Data Mining

Lecture 4
Decision Trees and Classification

http://www.informatik.uni-hamburg.de/WTM/

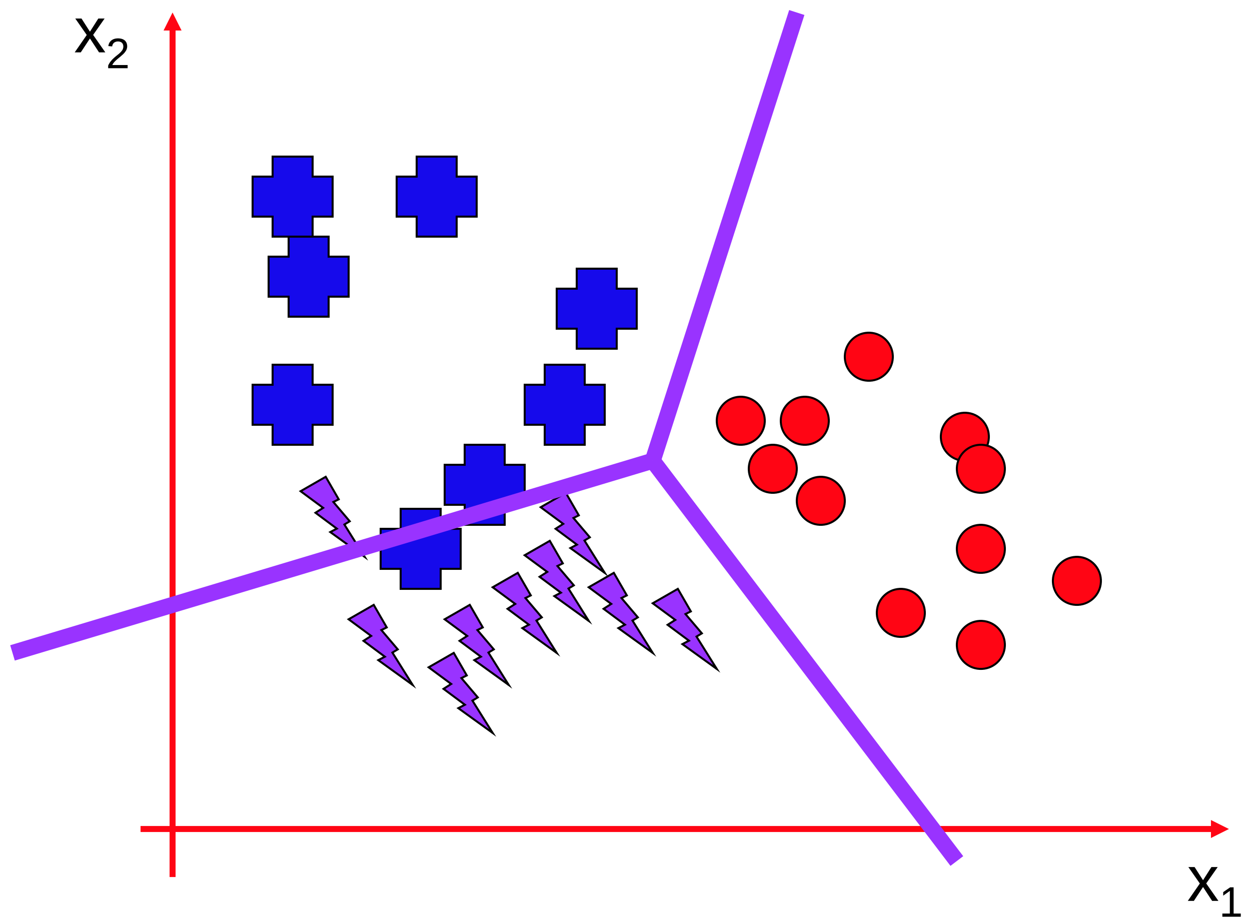
Overview

- Decision trees for classification
- Decision tree induction
- Criteria for attribute split
  - Information Gain
  - Gini Impurity
- Continuous attributes
- Gain ratio
- Missing values
- Pruning
- Limitations of decision trees

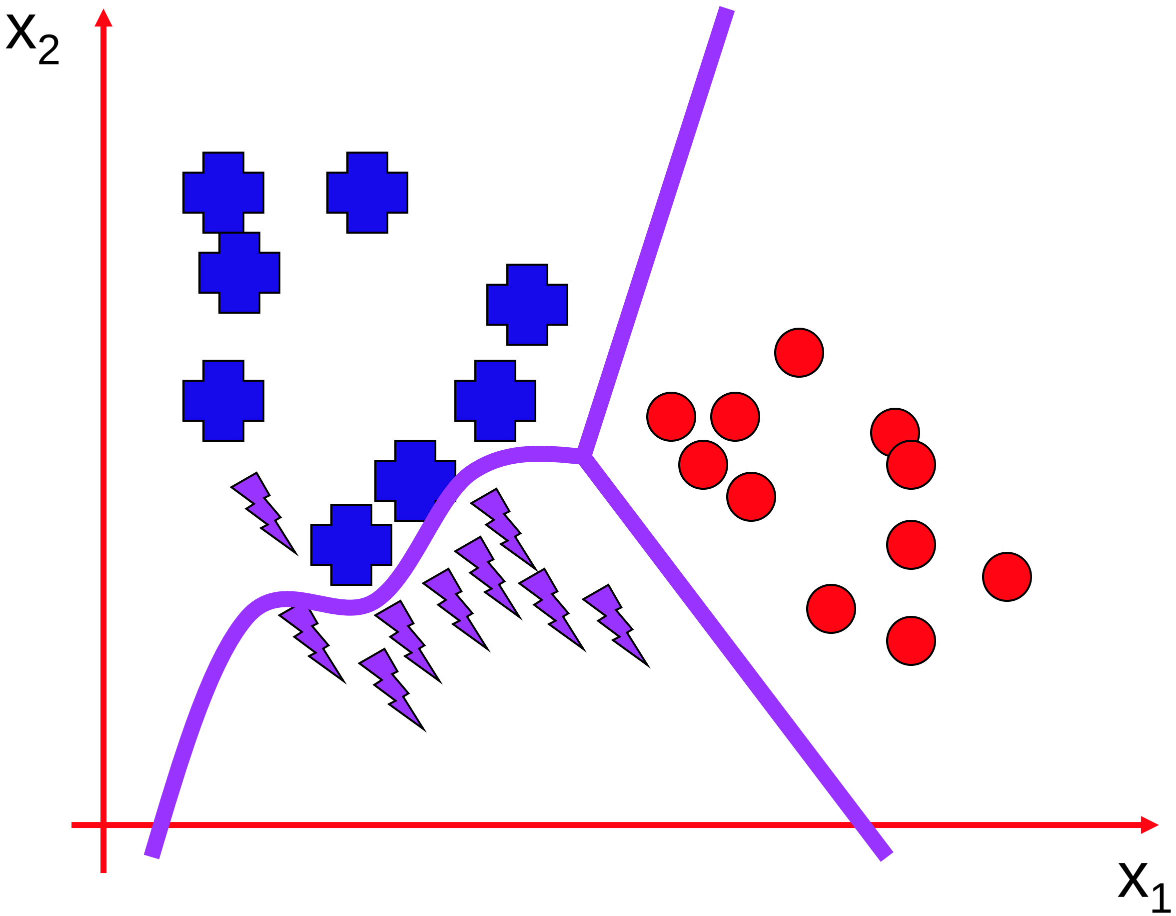
Motivation: Making decisions

Dan Gilbert: Why we make bad decisions,
TED talks,. Video online

Decision Boundaries



Decision Boundaries



History of Decision Trees

- 1966: Hunt, colleagues in psychology used full search decision tree methods to model human concept learning
- 1977: Breiman, Friedman, colleagues in statistics develop simultaneous Classification And Regression Trees (CART)
- 1986: Quinlan's landmark paper on ID3
- Late 1980s:Various improvements, i.e: coping with noise, continuous attributes, missing data, non-axis-parallel DTs
- 1993: Quinlan's updated algorithm, C4.5
- Towards 2000: Quinlan: More pruning, overfitting control heuristics (C5.0, etc.); combining DTs; incremental learning

Supervised vs. Unsupervised Learning (discrete outputs)

today

- **Supervised Learning (*Classification*)**
  - The training data (observations, measurements, etc.) are accompanied by ***categorical class labels*** (discrete or nominal)
  - New data is classified based on the training
- **Unsupervised Learning (*Clustering*)**
  - Given a set of measurements, observations, etc. of which the class labels are unknown
  - Aim: establishing the existence of clusters in the data

Supervised vs. Unsupervised Learning (continuous outputs)

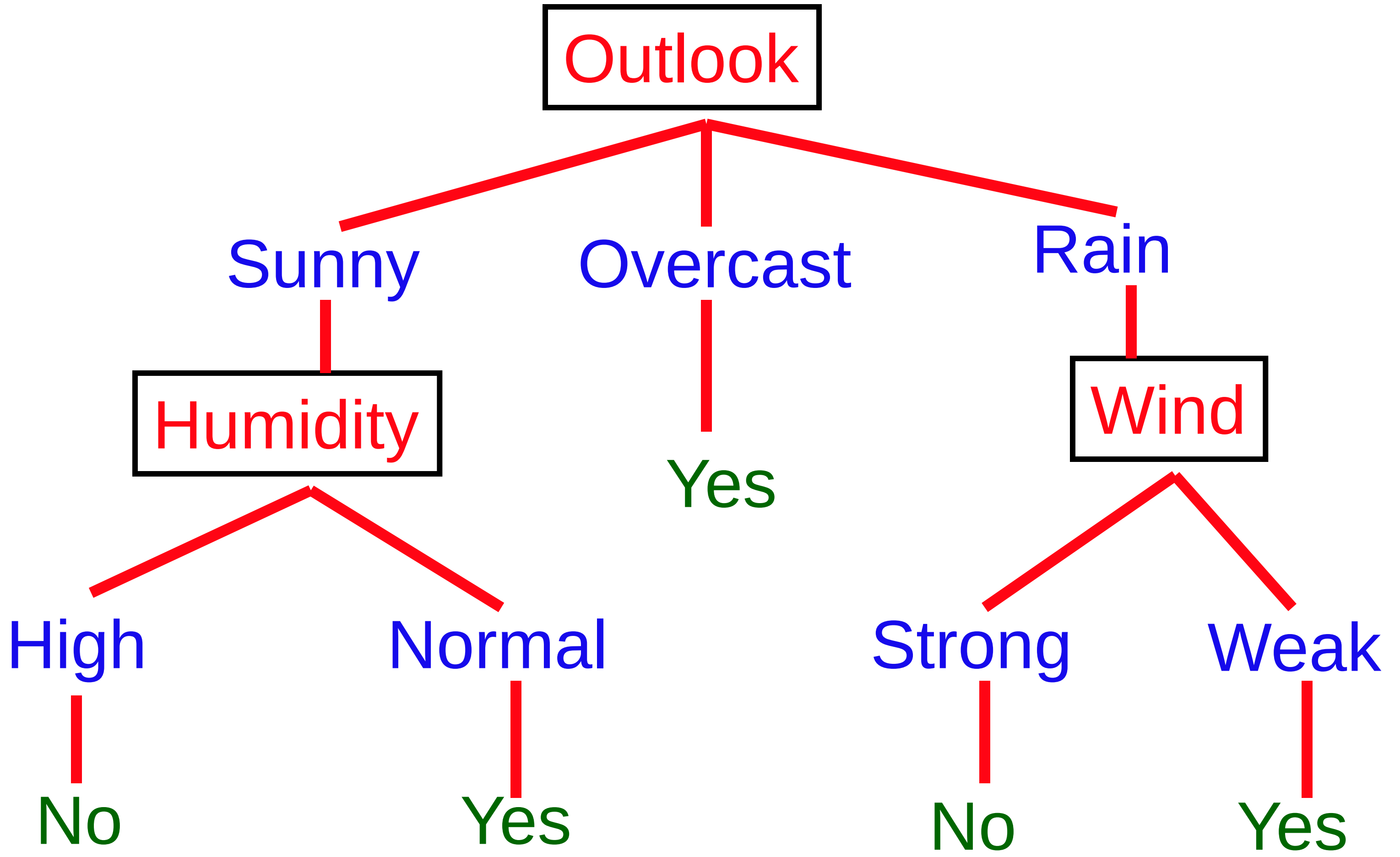
- **Supervised Learning (Regress*ion*)**
  - The training data (observations, measurements, etc.) are accompanied by ***continuous output values***
  - For new data where output values are missing, “predict” the most likely output values based on the training
- **Unsupervised Learning (*general case*)**
  - Given a set of measurements, observations, etc. of which the class labels are unknown
  - Aim: represent the data in another form
    - E.g. compressed via PCA, …

Decision Trees

- Split ***classification*** into a series of choices about features in turn
- Lay them out in a tree
- Progress down the tree to the leaves

Example: Anyone for Tennis?

Decide whether to play tennis based on the weather

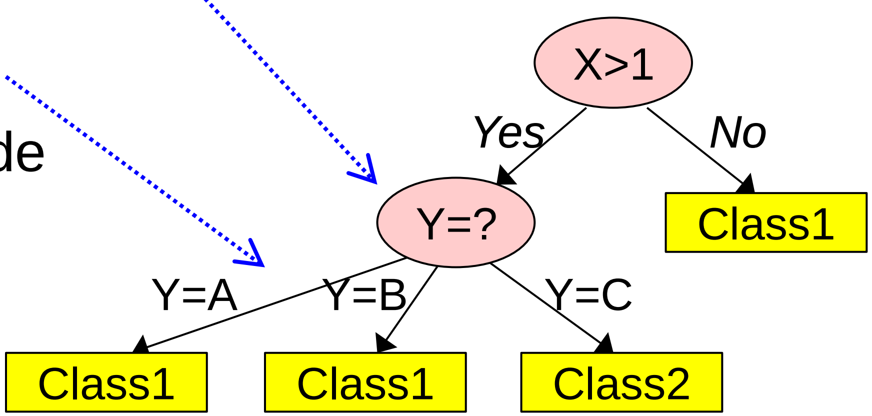


Rules and Decision Trees

- Tree can be turned into a set of rules:
  - if (outlook = sunny & humidity = normal) | (outlook = overcast)
    | (outlook = rain & wind = weak)
    then
    play tennis
- How do we generate the trees?
  - Need to choose features / attributes
  - Need to choose order of features / attributes

Decision Trees

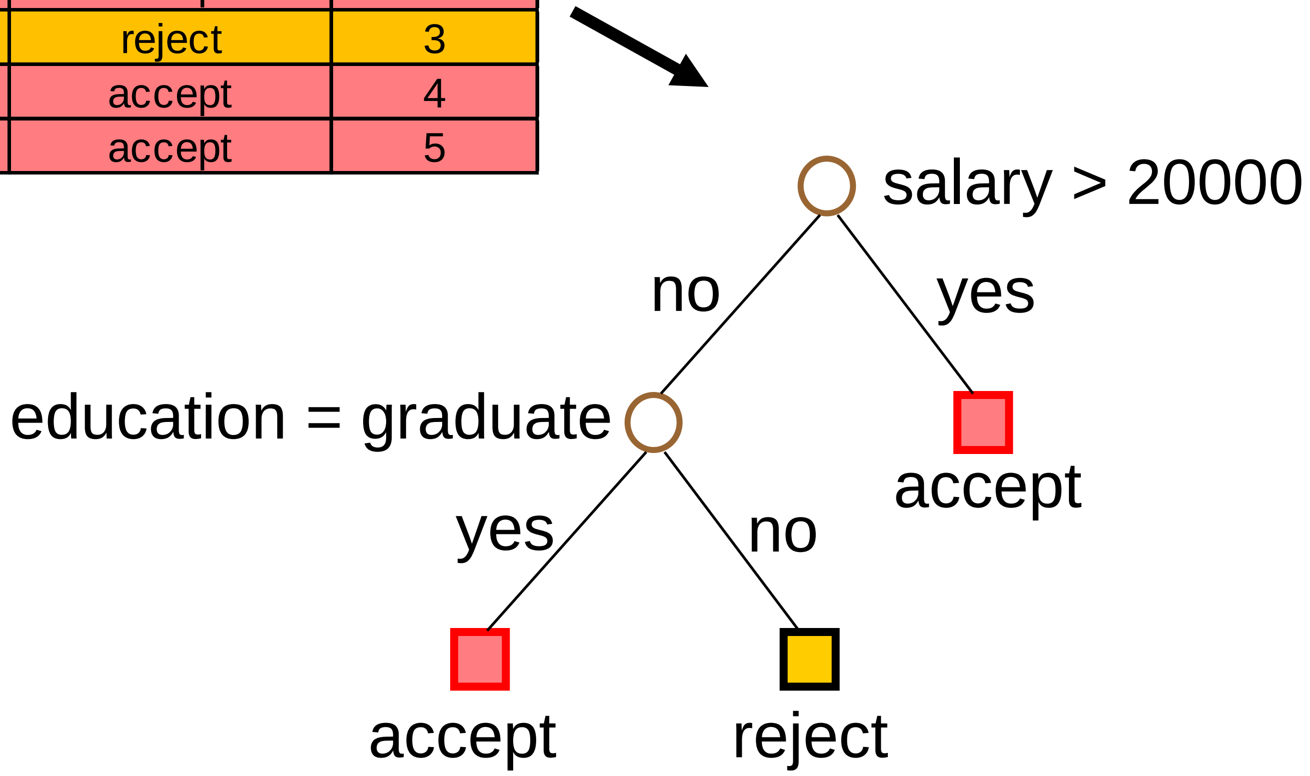
- Efficient method for producing ***classifiers*** from data
  - ***Supervised learning*** methods that construct decision trees from a set of input-output samples.
  - Guarantees that a simple tree is found
    - but not necessarily the simplest one
- Consists of
  - ***Nodes*** that are tests on the attributes
  - Outgoing ***branches*** of a node correspond to all the possible outcomes of the test at the node
  - ***Leaves*** that are sets of samples belonging to the same class



Example of Decision Tree for Credit Approval

Credit Analysis

salary	education	label	#
10000	high school	reject	1
40000	under graduate	accept	2
18000	under graduate	reject	3
75000	graduate	accept	4
15000	graduate	accept	5



Decision Tree for Classification

- Given:
  - Database of ***samples***, each assigned a ***class label***.

- Task: Develop a ***model***/profile for each class:
  - **Example profile** (good credit):

    (salary > 20k) or (salary <= 20k and education = graduate)
    => Credit = Good (approved)

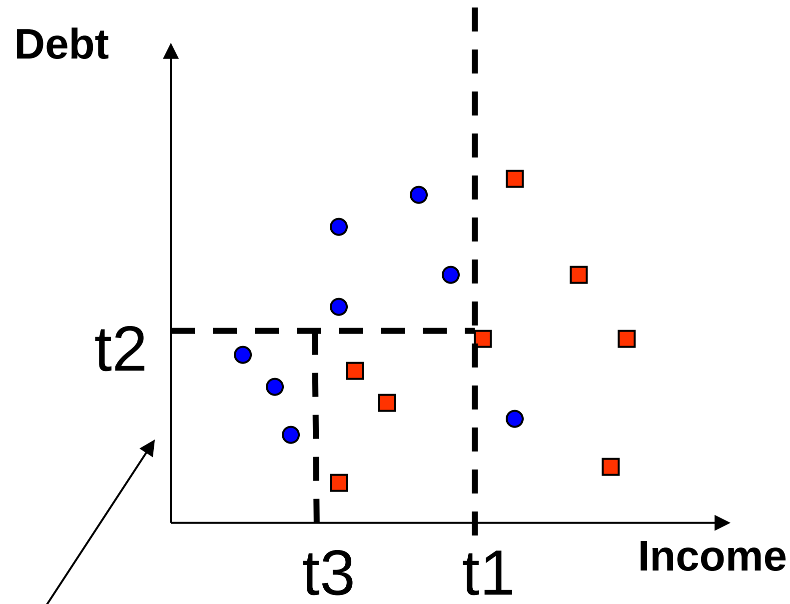
Classification by Decision Tree Induction

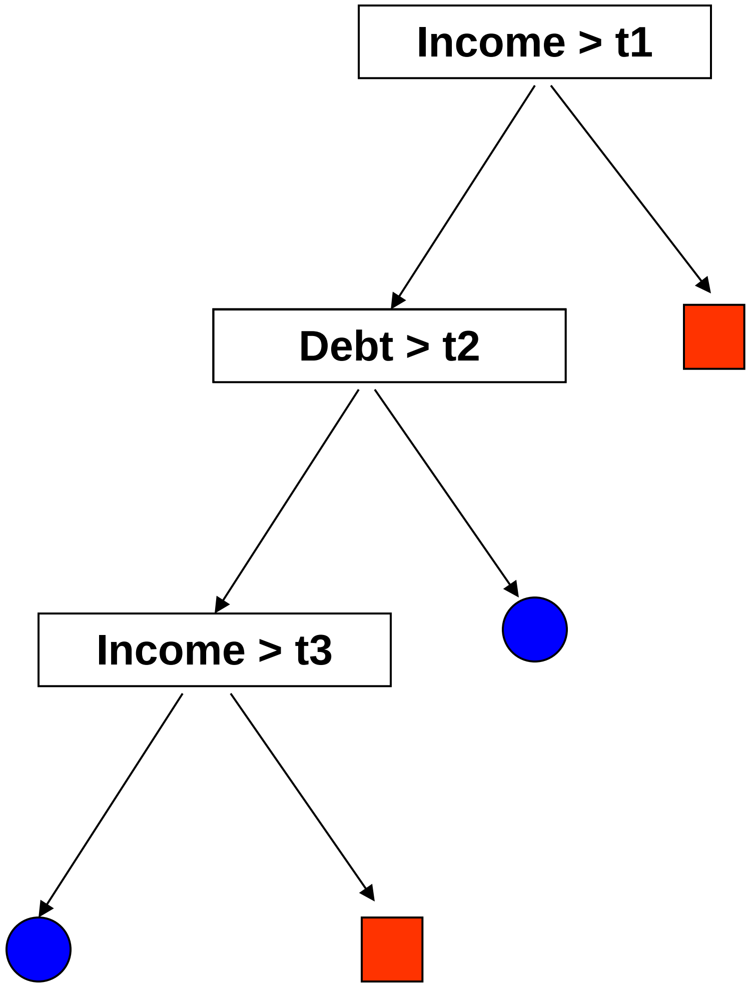
- Decision tree ***generation*** consists of two phases:

  1. Tree ***construction***:
     - At start, all the training examples are at the root.
     - Partition the examples recursively based on selected attributes.
  2. Tree ***pruning***:
     - Identify and remove branches that reflect noise or outliers.

- Decision tree ***use***: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Decision Tree: Example





Note: tree boundaries are piecewise linear and axis-parallel

Are all correctly classified?

Classification – a Two-Step Process

- ***Model construction***: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the ***class label attribute***
  - The set of tuples used for model construction is the ***training set***
  - The model is represented as decision trees, classification rules, or mathematical formulae (e.g. neural network)
- ***Model usage***: for classifying future or unknown objects
  - Evaluate the model
    - The known label of test sample is compared with the classified result from the model
    - E.g., ***accuracy*** rate is the percentage of test set samples that are correctly classified by the model
    - ***Test set*** is independent of training set (check for overfitting)
  - If the accuracy is acceptable, use the model to ***classify data*** tuples whose class labels are not known

Process (1): Model Construction

Training Data

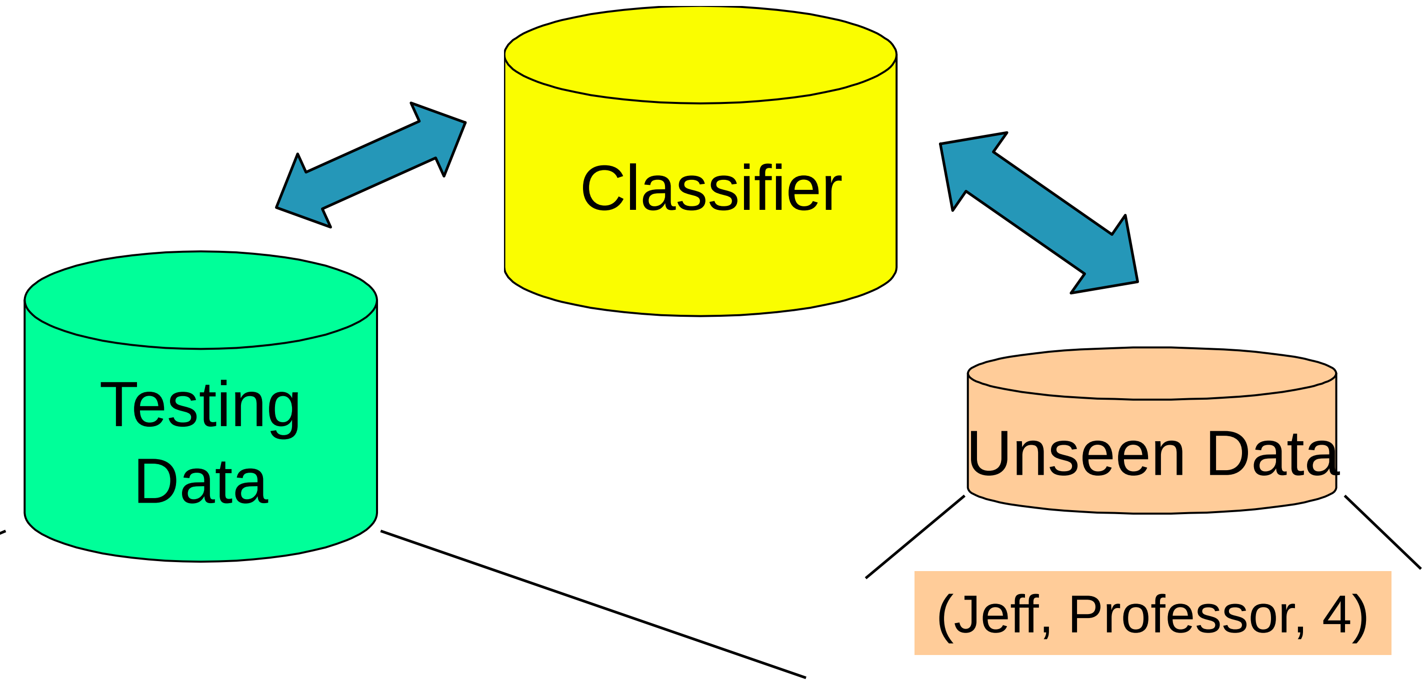
Classification Algorithms

Classifier (Model)

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	yes
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

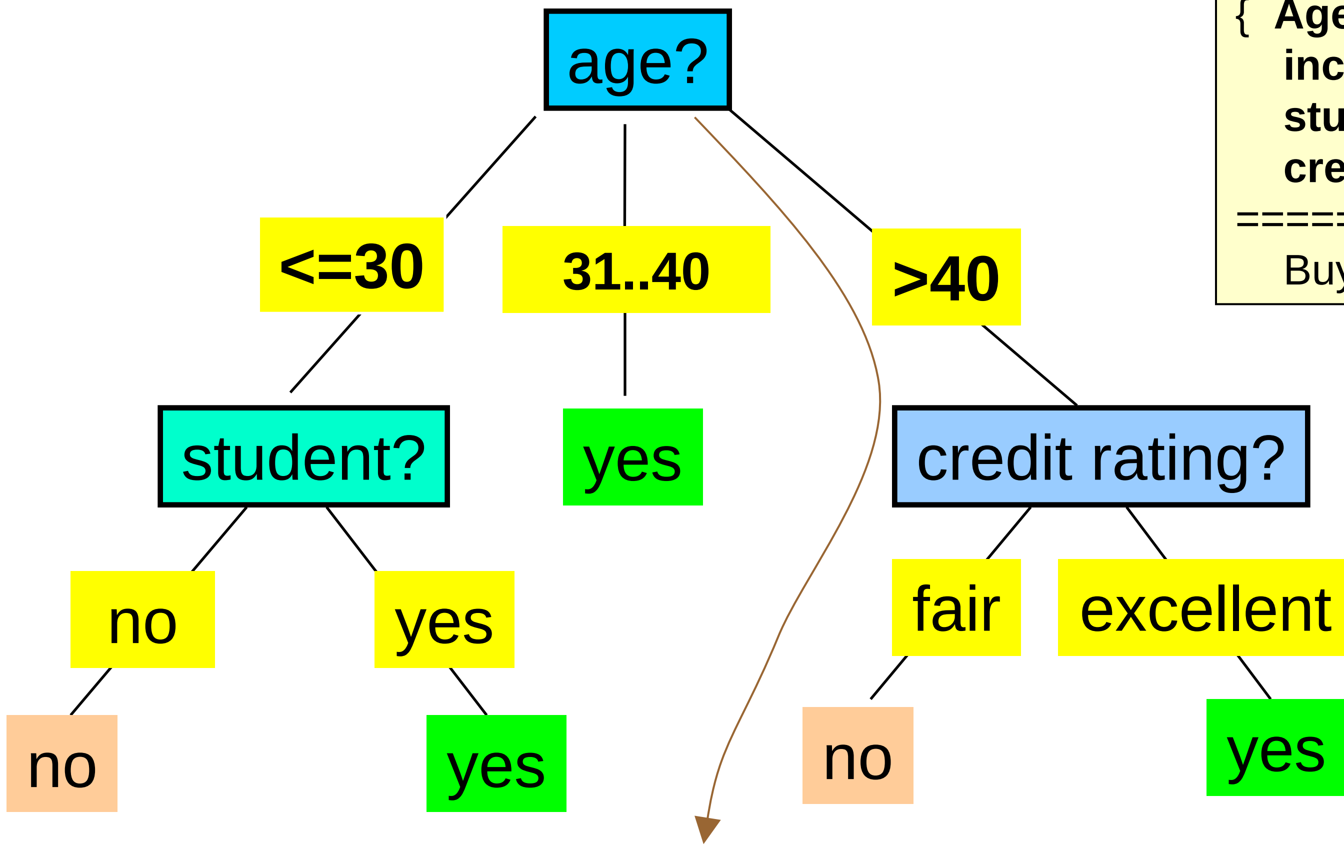
Yes!

Decision Tree Induction: Training Dataset

This follows an example of Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



Classify new sample:

{ **Age = 45** and
income = low and
student = no and
credit = fair}
=================
Buy computer = **?**

Answer (class):

NO

Decision Tree

- Requirements for a Decision Tree algorithm:
  1. Consistent attribute-value description for all data
  2. Predefined classes
  3. Discrete classes
  4. Sufficient data
  5. "Logical" classification models (not weighted decisions)

- Pros
  - ***Fast*** execution time
  - Generated trees (rules) are ***easy to interpret*** by humans
  - ***Scale well*** for large data sets
  - Can handle high dim. data

- Cons
  - Cannot capture ***correlations*** among attributes
  - Consider only ***axis-parallel*** cuts

Decision Tree Algorithms

- Classifiers from machine learning and statistical community:
  - ID3
  - C4.5 [Quinlan 93] $\rightarrow$ C5.0
  - CART (as an advance in applied statistics)

- Classifiers for large databases:
  - SLIQ, SPRINT
  - SONAR
  - Rainforest

- Aspects are quality of the tree, scalability, and memory use.

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a ***top-down recursive divide-and-conquer manner***
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., ***information gain***)
- Conditions for **stopping partitioning**
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning
    - ***majority voting*** is employed for classifying the leaf
  - There are no samples left

Decision Tree Algorithms: First Splitting

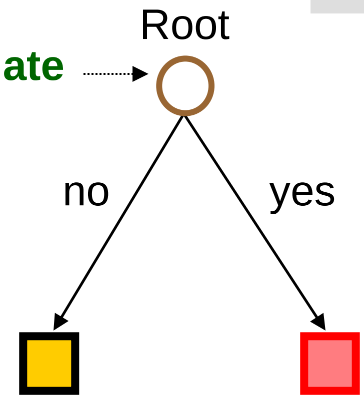
education

high school	reject	1
under graduate	accept	2
under graduate	reject	3
graduate	accept	4
graduate	accept	5

salary

10000	reject	1
15000	accept	5
18000	reject	3
40000	accept	2
75000	accept	4

education = graduate



high-school	reject	1
under-graduate	accept	2
under-graduate	reject	3

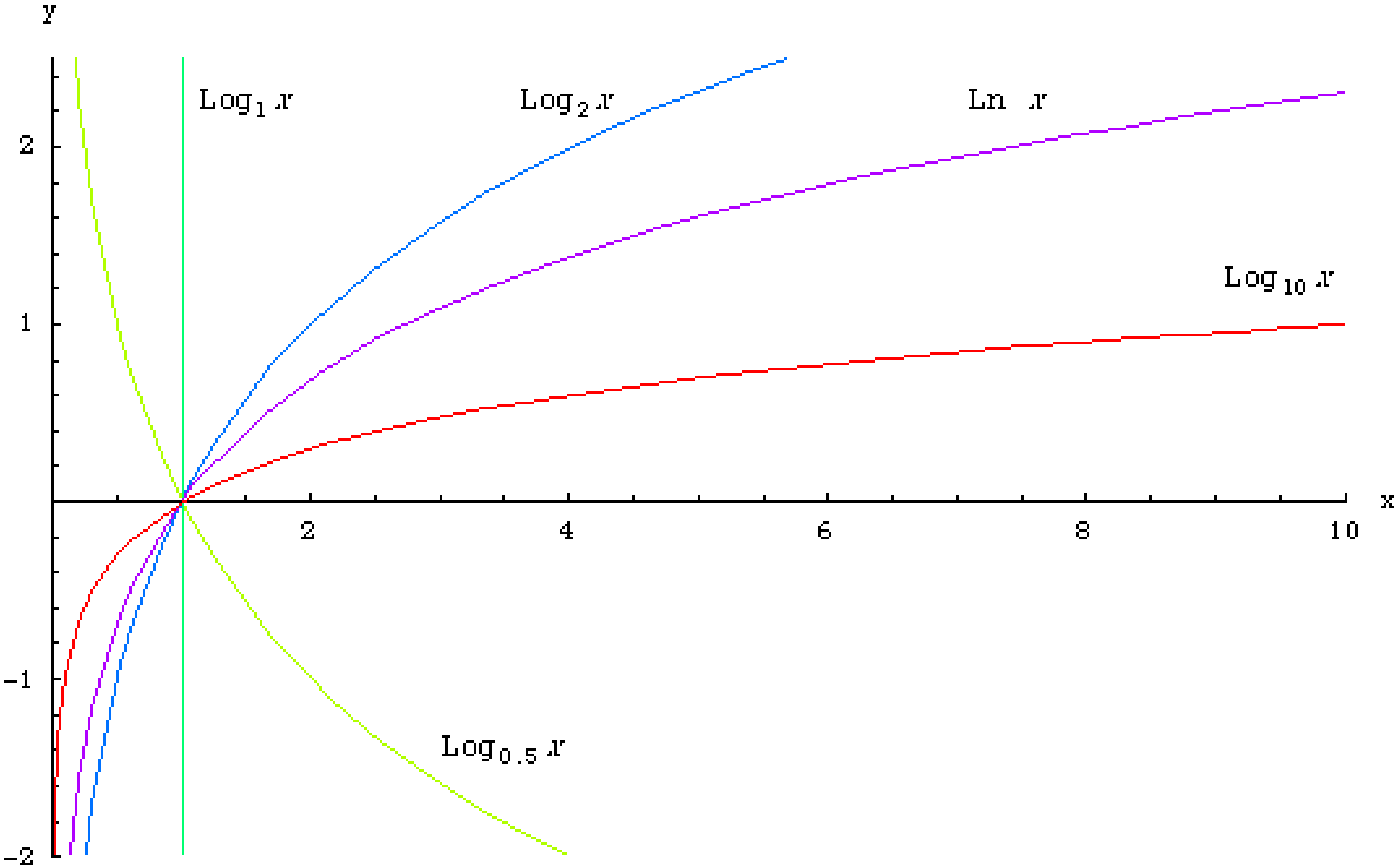
10000	reject	1
40000	accept	2
18000	reject	3

graduate	accept	4
graduate	accept	5

75000	accept	4
15000	accept	5

we did not explain how we selected “education” attribute for splitting

Reminder…$\log_2 p$



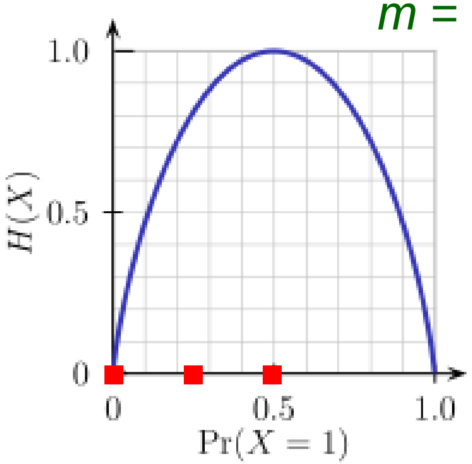
Brief Review of Entropy

- Entropy (Information Theory)
  - ***Measure of uncertainty*** associated with a random variable
    - Higher entropy ⇨ higher uncertainty
    - Lower entropy ⇨ lower uncertainty
  - Calculation: For a discrete random variable Y taking *m* distinct values $\{y_1,\ldots,y_m\}$, where $p_i=P(Y=y_i)$

$$H(Y) = \underbrace{\sum_{i=1}^{m} p_i}_{\text{weighted average over the classes}} \cdot \log(\underbrace{\frac{1}{p_i}}_{\text{surprise}}) = -\sum_{i=1}^{m} p_i \cdot \log(p_i)$$

Brief Review of Entropy

for
$m = 2$



$$H(Y) = \sum_{i=1}^{m} p_i \cdot \log(\frac{1}{p_i}) = -\sum_{i=1}^{m} p_i \cdot \log(p_i)$$

Three examples for m=2 classes:

- $p_1=0, p_2=1$

$$H(Y) = -0 \cdot \log(0) - 1 \cdot \log(1) = -0 \cdot (-\infty_{small}) - 1 \cdot 0 = 0$$

- $p_1=0.25, p_2=0.75$

$$H(Y) = -0.25 \cdot \log(0.25) - 0.75 \cdot \log(0.75) = -0.25 \cdot (-2) - 0.75 \cdot (-0.415)$$
$$= 0.811$$

- $p_1=0.5, p_2=0.5$

$$H(Y) = -0.5 \cdot \log(0.5) - 0.5 \cdot \log(0.5) = -0.5 \cdot (-1) - 0.5 \cdot (-1) = 1$$

Select the Attribute with highest Information Gain (ID3/C4.5)

- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, (m classes) estimated by $|C_{i,D}|/|D|$
- ***Information*** (entropy) to classify a tuple ***in D***:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- ***Average information*** needed, after using attribute A to split D into ***k partitions***:

$$Info_A(D) = \underbrace{\sum_{j=1}^{k} \frac{|D_j|}{|D|}}_{\text{weighted average over the partitions}} \cdot \underbrace{Info(D_j)}_{\text{entropy in partition}}$$

- ***Information gained*** by branching on attribute A:

$$Gain(A) = Info(D) - Info_A(D)$$

Information Gain – Example

- Class "buys_computer =yes" (9x)
- Class "buys_computer =no" (5x)

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$
$$= 0.94$$

$$Info_{age}(D) = \boxed{\frac{5}{14} I(2,3)} + \underbrace{\frac{4}{14} I(4,0)}_{0} + \frac{5}{14} I(3,2)$$
$$= 0.694$$

age	yes_i	no_i	$I(yes_i, no_i)$
<=30	2	3	0,971
31…40	4	0	0
>40	3	2	0,971

"age <=30" has 5 out of 14 samples, with 2 "yes" and 3 "no"

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Gain(age) = Info(D) - Info_{age}(D)$$
$$= \underline{0.246}$$

- Information gains for other splits:

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Decision Tree Algorithm

- Key idea: ***Recursive Partitioning***
  - Take all of your data.
  - Consider *all* possible values of *all* variables.
  - Select the variable/value (X=t_1) that produces the greatest ***separation*** in the target.
    - (X=t_1) is called a "split".
  - If $X < t_1$ then send data point to the "left" branch, otherwise, send data point to the "right" branch.
  - Now repeat same process on these two "nodes"

→ You get a "tree"

- Note: CART only uses *binary* splits.

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the **best split point** for A
  - Sort the value A in increasing order
  - Typically, the ***midpoint*** between a pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Attribute Selection Measure Comparison

- ***Information gain*** (ID3/C4.5)
  - All attributes are assumed to be categorical.
    - Can be modified for continuous-valued attributes.

- ***Gini impurity*** (IBM Intelligent Miner, CART, SLIQ, SPRINT)
  - All attributes are assumed continuous-valued.
    - Can be modified for categorical attributes.
  - Assume there exist several possible split values for each attribute.

Gini

- Corrado Gini, Italian statistician, 1884-1965
- ***Gini coefficient***
  - Used to show inequality of income distribution in a population
  - Large if unequal incomes,
    small if equal incomes
- ***Gini impurity***
  - A measure for the distribution of labels in a set
  - Large if many equally distributed labels,
    small if large probability only for few labels
- Hence, Gini index ≠ Gini coefficient
  - Attention: Both sometimes referred to as “Gini index”

Attribute Selection using *Gini Impurity*

- A data set *D* contains examples from *m* classes, and
- p_j is the relative frequency of class *j* in *D.*
- Then Gini impurity, *Gini*(*D*), is defined as:

$$Gini(D) = \underbrace{\sum_{j=1}^{m} p_j}_{\text{weighted average over the classes}} \underbrace{(1 - p_j)}_{\text{probability of incorrect labeling}} = 1 - \sum_{j=1}^{m} p_j^{\,2}$$

- Gini measures how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

- Should be minimized!

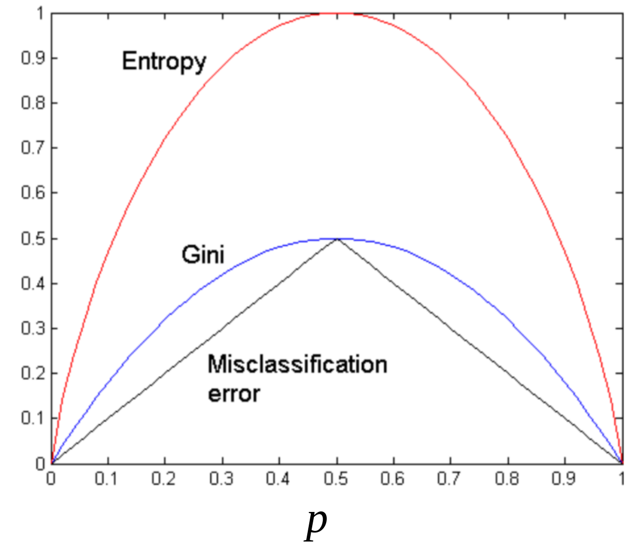
- Note: while $\Sigma_i\, p_i = 1$ always, not so $\Sigma_i\, p_i^2$

Attribute Selection using *Gini Impurity*

$$Gini(D) = \sum_{j=1}^{m} p_j(1 - p_j) = 1 - \sum_{j=1}^{m} p_j^{\,2}$$

- Minimum (1-1=0) when all records belong to one class
  → most interesting information for a split
- Maximum (1 - 1/*m*) when records are *equally distributed* among all classes
  → least interesting information

Gini impurity for *m*=2 classes



Splitting Based on *Gini* Impurity

- When a node p is split into k partitions, the ***quality*** of split is computed as

$$Gini_{\text{split}} = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

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

- Interpretation: weighted sum of *Gini* impurity for subsets i of samples caused by splitting
- If a data set D is split into two subsets D_1 and D_2 with sizes n_1 and n_2 respectively, the *Gini* impurity *Gini*(D) is defined as

$$Gini_{\text{split}}(D) = \frac{n_1}{n} Gini(D_1) + \frac{n_2}{n} Gini(D_2)$$

Splitting Based on *Gini* Impurity

- Need to enumerate all possible splitting points for each attribute
- The attribute that provides the smallest $Gini_{split}(D)$ is chosen to split the node

Gini Impurity – Example

- Class "buys_computer =yes" (9x)
- Class "buys_computer =no" (5x)

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

$$= 0.459$$

$$Gini_{age}(D) = \boxed{\frac{5}{14} Gini(2{,}3)} + \frac{4}{14} \underbrace{Gini(4{,}0)}_{0} + \frac{5}{14} Gini(3{,}2)$$

$$= 0.343$$

age	yes_i	no_i	$Gini(yes_i,no_i)$
<=30	2	3	0,48
31…40	4	0	0
>40	3	2	0,48

"age <=30" has 5 out of 14 samples, with 2 "yes" and 3 "no"

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- Compute Gini for other splits:

$$Gini(income) = \ldots$$

$$Gini(student) = \ldots$$

$$Gini(credit\_rating) = \ldots$$

- Consider also other splits, e.g. age {<=30 & 31…40} and {>40}
- Split at lowest value

Examples for Computing Gini (for *m*=2 classes)

- Gini Impurity for a given node t: $Gini(t) = 1 - \sum_{j=1}^{m} p(j \mid t)^2$

$p(j \mid t)$ is the *relative frequency* of class *j* at node *t*

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 **= 0**

Minimum

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)^2 – (5/6)^2 **= 0.278**

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)^2 – (4/6)^2 **= 0.444**

C1	**3**
C2	**3**

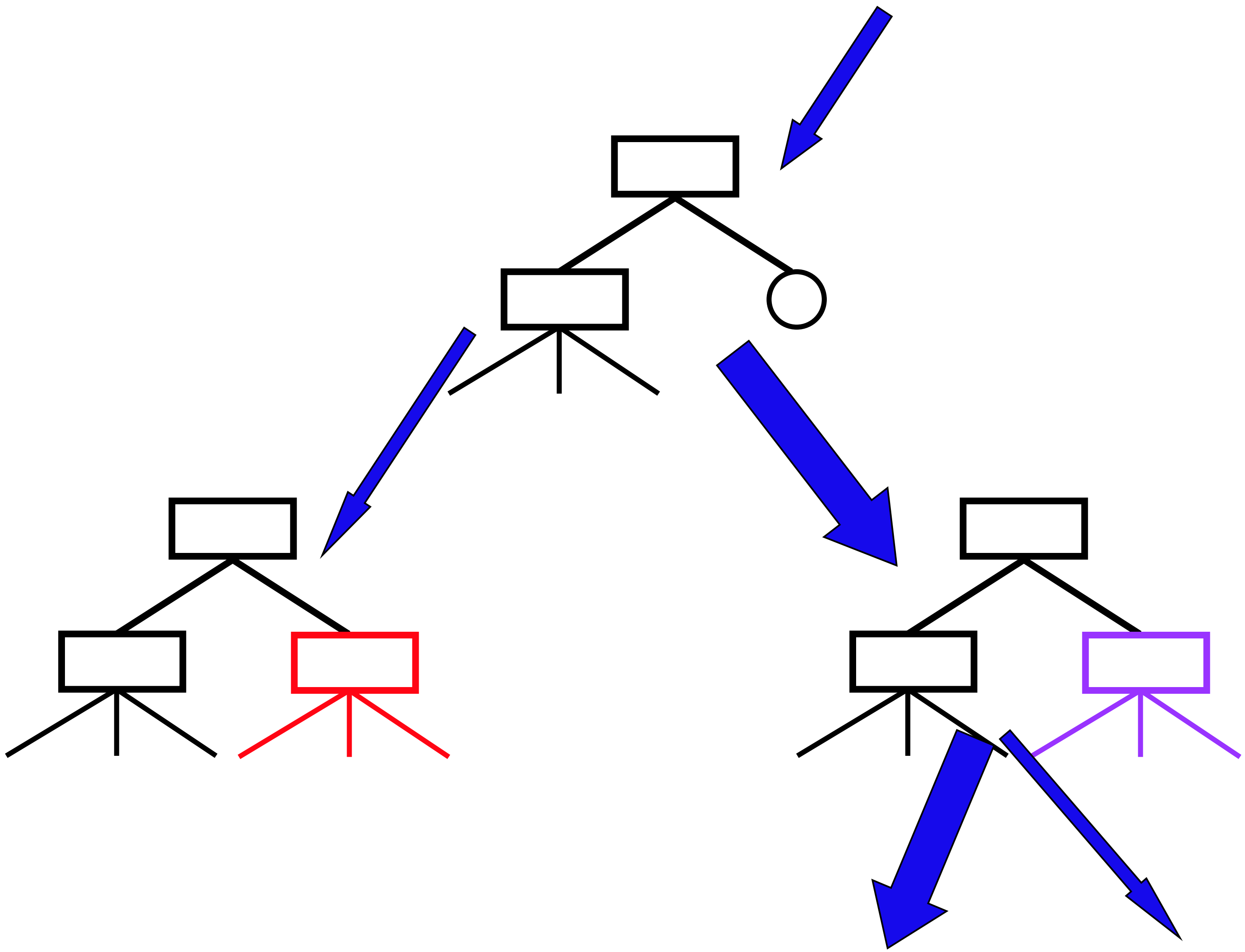
P(C1) = 3/6 P(C2) = 3/6

Gini = 1 – (3/6)^2 – (3/6)^2 **= 0.5** Maximum

Making decisions – Errors in value
Fresher after lunch

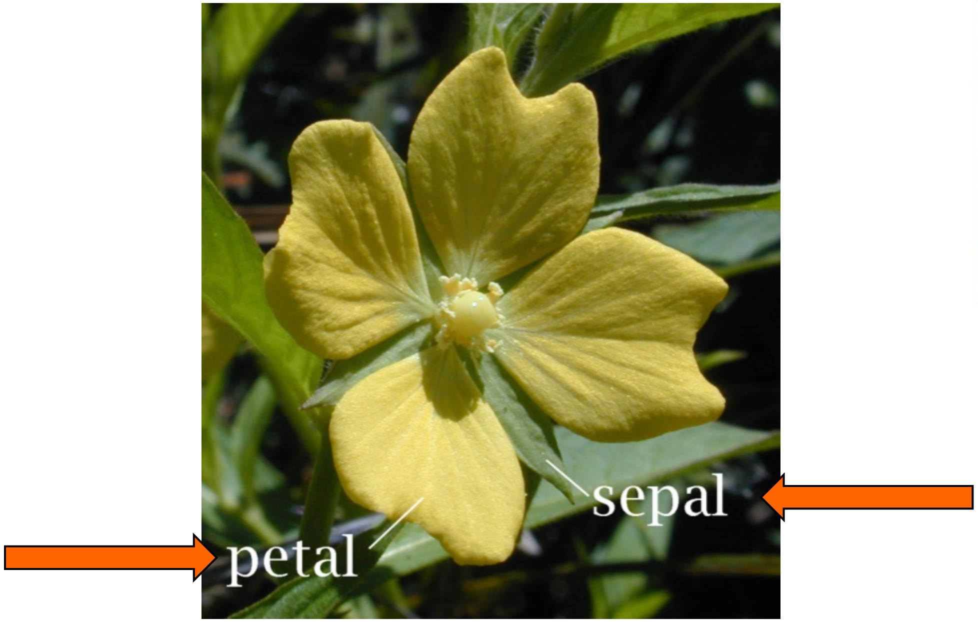
Dan Gilbert: Why we make bad decisions,
TED talks   Video online

Search



Matlab Demo of Decision Tree

- In Matlab, t = classregtree(X,y,'*Name*',*value*) creates a decision tree.
- **Example**: Create a classification tree for Fisher's iris data, a typical test case for many classification techniques.



Iris setosa

Iris versicolor

Iris virginica

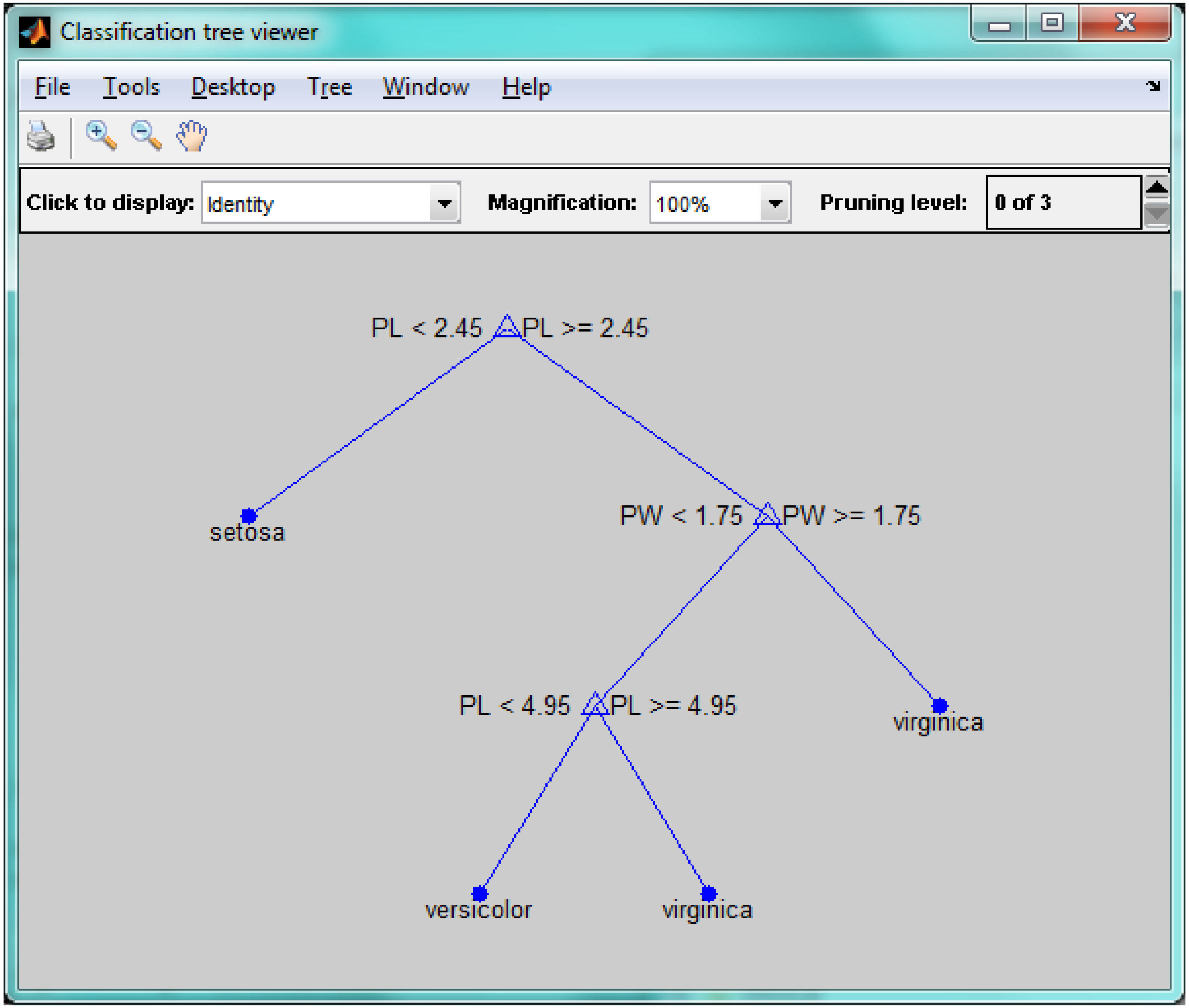
Matlab Demo of Decision Tree

- In Matlab, t = classregtree(X,y,'*Name*',*value*)   creates a decision tree.
- **Example**: Create a classification tree for Fisher's iris data, a typical test case for many classification techniques.
  - In this data set, four attributes (Sepal Length, Sepal Width, Petal Length and Petal Width) are considered in order to distinguish three species of flowers (*Iris setosa*, *Iris virginica* and *Iris versicolor*).
  - Commands:

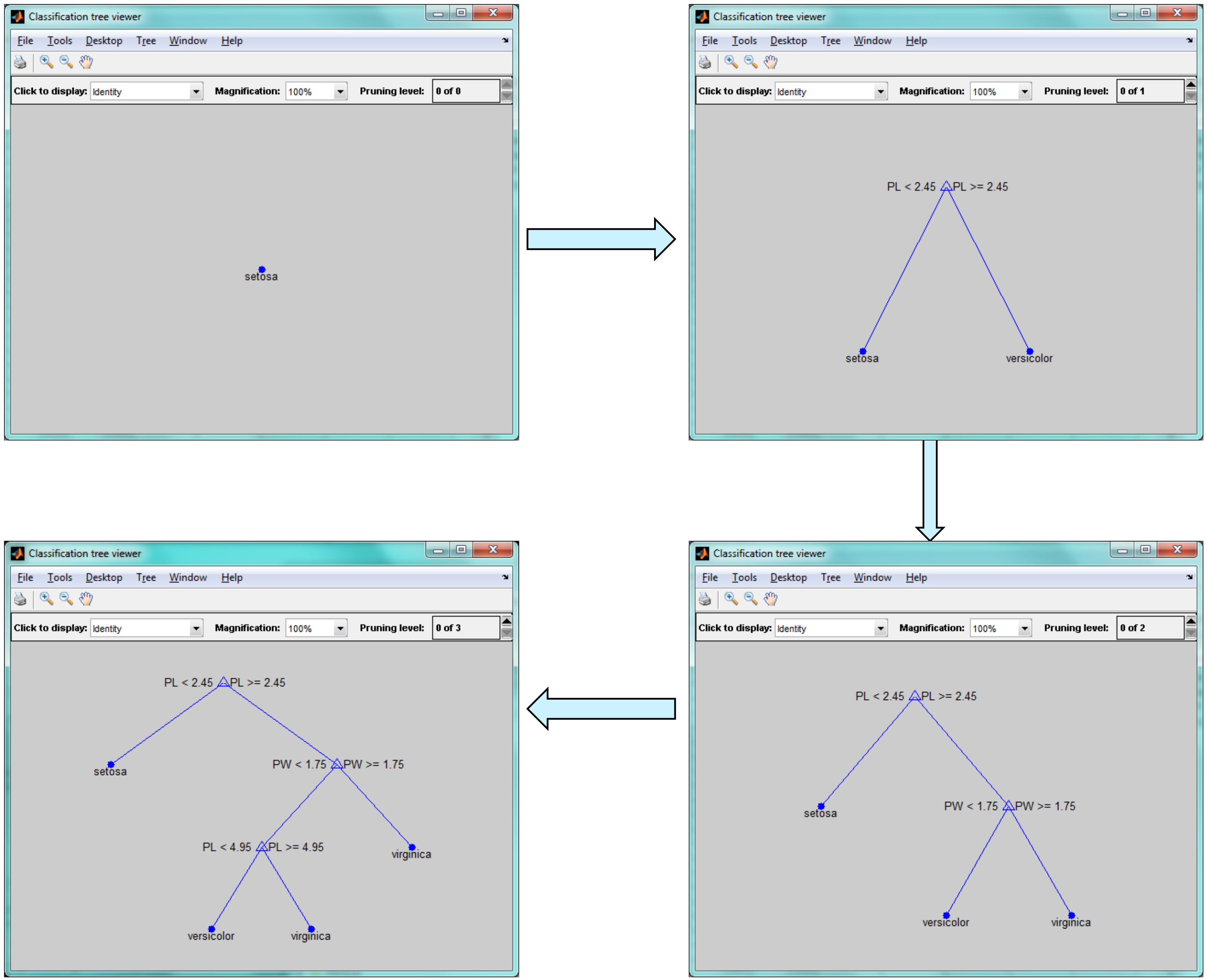
```
load fisheriris;
t = classregtree(meas,species,... 'names',{'SL'
'SW' 'PL' 'PW'});
```

  - Program generates a decision tree based on the data set.

Matlab Demo of Decision Tree



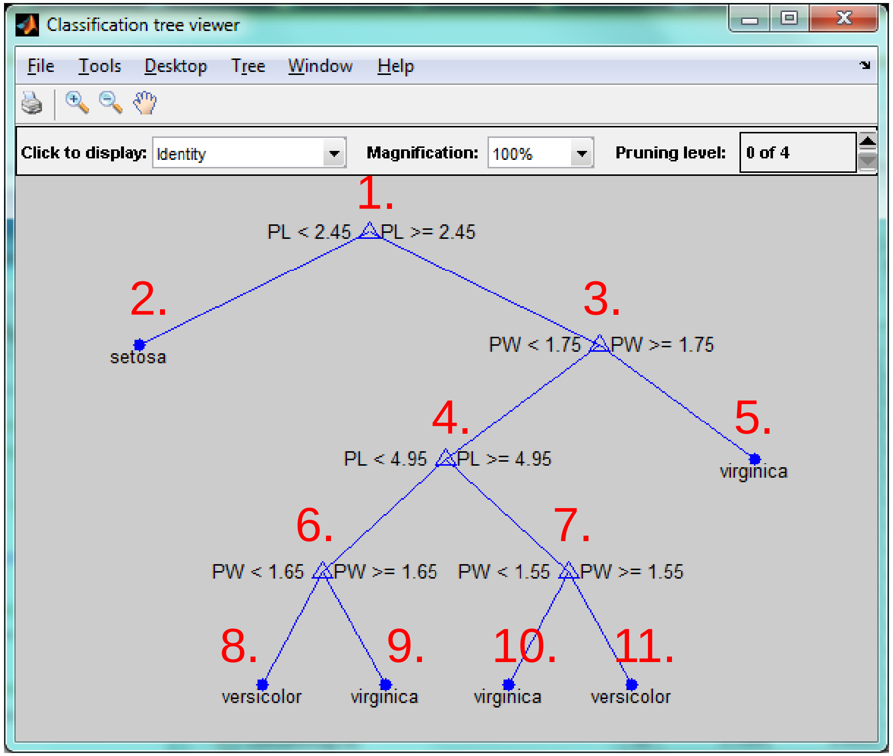
Matlab Demo of Decision Tree



Matlab Demo of Decision Tree

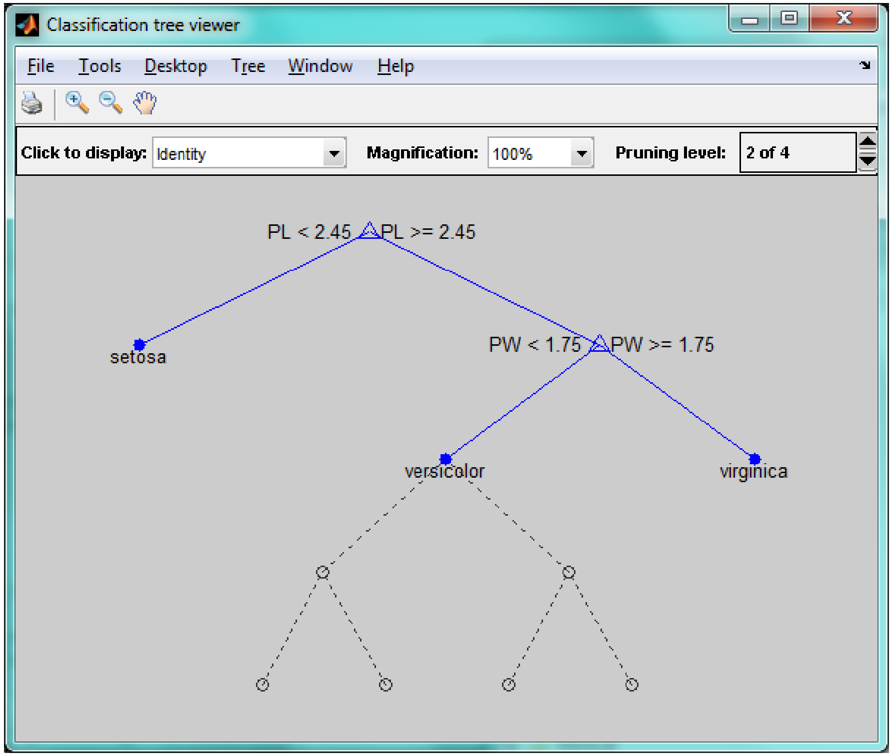
Final Decision tree for classification

1. if PL<2.45 then node 2 elseif PL>=2.45 then node 3
2. class = setosa
3. if PW<1.75 then node 4 elseif PW>=1.75 then node 5
4. if PL<4.95 then node 6 elseif PL>=4.95 then node 7
5. class = virginica
6. if PW<1.65 then node 8 elseif PW>=1.65 then node 9
7. if PW<1.55 then node 10 elseif PW>=1.55 then node 11
8. class = versicolor
9. class = virginica
10. class = virginica
11. class = versicolor



Matlab Demo of Decision Tree

- We can also prune the tree to avoid overfitting
- tt = prune(t,'level',2)



Advanced Decision Trees

- C4.5
  - Improved handling of continuous variables
  - C source code available
- C5
  - Quinlan made further improvements (boosting)
  - Many commercial data mining packages use the C5 algorithm
  - Source code available at a cost!
- CART
  - Breiman et al (Classification & regression trees, 1984)
  - similar to C4.5, boosting & bagging the data sets

Decision Tree Algorithms – C4.5

- ***Recursive building*** tree phase:

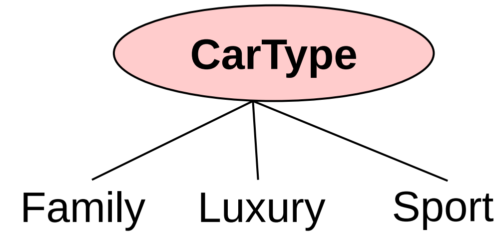
```
1. Initialize root node of tree.
2. while a node N that can be split:
3.     for each attribute A, evaluate splits on A,
4.     use best split to split N.
```

- Use ***entropy (information gain)*** to find best split

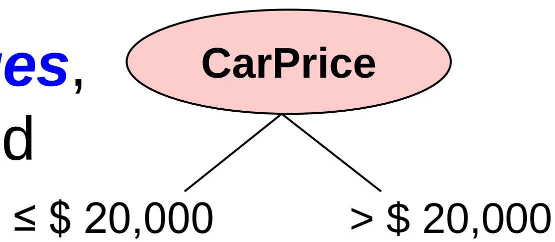
- Separate attribute lists maintained in each node of tree

C4.5 – Possible Mechanisms for Tests

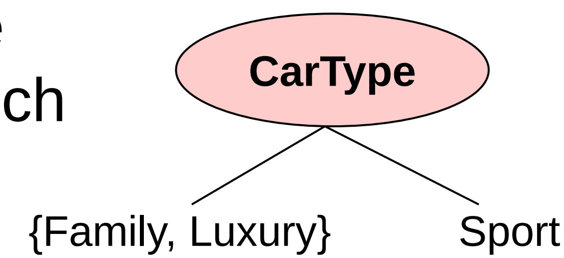
a. "standard" test on a ***discrete attribute***: one branch for each possible value of that attribute



b. If attribute Y has ***continuous numeric values***, binary test with outcomes $Y \leq Z$ and $Y > Z$ could be defined



c. possible values are allocated to a variable number of ***groups*** with one outcome/branch for each group



New example (1) Threshold Finding with Gain

Sometimes we have to find the threshold and the attribute

Database D

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
B	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

Attribute 2:

- After a sorting process, the set of values is: {65, 70, 75, 78, 80, 85, 90, 95, 96},
- … the set of potential threshold values *Z* is: {65, 70, 75, 78, 80, 85, 90, 95}.
- The optimal *Z* value is *Z*=80 (highest Inf. Gain)

9 are ≤ 80 … *7 are* Class**1** *2 are* Class**2**

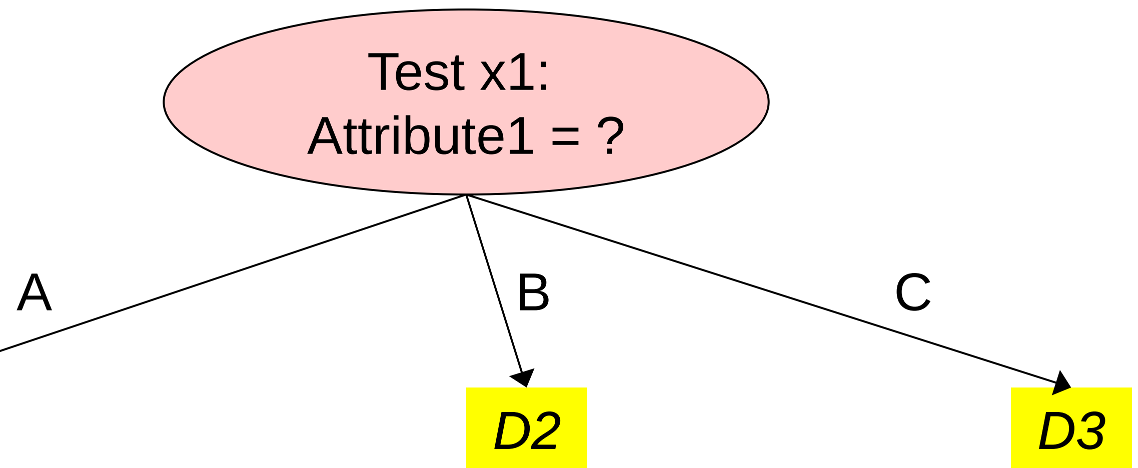
5 are > 80

- $\text{Info}_{Z=80}(D) = 9/14\cdot(-\,7/9\cdot\log_2(7/9) - 2/9\cdot\log_2(2/9)) + 5/14\cdot(-\,2/5\cdot\log_2(2/5) - 3/5\cdot\log_2(3/5)) = 0.837$ bits
- Gain(*Z=80*) = 0.940 – 0.837 = 0.103 bits

However, Attribute 1 gives the highest gain of 0.246 bits, and therefore this attribute will be selected for the first splitting

(are attributes with many values favoured in general … ?)

New Example (2) Initial Decision Tree



D1

Att. 2	Att. 3	Class
70	True	Class**1**
90	True	Class**2**
85	False	Class**2**
95	False	Class**2**
70	False	Class**1**

D2

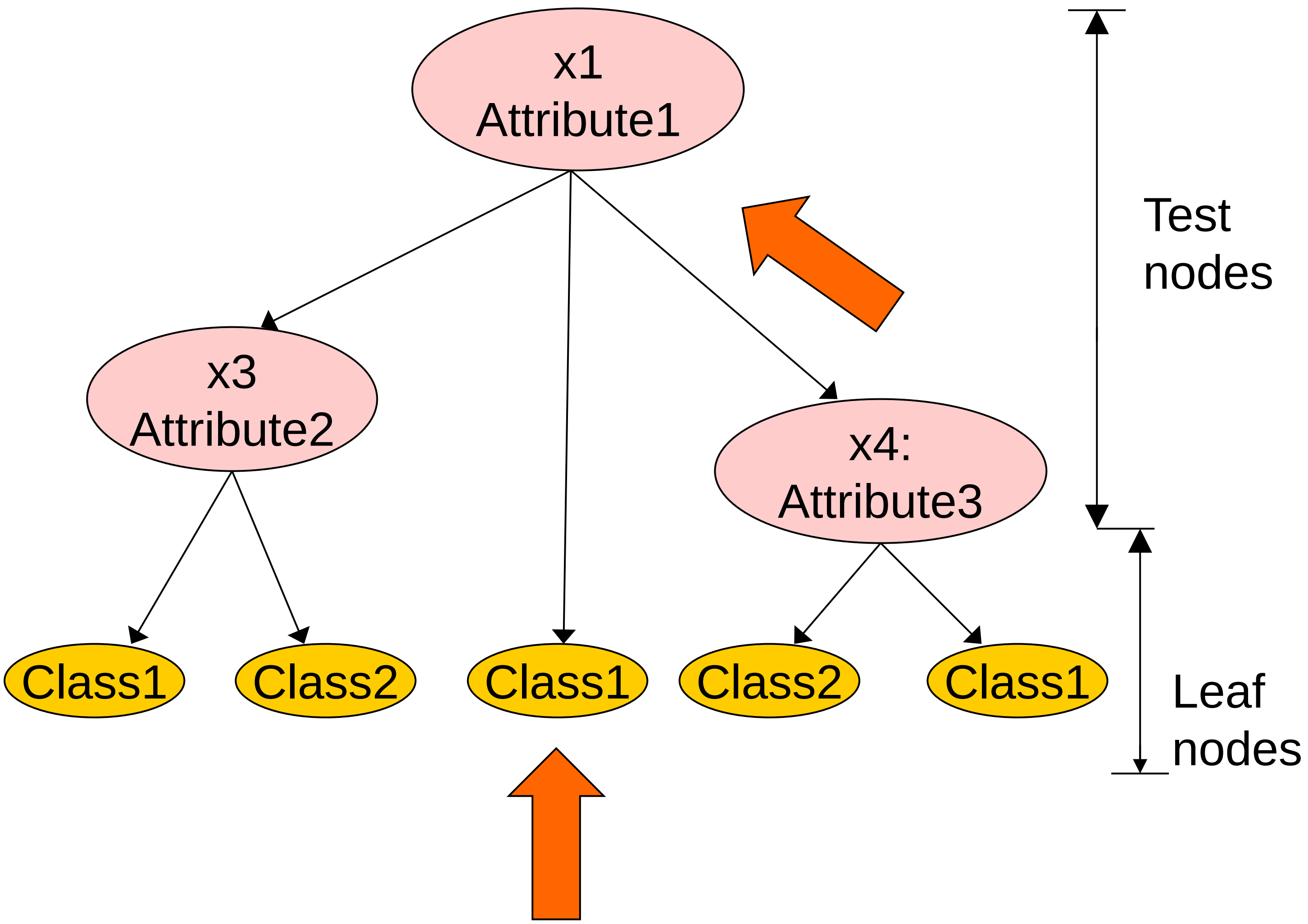
Att. 2	Att. 3	Class
90	True	Class**1**
78	False	Class**1**
65	True	Class**1**
75	False	Class**1**

D3

Att. 2	Att. 3	Class
80	True	Class**2**
70	True	Class**2**
80	False	Class**1**
80	False	Class**1**
96	False	Class**1**

Initial decision tree and subset cases for a database ***D***

New example (3) Final Decision Tree



All of them are in CLASS1

Final Decision Tree as Pseudo Code

- Decision Tree – **Pseudo-code Example:**

```
If        Attribute1 = A
          Then
                    If        Attribute2 <= 70
                              Then
                                        Classification = CLASS1;
                              Else
                                        Classification = CLASS2;
Elseif    Attribute1 = B
          Then
                                        Classification = CLASS1;
Elseif    Attribute1 = C
          Then
                    If        Attribute3 = True
                              Then
                                        Classification = CLASS2;
                    Else
                                        Classification = CLASS1.
```

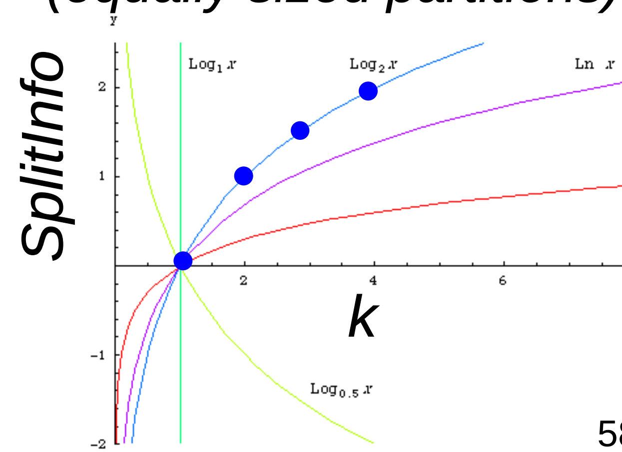
C4.5 Algorithm: Gain Ratio

- *Revision*: Measures we defined so far:
  - Entropy to classify a tuple in *D*: $Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$
  - Information needed (after using *A* to split *D* into *k* partitions) to classify *D*: $Info_A(D) = \sum_{j=1}^{k} \frac{|D_j|}{|D|} \cdot Info(D_j)$
  - Information gained for attribute *A*: $Gain(A) = Info(D) - Info_A(D)$
- Information gain measure is ***biased*** towards attributes with a large number of values *(this is also true for the Gini index)*
- C4.5 (a successor of ID3) uses gain ratio to ***normalize the information gain***

$$SplitInfo = -\sum_{j=1}^{k} \left(\frac{|D_j|}{|D|} \cdot \log_2 \left(\frac{|D_j|}{|D|} \right) \right)$$

$$GainRatio(A) = Gain(A) / SplitInfo(A)$$

(equally sized partitions)



Information Gain → Gain Ratio (prev. Example)

- Class "buys_computer =yes" (9x)
- Class "buys_computer =no" (5x)

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$
$$= 0.94$$

age	yes_i	no_i	$I(yes_i, no_i)$
<=30	2	3	0,971
31…40	4	0	0
>40	3	2	0,971

$$Info_{age}(D) = \boxed{\frac{5}{14} I(2,3)} + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2)$$
$$= 0.694$$

"age <=30" has 5 out of 14 samples, with 2 "yes" and 3 "no"

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$SplitInfo(age) = -\sum_{j=1}^{3}\left(\frac{|D_j|}{|D|}\cdot\log_2\left(\frac{|D_j|}{|D|}\right)\right)$$
$$= -\frac{5}{14}\log_2\left(\frac{5}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$
$$= 1.577$$

$$GainRatio(age) = 0.246 / 1.557 = 0.156$$

C4.5 Algorithm for Continuous Numeric Values

- Define binary test with outcomes $X \leq Z$ and $X > Z$, based on comparing the value of attribute against a ***threshold value*** Z
- ***Sort*** the training samples w.r.t. the values of the chosen attribute X
  - Number of these values is finite
  - Notation for sorted order: $\{v_1, v_2, \ldots, v_m\}$
- Examine ***all splits*** to find the ***optimal split***
  - m-1 possible splits on X.
  - Any threshold value between $\boldsymbol{v}_{\mathbf{i}}$ and $\boldsymbol{v}_{\mathbf{i+1}}$ has the same effect of dividing the cases into $\{v_1, v_2, \ldots, v_{\mathrm{i}}\}$ and $\{v_{\mathrm{i+1}}, v_{\mathrm{i+2}}, \ldots, v_{\mathrm{m}}\}$.
- Normal choice as representative threshold: ***midpoint*** of each interval: $(v_{\mathrm{i}} + v_{\mathrm{i+1}})/2$
  - C4.5 chooses the ***smaller*** value v_{i} of an interval $\{v_{\mathrm{i}}, v_{\mathrm{i+1}}\}$, rather than the midpoint – ensures that threshold values exist in the data

C4.5 Algorithm: Unknown Values

- In C4.5, samples with unknown values are ***distributed probabilistically*** according to the ***relative frequency*** of known values

- New information gain criterion for split in attribute X:

  $$Gain(X) = F \cdot (Info(D) - Info_X(D))$$

  - *Factor F* = number of samples in database with known value for a given attribute / total number of samples in a data set
  - *Factor F* here 13/14

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
?	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

C4.5 Algorithm: Unknown Values – Example (1)

13 remaining cases with values for Attribute1

$$\text{Info}(D) = -8/13 \ \log_2 (8/13) \ - 5/13 \ \log_2 (5/13) = \mathbf{0.961\ bits}$$

8 belong to CLASS1 *5 belong to CLASS2*

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
~~?~~	~~90~~	~~True~~	~~Class**1**~~
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

Test X_1 for the three values A, B, or C:

$$\begin{aligned}\text{Info}_{X1}(D) = \ & 5/13\,(-2/5 \log_2 (2/5) - 3/5 \log_2 (3/5)) \\ & + 3/13\,(-3/3 \log_2 (3/3) - \ 0/3 \log_2 (0/3)) \\ & + 5/13\,(-3/5 \log_2 (3/5) - 2/5 \log_2 (2/5)) \\ & = \ \mathbf{0.747\ bits}\end{aligned}$$

$$\mathbf{Gain\ (X_1) = 13/14} \cdot (0.961 - 0.747) = \mathbf{0.199\ bits}$$

Factor F

C4.5 Algorithm: Unknown Values – Example (2)

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
?	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

Distribution of samples into subsets with corresponding weight factors *w*

D1: Attribute1 = A

Att.2	Att.3	Class	*w*
70	True	Class**1**	1
90	True	Class**2**	1
85	False	Class**2**	1
95	False	Class**2**	1
70	False	Class**1**	1
90	***True***	***Class1***	***5/13***

D1: Attribute1 = B

Att.2	Att.3	Class	*w*
90	***True***	***Class1***	***3/13***
78	False	Class**1**	1
65	True	Class**1**	1
75	False	Class**1**	1

D1: Attribute1 = C

Att.2	Att.3	Class	*w*
80	True	Class**2**	1
70	True	Class**2**	1
80	False	Class**1**	1
80	False	Class**1**	1
96	False	Class**1**	1
90	***True***	***Class1***	***5/13***

C4.5 Algorithm: Generalizing Partitioning

- When a sample from ***D* with known value** is assigned to subset D_i, its probability belonging to ***D_i* is 1**, and in all other subsets is 0
- C4.5 associates with each sample (having ***missing value***) a ***weight w*** representing the ***probability*** that it belongs to each subset D_i :

  $$\boldsymbol{w_{\text{new}} = w_{\text{old}} \cdot P(D_i)}$$

- Splitting set *D* using test X_1 on Attribute1: New weights w_i will be probabilities, here: 5/13, 3/13, and 5/13, since initial w_{old} is 1

  $|D_1|$ = 5+**5/13**, $|D_2|$ = 3+**3/13**, and $|D_3|$ = 5+**5/13**

- The decision tree **leaves** are defined with two new parameters: **($|D_i|$/*E*)**
- $|D_i|$ is the sum of the ***fractional samples*** that reach the leaf, and

  E is the ***number of samples*** belonging to classes other than nominated class
- (3.4 / 0.4) means:
  - 3.4 (or 3 + 5/13) fractional training samples reached leaf,
  - 0.4 (or 5/13) of which did not belong to the class of the leaf

Partitioning – Example

- Decision tree for the database *D* with missing values:

```
If      Attribute1 == A
        Then
                If      Attribute2 <= 70
                        Then
                                Classification = CLASS1    (2.0 / 0);
                Else
                                Classification = CLASS2    (3.4 / 0.4);
Elseif  Attribute1 == B
        Then
                                Classification = CLASS1    (3.2 / 0);
Elseif  Attribute1 == C
        Then
                If      Attribute3 = True
                        Then
                                Classification = CLASS2    (2.4 / 0.4);
                Else
                                Classification = CLASS1    (3.0 / 0).
```

$(|Di|/E)$:
$|Di|$ = sum of the fractional samples that reach the leaf,
E = number of samples that belong to classes other than the nominated class.

Enhancements to Basic Decision Tree Induction (Intermediate Summary)

- Allow for ***continuous-valued attributes***
  - Partition the continuous attribute value into a discrete set of intervals, dynamically defined using the data’s attribute values
- Handle ***missing attribute values***
  - Assign probability to each of the possible values
- ***Attribute construction***
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication
- Challenge: ***incremental learning*** of decision trees

Decision Tree Algorithms – Building and Pruning

- ***Building phase***
  - Recursively split nodes using best splitting attribute for node.

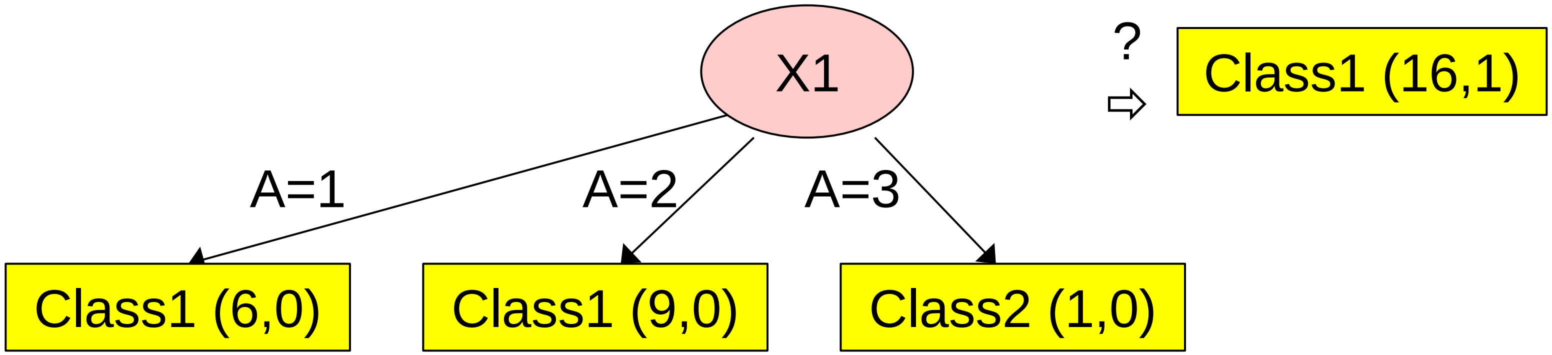
- ***Pruning phase***
  - Smaller imperfect decision tree generally achieves better accuracy on test data.
  - Prune leaf nodes recursively to prevent over-fitting.

Avoid Overfitting in Classification

- The generated tree may overfit the training data:
  - Too ***many branches***, some may reflect anomalies due to noise or outliers
  - Result: poor accuracy for unseen samples
- Two approaches to avoid overfitting:
  - ***Prepruning***: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - ***Postpruning***: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Pruning a Decision Tree

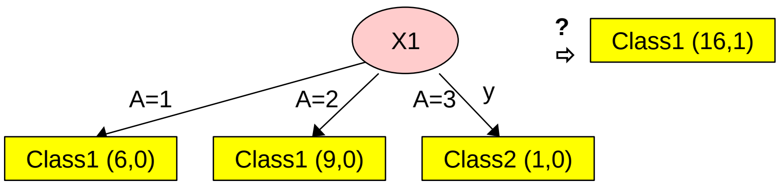
- ***Pruning***: Discarding one or more subtrees and replacing them with leaves
  - C4.5 follows a ***postpruning*** approach (pessimistic pruning)



will we replace this subtree with a single leaf node?

Pruning Decision Tree: **P**redicted **E**rror

$$PE = \sum_{i=1}^{nodes} n_i \cdot U_{25\%}$$



# of samples in the node

upper limit on error rate (for the node): from statistical tables for binomial distributions

- Using default confidence of 25%, **upper limits on the error rates** for all nodes are collected from statistical tables for binomial distributions:

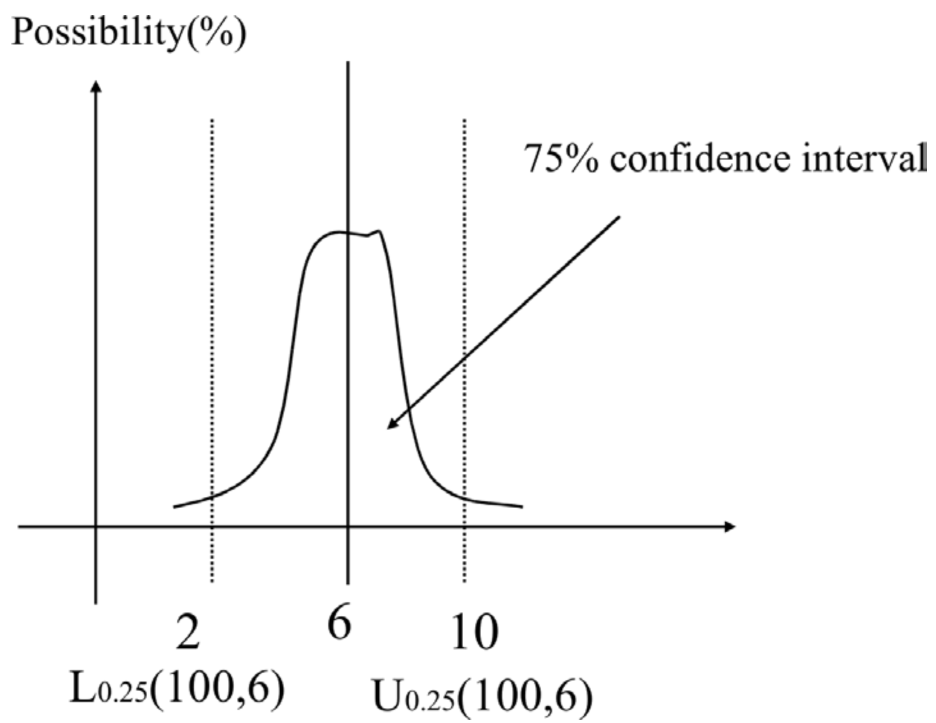
  Tree: $U_{25\%}(6,0) = 0.206$, $U_{25\%}(9,0) = 0.143$, $U_{25\%}(1,0) = 0.750$

  Node: $U_{25\%}(16,1) = 0.157$

- **Predicted errors** for the subtree and the replaced node are:
  - $PE_{tree} = 6 \cdot 0.206 + 9 \cdot 0.143 + 1 \cdot 0.750 = 3.257$
  - $PE_{node} = 16 \cdot 0.157 = 2.512$
  - Since $PE_{tree} > PE_{node}$, replace the subtree with the new leaf node.

$U_{CF}(|D_i|, E)$

- Consider classifying E examples incorrectly out of $|D_i|$ samples (like observing E events in $|D_i|$ trials in the binomial distribution)
- For a given confidence level CF, the upper limit on the error rate over the whole population is $U_{CF}(|D_i|, E)$ with CF% confidence.
- Example:
  - $U_{25\%}(100,6)$
  - 100 examples in a leaf
  - 6 examples misclassified
  - How large is the true error assuming a pessimistic estimate with a confidence of 25%?



Extracting Decision Rules from Trees

- Represent the knowledge in the form of ***IF-THEN*** rules
  - One rule is created for each path from the root to a leaf.
  - Each attribute-value pair along a path forms a conjunction.
  - The leaf node holds the class prediction.
- Rules are easier for humans to understand

Examples:

```
IF age = “<=30” AND student = “no”
THEN buys_computer = “no”
                    IF age = “<=30” AND student = “yes”
                    THEN buys_computer = “yes”
IF age = “31…40”
THEN buys_computer = “yes”
                    IF age = “>40” AND credit_rating = “excellent”
                    THEN buys_computer = “yes”
IF age = “>40” AND credit_rating = “fair”
THEN buys_computer = “no”
```

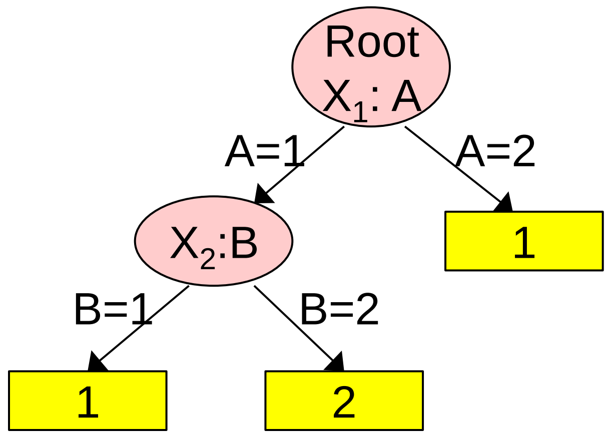
Rule Ordering

If more than one rule is triggered, we need ***conflict resolution***

- Size ordering
  - assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the *most attribute tests*)
- Class-based ordering
  - Rules for the most frequent class come first, or
  - Sort based on misclassification cost per class
- Rule-based ordering (decision list)
  - rules are organized into one long priority list, according to some measure of rule quality (e.g. accuracy, # attribute tests) or by experts

C4.5 Algorithm: Generating Decision Rules may not really simplify

Decision tree



Transformation

Paths into Rules

Decision rules

If A=1 and B=1 Then Class1
If A=1 and B=2 Then Class2
If A=2 Then Class1

Decision rules for database **D:**

Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
?	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	80	False	Class**1**
C	96	False	Class**1**

If Attribute1 = A and Attribute2 <= 70
Then Classification = CLASS1 (2.0 / 0);

If Attribute1 = A and Attribute2 > 70
Then Classification = CLASS2 (3.4 / 0.4);

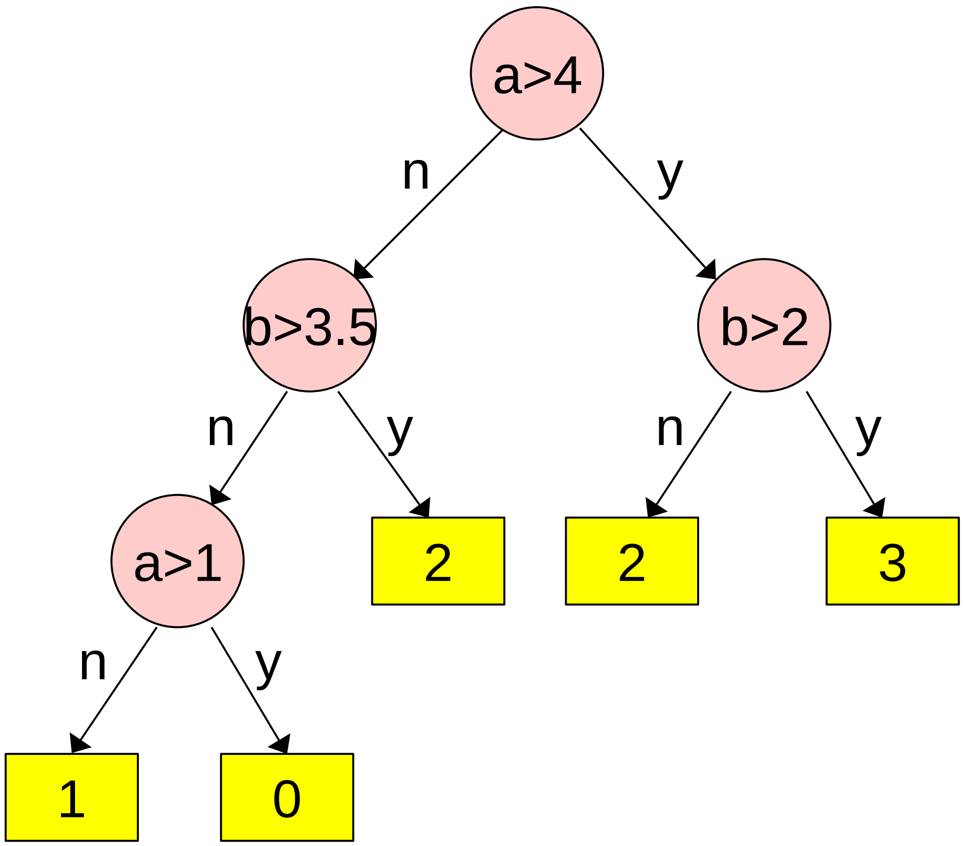
If Attribute1 = B
Then Classification = CLASS1 (3.2 / 0);

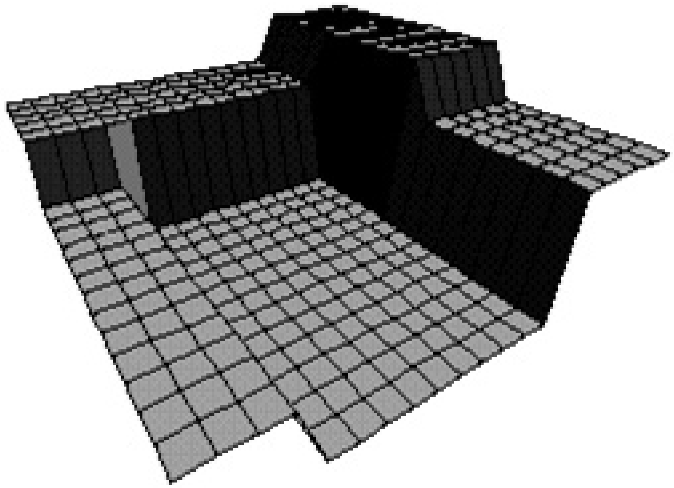
If Attribute1 = C and Attribute3 = True
Then Classification = CLASS2 (2.4 / 0.4);

If Attribute1 = C and Attribute3 = False
Then Classification = CLASS1 (3.0 / 0).

Bottom example is for previous partitioning data set (14 samples).

Limitations of Decision Trees and Decision Rules (1)

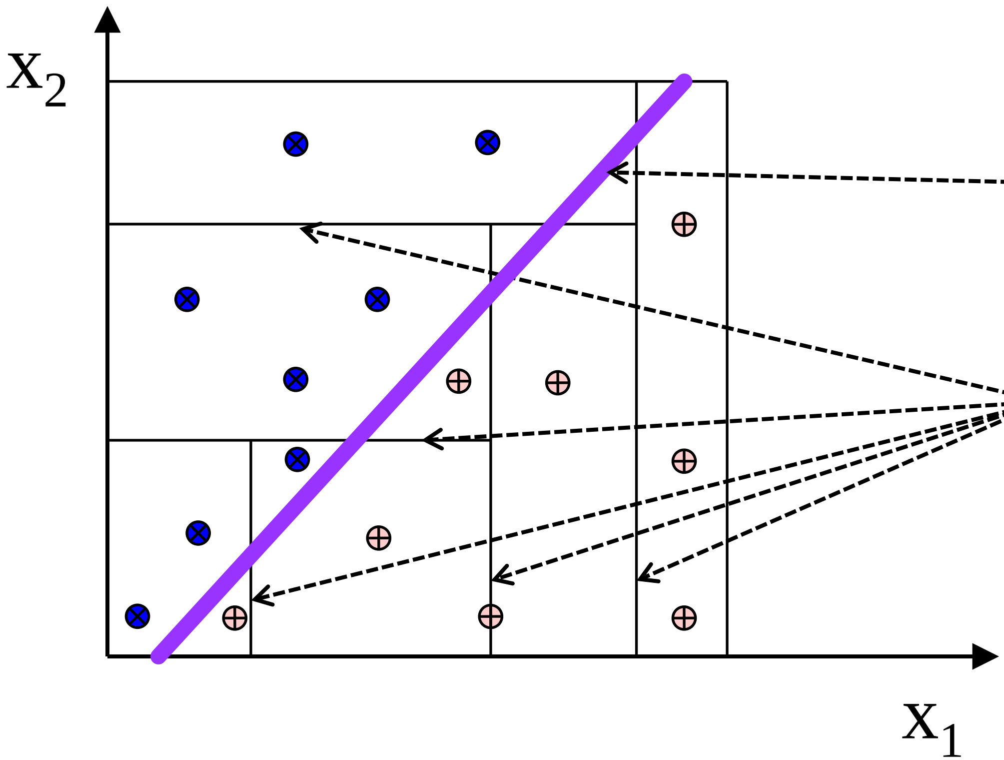




Example:

- 2D samples are classified using a third dimension for classes
- Problematic: classification function is much ***more complex*** with ***related attributes***

Limitations of Decision Trees and Decision Rules (2)



Classification through a linear combination of attributes

Rules as orthogonal hyperplanes

Limitations of Decision Trees and Decision Rules (3)

- Let a given class be supported, if ***any k out of n*** conditions are met.
- To represent this classifier with rules, it would be necessary to define $\binom{n}{k}$ regions only for one class

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}.$$

- **Example**: Medical diagnostic:
    - If 4 out of 11 symptoms support diagnosis of a given disease, then the corresponding classifier will generate 330 regions in 11-dimensional space for positive diagnosis only.

  ⇨ corresponds to 330 decision rules.

Limitations of Decision Trees and Decision Rules: Further Ideas

- Introducing new attributes, rather than removing old ones, can avoid sometimes-intensive fragmentation of the n-dimensional space:

Model: $(A1 \vee A2 \vee A3) \wedge (A4 \vee A5 \vee A6) \wedge (A7 \vee A8 \vee A9) \rightarrow$ **C1**

Solution 1:

$A1 \wedge A4 \wedge A7 \rightarrow C1$

$A1 \wedge A5 \wedge A7 \rightarrow C1$ → 27 combinations

$A1 \wedge A6 \wedge A7 \rightarrow C1$

…

Solution 2: Introduce **new derived attributes**:

$B1 = A1 \vee A2 \vee A3$

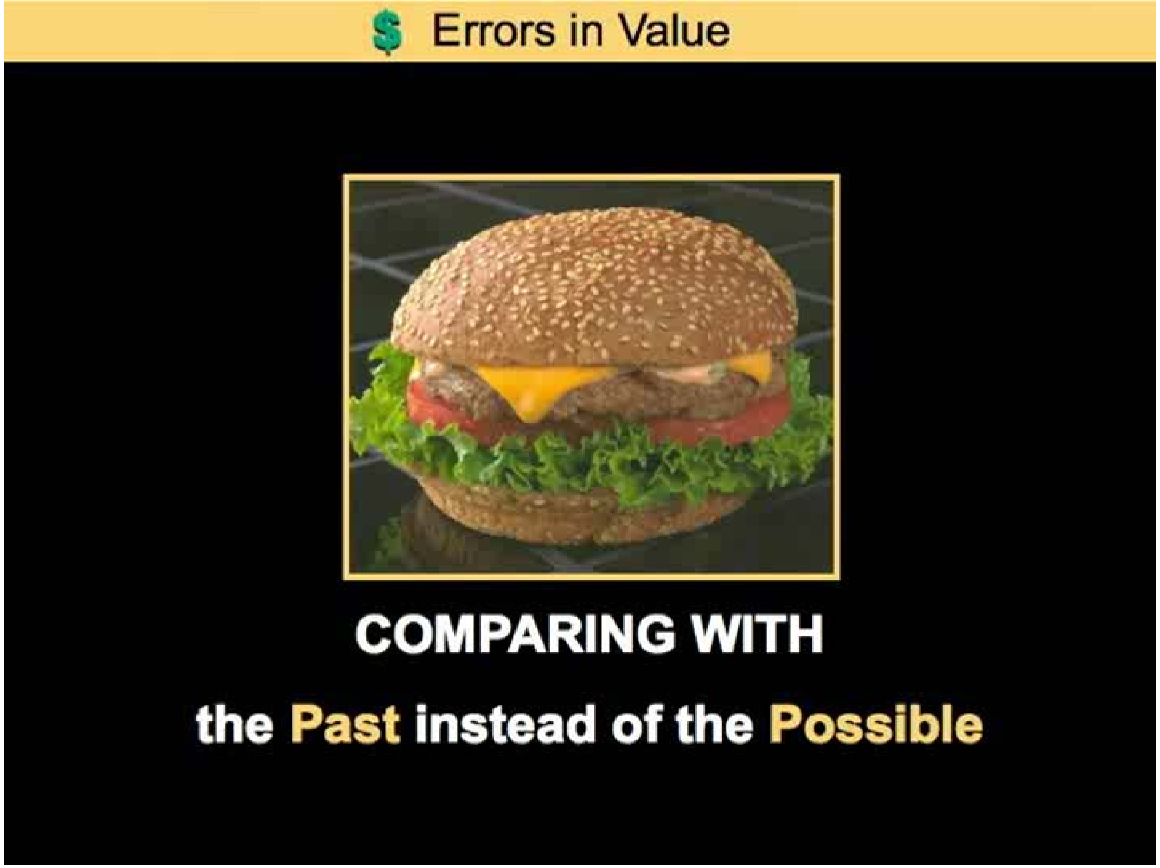
$B2 = A4 \vee A5 \vee A6$ → $\mathbf{B1 \wedge B2 \wedge B3 \rightarrow C1}$

$B3 = A7 \vee A8 \vee A9$

Decision Trees (Summary)

- Advantages
  - Automatically create tree representations from data
  - Trees can be converted to rules, can discover “new” rules
  - Identify most discriminating attribute first
    - Using Information Gain (Ratio) or Gini Impurity
  - Tree can handle discrete, continuous, mixed, and missing attributes
- Disadvantages
  - Trees can become large and difficult to understand
  - Can produce counter-intuitive rules
  - Examines attributes individually, but not inter-attribute relationships
  - Future splits not known when splitting → not globally optimal tree
  - Tree induction rules have no direct relation to training objective, i.e. minimizing the classification error

Limitations: Decisions over Time



Dan Gilbert: Why we make bad decisions, TED talks, 2008. Video online