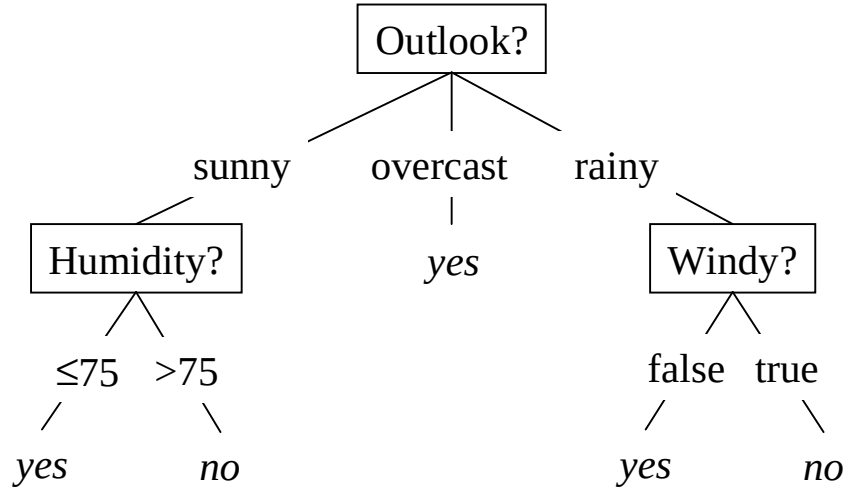
Decision Trees

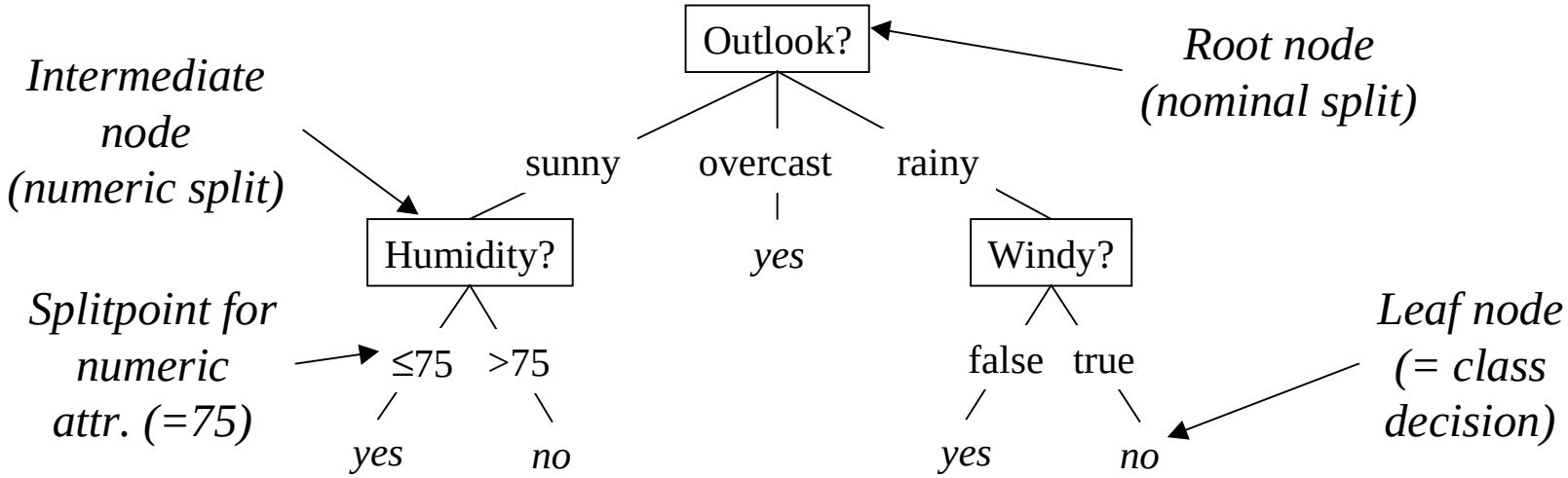


Lektor Dr.techn. Alexander K. Seewald
Österreichisches Forschungsinstitut
für Artificial Intelligence

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.

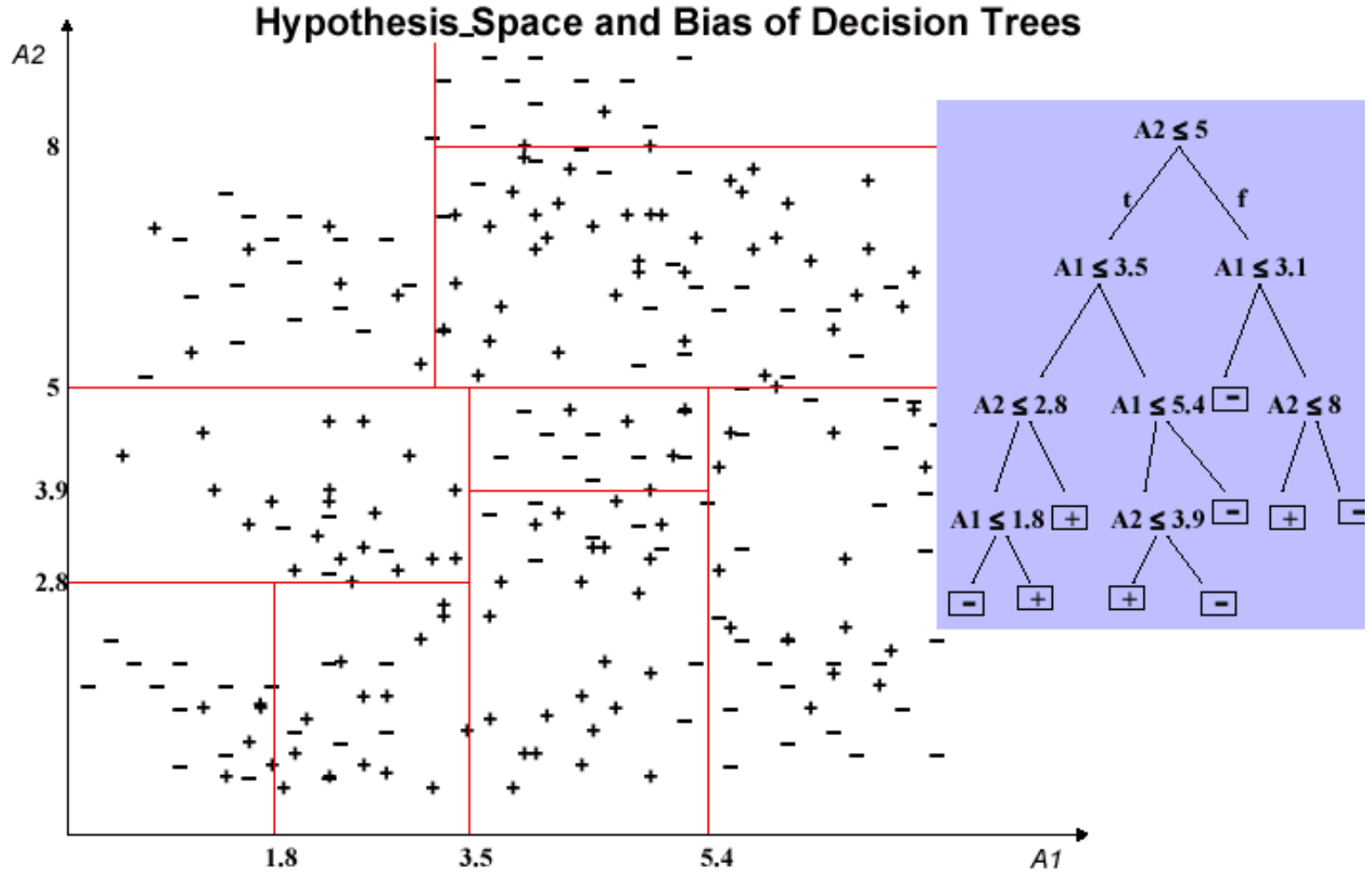
Pseudocode for ID3+ (i.e. ID3 extended with numeric attribute splits)

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)

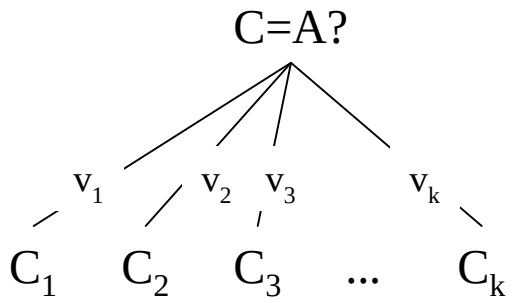
A ... some attribute with possible values v_1, ..., v_k

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

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

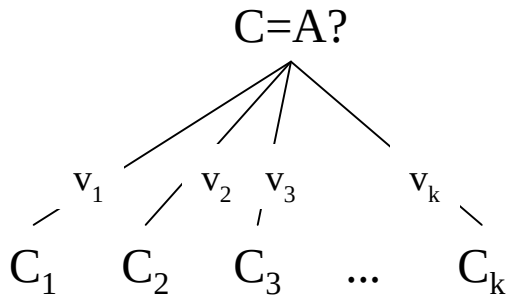
p, n ... number of positive / negative examples in C ($p+n = N$)

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]

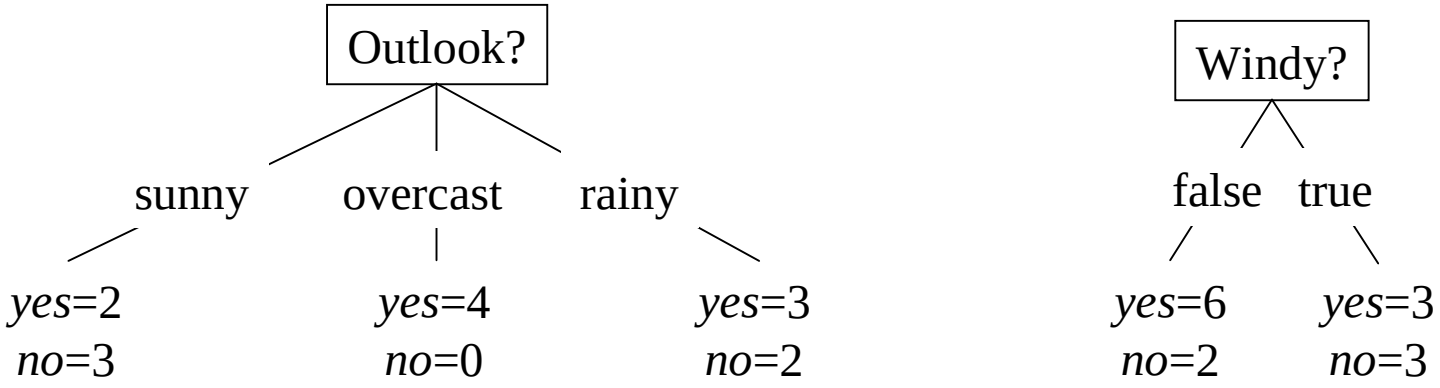
95.5 (8/5 vs. 1/0) → InfoGain = 0.048 [bits]

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



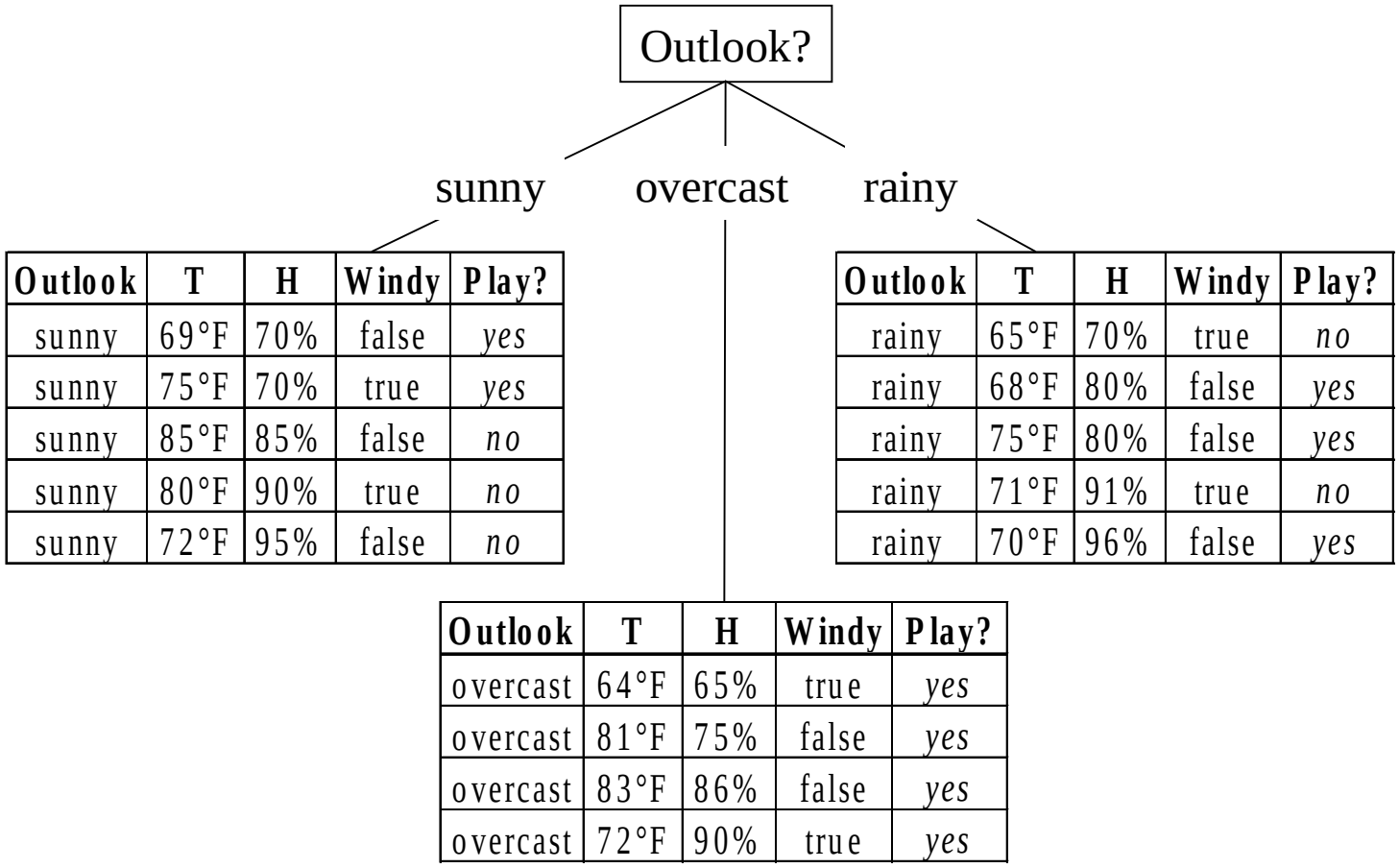
InfoGain(Outlook) = .940 – 5/14*.971 – 4/14*0.00 – 5/14*.971 = **0.246 [bits]**
InfoGain(Windy) = .940 – 8/14*.811 – 6/14*1.00 = 0.048 [bits]
Best splitpoint for numeric attribute humidity (82.5) = 0.152 [bits]
Best splitpoint for numeric attribute temperature (84.0) = 0.113 [bits]

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

Common Problem with DTs: Overfitting

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
sunny	***cool***	***normal***	***true***	***Don't Play***

Additional training example with incorrect class

```
outlook = sunny
|   humidity = high: no (3)
|   humidity = normal: yes (2)
outlook = overcast: yes (4)
outlook = rainy
|   windy = TRUE: no (2)
|   windy = FALSE: yes (3)
```

```
outlook = sunny
|   humidity = high: no (3)
|   humidity = normal
|   |   temperature = hot: yes (0)
|   |   temperature = mild: yes (1)
|   |   temperature = cool
|   |   |   windy = TRUE: no (1)
|   |   |   windy = FALSE: yes (1)
outlook = overcast: yes (4)
outlook = rainy
|   windy = TRUE: no (2)
|   windy = FALSE: yes (3)
```


alex@seewald.at / alex.seewald.at

How to avoid Overfitting for DTs

Pre-Pruning

Stop splitting a node further (even if it still contains examples of different classes, and even if some attributes are still available) if there seems to be no statistically significant correlation between attributes and classes

Post-Pruning

First construct (possibly complex) tree that is maximally consistent with the training data (i.e., has minimum error on training data)

Then simplify the tree by cutting off branches and subtrees that seem harmful.

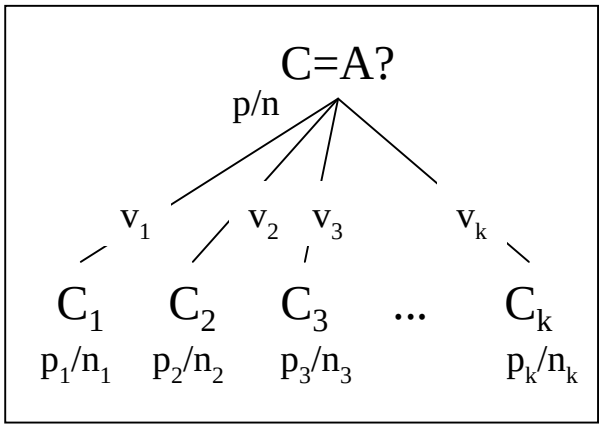
Effects of Pruning

- Simpler trees with lower accuracy on the training data but possibly higher accuracy on new, unseen data.
- Improves handling of attribute and class noise

alex@seewald.at / alex.seewald.at

Pre-Pruning

Pre-Pruning: X^2 Test (pronounced: Chi-Square)



If A is **completely irrelevant** to the class of an object in C, the expected value of p_i is $p_i' = p * |C_i|/|C|$ and the expected value of n_i is $n_i' = n * |C_i|/|C|$, where p and n are the number of positive resp. negative examples of the class.

⇒ The larger the differences $|p_i - p_i'|$ and $|n_i - n_i'|$, the smaller the likelihood that A is completely irrelevant.

⇒ Statistic $S = \sum_{i=1}^{k} \frac{(p_i - p_i')^2}{p_i'} + \frac{(n_i - n_i')^2}{n_i'}$ is approximately X^2 distributed with *k-1* d.o.f.

⇒ Perform X^2 test: S large enough? (Intuition: the smaller S, the higher the probability that A is irrelevant to the class (i.e., class is independent of A))

⇒ Prune (stop refining a node) if there is no relevant A at given confidence level

alex@seewald.at / alex.seewald.at

Post-Pruning

Reduced Error Pruning (Pseudocode)

1. Randomly split training examples TD into a training set TS (usually 70%) and a pruning (validation) set PS (usually 30%)
2. Learn a (possibly complex) tree from TS that is as consistent with the data as possible (i.e., that possibly overfits the data)
3. Perform tree simplification step:

   For each subtree T_i of T, tentatively replace T_i by a majority class leaf
4. Compare the accuracy on PS(!) for all modified subtrees with accuracy of original T on PS:

- If there is no T_i that improves accuracy on PS when removed: exit
- Otherwise: remove (and replace with leaf) T_i with maximum improvement.

5. Go to 3.

Question: Why additional pruning set PS? Why not use original training set TD for making pruning decisions?

alex@seewald.at / alex.seewald.at

Post-Pruning without Validation Set: PEP

Pessimistic Error Pruning (PEP)

(used in well-known C4.5 decision tree learner)

Replace subtree T_i by a majority class leaf, if and only if

$$E+0.5 < \sum J + L(T_i)/2 + SE$$

where...

- $\sum J$ number of training set errors for subtree T_i before replacing
- E number of training set errors when replacing T_i by a leaf (only for those examples which are within the subtree T_i)
- $L(T_i)$ total number of leaves in subtree T_i
- SE standard error: $\sqrt{\frac{(\sum J+L(T_i)/2)(\sum K-(\sum J+L(T_i)/2))}{\sum K}}$
- $\sum K$ number of examples in subtree T_i ($=p_i+n_i=|C_i|$)

\+ *no need for validation set - all training data can be used; very efficient*

– *heuristic is ad-hoc and not reasonably grounded in statistical theory*

alex@seewald.at / alex.seewald.at