IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

Data Mining

Week 2: Classification, Decision Tree

Pabitra Mitra
Computer Science and Engineering

Data Mining

Classification and Decision Trees

Pabitra Mitra

Computer Science and Engineering

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.

Classification

- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

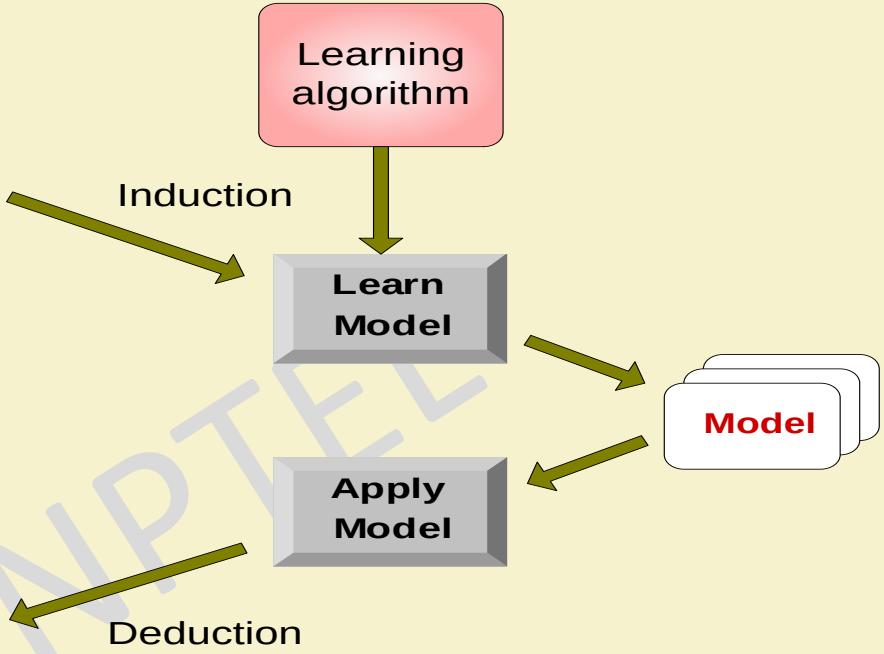
Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

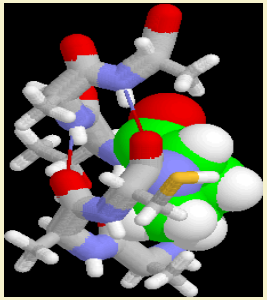
Test Set



Examples of Classification Task

- Classifying credit card transactions as legitimate or fraudulent

- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil



- Categorizing news stories as finance, weather, entertainment, sports, etc

Classification Techniques

- Decision Tree
- Bayes Classifier
- Nearest Neighbor
- Support Vector Machine
- Neural Networks
- Others….

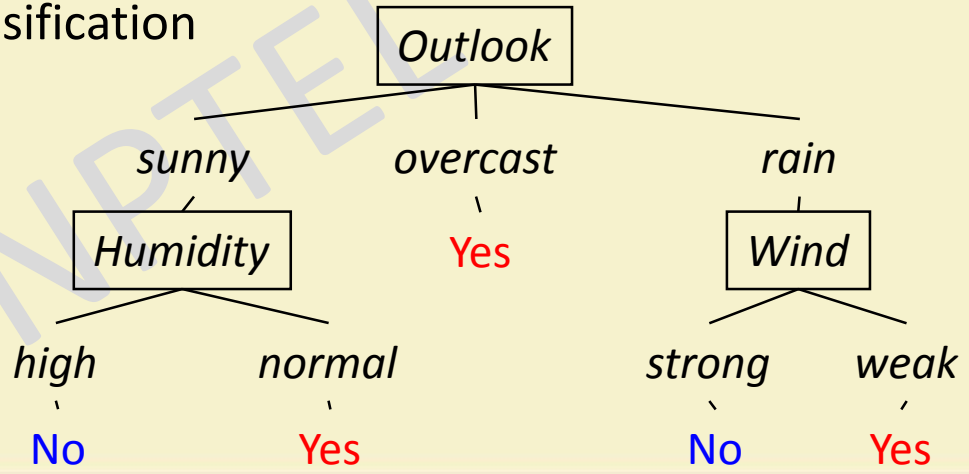
Training Examples

Day	Outlook	Temp	Humidity	Wind	Tennis?
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 tests an attribute
  - Each branch corresponds to attribute value
  - 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

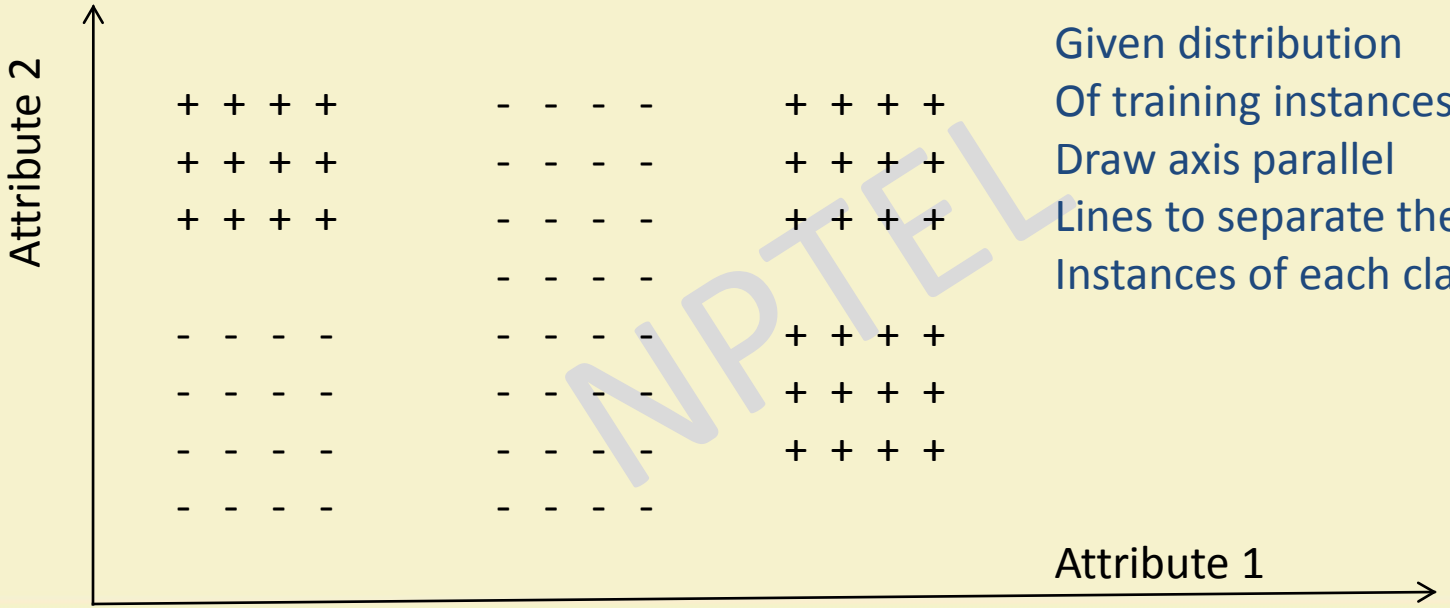
if outlook = rain AND wind = weak

then playtennis

Applications of Decision Trees

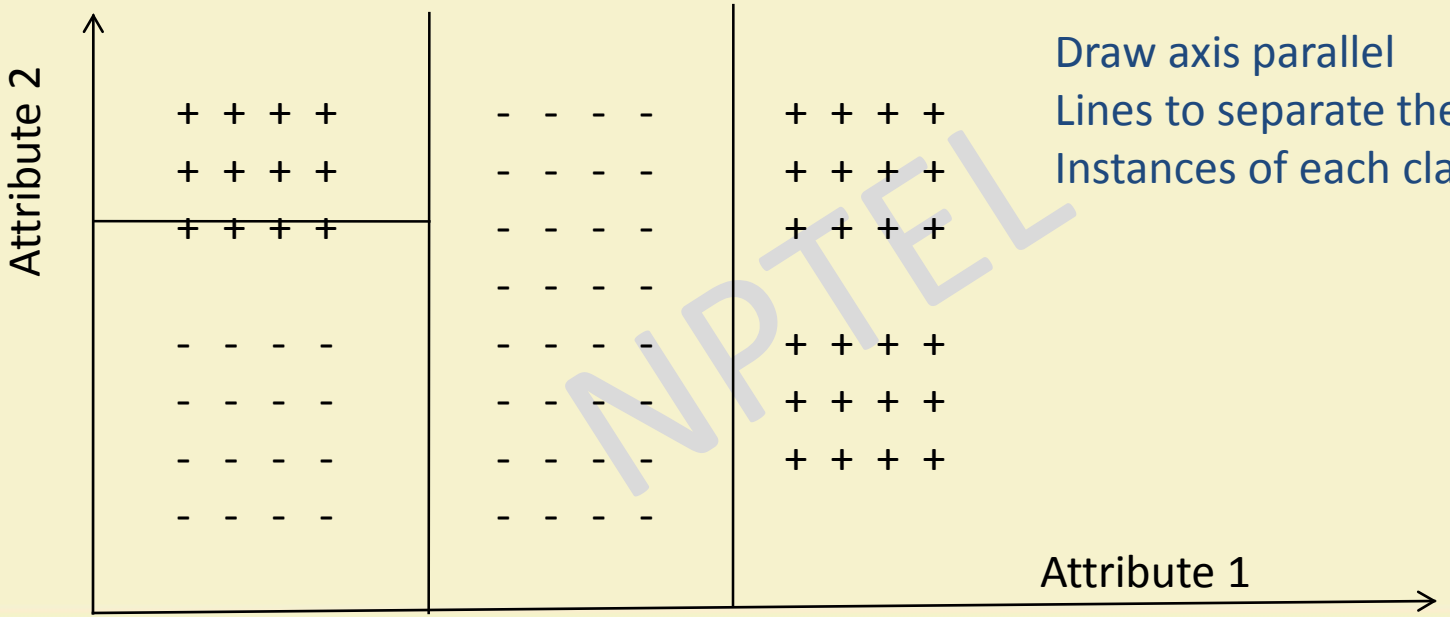
- Instances describable by a fixed set of attributes and their values
- Target function is discrete valued
  - 2-valued
  - N-valued
  - But can approximate continuous functions
- Disjunctive hypothesis space
- Possibly noisy training data
  - Errors, missing values, ...
- Examples:
  - Equipment or medical diagnosis
  - Credit risk analysis
  - Calendar scheduling preferences

Decision Trees



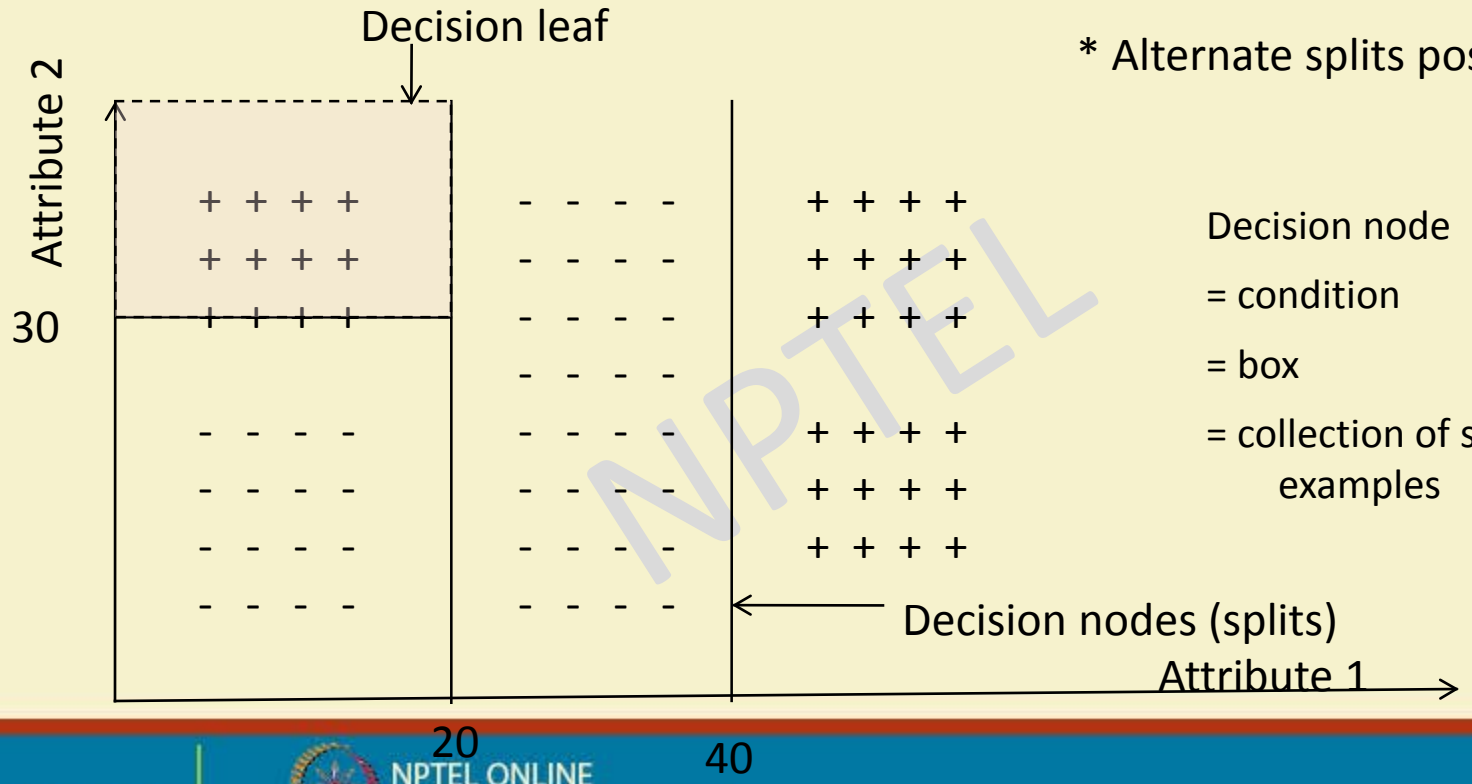
Given distribution
Of training instances
Draw axis parallel
Lines to separate the
Instances of each class

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

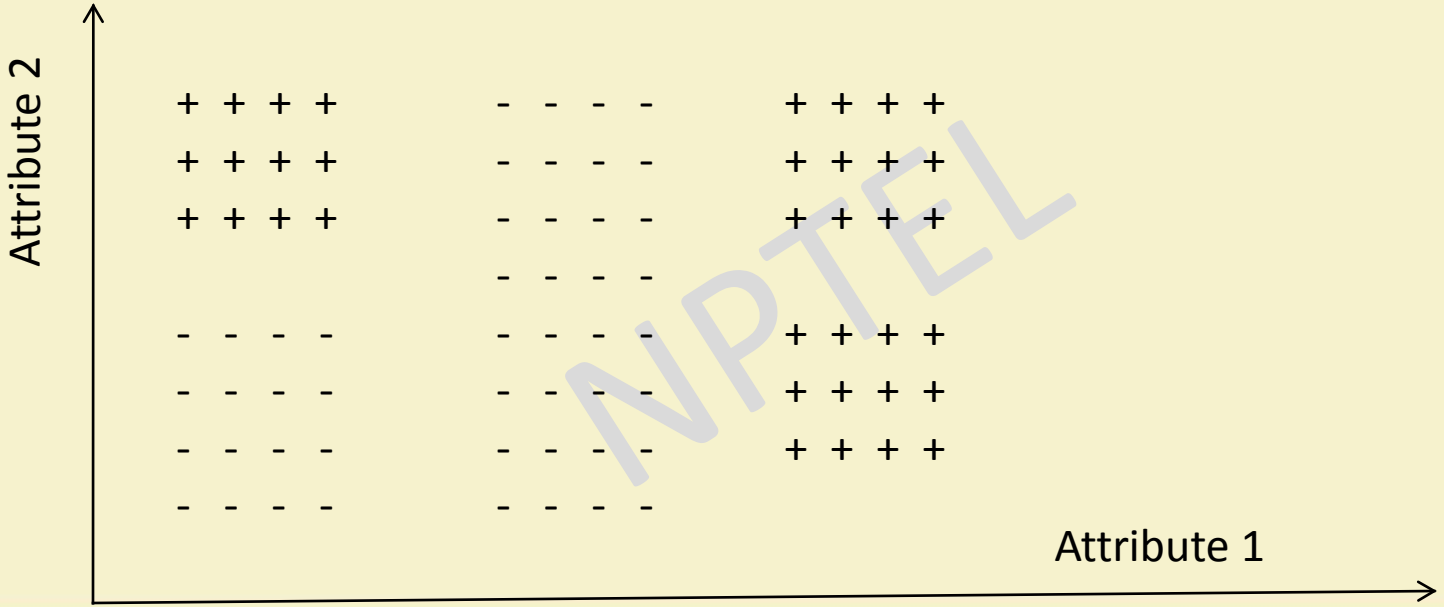
Decision Tree Construction

- Find the best structure
- Given a training data set

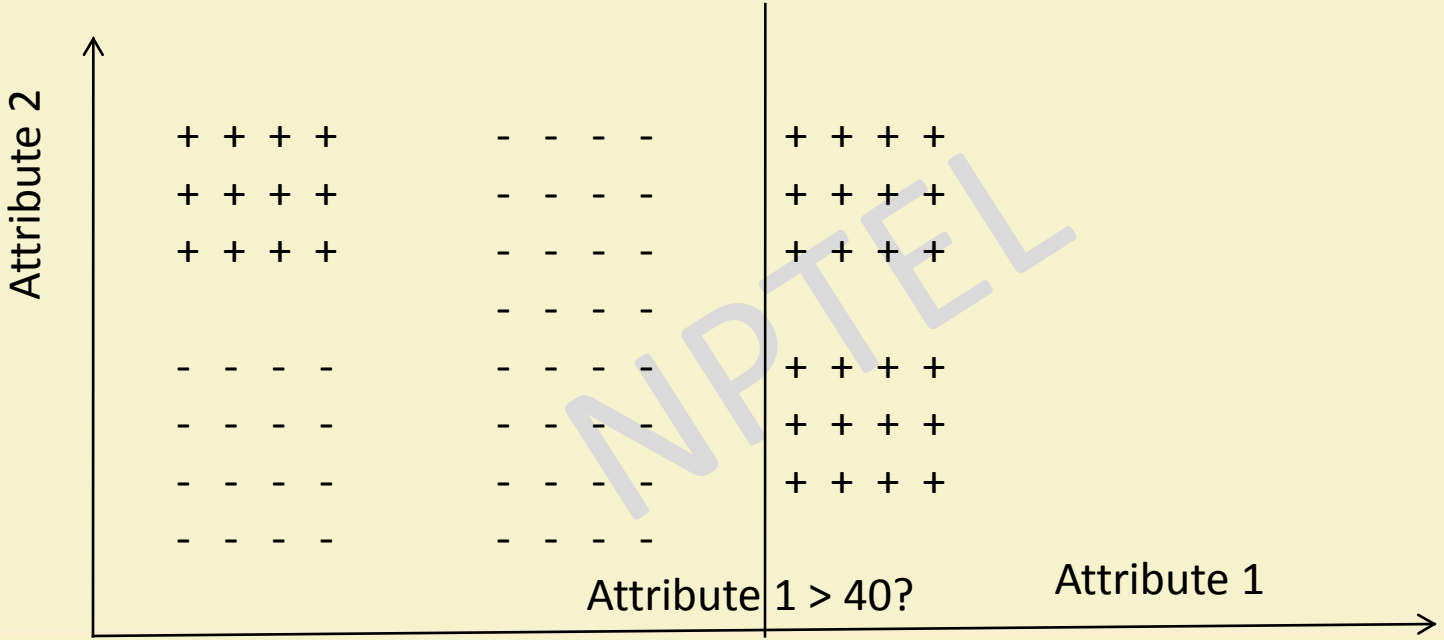
Top-Down Construction

- 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?

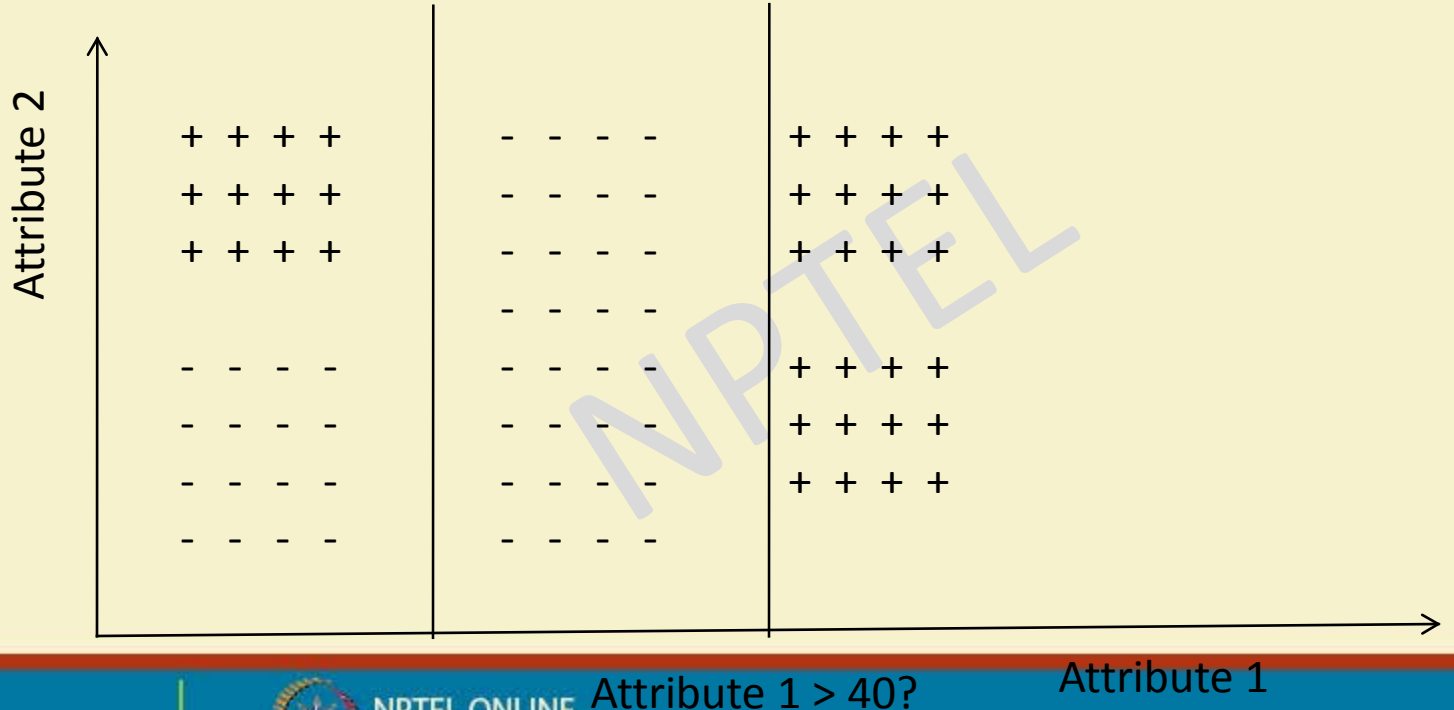
Best attribute to split?



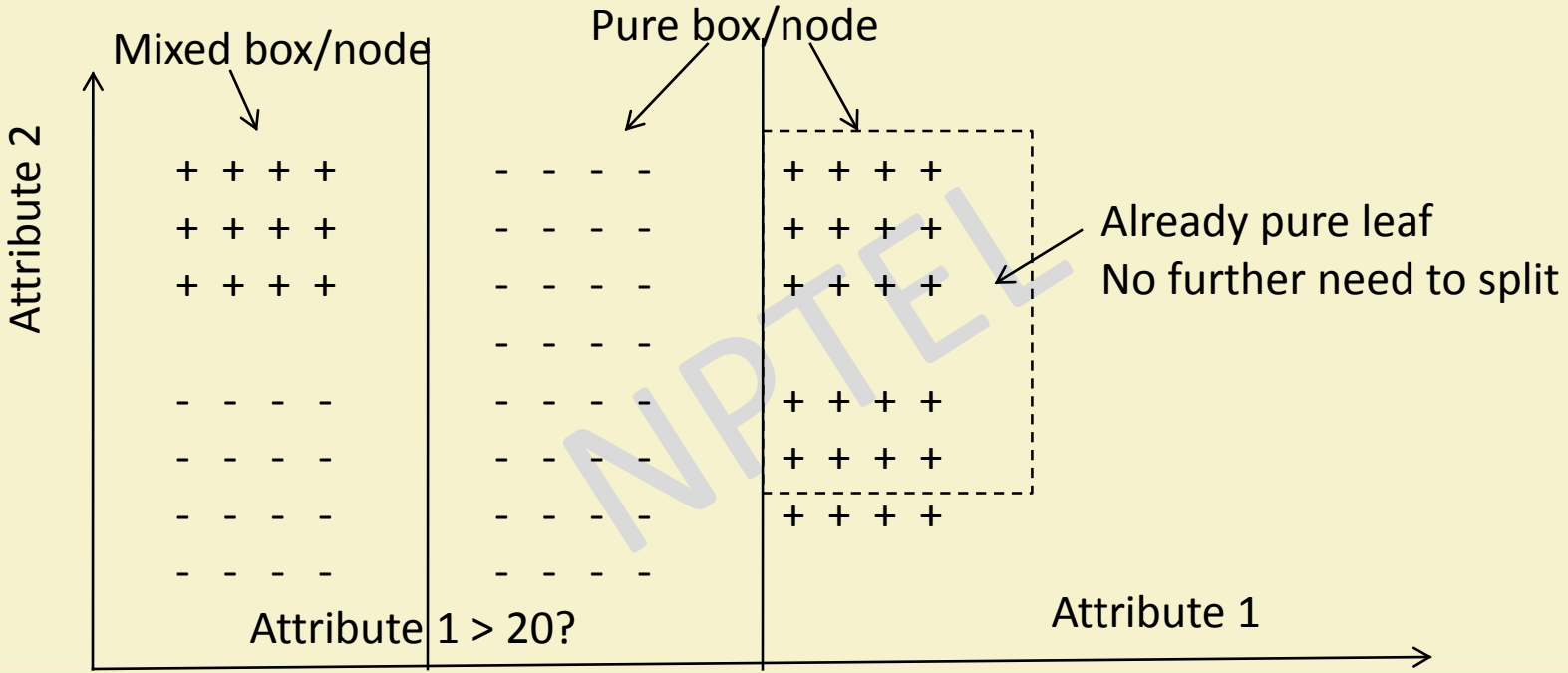
Best attribute to split?



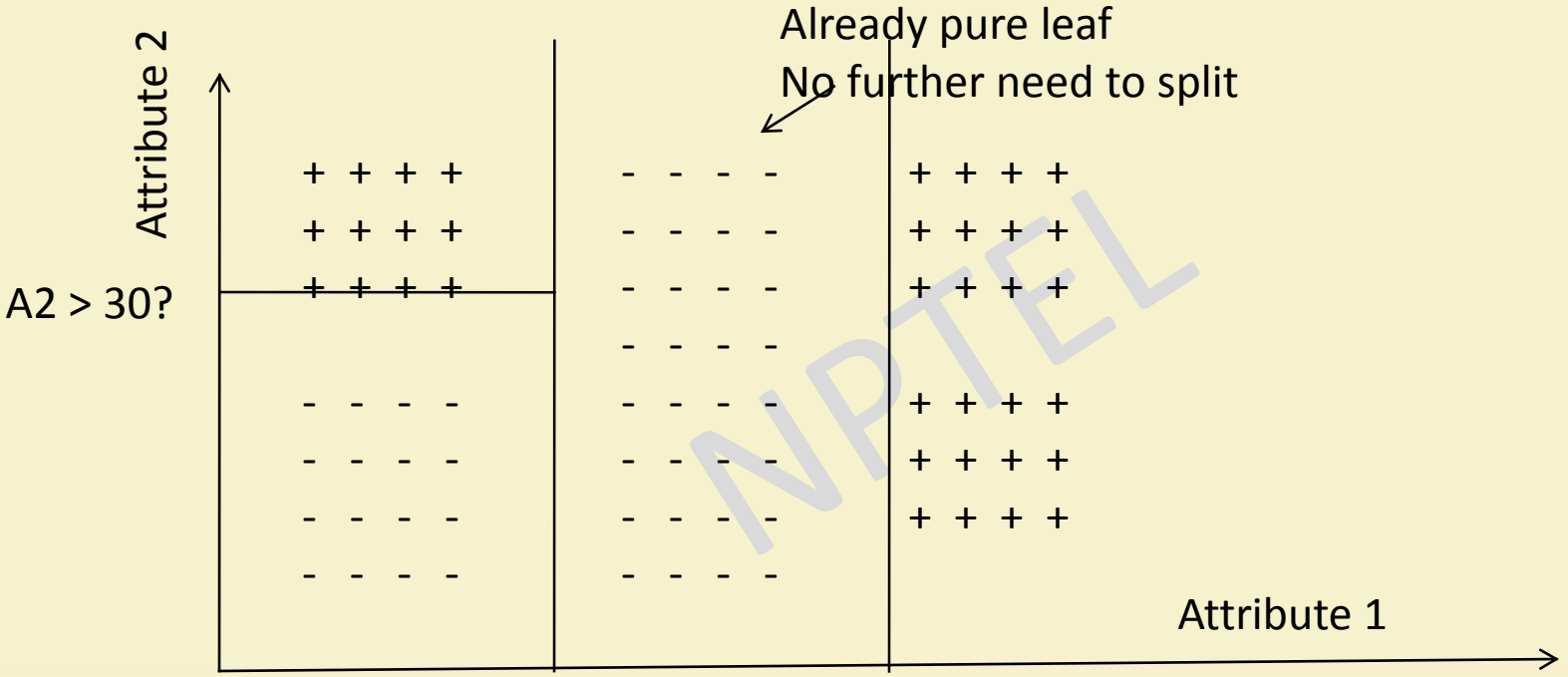
Best attribute to split?



Which split to make next?



Which split to make next?

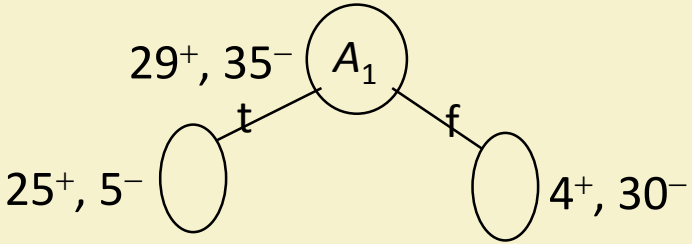


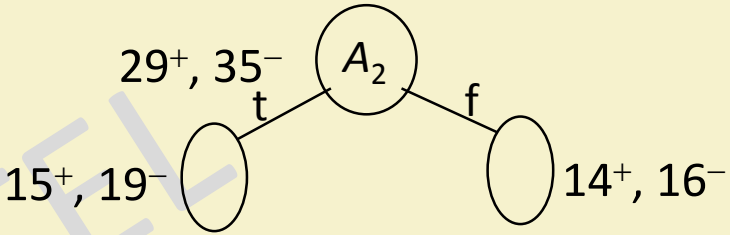
Principle of Decision Tree Construction

- 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

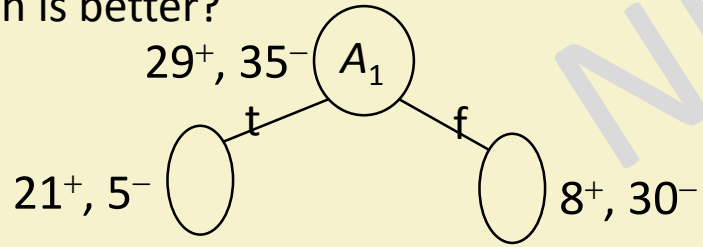
Choosing Best Attribute?

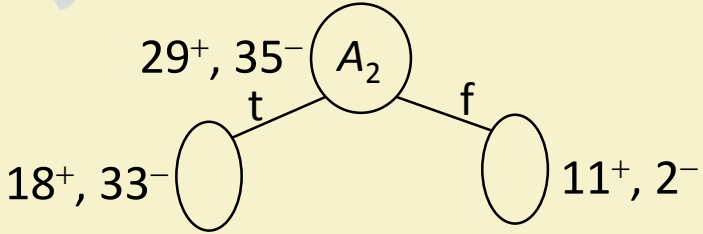
- Consider 64 examples, 29^+ and 35^-
- Which one is better?





- 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

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

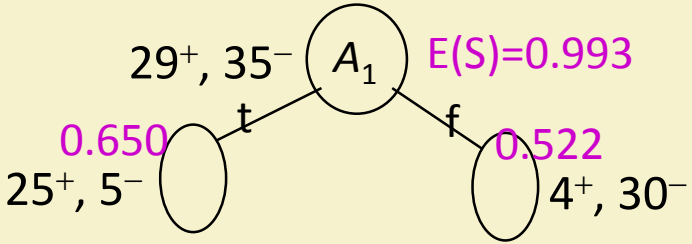
- Can be generalized to more than two values

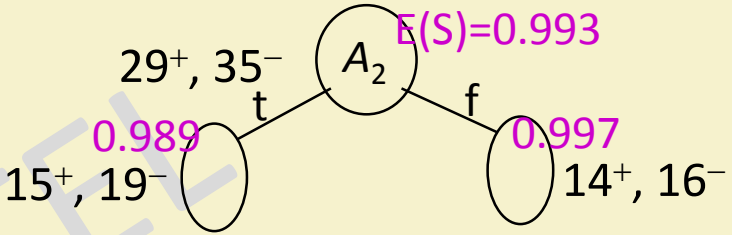
Entropy

- Entropy can also be viewed as measuring
  - purity of S,
  - uncertainty in S,
  - information in S, ...
- E.g.: values of entropy for p+=1, p+=0, p+=.5
- Easy generalization to more than binary values
  - Sum over pi *(-$\log_2$ pi) , i=1,n
    - i is + or – for binary
    - i varies from 1 to n in the general case

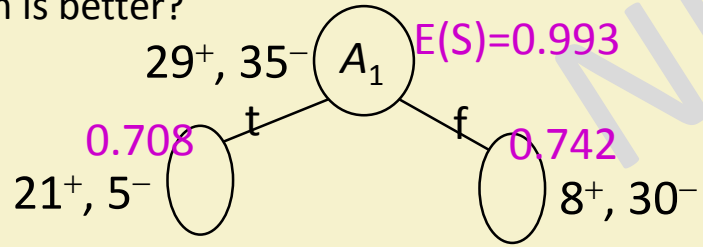
Choosing Best Attribute?

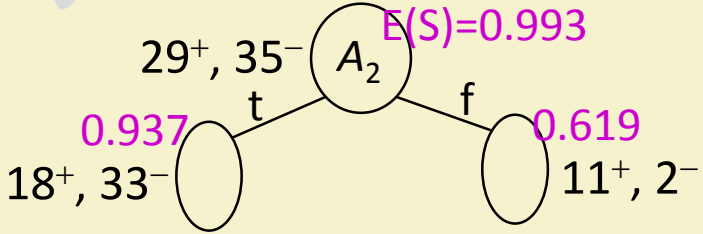
- Consider 64 examples ($29^+, 35^-$) and compute entropies:
- Which one is better?





- Which is better?

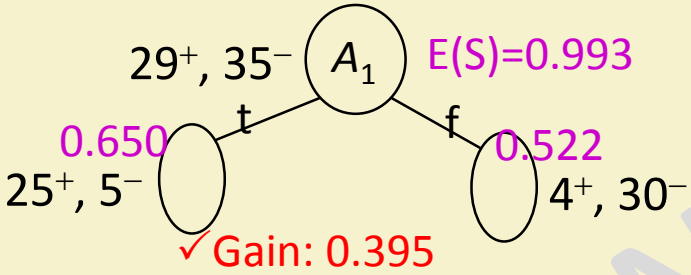


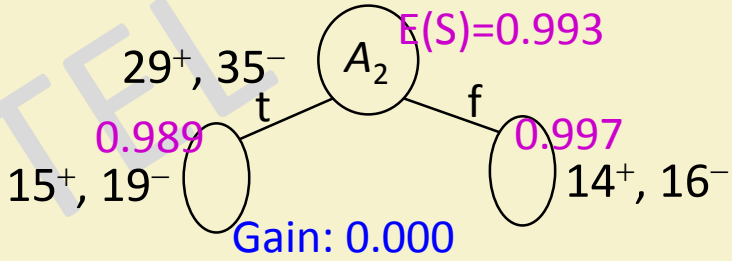


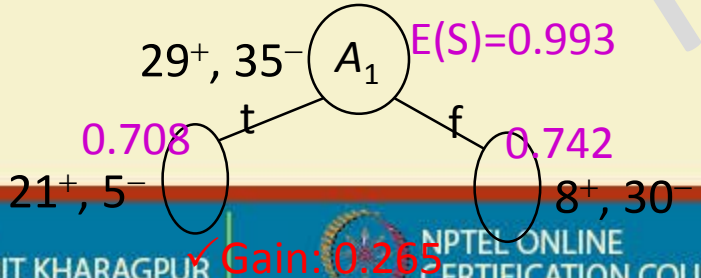
Information Gain

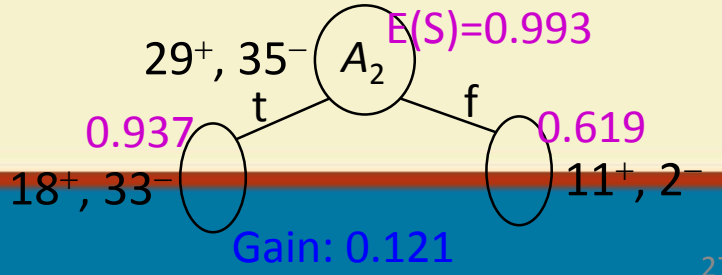
- $Gain(S,A)$: reduction in entropy after choosing attr. A

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









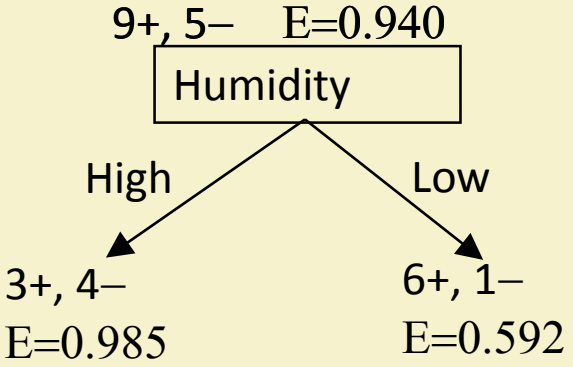
Gain function

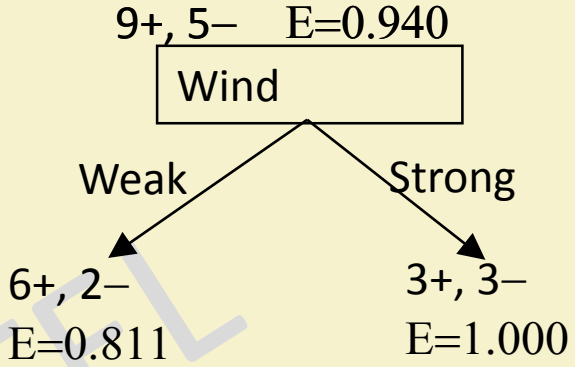
- Gain is measure of how much can
  - Reduce uncertainty
    - ❖ Value lies between 0,1
    - ❖ What is significance of
      - ➢ gain of 0?
        - example where have 50/50 split of +/- both before *and* after discriminating on attributes values
      - ➢ gain of 1?
        - Example of going from "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

Day	Outlook	Temp	Humidity	Wind	Tennis?
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





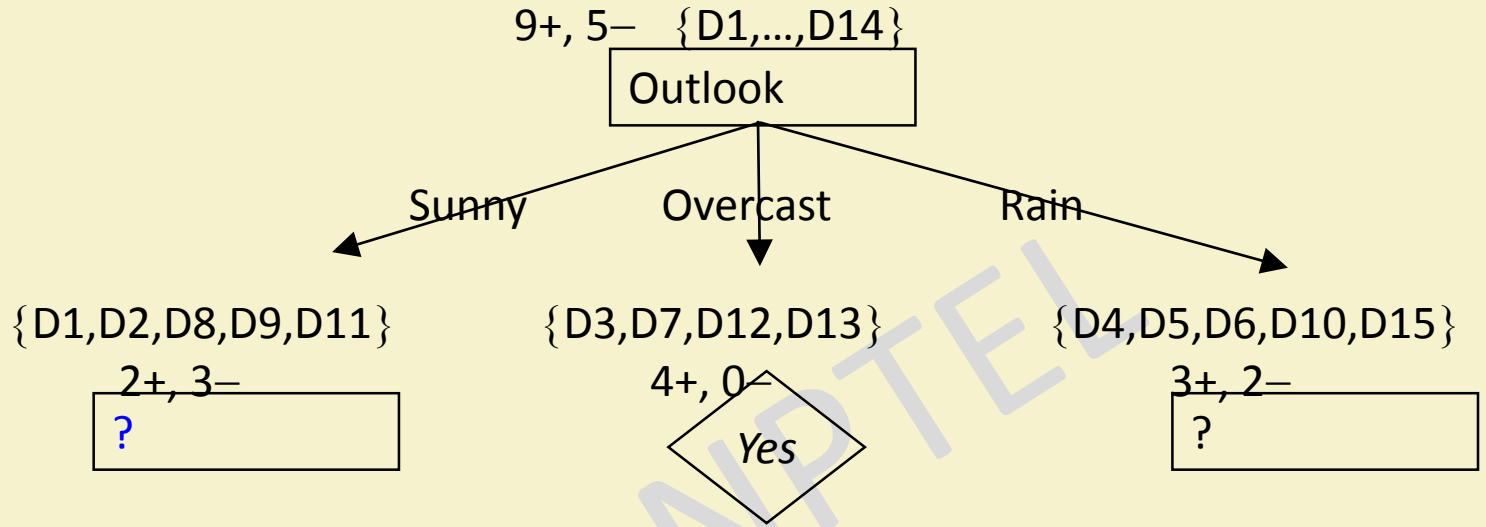
Gain (S, Humidity) = 0.151

Gain (S, Wind) = 0.048

Gain (S, Outlook) = 0.246

Gain (S, Temp) = 0.029

Sort the Training Examples



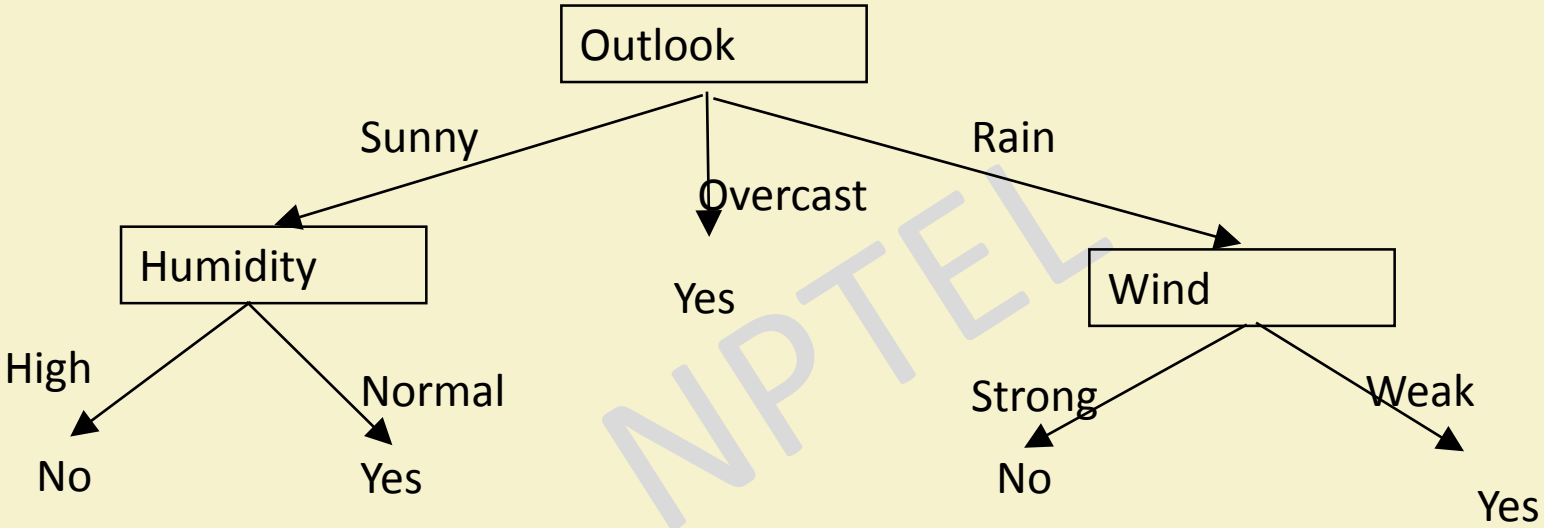
S_{sunny}= {D1,D2,D8,D9,D11}

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

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

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

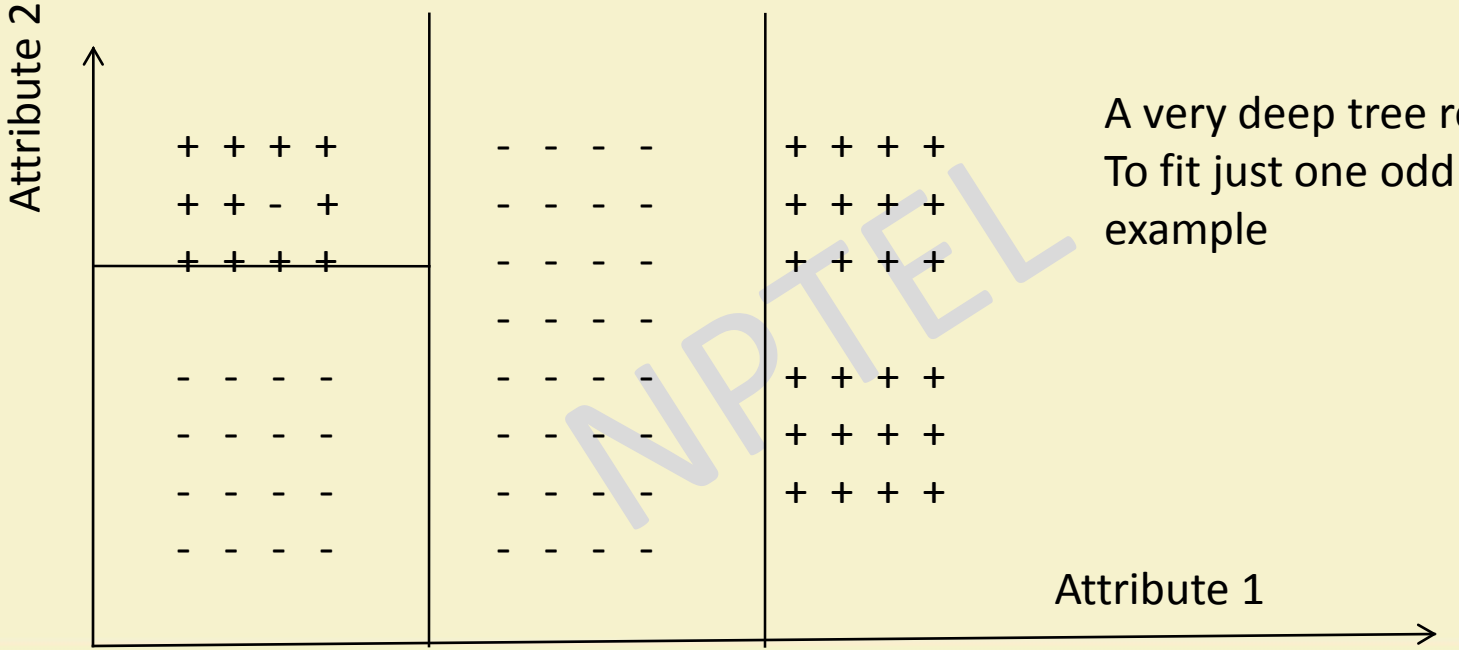
Final Decision Tree for Example



Overfitting the Data

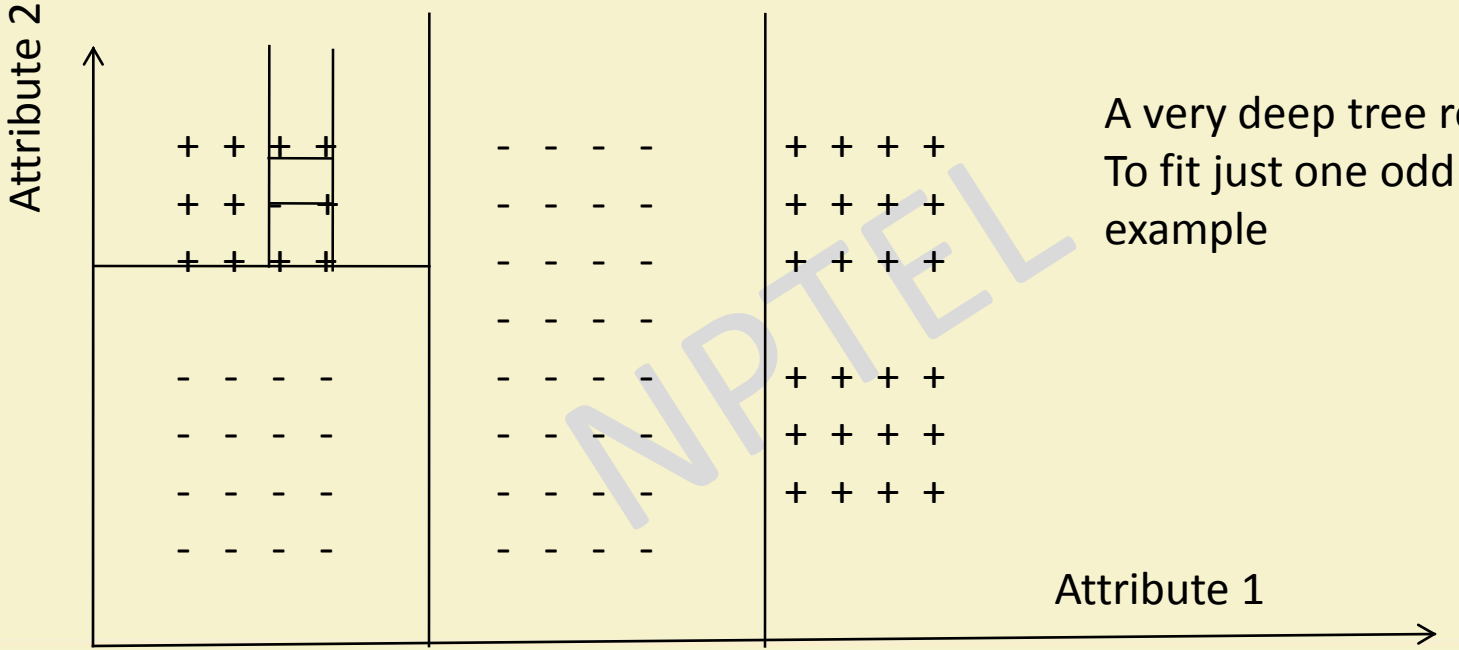
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data the tree is fitting
  - The algorithm might be making decisions based on very little data
- A hypothesis *h* is said to overfit 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

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 ?

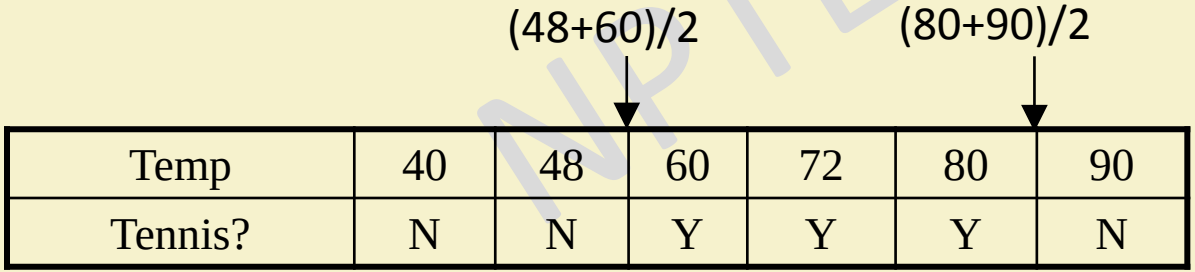
This is related to the notion of regularization that we will see in other contexts– keep the hypothesis simple.

Extensions of basic algorithm

- Continuous valued attributes
- Attributes with many values
- TE's with missing data
- Attributes with associated costs
- Other impurity measures
- Regression tree

Continuous Valued Attributes

- Create a discrete attribute from continuous variables
  - E.g., define critical Temperature = 82.5
- Candidate thresholds
  - chosen by gain function
  - can have more than one threshold
  - typically where values change quickly



Temp	40	48	60	72	80	90
Tennis?	N	N	Y	Y	Y	N

Unknown Attribute Values

- What if some examples are missing values of attribute *A*?
- Use training example anyway, sort through tree
  - if node *n* tests *A*, assign most common value of *A* among other examples sorted to node *n*
  - assign most common value of *A* among other examples with same target value
  - assign probability p_i to each possible value v_i of *A*
    - assign fraction p_i of example to each descendant in tree
- Classify test instances with missing values in same fashion
- Used in C4.5 – a popular decision tree software

Gini Index

- Another sensible measure of impurity (i and j are classes)

$$Gini = \sum_{i \neq j} p(i)p(j)$$

- After applying attribute A, the resulting Gini index is

$$Gini(A) = \sum_{v} p(v) \sum_{i \neq j} p(i|v)p(j|v)$$

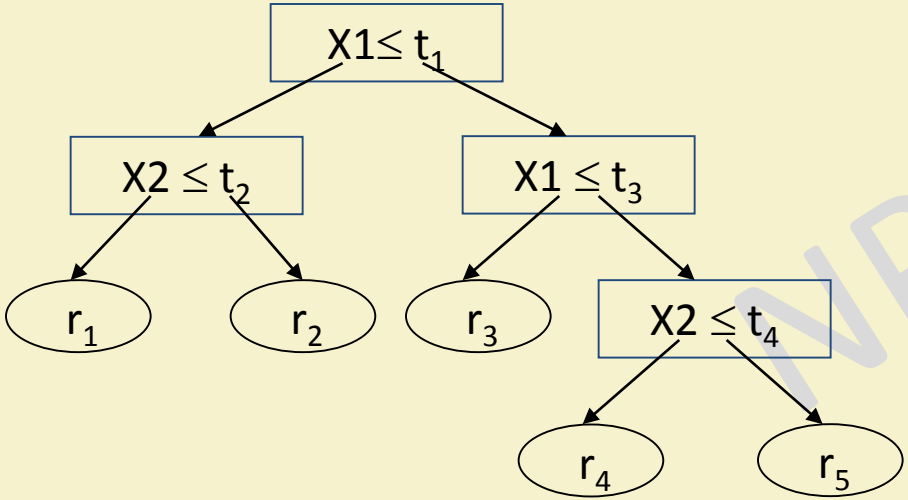
-

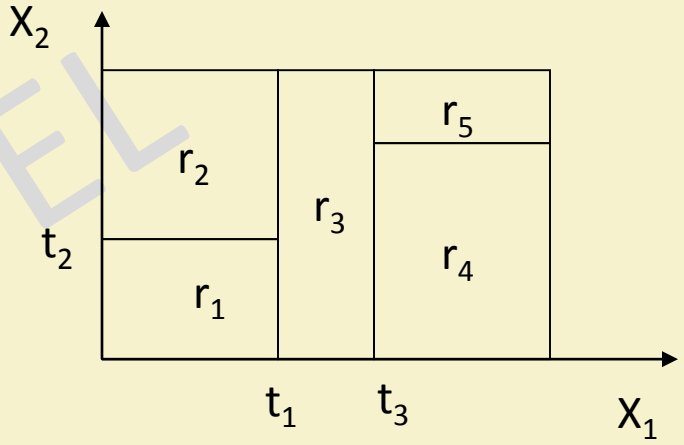
Regression Tree

- Similar to classification
- Use a set of attributes to predict the value (instead of a class label)
- Instead of computing information gain, compute the sum of squared errors
- Partition the attribute space into a set of rectangular subspaces, each with its own predictor
  - The simplest predictor is a constant value

Rectilinear Division

- A regression tree is a piecewise constant function of the input attributes





When Are Decision Trees Useful ?

- Advantages
  - Very fast: can handle very large datasets with many attributes
  - Flexible: several attribute types, classification and regression problems, missing values…
  - Interpretability: provide rules and attribute importance
- Disadvantages
  - Instability of the trees (high variance)
  - Not always competitive with other algorithms in terms of accuracy

Summary

- Decision trees are practical for concept learning
- Basic information measure and gain function for best first search of space of DTs
- ID3 procedure
  - search space is complete
  - Preference for shorter trees
- Overfitting is an important issue with various solutions
- Many variations and extensions possible

Software

- In R:
  - Packages tree and rpart
- C4.5:
  - http://www.cse.unwe.edu.au/~quinlan
- Weka
  - http://www.cs.waikato.ac.nz/ml/weka

End of Decision Tree