BABEŞ-BOLYAI UNIVERSITY
Faculty of Computer Science and Mathematics

# ARTIFICIAL INTELLIGENCE

**Intelligent systems**

Machine learning

Decision trees

# Topics

A. Short introduction in Artificial Intelligence (AI)

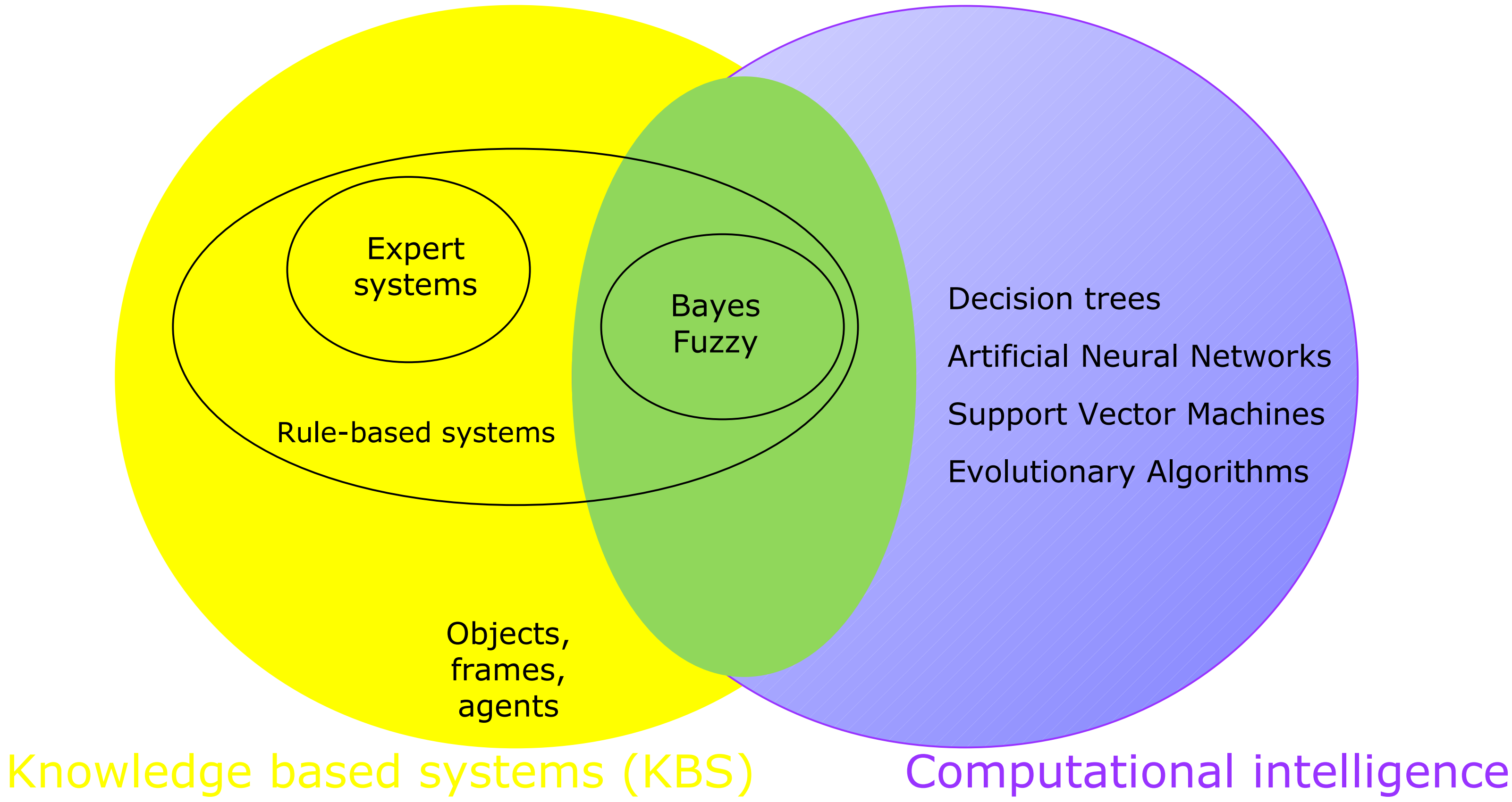
A. Solving search problems
   A. Definition of search problems
   B. Search strategies
      A. Uninformed search strategies
      B. Informed search strategies
      C. Local search strategies (Hill Climbing, Simulated Annealing, Tabu Search, Evolutionary algorithms, PSO, ACO)
      D. Adversarial search strategies

C. **Intelligent systems**
   A. Rule-based systems in certain environments
   B. Rule-based systems in uncertain environments (Bayes, Fuzzy)
   C. **Learning systems**
      A. **Decision Trees**
      B. Artificial Neural Networks
      C. Support Vector Machines
      D. Evolutionary algorithms
   D. Hybrid systems

# Useful information

- Chapter VI (18) of *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*
- Chapters 10 and 11 of *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- Chapter V of *D. J. C. MacKey, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003*
- Chapters 3 of *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*

# Intelligent systems



# Content

- Intelligent systems
  - Automatic learning systems (ALS)
    - Machine Learning - ML
      - Problem
      - Design
      - Typology
        - Supervised learning
        - Unsupervised learning
        - Reinforcement learning
        - Learning theory
    - Systems
      - Decision trees

# Intelligent systems – Machine Learning (ML)

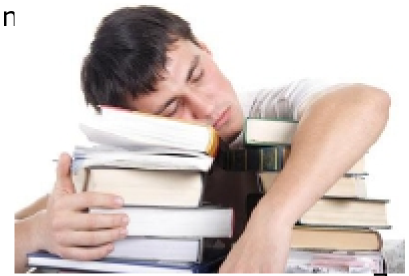
- Problem
  - “How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?”

- Applications
  - Image and voice recognition
    - Handwritten recognition
    - Face detection
  - Computer vision
    - Obstacle detection
    - Footprint recognition
  - Bio-surveillance
  - Robot control
  - Predictions
  - Medical diagnostic
  - Fraud detection

# Intelligent systems – Machine Learning (ML)

- Definition
  - Arthur Samuel (1959)
    - "field of study that gives computers the ability to learn without being explicity programmed"
  - Herbert Simon (1970)
    - "Learning is any process by which a system improves performance from experience."
  - Tom Mitchell (1998)
    - "a well-posed learning problem is defined as follows: He says that a computer program is set to learn from an experience E with respect to some task T and some performance measure P if its performance on T as measured by P improves with experience E"
  - Ethem Alpaydin (2010)
    - Programming computers to optimize a performance criterion using example data or past experience.

- Necessity
  - Better computational systems
    - To difficult or to expensive to be constructed manually
      - Systems that automatically adapt
        - Spam filters
      - Systems that discovers information in large database → data minin
        - Financial analysis
        - Text/image analyses
  - Understanding the biological systems



# Intelligent systems – Machine Learning (ML)

- Design
  - Improve of task T
    - Establish the goal (what has to be learn) – objective function – and its representation
    - Select a learning algorithms to perform the inference of the goal based on experience
  - Respect a performance metric P
    - Evaluation of the algorithm's performances
  - Based on experience E
    - Select an experience database

  - Example
    - T: playing checkers
    - P: percent of winning games
    - E: playing the game

    - T: handwritten recognition
    - P: percent of correct recognized words
    - E: database of images with different words

    - T: separate the spams
    - P: percent of correct classified emails
    - E: databases with annotated emails

# Intelligent systems – Machine Learning (ML)

- Design → choose the objective function
  - Which is the function that must be learn?
    - Ex.: checkers game → a function that:
      - Selects the next move
      - Evaluates a move
    - In order to identify the best move
  - Representation of objective function
    - Different representations
      - Tabel
      - Symbolic rules
      - Numeric functions
      - Probabilistic functions
    - Ex. Checkers game
      - A linear combinations of # white pieces, # black pieces, # of white compromised pieces, # black compromised pieces
    - There is a trade-off between
      - Expressiveness of representation
      - Easy of learning
    - Objective function computation
      - Polynomial time
      - Non-polynomial time

# Intelligent systems – Machine Learning (ML)

- Design → select a learning algorithm
  - Algorithm
    - By using the training data
    - Induce the hypothesis definition that
      - Match the data
      - Generalize the un - seen data
  - Main principle
    - Error minimisation (cost function – loss function)

- Design → evaluation of a learning system
  - Experimental
    - By comparing different methods on different data (cross-validation)
    - Collect data based on performances
      - Accuracy, training time, testing time
    - Statistical analyse of the differences
  - Theoretic
    - Mathematical analyse of algorithms and theorem proving
      - Computational complexity
      - Ability to match the training data
      - Complexity of the most relevant sample for learning

# Intelligent systems – Machine Learning (ML)

- Design → evaluation of a learning system
  - Comparing the performances of 2 algorithms for solving a given problem
    - Performance measures
      - Parameters of a statistic series
      - Proportion (percent) computed for a statistical series (ex. Accuracy)
    - Comparing based on confidence intervals
      - For a problem and 2 solving algorithms with performances $p_1$ and $p_2$
      - Confidence intervals $I_1=[p_1-\Delta_1,p_1+\Delta_1]$ şi $I_2=[p_2-\Delta_2,p_2+\Delta_2]$
      - If $I_1 \cap I_2=\emptyset$ ➔ algorithm 1 works better than algorithm 2 (for the given problem)
      - if $I_1 \cap I_2 \neq \emptyset$ ➔ impossible to decide
    - Confidence interval for the mean (average)
      - For a statistical series of n data, with computed mean m and dispersion $\sigma$, determine the confidence interval of the mean $\mu$
      - $P(-z \leq (m-\mu)/(\sigma/\sqrt{n}) \leq z) = 1 - \alpha$ ➔ $\mu \in [m-z\sigma/\sqrt{n}, m+z\sigma/\sqrt{n}]$
      - $P = 95\%$ ➔ $z = 1.96$
    - Confidence interval for accuracy
      - For an accuracy $p$ computed for $n$ data, determine the confidence interval of accuracy
      - $p \in [p-z(p(1-p)/n)^{1/2}, p+z(p(1-p)/n)^{1/2}]$
      - $P = 95\%$ ➔ $z = 1.96$

| $P=1-\alpha$ | $z$ |
|---|---|
| 99.9% | 3.3 |
| 99.0% | 2.577 |
| 98.5% | 2.43 |
| 97.5% | 2.243 |
| 95.0% | 1.96 |
| 90.0% | 1.645 |
| 85.0% | 1.439 |
| 75.0% | 1.151 |

# Intelligent systems – Machine Learning (ML)

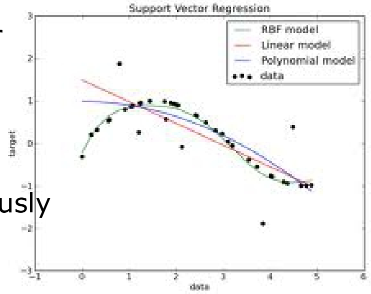
- Design → choose the training database
  - Based on
    - Direct experience
      - Pairs (in, out) that are useful for the objective function
      - Eg. Checkers game→ board game annotated by correct or incorrect move
    - Indirect experience
      - Useful feedback (unlike i/o pairs) for the objective function
      - Eg. Checkers game→ sequences of moves and the final score of the game
  - Data sources
    - Random generated examples
      - Positive and negative examples
    - Positive examples collected by a learner
    - Real examples
  - Content
    - Training data
    - Test data
  - Characteristics
    - Independent data
      - Otherwise → collective learning
    - Training and testing data must respect the same distribution law
      - Otherwise → *transfer learning/inductive transfer*
        - Vehicle recognition → truck recognition
        - Text analyses
        - Spam filters

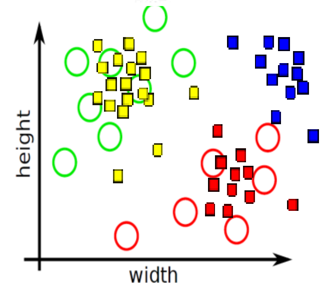
# Intelligent systems – Machine Learning (ML)

- Design → choose the training database
  - Characteristics extracted (attributes) from raw data
    - Quantitative characteristics → nominal or rational scale
      - Continuous values → weight
      - Discrete values → # of computers
      - Range values → event times
    - Qualitative characteristics
      - Nominal → colour
      - Ordinal → sound intensity (low, medium, high)
    - Structured
      - Trees – root is a generalisation of children (vehicle → car, bus, tractor, truck)
  - Data transformation
    - Standardisation → numerical attributes
      - Remove the scale effect (different scale and units)
      - Raw values are transformed in z scores
        - $Z_{ij} = (x_{ij} - \mu_j)/\sigma_j$, where $x_{ij}$ – value of $j^{th}$ attribute of $i^{th}$ instance, $\mu_j$ ($\sigma_j$) is the mean (standard deviation) of $j^{th}$ attribute for all instances
    - Selection of some attributes

# Intelligent systems – Machine Learning (ML)

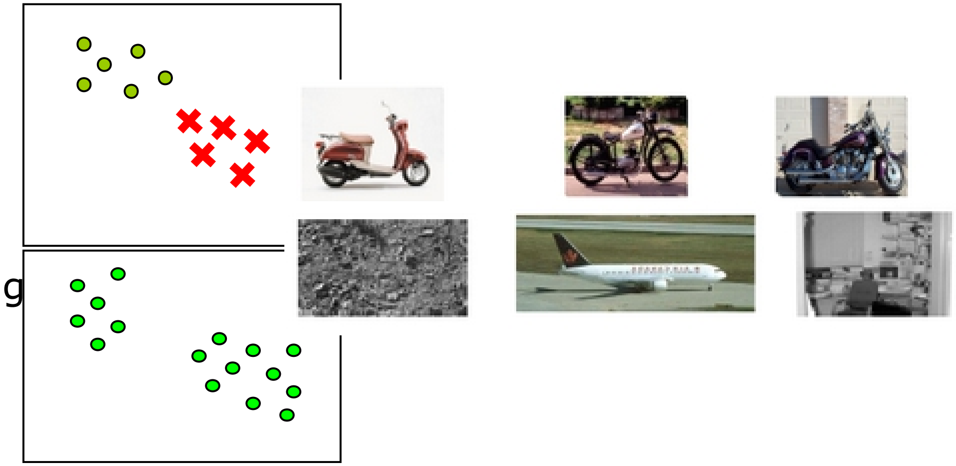
- Typology
  - Based on their aim (goal)
    - ISs for prediction
      - Aim: predict the output for a new input based on a previously learned model
      - Eg. predicting sales of a product for a time in the future based on price, calendar month, region, average income
    - ISs for regression
      - Aim: estimation of the (uni or multi - variable) function shape based on a previously learned model
      - Eg.: estimate the function that models the edge of a surface
    - ISs for classification
      - Aim: classify an object into one or more – known or unknown - categories based on their characteristics
      - Eg.: diagnostic systems for cancer: malign or benign or normal
    - ISs for planning
      - Aim: generate a sequence of optimal actions for performing a task
      - Eg.: planning the moves of a robot from a position to a source of energy



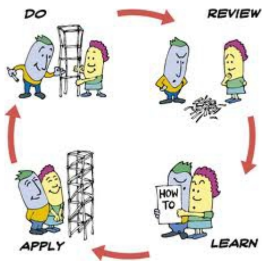




# Intelligent systems – Machine Learning (ML)

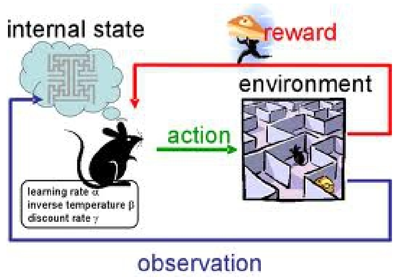
- Typology
  - Based on the experience learned during training process
    - ISs with supervised learning
    - ISs with unsupervised learning



    - ISs with active learning




    - ISs with reinforcement learning



# Intelligent systems – Machine Learning (ML)

- Supervised learning
  - Definire
  - Exemple
  - Proces
  - Calitatea învăţării
    - Metode de evaluare
    - Măsuri de performanţă
  - Tipologie

# Intelligent systems – Machine Learning (ML)

## Învăţare supervizată

- Scop
  - Furnizarea unei ieşiri corecte pentru o nouă intrare
- Definire
  - Se dă un set de date (exemple, instanţe, cazuri)
    - date de antrenament – sub forma unor perechi ($atribute\_data_i$, $ieşire_i$), unde
      - i =1,N (N = nr datelor de antrenament)
      - $atribute\_data_i$= ($atr_{i1}$, $atr_{i2}$, ..., $atr_{im}$), m – nr atributelor (caracteristicilor, proprietăţilor) unei date
      - $ieşire_i$
        - o categorie dintr-o mulţime dată (predefinită) cu k elemente (k – nr de clase) ➔ problemă de clasificare
        - un număr real ➔ problemă de regresie
    - date de test - sub forma (atribute_datai), i =1,n (n = nr datelor de test).
  - Să se determine
    - o funcţie (necunoscută) care realizează corespondenţa atribute – ieşire pe datele de antrenament
    - ieşirea (clasa/valoarea) asociată unei date (noi) de test folosind funcţia învăţată pe datele de antrenament
- Alte denumiri
  - Clasificare (regresie), învăţare inductivă
- Proces → 2 etape
  - Antrenarea
    - Învăţarea, cu ajutorul unui algoritm, a modelului de clasificare
  - Testarea
    - Testarea modelului folosind date de test noi (unseen data)
- Caracteristic
  - BD experimentală adnotată (pt. învăţare)

# Intelligent systems – Machine Learning (ML)

## Învăţare supervizată

- Tip de probleme
  - regresie
    - Scop: predicţia output-ului pentru un input nou
    - Output continuu (nr real)
    - Ex.: predicţia preţurilor
  - clasificare
    - Scop: clasificarea (etichetarea) unui nou input
    - Output discret (etichetă dintr-o mulţime predefinită)
    - Ex.: detectarea tumorilor maligne
- Exemple de probleme
  - Recunoaşterea scrisului de mână
  - Recunoaşterea imaginilor
  - Previziunea vremii
  - Detecţia spam-urilor

# Intelligent systems – Machine Learning (ML)

## Învăţare supervizată

- Calitatea învăţării
  - Definire
    - o măsură de performanţă a algoritmului
      - ex. acurateţea (Acc = nr de exemple corect clasificate / nr total de exemple)
    - calculată în
      - faza de antrenare
      - faza de testare
  - Metode de evaluare
    - Seturi disjuncte de antrenare şi testare
      - setul de antrenare poate fi împărţit în date de învăţare şi date de validare
      - setul de antrenare este folosit pentru estimarea parametrilor modelului (cei mai buni parametri obţinuţi pe validare vor fi folosiţi pentru construcţia modelului final)
      - pentru date numeroase
    - Validare încrucişată cu mai multe (h) sub-seturi egale ale datelor (de antrenament)
      - separararea datelor de h ori în (*h-1* sub-seturi pentru învăţare şi 1 sub-set pt validare)
      - dimensiunea unui sub-set = dimensiunea setului / h
      - performanţa este dată de media pe cele h rulări (ex. *h = 5* sau *h = 10)*
      - pentru date puţine
    - Leave-one-out cross-validation
      - similar validării încrucişate, dar h = nr de date → un sub-set conţine un singur exemplu
      - pentru date foarte puţine
  - Dificultăţi
    - Învăţare pe derost (overfitting) → performanţă bună pe datele de antrenament, dar foarte slabă pe datele de test

# Intelligent systems – Machine Learning (ML)

## Învăţare supervizată

- Calitatea învăţării
  - Măsuri de performanţă
    - Măsuri statistice
      - acurateţea
      - Precizia
      - Rapelul
      - Scorul F1
    - Eficienţa
      - În construirea modelului
      - În testarea modelului
    - Robusteţea
      - Tratarea zgomotelor şi a valorilor lipsă
    - Scalabilitatea
      - Eficienţa gestionării seturilor mari de date
    - Interpretabilitatea
      - Modelului de clasificare
    - Proprietatea modelului de a fi compact
    - Scoruri

# Intelligent systems – Machine Learning (ML)

## Învăţare supervizată

- Calitatea învăţării → Măsuri de performanţă → Măsuri statistice
  - Acurateţea
    - Nr de exemple corect clasificate / nr total de exemple
    - Opusul erorii
    - Calculată pe
      - Setul de validare
      - Setul de test
    - Uneori
      - Analiză de text
      - Detectarea intruşilor într-o reţea
      - Analize financiare

    este importantă doar o singură clasă (clasă pozitivă) → restul claselor sunt negative
  - Precizia (P)
    - nr. de exemple pozitive corect clasificate / nr. total de exemple clasificate ca pozitive
    - probabilitatea ca un exemplu clasificat pozitiv să fie relevant
    - *TP / (TP + FP)*
  - Rapelul (R)
    - nr. de exemple pozitive corect clasificate / nr. total de exemple pozitive
    - Probabilitatea ca un exemplu pozitiv să fie identificat corect de către clasificator
    - *TP/ (TP +FN)*
    - Matrice de confuzie → rezultate reale vs. rezultate calculat
  - Scorul F1
    - Combină precizia şi rapelul, facilitând compararea a 2 algoritmi
    - Media armonică a preciziei şi rapelului
    - *2PR/(P+R)*

| | | Rezultate reale | |
|---|---|---|---|
| | | Clasa pozitivă | Clasa(ele) negativă(e) |
| Rezultate calculate | Clasa pozitivă | *True positiv (TP)* | *False positiv (FP)* |
| | Clasa(ele) negativă(e) | *False negative (FN)* | *True negative (TN)* |

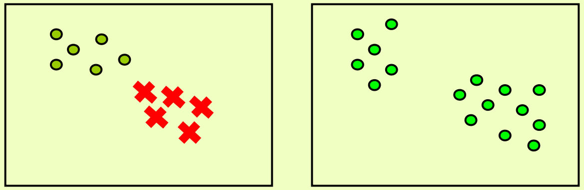
# Intelligent systems – Machine Learning (ML)

- Învăţare ne-supervizată
  - Definire
  - Exemple
  - Proces
  - Metode de evaluare şi măsuri de performanţă
  - Tipologie

# Intelligent systems – Machine Learning (ML)

Învăţare ne-supervizată

- Scop
  - Găsirea unui model sau a unei structuri utile a datelor
  - Împărţirea unor exemple neetichetate în submulţimi disjuncte (clusteri) astfel încât:
    - exemplele din acelaşi cluster sunt foarte similare
    - exemplele din clusteri diferiţi sunt foarte diferite
- Definire
  - Se dă un set de date (exemple, instanţe, cazuri)
    - Date de antrenament sub forma ***atribute_data***$_i$, unde
      - $i = 1, N$ ($N$ = nr datelor de antrenament)
      - ***atribute_data***$_i$= ($atr_{i1}$, $atr_{i2}$, ..., $atr_{im}$), $m$ – nr atributelor (caracteristicilor, proprietăţilor) unei date
    - Date de test sub forma (***atribute_data***$_i$), $i = 1, n$ ($n$ = nr datelor de test)
  - Se determină
    - o funcţie (necunoscută) care realizează gruparea datelor de antrenament în mai multe clase
      - Nr de clase poate fi pre-definit ($k$) sau necunoscut
      - Datele dintr-o clasă sunt asemănătoare
    - clasa asociată unei date (noi) de test folosind gruparea învăţată pe datele de antrenament
  - Învăţare supervizată vs. învăţare ne-supervizată



  - Distanţe între 2 elemente p şi q ϵ Rm
    - Euclideana → d(p,q)=sqrt(Σj=1,2,...,m(pj-qj)2)
    - Manhattan → d(p,q)=Σj=1,2,...,m|pj-qj|
    - Mahalanobis → d(p,q)=sqrt(p-q)S-1(p-q)), unde S este matricea de variaţie şi covariaţie (S= E[(p-E[p])(q-E[q])])
    - Produsul intern → d(p,q)=Σj=1,2,...,mpjqj
    - Cosine → d(p,q)=Σj=1,2,...,mpjqj / (sqrt(Σj=1,2,...,mpj2) * sqrt(Σj=1,2,...,mqj2))
    - Hamming → numărul de diferenţe între p şi q
    - Levenshtein → numărul minim de operaţii necesare pentru a-l transforma pe p în q
  - Distanţă vs. Similaritate
    - Distanţa → min
    - Similaritatea → max

# Intelligent systems – Machine Learning (ML)

Învăţare ne-supervizată

- Alte denumiri
  - Clustering
- Procesul → 2 paşi
  - Antrenarea → Învăţarea (determinarea), cu ajutorul unui algoritm, a clusterilor existenţi
  - Testarea → Plasarea unei noi date într-unul din clusterii identificaţi în etapa de antrenament
- Caracteristic
  - Datele nu sunt adnotate (etichetate)
- Tip de probleme
  - Identificara unor grupuri (clusteri)
    - Analiza genelor
    - Procesarea imaginilor
    - Analiza reţelelor sociale
    - Segmentarea pieţei
    - Analiza datelor astronomice
    - Clusteri de calculatoare
  - Reducerea dimensiunii
  - Identificarea unor cauze (explicaţii) ale datelor
  - Modelarea densităţii datelor
- Exemple de probleme
  - Gruparea genelor
  - Studii de piaţă pentru gruparea clienţilor (segmentarea pieţei)
  - news.google.com

# Intelligent systems – Machine Learning (ML)

## Învăţare ne-supervizată

- Calitatea învăţării (validarea clusterizări):
  - Criterii interne → Similaritate ridicată în interiorul unui cluster şi similaritate redusă între clusteri
    - Distanţa în interiorul clusterului
    - Distanţa între clusteri
    - Indexul Davies-Bouldin
    - Indexul Dunn
  - Criteri externe → Folosirea unor benchmark-uri formate din date pre-grupate
    - Compararea cu date cunoscute – în practică este imposibil
    - Precizia
    - Rapelul
    - F-measure

# Învăţare ne-supervizată

- Calitatea învăţării → Criterii interne
  - Distanţa în interiorul clusterului $c_j$ care conţine $n_j$ instanţe
    - Distanţa medie între instanţe (average distance) $D_a (c_j) = \Sigma_{xi1,xi2\epsilon cj} ||x_{i1} - x_{i2}|| / (n_j(n_j-1))$
    - Distanţa între cei mai apropiaţi vecini $D_{nn} (c_j) = \Sigma_{xi1\epsilon cj} min _{xi2\epsilon cj}||x_{i1} - x_{i2}|| / n_j$
    - Distanţa între centroizi $D_c (c_j) = \Sigma_{xi,\epsilon cj} ||x_i - \mu_j|| / n_j$, unde $\mu_j = 1/ n_j\Sigma_{xi\epsilon cj}x_i$
  - Distanţa între 2 clusteri $c_{j1}$ şi $c_{j2}$
    - Legătură simplă $d_s(c_{j1}, c_{j2}) = min _{xi1\epsilon cj1,\ xi2\epsilon cj2} \{||x_{i1} - x_{i2}||\}$
    - Legătură completă $d_{co}(c_{j1}, c_{j2}) = max _{xi1\epsilon cj1,\ xi2\epsilon cj2} \{||x_{i1} - x_{i2}||\}$
    - Legătură medie $d_a(c_{j1}, c_{j2}) = \Sigma _{xi1\epsilon cj1,\ xi2\epsilon cj2} \{||x_{i1} - x_{i2}||\} / (n_{j1} * n_{j2})$
    - Legătură între centroizi $d_{ce}(c_{j1}, c_{j2}) = ||\mu_{j1} - \mu_{j2}||$
  - Indexul Davies-Bouldin → min ➔ clusteri compacţi
    - $DB = 1/nc*\Sigma_{i=1,2,...,nc}max_{j=1,\ 2,\ ...,\ nc,\ j \neq i}((\sigma_i + \sigma_j)/d(\mu_i, \mu_j))$, unde:
      - $nc$ – numărul de clusteri
      - $\mu_i$ – centroidul clusterului $i$
      - $\sigma_i$ – media distanţelor între elementele din clusterul $i$ şi centroidul $\mu_i$
      - $d(\mu_i, \mu_j)$ – distanţa între centroidul $\mu_i$ şi centroidul $\mu_j$
  - Indexul Dunn
    - Identifică clusterii denşi şi bine separaţi
    - $D=d_{min}/d_{max}$, unde:
      - $d_{min}$ – distanţa minimă între 2 obiecte din clusteri diferiţi – distanţa intra-cluster
      - $d_{max}$ – distanţa maximă între 2 obiecte din acelaşi cluster – distanţa inter-cluster

# Intelligent systems – Machine Learning (ML)

## Învăţare ne-supervizată

- Tipologie
  - După modul de formare al clusterilor
    - Ierarhic
      - se crează un arbore taxonomic (dendogramă)
        - crearea clusterilor → recursiv
        - nu se cunoaşte *k* (nr de clusteri)
      - aglomerativ (de jos în sus) → clusteri mici spre clusteri mari
      - diviziv (de sus în jos) → clusteri mari spre clusteri mici
      - Ex. Clustering ierarhic aglomrativ
    - Ne-ierarhic
      - Partiţional → se determină o împărţire a datelor → toţi clusterii deodată
      - Optimizează o funcţie obiectiv definită local (doar pe anumite atribute) sau global (pe toate atributele) care poate fi:
        - Pătratul erorii – suma patratelor distanţelor între date şi centroizii clusterilor → min (ex. K-means)
        - Bazată pe grafuri (ex. Clusterizare bazată pe arborele minim de acoperire)
        - Pe modele probabilistice (ex. Identificarea distribuţiei datelor → Maximizarea aşteptărilor)
        - Pe cel mai apropiat vecin
      - Necesită fixarea apriori a lui k → fixarea clusterilor iniţiali
        - Algoritmii se rulează de mai multe ori cu diferiţi parametri şi se alege versiunea cea mai eficientă
      - Ex. K-means, ACO
    - bazat pe densitatea datelor
      - Densitatea şi conectivitatea datelor
        - Formarea clusterilor de bazează pe densitatea datelor într-o anumită regiune
        - Formarea clusterilor de bazează pe conectivitatea datelor dintr-o anumită regiune
      - Funcţia de densitate a datelor
        - Se încearcă modelarea legii de distribuţie a datelor
      - Avantaj:
        - Modelarea unor clusteri de orice formă
    - Bazat pe un grid
      - Nu e chiar o metodă nouă de lucru
        - Poate fi ierarhic, partiţional sau bazat pe densitate
      - Pp segmentarea spaţiului de date în zone regulate
      - Obiectele se plasează pe un grid multi-dimensional
      - Ex. ACO

# Intelligent systems – Machine Learning (ML)

## Învăţare ne-supervizată

- Tipologie
  - După modul de lucru al algoritmului
    - Aglomerativ
      1. Fiecare instanţă formează iniţial un cluster
      2. Se calculează distanţele între oricare 2 clusteri
      3. Se reunesc cei mai apropiaţi 2 clusteri
      4. Se repetă paşii 2 şi 3 până se ajunge la un singur cluster sau la un alt criteriu de stop
    - Diviziv
      1. Se stabileşte numărul de clusteri (k)
      2. Se iniţializează centrii fiecărui cluster
      3. Se determină o împărţire a datelor
      4. Se recalculează centrii clusterilor
      5. Se reptă pasul 3 şi 4 până partiţionarea nu se mai schimbă (algoritmul a convers)
  - După atributele considerate
    - Monotetic – atributele se consideră pe rând
    - Politetic – atributele se consideră simultan
  - După tipul de apartenenţă al datelor la clusteri
    - Clustering exact (*hard clustering*)
      - Asociază fiecarei intrări $\boldsymbol{x}_i$ o etichetă (clasă) $c_j$

    1. Clustering fuzzy
       1. Asociază fiecarei intrări $\boldsymbol{x}_i$ un grad (probabilitate) de apartenenţă $f_{ij}$ la o anumită clasă $c_j$ ➔ o instanţă $\boldsymbol{x}_i$ poate aparţine mai multor clusteri

# Intelligent systems – Machine Learning (ML)

- Tipologie
  - În funcţie de experienţa acumulată în timpul învăţării
    - SI cu învăţare supervizată
    - SI cu învăţare nesupervizată
    - **SI cu învăţare activă**
    - SI cu învăţare cu întărire
  - În funcţie de modelul învăţat (algoritmul de învăţare)
    - Arbori de decizie
    - Reţele neuronale artificiale
    - Algoritmi evolutivi
    - Maşini cu suport vectorial
    - Modele Markov ascunse

# Intelligent systems – Machine Learning (ML)

- Învăţare activă
  - Algoritmul de învăţare poate primi informaţii suplimentare în timpul învăţării pentru a-şi îmbunătăţi performanţa
    - Ex. pe care din datele de antrenament este mai uşor să se înveţe modelul de decizie

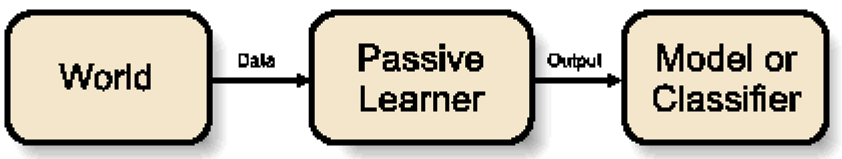



Figure 1.1: General schema for a passive learner.


Figure 1.2: General schema for an active learner.

# Intelligent systems – Machine Learning (ML)

- Tipologie
  - În funcţie de experienţa acumulată în timpul învăţării
    - SI cu învăţare supervizată
    - SI cu învăţare nesupervizată
    - SI cu învăţare activă
    - **SI cu învăţare cu întărire**
  - În funcţie de modelul învăţat (algoritmul de învăţare)
    - Arbori de decizie
    - Reţele neuronale artificiale
    - Algoritmi evolutivi
    - Maşini cu suport vectorial
    - Modele Markov ascunse

# Intelligent systems – Machine Learning (ML)

## Învăţare cu întărire

- Scop
  - Învăţarea, de-a lungul unei perioade, a unui mod de acţiune (comportament) care să maximizeze recompensele (câştigurile) pe termen lung
- Tip de probleme
  - Ex. Dresarea unui câine (good and bad dog)
- Caracteristic
  - Interacţiunea cu mediul (acţiuni → recompense)
  - Secvenţă de decizii
- Învăţare supervizată
  - Decizie → consecinţă (cancer malign sau benign)

# Intelligent systems – Machine Learning (ML)

- Typology
  - Based on algorithm
    - **Decision trees**
    - Artificial Neural Networks
    - Evolutionary algorithms
    - Support Vector Machines
    - Hidden Markov Models

# Intelligent systems – decision trees (DT)

- Decision trees (DTs)
  - Aim
  - Definition
  - Solved problems
  - Example
  - Process
  - Tools
  - Advantages and limits

# Intelligent systems – decision trees (DT)

- Aim
  - Divide a collection of articles in smaller sets by successively applying some decision rules → asking more questions
    - Each question is addressed based on the answer of the previous question
  - Elements are characterized by non-metric information
- Definition
  - Decision tree
    - A special graph → bicolour and oriented tree
    - Contains three node types:
      - Decision nodes → possibilities of decider (a test on an attribute of item that must be classified)
      - Hazard nodes → random events outside the control of decider (exam results, therapy consequences)
      - Result nodes → final states that have a utility or a label
    - Decision and hazard nodes alternate on the tree levels
    - Result nodes → leaf (terminal nodes)
    - (oriented) Edges of the tree consequences of decisions (can be probabilistic)
  - Each internal node corresponds to an attribute
  - Each branch under a node (attribute) corresponds to the value of that attribute
  - Each leaf corresponds to a class

# Intelligent systems – decision trees (DT)

- Problems solved by DTs
  - Problem's instances are represented by a fixed number of attributes, each attribute having a finite number of values
  - Objective function takes discrete values
  - DT represents a disjunction of more conjunctions, each conjunction being "atribute $a_i$ has value $v_j$"
  - Training data could contain errors
  - Training data could be incomplete
    - Some data have not all attributes

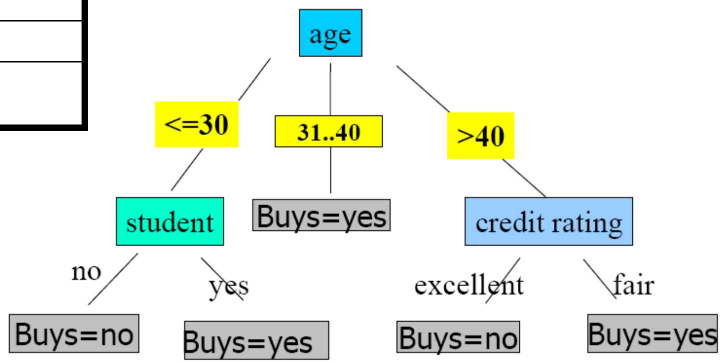
  - Classification problem
    - Binary classification
      - Instances are [(attribute$_{ij}$, value*ij*), class$_i$, i=1,2,...,n, j=1,2,...,m, classi taking 2 values]
    - Multi-class (k-class)
      - Instances are [(attribute$_{ij}$, value$_{ij}$), class$_i$, i=1,2,...,n, j=1,2,...,m, classi taking *k* values]

  - Regression problems
    - DTs are constructed in a similar manner to those of classification problems, but instead to label each node by the label of a class, each node has associated a real value or a function that depends on the inputs of that node
    - Input space is split in decision regions by parallel cuttings to Ox and Oy
    - Discrete outputs are transformed in continuous functions
    - Quality of problem solving
      - Prediction error (square or absolute)

# Intelligent systems – decision trees (DT)

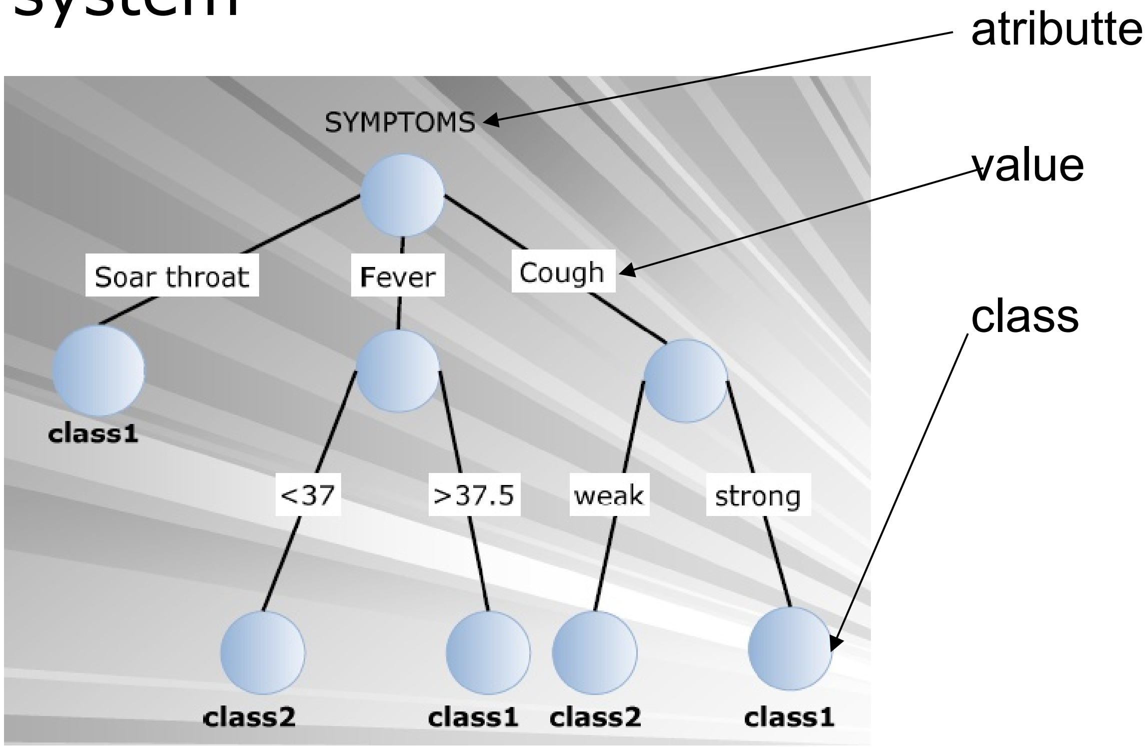
- Example

| rec | Age | Income | Student | Credit_rating | Buys_computer(CLASS) |
|---|---|---|---|---|---|
| r1 | <=30 | High | No | Fair | No |
| r2 | <=30 | High | No | Excellent | No |
| r3 | 31…40 | High | No | Fair | Yes |
| r4 | >40 | Medium | No | Fair | Yes |
| r5 | >40 | Low | Yes | Fair | Yes |
| r6 | >40 | Low | Yes | Excellent | No |
| r7 | 31…40 | Low | Yes | Excellent | Yes |
| r8 | <=30 | Medium | No | Fair | No |
| r9 | <=30 | Low | Yes | Fair | Yes |
| r10 | >40 | Medium | Yes | Fair | Yes |
| r11 | <=30 | Medium | Yes | Excellent | Yes |
| r12 | 31…40 | Medium | No | Excellent | Yes |
| r13 | 31…40 | High | Yes | Fair | Yes |
| r14 | >40 | Medium | No | Excellent | No |



# Intelligent systems – decision trees (DT)

- Example
  - Medical system



# Intelligent systems – decision trees (DT)

- Example
  - Credits

Approved or not

| ID | Age | Has_Job | Own_House | Credit_Rating | Class |
|---|---|---|---|---|---|
| 1 | young | false | false | fair | No |
| 2 | young | false | false | good | No |
| 3 | young | true | false | good | Yes |
| 4 | young | true | true | fair | Yes |
| 5 | young | false | false | fair | No |
| 6 | middle | false | false | fair | No |
| 7 | middle | false | false | good | No |
| 8 | middle | true | true | good | Yes |
| 9 | middle | false | true | excellent | Yes |
| 10 | middle | false | true | excellent | Yes |
| 11 | old | false | true | excellent | Yes |
| 12 | old | false | true | good | Yes |
| 13 | old | true | false | good | Yes |
| 14 | old | true | false | excellent | Yes |
| 15 | old | false | false | fair | No |

# Intelligent systems – decision trees (DT)

- Process
  - Tree construction (induction)
    - Based on training data
    - Works bottom-up or top-down (splitting)
  - Using the tree as a problem solver
    - All decisions performed along a path from the root to a leaf form a rule
    - Rules from DT are used for labeling new data
  - Pruning
    - Identify and move/eliminate branches that reflect noise or exceptions

# Intelligent systems – decision trees (DT)

- Process → Tree construction (induction)
  - Split the training data into subsets based on the characteristics of data
    - A node → Question related to a property
    - Branches of a node → possible answers to the question of the node
    - Initially, all examples are located in the root
      - An attribute gives the root ant its values give the branches
    - On next levels, examples are partitioned based on their attributes → order of attributes
      - For each node, an attribute is (recursively) chosen – its values → branches
    - Splitting → greedy decision making
  - Iterative process
    - Stop conditions
      - All examples from a node belong to the same class → node is a leaf and is labeled by $class_i$
      - There are no examples → node becomes a leaf and is labeled by the majority class of training data
      - There are no attributes

# Intelligent systems – decision trees (DT)

- Process → Tree construction (induction)
  - Example

| rec | Age | Income | Student | Credit_rating | Buys_computer(CLASS) |
|---|---|---|---|---|---|
| r1 | <=30 | High | No | Fair | No |
| r2 | <=30 | High | No | Excellent | No |
| r3 | 31…40 | High | No | Fair | Yes |
| r4 | >40 | Medium | No | Fair | Yes |
| r5 | >40 | Low | Yes | Fair | Yes |
| r6 | >40 | Low | Yes | Excellent | No |
| r7 | 31…40 | Low | Yes | Excellent | Yes |
| r8 | <=30 | Medium | No | Fair | No |
| r9 | <=30 | Low | Yes | Fair | Yes |
| r10 | >40 | Medium | Yes | Fair | Yes |
| r11 | <=30 | Medium | Yes | Excellent | Yes |
| r12 | 31…40 | Medium | No | Excellent | Yes |
| r13 | 31…40 | High | Yes | Fair | Yes |
| r14 | >40 | Medium | No | Excellent | No |

# Intelligent systems – decision trees (DT)

- Process → Tree construction (induction)
  - Example
    - Attribute *age* is selected for the root

age

<=30

| income | student | credit | class |
|---|---|---|---|
| high | no | fair | no |
| high | no | excellent | no |
| medium | no | fair | no |
| low | yes | fair | yes |
| medium | yes | excellent | yes |

>40

| income | student | credit | class |
|---|---|---|---|
| medium | no | fair | yes |
| low | yes | fair | yes |
| low | yes | excellent | no |
| medium | yes | fair | yes |
| medium | no | excellent | no |

31…40

| income | student | credit | class |
|---|---|---|---|
| high | no | fair | yes |
| low | yes | excellent | yes |
| medium | no | excellent | yes |
| high | yes | fair | yes |

age

<=30

| income | student | credit | class |
|---|---|---|---|
| high | no | fair | no |
| high | no | excellent | no |
| medium | no | fair | no |
| low | yes | fair | yes |
| medium | yes | excellent | yes |

>40

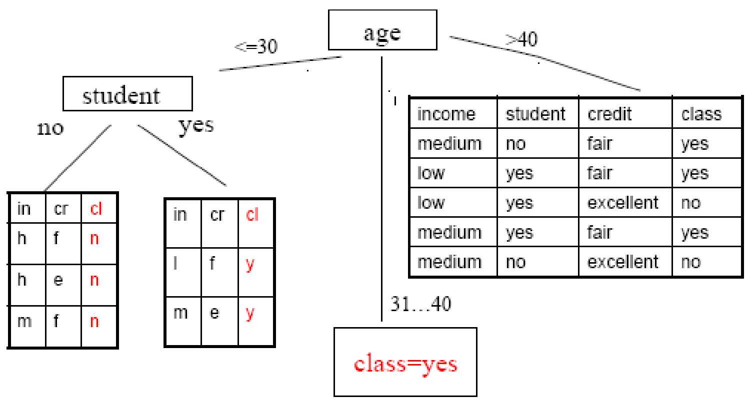
| income | student | credit | class |
|---|---|---|---|
| medium | no | fair | yes |
| low | yes | fair | yes |
| low | yes | excellent | no |
| medium | yes | fair | yes |
| medium | no | excellent | no |

31…40

class=yes

# Intelligent systems – decision trees (DT)

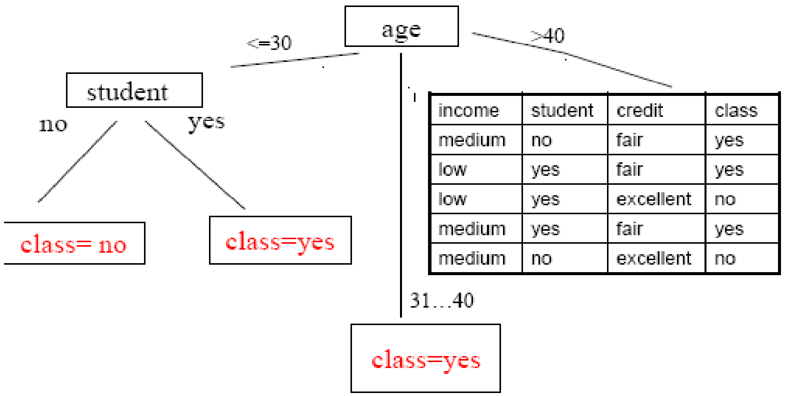
- Process → Tree construction (induction)
  - Example
    - Attribute *age* is selected for the root
    - Attribute *student* is selected on branch *age <=30*




| in | cr | cl |
|---|---|---|
| h | f | n |
| h | e | n |
| m | f | n |

| in | cr | cl |
|---|---|---|
| l | f | y |
| m | e | y |

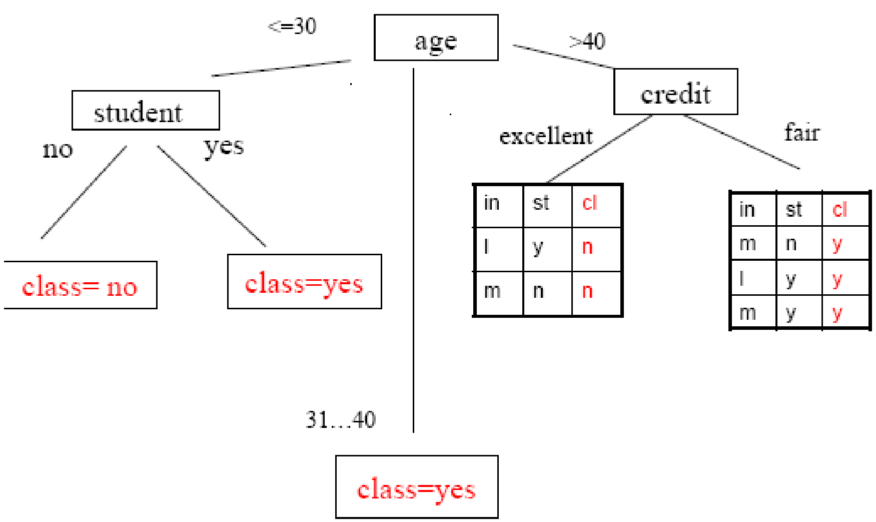
| income | student | credit | class |
|---|---|---|---|
| medium | no | fair | yes |
| low | yes | fair | yes |
| low | yes | excellent | no |
| medium | yes | fair | yes |
| medium | no | excellent | no |

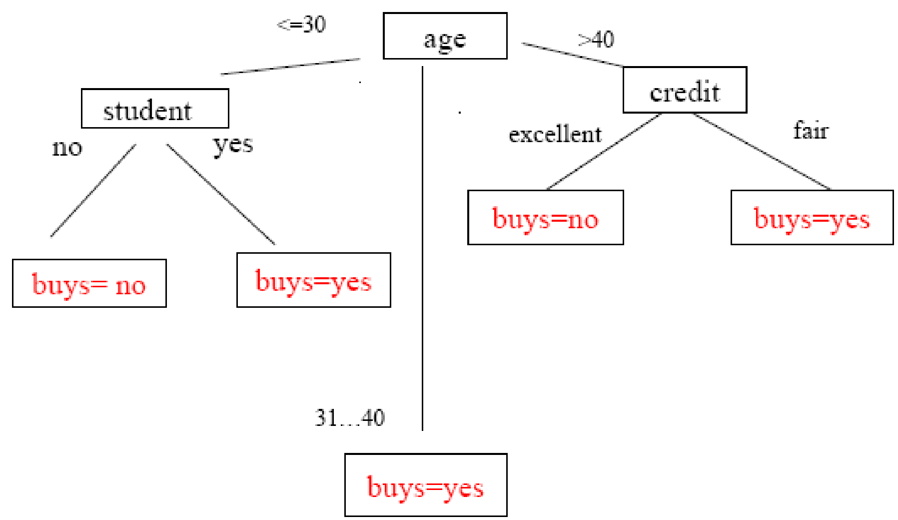



| income | student | credit | class |
|---|---|---|---|
| medium | no | fair | yes |
| low | yes | fair | yes |
| low | yes | excellent | no |
| medium | yes | fair | yes |
| medium | no | excellent | no |

# Intelligent systems – decision trees (DT)

- Process → Tree construction (induction)
  - Example
    - Attribute *age* is selected for the root
    - Attribute *student* is selected on branch *age <=30*
    - Attribute *credit* is selected on branch *age > 40*



# Intelligent systems – decision trees (DT)

- Process → tree construction → ID3/C4.5 algorithm
  - Greedy, recursive, top-down, divide-and-conquer

```
generate(D, A){              //D – a partitioning of training data, A – list of attributes
      create a new node N
      if examples from D belong to a single class C then
              node N becomes a leaf and is labeled by C
              return node N
      else
              if A=∅ then
                 node N becomes a leaf and is labeled by majority class of D
                 return node N
              else
                 separation_attribute = AttributeSelection(D, A)
                 label node N by separation_attribute
                 for all possible values vj of separation_attribute
                          let Dj – set of examples from D that have separation_attribute=vj
                          if Dj =∅ then
                                          add a leaf (to node N) labeled by majority class of D
                          else
                              add a node (to node N) return by generate(Dj, A–separation_attribute)
                return node N
}
```

# Intelligent systems – decision trees (DT)

- Process → tree construction → ID3/C4.5 algorithm
  - AttributeSelection(D,A) → select the attribute that corresponds to a node (root or internal node)
    - Random
    - Attribute with the fewest/most values
    - Based on a pre-established order
      - Information gain
      - Gain rate
      - Gini index
      - Distance between partitions created by the attribute

# Intelligent systems – decision trees (DT)

- Process → tree construction → ID3/C4.5 algorithm → Attribute Selection
  - Information gain
    - An impurity measure
      - 0 (minim) – if all examples belong to the same class
      - 1 (maxim) – if examples are uniform distributed over classes
    - Based on data entropy
      - Expected number of bits required by coding the class of an element from data
      - Binary classification (2 classes): $E(S) = - p_+ \log_2 p_+ - p_- \log_2 p_-$ where
        - $p_+$ - proportion of positive examples in dataset S
        - $p_-$ - proportion of negative examples in dataset S
      - Multi-class classification: $E(S) = \sum_{i=1, 2, \ldots, k} - p_i \log_2 p_i$ – data entropy related to target attribute (output attribute), where
        - $p_i$ – proportion of examples from class *i* in dataset S
    - Information gain of an attribute
      - How the elimination of attribute *a* reduces the dataset's entropy
      - $Gain(S, a) = E(S) - \sum_{v \in valori(a)} |S_v| / |S| \, E(S_v)$
      - $\sum_{v \in values(a)} |S_v| / |S| \, E(S_v)$ – expected information

# Intelligent systems – decision trees (DT)

Process → tree construction → ID3/C4.5 algorithm → Attribute Selection

- Information gain
  - Example

| | a1 | a2 | a3 | Clasa |
|---|---|---|---|---|
| d1 | mare | roşu | cerc | clasa 1 |
| d2 | mic | roşu | pătrat | clasa 2 |
| d3 | mic | roşu | cerc | clasa 1 |
| d4 | mare | albastru | cerc | clasa 2 |

$S = \{d1, d2, d3, d4\} \Rightarrow p_+ = 2 / 4,\ p_- = 2 / 4 \Rightarrow E(S) = - p_+ \log_2 p_+ - p_- \log_2 p_- = 1$

$S_{v=mare} = \{d1, d4\} \Rightarrow p_+ = ½,\ p_- = ½ \Rightarrow E(S_{v=mare}) = 1$

$S_{v=mic} = \{d2, d3\} \Rightarrow p_+ = ½,\ p_- = ½ \Rightarrow E(S_{v=mic}) = 1$

$S_{v=rosu} = \{d1, d2, d3\} \Rightarrow p+ = 2/3,\ p- = 1/3 \Rightarrow E(S_{v=rosu}) = 0.923$

$S_{v=albastru} = \{d4\} \Rightarrow p+ = 0,\ p- = 1 \Rightarrow E(S_{v=albastru}) = 0$

$S_{v=cerc} = \{d1, d3, d4\} \Rightarrow p+ = 2/3,\ p- = 1/3 \Rightarrow E(S_{v=cerc}) = 0.923$

$S_{v=patrat} = \{d2\} \Rightarrow p+ = 0,\ p- = 1 \Rightarrow E(S_{v=patrat}) = 0$

$Gain(S, a) = E(S) - \sum_{v \in values(a)} |S_v| / |S|\ E(S_v)$

$Gain(S, a_1) = 1 - (|S_{v=mare}| / |S|\ E(S_{v=mare}) + |S_{v=mic}| / |S|\ E(S_{v=mic})) = 1 - (2/4 * 1 + 2/4 * 1) = 0$

$Gain(S, a_2) = 1 - (|S_{v=rosu}| / |S|\ E(S_{v=rosu}) + |S_{v=albastru}| / |S|\ E(S_{v=albastru})) = 1 - (3/4 * 0.923 + 1/4 * 0) = 0.307$

$Gain(S, a_3) = 1 - (|S_{v=cerc}| / |S|\ E(S_{v=cerc}) + |S_{v=patrat}| / |S|\ E(S_{v=patrat})) = 1 - (3/4 * 0.923 + 1/4 * 0) = 0.307$

# Intelligent systems – decision trees (DT)

- Process → tree construction → ID3/C4.5 algorithm → Attribute Selection
  - Gain rate
    - Penalises an attribute by integrating a new term – *split information* – that depends on spreading degree and on uniformity degree of separation
      - *Split information* – entropy related to possible values of attribute a
      - Sv – proportion of examples from dataset S that have attribute a with value v
      - *splitInformation(S,a)=* $-\sum_{v=value(a)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$

# Intelligent systems – decision trees (DT)

- Process
  - Tree construction
  - Using the tree as a problem solver
    - Main idea
      - Extract the rules from the constructed tree
        - IF *age* = *"<=30"* AND *student* = *"no"* THEN *buys_computer* = *"no"*
        - IF *age* = *"<=30"* AND *student* = *"yes"* THEN *buys_computer* = *"yes"*
        - IF *age* = *"31…40"* THEN *buys_computer* = *"yes"*
        - IF *age* = *">40"* AND *credit_rating* = *"excellent"* THEN *buys_computer* = *"no"*
        - IF *age* = *">40"* AND *credit_rating* = *"fair"* THEN *buys_computer* = *"yes"*
      - Use the rules for classifying the test data (new data)
        - Let x a data without class → rules can be written as predicates
        - IF *age( x, <=30)* AND *student(x, no)* THEN *buys_computer (x, no)*
        - IF *age(x, <=30)* AND *student (x, yes)* THEN *buys_computer (x, yes)*

# Intelligent systems – decision trees (DT)

- Process
  - Tree construction
  - Using the tree as a problem solver
    - Difficulties
      - *Underfitting* → DT constructed on training data is to simple → large classification error during training and testing
      - *Overfitting* → DT constructed on training data match the training data, but it ca not generalize new data
    - Solutions
      - Pruning → remove some branches (un - useful, redundant) → small tree
      - Cross-validation

# Intelligent systems – decision trees (DT)

- Process
  - Tree construction
  - Using the tree as a problem solver
  - Pruning
    - Why?
      - After the DT is constructed, classification rules are extracted in order to represent the knowledge as if-then rules (easy to understand)
      - A rule is create by traversing the DT from root to a leaf
      - Each pair (attribute, value) – (node, edge) – is a conjunction in the premise of the rule (if part), except the last node of the path that is a leaf and represents the consequence (output, then part) of the rule
    - Typology
      - *pre-pruning*
        - Increasing the tree is stopped during construction by stopping the division of nodes that become leaf labeled by majority class of examples from that node
      - *post-pruning*
        - After the DT is constructed, eliminate the branches of some nodes that become leaf → classification error reduces (on testing data)

# Intelligent systems – decision trees (DT)

- Tools
  - http://webdocs.cs.ualberta.ca/~aixplore/learning/D
  - WEKA → J48
  - http://id3alg.altervista.org/
  - http://www.rulequest.com/Personal/c4.5r8.tar.gz

- Biblio
  - http://www.public.asu.edu/~kirkwood/DAStuff/decisiontrees/index.html

# Intelligent systems – decision trees (DT)

- Advantages
  - Easy to understand and interpret
  - Can use nominal or categorized data
  - Decision logic can be easy followed (rules are visible)
  - Works better with large data

- Disadvantages
  - Instability → change the training data
  - Complexity → representation
  - Difficult to use
  - The DT construction is expensive
  - The DT construction requires a lot of information

# Intelligent systems – decision trees (DT)

- Difficulties
  - There can be more trees
    - To small
    - With a better accuracy (easy to be read and with good performances)
    - Identify the best tree → NP-problem
  - Select the best tree
    - Heuristic algorithms
    - ID3 → the smallest tree
      - Occam theorem: "always choose the simplest explanation"
  - Continuous attributes
    - Range splitting
      - How many intervals?
      - How large intervals?
  - To large trees
    - Pre - pruning → stops to construct the tree earlier
    - Post-pruning → remove some branches

# Review

- Automatic learning systems

  - Machine Learning – ML
    - Supervised learning → annotated train data (by label from a predefined set) and test data have to be annotated by using the learnt model (by one of the known labels)
    - Unsupervised learning → not-annotated train data; a labeling model has to be learnt in order to annotate the test data; the set of labels is unknown befor training

  - Systems
    - Decision trees
      - Each internal node → attribute
      - Each branch of a node (attribute) → value of that attribute
      - Each leaf → class (label) – contains all data from that class

# Next lecture

A. Short introduction in Artificial Intelligence (AI)

A. Solving search problems
   A. Definition of search problems
   B. Search strategies
      A. Uninformed search strategies
      B. Informed search strategies
      C. Local search strategies (Hill Climbing, Simulated Annealing, Tabu Search, Evolutionary algorithms, PSO, ACO)
      D. Adversarial search strategies

C. **Intelligent systems**
   A. Rule-based systems in certain environments
   B. Rule-based systems in uncertain environments (Bayes, Fuzzy)
   C. **Learning systems**
      A. **Decision Trees**
      B. **Artificial Neural Networks**
      C. Support Vector Machines
      D. Evolutionary algorithms
   D. Hybrid systems

# Next lecture – useful information

- Chapter VI (19) of *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*

- Chapter 8 of *Adrian A. Hopgood, Intelligent Systems for Engineers and Scientists, CRC Press, 2001*

- Chapters 12 and 13 of *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*

- Chapter V of *D. J. C. MacKey, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003*

- Chapter 4 of *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*

- Presented information have been inspired from different bibliographic sources, but also from past AI lectures taught by:
  - PhD. Assoc. Prof. Mihai Oltean – www.cs.ubbcluj.ro/~moltean
  - PhD. Assoc. Prof. Crina Groşan - www.cs.ubbcluj.ro/~cgrosan
  - PhD. Prof. Horia F. Pop - www.cs.ubbcluj.ro/~hfpop