Data Mining

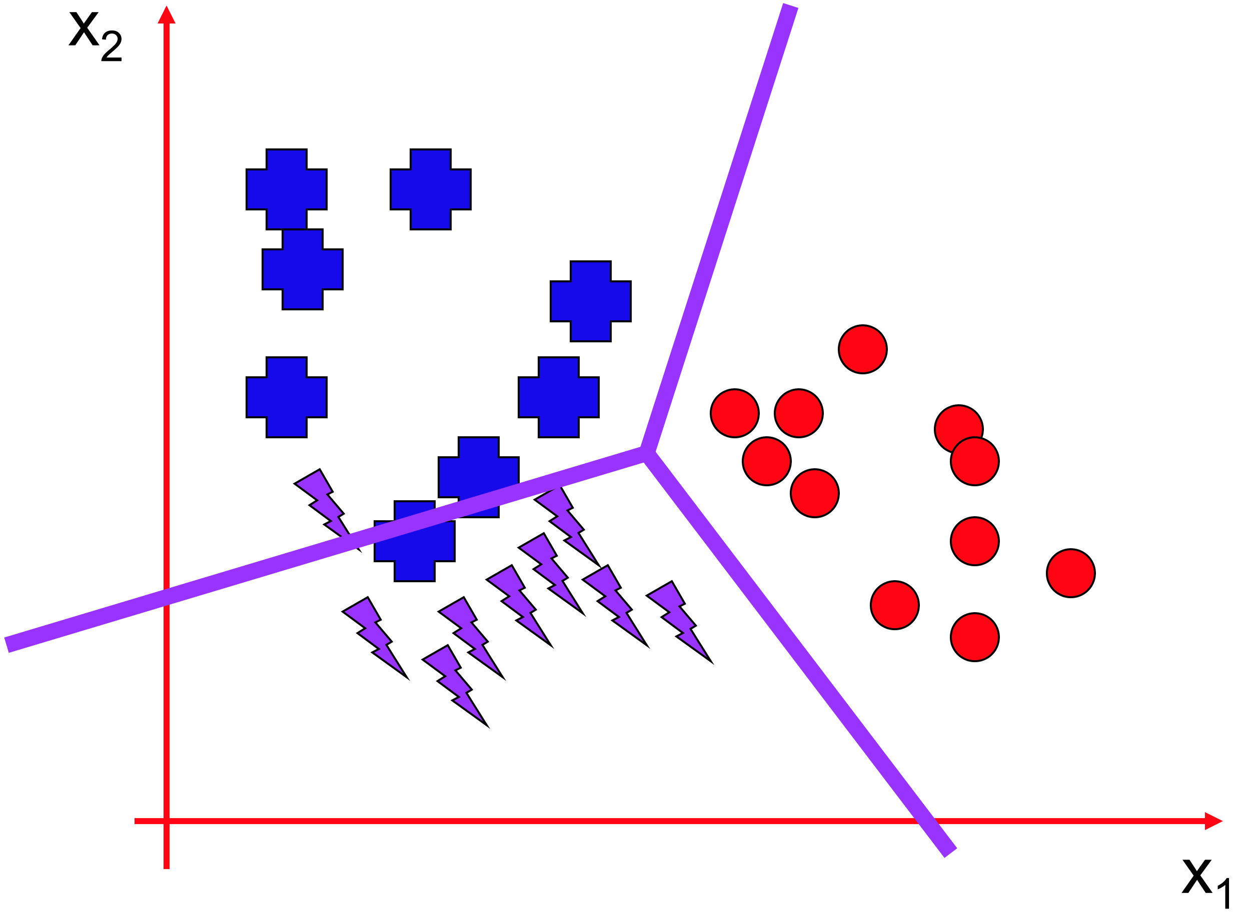
Lecture 5
Decision Trees and Classification

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

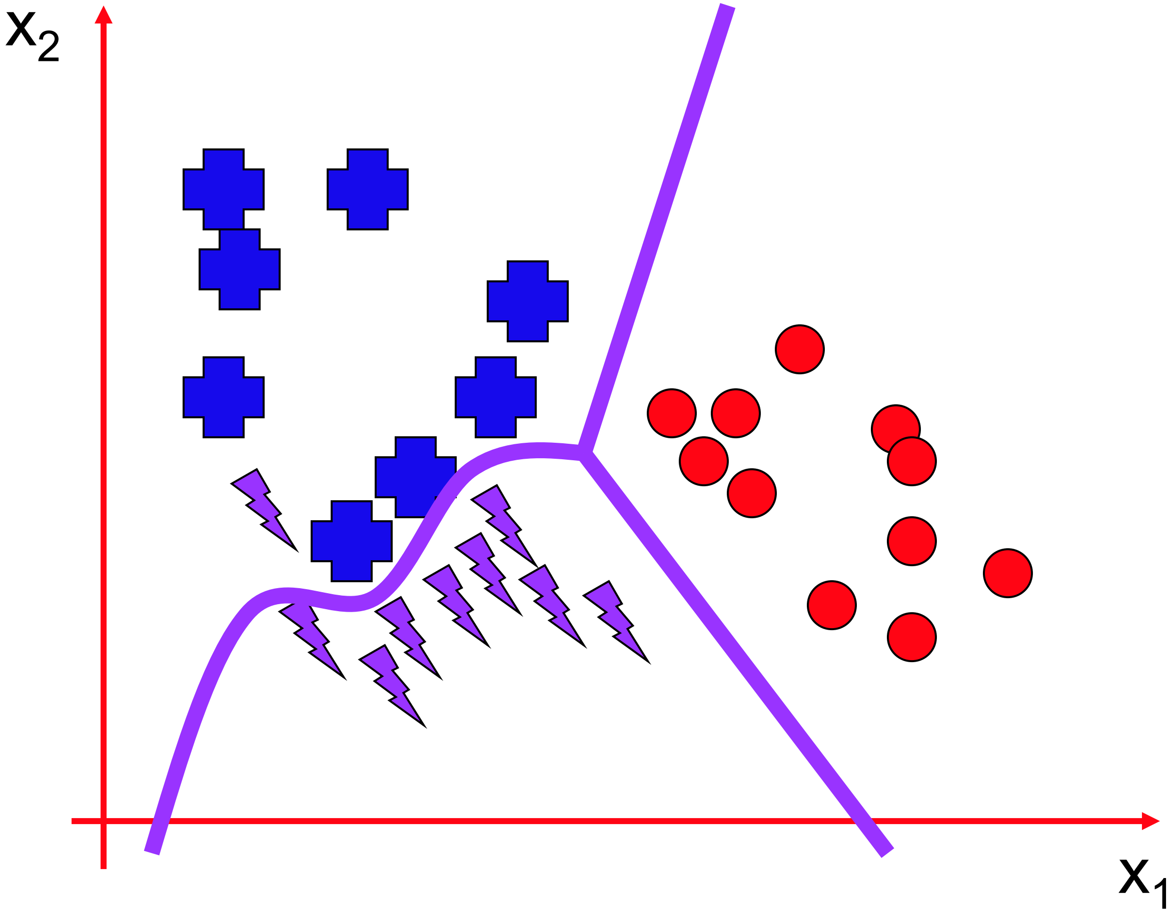
Motivation: Making decisions

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

Supervised vs. Unsupervised Learning

- **Supervised Learning (*Classification*)**
  - Supervision: The training data (observations, measurements, etc.) are accompanied by ***labels*** indicating the class of the observations
  - New data is classified based on the training set
- **Unsupervised Learning (*Clustering*)**
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. numeric Prediction

- ***Discrete Classification***
  - assigns categorical class labels (discrete or nominal)
  - learns a model based on a training set and the values (***class labels***) of a classifying attribute and uses it in classifying new data
- ***Numeric Prediction***
  - models continuous-valued functions, i.e., predicts unknown or missing values
- **Typical applications**
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Decision Trees

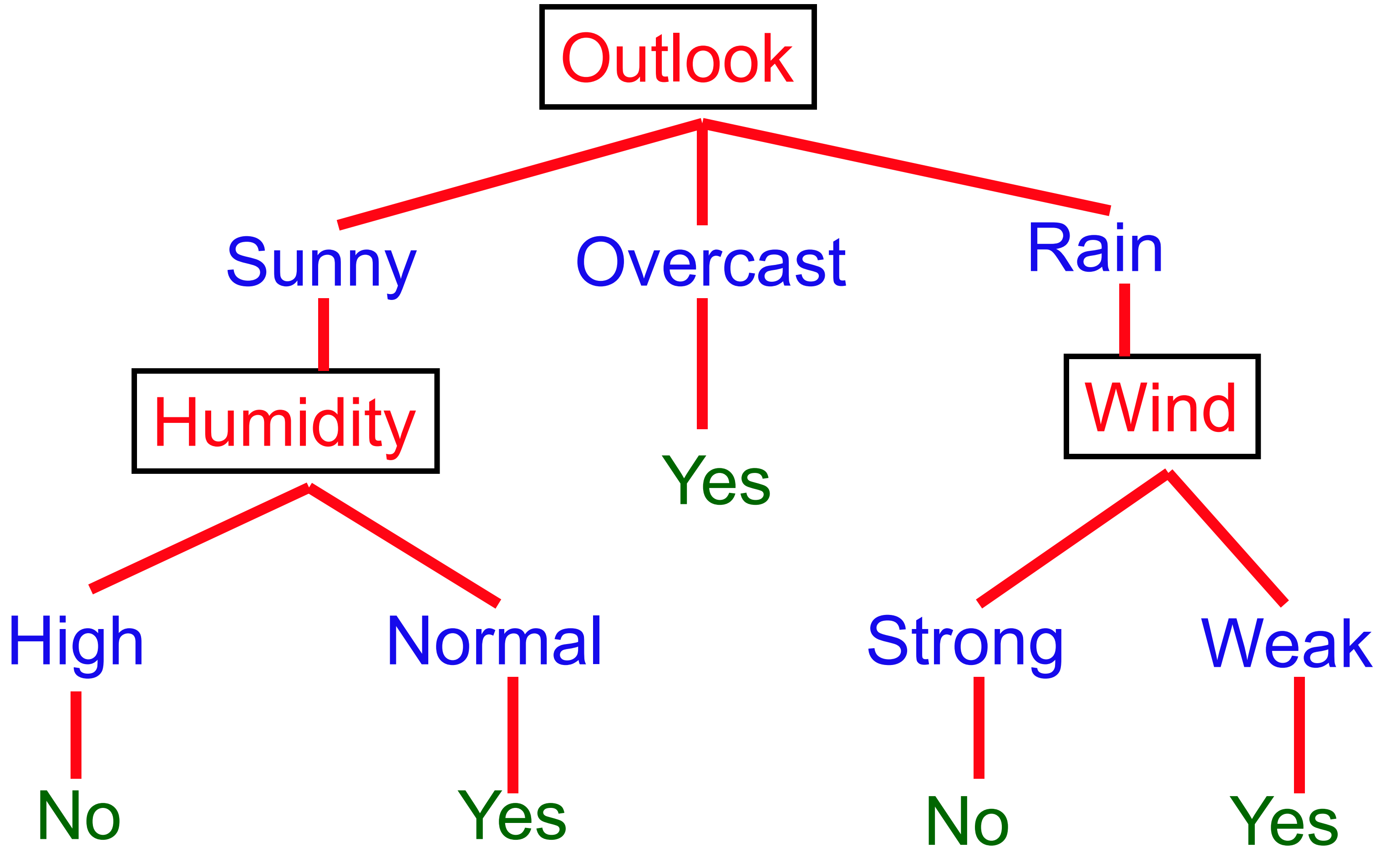
- Split classification into a series of choices about features in turn

- Lay them out in a tree

- Progress down the tree to the leaves

Example: Anyone for Tennis?

Bottom leaves show decision whether to play tennis

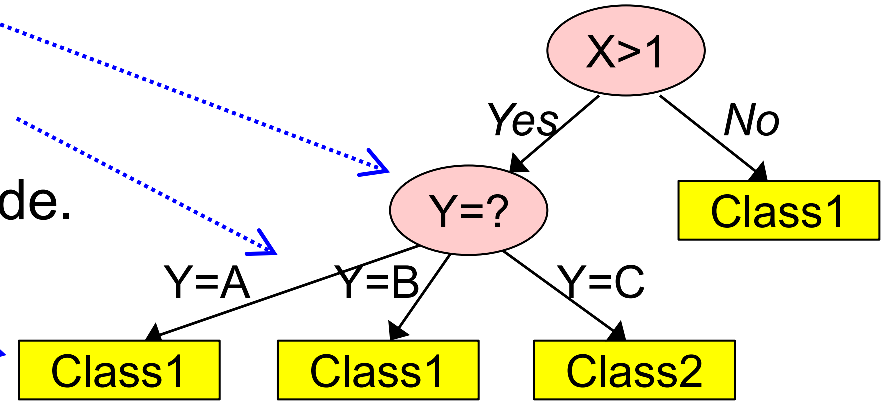


Rules and Decision Trees

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

Decision Trees

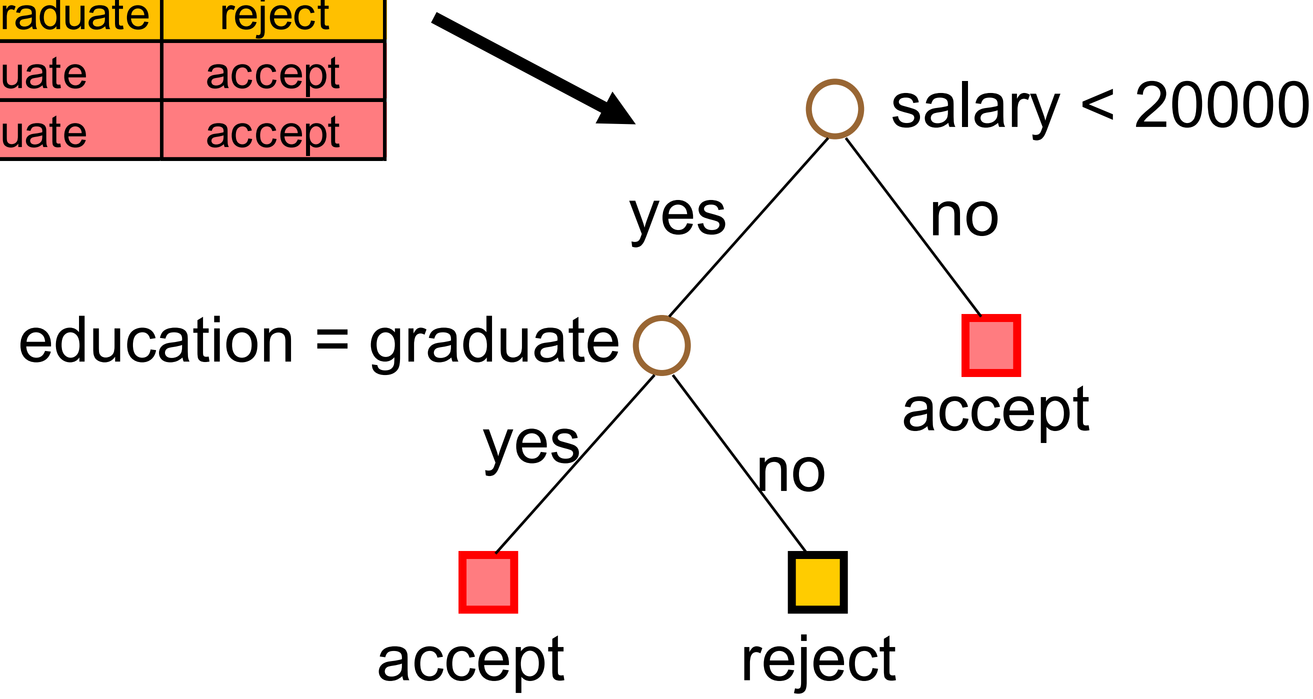
- Efficient method for producing ***classifiers*** from data
  - ***Supervised learning*** methods that construct decision trees from a set of input-output samples.
  - Guarantees that a simple, but not necessary the simplest, tree is found.
- Consists of
  - ***Nodes*** that are tests on the attributes.
  - 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
40000	under graduate	accept
15000	under graduate	reject
75000	graduate	accept
18000	graduate	accept



Decision Tree for Classification

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

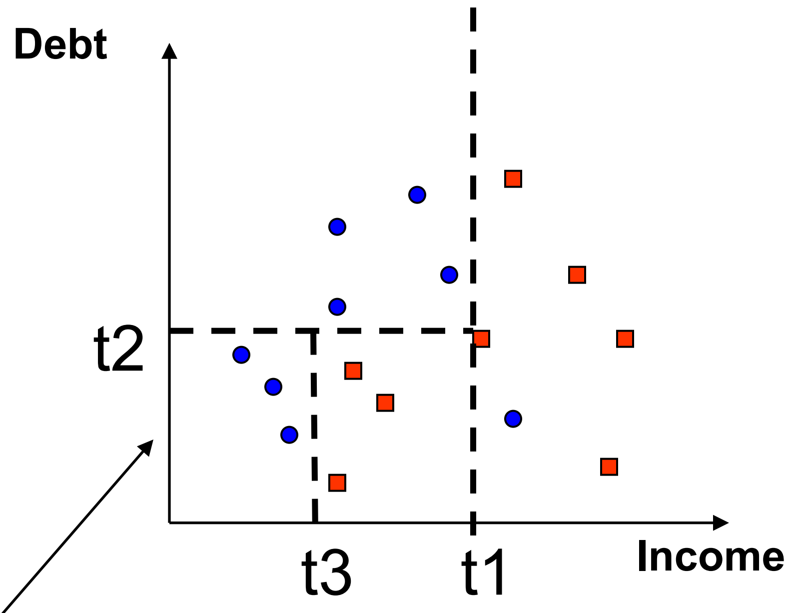
- Task: Develop a model/profile for each class:
  - **Example profile** (good credit):

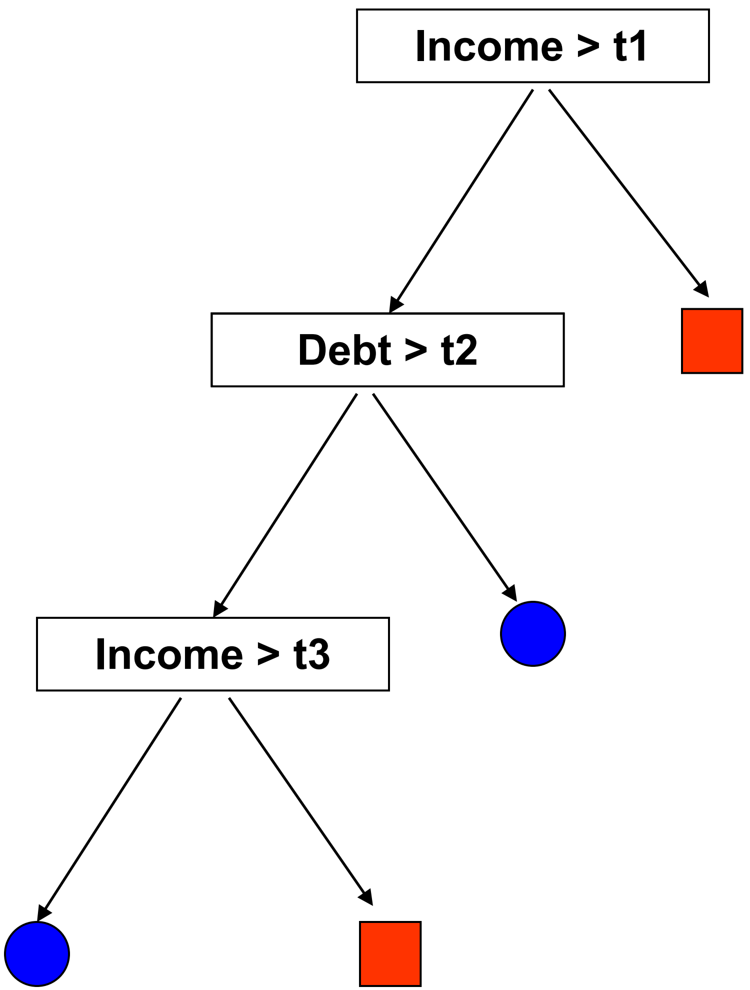
    (25 <= age <= 40 and income > 40k) or (married = YES)
    => Credit = Good (approved)

Classification by Decision Tree Induction

- Decision tree ***generation*** consists of two phases:
  - Tree ***construction***:
    - At start, all the training examples are at the root.
    - Partition the examples recursively based on selected attributes.
  - Tree ***pruning***:
    - Identify and remove branches that reflect noise or outliers.
- ***Use*** of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Decision Tree: Example





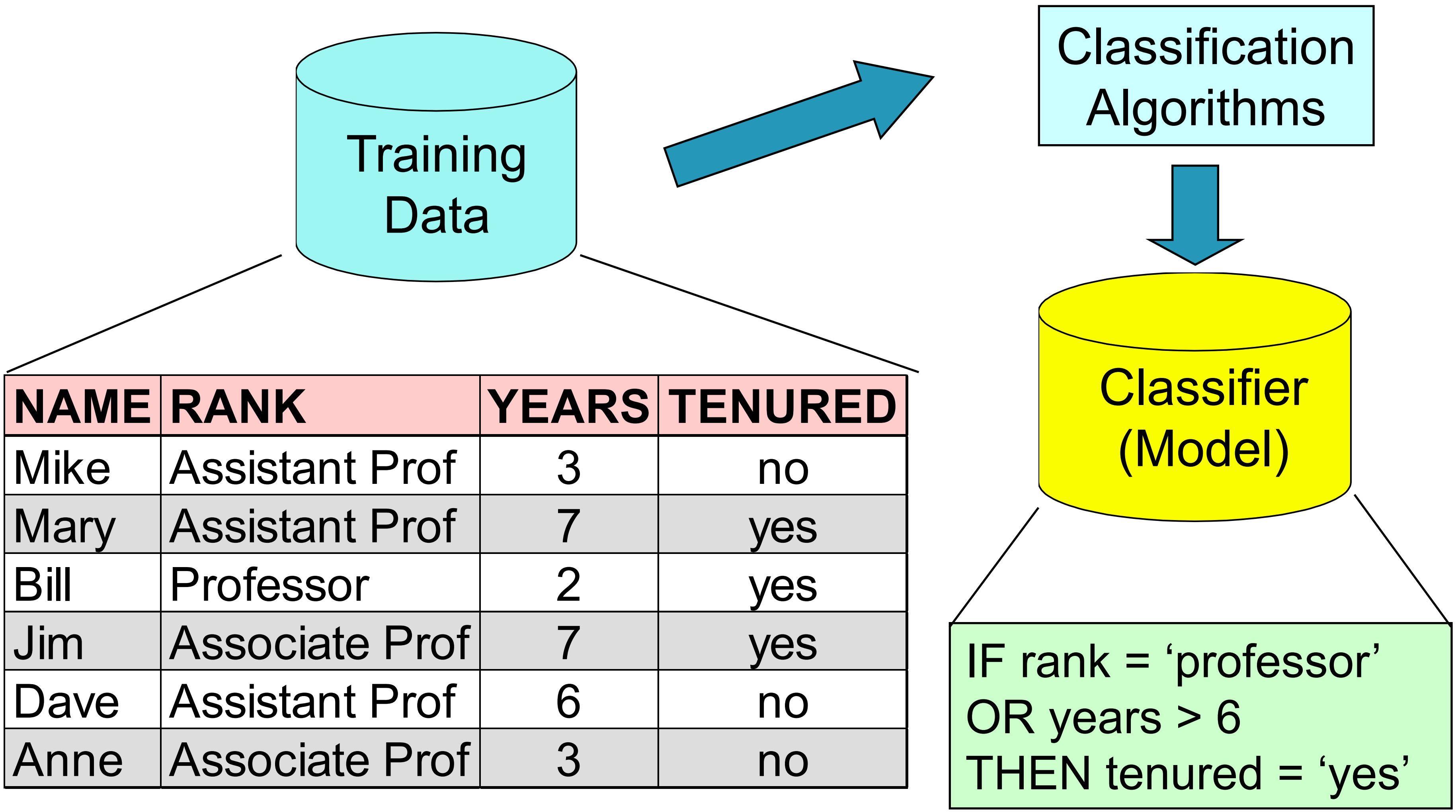
Note: tree boundaries are piecewise linear and axis-parallel

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 ***training set***
  - The model is represented as classification rules, decision trees, or mathematical formulae
- ***Model usage***: for classifying future or unknown objects
  - ***Estimate accuracy*** of the model
    - The known label of test sample is compared with the classified result from the model
    - ***Accuracy*** rate is the percentage of test set samples that are correctly classified by the model
    - ***Test set*** is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to ***classify data*** tuples whose class labels are not known

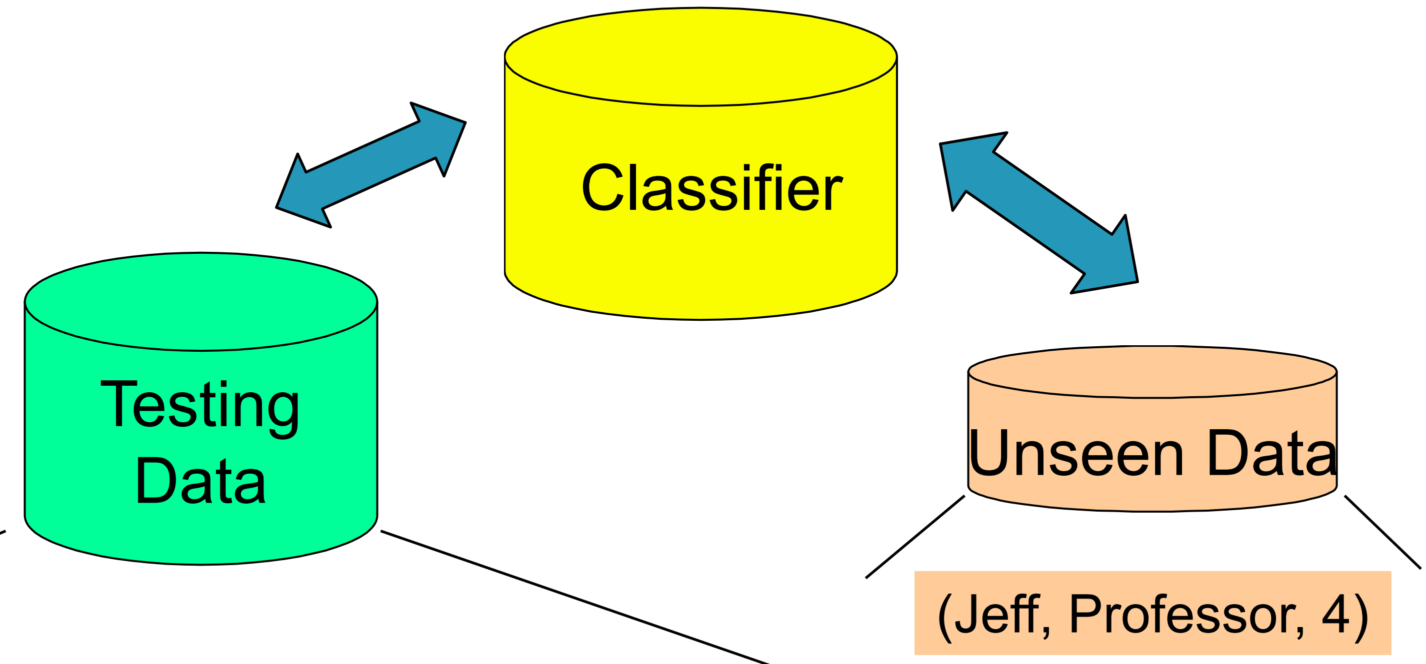
Process (1): Model Construction



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

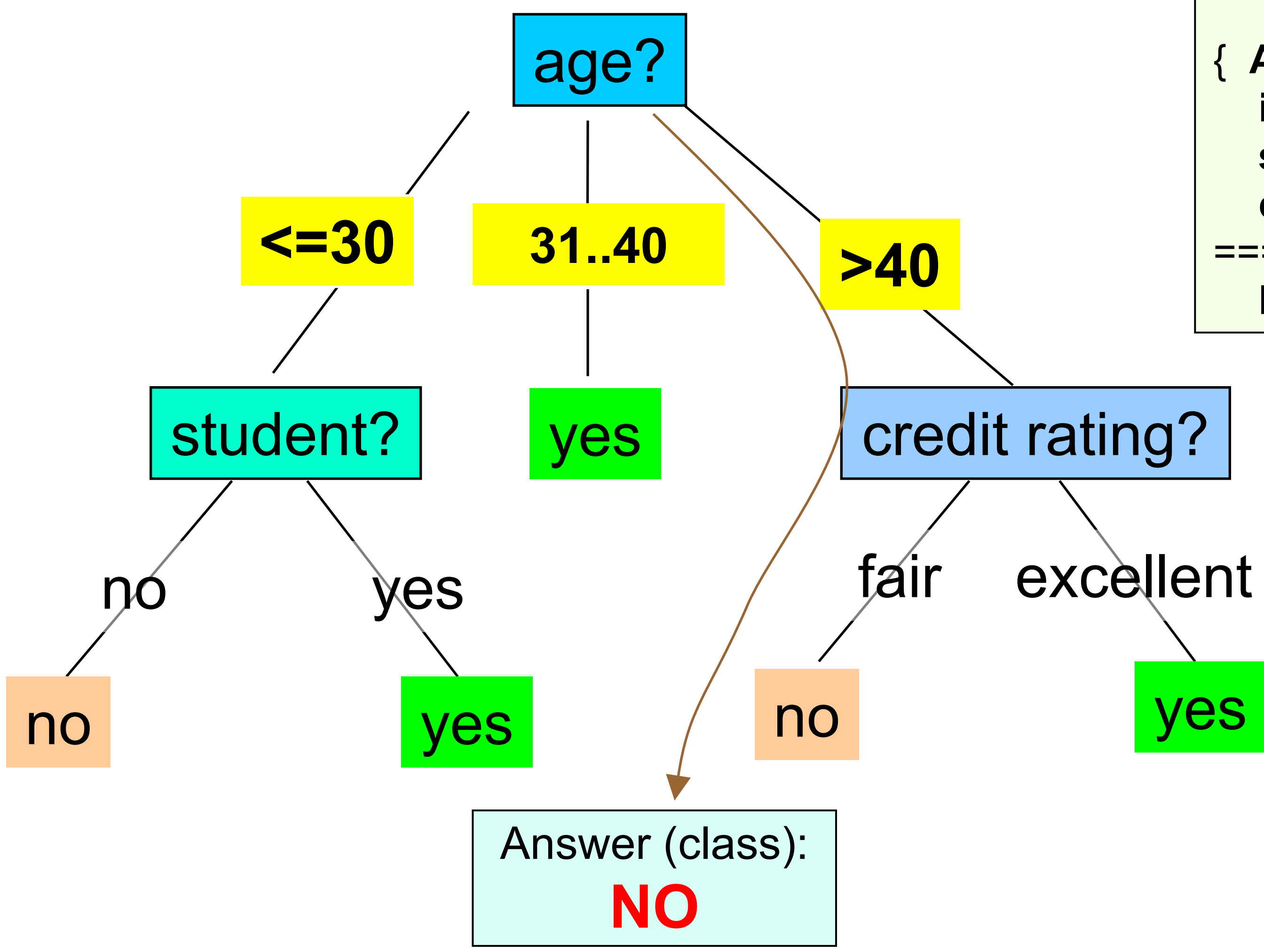
This follows an example of Quinlan's ID3

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 = **?**



Decision Tree

- Requirements for a Decision Tree algorithm:
  1. Attribute-value description
  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

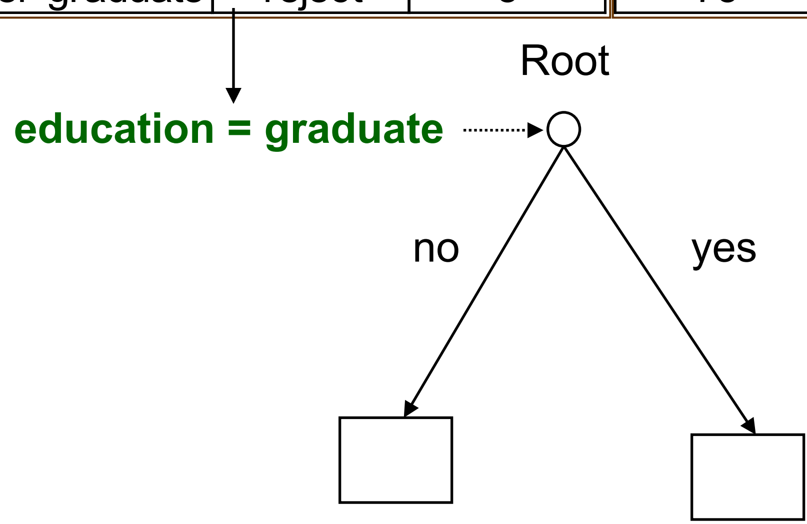
Algorithm for Decision Tree Induction

- 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

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

10	reject	1
15	accept	3
18	reject	5
40	accept	2
75	accept	4



high-school	reject	1
under-graduate	accept	2
under-graduate	reject	5

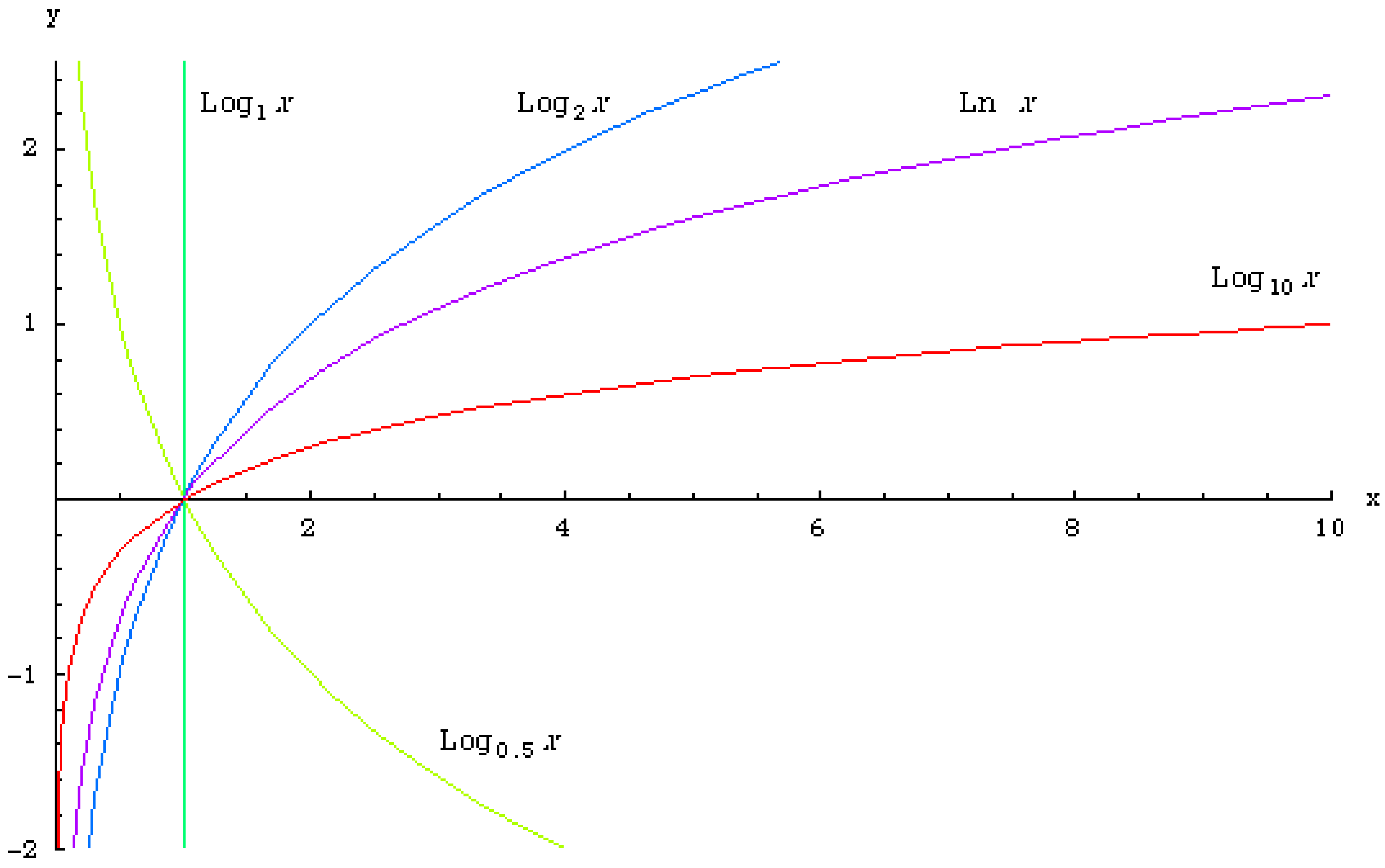
10	reject	1
18	reject	5
40	accept	2

graduate	accept	3
graduate	accept	4

15	accept	3
75	accept	4

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

Reminder…$\log_2 p$



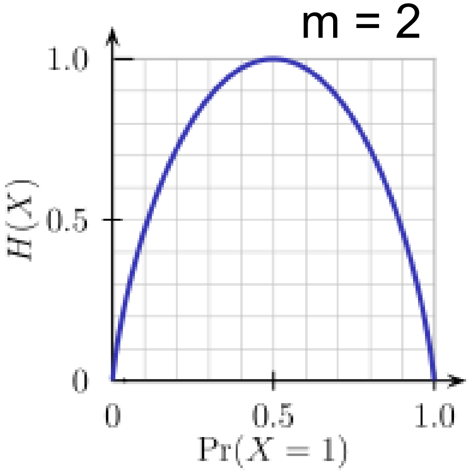
Brief Review of Entropy

- Entropy (Information Theory)
  - ***Measure of uncertainty*** associated with a random variable
  - Calculation: For a discrete random variable Y taking m distinct values $\{y_1,\ldots,y_m\}$,

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

where $p_i=P(Y=y_i)$

  - Interpretation
    - Higher entropy ⇨ higher uncertainty
    - Lower entropy ⇨ lower uncertainty



- Conditional Entropy

$$H(Y \mid X) = -\sum_{x} p(x) \cdot H(Y \mid X = x)$$

Attribute Selection Measure: Information Gain (ID3/C4.5)

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

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

- ***Information needed*** (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

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

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

Information Gain – Example

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

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

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

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

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

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

Similarly,

$$Gain(income) = 0.029$$

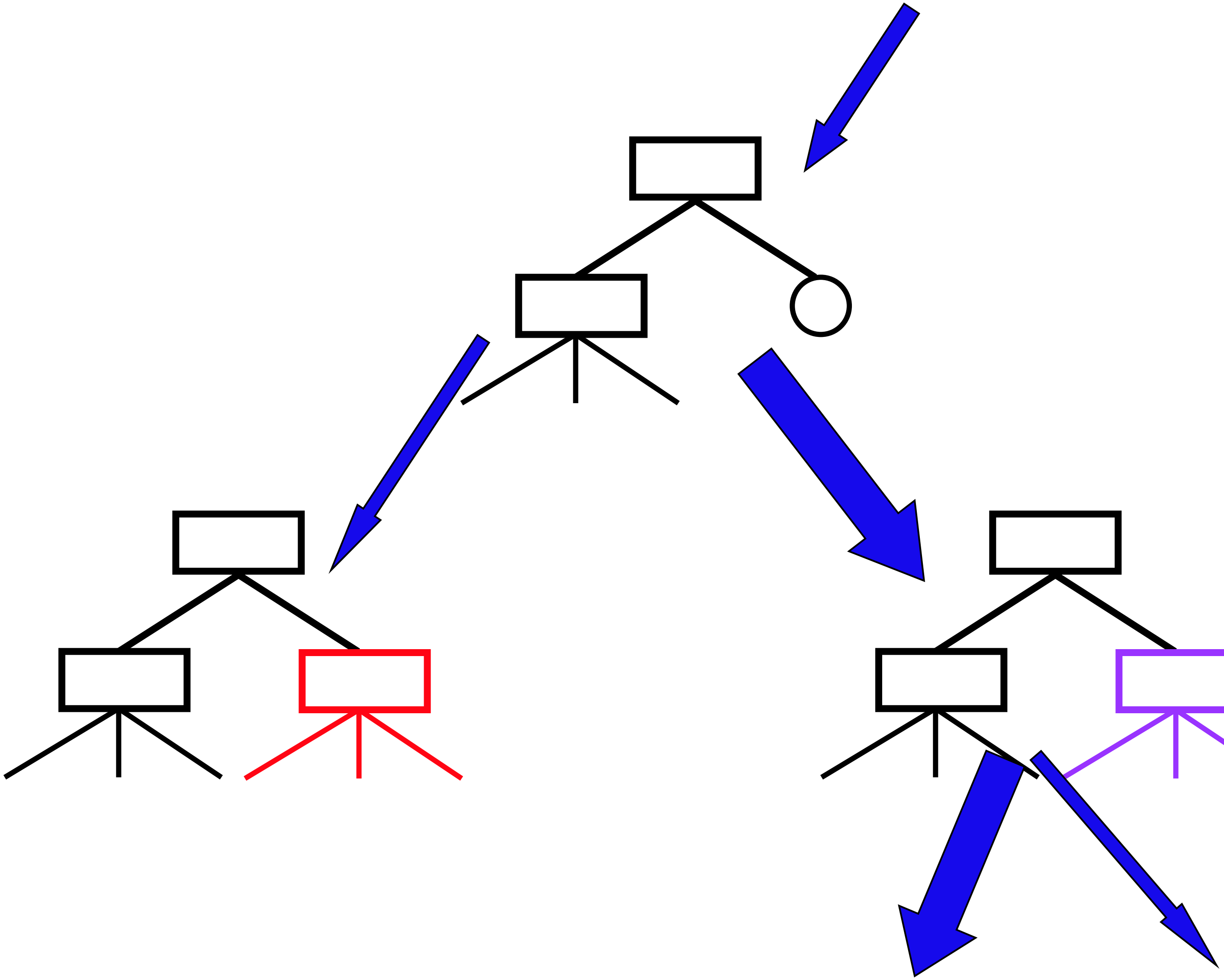
$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Making decisions – Errors in value
Fresher after lunch

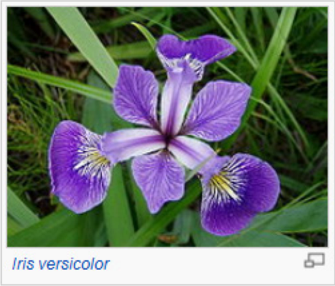
Dan Gilbert: Why we make bad decisions,
TED talks   Video online

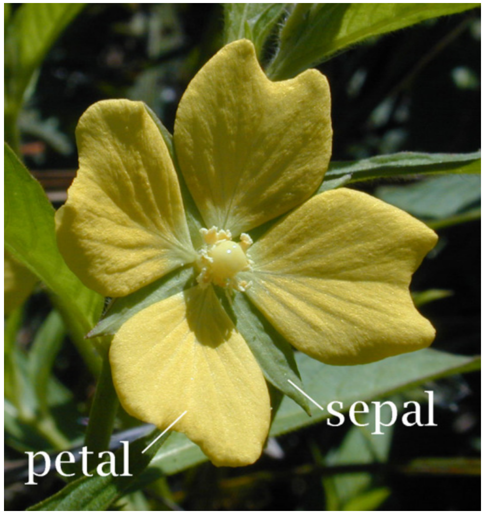
Search



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.





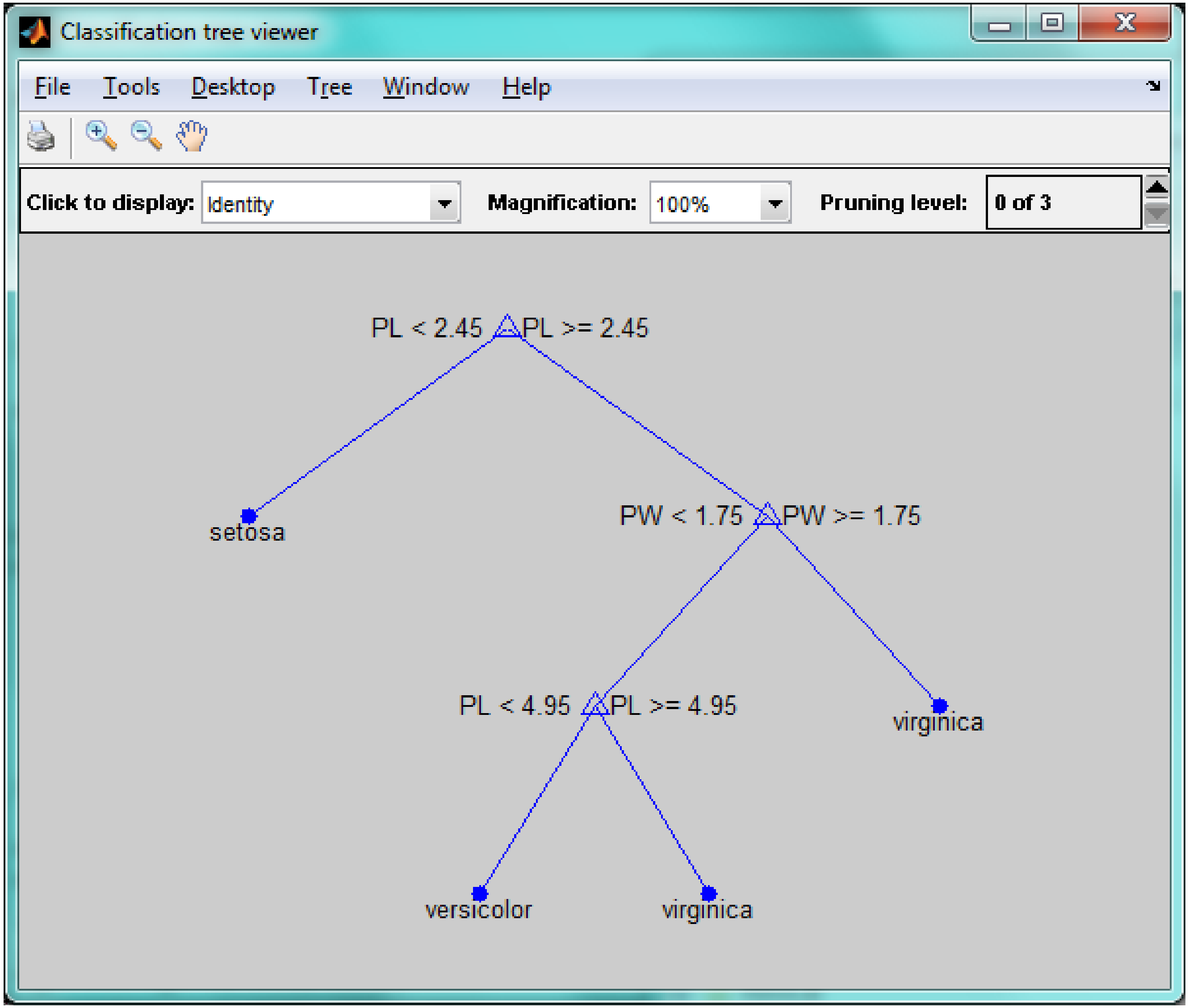
Matlab Demo of Decision Tree

- 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:

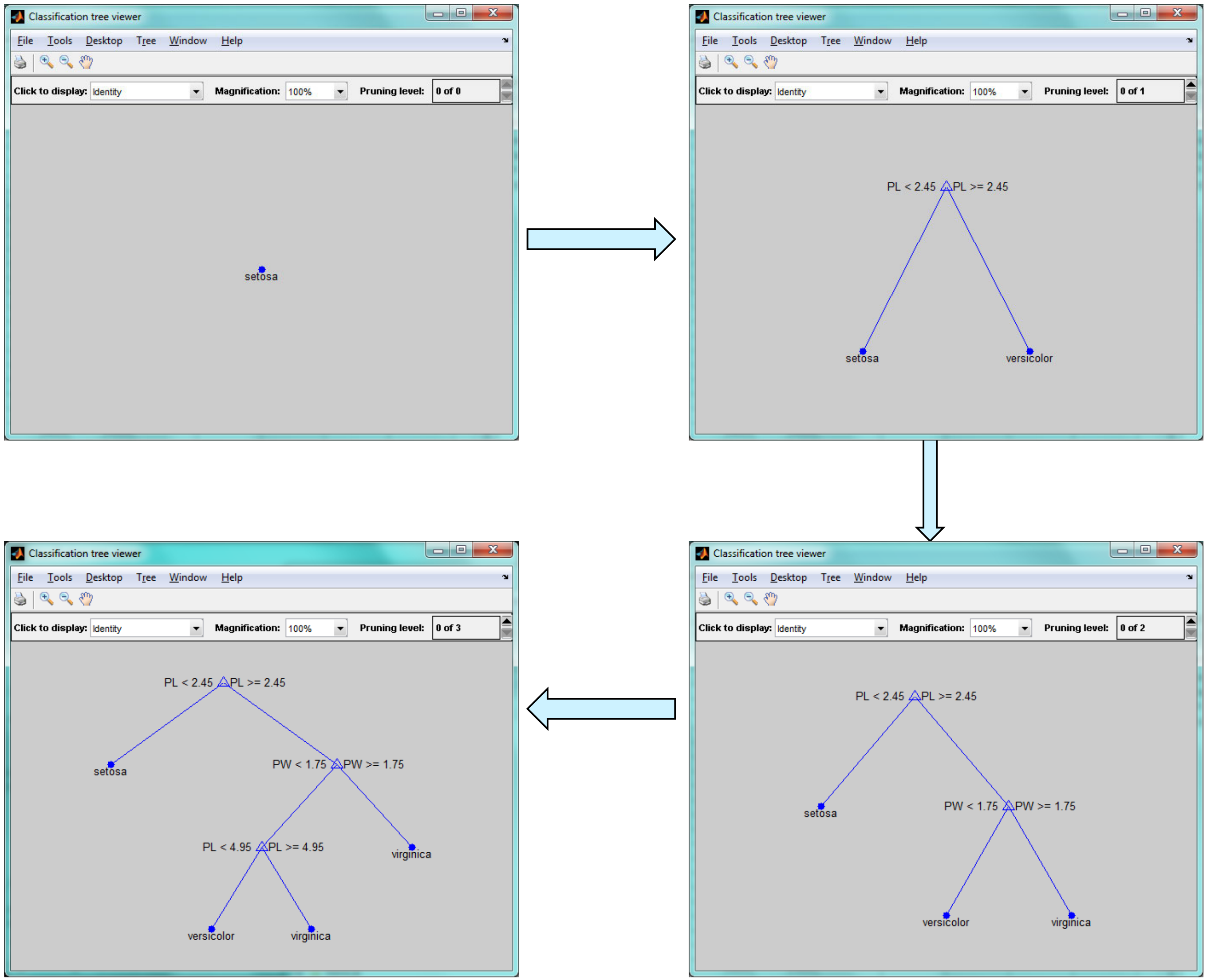
```
load fisheriris;
t = classregtree(meas,species,... 'names',{'SL'
'SW' 'PL' 'PW'});
```

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

Matlab Demo of Decision Tree

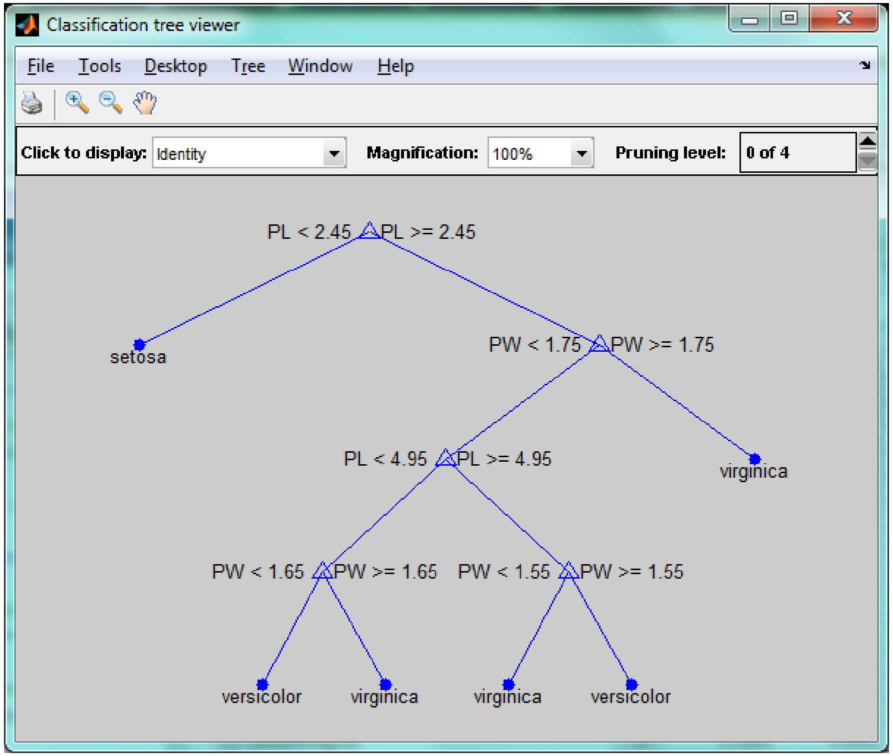


Matlab Demo of Decision Tree



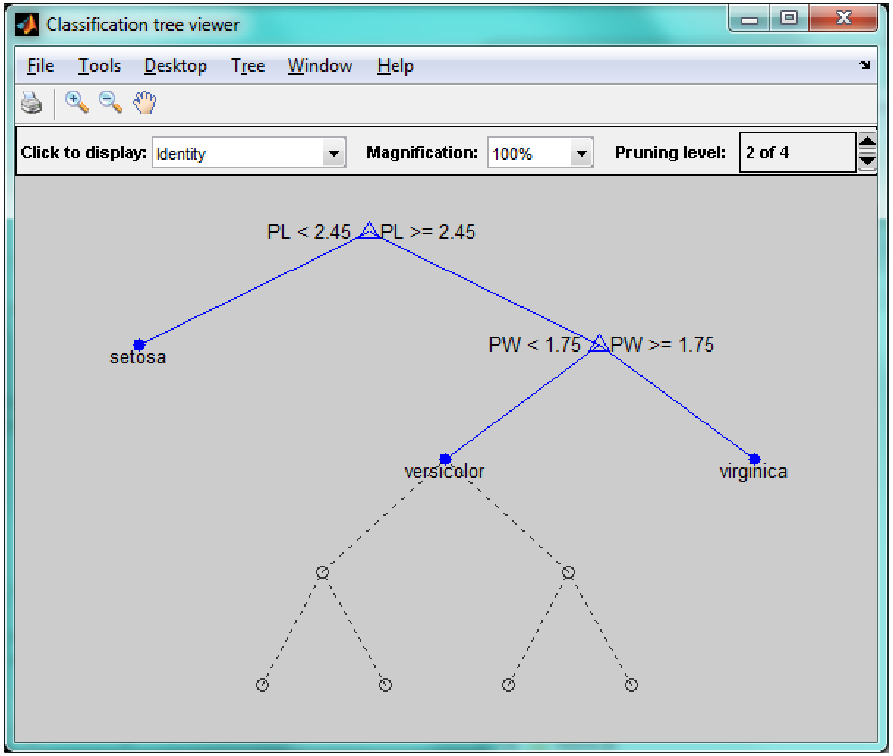
Matlab Demo of Decision Tree

- 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:

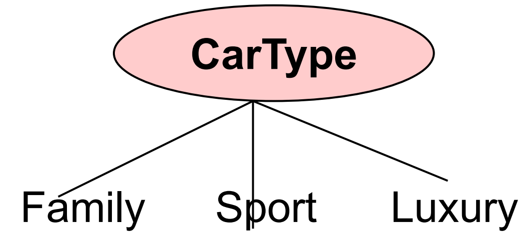
```
1. Initialize root node of tree.
2. while a node N that can be split:
3.     for each attribute A, evaluate splits on A,
4.     use best split to split N.
```

- Use ***Entropy index*** to find best split

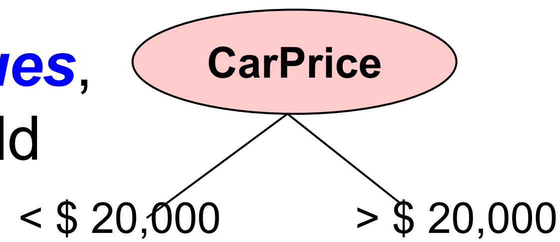
- Separate attribute lists maintained in each node of tree

C4.5 – Possible Mechanisms for Tests

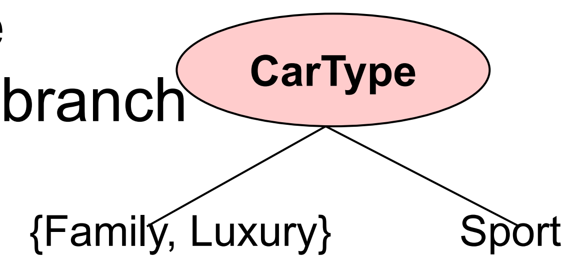
a. “standard” test on a ***discrete attribute***: one branch for each possible value of that attribute



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 and branch for each group



New example (1) Threshold Finding with Gain

Sometimes we have to find the threshold and the attribute

Database T

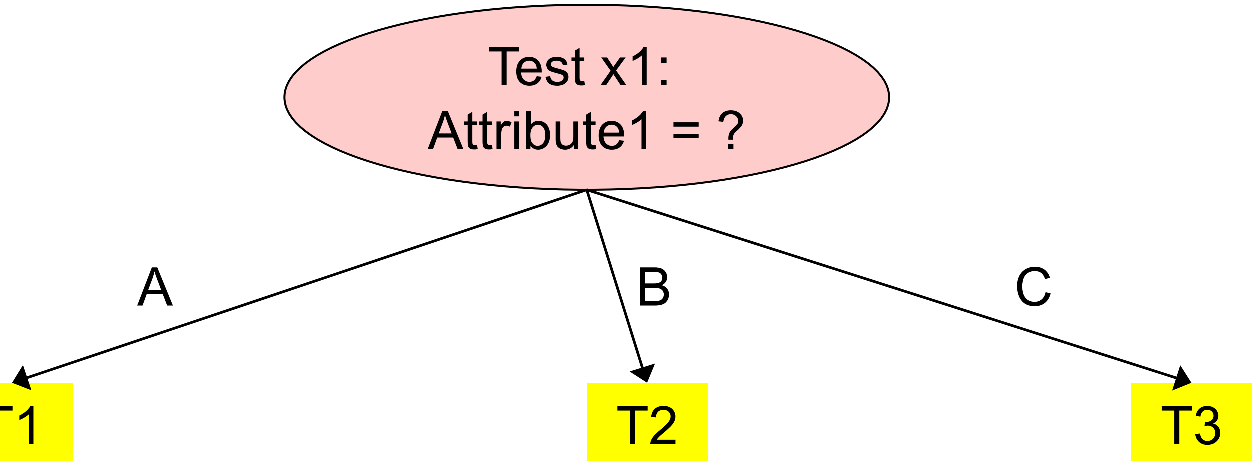
Attribute 1	Attribute 2	Attribute 3	Class
A	70	True	Class**1**
A	90	True	Class**2**
A	85	False	Class**2**
A	95	False	Class**2**
A	70	False	Class**1**
B	90	True	Class**1**
B	78	False	Class**1**
B	65	True	Class**1**
B	75	False	Class**1**
C	80	True	Class**2**
C	70	True	Class**2**
C	80	False	Class**1**
C	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), and the corresponding process of information gain computation for the test *x*3 (Attribute2 ≤ 80 or Attribute2 > 80)
- $\text{Info}_{x3}(T) = 9/14\cdot(-\,7/9\cdot\log_2(7/9) - 2/9\cdot\log_2(2/9))$
  $+\ 5/14\cdot(-\,2/5\cdot\log_2(2/5) - 3/5\cdot\log_2(3/5))$
  $=\ 0.837$ bits
- Gain(*x*3) = 0.940 – 0.837 = 0.103 bits

Attribute1 gives the highest gain of 0.246 bits, and therefore this attribute will be selected for the first splitting

New Example (2) Initial Decision Tree



T1

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**

T2

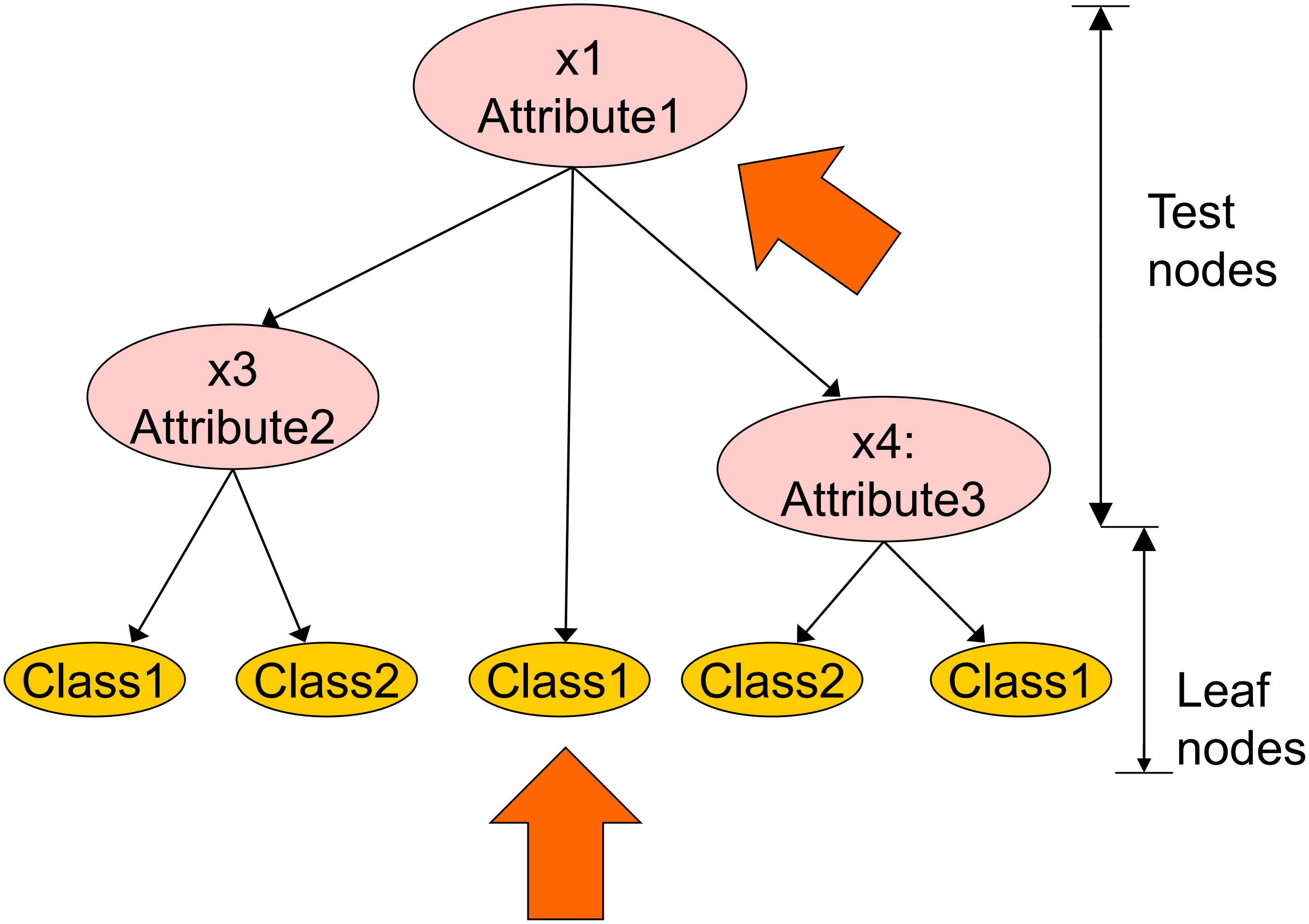
Att. 2	Att. 3	Class
90	True	Class**1**
78	False	Class**1**
65	True	Class**1**
75	False	Class**1**

T3

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 **T**

New example (3) Final Decision Tree



All of them are in CLASS1

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 v partitions) to classify D: $Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$
  - 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
- C4.5 (a successor of ID3) uses gain ratio to overcome this problem (***normalization*** of ***information gain***)

$$SplitInfo = -\sum_{i=1}^{n} \left(\frac{|T_i|}{|T|} \cdot \log_2\left(\frac{|T_i|}{|T|} \right) \right) \qquad GainRatio(X) = Gain(X) / SplitInfo(X)$$

  - **Example**:
    SplitInfo(x1) = – $^5/_{14} \cdot \log_2(^5/_{14})$ – $^4/_{14} \cdot \log_2(^4/_{14})$ - $^5/_{14} \cdot \log_2(^5/_{14})$ = 1.577 bits
    GainRatio(x1) = 0.246 / 1.557 = 0.156

    (x1 was on attribute 1 – see on one of the previous slides)

C4.5 Algorithm for Continuous Numeric Values

- Define binary test with outcomes $Y \leq Z$ and $Y > Z$, based on comparing the value of attribute against a threshold value Z
- Threshold value Z:
  - Sort training samples on the values of chosen attribute Y
    - Number of these values is ***finite***
    - Notation for sorted order: $\{v_1, v_2, \ldots, v_m\}$
  - 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 $\{v_{\mathrm{i+1}}, v_{\mathrm{i+2}}, \ldots, v_{\mathrm{m}}\}$.
    - m-1 possible splits on Y,
    - ***Optimal split***: examine all ***systematically***
  - Normal choice as representative threshold: midpoint of each interval: $(v_{\mathrm{i}} + v_{\mathrm{i+1}})/2$
    - C4.5 chooses a ***smaller*** value v_{i} for every interval $\{v_{\mathrm{i}}, v_{\mathrm{i+1}}\}$, rather ***than*** the ***midpoint*** itself as the threshold

C4.5 Algorithm: Unknown Values

- In C4.5 it is accepted as a principle that
  - Samples with the unknown values are ***distributed probabilistically*** according to the ***relative frequency*** of known values

- The new gain criterion will have the form:

$$Gain(x) = F\ (\ Info(T) - Info_x(T))$$

  - *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)

$\text{Info}(T) = -8/13\ \log_2 (8/13)\ -5/13\ \log_2 (5/13)$

$= \mathbf{0.961\ bits}$

Eight out of the thirteen cases with values for Attribute1 belong to CLASS1 and five cases to CLASS2

test x_1 represents the selection of one of three values A, B, or C

$$\begin{aligned}\text{Info}_{x1}(T) &= 5/13\ (-2/5 \log_2 (2/5) - 3/5 \log_2 (3/5)) \\ &+ 3/13\ (-3/3 \log_2 (3/3) - 0/3 \log_2 (0/3)) \\ &+ 5/14\ (-3/5 \log_2 (3/5) - 2/5 \log_2 (2/5)) \\ &= \mathbf{0.747\ bits}\end{aligned}$$

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**

Gain (x_1) = **13/14** (0.961 – 0.747) = **0.199 bits**

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 in subsets with corresponding weight factors

T1: 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***

T1: 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

T1: 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 ***T* with known value** is assigned to subset T_i, its probability belonging to ***T_i* is 1**, and in all other subsets is 0
- C4.5 associates with each sample (having ***missing value***) in each subset T_i a **weight *w*** representing the ***probability*** that the case belongs to each subset:

$$\mathbf{w_{new} = w_{old} \cdot P(T_i)}$$

- Splitting set T using test x_1 on Attribute1. New weights w_i will be ***equal to probabilities, in this case***: 5/13, 3/13, and 5/13, because initial (old) value for w is equal to 1

$|T_1| = 5+\mathbf{5/13}$, $|T_2| = 3+\mathbf{3/13}$, and $|T_3| = 5+\mathbf{5/13}$.

- The decision tree leaves are defined with two new parameters: $\mathbf{(|T_i| / E)}$
- $|T_i|$ is the sum of the ***fractional samples*** that reach the leaf, & ***E*** is the ***number of samples*** that belong to classes other than nominated class
- (3.4 / 0.4) means 3.4 (or 3 + 5/13) fractional training samples reached leaf, of which 0.4 (or 5/13) did not belong to the class of the leaf

Partitioning – Example

- Decision tree for the database T 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);
                      Else
                            Classification = CLASS1     (3.0 / 0).
```

(|Ti|/E). |Ti| is the sum of the fractional samples that reach the leaf,
E is the 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***
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle ***missing attribute values***
  - Assign the most common value of the attribute
  - 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

Decision Tree Algorithms – Building and Pruning

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

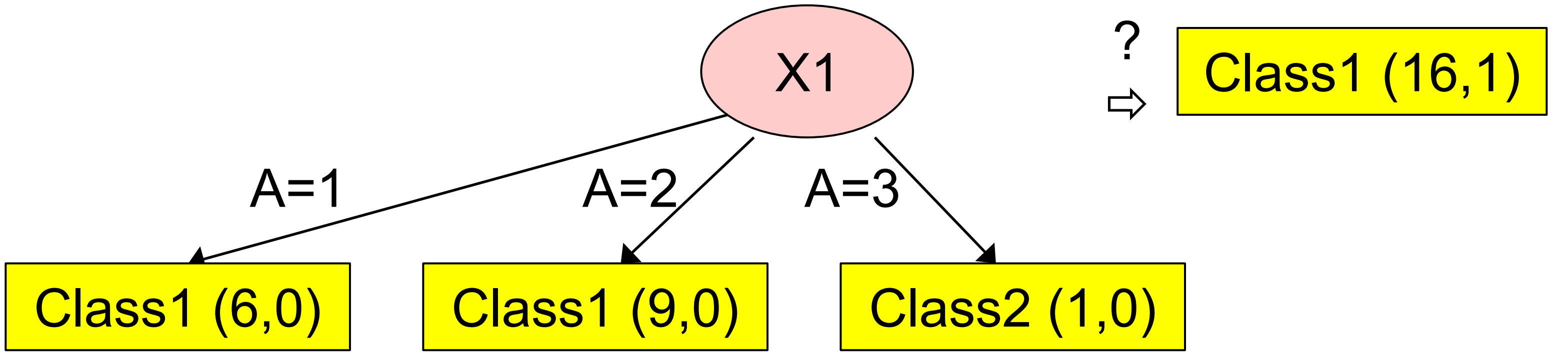
- ***Pruning phase***
  - Smaller imperfect decision tree generally achieves better accuracy.
  - Prune leaf nodes recursively to prevent over-fitting.

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 is in 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

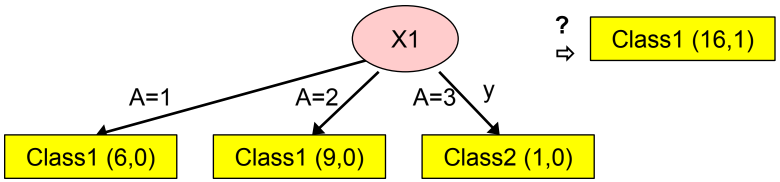
- ***Pruning***: Discarding one or more subtrees and replacing them with leaves
  - C4.5 follows a ***postpruning*** approach (pessimistic pruning)



will we replace this subtree with a single leaf node?

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

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



# of samples in the node

confidence limit (for the node): from statistical tables for binominal distributions

- Using default confidence of 25%, **upper confidence limits** for all nodes are collected from statistical tables:

  U25%(6,0) = 0.206, U25%(9,0) = 0.143, U25%(1,0) = 0.750,

  and U25%(16,1) = 0.157.

- **Predicted errors** for the initial tree and replaced node are:
  - PEtree = 6 · 0.206 + 9 · 0.143 + 1 · 0.750 = 3.257
  - PEnode = 16 · 0.157 = 2.512
  - Since PEtree > Penode, replace the subtree with the new leaf node.

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*”

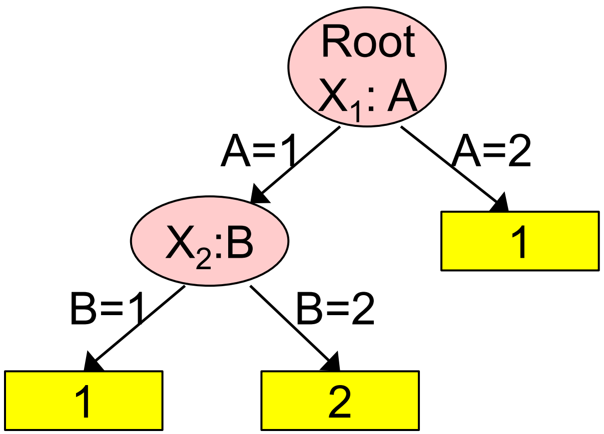
Rule Ordering

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: decreasing order of misclassification cost per class
- Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality 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 **T:**

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);

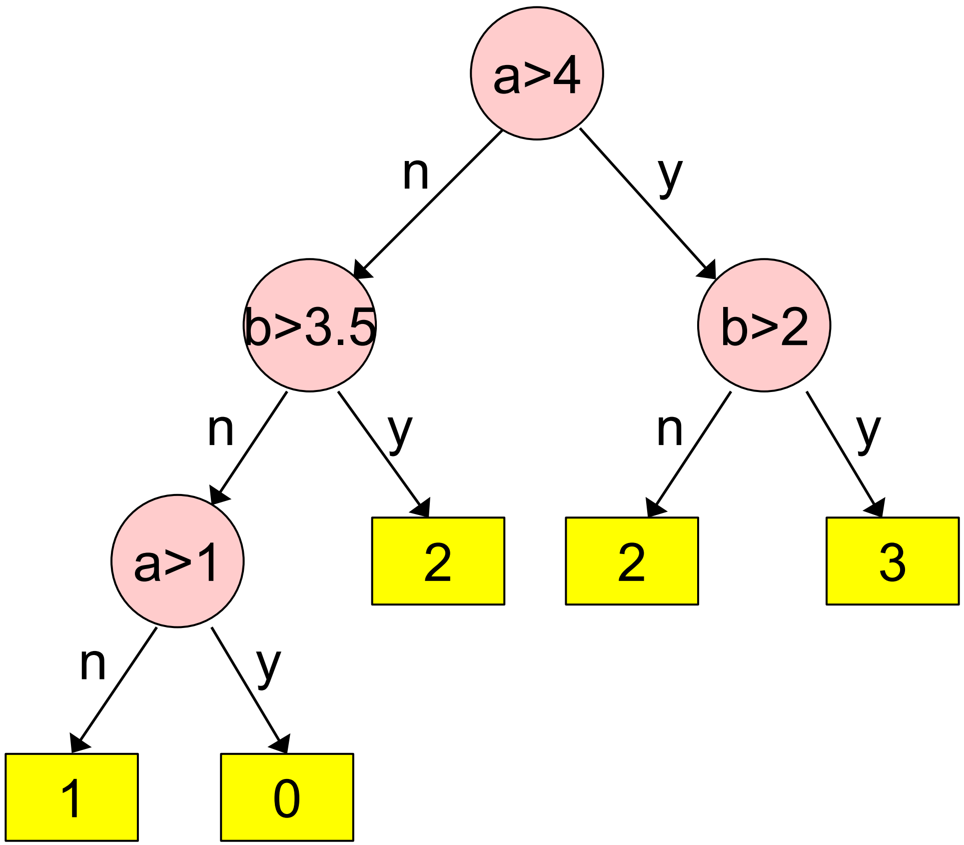
If Attribute1 = B
Then Classification = CLASS1 (3.2 / 0);

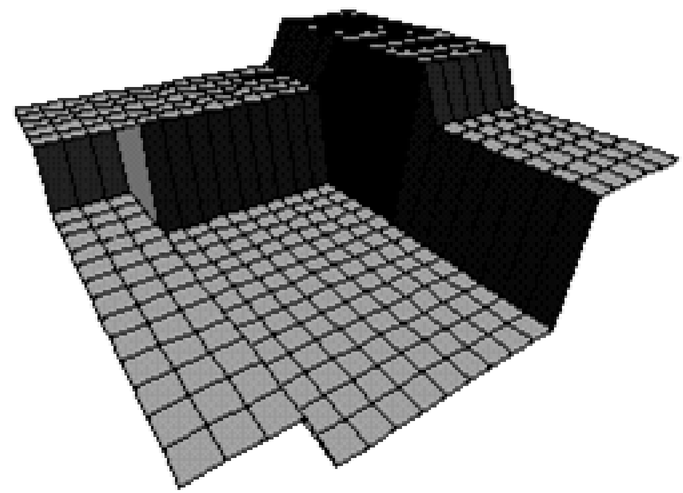
If Attribute1 = C and Attribute3 = True
Then Classification = CLASS2 (2.4 / 0);

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)

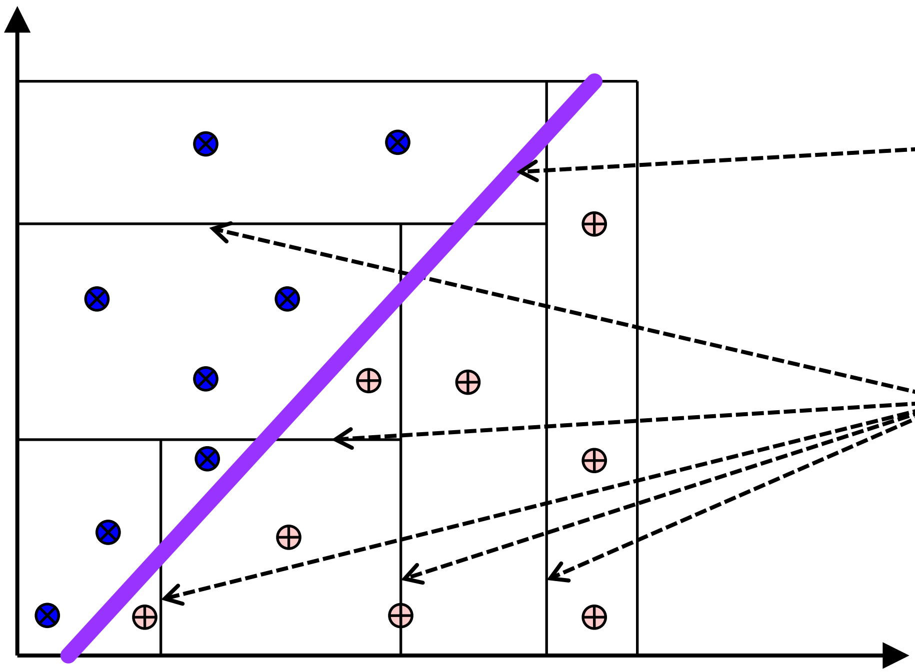




Example:

- 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)

- Given class is supported if ***k out of n*** conditions are presented.
- 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**:

$B1 = A1 \vee A2 \vee A3$

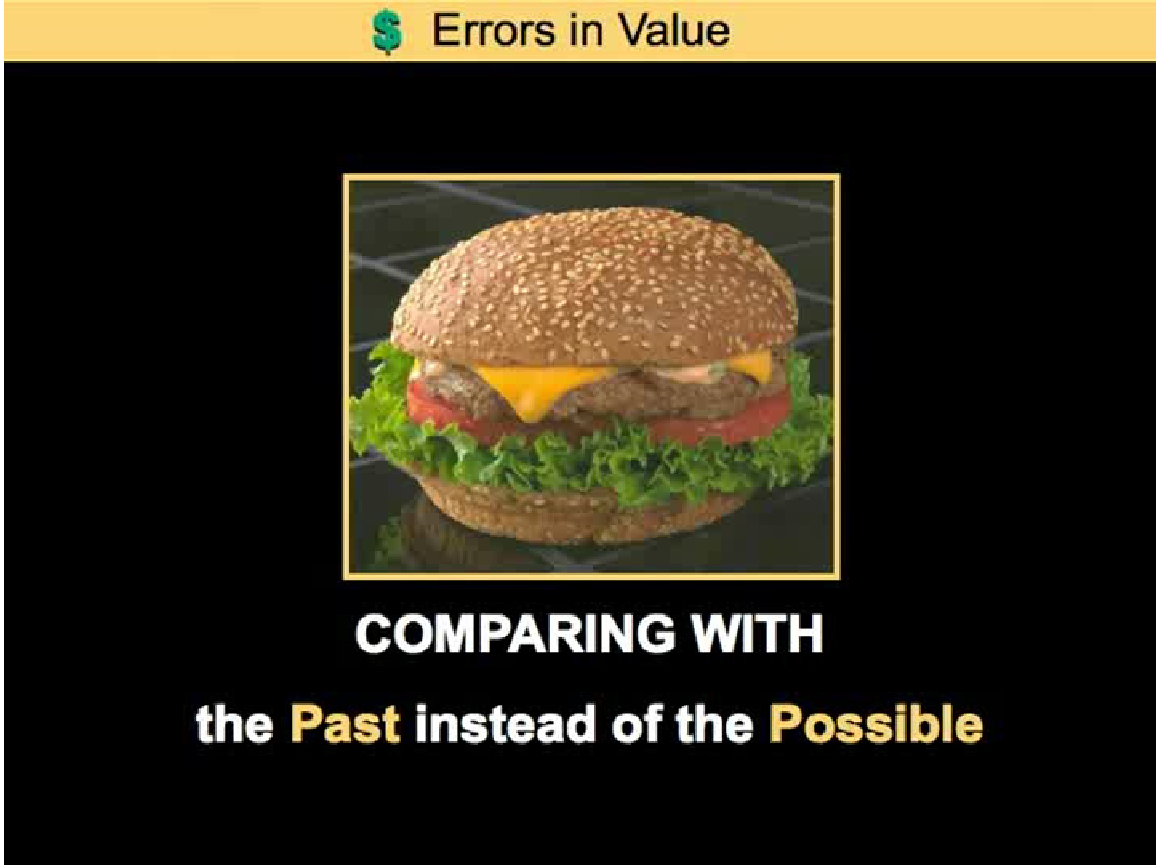
$B2 = A4 \vee A5 \vee A6$ $\rightarrow$ $\mathbf{B1 \wedge B2 \wedge B3 \rightarrow C1}$

$B3 = A7 \vee A8 \vee A9$

Decision Trees (Summary)

- Advantages
  - automatically creates tree representations from data
  - can discover “new” rules (watch out for counter-intuitive rules)
  - extensively used in data mining
  - identifies most discriminating attribute first
  - trees can be converted to rules
- Disadvantages
  - several identical examples have same effect as a single example
  - trees can become large and difficult to understand
  - cannot deal with contradictory examples
  - examines attributes individually: does not consider effects of inter-attribute relationships
  - can produce counter-intuitive rules

Limitations: Decisions over Time



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