

Algoritmul de învățare al perceptronului

Rețele neuronale reprezintă un model matematic inspirat de principiile funcționării unui creier biologic. Creierul uman are în componență zeci de miliarde de *neuroni*, fiecare neuron fiind conectat cu zeci de mii de alți neuroni. Conexiunile dintre neuroni se realizează prin *sinapse*. Simplificând, un neuron funcționează în felul următor (Figura 1 - stânga): neuronul primește *semnale electrice* de la *axonii* (pre-sinapse) altor neuroni, ce trec prin sinapse, prin intermediul *dendritelor* (post-sinapse), procesează aceste semnale iar dacă semnalul electric este suficient de puternic neuronul se *activează* și emite un semnal electric prin intermediul axonilor.

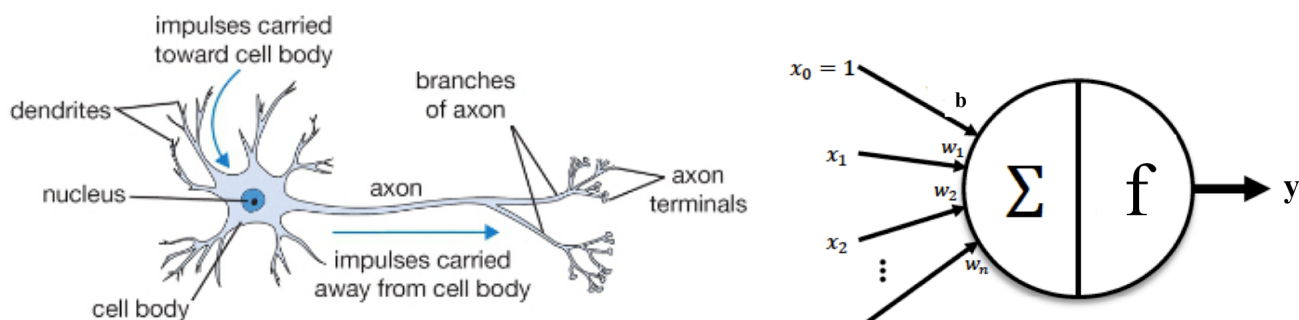


Figura 1. Stânga: structura unui neuron biologic. Dreapta: structura unui neuron artificial (modelul McCulloch-Pitts)

Primul model computațional al unui neuron a fost propus de cercetătorii McCulloch și Pitts (Figura 1 - dreapta). Deseori, acest model este asociat cu un perceptron. Semnalele electrice primite de la alți neuroni sunt modelate ca *valori numerice de intrare* (intrări - x_1, x_2, \dots, x_n), procesarea semnalelor este modelată printr-o *funcție de integrare* (de obicei funcția de integrare liniară = suma) aplicată intrărilor care sunt ponderate (cu ponderile w_1, w_2, \dots, w_n), activarea neuronului este modelată de o *funcție de transfer* f . Neuronul artificial are bias-ul b (starea curentă a neuronului), care este adunat intrărilor ponderate (semnalul $x_0 = 1$), rezultând argumentul funcției de transfer. O funcție de transfer f cu care vom lucra în acest laborator este funcția :

$$\text{hardlims}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x \leq 0 \end{cases}$$

Mulțimi liniar separabile

Două mulțimi de puncte dintr-un spațiu n -dimensional se numesc liniar separabile dacă ele pot fi separate de un hiperplan. În acest caz identificăm cele două mulțimi de puncte cu două clase (vom folosi în cele ce urmează clasele -1 și 1). Problemele de clasificare binară în care cele două clase sunt liniar separabile pot fi rezolvate cu succes de un perceptron.

Un perceptron cu vectorul de ponderi $\mathbf{w} = (w_1, w_2, \dots, w_n)$ și bias-ul b implementează pentru vectorul de intrare $\mathbf{x} = (x_1, x_2, \dots, x_n)$ ecuația hiperplanului dat de: $\mathbf{w}^T \mathbf{x} + b = 0$. Pentru funcția de transfer *hardlims*, vectorul \mathbf{x} este clasificat în clasa $y = \text{hardlims}(\mathbf{w}^T \mathbf{x} + b)$.

Pentru problemele de clasificare binare liniar separabile, *algoritmul de învățare a perceptronului* (propus de Rosenblatt în 1957) arată că într-un număr finit de pași, putem găsi (*învăța, antrena*) un vector de ponderi \mathbf{w}^* și un bias b^* care definesc un hiperplan de separare al celor două clase.

Pentru mulțimea de antrenare $S = \{(\mathbf{x}^1, t^1), (\mathbf{x}^2, t^2), \dots, (\mathbf{x}^m, t^m)\}$ vrem să găsim \mathbf{w}^* și b^* astfel încât pentru fiecare exemplu \mathbf{x}^i ieșirea y^i a perceptronului să fie aceeași cu eticheta t^i , adică $y^i = \text{hardlims}(\mathbf{w}^{*T} \mathbf{x}^i + b^*)$ să fie egală cu t^i . O variantă a algoritmului de învățare a perceptronului este cea *incrementală* (sau *online*). În

care vectorul de ponderi \mathbf{w} și bias-ul b curenți sunt modificați după fiecare exemplu misclasificat. În general, pentru a obține soluția (\mathbf{w}^*, b^*) ce definește hiperplanul de separare, algoritmul necesită să parcurgă mulțimea de antrenare a exemplurilor de mai multe ori. O singură parcurgere a mulțimii de antrenare se numește *epocă*. Algoritmul se oprește în momentul în care de-a lungul unei epoci nu se modifică vectorul de ponderi \mathbf{w} și bias-ul b curenți, semn că toate exemplele sunt clasificate corect cu parametri actuali. În cele ce urmează prezentăm algoritmul lui Rosenblatt, varianta online.

Algoritmul de învățare al perceptronului, varianta online

Input: mulțimea de antrenare $S = \{(\mathbf{x}^1, t^1), (\mathbf{x}^2, t^2), \dots, (\mathbf{x}^m, t^m)\}$, cu $t^i = 1$ sau -1

Output: vectorul de ponderi \mathbf{w}^* și bias-ul b^* (pentru cazul de liniar separabilitate aceștia determină hiperplanul de separare);

Iterația 0: Inițializează \mathbf{w} și b : $\mathbf{w} = \mathbf{w}^0$, $b = b^0$, $k = 0$ iterația curentă.

Iterația $k+1$:

1. se selectează exemplul de antrenare \mathbf{x}^i cu $i = (k \bmod m) + 1$;
2. se calculează y^i , eticheta exemplului \mathbf{x}^i , $y^i = \text{hardlims}((\mathbf{w}^k)^T \mathbf{x}^i + b^k)$;
3. dacă $y^i = t^i$, se păstrează parametri curenți, adică $\mathbf{w}^{k+1} = \mathbf{w}^k$, $b^{k+1} = b^k$, altfel se modifică parametri astfel: $\mathbf{w}^{k+1} = \mathbf{w}^k + t^i \mathbf{x}^i$, $b^{k+1} = b^k + t^i$
4. dacă k e multiplul lui m (suntem la sfârșitul epocii k/m)
 - a. dacă în epoca k/m nu s-au modificat parametri \mathbf{w} și b atunci $\mathbf{w}^* = \mathbf{w}$, $b^* = b$. Algoritmul se termină.
5. $k = k+1$;

Codul de la care porniți în acest laborator implementează algoritmul de învățare al lui Rosenblatt pentru mulțimea de antrenare $S = \{((0,0),-1), ((0,1),1), ((1,0),1), ((1,1),1)\}$. De asemenea, codul vă prezintă și implementarea din scikit-learn a perceptronului.

Realizați următoarele:

- a. parcurgeți codul deja scris și observați cele două implementări. Ce se întâmplă dacă folosiți clasele 0 și 1 în algoritmul de învățare al lui Rosenblatt? Cum procedați în acest caz pentru a putea învăța hiperplanul de separare dintre cele două clase?
- b. generați un nor de 1000 de puncte (x_{i1}, x_{i2}) în pătratul de dimensiuni $[-1, 1] \times [-1, 1]$ și etichetați fiecare punct (x_{i1}, x_{i2}) cu eticheta 1 dacă $x_{i1} - x_{i2} \geq 0$ sau cu eticheta 0 altfel. Plotați punctele în plan asociind o culoare punctelor cu aceeași etichetă. Antrenați un perceptron care clasifică corect aceste puncte. Plotați dreapta de separare.
- c. scrieți codul Python care implementează un perceptron ce rezolvă corect problema de separare în cazul în care avem 4 puncte $(0,0)$, $(1,0)$, $(0,1)$, $(1,1)$ iar etichetele lor sunt date de operația AND aplicate componentelor punctelor. Puteți scrie un perceptron care rezolvă corect problema pentru cazul XOR?
- d. Pentru mulțimi liniar neseeparabile algoritmul lui Rosenblatt nu va converge. O variantă posibilă este de a returna parametri \mathbf{w} și b care au cel mai mare număr de exemple consecutive care sunt clasificate corect (pentru fiecare epocă alegeți exemplele în mod aleator generând o permutare a indicilor). Cum ați implementa un asemenea algoritm?