Calcul neuronal și calcul evolutiv

- Despre ce este vorba ?
- Calcul inteligent
- Calcul neuronal - specific
- Calcul evolutiv – specific
- Tehnici înrudite
- Structura cursului … aspecte organizatorice

Despre ce este vorba ?

- … despre rezolvarea problemelor (dificile)
- Există diferite clase de probleme dificile:
  - Dificile pentru calculatoare dar ușoare pentru oameni (recunoașterea vorbirii, recunoașterea imaginilor, recunoașterea caracterelor etc)
  - Dificile atât pentru calculatoare cât și pentru oameni (probleme de optimizare combinatorială de dimensiuni mari, probleme de optimizare neliniară, probleme de planificare etc)

Probleme dificile

- Problemele dificile atât pentru calculatoare cât și pentru oameni sunt cele caracterizate printr-un spațiu mare de căutare și prin faptul că nu se cunosc algoritmi care să permită rezolvarea lor într-un interval timp a cărui dimensiune depinde polinomial de dimensiunea problemei (probleme NP dificile)

Exemple clasice:

- Problema satisfiabilității (SAT): determinarea valorilor de adevăr ale unor variabile pentru care o expresie logică este adevarată. Pentru n variabile spațiul de căutare are dimensiunea 2^n
- Problema comis voiajorului (TSP): determină un circuit de cost minim care vizitează n locații. Dimensiunea spațiului de căutare este (n-1)!

  (în cazul problemelor simetrice este (n-1)!/2)

Probleme dificile

- Problemele dificile pentru calculatoare dar mai ușoare pentru oameni sunt cele “rău-puse”, adică cele pentru care este dificil de identificat un model abstract care să reflecte toate particularitățile problemei

- Sa considerăm următoarele două probleme:
  - clasificarea angajatilor unei firme în două categorii în funcție de valoarea venitului: cei care au peste venitul mediu (din cadrul firmei) într-o categorie și cei care au sub venitul mediu în altă categorie

  - clasificarea angajaților unei firme în două categorii în funcție de credibilitatea relativ la acordarea unui împrumut

Probleme dificile

- In cazul primei probleme este ușor să se identifice un model formal (o regulă de clasificare):

  IF venit > venit_mediu THEN Class 1
  ELSE Class 2

- In cazul celei de a doua probleme lucrurile sunt mai complicate întrucât trebuie luați în calcul mai mulți factori intercorelați (situație financiară, stare de sănătate, situație familială, perspective în carieră etc.). Un expert bancar poate rezolva o astfel de problemă bazându-se pe experiența dobândită de-a lungul timpului precum și pe elemente subiective dificil de cuantificat

Probleme dificile

- Diferențe între probleme bine-puse și probleme rău-puse

Problema bine-pusă:

- i se poate asocia un model formal
- există algoritm de rezolvare

Problema rău-pusă:

- nu este ușor de formalizat
- există doar exemple de rezolvare
- datele despre problema pot fi incomplete sau inconsistente
- metodele clasice sunt inaplicabile

Probleme dificile

Metodele de rezolvare a problemelor rău-puse trebuie să se caracterizeze prin:

- Abilitatea de a extrage modele din exemple
- Adaptabilitate la modificari în proprietățile problemei (probleme cu caracter dinamic)
- Robustețe la erori sau zgomot în datele de intrare
- Capacitate de a produce rezultatul folosind un volum rezonabil de resurse (timp de calcul, spațiu memorie)

Domeniul ce se ocupă cu dezvoltarea acestor metode este denumit uneori “computational intelligence” iar alteori “soft computing”

Calcul inteligent

Computational Intelligence

"is a branch of the study of artificial intelligence; it aims to use learning, adaptive, or evolutionary algorithms to create programs that are, in some sense, intelligent. " [Wikipedia, 2012]

"Studiul mecanismelor adaptive care permit simularea comportamen-tului inteligent în medii complexe sau cu caracter dinamic"

Terminologie:

Calcul inteligent ?

Inteligența computațională ?

Calcul tolerant la erori ?

Computational Intelligence

"is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which traditional approaches, i.e., first principles modeling or explicit statistical modeling, are ineffective or infeasible. It primarily includes artificial neural networks, evolutionary computation and fuzzy logic." [Wikipedia - 2014]

Calcul inteligent

Soft Computing

"is a term applied to a field within computer science which is characterized by the use of inexact solutions to computationally hard tasks such as the solution of NP-complete problems, for which there is no known algorithm that can compute an exact solution in polynomial time. Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. In effect, the role model for soft computing is the human mind." [Wikipedia - 2014]

Soft Computing

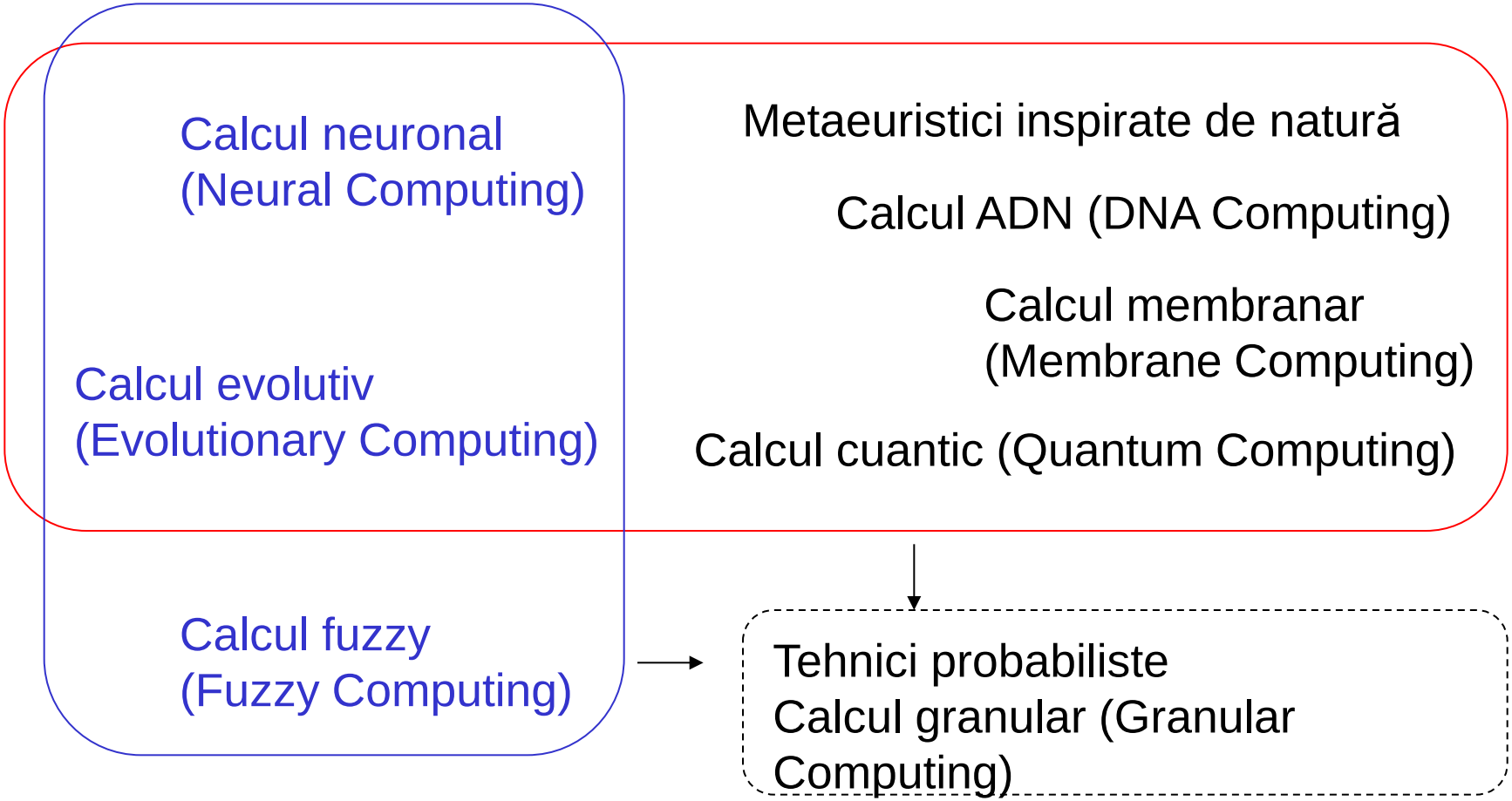
"is a collection of new techniques in computer science, especially in artificial intelligence; unlike hard computing, it is tolerant of imprecision, uncertainty and partial truth. In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost." [Wikipedia - 2012]

Calcul inteligent

Direcții de lucru:	Instrument:	Sursa de inspirație:
Calcul neuronal	Rețea neuronală artificială	Creierul uman
Calcul evolutiv	Algoritm evolutiv	Procesele de evoluție din natură
Calcul fuzzy	Sistem fuzzy	Specificul raționamentului uman și a limbajului natural

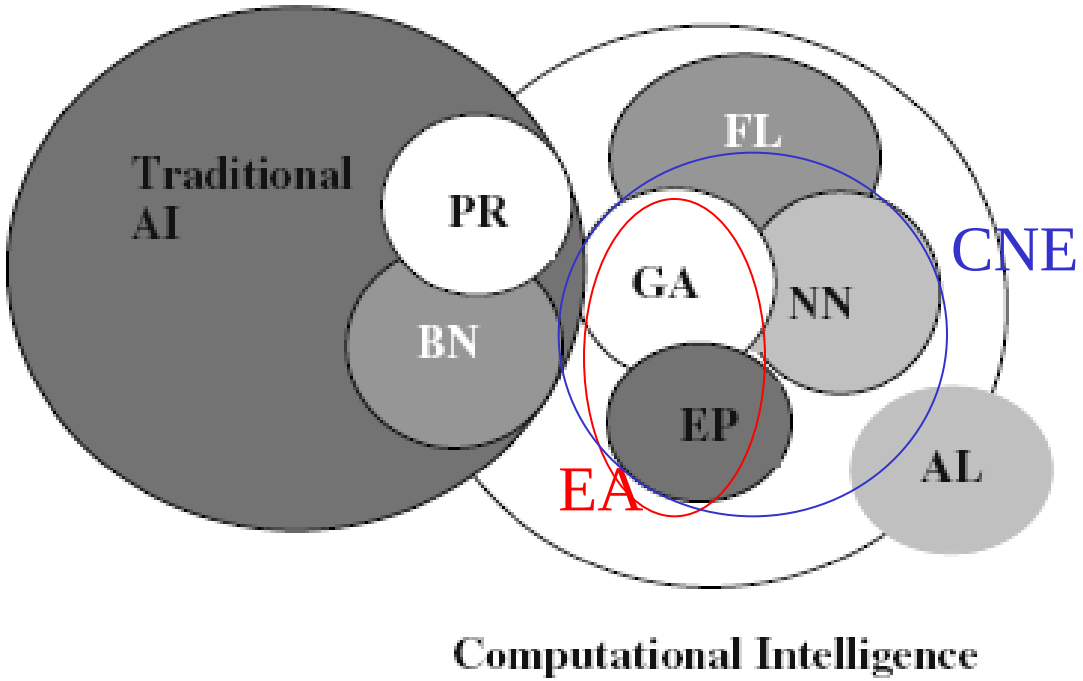
Calcul inteligent

Calcul natural = metode de calcul inspirate de natură



Calcul inteligent

[A. Konar – Computational Intelligence, 2007]



PR= Probabilistic reasoning, BN= Belief networks, FL= Fuzzy logic, NN= Neural nets, GA= Genetic algorithms, EP= Evolutionary programming, AL= Artificial life.

Calcul neuronal - specific

Principiile calculului neuronal

Modelul biologic

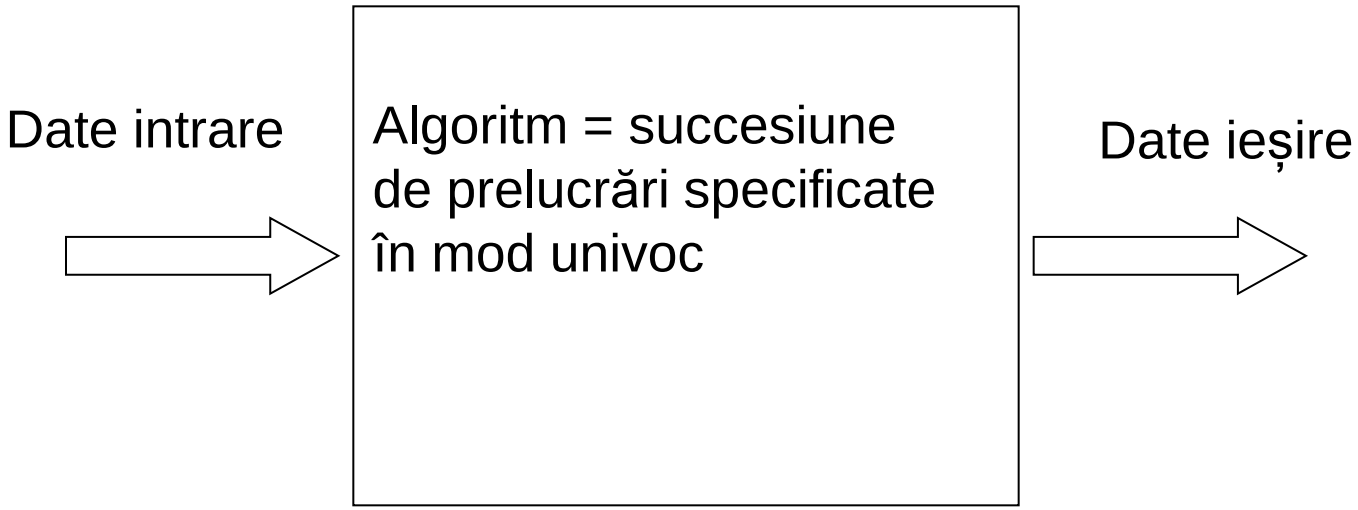
Structura unei rețele neuronale

Tipuri de rețele neuronale

Aplicații ale calculului neuronal

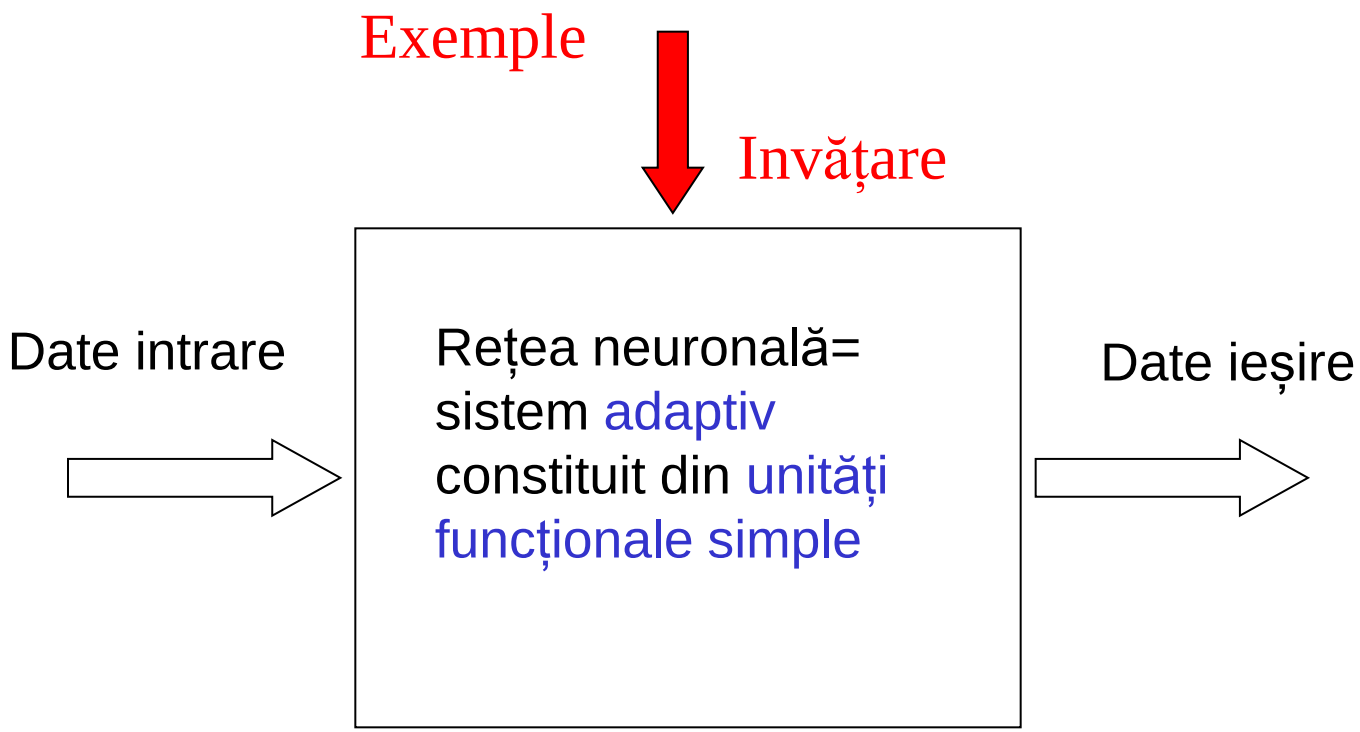
Principiile calculului neuronal

Rezolvarea in manieră clasică a unei probleme



Principiile calculului neuronal

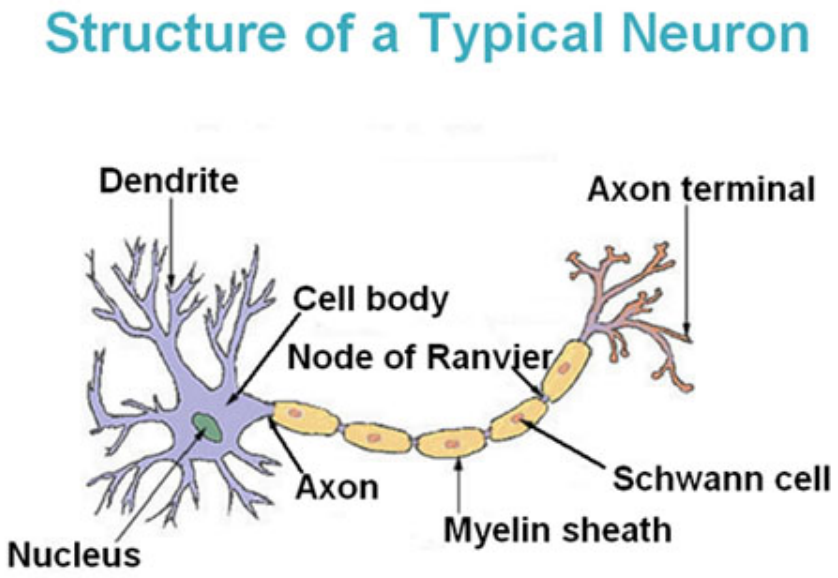
Utilizarea unei rețele neuronale (sau a oricărui model bazat pe învățare automată)

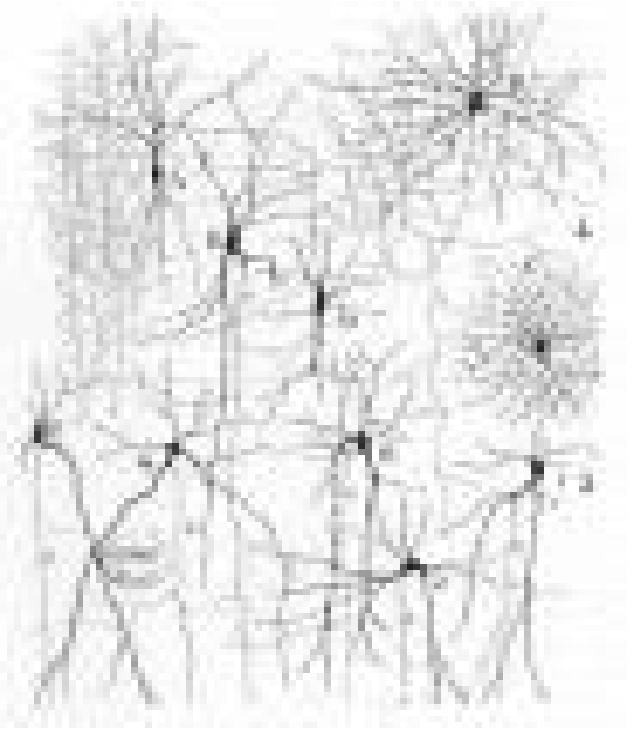


Calcul neuronal – modelul biologic

Creierul uman

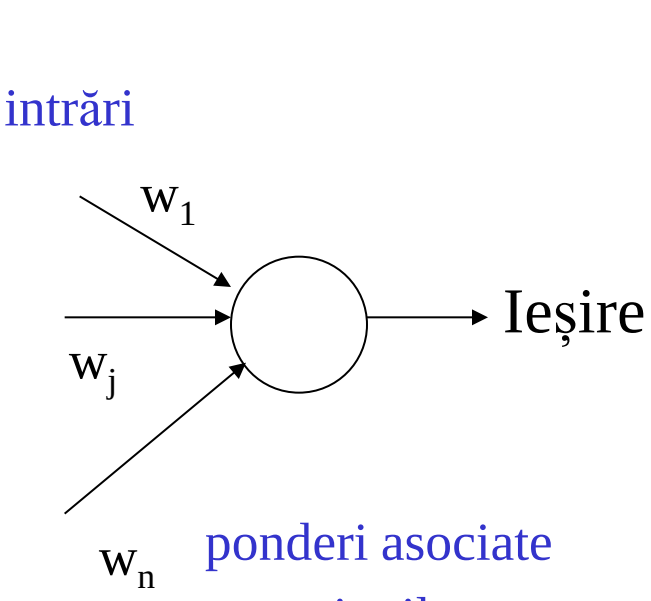
cca 10^{10} neuroni, cca 10^{14} interconexiuni





Structura unei rețele neuronale

Rețea neuronală artificială = ansamblu de unități simple de prelucrare (neuroni) interconectate



$$y = f\left(\sum_{j=1}^{n} w_j x_j - w_0\right)$$

$$u = \sum_{j=1}^{n} w_j x_j - w_0$$

Funcție de agregare a intrărilor

$$f(u) = \mathrm{sgn}(u) \qquad f(u) = \tanh(u)$$

$$f(u) = H(u) \qquad f(u) = \frac{1}{1+\exp(-u)}$$

Funcții de activare (calcul semnal de ieșire)

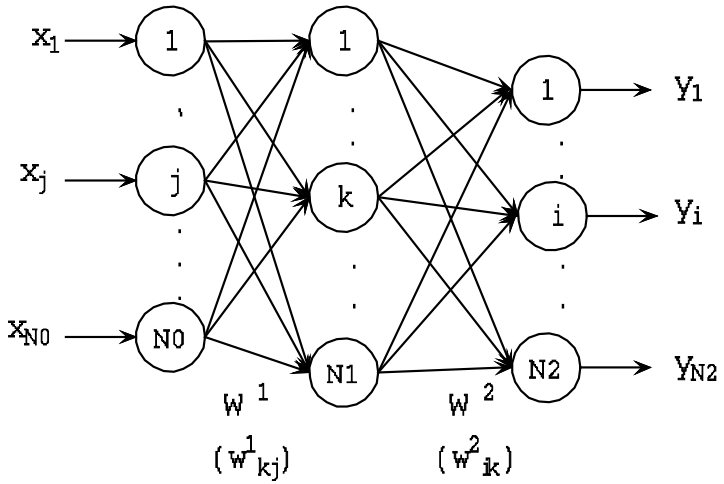
Structura unei rețele neuronale

Structura unei rețele neuronale artificiale:

- Arhitectura

- Funcționare

- Antrenare: determinarea parametrilor ajustabili

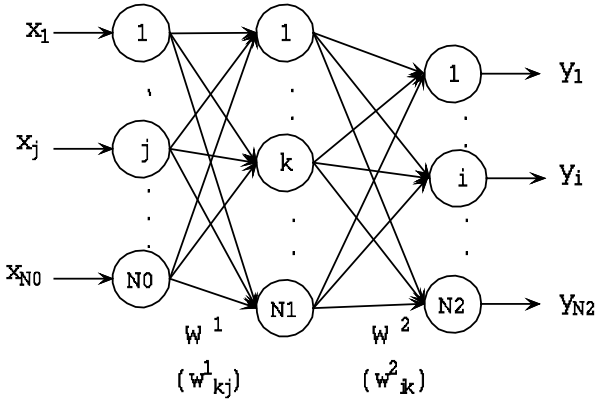


$$y_i = f\left(\sum_{k=0}^{N1} w_{ik}^2 f\left(\sum_{j=0}^{N0} w_{kj}^1 x_j\right)\right),\ i=\overline{1,N2}$$

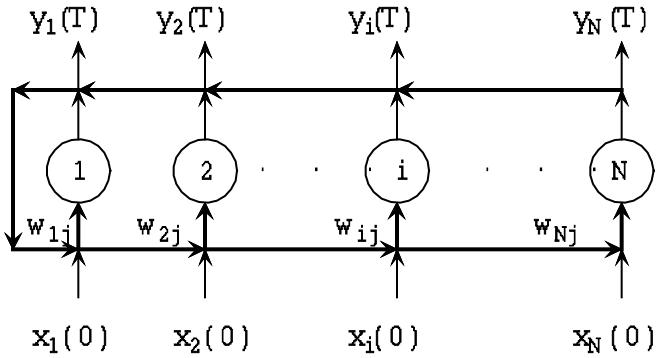
Rețea feedforward cu un nivel ascuns

Structura unei rețele neuronale

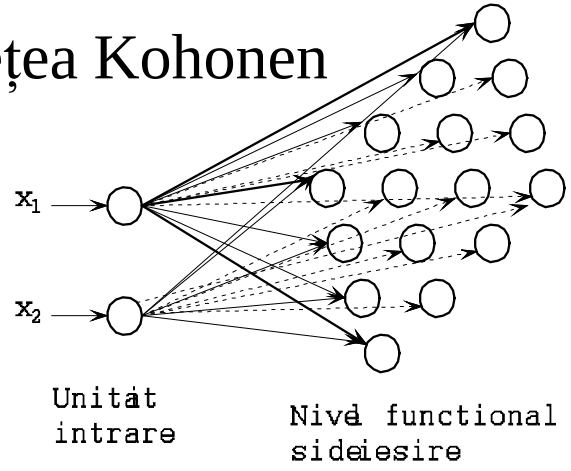
Tipuri de rețele neuronale:



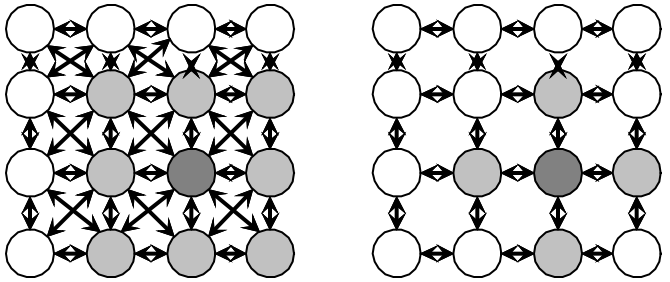
Rețea Hopfield



Rețea Kohonen



Rețea celulara



Antrenarea rețelelor neuronale

Antrenare (învățare) = extragerea modelului corespunzător problemei pornind de la exemple

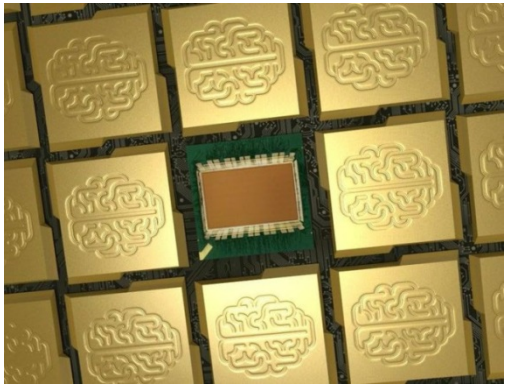
= determinarea parametrilor rețelei (ponderile asociate conexiunilor)

Tipuri de învățare:

- Supervizată (se cunosc atât datele de intrare cât și răspunsurile corecte)
- Nesupervizată (se cunosc doar datele de intrare)
- De tip recompensă / penalizare (se știe doar dacă răspunsul este corect sau nu însă nu se poate cuantifica abaterea față de răspunsul corect)

Aplicații ale calculului neuronal

- Clasificare și recunoaștere
  - Clasificarea supervizată și nesupervizată a datelor
  - Recunoașterea caracterelor / imaginilor / vorbirii
- Aproximare și estimare
  - Predicție în serii temporale
- Modelare și control
  - Modelarea unor sisteme neliniare
- Optimizare
  - Probleme de rutare
  - Probleme de planificare
- Prelucrarea și analiza semnalelor
  - Filtrare adaptivă



TrueNorth [IBM 2014]
(brain like chip)
- 4096 procesoare care imită funcționarea unui milion de neuron si 256 milioane de sinapse
- model de neuron bazat pe pulsuri

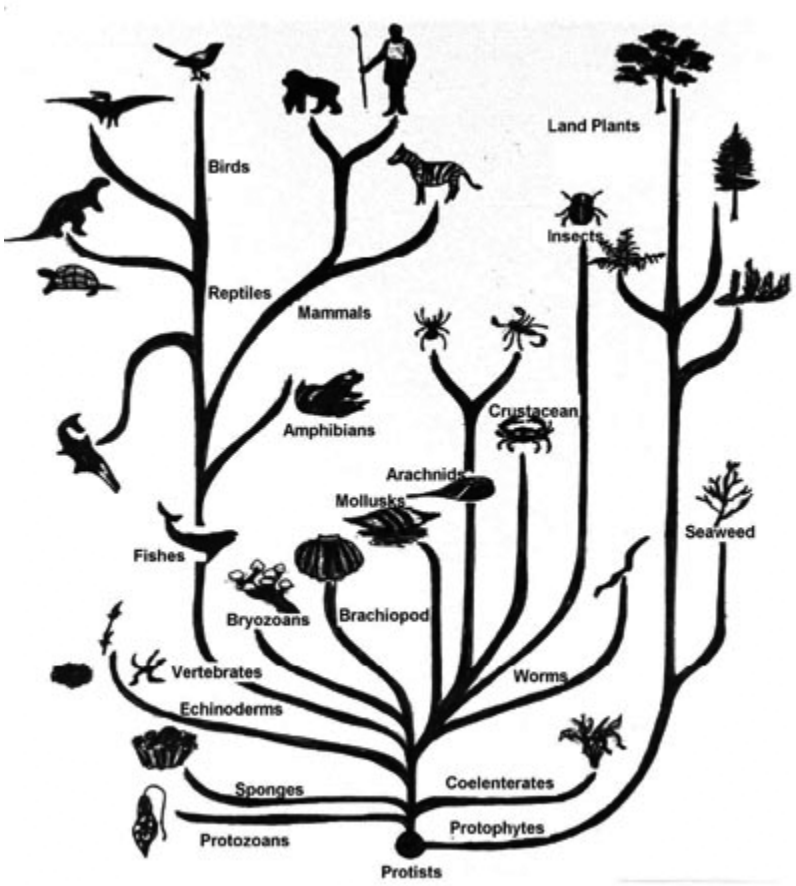
Calcul evolutiv - specific

Principiile calculului evolutiv

Structura unui algoritm evolutiv

Clase de algoritmi evolutivi

Aplicații ale calculului evolutiv



Principiile calculului evolutiv

- Este inspirat de procesele de evoluție din natură bazate pe principiile eredității și a supraviețuirii celui mai bun individ (cel care s-a adaptat cel mai bine mediului)

- Se bazează pe determinarea soluției unei probleme prin explorarea spațiului soluțiilor potențiale folosind o populație de “căutatori” (agenți sau indivizi)

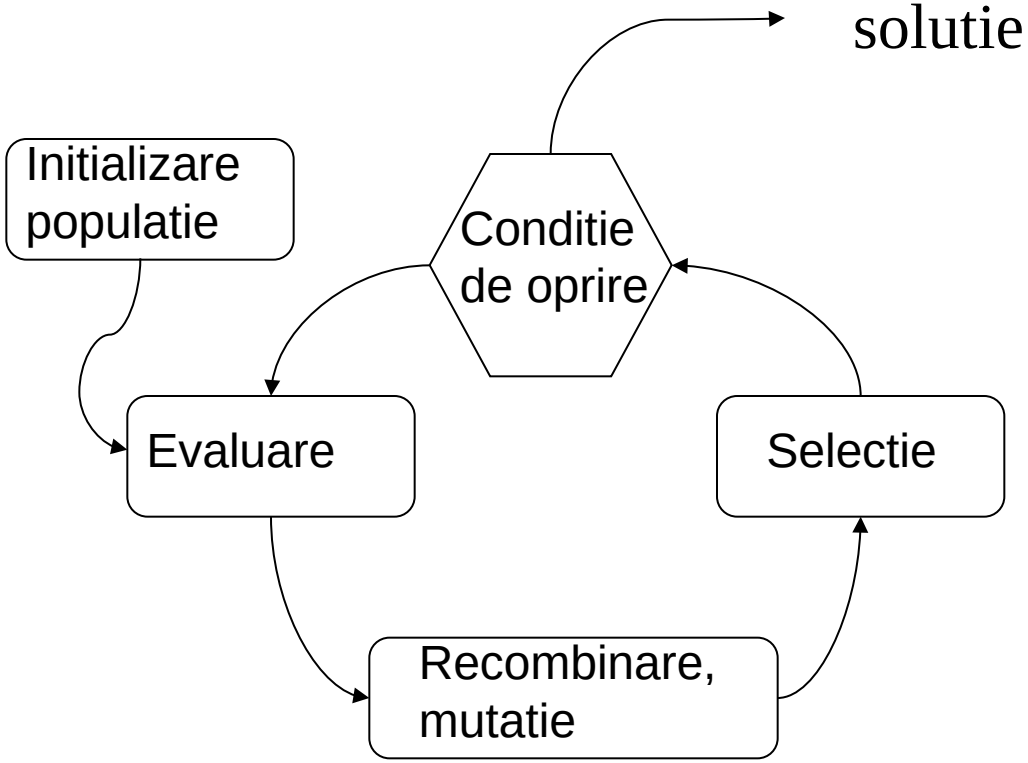
- Elementele populației sunt codificate în funcție de specificul problemei (șiruri de biti, vectori de valori reale, arbori etc.)

Principiile calculului evolutiv

Analogia dintre procesele de evolutie din natura și rezolvarea problemelor

Evolutia in natura		**Rezolvarea problemelor**
Mediu	⟷	Problema
Individ	⟷	Soluție candidat
Grad de adaptare la mediu	⟷	Calitate solutie

Structura unui algoritm evolutiv



Algoritm evolutiv = proces iterativ constând în aplicarea succesivă a unor operatori

- recombinare
- mutație
- selecție

asupra unei populații inițializate aleator

Clase de algoritmi evolutivi

- Algoritmi genetici:
  - Codificare binară a informației
  - Incrucișarea (recombinarea) este operatorul principal
  - Mutația este operator secundar
  - Adecvate pentru probleme de optimizare combinatorială

- Strategii evolutive:
  - Codificare reală a informației
  - Mutația este operatorul principal
  - Incrucișarea este operator secundar
  - Adecvate pentru rezolvarea de probleme de optimizare în domenii continue

Clase de algoritmi

- Programare genetică:
  - Elementele populației sunt structuri (arbori, expresii, programe etc)
  - Permit generarea unor structuri de calcul prin procese de evoluție

- Programare evolutivă:
  - Codificare reală a informației
  - Mutația este singurul operator
  - Utilizată în rezolvarea problemelor de optimizare continuă

Tendința curentă: hibridizarea diferitelor variante

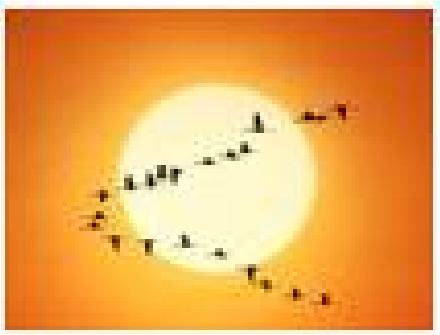
Tipuri de probleme

- Optimizare cu și fără restricții
  - functii nediferențiabile, discontinue sau a căror evaluare se bazează pe simulari
  - apar frecvent în proiectarea sistemelor din inginerie, în rezolvarea problemelor de planificare etc
- Optimizare multimodală
  - Pentru funcții cu mai multe optime globale sau locale
  - Apar în analiza datelor
- Optimizare multicriterială
  - Există mai multe criterii de optimi care sunt conflictuale
  - Apar in multe domenii aplicative
- Optimizare în medii dinamice sau nedeterministe
  - Criteriul de optim variază în timp sau este influențat de factori aleatori

Domenii de aplicabilitate

- Probleme de planificare a activităților (generarea automată a orarelor sau planificarea sarcinilor).
- Programare automată (generarea unor programe care sau structuri computationale cum sunt automatele celulare si rețelele de sortare).
- Probleme de predicție (evoluția unui activ financiar sau a cursului valutar).
- Prelucrarea imaginilor (proiectarea filtrelor pentru imagini și analiza imaginilor).
- Proiectarea rețelelor neuronale (stabilirea arhitecturii si/sau a ponderilor).
- Probleme de identificare a unor structuri (bioinformatică – identificarea structurii proteinelor)
- Simularea unor activități creative (generare de imagini, muzica etc.)

Tehnici înrudite



- Modele inspirate de inteligența mulțimilor

  PSO – Particle Swarm Optimization

  Creatori: Eberhart, Kennedy -1995

  http://www.swarmintelligence.org/

  - Sunt inspirate de comportamentul stolurilor de păsări, a bancurilor de pești sau a roiurilor de albine
  - In procesul de cautare a solutiei fiecare “individ” este ghidat de:
    - Experiența colectivă
    - Experiența individuală
  - Aplicații: probleme de optimizare în spații continue

Tehnici înrudite

- Modelul coloniilor de furnici

  ACO – Ant Colony Optimization
  [http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html]

  AS – Ant Systems

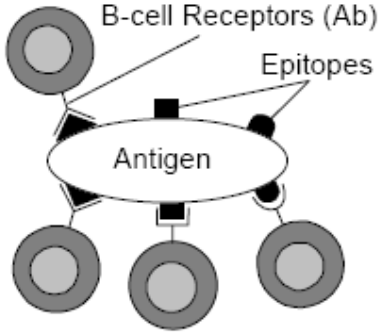
  Creator: Marco Dorigo - 1992

  – Sunt inspirate de comportamentul furnicilor în căutarea hranei și organizarea cuibului
  – Este esențial conceptul de feromon (substanță chimică care permite comunicarea între furnici)

  Aplicații:

  - Optimizare (probleme de rutare)
  - Analiza datelor (grupare)
  - Clasificarea imaginilor

Tehnici înrudite



- Modelul sistemului imunitar

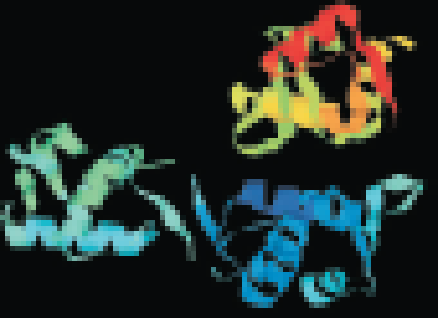
  AIS– Artificial Immune Systems
  [http://www.dca.fee.unicamp.br/~lnunes/immune.html]

  Creator: Leandro Castro - 1999

  – este inspirat de abilitatea sistemului imunitar biologic de a recunoaște agenții patogeni și de a reacționa la orice atac

  Aplicații:

  - Sisteme de detecție a intrusilor
  - Analiza datelor (grupare)



Tehnici înrudite

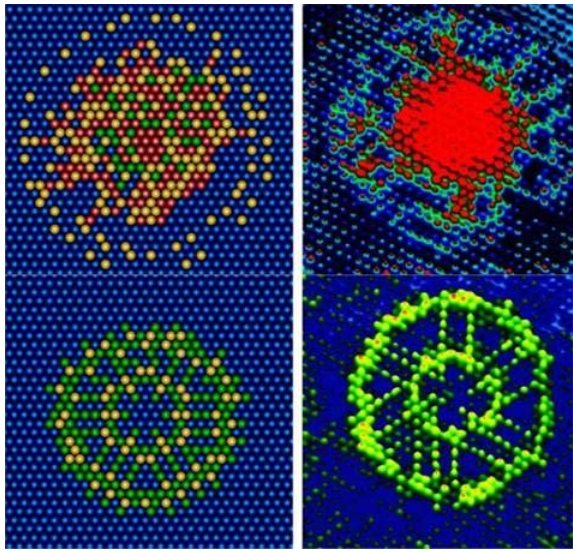
- Calcul ADN

Primul experiment: Adleman (1994) – problema comis voiajorului pentru 7 orase

Stare curentă:

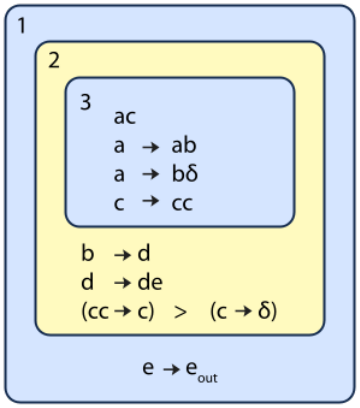
- Calculator de scară moleculară bazat pe principiile biologiei computaționale (2010)
- Algoritmi inspirați de transformările structurilor ADN (tăiere, clonare, filtrare)

Calculator ADN = colecție de secvente ADN special selectate a căror combinare (prin transformari biochimice) conduce la rezultatul problemei



Structura de grosimea a doua molecule – genereaza diferite “pattern”-uri

Tehnici înrudite



- Membrane Computing (P-systems) [http://ppage.psystems.eu]
- Primul model: P-sistem propus Gh. Paun (1998)
- P-sistem = model abstract de calcul inspirat de structura compartimentată a celulelor vii și de modul în care se desfășoară procesele chimice în cadrul acestora.
- P-sistemele prelucrează multiseturi de obiecte simbolice plasate într-o structură ierarhică de membrane
- Există multe rezultate teoretice referitoare la puterea de calcul a sistemelor membranare însă mai puține aplicații (domeniile de aplicabilitate includ biologia, lingvistica, management)

Tehnici înrudite

- Artificial Bee Colony (Karaboga, 2005) http://mf.erciyes.edu.tr/abc/
  - Inspirate de comportamentul albinelor în căutarea polenului
- Biogeography Based Optimization (BBO Algorithms) - D. Simon, 2008 http://academic.csuohio.edu/simond/bbo/
  - Inspirate de procesele de migrare a speciilor
- Fireflies Algorithm (X.S. Yang, 2008)
  - Inspirate de comportamentul licuricilor
- Cuckoo Search (X.S. Yang & S. Deb, 2009)
  - Inspirate de comportamentul cucilor în utilizarea cuiburilor altor specii
- Bat Algorithm (X.S. Yang & A.H Gandomi, 2010)
  - Inspirate de abilitatea liliecilor de a se orienta

Bibliografie

A.Engelbrecht: Computational Intelligence. An Introduction, John Wiley and Sons, 2007

L. Rutkowski: Computational Intelligence: Methods and Techniques, Springer, 2008

A.Konar: Computational Intelligence: Principles, Techniques and Applications, Springer, 2005

Z. Michalewicz, D. Fogel: How to Solve It. Modern Heuristics. Springer, 1999

Structura curs

- Rețele neuronale artificiale
  - probleme de asociere (aproximare, clasificare)
  - probleme de optimizare
- Algoritmi aleatori de optimizare
  - Simulated annealing
- Calcul evolutiv
  - Algoritmi genetici
  - Strategii evolutive
  - Programare evolutivă
  - Programare genetică
  - Modele paralele și distribuite in calculul evolutiv
  - Algoritmi evolutivi pentru optimizare multi-criteriala
- Proiectarea evolutivă a rețelelor neuronale
- Tehnici înrudite: sisteme de particule, modelul coloniilor de furnici, sisteme imunitare artificiale, algoritmi bazati pe estimarea distributiei in starilor in spatiul solutiilor, algoritmi cuantici

Structura laborator

Lab 1: Probleme de clasificare (pattern recognition) – rețele neuronale feedforward

Lab 2: Probleme de aproximare și predicție – rețele neuronale feedforward și recurente

Lab 3: Probleme de optimizare combinatorială - SA, GA

Lab 4: Probleme de optimizare în domenii continue - ES

Lab 5: Design evolutiv - GP

Lab 6: Probleme de optimizare multicriterială - MOEA

Lab 7: Aplicații ale tehnicilor înrudite (ACO, PSO, AIS)

Medii de experimentare:

Weka, Scilab, R

Evaluare

Materiale pentru curs: http://www.info.uvt.ro/~dzaharie/cne2014

Mod de evaluare:

Proiect final: 60-80%

Test scris: 20%

Teme laborator: 20%