Decision Trees

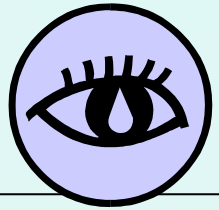
JERZY STEFANOWSKI
Institute of Computing Science
Poznań University of Technology

Doctoral School , Catania-Troina, April, 2008

Aims of this module

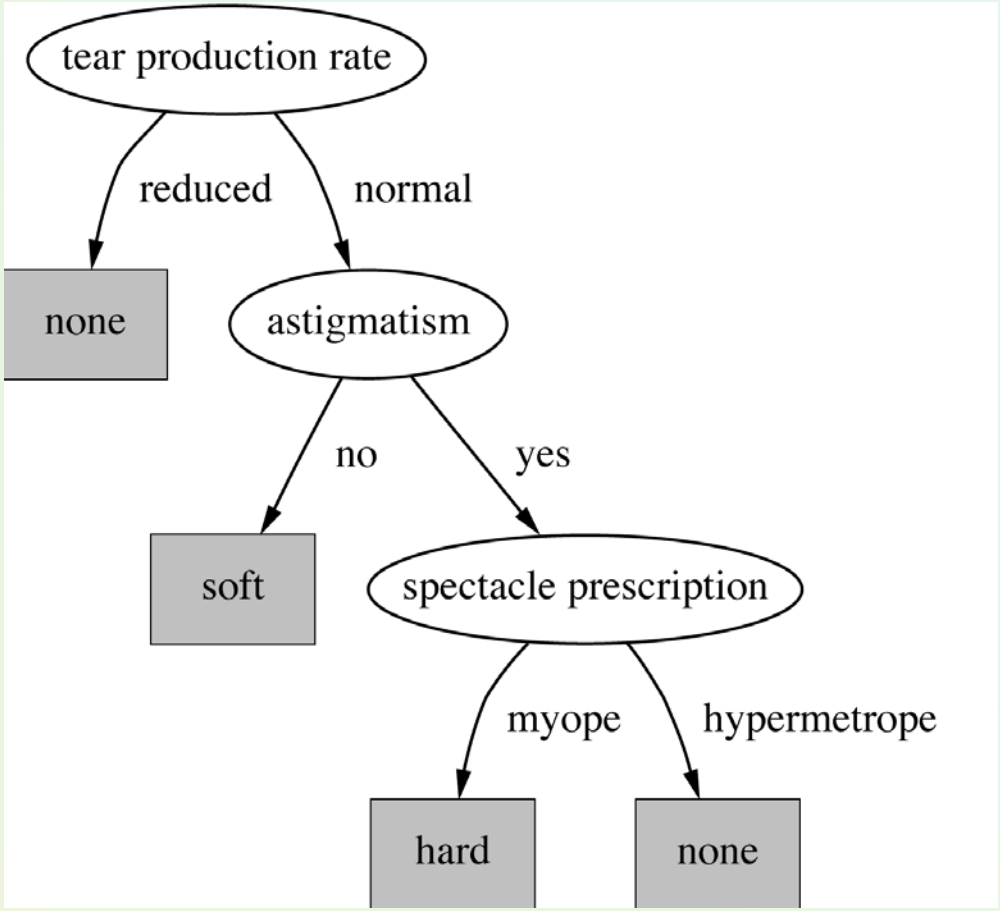
- The decision tree representation.
- The basic algorithm for inducing trees.
- Heuristic search (which is the best attribute):
  - Impurity measures, entropy, …
- Handling real / imperfect data.
- Overfitting and pruning decision trees.
- Some examples.

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

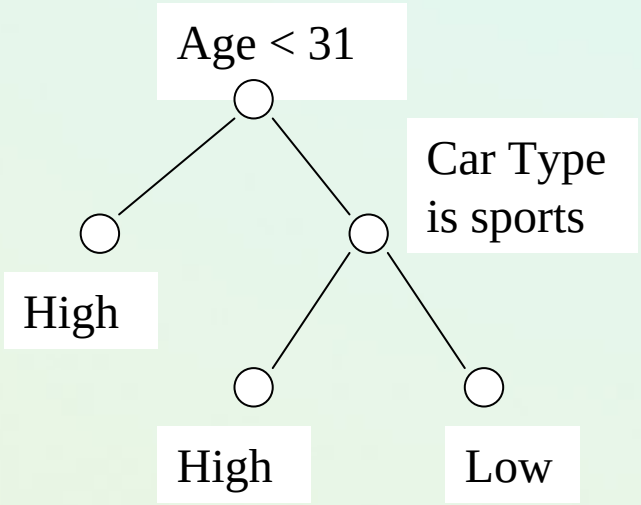
Induction of decision trees

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



Appropriate problems for decision trees learning

- Classification problems: assign an object into one of a discrete set of possible categories / classes.
- Characteristics:
  - Instances describable by attribute--value pairs (discrete or real-valued attributes).
  - Target function is discrete valued (boolean or multi-valued; if real valued, then regression trees).
  - Disjunctive hypothesis my be required.
  - Training data may be noisy (classification errors and/or mistakes in attribute values).
  - Training data may contain missing attribute values.

General issues

- Basic algorithm: a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner.
  - TDIDT $\rightarrow$ 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

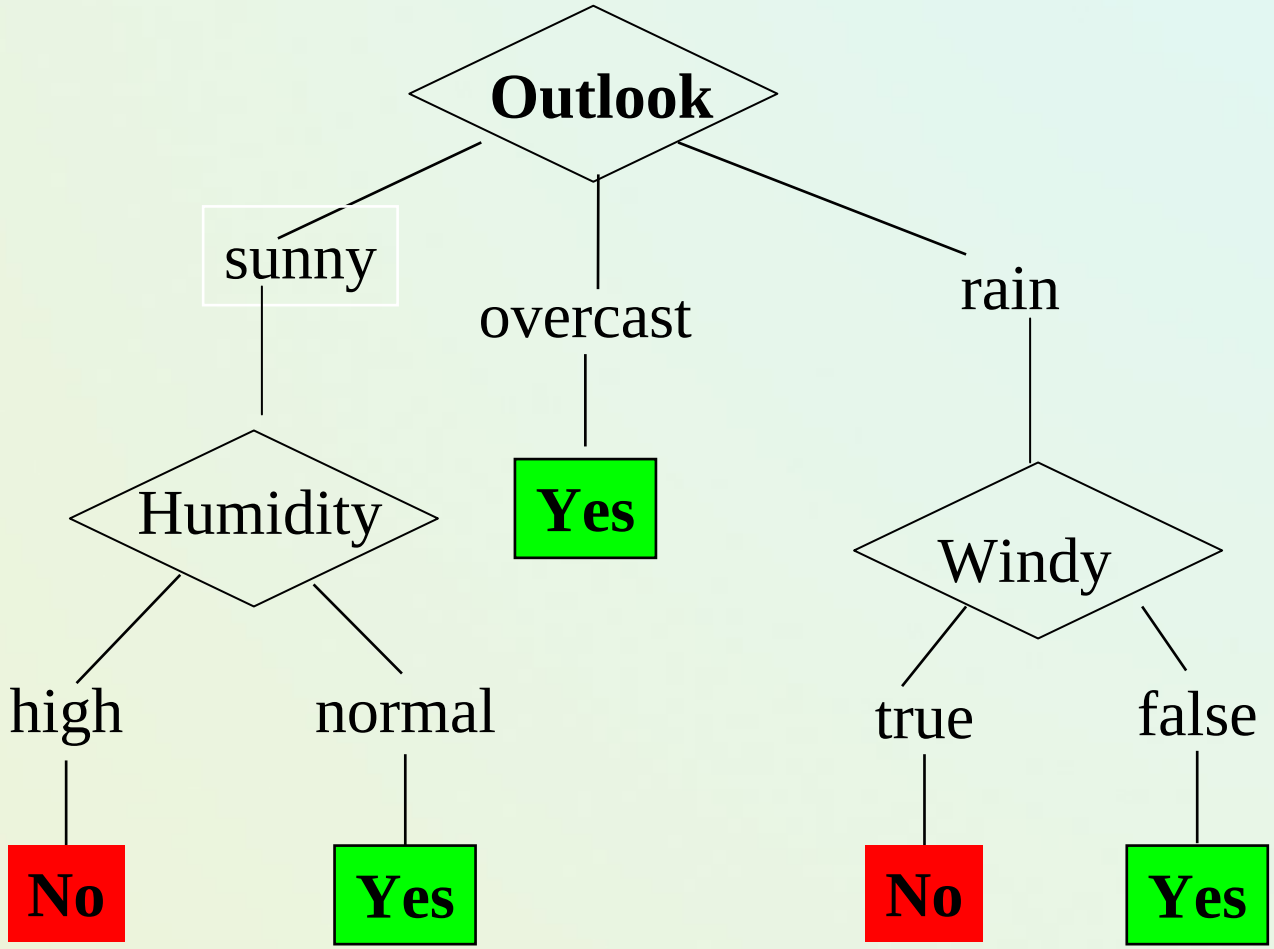
- All possible sequences of all possible tests
- very large search space, e.g., if N binary attributes:
  - 1 null tree
  - N trees with 1 (root) test
  - N*(N-1) trees with 2 tests
  - N*(N-1)*(N-1) trees with 3 tests
  - and so on
- size of search space is exponential in number of attributes
  - too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

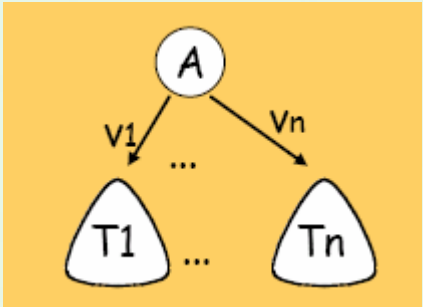
Note:
All attributes are nominal

Example Tree for “Play?”



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.

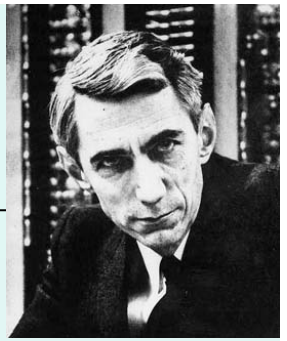
- The attribute that is most useful for classifying examples.
- We need a goodness / (im)purity function $\rightarrow$ measuring how well a given attribute separates the learning examples according to their classification.
- Heuristic: prefer the attribute that produces the “purest” sub-nodes and leads 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 Φ should satisfies:
  - $\Phi(P) \geq 0$; $\Phi(P)$ is minimal if $\exists i$ such that pi=1;
    $\Phi(P)$ is maximal if $\forall i\ 1 \leq i \leq k$, 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*:

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

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

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

- Choose the attribute with the maximal info gain:

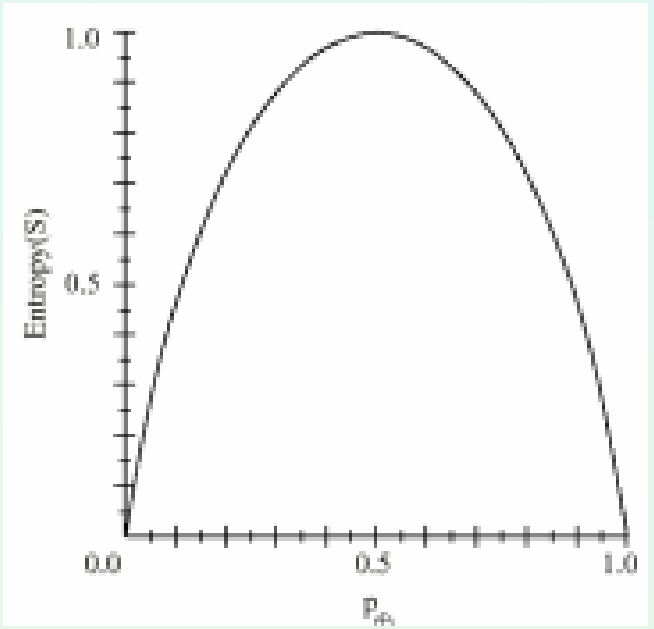
$$entropy(S) - entropy(S \mid A)$$

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*

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

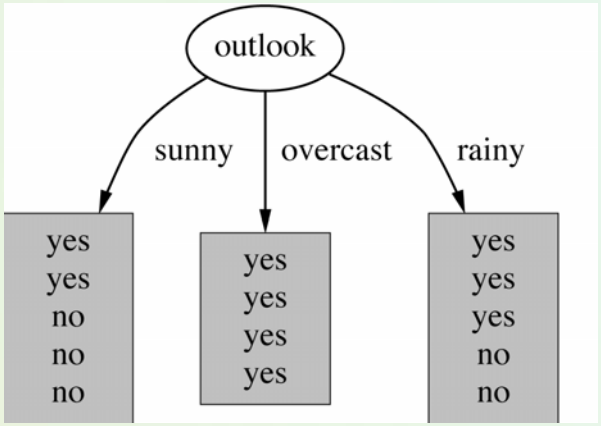
$$p_{no} = -\left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) = 0.53$$

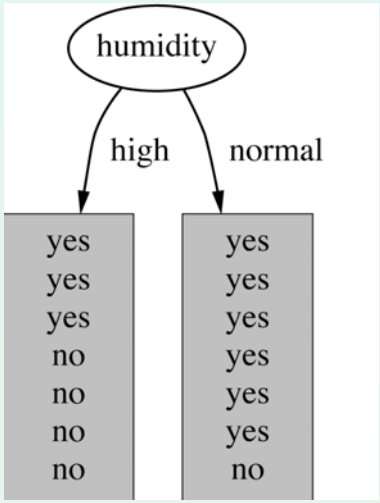
$$E(S) = p_{yes} + p_{no} = 0.94$$

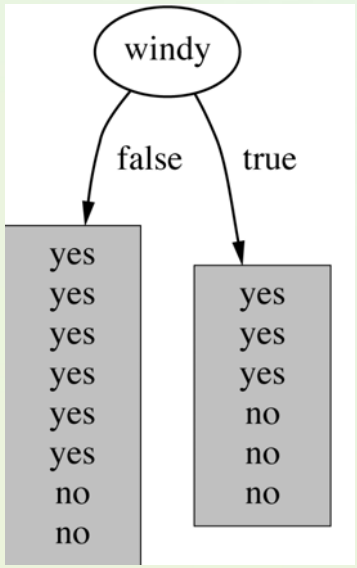
Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes

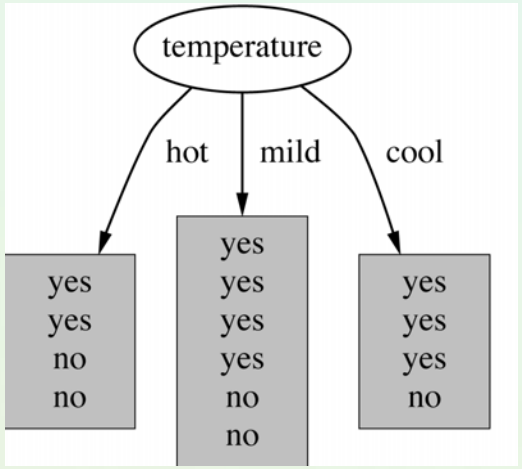
Outlook	Temp.	Humidity	Windy	play
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

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)

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

- Information gain for attributes from weather data:

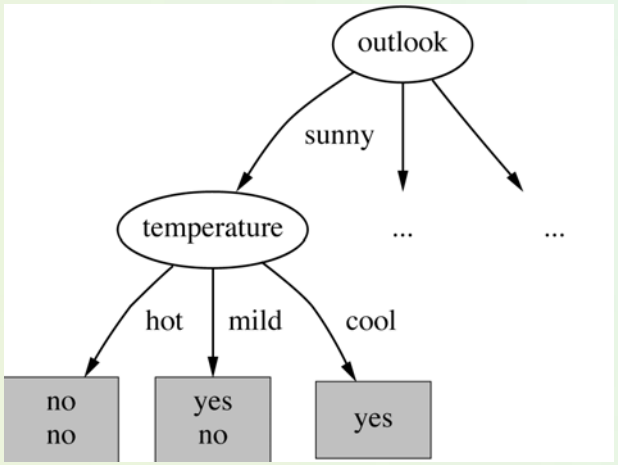
$$\text{gain("Outlook")} = 0.247$$

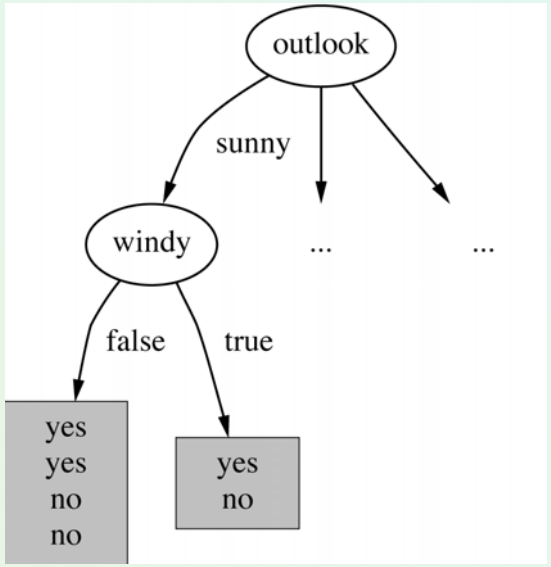
$$\text{gain("Temperature")} = 0.029$$

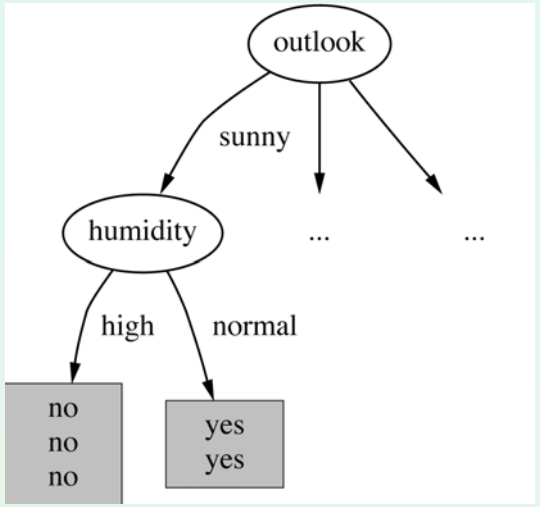
$$\text{gain("Humidity")} = 0.152$$

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

Continuing to split





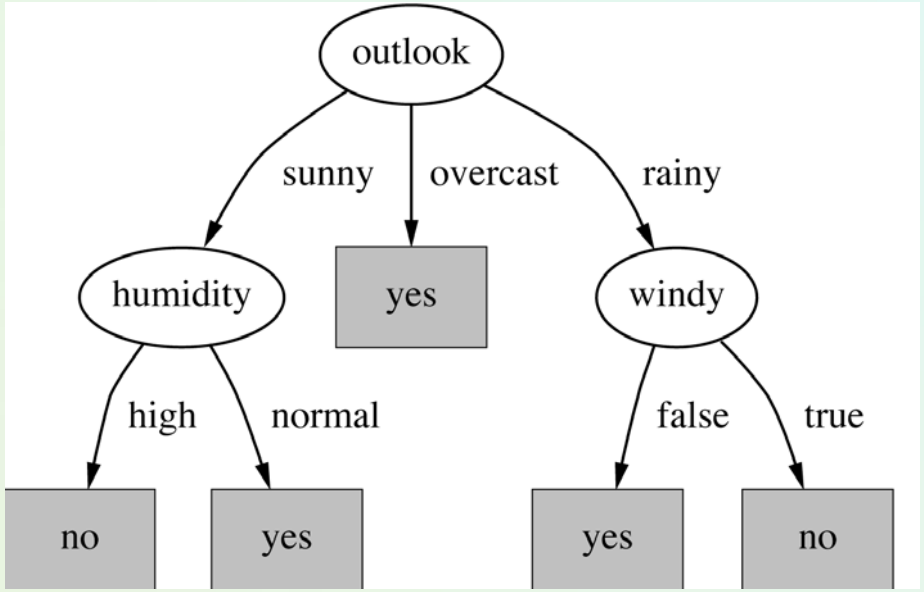


$\text{gain}(\text{"Humidity"}) = 0.971$

$\text{gain}(\text{"Temperature"}) = 0.571$

$\text{gain}(\text{"Windy"}) = 0.020$

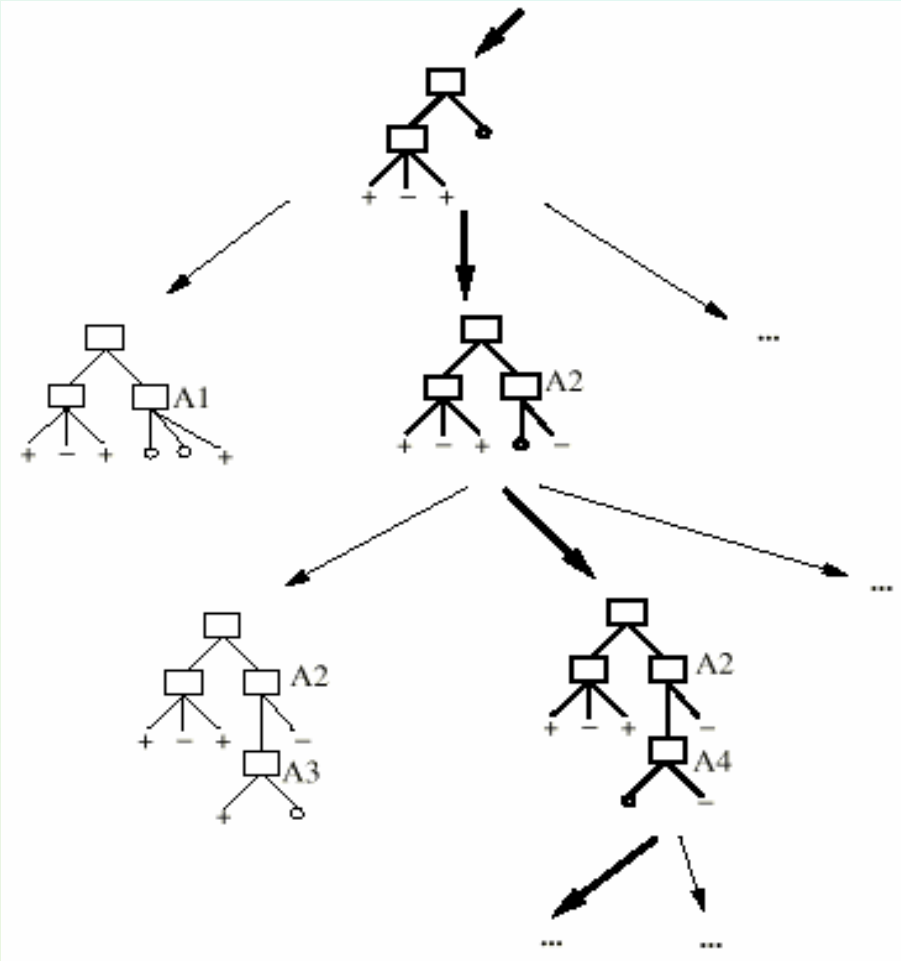
The final decision tree



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

Hypothesis Space Search in ID3

- ID3 performs a simple-to-complex, hill climbing search through this space.
- ID3 performs no backtracking in its search.
- ID3 uses all training instances at each step of the search.
- Preference for short trees.
- Preference for trees with high information gain attributes near the root.



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

- 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 (*need to enumerate all possible splitting points for each attribute*).

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 $\rightarrow$ Why simple trees should be preferred?

  1. The number of simple hypothesis that may accidentally fit the data is small, so chances that simple hypothesis uncover some interesting knowledge about the data are larger.
  2. Simpler trees have higher bias and thus lower variance, they should not overfit the data that easily.
  3. 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!

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, 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

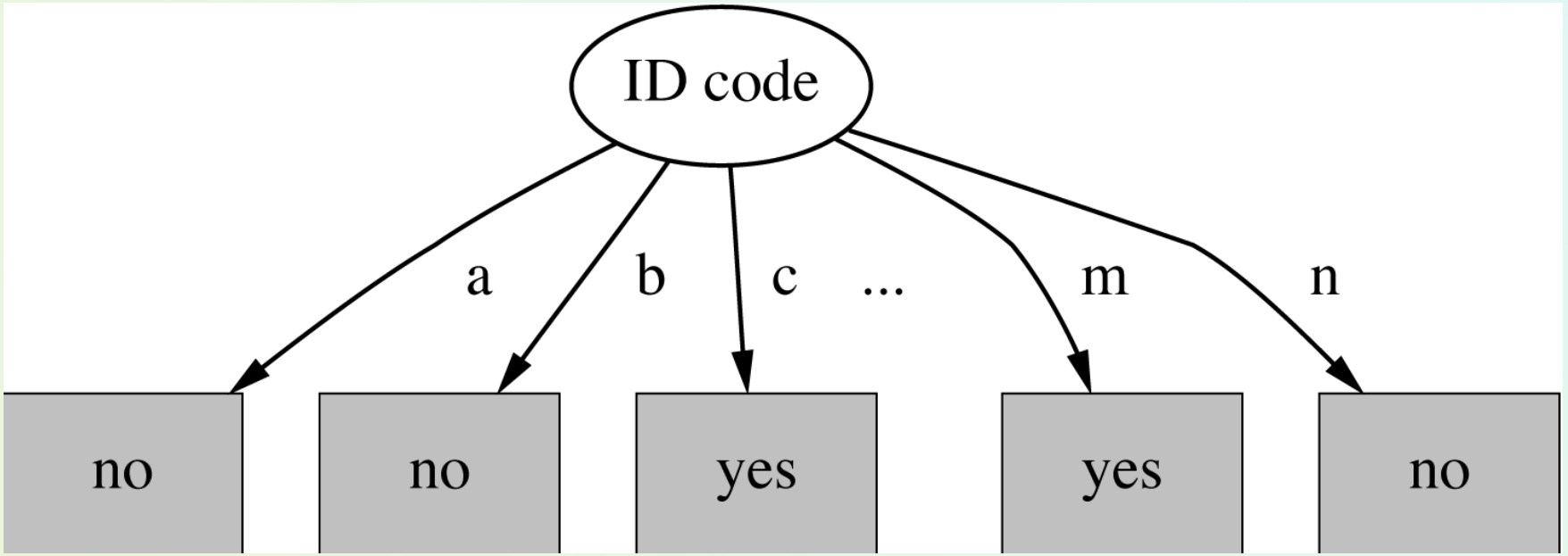
  ⇒Information gain is biased towards choosing attributes with a large number of values

  ⇒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: 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)$\in$X where X$\subset$domain(A)

Binary tree (Quinlan's C4.5 output)

```
Pruned decision tree:

A9 = t:
|   A15 > 228 : + (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 $\rightarrow$ 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

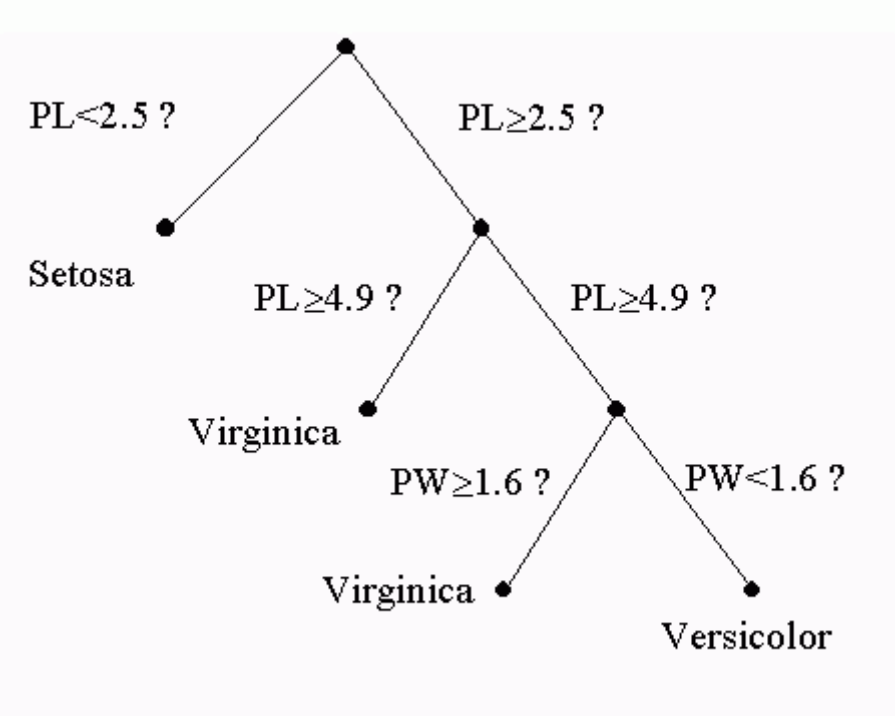
  - Info([4,2],[5,3])
    = 6/14 info([4,2]) + 8/14 info([5,3])
    = 0.939

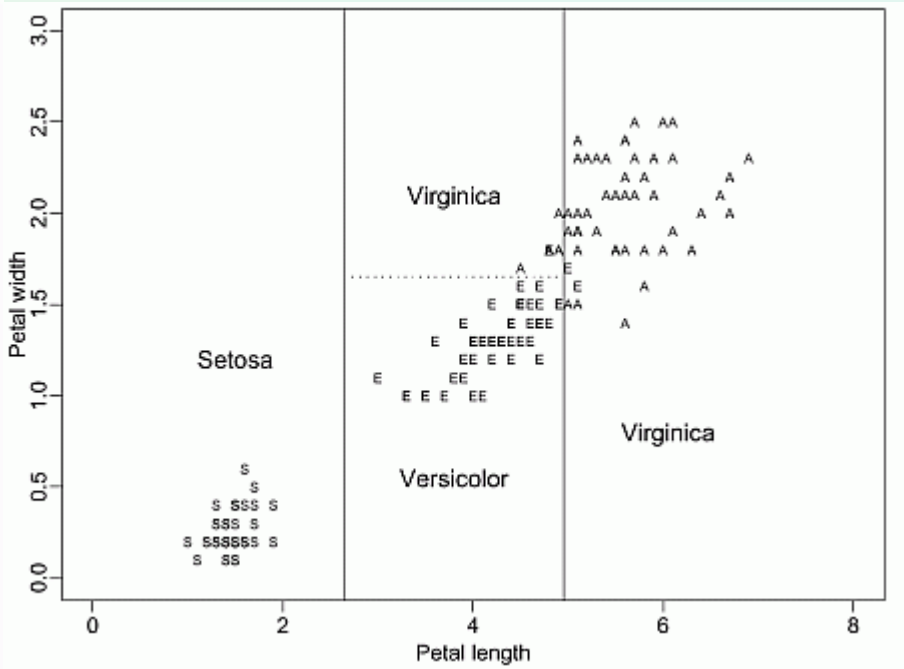
- Place split points halfway between values
- Can evaluate all split points in one pass!

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*.
  - More advanced approaches ….

Handling noise and imperfect examples

Sources of imperfection.

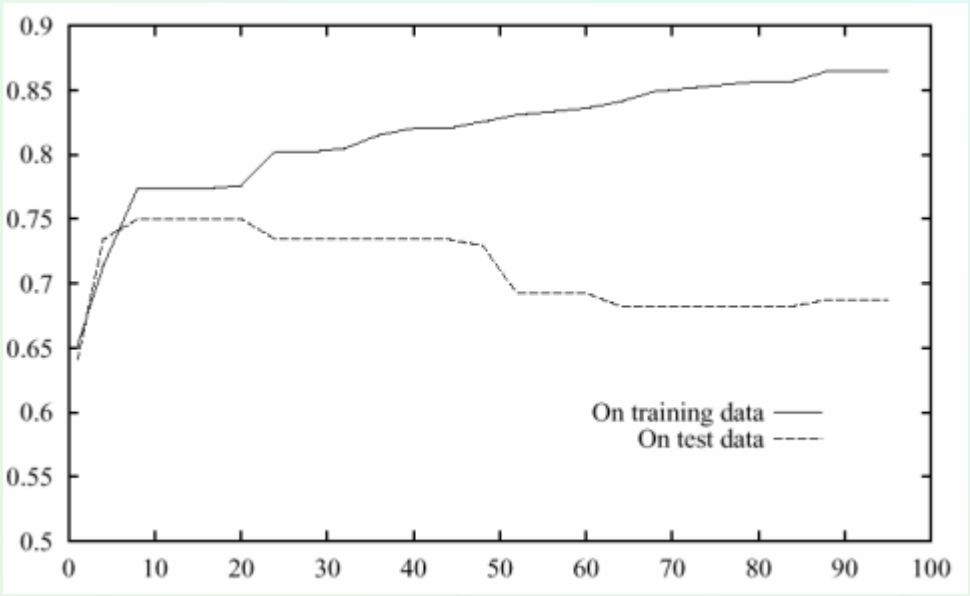
- Random errors (noise) in training examples
  - erroneous attribute values.
  - erroneous classification.
- Too sparse training examples.
- Inappropriate / insufficient set of attributes.
- Missing attribute values.

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, …
- 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).

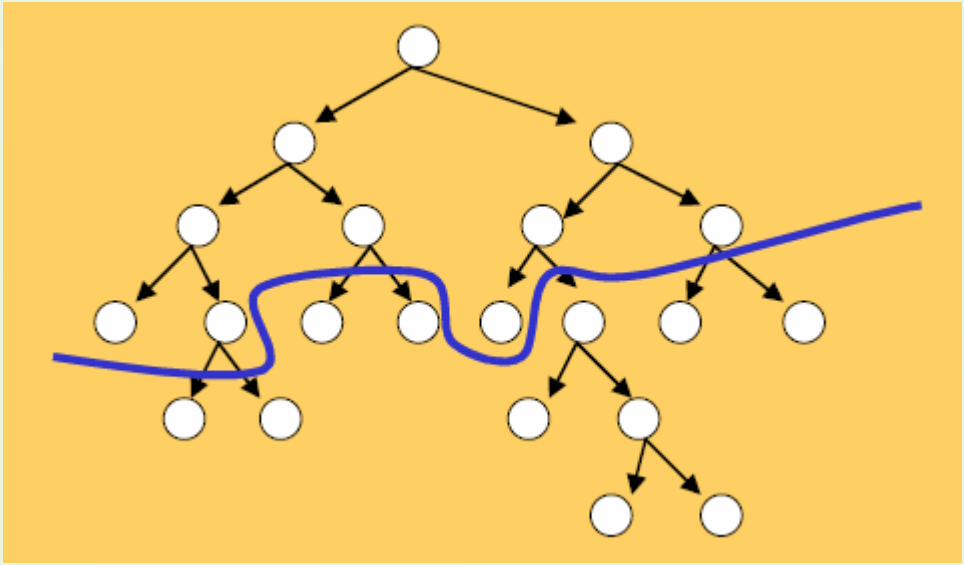
Overfitting in Decision Tree Construction

- 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 criterion is not greater than a certain threshold.
- The change of probability distribution is not significant.
- …

Reduced Error pruning

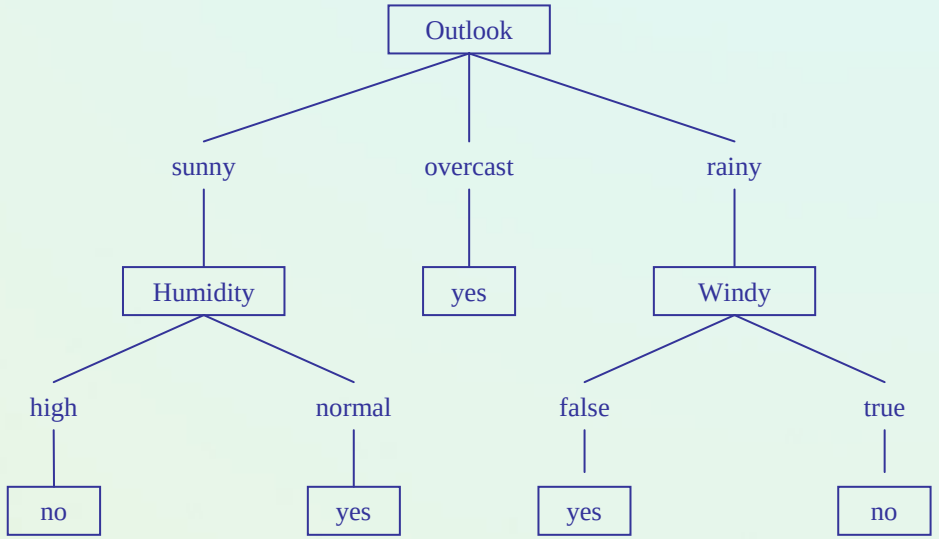
Split data into training and validation sets.

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*.

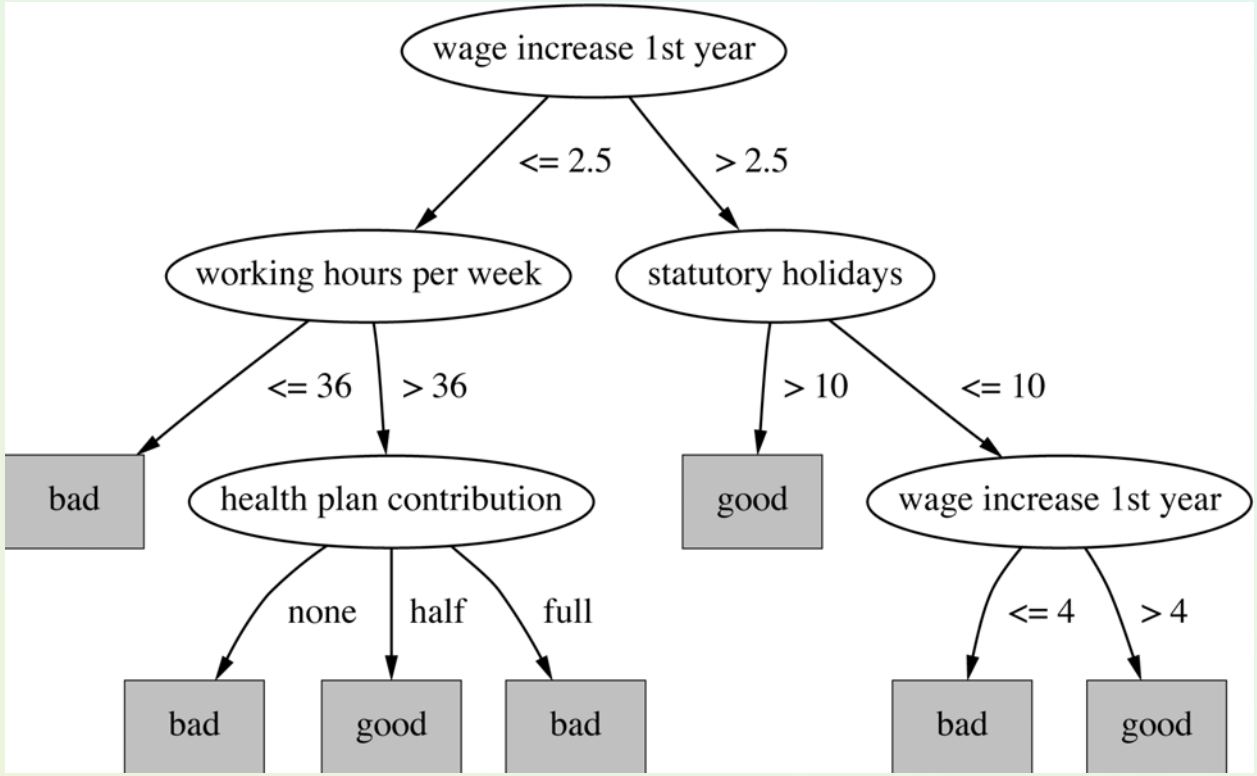
Do until further pruning is harmful:

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.



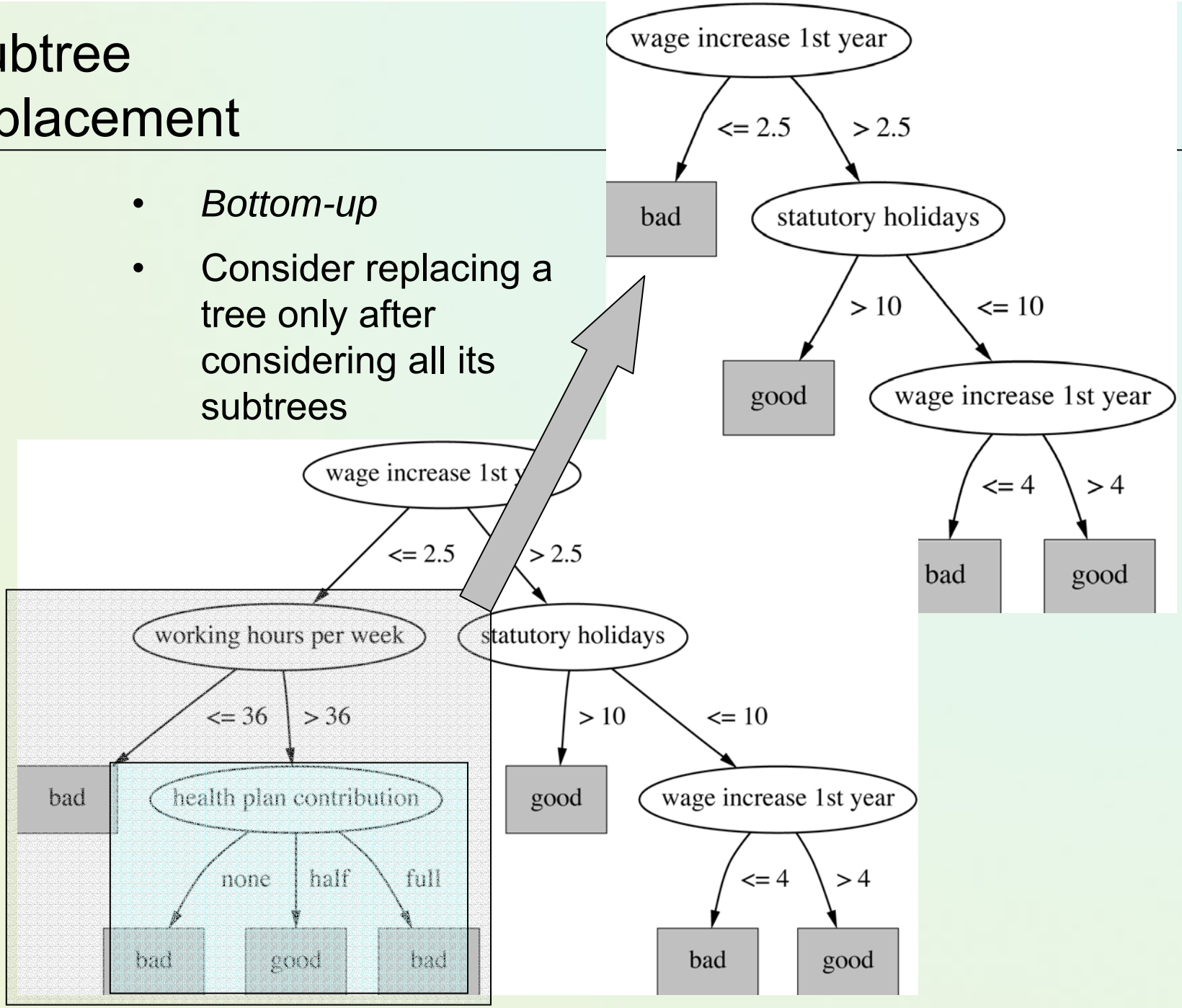
Post-pruning

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



Subtree replacement

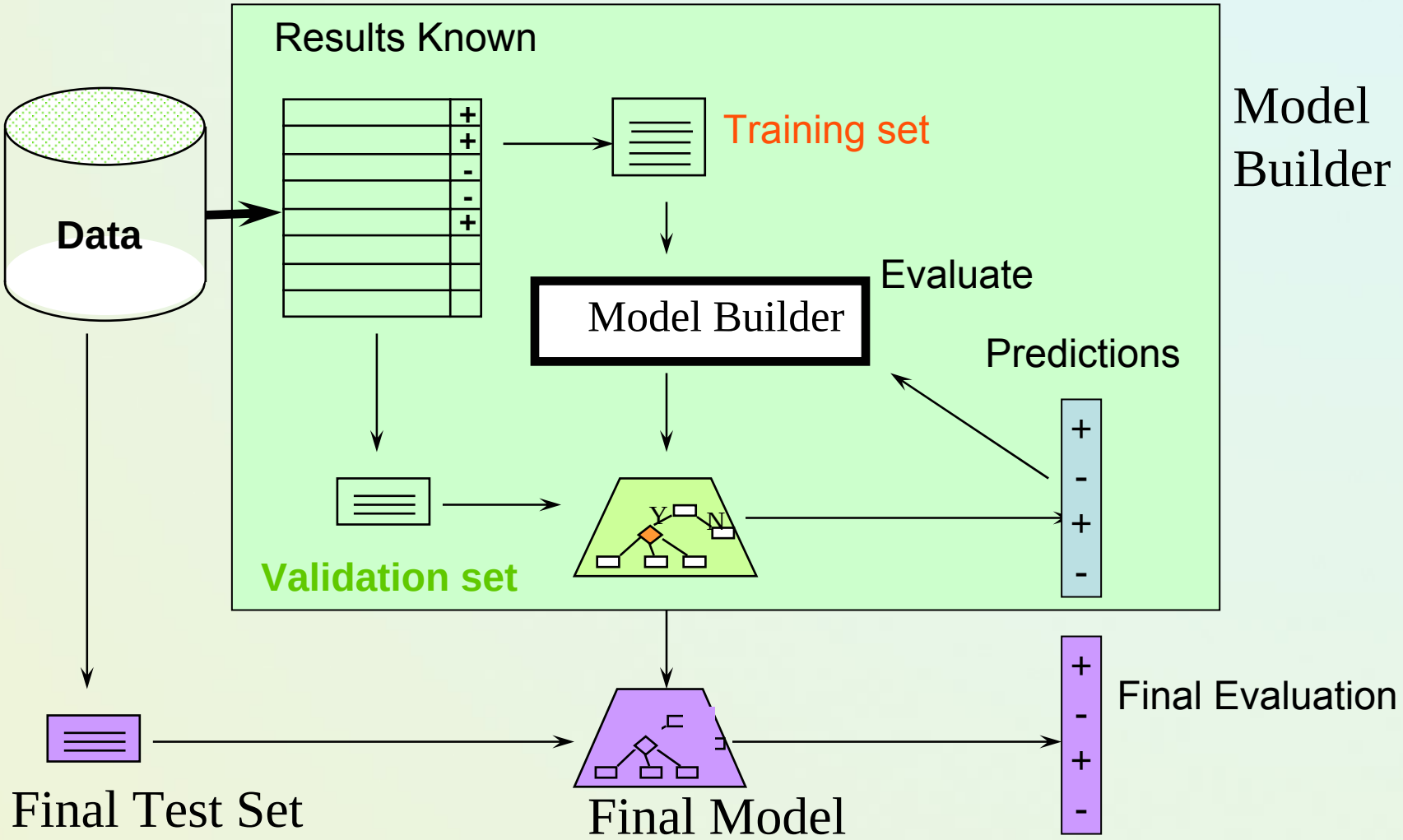
- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees



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
  - Option *c* parameter – default value 0,25:
    the smaller value, the stronger pruning!

Classification: Train, Validation, Test split



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
- recent data mining research contributes to
  - Scalability
  - Generalization-based classification
  - Parallel and distributed processing

Classifying Large Dataset

- Decision trees seem to be a good choice
  - relatively faster learning speed than other classification methods
  - can be converted into simple and easy to understand classification rules
  - can be used to generate SQL queries for accessing databases
  - has comparable classification accuracy with other methods
  - Objectives
- Classifying data-sets with millions of examples and a few hundred even thousands attributes with reasonable speed.

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)

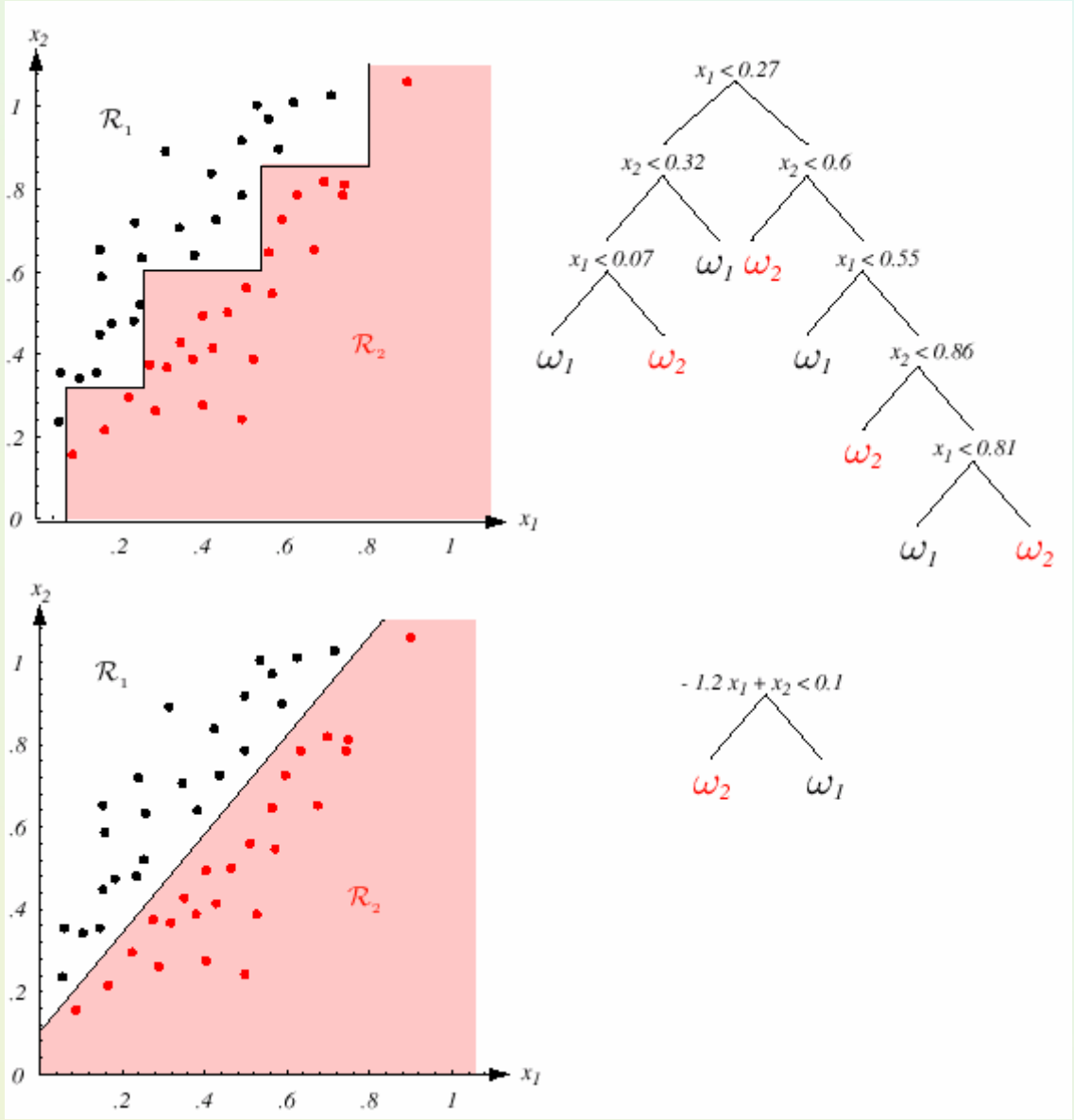
Previous Efforts on Scalability

- 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.

Weaknesses of Decision Trees

- Large or complex trees can be just as unreadable as other models
- Trees don't easily represent some basic concepts such as M-of-N, parity, non-axis-aligned classes…
- Don't handle real-valued parameters as well as Booleans
- If model depends on summing contribution of many different attributes, DTs probably won't do well
- DTs that look very different can be same/similar
- Propositional (as opposed to 1st order logic)
- Recursive partitioning: run out of data fast as descend tree

Oblique trees



Univariate, or monothetic trees,

mult-variate, or oblique trees.

Figure from Duda, Hart & Stork, Chap. 8

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.

Applications

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

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.

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?