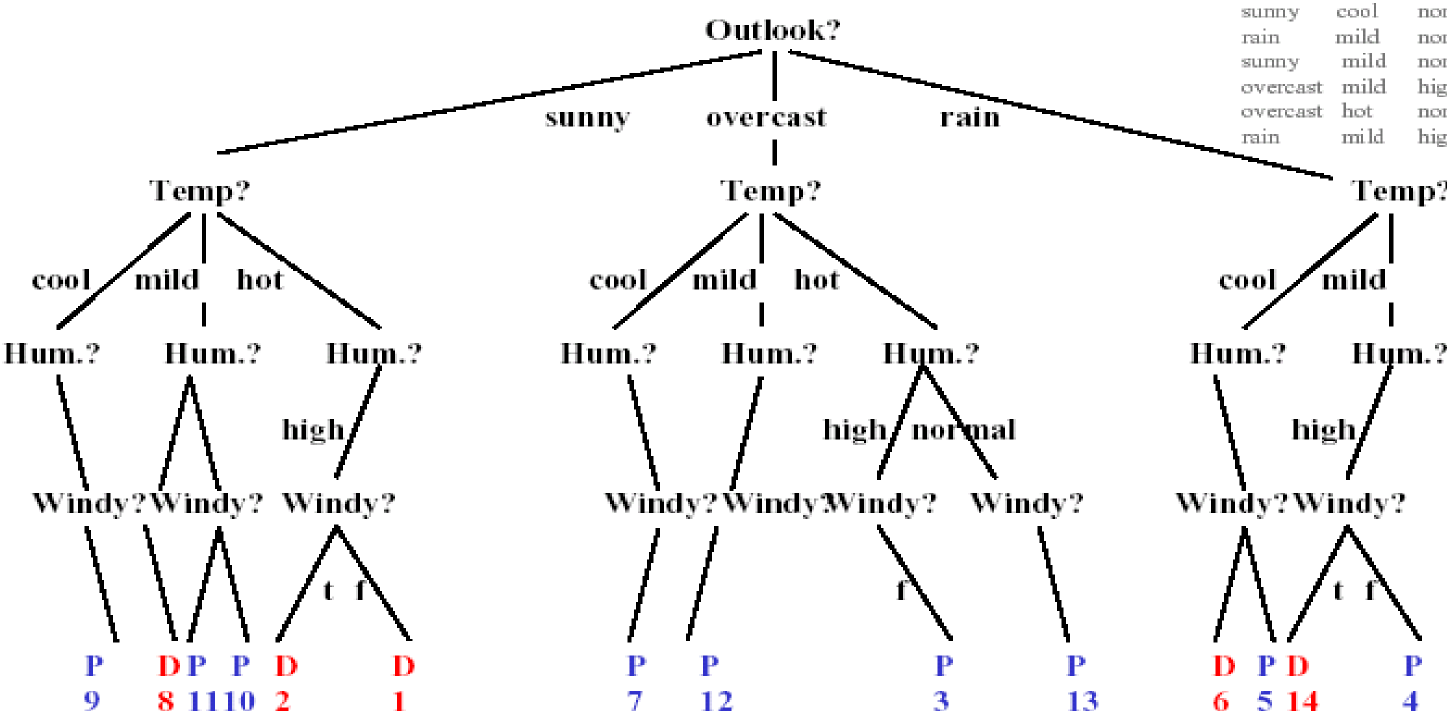
Outlook	Temp.	Hum.	Windy	CLASS
sunny	hot	high	false	Don't Play
sunny	hot	high	true	Don't Play
overcast	hot	high	false	Play
rain	mild	high	false	Play
rain	cool	normal	false	Play
rain	cool	normal	true	Don't Play
overcast	cool	normal	true	Play
sunny	mild	high	false	Don't Play
sunny	cool	normal	false	Play
rain	mild	normal	false	Play
sunny	mild	normal	true	Play
overcast	mild	high	true	Play
overcast	hot	normal	false	Play
rain	mild	high	true	Don't Play



How to learn a decision tree?

Naive idea 2: keep only those branches for which class is known

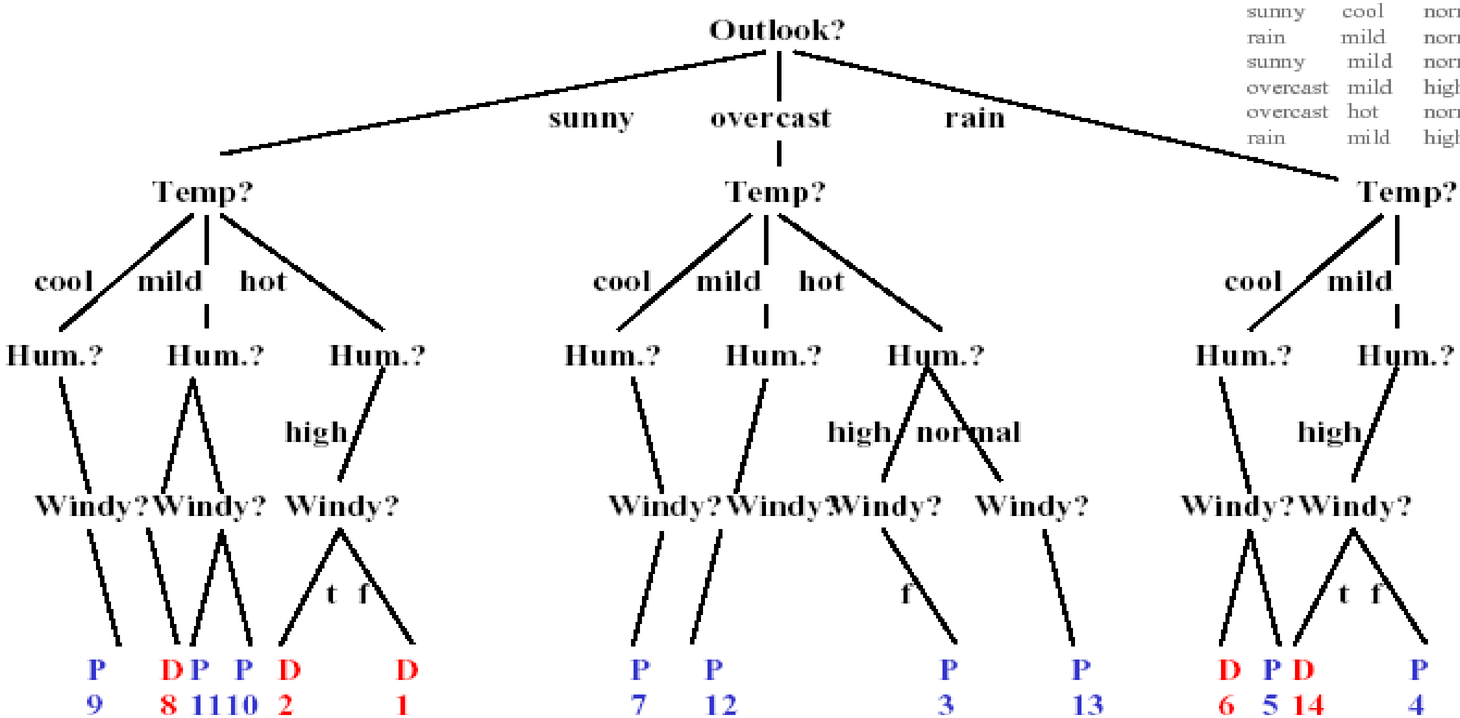
Outlook	Temp.	Hum.	Windy	CLASS
sunny	hot	high	false	Don't Play
sunny	hot	high	true	Don't Play
overcast	hot	high	false	Play
rain	mild	high	false	Play
rain	cool	normal	false	Play
rain	cool	normal	true	Don't Play
overcast	cool	normal	true	Play
sunny	mild	high	false	Don't Play
sunny	cool	normal	false	Play
rain	mild	normal	false	Play
sunny	mild	normal	true	Play
overcast	mild	high	true	Play
overcast	hot	normal	false	Play
rain	mild	high	true	Don't Play



How to learn a decision tree?

Naive idea 3: remove nodes with only one branch, and subtrees with the same class in all branches
(no distinction necessary to be consistent with training data)

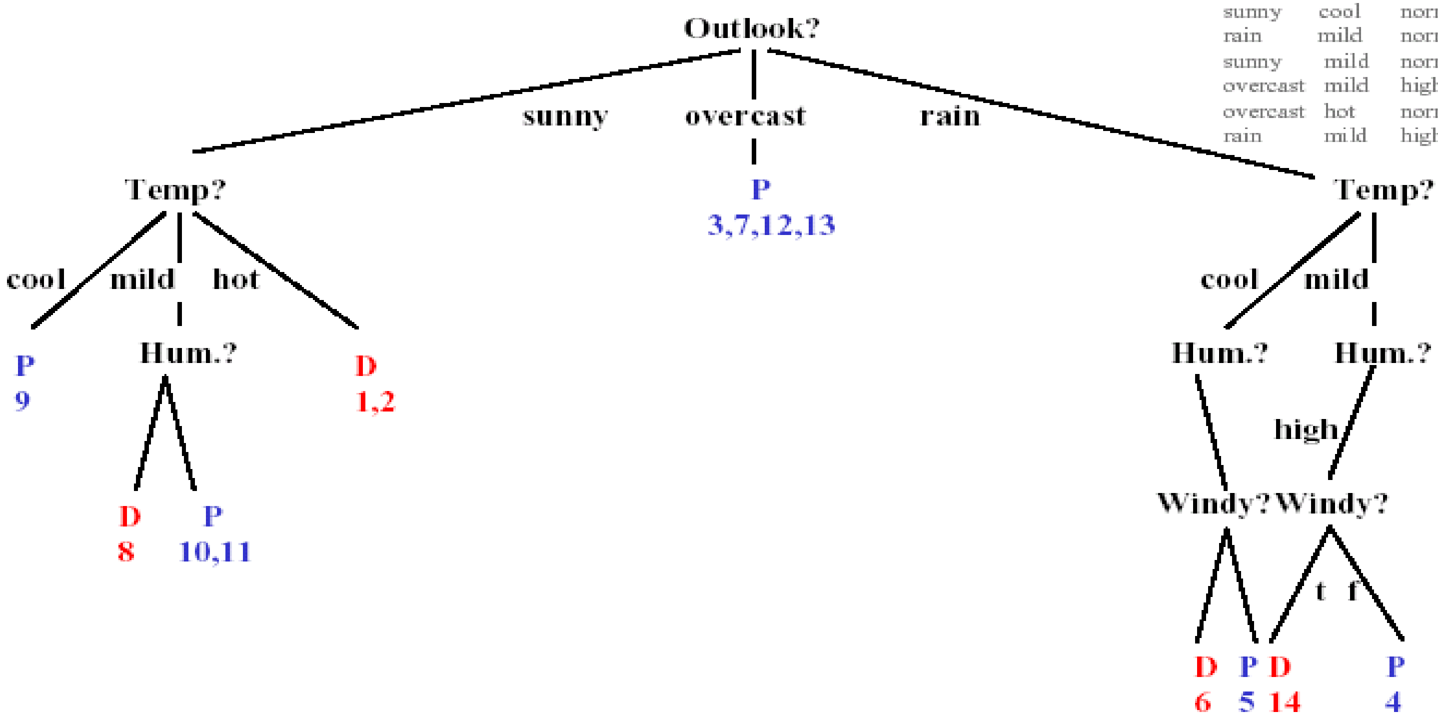
Outlook	Temp.	Hum.	Windy	CLASS
sunny	hot	high	false	Don't Play
sunny	hot	high	true	Don't Play
overcast	hot	high	false	Play
rain	mild	high	false	Play
rain	cool	normal	false	Play
rain	cool	normal	true	Don't Play
overcast	cool	normal	true	Play
sunny	mild	high	false	Don't Play
sunny	cool	normal	false	Play
rain	mild	normal	false	Play
sunny	mild	normal	true	Play
overcast	mild	high	true	Play
overcast	hot	normal	false	Play
rain	mild	high	true	Don't Play



How to learn a decision tree?

Naive idea 3: remove nodes with only one branch, and subtrees with the same class in all branches
(no distinction necessary to be consistent with training data)

Outlook	Temp.	Hum.	Windy	CLASS
sunny	hot	high	false	Don't Play
sunny	hot	high	true	Don't Play
overcast	hot	high	false	Play
rain	mild	high	false	Play
rain	cool	normal	false	Play
rain	cool	normal	true	Don't Play
overcast	cool	normal	true	Play
sunny	mild	high	false	Don't Play
sunny	cool	normal	false	Play
rain	mild	normal	false	Play
sunny	mild	normal	true	Play
overcast	mild	high	true	Play
overcast	hot	normal	false	Play
rain	mild	high	true	Don't Play



Problem of “naive“ approaches:

- Attributes are used in the order of their occurrence in the data
- The same attributes are always used in the same layer of a tree
- A lot trees can be constructed, which are consistent with the trainings data…
  - Which one is the ‘best‘?
  - Need of general formal criteria to describe the target decision tree

Requirements:

- Consistent with trainings data
  – Should predict the desired class for every known trainings example
- Generalization
  – Should be more general than the sum of the input examples, to also predict new cases
- Correctness
  – Correct prediction of new cases
- Simplicity (Ozzam‘s Razor)
  – Search for the simplest (smallest) tree, which is consistent with the trainings data
  – Exhaustive search is not feasible (combinatorial complexity)
  – Heuristic search strategies (e.g. ID3)

ID3 Algorithm:

- Heuristic Approach
  - No guarantee to find the simplest tree (only a exhaustive approach could guarantee that)
- Recursive Approach
  - Stepwise generation of the decision tree – start with empty tree
- Greedy Approach
  - 'Greedy' selection of the next attribute, i.e. the local optimization criteria will be maximized

ID3 Algorithm

- Stepwise recursive construction heuristic
- Start with root and all input training examples

Main loop:

1. If all examples in current node belong to the same class C
   => make current node a leaf (with class label C) and EXIT loop
2. Select attribute A that is “best” for current node, and assign A as decision attribute to current node
3. Create a branch + successor node for each possible value of A
4. Split training examples associated with current node into subsets according to their value of A; assign each subset to its respective branch/subnode
5. For each subnode: recursively call ID3

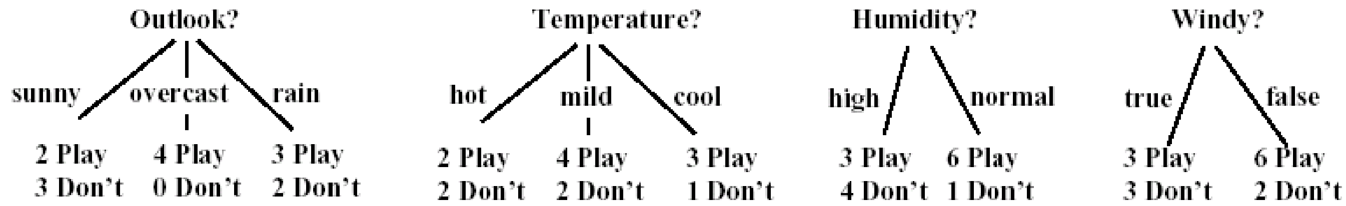
Decision trees
Decision Tree Learning

HeuristicLab

OBERÖSTERREICH

Selection of the 'best' attribute

Intuitive: The best attribute is the one, which differentiates best between the classes and therefore increases the probability to construct simple trees



Which attribute provides the best discriminability and therefore the most information gain?

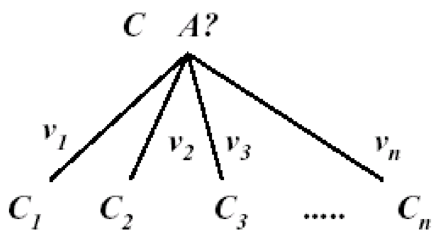
Outlook	Temp.	Hum.	Windy	CLASS
sunny	hot	high	false	Don't Play
sunny	hot	high	true	Don't Play
overcast	hot	high	false	Play
rain	mild	high	false	Play
rain	cool	normal	false	Play
rain	cool	normal	true	Don't Play
overcast	cool	normal	true	Play
sunny	mild	high	false	Don't Play
sunny	cool	normal	false	Play
rain	mild	normal	false	Play
sunny	mild	normal	true	Play
overcast	mild	high	true	Play
overcast	hot	normal	false	Play
rain	mild	high	true	Don't Play

Introduction of new measurements:

Entropy, Information gain

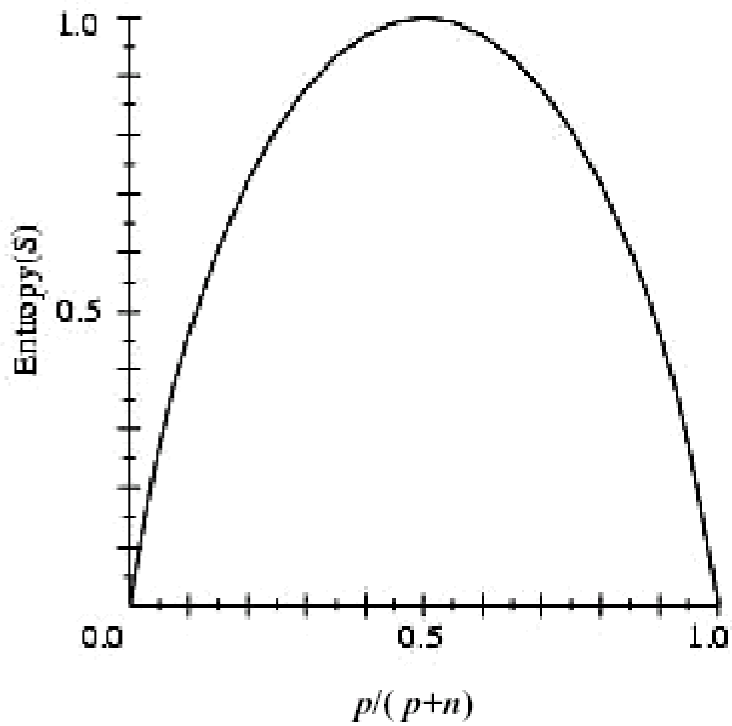
Entropy:

- Notation (on the supposition that we have 2 classes)
  - A Attributes with (positive) Values $v_1, \ldots\ldots v_n$
  - C Set of training examples of the current node
  - N Number of examples in C (N=|C|)
  - p,n Number of positive/negative examples in C (p+n=N)
  - p_i, n_i Number of positive/negative examples in sub node C_i



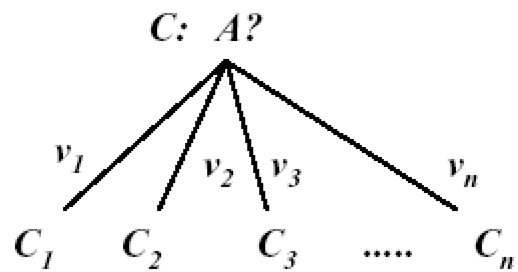
$$Entropy(C) = -p/(p+n)\ log_2\ p/(p+n) - n/(p+n)\ log_2\ n/(p+n)$$

Entropy:



Decision trees
Decision Tree Learning

Information gain:

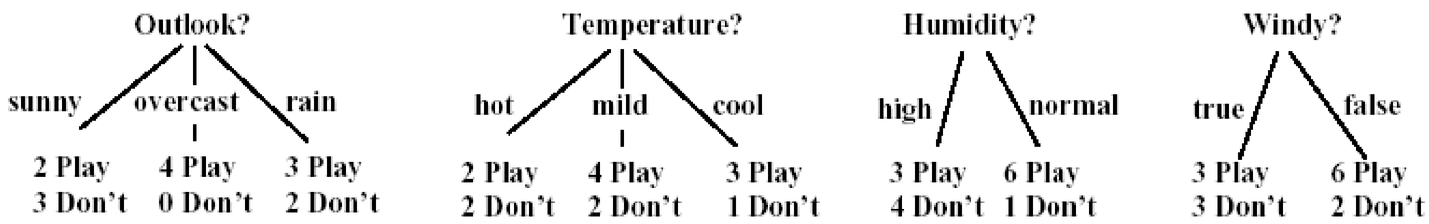


$$InfoGain(C,A) = Entropy(C) - \sum |C_i|/|C| * Entropy(C_i)$$

- Information gain is the expected reduction of the Entropy, if the data is classified according to A
- ID3 selects the attribute with the biggest information gain
- Maximizing the information gain is equivalent to minimizing the second term

Decision trees
Decision Tree Learning

Example:



Class distribution at the root (i.e., in the full training set): **9 Play, 5 Don't Play**

=> $Entropy(C) = -\,9/14 \cdot log_2(9/14) - 5/14 \cdot log_2(5/14) = 0.940$ [bits]

max.

$Gain(C,Outlook) = .940 - 5/14 \cdot .971 - 4/14 \cdot 0.00 - 5/14 \cdot .971 = 0.246$ [bits]

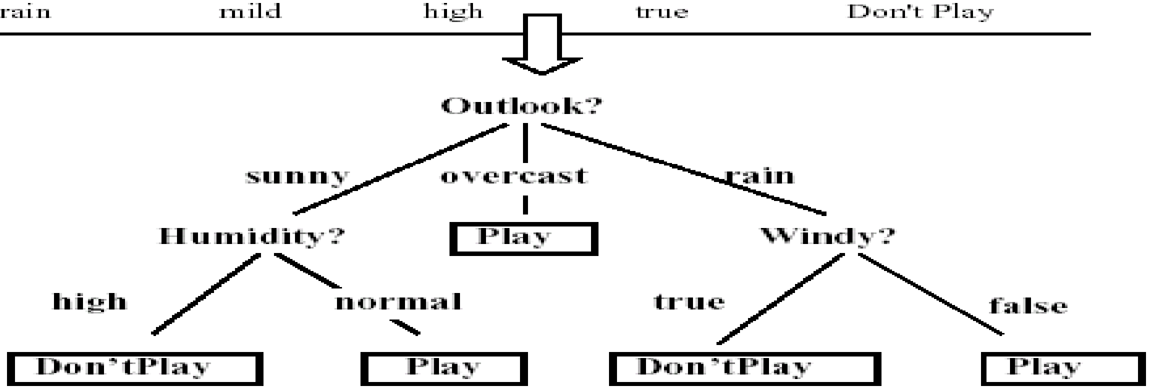
$Gain(C,Temperature) = .940 - 4/14 \cdot 1.00 - 6/14 \cdot .918 - 4/14 \cdot .811 = 0.029$ [bits]

$Gain(C,Humidity) = .940 - 7/14 \cdot .985 - 7/14 \cdot .592 = 0.151$ [bits]

$Gain(C,Windy) = .940 - 6/14 \cdot 1.00 - 8/14 \cdot .811 = 0.048$ [bits]

ID3 Algorithm:

Outlook	Temp.	Humidity	Windy	CLASS
sunny	hot	high	false	Don't Play
sunny	hot	high	true	Don't Play
overcast	hot	high	false	Play
rain	mild	high	false	Play
rain	cool	normal	false	Play
rain	cool	normal	true	Don't Play
overcast	cool	normal	true	Play
sunny	mild	high	false	Don't Play
sunny	cool	normal	false	Play
rain	mild	normal	false	Play
sunny	mild	normal	true	Play
overcast	mild	high	true	Play
overcast	hot	normal	false	Play
rain	mild	high	true	Don't Play



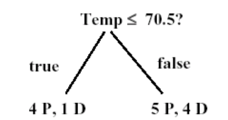
Decision trees
Decision Tree Learning

Extension of Decision Tree Learning:

- Numerical attributes:
  - There shouldn't be an additional path for every instance of a numeric attribute
  - Solution: choose a suitable partition
- Missing attribute entries:
  - Ignore examples with missing attribute entries (wasteful)
  - Consider missing entries as additional class value (often leads to undesirable results)
  - Use default-values (most frequent entry….)
  - ……………

Outlook	Temp.	Humidity	Windy	CLASS
sunny	85	high	false	Don't Play
sunny	80	high	true	Don't Play
overcast	83	high	false	Play
rain	70	high	false	Play
rain	68	normal	false	Play
rain	65	normal	true	Don't Play
overcast	64	normal	true	Play
sunny	72	high	false	Don't Play
sunny	69	normal	false	Play
rain	75	normal	false	Play
sunny	75	normal	true	Play
overcast	72	high	true	Play
overcast	81	normal	false	Play
rain	71	high	true	Don't Play

Temperature:	64	65	68	69	70	71	72	75	80	81	83	85
Class: Play	1	0	1	1	1	0	1	2	0	1	1	0
Don't Play	0	1	0	0	0	1	1	0	1	0	0	1



Outlook	Temp.	Humidity	Windy	CLASS
sunny	85	high	false	Don't Play
sunny	?	high	true	Don't Play
overcast	83	high	false	Play
rain	70	high	false	Play
rain	68	?	false	Play
rain	65	normal	true	Don't Play
overcast	64	normal	true	Play
sunny	72	?	false	Don't Play
sunny	69	normal	false	Play
rain	75	normal	?	Play
sunny	75	normal	true	Play
overcast	72	high	true	Play
?	?	normal	false	Play
rain	71	high	true	Don't Play

Undesirable properties

- Comprehensibility
- Simplicity
- Accuracy
  - concerning the training data
  - concerning new data!

Measurement

- Complexity
- Quality of prediction
  - Accuracy of the learned model with unknown data
  - Use partition of data for learning and the rest for testing
  - systematically with cross validation

Complexity of search tree und Overfitting

- **Decision tree, which represents the training data best (doesn’t have to be the best model!)**
  - Training examples are perhaps incorrect (noisy)
  - Maybe the target can’t be exactly represented in the given representation

  Overfitting
    - The constructed model can contain irrelevant properties
    - Model is more complex than necessary (converges to noisy parts)
    - Prediction of Model is insufficient
  - Solution: Learn simplified model, which doesn’t exactly converge to the training data (Pruning)

Avoidance of Overfitting: Pruning Concept

- Pre Pruning:
  - Abort of further segmentation of nodes (even if it contains instances of different classes) if no statistical relevant correlation between attributes and classes can be detected.
- Post Pruning:
  - Construction of a maximal consistent search tree with the training data
  - Pruning of sub trees, which have no significant influence

$\Rightarrow$ Simpler search trees, which approximate the training data worse and new data better