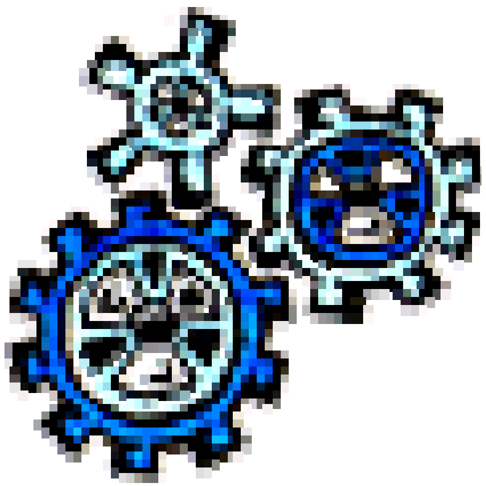
Knowledge discovery & data mining

Classification & fraud detection

Fosca Giannotti and
Dino Pedreschi
Pisa KDD Lab
CNUCE-CNR & Univ. Pisa
http://www-kdd.di.unipi.it/



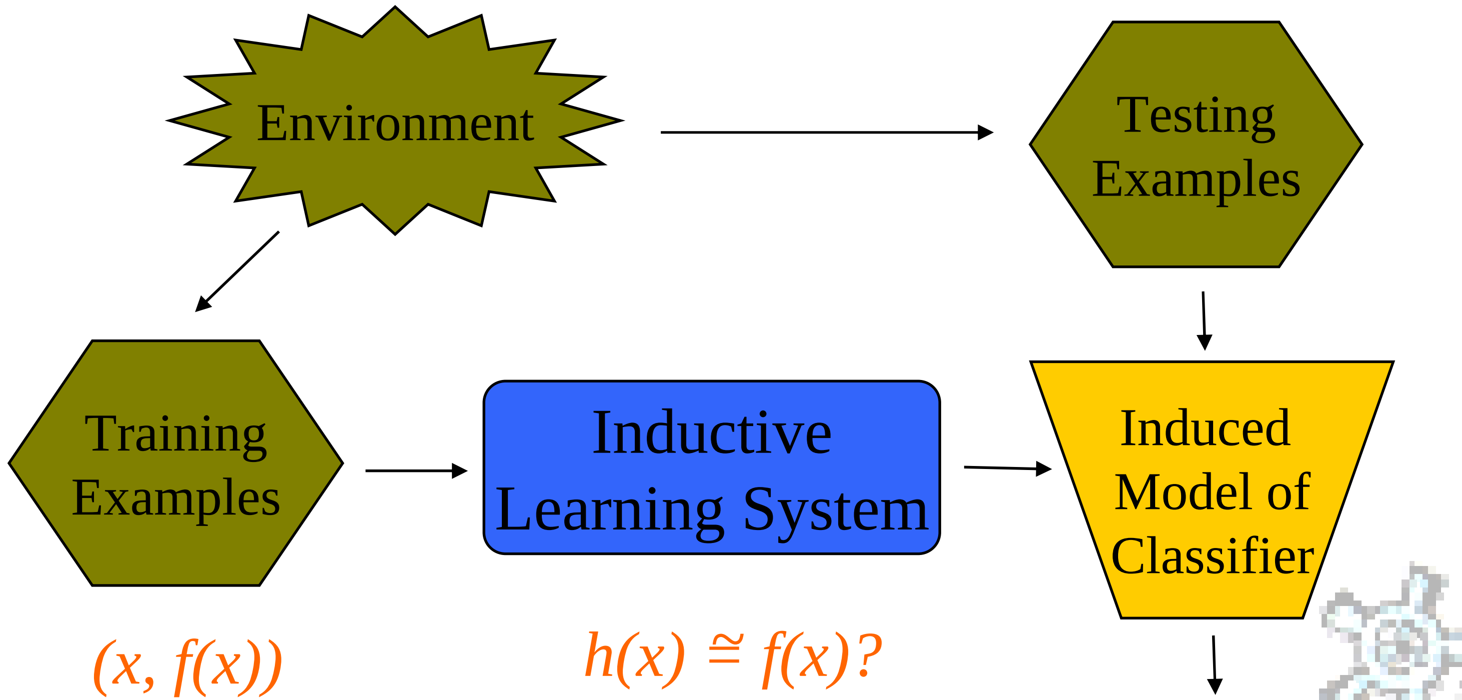
Module outline

- The classification task
- Main classification techniques
  - Bayesian classifiers
  - Decision trees
  - Hints to other methods
- Application to a case-study in fiscal fraud detection: audit planning

The classification task

- Input: a training set of tuples, each labelled with one class label
- Output: a model (classifier) which assigns a class label to each tuple based on the other attributes.
- The model can be used to predict the class of new tuples, for which the class label is missing or unknown
- Some natural applications
  - credit approval
  - medical diagnosis
  - treatment effectiveness analysis

Basic Framework for Inductive Learning



A problem of representation and search for the best hypothesis, h(x).

Output Classification

(x, h(x))

Train & test

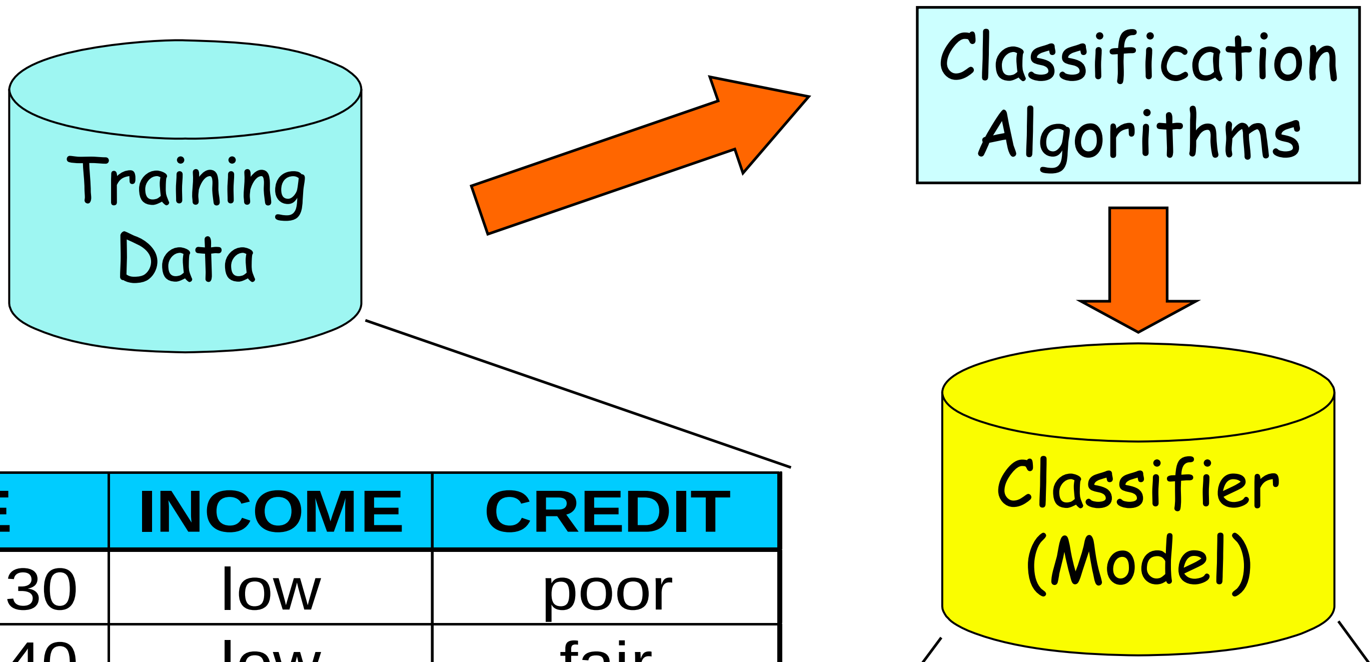
- The tuples (observations, samples) are partitioned in training set + test set.
- Classification is performed in two steps:

1. training - build the model from training set
2. test - check accuracy of the model using test set

Train & test

- Kind of models
  - IF-THEN rules
  - Other logical formulae
  - Decision trees
- Accuracy of models
  - The known class of test samples is matched against the class predicted by the model.
  - Accuracy rate = % of test set samples correctly classified by the model.

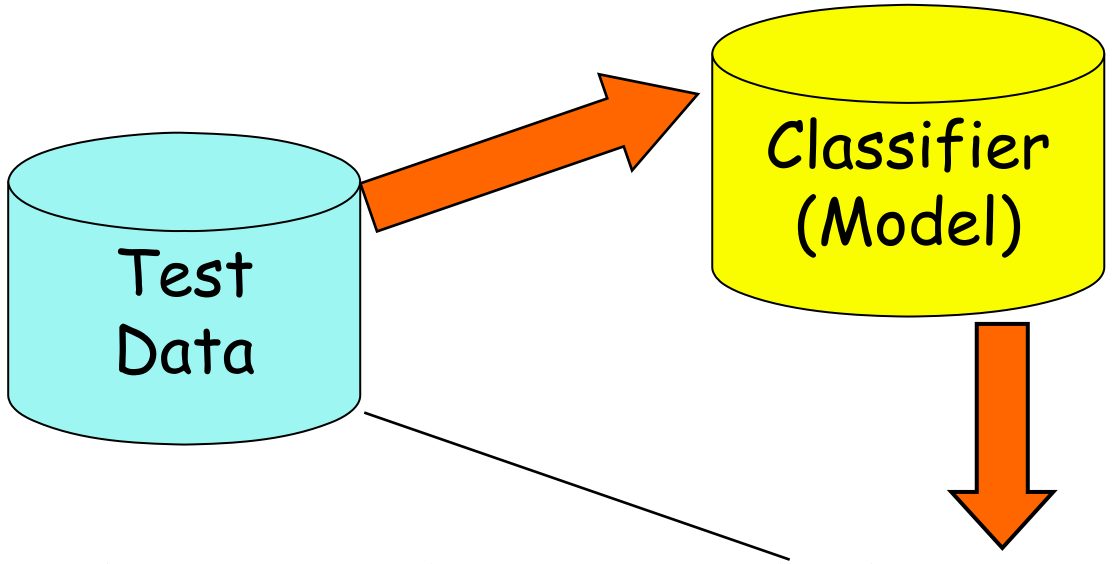
Training step



NAME	AGE	INCOME	CREDIT
Mary	20 - 30	low	poor
James	30 - 40	low	fair
Bill	30 - 40	high	good
John	20 - 30	med	fair
Marc	40 - 50	high	good
Annie	40 - 50	high	good

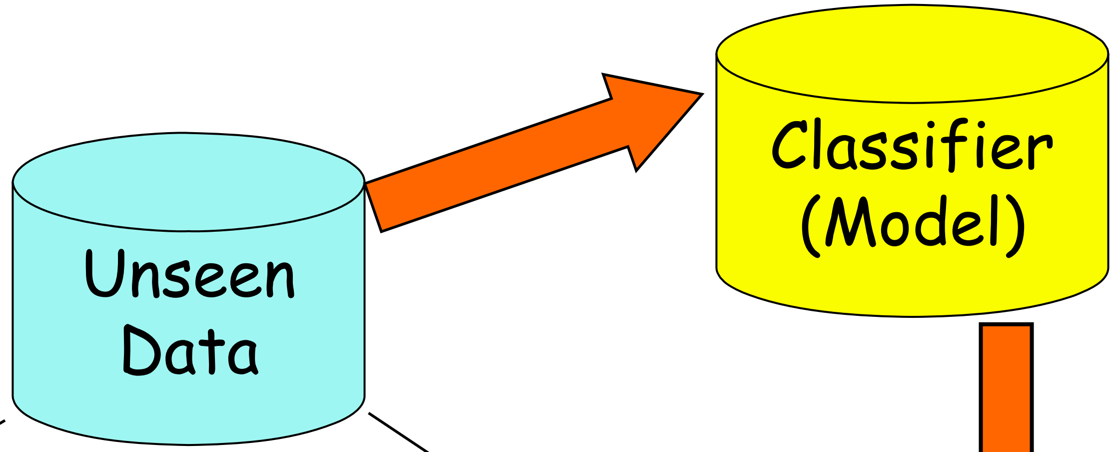
IF *age* = 30 - 40
OR *income* = high
THEN *credit* = good

Test step



NAME	AGE	INCOME	CREDIT
Paul	20 - 30	high	good
Jenny	40 - 50	low	fair
Rick	30 - 40	high	fair

CREDIT
fair
fair
good



NAME	AGE	INCOME
Doc	20 - 30	high
Phil	30 - 40	low
Kate	40 - 50	med

CREDIT
fair
poor
fair

Machine learning terminology

- **Classification = supervised learning**
  - use training samples with known classes to classify new data
- **Clustering = unsupervised learning**
  - training samples have no class information
  - guess classes or clusters in the data

Comparing classifiers

- Accuracy
- Speed
- Robustness
  - w.r.t. noise and missing values
- Scalability
  - efficiency in large databases
- Interpretability of the model
- Simplicity
  - decision tree size
  - rule compactness
- Domain-dependent quality indicators

Classical example: play tennis?

- Training set from Quinlan's book

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Module outline

- The classification task
- Main classification techniques
  - Bayesian classifiers
  - Decision trees
  - Hints to other methods
- Application to a case-study in fraud detection: planning of fiscal audits

Bayesian classification

- The classification problem may be formalized using a-posteriori probabilities:
- $P(C|X)$ = prob. that the sample tuple $X=<x_1,\ldots,x_k>$ is of class C.

- E.g. $P(class=N \mid outlook=sunny, windy=true, \ldots)$

- Idea: assign to sample X the class label C such that $P(C|X)$ is maximal

Estimating a-posteriori probabilities

- Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) \,/\, P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative freq of class C samples
- C such that $P(C|X)$ is maximum =
  C such that $P(X|C) \cdot P(C)$ is maximum

- Problem: computing $P(X|C)$ is unfeasible!

Naïve Bayesian Classification

- Naïve assumption: attribute independence

$$P(x_1,\dots,x_k|C) = P(x_1|C)\cdot\dots\cdot P(x_k|C)$$

- If i-th attribute is categorical:
  $P(x_i|C)$ is estimated as the relative freq of samples having value x_i as i-th attribute in class C
- If i-th attribute is continuous:
  $P(x_i|C)$ is estimated thru a Gaussian density function
- Computationally easy in both cases

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

Play-tennis example: classifying X

- An unseen sample X = <rain, hot, high, false>

- $P(X|p) \cdot P(p) =$
  $P(rain|p) \cdot P(hot|p) \cdot P(high|p) \cdot P(false|p) \cdot P(p) =$
  $3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$
- $P(X|n) \cdot P(n) =$
  $P(rain|n) \cdot P(hot|n) \cdot P(high|n) \cdot P(false|n) \cdot P(n) =$
  $2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = 0.018286$

- Sample X is classified in class n (don't play)

The independence hypothesis…

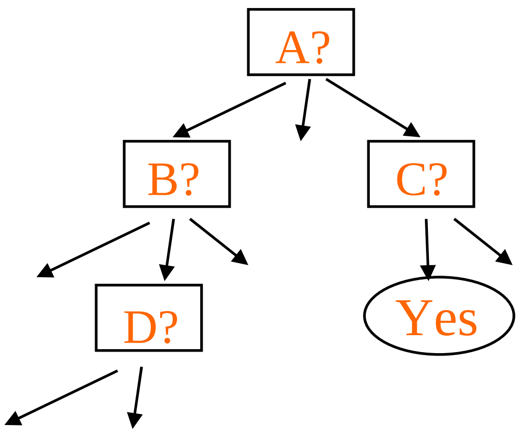
- … makes computation possible
- … yields optimal classifiers when satisfied
- … but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
  - Bayesian networks, that combine Bayesian reasoning with causal relationships between attributes
  - Decision trees, that reason on one attribute at the time, considering most important attributes first

Module outline

- The classification task
- Main classification techniques
  - Bayesian classifiers
  - Decision trees
  - Hints to other methods
- Application to a case-study in fraud detection: planning of fiscal audits

Decision trees

- A tree where
- internal node = test on a single attribute
- branch = an outcome of the test
- leaf node = class or class distribution

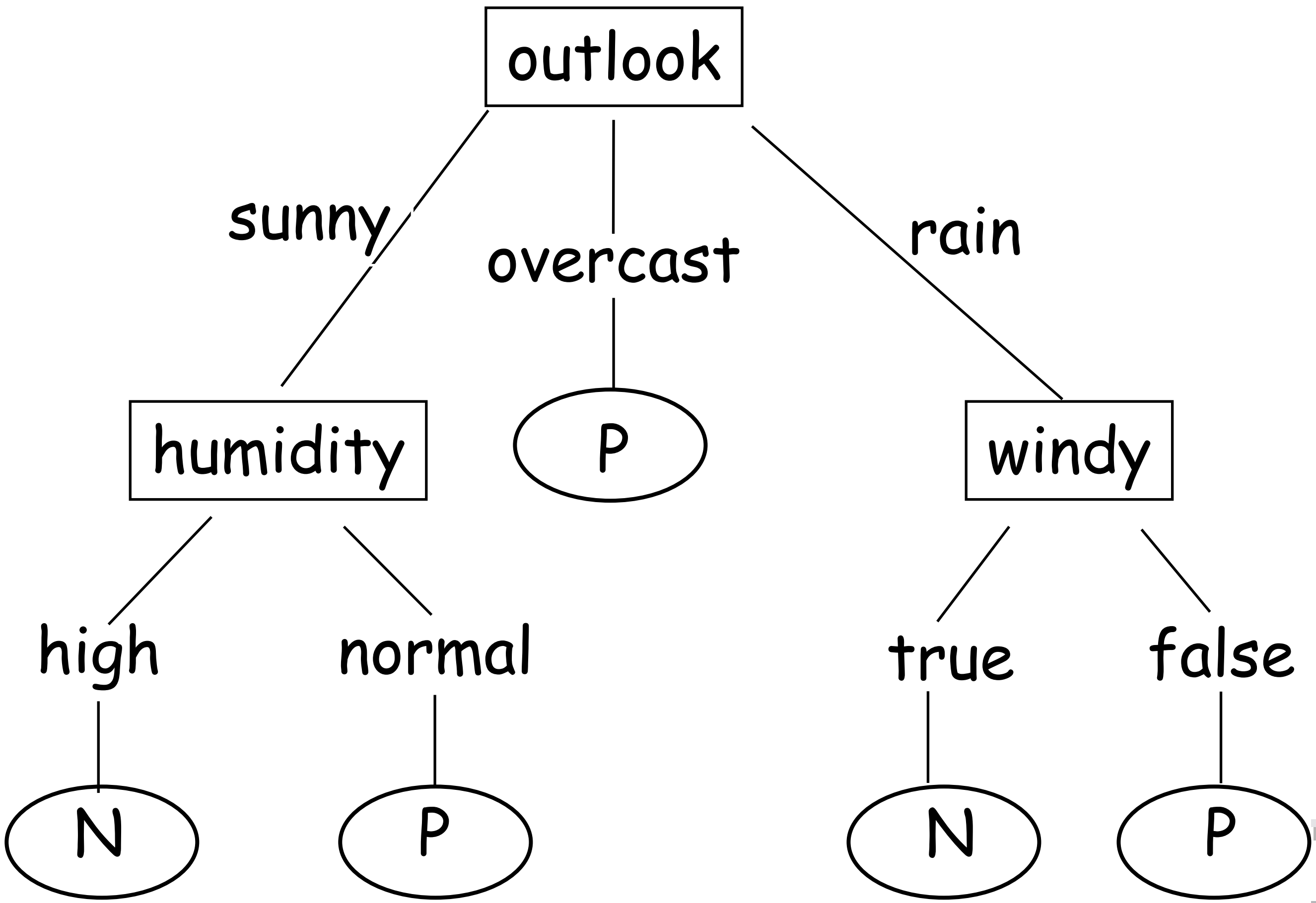


Classical example: play tennis?

- **Training set from Quinlan's book**

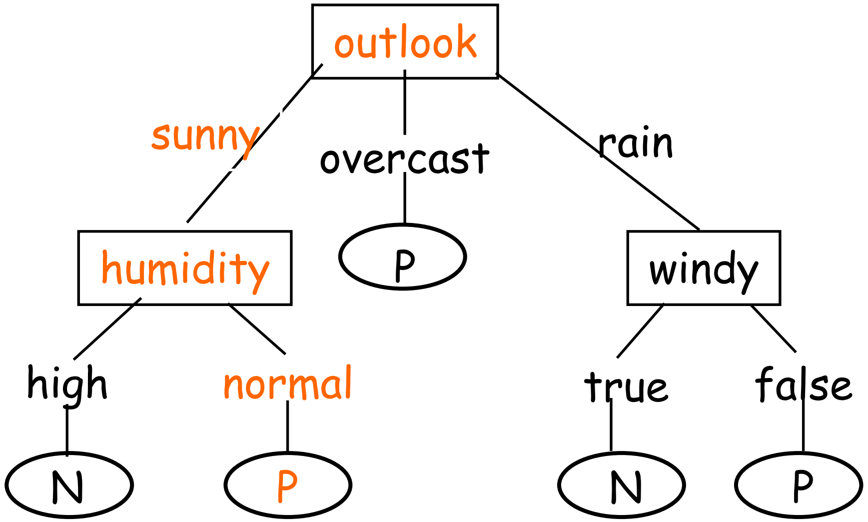
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Decision tree obtained with ID3 (Quinlan 86)



From decision trees to classification rules

- One rule is generated for each path in the tree from the root to a leaf
- Rules are generally simpler to understand than trees



IF outlook=sunny
AND humidity=normal
THEN play tennis

Decision tree induction

- Basic algorithm
  - top-down recursive
  - divide & conquer
  - greedy (may get trapped in local maxima)
- Many variants:
  - from machine learning: ID3 (Iterative Dichotomizer), C4.5 (Quinlan 86, 93)
  - from statistics: CART (Classification and Regression Trees) (Breiman et al 84)
  - from pattern recognition: CHAID (Chi-squared Automated Interaction Detection) (Magidson 94)
- Main difference: divide (split) criterion

Generate_DT(samples, attribute_list) =

1) Create a new node N;
2) If samples are all of class C then label N with C and exit;
3) If attribute_list is empty then label N with majority_class(N) and exit;
4) Select best_split from attribute_list;
5) For each value v of attribute best_split:
   - Let S_v = set of samples with best_split=v ;
   - Let N_v = Generate_DT(S_v, attribute_list \ best_split) ;
   - Create a branch from N to N_v labeled with the test best_split=v ;

Criteria for finding the best split

- Information gain (ID3 – C4.5)
  - Entropy, an information theoretic concept, measures impurity of a split
  - Select attribute that maximize entropy reduction
- Gini index (CART)
  - Another measure of impurity of a split
  - Select attribute that minimize impurity
- χ^2 contingency table statistic (CHAID)
  - Measures correlation between each attribute and the class label
  - Select attribute with maximal correlation

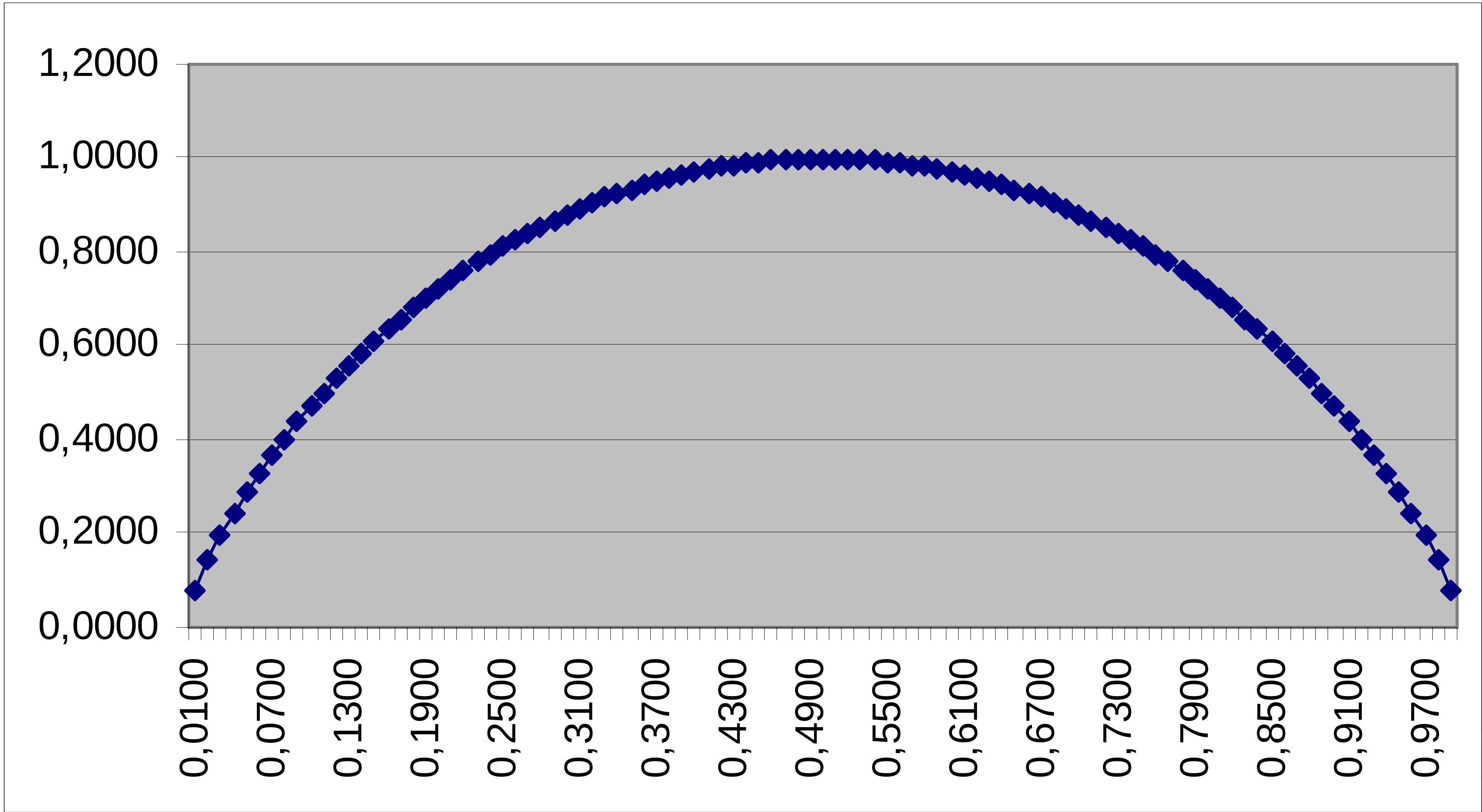
Information gain (ID3 - C4.5)

E.g., two classes, *Pos* and *Neg*, and dataset *S* with *p* *Pos*-elements and *n* *Neg*-elements.

- Amount of information to decide if an arbitrary example belongs to *Pos* or *Neg*:
- $fp = p \,/\, (p+n)$ $\qquad\qquad fn = n \,/\, (p+n)$

$$I(p,n) = -\ fp \cdot \log_2(fp) - fn \cdot \log_2(fn)$$

Entropy


1,2000
1,0000
0,8000
0,6000
0,4000
0,2000
0,0000
0,0100
0,0700
0,1300
0,1900
0,2500
0,3100
0,3700
0,4300
0,4900
0,5500
0,6100
0,6700
0,7300
0,7900
0,8500
0,9100
0,9700

Information gain (ID3 - C4.5)

- Entropy = information needed to classify samples in a split according to an attribute
- Splitting S with attribute A results in partition
  $\{S_1, S_2, \ldots, S_k\}$
- p_i (resp. n_i) = # elements in S_i from *Pos* (resp. *Neg*)

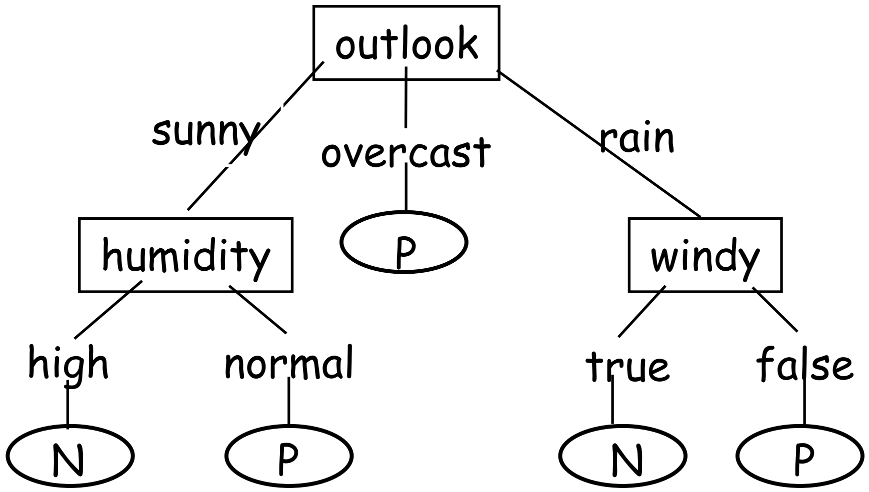
$$E(A) = \sum_{i \in [1,k]} I(p_i, n_i) \cdot (p_i + n_i) / (p+n)$$

$$gain(A) = I(p,n) - E(A)$$

- Select A which maximizes $gain(A)$
- Extensible to continuous attributes

Information gain - play tennis example

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



- **Choosing best split at root node:**
  - ***gain*(outlook) = 0.246**
  - ***gain*(temperature) = 0.029**
  - ***gain*(humidity) = 0.151**
  - ***gain*(windy) = 0.048**
- **Criterion biased towards attributes with many values – corrections proposed (*gain ratio*)**

Gini index (CART)

- E.g., two classes, *Pos* and *Neg*, and dataset *S* with *p* *Pos*-elements and *n* *Neg*-elements.
- $fp = p\ /\ (p+n)$ $\qquad$ $fn = n\ /\ (p+n)$

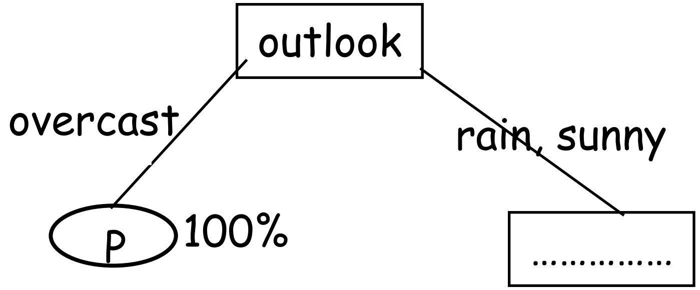
$$gini(S) = 1 - fp^2 - fn^2$$

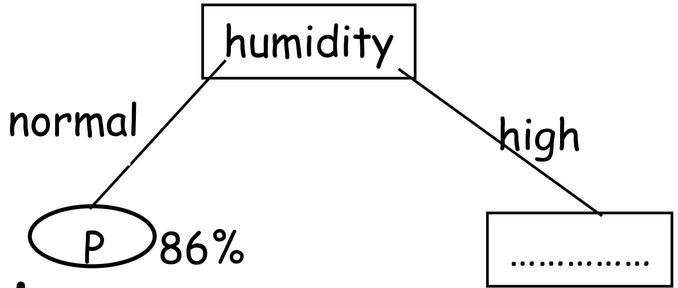
- If dataset *S* is split into S_1, S_2 then

$$gini_{split}(S_1, S_2) = gini(S_1)\cdot(p_1+n_1)/(p+n) + gini(S_2)\cdot(p_2+n_2)/(p+n)$$

Gini index - play tennis example

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

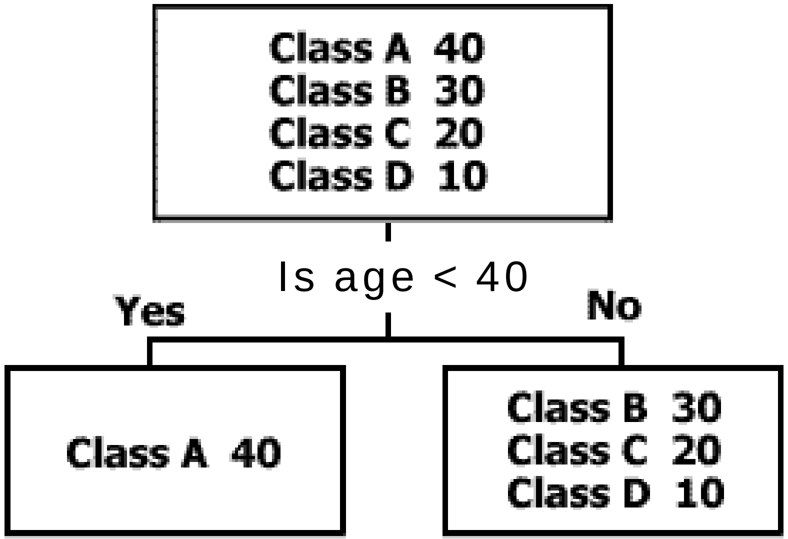


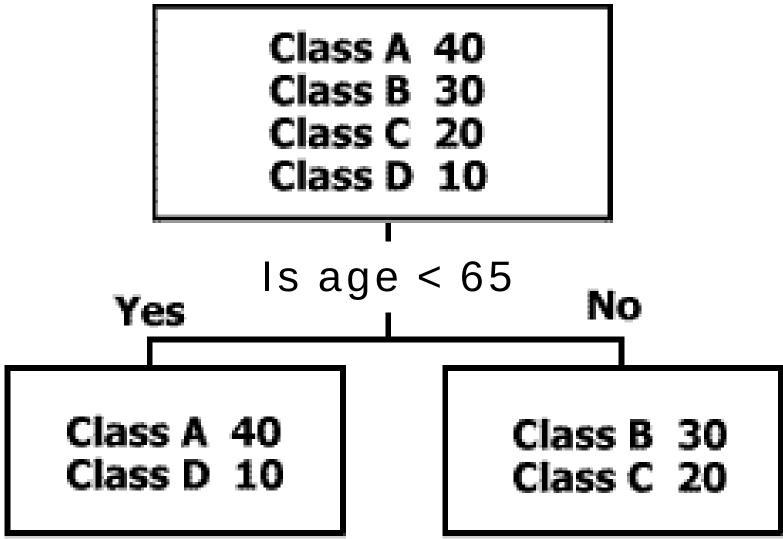


- ⌘ **Two top best splits at root node:**
- ⌘ **Split on outlook:**
  - ⌘ S_1: {overcast} (4Pos, 0Neg) S_2: {sunny, rain}
- ⌘ **Split on humidity:**
  - ⌘ S_1: {normal} (6Pos, 1Neg) S_2: {high}

Entropy vs. Gini (on continuous attributes)

- Gini tends to isolate the largest class from all other classes
- Entropy tends to find groups of classes that add up to 50% of the data





Other criteria in decision tree construction

- Branching scheme:
  - binary vs. *k*-ary splits
  - categorical vs. continuous attributes
- Stop rule: how to decide that a node is a leaf:
  - all samples belong to same class
  - *impurity* measure below a given threshold
  - no more attributes to split on
  - no samples in partition
- Labeling rule: a leaf node is labeled with the class to which most samples at the node belong

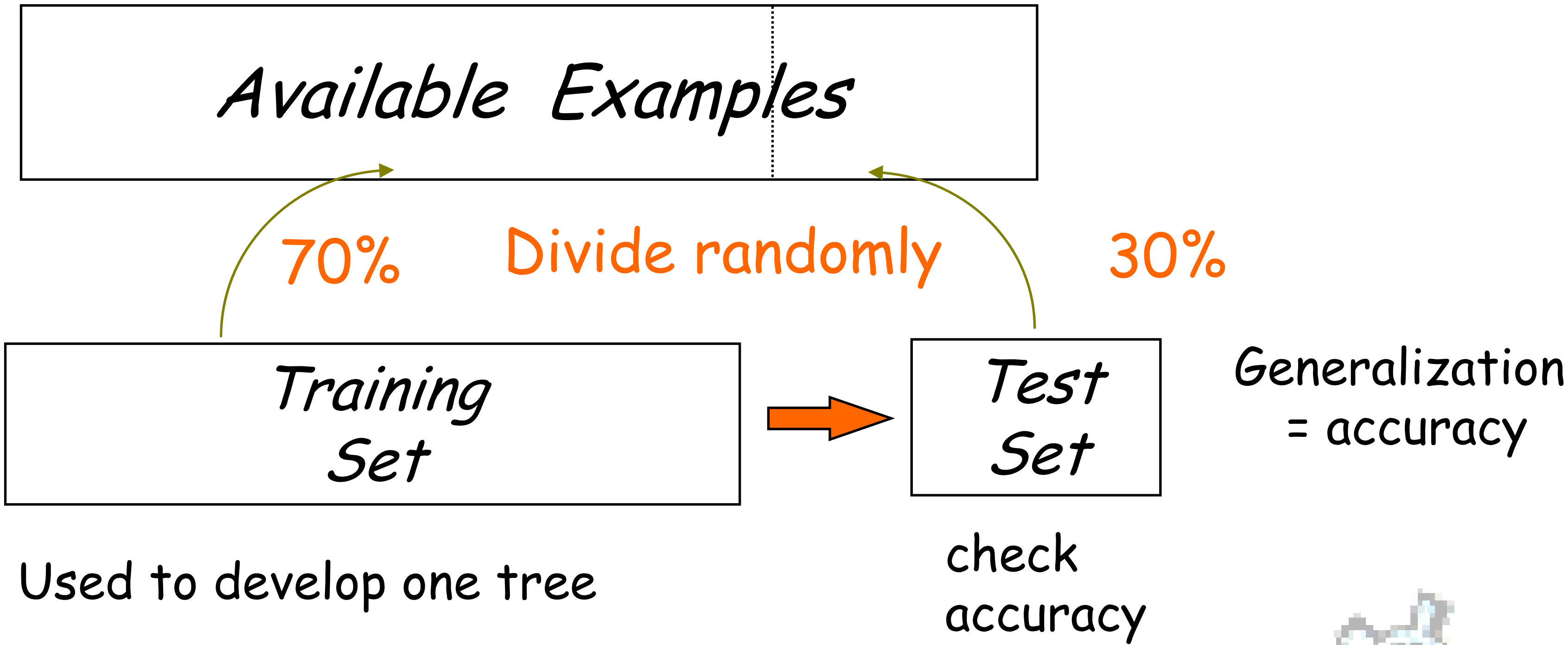
The *overfitting* problem

- Ideal goal of classification: find the simplest decision tree that fits the data and generalizes to unseen data
  - intractable in general
- A decision tree may become too complex if it *overfits* the training samples, due to
  - noise and outliers, or
  - too little training data, or
  - local maxima in the greedy search
- Two heuristics to avoid overfitting:
  - Stop earlier: Stop growing the tree earlier.
  - Post-prune: Allow overfit, and then simplify the tree.

Stopping vs. pruning

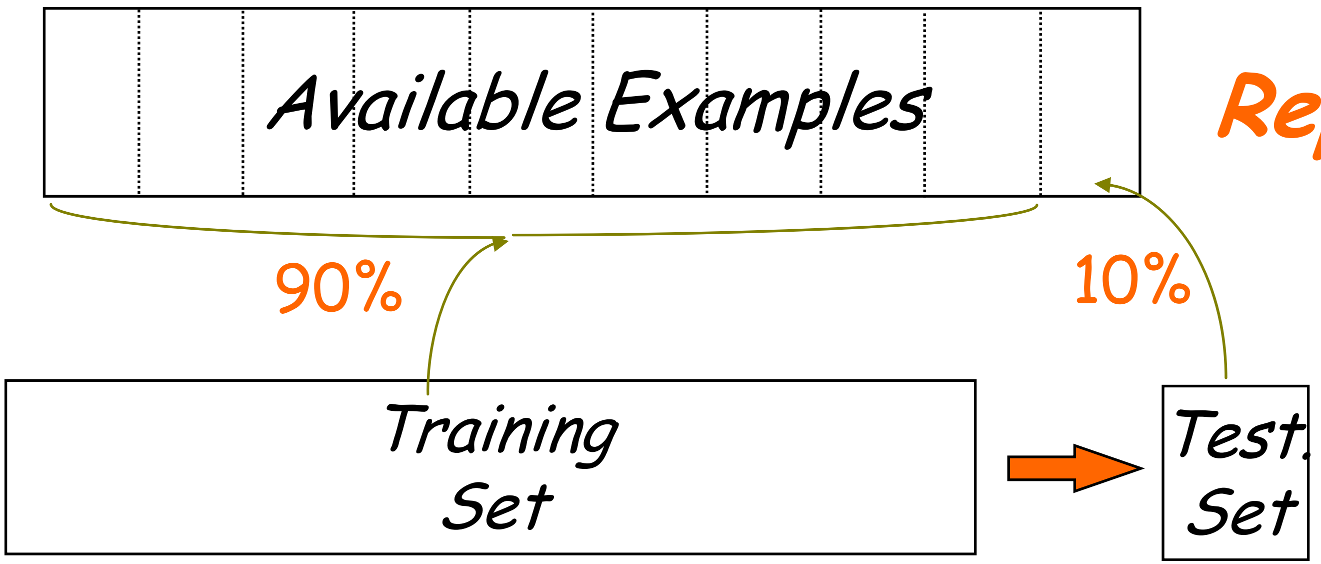
- **Stopping**: Prevent the split on an attribute (predictor variable) if it is below a level of statistical significance - simply make it a leaf (CHAID)
- **Pruning**: After a complex tree has been grown, replace a split (subtree) with a leaf if the *predicted* validation error is no worse than the more complex tree (CART, C4.5)
- Integration of the two: **PUBLIC** (Rastogi and Shim 98) – estimate pruning conditions (lower bound to minimum cost subtrees) during construction, and use them to stop.

If dataset is large



If data set is not so large

- **Cross-validation**



Repeat 10 times

Generalization
= mean and stddev
of accuracy

Used to develop 10 different tree

Tabulate
accuracies

Categorical vs. continuous attributes

- Information gain criterion may be adapted to continuous attributes using binary splits
- Gini index may be adapted to categorical.

- Typically, discretization is not a pre-processing step, but is performed dynamically during the decision tree construction.

Summarizing …

tool→	C4.5	CART	CHAID
arity of split	binary and K-ary	binary	K-ary
split criterion	information gain	gini index	χ^2
stop vs. prune	prune	prune	stop
type of attributes	categorical +continuous	categorical +continuous	categorical

Scalability to large databases

- What if the dataset does not fit main memory?
- Early approaches:
  - Incremental tree construction (Quinlan 86)
  - Merge of trees constructed on separate data partitions (Chan & Stolfo 93)
  - Data reduction via sampling (Cattlet 91)
- Goal: handle order of 1G samples and 1K attributes
- Successful contributions from data mining research
  - SLIQ (Mehta et al. 96)
  - SPRINT (Shafer et al. 96)
  - PUBLIC (Rastogi & Shim 98)
  - RainForest (Gehrke et al. 98)

Module outline

- The classification task
- Main classification techniques
  - Decision trees
  - Bayesian classifiers
  - Hints to other methods
- Application to a case-study in fraud detection: planning of fiscal audits

Backpropagation

- Is a **neural network** algorithm, performing on multilayer feed-forward networks (Rumelhart et al. 86).
- A network is a set of **connected input/output** units where each connection has an associated **weight**.
- The weights are adjusted during the training phase, in order to correctly predict the class label for samples.

Backpropagation

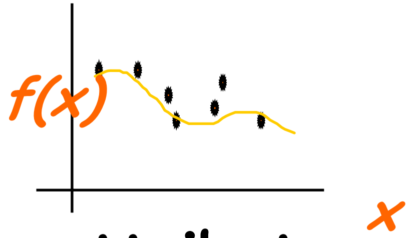
PROS

- High accuracy
- Robustness w.r.t. noise and outliers

CONS

- Long training time
- Network topology to be chosen empirically
- Poor interpretability of learned weights

Prediction and (statistical) regression



- Regression = construction of models of continuous attributes as functions of other attributes
- The constructed model can be used for prediction.
- E.g., a model to predict the sales of a product given its price
- Many problems solvable by linear regression, where attribute Y (response variable) is modeled as a linear function of other attribute(s) X (predictor variable(s)):

$$Y = a + b \cdot X$$

- Coefficients a and b are computed from the samples using the least square method.

Other methods (not covered)

- *K*-nearest neighbors algorithms
- Case-based reasoning
- Genetic algorithms
- Rough sets
- Fuzzy logic
- Association-based classification (Liu et al 98)

Module outline

- The classification task
- Main classification techniques
  - Decision trees
  - Bayesian classifiers
  - Hints to other methods
- Application to a case-study in fraud detection: planning of fiscal audits

Fraud detection and audit planning

- A major task in fraud detection is constructing *models* of fraudulent behavior, for:
  - preventing future frauds (*on-line* fraud detection)
  - discovering past frauds (*a posteriori* fraud detection)
- Focus on a posteriori FD: analyze historical audit data to plan effective future audits
- Audit planning is a key factor, e.g. in the fiscal and insurance domain:
  - tax evasion (from incorrect/fraudulent tax declarations) estimated in Italy between 3% and 10% of GNP

Case study

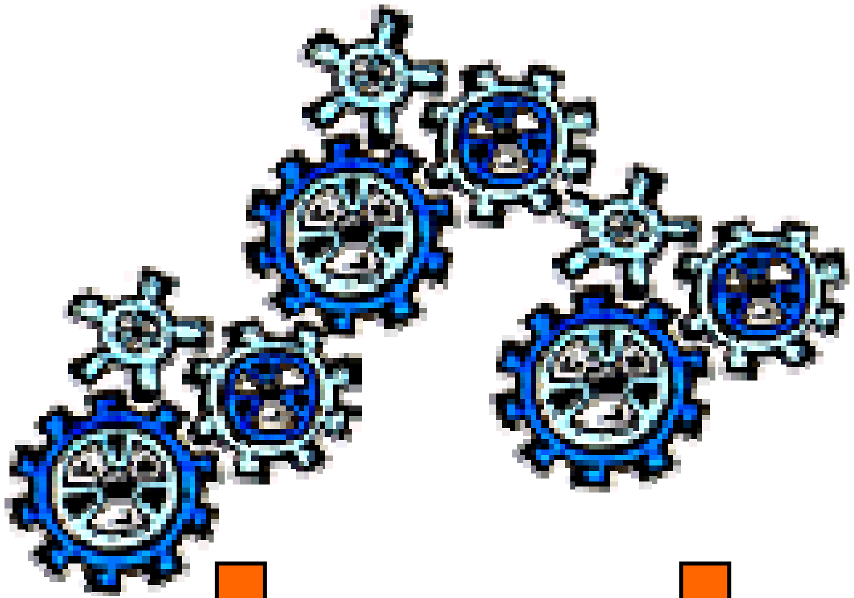
- Conducted by our Pisa KDD Lab (Bonchi et al 99)
- A data mining project at the Italian Ministry of Finance, with the aim of assessing:
  - the potential of a KDD process oriented to planning audit strategies;
  - a methodology which supports this process;
  - an integrated logic-based environment which supports its development.

Audit planning

- Need to face a trade-off between conflicting issues:
  - *maximize audit benefits*: select subjects to be audited to maximize the recovery of evaded tax
  - *minimize audit costs*: select subjects to be audited to minimize the resources needed to carry out the audits.
- Is there a KDD methodology which may be tuned according to these options?
- How extracted knowledge may be combined with domain knowledge to obtain useful audit models?

Autofocus data mining

policy options, business rules



fine parameter tuning of mining tools

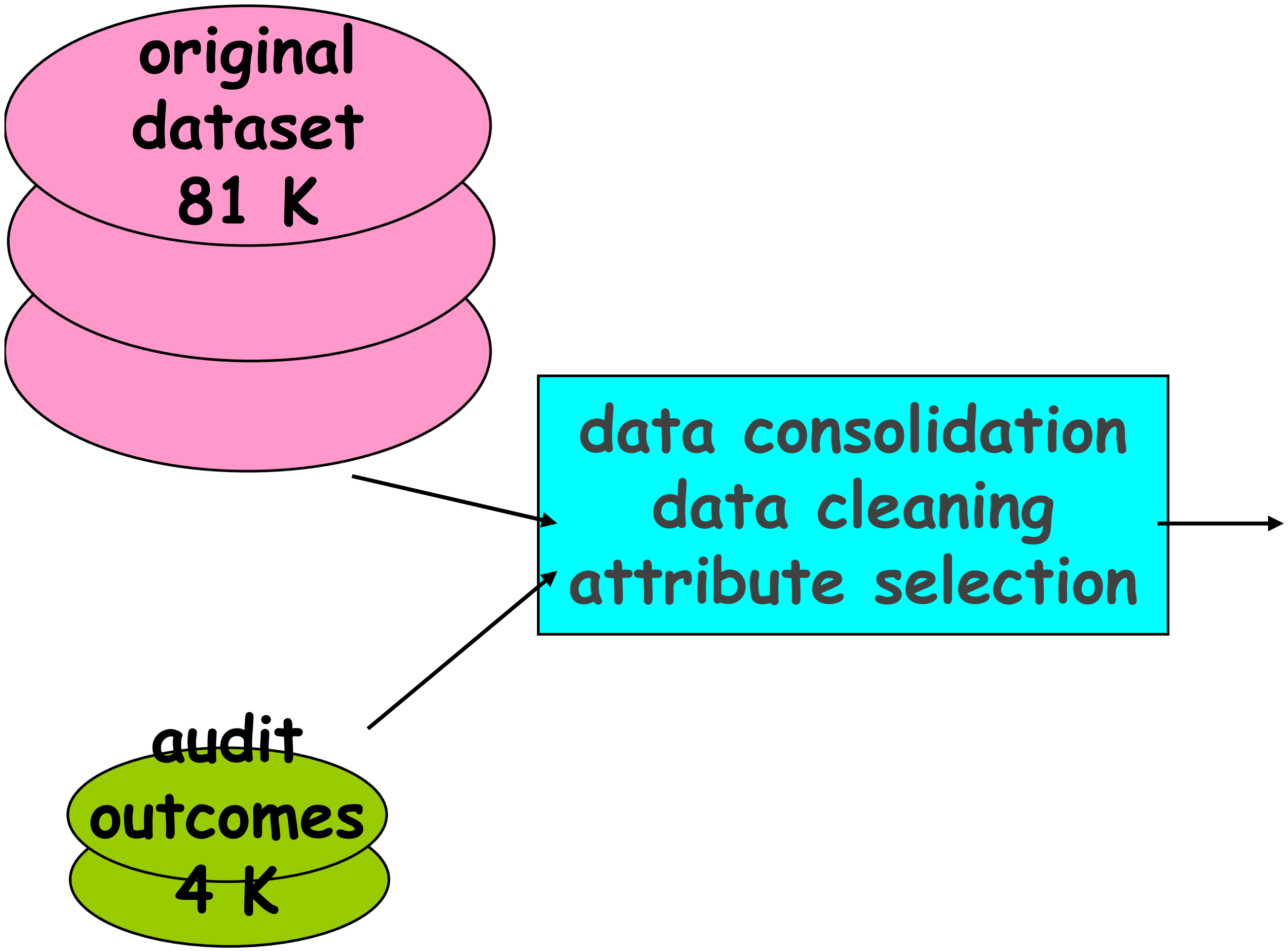
Classification with decision trees

- **Reference technique:**
  - **Quinlan's C4.5, and its evolution C5.0**
- **Advanced mechanisms used:**
  - **pruning factor**
  - **misclassification weights**
  - **boosting factor**

Available data sources

- **Dataset: tax declarations, concerning a targeted class of Italian companies, integrated with other sources:**
  - social benefits to employees, official budget documents, electricity and telephone bills.
- **Size: 80 K tuples, 175 numeric attributes.**
- **A subset of 4 K tuples corresponds to the *audited* companies:**
  - outcome of audits recorded as the *recovery* attribute (= *amount of evaded tax ascertained*)

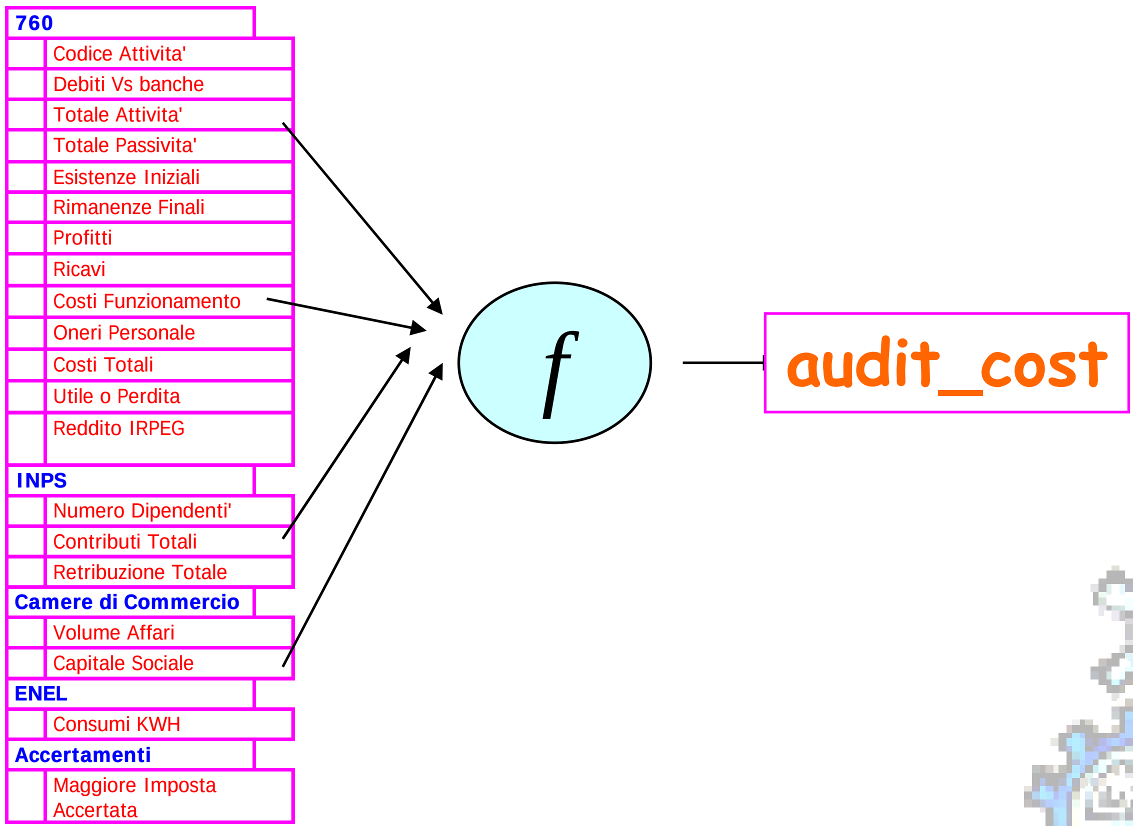
Data preparation



- **TAX DECLARATION**
  - Codice Attivita'
  - Debiti Vs banche
  - Totale Attivita'
  - Totale Passivita'
  - Esistenze Iniziali
  - Rimanenze Finali
  - Profitti
  - Ricavi
  - Costi Funzionamento
  - Oneri Personale
  - Costi Totali
  - Utile o Perdita
  - Reddito IRPEG
- **SOCIAL BENEFITS**
  - Numero Dipendenti'
  - Contributi Totali
  - Retribuzione Totale
- **OFFICIAL BUDGET**
  - Volume Affari
  - Capitale Sociale
- **ELECTRICITY BILLS**
  - Consumi KWH
- **AUDIT**
  - Recovery

Cost model

- A derived attribute audit_cost is defined as a function of other attributes



Cost model and the target variable

- recovery of an audit after the audit cost
  actual_recovery = recovery - audit_cost

- target variable (class label) of our analysis is set as the Class of Actual Recovery (c.a.r.):

- $$c.a.r. = \begin{cases} negative & \text{if } actual\_recovery \leq 0 \\ positive & \text{if } actual\_recovery > 0. \end{cases}$$

Training set & test set

- Aim: build a binary classifier with c.a.r. as target variable, and evaluate it
- Dataset is partitioned into:
  - training set, to build the classifier
  - test set, to evaluate it
- Relative size: incremental samples approach
- In our case, the resulting classifiers improve with larger training sets.
- Accuracy test with 10-fold cross-validation

Quality assessment indicators

- The obtained classifiers are evaluated according to several indicators, or metrics
- Domain-independent indicators
  - confusion matrix
  - misclassification rate
- Domain-dependent indicators
  - audit #
  - actual recovery
  - profitability
  - relevance

Domain-independent quality indicators

- **confusion matrix** (of a given classifier)

negative	positive	← classified as
TN	FP	actual class negative
FN	TP	actual class positive

TN (TP): true negative (positive) tuples
FN (FP): false negative (positive) tuples

misclassification rate =

$$\# (FN \cup FP) / \# \text{test-set}$$

Domain-dependent quality indicators

- audit # (of a given classifier): number of tuples classified as positive =
  $$\#(FP \cup TP)$$
- actual recovery: total amount of actual recovery for all tuples classified as positive
- profitability: average actual recovery per audit
- relevance: ratio between profitability and misclassification rate

The REAL case

- Classifiers can be compared with the REAL case, consisting of the whole test-set:

- audit # (REAL) = 366
- actual recovery(REAL) = 159.6 M euro

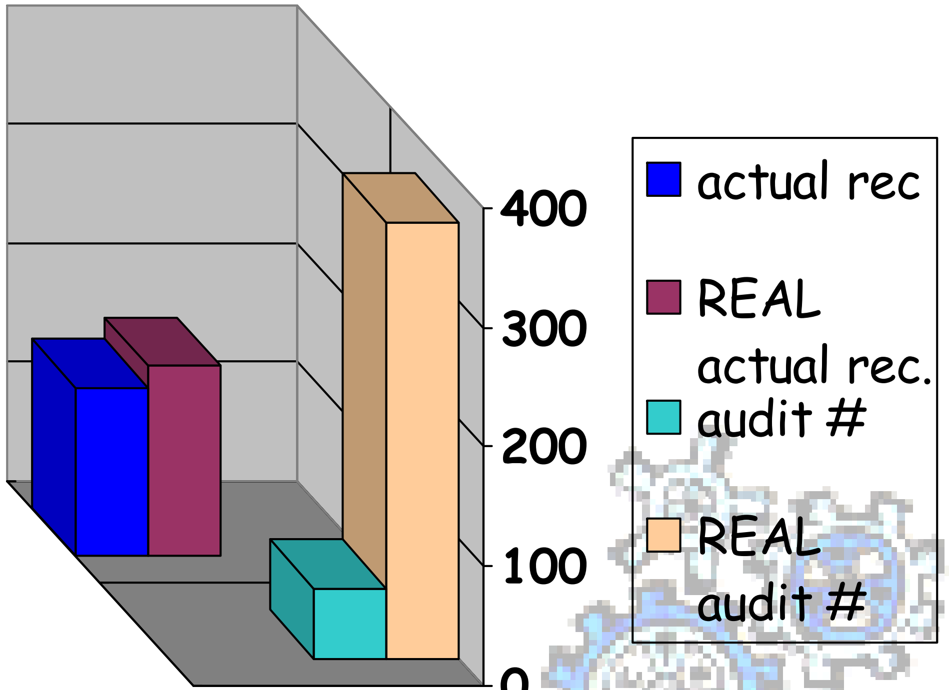
Controlling classifier construction

- *maximize audit benefits*: minimize FN
- *minimize audit costs*: minimize FP
- hard to get both!
  - unbalance tree construction towards eiher negatives or positives
- which parameters may be tuned?
  - misclassification weights, e.g., trade 1 FN for 10 FP
  - replication of minority class
  - boosting and pruning level

Model evaluation: classifier 1 (min FP)

- no replication in training-set (unbalance towards negative)
- 10-trees adaptive boosting

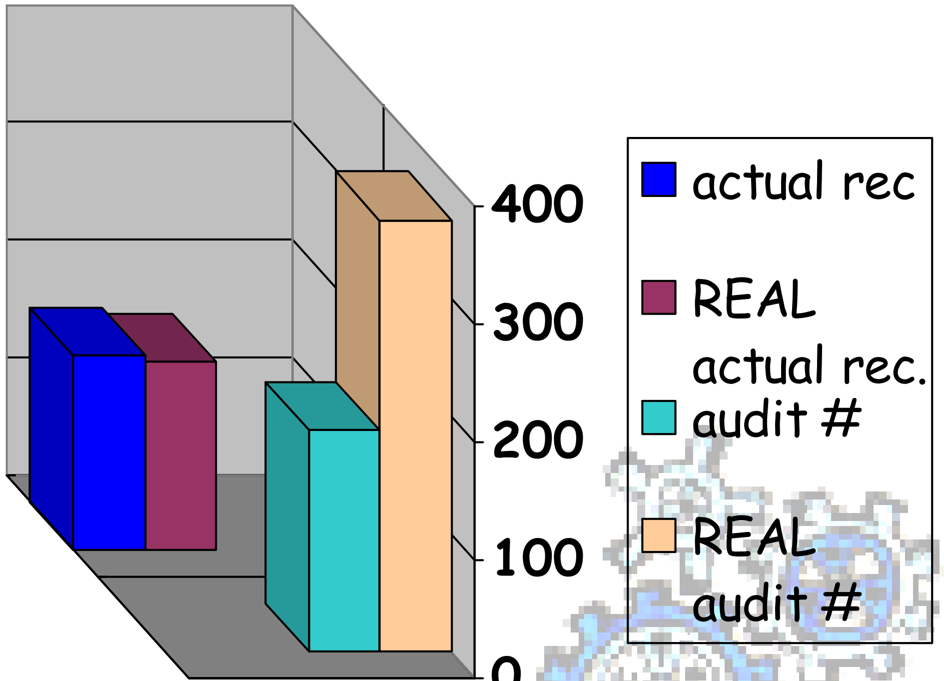
- *misc. rate* = 22%
- *audit #* = 59 (11 FP)
- *actual rec.*= 141.7 Meuro
- *profitability* = 2.401



Model evaluation: classifier 2 (min FN)

- replication in training-set (balanced neg/pos)
- misc. weights (trade 3 FP for 1 FN)
- 3-trees adaptive boosting

- *misc. rate* = 34%
- *audit #* = 188 (98 FP)
- *actual rec.*= 165.2 Meuro
- *profitability* = 0.878



What have we achieved?

- Idea of a KDD methodology tailored for a vertical application: audit planning
  - define an audit cost model
  - monitor training- and test-set construction
  - assess the quality of a classifier
  - tune classifier construction to specific policies
- Its formalization requires a flexible KDSE – knowledge discovery support environment, supporting:
  - integration of deduction and induction
  - integration of domain and induced knowledge
  - separation of conceptual and implementation level

References - classification

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 13, 1997.
- F. Bonchi, F. Giannotti, G. Mainetto, D. Pedreschi. Using Data Mining Techniques in Fiscal Fraud Detection. In Proc. DaWak'99, First Int. Conf. on Data Warehousing and Knowledge Discovery, Sept. 1999.
- F. Bonchi , F. Giannotti, G. Mainetto, D. Pedreschi. A Classification-based Methodology for Planning Audit Strategies in Fraud Detection. In Proc. KDD-99, ACM-SIGKDD Int. Conf. on Knowledge Discovery & Data Mining, Aug. 1999.
- J. Catlett. *Megainduction: machine learning on very large databases*. PhD Thesis, Univ. Sydney, 1991.
- P. K. Chan and S. J. Stolfo. Metalearning for multistrategy and parallel learning. In Proc. 2nd Int. Conf. on Information and Knowledge Management, p. 314-323, 1993.
- J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.
- J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In Proc. KDD'95, August 1995.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.
- B. Liu, W. Hsu and Y. Ma. Integrating classification and association rule mining. In Proc. KDD'98, New York, 1998.

References - classification

- J. Magidson. The CHAID approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, *Advanced Methods of Marketing Research*, pages 118-159. Blackwell Business, Cambridge Massechusetts, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. In Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96), Avignon, France, March 1996.
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Diciplinary Survey. *Data Mining and Knowledge Discovery* 2(4): 345-389, 1998
- J. R. Quinlan. Bagging, boosting, and C4.5. In Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96), 725-730, Portland, OR, Aug. 1996.
- R. Rastogi and K. Shim. Public: A decision tree classifer that integrates building and pruning. In Proc. 1998 Int. Conf. Very Large Data Bases, 404-415, New York, NY, August 1998.
- J. Shafer, R. Agrawal, and M. Mehta. SPRINT : A scalable parallel classifier for data mining. In Proc. 1996 Int. Conf. Very Large Data Bases, 544-555, Bombay, India, Sept. 1996.
- 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.
- D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning internal representation by error propagation. In D. E. Rumelhart and J. L. McClelland (eds.) *Parallel Distributed Processing*. The MIT Press, 1986