METODE INTELIGENTE DE REZOLVARE A PROBLEMELOR REALE

Laura Dioşan

Tema 4

Conținut

- Instruire automata (Machine Learning - ML)
  - Problematică
  - Proiectarea unui sistem de învăţare automată
  - Tipologie
    - Învăţare supervizată
    - Învăţare nesupervizată
    - Învăţare cu întărire
    - Teoria învăţării
- De citit:
  - S.J. Russell, P. Norvig – Artificial Intelligence - A Modern Approach → capitolul 18, 19, 20
  - Documentele din directoarele: ML, classification, clustering

Învățare automată

- Problematica
  - “How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?”
- Aplicaţii
  - Recunoaştere de imagini şi semnal vocal
  - Computer vision
  - Supraveghere bio
  - Controlul roboţilor

Învățare automată

- Arthur Samuel (1959)
  - "field of study that gives computers the ability to learn without being explicity programmed"
  - Înzestrarea computerelor cu abilitatea de a învăţa pe baza experienţei
- 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"
- Herbert Simon:
  - "Learning is any process by which a system improves performance from experience."

- De ce?
  - Sisteme computaţionale mai bune
    - Sisteme dificil sau prea costisitor de construit manual
      - Sisteme care se adaptează automat
        - Filtre de spam
      - Descoperirea de informaţii în baze de date mari → data mining
        - Analize financiare
        - Analize de text/imagini
  - Înţelegerea organismelor biologice

Învățarea

- Îmbunătăţirea task-ului T
- respectând o metrică de performanţă P şi
- bazându-se pe experienţa E

- Ex. :
  - T: jucarea jocului de dame
  - P: procentul de jocuri câştigate împotriva unui oponent oarecare
  - E: exersarea jocului împotriva lui însuşi

  - T: recunoaşterea scrisului de mână
  - P: procentul de cuvinte recunoscute corect
  - E: baze de date cu imagini cu cuvinte corect adnotate

  - T: separarea spam-urilor de mesajele obişnuite
  - P: procentul de email-uri corect clasificate (spam sau normal)
  - E: baze de date cu email-uri adnotate

Proiectarea unui sistem de învățare automată

- Îmbunătăţirea task-ului T
  - stabilirea scopului (ceea ce trebuie învăţat) - funcţiei obiectiv – şi reprezentarea sa
  - alegerea unui algoritm de învăţare care să realizeze inferenţa (previziunea) scopului pe baza experienţei
- respectând o metrică de performanţă P
  - evaluarea performanţelor algortimului ales
- bazându-se pe experienţa E
  - alegerea bazei de experienţă

Proiectare – Alegerea funcției obiectiv

- Care este funcţia care trebuie învăţată?
  - Ex. pt jocul de dame
    - o funcţie care
      - alege următoarea mutare
      - evaluează o mutare
    - obiectivul fiind alegerea celei mai bune mutări

Proiectare – Reprezentarea funcției obiectiv

- Diferite reprezentări
  - tablou (tabel)
  - reguli simbolice
  - funcţie numerică
  - funcţii probabilistice
  - ex. jocul de dame
    - Combinaţie liniară a nr. de piese albe, nr. de piese negre, nr. de piese albe compromise la următoarea mutare, nr. de piese negre compromise la următoarea mutare
- Există un compromis între
  - expresivitatea reprezentării şi
  - uşurinţa învăţării
- Calculul funcţiei obiectiv
  - timp polinomial
  - timp non-polinomial

Proiectare – Alegerea unui algoritm de învățare

- Algoritmul
  - folosind datele de antrenament
  - induce definirea unor ipoteze care
    - să se potirvească cu datele de antrenament şi
    - să generalizeze cât mai bine datele ne-văzute (datele de test)

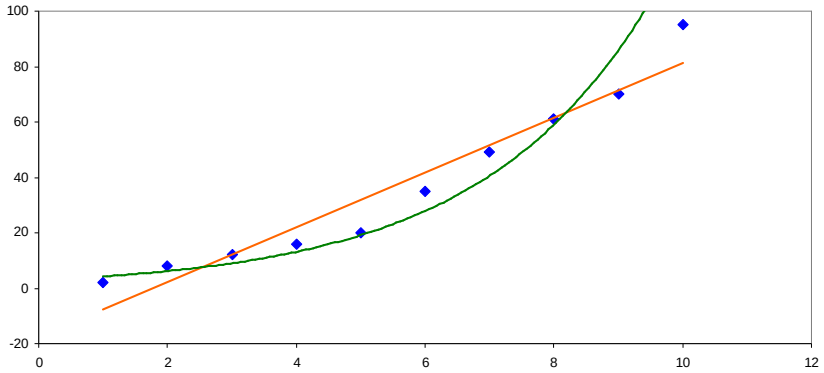
- Principiul de lucru
  - minimizarea unei erori (funcţie de cost – *loss function*)

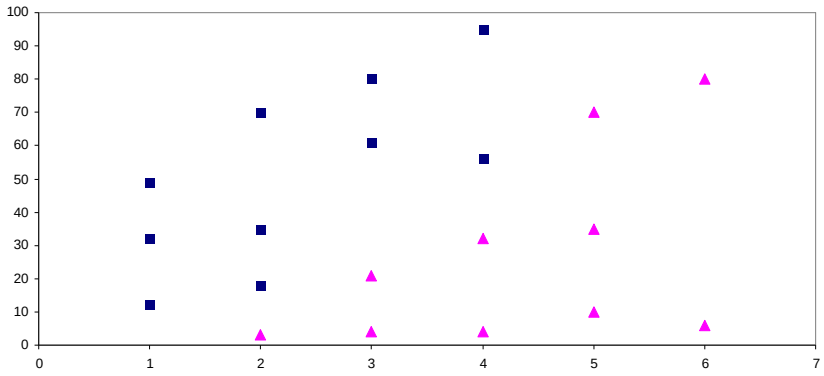
Proiectare – Învățare automată – tipologie

- Învăţare supervizată
- Învăţare nesupervizată
- Învăţare cu întărire

Învățare supervizată

- Scop:
  - Furnizarea unei ieşiri corecte pentru o nouă intrare
- 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
- Caracteristic
  - BD experimentală adnotată (pt. învăţare)

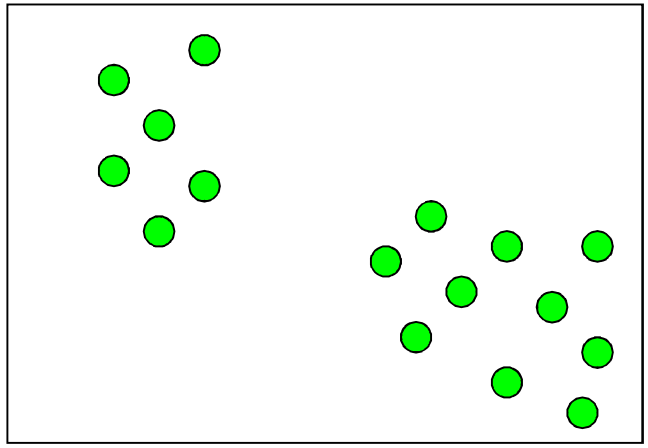




Învățare nesupervizată

- Scop
  - Găsirea unui model sau a unei structuri utile a datelor

- 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



- Caracteristic
  - Datele nu sunt adnotate (etichetate)

Învățare 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 benig

Teoria învățării

- De unde ştim dacă un algoritm de învăţare funcţionează?
- Cum se aproximează funcţiile?
- Câte date de antrenament sunt necesare?

Proiectare – Evaluarea unui sistem de învățare

- Experimental
  - Compararea diferitelor metode pe diferite date (cross-validare)
  - Colectarea datelor pe baza performanţei
    - Acurateţe, timp antrenare, timp testare
  - Aprecierea diferenţelor dpdv statistic
- Teoretic
  - Analiza matematică a algoritmilor şi demonstrarea de teoreme
    - Complexitatea computaţională
    - Abilitatea de a se potrivi cu datele de antrenament
    - Complexitatea eşantionului relevant pentru o învăţare corectă

Proiectare – Evaluarea unui sistem de învățare

Compararea performanţelor a 2 algoritmi în rezolvarea unei probleme

- Indicatori de performanţă
  - Parametrii ai unei serii statistice
    - Ex. media
    - Legea normală
  - Proporţie calculată pentru serie statistică
    - Indicator variabil, variabilă dummy
    - Ex. Acurateţea
    - Legea binomială
- Comparare pe baza intervalelor de încredere

Proiectare – Evaluarea unui sistem de învățare

Compararea performanţelor a 2 algoritmi în rezolvarea unei probleme → Interval de încredere pentru medie

- Pt o serie statistică:
  - de volum n
  - cu media (calculată) m
  - dispersia σ
- să se determine intervalul de încredere al valorii medii μ

- $P(-z \leq (m-\mu)/(\sigma/\sqrt{n}) \leq z) = 1 - \alpha$
  - → $\mu \epsilon [m - z\sigma/\sqrt{n},\ m + z\sigma/\sqrt{n}]$
- $P = 95\%$ → $z = 1.96$

- Ex. Problema rucsacului rezolvată cu ajutorul algoritmilor evolutivi

$P=1-\alpha$	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

Proiectare – Evaluarea unui sistem de învățare

Compararea performanţelor a 2 algoritmi în rezolvarea unei probleme → Interval de încredere pentru acurateţe

- Pt o performanţă *p* (acurateţe)
  - calculată pentru *n* date
- să se determine intervalul de încredere

- $[p-z(p(1-p)/n)^{1/2}, p+z(p(1-p)/n)^{1/2}]$
- $P = 95\% \rightarrow z = 1.96$

- Ex. Problemă de clasificare rezolvată cu ajutorul Maşinilor cu suport vectorial

P=1-α	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

Proiectare – Evaluarea unui sistem de învățare

Compararea performanţelor a 2 algoritmi în rezolvarea unei probleme
→ pe baza intervalelor de încredere

- Pp o problemă şi 2 algoritmi care o rezolvă
- Performanţele algoritmilor: p_1 şi p_2
- Intervalele de încredere corespunzătoare celor 2 performanţe
  - $I_1=[p_1-\Delta_1,p_1+\Delta_1]$ şi
  - $I_2=[p_2-\Delta_2,p_2+\Delta_2]$
- Dacă p_1 e mai bună ca p_2 *şi*
  - dacă $I_1 \cap I_2=\emptyset$ → algoritmul 1 este mai bun decât algoritmul 2 (pt problema dată)
  - dacă $I_1 \cap I_2 \neq \emptyset$ → nu se poate spune care algoritm este mai bun

Proiectare – Alegerea bazei de experienţă

- Experienţă directă
  - Perechi (intrare, ieşire) utile pt. funcţia obiectiv
  - Ex. Jocul de dame
    - tablă de joc etichetată cu mutare corectă sau incorectă
- Experienţă indirectă
  - Feedback util (diferit de perechile I/O) pt funcţia obiectiv
  - Ex. Jocul de dame
    - secvenţe de mutări şi scorul final asociat jocului

Proiectare – Alegerea bazei de experienţă

- Surse de date
  - Exemple generate aleator
    - Exemple pozitive şi negative
  - Exemple pozitive colectate de un “învăţător” benevol
  - Exemple reale

Proiectare – Alegerea bazei de experienţă

- Compoziţie
  - Date de antrenament
  - Date de test

- Caracteristici
  - Datele de antrenament şi de test trebuie
    - să fie independente
      - Dacă nu → clasificare colectivă
    - să urmeze aceeaşi lege de distribuţie
      - Dacă nu → învăţare prin transfer (transfer learning/inductive transfer)
        - recunoaşterea maşinilor → recunoaşterea camioanelor
        - analiza textelor
        - filtre de spam

Proiectare – Alegerea bazei de experienţă

- Tipuri de atribute ale datelor
  - Cantitative → scară nominală sau raţională
    - Valori continue → greutatea
    - Valori discrete → numărul de computere
    - Valori de tip interval → durata unor evenimente
  - Calitative
    - Nominale → culoarea
    - Ordinale → intensitatea sunetului (joasă, medie, înaltă)
  - Structurate
    - Arbori – rădăcina este o generalizare a copiilor (vehicol → maşină, autobus, tractor, camion)

Proiectare – Alegerea bazei de experienţă

- Transformări asupra datelor
  - Standardizare → atribute numerice
    - Înlăturarea efectelor de scară (scări şi unităţi de măsură diferite)
    - Valorile brute se transformă în scoruri z
      - $Z_{ij} = (x_{ij} - \mu_j)/\sigma_j$, unde x_{ij} – valoarea atributului al j-lea al instanţei i, μ_j (σ_j) este media (abaterea) atributelor j pt. toate instanţele
  - Selectarea anumitor atribute

Învățare automată

- Învăţare supervizată
- Teoria învăţării
- Învăţare nesupervizată
- Învăţare cu întărire

Învățare automată

- **Învăţare supervizată**
- Învăţare nesupervizată
- Învăţare cu întărire
- Teoria învăţării

Învățare supervizată

- Definire
- Exemple
- Proces
- Metode de evaluare şi măsuri de performanţă
- Tipologie
- Algoritmi

Învățare supervizată – definire

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_data$_i$), 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

Învățare supervizată – exemple

- Recunoaşterea scrisului de mână
- Recunoaşterea imaginilor
- Previziunea vremii
- Detecţia spam-urilor

Învățare supervizată – proces

Procesul

- 2 paşi:
  - Antrenarea
    - Învăţarea, cu ajutorul unui algoritm, a modelului de clasificare
  - Testarea
    - Testarea modelului folosind date de test noi (*unseen data*)

Calitatea învăţării

- o măsură de performanţă a algoritmului → ex. acurateţea
  - Acc = nr de exemple corect clasificate / nr total de exemple
- calculată în:
  - faza de antrenare
    - Ansamblul de date antrenament se împarte în
      - Date de învăţare
      - Date de validare
    - Performanţa se apreciază pe sub-ansamblul de validare
      - O singură dată
      - De mai multe ori → validare încrucişată (cross-validation)
  - faza de testare
- probleme
  - Învăţare pe derost (*overfitting*) → performanţă bună pe datele de antrenament, dar foarte slabă pe datele de test

Învățare supervizată – evaluare

Metode de evaluare

- Seturi disjuncte de antrenare şi testare
  - pt. date numeroase
  - setul de antrenare
    - poate fiîmpărţit în
      - Date de învăţare
      - Date de validare
    - Folosit pentru estimarea parametrilor modelului
      - Cei mai buni parametri obţinuţi pe validare vor fi folosiţi pentru construcţia modelului final

- 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
    - 1 sub-set pt validare
  - dimensiunea unui sub-set = dimensiunea setului / *h*
  - performanţa este dată de media pe cele *h* rulări
    - *h = 5* sau *h = 10*
  - pt date puţine

- Leave-one-out cross-validation
  - similar validării încrucişate, dar *h* = nr de date ➔ un sub-set conţine un singur exemplu
  - pt. date foarte puţine

Învățare supervizată – evaluare

Măsuri de performanţă

- Măsuri statistice
- 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

Învățare supervizată – evaluare

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

Învățare supervizată – evaluare

Măsuri de performanţă

- Măsuri statistice
  - Precizia şi Rapelul
    - 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)*
    - Matricea de confuzie
      - Rezultate reale vs. rezultate calculate
  - 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)*

Învățare supervizată – evaluare

Condiţii fundamentale

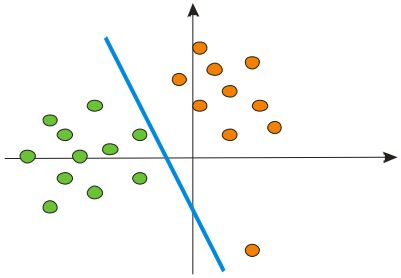
- Distribuţia datelor de antrenament şi test este aceeaşi
  - În practică, o astfel de condiţie este adesea violată
- Exemplele de antrenament trebuie să fie reprezentative pentru datele de test

Învățare supervizată – tipologie

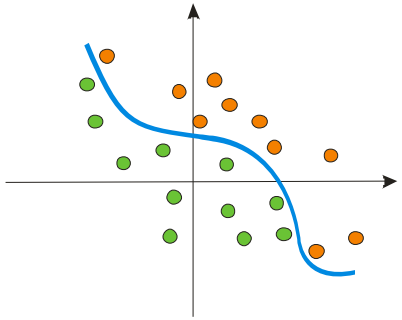
- După numărul de date de ieşire (probleme de clasificare)
  - Binară
    - Ieşiri (output-uri) binare → $k = 2$
      - Ex. 1 şi -1; 0 şi 1; acceptat şi refuzat
  - Multi-clasă
    - Ieşiri multiple → $k > 2$
      - Ex. 1, 0 şi -1; mic, mediu, mare şi foarte mare

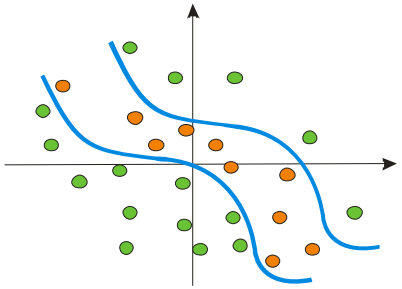
Învățare supervizată – tipologie

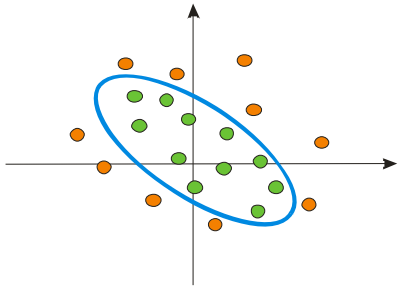
- După forma clasificatorului
  - Clasificare liniară



  - Clasificare ne-liniară
    - se crează o reţea de clasificatori liniari
    - se mapează datele într-un spaţiu nou (mai mare) unde ele devin separabile

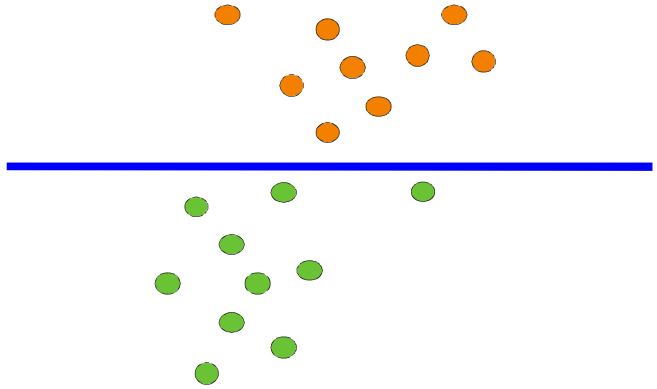




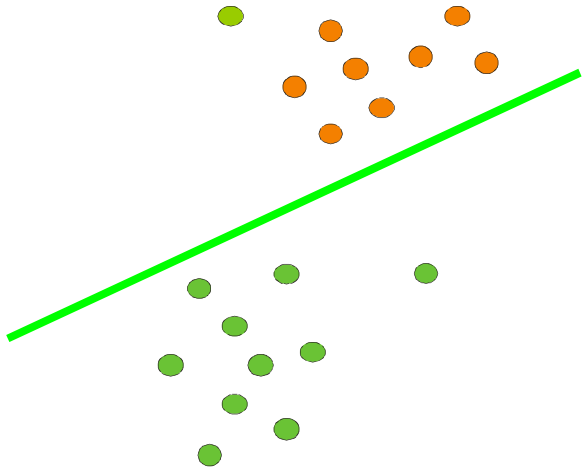


Învățare supervizată – tipologie

- După caracteristicile datelor
  - Clasificare pt date perfect separabile
    - Clasificare fără eroare



  - Clasificare pt date ne-separabile
    - Clasificare cu o anumită eroare (anumite date sunt plasate eronat în clase)



Învățare supervizată – tipologie

- După algoritm
  - Bazată doar pe instanţe
    - Foloseşte direct datele, fără a crea un model de separare
    - Ex. algoritmul cel mai apropiat vecin (*k-nearest neighbour*)
  - Generative
    - Construieşte un model probabilistic
    - Ex. reţele Bayesiene
  - Discriminative
    - Estimează o separare al datelor
    - Ex. arbori de decizie, reţele neuronale artificiale, maşini cu suport vectorial, algoritmi evolutivi

Învățare supervizată – algoritmi

- Cel mai apropiat vecin
- Arbori de decizie
- Sisteme bazate pe reguli
- Reţele neuronale
- Maşini cu suport vectorial

clasificare

- Algoritmi evolutivi → regresie

Învățare supervizată – algoritmi

Problemă de clasificare

- 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)
    - date de test – sub forma (atribute_data$_i$), 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) asociată unei date (noi) de test folosind funcţia învăţată pe datele de antrenament

Învățare supervizată – algoritmi

Problemă de regresie

- 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$_i$
        - un număr real
    - date de test – sub forma (atribute_data$_i$), 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

Învățare supervizată – algoritmi

Cel mai apropiat vecin (*k-nearest neighbour*)

- Cel mai simplu algoritm de clasificare
- În etapa de antrenament, algoritmul doar citeşte datele de intrare (atributele şi clasa fiecarei 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
- Căutarea vecinilor se bazează pe:
  - distanţa Euclidiană – atribute continue
  - distanţa Hamming – analiza textelor
  - alte distanţe

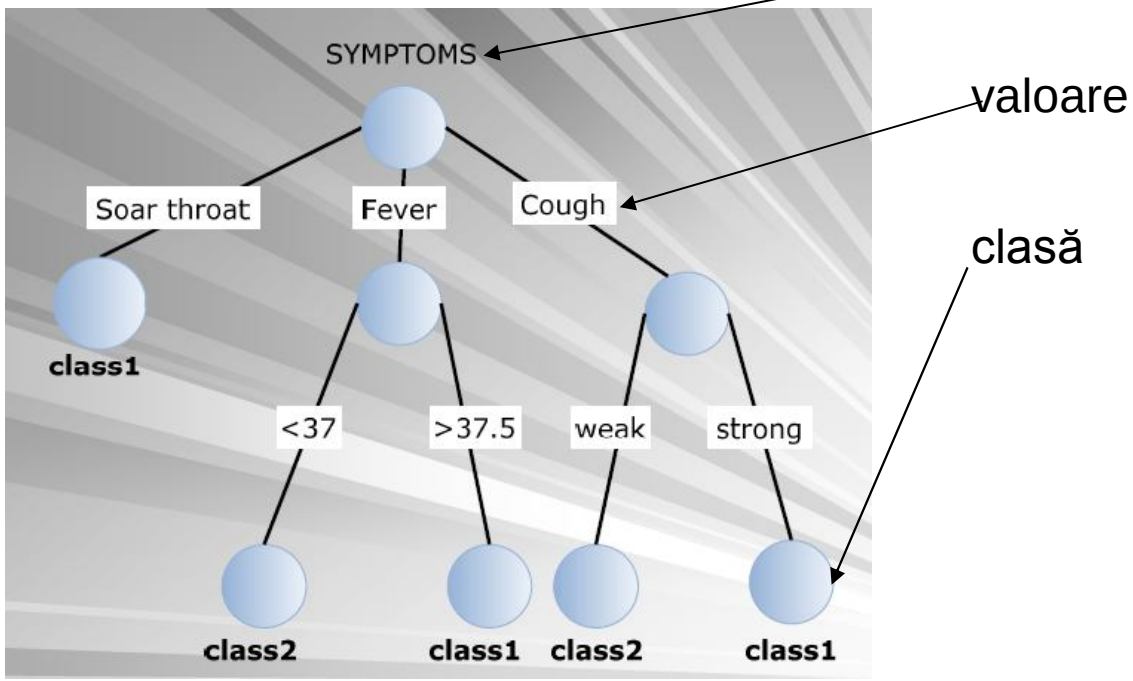
Învățare supervizată – algoritmi

Arbori de decizie

- Fiecare nod intern corespunde unui atribut
- Fiecare ramură de sub un nod (atribut) corespunde unei valori a atributului
- Fiecare frunză corespunde unei clase
- Cel mai cunoscut: C4.5 al lui Ross Quinlan

Învățare supervizată – algoritmi
Arbori de decizie

- Exemplu – sistem medical



Învățare supervizată – algoritmi
Arbori de decizie

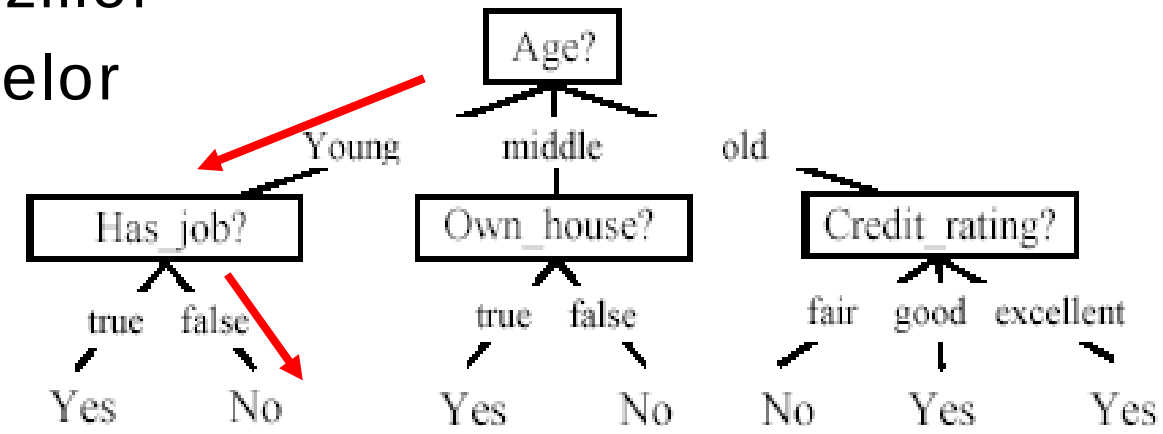
- Exemplu – acordarea de credite

aprobat sau nu

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**

Învățare supervizată – algoritmi
Arbori de decizie

- Arbori cu noduri
  - Interne →deciziilor
  - Frunză → claselor



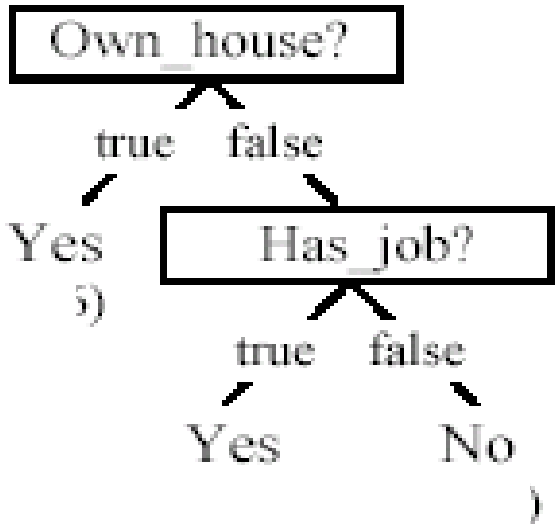
  - Date noi →

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

NO

Învățare supervizată – algoritmi
Arbori de decizie

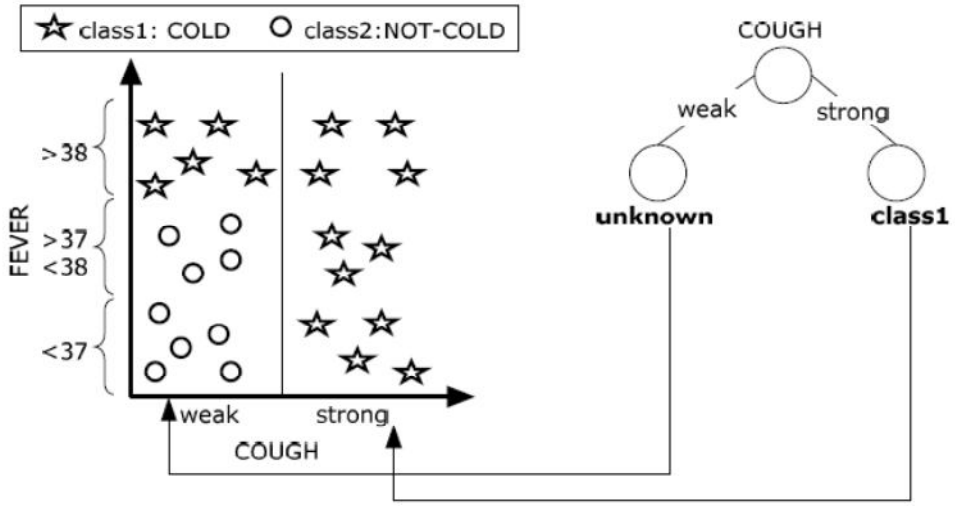
- Pot exista mai mulţi arbori
  - Cât mai mici
  - Cu o acurateţe cât mai mare (uşor de "citit" şi cu performanţe bune)
  - Găsirea celui mai bun arbore → problemă NP-dificilă
- Construirea şi alegerea arborelui
  - Algoritmi euristici
  - ID3 → cel mai mic arbore acceptabil
    - → teorema lui Occam: "always choose the simplest explanation"



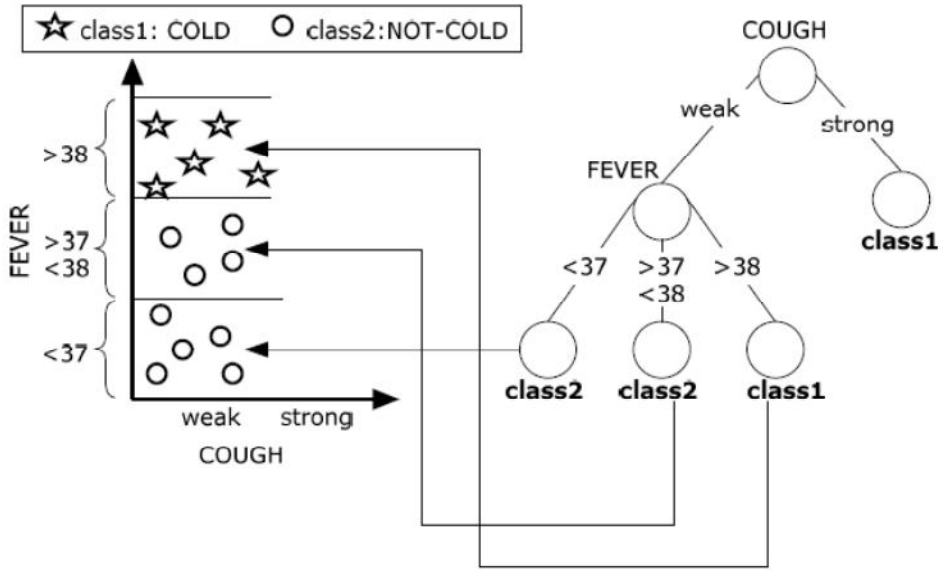
Învățare supervizată – algoritmi
Arbori de decizie

- O soluţie bazată pe ***divide-and-conquer***
  - Arbore construit recursiv de sus în jos
  - La început toate exemplele sunt plasate în rădăcină
  - Pe următoarele nivele exemplele sunt partiţionate în funcţie de atribute ➔ ordinea considerării atributelor
    - C4.5 se bazează pe câştigul de informaţie (information gain) – teoria informaţiei
  - Partiţionarea se opreşte când:
    - toate exemplele dintr-un nod aparţin aceleaşi clase
    - nu mai pot fi considerate noi atribute
    - nu mai sunt exemple

Învățare supervizată – algoritmi
Arbori de decizie



Învățare supervizată – algoritmi
Arbori de decizie



Învățare supervizată – algoritmi
Arbori de decizie

- Alegerea celui mai bun atribut
  - clasificare binară
    - p_+ - proporţia exemplelor pozitive în setul de date S
    - p_- - proporţia exemplelor negative în setul de date S
    - Entropia – măsoară impuritatea datelor
      - $E(S) = - p_+ log_2 p_+ - p_- log_2 p_-$
  - clasificare cu mai multe clase
    - p_i – proporţia exemplelor din clasa i în setul de date S
    - $E(S) = \Sigma_{i=1, 2, ..., k} - p_i log_2 p_i$
  - câştigul de informaţie (*information gain*) al unei caracterisitici a (al unui atribut) a datelor
    - Reducerea entropiei setului de date ca urmare a eliminării atributului a
    - $Gain(S, a) = E(S) - \Sigma_{v \in valori(a)} |S_v| / |S| \, E(S_v)$

Învățare supervizată – algoritmi
Arbori de decizie

	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

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

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

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

$S_{v=rosu} = \{d1, d2, d3\} \rightarrow p+ = 2/3,\ p- = 1/3 \rightarrow E(S_{v=rosu}) = 0.923$
$S_{v=albastru} = \{d4\} \rightarrow p+ = 0,\ p- = 1 \rightarrow E(S_{v=albastru}) = 0$

$S_{v=cerc} = \{d1, d3, d4\} \rightarrow p+ = 2/3,\ p- = 1/3 \rightarrow E(S_{v=cerc}) = 0.923$
$S_{v=patrat} = \{d2\} \rightarrow p+ = 0,\ p- = 1 \rightarrow E(S_{v=patrat}) = 0$

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

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

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

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

Învățare supervizată – algoritmi
Arbori de decizie

Probleme

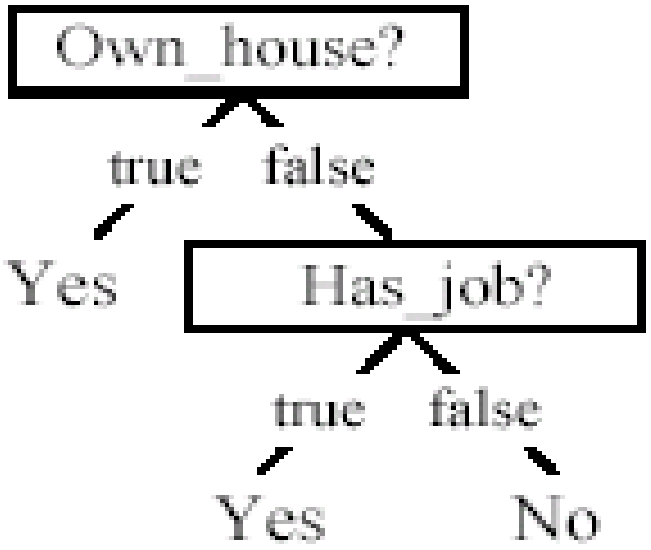
- Atribute continue
  - Separarea în intervale
    - Câte intervale?
    - Cât de mari sunt intervalele?
- Arbori prea adânci sau prea stufoşi
  - pre-elagaj (pre-pruning) → oprirea construirii arborelui mai devreme
  - post-elagaj (post-pruning) → înlăturarea anumitor ramuri

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

Învățare supervizată – algoritmi
Sisteme bazate pe reguli

- Transformarea AD într-un set de reguli
- Fiecare drum din arbore ➔ regulă
- Regulile *IF-THEN* pot fi identificate din date

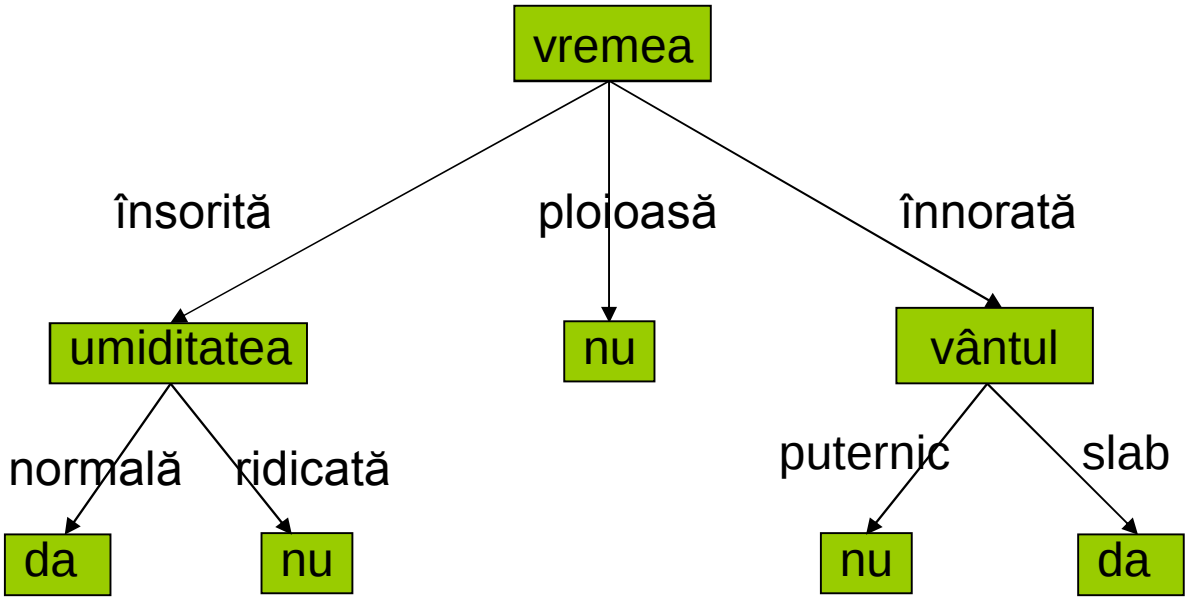


Own_house = true → Class =Yes
Own_house = false, Has_job = true → Class = Yes
Own_house = false, Has_job = false → Class = No

Învățare supervizată – algoritmi
Sisteme bazate pe reguli

Conversia unui AD în reguli

- Vremea propice pentru tenis



- DACĂ *vremea=înnorată* şi *vântul=slab* ATUNCI *joc tenis*

Învățare supervizată – algoritmi
Sisteme bazate pe reguli

- Secvenţele de reguli = liste de decizii
- Găsirea regulilor
  - Acoperire secvenţială
    - Se învaţă o regulă
    - Se elimină exemplele care respectă regula
    - Se caută noi reguli

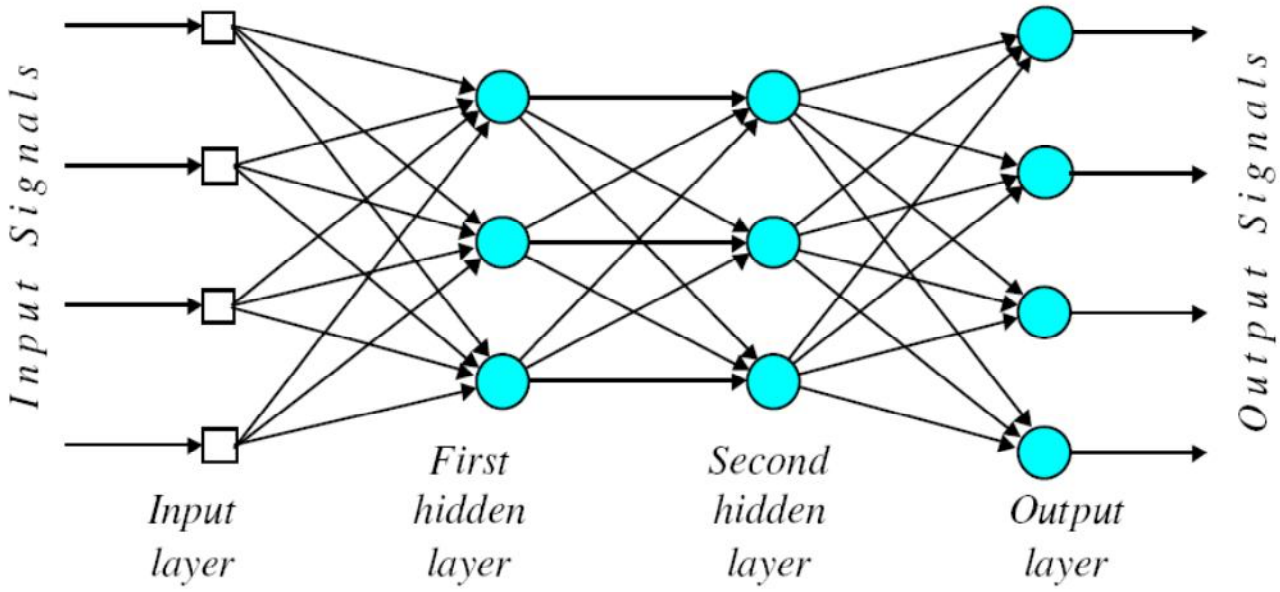
Învățare supervizată – algoritmi

Clasificare

- Clasificare binară pt orice fel de date de intrare (discrete sau continue)
  - Datele pot fi separate de:
    - o dreaptă → $ax + by + c = 0$ (dacă $m = 2$)
    - un plan → $ax + by + cz + d = 0$ (dacă $m = 3$)
    - un hiperplan $\Sigma a_i x_i + b = 0$ (dacă $m > 3$)
  - Cum găsim valorile optime pt. a, b, c, d, a_i?
    - Reţele neuronale artificiale
    - Maşini cu suport vectorial

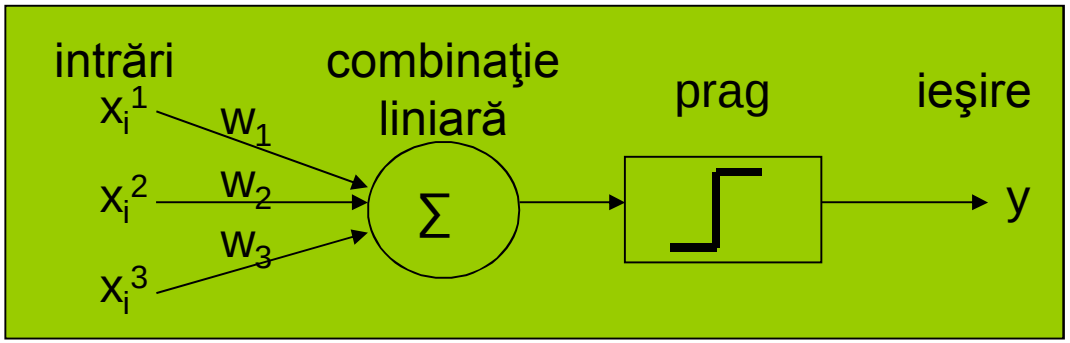
Învățare supervizată – algoritmi
Rețele neuronale artificiale

- Similar unei reţele neuronale biologice
- O mulţime de neuroni dispuşi ca într-un graf (un nod → un neuron) pe mai multe straturi (*layere*)
  - Strat de intrare
    - Conţine *m* (nr de atribute al unei date) noduri
  - Strat de ieşire
    - Conţine *r* (nr de ieşiri) noduri
  - Straturi intermediare (ascunse) – rol în "complicarea" reţelei
    - Diferite structuri
    - Diferite mărimi



Învățare supervizată – algoritmi
Rețele neuronale artificiale

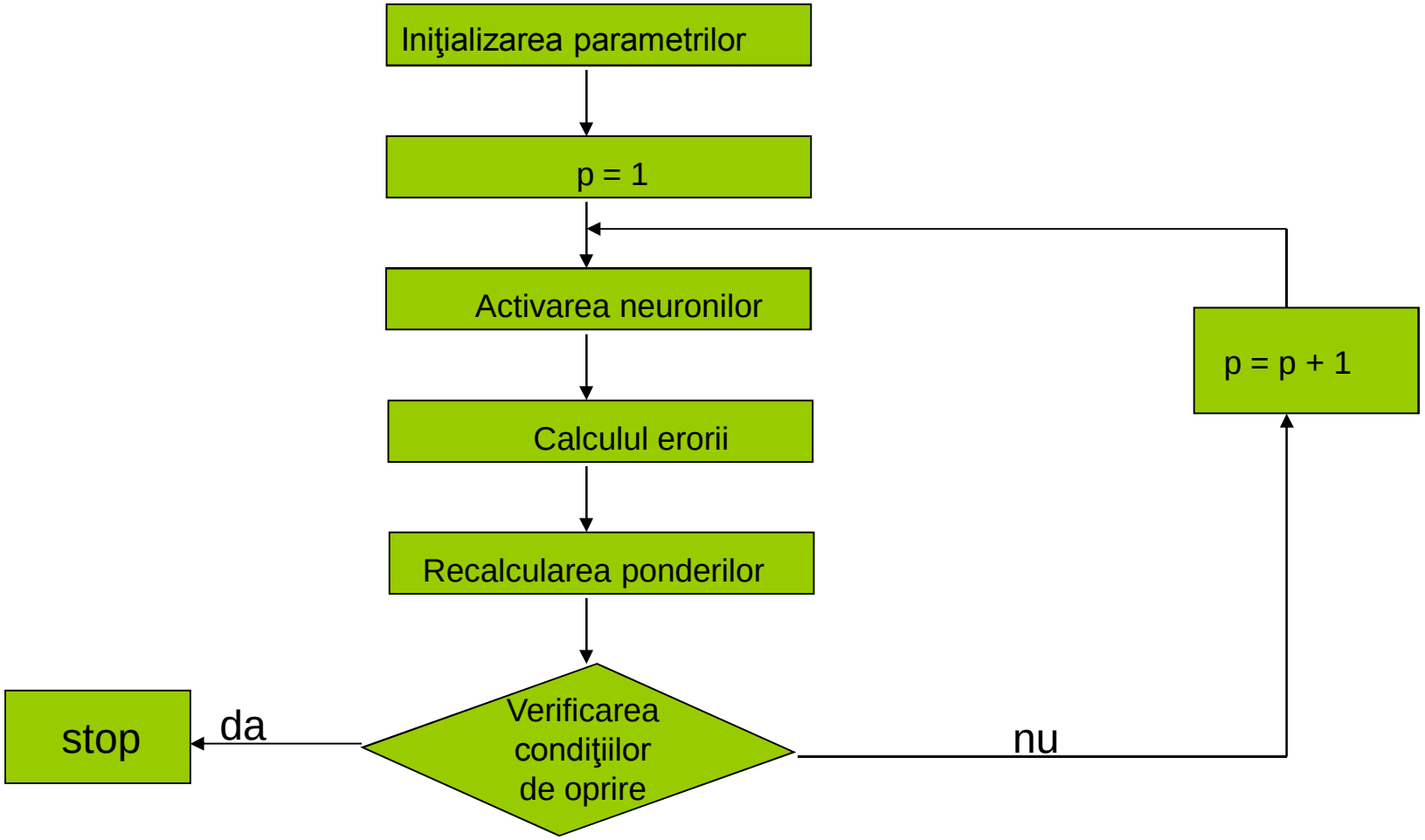
- O mulţime de neuroni
  - un neuron dotat cu
    - Intrări
    - Ieşiri
    - Funcţie de activare
      - Funcţia liniară Y = aX + b
      - Funcţia prag Y = 1, dacă X ≥ prag, 0 altfel
      - Funcţia sigmoidală Y = 1 / e^{-X}
      - Funcţia semn Y = +1, dacă X ≥ 0, -1 altfel
- Cum învaţă reţeaua?
  - Să pp. că avem *n = 10* date de antrenament cu câte *m = 3* atribute fiecare, o iesire cu 2 valori posibile (*k = 2* clase) şi o reţea



  - Pe baza datelor de antrenament se identifică valorile optime ale ponderilor w_i
  - Algoritmi
    - Perceptron – algoritmul iniţial pentru învăţarea unei reţele cu un singur strat ascuns format dintr-un singur neuron
    - Backpropagation – algoritm pentru învăţarea unei reţele cu mai multe straturi ascunse

Învățare supervizată – algoritmi
Rețele neuronale artificiale

- Algoritm de învăţare a ponderilor



Învățare supervizată – algoritmi
Rețele neuronale artificiale

Perceptron

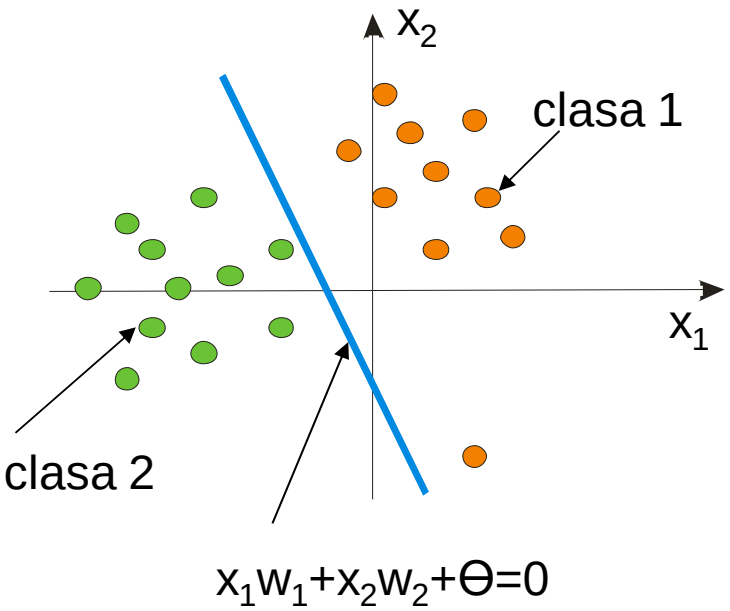
- Pp. ca avem n date cu câte m atribute fiecare şi o singură ieşire binară ((x_{p1}, x_{p2}, ..., x_{pm}, y_p)) cu $p = 1, 2, ..., n$ şi că folosim funcţia de activare sumă (pt un neuron)

- Se începe cu valori aleatorii pentru parametrii algoritmului:
  - ponderi (în intervalul [-0.5, 0.5]),
  - prag (Θ) şi
  - rata de învăţare (α)

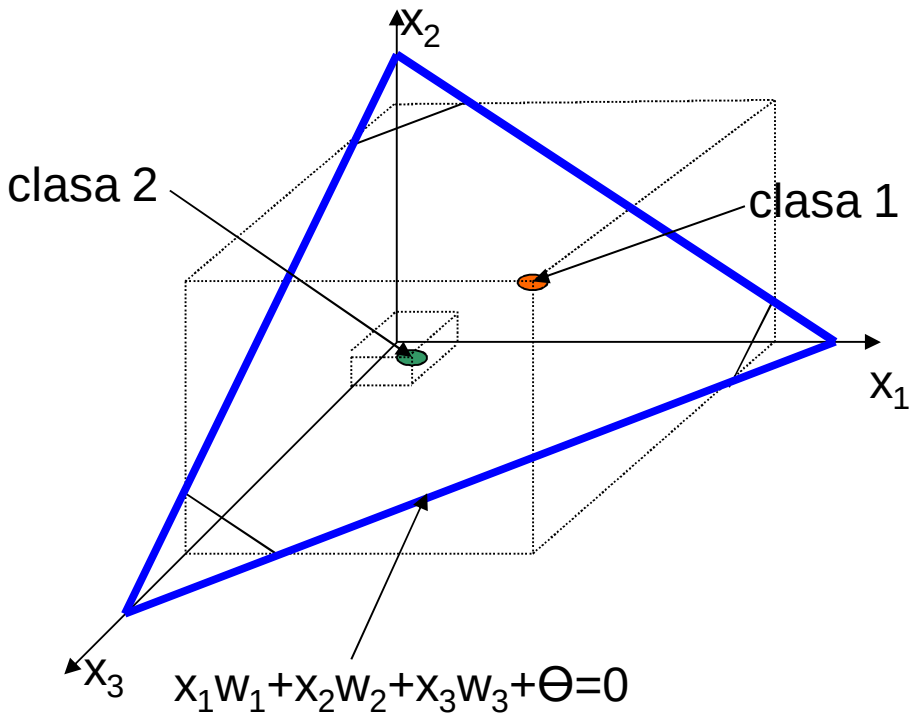
- Iteraţia p ($p = 1, n$)
  - Se prezintă reţelei exemplul p de date $(x_{p1}, x_{p2}, ..., x_{pm})$
  - Se activează neuronul = se calculează ieşirea $y_c(p) = \sum_{t=1,m} w_t(p)x_{pt} + \Theta$
  - Se calculează eroarea $e(p) = y_p - y_c(p)$
    - Dacă eroare e pozitivă → trebuie crescută ieşirea calculată
    - Dacă eroare e negativă → trebuie redusă ieşirea calculată
  - Se recalculează ponderile w_t pe baza corecţiei Δw_t
    - $w_t(p+1) = w_t(p) + \Delta w_t(p)$, unde $\Delta w_t = \alpha\, x_{pt}\, e(p)$, $t=1,2,...,m$
- Se trece la iteraţia următoare ($p = p + 1$) şi se reia activarea neuronului
  - La epuizarea datelor de antrenament se reîncepe cu primele date ($p = 1$)
  - n iteraţii = o epocă

- Algoritmul se opreşte când
  - Eroarea este nulă
  - Nu se mai obţin îmbunătăţiri

Învățare supervizată – algoritmi
Rețele neuronale artificiale



Clasificare binară cu m=2 intrări



Clasificare binară cu m=3 intrări

Învățare supervizată – algoritmi
Rețele neuronale artificiale

Perceptron - exemplu

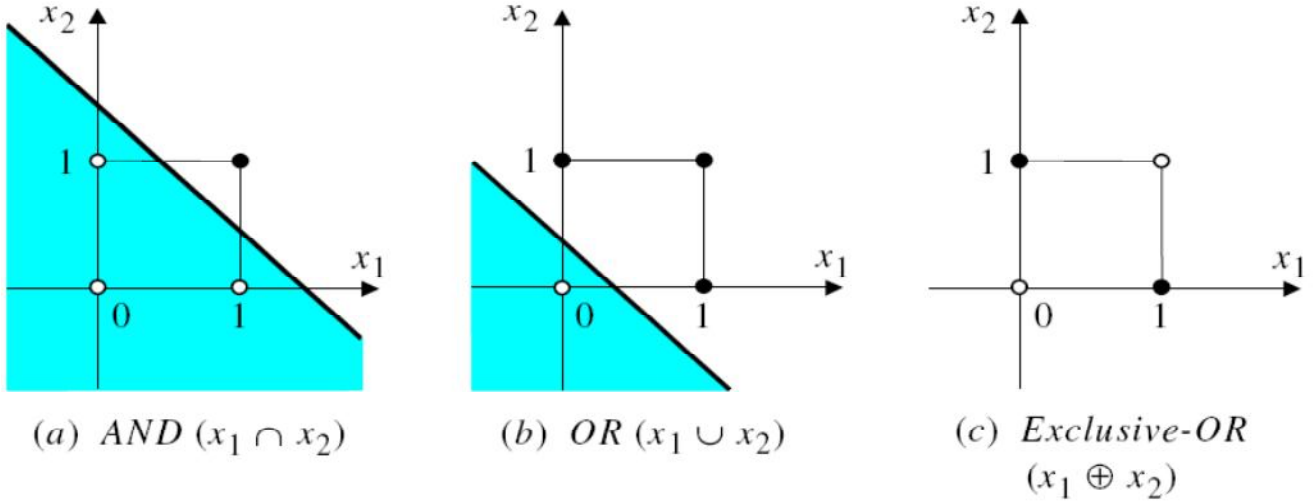
Epoch	Inputs		Desired output	Initial weights		Actual output	Error	Final weights	
	x_1	x_2	Y_d	w_1	w_2	Y	e	w_1	w_2
1	0	0	0	0.3	–0.1	0	0	0.3	–0.1
	0	1	0	0.3	–0.1	0	0	0.3	–0.1
	1	0	0	0.3	–0.1	1	–1	0.2	–0.1
	1	1	1	0.2	–0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	–1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	–1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	–1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$

Învățare supervizată – algoritmi
Rețele neuronale artificiale

Perceptron - limitări

- Un perceptron poate învăţa operaţiile AND şi OR, dar nu poate învăţa operaţia XOR (nu e liniar separabilă)



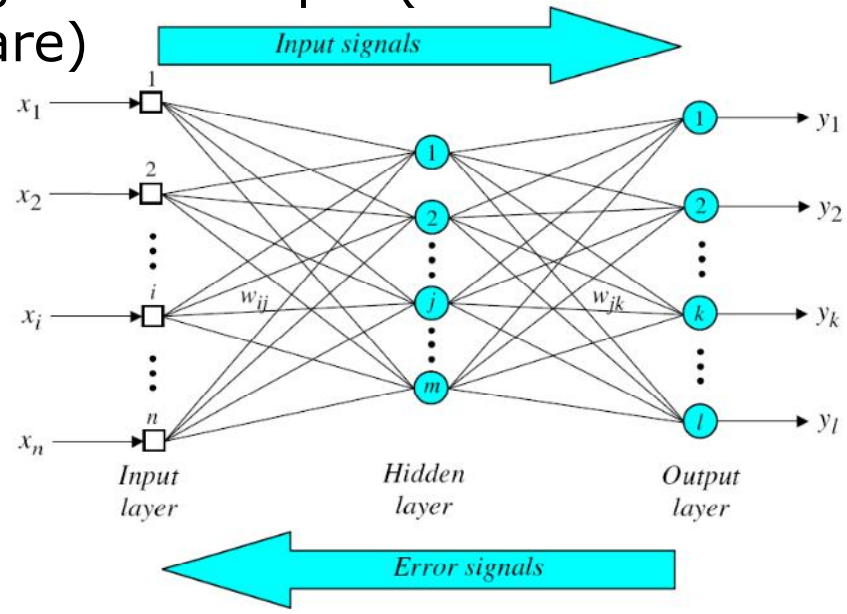
- Nu poate clasifica date non-liniar separabile
  - ➔ soluţia = mai mulţi neuroni

Învățare supervizată – algoritmi
Rețele neuronale artificiale

- Învăţarea – algoritmul backpropagation
  - Similar perceptronului
  - 2 etape
    - Propagarea intrării din stratul de intrare către stratul de ieşire şi calcularea ieşirii
    - Calcularea erorii şi propagarea ei înapoi (de la stratul de ieşire către cel de intrare)

      prin modificarea

      ponderilor



Învățare supervizată – algoritmi
Rețele neuronale artificiale

Algoritmul backpropagation

- Pp. ca avem n date cu câte m atribute fiecare şi o singură ieşire ((x_{p1}, x_{p2}, ..., x_{pm}, y_p)) cu $p = 1, 2, ..., n$ şi că folosim funcţia de activare sumă (pt un neuron) şi un singur strat ascuns cu H noduri

- Algoritm
  1. Iniţializarea
     - ponderilor între oricare 2 noduri ale reţelei,
     - a pragurilor corespunzătoare nodurilor din straturile ascuse si din stratul de ieşire şi
     - a ratei de învăţare
  2. Activarea
     - Se calculează ieşirile corespunzătoare nodurilor din stratul ascuns
       - $y_h(p) = \Sigma_{t=1,m} w_{th}(p)x_t(p) + \Theta_h$, $h = 1, 2, ..., H$
     - Se calculează ieşirea corespunzătoarea nodului din stratul de ieşire
       - $y_c(p) = \Sigma_{h=1,H} w_{hf}(p)y_h(p) + \Theta$
  3. Se calculează eroarea nodului de ieşire
       - $e(p) = y(p) - y_c(p)$
  4. Se recalculează ponderile dintre stratul ascuns şi nodul de ieşire w_{hf}, unde $h = 1, 2, ..., H$
     - $w_{hf}(p+1) = w_{hf}(p) + \alpha\, y_h(p)\, e(p)$
  5. Se calculează erorile nodurilor din stratul ascuns prin propagare înapoi
     - $e_h(p) = e(p)\, w_{hf}(p)$, unde $h = 1, 2, ..., H$
  6. Se recalculează ponderile w_{th}, unde $h = 1, 2, ..., H$ şi $t = 1, 2, ..., m$ pe baza corecţiei
     - $w_{th}(p+1) = w_{th}(p) + \alpha\, x_t(p)\, e_h(p)$
  7. Se revine la pasul 2

Învățare supervizată – algoritmi
Rețele neuronale artificiale

algoritmul backpropagation

- Pp. ca avem n date cu câte m atribute fiecare şi *r* ieşiri ((x_{p1}, x_{p2}, ..., x_{pm}, y_{p1}, y_{p2}, ...,y_{pr})) cu $p = 1, 2, ..., n$ şi că folosim funcţia de activare sumă (pt un neuron) şi un singur strat ascuns cu H noduri

- Algoritm
  1. Iniţializarea
     - ponderilor între oricare 2 noduri ale reţelei şi
     - a pragurilor corespunzătoare nodurilor din straturile ascuse si din stratul de ieşire
  2. Activarea
     - Se calculează ieşirile conrespunzătoare nodurilor din stratul ascuns
       - $y_h(p) = \sum_{t=1,m} w_{th}(p)x_t(p) + \Theta_h$, $h = 1, 2, ..., H$
     - Se calculează ieşirile corespunzătoare nodurilor din stratul de ieşire
       - $y_{cj}(p) = \sum_{h=1,H} w_{hj}(p)y_h(p) + \Theta_j$, $j = 1, 2, ..., r$
  3. Se calculează erorile nodurilor de ieşire
       - $e_j(p) = y_j(p) - y_{cj}(p)$, $j = 1, 2, ..., r$
  4. Se recalculează ponderile whj, unde $h = 1, 2, ..., H$ şi $j = 1, 2, ..., r$
     - $w_{hj}(p+1) = w_{hj}(p) + \alpha\, y_h(p)\, e_j(p)$
  5. Se calculează erorile nodurilor din stratul ascuns prin propagare înapoi
     - $e_h(p) = \sum_{j=1,k} e_j(p)\, w_{hj}(p)$, unde $h = 1, 2, ..., H$ şi $j = 1, 2, ..., r$
  6. Se recalculează ponderile w_{th}, unde $h = 1, 2, ..., H$ şi $t = 1, 2, ..., m$
     - $w_{th}(p+1) = w_{th}(p) + \alpha\, x_t(p)\, e_h(p)$
  7. Se revine la pasul 2

Învățare supervizată – algoritmi
Maşini cu suport vectorial (MSV)

- Dezvoltate de Vapnik în 1970
- Popularizate după 1992
- Clasificatori liniari care identifică un hiperplan de separaţie între clasa pozitivă şi cea negativă
- Au o fundamentare teoretică foarte riguroasă
- Funcţionează foarte bine pentru date de volum mare (analiza textelor)

Învățare supervizată – algoritmi
Maşini cu suport vectorial

Concepte de bază

- Un set de date
  - De antrenament
    - $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, unde
      - $\mathbf{x}_i = (x_1, x_2, \ldots, x_m)$ este un vector de intrare într-un spaţiu real $X \subseteq R^m$ şi
      - y_i este eticheta clasei (valoarea de ieşire), $y_i \in \{1, -1\}$
        - 1 → clasă pozitivă,
        - -1 → clasă negativă

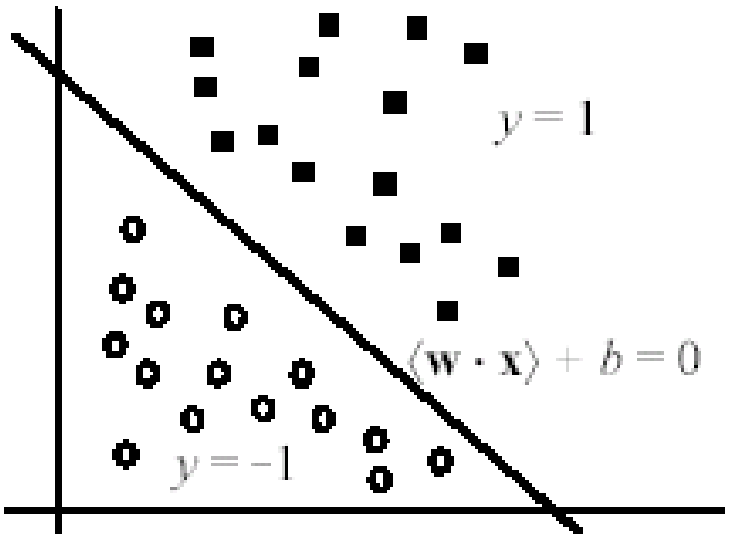
- MSV găseşte o funcţie liniară de forma

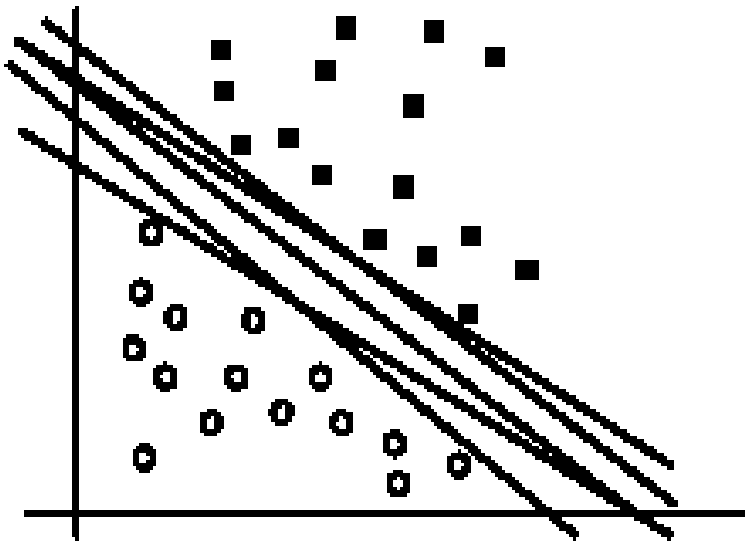
$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b, \text{ (}\mathbf{w}\text{: vector pondere)}$$

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

Învățare supervizată – algoritmi
Maşini cu suport vectorial

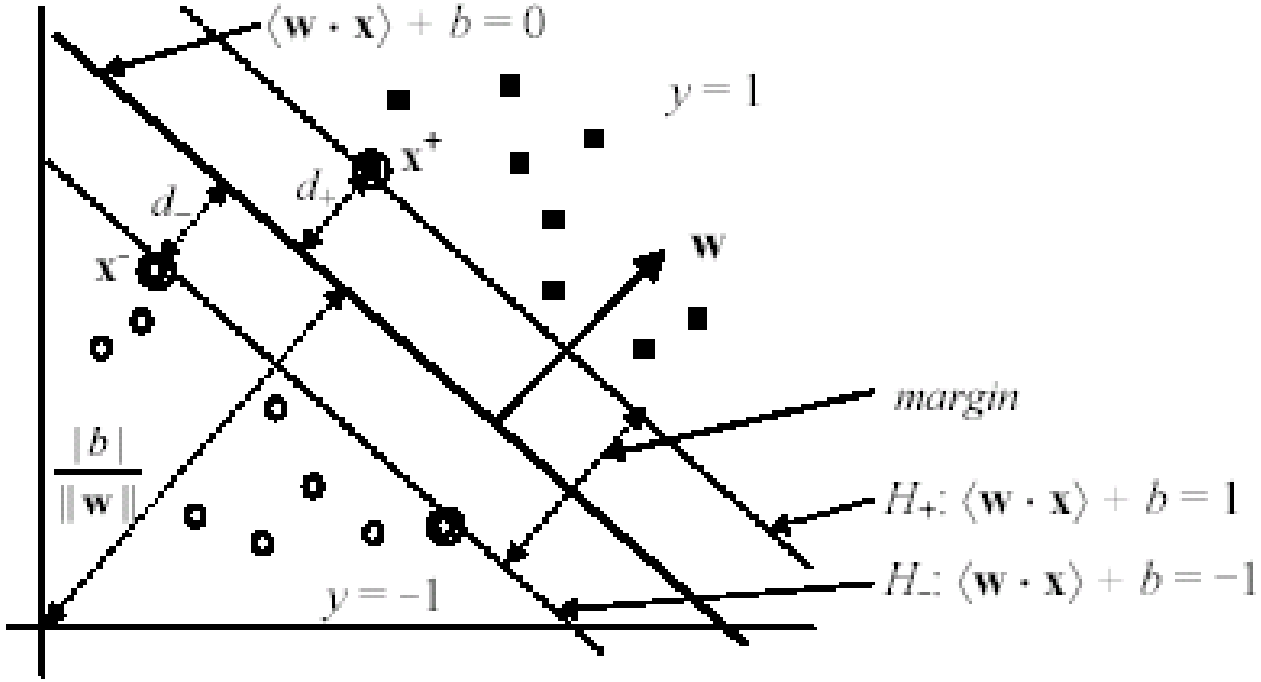
- Hiperplanul de decizie care separă cele 2 clase este:
  - $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$
- Pot exista mai multe hiperplanuri
  - Care este cel mai bun?





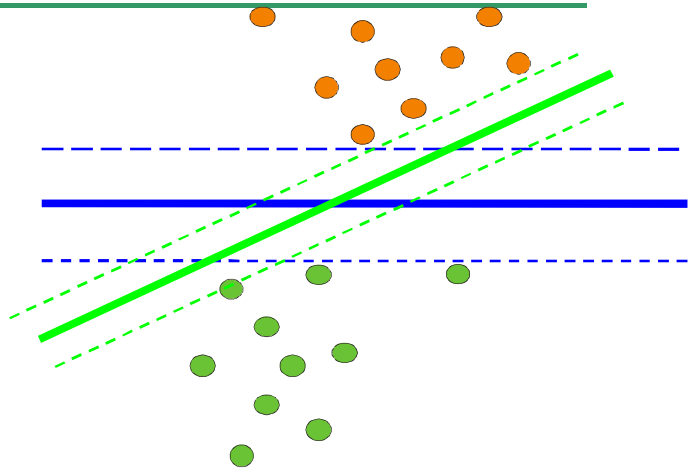
Învățare supervizată – algoritmi
Maşini cu suport vectorial

- MSV caută hiperplanul cu cea mai largă margine (cel care micşorează eroarea de generalizare)
  - Algoritmul SMO (*Sequential minimal optimization*)

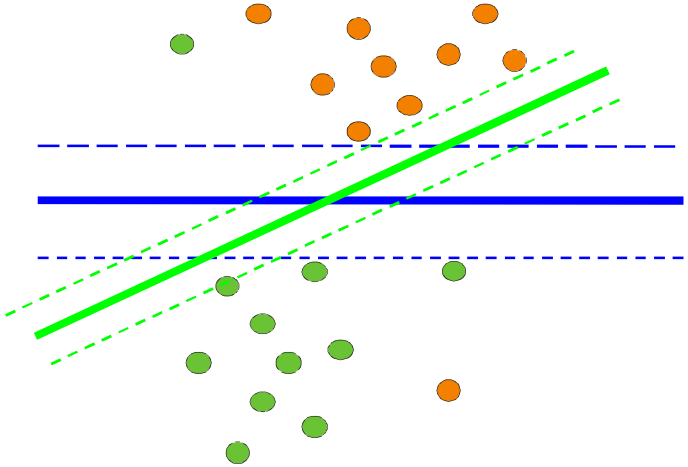


Învățare supervizată – algoritmi
Maşini cu suport vectorial

- Cazuri de date
  - Liniar separabile
    - Separabile
      - Eroarea = 0



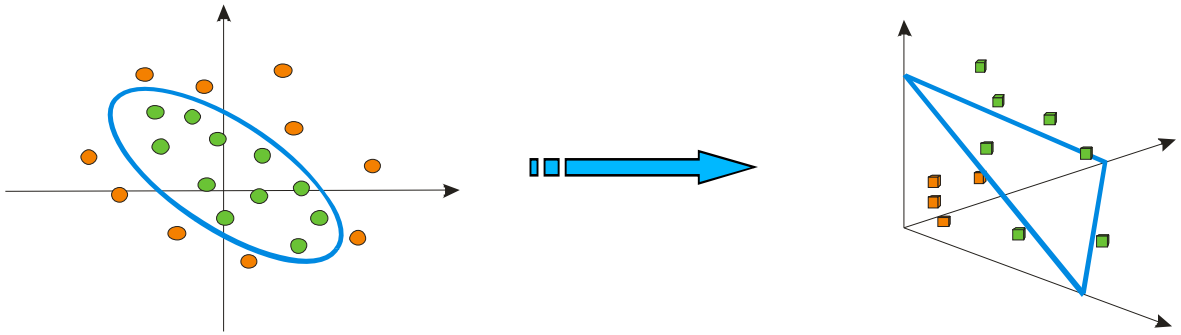
    - Ne-separabile
      - Se relaxează constrângerile → se permit unele erori
      - C – coeficient de penalizare



Învățare supervizată – algoritmi
Maşini cu suport vectorial

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: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_1 + c_1 * K_2 * exp(c_2 + K_3)$

Învățare supervizată – algoritmi
Maşini cu suport vectorial

- Probleme
  - 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/

Învățare supervizată – algoritmi

Regresie

- Studiul legăturii între variabile
- 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)
      - $\boldsymbol{atribute\_data_i} = (atr_{i1}, atr_{i2}, \ldots, atr_{im})$, m – nr atributelor (caracteristicilor, proprietăţilor) unei date
      - $ieşire_i$ – un număr real
    - date de test
      - sub forma ($\boldsymbol{atribute\_data_i}$), $i = 1,n$ (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

Învățare supervizată – algoritmi
Algoritmi evolutivi

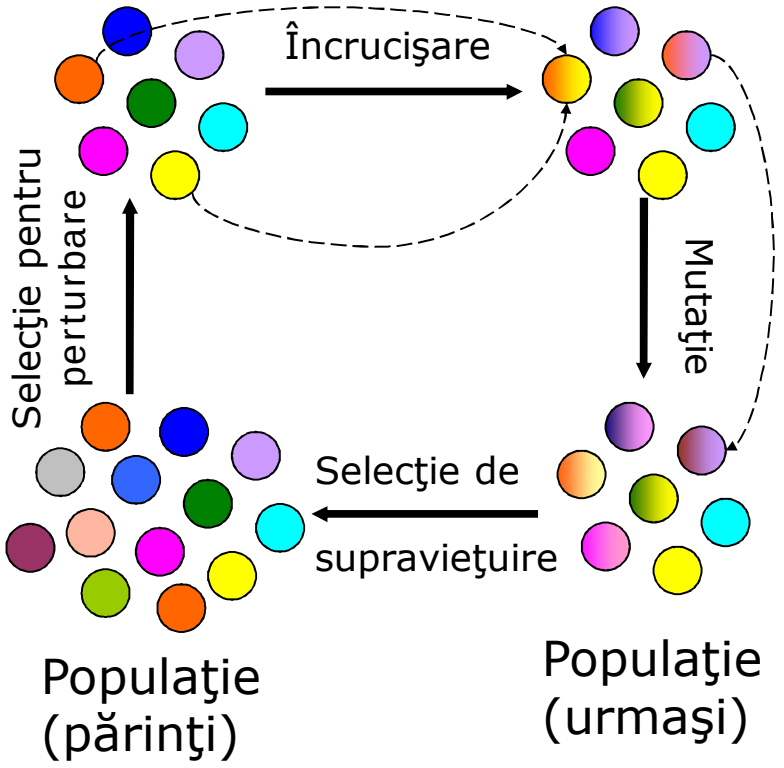
- Algoritmi
  - Inspiraţi din natură (biologie)
  - Iterativi
  - Bazaţi pe
    - populaţii de potenţiale soluţii
    - căutare aleatoare ghidată de
      - Operaţii de selecţie naturală
      - Operaţii de încrucişare şi mutaţie
  - Care procesează în paralel mai multe soluţii
- Metafora evolutivă

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
Încrucişare şi mutaţie	Operatori de căutare
Mediu	Spaţiul de căutare al problemei

Învățare supervizată – algoritmi

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
```



Învățare supervizată – algoritmi
Programare genetică

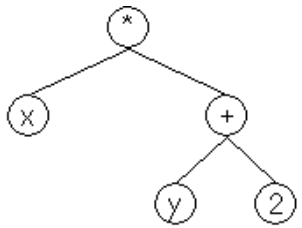
- Un tip particular de algoritmi evolutivi
- Cromozomi
  - sub formă de arbore care codează mici programe
- Fitness-ul unui cromozom
  - Performanţa programului codat în el

- http://www.genetic-programming.org/

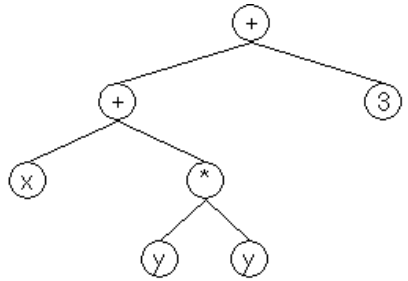
Învățare supervizată – algoritmi
Programare genetică

- Reprezentare
  - Cromozomul = un arbore cu noduri de tip
    - Funcţie → operatori matematici (+,-,*,/,*sin*,*log*,...)
    - Terminal → atribute ale datelor problemei sau constante (*x*,*y*,*z*,*a*,*b*,*c*,...)
  - care codează expresia matematică a unei funcţii

$x(y+2)$



$x+y^2+3$



Învățare supervizată – algoritmi
Programare genetică

- Fitness
  - Eroarea de predicţie
  - pp următoarele date de intrare (2 atribute şi o ieşire) şi 2 cromozomi:
    - $c_1 = 3x_1 - x_2 + 5$
    - $c_2 = 3x_1 + 2x_2 + 2$

$f^*(x1,x2) = 3x_1 + 2x_2 + 1$ – necunoscută

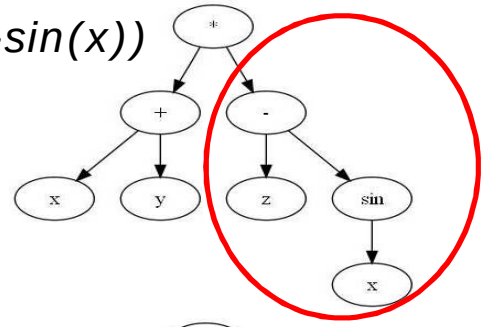
x_1	x_2	$f^*(x_1,x_2)$	$f_1(x_1,x_2)$	$f_2(x_1,x_2)$	$\|f^* - f_1\|$	$\|f^* - f_2\|$
1	1	6	7	7	1	1
0	1	3	4	4	1	1
1	0	4	8	5	4	1
-1	1	0	1	1	1	1
					$\Sigma = 7$	$\Sigma = 4$

➔ c_2 e mai bun ca c_1

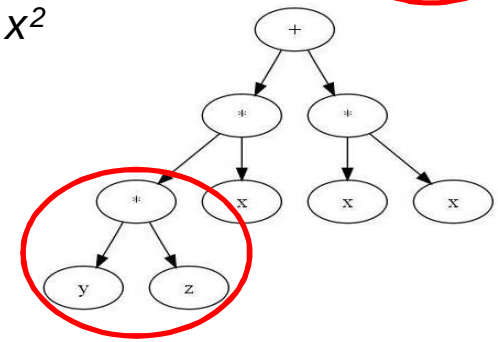
Învățare supervizată – algoritmi
Programare genetică

- Iniţializare
  - Generare aleatoare de arbori corecţi → expresii matematice valide
- Încrucişare
  - Cu punct de tăietură – se interchimbă doi sub-arbori

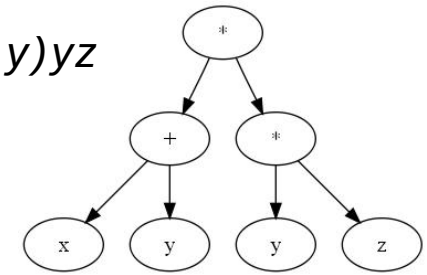
$p_1 = (x+y)*(z-sin(x))$



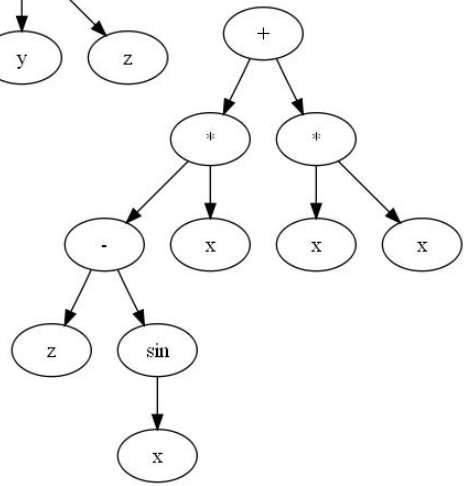
$p_2 = xyz + x^2$



$f_1 = (x+y)yz$



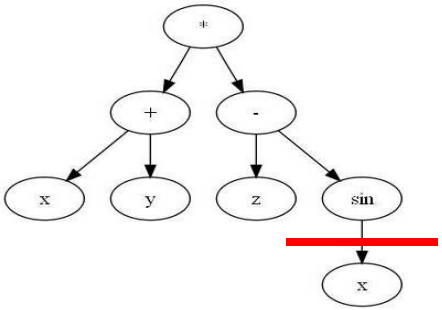
$f_2 = (z-sin(x))x + x^2$



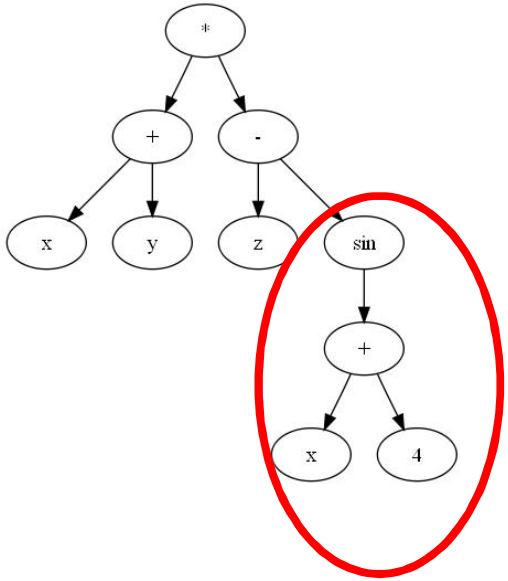
Învățare supervizată – algoritmi
Programare genetică

- Mutaţie
  - Generarea unui nou sub-arbore

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



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



Învățare automată

- Învăţare supervizată
- **Învăţare ne-supervizată**
- Învăţare cu întărire
- Teoria învăţării

Învățare ne-supervizată

- Definire
- Exemple
- Proces
- Metode de evaluare şi măsuri de performanţă
- Tipologie
- Algoritmi

Învățare ne-supervizată – definire

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

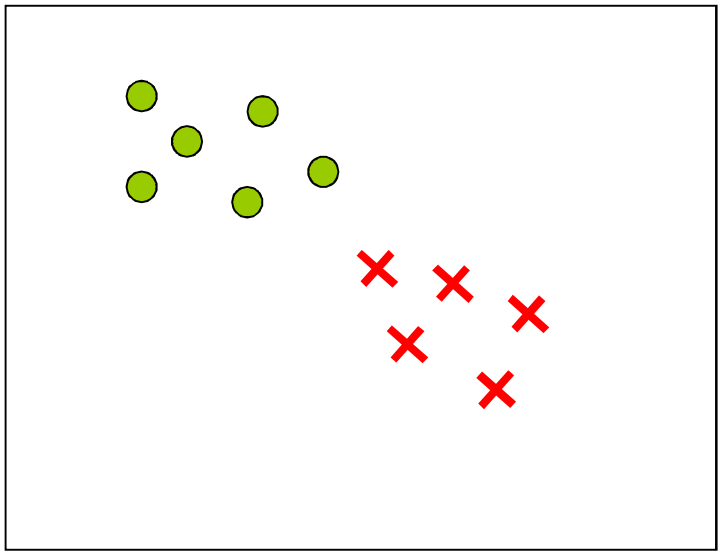
- Se dă
  - un set de date (exemple, instanţe, cazuri)
    - Date de antrenament
      - Sub forma $\boldsymbol{atribute\_data_i}$, unde
        - $i = 1,N$ (N = nr datelor de antrenament)
        - $\boldsymbol{atribute\_data_i} = (atr_{i1}, atr_{i2}, \ldots, atr_{im})$, m – nr atributelor (caracteristicilor, proprietăţilor) unei date
    - Date de test
      - Sub forma ($\boldsymbol{atribute\_data_i}$), $i = 1,n$ (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

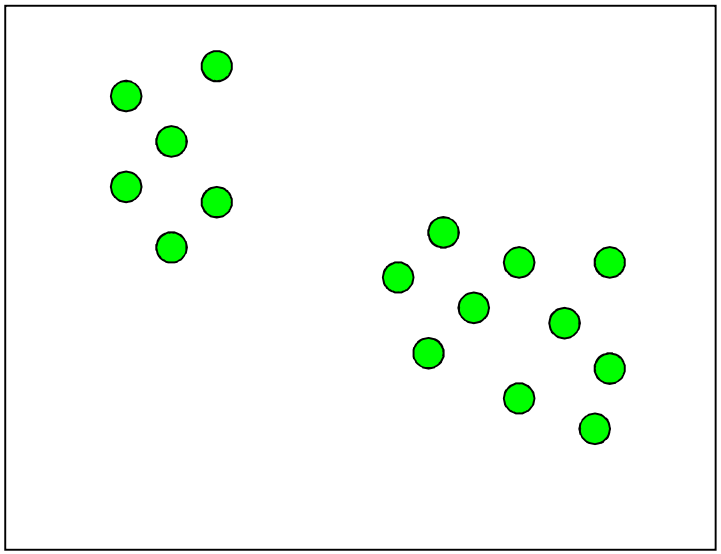
Alte denumiri

- Clustering

Învățare ne-supervizată – definire

- Supervizată *vs.* Ne-supervizată





Învățare ne-supervizată – definire

- Distanţe între 2 elemente ***p*** şi ***q*** ϵR^m
  - Euclideana
    - $d(\boldsymbol{p},\boldsymbol{q}) = sqrt(\sum_{j=1,2,\ldots,m}(p_j - q_j)^2)$
  - Manhattan
    - $d(\boldsymbol{p},\boldsymbol{q}) = \sum_{j=1,2,\ldots,m}|p_j - q_j|$
  - Mahalanobis
    - $d(\boldsymbol{p},\boldsymbol{q}) = sqrt((\boldsymbol{p}-\boldsymbol{q})S^{-1}(\boldsymbol{p}-\boldsymbol{q}))$,
      - unde *S* este matricea de variaţie şi covariaţie ($S = E[(\boldsymbol{p}-E[\boldsymbol{p}])(\boldsymbol{q}-E[\boldsymbol{q}])]$)
  - Produsul intern
    - $d(\boldsymbol{p},\boldsymbol{q}) = \sum_{j=1,2,\ldots,m} p_j q_j$
  - Cosine
    - $d(\boldsymbol{p},\boldsymbol{q}) = \sum_{j=1,2,\ldots,m} p_j q_j \,/\, (sqrt(\sum_{j=1,2,\ldots,m} p_j^2) * sqrt(\sum_{j=1,2,\ldots,m} q_j^2))$
  - 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

Învățare ne-supervizată – exemple

- Gruparea genelor
- Studii de piaţă pentru gruparea clienţilor (segmentarea pieţei)
- news.google.com

Învățare ne-supervizată – proces

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

Calitatea învăţării (validarea clusterizări):

- Criterii interne
  - Similaritate ridicată în interiorul unui cluster şi similaritate redusă între clusteri
- Criteri externe
  - Folosirea unor benchmark-uri formate din date pre-grupate

Învățare ne-supervizată – evaluare

Măsuri de performanţă

- Criterii interne
  - Distanţa în interiorul clusterului
  - Distanţa între clusteri
  - Indexul Davies-Bouldin
  - Indexul Dunn

- Criteri externe
  - Compararea cu date cunoscute – în practică este imposibil
  - Precizia
  - Rapelul
  - F-measure

Învățare ne-supervizată – evaluare

Măsuri de performanţă

- Criterii interne
  - Distanţa în interiorul clusterului c_j care conţine n_j instanţe
    - Distanţa medie între instanţe (average distance)
      - $D_a\ (c_j) = \sum_{xi1,xi2 \epsilon cj} ||x_{i1} - x_{i2}|| / (n_j(n_j\text{-}1))$
    - Distanţa între cei mai apropiaţi vecini (nearest neighbour distance)
      - $D_{nn}\ (c_j) = \sum_{xi1 \epsilon cj} min_{\ xi2 \epsilon cj} ||x_{i1} - x_{i2}|| / n_j$
    - Distanţa între centroizi
      - $D_c\ (c_j) = \sum_{xi,\epsilon cj} ||x_i - \mu_j|| / n_j$, $unde\ \mu_j = 1/ n_j \sum_{xi \epsilon cj} x_i$

Învățare ne-supervizată – evaluare

Măsuri de performanţă

- Criterii interne
  - 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
      - $d_{ce}(c_{j1}, c_{j2}) = ||\mu_{j1} - \mu_{j2}||$

Învățare ne-supervizată – evaluare

Măsuri de performanţă

- Criterii interne
  - Indexul Davies-Bouldin → min ➔ clusteri compacţi
    - $DB = 1/nc * \sum_{i=1,2,\ldots,nc} max_{j=1,2,\ldots,nc, j \neq i}((\sigma_i + \sigma_j)/d(\mu_i, \mu_j))$
    - unde:
      - nc – numărul de clusteri
      - μ_i – centroidul clusterului i
      - σ_i – media distanţelor între elementele din clusterul i şi centroidul μ_i
      - $d(\mu_i, \mu_j)$ – distanţa între centroidul μ_i şi centroidul μ_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

Învățare ne-supervizată - tipologie

- După modul de formare al clusterilor
  - C. ierarhic
  - C. ne-ierarhic (partiţional)
  - C. bazat pe densitatea datelor
  - C. bazat pe un grid

Învățare 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

Învățare ne-supervizată - tipologie

- După modul de formare al clusterilor
  - Ne-ierarhic
    - Partiţional → se determină o împărţire a datelor → toţi clusterii deodată
    - Optimizează o funcţie obiectiv definită
      - Local – doar pe anumite atribute
      - 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*

Învățare ne-supervizată - tipologie

- După modul de formare al clusterilor
  - 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

Învățare ne-supervizată - tipologie

- După modul de formare al clusterilor
  - 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

Învățare 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

Învățare ne-supervizată - tipologie

- După tipul de apartenenţă al datelor la clusteri
  - Clustering exact (*hard clustering*)
    - Asociază fiecarei intrări $\boldsymbol{x_i}$ o etichetă (clasă) c_j
  - Clustering fuzzy
    - 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

Învățare ne-supervizată – algoritmi

- Clustering ierarhic aglomerativ
- K-means
- AMA
- Modele probabilistice
- Cel mai apropiat vecin
- Fuzzy
- Reţele neuronale artificiale
- Algoritmi evolutivi
- ACO

Învățare ne-supervizată – algoritmi
Clustering ierarhic aglomerativ

- Se consideră o distanţă între 2 instanţe $d(x_{i1}, x_{i2})$
- Se formează *N* clusteri, fiecare conţinând câte o instanţă
- Se repetă
  - Determinarea celor mai apropiaţi 2 clusteri
  - Se reunesc cei 2 clusteri ➔ un singur cluster
- Până când se ajunge la un singur cluster (care conţine toate instanţele)

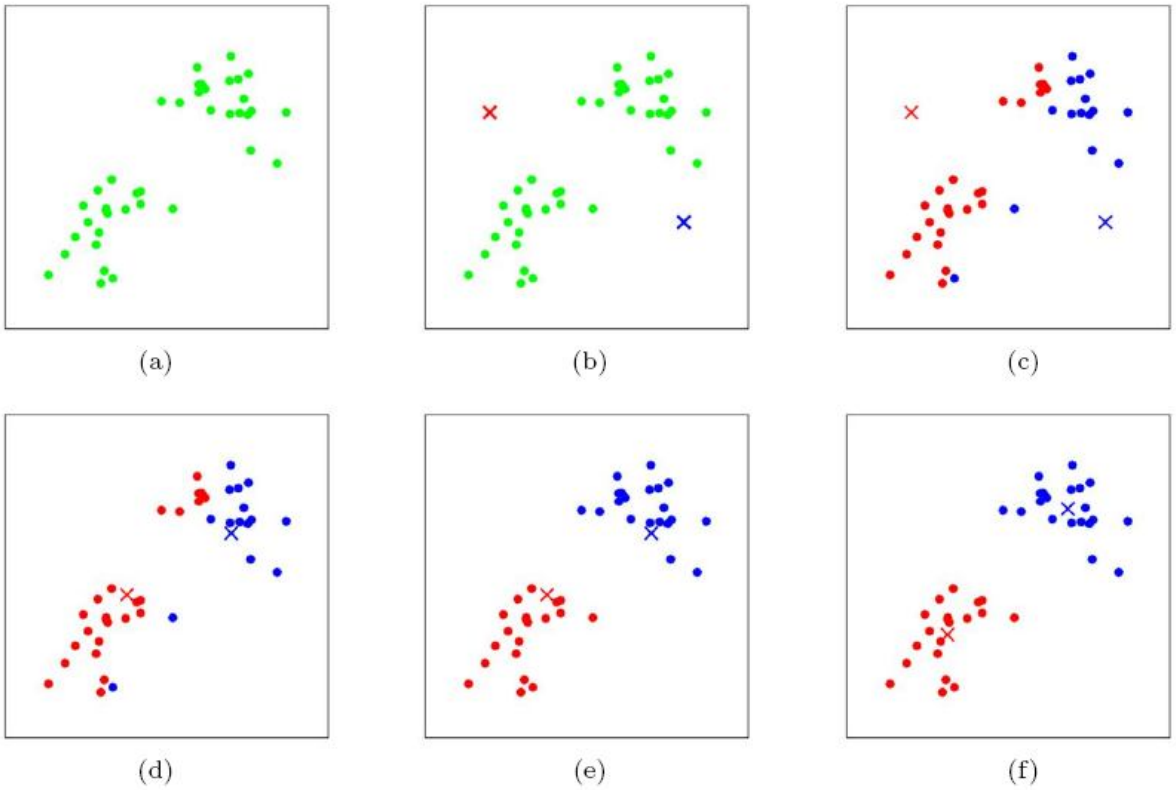
Învățare ne-supervizată – algoritmi
Clustering ierarhic agloemrativ

- Distanţa între 2 clusteri c_i şi c_j:
  - Legătură simplă → minimul distanţei între obiectele din cei 2 clusteri
    - $d(c_i, c_j) = max_{xi1 \epsilon ci,\ xi2 \epsilon cj}\ sim(\boldsymbol{x_{i1}}, \boldsymbol{x_{i2}})$
  - Legătură completă → maximul distanţei între obiectele din cei 2 clusteri
    - $d(c_i, c_j) = min_{xi1 \epsilon ci,\ xi2 \epsilon cj}\ sim(\boldsymbol{x_{i1}}, \boldsymbol{x_{i2}})$
  - Legătură medie → media distanţei între obiectele din cei 2 clusteri
    - $d(c_i, c_j) = 1 / (n_i * n_j) \sum_{xi1 \epsilon ci} \sum_{xi2 \epsilon cj} d(\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2})$
  - Legătură medie peste grup → distanţa între mediile (centroizii) celor 2 clusteri
    - $d(c_i, c_j) = \rho(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$, ρ – *distanţă*, $\boldsymbol{\mu}_j = 1/n_j \sum_{xi \epsilon cj} \boldsymbol{x}_i$

Învățare ne-supervizată – algoritmi K-means (algoritmul Lloyd/iterația Voronoi)

- Pp că se vor forma k clusteri
- Iniţializează *k* centroizi $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$
  - Un centroid $\boldsymbol{\mu}_j$ ($i=1,2, \ldots, k$) este un vector cu *m* valori (*m* – nr de atribute)
- Repetă până la convergenţă
  - Asociază fiecare instanţă celui mai apropiat centroid → pentru fiecare instanţă $\boldsymbol{x}_i$, $i = 1, 2, \ldots, N$
    - $c_i = arg\ min_{j = 1, 2, \ldots, k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2$
  - Recalculează centroizii prin mutarea lor în media instanţelor asociate fiecăruia → pentru fiecare cluster c_j, $j = 1, 2, \ldots, k$
    - $\boldsymbol{\mu}_j = \Sigma_{i=1,2, \ldots N} 1_{ci=j}\ \boldsymbol{x}_i \ / \ \Sigma_{i=1,2, \ldots N}\ 1_{ci=j}$

Învățare ne-supervizată – algoritmi K-means



Învățare ne-supervizată – algoritmi K-means

- Iniţializarea a k centroizi $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$
  - Cu valori generate aleator (în domeniul de definiţie al problemei)
  - Cu k dintre cele N instanţe (alese în mod aleator)

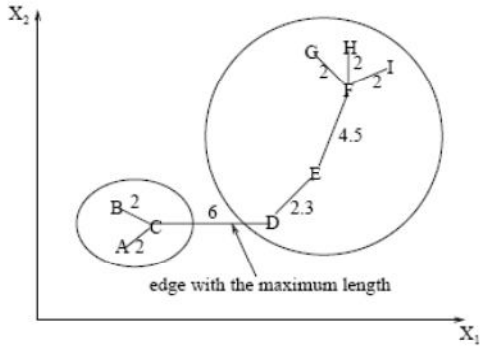
- Algoritmul converge întotdeauna?
  - Da, pt că avem funcţia de distorsiune J
    - $J(c, \mu) = \Sigma_{i=1,2,\ldots,N} ||\boldsymbol{x}_i - \boldsymbol{\mu}_{cj}||^2$

    care este descrescătoare
  - Converge într-un optim local
  - Găsirea optimului global → NP-dificilă

Învățare ne-supervizată – algoritmi

Clusterizare bazată pe arborele minim de acoperire (AMA)

- Se construieşte AMA al datelor
- Se elimină din arbore cele mai lungi muchii, formânduse clusteri



Învățare ne-supervizată – algoritmi
Modele probabilistice

- http://www.gatsby.ucl.ac.uk/~zoubin/course04/ul.pdf
- http://learning.eng.cam.ac.uk/zoubin/nipstut.pdf

Învățare ne-supervizată – algoritmi
Cel mai apropiat vecin

- Se etichetează câteva dintre instanţe
- Se repetă până la etichetarea tuturor instanţelor
  - O instanţă ne-etichetată va fi inclusă în clusterul instanţei cele mai apropiate
    - dacă distanţa între instanţa neetichetată şi cea etichetată este mai mică decât un prag

Învățare ne-supervizată – algoritmi
Clusterizare fuzzy

- Se stabileşte o partiţionare fuzzy iniţială
  - Se construieşte matricea gradelor de apartenenţă U, unde u_{jj} – gradul de apartenenţă al instanţei $\boldsymbol{x_i}$ ($i=1,2, \ldots, N$) la clusterul c_j ($j = 1, 2, \ldots, k$) ($u_{ij} \in [0,1]$)
    - Cu cât u_{ij} e mai mare, cu atât e mai mare încrederea că instanţa $\boldsymbol{x_i}$ face parte din clusterul c_j

- Se stabileşte o funcţie obiectiv
  - $E^2(U) = \sum_{i=1,2, \ldots, N} \sum_{j=1,2,\ldots,k} u_{ij} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2$,
    - unde $\boldsymbol{\mu}_j = \sum_{i=1,2, \ldots, N} u_{ij} \boldsymbol{x}_i$ – centrul celui de-al j-lea fuzzy cluster
  - care se optimizează (min) prin re-atribuirea instanţelor (în clusteri noi)

- Clusering fuzzy ➔ clusterizare *hard* (fixă)
  - impunerea unui prag funcţiei de apartenenţă u_{ij}

Învățare ne-supervizată – algoritmi
Algoritmi evolutivi

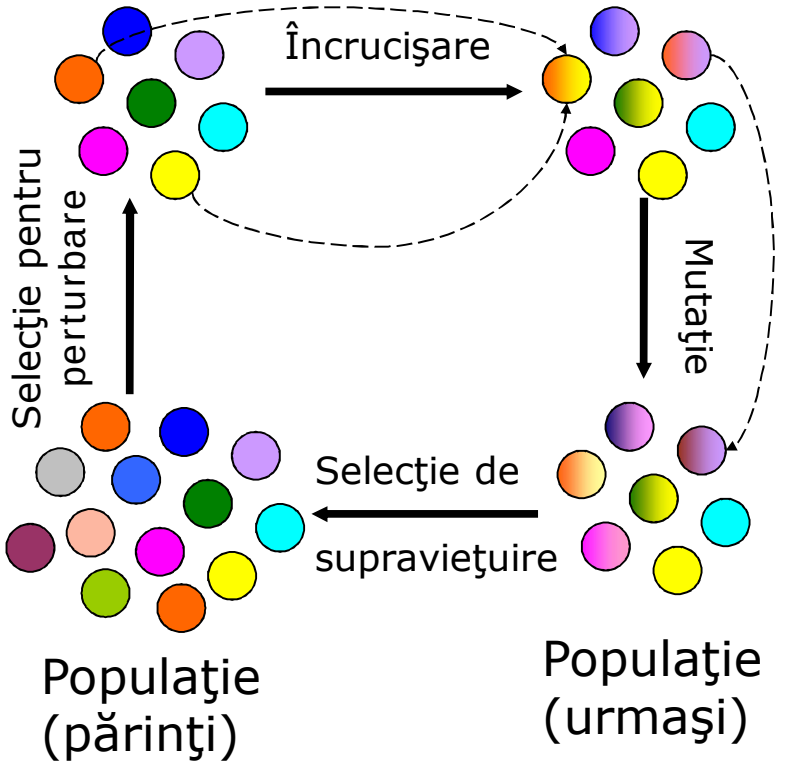
- Algoritmi
  - Inspiraţi din natură (biologie)
  - Iterativi
  - Bazaţi pe
    - populaţii de potenţiale soluţii
    - căutare aleatoare ghidată de
      - Operaţii de selecţie naturală
      - Operaţii de încrucişare şi mutaţie
  - Care procesează în paralel mai multe soluţii
- Metafora evolutivă

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
Încrucişare şi mutaţie	Operatori de căutare
Mediu	Spaţiul de căutare al problemei

Învățare ne-supervizată – algoritmi

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
```

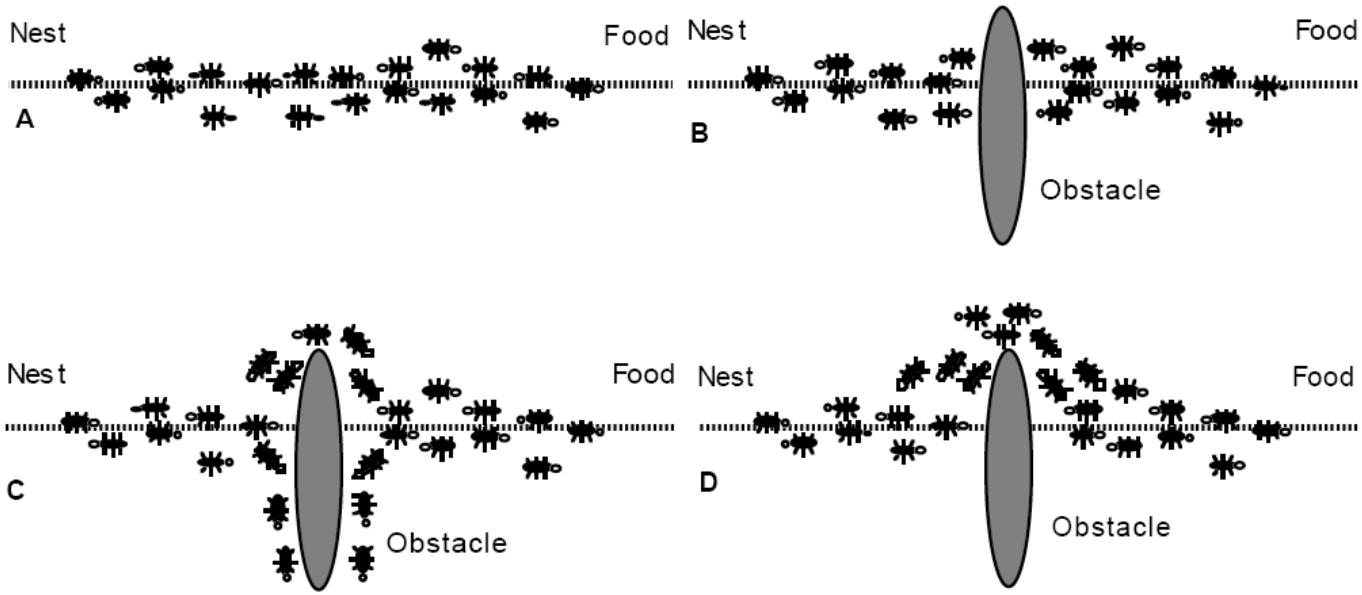


Învățare ne-supervizată – algoritmi
Algoritmi evolutivi

- Reprezentare
  - Cromozomul = o partiţionare a datelor
    - Ex. 2 clusteri ➔ cromozom = vector binar
    - Ex. *K* clusteri ➔ cromozom = vector cu valori din {*1,2,…,k*}
- Fitness
  - Calitatea partiţionării
- Iniţializare
  - Aleatoare
- Încrucişare
  - Punct de tăietură
- Mutaţie
  - Schimbarea unui element din cromozom

Învățare ne-supervizată – algoritmi ACO

- Preferinţa pentru drumuri cu nivel ridicat de feromon
- Pe drumurile scurte feromonul se înmulţeşte
- Furnicile comunică pe baza urmelor de feromon



Învățare ne-supervizată – algoritmi ACO

- Algoritm de clusterizare bazat pe un grid
- Obiectele se plasează aleator pe acest grid, urmând ca furnicuţele să le grupeze în funcţie de asemănarea lor
- 2 reguli pentru furnicuţe
  - Furnica “ridică” un obiect-obstacol
    - Probabilitatea de a-l ridica e cu atât mai mare cu cât obiectul este mai izolat (în apropierea lui nu se află obiecte similare)
    - $p(ridica)=(k^+/(k^+ +f))^2$
  - Furnica “depune” un obiect (anterior ridicat) într-o locaţie nouă
    - Probabilitatea de a-l depune e cu atât mai mare cu cât în vecinătatea locului de plasare se afla mai multe obiecte asemănătoare
    - $p(depune)=(f/(k^- +f))^2$
  - k^+, k^- - constante
  - f – procentul de obiecte similare cu obiectul curent din memoria furnicuţei
- Furnicuţele
  - au memorie
    - reţin obiectele din vecinătatea poziţiei curente
  - se mişcă ortogonal (N, S, E, V) pe grid pe căsuţele neocupate de alte furnici

Învățare automată

- Învăţare supervizată
- Învăţare ne-supervizată
- **Învăţare cu întărire**
- Teoria învăţării

Învățare cu întărire

- Definire
- Exemple
- Proces
- Metode de evaluare şi măsuri de performanţă
- Tipologie
- Algoritmi

Învățare cu întărire – definire

Învăţarea unui anumit comportament în vederea realizării unei sarcini ➔ execuţia unei acţiuni ➔ primeşte un feedback (cât de bine a acţionat pentru îndeplinirea sarcinii) ➔ execuţia unei noi acţiuni

Învăţare cu întărire

- Se primeşte o recompensă (întărire pozitivă) – dacă sarcina a fost bine îndeplinită
- Se primeşte o pedeapsă (întărire negativă) – dacă sarcina nu a fost bine îndeplinită

Definire

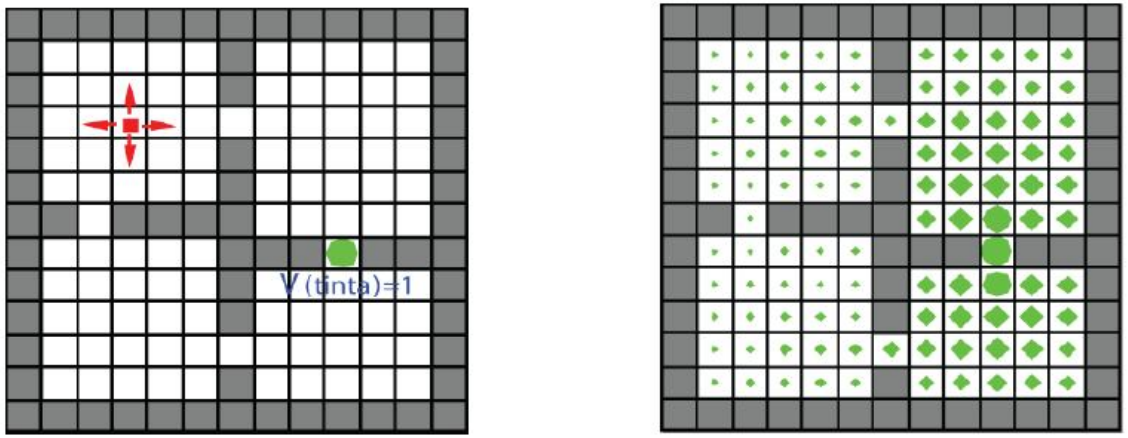
- Se dau
  - Stări ale mediului
  - Acţiuni posibile de executat
  - Semnale de întărire (scalare) – recompense sau pedepse
- Se determină
  - O succesiune de acţiuni care să maximizeze măsura de întărire (recompensa)

Alte denumiri

- Reinforcement learning
- Învăţare împrospătată

Învățare ne-supervizată – definire

- Exemplu: plecând din căsuţa roşie să se găsească un drum până la căsuţa verde



- Agentul învaţă prin interacţiunea cu mediul şi prin observarea rezultatelor obţinute din aceste interacţiuni
  - Este vorba de "cauză şi efect" -- modul în care oamenii îşi formează cunoaşterea aupra mediului pe parcursul vieţii
  - Acţiunile pot afecta şi recompensele ulterioare, nu numai pe cele imediate (efect întârziat)

Învățare cu întărire – definire

- Învăţare supervizată
  - Învăţarea pe baza unor exemple oferite de un expert extern care deţine o bază importantă de cunoştinţe
- Învăţare cu întărire
  - Învăţarea pe baza interacţiunii cu mediul

Învățare cu întărire – exemple

- Robotică
  - Controlul membrelor
  - Controlul posturii
  - Preluarea mingii în fotbalul cu roboţii
- Cercetări operaţionale
  - Stabilirea preţurilor
  - Rutare
  - Planificarea task-urilor

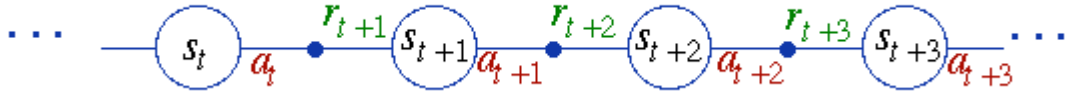
Învățare cu întărire – proces

Procesul

- Agentul observă o stare de intrare
- Agentul alege o acţiune pe baza unei funcţii de decizie (o **strategie**)
- Agentul execută acţiunea aleasă
- Agentul primeşte o recompensă/pedeapsă numerică de la mediu
- Agentul reţine recompensa/pedeapsa primită

Învățare cu întărire – proces

- Mediul este modelat ca un proces de decizie de tip Markov
  - *S* – mulţimea stărilor posibile
  - *A(s)* – acţiuni posibile în starea *s*
  - *P(s, s', a)* – probabilitatea de a trece din starea *s* în starea *s'* prin acţiunea *a*
  - *R(s, s', a)* – recompensa aşteptată în urma trecerii din starea *s* în starea *s'*prin acţiunea *a*
  - γ – rata de discount pentru recompensele întârziate



- Obiectivul
  - Găsirea unei politici $\pi : s \in S \rightarrow a \in A(s)$ care maximizează
    - valoarea (recompensa viitoare aşteptată) a unei stări
      - $V^{\pi}(s) = E\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s_t = s, \pi \}$
    - calitatea fiecărei perechi stare-acţiune
      - $Q^{\pi}(s,a) = E\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s_t = s, a_t = a, \pi \}$

Învățare cu întărire – evaluare

Măsuri de performanţă

- Recompensa acumulată pe parcursul învăţarii
- Numărul de paşi necesari învăţării

Învățare cu întărire – algoritmi

- Q-learning
  - Calitatea unei combinaţii stare-acţiune
- SARSA (*State-Action-Reward-State-Action*)

Învățare automată

- Învăţare supervizată
- Învăţare ne-supervizată
- Învăţare cu întărire
- **Teoria învăţării**