BABEŞ-BOLYAI UNIVERSITY
Faculty of Mathematics and Computer Science

Inteligenţă Artificială

9: Învăţare Automată

Camelia Chira

cchira@cs.ubbcluj.ro

Sisteme inteligente

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

- Cum sa proiectam roboti mobili autonomi care invata din experienta?
- Cum sa folosim inregistrari medicale existente pentru a invata ce pacienti viitori vor raspunde mai bine caror tratamente?
- Cum sa dezvoltam motoare de cautare care se adapteaza automat intereselor individuale ale utilizatorilor?

- *Învăţare Automată - Machine Learning (ML)*
  - Dezvoltarea de metode care pot detecta automat forme (patterns) in date si care pot apoi folosi formele descoperite pentru a prezice date viitoare
  - Extragerea de forme si tendinte importante din date pentru a intelege ce ne spun datele

Exemple ML

- Clasificare
  - Filtru spam
  - Detectarea obiectelor
  - Predictia vremii
- Regresie
  - Predictia vremii
  - Bursa de actiuni
  - Imobiliare
- Ranking
  - Cautare online
  - Gasire elemente similare
- Clustering
  - Gruparea elementelor similare
  - Gruparea genelor
  - Gruparea paginilor web

Aplicatii

- Recunoaştere de imagini şi semnal vocal
- Recunoaşterea scrisului de mână
- Detecţia feţelor
- Detecţia obstacolelor
- Recunoaşterea amprentelor
- Controlul roboţilor
- Predicţia vremii
- Diagnosticare medicală
- Detecţia fraudelor
- etc

Învăţare Automată: tipuri de învăţare

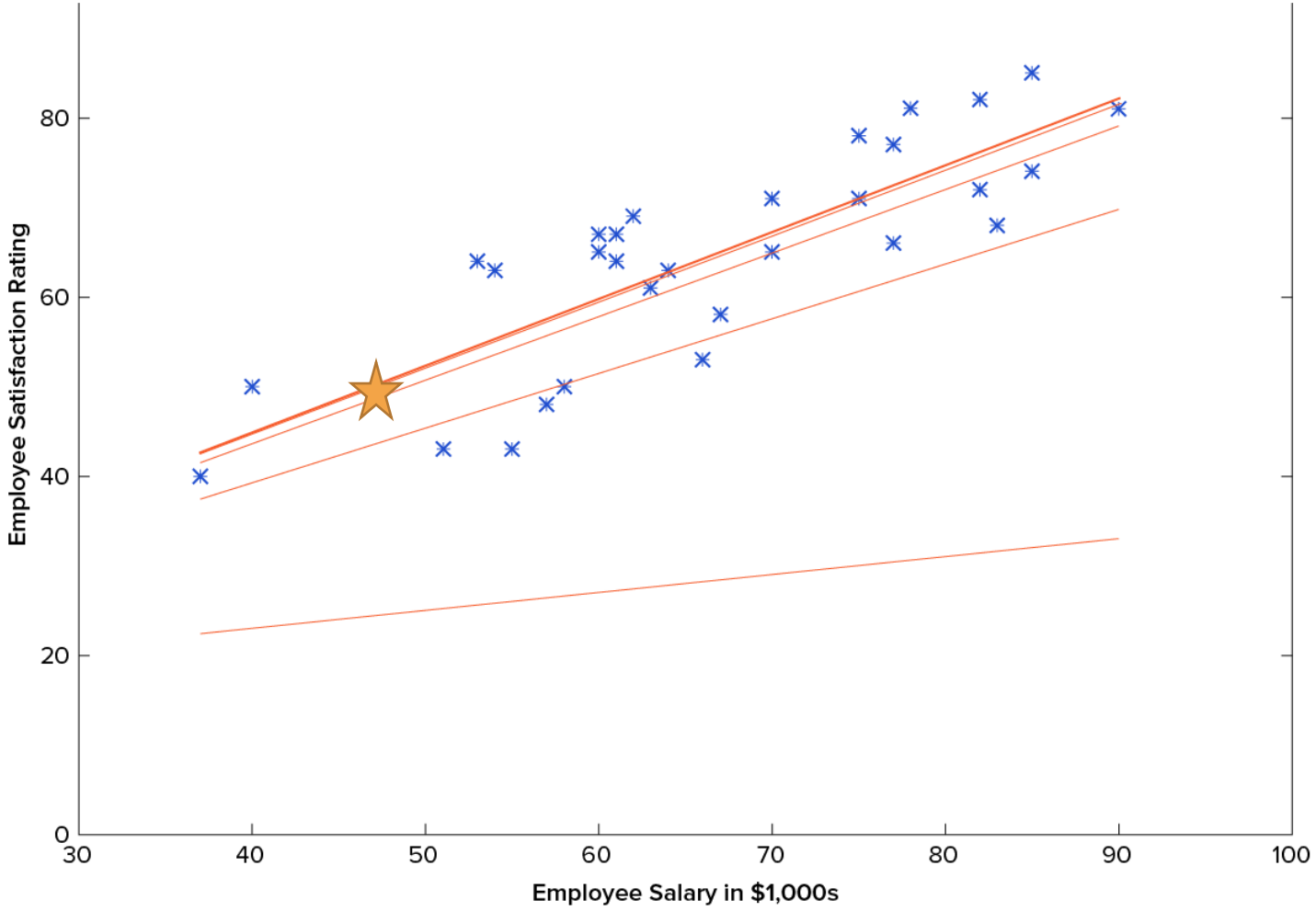
- *Invatare supervizata*
  - Regresie
    - Furnizarea unei iesiri corecte pentru o intrare noua
    - Output continuu – numar real
    - Ex. Predictia preturilor
  - Clasificare
    - Clasificarea unei intrari noi
    - Output discret – eticheta intr-o multime predefinita
    - Ex. Detectarea tumorilor maligne
- *Invatare nesupervizata*
  - Clustering
    - Gasirea unui model sau structuri utile a datelor
    - Output grupuri de date - datele se impart in grupuri astfel incat datele din acelasi grup sa fie similare iar cele din grupuri diferite sa fie foarte diferite
    - Ex. Analiza genelor

Exemple: probleme de regresie

- Predictia pretului unei actiuni pe baza unui istoric
- Predictia nivelului de satisfactie a angajatilor pe baza castigurilor salariale
- Predictia valorii unei case pe baza unor atribute e.g. suprafata, locatie, numar bai, numar dormitoare, gradina, etc

Exemple modele
- Linear regression
- Logistic regression
- Arbori de decizie
- Retele neuronale

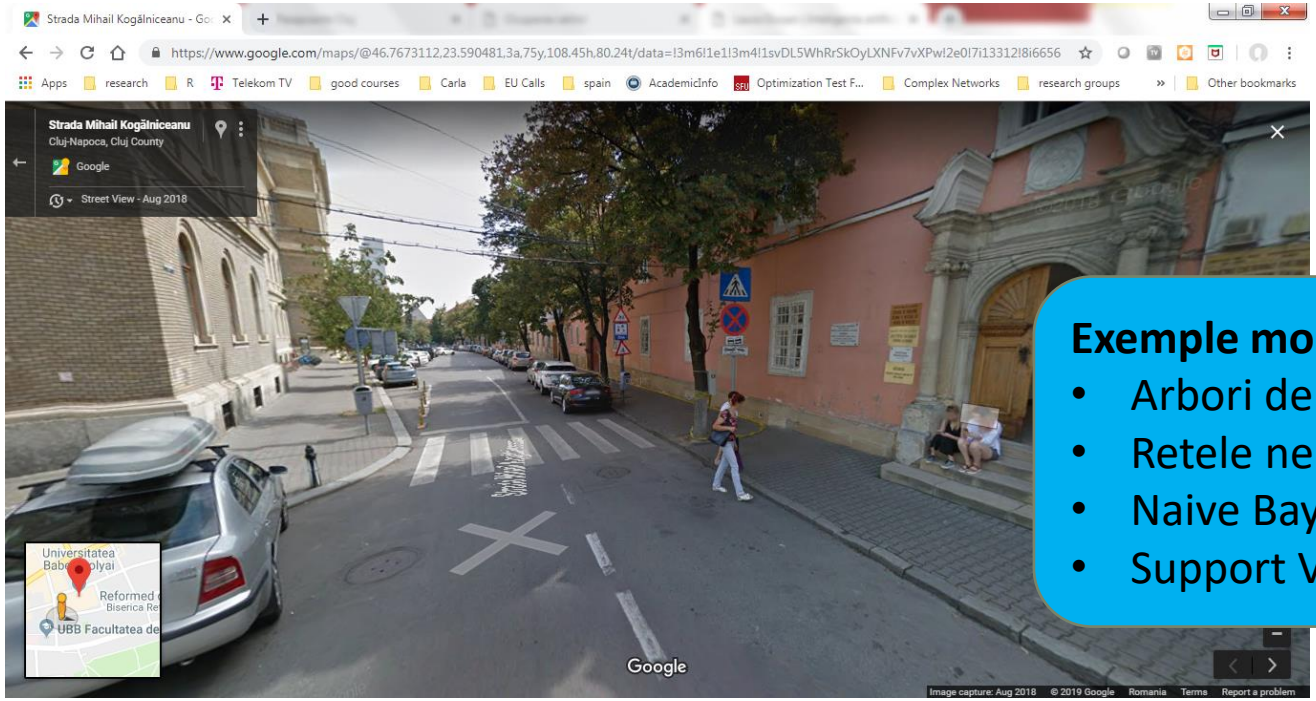
Exemplu: predictia nivelului de satisfactie



Sursa: https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer

Exemple: probleme de clasificare

- Diagnoza medicala: clasificarea celulelor tumorale in benigne sau maligne
- Spam filtering: clasificarea emailurilor ca spam sau utile
- Face detection: detectarea fetelor in imagini



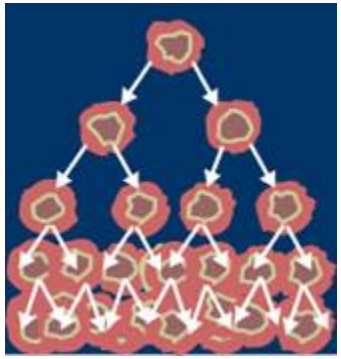
Exemple modele

- Arbori de decizie
- Retele neuronale
- Naive Bayes
- Support Vector Machines

Exemplu: diagnoza medicala

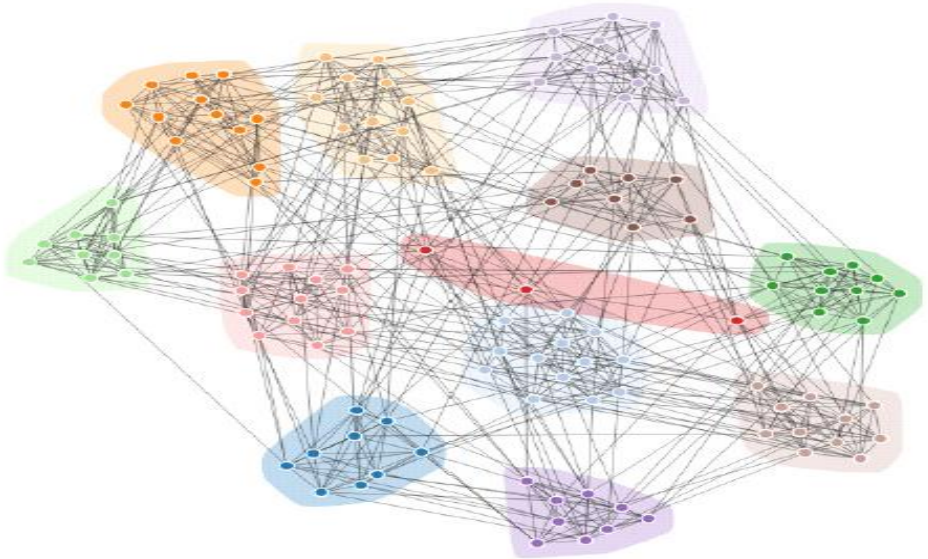
Exemplu de set de date (breast-cancer-wisconsin -arff format)

@relation wisconsin-breast-cancer
@attribute Clump_Thickness integer [1,10]
@attribute Cell_Size_Uniformity integer [1,10]
@attribute Cell_Shape_Uniformity integer [1,10]
@attribute Marginal_Adhesion integer [1,10]
@attribute Single_Epi_Cell_Size integer [1,10]
@attribute Bare_Nuclei integer [1,10]
@attribute Bland_Chromatin integer [1,10]
@attribute Normal_Nucleoli integer [1,10]
@attribute Mitoses integer [1,10]
@attribute Class { benign, malignant}
@data
5,1,1,1,2,1,3,1,1,benign
5,4,4,5,7,10,3,2,1,benign
3,1,1,1,2,2,3,1,1,benign
8,10,10,8,7,10,9,7,1,malignant
1,1,1,1,2,10,3,1,1,benign



Exemple: probleme de clustering

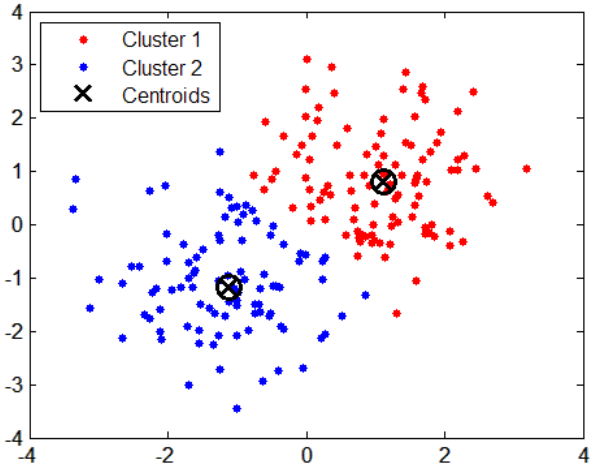
- Clasificarea documentelor in functie de topic (categorii multiple)
- Gruparea clientilor in functie de interese, istoricul cumparaturilor sau monitorizarea activitatilor
- Gruparea genelor din date de tip time-series microarray



Exemple modele

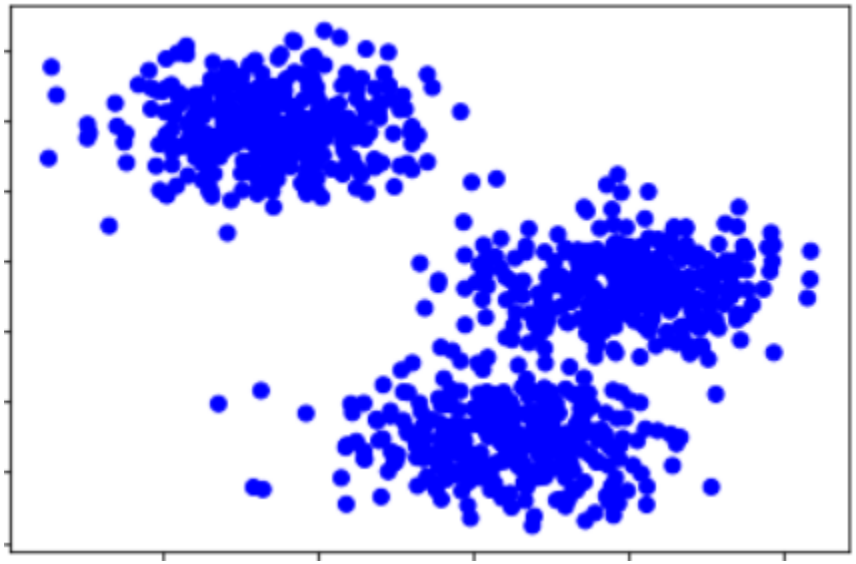
- K-means
- Fuzzy C-means
- Hierarchical clustering

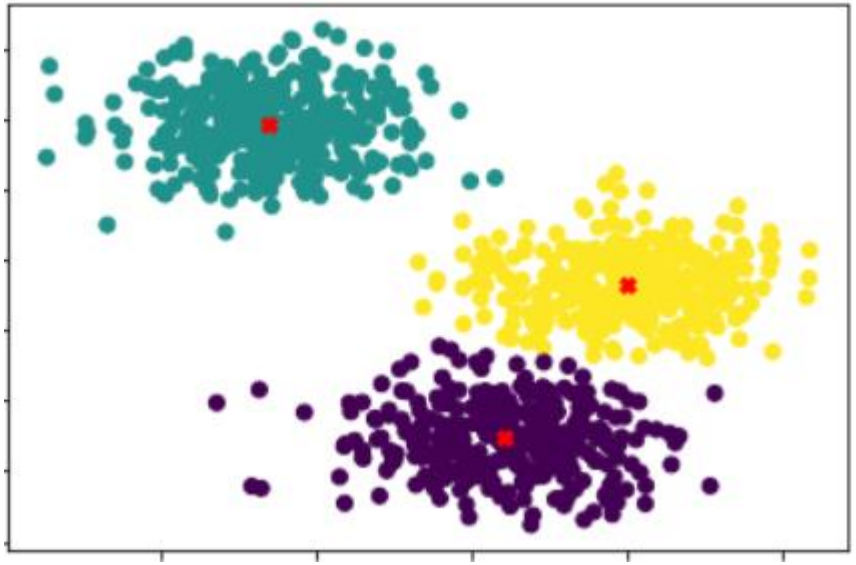
Exemplu: clustering people



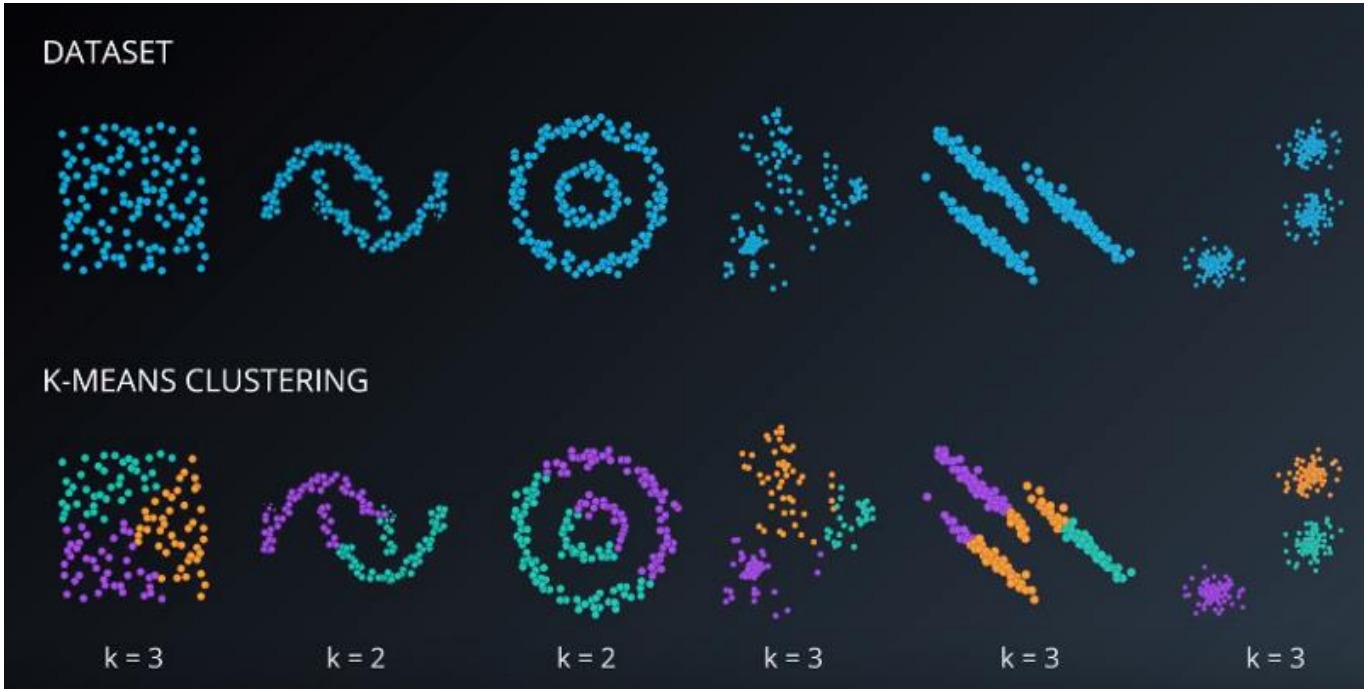
- Identificarea clientilor cu aceleasi interese pe baza cumparaturilor online

K means





K means



See https://scikit-learn.org/stable/modules/clustering.html

Concepte de baza

- Ce se cunoaste?
  - O colectie de inregistrari pentru care se cunoaste clasa careia ii aparțin (set de date etichetate)
  - Fiecare inregistrare conține un set de atribute si eticheta clasei de care apartine
- Ce se cauta?
  - Un model pentru relatia dintre clasa unei inregistrari și atribute
  - Acest model de clasificare mapeaza o inregistrare cu anumite atribute la o anumita clasa
- De ce?
  - Scopul este de a utiliza modelul de invatare gasit pentru a putea fi aplicat unor date noi si determina clasa corespunzatoare pe baza atributelor

Modele de clasificare

- Proces: etapa de antrenare, etapa de clasificare
- Performanta algoritmului
  - Acuratetea calculata in faza de antrenare, faza de testare
    - Acc = nr de exemple corect clasificate / nr total de exemple
- Metode de evaluare
  - Seturi disjuncte de antrenare si testare
    - setul de antrenare poate fi imparţit in date de invatare si date de validare
    - setul de antrenare este folosit pentru estimarea parametrilor modelului (cei mai buni parametri obtinuti pe validare vor fi folositi pentru constructia modelului final)
  - Validare incrucisata cu mai multe (h) sub-seturi egale ale datelor (de antrenament)
    - separararea datelor de h ori in *h-1* sub-seturi pentru invatare si 1 sub-set pt validare
    - dimensiunea unui sub-set = dimensiunea setului / h
    - performanta este data de media pe cele h rulari (ex. *h = 5* sau *h = 10)*
  - Leave-one-out cross-validation

Masuri de performanta

- Fie o problema de clasificare cu 2 clase:
  - Clasa C1 - pozitiva
  - Clasa C2 - negativa

- *Matricea de confuzie:*

		Rezultat clasificator	
		C1 (clasa pozitiva)	**C2 (clasa negativa)**
Clasa reala	**C1 (clasa pozitiva)**	TP	FN
	C2 (clasa negativa)	FP	TN

TP: True Positive = nr de cazuri C1 care sunt clasificate (*corect*) în C1
TN: True Negative = nr de cazuri C2 care sunt clasificate (*corect*) în C2
FP: False Positive = nr de cazuri C2 dar care sunt clasificate (*incorect*) în C1
FN: False Negative = nr de cazuri C1 dar care sunt clasificate (*incorect*) în C2

Masuri de performanta

- **Acuratete = (TP+TN)/(TP+TN+FP+FN)**
  - nr date clasificate corect/ nr total de date
- **Sensitivitate = TP/(TP+FN)**
  - TP rate sau **recall** (rata de regasire)
- **Specificitate = TN/ (TN+FP)**
  - TN rate
  - FP rate = 1-specificitate=FP/(TN+FP)
- **Precizie = TP/(TP+FP)**
  - nr cazuri real pozitive/ nr cazuri clasificate ca fiind positive (**precision**)

- ✓ Toate aceste valori sunt in [0,1]
- ✓ Valori mari => performanta buna

TP: True Positive = nr de cazuri C1 care sunt clasificate (*corect*) în C1
TN: True Negative = nr de cazuri C2 care sunt clasificate (*corect*) în C2
FP: False Positive = nr de cazuri C2 dar care sunt clasificate (*incorect*) în C1
FN: False Negative = nr de cazuri C1 dar care sunt clasificate (*incorect*) în C2

Modele de clasificare

- Un model de clasificare (sau clasificator) trebuie sa fie:
  - Acurat (sa identifice clasa corecta)
  - Compact (usor de interpretat de catre utilizator)
  - Eficient in etapa de antrenare si etapa de clasificare

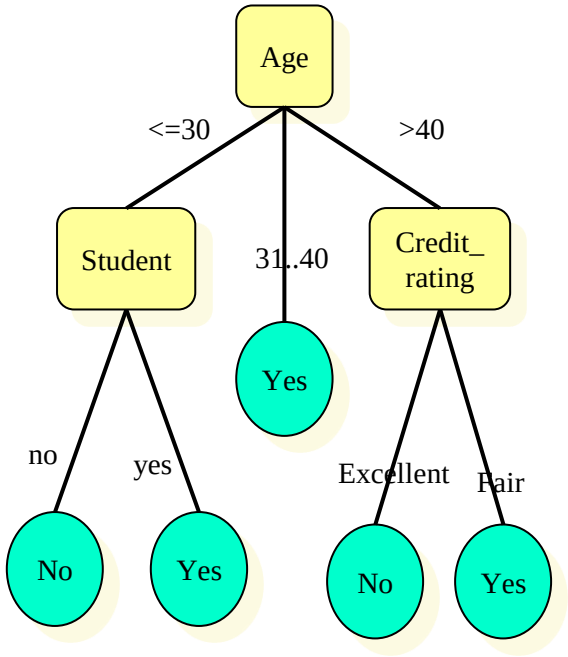
- Exemple de modele de clasificare
  - Arbori de decizie (decision trees)
  - Reguli de clasificare (classification rules)
  - Retele neuronale (neural networks)
  - Masini cu suport vectorial (Support vector machines)
  - Modele probabiliste
  - etc

Arbori de decizie

- Scop
  - Divizarea unei colecţii de articole în seturi mai mici prin aplicarea succesivă a unor reguli de decizie
- Definire
  - Graf special cu mai multe tipuri de noduri
  - Fiecare nod intern corespunde unui atribut
  - Fiecare ramură de sub un nod (atribut) corespunde unei valori a atributului
  - Fiecare frunză corespunde unei clase
- Tipuri de probleme
  - Exemplele (instanţele) sunt reprezentate printr-un număr fix de atribute, fiecare atribut putând avea un număr limitat de valori
  - Probleme de clasificare: binara / multi-clasa
  - Probleme de regresie

Arbori de decizie: exemplu

RID	Age	Income	Student	Credit_rating	Class: buys_computer
1	<=30	High	No	Fair	No
2	<=30	High	No	Excellent	No
3	31..40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31..40	Low	Yes	Excellent	Yes
8	<=30	Medium	No	Fair	No
9	<=30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<=30	Medium	Yes	Excellent	Yes
12	31..40	Medium	No	Excellent	Yes
13	31..40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No



Arbori de decizie: proces

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

Construirea arborelui

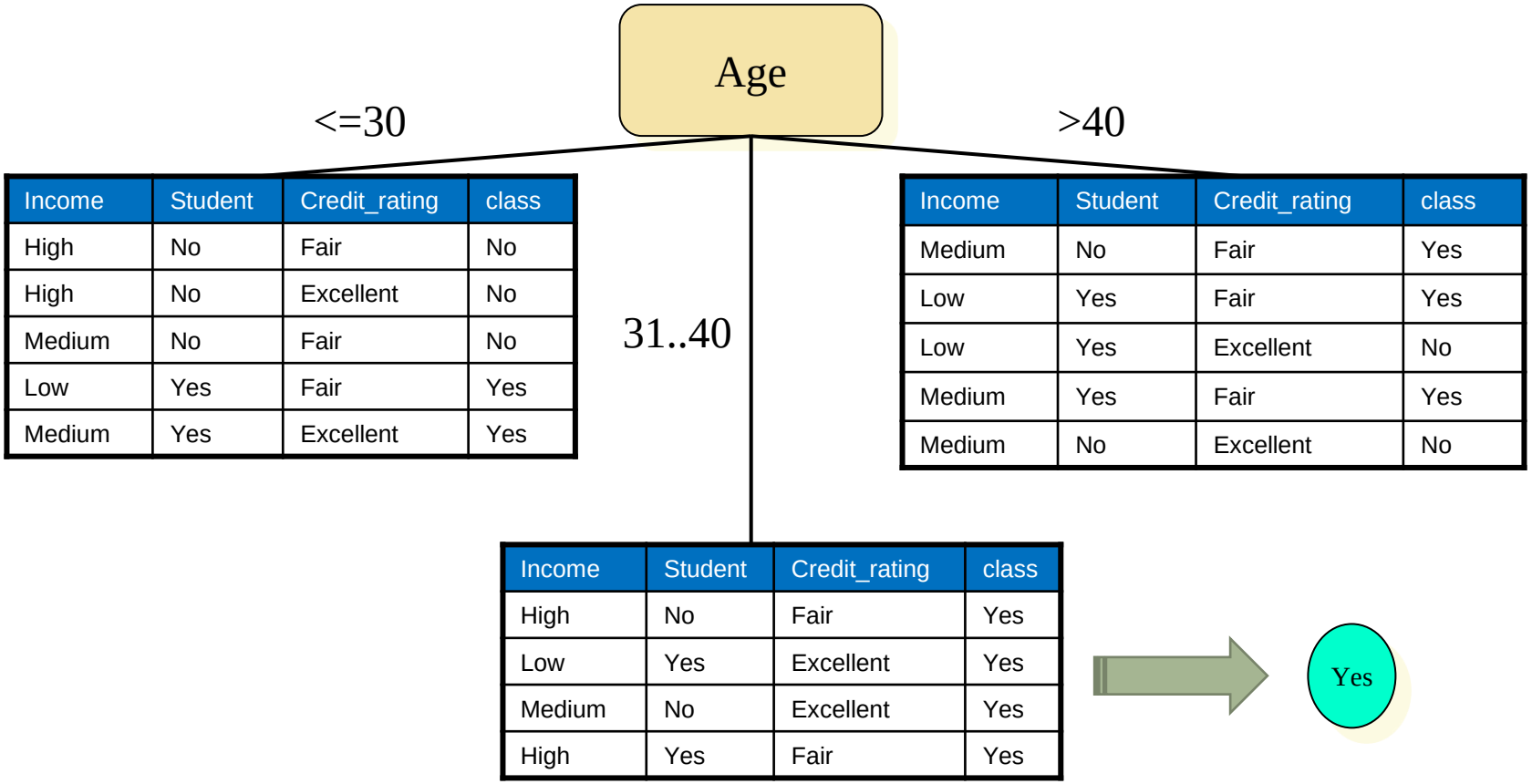
- Divizarea datelor de antrenament în subseturi pe baza caracteristicilor datelor
  - Un nod - întrebare legată de o anumită proprietate a unui obiect dat
  - Ramurile ce pleacă din nod - etichetate cu posibilele răspunsuri la întrebarea din nodul curent
  - La început toate exemplele sunt plasate în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

Construirea auborelui: exemplu

RID	Age	Income	Student	Credit_rating	Class: buys_computer
1	<=30	High	No	Fair	No
2	<=30	High	No	Excellent	No
3	31..40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31..40	Low	Yes	Excellent	Yes
8	<=30	Medium	No	Fair	No
9	<=30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<=30	Medium	Yes	Excellent	Yes
12	31..40	Medium	No	Excellent	Yes
13	31..40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

Construirea auborelui: exemplu

- Pentru radacina se alege atributul '*Age*'



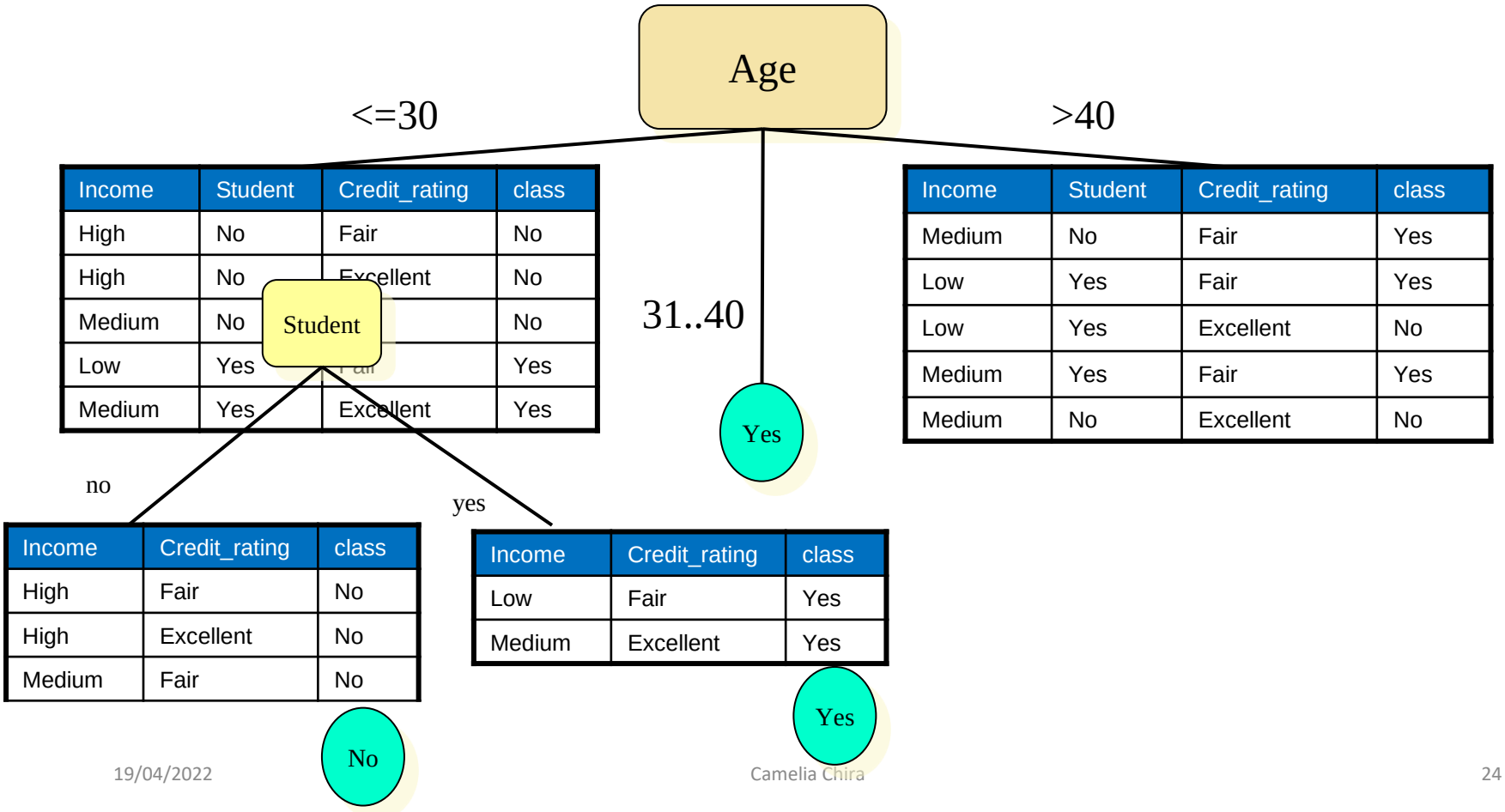
Income	Student	Credit_rating	class
High	No	Fair	No
High	No	Excellent	No
Medium	No	Fair	No
Low	Yes	Fair	Yes
Medium	Yes	Excellent	Yes

Income	Student	Credit_rating	class
Medium	No	Fair	Yes
Low	Yes	Fair	Yes
Low	Yes	Excellent	No
Medium	Yes	Fair	Yes
Medium	No	Excellent	No

Income	Student	Credit_rating	class
High	No	Fair	Yes
Low	Yes	Excellent	Yes
Medium	No	Excellent	Yes
High	Yes	Fair	Yes

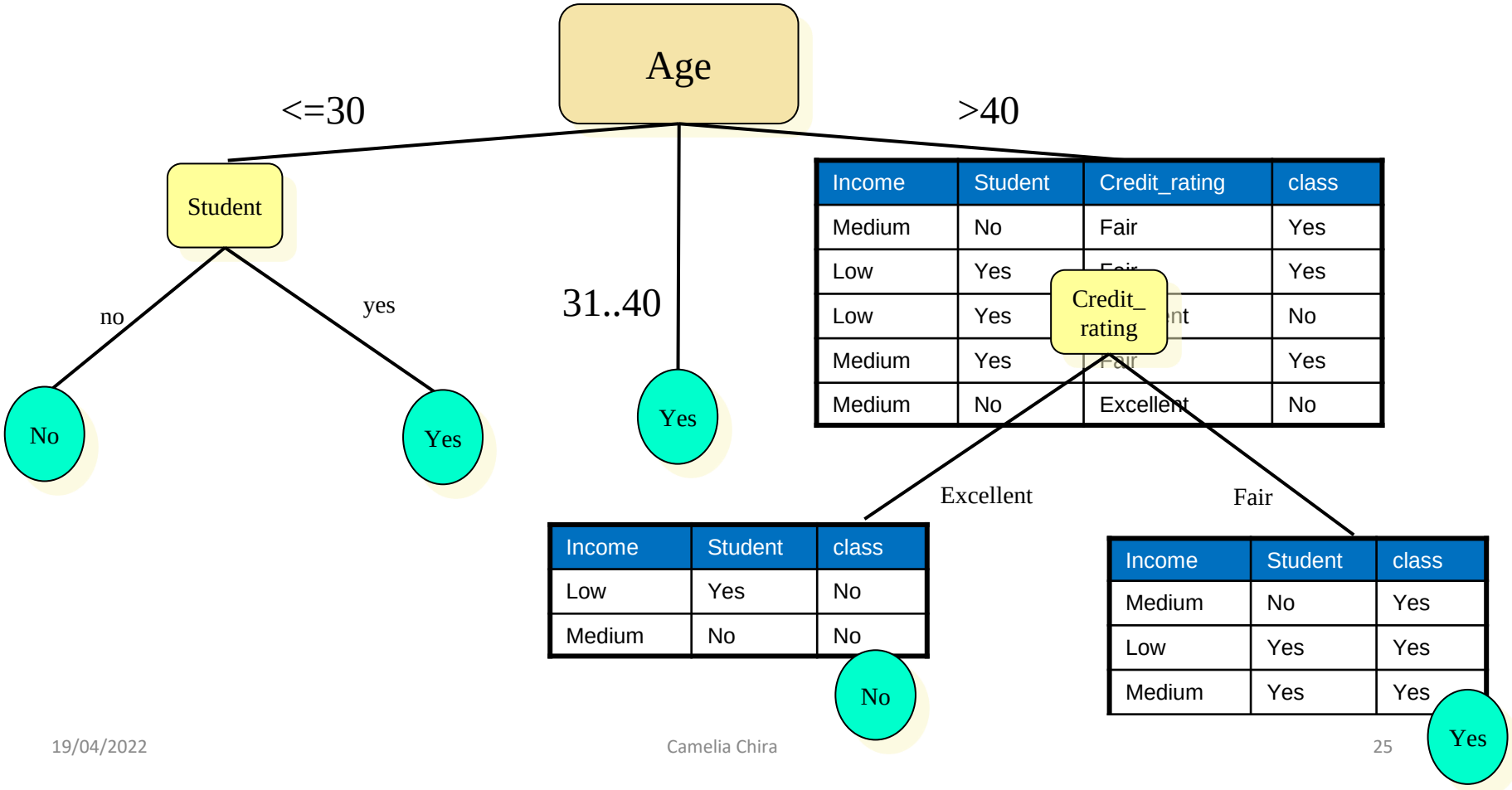
Construirea auborelui: exemplu

- Pentru ramura age <= 30 se alege atributul '*Student*'

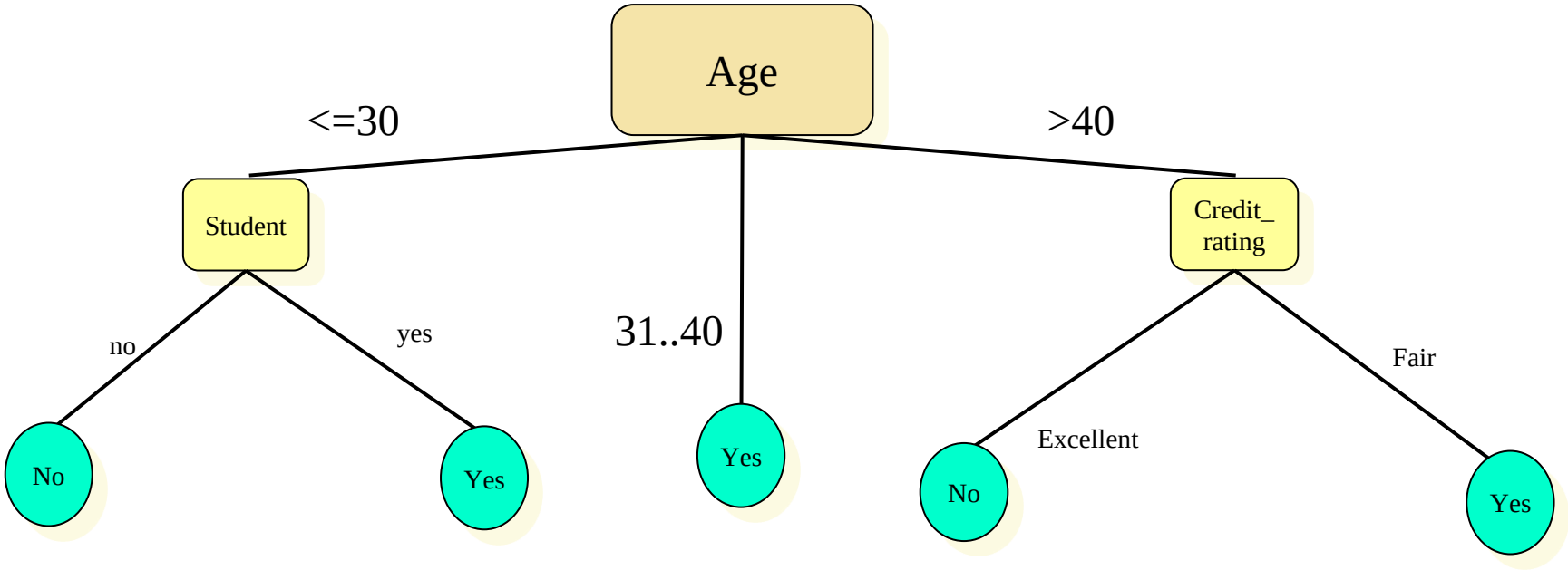


Construirea auborelui: exemplu

- Pentru ramura age > 40 se alege atributul '*Credit_rating*'



Construirea auborelui: exemplu



Agloritmi pentru construirea arborelui de decizie

- Algoritmul ID3
  - Intrare: set de date
  - Iesire: arbore de decizie (noduri interne etichetate cu atribute, noduri frunză etichetate cu clase, muchii etichetate cu valori ale atributelor)

- Algoritmul C4.5
  - Imbunatatire a algoritmul ID3 pentru a trata atribute continue, valori absente

- J48 –implementarea din Weka a algoritmului C4.5
  - https://www.cs.waikato.ac.nz/ml/weka/

Algoritmul ID3/C4.5

generare(D, A){ //D – partiţionare a exemplelor de antrenament, A – lista de atribute

Crearea unui nod nou N

Dacă exemplele din D fac parte dintr-o singură clasă C **atunci**

nodul N devine frunză şi este etichetat cu C

returnează nodul N

Altfel

Dacă A=∅ **atunci**

nodul N devine frunză şi este etichetat cu clasa majoritară în D

returnează nodul N

Altfel

atribut_separare = *Selectează_atribut*(D, A)

Etichetează nodul N cu atribut_separare

Pentru fiecare valoare posibilă vj a lui atribut_separare

Fie Dj mulţimea exemplelor din D pentru care atribut_separare = vj

Dacă Dj = ∅ **atunci**

Ataşează nodului N o frunză etichetată cu clasa majoritară în D

Altfel

Ataşează nodului N un nod returnat de generare(Dj, A –atribut_separare)

returnează nodul N

}

Algoritmul ID3/C4.5: alegerea atributului de ramificare

- *Selectează_atribut(D, A)*
  - Aleatoare
  - Atributul cu cele mai puţine/multe valori
  - Atributul cu cel mai mare câştig de informaţie (information gain)
    - Rata câştigului (gain ratio)
- Câştigul de informaţie
  - O măsură de impuritate
    - 0 (minimă) dacă toate exemplele fac parte din aceeaşi clasă
    - 1 (maximă) dacă avem număr egal de exemple din fiecare clasă
  - Se bazează pe entropia datelor
    - Clasificare binară: $H(D)= - p_1 log_2 p_1 - p_2 log_2 p_2$, *p1* proporţia exemplelor pozitive în setul de date D, *p2* proporţia exemplelor negative în setul de date *D*
    - Clasificare cu mai multe clase: $H(D) = \Sigma - p_i log_2 p_i$
  - Câştigul de informaţie al unui atribut al datelor
    - Reducerea entropiei setului de date ca urmare a eliminării atributului

Alegerea atributului de ramificare - Information Gain

- Set de date distribuit in k clase: D={C1,C2,…,Ck}
- Distribuţia de probabilitate (p1,p2, …, pk), *pi=card(Ci)/card(D)*
- Fie A un atribut şi $v_1,v_2,\ldots,v_{mA}$ valorile posibile ale acestui atribut
- Fie Dj setul de date din D pt care atributul A are valoarea vj şi Pj distribuţia datelor din Dj în cele k clase
- C_{ji} = datele din clasa C_i care au valoarea v_j pt atributul A
- Câştigul informational - Information Gain (IG) obtinut prin partitionarea setului de date folosind atributul A:

$$IG(D,A) = H(D) - \sum_{j=1}^{m_A} P(D_j \mid A = v_j) H(D_j \mid A = v_j)$$

$$H(D) = -\sum_{i=1}^{k} p_i \log p_i \qquad P(D_j \mid A = v_j) = \frac{card(D_j)}{card(D)}$$

$$H(D_j \mid A = v_j) = -\sum_{i=1}^{k} p_{ij} \log p_{ij}, \quad p_{ij} = \frac{card(C_{ji})}{card(C_i)}$$

Information Gain (IG): Exemplu

RID	Age	Income	Student	Credit_rating	Class: buys_computer
1	<=30	High	No	Fair	No
2	<=30	High	No	Excellent	No
3	31..40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31..40	Low	Yes	Excellent	Yes
8	<=30	Medium	No	Fair	No
9	<=30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<=30	Medium	Yes	Excellent	Yes
12	31..40	Medium	No	Excellent	Yes
13	31..40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

Distribuţia claselor

- C_1="yes", C_2="no"
- $p_1=9/14$, $p_2=5/14$

$$H(p_1, p_2) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

$$H(p_1, p_2) = 0.940$$

Information Gain (IG): Exemplu

- Calcul entropie pt fiecare atribut

RID	Age	Income	Student	Credit_rating	Class: buys_computer
1	<=30	High	No	Fair	No
2	<=30	High	No	Excellent	No
3	31..40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31..40	Low	Yes	Excellent	Yes
8	<=30	Medium	No	Fair	No
9	<=30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<=30	Medium	Yes	Excellent	Yes
12	31..40	Medium	No	Excellent	Yes
13	31..40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

Age:
"<=30": 3 No, 2 Yes, 5 total
"31..40": 0 No, 4 Yes, 4 total
">40": 3 Yes, 2 No, 5 total

H("<=30")
*=-3/5*log(3/5)-2/5*log(2/5)*
=0.971

H("31..40")
= -0*log(0)-4/4*log(4/4)
=0

H(">40")
*=-3/5*log(3/5)-2/5*log(2/5)*
=0.971

Information Gain (IG): Exemplu

- Calcul entropie pt fiecare atribut: *Age*

H(age) = 5/14**H("<=30")*+4/14**H("31..40")* +5/14**H(">40")* = 0.694

- *Castigul informational pentru atributul Age:*

$$IG(Age) = H(p1,p2)-H(age)= 0.246$$

- Similar, se calculeaza:

IG(Income)=0.029
IG(Student)=0.151
IG(Credit_rating)=0.048

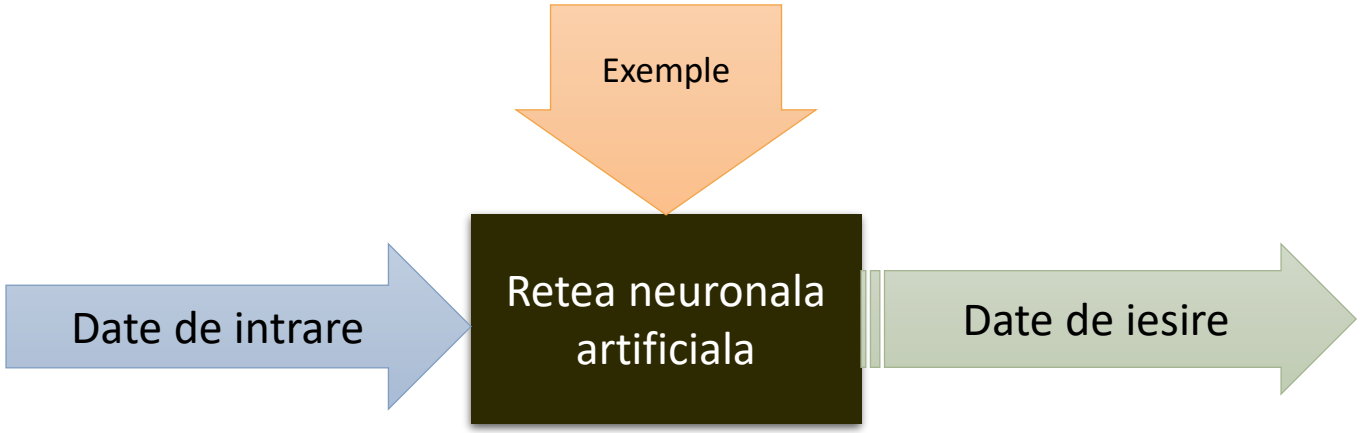
Utilizarea arborilor de decizie

- Se extrag regulile formate în arborele anterior construit
- Reguli extrase din arborele dat în exemplul anterior:
  - IF *age = "<=30"* AND *student = "no"* THEN *buys_computer = "no"*
  - IF *age = "<=30"* AND *student = "yes"* THEN *buys_computer = "yes"*
  - IF *age = "31…40"* THEN *buys_computer = "yes"*
  - IF *age = ">40"* AND *credit_rating = "excellent"* THEN *buys_computer = "no"*
  - IF *age = ">40"* AND *credit_rating = "fair"* THEN *buys_computer = "yes"*

- Regulile sunt folosite pentru a clasifica datele de test (date noi). Fie *x* o dată pentru care nu se știe clasa de apartenență
  - Regulile se pot scrie sub forma unor predicate astfel:
    - IF *age(x, <=30)* AND *student(x, no)* THEN *buys_computer (x, no)*
    - IF *age(x, <=30)* AND *student (x, yes)* THEN *buys_computer (x, yes)*

Retele neuronale artificiale

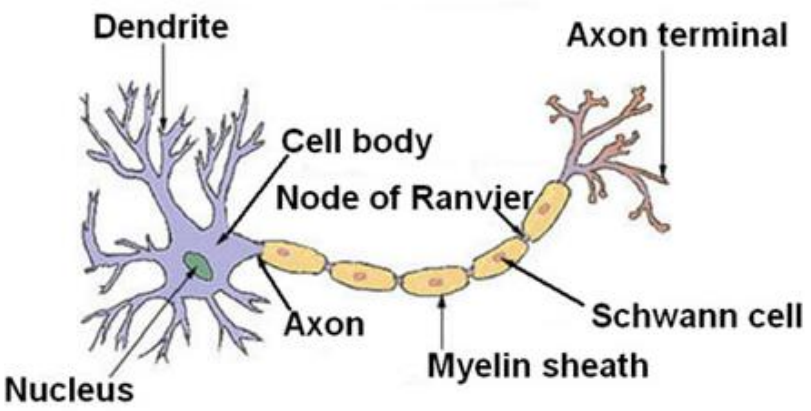
- Sisteme artificiale inspirate de structura si functionarea creierului
- Constituite din mai multe *unitati functionale interconectate (neuroni)*
- Sistem adaptiv capabil sa furnizeze raspunsuri pentru o problema dupa ce a fost antrenat pentru probleme similare

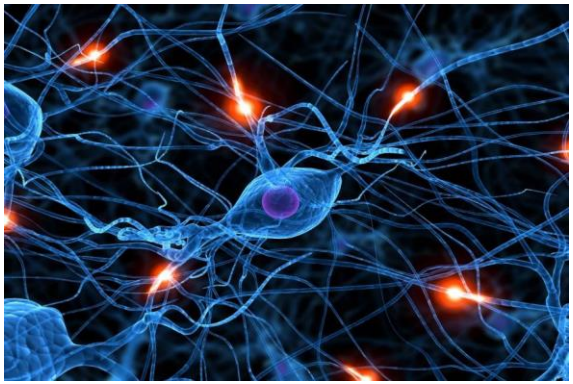


- Clasificatori de tip black-box

Retele neuronale – modelul biologic

- Creierul uman: ~10.000.000.000 de neuroni conectaţi prin sinapse
- Neuron: are un corp (soma), un axon şi multe dendrite
- Un neuron poate fi într-una din 2 stări: *activ* – dacă informaţia care intră în neuron depăşeşte un anumit prag de stimulare, *pasiv* – altfel
- Sinapsă: legătura între axon-ul unui neuron şi dendritele altui neuron
- 5.000 de conexiuni / neuron (în medie)



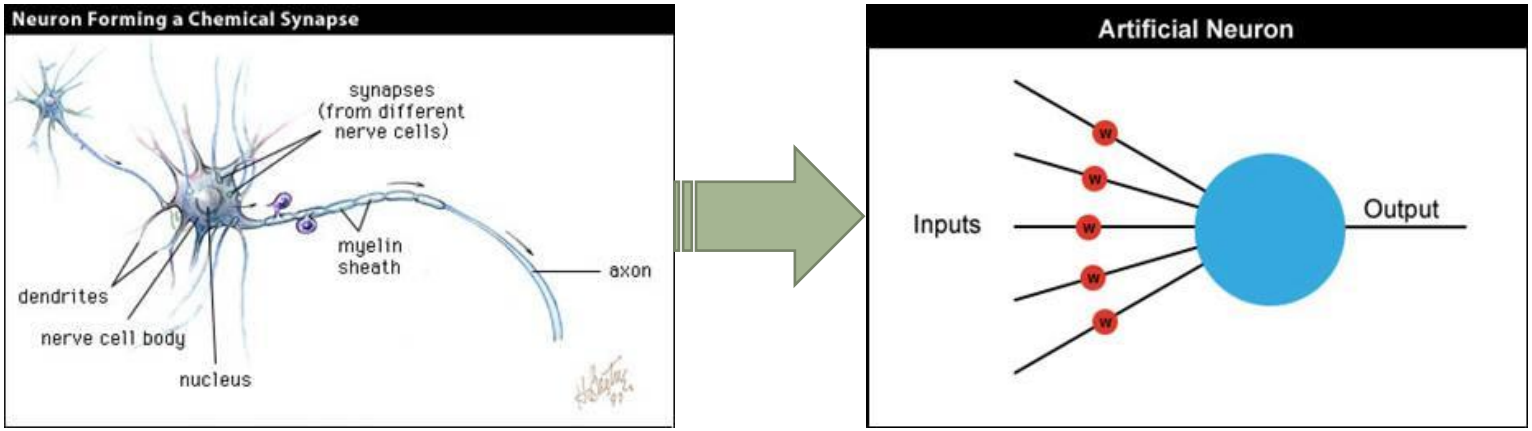


Retele neuronale

Retea neuronala: ansamblu de unitati functionale (neuroni) interconectate

Unitate functionala: model simplificat al neuronului biologic care efectueaza prelucrari simple asupra unor date de intrare

=> Model foarte simplificat la creierului



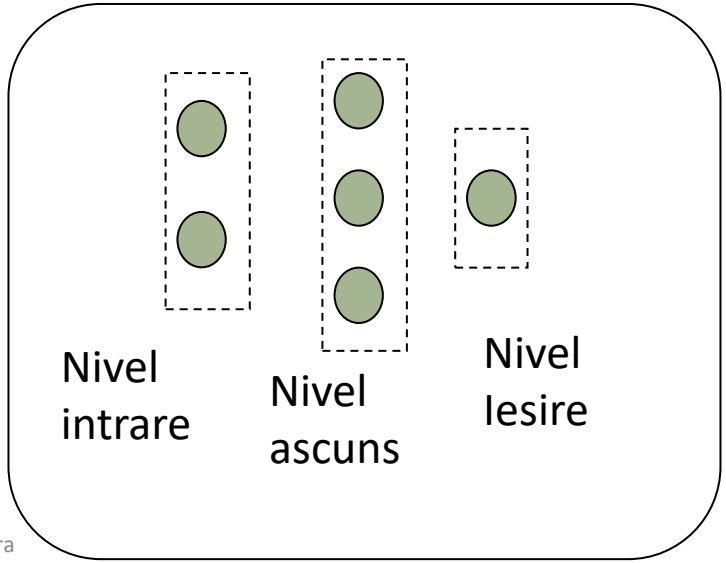
Retele neuronale artificiale (RNA)

- RNA: set de neuroni artificiali(unităţi functionale) interconectaţi
  - Fiecare neuron primeşte mai multe semnale de intrare si produce un semnal de ieşire
  - Reteaua primeşte un vector de intrare (prin neuronii de intrare) şi produce un vector de ieşire (prin neuroniide ieşire)

- Arhitectura = graf orientat etichetat; fiecare arc are asociată o pondere numerică care modelează permeabilitatea sinaptică
- Funcţionare = procesul prin care RNA transformă un vector de intrare într-un vector de ieşire
- Antrenare = procesul prin care sunt stabilite valorile ponderilor sinaptice şi ale altor parametri ai reţelei

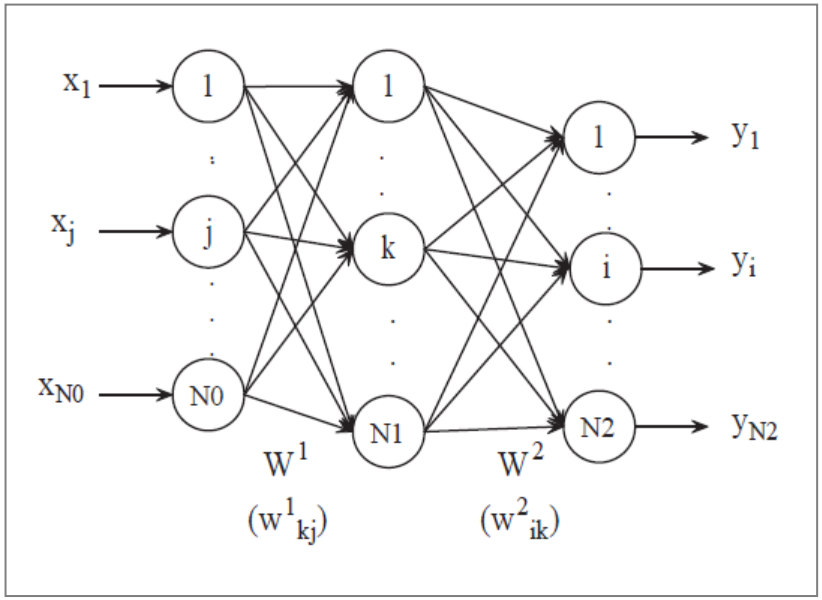
Arhitectura

- Modul in care sunt amplasate si interconectate unitatile functionale (neuronii)
- RNA ⇔ graf orientat etichetat
  - Noduri: neuroni
  - Arce: modul in care sunt conectati neuronii
  - Etichete: ponderile conexiunilor
- Retele organizate pe nivele
  - Un nivel de unitati de intrare
  - Unul sau mai multe nivele de unitati ascunse
  - Un nivel de iesire

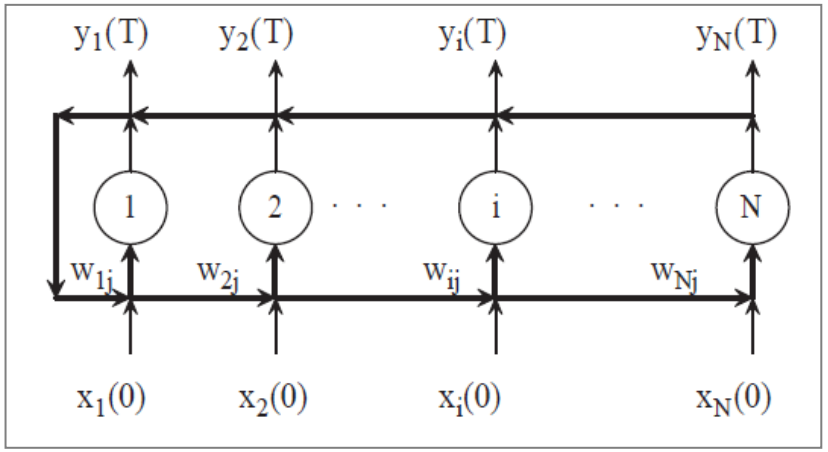


Principalele tipuri de arhitecturi

- Feed-forward
  - Graful suport nu conţine cicluri (neuronii sunt de obicei plasaţi pe mai multe nivele)
  - Semnalul de ieşire poate fi calculat prin compunerea unor funcţii de agregareşi de activare



- Recurentă
  - Graful suport conţine cicluri
  - Semnalul de ieşire este calculat prin simularea unui sistem dinamic (proces iterativ)



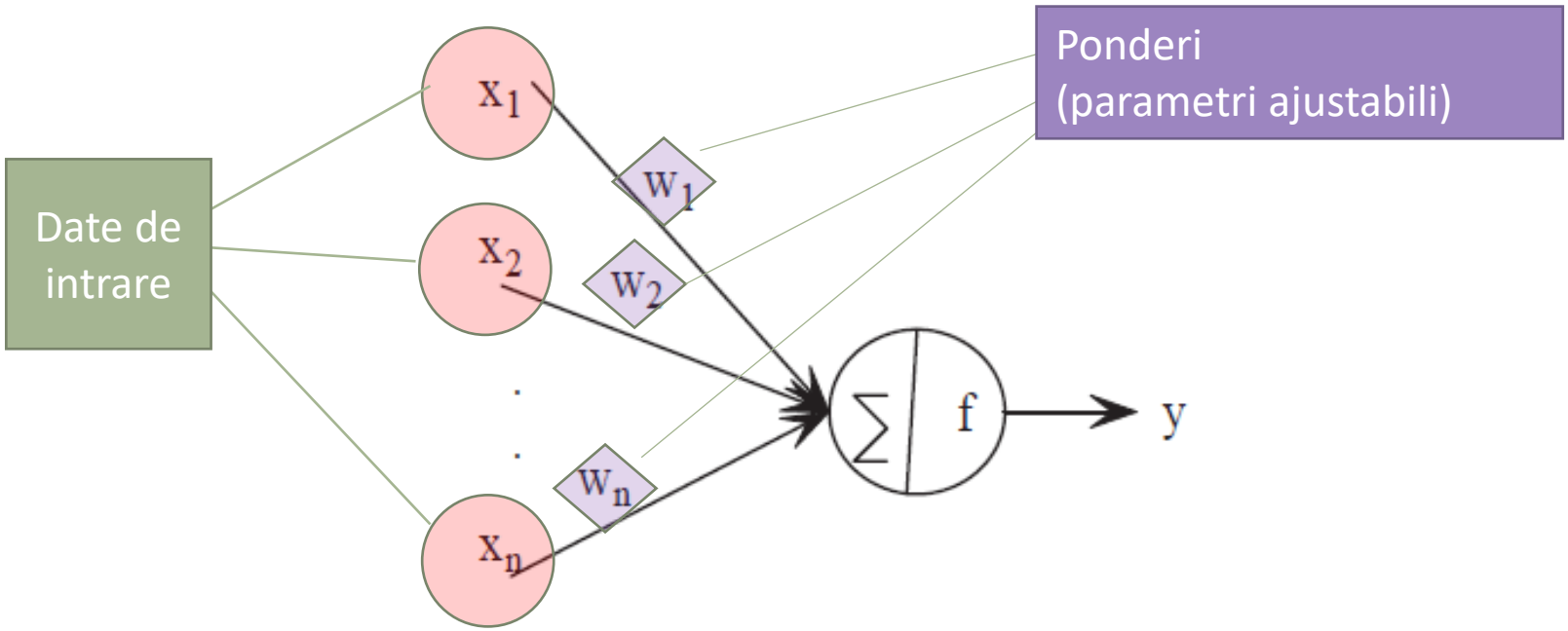
Proiectarea RNA

- Alegerea arhitecturii: număr de nivele, număr de unităţi pe fiecare nivel, funcţii de activare, tip interconectare
- Antrenare: determinarea valorilor ponderilor folosind un set de antrenare şi un algoritm de învăţare
- Validare/testare: analiza comportamentului reţelei pentru exemple care nu fac parte din setul de antrenare

- Pentru o problema de clasificare a unor date n-dimensionale in m clase reţeaua ar trebui să aibă:
  - n unităţi de intrare
  - m unităţi de ieşire

Unitati functionale

- Sau *neuroni* sau *elemente de procesare*
- Primeste semnale de intrare si produce un semnal de iesire (forma numerica)



Rolul unitatilor functionale

- Unitati de intrare
  - Primesc semnale din partea mediului
  - Rol: retransmit semnalul primit catre alti neuroni
- Unitati ascunse
  - Conectate doar cu alte unitati ale retelei
  - Nu comunica direct cu mediul extern
  - Rol: colecteaza si prelucreaza , distribuie semnalul de iesire catre alti neuroni
- Unitati de iesire
  - Rol: colecteaza semnale de la alti neuroni, le prelucreaza si transmit iesirea mediului extern

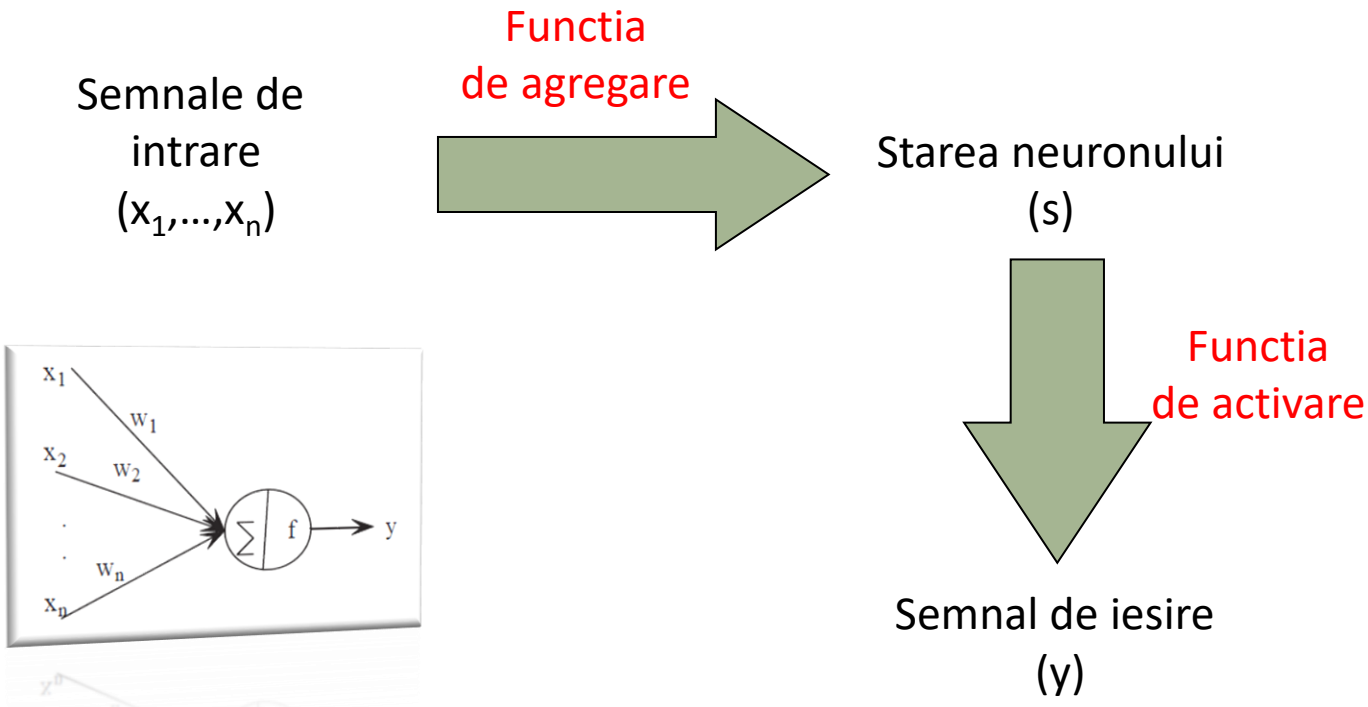
Model RNA

- Retea cu p neuroni
- Conectati printr-o multime de *ponderi sinaptice*
- Fiecare neuron are *n* intrari si o iesire *y*
- Intrari: $x_1, x_2, \ldots, x_n$ (numere reale)
- Ponderi sinaptice: $w_1, w_2, \ldots, w_n$ (numere reale)
- Fiecare neuron calculeaza starea sa interna s:

$$s = \sum_{j=1}^{n} w_j x_j$$

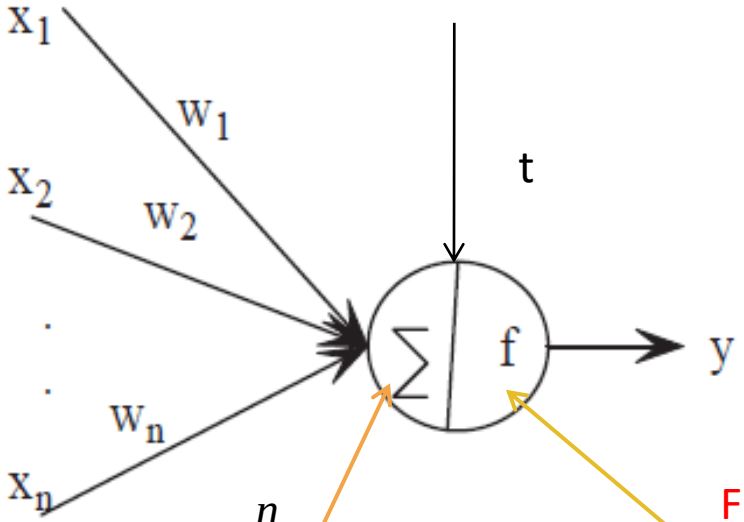
RNA: un neuron

- Combinarea semnalelor de intrare se realizează printr-o funcție de agregare
- Semnalul de ieșire aplicand o funcție de activare



RNA: un neuron

- Fiecare neuron este caracterizat de un prag de activare notat cu *t*
- Iesirea y a neuronului este +1 daca activarea totala ≥ t (modelul McCulloch-Pitts)



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

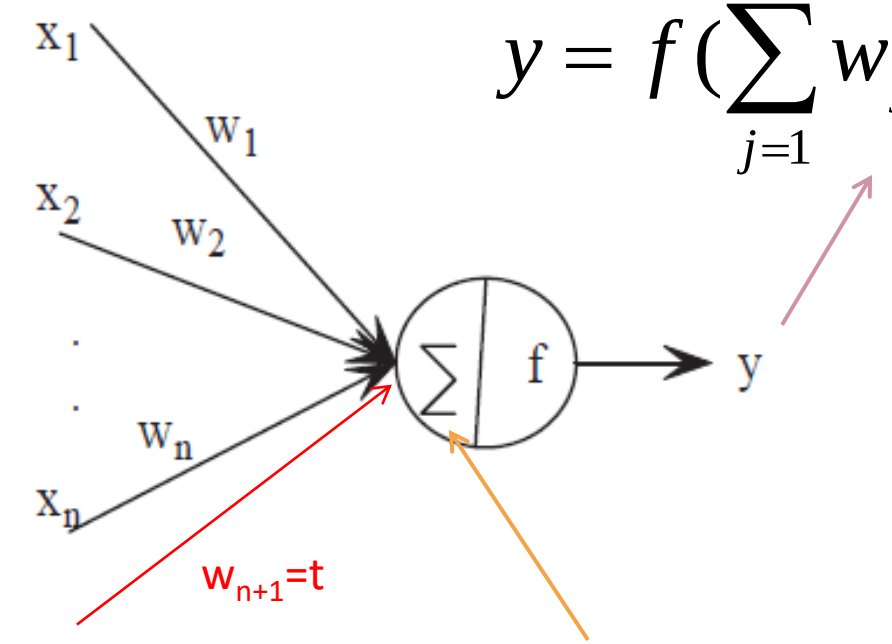
Functia de agregare $\sum_{j=1}^{n} w_j x_j$

Functia de activare, ex.

$$f : R \to R$$

$$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

RNA: un neuron



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

$$\sum_{j=1}^{n+1} w_j x_j = \sum_{j=1}^{n} w_j x_j - t$$

Functia de agregare

Suma ponderata

$$s = \sum_{i=1}^{n} w_i x_i - t$$

Distanta euclidiana

$$s = \sum_{i=1}^{n} (w_i - x_i)^2$$

Neuron multiplicativ

$$s = \prod_{i=1}^{n} x_i^{w_i}$$

Conexiuni de ordin superior

$$s = \sum_{i=1}^{n} w_i x_i + \sum_{i,j=1}^{n} w_{ij} x_i x_j + \ldots$$

Functia de activare

- Depinde de modelul de retea neuronala studiat
- Se mai numeste: functie de *raspuns*, functie *neuronala*, functie de *iesire*, functie de *transfer*

Functia prag (Heaviside)

$$f : R \to \{0,1\}$$

$$f(x) = H(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

Functia signum

$$f : R \to \{-1,1\}$$

$$f(x) = \text{sgn}(x) = \begin{cases} 1, x \geq 0 \\ -1, x < 0 \end{cases}$$

Functie liniara

$$f(x) = x$$

$$f(x) = \max(0, x)$$

Functie sigmoidala

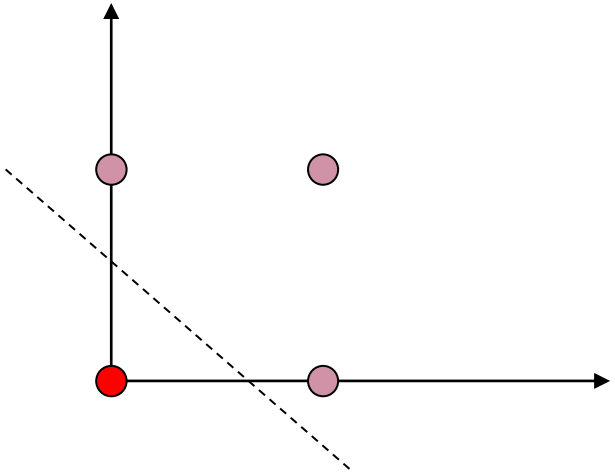
$$f : R \to (0,1)$$

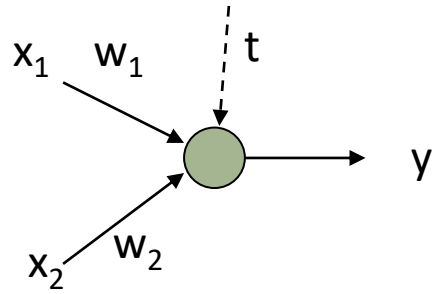
$$f(x) = \frac{1}{1+e^{-kx}}, k > 0$$

Ce poate face un singur neuron?

- Reprezenta (invata) functii booleene simple
- Rezolva probleme simple de clasificare binara (probleme linear separabile)

OR	0	1
0	0	1
1	1	1



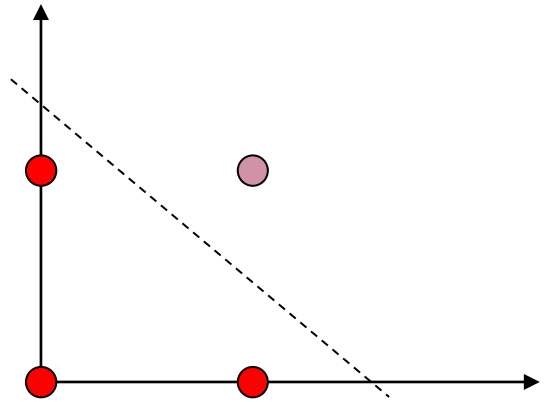


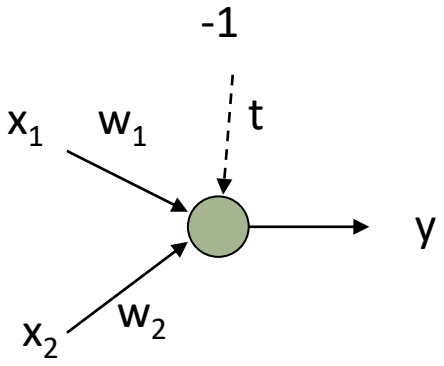
$y=H(w_1x_1+w_2x_2-t)$

Ex: $w_1=w_2=1$, $t=0.5$

Rezolvarea functiei AND

AND	0	1
0	0	0
1	0	1





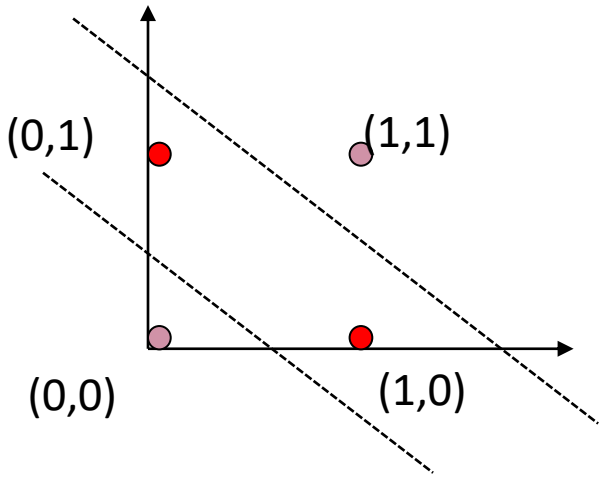
$y=H(w_1x_1+w_2x_2-t)$

Ex: $w_1=w_2=1$, $t=1.5$

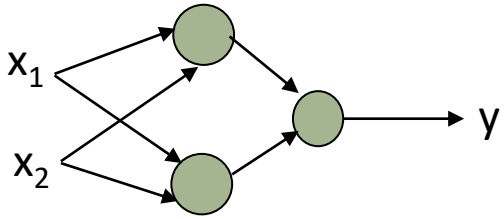
Probleme neliniar separabile

- Probleme linear separabile (ex. AND, OR): suficient o retea cu un singur nivel (cu un neuron)
- Limitele perceptronului: clase care nu sunt liniar separabile

XOR	0	1
0	0	1
1	1	0

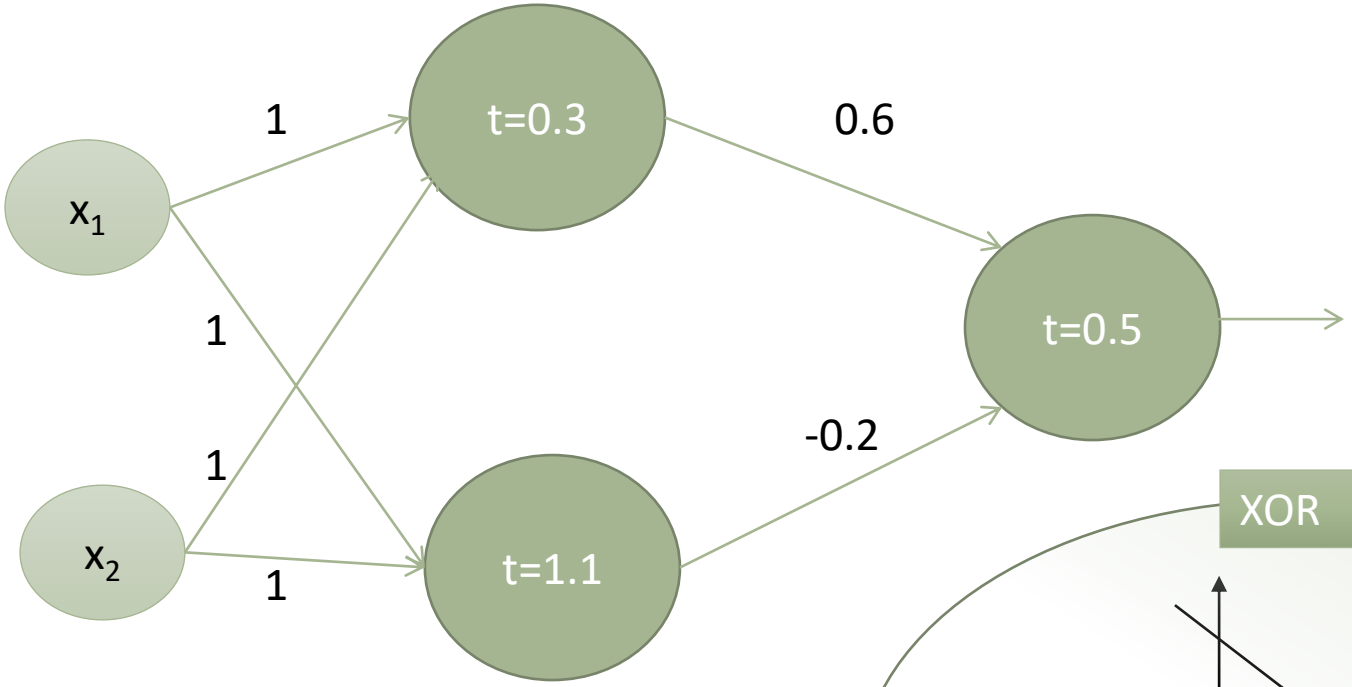


Pentru probleme neliniar spearabile este necesar cel putin un nivel ascuns!

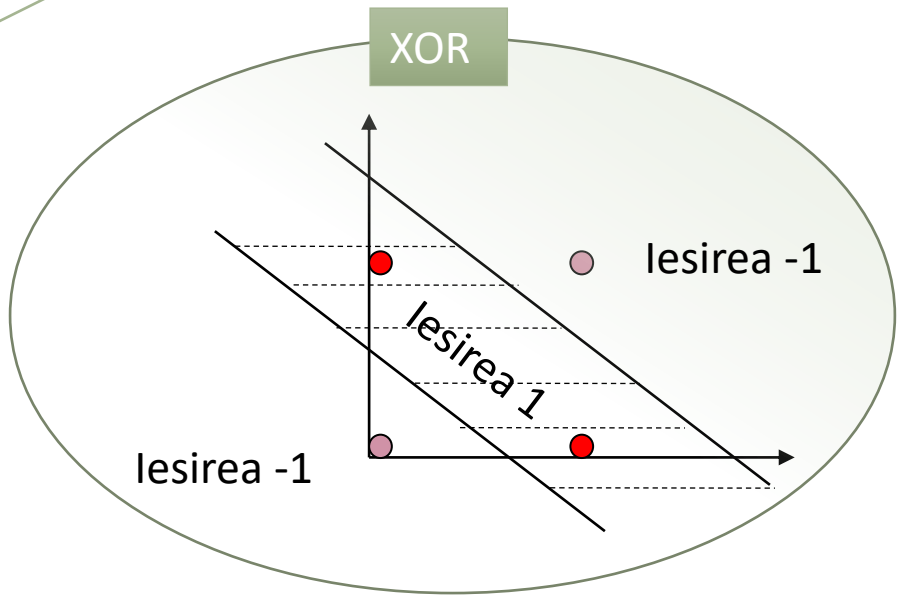


Exemplu perceptron 2 straturi

capabil sa imparta planul in 3 regiuni de decizie

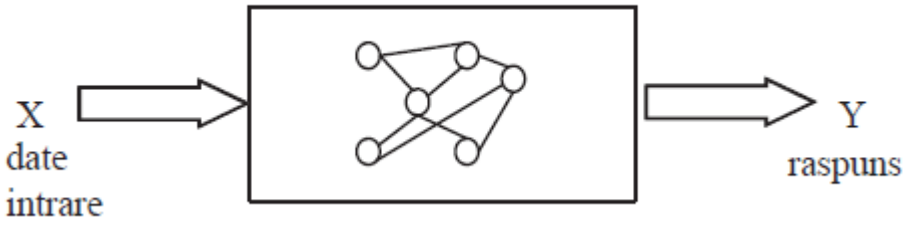


$$f(x) = \text{sgn}(x) = \begin{cases} 1, x \geq 0 \\ -1, x < 0 \end{cases}$$



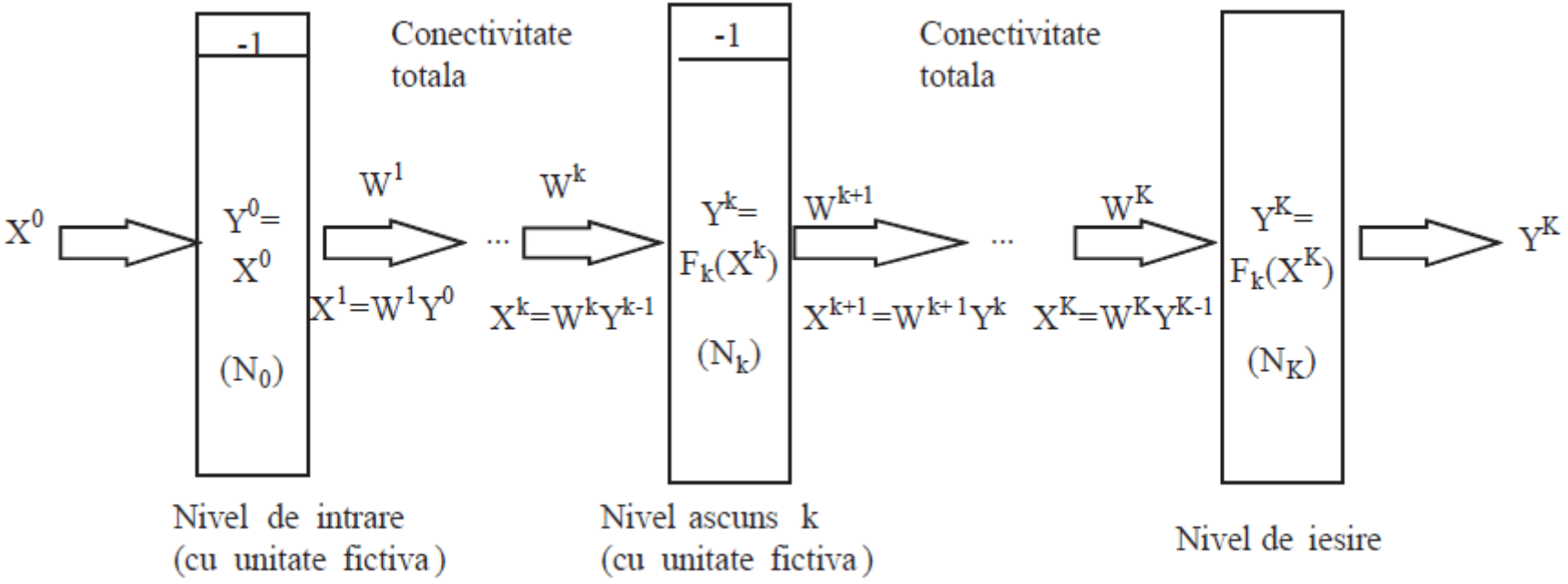
Functionare

- Modul in care reteaua transforma un semnal de intrare X intr-un semnal de iesire Y
- Depinde de modul in care functioneaza neuronii si de cum sunt conectati
- Important: *ponderile asociate conexiunilor*

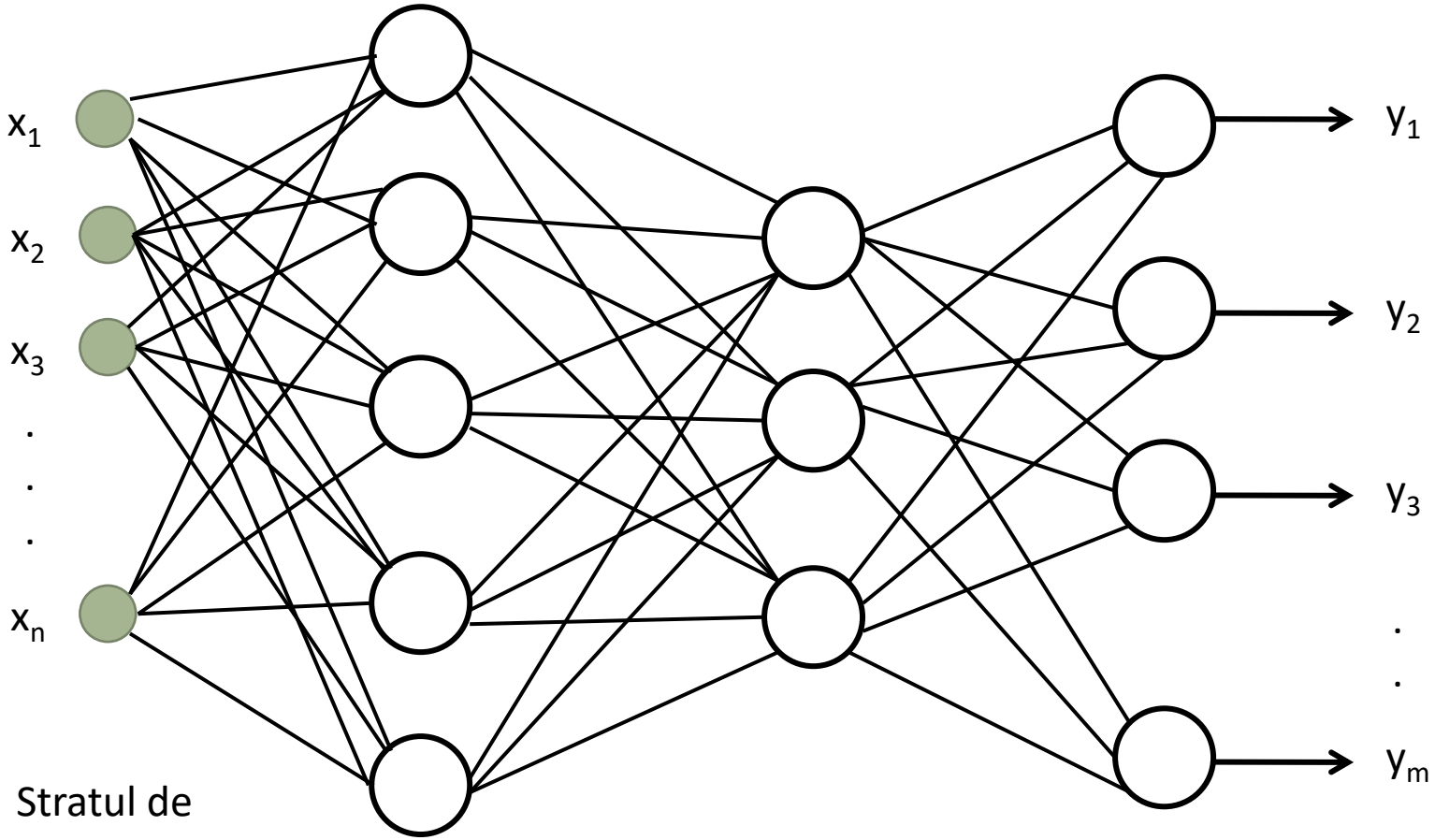


Retele feedforward

- X = vector intrare, Y= vector ieșire, F=funcție de activare



Retele feedforward: exemplu

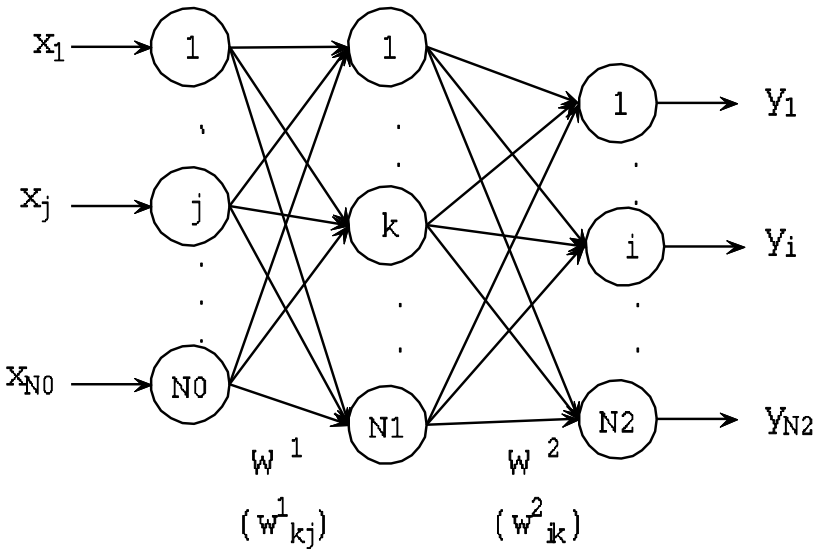


Functionare Feedforward: Y=F(X)

- Semnalul de iesire Y se poate obtine prin aplicarea unei functii asupra intrarilor
- Caz particular: 1 nivel ascuns
- Parametrii modelului: matricile cu ponder W^1 si W^2

X: vector intrare

Y: vector iesire



W^1,W^2: Matrici ponderi

$$y_i = f_2\left(\sum_{k=0}^{N1} w_{ik}^2 f_1\left(\sum_{j=0}^{N0} w_{kj}^1 x_j\right)\right),$$

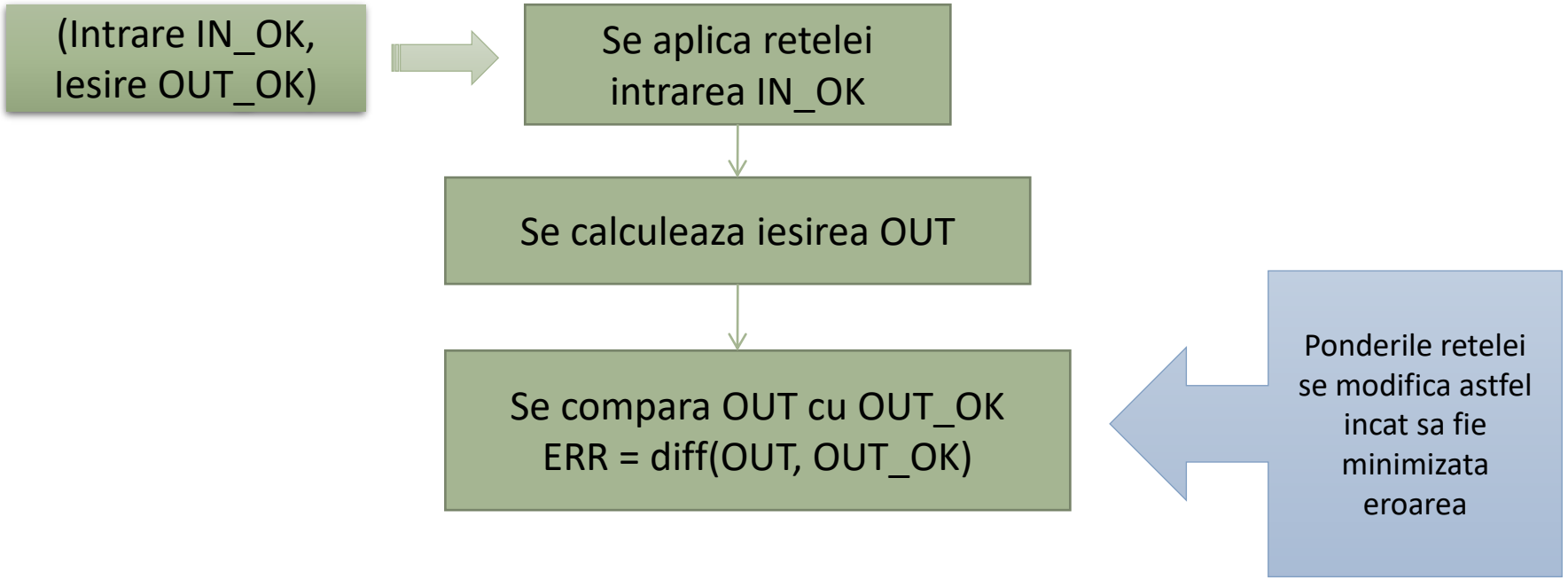
$$i = \overline{1, N2}$$

Deep Learning: rețele cu număr mare de nivele (**Deep Neural Networks**) folosite mai ales pentru recunoașterea imaginilor și a vorbirii

Antrenarea (invatarea) RNA

- Modificarea parametrilor retelei pentru o comportare adecvata problemei
- Scop: stabilirea valorii optime a ponderilor dintre 2 noduri
- Capacitatea de generalizare: RNA este capabila sa produca raspunsuri la date pentru care nu a fost antrenata
- Procesul de invatare
  - Multime de informatii
  - Algoritm de adaptare la informatiile primite
- Invatare supervizata
  - Se prezinta retelei o multime de exemple de instruire (set de antrenare)
  - Se caută valorile optime ale ponderilor între oricare 2 noduri ale reţelei prin minimizarea erorii (diferenţa între rezultatul real y şi cel calculat de către reţea)

Invatare supervizata



- Testarea retelei antrenate: se retine din setul de antrenare un subset de testare
- Asigurarea unei bune capacitati de generalizare: mentinerea unui nivel acceptabil de eroare pe setul de antrenare in scopul evitarii *suprainvatarii*

Antrenarea RNA

- Set de antrenare: $\{(x^1,d^1), \ldots, (x^L,d^L)\}$
  x= vector intrare, d= vector de ieșire corect

- Funcție de eroare (suma pătratelor erorilor)

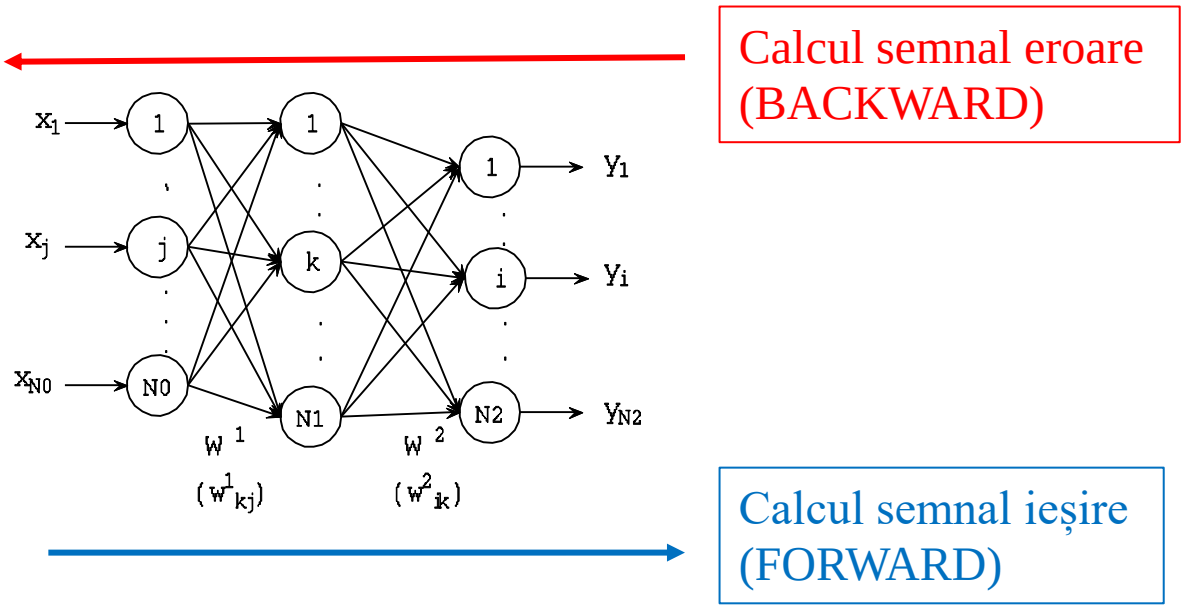
$$E(W) = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{N2}\left(d_i^l - f_2\left(\sum_{k=0}^{N1} w_{ik} f_1\left(\sum_{j=0}^{N0} w_{kj} x_j^l\right)\right)\right)^2$$

- Scopul antrenării: minimizarea funcției de eroare

- Metode de antrenare
  - Backpropagation (propagarea inapoi a erorii)
  - Algoritmi evolutivi
  - Simulated Annealing

Algoritmul Back Propagation

- Pentru fiecare exemplu din setul de antrenare:
  - se determină semnalul de ieşire
  - se calculează eroarea la nivelul de ieşire
  - se propagă eroarea înapoi în reţea şi se reţine factorul delta corespunzător fiecărei ponderi
  - se aplică ajustarea corespunzătoare fiecărei ponderi



Algoritmul Back Propagation

Initializarea aleatoare a ponderilor: aleator din [0,1] sau [-1,1]

REPEAT

FOR i = 1, L **DO**

Etapa FORWARD: Se propagă informaţia înainte şi se calculează ieşirea corespunzătoare fiecărui neuron al reţelei

$$x_k^l = \sum_{j=0}^{N0} w_{kj}^1 x_j^l,\ y_k^l = f_1(x_k^l),\ x_i^l = \sum_{k=0}^{N1} w_{ik}^2 y_k^l,\ y_i^l = f_2(x_i^l)$$

Etapa BACKWORD: Se stabileşte şi se propagă eroarea înapoi

$$\delta_i^l = f_2^{'}(x_i^l)(d_i^l - y_i^l),\ \delta_k^l = f_1^{'}(x_k^l)\sum_{i=1}^{N2} w_{ik}^2 \delta_i^l$$

Etapa de ajustare: Se ajustează ponderile

$$w_{kj}^1 = w_{kj}^1 + \eta \delta_k^l x_j^l,\ \ w_{ik}^2 = w_{ik}^2 + \eta \delta_i^l y_k^l$$

Probleme

- Viteza mica de convergenta - eroarea descreste prea incet
- Oscilatii - valoarea erorii oscileaza in loc sa descreasca in mod constant
- Minime locale - procesul de invatare se blocheaza intr-un minim local al functiei de eroare
- Supra-antrenarea si capacitatea limitata de generalizare

Rezolvarea unei probleme -RNA

- Stabilirea arhitecturii initiale
- Alegerea tipului neuronilor
- Stabilirea parametrilor ajustabili
  - O instantiere a parametrilor ⇔ O anumita functie asociata RNA
- Algoritmul de invatare trebuie sa fie potrivit cu arhitectura RNA si cu cantitatea de informatie de care se dispune despre problema
- Antrenarea RNA pentru a rezolva o anumita problema
- Testarea RNA ⇔ verificarea corectitudinii raspunsurilor RNA cand primeste date de intrare care nu apartin multimii de instruire dar pentru care se cunoaste raspunsul corect
- Utilizarea RNA

Cursul urmator...

Modele hibride