INTELIGENȚĂ ARTIFICIALĂ

Sisteme inteligente

Sisteme care învaţă singure

– kNN si programare genetică –

Laura Dioşan

Sumar

A. Scurtă introducere în Inteligenţa Artificială (IA)

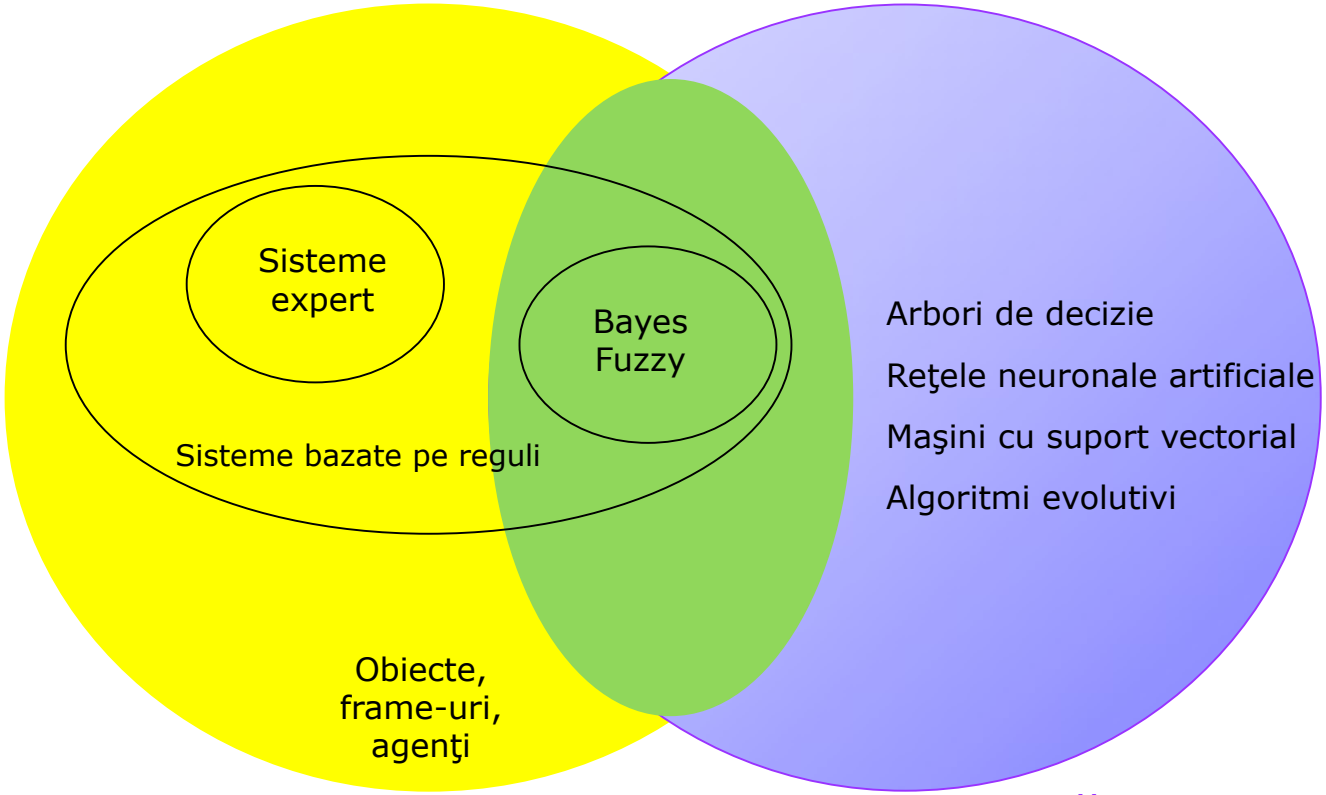
B. Rezolvarea problemelor prin căutare
- Definirea problemelor de căutare
- Strategii de căutare
  - Strategii de căutare neinformate
  - Strategii de căutare informate
  - Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
  - Strategii de căutare adversială

C. Sisteme inteligente
- Sisteme care învaţă singure
  - Arbori de decizie
  - Reţele neuronale artificiale
  - kNN
  - Algoritmi evolutivi
  - Maşini cu suport vectorial
- Sisteme bazate pe reguli
- Sisteme hibride

Materiale de citit şi legături utile

- capitolul 15 din *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- Capitolul 9 din *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*
- capitolul VI (18) din *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*
- capitolul 10 şi 11 din *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- capitolul V din *D. J. C. MacKey, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003*
- capitolul 3 din *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*

Sisteme inteligente



Sisteme inteligente – SIS – Învățare automată

- 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):
    - Reţele neuronale artificiale
    - **Maşini cu suport vectorial (MSV)**
    - **Algoritmi evolutivi**
    - **kNN**
    - Arbori de decizie
    - Modele Markov ascunse

Materiale de citit şi legături utile

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

Sisteme inteligente – SIS – Învățare automată

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
  - **Maşini cu suport vectorial (MSV)**
  - Algoritmi evolutivi
  - Modele Markov ascunse

Sisteme inteligente – SIS – MSV

- Maşini cu suport vectorial (MSV)
  - Definire
  - Tipuri de probleme rezolvabile
  - Avantaje
  - Dificultăţi
  - Tool-uri

Sisteme inteligente – SIS – MSV

- Definire
  - Dezvoltate de Vapnik în 1970
  - Popularizate după 1992
  - Clasificatori liniari care identifică un hiperplan de separare a clasei pozitive de clasa negativă
  - Au o fundamentare teoretică foarte riguroasă
  - Funcţionează foarte bine pentru date de volum mare (analiza textelor, analiza imaginilor)

  - **Reamintim**
    - Problemă de învăţare supervizată în care avem un set de date de forma:
      - (x^d, t^d), cu:
      - $x^d \in \mathbf{R}^m \rightarrow x^d=(x^d_1, x^d_2,..., x^d_m)$
      - $t^d \in \mathbf{R} \rightarrow t^d \in \{1, -1\}$, 1 → clasă pozitivă, -1 → clasă negativă
      - cu d = 1,2,...,n,n+1,n+2,...,N
    - Primele n date (se cunosc x^d şi t^d) vor fi folosite drept bază de antrenament a MSV
    - Ultimele N-n date (se cunosc doar x^d, fără t^d) vor fi folosite drept bază de testare a MSV

Sisteme inteligente – SIS – MSV

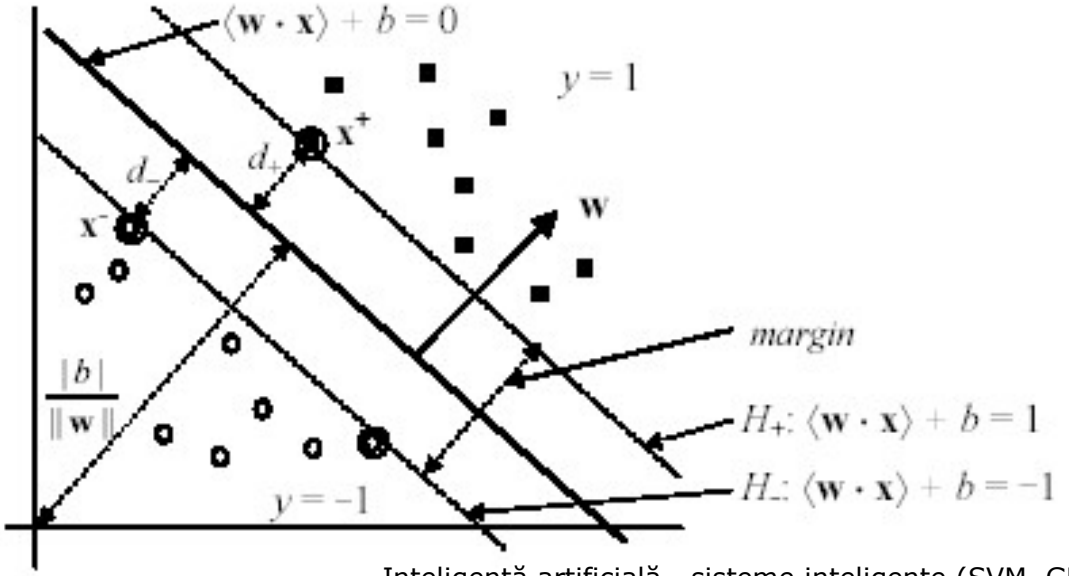
- Definire
  - MSV găseşte o funcţie liniară de forma $f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$, ($\mathbf{w}$ -vector pondere) a.î.

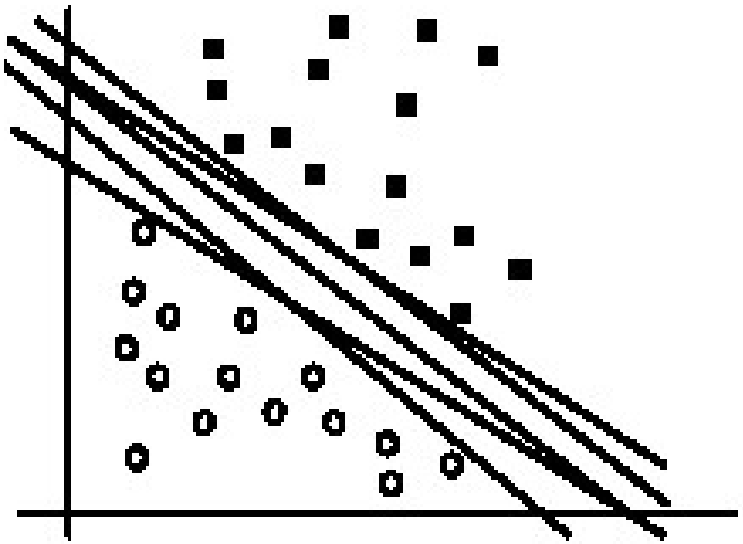
$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

  - $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ → hiperplanul de decizie care separă cele 2 clase

Sisteme inteligente – SIS – MSV

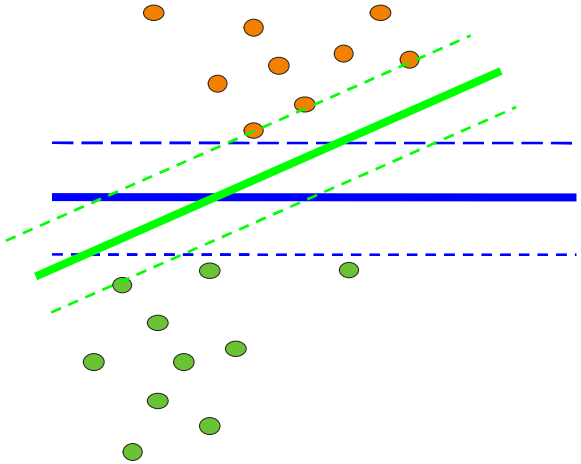
- Definire
  - Pot exista mai multe hiperplane
    - Care este cel mai bun hiperplan?
  - MSV caută hiperplanul cu cea mai largă margine (cel care micşorează eroarea de generalizare)
    - Algoritmul SMO (*Sequential minimal optimization*)



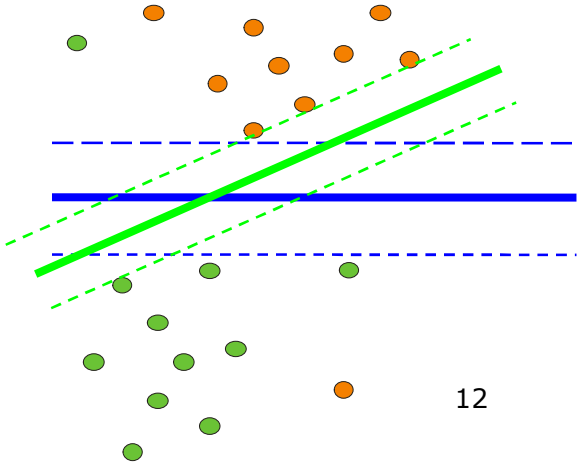


Sisteme inteligente – SIS – MSV

- Tipuri de probleme rezolvabile
  - Probleme de clasificare → Cazuri de date
    - Liniar separabile
      - Separabile
        - Eroarea = 0



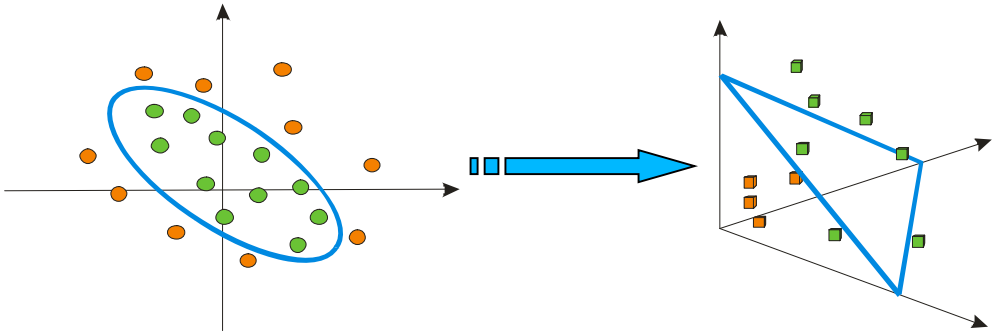
      - Ne-separabile
        - Se relaxează constrângerile → se permit unele erori
        - C – coeficient de penalizare



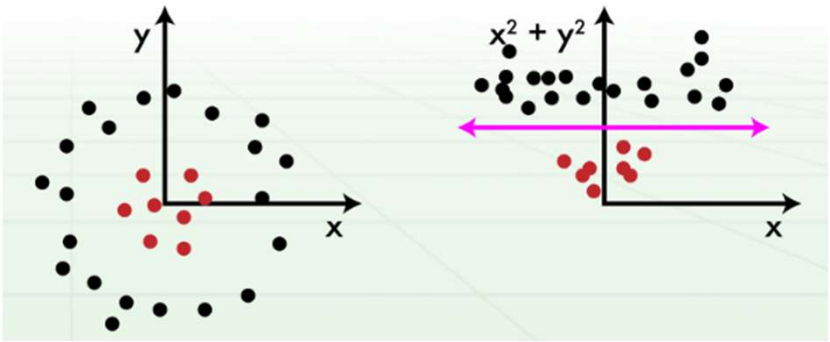
Sisteme inteligente – SIS – MSV

Cazuri de date

- Non-liniar separabile
  - Spaţiul de intrare se transformă într-un spaţiu cu mai multe dimensiuni (*feature space*), cu ajutorul unei funcţii kernel, unde datele devin liniar separabile



  - Kernele posibile
    - Clasice
      - Polynomial kernel: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle + 1)^d$
      - RBF kernel: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = exp(- \sigma |\boldsymbol{x}_1 - \boldsymbol{x}_2|^2)$
    - Kernele multiple
      - Liniare: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum w_i K_i$
      - Ne-liniare
        - Fără coeficienţi: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_1 + K_2 * exp(K_3)$
        - Cu coeficienţi: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_1 + c_1 * K_2 * exp(c_2 + K_3)$

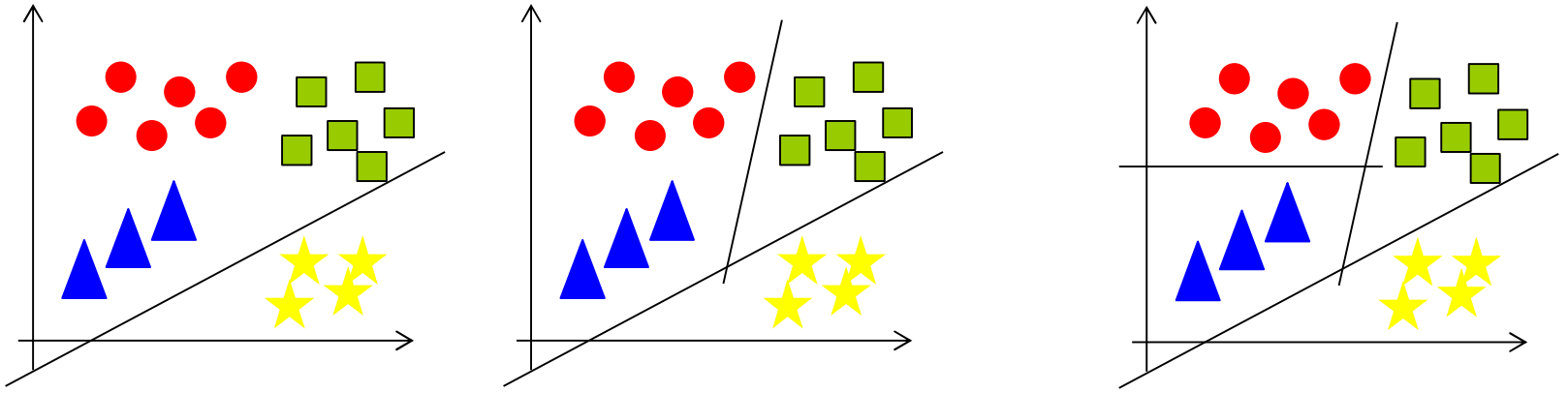


Sisteme inteligente – SIS – MSV

- Configurarea MSV
  - Parametrii unei MSV
    - Coeficientul de penalizare C
      - C – mic → convergenţă lentă
      - C – mare → convergenţă rapidă
    - Parametrii funcţiei kernel (care kernel şi cu ce parametri)
      - Dacă m (nr de atribute) este mult mai mare decât n (nr de instanţe)
        - MSV cu kernel liniar (MSV fără kernel) → $K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = \boldsymbol{x}^{d1} \cdot \boldsymbol{x}^{d2}$
      - Dacă m (nr de atribute) este mare, iar n (nr de instanţe) este mediu
        - MSV cu kernel Gaussian $K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = \exp(-||\boldsymbol{x}^{d1} - \boldsymbol{x}^{d2}||^2/2\sigma^2)$
        - σ – dispersia datelor de antrenament
        - Atributele instanţelor trebuie normalizate (scalate la (0,1))
      - m (nr de atribute) este mic, iar n (nr de instanţe) este mare
        - Se adaugă noi atribute, iar apoi
        - MSV cu kernel liniar

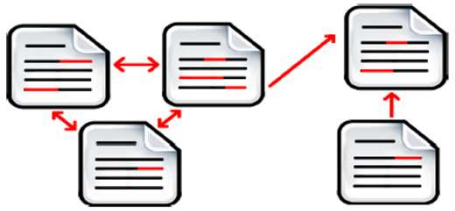
Sisteme inteligente – SIS – MSV

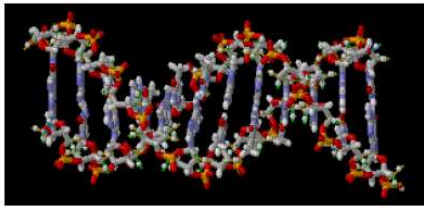
- MSV pentru probleme de clasificare supervizate cu mai mult de 2 clase
  - Una vs. restul (one *vs.* all)

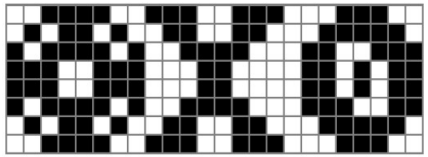


Sisteme inteligente – SIS – MSV

- MSV structurate
  - Învăţare automată
    - Normală f: $\mathcal{X}$ → $\mathbf{R}$
      - Intrări de orice fel
      - Ieşiri numerice (naturale, întregi, reale)
    - Structurată: $\mathcal{X}$ → $\mathcal{Y}$
      - Intrări de orice fel
      - Ieşiri de orice fel (simple sau structurate)
  - Informaţii structurate
    - Texte şi hiper-texte
    - Molecule şi structuri moleculare
    - Imagini







Sisteme inteligente – SIS – MSV

- MSV structurate
  - Aplicaţii
    - Procesarea limbajului natural
      - Traduceri automate (ieşiri → propoziţii)
      - Analiza sintactică şi/sau morfologică a propoziţiilor (ieşiri → arborele sintactic şi/sau morfologic)
    - Bioinformatică
      - Predicţia unor structuri secundare (ieşirile → grafe bi-partite)
      - Predicţia funcţionării unor enzime (ieşirile → *path*-uri în arbori)
    - Procesarea vorbirii
      - Transcrieri automate (ieşiri → propoziţii)
      - Transformarea textelor în voce (ieşiri → semnale audio)
    - Robotică
      - Planificare (ieşirile → secvenţe de acţiuni)

Sisteme inteligente – SIS – MSV

- Avantaje
  - Pot lucra cu orice fel de date (liniar separabile sau nu, distribuit uniform sau nu, cu distribuţie cunoscută sau nu)
    - Funcţiile kernel care crează noi atribute (features) → straturile ascunse dintr-o RNA
  - Dacă problema e convexă oferă o soluţie unică → optimul global
    - RNA pot asocia mai multe soluţii → optime locale
  - Selectează automat mărimea modelului învăţat (prin vectorii suport)
    - În RNA straturile ascunse trebuie configurate de către utilizator *apriori*
  - Nu învaţă pe derost datele (overfitting)
    - RNA se confruntă cu problema overfitting-ului chiar şi cand modelul se învaţă prin validare încrucişată
- Dificultăţi
  - Doar atribute reale
  - Doar clasificare binară
  - Background matematic dificil
- Tool-uri
  - LibSVM → http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Weka → SMO
  - SVMLight → http://svmlight.joachims.org/
  - SVMTorch → http://www.torch.ch/
  - http://www.support-vector-machines.org/

Sisteme inteligente – SIS – Învățare automată

- Programare genetică
  - Definire
  - Proiectare
  - Avantaje
  - Limite
  - Versiuni

Sisteme inteligente – SIS – PG

Reamintim

- Învăţare supervizată → problemă de regresie (Studiul legăturii între variabile)
  - Se dă un set de *n* 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)
      - $\boldsymbol{atribute\_data_i} = (atr_{i1}, atr_{i2}, ..., atr_{im})$, m – nr atributelor (caracteristicilor, proprietăţilor) unei date
      - $ieşire_i$ – un număr real
    - date de test
      - sub forma ($\boldsymbol{atribute\_data_i}$), $i = n+1,N$ (N-n = nr datelor de test)
  - Să se determine
    - o funcţie (necunoscută) care realizează corespondenţa atribute – ieşire pe datele de antrenament
    - Ieşirea (valoarea) asociată unei date (noi) de test folosind funcţia învăţată pe datele de antrenament
- Cum găsim forma (expresia) funcţiei?
  - Algoritmi evolutivi → Programare genetică

Sisteme inteligente – SIS – PG

Reamintim

- Algoritmi evolutivi
  - Inspiraţi din natură (biologie)
  - Iterativi
  - Bazaţi pe
    - populaţii de potenţiale soluţii
    - căutare aleatoare ghidată de
      - Operaţii de selecţie naturală
      - Operaţii de încrucişare şi mutaţie
  - Care procesează în paralel mai multe soluţii
- Metafora evolutivă

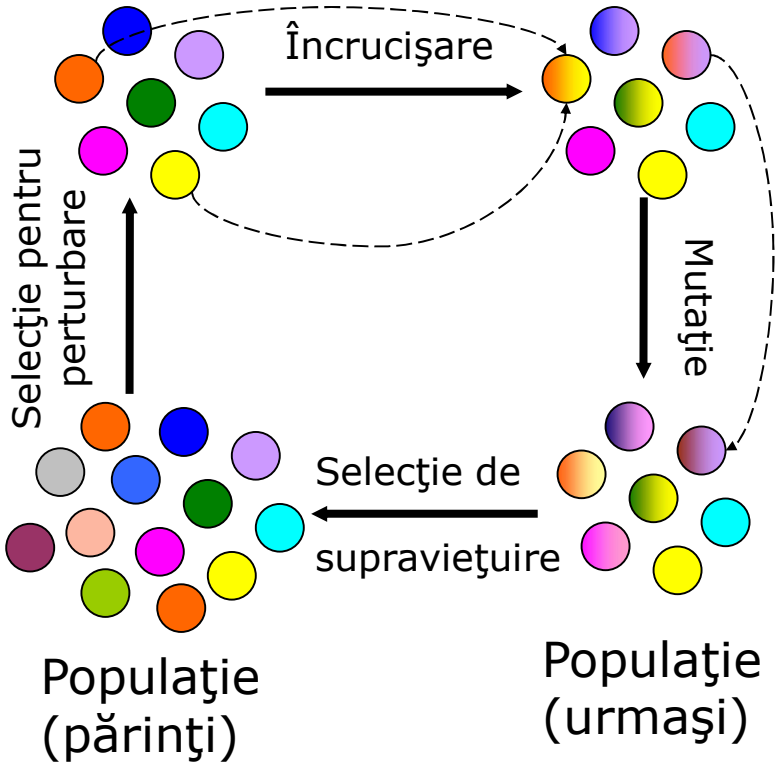
Evoluţie naturală	Rezolvarea problemelor
Individ	Soluţie potenţială (candidat)
Populaţie	Mulţime de soluţii
Cromozom	Codarea (reprezentarea) unei soluţii
Genă	Parte a reprezentării
Fitness (măsură de adaptare)	Calitate
Încrucişarea şi mutaţie	Operatori de căutare
Mediu	Spaţiul de căutare al problemei

Sisteme inteligente – SIS – PG

Reamintim

- Algoritmi evolutivi

```
Initializare populaţie P(0)
Evaluare P(0)
g := 0; //generaţia
CâtTimp (not condiţie_stop) execută
    Repetă
       Selectează 2 părinţi p1 şi p2 din P(g)
       Încrucişare(p1,p2) =>o1 şi o2
       Mutaţie(o1) => o1*
       Mutaţie(o2) => o2*
       Evaluare(o1*)
       Evaluare(o2*)
       adăugare o1* şi o* în P(g+1)
    Până când P(g+1) este completă
    g := g + 1
Sf CâtTimp
```

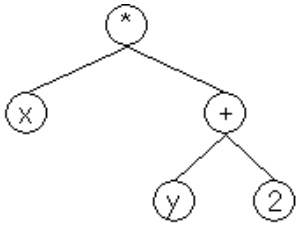


Sisteme inteligente – SIS – PG

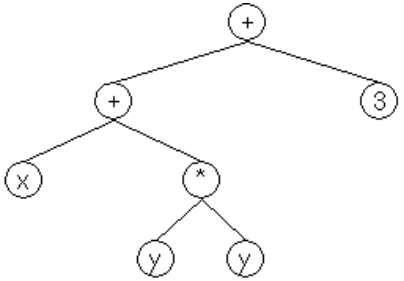
- Definire
  - Propusă de john Koza în 1988
  - http://www.genetic-programming.org/
  - Un tip particular de algoritmi evolutivi
  - Cromozomi
    - sub formă de arbore care codează mici programe
  - Fitness-ul unui cromozom
    - Performanţa programului codat în el
  - Scopul PG
    - Evoluarea de programe de calculator
    - AG evoluează doar soluţii pentru probleme particulare

Sisteme inteligente – SIS – PG

- Proiectare
  - Reprezentarea cromozomilor
    - Foarte importantă, dar este o sarcină dificilă
    - Cromozomul = un arbore cu noduri de tip
      - Funcţie → operatori matematici (+,-,*,/,*sin*,*log*, *if*,...)
      - Terminal → atribute ale datelor problemei sau constante (*x,y,z,a,b,c,...*)
    - care codează expresia matematică a unui program (problema regresiei → a unei funcţii)



$x(y+2)$



$x+y^2+3$

Sisteme inteligente – SIS – PG

- Proiectare
  - Fitness
    - Eroarea de predicţie – diferenţa între ceea ce dorim să obţinem şi ceea ce obţinem de fapt
    - pp o problemă de regresie cu următoarele date de intrare (2 atribute şi o ieşire) şi 2 cromozomi:
      - $c_1 = 3x_1-x_2+5 = (((3, x1, *), x2, -), 5, +)$
      - $c_2 = 3x_1+2x_2+2$ $f^*(x1,x2)=3x_1+2x_2+1$ – necunoscută

x_1	x_2	$f^*(x_1,x_2)$	$f_1(x_1,x_2)$	$f_2(x_1,x_2)$	$\lvert f^*-f_1\rvert$	$\lvert f^*-f_2\rvert$
1	1	6	7	7	1	1
0	1	3	4	4	1	1
1	0	4	8	5	4	1
-1	1	0	1	1	1	1
					∑=7	∑= 4

→ c_2 e mai bun ca c_1

Sisteme inteligente – SIS – PG

- Proiectare
  - Fitness
    - Eroarea de predicţie – diferenţa între ceea ce dorim să obţinem şi ceea ce obţinem de fapt
    - pp o problemă de clasificare cu următoarele date de intrare (2 atribute şi o ieşire) şi 2 cromozomi:
      - $c_1 = 3x_1-x_2+5$ *f1(1,1) = 7 -: sigm(7) – (0,1) -: Theta*
      - $c_2 = 3x_1+2x_2+2$

x_1	x_2	$f^*(x_1,x_2)$	$f_1(x_1,x_2)$	$f_2(x_1,x_2)$	$\|f^*-f_1\|$	$\|f^*-f_2\|$
1	1	Yes	Yes	Yes	0	0
0	1	No	Yes	No	1	0
1	0	Yes	No	No	1	1
-1	1	Yes	No	yes	1	0
					∑=3	∑= 1

➔ c_2 e mai bun ca c_1

Sisteme inteligente – SIS – PG

- Proiectare
  - Iniţializarea cromozomilor
    - Generare aleatoare de arbori corecţi → programe valide (expresii matematice valide)
    - Se stabileşte o adâncime maximă a arborilor D_{max}
    - 3 metode de iniţializare
      - *Full* → fiecare ramură a rădăcinii are adâncimea D_{max}
        - Nodurile aflate la o adâncime d < Dmax se iniţializează cu una dintre funcţiile din F
        - Nodurile aflate la o adâncime d = Dmax se iniţializează cu unul dintre terminalele din T
      - *Grow* → fiecare ramură a rîdăcinii are o adâncime < D_{max}
        - Nodurile aflate la o adâncime d < Dmax se iniţializează cu un element din $F \cup T$
        - Nodurile aflate la o adâncime d = Dmax se iniţializează cu unul dintre terminalele din T
      - *Ramped half and half* → ½ din populaţia de cromozomi se iniţializează folosind metoda *full*, ½ din populaţia de cromozomi se iniţializează folosind metoda *grow*

Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Selecţia pentru recombinare
    - similar oricărui algoritm evolutiv
    - recomandare → selecţie proporţională
    - over-selection → pentru populaţii f mari
      - Se ordonează populaţia pe baza fitness-ului şi se împarte în 2 grupuri:
        - Grupul 1: cei mai buni x% cromozomi din populaţie
        - Grupul 2: restul de (100-x)% cromozomi din populaţie
        - Pentru populaţii cu 1000, 2000, 4000, 8000 de cromozomi, x este stabilit la 32%, 16%, 8%, respectiv 4%
      - 80% din operaţiile de selecţie vor alege cromozomi din grupul 1, 20% din grupul 2

Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Selecţia de supravieţuire
    - Scheme
      - Generaţională
      - steady-state
    - Probleme
      - *Bloat* → supravieţuirea celui mai "gras" individ (dimensiunea cromozomilor creşte de-a lungul evoluţiei)
      - Soluţii
        - Interzicerea operatorilor de variaţie care produc descendenţi prea mari
        - Presiunea economiei (zgârceniei) – penalizarea cromozomilor prea mari

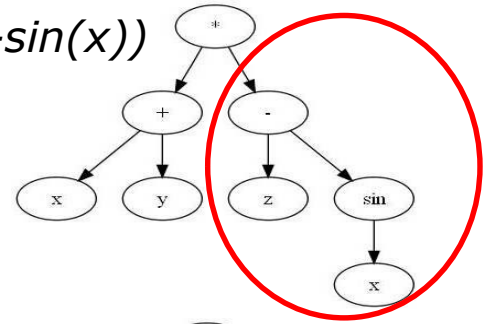
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Încrucişare şi mutaţie
    - Parametri
      - O probabilitate *p* de alegere între încrucişare şi mutaţie
        - p = 0 (cf. Koza) sau p = 0.05 (cf. Banzhaf)
      - O probabilitate p_c şi respectiv p_m de stabilire a nodului care urmează a fi supus modificării
    - Dimensiunea descendenţilor diferă de dimensiune părinţilor

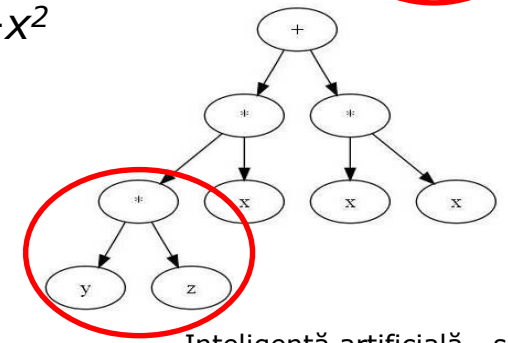
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Încrucişare
    - Cu punct de tăietură – se interchimbă doi sub-arbori
    - Punctul de tăietură se generează aleator

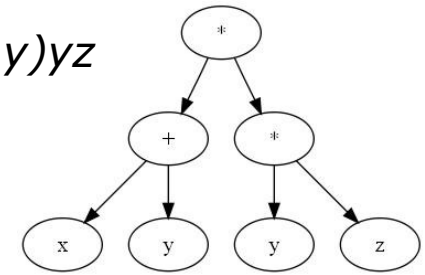
$p_1=(x+y)*(z-sin(x))$



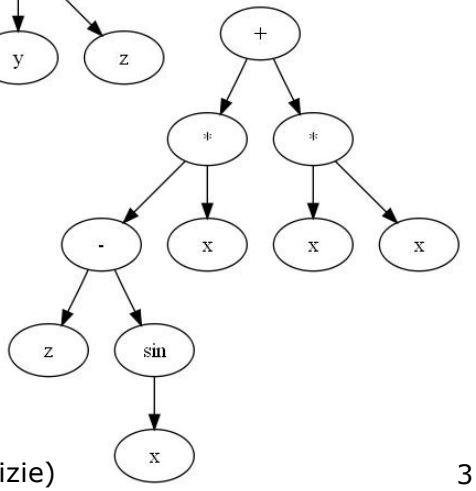
$p_2=xyz+x^2$



$f_1=(x+y)yz$



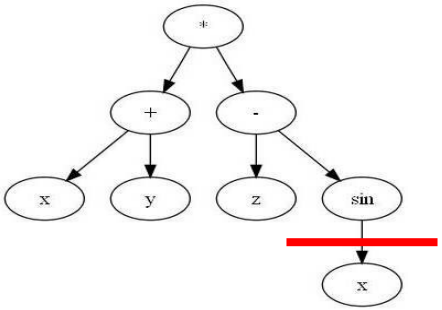
$f_2=(z-sin(x))x+x^2$



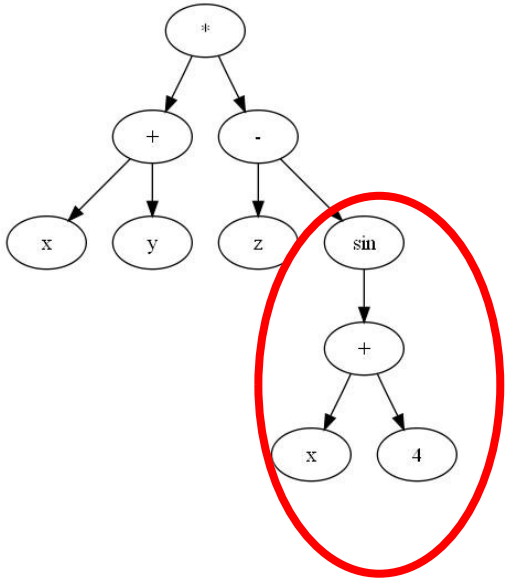
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *grow* → Înlocuirea unei frunze cu un nou sub-arbore

p=(x+y)(z-sin(x))*



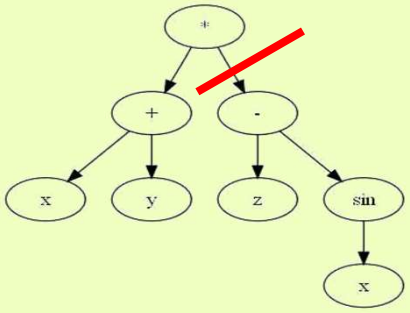
f=(x+y)(z-sin(x+4))*



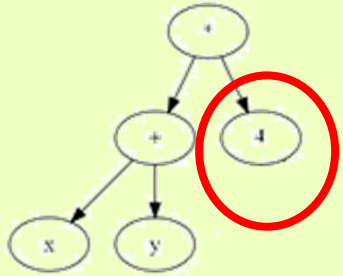
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *shrink* → Înlocuirea unui sub-arbore cu o frunză

p=(x+y)(z-sin(x))*



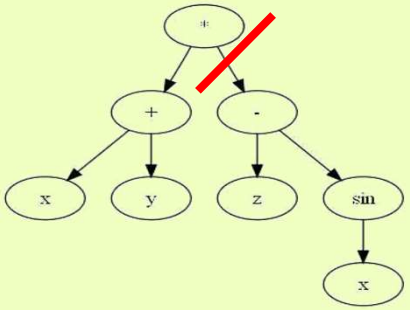
*f=(x+y)*4*



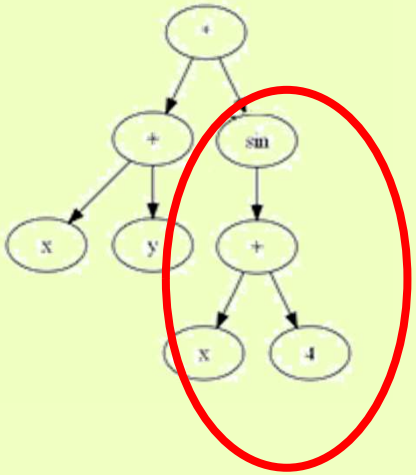
Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *Koza* → Înlocuirea unui nod (intern sau frunză) cu un nou sub-arbore

p=(x+y)(z-sin(x))*



*f=(x+y)*sin(x+4)*



Sisteme inteligente – SIS – PG

- Proiectare
  - Operatori genetici → Mutaţie
    - Mutaţie de tip *switch*
      - selectarea unui nod intern şi re-ordonarea sub-arborilor săi
    - Mutaţie de tip cycle
      - selectarea unui nod şi înlocuirea lui cu unul nou de acelaşi tip (intern – cu o funcţie – sau frunză – cu un terminal)

Sisteme inteligente – SIS – PG

- Comparaţie AG şi PG
  - Forma cromozomilor
    - AG – cromozomi liniari
    - PG – cromozomi ne-liniari
  - Dimensiunea cromozomilor
    - AG – fixă
    - PG – variabilă (în adâncime sau lăţime)
  - Schema de creare a descendenţilor
    - AG – încrucişare şi mutaţie
    - PG – încrucişare sau mutaţie

Sisteme inteligente – SIS – PG

- Avantaje
  - PG găseşte soluţii problemelor care nu au o soluţie optimă
    - Un program pentru conducerea maşinii → nu există o singură soluţie
    - Unele soluţii implică un condus sigur, dar lent
    - Alte soluţii implică o viteză mare, dar un risc ridicat de accidente
    - Coducerea maşinii ←→ compromis între viteză mare şi siguranţă
  - PG este utilă în problemele a căror variabile se modifică frecvent
    - Conducerea maşinii pe autostradă
    - Conducerea maşinii pe un drum forestier
- Limite
  - Timpul mare necesar evoluţiei pentru identificarea soluţiei

Sisteme inteligente – SIS – PG

- Versiuni ale PG
  - PG liniară (Cramer, Nordin)
  - Gene Expression Programming (Ferreira)
  - Multi Expression Programing (Oltean)
  - Gramatical Evolution (Ryan, O'Neill)
  - Cartesian Genetic Programming (Miller)

Sisteme inteligente – SIS – PG

- PG liniară
  - Evoluarea de programe scrise într-un limaj imperativ (calculul fitness-ului nu necesită interpretare) → viteză mare de lucru
  - Reprezentare
    - Vector de instrucţiuni, fiecare instrucţiune fiind de forma (pp ca aritatea maximală a unei funcţii din F este n):
      - Index_op, registru_out, registru_in$_1$, registru_in$_2$,...,registru_in$_n$

- $v_i = v_j * v_k$ // instruction operating on two registers
- $v_i = v_j * c$ // instruction operating on one register and one constant
- $v_i = sin(v_j)$ // instruction operating on one register

void LGP_program (double $v[11]$)
{
$v[8] = v[0] - 10;$
$v[6] = v[2] * v[0];$
$v[5] = v[8] * 7;$
$v[4] = v[2] - v[0];$
$v[10] = v[1]/v[4];$
$v[3] = sin(v[1]);$
$v[1] = v[8] - v[6];$
$v[7] = v[10] * v[3];$
$v[9] = v[0] + v[7];$
$v[2] = v[7] + 3;$
}

void LGP_effective_program (double $v[11]$)
{
$v[4] = v[2] - v[0];$
$v[10] = v[1]/v[4];$
$v[3] = sin(v[1]);$
$v[7] = v[10] * v[3];$
$v[9] = v[0] + v[7];$
}

Sisteme inteligente – SIS – PG

- PG liniară
  - Iniţializare
    - Aleatoare
    - Restricţii
      - Lungimea iniţială a cromozomului (nr de instrucţiuni)
  - Operatori genetici de variaţie
    - Încrucişare – cu 2 puncte de tăietură
    - Mutaţie
      - Micro mutaţie → schimbarea unui operand sau operator (nu se modifică dimensiunea cromozomului)
      - Macro mutaţie → inserarea sau eliminarea unei instrucţiuni (se modifică dimensiunea cromozomului)

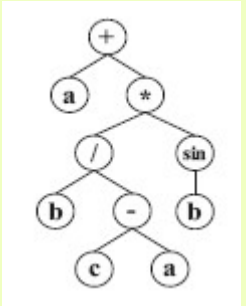
Sisteme inteligente – SIS – PG

- PG liniară
  - Avantaje
    - Evoluare într-un limbaj de nivel redus (low-level)
  - Dezavantaje
    - Numărul de regiştri necesari (numărul de atribute ale problemei)
  - Resurse
    - Register Machine Learning Technologies http://www.aimlearning.com
    - *Peter Nordin's home page http://fy.chalmers.se/~pnordin*
    - *Wolfgang Banzhaf's home page http://www.cs.mun.ca/~banzhaf*
    - *Markus Brameier's home page http://www.daimi.au.dk/~brameier*

Sisteme inteligente – SIS – PG

- Gene Expression Programming (GEP)
  - Ideea de bază
    - Reprezentarea liniară a expresiilor codabile în arbori (prin parcuregerea în lăţime a acestora – breadth-first)

$$C = +a * /Sb - bcacabbc$$



  - Reprezentare
    - Un cromozom este format din mai multe gene
      - Legate între ele prin + sau *
    - Fiecare genă este formată din:
      - Cap
        - conţine h elemente → funcţii şi terminale
      - Coadă
        - conţine doar terminale, în număr de t = (n-1)*h+1, unde n – aritatea maximă a unei funcţii din F

Sisteme inteligente – SIS – PG

- GEP
  - Iniţializare
    - Aleatoare, cu elemente din F şi T conform regulilor precizate anterior
  - Operatori de variaţie
    - Încrucişare
      - La nivel de alelă
        - Cu un punct de tăietură
        - Cu două puncte de tăietură
      - Încrucişare la nivel de genă
        - Cromozomii schimbă între ei anumite gene (plasate pe aceeaşi poziţie)
    - Mutaţie
      - La nivel de alelă
        - se modifică un element din cap sau coadă, respectând regulile de la iniţializare
    - Transpoziţii

Sisteme inteligente – SIS – PG

- GEP
  - Avantaje
    - Codarea în cromozomi a unor programe corecte datorită separării unei gene în cap şi coadă
  - Dezavantaje
    - Cromozomii multi-genă
      - Câte gene?
      - Cum se leagă genele între ele?
  - Resurse
    - Gene Expression Programming website, http://www.gepsoft.com
    - *Heitor Lopes's home page* http://www.cpgei.cefetpr.br/~hslopes/index-english.html
    - *Xin Li's home page* *http://www.cs.uic.edu/~xli1*
    - *GEP in C#* http://www.c-sharpcorner.com/Code/2002/Nov/GEPAlgorithm.asp

Sisteme inteligente – SIS – PG

- Multi Expression Programming (MEP)
  - Ideea de bază
    - Cromozomul este format din mai multe gene, fiecare genă → cod cu 3 adrese
      - Similar PG liniară, dar mai rapid
  - Reprezentare
    - Liniară
    - O genă conţine o funcţie (unară sau binară) şi pointeri spre argumentele sale
    - Cromozomul codează mai multe potenţiale soluţii → fiecare soluţie corespunde unei gene
      - Calitatea unei soluţii (gene) = suma (peste datele de antrenament) între ceea ce trebuia obţinut şi ceea ce se obţine
      - Calitatea unui cromozom = fitness-ul celei mai bune gene

Sisteme inteligente – SIS – PG

- MEP
  - Iniţializare
    - Prima genă trebuie să fie un terminal
    - Restul genelor pot conţine
      - un terminal sau
      - o funcţie (unară sau binară) şi pointeri spre argumentele sale
        - Argumentele unei funcţii poziţionată în a i-a genă trebuie să fie poziţionate în cromozom la indici mai mici decât i
  - Operatori de variaţie
    - Încrucişare → schimbarea unor gene între părinţi
      - Cu un punct de tăietură
      - Cu două puncte de tăietură
      - Uniformă
    - Mutaţie → modificarea unei gene
      - Prima genă → generarea unui nou terminal
      - Restul genelor → generarea unui terminal sau a unei funcţii (simbolul funcţiei şi argumentele funcţiei)
        - Generarea are loc la fel ca la iniţializare

Sisteme inteligente – SIS – PG

- MEP
  - Avantaje
    - Ieşire dinamică corespunzătoare unui cromozom
      - Complexitatea programului (expresiei) căutat(e)
      - Programe (expresii) de lungime variabilă obţinute fără operatori speciali
      - Programe de lungime exponenţială codate în cromozomi de lungime polinomială
  - Dezavantaje
    - Complexitatea decodării pt date de antrenament necunoscute → evoluarea strategiilor de joc
  - Resurse
    - Mihai Oltean's home page http://www.cs.ubbcluj.ro/~
    - *Crina Gro»san's home page http://www.cs.ubbcluj.ro/~cgrosan*
    - MEP web page http://www.mep.cs.ubbcluj.ro
    - *MEP in C# http://www.c-sharpcorner.com*

Sisteme inteligente – SIS – PG

- Grammatical Evolution (GE)
  - Ideea de bază
    - Evoluarea de programe în forma Backus-Naur (program exprimat sub forma unei gramatici cu simboluri terminale şi non-terminale, simbol de start, reguli/producţii)
  - Reprezentare
    - String binar de codons (grupuri de 8 biţi) → care regulă a gramaticii tb aplicată
    - Exemplu
      - G={N,T,S,P}, N={+, -,*, /, sin, (,)}, T= {expr, op2, op1}, S=⟨expr⟩, iar P este:
        - ⟨expr⟩ ::=a|b|c| ⟨expr⟩ ⟨op2⟩ ⟨expr⟩|(⟨expr⟩ ⟨op2⟩ ⟨expr⟩)| ⟨op1⟩ ⟨expr⟩
        - ⟨op2⟩::=+|-|*|/,
        - ⟨op1⟩::=sin
      - C^*_{GE}=(9 12 12 3 15 7 11 4 2 5 0 6 11 0 1 7 12)
      - S= ⟨expr⟩ ➔ ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ➔ a⟨op2⟩ ⟨expr⟩ ➔a + ⟨expr⟩ ➔ a + ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ➔ a + ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ⟨op2⟩ ⟨expr⟩ ➔ a + b ⟨op2⟩ ⟨expr⟩ ⟨op2⟩ ⟨expr⟩

$$C_{GE} = (00001001\ \ 00001100\ \ 00001100\ \ 00000011\ \ 00001111\ \ 00000111\ \ 00001011\ \ 00000100\ \ 00000010\ \ 00000101\ \ 00000000\ \ 00000110\ \ 00001011\ \ 00000000\ \ 00000001\ \ 00000111\ \ 00001100)$$

$$a + b/\langle expr\rangle\ \langle op_2\rangle\ \langle expr\rangle$$
$$a + b/(\langle expr\rangle\ \langle op_2\rangle\ \langle expr\rangle)\ \langle op_2\rangle\ \langle expr\rangle$$
$$a + b/(c\ \langle op_2\rangle\ \langle expr\rangle)\ \langle op_2\rangle\ \langle expr\rangle$$
$$a + b/(c - \langle expr\rangle)\ \langle op_2\rangle\ \langle expr\rangle$$
$$a + b/(c - a)\ \langle op_2\rangle\ \langle expr\rangle$$
$$a + b/(c - a) * \langle expr\rangle$$
$$a + b/(c - a) * \langle op_1\rangle\ \langle expr\rangle$$
$$a + b/(c - a) * \sin\langle expr\rangle$$

$$E = a + b/(c - a) * \sin(b)$$

Sisteme inteligente – SIS – PG

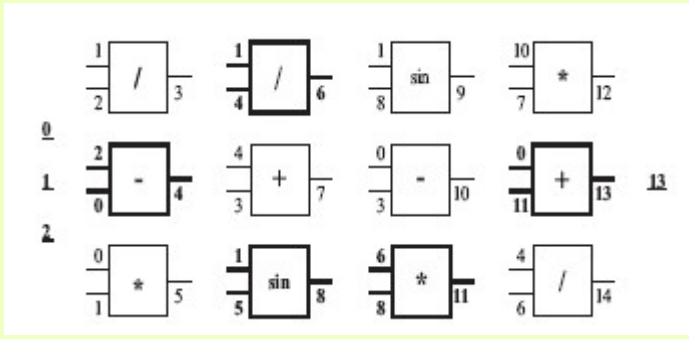
- GE
  - Iniţializare
    - Stringul binar este iniţializat aleator cu 0 şi 1 fără restricţii ➔ programe valide
    - Decodarea se termină când s-a obţinut un program complet
      - Dacă s-au terminat codons şi încă nu s-a format tot programul, se reiau codons de la primul element → wrapping
  - Operatori de variaţie
    - Încrucişare
      - Cu un punct de tăietură
    - Mutaţie
      - Schimbarea probabilistică a unui bit în opusul său
    - Duplicare
      - O secvenţă de gene este copiată la sfârşitul cromozomului
    - Pruning
      - Eliminarea genelor ne-folosite în procesul de transformare (decodare) a cromozomului

Sisteme inteligente – SIS – PG

- GE
  - Avantaje
    - Evoluarea de programe scrise în limbaje a căror instrucţiuni pot fi exprimate ca reguli de tip BNF
    - Reprezentarea poate fi schimbată prin modificarea gramaticii
  - Dezavantaje
    - Wrapping-ul la infinit → limitarea repetărilor şi penalizarea cromozomilor care depăşesc un anumit prag de repetări
  - Resurse
    - Grammatical Evolution web page, http://www.grammatical-evolution.org
    - *Conor Ryan's home page, http://www.csis.ul.ie/staff/conorryan*
    - *Michael O'Neill's home page, http://ncra.ucd.ie/members/oneillm.html*
    - *John James Collins's home page, http://www.csis.ul.ie/staff/jjcollins*
    - *Maarten Keijzer's home page, http://www.cs.vu.nl/~mkeijzer*
    - *Anthony Brabazon's home page http://ncra.ucd.ie/members/brabazont.*html

Sisteme inteligente – SIS – PG

- Cartesian Genetic Programming (CGP)
  - Ideea de bază
    - Cromozomi sub formă de graf (matrice) ➔ programe mai complexe decât cele din arbori
  - Reprezentare
    - În sistem cartezian (matrice de noduri)
    - Un nod are asociate
      - O funcţie
      - Intrări
      - Ieşiri
    - Ouputul cromozomului
      - Outputul oricărui nod



$$C = (1,2,3,\ \mathbf{2,0,1},\ 0,1,2,\ \mathbf{1,4,3},\ 4,3,0,\ \mathbf{1,5,4},\ 1,8,4,\ 0,3,1,\ \mathbf{6,8,2},\ 10,7,2,\ \mathbf{0,11,0},\ 4,6,3,\ 13)$$

Sisteme inteligente – SIS – PG

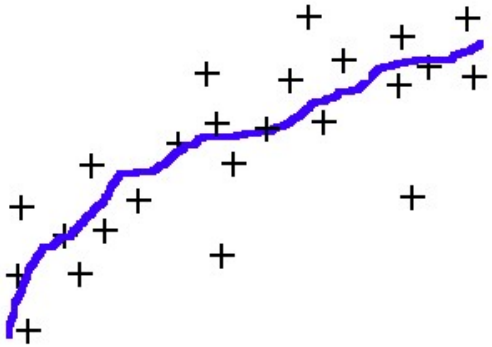
- CGP
  - Iniţializare
    - Aleatoare
    - Intrările oricărui nod trebuie să fie noduri de pe coloanele anterioare
      - Nodurile de pe prima coloană au ca intrări caracteristicile datelor de antrenament
  - Operatori de variaţie
    - Încrucişare
      - Nu se aplică
    - Mutaţie
      - Modificarea elementelor unui nod

Sisteme inteligente – SIS – PG

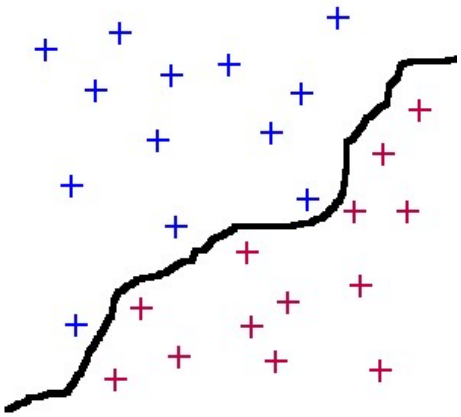
- CGP
  - Avantaje
    - Evoluarea indicelui nodului care furnizează ieşirea programului codat în cromozom
    - Programul evoluat poate avea una sau mai multe ieşiri
  - Dezavantaje
    - Stabilirea numărului de coloane influenţează rezultatele obţinute
  - Resurse
    - Julian. F. Miller's home page http://www.elec.york.ac.uk/intsys/users/jfm7
    - *Lukás Sekanina's home page http://www.fit.vutbr.cz/~sekanina/*

Sisteme inteligente – SIS – PG

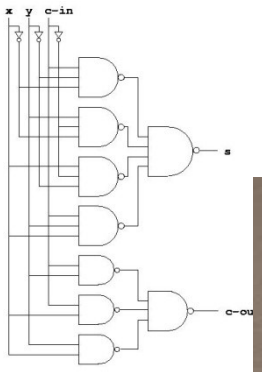
- Aplicaţii
  - Probleme în care există o relaţie între intrări şi ieşiri
  - Probleme de regresie

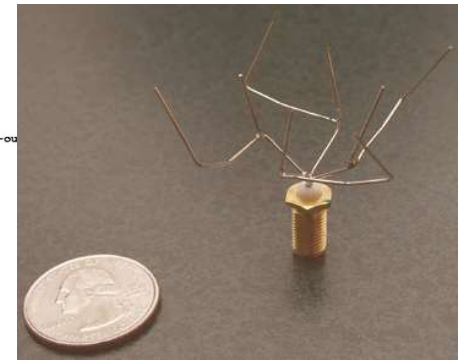


  - Probleme de clasificare



Sisteme inteligente – SIS – PG



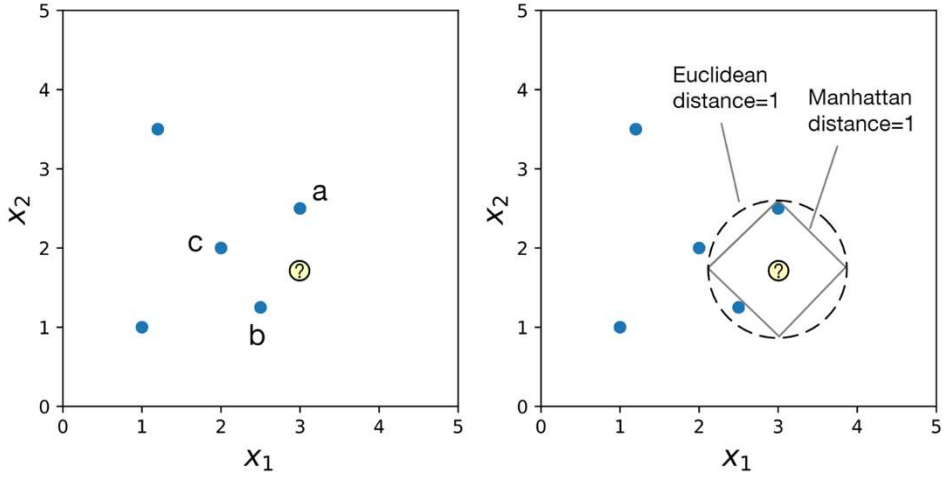


- Aplicaţii
  - Probleme de design
    - Evoluarea de circuite digitale
    - Evoluarea de antene
      - http://idesign.ucsc.edu/projects/evo_antenna.html
    - Evoluarea de programe (scrise într-un anumit limbaj)
    - Evoluarea de picturi şi muzică
      - http://www.cs.vu.nl/~gusz/
    - Altele
      - http://www.genetic-programming.com/humancompetitive.html

Sisteme inteligente – SIS - kNN

Cei mai apropiați k vecini (*k-nearest neighbours - kNN*)

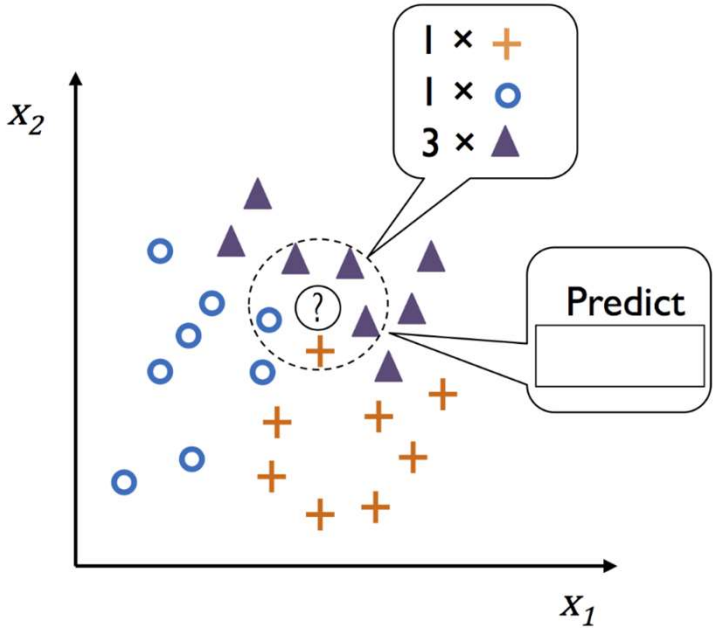
- Unul dintre cei mai simpli algoritmi de clasificare
- În etapa de antrenament, algoritmul doar citeşte datele de intrare (atributele şi clasa fiecărei instanţe)
- În etapa de testare, pentru o nouă instanţă (fără clasă):
  - se caută (printre instanţele de antrenament) **cei mai apropiaţi *k* vecini**
    - distanţa Minkowski (Manhattan, Euclidiană) – atribute continue
    - distanţa Hamming, Levensthein – analiza textelor
    - alte distanţe (funcţii kernel)



  - se preia clasa majoritară a acestor *k* vecini

Sisteme inteligente – SIS - kNN

- Unul dintre cei mai simpli algoritmi de clasificare
- În etapa de antrenament, algoritmul doar citeşte datele de intrare (atributele şi clasa fiecărei instanţe)
- În etapa de testare, pentru o nouă instanţă (fără clasă)
  - se caută (printre instanţele de antrenament) cei mai apropiaţi *k* vecini şi
  - **se preia clasa majoritară a acestor *k* vecini**



Majority vote:

Plurality vote:

Majority vote: None

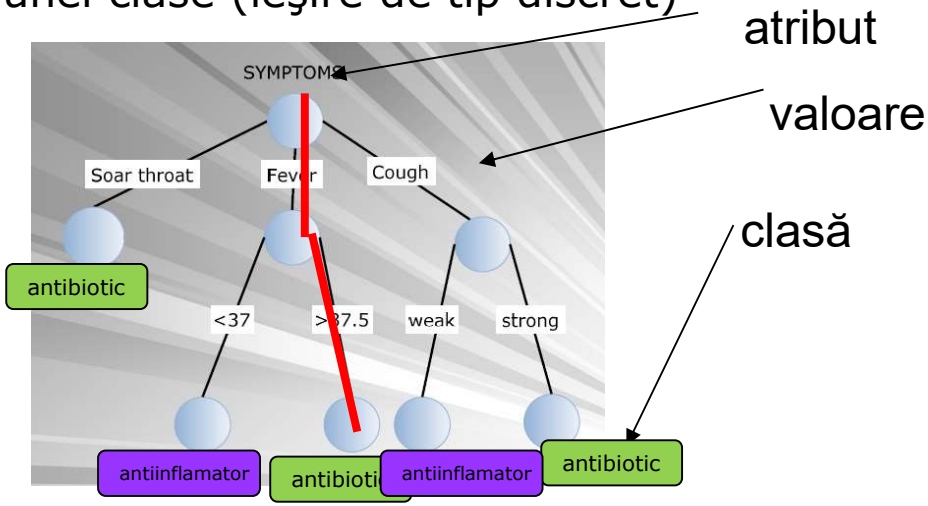
Plurality vote:

Sisteme inteligente – SIS - kNN

- Tool-uri
  - Sklearn (python)
    - https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
  - Weka (java)
    - https://weka.sourceforge.io/doc.dev/weka/classifiers/lazy/IBk.html
- Biblio
  - https://github.com/rasbt/stat479-machine-learning-fs19/tree/master/02_knn

Sisteme inteligente – SIS – Arbori de decizie

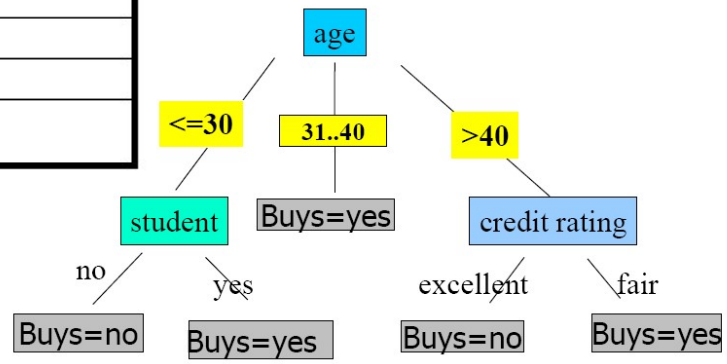
- Scop
  - Divizarea unei colecţii de articole în seturi mai mici prin aplicarea succesivă a unor reguli de decizie → adresarea mai multor întrebări
    - Fiecare întrebare este formulată în funcţie de răspunsul primit la întrebarea precedentă
  - Elementele se caracterizează prin informaţii non-metrice
- Definire
  - Fiecare nod intern corespunde unui atribut
  - Fiecare ramură de sub un nod (atribut) corespunde unei valori a atributului
  - Fiecare frunză corespunde unei clase (ieşire de tip discret)



Sisteme inteligente – SIS – Arbori de decizie

- Exemplu

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



Sisteme inteligente – SIS – Arbori de decizie

- Definire
  - Arborele de decizie
    - Un graf special → arbore orientat bicolor
    - Conţine noduri de 3 tipuri:
      - Noduri de decizie → posibilităţile decidentului (ex. Diversele examinări sau tratamente la care este supus pacientul) şi indică un test pe un atribut al articolului care trebuie clasificat
      - Noduri ale hazardului – evenimente aleatoare în afara controlului decidentului (rezultatul examinărilor, efectul terapiilor)
      - Noduri rezultat – situaţiile finale cărora li se asociază o utilitate (apreciată aprioric de către un pacient generic) sau o etichetă
    - Nodurile de decizie şi cele ale hazardului alternează pe nivelele arborelui
    - Nodurile rezultat – noduri terminale (frunze)
    - Muchiile arborelui (arce orientate) → consecinţele în timp (rezultate) ale deciziilor, respectiv ale realizării evenimentelor aleatoare (pot fi însoţite de probabilităţi)
  - Fiecare nod intern corespunde unui atribut
  - Fiecare ramură de sub un nod (atribut) corespunde unei valori a atributului
  - Fiecare frunză corespunde unei clase

Sisteme inteligente – SIS – Arbori de decizie

- Tipuri de probleme
  - Exemplele (instanţele) sunt reprezentate printr-un număr fix de atribute, fiecare atribut putând avea un număr limitat de valori
  - Funcţia obiectiv ia valori de tip discret
  - AD reprezintă o disjuncţie de mai multe conjuncţii, fiecare conjuncţie fiind de forma atributul a_i are valoarea v_j
  - Datele de antrenament pot conţine erori
  - Datele de antrenament pot fi incomplete
    - Anumitor exemple le pot lipsi valorile pentru unele atribute
  - **Probleme de clasificare**
    - Binară
      - exemple date sub forma [($atribut_{ij}$, $valoare_{ij}$), $clasă_i$, i=1,2,...,n, j=1,2,...,m, $clasă_i$ putând lua doar 2 valori]
    - Multi-clasă
      - exemple date sub forma [($atribut_{ij}$, $valoare_{ij}$), $clasă_i$, i=1,2,...,n, j=1,2,...,m, $clasă_i$ putând lua k valori]
  - **Probleme de regresie**
    - AD se construiesc similar cazului problemei de clasificare, dar în locul etichetării fiecărui nod cu eticheta unei clase se asociază nodului o valoare reală sau o funcţie dependentă de intrările nodului respectiv
    - Spaţiul de intrare se împarte în regiuni de decizie prin tăieturi paralele cu axele Ox şi Oy
    - Are loc o transformare a ieşirilor discrete în funcţii continue
    - Calitatea rezolvării problemei
      - Eroare (pătratică sau absolută) de predicţie

Sisteme inteligente – SIS – Arbori de decizie

- Proces
  - Construirea (creşterea, inducţia) arborelui
    - Se bazează pe un set de date de antrenament
    - Lucrează de jos în sus sau de sus în jos (prin divizare – *splitting*)
  - Utilizarea arborelui ca model de rezolvare a problemelor
    - Ansamblul decizilor efectuate de-a lungul unui drum de la rădăcină la o frunză formează o regulă
    - Regulile formate în AD sunt folosite pentru etichetarea unor noi date
  - Tăierea (curăţirea) arborelui (pruning)
    - Se identifică şi se mută/elimină ramurile care reflectă zgomote sau excepţii

Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD
  - Divizarea datelor de antrenament în subseturi pe baza caracteristicilor datelor
    - Un nod → întrebare legată de o anumită proprietate a unui obiect dat
    - Ramurile ce pleacă din nod → etichetate cu posibilele răspunsuri la întrebarea din nodul curent
    - La început toate exemplele sunt plasate în rădăcină
      - La pornire, un atribut va fi rădăcina arborelui, iar valorile atributului vor deveni ramuri ale rădăcinii
    - Pe următoarele nivele exemplele sunt partiţionate în funcţie de atribute ➔ ordinea considerării atributelor
      - Pentru fiecare nod se alege în mod recursiv câte un atribut (cu valorile lui pe ramurile descendente din nodul curent)
    - Divizarea → greedy în luarea decizilor
  - Proces iterativ
    - Reguli de oprire
      - toate exemplele aferente unui nod fac parte din aceeaşi clasă → nodul devine frunză şi este etichetat cu Ci
      - Nu mai sunt exemple → nodul devine frunză şi este etichetat cu clasa majoritară în setul de date de antrenament
      - nu mai pot fi considerate noi atribute

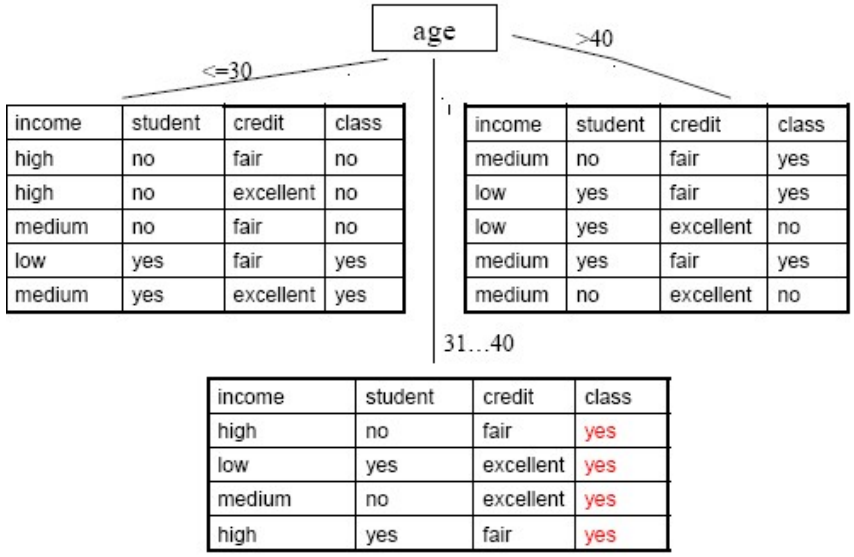
Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD
  - Exemplu

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

Sisteme inteligente – SIS – Arbori de decizie

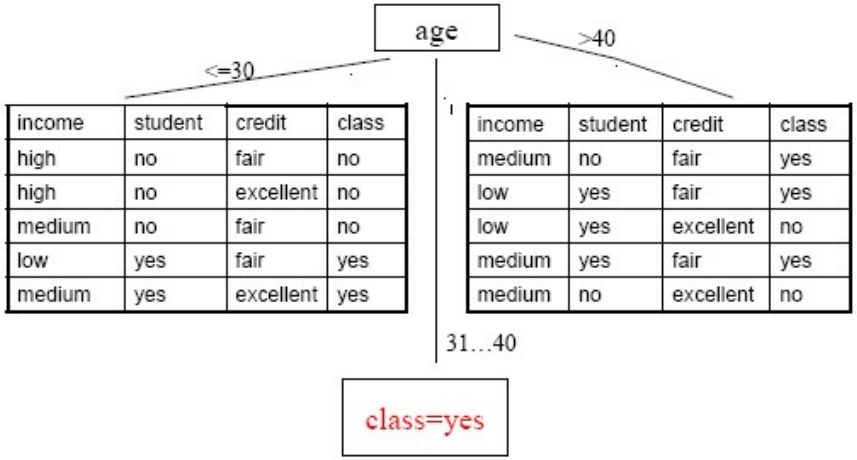
- Proces → Construirea AD
  - Exemplu
    - Pentru rădăcină se alege atributul *age*



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

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

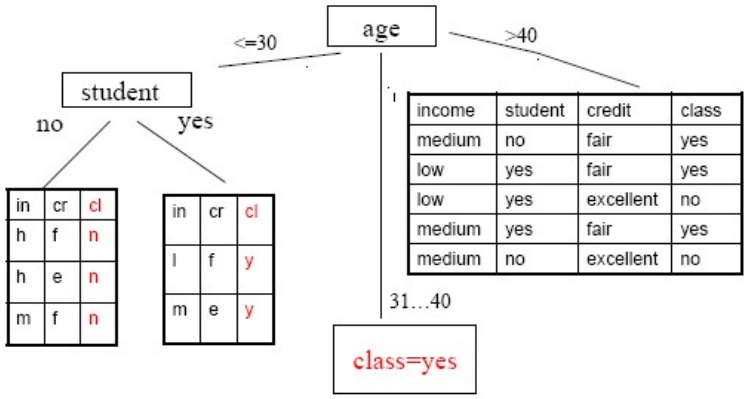
income	student	credit	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

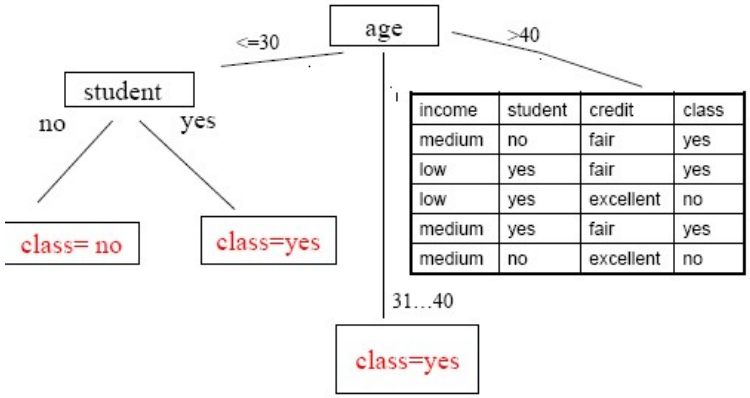


income	student	credit	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

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

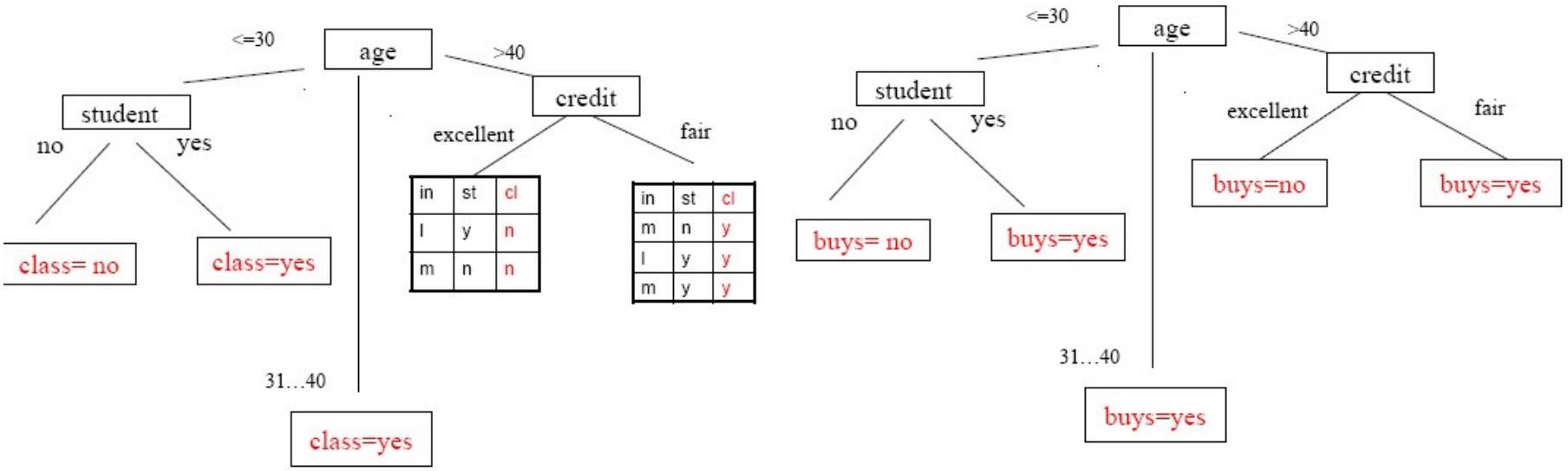
- Proces → Construirea AD
  - Exemplu
    - Pentru rădăcină se alege atributul *age*
    - Pe ramura <=30 se alege atributul student





Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD
  - Exemplu
    - Pentru rădăcină se alege atributul *age*
    - Pe ramura <=30 se alege atributul student
    - Pe ramura > 40 se alege atributul credit



Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD → Algoritmul ID3/C4.5
  - Greedy, recursiv, top-down, divide-and-conquer

```
generare(D, A){     //D – partiţionare a exemplelor de antrenament, A – lista de atribute
    Crearea unui nod nou N
    Dacă exemplele din D fac parte dintr-o singură clasă C atunci
        nodul N devine frunză şi este etichetat cu C
        returnează nodul N
    Altfel
        Dacă A=∅ atunci
            nodul N devine frunză şi este etichetat cu clasa majoritară în D
            returnează nodul N
        Altfel
            atribut_separare = Selectează_atribut(D, A)
            Etichetează nodul N cu atribut_separare
            Pentru fiecare valoare posibilă vj a lui atribut_separare
                Fie Dj mulţimea exemplelor din D pentru care atribut_separare = vj
                Dacă Dj =∅ atunci
                    Ataşează  nodului N o frunză etichetată cu clasa majoritară în D
                Altfel
                    Ataşează nodului N un nod returnat de generare(Dj, A –atribut_separare)
            Returnează nodul N
}
```

Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD → Algoritmul ID3/C4.5
  - Selectează_atribut(D, A) → Alegerea atributului aferent unui nod (rădăcină sau intern)
    - Aleatoare
    - Atributul cu cele mai puţine/multe valori
    - Pe baza unei ordini prestabilite a atributelor
      - Câştigul de informaţie
      - Rata câştigului
      - Indicele Gini
      - Distanţa între partiţiile create de un atribut

Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD → Algoritmul ID3/C4.5 → Selectare atribut
  - Câştigul de informaţie
    - O măsură de impuritate
      - 0 (minimă) dacă toate exemplele fac parte din aceeaşi clasă
      - 1 (maximă) dacă avem număr egal de exemple din fiecare clasă
    - Se bazează pe entropia datelor
      - măsoară impuritatea datelor
      - numărul sperat (aşteptat) de biţi necesari pentru a coda clasa unui element oarecare din setul de date
      - clasificare binară (cu 2 clase): $E(S) = - p_+ log_2 p_+ - p_- log_2 p_-$, unde
        - p_+ - proporţia exemplelor pozitive în setul de date S
        - p_- - proporţia exemplelor negative în setul de date S
      - clasificare cu mai multe clase: $E(S) = \sum_{i=1, 2, ..., k} - p_i log_2 p_i$ – entropia datelor relativ la atributul ţintă (atributul de ieşire), unde
        - p_i – proporţia exemplelor din clasa i în setul de date S
    - câştigul de informaţie (*information gain*) al unei caracterisitici a (al unui atribut al) datelor
      - Reducerea entropiei setului de date ca urmare a eliminării atributului *a*
      - $Gain(S, a) = E(S) - \sum_{v \in valori(a)} |S_v| / |S| \, E(S_v)$
      - $\sum_{v \in valori(a)} |S_v| / |S| \, E(S_v)$ – *informaţia scontată*

Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD → Algoritmul ID3/C4.5 → Selectare atribut
  - Câştigul de informaţie
    - exemplu

	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

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

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

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

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

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

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

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

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

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

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

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

Sisteme inteligente – SIS – Arbori de decizie

- Proces → Construirea AD → Algoritmul ID3/C4.5 → Selectare atribut
  - Rata câştigului
    - Penalizează un atribut prin încorporarea unui termen – *split information* – sensibil la gradul de împrăştiere şi uniformitate în care atributul separă datele
      - *Split information* – entropia relativ la valorile posibile ale atributului *a*
      - *Sv* – proporţia exemplelor din setul de date *S* care au atributul *a* evaluat cu valoarea *v*
      - *splitInformation(S,a)=* $$-\sum_{v=value(a)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

Sisteme inteligente – SIS – Arbori de decizie

- Studiu de caz - Problema vampirilor

Items	Shadow	Complexion	Garlic	Accent	Vampire
i1	?	Pale	Yes	None	No
i2	Yes	Ruddy	Yes	None	No
i3	?	Ruddy	No	None	Yes
i4	No	Average	No	Heavy	Yes
i5	?	Average	No	Odd	Yes
i6	Yes	Pale	No	Heavy	No
i7	Yes	Average	No	Heavy	No
i8	?	Ruddy	Yes	Odd	No

Sisteme inteligente – SIS – Arbori de decizie

- Proces
  - Construirea arborelui
  - Utilizarea arborelui ca model de rezolvare a problemelor
    - Ideea de bază
      - Se extrag regulile formate în arborele anterior construit → Reguli extrase din arborele dat în exemplul anterior:
        - 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"*
      - Regulile sunt folosite pentru a clasifica datele de test (date noi). Fie *x* o dată pentru care nu se ştie clasa de apartenenţă → Regulile se pot scrie sub forma unor predicate astfel:
        - 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)*

Sisteme inteligente – SIS – Arbori de decizie

- Proces
  - Construirea arborelui
  - Utilizarea arborelui ca model de rezolvare a problemelor
    - Dificultăţi
      - *Underfitting* (sub-potrivire) → AD indus pe baza datelor de antrenament este prea simplu ➔ eroare de clasificare mare atât în etapa de antrenare, cât şi în cea de testare
      - *Overfitting* (supra-potrivire, învăţare pe derost) → AD indus pe baza datelor de antrenament se potriveşte prea accentuat cu datele de antrenament, nefiind capabil să generalizeze pentru date noi
      - Soluţii:
        - fasonarea arborelui (pruning) → Îndepărtarea ramurilor nesemnificative, redundante ➔ arbore mai puţin stufos
        - validare cu încrucişare

Sisteme inteligente – SIS – Arbori de decizie

- Proces
  - Construirea arborelui
  - Utilizarea arborelui ca model de rezolvare a problemelor
  - Tăierea (fasonarea) arborelui
    - Necesitate
      - Odată construit AD, se pot extrage reguli (de clasificare) din AD pentru a putea reprezenta cunoştinţele sub forma regulilor *if-then* atât de uşor de înţeles de către oameni
      - O regulă este creată (extrasă) prin parcurgerea AD de la rădăcină până la o frunză
      - Fiecare pereche (atribut,valoare), adică (nod, muchie), formează o conjuncţie în ipoteza regulii (partea dacă), mai puţin ultimul nod din drumul parcurs care este o frunză şi reprezintă consecinţa (ieşirea, partea atunci) regulii
    - Tipologie
      - Prealabilă (*pre-pruning*)
        - Se opreşte creşterea arborelui în timpul inducţiei prin sistarea divizării unor noduri care devin astfel frunze etichetate cu clasa majoritară a exemplelor aferente nodului respectiv
      - Ulterioară (*post-pruning*)
        - După ce AD a fost creat (a crescut) se elimină ramurile unor noduri care devin astfel frunze ➔ se reduce eroarea de clasificare (pe datele de test)

Sisteme inteligente – SIS – Arbori de decizie

- Tool-uri
  - http://webdocs.cs.ualberta.ca/~aixplore/learning/DecisionTrees/Applet/DecisionTreeApplet.html
  - 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

Sisteme inteligente – SIS – Arbori de decizie

- Avantaje
  - Uşor de înţeles şi interpretat
  - Permit utilizarea datelor nominale şi categoriale
  - Logica deciziei poate fi urmărită uşor, regulile fiind vizibile
  - Lucrează bine cu seturi mari de date

- Dezavantaje
  - Instabilitate → modificarea datelor de antrenament
  - Complexitate → reprezentare
  - Greu de manevrat
  - Costuri mari pt inducerea AD
  - Inducerea AD necesită multă informaţie

Sisteme inteligente – SIS – Arbori de decizie

- Dificultăţi
  - Existenţa mai multor arbori
    - Cât mai mici
    - Cu o acurateţe cât mai mare (uşor de "citit" şi cu performanţe bune)
    - Găsirea celui mai bun arbore → problemă NP-dificilă
  - Alegerea celui mai bun arbore
    - Algoritmi euristici
    - ID3 → cel mai mic arbore acceptabil
      - → teorema lui Occam: "always choose the simplest explanation"
  - Atribute continue
    - Separarea în intervale
      - Câte intervale?
      - Cât de mari sunt intervalele?
  - Arbori prea adânci sau prea stufoşi
    - Fasonarea prealabilă (pre-pruning) → oprirea construirii arborelui mai devreme
    - Fasonarea ulterioară (post-pruning) → înlăturarea anumitor ramuri

Învățare supervizată – algoritmi
Arbori de decizie

- Tool-uri
  - http://webdocs.cs.ualberta.ca/~aixplore/learning/DecisionTrees/Applet/DecisionTreeApplet.html
  - WEKA → J48
  - http://id3alg.altervista.org/
  - http://www.rulequest.com/Personal/c4.5r8.tar.gz
  - https://scikit-learn.org/stable/modules/tree.html

- Biblio
  - http://www.public.asu.edu/~kirkwood/DAStuff/decisiontrees/index.html
  - https://github.com/rasbt/stat479-machine-learning-fs19/tree/master/06_trees

Recapitulare

- Sisteme care învaţă singure (SIS)
  - Algoritmi de programare genetică (PG)
    - Algoritmi evolutivi cu cromozomi sub formă de arbore
    - Cromozomii
      - Arborescenţi
      - Matriciali
      - Liniari
    - codează potenţiale soluţii de tipul
      - Expresiilor matematice → probleme de regresie/clasificare
      - Expressilor de tip Boolean → probleme de tip EvenParity / proiectare de circuite digitale
      - Programelor → evoluarea de cod sursă pentru rezolvarea unor probleme

Cursul următor

A. Scurtă introducere în Inteligenţa Artificială (IA)

B. Rezolvarea problemelor prin căutare
- Definirea problemelor de căutare
- Strategii de căutare
  - Strategii de căutare neinformate
  - Strategii de căutare informate
  - Strategii de căutare locale (Hill Climbing, Simulated Annealing, Tabu Search, Algoritmi evolutivi, PSO, ACO)
  - Strategii de căutare adversială

C. Sisteme inteligente
- **Sisteme care învaţă singure**
  - Arbori de decizie
  - Reţele neuronale artificiale
  - Maşini cu suport vectorial
  - Algoritmi evolutivi
- Sisteme bazate pe reguli
- Sisteme hibride

Cursul următor –
Materiale de citit şi legături utile

- capitolul 15 din *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- Capitolul 9 din *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*

- Informaţiile prezentate au fost colectate din diferite surse de pe internet, precum şi din cursurile de inteligenţă artificială ţinute în anii anteriori de către:
  - Conf. Dr. Mihai Oltean – www.cs.ubbcluj.ro/~moltean
  - Lect. Dr. Crina Groşan - www.cs.ubbcluj.ro/~cgrosan
  - Prof. Dr. Horia F. Pop - www.cs.ubbcluj.ro/~hfpop