

Decision Tree

Machine Learning

Hamid R Rabiee – Zahra Dehghanian
Spring 2025




Sharif University
of Technology

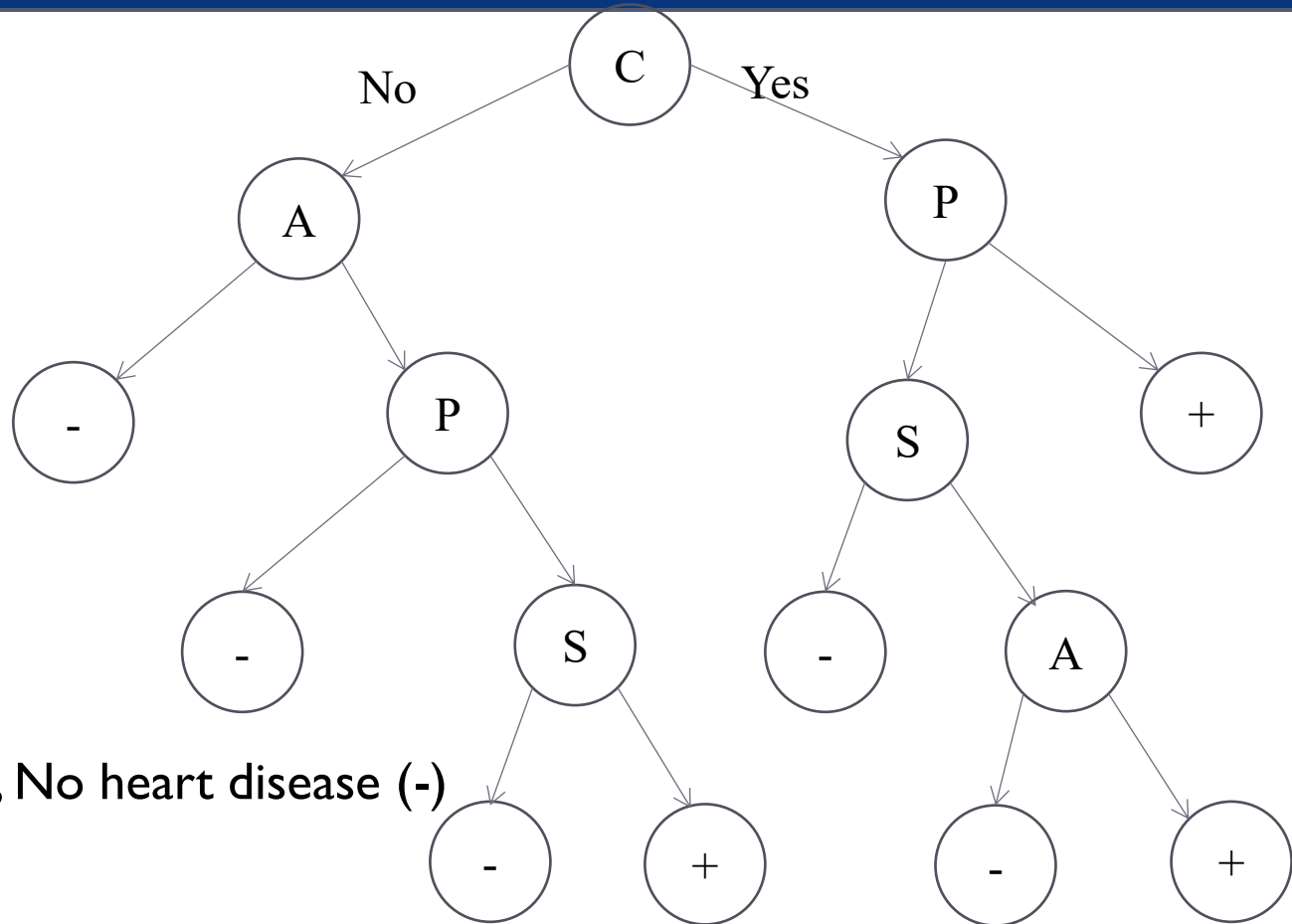
Decision tree

- One of the most intuitive classifiers that is easy to understand and construct
 - However, it also works very (very) well
- Categorical features are preferred. If feature values are continuous, they are discretized first.
- Application: Database mining

Example

- **Attributes:**
 - A: age>40
 - C: chest pain
 - S: smoking
 - P: physical test

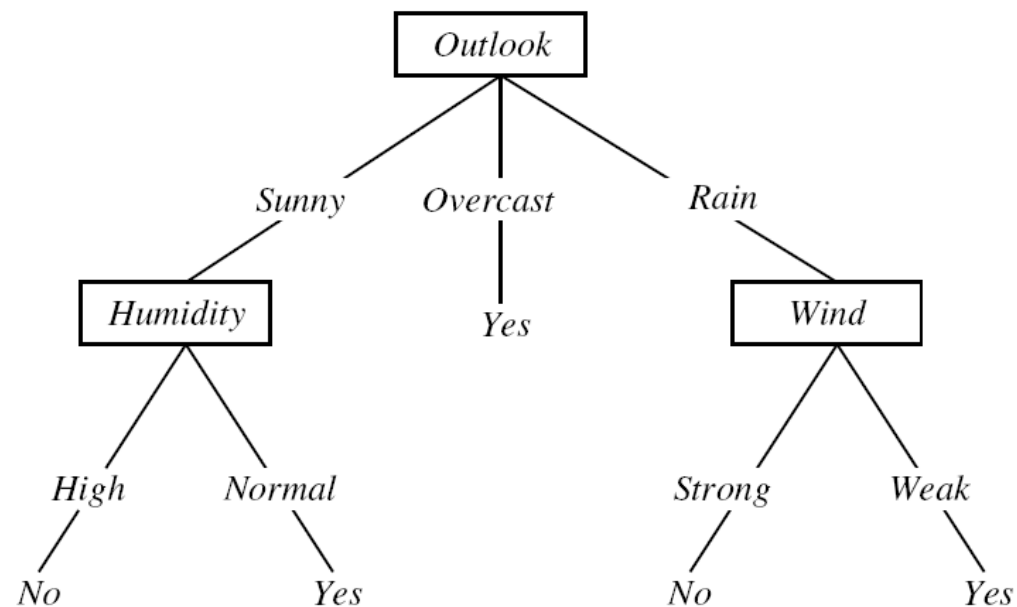
- Label: 
- Heart disease (+), No heart disease (-)



Decision tree: structure

- Leaves (terminal nodes) represent target variable
 - Each leaf represents a class label
- Each internal node denotes a test on an attribute
 - Edges to children for each of the possible values of that attribute

Day	Outlook	Temperature	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 tree: learning

- Decision tree learning: construction of a decision tree from training samples.
 - Decision trees used in data mining are usually classification trees
- There are many specific decision-tree learning algorithms, such as:
 - ID3
 - C4.5
- Approximates functions of usually discrete domain
 - The learned function is represented by a decision tree

Decision tree learning

- Learning an optimal decision tree is NP-Complete
 - Instead, we use a **greedy** search based on a heuristic
 - We cannot guarantee to return the globally-optimal decision tree.
- The most common strategy for DT learning is a greedy top-down approach
 - chooses a variable at each step that best splits the set of items.
- Tree is constructed by splitting samples into subsets based on an attribute value test in a recursive manner

How to construct basic decision tree?

- We prefer decisions leading to a simple, compact tree with few nodes
- Which attribute at the root?
 - Measure: how well the attributes split the set into homogeneous subsets (having same value of target)
 - Homogeneity of the target variable within the subsets.
- How to form descendant?
 - Descendant is created for each possible value of A
 - Training examples are sorted to descendant nodes

Constructing a decision tree

Function FindTree(S,A) \longrightarrow S: samples, A: attributes

If empty(A) or all labels of the samples in S are the same
status = leaf

class = most common class in the labels of S

else

status = internal

a \leftarrow **bestAttribute(S,A)**

LeftNode = FindTree(S(a=1), A \ {a})

RightNode = FindTree(S(a=0), A \ {a})

end

end

Recursive calls to create left and right subtrees
S(a=1) is the set of samples in S for which a=1

ID3

Function FindTree(S,A) // S: samples,A: attributes

If empty(A) or all labels of the samples in S are the same:

status = leaf

class = most common class in the labels of S

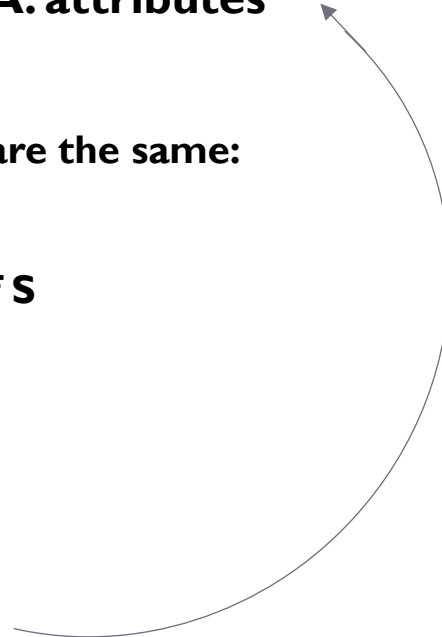
else:

status = internal

a = **bestAttribute(S,A)**

LeftNode = FindTree(S(a=1),A \ {a})

RightNode = FindTree(S(a=0),A \ {a})



ID3

ID3 (Examples, Target_Attribute, Attributes)

Create a root node for the tree

If all examples are positive, return the single-node tree Root, with label = +

If all examples are negative, return the single-node tree Root, with label = -

If number of predicting attributes is empty then

return Root, with label = most common value of the target attribute in the examples

else

A = The Attribute that best classifies examples.

Testing attribute for Root = A.

for each possible value, v_i , of A

Add a new tree branch below Root, corresponding to the test $A = v_i$.

Let Examples(v_i) be the subset of examples that have the value for A

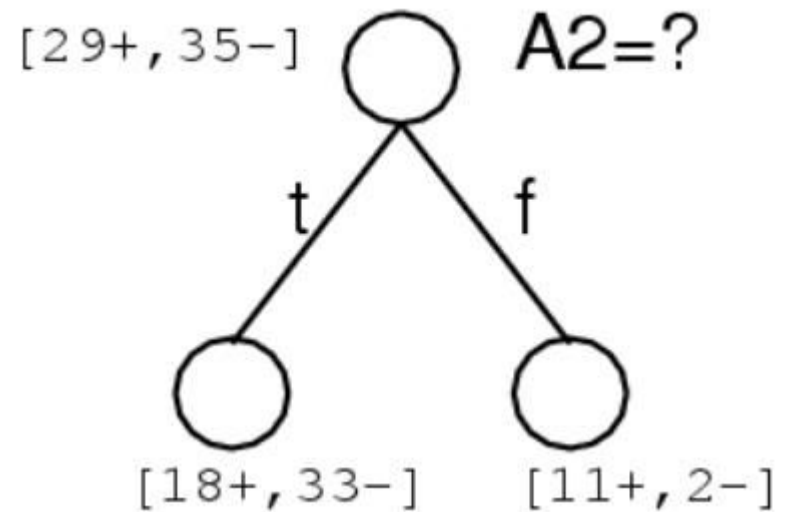
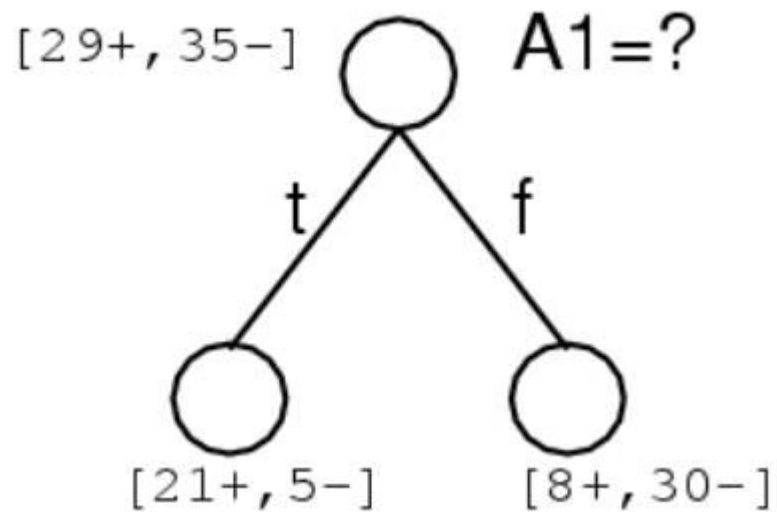
if Examples(v_i) is empty then

below this new branch add a leaf node with label = most common target value in the examples

else below this new branch add subtree **ID3 (Examples(v_i), Target_Attribute, Attributes – {A})**

return Root

Which attribute is the best?



Which attribute is the best?

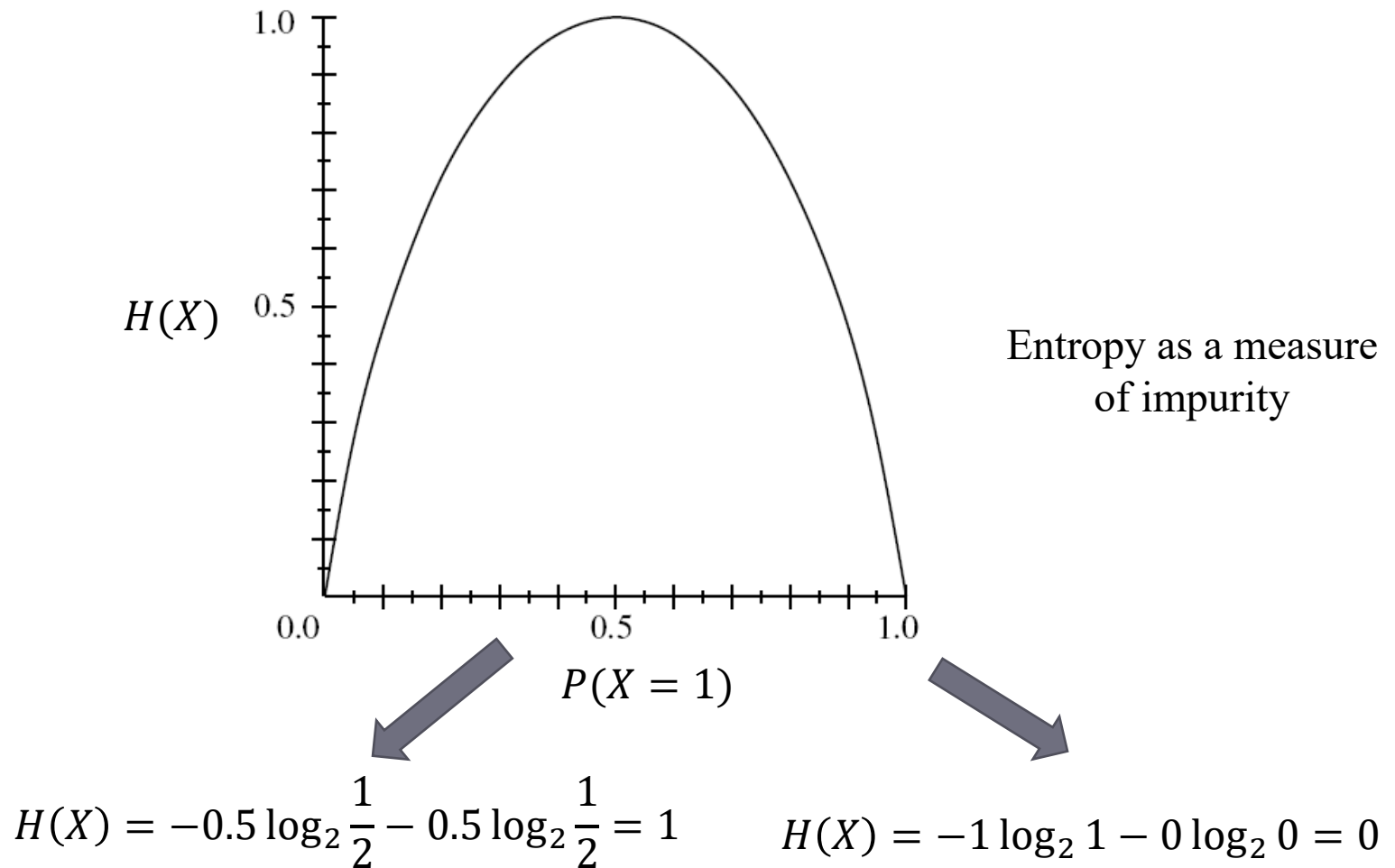
- A variety of heuristics for picking a good test
 - **Information gain**: originated with ID3 (Quinlan, 1979).
 - Gini impurity
 - ...
- These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

Entropy

$$H(X) = - \sum_{x_i \in X} P(x_i) \log P(x_i)$$

- Entropy measures the uncertainty in a specific distribution
- Information theory:
 - $H(X)$: expected number of bits needed to encode a randomly drawn value of X (under most efficient code)
 - Most efficient code assigns $-\log P(X = i)$ bits to encode $X = i$
 - \Rightarrow expected number of bits to code one random X is $H(X)$

Entropy for a Boolean variable



Information Gain (IG)

$$\text{Gain}(S, A) \equiv H_S(Y) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H_{S_v}(Y)$$

- A : variable used to split samples
- Y : target variable
- S : samples

Mutual Information

- The expected reduction in entropy of Y caused by knowing X :

$$I(X, Y) = H(Y) - H(Y|X)$$
$$= - \sum_i \sum_j P(X = i, Y = j) \log \frac{P(X = i)P(Y = j)}{P(X = i, Y = j)}$$

- Mutual information in decision tree:
 - $H(Y)$: Entropy of Y (i.e., labels) before splitting samples
 - $H(Y|X)$: Entropy of Y after splitting samples based on attribute X
 - It shows expectation of label entropy obtained in different splits (where splits are formed based on the value of attribute X)

Conditional entropy

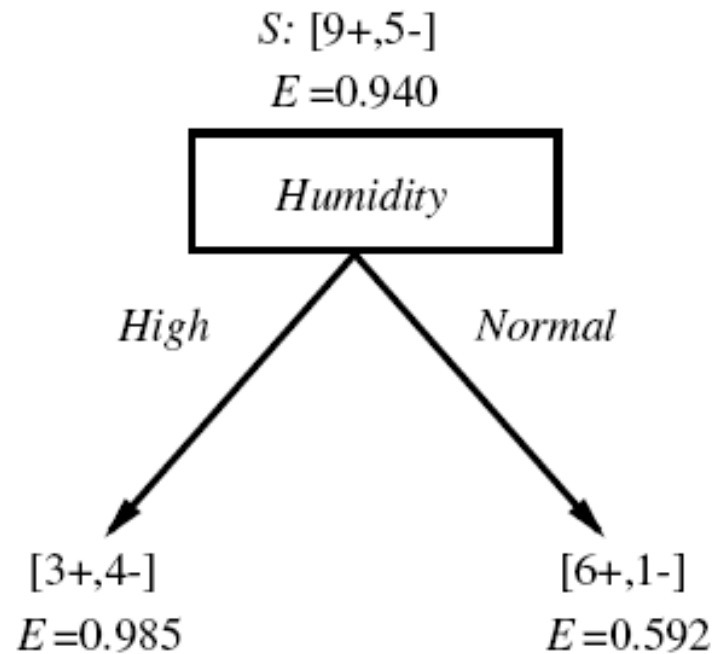
$$H(Y|X) = - \sum_i \sum_j P(X = i, Y = j) \log P(Y = j|X = i)$$

$$H(Y|X) = \sum_i \underbrace{P(X = i)}_{\text{probability of following i-th value for } X} \underbrace{\sum_j -P(Y = j|X = i) \log P(Y = j|X = i)}_{\text{Entropy of } Y \text{ for samples with } X = i}$$

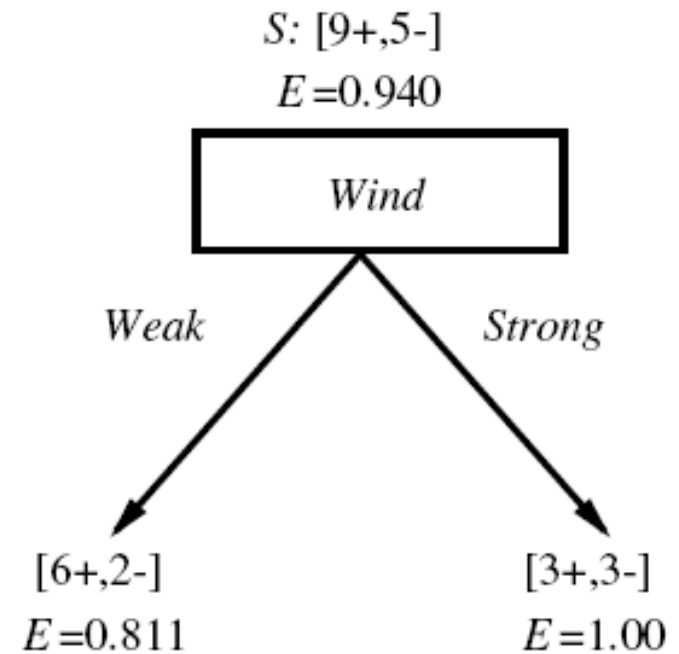
probability of following i-th value for X

Entropy of Y for samples with $X = i$

Information Gain: Example

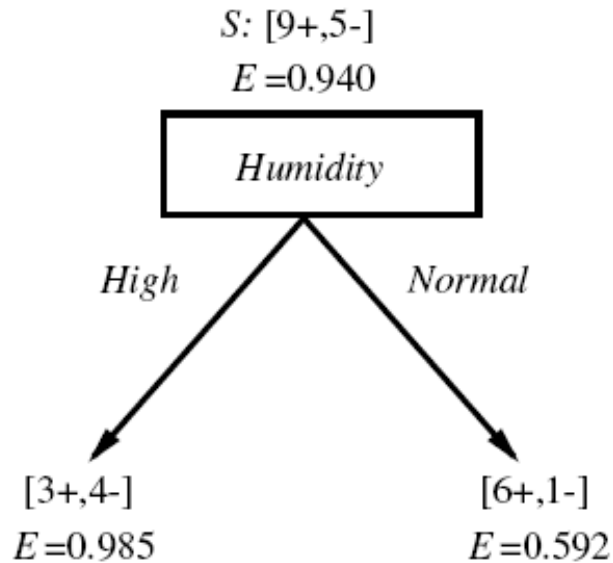


$$\begin{aligned}
 &H(Y|Humidity) \\
 &= \frac{7}{14} \times H(Y|Humidity = High) + \\
 &\quad \frac{7}{14} \times H(Y|Humidity = Normal) \\
 &= \frac{7}{14} \times 0.985 + \frac{7}{14} \times 0.592
 \end{aligned}$$

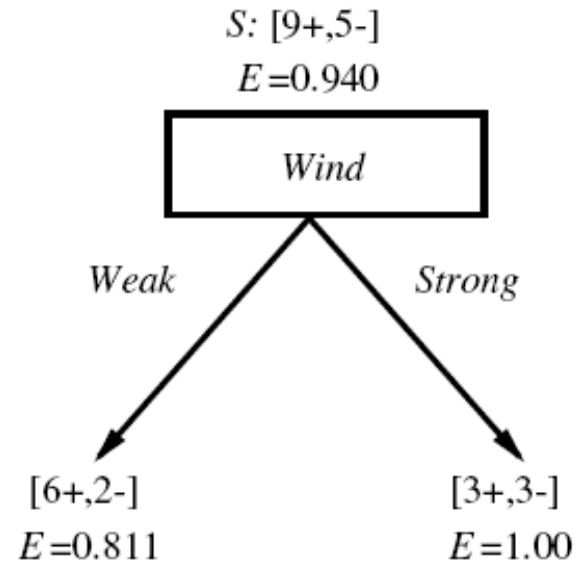


$$\begin{aligned}
 &H(Y|Wind) \\
 &= \frac{8}{14} \times H(Y|Wind = Weak) + \\
 &\quad \frac{6}{14} \times H(Y|Wind = Strong) = \frac{8}{14} \times 0.811 + \frac{6}{14} \times 1
 \end{aligned}$$

Conditional entropy: example



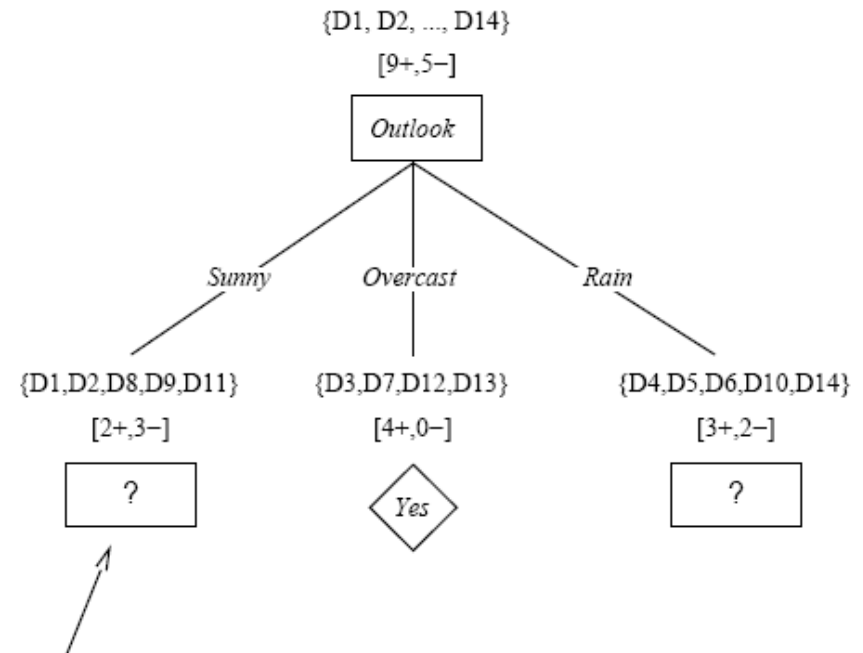
$$\begin{aligned} \text{Gain}(S, \text{Humidity}) &= .940 - (7/14).985 - (7/14).592 \\ &= .151 \end{aligned}$$



$$\begin{aligned} \text{Gain}(S, \text{Wind}) &= .940 - (8/14).811 - (6/14)1.0 \\ &= .048 \end{aligned}$$

Information Gain: Example

Day	Outlook	Temperature	Humidity	Wind	Play
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



Which attribute should be tested here?

$$S_{\text{Sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{Sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{Sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{Sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

How to find the best attribute?

- Information gain as our criteria for a good split
 - attribute that maximizes information gain is selected
- When a set of S samples have been sorted to a node, choose j -th attribute for test in this node where:

$$\begin{aligned} j &= \operatorname{argmax}_{i \in \text{remaining atts.}} \operatorname{Gain}(S, X_i) \\ &= \operatorname{argmax}_{i \in \text{remaining atts.}} H_S(Y) - H_S(Y|X_i) \\ &= \operatorname{argmin}_{i \in \text{remaining atts.}} H_S(Y|X_i) \end{aligned}$$

Other splitting criteria

- Information gain are biased in favor of those attributes with more levels.
 - More complex measures to select attribute
- Example: attribute Date
- Gain Ratio:

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{-\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log \frac{|S_v|}{|S|}}$$

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

(NOTE: $p(j | t)$ is the relative frequency of class j at node t).

- Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information
- Minimum (0) when all records belong to one class, implying most interesting information

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

Splitting Based on GINI

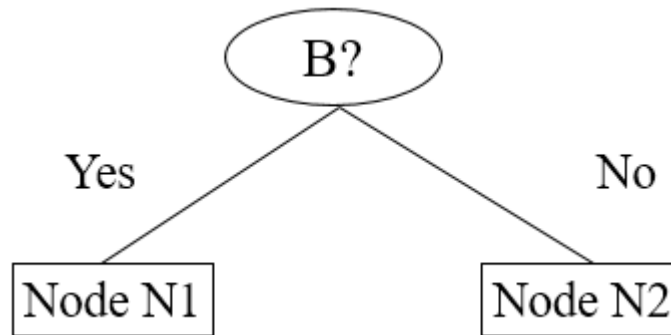
- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i ,
 n = number of records at node p .

Binary Attributes: Computing GINI Index

- Splits into two partitions



	Parent
C1	6
C2	6
Gini = 0.500	

$$\begin{aligned} \text{Gini}(N1) &= 1 - (5/7)^2 - (2/7)^2 \\ &= 0.194 \end{aligned}$$

$$\begin{aligned} \text{Gini}(N2) &= 1 - (1/5)^2 - (4/5)^2 \\ &= 0.528 \end{aligned}$$

	N1	N2
C1	5	1
C2	2	4
Gini=0.333		

Decision Tree

$$\begin{aligned} \text{Gini(Children)} &= 7/12 * 0.194 + \\ &\quad 5/12 * 0.528 \\ &= 0.333 \end{aligned}$$



ID3 algorithm: Properties

- The algorithm
 - either reaches homogenous nodes
 - or runs out of attributes
- Guaranteed to find a tree consistent with any conflict-free training set
 - ID3 hypothesis space of all DTs contains all discrete-valued functions
 - Conflict free training set: identical feature vectors always assigned the same class
- But not necessarily find the simplest tree (containing minimum number of nodes).
 - a greedy algorithm with locally-optimal decisions at each node (no backtrack).

Decision tree learning: Function approximation problem

- **Problem Setting:**

- Set of possible instances X
- Unknown target function $f: X \rightarrow Y$ (Y is discrete valued)
- Set of function hypotheses $H = \{ h \mid h : X \rightarrow Y \}$
 - h is a DT where tree sorts each x to a leaf which assigns a label y

- **Input:**

- Training examples $\{(x^{(i)}, y^{(i)})\}$ of unknown target function f

- **Output:**

- Hypothesis $h \in H$ that best approximates target function f

Decision tree hypothesis space

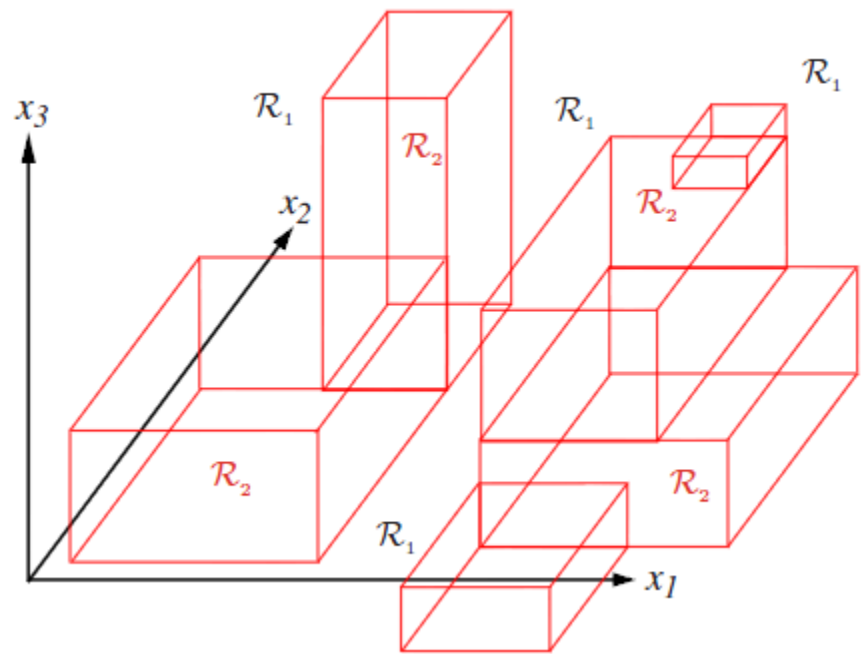
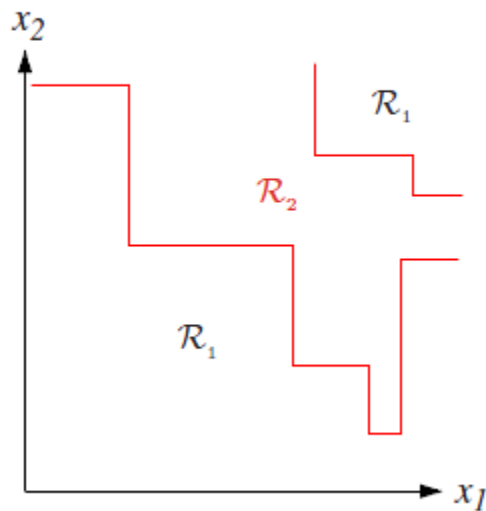
- Suppose attributes are Boolean
- Disjunction of conjunctions
- Which trees to show the following functions?
 - $y = x_1 \text{ and } x_5$
 - $y = x_1 \text{ or } x_4$
 - $y = (x_1 \text{ and } x_5) \text{ or } (x_2 \text{ and } \neg x_4) \text{ ?}$

Decision tree as a rule base

- Decision tree = a set of rules
- Disjunctions of conjunctions on the attribute values
 - Each path from root to a leaf = conjunction of attribute tests
 - All of the leafs with $y = i$ are considered to find the rule for $y = i$

How partition instance space?

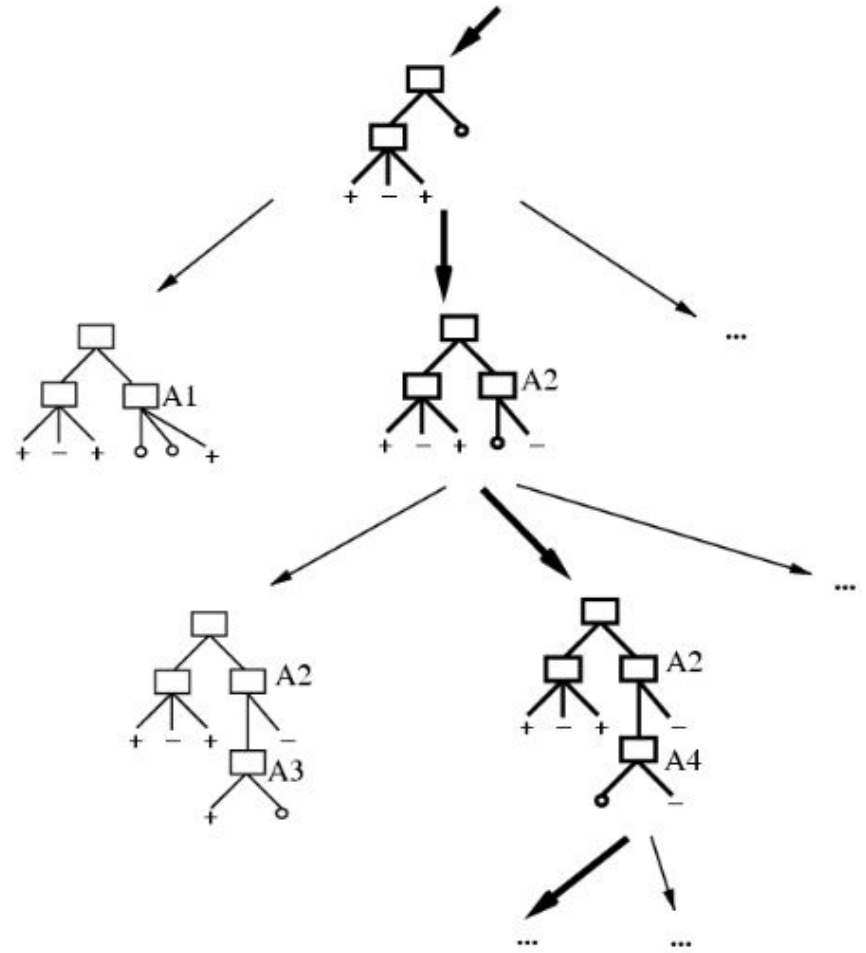
- Decision tree
 - Partition the instance space into axis-parallel regions, labeled with class value



[Duda & Hart 's Book]

ID3 as a search in the space of trees

- **ID3**: heuristic search through space of DTs
 - Performs a simple to complex hill-climbing search (begins with empty tree)
 - prefers simpler hypotheses due to using IG as a measure of selecting attribute test
- IG gives a bias for trees with minimal size.
 - ID3 implements a search (preference) bias instead of a restriction bias.



Why prefer short hypotheses?

- Why is the optimal solution the smallest tree?
- Fewer short hypotheses than long ones
 - a short hypothesis that fits the data is less likely to be a statistical coincidence
 - Lower variance of the smaller trees

Ockham (1285-1349) *Principle of Parsimony*:

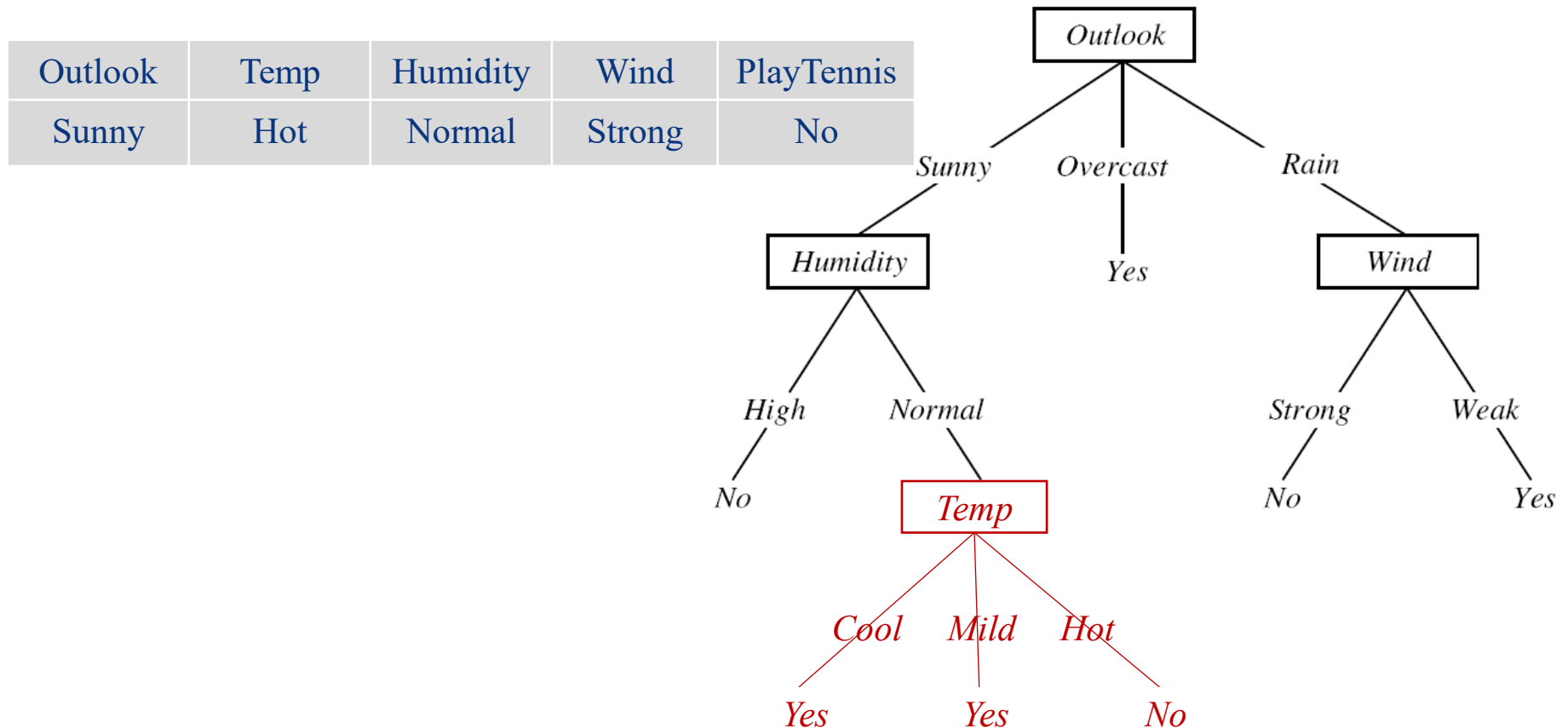
“One should not increase, beyond what is necessary, the number of entities required to explain anything.”

Over-fitting problem

- ID3 perfectly classifies training data (for consistent data)
 - It tries to memorize every training data
 - Poor decisions when very little data (it may not reflect reliable trends)
 - Noise in the training data: the tree is erroneously fitting.
 - A node that “should” be pure but had a single (or few) exception(s)?
- For many (non relevant) attributes, the algorithm will continue to split nodes
 - leads to over-fitting!

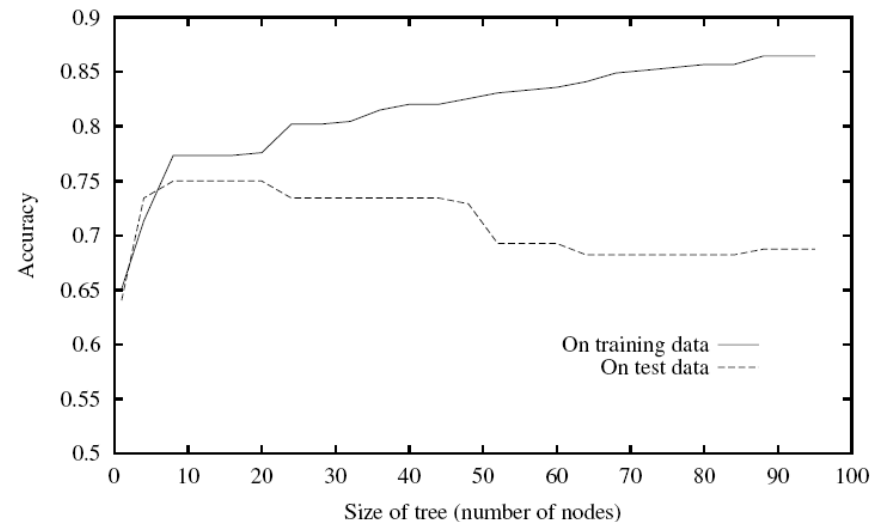
Over-fitting problem: an example

- Consider adding a (noisy) training example:



Over-fitting in decision tree learning

- Hypothesis space H : decision trees
- Training (empirical) error of $h \in H$: $error_{train}(h)$
- Expected error of $h \in H$: $error_{true}(h)$
- h overfits training data if there is a $h' \in H$ such that
 - $error_{train}(h) < error_{train}(h')$
 - $error_{true}(h) > error_{true}(h')$



A question?

- How can it be made smaller and simpler?
 - Early stopping
 - When should a node be declared as a leaf?
 - If a leaf node is impure, how should the category label be assigned?
 - Pruning?
 - Build a full tree and then post-process it

Avoiding overfitting

- **Stop growing** when the data split is not statistically significant.
- Grow full tree and then **prune** it.
 - More successful than stop growing in practice.
- How to select “best” tree:
 - Measure performance over separate validation set
 - MDL: minimize
 - $size(tree) + size(missclassifications(tree))$

Reduced-error pruning

- Split data into train and validation set
- Build tree using training set
- Do until further pruning is harmful:
 - Temporarily remove sub-tree rooted at node
 - Replace it with a leaf labeled with the current majority class at that node
 - Measure and record error on validation set

Produces smallest version of the most accurate sub-tree.

C4.5

- C4.5 is an extension of ID3
 - Learn the decision tree from samples (allows overfitting)
 - Convert the tree into the **equivalent set of rules**
 - Prune (generalize) each rule by removing any precondition that results in improving estimated accuracy
 - Sort the pruned rules by their estimated **accuracy**
 - consider them in sequence when classifying new instances
- Why converting the decision tree to rules before pruning?
 - Distinguishing among different contexts in which a decision node is used
 - Removes the distinction between attribute tests that occur near the root and those that occur near the leaves

Continuous attributes

- Tests on continuous variables as Boolean?
- Either use threshold to turn into binary or discretize
- It's possible to compute information gain for all possible thresholds (there are a finite number of training samples)

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

- Harder if we wish to assign more than two values (can be done recursively)

Classification

		Predicted Class	
		<i>Class</i> = 1	<i>Class</i> = 0
Actual Class	<i>Class</i> = 1	f_{11}	f_{10}
	<i>Class</i> = 0	f_{01}	f_{00}

Confusion Matrix

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Ranking classifiers

Top 8 are all based on various extensions of decision trees

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	.939	.963	.938	.929*	.880	.896	.896	.917
RF	PLT	.872*	.805	.934*	.957	.931	.930	.851	.858	.892	.898
BAG-DT	—	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	—	.872	.790	.934*	.957	.931	.930	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	.861	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	—	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	—	.834*	.816	.939	.963	.938	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	—	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	—	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	—	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774

[Rich Caruana & Alexandru Niculescu-Mizil, An Empirical Comparison of Supervised Learning Algorithms, ICML 2006]

Decision tree advantages

- Simple to understand and interpret
- Requires little data preparation and also can handle both numerical and categorical data
- Time efficiency of learning decision tree classifier
 - Can be used on large datasets
- Robust: Performs well even if its assumptions are somewhat violated

Reference

- T. Mitchell, “Machine Learning”, 1998. [Chapter 3]
- C. Bishop, “Pattern Recognition and Machine Learning”, Chapter 6.1-6.2, 7.1.
- Course CE-717, Dr. M.Soleymani