# Discovering Decision Trees

JERZY STEFANOWSKI
Institute of Computing Science
Poznań University of Technology

Lecture 5 SE Master Course, 2008/9 = revised 2010

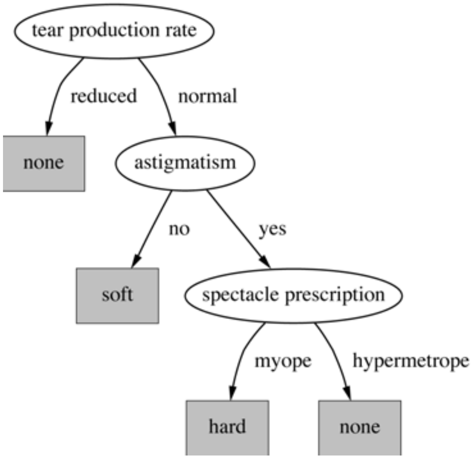
## Aims of this module

- The decision tree representation.
- The basic algorithm for inducing trees (Quinaln’s ID3).
- Heuristic search (which is the best attribute):
  - Impurity measures, entropy, gini index…
- Handling real / imperfect data (extensions in C4.5).
  - Multivalued attributes and binary trees
  - Continuous valued attributes
- Overfitting and pruning decision trees.
- Some examples.
- Software implementations

# The contact lenses data

| Age | Spectacle prescription | Astigmatism | Tear production rate | Recommended lenses |
|---|---|---|---|---|
| Young | Myope | No | Reduced | None |
| Young | Myope | No | Normal | Soft |
| Young | Myope | Yes | Reduced | None |
| Young | Myope | Yes | Normal | Hard |
| Young | Hypermetrope | No | Reduced | None |
| Young | Hypermetrope | No | Normal | Soft |
| Young | Hypermetrope | Yes | Reduced | None |
| Young | Hypermetrope | Yes | Normal | hard |
| Pre-presbyopic | Myope | No | Reduced | None |
| Pre-presbyopic | Myope | No | Normal | Soft |
| Pre-presbyopic | Myope | Yes | Reduced | None |
| Pre-presbyopic | Myope | Yes | Normal | Hard |
| Pre-presbyopic | Hypermetrope | No | Reduced | None |
| Pre-presbyopic | Hypermetrope | No | Normal | Soft |
| Pre-presbyopic | Hypermetrope | Yes | Reduced | None |
| Pre-presbyopic | Hypermetrope | Yes | Normal | None |
| Presbyopic | Myope | No | Reduced | None |
| Presbyopic | Myope | No | Normal | None |
| Presbyopic | Myope | Yes | Reduced | None |
| Presbyopic | Myope | Yes | Normal | Hard |
| Presbyopic | Hypermetrope | No | Reduced | None |
| Presbyopic | Hypermetrope | No | Normal | Soft |
| Presbyopic | Hypermetrope | Yes | Reduced | None |
| Presbyopic | Hypermetrope | Yes | Normal | None |

# A decision tree for this problem



witten&eibe

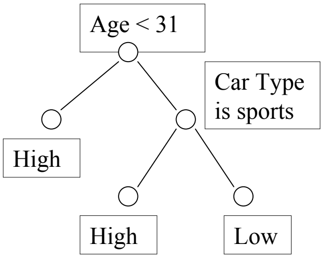
# Induction of decision trees

Decision tree: a directed graph, where nodes corresponds to some tests on attributes, a branch represents an outcome of the test and a leaf corresponds to a class label.

A new case is classified by following a matching path to a leaf node.

The problem: given a learning set, induce automatically a tree

| Age | Car Type | Risk |
|---|---|---|
| 20 | Combi | High |
| 18 | Sports | High |
| 40 | Sports | High |
| 50 | Family | Low |
| 35 | Minivan | Low |
| 30 | Combi | High |
| 32 | Family | Low |
| 40 | Combi | Low |



# General issues

- Basic algorithm: a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner.
  - TDIDT → Top Down Induction of Decision Trees.
- Key issues:
  - **Splitting criterion**: splitting examples in the node / how to choose attribute / test for this node.
  - **Stopping criterion**: when should one stop growing the branch of the tree.
  - **Pruning**: avoiding overfitting of the tree and improving classification performance for the difficult data.
- Advantages:
  - mature methodology, efficient learning and classification.

## Search space

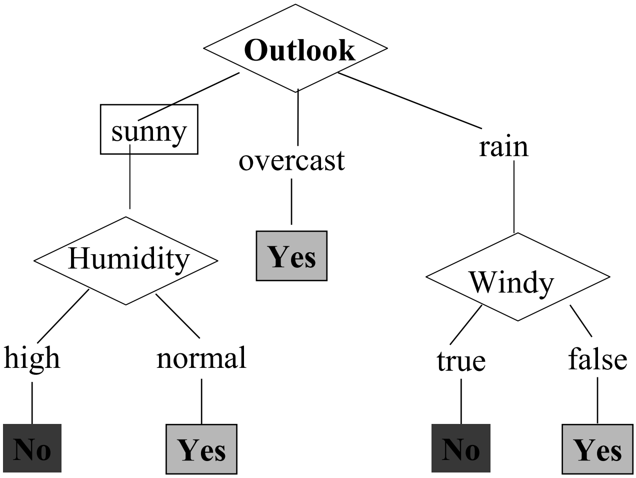
- All possible sequences of all possible tests
- Very large search space, e.g., if N binary attributes:
  - 1 null tree
  - N trees with 1 (root) test
  - N*(N-1) trees with 2 tests
  - N*(N-1)*(N-1) trees with 3 tests
  - and so on
- Size of search space is exponential in number of attributes
  - too big to search exhaustively!!!!

## Weather Data: Play or not Play?

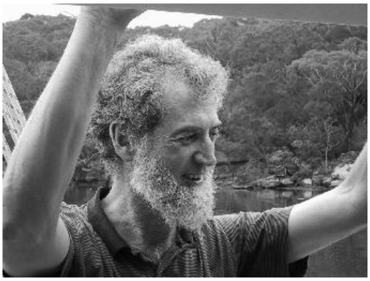
| Outlook | Temperature | Humidity | Windy | Play? |
|---|---|---|---|---|
| sunny | hot | high | false | No |
| sunny | hot | high | true | No |
| overcast | hot | high | false | Yes |
| rain | mild | high | false | Yes |
| rain | cool | normal | false | Yes |
| rain | cool | normal | true | No |
| overcast | cool | normal | true | Yes |
| sunny | mild | high | false | No |
| sunny | cool | normal | false | Yes |
| rain | mild | normal | false | Yes |
| sunny | mild | normal | true | Yes |
| overcast | mild | high | true | Yes |
| overcast | hot | normal | false | Yes |
| rain | mild | high | true | No |

*Note:*
*All attributes*
*are nominal*

# Example Tree for “Play?”


Outlook
sunny
overcast
rain
Humidity
Yes
Windy
high
normal
true
false
No
Yes
No
Yes

# J.Ross Quinlan

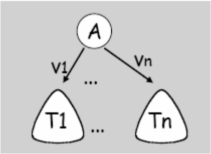


Ross Quinlan completed a PhD in Computer Science at the University of Washington in 1968. He has developed several algorithms used in machine learning and data mining such as ID3, C4.5, FOIL, and more recent commercial systems such as See5 and Cubist. He has held permanent appointments at the University of Sydney, University of Technology Sydney, Rand Corporation, and visiting appointments at Carnegie-Mellon University, MIT, GTE, and Stanford University. He currently heads a small data mining tools company and is an Adjunct Professor at the University of New South Wales. He is a Fellow of the American Association for Artificial Intelligence and the Australian Computer Society.

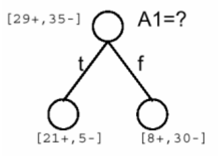
- More of his papers – have a look at http://www.rulequest.com/Personal/
- See also http://en.wikipedia.org/wiki/Ross_Quinlan

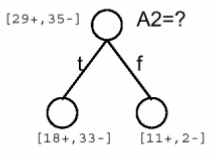
## Basic TDIDT algorithm (simplified Quinlan’s ID3)

- At start, all training examples *S* are at the root.
- **If** all examples from S belong to the same class Kj  
  **then** label the root with Kj  
  **else**
  - select the „best” attribute A
  - divide S into S1, …, Sn according to values v1, …, vn of attribute A
  - Recursively build subtrees T1, …, Tn for S1, …,Sn



## Which attribute is the best?





$P_+$ and $P_-$ are *a priori* class probabilities in the node S, test divides the S set into $S_t$ and $S_f$.

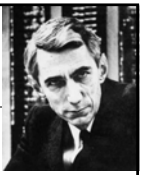
- The attribute that is most useful for classifying examples.
- We need a goodness / (im)purity function → measuring how well a given attribute separates the learning examples according to their classification.
- Heuristic: prefer the attribute that produces the "purest" sub-nodes and leads to the smallest tree.

## A criterion for attribute selection

Impurity functions:

- Given a random variable *x* with *k* discrete values, distributed according to P={p1,p2,…pk}, a impurity function $\Phi$ should satisfies:
  - $\Phi$(P)≥0 ; $\Phi$(P) is minimal if $\exists$i such that pi=1;
    $\Phi$(P) is maximal if $\forall$i 1≤i ≤ k , pi=1/k
    $\Phi$(P) is symmetrical and differentiable everywhere in its range
- The goodness of split is a reduction in impurity of the target concept after partitioning S.
- Popular function: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces

## Computing information entropy



- Entropy information (originated from Shannon)
  - Given a probability distribution, the info required to predict an event is the distribution's *entropy*
  - Entropy gives the information required in bits (this can involve fractions of bits!)
  - The amount of information, needed to decide if an arbitrary example in *S* belongs to class $K_j$ ($p_j$ - prob. it belongs to $K_j$ ).
- Basic formula for computing the entropy for examples in *S*:

$$\mathrm{entropy}(S) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

- A conditional entropy for splitting examples *S* into subsets *Si* by using an attribute A:

$$entropy(S \mid A) = \sum_{i=1}^{m} \frac{|S_i|}{|S|} \cdot entropy(S_i)$$

- Choose the attribute A with the maximal **info gain**:

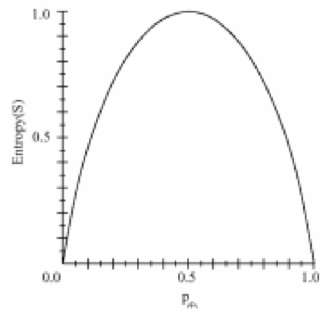
$$entropy(S) - entropy(S \mid A)$$

# Entropy interpretation

- Binary classification problem

$$E(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

- The entropy function relative to a Boolean classification, as the proportion **p+** of positive examples varies between 0 and 1
- Entropy of "pure" nodes (examples from one class) is 0;
- Max. entropy is for a node with mixed samples $P_i$=1/2.



Plot of *Ent*(S)
for $P_+ = 1 - P_-$

# Weather Data: Play or not Play?

| Outlook | Temperature | Humidity | Windy | Play? |
|---|---|---|---|---|
| sunny | hot | high | false | No |
| sunny | hot | high | true | No |
| overcast | hot | high | false | Yes |
| rain | mild | high | false | Yes |
| rain | cool | normal | false | Yes |
| rain | cool | normal | true | No |
| overcast | cool | normal | true | Yes |
| sunny | mild | high | false | No |
| sunny | cool | normal | false | Yes |
| rain | mild | normal | false | Yes |
| sunny | mild | normal | true | Yes |
| overcast | mild | high | true | Yes |
| overcast | hot | normal | false | Yes |
| rain | mild | high | true | No |

*Note:*
*All attributes*
*are nominal*

# Entropy Example from the Dataset

Information before split / no attributes, only decision class label distribution

In the Play dataset we had two target classes: *yes* and *no*

Out of 14 instances, 9 classified *yes*, rest *no*

$$p_{yes} = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) = 0.41$$

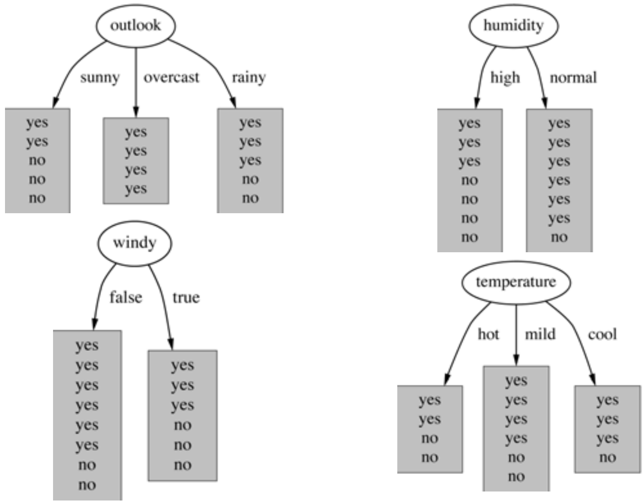
$$p_{no} = -\left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) = 0.53$$

$$E(S) = p_{yes} + p_{no} = 0.94$$

| Outlook | Temp. | Humidity | Windy | Play |
|---|---|---|---|---|
| Sunny | Hot | High | False | No |
| Sunny | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Rainy | Mild | High | False | Yes |
| Rainy | Cool | Normal | False | Yes |
| Rainy | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |

| Outlook | Temp. | Humidity | Windy | play |
|---|---|---|---|---|
| Sunny | Mild | High | False | No |
| Sunny | Cool | Normal | False | Yes |
| Rainy | Mild | Normal | False | Yes |
| Sunny | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Rainy | Mild | High | True | No |

# Which attribute to select?



witten&eibe

## Example: attribute "Outlook"

- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0$$

***Note: log(0) is not defined, but we evaluate 0*log(0) as zero***

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971$$

- Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693$$

## Computing the information gain

- Information gain:

(information before split) – (information after split)

$$\text{gain}(\text{"Outlook"}) = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247$$

- Information gain for attributes from weather data:

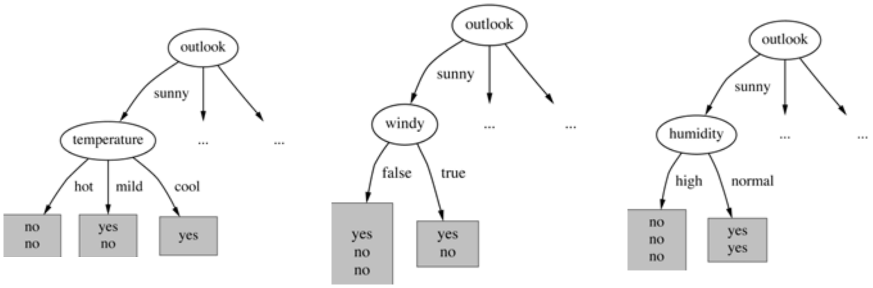
$$\text{gain}(\text{"Outlook"}) = 0.247$$

$$\text{gain}(\text{"Temperature"}) = 0.029$$

$$\text{gain}(\text{"Humidity"}) = 0.152$$

$$\text{gain}(\text{"Windy"}) = 0.048$$

## Continuing to split

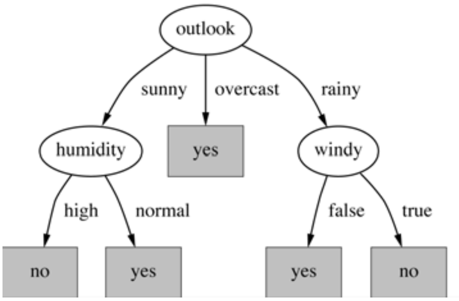


gain("Humidity") = 0.971

gain("Temperature") = 0.571

gain("Windy") = 0.020

## The final decision tree



What we have used → it is R.Quinlan's ID3 algorithm!

# ID3 algorithm (Quinlan)

Informally:

- Determine the attribute with the highest information gain on the training set (node or its subset in sub-nodes).
- Use this attribute as the root, create a branch for each of the values the attribute can have.
- Split training examples to branches depending on their attribute value.
- For each branch (splitted subsets):
  - **IF** training examples are perfectly classified, **THEN** STOP and assign a class label to this leaf
  - **ELSE** repeat the process with subset of the training set that is assigned to that branch.

# Real examples of decision trees

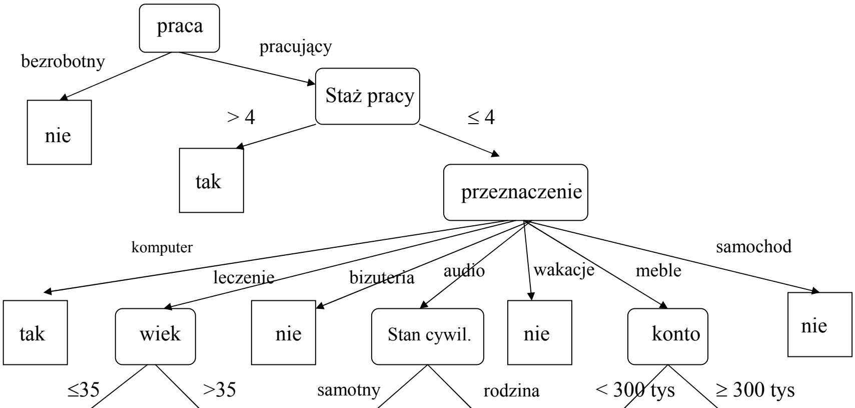
Medicine - Predicting C-Section Risk

- Learned from Medical Records of 1000 Women
- Negative Examples are Cesarean Sections
  - Prior distribution: [833+, 167-] 0.83+, 0.17-
  - *Fetal-Presentation* = 1: [822+, 167-] 0.88+, 0.12-
    - *Previous-C-Section* = 0: [767+, 81-] 0.90+, 0.10-
      - ***Primiparous* = 0: [399+, 13-] 0.97+, 0.03-**
      - ***Primiparous* = 1: [368+, 68-] 0.84+, 0.16-**
        - ***Fetal-Distress* = 0: [334+, 47-] 0.88+, 0.12-**
          - ***Birth-Weight* < 3349 0.95+, 0.05-**
          - ***Birth-Weight* ≥ 3347 0.78+, 0.22-**
        - ***Fetal-Distress* = 1: [34+, 21-] 0.62+, 0.38-**
    - *Previous-C-Section* = 1: [55+, 35-] 0.61+, 0.39-
  - *Fetal-Presentation* = 2: [3+, 29-] 0.11+, 0.89-
  - *Fetal-Presentation* = 3: [8+, 22-] 0.27+, 0.73-

## Japanese credit data

- W przykładzie wykorzystano dane dotyczące 125 osób ubiegających się o kredyty konsumpcyjne w pewnym banku w Japonii (archiwum Univ. Irvine).
- Osoby scharakteryzowane za pomocą 10 cech jakościowych i ilościowych, np. sytuacja zawodowa, przeznaczenie kredytu, płeć, stan cywilny, wiek, zarobki, stan konta, deklarowane raty, staż pracy w zakładzie (lata), ...
- Klienci byli podzielenie na dwie grupy: dobrych (mogą uzyskać kredyt) i ryzykownych.
- Cel analizy: identyfikacja reguł decyzji kredytowych, poszukiwanie profilu klientów którzy nie powinni otrzymać kredytu

## Final tree for credit data



## Other splitting criteria

- Gini index (CART, SPRINT)
  - select attribute that minimize impurity of a split
- $\chi 2$ contingency table statistics (CHAID)
  - measures correlation between each attribute and the class label
  - select attribute with maximal correlation
- Normalized Gain ratio (Quinlan 86, C4.5)
  - normalize different domains of attributes
- Distance normalized measures (Lopez de Mantaras)
  - define a distance metric between partitions of the data.
  - chose the one closest to the perfect partition
- Orthogonal (ORT) criterion
- AUC-splitting criteria (Ferri et at.)
- There are many other measures. Mingers'91 provides an experimental analysis of effectiveness of several selection measures over a variety of problems.
- Look also in a study by D.Malerba, …

## *Gini* Index – a solution from CART

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

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

  where $p_j$ is the relative frequency of class *j* in *T.*
- If a data set *T* is split into two subsets $T_1$ and $T_2$ with sizes $N_1$ and $N_2$ respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

  $$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node.

## Extracting Classification Rules from Decision Trees

- The knowledge represented in decision trees can be extracted and represented in the form of classification IF-THEN rules.
- One rule is created for each path from the root to a leaf node.
- Each attribute-value pair along a given path forms a conjunction in the rule antecedent; the leaf node holds the class prediction, forming the rule consequent.

## Extracting Classification Rules from Decision Trees

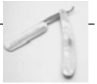
An example for the Weather nominal dataset:

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

However:

- Dropping redundant conditions in rules and rule post-pruning
- Classification strategies with rule sets are necessary

## Occam’s razor: prefer the simplest hypothesis that fits the data.

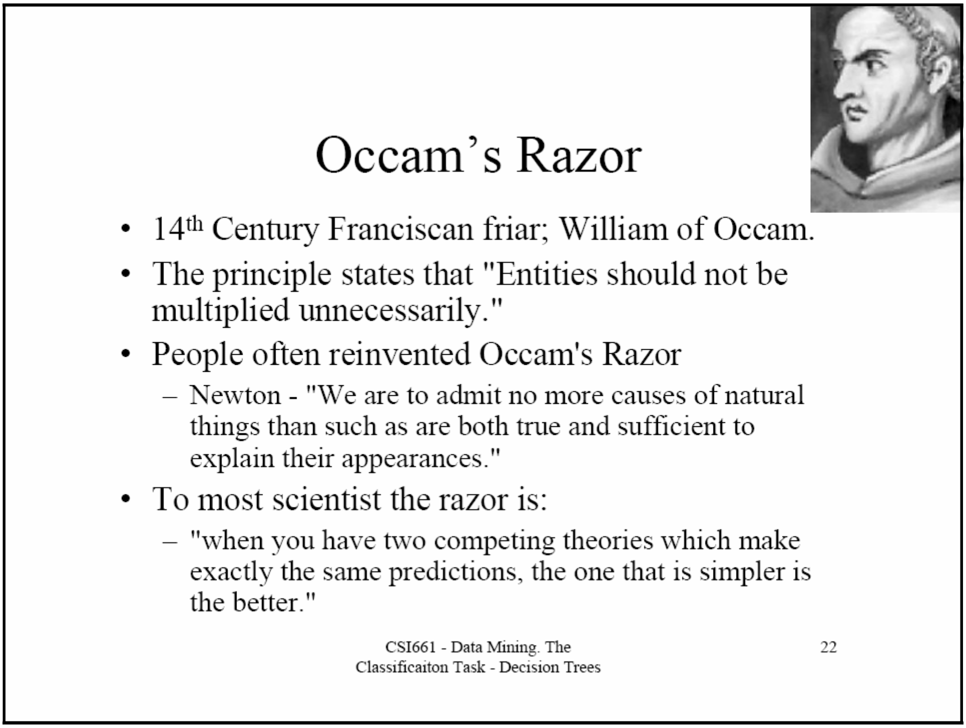


- Inductive bias → Why simple trees should be preferred?
  1. The number of simple hypotheses that may accidentally fit the data is small, so chances that simple hypothesis uncover some interesting knowledge about the data are larger.
  2. A larger tree that fits data might be coincidence
  3. Simpler trees have lower variance, they should not overfit the data that easily.
  4. Simpler trees do not partition the feature space into too many small boxes, and thus may generalize better, while complex trees may end up with a separate box for each training data sample.

  Still, even if the tree is small ...

  for small datasets with many attributes several equivalent (from the accuracy point of view) descriptions may exist.

  => one tree may not be sufficient, we need a forest of „healthy” trees! (see the lecture on ensembles and …)



## Occam’s Razor

- 14th Century Franciscan friar; William of Occam.
- The principle states that "Entities should not be multiplied unnecessarily."
- People often reinvented Occam's Razor
  - Newton - "We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances."
- To most scientist the razor is:
  - "when you have two competing theories which make exactly the same predictions, the one that is simpler is the better."

## Using decision trees for real data

- Some issues:
  - Highly branching attributes,
  - Handling continuous and missing attribute values
  - Overfitting
  - Noise and inconsistent examples
  - ….
- Thus, several extension of tree induction algorithms, see e.g. Quinlan C4.5, CART, CHAiD, Assistant86, …

## Highly-branching attributes

- Problematic: attributes with a large number of values (extreme case: ID code)
- Subsets are more likely to be pure if there is a large number of values

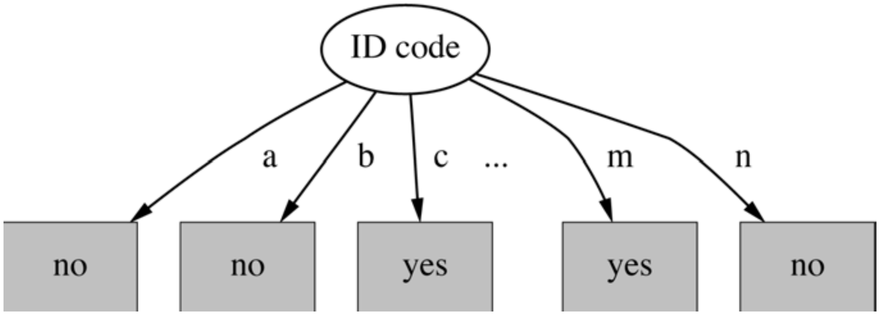
  ⇒Information gain is biased towards choosing attributes with a large number of values

  ⇒This may result in *overfitting* (selection of an attribute that is non-optimal for prediction)

# Weather Data with ID code

| ID | Outlook | Temperature | Humidity | Windy | Play? |
|---|---|---|---|---|---|
| a | sunny | hot | high | false | No |
| b | sunny | hot | high | true | No |
| c | overcast | hot | high | false | Yes |
| d | rain | mild | high | false | Yes |
| e | rain | cool | normal | false | Yes |
| f | rain | cool | normal | true | No |
| g | overcast | cool | normal | true | Yes |
| h | sunny | mild | high | false | No |
| i | sunny | cool | normal | false | Yes |
| j | rain | mild | normal | false | Yes |
| k | sunny | mild | normal | true | Yes |
| l | overcast | mild | high | true | Yes |
| m | overcast | hot | normal | false | Yes |
| n | rain | mild | high | true | No |

# Split for ID Code Attribute



Entropy of split = 0 (since each leaf node is “pure”, having only one case.

Information gain is maximal for ID code

## Gain ratio

- *Gain ratio*: a modification of the information gain that reduces its bias on high-branch attributes.
- Gain ratio takes number and size of branches into account when choosing an attribute.
  - It corrects the information gain by taking the *intrinsic information* of a split into account (i.e. how much info do we need to tell which branch an instance belongs to).

## Gain Ratio and Intrinsic Info

- Intrinsic information (a kind of a correction factor): entropy of distribution of instances into branches

$$IntrinsicInfo(S,A) \equiv -\sum \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}.$$

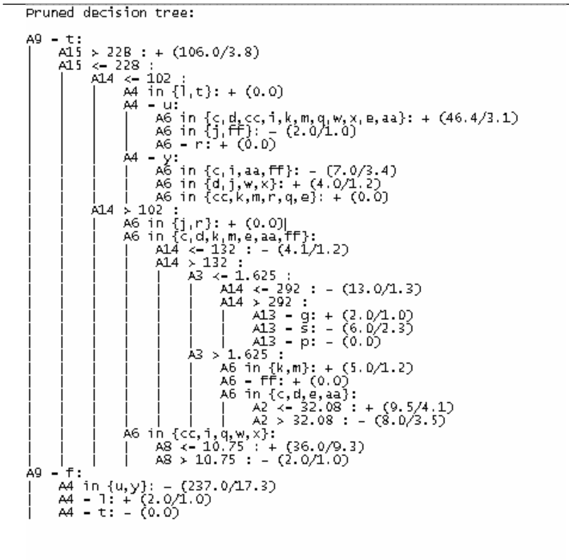
- *Gain ratio* (Quinlan′86) normalizes info gain by:

$$GainRatio(S,A) = \frac{Gain(S,A)}{IntrinsicInfo(S,A)}.$$

# Binary Tree Building

- Sometimes it leads to smaller trees or better classifiers.
- The form of the split used to partition the data depends on the type of the attribute used in the split:
    - for a continuous attribute A, splits are of the form value(A)<x where x is a value in the domain of A.
    - for a categorical attribute A, splits are of the form value(A)∈X where X⊂domain(A)

# Binary tree (Quinlan’s C4.5 output)



```
Pruned decision tree:

A9 - t:
|   A15 > 22B : + (106.0/3.8)
|   A15 <- 228 :
|   |   A14 <- 102 :
|   |   |   A4 in {l,t}: + (0.0)
|   |   |   A4 - u:
|   |   |   |   A6 in {c,d,cc,i,k,m,q,w,x,e,aa}: + (46.4/3.1)
|   |   |   |   A6 in {j,ff}: - (2.0/1.0)
|   |   |   |   A6 - r: + (0.0)
|   |   |   A4 - y:
|   |   |   |   A6 in {c,i,aa,ff}: - (7.0/3.4)
|   |   |   |   A6 in {d,j,w,x}: + (4.0/1.2)
|   |   |   |   A6 in {cc,k,m,r,q,e}: + (0.0)
|   |   A14 > 102 :
|   |   |   A6 in {j,r}: + (0.0)|
|   |   |   A6 in {c,d,k,m,e,aa,ff}:
|   |   |   |   A14 <- 132 : - (4.1/1.2)
|   |   |   |   A14 > 132 :
|   |   |   |   |   A3 <- 1.625 :
|   |   |   |   |   |   A14 <- 292 : - (13.0/1.3)
|   |   |   |   |   |   A14 > 292 :
|   |   |   |   |   |   |   A13 - g: + (2.0/1.0)
|   |   |   |   |   |   |   A13 - s: - (6.0/2.3)
|   |   |   |   |   |   |   A13 - p: - (0.0)
|   |   |   |   |   A3 > 1.625 :
|   |   |   |   |   |   A6 in {k,m}: + (5.0/1.2)
|   |   |   |   |   |   A6 - ff: + (0.0)
|   |   |   |   |   |   A6 in {c,d,e,aa}:
|   |   |   |   |   |   |   A2 <- 32.08 : + (9.5/4.1)
|   |   |   |   |   |   |   A2 > 32.08 : - (8.0/3.5)
|   |   |   A6 in {cc,i,q,w,x}:
|   |   |   |   A8 <- 10.75 : + (36.0/9.3)
|   |   |   |   A8 > 10.75 : - (2.0/1.0)
A9 - f:
|   A4 in {u,y}: - (237.0/17.3)
|   A4 - l: + (2.0/1.0)
|   A4 - t: - (0.0)
```

- Crx (Credit Data) UCI ML Repository

# Continuous valued attributes

- The real life data often contains numeric information or mixtures of different type attributes.
- It should properly handled (remind problems with highly valued attributes).
- Two general solutions:
    - The discretization in a pre-processing step (transforming numeric values into ordinal ones by finding sub-intervals)
    - Adaptation of algorithms → binary tree, new splitting conditions (A < t),…
        - While evaluating attributes for splitting condition in trees, dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals.

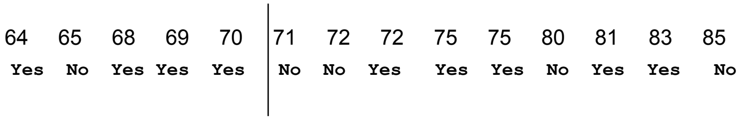
# Weather data - numeric

| Outlook | Temperature | Humidity | Windy | Play |
|---|---|---|---|---|
| Sunny | 85 | 85 | False | No |
| Sunny | 80 | 90 | True | No |
| Overcast | 83 | 86 | False | Yes |
| Rainy | 75 | 80 | False | Yes |
| … | … | … | … | … |

```
If outlook = sunny and humidity > 83 then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity < 85 then play = yes
If none of the above then play = yes
```

# Example

- Split on temperature attribute:



| 64 | 65 | 68 | 69 | 70 | 71 | 72 | 72 | 75 | 75 | 80 | 81 | 83 | 85 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Yes | No | Yes | Yes | Yes | No | No | Yes | Yes | Yes | No | Yes | Yes | No |

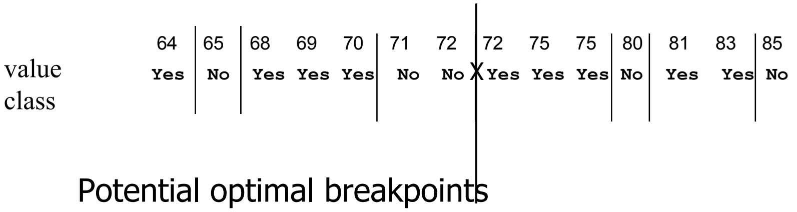
  - E.g. temperature < 71.5: yes/4, no/2
    temperature ≥ 71.5: yes/5, no/3

  - Info([4,2],[5,3])
    = 6/14 info([4,2]) + 8/14 info([5,3])
    = 0.939

- Place split points halfway between values
- Can evaluate all split points in one pass!

# Speeding up

- Entropy only needs to be evaluated between points of different classes (Fayyad & Irani, 1992)



| value | 64 | 65 | 68 | 69 | 70 | 71 | 72 | 72 | 75 | 75 | 80 | 81 | 83 | 85 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class | Yes | No | Yes | Yes | Yes | No | No X | Yes | Yes | Yes | No | Yes | Yes | No |

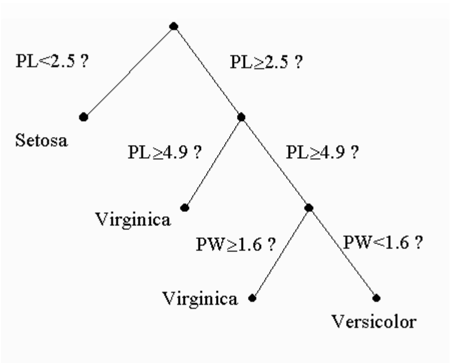
Potential optimal breakpoints

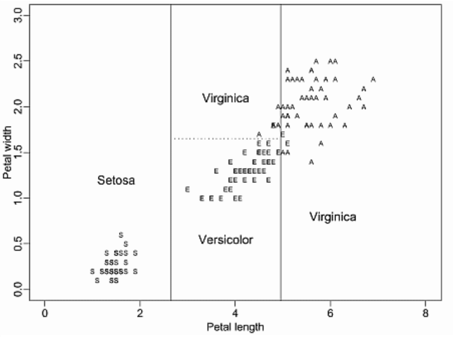
Breakpoints between values of the same class cannot be optimal

# Graphical interpretation – decision boundaries

Hierarchical partitioning of feature space into hyper-rectangles.

Example: Iris flowers data, with 4 features; displayed in 2-D.





# Summary for Continuous and Missing Values

- Sort the examples according to the continuous attribute A, then identify adjacent examples that differ in their target classification, generate a set of candidate thresholds, and select the one with the maximum gain.
- Extensible to split continuous attributes into multiple intervals.
- Assign missing attribute values either
  - Assign the most common value of *A(x).*
  - Assign probability to each of the possible values of *A*. These probabilities are estimated based on the observed frequencies of the values of *A*. These probabilities are used in the information gain measure.
  - More advanced approaches ….

## Missing values – advanced (C4.5 solution)

Split instances with missing values into pieces

  - A piece going down a branch receives a weight proportional to the popularity of the branch
  - weights sum to 1

- Info gain works with fractional instances
  - use sums of weights instead of counts
- During classification, split the instance into pieces in the same way
  - Merge probability distribution using weights

## Handling noise and imperfect examples

Sources of imperfection.

- Random errors (noise) in training examples
  - erroneous attribute values.
  - erroneous classification.
- Too sparse training examples.
- Inappropriate / insufficient set of attributes.
- Missing attribute values.

# A Problem of Weather Data Again …

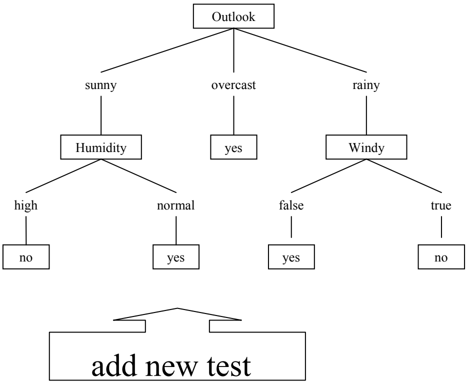
| Outlook | Temperature | Humidity | Windy | Play? |
|---|---|---|---|---|
| sunny | hot | high | false | No |
| sunny | hot | high | true | No |
| overcast | hot | high | false | Yes |
| rain | mild | high | false | Yes |
| rain | cool | normal | false | Yes |
| rain | cool | normal | true | No |
| overcast | cool | normal | true | Yes |
| sunny | mild | high | false | No |
| **sunny** | **cool** | **normal** | **false** | **Yes** |
| rain | mild | normal | false | Yes |
| sunny | mild | normal | true | Yes |
| overcast | mild | high | true | Yes |
| overcast | hot | normal | false | Yes |
| rain | mild | high | true | No |

*Note:*
*All examples are consistent*

# Inconsistent examples and overfitting

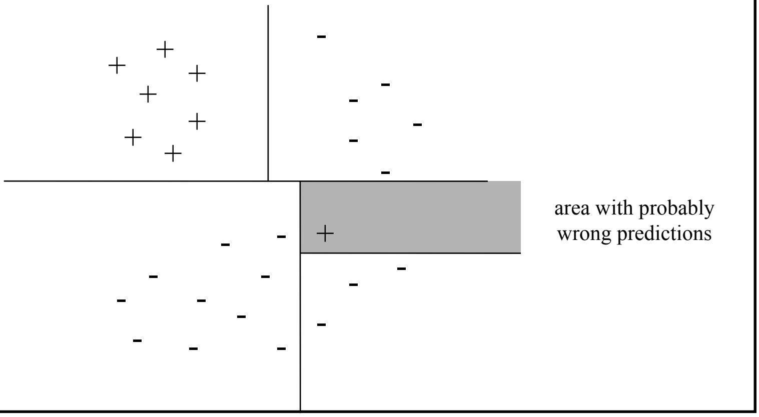
- ***Noisy training instances.*** Consider an noisy training example:

Outlook = *Sunny;*Temperature = *Cool;* Humidity = *Normal;* Wind = *False;* PlayTennis = *No*



# Reasons for Overfitting

• ***Small number of instances are associated with leaf nodes.*** In this case it is possible that for coincidental regularities to occur that are unrelated to the actual target concept.

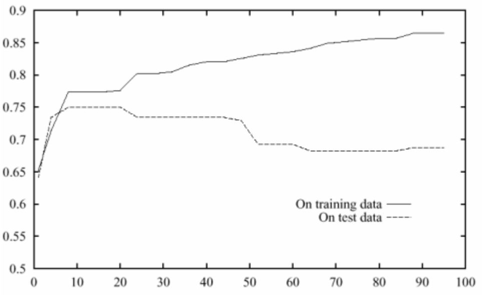


# Overfitting the Data

- The basic algorithm → grows each branch of the tree just deeply enough to sufficiently classify the training examples.
  - Reasonable for perfect data and a descriptive perspective of KDD; However, …
  - Occam razor and generality abilities
- When there is „noise" in the dataset or the data is not representative sample of the true target function …
  - The tree may *overfit* the learning examples
- Definition: The tree / classifier *h* is said to overfit the training data, if there exists some alternative tree *h'*, such that it has a smaller error than *h* over the entire distribution of instances (although *h* may has smaller error than *h'* on the training data).

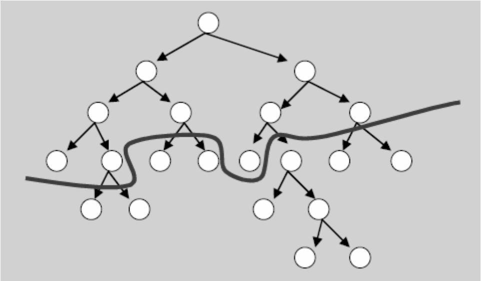
## Overfitting in Decision Tree Construction

- Accuracy as a function of the number of tree nodes: on the training data it may grow up to 100%, but the final results may be worse than for the majority classifier!



## Tree pruning

- Avoid overfitting the data by tree pruning.
- After pruning the classification accuracy on unseen data may increase!



## Avoid Overfitting in Classification - Pruning

- Two approaches to avoid overfitting:
  - (Stop earlier / Forward pruning): Stop growing the tree earlier – extra stopping conditions, e.g.
    1. Stop splitting the nodes if the number of samples is too small to make reliable decisions.
    2. Stop if the proportion of samples from a single class (node purity) is larger than a given threshold - forward pruning
  - (Post-pruning): Allow overfit and then post-prune the tree.
    - Estimation of errors and tree size to decide which sub-tree should be pruned.

## Remarks on pre-pruning

- The number of cases in the node is less than the given threshold.
- The probability of predicting the strongest class in the node is sufficiently high.
- The best splitting evaluation criterion (e.g. entropy) is not greater than a certain threshold.
- The change of probability distribution is not significant.
  - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
  - Most popular test: *chi-squared test*
  - Only statistically significant attributes were allowed to be selected by information gain procedure
- …

## Remarks on pre-pruning (2)

It seems to be right but remember about …

- hard to properly evaluate node split without seeing what splits would follow it (use a lookahead technique?)
- some attributes useful only in combination with other attributes

On the other hand

It is less computationally expensive than post-pruning!

## Reduced Error Post-pruning

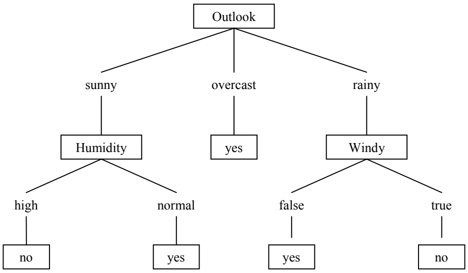
**Split data into training and validation sets.**

**Pruning a decision node *d* consists of:**

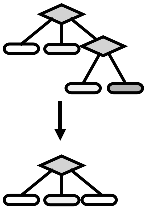
1. removing the subtree rooted at *d*.
2. making *d* a leaf node.
3. assigning *d* the most common classification of the training instances associated with *d*.

**Do until further pruning is harmful:**

1. Evaluate impact on validation set of pruning each possible node (plus those below it).
2. Greedily remove the one that most improves validation set accuracy.



## Reduced-Error Pruning – REP method



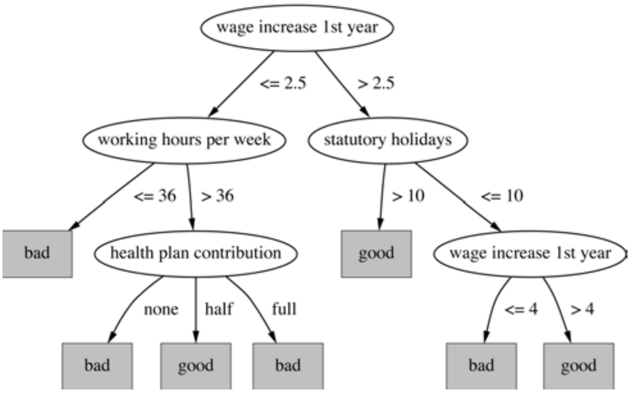
- Post-Pruning, Cross-Validation Approach
- Split Data into Training and Validation Sets
- Function *Prune*(*T, node*)
  - Remove the subtree rooted at *node*
  - Make *node* a leaf (with majority label of associated examples)
- Algorithm *Reduced-Error-Pruning* (*D*)
  - Partition *D* into $D_{train}$ (training / “growing”), $D_{validation}$ (validation / “pruning”)
  - Build complete tree *T* using *ID3* on $D_{train}$
  - UNTIL accuracy on $D_{validation}$ decreases DO

```
UNTIL accuracy on $D_{validation}$ decreases DO
  FOR each non-leaf node candidate in T
      Temp[candidate] ← Prune (T, candidate)
      Accuracy[candidate] ← Test (Temp[candidate], $D_{validation}$)
  T ← T’ ∈ Temp with best value of Accuracy (best increase; greedy)
```

  - RETURN (pruned) *T*

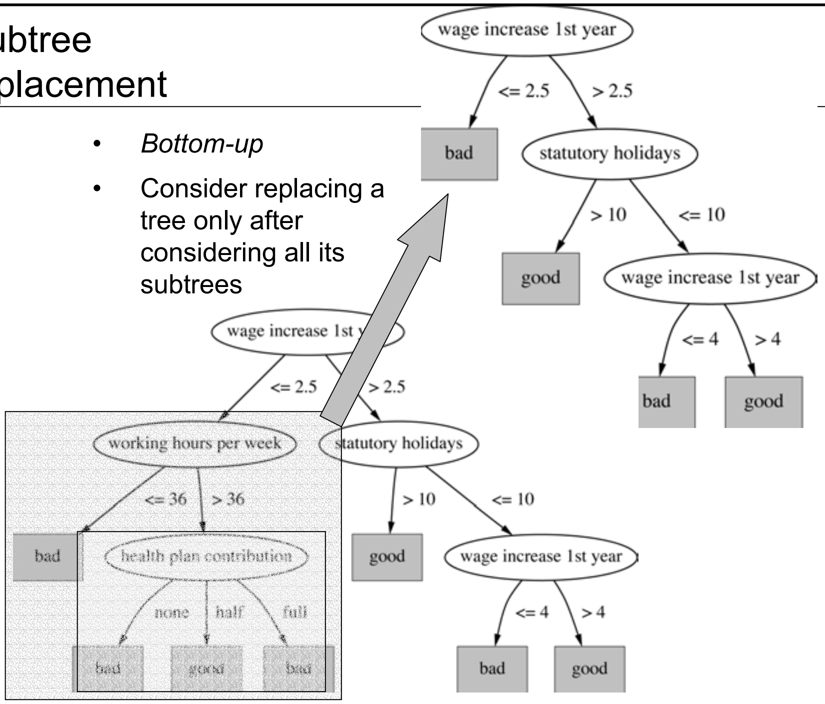
## Post-pruning

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees
- Ex: labor negotiations



# Subtree replacement

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees



# Cost-complexity approach

- Consider both size and error estimate of the tree.

$$\alpha \cdot R(t) + e(t)$$

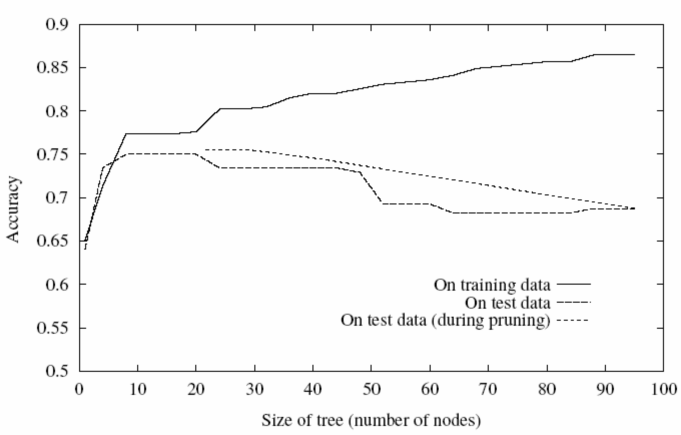
- where *t* – the current tree, *R*(*t*) – size of the tree, *e*(*t*) – estimation of the error while classifying objects, $\alpha$ - technical coefficient.
- Try to minimize it!
- A strategy to look through a family of reduced trees.
  - More advanced version in CART
- MDL principle - …

# Remarks to post-pruning

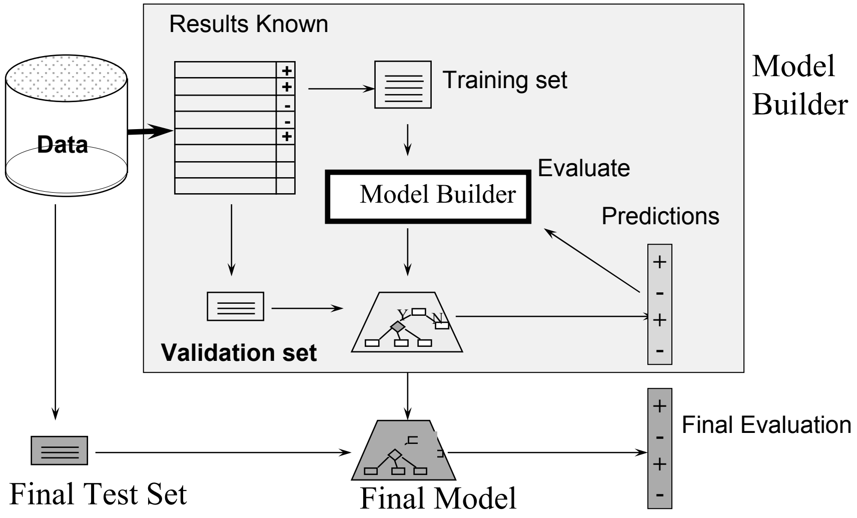
- Approaches to determine the correct final tree size:
  - Different approaches to error estimates
  - Separate training and testing sets or use cross-validation.
  - Use all the data for training, but apply a statistical test to estimate whether expanding or pruning a node may improve over entire distribution.
- Rule post-pruning (C4.5): converting to rules before pruning.
- C4.5 method – estimate of pessimistic error
  - Derive confidence interval from training data
  - Use a heuristic limit, derived from this, for pruning
  - Shaky statistical assumptions (based on training data)
  - Seems to work OK in practice
  - Option *c* parameter – default value 0,25:
    the smaller value, the stronger pruning!

# Reduced post-pruning

- Separate training and testing sets or use an extra validation (pruning one).



## Classification: Train, Validation, Test split


Results Known

Model Builder

Data

Training set

Model Builder

Evaluate

Predictions

Validation set

Final Test Set

Final Model

Final Evaluation

## From trees to rules (C4.5rule)

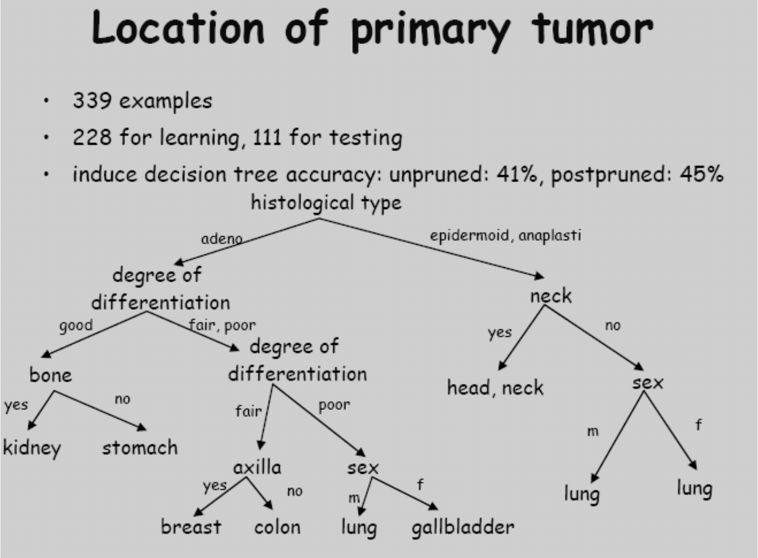
- Simple way: one rule for each leaf
- C4.5rules: greedily prune conditions from each rule if this reduces its estimated error
  - Can produce duplicate rules
  - Check for this at the end
- Then
  - look at each class in turn
  - consider the rules for that class
  - find a “good” subset (guided by MDL)
- Then rank the subsets to avoid conflicts
- Finally, remove rules (greedily) if this decreases error on the training data

## *Complexity of tree induction

- Assume
  - $m$ attributes
  - $n$ training instances
  - tree depth $O(\log n)$
- Building a tree $O(m\, n \log n)$
- Subtree replacement $O(n)$
- Subtree raising $O(n (\log n)^2)$
  - Every instance may have to be redistributed at every node between its leaf and the root
  - Cost for redistribution (on average): $O(\log n)$
- Total cost: $O(m\, n \log n) + O(n (\log n)^2)$

witten & eibe

## A real life example of tree pruning [U.Lubljana]



### Location of primary tumor

- 339 examples
- 228 for learning, 111 for testing
- induce decision tree accuracy: unpruned: 41%, postpruned: 45%

## Classification and Massive Databases

- Classification is a classical problem extensively studied by
  - statisticians
  - AI, especially machine learning researchers
- Database researchers re-examined the problem in the context of large databases
  - most previous studies used small size data, and most algorithms are memory resident
  - Classifying data-sets with millions of examples and a few hundred even thousands attributes with reasonable speed
- recent data mining research contributes to
  - Scalability
  - Generalization-based classification
  - Parallel and distributed processing

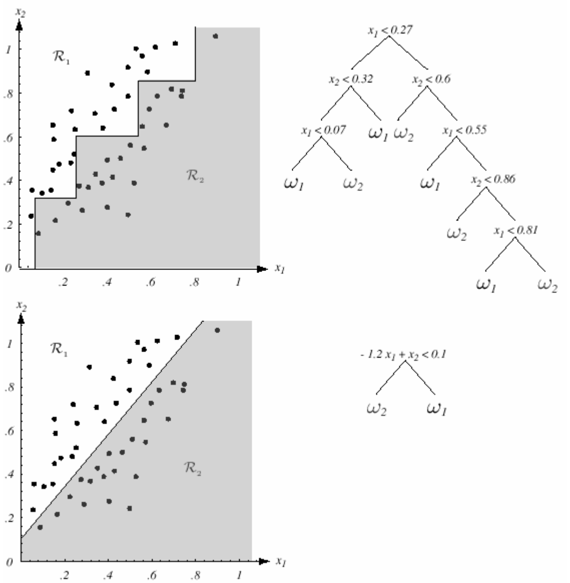
## Scalable Decision Tree Methods

- Most algorithms assume data can fit in memory.
- Data mining research contributes to the scalability issue, especially for decision trees.
- Successful examples
  - SLIQ (EDBT'96 -- Mehta et al.'96)
  - SPRINT (VLDB96 -- J. Shafer et al.'96)
  - PUBLIC (VLDB98 -- Rastogi & Shim'98)
  - RainForest (VLDB98 -- Gehrke, et al.'98)

## Previous Efforts on Scalability

- Incremental tree construction (Quinlan'86)
  - using partial data to build a tree.
  - testing other examples and those mis-classified ones are used to rebuild the tree interactively.
- Data reduction (Cattlet'91)
  - reducing data size by sampling and discretization.
  - still a main memory algorithm.
- Data partition and merge (Chan and Stolfo'91)
  - partitioning data and building trees for each partition.
  - merging multiple trees into a combined tree.
  - experiment results indicated reduced classification accuracy.

## Oblique trees



Univariate, or monothetic trees,

mult-variate, or oblique trees.

Figure from
Duda, Hart & Stork,
Chap. 8

## History of Decision Tree Research

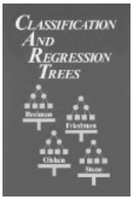
- 1960’s
  - 1966: Hunt, colleagues in psychology used full search decision tree methods to model human concept learning
- 1970’s
  - 1977: Breiman, Friedman, colleagues in statistics develop simultaneous Classification And Regression Trees (*CART*)
  - 1979: Quinlan’s first work with proto-*ID3*
- 1980’s
  - 1984: first mass publication of *CART* software (now in many commercial codes)
  - 1986: Quinlan’s landmark paper on *ID3*
  - Variety of improvements: coping with noise, continuous attributes, missing data, non-axis-parallel DTs, etc.
- 1990’s
  - 1993: Quinlan’s updated algorithm, *C4.5*
  - More pruning, overfitting control heuristics (*C5.0*, etc.); combining DTs

## C4.5 History

- ID3, CHAID – 1960s
- C4.5 innovations (Quinlan):
  - permit numeric attributes
  - deal sensibly with missing values
  - pruning to deal with for noisy data
- C4.5 - one of best-known and most widely-used learning algorithms
  - Last research version: C4.8, implemented in Weka as J4.8 (Java)
  - Commercial successor: C5.0 (available from Rulequest)

## CART – Classification And Regression Tree

- Developed 1974-1984 by 4 statistics professors
  - Leo Breiman (Berkeley), Jerome Friedman (Stanford), Charles Stone (Berkeley), Richard Olshen (Stanford)
- Focused on accurate assessment when data is noisy
- Currently distributed by Salford Systems

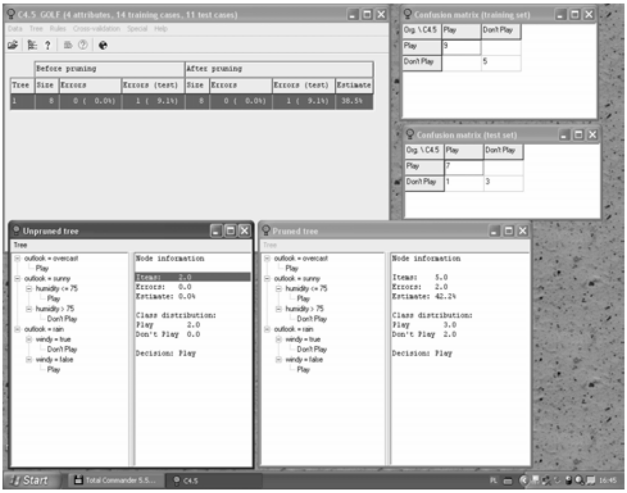


## Software implementations

- ID3 and C4.5 was easier to get free (with source code in C)
  - J4.8 available in WEKA
  - The best version C5.0 is a commercial tool www.rulequest.com
- CART → was not so easy to get free
  - Commercialy distributed by Salford Systems
    - www.salford-systems.com
  - Basis versions available in typical statistical / data mining software as Statsoft, SAS, Clementine, SPSS, etc.
- Many others – see W.Buntine IND2

# C4.5 code + our PP interface

- You can use it during lab classes


C4.5 GOLF (4 attributes, 14 training cases, 11 test cases)
Data Tree Rules Cross-validation Special Help
Confusion matrix (training set)
Confusion matrix (test set)
Unpruned tree
Pruned tree
Tree
Start
Total Commander 5.5...
C4.5
16:45

|  | Before pruning |  |  | After pruning |  |  |  |
|---|---|---|---|---|---|---|---|
| Tree | Size | Errors | Errors (test) | Size | Errors | Errors (test) | Estimate |
| 1 | 8 | 0 ( 0.0%) | 1 ( 9.1%) | 8 | 0 ( 0.0%) | 1 ( 9.1%) | 38.5% |

Confusion matrix (training set)

| Org. \ C4.5 | Play | Don't Play |
|---|---|---|
| Play | 9 |  |
| Don't Play |  | 5 |

Confusion matrix (test set)

| Org. \ C4.5 | Play | Don't Play |
|---|---|---|
| Play | 7 |  |
| Don't Play | 1 | 3 |

Unpruned tree

```
outlook = overcast
    Play
outlook = sunny
    humidity <= 75
        Play
    humidity > 75
        Don't Play
outlook = rain
    windy = true
        Don't Play
    windy = false
        Play
```

```
Node information

Items:     2.0
Errors:    0.0
Estimate:  0.0%

Class distribution:
Play        2.0
Don't Play  0.0

Decision: Play
```

Pruned tree

```
outlook = overcast
    Play
outlook = sunny
    humidity <= 75
        Play
    humidity > 75
        Don't Play
outlook = rain
    windy = true
        Don't Play
    windy = false
        Play
```

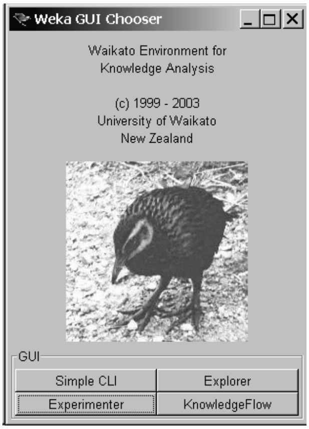
```
Node information

Items:     5.0
Errors:    2.0
Estimate:  42.2%

Class distribution:
Play        3.0
Don't Play  2.0

Decision: Play
```

## WEKA – Machine Learning and Data Mining Workbench



Waikato Environment for
Knowledge Analysis

(c) 1999 - 2003
University of Waikato
New Zealand

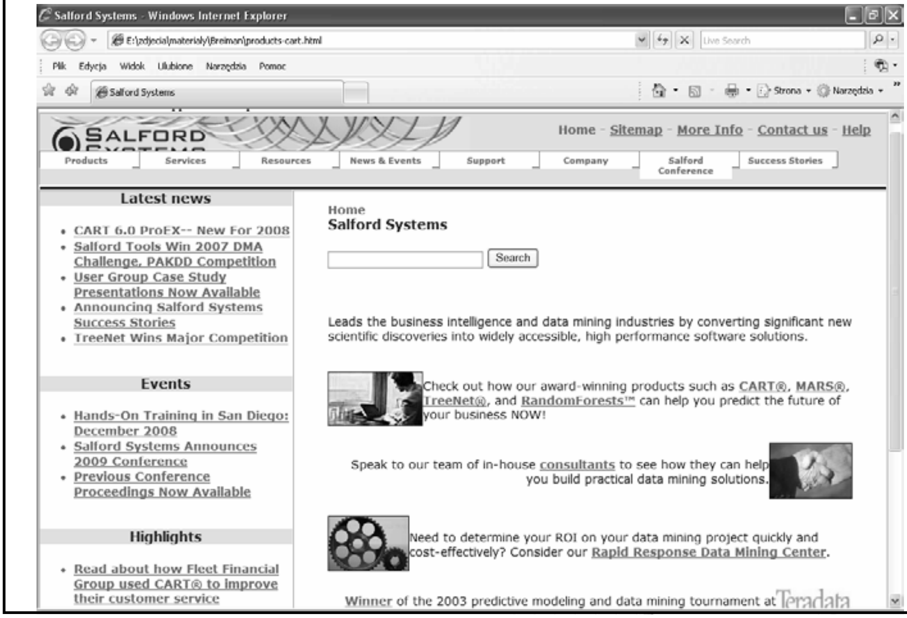
J4.8 – Java implementation of C4.8

Many more decision-tree and other machine learning methods

# *MLC++:* Machine Learning Library

- *MLC++* (Ron Kohavi idea)
  - http://www.sgi.com/Technology/mlc
  - An object-oriented machine learning library
  - Contains a suite of inductive learning algorithms (including *ID3*)
  - Supports incorporation, reuse of other DT algorithms (*C4.5*, etc.)
  - Automation of statistical evaluation, cross-validation
- Wrappers
  - Optimization loops that iterate over inductive learning functions (inducers)
  - Used for performance tuning (finding subset of *relevant* attributes, etc.)
- Combiners
  - Optimization loops that iterate over or interleave inductive learning functions
  - Used for performance tuning (finding subset of *relevant* attributes, etc.)
  - Examples: bagging, boosting of *ID3*, *C4.5*
- Graphical Display of Structures
  - Visualization of DTs (AT&T *dotty*, SGI *MineSet TreeViz*)
  - General logic diagrams (projection visualization)

# CART from Salford Systems



**Latest news**

- CART 6.0 ProEX-- New For 2008
- Salford Tools Win 2007 DMA Challenge, PAKDD Competition
- User Group Case Study Presentations Now Available
- Announcing Salford Systems Success Stories
- TreeNet Wins Major Competition

**Events**

- Hands-On Training in San Diego: December 2008
- Salford Systems Announces 2009 Conference
- Previous Conference Proceedings Now Available

**Highlights**

- Read about how Fleet Financial Group used CART® to improve their customer service

Home

**Salford Systems**

Leads the business intelligence and data mining industries by converting significant new scientific discoveries into widely accessible, high performance software solutions.

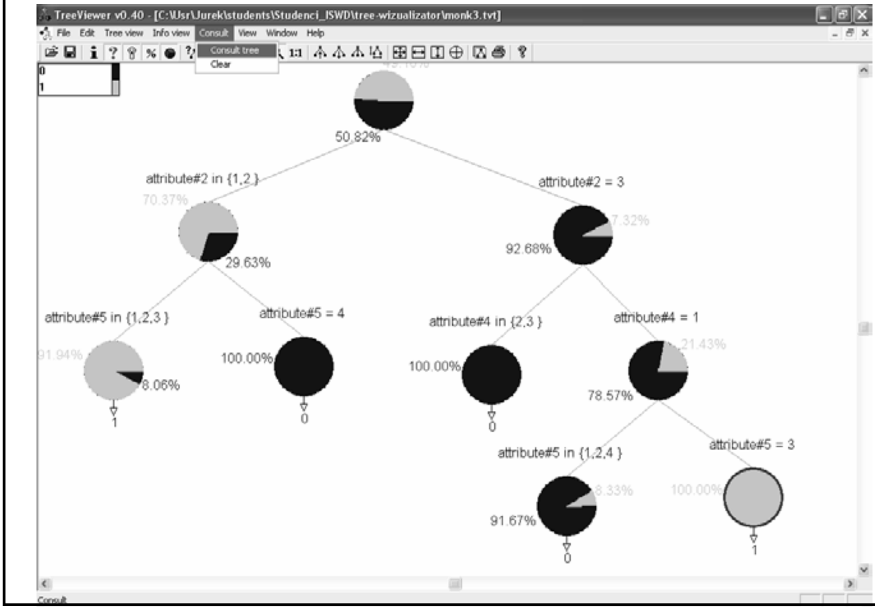
Check out how our award-winning products such as CART®, MARS®, TreeNet®, and RandomForests™ can help you predict the future of your business NOW!

Speak to our team of in-house consultants to see how they can help you build practical data mining solutions.

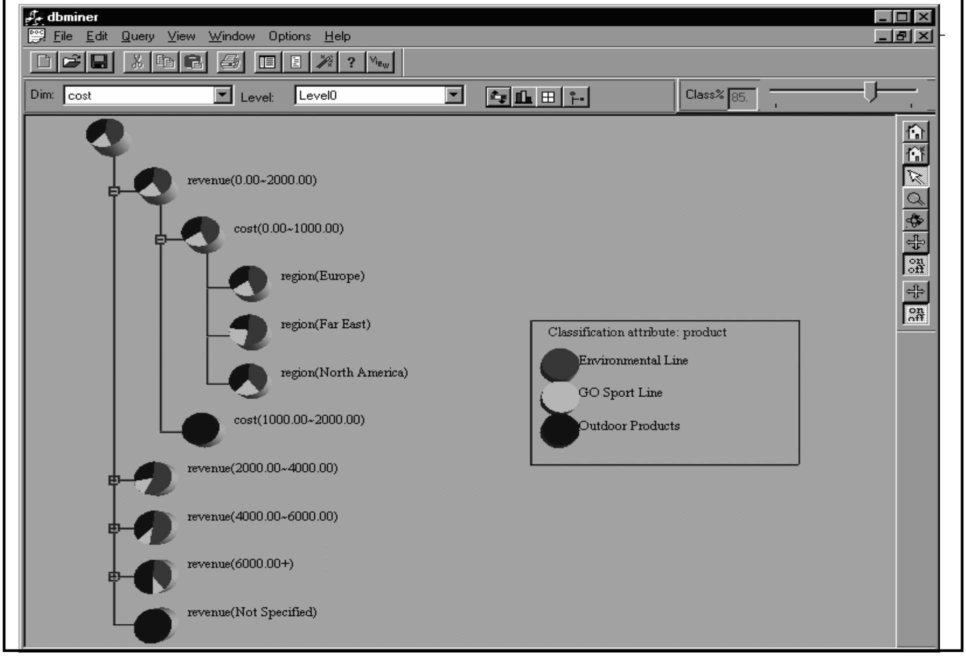
Need to determine your ROI on your data mining project quickly and cost-effectively? Consider our Rapid Response Data Mining Center.

Winner of the 2003 predictive modeling and data mining tournament at Teradata

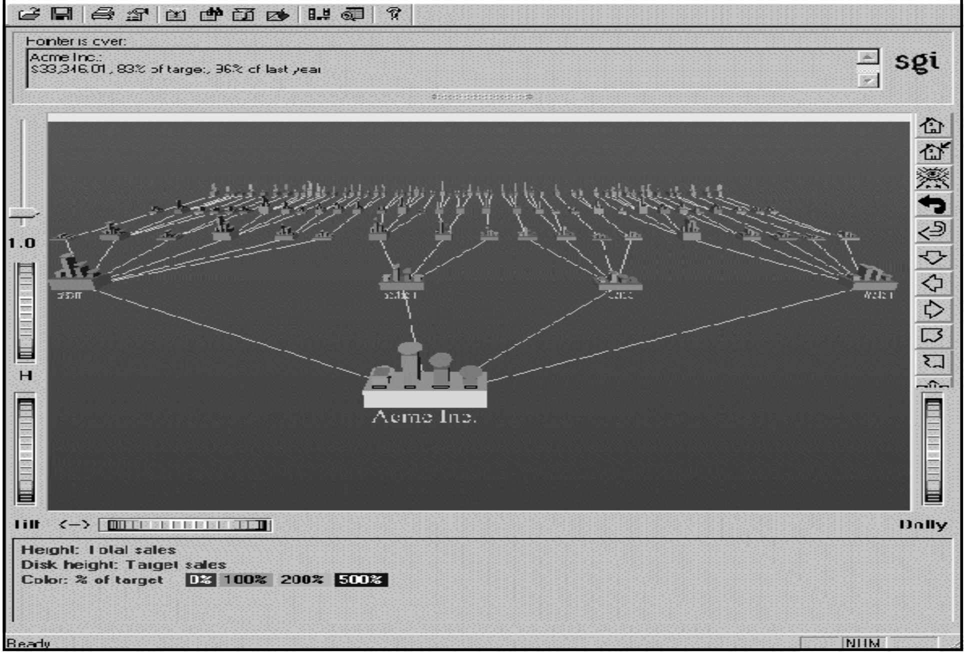
## Other tools for visualisation



## Presentation of Classification Results



## Visualization of a Decision Tree in SGI/MineSet 3.0



Pointer is over:

Acme Inc.:
$33,316.01 , 83% of target, 36% of last year

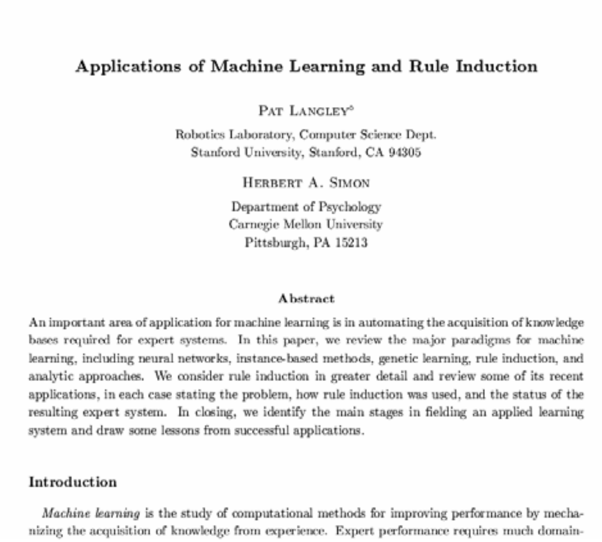
Height: Total sales
Disk height: Target sales
Color: % of target 0% 100% 200% 500%

Ready

## Applications

- Treatment effectiveness
- Credit Approval
- Store location
- Target marketing
- Insurance company (fraud detection)
- Telecommunication company (client classification)
- Many others …

## More about applications - see



### Applications of Machine Learning and Rule Induction

PAT LANGLEY[◇]
Robotics Laboratory, Computer Science Dept.
Stanford University, Stanford, CA 94305

HERBERT A. SIMON
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

**Abstract**

An important area of application for machine learning is in automating the acquisition of knowledge bases required for expert systems. In this paper, we review the major paradigms for machine learning, including neural networks, instance-based methods, genetic learning, rule induction, and analytic approaches. We consider rule induction in greater detail and review some of its recent applications, in each case stating the problem, how rule induction was used, and the status of the resulting expert system. In closing, we identify the main stages in fielding an applied learning system and draw some lessons from successful applications.

**Introduction**

*Machine learning* is the study of computational methods for improving performance by mechanizing the acquisition of knowledge from experience. Expert performance requires much domain-

- P.Langley, H.Simon paper in Michalski, Bratko, Kubat book on Machine Learning and Data Mining

## When to use decision trees

- One needs both symbolic representation and good classification performance.
- Problem does not depend on many attributes
  - Modest subset of attributes contains relevant info
- Linear combinations of features not critical.
- Speed of learning is important.

## Summary Points

1. Decision tree learning provides a practical method for classification learning.
2. ID3-like algorithms offer symbolic knowledge representation and good classifier performance.
3. The inductive bias of decision trees is preference (search) bias.
4. Overfitting the training data is an important issue in decision tree learning.
5. A large number of extensions of the decision tree algorithm have been proposed for overfitting avoidance, handling missing attributes, handling numerical attributes, etc.
6. There exists generalizations for mining massive data sets

## References

- Mitchell, Tom. M. 1997. *Machine Learning.* New York: McGraw-Hill
- Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*
- Stuart Russell, Peter Norvig, 1995. *Artificial Intelligence: A Modern Approach*. New Jersey: Prantice Hall.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984.
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Diciplinary Survey, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- S. M. Weiss and C. A. Kulikowski. Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufman, 1991.

## Any questions, remarks?

?