8.11.2022

- noțiuni / concepte
- clasificare / grupări

Inteligență Artificială

Cursul 8

Rețele Neuronale

Istoric

- '43 → a fost propus primul model matematic, dar care nu putea învăța
- '51 → primul circuit electronic
- '58 → primul perceptron (= o rețea neuronală formată dintr-o singură unitate
  - nu poate învăța orice funcție (ex. XOR) motiv.
  - pt. care statul a stat până ce s-a descoperit că pot fi folosiți mai mulți perceptroni.
- 2006 → se pun bazele Deep Neural Network

Ex de rețele neuronale

- recunoașterea caracterelor de mână
- recunoaștere facială
- recunoașterea limbajului natural și traducere

Se antrenează pe seturi de date foarte mari

Modelul biologic (neuronul) pt a putea face legături cu modelul artificial

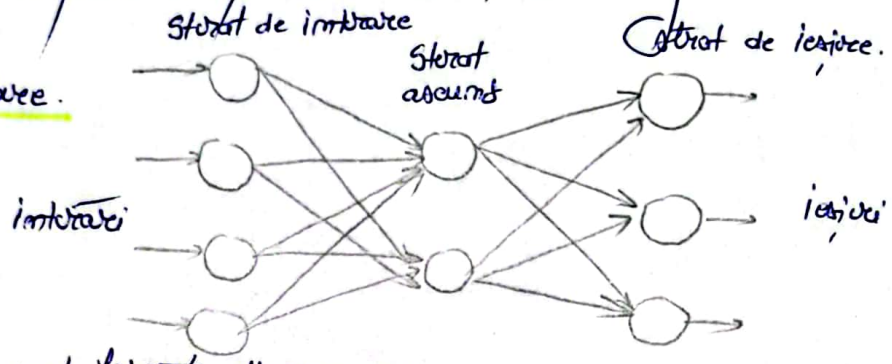
Modelul artificial este inspirat din biologie: neuronii se conectează cu ajutorul dendritelor și o sinapsă și astfel este transmis semnalul electric (informația) de la un neuron la altul. prin axon.

ex de rețea neuronală cu mai multe unități.

Vom avea o rețea → stratul de intrare
→ stratul ascuns } → reprezentate sub forma unui
→ stratul de ieșire } digraf cu un singur Arc.

Antrenarea presupune determinarea parametrilor rețelei, pornind de la datele de antrenare.

Acestea reprezintă sisteme adaptive de tip „black box", care extrag un model printr-un proces de învățare.



Rețelele neuronale artificiale sunt formate din unități, care reprezintă modele funcționale.
computaționale simplificate ale neuronului.

- semnale de intrare
- ponderi sinaptice atașate conexiunilor.
- prag de activare
- ieșire

RN biologică	RN artificială
corpul celulei	neuron
dendrite	intrări
axon	ieșire
sinapsă	pondere

Corpul celulei calculează o funcție.

Funcție de activare → va avea un output care este transmis prin axon

Tipuri de învățare
- supervizată (datele sunt etichetate, adică pt fiecare știm clasa căreia aparține)
- nesupervizată (datele nu sunt etichetate, deci clasele le vom identifica doar pe baza atributelor pe care le avem)
- întărire

Învățarea supervizată
- clasificare → dată o mulțime de instanțe, să se identifice clasa
- regresie → dată o succesiune de valori, să se determine relația dintre 2 sau mai multe variabile.

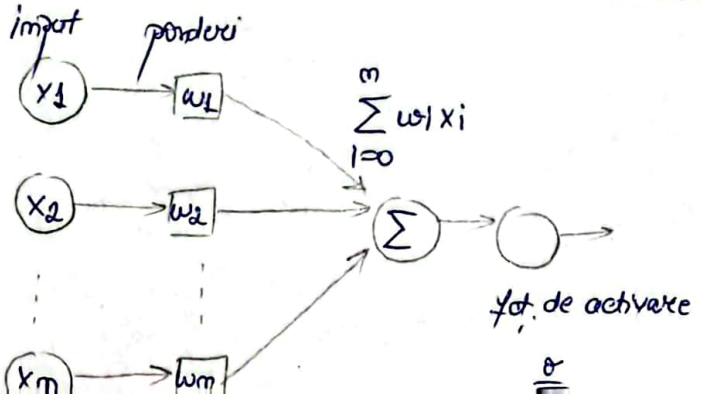
Diferența dintre clasificare și regresie: tipul ieșirii este diferit:
- clasificare → discret
- regresie → continuu.

Perceptronul - Antrenarea perceptronului.

Un perceptron are ca intrare valori $x_2, \dots x_m$ și valorile asociate $w_1, w_2, \dots, w_m$.

Se va calcula o combinație liniară (Σ) și se va trece rezultatul printr-o fct. de activare.

(Se activează neuronul doar în anumite situații).



$$\theta(x_1, \dots, x_m) = \begin{cases} 1 \text{ dacă } w_0 + w_1 x_1 + \dots + w_m x_m > 0 \\ -1 \text{ altfel.} \end{cases}$$

③

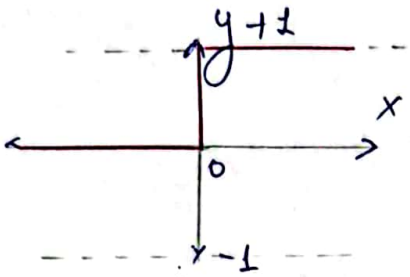
Învățarea perceptronului prin alegerea ponderilor $w_0 \dots w_m$.

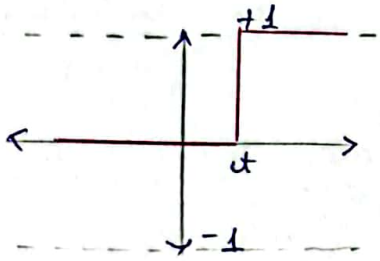
$\sum_{i=0}^{m} w_i x_i > 0$ sau $\vec{w} \cdot \vec{x} > 0$.

Notație vectorială: $o(\vec{x}) = F(\vec{w} \cdot \vec{x})$

Funcția de activare treaptă

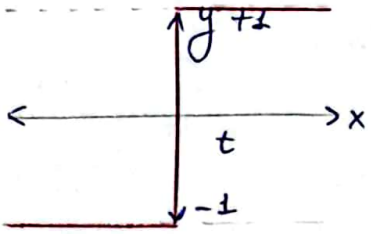
$$F(y) = \begin{cases} 1 & \text{dacă } y \geq 0 \\ 0 & \text{altfel} \end{cases}$$

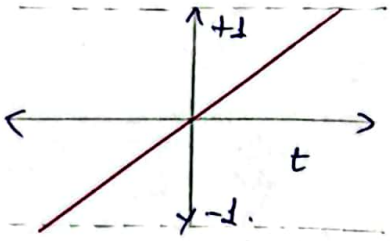




Funcția de activare semn.

$$F(y) = \begin{cases} 1 & \text{dacă } y \geq 0 \\ -1 & \text{altfel.} \end{cases}$$





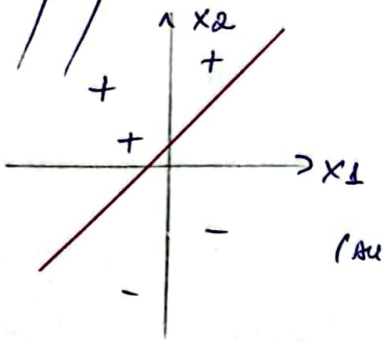
Scopul perceptronului este de o clasifica un set de date în clase pozitive și negative.

2 clase, ca la Rd.

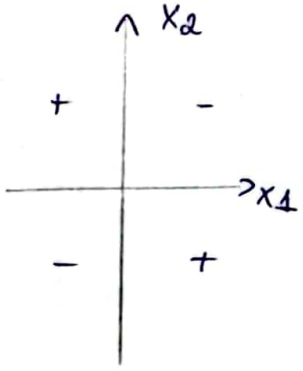
Există date care nu sunt separabile liniar, date există șiruri (ex XOR) care nu pot fi separate de o linie în cele 2 clase.

$\vec{w} \cdot \vec{x} > 0 \Rightarrow o = 1$ și $o = -1$.

Ec. hiperplanului: $\vec{w} \cdot \vec{x} = 0$.



(sunt separabile)



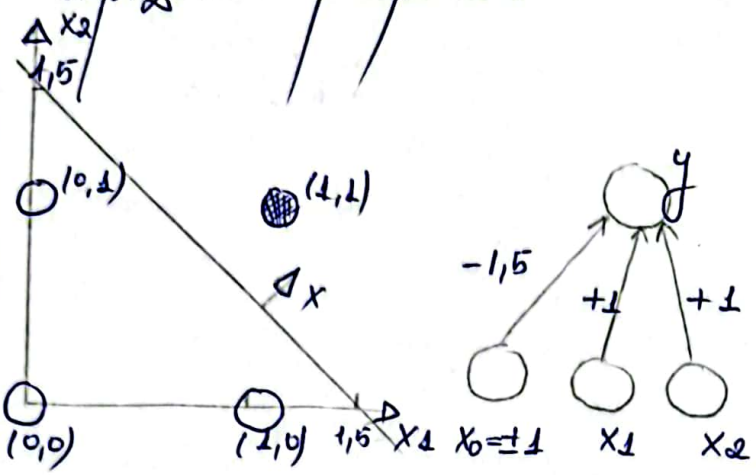
(nu sunt separabile liniar)

Un perceptron poate fi utilizat și în reprezentarea funcțiilor booleene.

ex1) AND, vom lua ponderile $w_0 = -1,5$, $w_1 = w_2 = 1$.

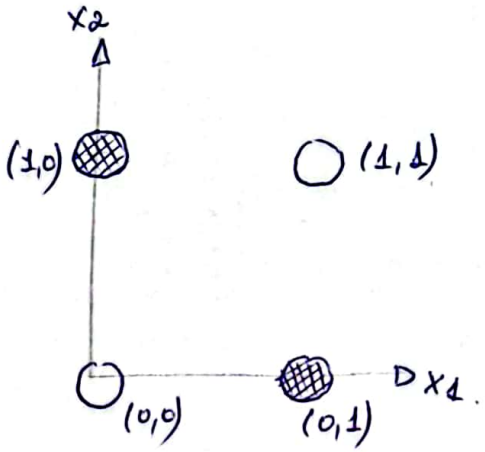
x_1	x_2	y_c
0	0	0
0	1	0
1	0	0
1	1	1

reprezentarea perceptronului.



ex.2) XOR. Nu poate fi reprezentată de un singur perceptron, dar poate fi printr-o rețea de perceptroni.

x_1	x_2	y_c
0	0	0
1	0	1
0	1	1
1	1	0



Yet, booleană poate fi reprezentată de o rețea de unități interconectate.

Reguli de antrenare

Pasul 1)

1) Regula de antrenare a perceptronului → prin învățarea ponderilor. (identifică vectorul de ponderi. a.î. perceptronul să returneze ieșirea corectă

Pasul 2) Generează ponderi aleatorii → calculează ieșirea pt. fiecare exemplu de antrenare și modifică ponderile. atunci când clasifică greșit un exemplu.

Pasul 1+2 se repetă până când perceptronul învață să clasifice corect toate datele.

Ponderile sunt modificate conform regulii de antrenare a perceptronului:

$$w_i = w_i + \Delta w_i, \text{ unde } \Delta w_i = \eta (t - o) x_i$$

- t = ieșirea dorită pt. exemplul de antrenare.
- o = ieșirea generată de perceptron
- η = rata de învățare (const pozitivă)

Intuitiv.
- dacă exemplul este clasificat corect $t - o = 0$, $\Delta w_i = 0 \Rightarrow$ ponderile nu sunt actualizate
- dacă perceptronul returnează -1 atunci când ieșirea corectă este $+1$.

Și $\eta = 0,1$, $x_i = 0,8$, atunci $\Delta w_i = 0,1 (1 - (-1)) \cdot 0,8 = 0,16$.

- dacă perceptronul returnează $+1$ când ieșirea corectă este -1, atunci ponderea scade.

Perceptronul poate fi folosit atunci când datele sunt separate liniar $\Rightarrow$ procedura converge.

atunci când vectorul de ponderi clasifică toate exemplele de antrenare.

2. Regula delta → utilizează gradient descent pt. a căuta în spațiul vectorilor de ponderi

Dorim antrenarea unei unități liniare pt. care ieșirea este $o(\vec{x}) = \vec{w} \cdot \vec{x}$.

Eroarea la antrenare pt. vectorul de ponderi w este: $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$

Reprezentarea spațiului este o f(d) convexă cu un singur minim local. Alg trebuie să poată identifica acest minimum pt o identifica direcția dată de Gradient Descent.

Stochastic Gradient Descent rezolvă { problema convergență lentă / existența mai multor minime locale

Aceasta pp. actualizarea ponderilor incremental, calculând eroarea pt. fiecare exemplu individual: $\Delta w_i = \eta (t-o) x_i$ { t = valoarea dorită; o = ieșire reală; x_i = a i-a intrare pt. exemplul de antrenare

regula Delta

Rețele neuronale multi-strat = o rețea cu propagare înainte (Feed-forward) care conține { → un strat de intrare; → unul sau mai multe straturi ascunse; → strat de ieșire } se realizează calculele

Eroarea este propagată înapoi către stratul ascuns pt. a actualiza ponderile.

ex. rețea antrenată să recunoască limbile locale.

Semnalul vocal este reprezentat de 2 parametri numerici, obținuți din analiza spectrală a sunetului.

Prop. de aproximare universală: o rețea neuronală cu un strat ascuns, cu un nr. posibil infinit de neuroni, poate aproxima orice f(d) reală continuă.

Unitatea sigmoid. · calculează o combinație liniară a intrărilor, apoi aplică un prag.

Ieșirea este o fct continuă a intrărilor: $o = \sigma(\vec{w} \cdot \vec{x})$, $\sigma(y) = \frac{1}{1+e^{-y}}$

σ = fct. sigmoidă

$\frac{d\sigma(y)}{dy} = \sigma(y) \cdot (1-\sigma(y))$ → derivata

Pentru antrenarea unei rețele neuronale multistrat se folosește alg. **backpropagation**.

Alg. are 2 faze: se propagă semnalul înainte, către stratul de output
eroarea se propagă înapoi, de la stratul de output la stratul ascuns.

Pași. 1) **Inițializarea**: alege nr. de intrări, unități ascunse și de ieșire, ponderile și prag urile de valori.

2) **Activarea**: rețeaua se activează prin aplicarea vect. de antrenare $\vec{x}$
se calculează ieșirile neuronilor din stratul ascuns.
se calculează ieșirile neuronilor din stratul de ieșire: $o: \sigma(\vec{w} \cdot \vec{x})$

epocă = iterarea peste toate exemplele de antrenare

Se antrenează până când eroarea medie pătratică ajunge sub un prag acceptabil