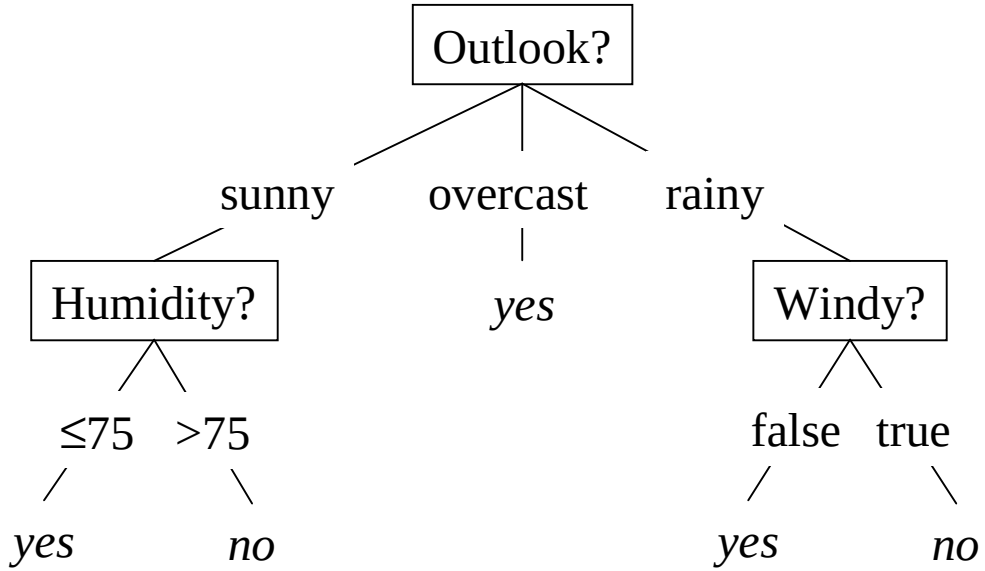
Decision Tree Learning et al.



Univ.-Lektor Dr.techn. Alexander K. Seewald
Österreichisches Forschungsinstitut
für Artificial Intelligence

alex@seewald.at / alex.seewald.at

Prelude: 1R, A Simple Rule Learner

ZeroR (*0R*) predicts most common class, i.e. the best prediction that can be obtained without referring to any attribute at all. Equivalent to nearest-neighbor with k=N.

OneR (*1R*) predicts best rule based on a single attribute, i.e. the best prediction that can be obtained using just one attribute.

[Holte 1993, *Machine Learning Journal (11)*]

How does 1R work?

- Generate one rule for each attribute. Choose the rule which maximizes proportion of correct prediction on training data TD (aka *Accuracy*).
- For qualitative variables, choose intervals of maximum length so that in each interval a majority class for more than SMALL values exists.
- For quantitative variable j, predict (for each value k separately) the most common class among all examples for this attribute and value.
- Treat missing values as one additional value for each attribute.

- *High bias, low variance. Works surprisingly well on some real-life datasets.*

alex@seewald.at / alex.seewald.at

Pseudocode for 1R (taken from Holte,1993)

1. In the training set count the number of examples in class C having value V for attribute A: store this information in a 3D array,COUNT[C,V,A].
2. The default class is the one having the most examples in the training set. The accuracy of the default class is the number of training examples in the default class divided by the total number of training examples.
3. FOR EACH NUMERICAL ATTRIBUTE, A, create a nominal version of A by defining a finite number of intervals of values. These intervals become the "values" of the nominal version of A. For example, if A's numerical values are partitioned into three intervals, the nominal version of A will have three values "interval 1", "interval 2", and "interval 3". COUNT[C,V,A] reflects this transformation: COUNT[C,"interval I",A] is the sum of COUNT[C,V,A] for all V in interval I. Definitions:

- Class C is optimal for attribute A, value V, if it maximizes COUNT[C,V,A].
- Class C is optimal for attribute A, interval I, if it maximizes COUNT[C,"interval I",A].

Values are partitioned into intervals so that every interval satisfies the following constraints:

(a) there is at least one class that is "optimal" for more than SMALL of the values in the interval. This constraint does not apply to the rightmost interval.

(b) If V[I] is the smallest value for attribute A in the training set that is larger than the values in interval I then there is no class C that is optimal both for V[I] and for interval I.

4. FOR EACH ATTRIBUTE, A, (use the nominal version of numerical attributes):

(a) Construct a hypothesis involving attribute A by selecting, for each value V of A (and also for "missing"), an optimal class for V. If several classes are optimal for a value, choose among them randomly

(b) add the constructed hypothesis to a set called HYPOTHESES. This set will ultimately contain one hypothesis for each attribute.

5. 1R: choose the rule from the set HYPOTHESES having the highest accuracy on the training set (if there are several "best" rules, choose among them at random).

alex@seewald.at / alex.seewald.at

1R Training Example

1R generates one rule per attribute (*SMALL*=5 here)

out. {overcast ⇒ *yes* (4/0), rainy ⇒ *yes* (3/2), sunny⇒*no*(2/3) } **71%**

windy { false ⇒ *yes* (6/2), true ⇒ *yes*(3/3) } 64%

hum. { <82.5 ⇒ *yes* (6/1), ≥82.5 ⇒ *no* (3/4) } **71%**

- Sort by humidity. Begin at top, noting for each value which class is optimal (may be many, e.g. for 90: *yes* and *no*). Count how often each class is optimal; continue until at least one class is optimal more than SMALL times = cond. (a).
- Apply cond. (b) to see if interval can be extended. If not, set interval end halfway between value and next value (=(80+85)/2 here)
- Continue for additional intervals, or until end has been reached.
- Missing values (if present) are put into a separate pseudo-interval with assigned optimal class.

temp {<77.5⇒ *yes* (7/3), ≥77.5⇒ *yes* (2/2) } 64%

Outlook	T	H	Windy	Play?
overcast	64°F	65%	true	*yes*
rainy	65°F	70%	true	*no*
sunny	69°F	70%	false	*yes*
sunny	75°F	70%	true	*yes*
overcast	81°F	75%	false	*yes*
rainy	68°F	80%	false	*yes*
rainy	75°F	80%	false	*yes*
sunny	85°F	85%	false	*no*
overcast	83°F	86%	false	*yes*
sunny	80°F	90%	true	*no*
overcast	72°F	90%	true	*yes*
rainy	71°F	91%	true	*no*
sunny	72°F	95%	false	*no*
rainy	70°F	96%	false	*yes*

For comparison:
Baseline (0R): 64%

alex@seewald.at / alex.seewald.at

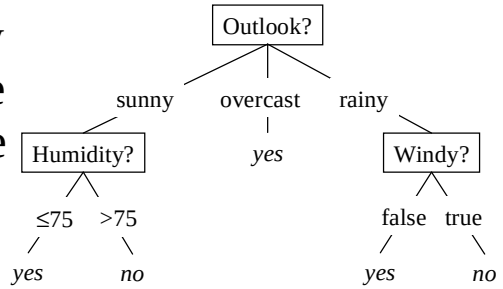
How to improve on 1R?

How to reduce 1R's bias / increase its complexity?

- Combine 1R rules of several attributes (~ 1R*, Holte 1993)
  - \+ slightly more expressive – very limited improvement
- Allow rules based on more than one attribute
  - \+ much more expressive
  - – curse of dimensionality: 2^p if we consider all possible attribute subsets(!)
  - – worse estimation of optimal class: for some combinations of attributes, very few or even no examples may exist.

- **Divide-and-Conquer:** Apply algorithm recursively! Choose best attribute at top and then recursively create rules for each subset instead of just counting the most common class. Repeat until "pure" (=only examples of the same class). This creates a *decision tree* of attribute and class values.

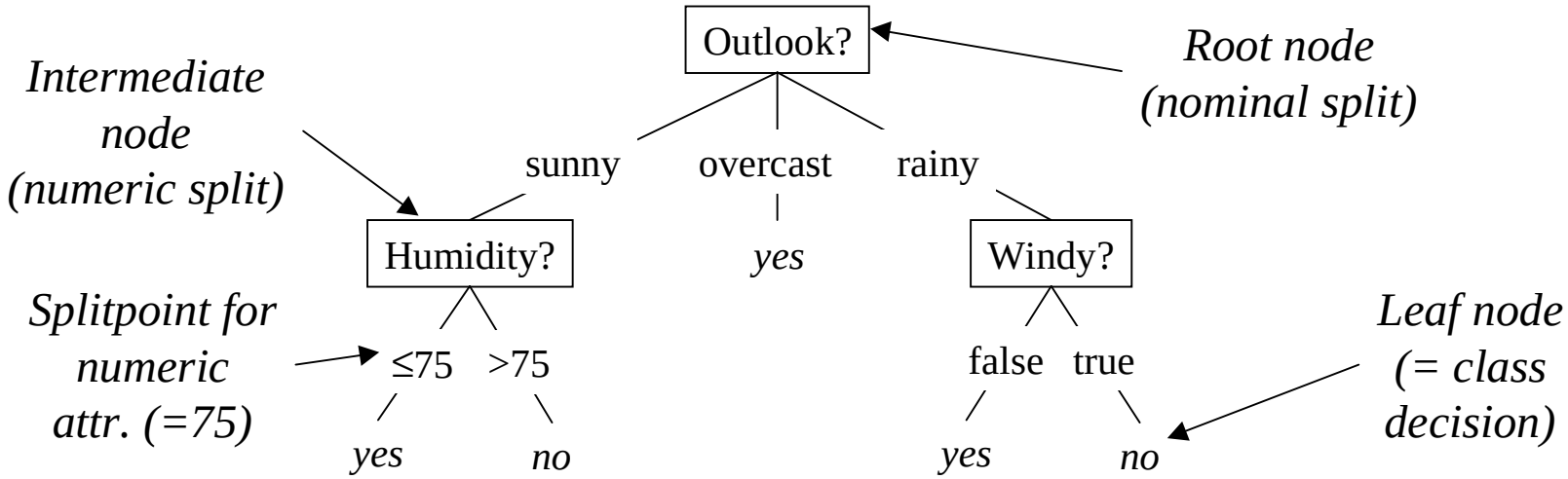
  Works slightly differently. Numeric attributes are only *split* (i.e. propagated into different subtrees) at a single value. Attributes are chosen by information gain, since accuracy is not available until the whole tree has been constructed.



alex@seewald.at / alex.seewald.at

What is a Decision Tree?

A recursive structure of attribute/class value decisions...



...which is equivalent to a set of rules, one for each path from the root node:

outlook=sunny & humidity≤75 ⇒ *yes*

outlook=sunny & humidity>75 ⇒ *no*

outlook=overcast ⇒ *yes*

outlook=rainy & windy=false ⇒ *yes*

outlook=rainy & windy=true ⇒ *no*

alex@seewald.at / alex.seewald.at

Basic Observations

Some basic observations on decision trees in general

- If we have once split on a nominal (qualitative) attribute, another split on the same attribute is meaningless - all examples within each subtree already have the same value for this attribute. Multiple splits on numeric (quantitative) attributes *are* possible, with different splitpoints, but at most s-1 for s unique values – $\log_2(s)$ if we always split at the median value of each set.

- The number of examples in each subtree will be smaller than the number at each node, provided we follow 1. and the attribute is not constant over all examples. In the latter case, splitting is also meaningless.

1. and 2. show that this process will stop at (possibly multiple) examples with exactly the same attribute values - ***regardless of how we split!***

alex@seewald.at / alex.seewald.at

Basic Observations (2)

3. Provided training data is consistent (i.e. no two examples have exactly the same values for all attributes and different classes), and we do not stop before each leaf contains examples with the same values for all attributes, each tree stores the full training data (disregarding example order), again **regardless of how we split**. There are exponentially many trees which are fully consistent with training data (i.e., 100% accuracy)

Problem: Which tree to choose among those consistent with training data?

Common Heuristic: *Ockhams Razor*

"Non sunt multiplicanda entia praeter necessitatem."

(Entities should not be multiplied beyond necessity.)

William of Ockham (1290? - 1349?)

Translation: Prefer the smallest/simplest theory among all consistent ones.

However, it is generally not feasible to exhaustively search for the smallest tree.

alex@seewald.at / alex.seewald.at

Top-Down Induction of Decision Trees

Prominent members: ID3, C4.5, CART, ...

Recursive algorithms

- Creates decision tree step-by-step
- Begins with an empty tree

Heuristic algorithms

- Aims at constructing a small tree, but cannot guarantee that it will find the smallest tree – since that would mean constructing *all* possible trees.

Greedy algorithms

- At each step, makes decision (which attribute/splitpoint to choose) based on a local optimality criterion (information gain)
- Blind to attributes that are relevant only in combination

alex@seewald.at / alex.seewald.at

ID3+

- *Bias:* Low. Smaller trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred to those that do not.
- *Variance*: High. A single example may change the tree completely.

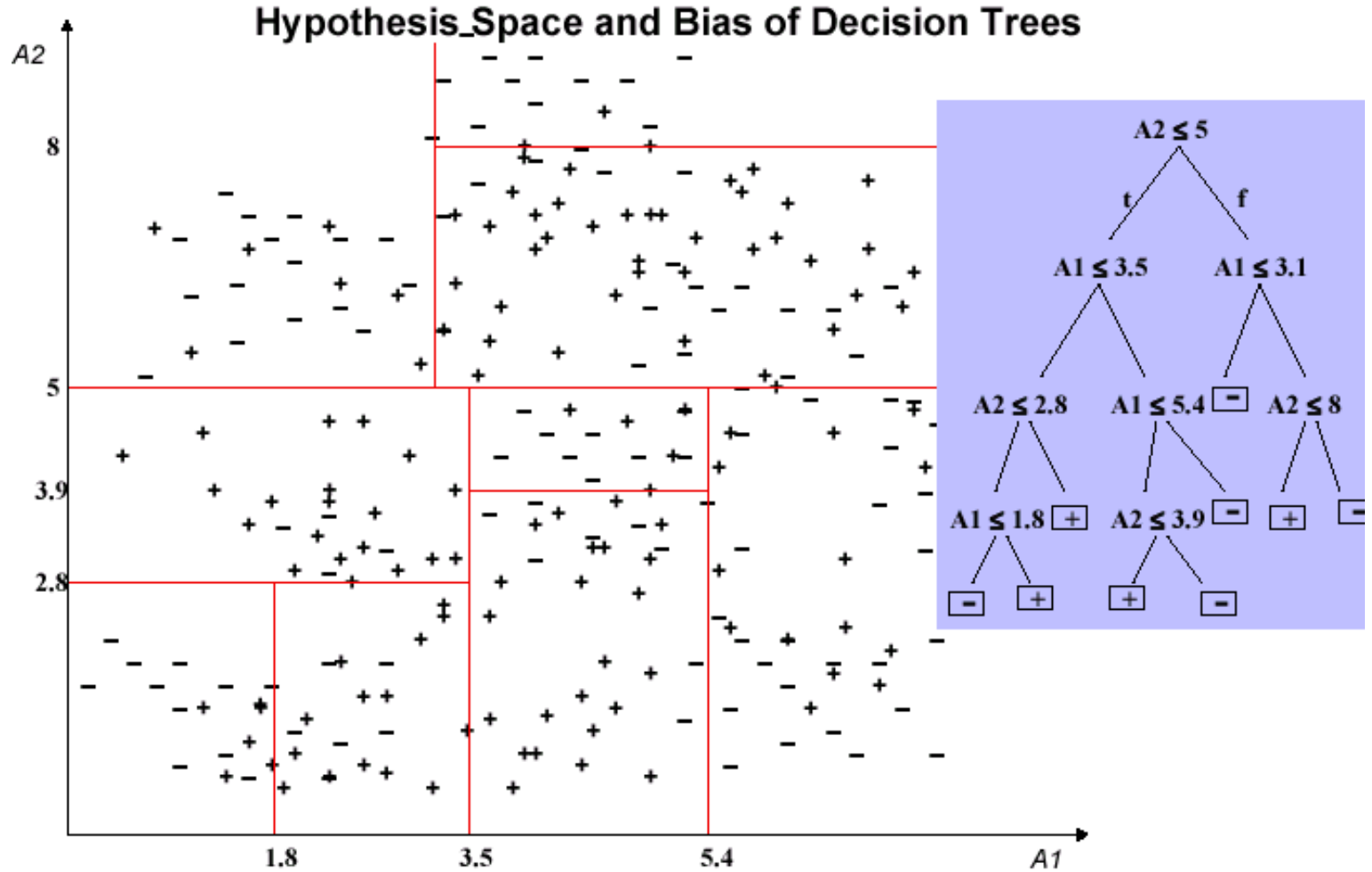
Pseudocode for ID3+ (i.e. ID3 extended with numeric attribute splits)

Start with root node and given all training examples

```
If all examples in current node belong to same class =>
   make current node a leaf node and EXIT
Select best nominal attribute, or best attribute /
   splitpoint combination for numeric attribute.
Create branch + subnodes for all values for nominal
   best attribute, or for < and >=splitpoint if best
   attribute is numeric.
Split training examples according to values of best
   attribute into subsets for each subnode.
Call ID3+ recursively for each subnode node with the
   appropriate subset of training examples.
```


alex@seewald.at / alex.seewald.at

Decision Tree - Bias & Variance



Low bias, high variance. Numeric attributes are split binary via splitpoint. Concept boundaries are axis-parallel hyperrectangles (see above)

alex@seewald.at / alex.seewald.at

What is the best local split?

Intuition: The attribute which best discriminates between the classes, and thus is likely to create a small tree.

⇒ **Information gain:** Expected increase in information (=reduce in entropy) if data are split by the values of the attribute (Information Theory by Shannon)

Notation (assumes two classes)

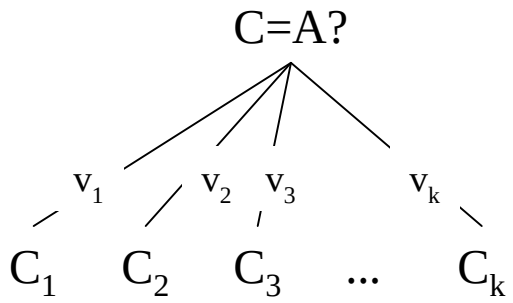
A ... some attribute with possible values v_1, ..., v_k

C ... set of training examples associated with current node

N ... number of examples in C (N = |C|)

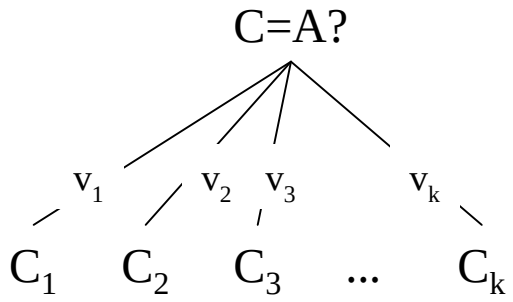
p, n ... number of positive / negative examples in C ($p+n = N$)

p_i, n_i ... number of positive / negative examples in subnode C_i



alex@seewald.at / alex.seewald.at

Entropy & InfoGain



Entropy(C) = – p/(p+n) log_2 p/(p+n) – n/(p+n) log_2 n/(p+n)

Entropy is a measure of the "impurity" of set C with respect to the class labels

InfoGain(C,A) = Entropy(C) – $\Sigma|C_i|/|C|$ * Entropy(C_i)

InfoGain is the expected reduction in entropy if the data is split along values of attributes A. *ID3 selects attribute with highest InfoGain in each step.*

Note: Entropy(C) is independent of A → maximizing InfoGain(C,A) is equivalent to minimizing the second term, i.e the weighted sum of entropies.

alex@seewald.at / alex.seewald.at

Splitting on numeric attributes

For numeric attributes, splitting on all possible values leads to weak generalization

→ Binary split on a single value (=threshold, splitpoint). This is done by considering all (reasonable) split points and computing InfoGain for each of them. Among all information gain values from nominal attributes, and all splitpoints from numeric attributes, the maximum is chosen by ID3.

Example: Split on humidity from the weather dataset. E(C) = 0.940 [bits]

67.5 (1/0 vs. 8/5) → InfoGain = 0.048 [bits]

72.5 (3/1 vs. 6/4) → InfoGain = 0.015 [bits]

82.5 (6/1 vs. 3/4) → **InfoGain= 0.152 [bits]** (best splitpoint for humidity)

85.5 (6/2 vs. 3/3) → InfoGain = 0.048 [bits]

88.0 (7/2 vs. 2/3) → InfoGain = 0.102 [bits]

90.5 (8/3 vs. 1/2) → InfoGain = 0.079 [bits]

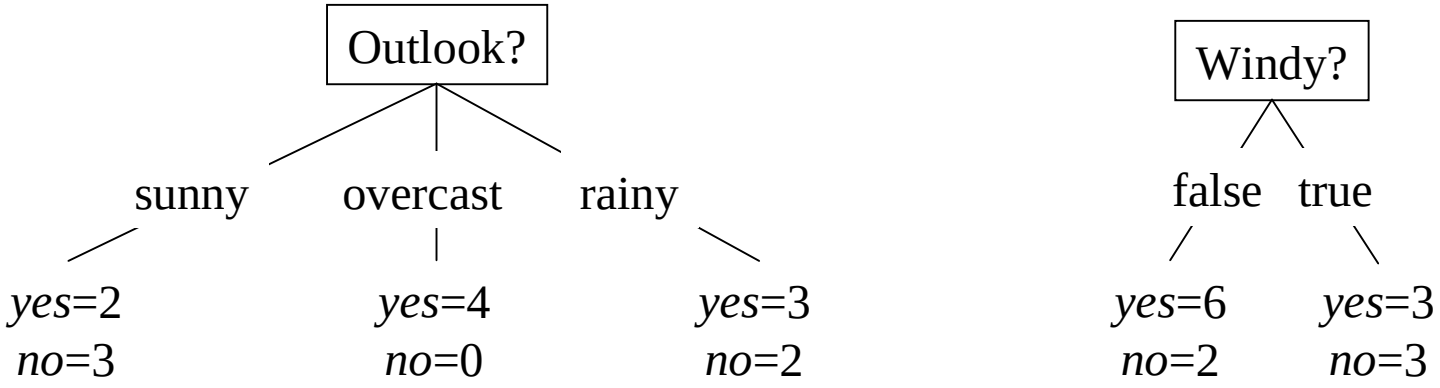
95.5 (8/5 vs. 1/0) → InfoGain = 0.048 [bits]

Hum. =	65	70	75	80	85	86	90	91	95	96
Play=yes	1	2	1	2	0	1	1	0	0	1
Play=no	0	1	0	0	1	0	1	1	1	0

alex@seewald.at / alex.seewald.at

Example: Choose root node for weather

Choose attribute with highest InfoGain



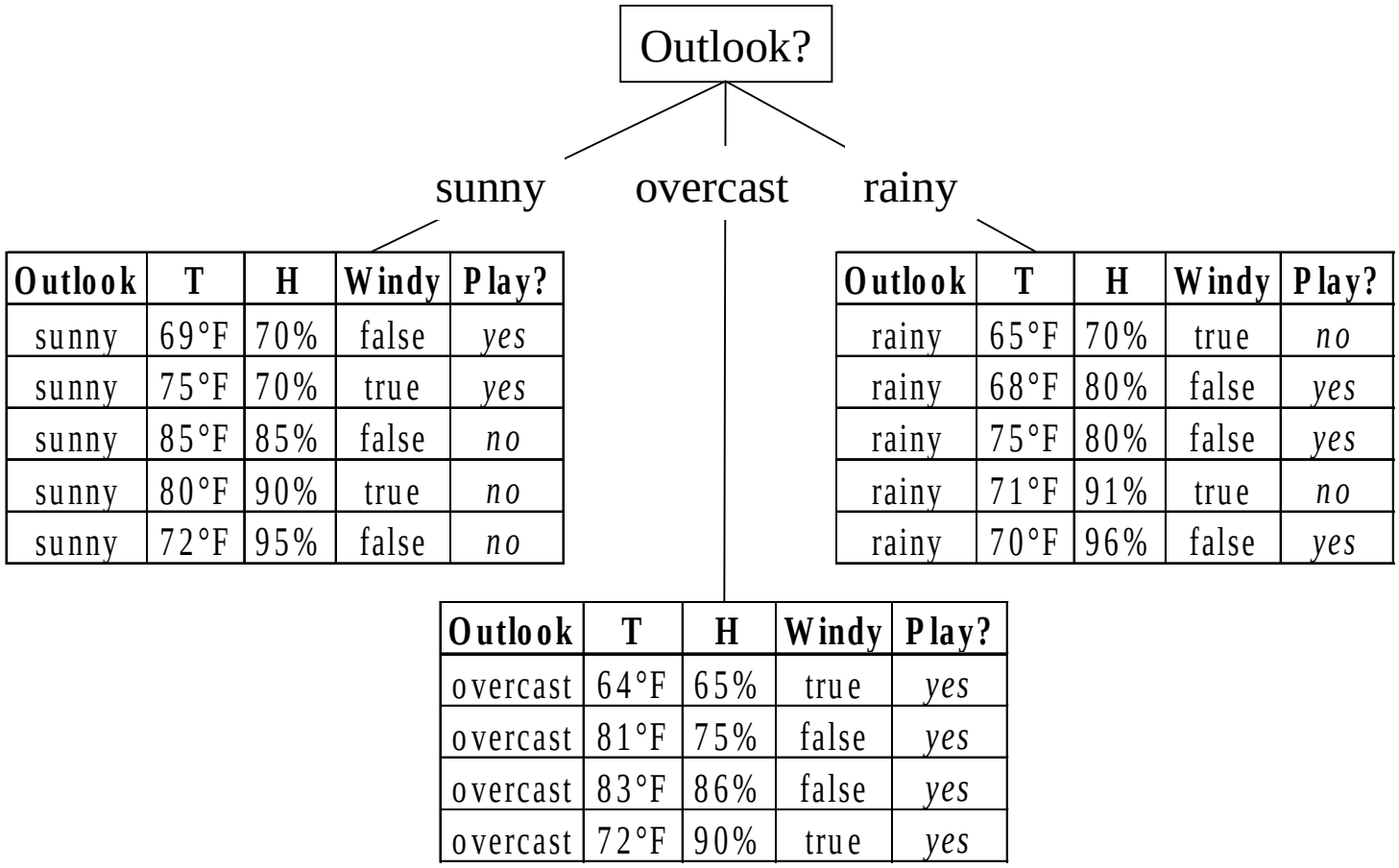
InfoGain(Outlook) = .940 – 5/14*.971 – 4/14*0.00 – 5/14*.971 = **0.246 [bits]**
InfoGain(Windy) = .940 – 8/14*.811 – 6/14*1.00 = 0.048 [bits]
Best splitpoint for numeric attribute humidity (82.5) = 0.152 [bits]
Best splitpoint for numeric attribute temperature (84.0) = 0.113 [bits]

Overall best split: Outlook = Root node

Propagate examples into three subnodes according to values of Outlook...

alex@seewald.at / alex.seewald.at

Example: Choose root node for weather (2)



Outlook	T	H	Windy	Play?
sunny	69°F	70%	false	*yes*
sunny	75°F	70%	true	*yes*
sunny	85°F	85%	false	*no*
sunny	80°F	90%	true	*no*
sunny	72°F	95%	false	*no*

Outlook	T	H	Windy	Play?
rainy	65°F	70%	true	*no*
rainy	68°F	80%	false	*yes*
rainy	75°F	80%	false	*yes*
rainy	71°F	91%	true	*no*
rainy	70°F	96%	false	*yes*

Outlook	T	H	Windy	Play?
overcast	64°F	65%	true	*yes*
overcast	81°F	75%	false	*yes*
overcast	83°F	86%	false	*yes*
overcast	72°F	90%	true	*yes*

Call ID3+ recursively for each subnode node with the appropriate subset of training examples **(see above)**

alex@seewald.at / alex.seewald.at

Lecture Plan Update

Change in (Lecture) Plan: We will first finish the most important learning systems for another few weeks, and then look at the advantages/disadvantages and specific examples using these learning algorithms.

Next Lesson (in two weeks **at IMKAI**)

- Overfitting avoidance for decision trees: Pre/Postpruning
- Dealing with missing values in decision trees & in general
- Common preprocessing transformations
- Bayesian Methods

...

alex@seewald.at / alex.seewald.at