15-381: AI
Decision Tree Learning – Part I

October 18, 2009

Manuela Veloso

Chapter 18, Russell and Norvig

Thanks to all past instructors

Carnegie Mellon

Outline

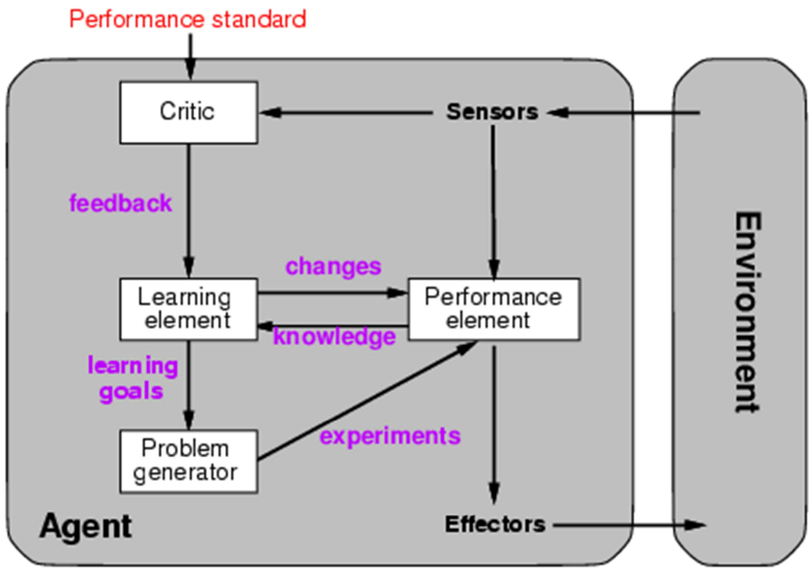
- Learning agents
- Inductive learning
- Decision tree learning

Learning

- Definition?...
- Gathering more knowledge
  - "Knowing more than was known before learning"
- Learning "substitutes" the need to model apriori
- Experience, feedback, refinement
- Learning modifies the agent's decision mechanisms to improve performance

In general, it is hard to define learning, as we don't really know what learning is.

Learning agents



In this example, the learning agent is something that learns by observing and interacting with the environment.

Learning "Element"

- A bit of "magic":
  - Which **components** of the performance element are to be learned
  - What **feedback** is available to learn these components
  - What **representation** is used for the components
- Type of feedback:
  - Supervised learning: correct labels/answers
  - Unsupervised learning: correct labels/answers missing
  - Reinforcement learning: occasional rewards

Magic == lots of hacks

Need a hypothesis representation that is general enough to express what you want to learn, but specific enough that the search for the correct hypothesis isn't too large.

Inductive Learning

- Simplest form: learn a function from **examples**

f is the target function

An example is a pair (x, $f(x)$) – *supervised learning*

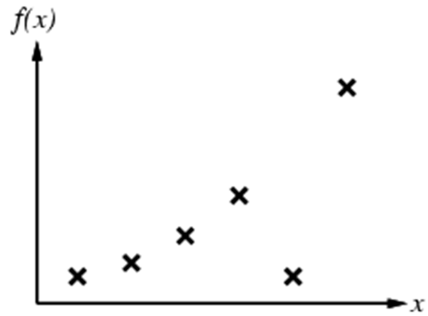
Problem: find a hypothesis h

such that $h \approx f$

given a training set of examples

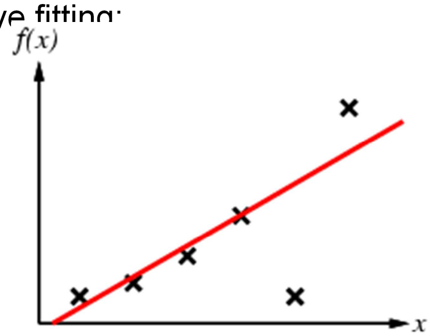
Inductive Learning Method

- Construct/adjust *h* to agree with *f* on training set
- *h* is consistent if it agrees with *f* on all examples
- E.g., curve fitting:
-



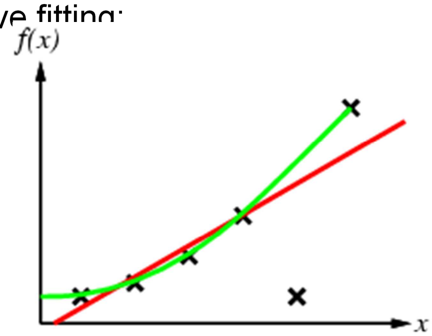
Inductive Learning Method

- Construct/adjust *h* to agree with *f* on training set
- *h* is consistent if it agrees with *f* on all examples
-
- E.g., curve fitting:
-



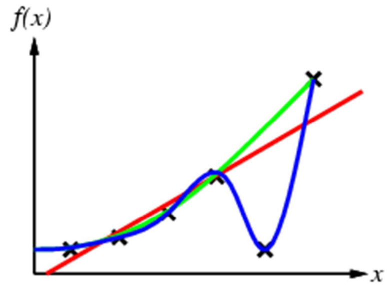
Inductive Learning Method

- Construct/adjust *h* to agree with *f* on training set
- *h* is consistent if it agrees with *f* on all examples
-
- E.g., curve fitting:
-



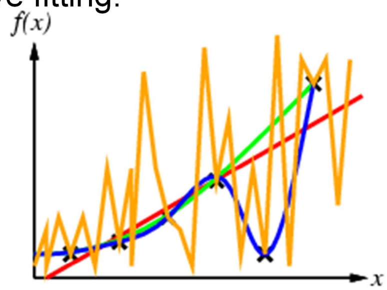
Inductive Learning Method

- Construct/adjust *h* to agree with *f* on training set
- *h* is consistent if it agrees with *f* on all examples
- E.g., curve fitting:
-



Inductive Learning Method

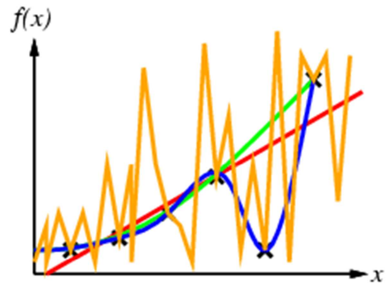
- Construct/adjust *h* to agree with *f* on training set
- *h* is consistent if it agrees with *f* on all examples
- E.g., curve fitting:



Given any number of points, it is possible to find a hypothesis that is consistent for every training point. However we want the hypothesis to also predict points that were not training points. This slide is an example of overfitting. Yes, it is consistent with ever training example, but it is a poor predictor of points not in the training data set.

Inductive Learning Method - Bias

- Hypothesis "form" – **bias** on the learning outcome
- Ockham's razor: prefer the simplest hypothesis consistent with data



There is an implicit bias in the hypothesis's form. For example, if your hypothesis is a 4^{th} degree polynomial, where you are learning the coefficient of each term, you can only represent curves of degree 4 and less.

Ockham's razor: when two hypothesis perform similarly, prefer the simpler one (ie if a line and a curve both fit points, then prefer a line).

Attribute-Based Data Sets

- Examples described by attribute values (Boolean, discrete, continuous)
- Function is class, wait/not wait for table at restaurant

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)

How should data be represented? Here, each row is one example, each column is an attribute.

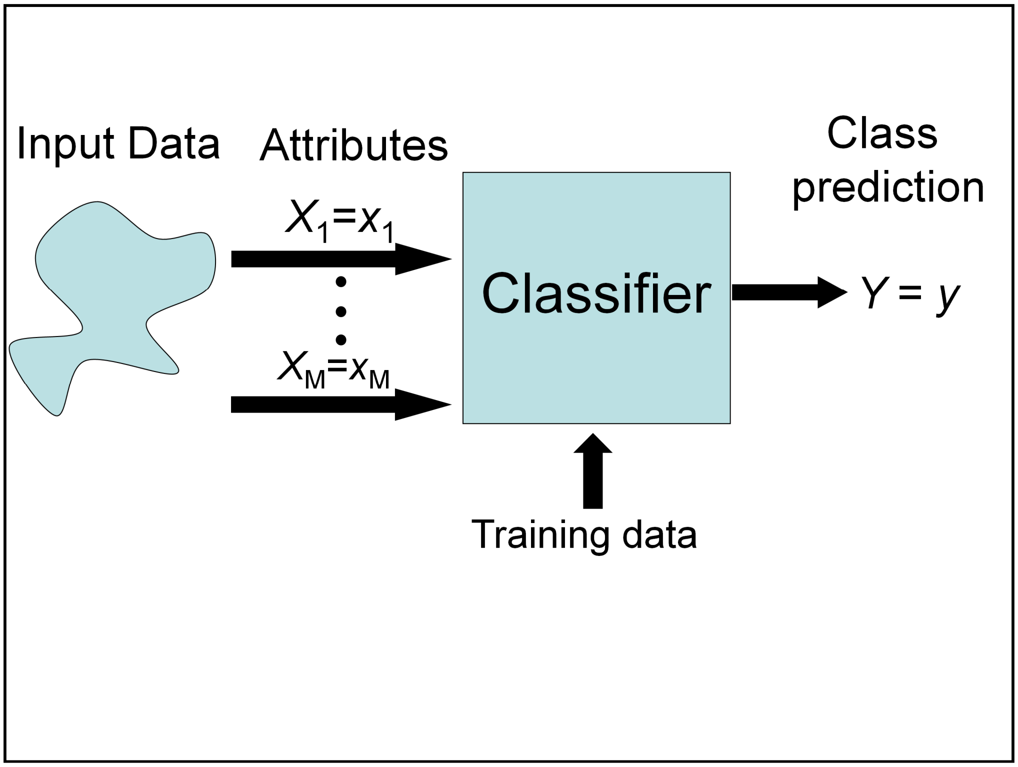
Each attribute takes a value. Boolean attributes take true/false, etc. Some attributes are discrete, some are continuous. As you'll see later on, continuous valued attributes can complicate things.

Attribute-Based Data Set

- *Type*: drama, comedy, thriller
- *Company*: MGM, Columbia
- *Director*: Bergman, Spielberg, Hitchcock
- *Mood*: stressed, relaxed, normal

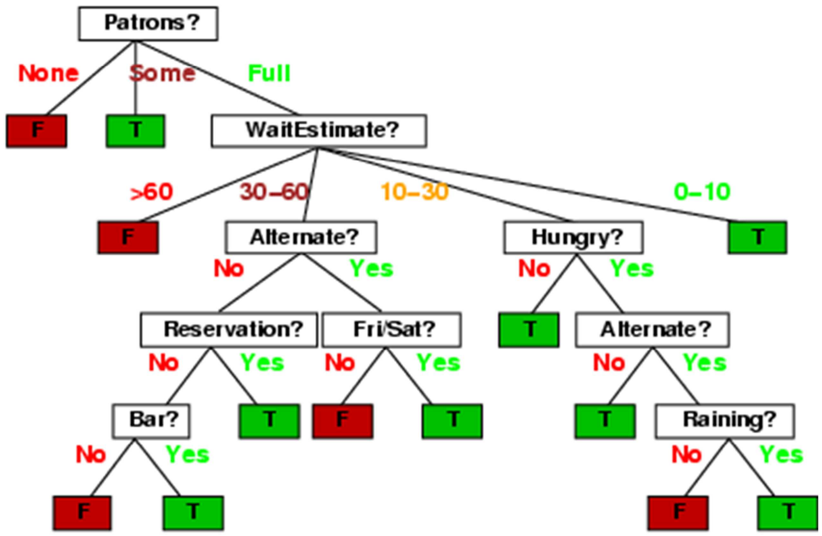
Movie	Type	Company	Director	Mood	Likes-movie?
m_1	thriller	MGM	Bergman	normal	No
m_2	comedy	Columbia	Spielberg	stressed	Yes
m_3	comedy	MGM	Spielberg	relaxed	No
m_4	thriller	MGM	Bergman	relaxed	No
m_5	comedy	MGM	Hitchcock	normal	Yes
m_6	drama	Columbia	Bergman	relaxed	Yes
m_7	drama	Columbia	Bergman	normal	No
m_8	drama	MGM	Spielberg	stressed	No
m_9	drama	MGM	Hitchcock	normal	Yes
m_{10}	comedy	Columbia	Spielberg	relaxed	No
m_{11}	thriller	MGM	Spielberg	normal	No
m_{12}	thriller	Columbia	Hitchcock	relaxed	No

A learner should learn that the type, director and your mood are important to whether or not you liked the movie, while the company is not so important


Input Data
Attributes
Class prediction
$X_1=x_1$
$X_M=x_M$
Classifier
$Y = y$
Training data

Data Representation - Decision Trees

- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:



This is a decision tree to decide whether or not you are going to wait for a table at a restaurant.

Here, hypotheses are in the form of decision trees. There are many possible decision trees (hypotheses). You are searching for the simplest decision tree (hypothesis) that is most consistent with the training data

Is this example the simplest?

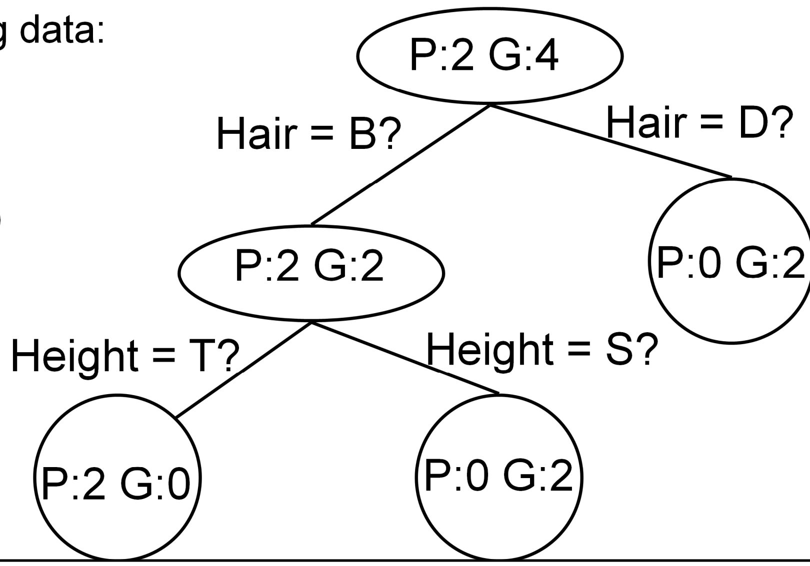
Expressiveness

- Decision trees can express any function of the input attributes
  - E.g., for Boolean functions, truth table row → path to leaf
- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*)
  - But… it probably won't generalize to new examples – goal of learning…
- Goal: Find more "compact" decision trees

Building a Decision Tree

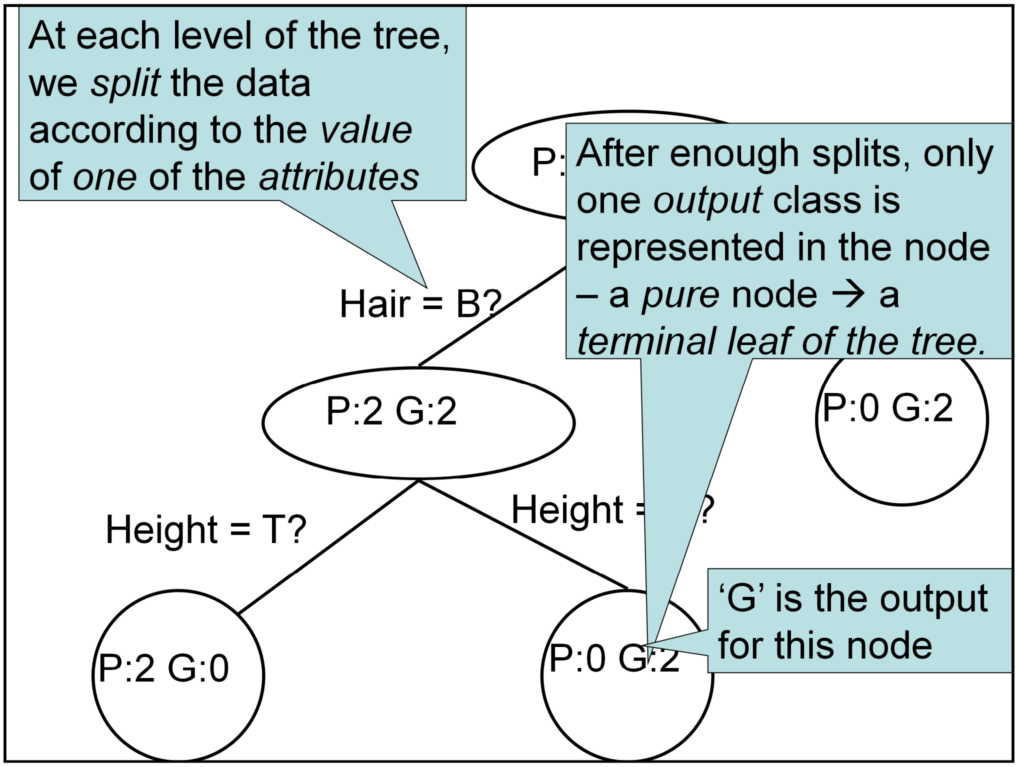
- Three variables:
  - Hair = {blond, dark}
  - Height = {tall,short}
  - Country = {Gromland, Polvia}

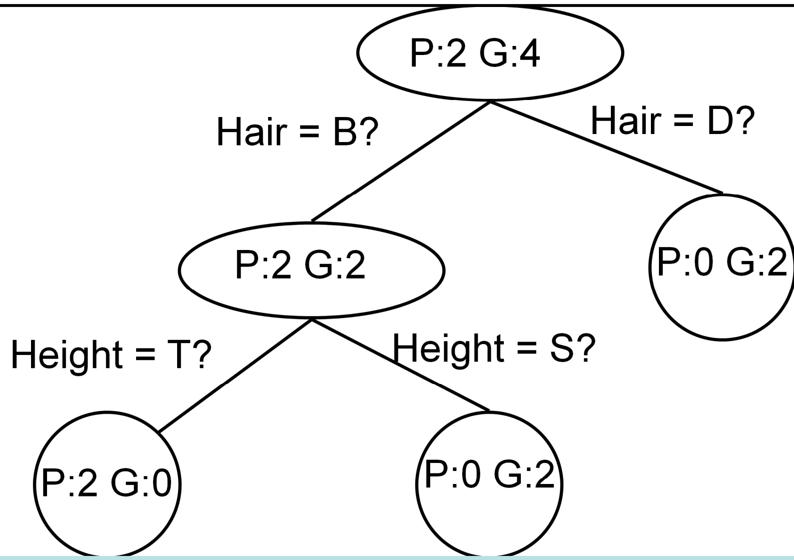
Training data:
(B,T,P)
(B,T,P)
(B,S,G)
(D,S,G)
(D,T,G)
(B,S,G)



Trying to classify examples as P or G based on their attributes

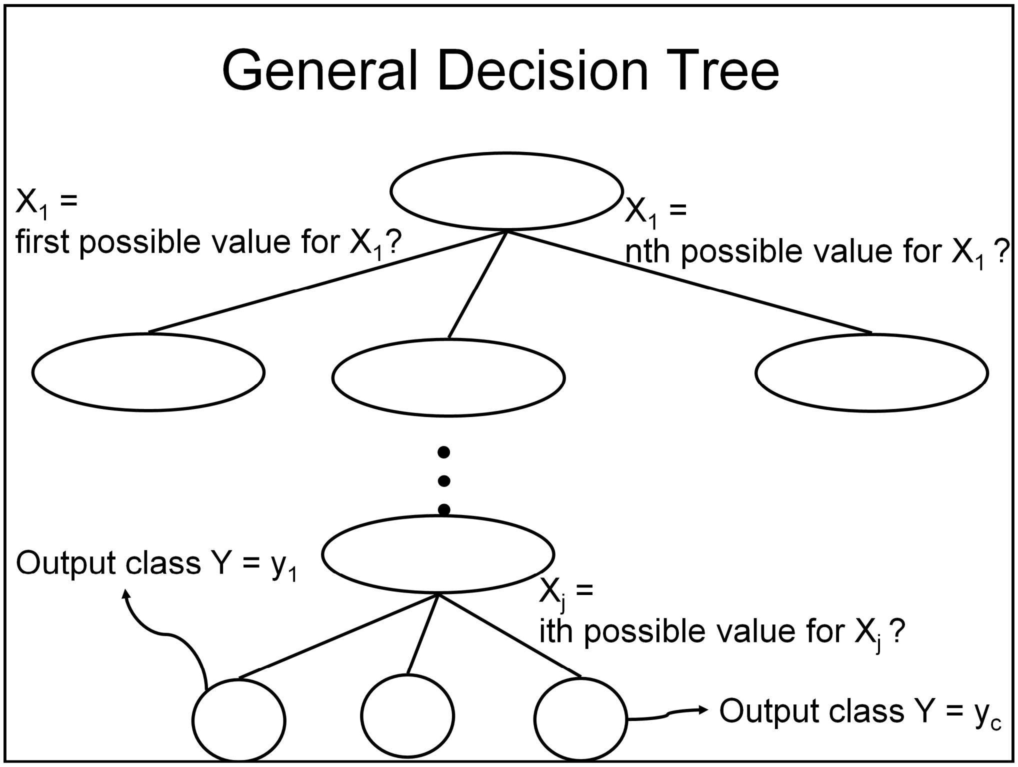
If tree height is what determines which hypothesis is simpler, there is no simpler, consistent hypothesis


At each level of the tree, we *split* the data according to the *value* of *one* of the *attributes*
P:
After enough splits, only one *output* class is represented in the node – a *pure* node → a *terminal leaf of the tree.*
Hair = B?
P:2 G:2
P:0 G:2
Height = T?
Height =
?
'G' is the output for this node
P:2 G:0
P:0 G:2



The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:

(B,T) is classified as P; (D,S) is classified as G



General Decision Tree
X_1 =
first possible value for X_1?
X_1 =
nth possible value for X_1 ?
Output class Y = y_1
X_j =
ith possible value for X_j ?
Output class Y = y_c

Basic Questions

- How **to choose the attribute/value to split** on at each level of the tree?
- When **to stop splitting**? When should a node be declared a leaf?
- If a leaf node is impure, how should the **class label be assigned**?
- If the tree is too large, how can it be **pruned**?

Set of Training Examples

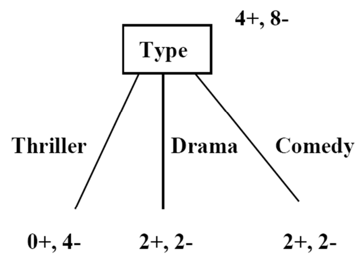
- *Type*: drama, comedy, thriller
- *Company*: MGM, Columbia
- *Director*: Bergman, Spielberg, Hitchcock
- *Mood*: stressed, relaxed, normal

Movie	Type	Company	Director	Mood	Likes-movie?
m_1	thriller	MGM	Bergman	normal	No
m_2	comedy	Columbia	Spielberg	stressed	Yes
m_3	comedy	MGM	Spielberg	relaxed	No
m_4	thriller	MGM	Bergman	relaxed	No
m_5	comedy	MGM	Hitchcock	normal	Yes
m_6	drama	Columbia	Bergman	relaxed	Yes
m_7	drama	Columbia	Bergman	normal	No
m_8	drama	MGM	Spielberg	stressed	No
m_9	drama	MGM	Hitchcock	normal	Yes
m_{10}	comedy	Columbia	Spielberg	relaxed	No
m_{11}	thriller	MGM	Spielberg	normal	No
m_{12}	thriller	Columbia	Hitchcock	relaxed	No

Choosing the "Best" Attribute

- Ideal attribute – partitions examples into all positive or all negative (or from the same class in each partition).
- Attribute that results in higher "discrimination."

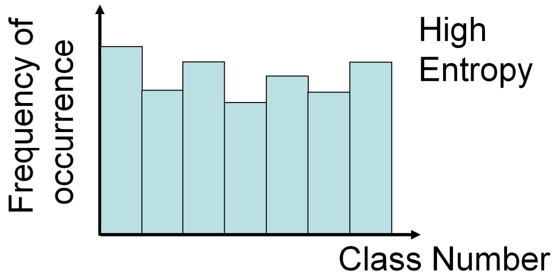
How *good* is the attribute "Type" of movie?

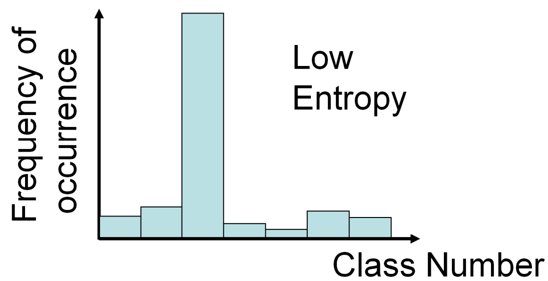


Any attribute splits the data (though some groups might be empty). Here, the type attribute splits the data into three groups.

The optimal attribute would split the data into pure nodes. Here, it would be great if just knowing the type of movie would tell you whether or not the movie was liked.

Entropy *(Shannon and Weaver 1949)*





The entropy captures the degree of “purity” of the data distribution

High entropy because all the information is mixed up, it is chaotic.

Low entropy because the information is organized and orderly.

Entropy *(Shannon and Weaver 1949)*

- In general, entropy is the average number of bits necessary to encode *n* values:

$$\boxed{H = -\sum_{i=1}^{n} - P_i \log_2 P_i}$$

- P_i = probability of occurrence of value *i*
  - High entropy → All the classes are (nearly) equally likely
  - Low entropy → A few classes are likely; most of the classes are rarely observed

Know the formula for entropy, it is very, very useful!

Log base conversion from base a to base b: loga(x) = logb(x)/logb(a)

Entropy for Attribute Choice

- Measure of *information* provided by the attribute
- **Entropy** of a set of examples S as the information content of S.

$$\text{Entropy(S)} = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$\text{where } p_i = \frac{|S_i|}{|S|}$$

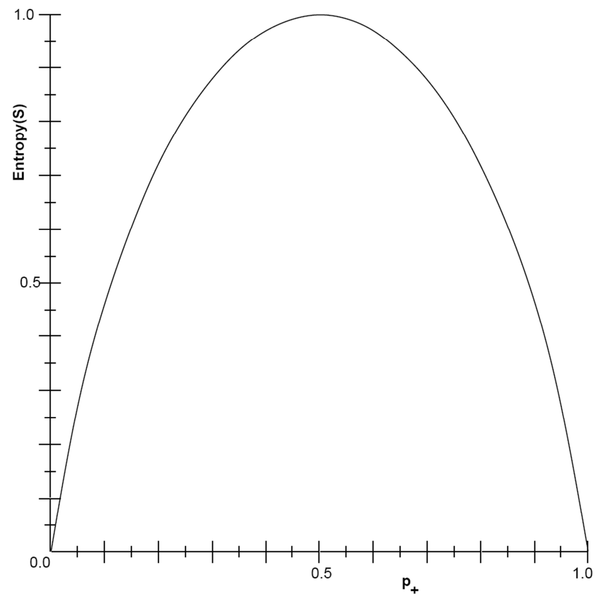
- c classes, Si size of the data set for class i

Entropy

- Unit – 1 bit of information =
  - the information content of the actual answer when there are two possible answers equally probable.

$$E(S) = -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{2}\log_2 \tfrac{1}{2} = 1$$

Entropy Relative – Boolean Classification



Imagine having a basket full of red and black balls. If p is probability of being black, then when everything is red, p is 0 and entropy is 0. When everything is black, and p is 1, then entropy is 0. If half are red and half are black, p is 0.5 and entropy is 1.

Entropy – Example

- $|S| = 12$, $c = 2(+,-)$, $|S_+| = 4$, $|S_-| = 8$

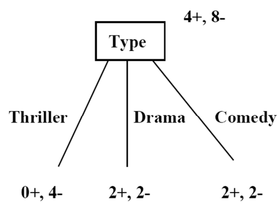
$$
\begin{aligned}
E(S) &= -\frac{4}{12}\log_2\frac{4}{12} - \frac{8}{12}\log_2\frac{8}{12} \\
&= 0.918
\end{aligned}
$$

4 positive examples and 8 negative examples. Entropy is still very high.

Choosing the "Best" Attribute

- Attribute with **highest information gain** – expected reduction in entropy of the set S if partitioned according to attribute A.

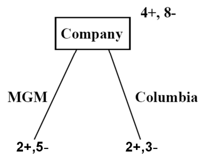
$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{i=1}^{\text{values}(A)} \frac{|S_{v_i}|}{|S|}\text{Entropy}(S_{v_i})$$



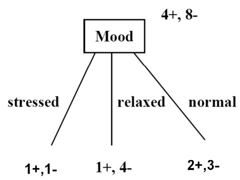
$$\begin{aligned}\text{Gain}(S,\text{Type}) = E(4^+,8^-) & -\tfrac{4}{12}E(0^+,4^-) \\ & -\tfrac{4}{12}E(2^+,2^-) \\ & -\tfrac{4}{12}E(2^+,2^-) \\ = 0.252 &\end{aligned}$$

Split on the attribute that gives the most information gain (reduces the entropy the most). This is not necessarily optimal, but it is a good heuristic.

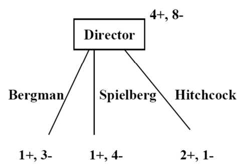
Other Attributes



$$E(4^+,8^-) - \tfrac{7}{12}E(2^+,5^-) - \tfrac{5}{12}E(2^+,3^-)$$
$$= 0.0102$$

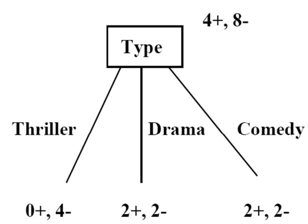


$$E(4^+,8^-) - \tfrac{2}{12}E(1^+,1^-) - \tfrac{5}{12}E(1^+,4^-) - \tfrac{5}{12}E(2^+,3^-)$$
$$= 0.046$$



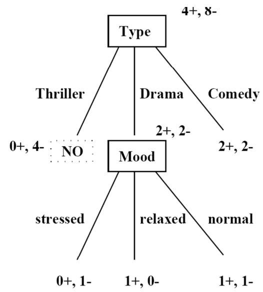
$$E(4^+,8^-) - \tfrac{4}{12}E(1^+,3^-) - \tfrac{5}{12}E(1^+,4^-) - \tfrac{3}{12}E(2^+,1^-)$$
$$= 0.118$$

Commitment to One Hypothesis



What is the next *best* attribute? Recursive.

Split Data and Continue

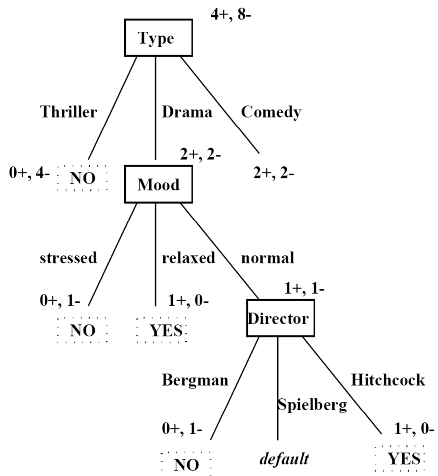


$$\begin{aligned}\text{Gain}(S_d, \text{Mood}) &= E(2^+,2^-) - \tfrac{1}{4}E(0^+,1^-) - \tfrac{1}{4}E(1^+,0^-) - \tfrac{2}{4}E(1^+,1^-) \\ &= 0.5\end{aligned}$$

$$\begin{aligned}\text{Gain}(S_d, \text{Company}) &= E(2^+,2^-) - \tfrac{2}{4}E(1^+,1^-) - \tfrac{2}{4}E(1^+,1^-) \\ &= 0\end{aligned}$$

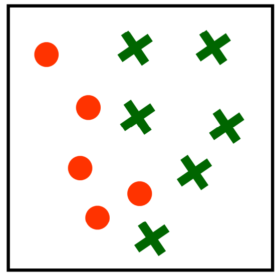
$$\begin{aligned}\text{Gain}(S_d, \text{Director}) &= E(2^+,2^-) - \tfrac{2}{4}E(1^+,1^-) - \tfrac{1}{4}E(0^+,1^-) - \tfrac{2}{4}E(1^+,0^-) \\ &= 0.5\end{aligned}$$

Learned Tree



You need some default answer for missing data.

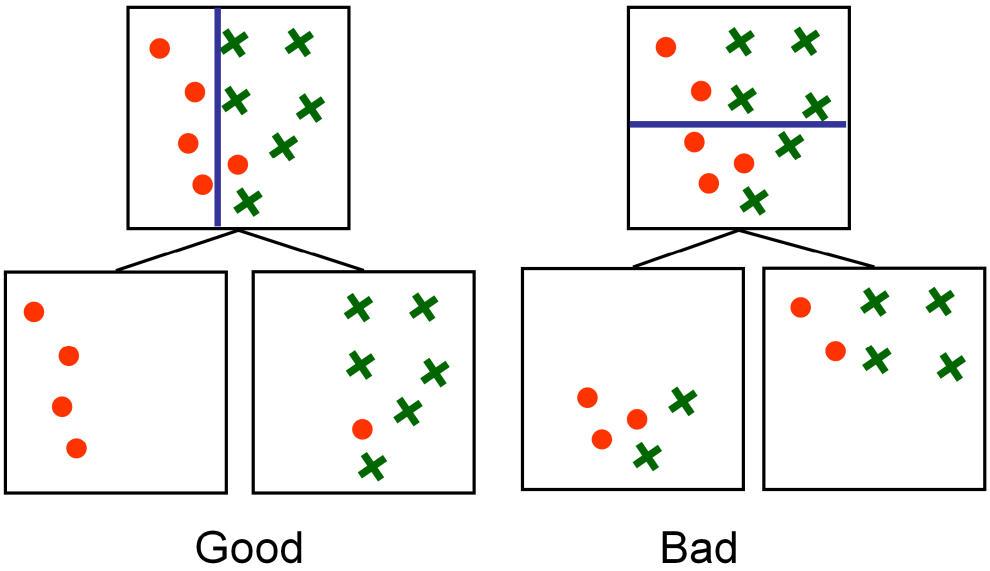
Continuous Case- How to Split?

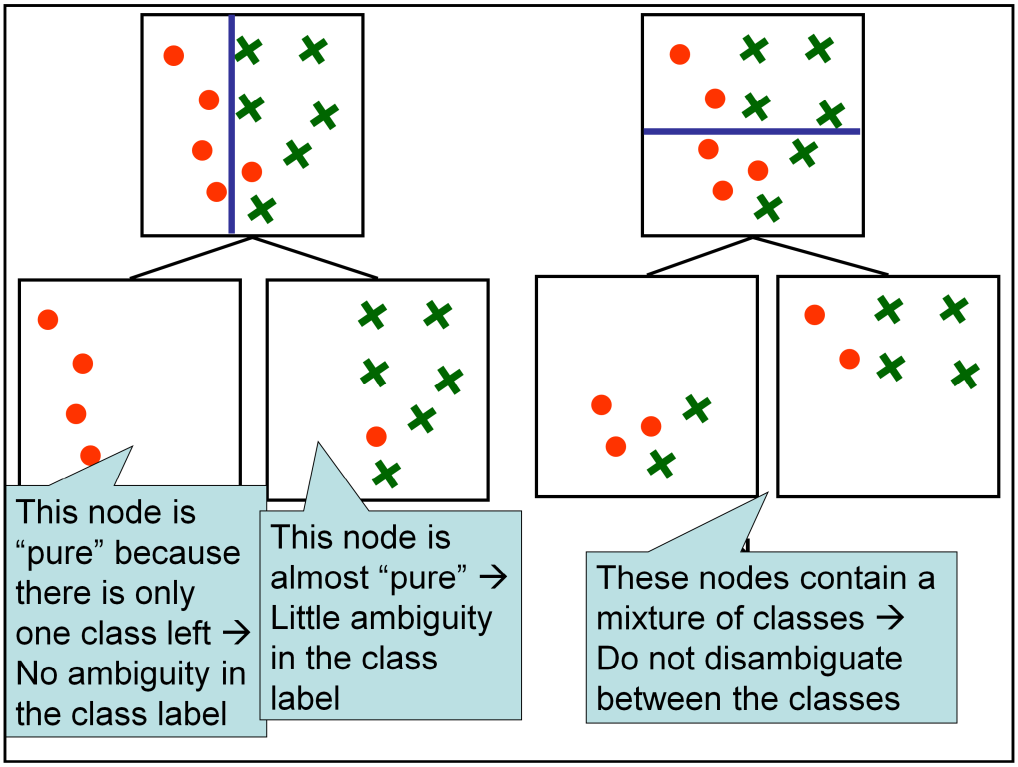


- Two classes (red circles/green crosses)
- Two attributes: X_1 and X_2
- 11 points in training data
- Idea → Construct a decision tree such that the leaf nodes predict correctly the class for all the training examples

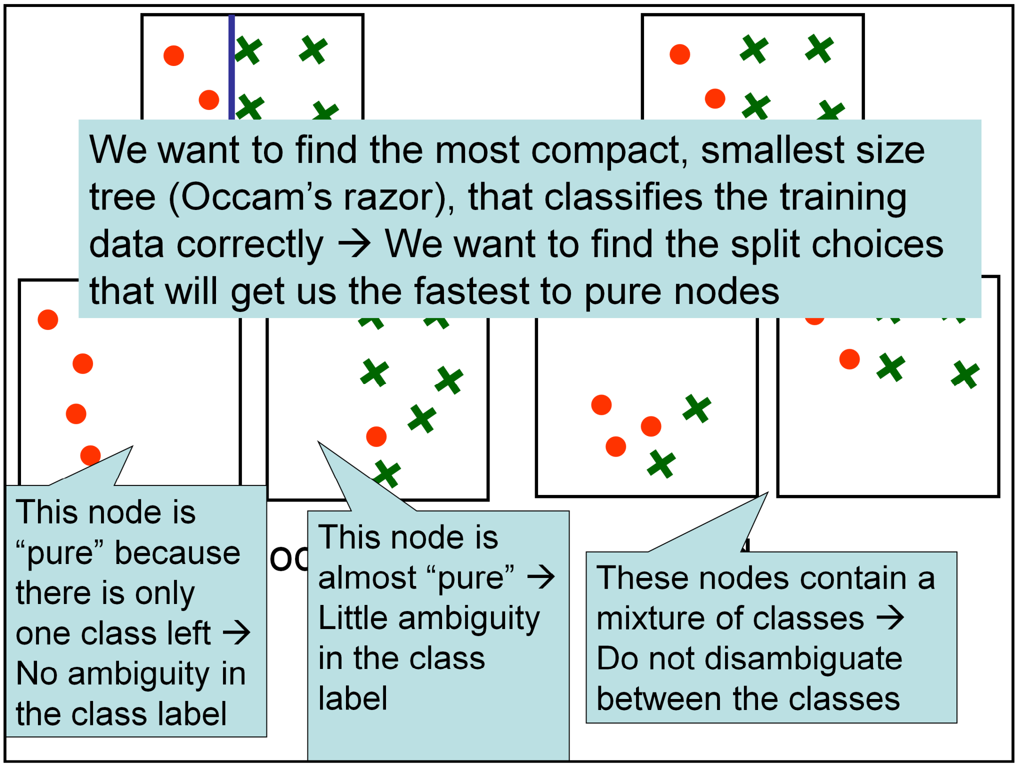
Learn which values of each attribute is the best to split on

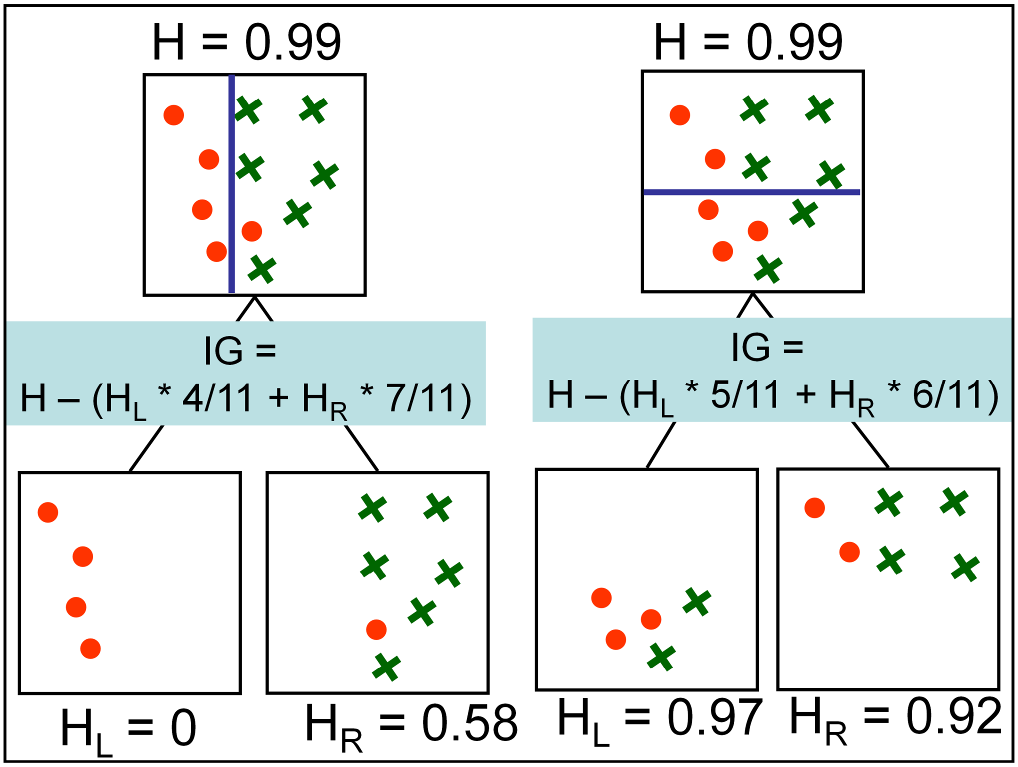
Good and Bad Splits

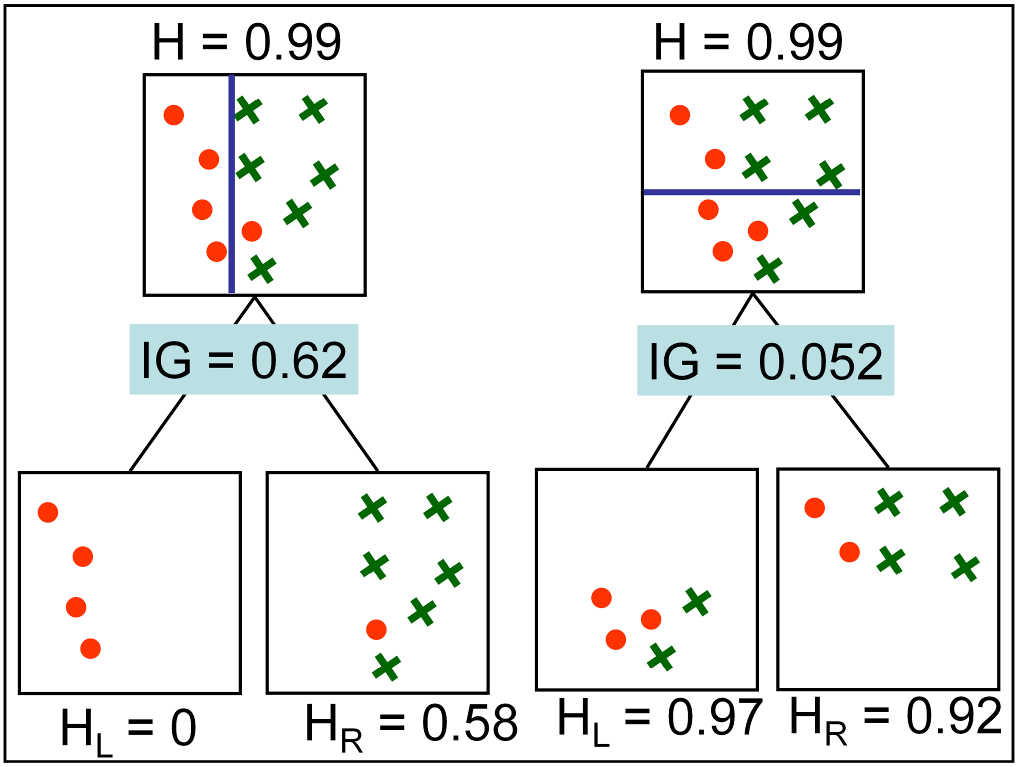




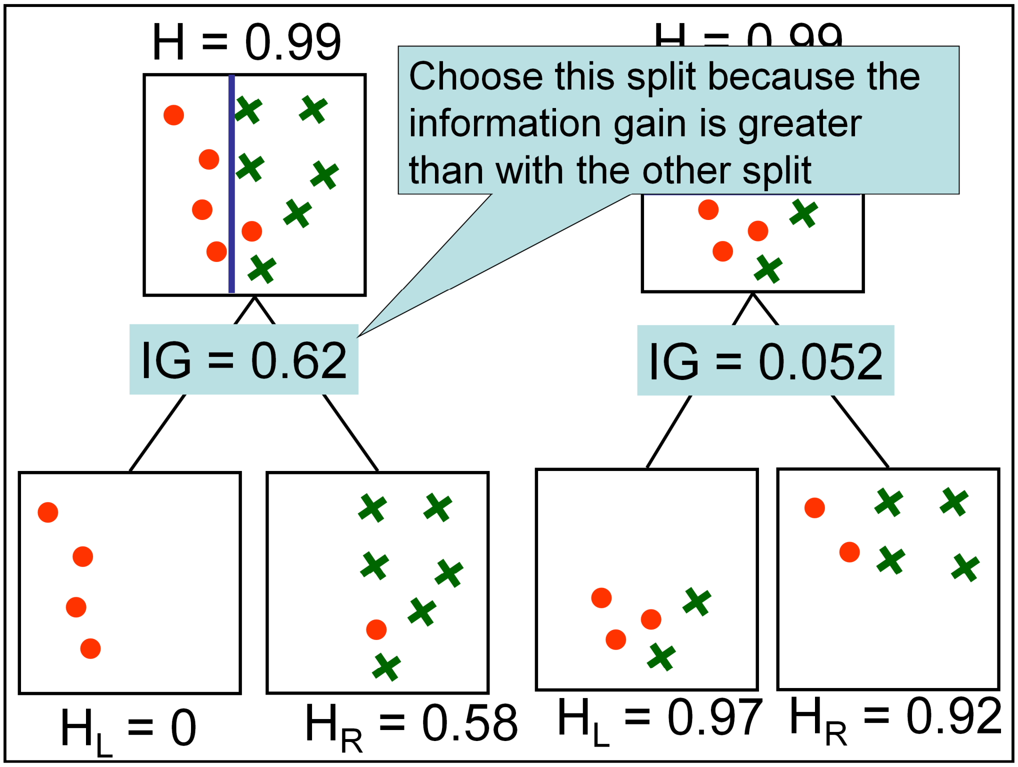
This node is "pure" because there is only one class left → No ambiguity in the class label
This node is almost "pure" → Little ambiguity in the class label
These nodes contain a mixture of classes → Do not disambiguate between the classes


We want to find the most compact, smallest size tree (Occam's razor), that classifies the training data correctly → We want to find the split choices that will get us the fastest to pure nodes
This node is "pure" because there is only one class left → No ambiguity in the class label
This node is almost "pure" → Little ambiguity in the class label
These nodes contain a mixture of classes → Do not disambiguate between the classes

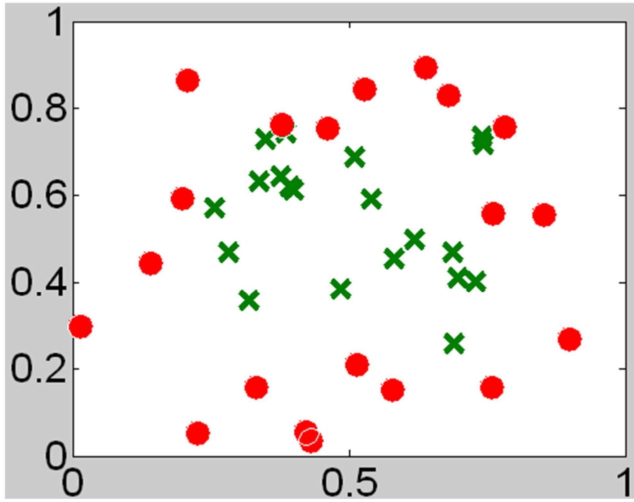

H = 0.99
H = 0.99
IG =
H – (H_L * 4/11 + H_R * 7/11)
IG =
H – (H_L * 5/11 + H_R * 6/11)
H_L = 0
H_R = 0.58
H_L = 0.97
H_R = 0.92



H = 0.99
H = 0.99
IG = 0.62
IG = 0.052
H_L = 0
H_R = 0.58
H_L = 0.97
H_R = 0.92


H = 0.99
H = 0.99
Choose this split because the information gain is greater than with the other split
IG = 0.62
IG = 0.052
H_L = 0
H_R = 0.58
H_L = 0.97
H_R = 0.92

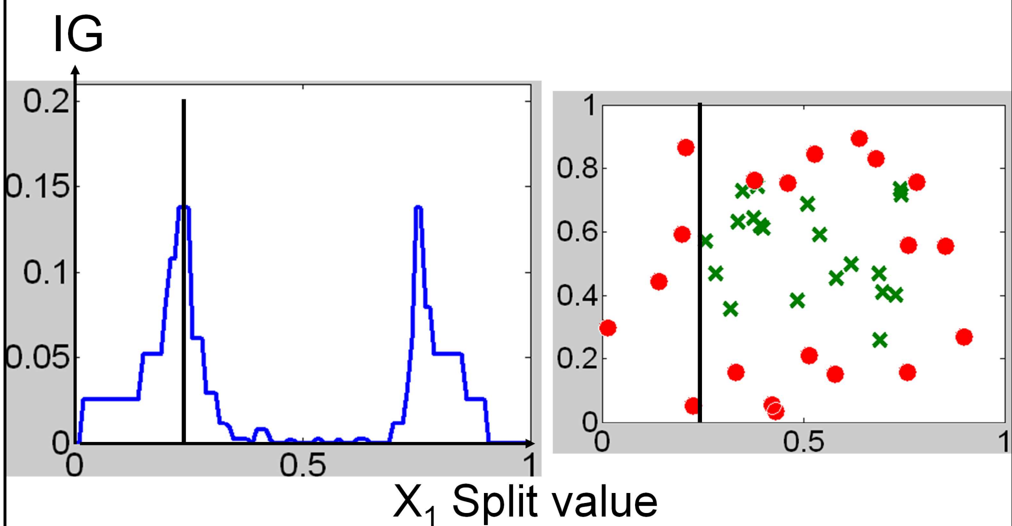
More Complete Example



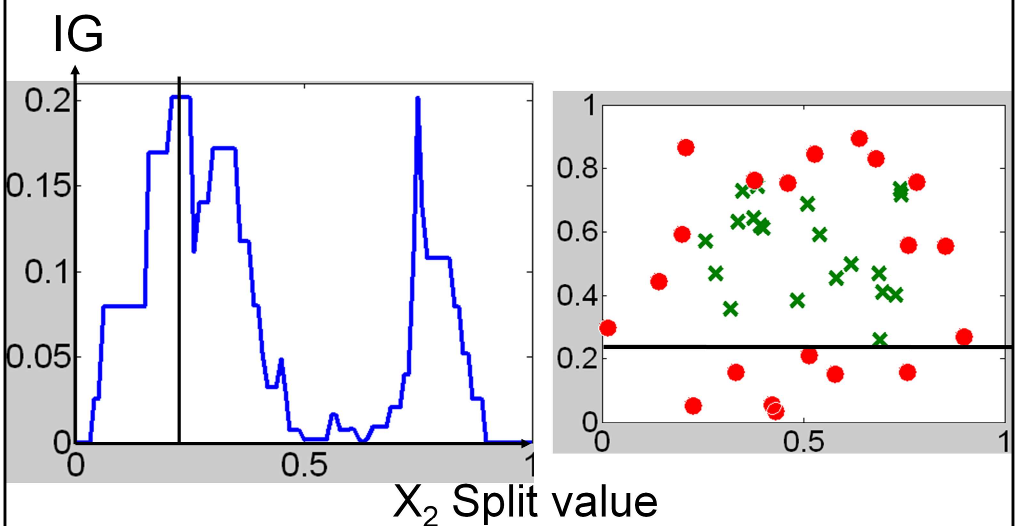
= 20 training examples from class A

= 20 training examples from class B

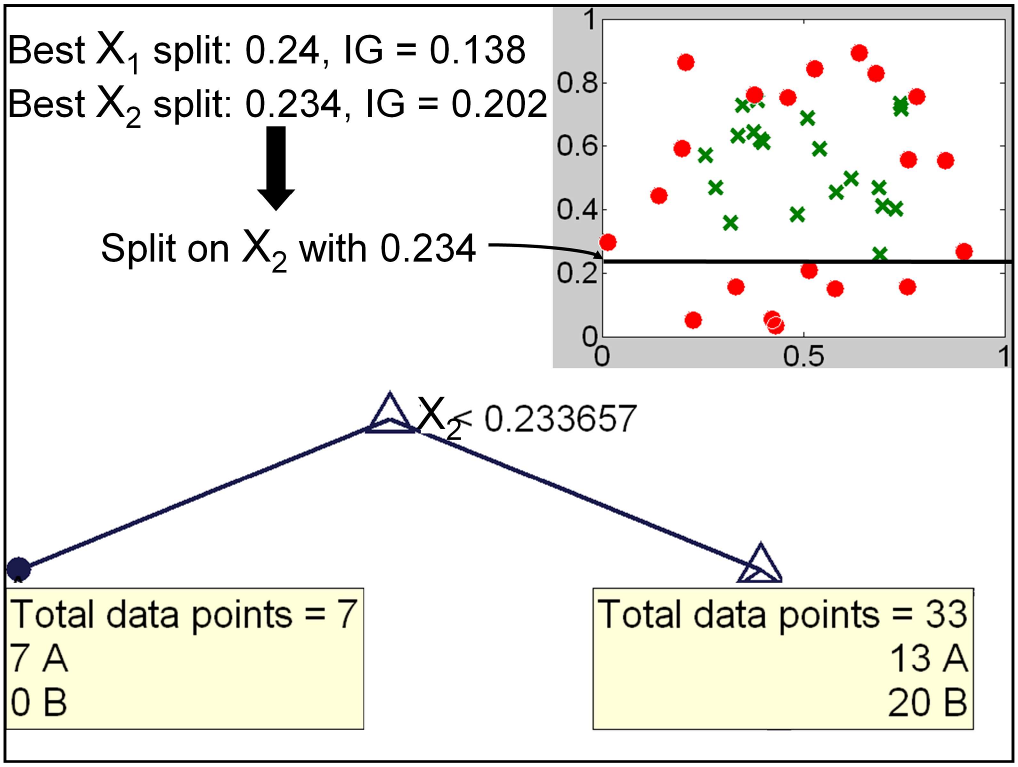
Attributes = X_1 and X_2 coordinates



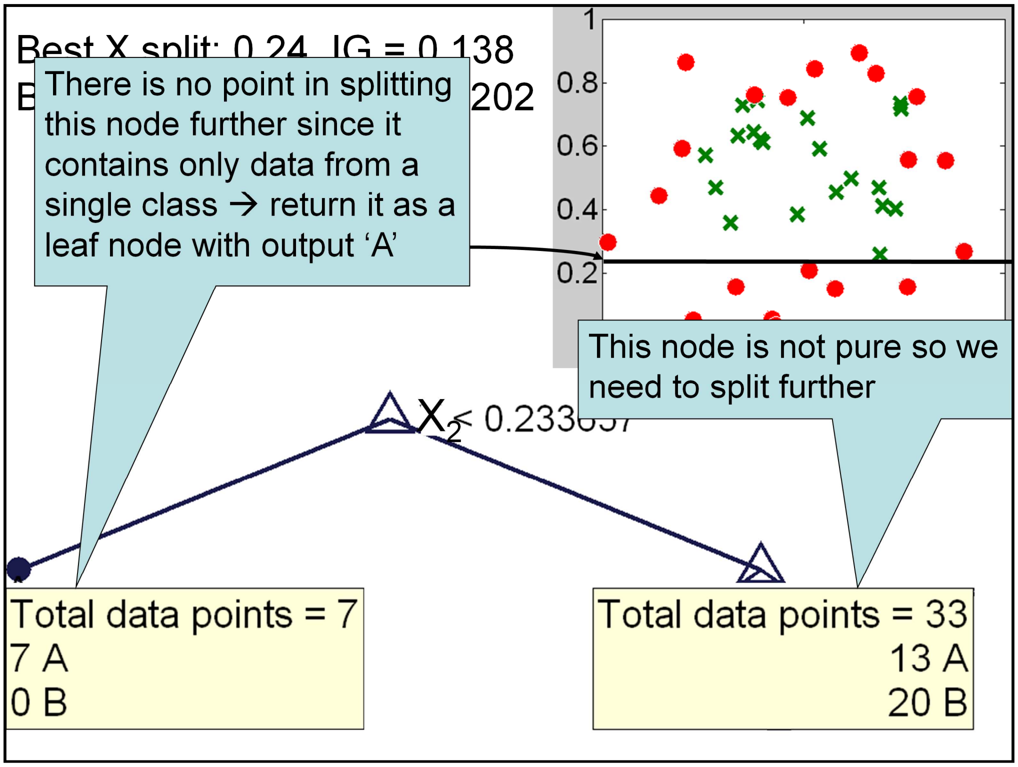
Best split value (max Information Gain) for X_1 attribute: 0.24 with IG = 0.138



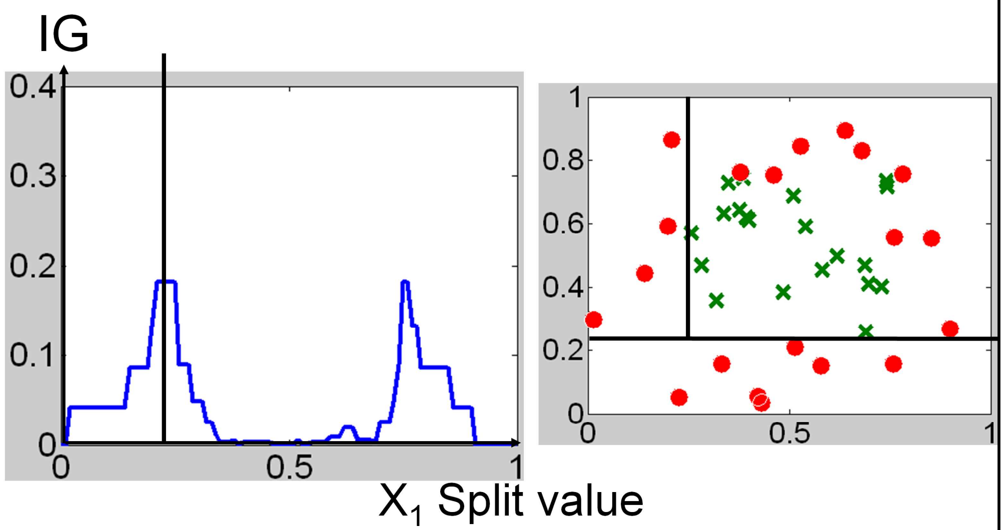
Best split value (max Information Gain) for X_2 attribute: 0.234 with IG = 0.202



Best X1 split: 0.24, IG = 0.138
Best X2 split: 0.234, IG = 0.202
Split on X2 with 0.234
X2 < 0.233657
Total data points = 7
7 A
0 B
Total data points = 33
13 A
20 B

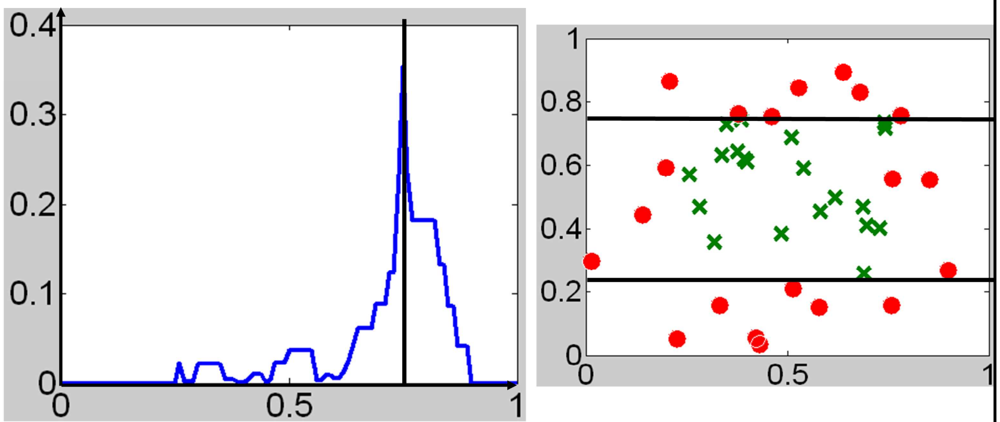


Best X split: 0.24, IG = 0.138
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output 'A'
202
1
0.8
0.6
0.4
0.2
This node is not pure so we need to split further
Total data points = 7
7 A
0 B
Total data points = 33
13 A
20 B



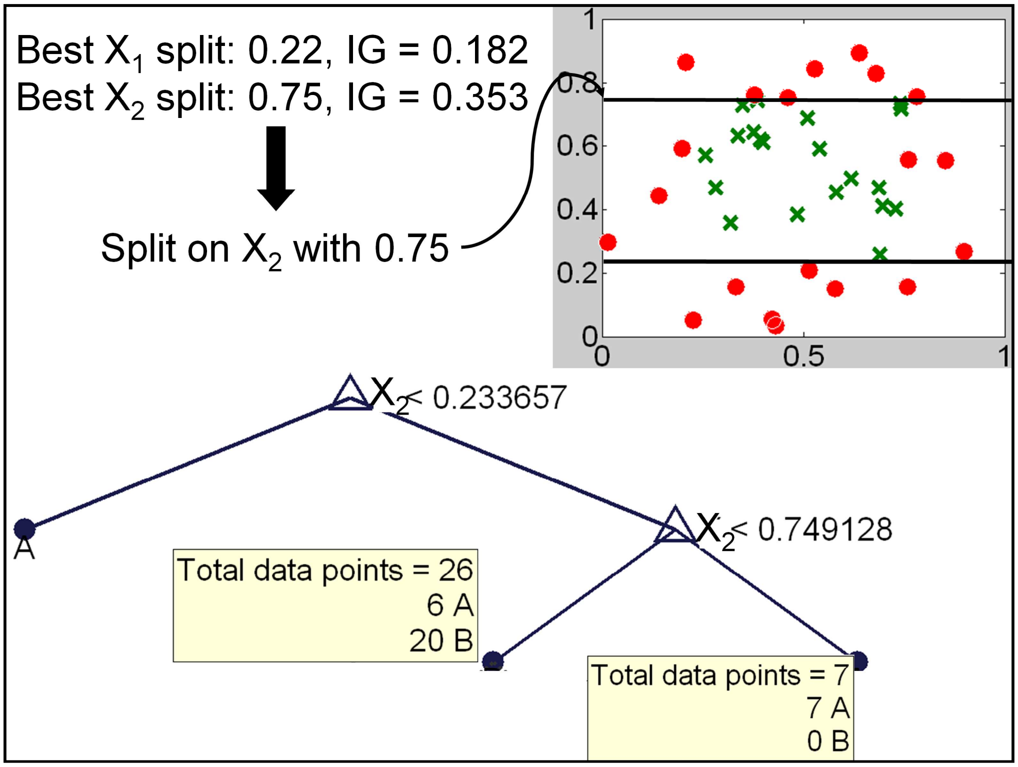
Best split value (max Information Gain) for X_1 attribute: 0.22 with IG ~ 0.182

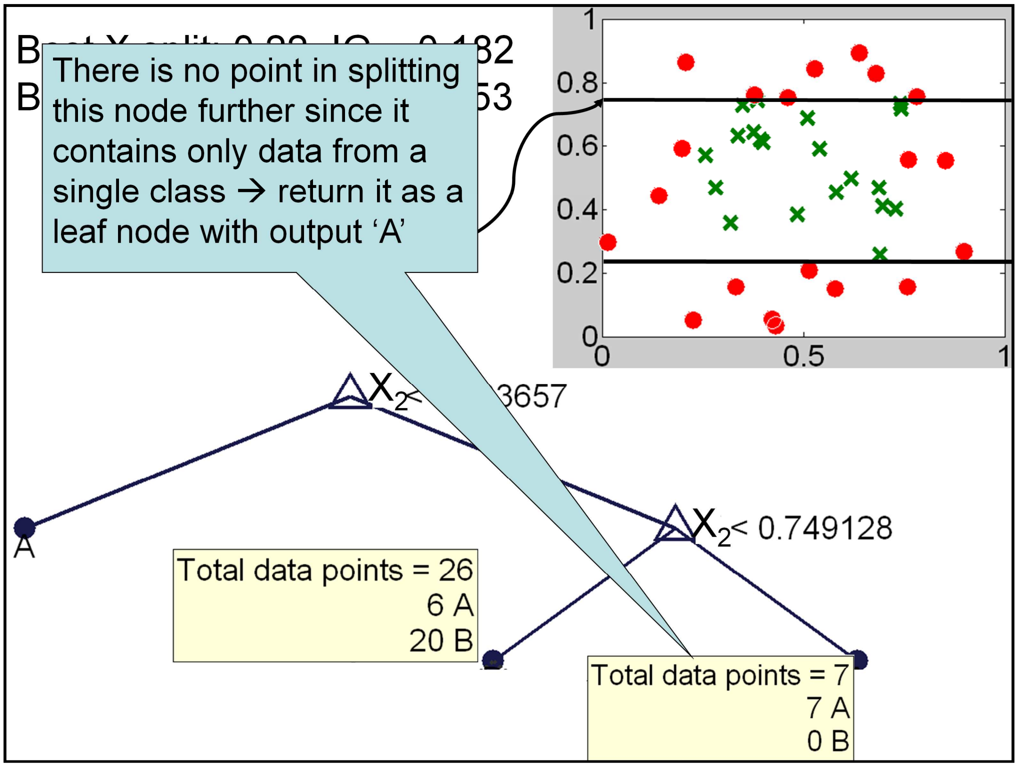
IG

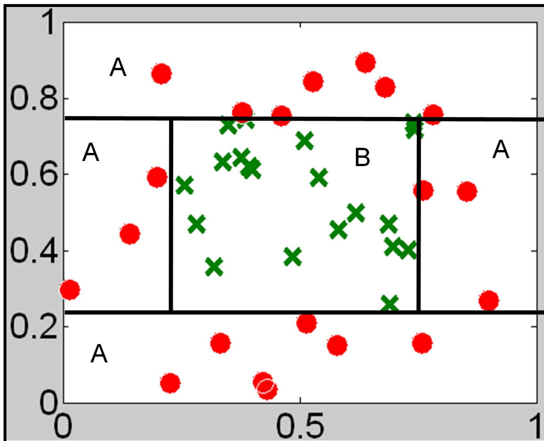


X_2 Split value

Best split value (max Information Gain) for X_2 attribute: 0.75 with IG ~ 0.353

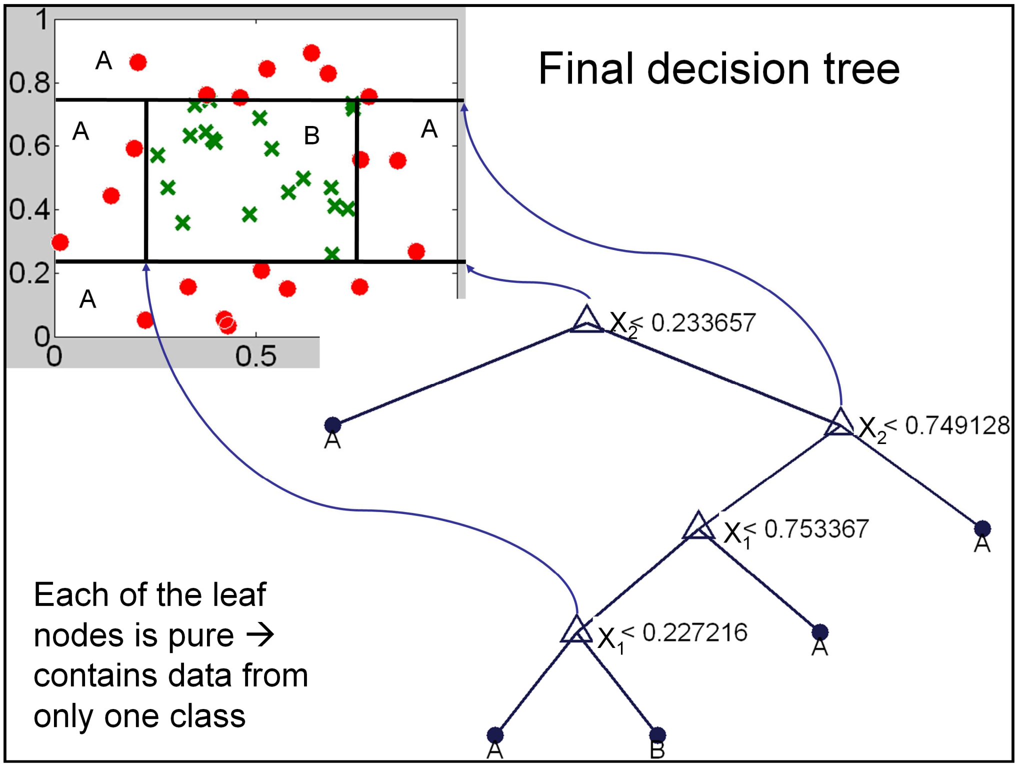

Best X_1 split: 0.22, IG = 0.182
Best X_2 split: 0.75, IG = 0.353
Split on X_2 with 0.75
X_2 < 0.233657
A
X_2 < 0.749128
Total data points = 26
6 A
20 B
Total data points = 7
7 A
0 B


There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output ‘A’
A
Total data points = 26
6 A
20 B
$X_2 <$ 0.749128
Total data points = 7
7 A
0 B
1
0.8
0.6
0.4
0.2
0
0
0.5
1

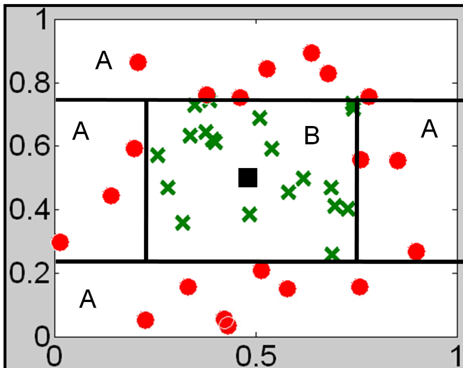


1
0.8
0.6
0.4
0.2
0
0
0.5
1
A
A
B
A
A

Final decision tree



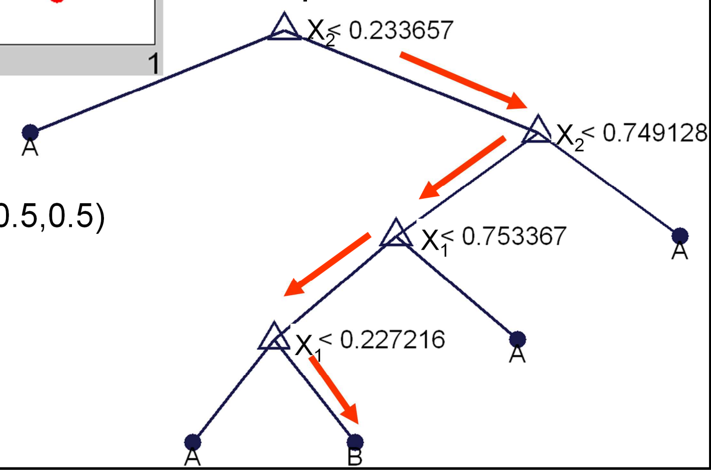
Each of the leaf nodes is pure → contains data from only one class



Final decision tree

Given an input (X,Y) → Follow the tree down to a leaf.

Return corresponding output class for this leaf



Example (X,Y) = (0.5,0.5)

Induction of Decision Trees

ID3 (*examples*, *attributes*)
if there are no *examples*
  then return *default*
  else
    if all *examples* are members of the same class
      then return the class
      else
        if there are no *attributes*
          then return Most_Common_Class (*examples*)
          else
            best_attribute ← **Choose_Best** (*examples*, *attributes*)
            root ← Create_Node_Test (*best_attribute*)
            for each value v_i of *best_attribute*
              $examples_{v_i}$ ← subset of *examples* with *best_attribute* = v_i
              subtree ← **ID3**($examples_{v_i}$, *attributes* - *best_attribute*)
              set *subtree* as a child of *root* with label v_i
return *root*

Basic Questions

- How to choose the attribute/value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

Comments on Tree Termination

- Zero entropy in data set, perfect classification
  - Type = thriller, 0+ 4-
  - Type = drama, Mood = stressed, 0+ 1-
  - Type = drama, Mood = relaxed, 1+ 0-
  - Type = drama, Mood = normal, Director = Bergman, 0+ 1-
  - Type = drama, Mood = normal, Director = Hitchcock, 1+ 0-
- No examples availables
  - Type = drama, Mood = normal, Director = Spielberg
- Indistinguishable data
  - Type = comedy, attribute Company:
    - ❖ m2, comedy, Columbia, Yes, and m10, comedy, Columbia, No
    - ❖ m3, comedy, MGM, No, and m5, comedy, MGM, Yes
  - Type = comedy, attribute Director:
    - ❖ m2, Spielberg, Yes, and m3, Spielberg, No
    - ❖ m5, Hitchcock, Yes, and m10, Spielberg, No
  - (Type = comedy, attribute Mood, could have split?).

Errors

Split labeled data D into *training* and *validation* sets

- Training set error - fraction of training examples for which the decision tree disagrees with the true classification

- Validation set error - fraction of testing examples - from given labeled examples - for which the decision tree disagrees with the true classification

Overfitting

Tree too specialized – "perfectly" fit the training data.

A tree T *overfits* the training data, if there is an alternate T' such that

- for training set, error with T < error with T'
- for complete D, error with T > error with T'

- Node Pruning
- Rule Post-pruning
- Cross validation

Reduced-Error Pruning

- Remove subtree, make node a leaf node, assign the most common classification of the examples of that node.
- Check validation set error
- Continue pruning until error does not increase with respect to the error of the unpruned tree
- Rule post-pruning: represent tree as set of rules; remove rules independently; check validation set error; stop with same criteria

Other Issues

- Attributes with many values
  - Information gain may select it.
  - Consider splitting value and use the ratio between Gain and the splitting value
- Attributes with cost
  - Use other metrics
- Attributes with missing values
  - Infer most common value from examples at node

Summary

- Inductive learning – supervised learning – classification
- Decision trees represent hypotheses
- DT learning driven by information gain heuristic
- Recursive algorithm to build decision tree
- Next class:
  - More on continuous values
  - Missing, noisy attributes
  - Overfitting, pruning
  - Different attribute selection criteria