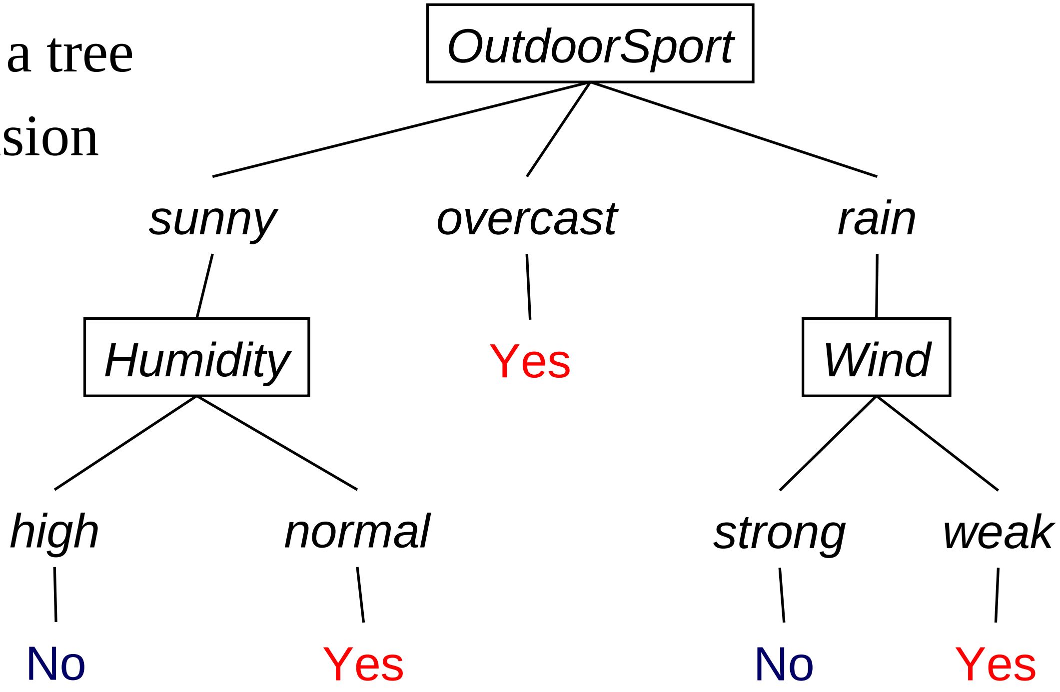
Artificial Intelligence

CS60045

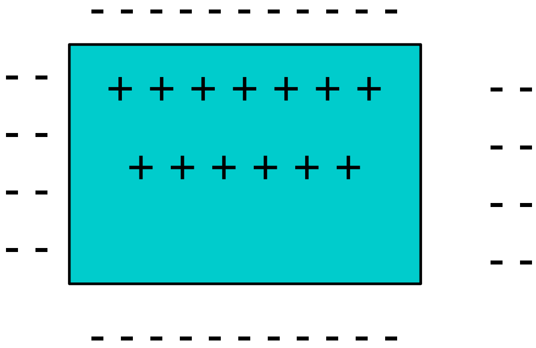
Decision Tree Learning

Representation of Concepts

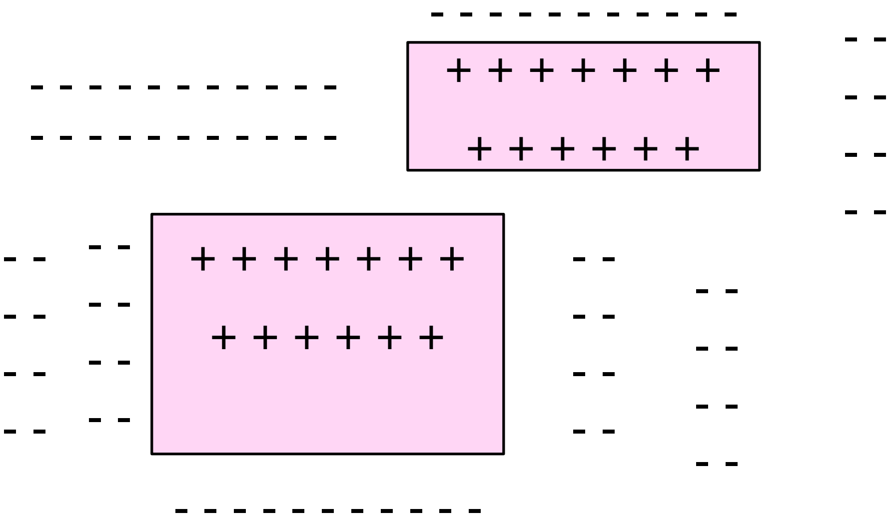
- Decision trees: disjunction of conjunction of attributes
  - (Sunny AND Normal) OR (Overcast) OR (Rain AND Weak) +
  - More powerful representation
  - Larger hypothesis space H
  - Can be represented as a tree
  - Common form of decision making in humans



Rectangle learning….



Conjunctions
(single rectangle)



Disjunctions of Conjunctions
(union of rectangles)

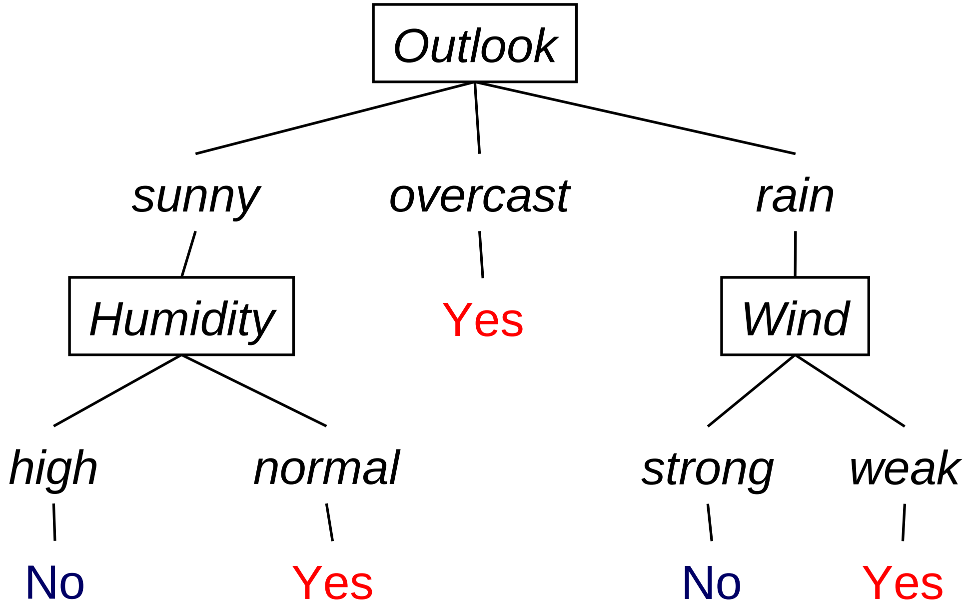
Training Examples

Day	Outlook	Temp	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*

Decision Trees

- Decision tree to represent learned target functions
  - Each internal node <u>tests</u> an attribute
  - Each branch corresponds to <u>attribute value</u>
  - Each leaf node assigns a classification

- Can be represented by logical formulas



Representation in Decision Trees

- Example of representing rule in DT’s:

  if outlook = sunny AND humidity = normal

  OR

  if outlook = overcast

  OR

  if outlook = rain AND wind = weak

  then playtennis

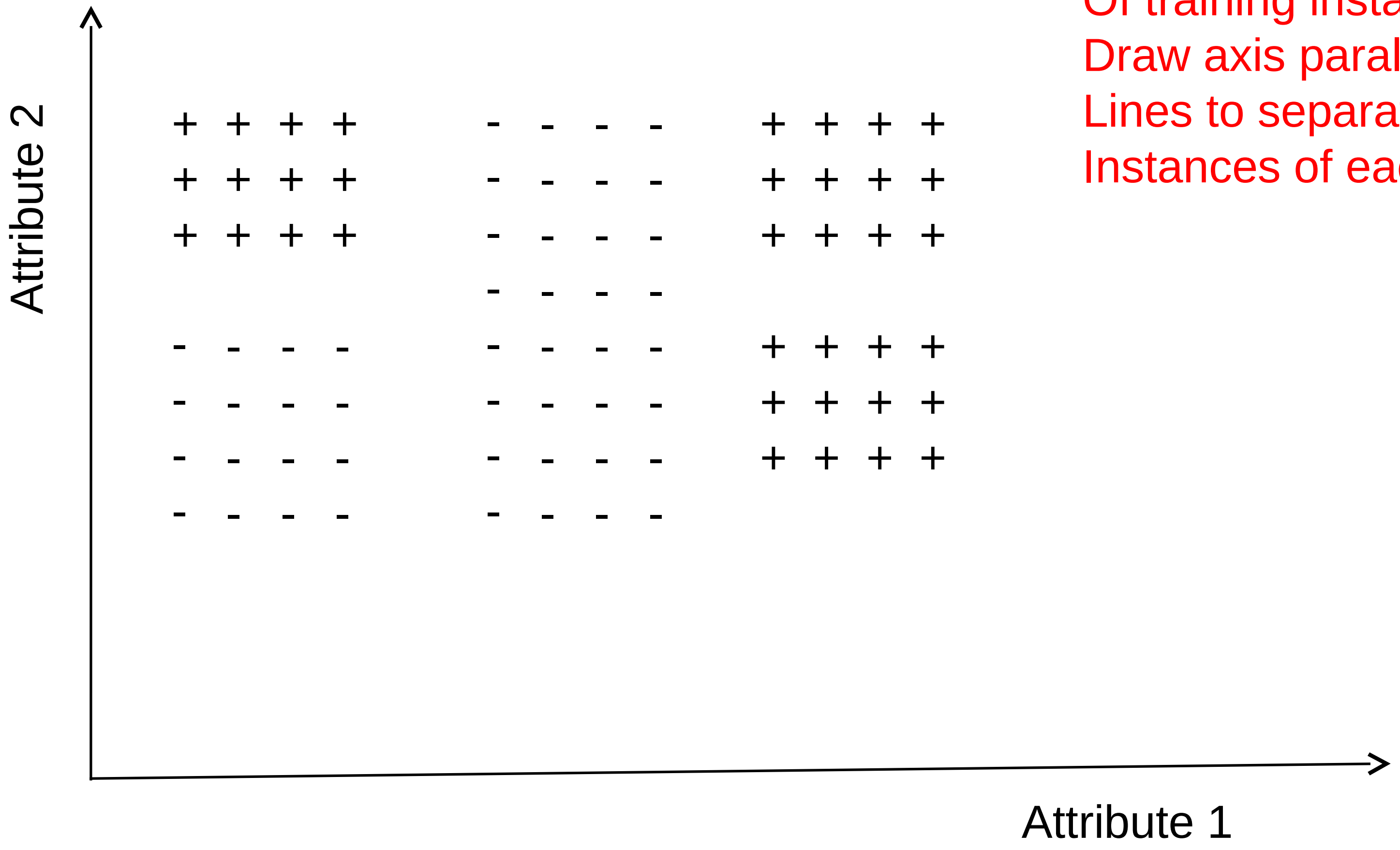
- Decision Tree Construction:
  - Find the best structure
  - Given a training data set

Applications of Decision Trees

- Instances describable by a fixed set of attributes and their values
- Target function is discrete valued
  - 2-valued
  - N-valued
  - But can approximate continuous functions
- Disjunctive hypothesis space
- Possibly noisy training data
  - Errors, missing values, …

- Examples:
  - Equipment or medical diagnosis
  - Credit risk analysis
  - Calendar scheduling preferences

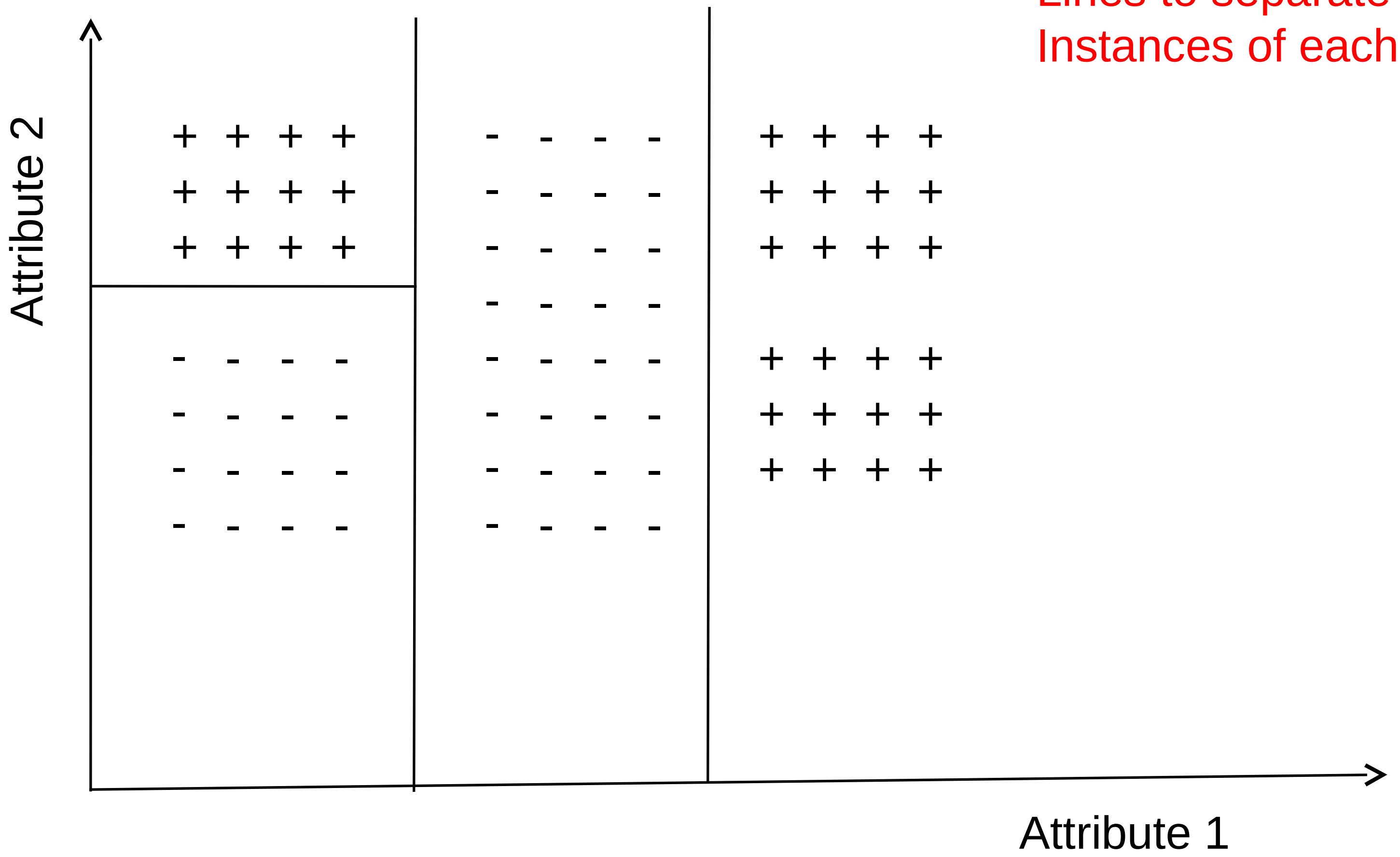
Decision Trees



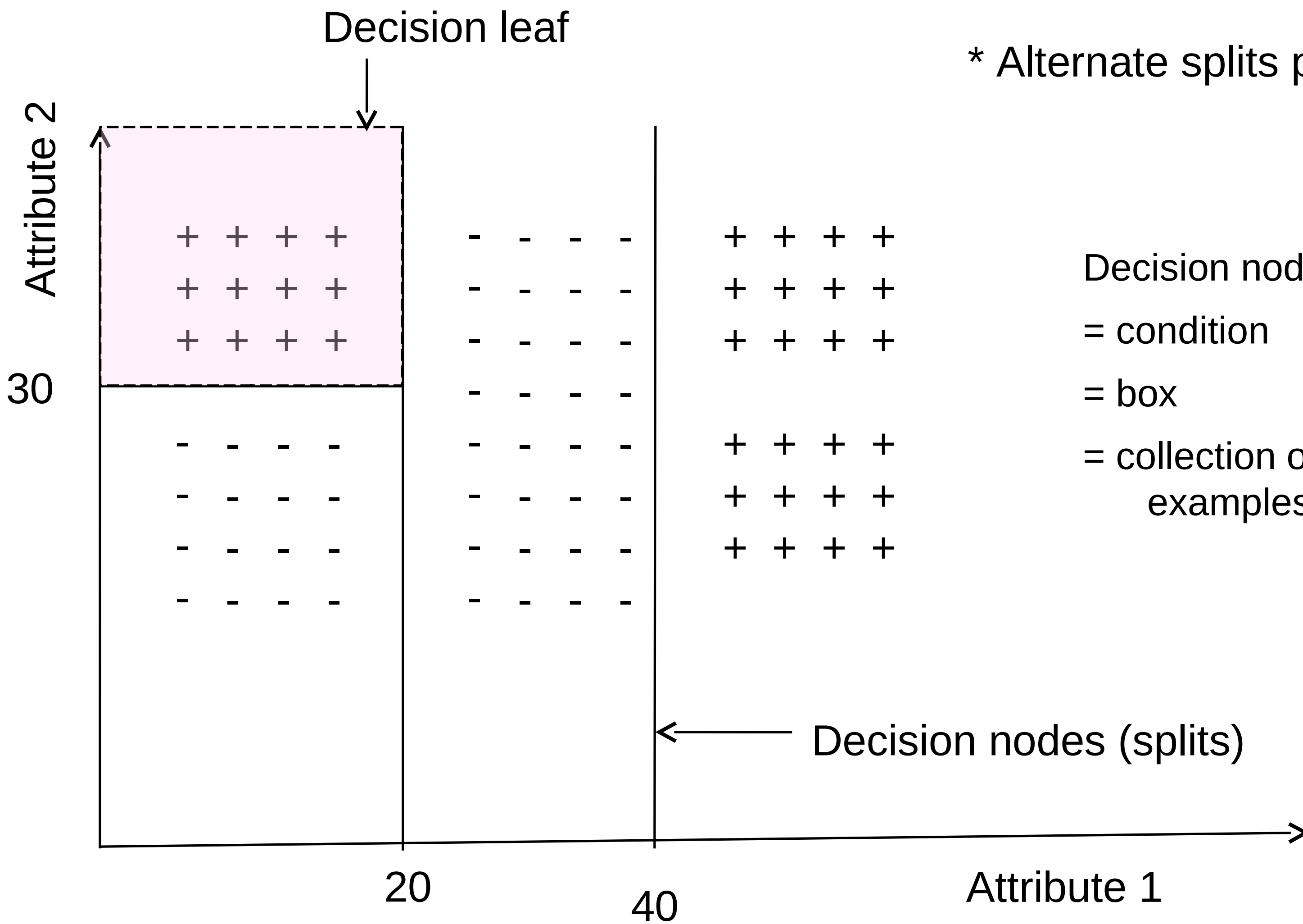
Given distribution
Of training instances
Draw axis parallel
Lines to separate the
Instances of each class

Decision Tree Structure

Draw axis parallel
Lines to separate the
Instances of each class



Decision Tree Structure



* Alternate splits possible

Decision node

= condition

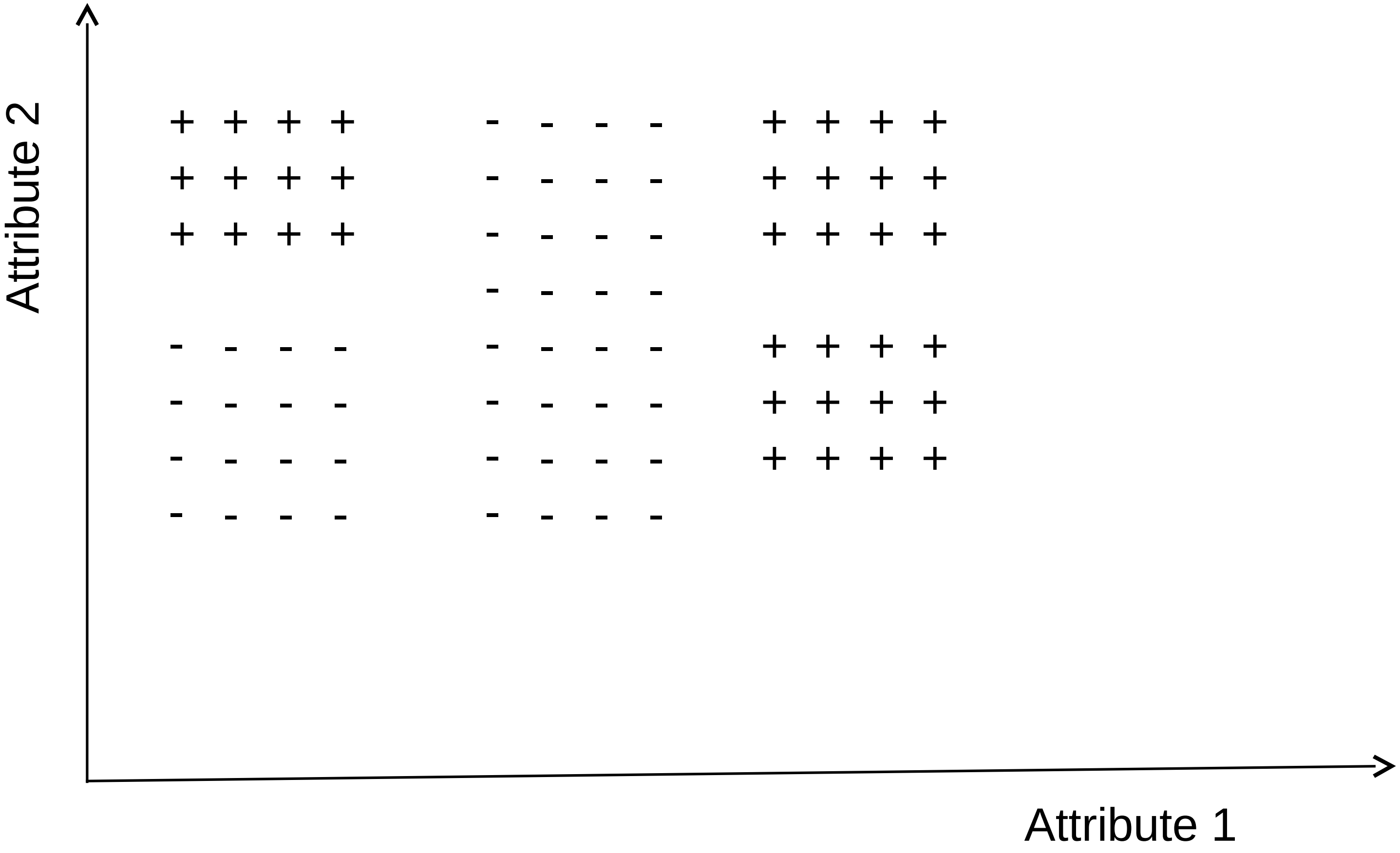
= box

= collection of satisfying examples

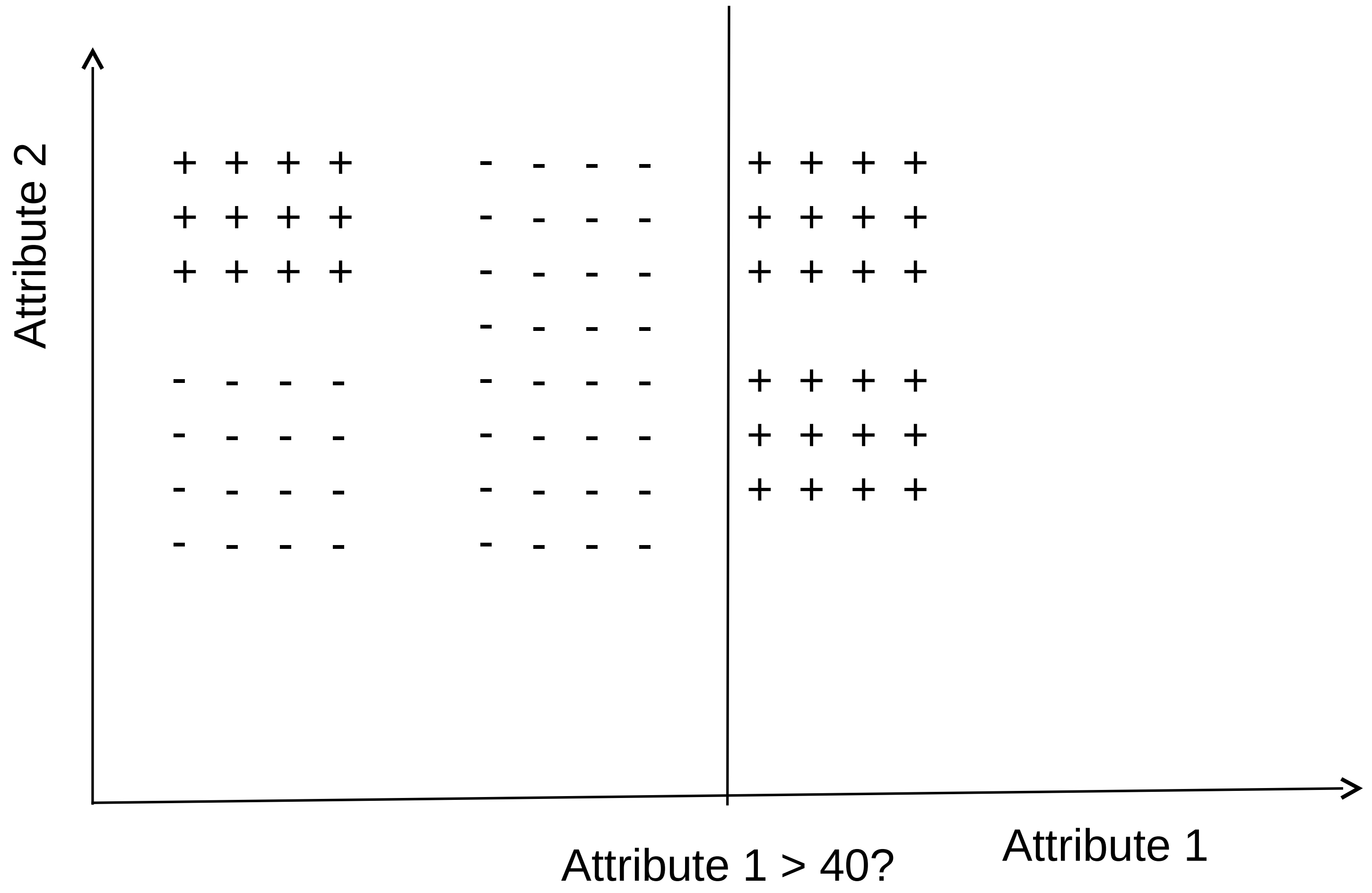
Top-Down Construction

- Start with empty tree
- Main loop:
  1. Split the “best” decision attribute (*A)* for next node
  2. Assign *A* as decision attribute for node
  3. For each value of *A*, create new descendant of node
  4. Sort training examples to leaf nodes
  5. If training examples perfectly classified, STOP,
     Else iterate over new leaf nodes
- Grow tree just deep enough for perfect classification
  - If possible (or can approximate at chosen depth)
- Which attribute is best?

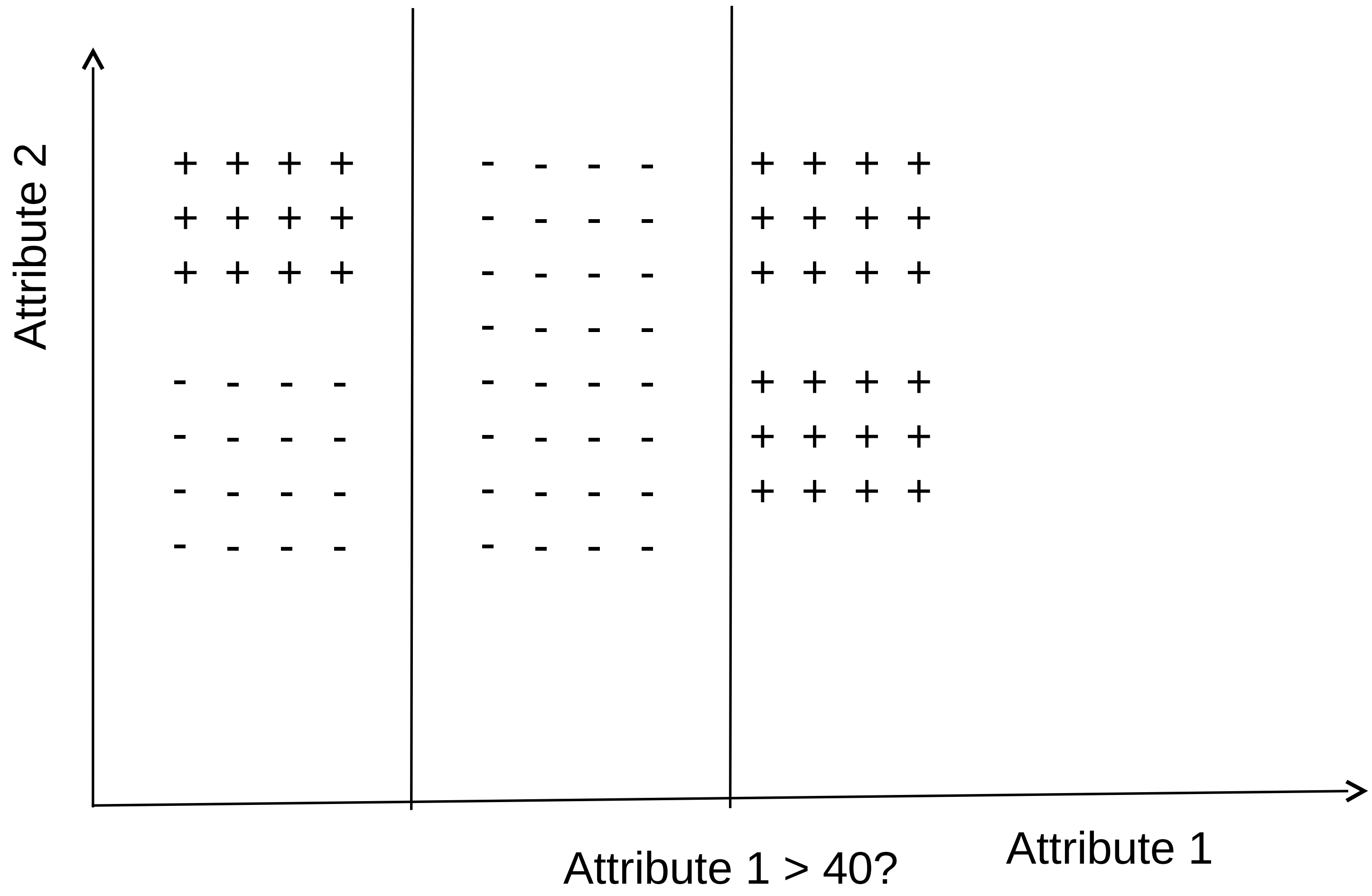
Best attribute to split?



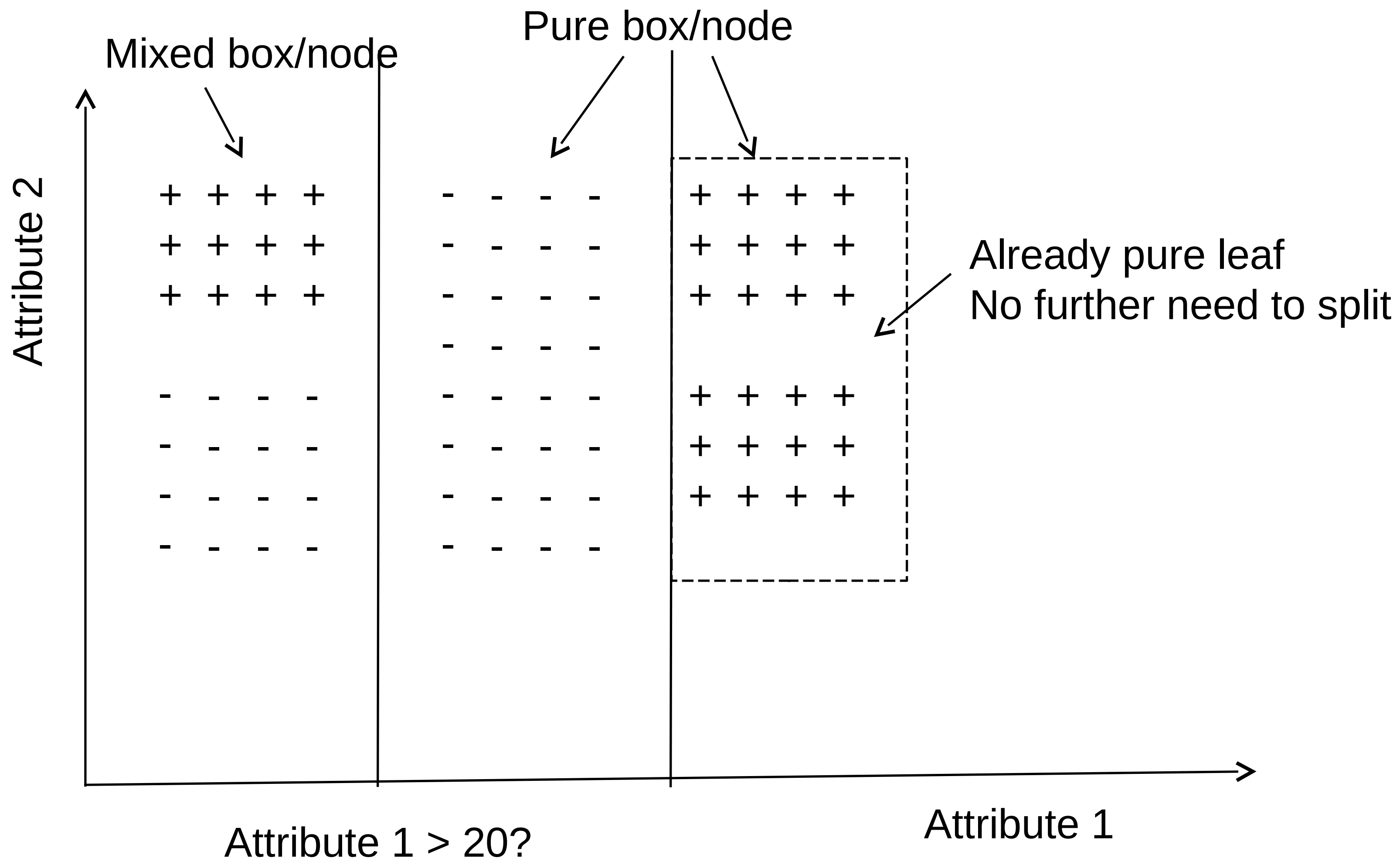
Best attribute to split?



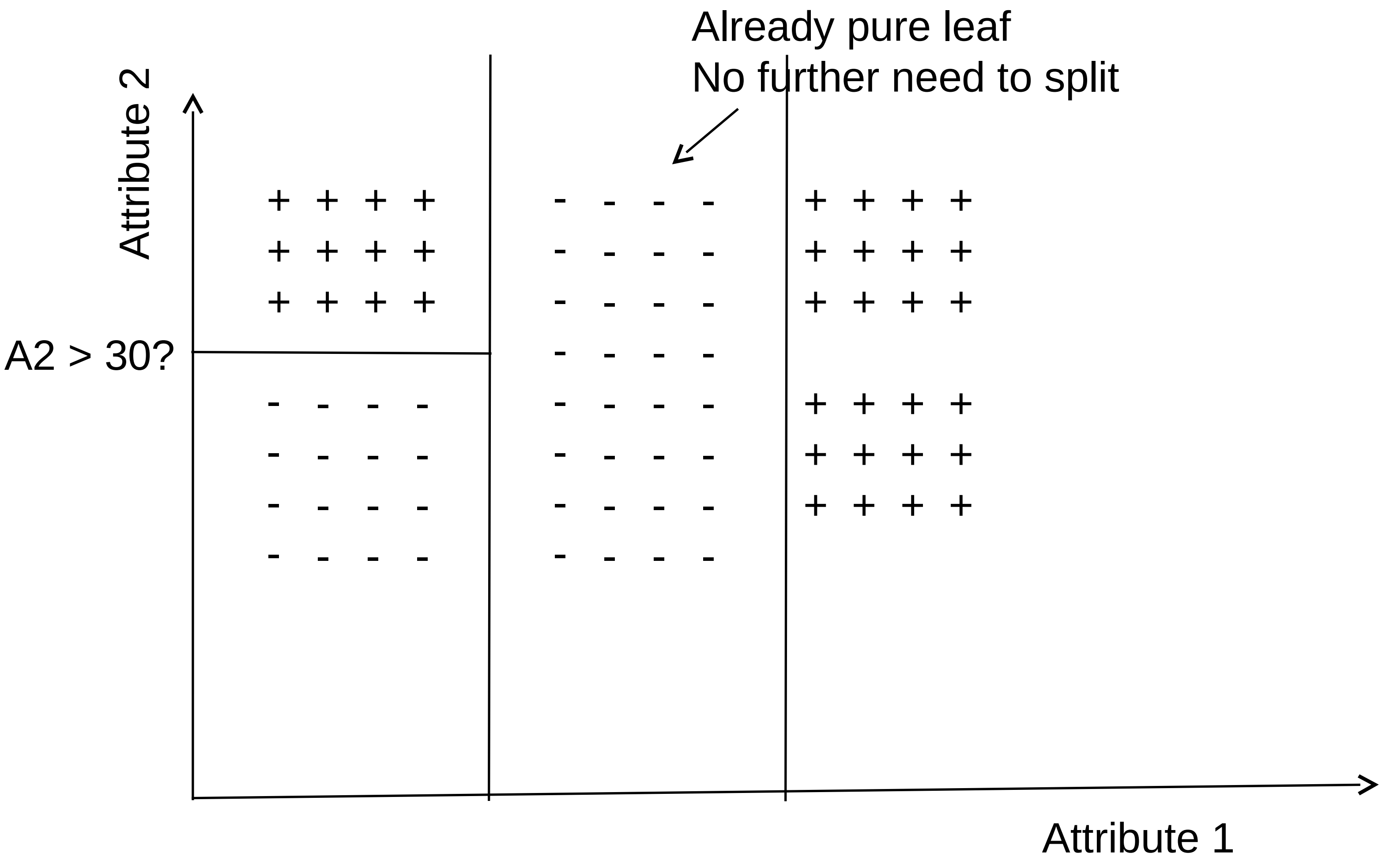
Best attribute to split?



Which split to make next?



Which split to make next?

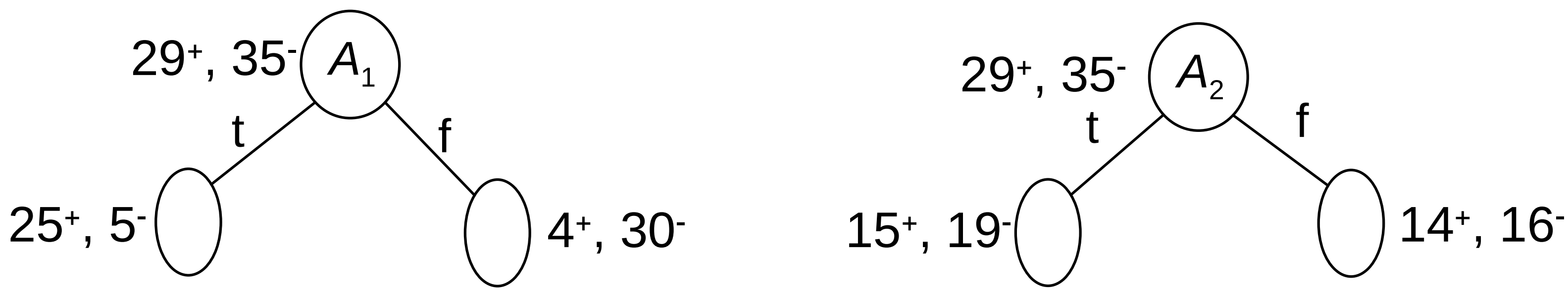


Principle of Decision Tree Construction

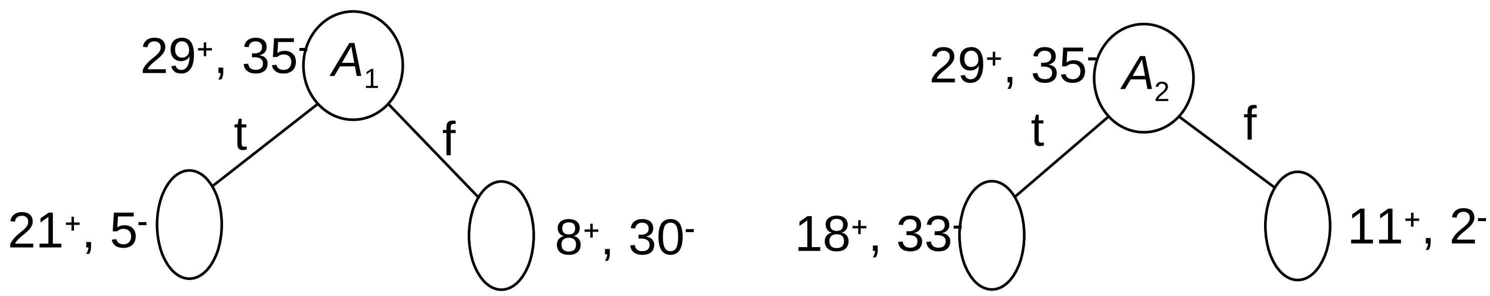
- Finally we want to form pure leaves
  - Correct classification
- Greedy approach to reach correct classification
  1. Initially treat the entire data set as a single box
  2. For each box choose the spilt that reduces its impurity (in terms of class labels) by the maximum amount
  3. Split the box having highest reduction in impurity
  4. Continue to Step 2
  5. Stop when all boxes are pure

Choosing Best Attribute?

- Consider 64 examples, 29^+ and 35^-
- Which one is better?



- Which is better?



Entropy

- A measure for
  - uncertainty
  - purity
  - information content
- Information theory: optimal length code assigns ($-\log_2 p$) bits to message having probability p
- S is a sample of training examples
  - p_+ is the proportion of positive examples in S
  - p_- is the proportion of negative examples in S
- Entropy of S: average optimal number of bits to encode information about certainty/uncertainty about S

$$Entropy(S) = p_+(-\log_2 p_+) + p_-(-\log_2 p_-) = -p_+\log_2 p_+ - p_-\log_2 p_-$$

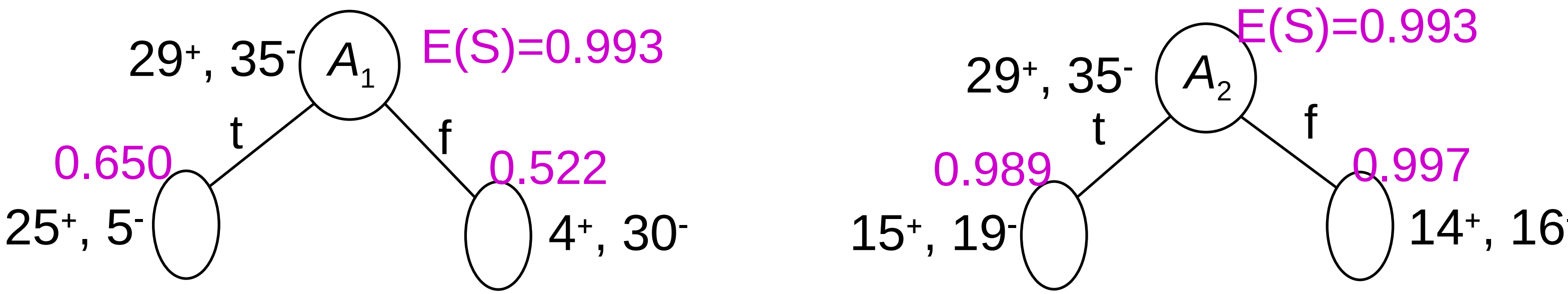
- Can be generalized to more than two values

Entropy

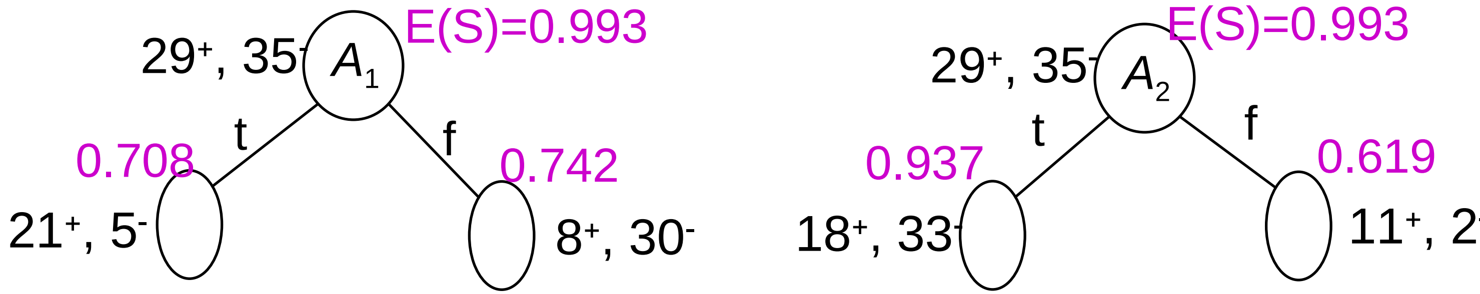
- Entropy can also be viewed as measuring
  - purity of S,
  - uncertainty in S,
  - information in S, …
- Example: values of entropy for p+=1, p+=0, p+=.5
- Easy generalization to more than binary values
  - $\Sigma^{n}_{i=1}\ p_i *(-\log_2 p_i)$, i=1..n
    - i is + or – for binary
    - i varies from 1 to n in the general case

Choosing Best Attribute?

- Consider 64 examples (29^+,35^-) and compute entropies:
- Which one is better?



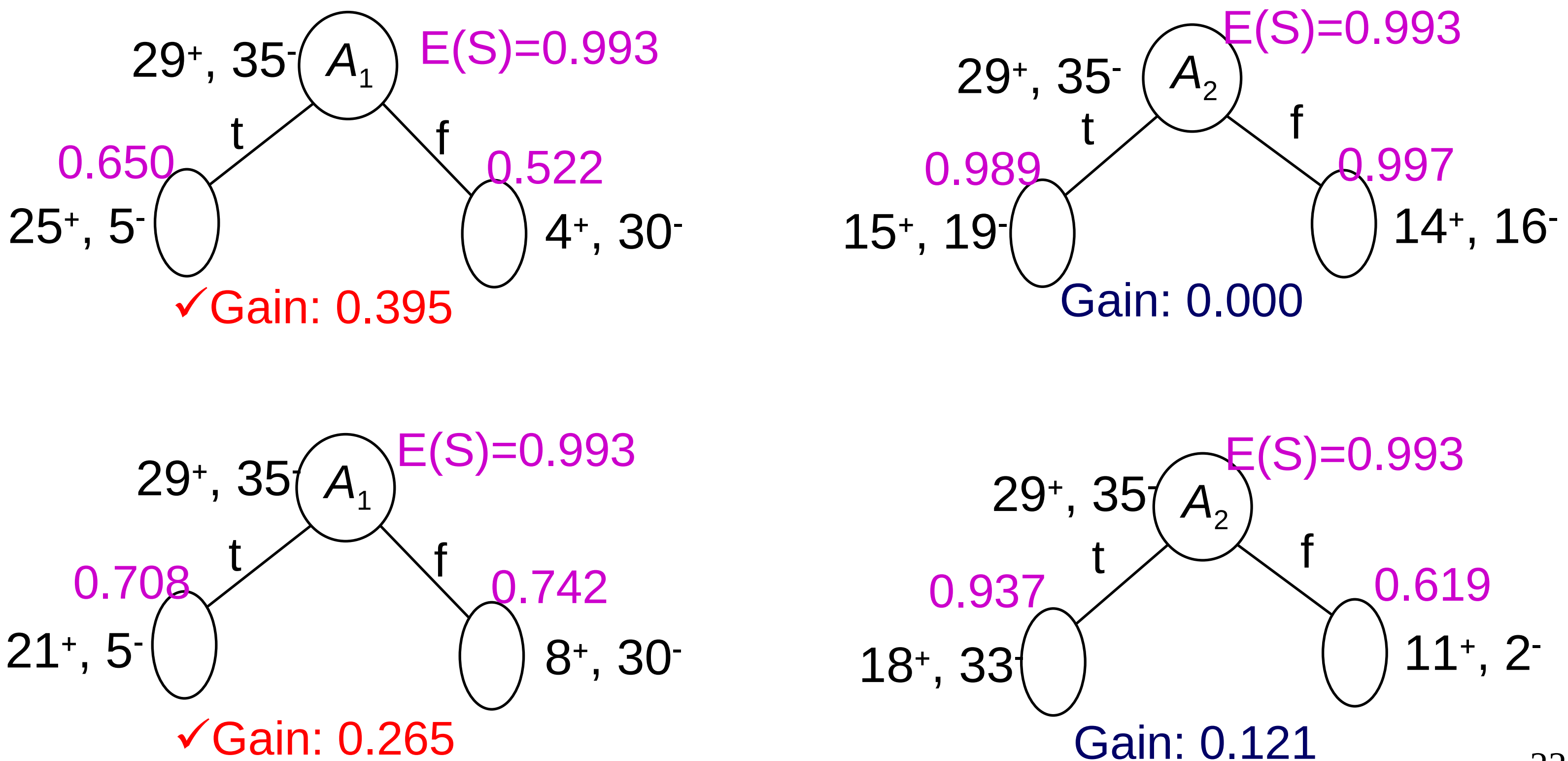
- Which is better?



Information Gain

- *Gain(S,A)*: reduction in entropy after choosing attr. *A*

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



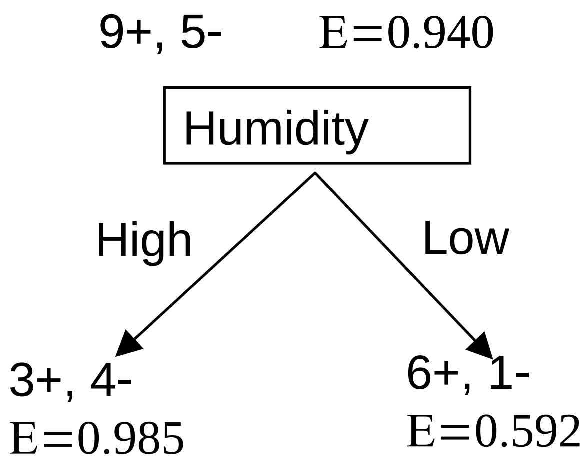
Gain function

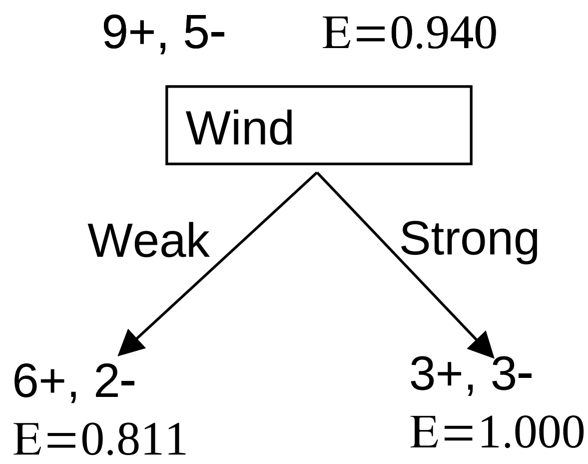
- Gain is measure of how much can
  - Reduce uncertainty
    - Value lies between 0,1
    - What is significance of
      - gain of 0?
        - example where have 50/50 split of +/- both before *and* after discriminating on attributes values
      - gain of 1?
        - Example of going from “perfect uncertainty” to perfect certainty after splitting example with predictive attribute
  - Find “patterns” in TE’s relating to attribute values
    - Move to locally minimal representation of TE’s

Training Examples (re-look)

Day	Outlook	Temp	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*

Determine the Root Attribute





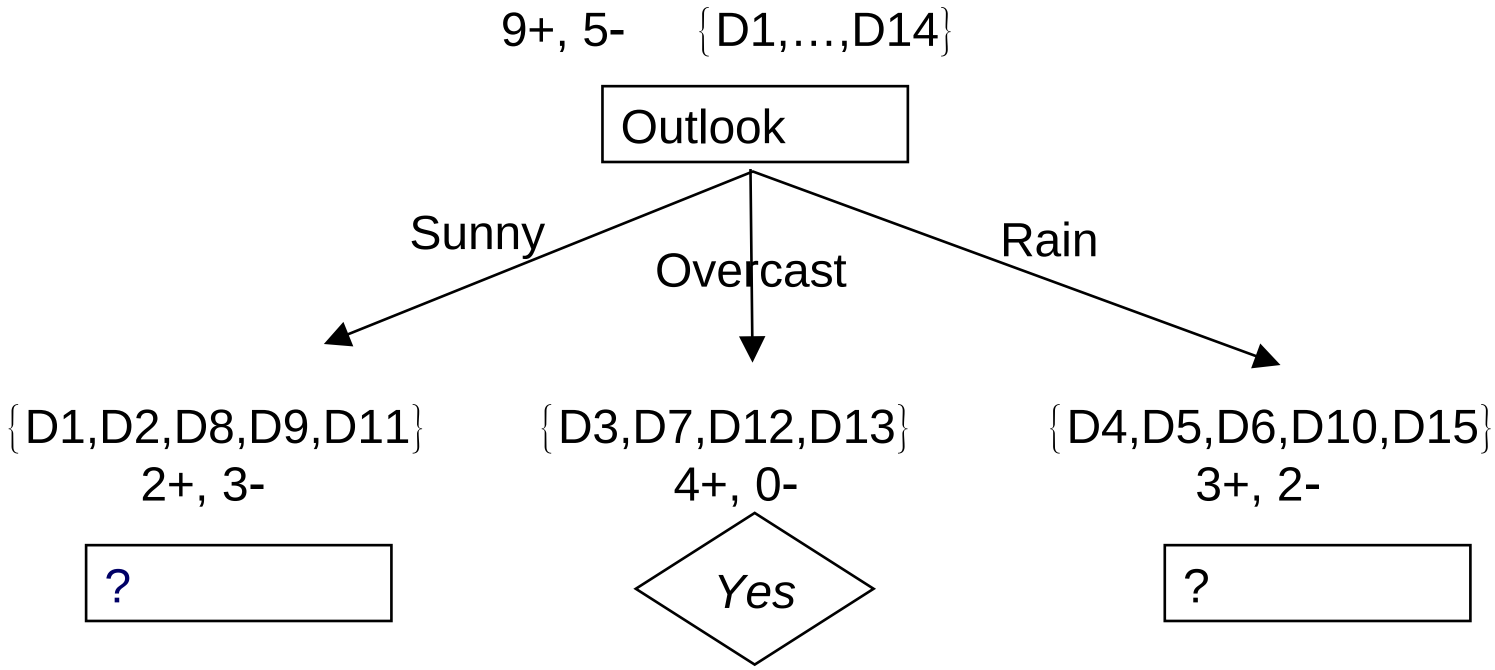
Gain (S, Humidity) = 0.151

Gain (S, Wind) = 0.048

Gain (S, Outlook) = 0.246

Gain (S, Temp) = 0.029

Sort the Training Examples



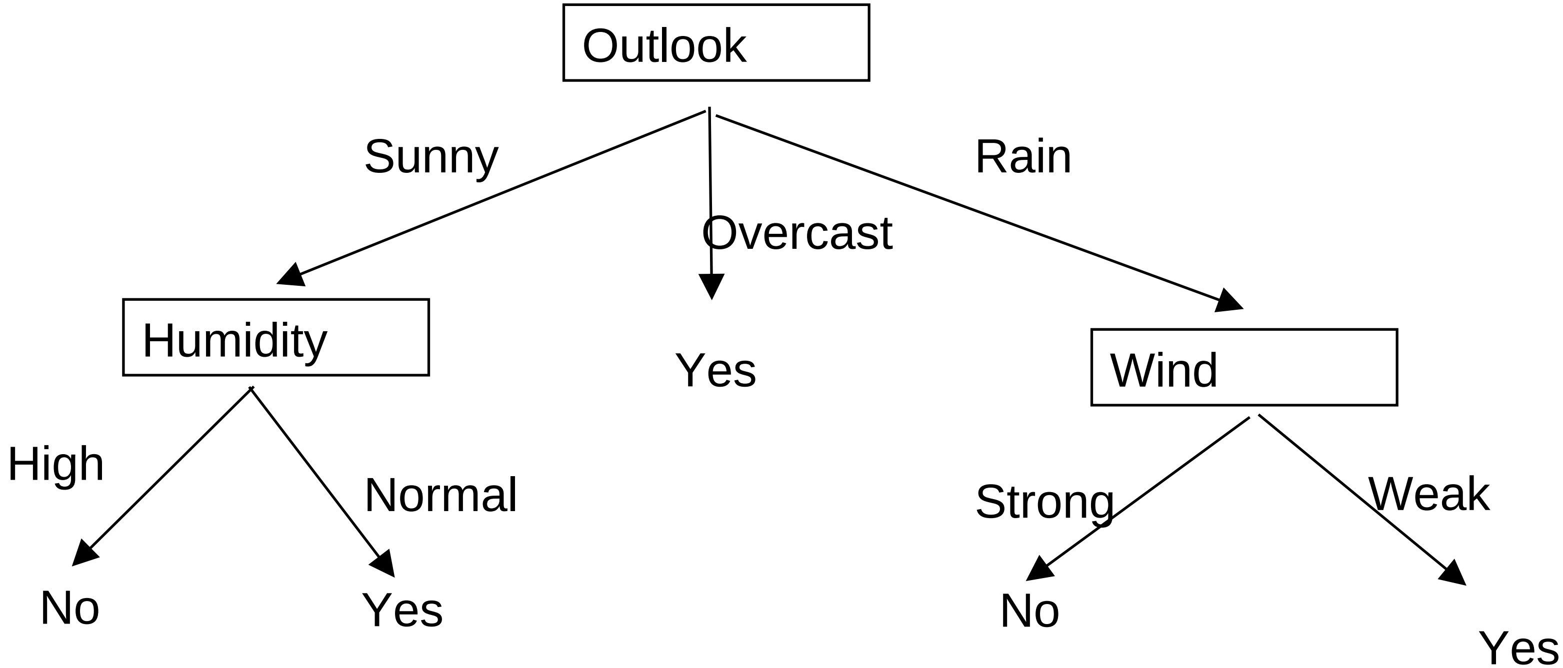
S_{sunny}= {D1, D2, D8, D9, D11}

Gain (S_{sunny}, Humidity) = .970

Gain (S_{sunny}, Temp) = .570

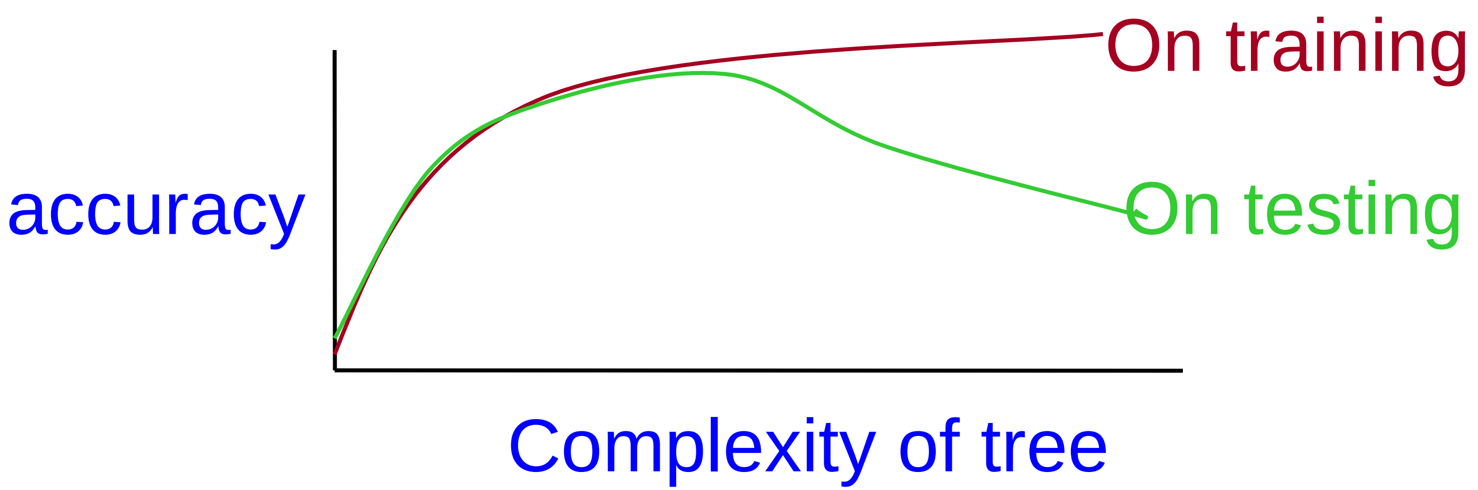
Gain (S_{sunny}, Wind) = .019

Final Decision Tree for Example

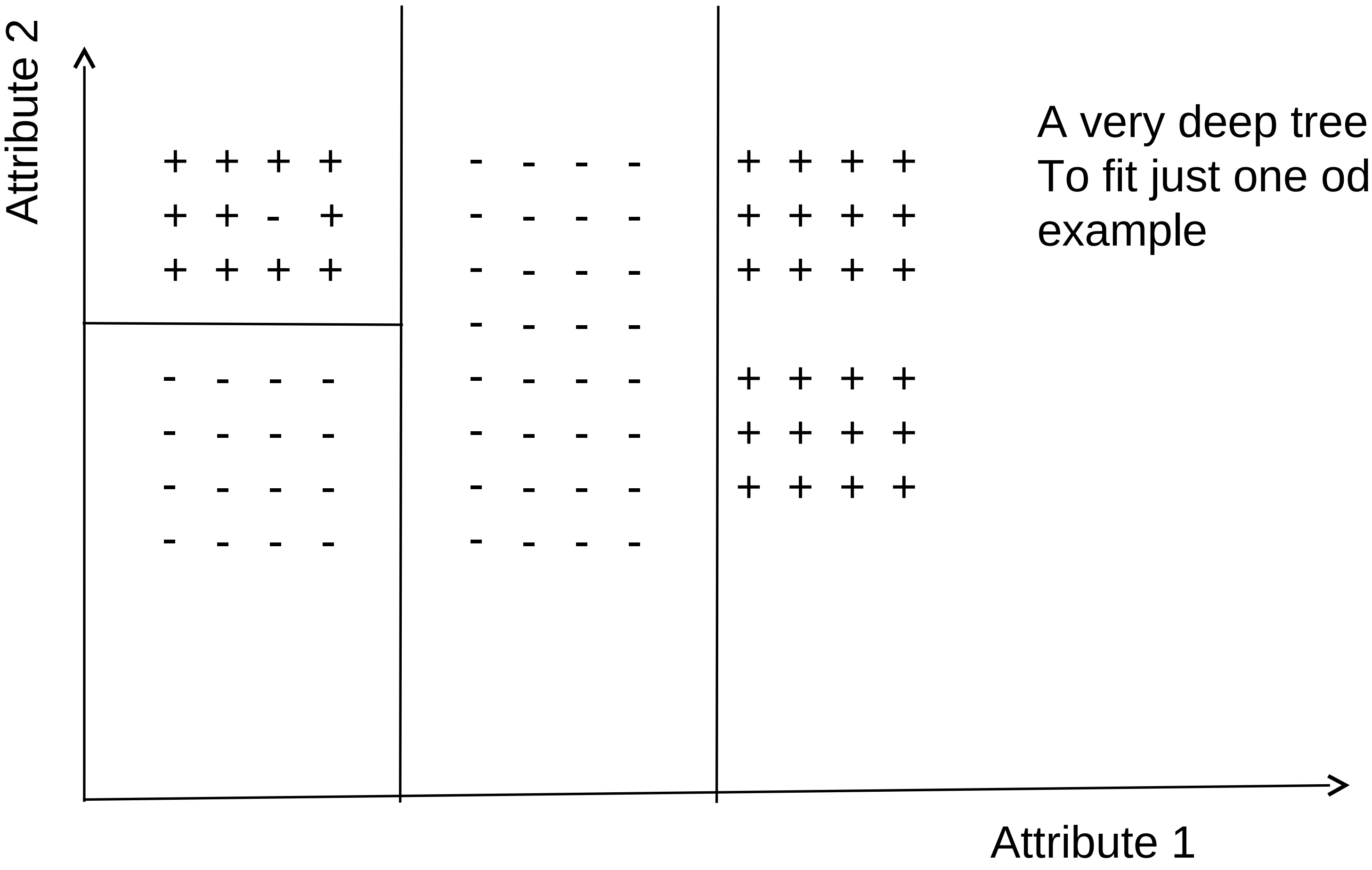


Overfitting the Data

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data the tree is fitting
  - The algorithm might be making decisions based on very little data
- A hypothesis *h* is said to overfit the training data if the is another hypothesis, h', such that *h* has smaller error than *h'* on the training data but *h* has larger error on the test data than *h'*.

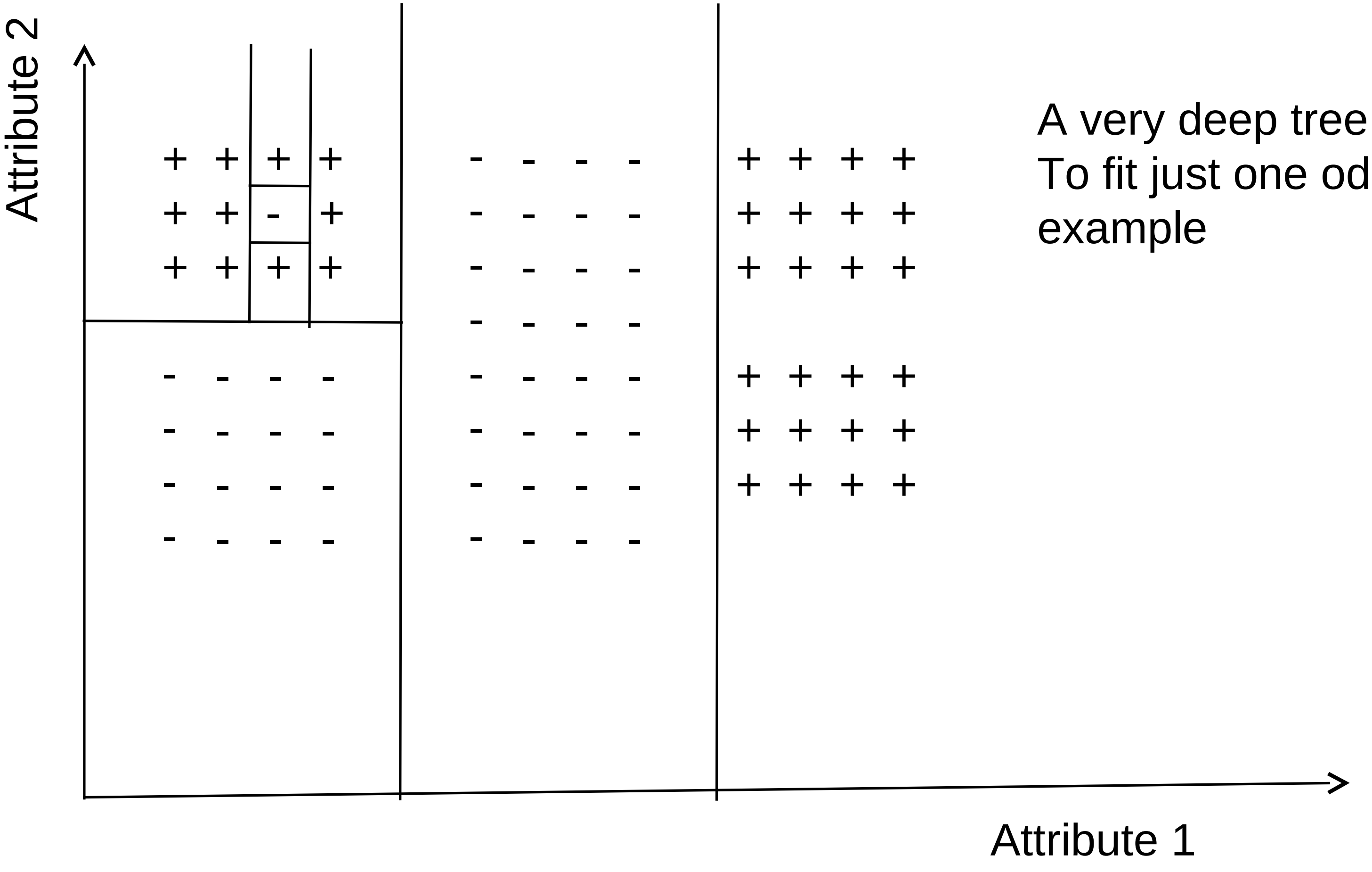


Overfitting



A very deep tree required
To fit just one odd training
example

When to stop splitting further?



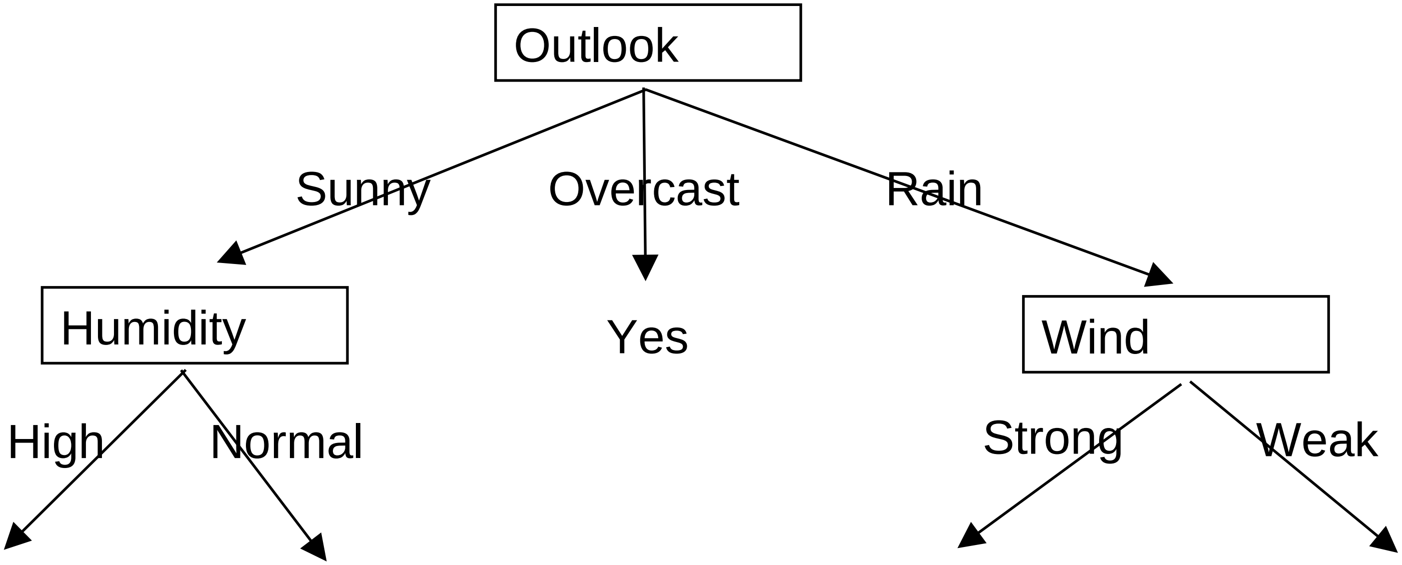
A very deep tree required
To fit just one odd training
example

Overfitting in Decision Trees

- Consider adding *noisy* training example (should be +):

Day	Outlook	Temp	Humidity	Wind	Tennis?
D15	Sunny	Hot	Normal	Strong	*No*

- What effect on earlier tree?



Overfitting: An Example

Noise or other coincidental regularities

Avoiding Overfitting

- Two basic approaches
  - Prepruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Postpruning: Grow the full tree and then remove nodes that seem not to have sufficient evidence. (more popular)

- Methods for evaluating subtrees to prune:
  - Cross-validation: Reserve hold-out set to evaluate utility (more popular)
  - Statistical testing: Test if the observed regularity can be dismissed as likely to be occur by chance
  - Minimum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions ?

    This is related to the notion of regularization that we will see in other contexts– keep the hypothesis simple.

Reduced-Error Pruning

- A post-pruning, cross validation approach
  - Partition training data into "grow" set and "validation" set.
  - Build a complete tree for the "grow" data
  - Until accuracy on validation set decreases, do:

    For each non-leaf node in the tree

    Temporarily prune the tree below; replace it by majority vote.

    Test the accuracy of the hypothesis on the validation set

    Permanently prune the node with the greatest increase

    in accuracy on the validation test.
- Problem: Uses less data to construct the tree
- Sometimes done at the rules level

General Strategy: Overfit and Simplify

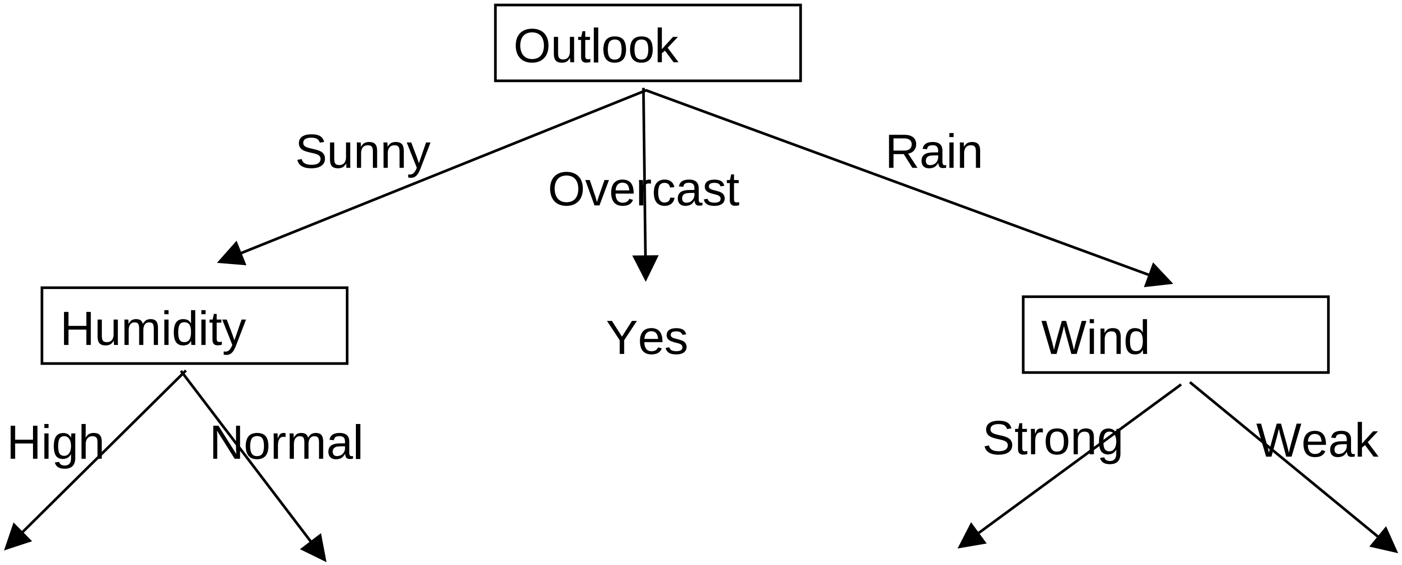
Rule post-pruning

- Allow tree to grow until best fit (allow overfitting)

- Convert tree to equivalent set of rules
  - One rule per leaf node
  - Prune each rule independently of others
    - Remove various preconditions to improve performance
  - Sort final rules into desired sequence for use

Example of Rule post pruning

- IF (Outlook = Sunny) ^ (Humidity = High)
  – THEN PlayTennis = No
- IF (Outlook = Sunny) ^ (Humidity = Normal)
  – THEN PlayTennis = Yes



Thank You!