Machine Learning

Amirkabir University of Technology
(Tehran Polytechnic)

Lecture 3.
Supervised learning
Decision Trees

Alireza Rezvanian

Fall 2023

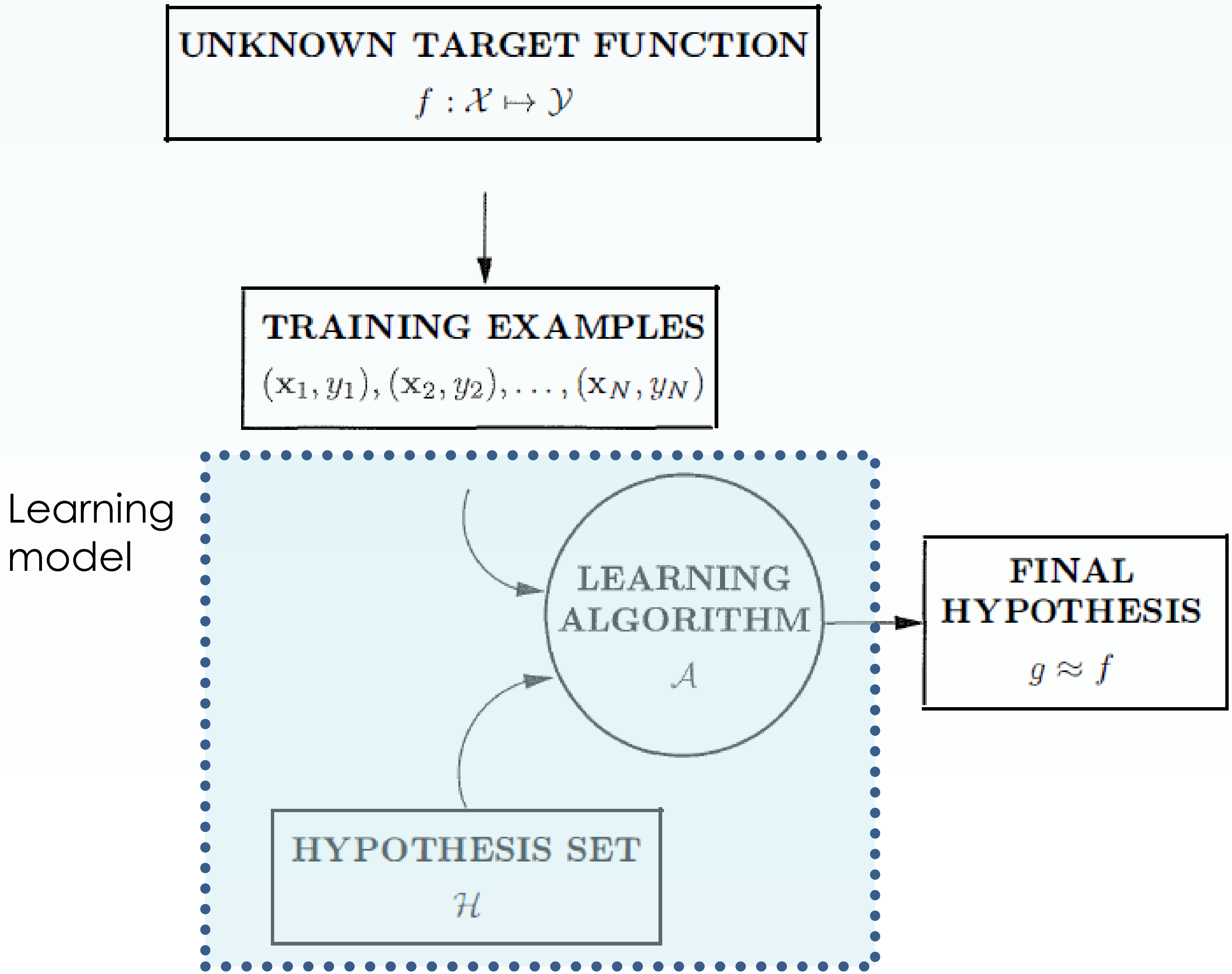
Last update: Oct. 29, 2022

Amirkabir University of Technology (Tehran Polytechnic)

Outline

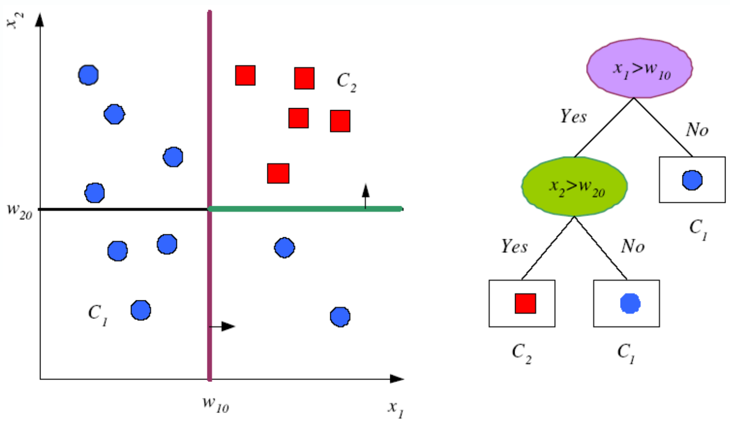
- Decision Trees
- Entropy
- Information Gain
- Learning Decision Trees
- ID3 algorithm
- C4.5
- Overfitting

Supervised Learning



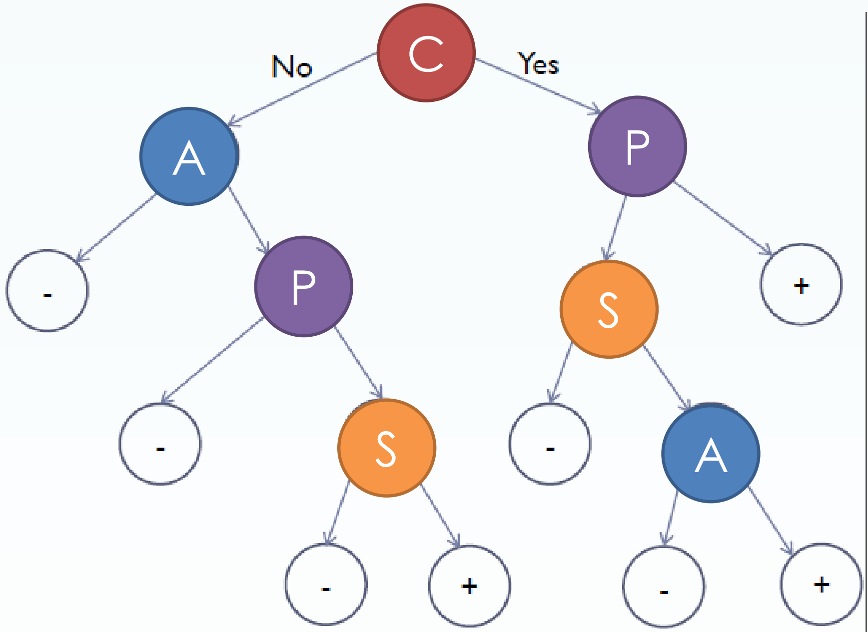
Decision Trees

- One of the most intuitive classifiers that is easy to understand and construct
  - However, it also works very well
- Categorical features are preferred. If feature values are continuous, they are discretized first.
- Synonyms of Decision Trees
  - Classification and Regression Trees (CART)
  - Algorithms for learning decision trees:
    - ID3
    - C4.5
  - Random Forests
    - Multiple decision trees



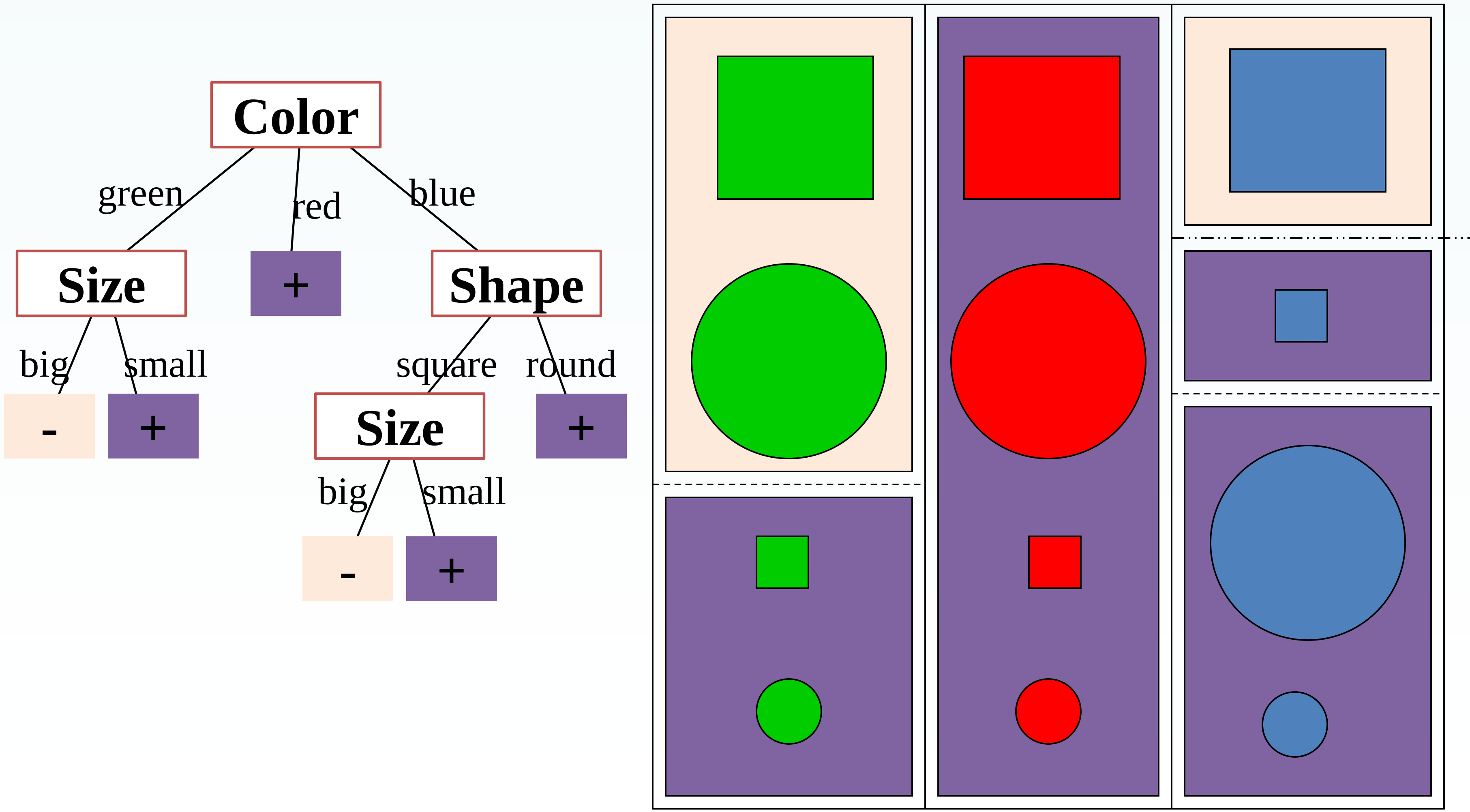
Example

- Each internal node denotes a test on an attribute
  - Attributes:
    - **A**: age>40
    - **C**: chest pain
    - **S**: smoking
    - **P**: physical test



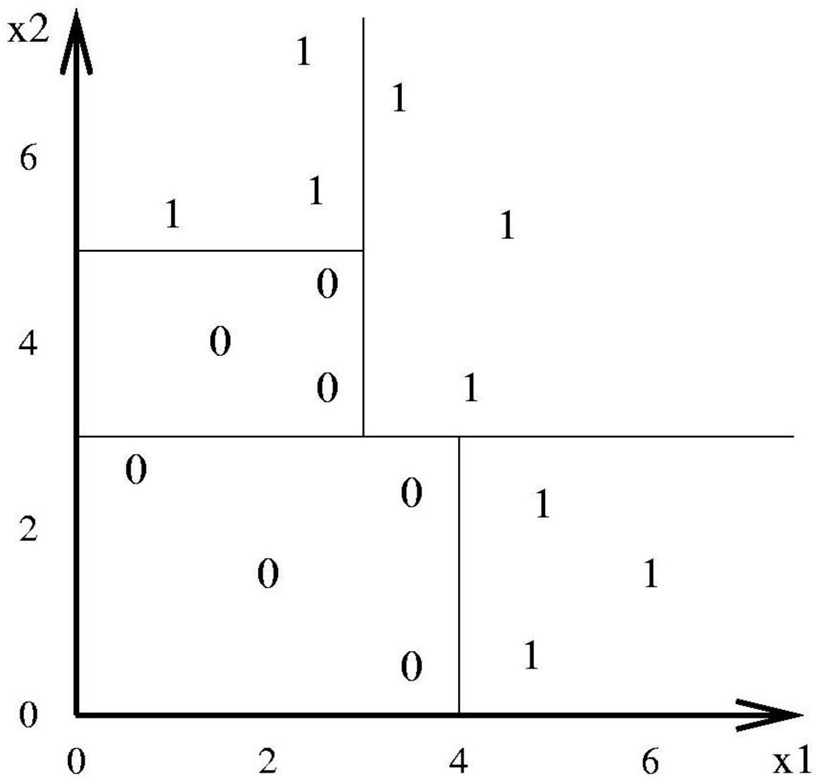
- Leaves (terminal nodes) represent target variable
  - Label
    - Heart disease (**+**), No heart disease (**-**)

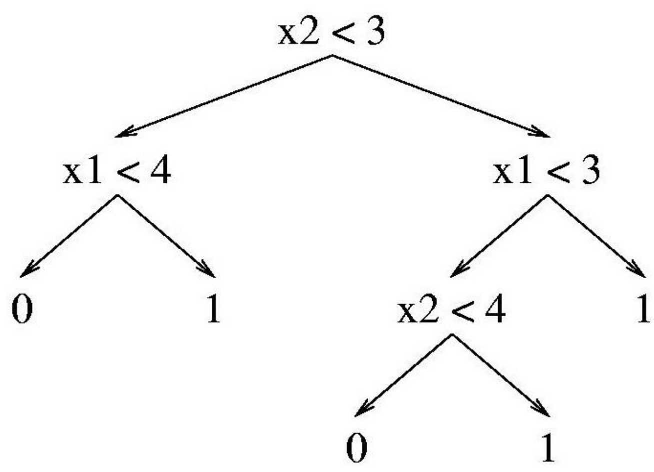
Decision tree-induced partition – example



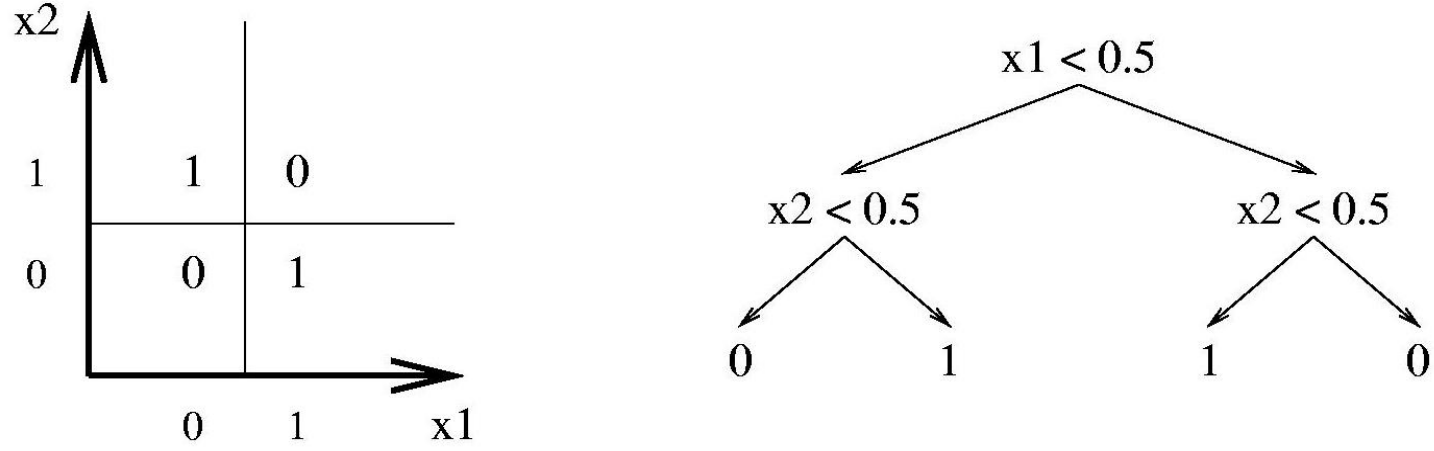
Decision tree decision boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision trees can represent any Boolean function



The tree will in the worst case require exponentially many nodes, however.

Comments

- Not all features/attributes need to appear in the tree.
- A features/attribute X_i may appear in multiple branches.
- On a path, no feature may appear more than once.
  - Not true for continuous features. We'll see later.
- Many trees can represent the same concept
- But, not all trees will have the same size!
  - e.g., $Y = (A \wedge B) \vee (\neg A \wedge C)$ (A and B) or (not A and C)

Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]

- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse
    - "Iterative Dichotomizer" (ID3)
    - C4.5 (ID3+improvements)
  - No guarantee to make the globally-optimal decision tree

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

Entropy and Information Gain

- A variety of heuristics for picking a good test
  - Information gain: originated with ID3 (Quinlan,1979).
  - Gini index
- 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

Bits

You are watching a set of independent random samples of X

You see that X has four possible values

P(X=A) = 1/4	P(X=B) = 1/4	P(X=C) = 1/4	P(X=D) = 1/4

So you might see: BAACBADCDADDDA…

You transmit data over a binary serial link. You can encode each reading with two bits (e.g. A = 00, B = 01, C = 10, D = 11)

0100001001001110110011111100…

Fewer Bits

Someone tells you that the probabilities are not equal

P(X=A) = 1/2	P(X=B) = 1/4	P(X=C) = 1/8	P(X=D) = 1/8

It's possible…

…to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

A	0
B	10
C	110
D	111

(This is just one of several ways)

General Case

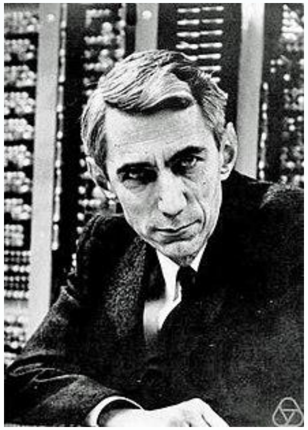
Suppose **X** can have one of ***m*** values… $V_1, V_2, \ldots V_m$

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	….	$P(X=V_m) = p_m$

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from X's distribution? It's

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m$$

$$= -\sum_{j=1}^{m} p_j \log_2 p_j$$



Claude Shannon

H(X) = The entropy of X

- "High Entropy" means X is from a uniform (boring) distribution
- "Low Entropy" means X is from varied (peaks and valleys) distribution

General Case

Suppose **X** can have one of ***m*** values… $V_1, V_2, \ldots V_m$

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	….	$P(X=V_m) = p_m$

What's the smallest possible num… average, per symbol, needed … of symb… distribu…

$H \ldots \log_2 \ldots$

$= -\sum \ldots \log_2 p \ldots$

H(X) = The entr…

A histogram of the frequency distribution of values of X would be flat

..and so the values sampled from it would be all over the place

A histogram of the frequency distribution of values of X would have many lows and one or two highs

..and so the values sampled from it would be more predictable

- "High Entropy" means X is from a uniform (boring) distribution
- "Low Entropy" means X is from varied (peaks and valleys) distribution

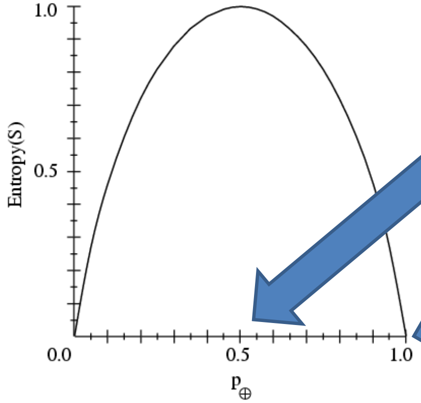
Entropy

Entropy ***H(Y)*** of a random variable ***Y***

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation: H(Y) is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)



$H(X) = -0.5\log(1/2) - 0.5\log(1/2) = 1$

$H(X) = -1\log(1) - 0\log(0) = 0$

Entropy

p(head)=0.5	p(head)=0.51	p(head)=?
p(tail)=0.5	p(tail)=0.49	p(tail)=?
H=1	H=0.9997	H=?

Learning Decision Trees

- A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output.
- If you are going to collect information from someone (e.g. asking questions sequentially in a decision tree), the "**best**" question is the one with the highest information gain
- To decide which attribute should be tested first, simply find the one with the highest information gain.
  - Then recurse …

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

What is Information Gain?

- **Information Gain (IG)** measures the expected reduction in entropy

$$IG(Y|X) = H(Y) - H(Y \mid X)$$

$$H(Y) = -\sum_i p(Y = v_i) \log_2 p(Y = v_i)$$

$$H(Y \mid X) = -\sum_j p(X = v_j) H(Y \mid X = v_j)$$

- Formally, IG named as Gain(S, F) for feature *F* with respect to collection of samples *S*

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

Constructing a decision tree

S: samples, A: attributes

```
Function FindTree(S,A)
  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})
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

ID3 (Examples,Target_Attribute, Attributes)

Create a root node for the tree

If all examples are positive, **then**

return the single-node tree Root, with label = +

If all examples are negative, **then**

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

Entropy of data set

- Dataset S has 9 positive, 5 negative examples
- Entropy of S is:

Entropy (9+, 5-) = $-(9/14)\log_2(9/14) - (5/14)\log_2(5/14) =$ 0.94

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*

Entropy is zero if all in S belong to the same class

Information gain for Outlook

Values(outlook)=Sunny, overcast, rainy

S=[9+, 5-]

S_{sunny}=[2+, 3-]

$S_{overcast}$=[4+, 0-]

S_{rainy}=[3+, 2-]

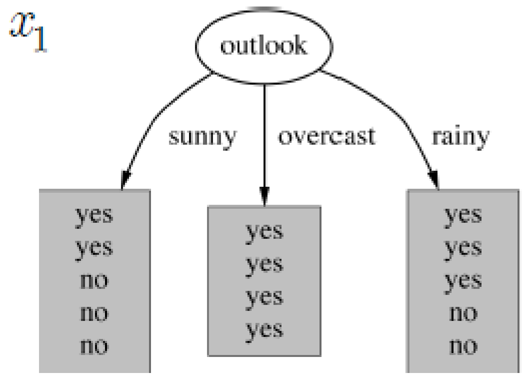
$$Gain(S, outlook) = Entropy(S) - \sum_{v\in[sunny,overcast,rainy]} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - (\frac{5}{14})Entropy(S_{sunny}) - (\frac{4}{14})Entropy(S_{overcast}) - \left(\frac{5}{14}\right) Entropy\left(S_{rainy}\right)$$

$$= 0.94 - (\frac{5}{14})0.97 - \left(\frac{4}{14}\right)0 - \left(\frac{5}{14}\right)0.97$$

$$= 0.693$$

Information gain for attribute x_1 (Outlook)



Info[2,3]=0.97 bits

Info[3,2]=0.97 bits

Info[4,0]=0 bits

For first value (*Sunny*) there are 2 positive and 3 negative examples

Info[2,3]=entropy(2/5, 3/5)

$= -2/5 \log_2 2/5 - 3/5 \log_2 3/5$

= 0.97 bits

Average info of subtree(weighted)=

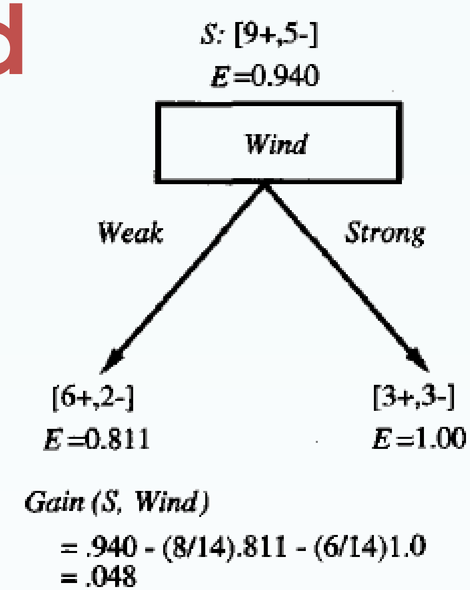
$0.97 \times 5/14 +$

$0 \times 4/14 +$

$0.97 \times 5/14 = 0.693$ bits

- Info of all training samples , info[9,5] = 0.94
- *IG*(*S*, *Outlook*) = 0.94 - 0.693 = 0.247 bits

Information gain for wind



Values(wind)=weak, strong

S=[9+, 5-]

S_{weak}=[6+, 2-]

S_{strong}=[3+, 3-]

$$Gain(S, wind) = Entropy(S) - \sum_{v \in [weak, strong]} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - (\frac{8}{14}) Entropy(S_{weak}) - (\frac{6}{14}) Entropy(S_{strong})$$

$$= 0.94 - (\frac{8}{14}) 0.0811 - \left(\frac{6}{14}\right) 1.000$$

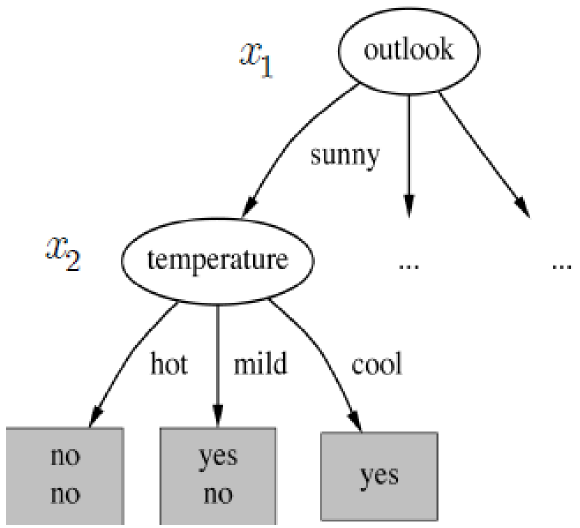
= 0.048

Information gain for each attribute

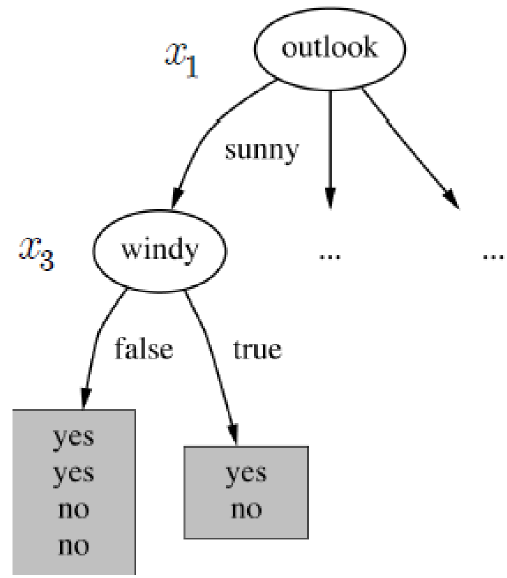
- IG(S, Outlook) = 0.94 – 0.693 = 0.247
- IG(S, Temperature)= 0.94 – 0.911 = 0.029
- IG(S, Humidity)= 0.94 – 0.788 = 0.152
- IG(S, Windy)= 0.94 – 0.892 = 0.048
- arg max {0.247, 0.029, 0.152, 0.048} = *Outlook*
- Select *Outlook* as the splitting attribute of tree

Expanded Tree Stumps for x_1

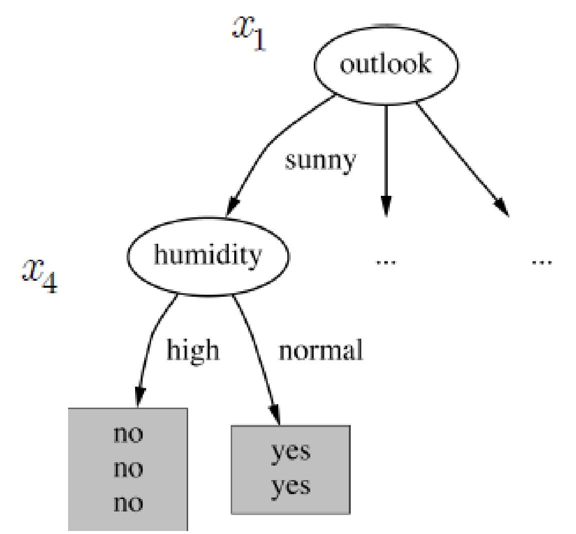
Outlook=Sunny



Info(2,3)=0.97
Info(0,2)=info(1,0)=0
Info(1,1)=0.5
Ave Info=0+0+(1/5)
$Gain(temp)$=0.97- 0.2 =0.77 bits



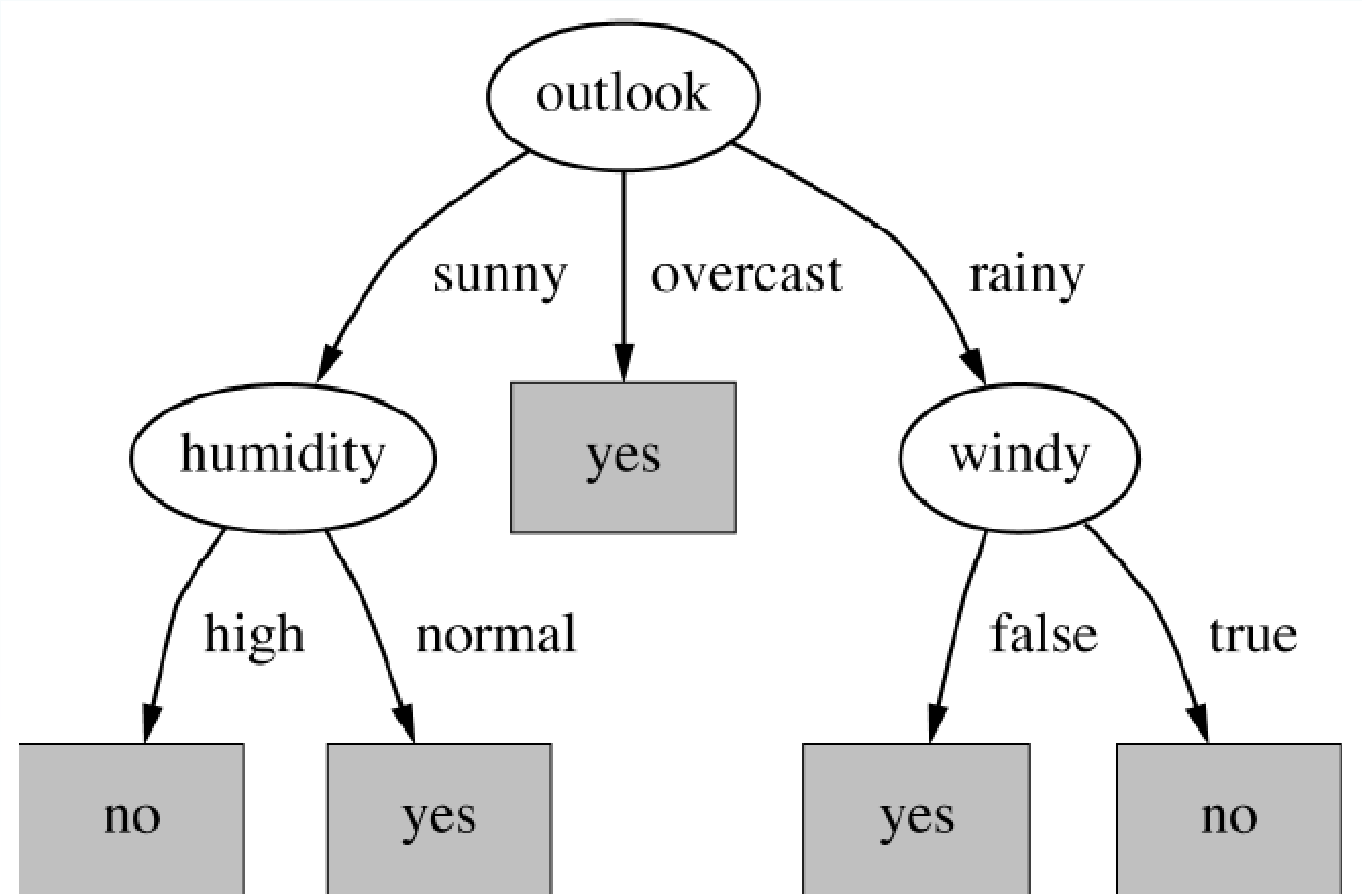
Info(3,3)=1
Info(2,2)=Info(1,1)=1
$Gain(windy)$=1-1
=0 bits



Info (3,2)= 0.97
Info (0,3)=Info(2,0)=0
$Gain(humidity)$=0.97 bits

Since *Gain*(*humidity*) is highest, select humidity as splitting attribute.
No need to split further

Decision Tree for the Weather Data



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).

Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees that overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

Rule post pruning

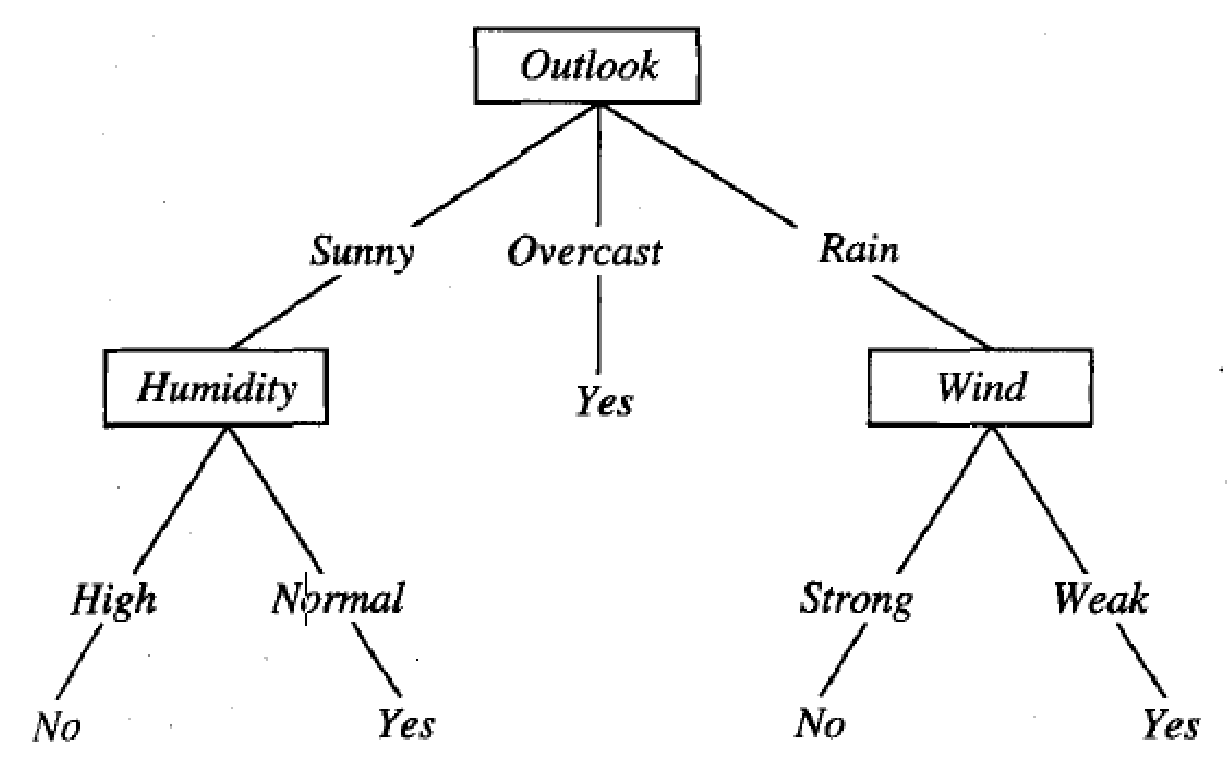
In practice, one quite successful method for finding high accuracy hypotheses is a technique we shall call rule post-pruning. A variant of this pruning method is used by **C4.5** (Quinlan 1993), which is an outgrowth of the original ID3 algorithm

1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances

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 results in improving estimated accuracy that
  - 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

C4.5



- **IF** (Outlook = Sunny) $\wedge$ (Humidity = High) **THEN** PalyTennis = **NO**
- **IF** (Outlook = Sunny) $\wedge$ (Humidity = Normal) **THEN** PalyTennis = **YES**
- **IF** (Outlook = Rain) $\wedge$ (Wind = Strong) **THEN** PalyTennis = **NO**
- **IF** (Outlook = Rain) $\wedge$ (Wind = Weak) **THEN** PalyTennis = **YES**

ID3 vs C4.5

	ID3	C4.5
Pruning	×	√
Features	nominal	nominal + numeric
missing values	×	√
Gain	Basic Info Gain	Gain Ratio

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)

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

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

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

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

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.
  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART, IBM IntelligentMiner)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  where p_j is the relative frequency of class j in D
- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as
- Reduction in Impurity:

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

- Ex. D has 9 tuples in Class = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Overfitting

- Definition: If your machine learning algorithm fits noise (i.e. pays attention to parts of the data that are irrelevant) it is overfitting.
- Fact (theoretical and empirical): If your machine learning algorithm is overfitting then it may perform less well on test set data.
- Ockham (1285-1349) Principle of Parsimony: "One should not increase, beyond what is necessary, the number of entities required to explain anything."

Handling Training Examples with Missing Attribute Values

- In certain cases, the available data may be missing values for some attributes.
  - Assign with the value that is most common among training examples at node n.
  - Assign with the most common value among examples at node n that have the same class.
  - Assign a probability to each of the possible values rather than simply assigning the most common value

Handling Attributes with Differing Costs

- In some learning tasks the instance attributes may have associated costs.
  - Tan and Schlimmer

$$\frac{Gain^2(S,A)}{Cost(A)}$$

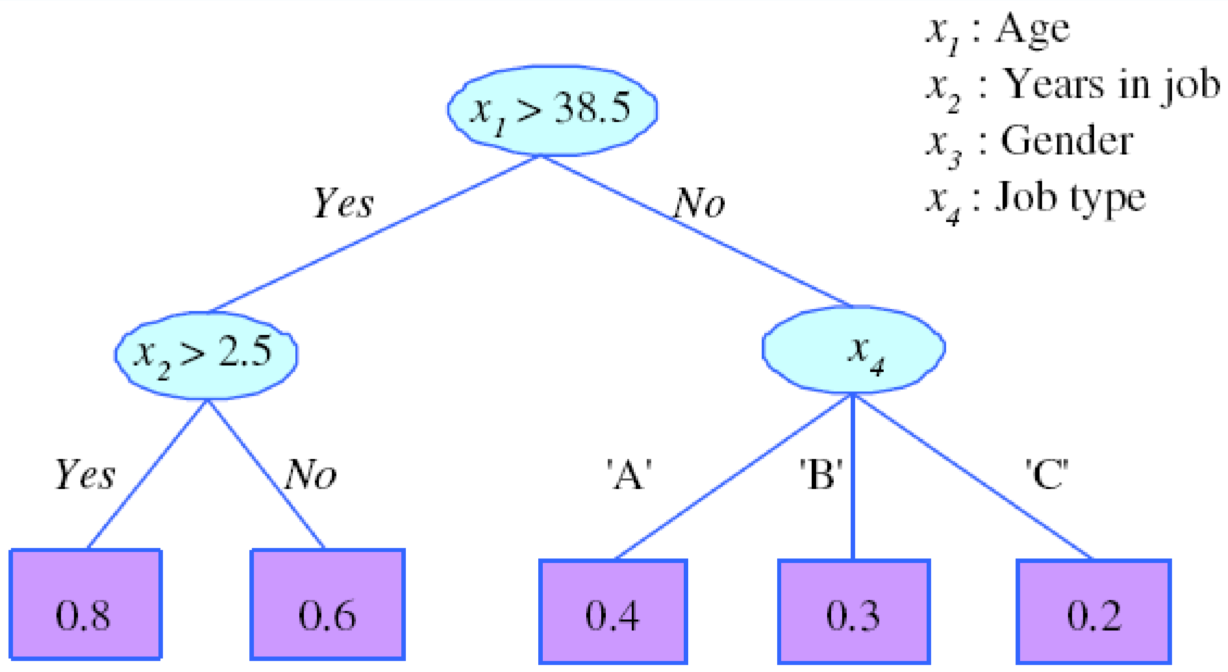
  - Nunez

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$

$w \in [0, 1]$

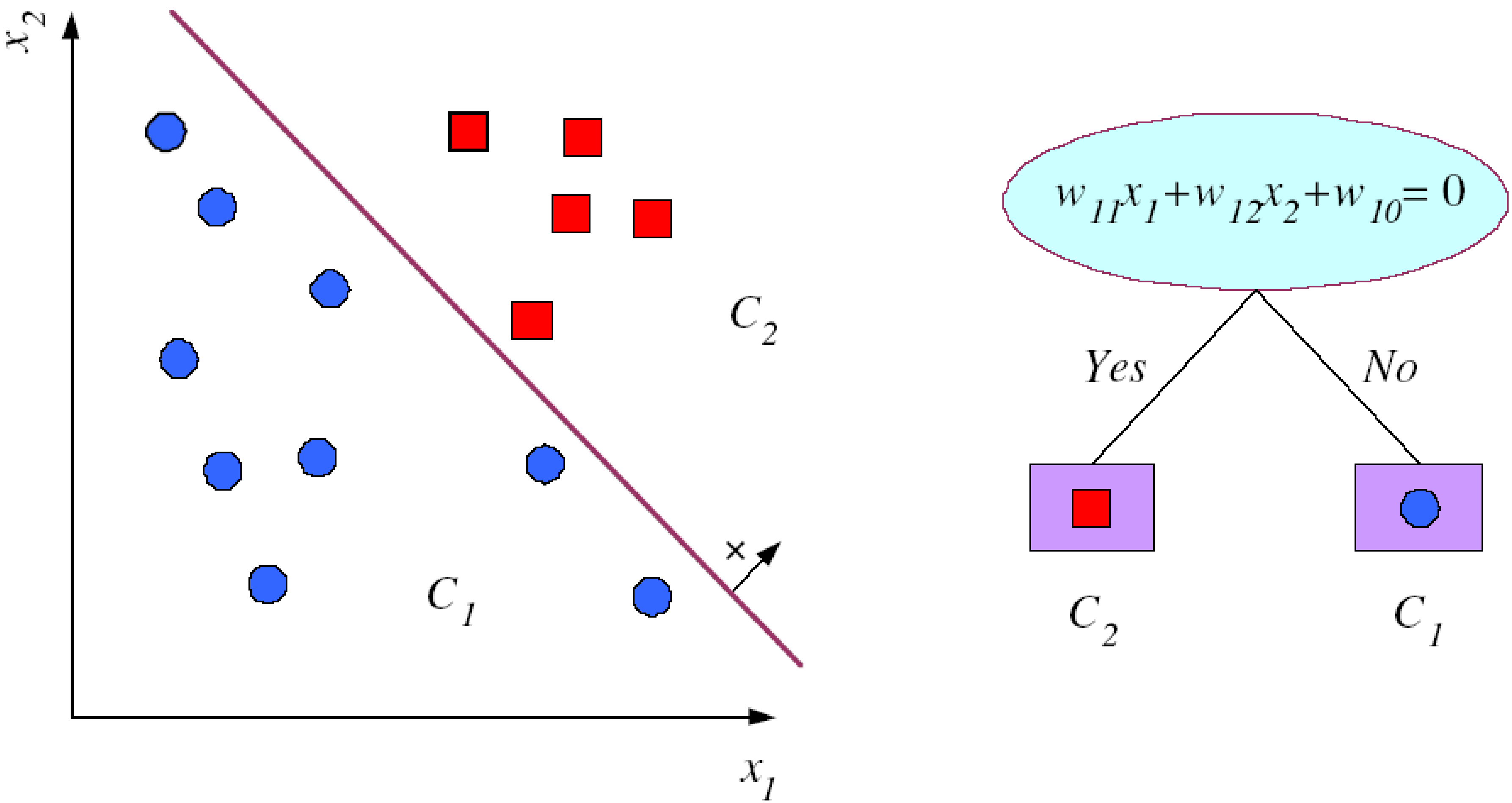
Rule Extraction from Trees

C4.5 Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

Multivariate Trees



Conclusions

- Decision trees are the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Easy to interpret
  - Computationally cheap
- It's possible to get in trouble with overfitting
- They do classification: predict a categorical output from categorical and/or real inputs

Reading

- C. M. Bishop, **Pattern recognition and machine learning**, Springer, 2006. (ch. 2)
- E. Alpaydin, **Introduction to Machine Learning**, 4th ed., The MIT Press, 2020. (ch. 9)
- T. Mitchel, **Machine learning**, McGraw-Hill Education, 1998. (ch. 8)