# Introducere în rețele neuronale

## 1. Introducere

Pornind de la modelul biologic al neuronului, în anul 1943, cercetătorii Warren S. McCulloch și Walter Pitts au propus pentru prima data în revista Bulletin of Mathematical Biophysics un model matematic capabil să imite în linii mari modelul biologic neuronal.

Neuronul McCulloch-Pitts este un neuron ce se activează binar. Asemeni neuronului biologic, acesta are sinapse excitatorii și inhibitorii, în funcție de valorile ponderilor. Astfel, conexiunile excitatorii au ponderi pozitive, pe când cele inhibitorii au ponderi negative, iar neuronul este asociat cu o valoare de prag. Neuronul se activează dacă intrarea este mai mare decât valoarea pragului și rămâne inactiv daca valoarea de intrare este mai mică decât pragul acestuia. Arhitectura neuronului este prezentată în Fig. 1.1, unde x reprezintă mărimile de intrare, w reprezintă ponderile și y ieșirea. Ponderile rețelei pot lua doar valorile -1 sau 1, -1 fiind pentru conexiune inhibitorie și 1 pentru cea excitatorie, iar neuronul este activat după formula:

$$\sum_{i=1}^{n} wi * xi \geq T \tag{1.1}$$

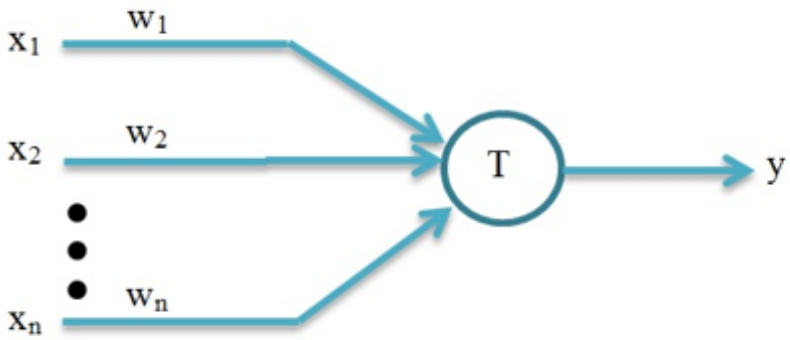


Fig. 1.1 – Arhitectura neuronului McCulloch-Pitts

Mai târziu, în anul 1957, Frank Rosenblatt a dezvoltat algoritmul Perceptron, algoritm ce intenționa să fie o mașinărie, mai degrabă decât un program, utilizată pentru recunoașterea de imagini. Acest algoritm, spre deosebire de cel dezvoltat de McCulloch-Pitts, este unul supervizat, care ajustează ponderile astfel încât eroarea de clasificare să fie minimă.

Perceptronul este cea mai simplă formă a rețelelor neuronale utilizat pentru clasificarea pattern-urilor linear separabile. Acesta conține un singur neuron cu sinapse ce pot fi ajustate prin modificarea ponderilor și a bias-ului. Prin acest algoritm, Rosenblatt a demonstrat că dacă două clase sunt liniar separabile, atunci perceptronul converge către o suprafață de decizie de forma unui hiperplan între cele două clase. Deoarece perceptronul are un singur neuron, acesta este limitat la clasificarea vectorilor în doar două clase. Spre deosebire de algoritmul dezvoltat de McCulloch și Pitts, neuronul nu mai este reprezentat de un prag, ci acesta constituie o funcție de activare, de forma:

$$f(k) = \begin{cases} 0, dacă\ k < 0 \\ 1, dacă\ k > 0 \end{cases}, \text{ unde } k = \sum_{i=1}^{n} wi * xi \tag{1.2}$$

Ieșirea rețelei fiind calculată astfel:

$$y = f(k) \tag{1.3}$$

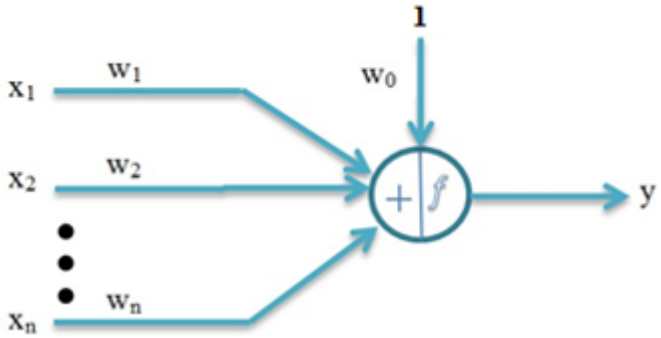


Fig. 1.2 – Arhitectura perceptronului simplu

În Fig. 1.2 este prezentată arhitectura neuronului perceptron, unde cu *x* sunt notate mărimile de intrare, cu *w* ponderile și cu *y* ieșirea neuronului.

Mai târziu, în anul 1960, Bernard Widrow și Marcian E. Hoff au introdus ADALINE (ADAptive LInear NEuron), un sistem adaptiv, rapid și precis, fiind prima rețea neuronală utilizată pe scară largă. ADALINE putea fi găsit în aproape orice telefon analog ca metodă de eliminare a ecoului în timp real. Îmbunătățirea rețelei ADALINE, în comparație cu Perceptron a fost adusă de antrenarea acesteia cu regula Widrow-Hoff, denumită și regula delta sau algoritmul LMS (least mean square).

Regula Widrow-Hoff reprezintă un algoritm de gradient negativ care are rolul de a minimiza eroarea medie pătratică. Astfel, presupunem intrarea alcătuită dintr-un singur vector **X**, de n caracteristici, intrare ce are ieșirea corectă notată cu d (d fiind un scalar), iar ponderile curente vor fi notate cu **W** (**W** fiind un vector de n valori). Atunci, eroarea va fi:

$$E(\boldsymbol{W}) = (d - \boldsymbol{W}^T * \boldsymbol{X})^2 \tag{1.4}$$

Iar gradientul este dat de formula:

$$\nabla E = \langle \frac{\partial E}{\partial w_0} \dots \frac{\partial E}{\partial w_n} \rangle \tag{1.5}$$

$$\frac{\partial E}{\partial w_j} = 2 * (\boldsymbol{d} - \boldsymbol{W}^T * \boldsymbol{X}) * \boldsymbol{X} \tag{1.6}$$

Unde $j = \overline{1,n}$, $n$ fiind numărul de intrări în rețea.

Deoarece trebuie specificat pasul, η, pentru gradientul negativ, 2 poate fi introdus în rata de învățare η. Deoarece gradientul negativ presupune avansarea în direcția -∇E, forma finală a algoritmului devine:

$$w_i(t+1) = w_i(t) + \eta(\mathrm{d} - \boldsymbol{W}^T\boldsymbol{X})x_i \tag{1.7}$$

Unde $w_i(t+1)$ reprezintă ajustarea ponderii pentru următoarea iterație.

Această procedură de ajustare a ponderilor se mai numește și regula delta deoarece, în general, eroarea $\boldsymbol{W}^T * \boldsymbol{X} - d$ se notează cu δ.

În anul 1969, Marvin Minsky și Seymour Papert au publicat o analiză matematică precisă a perceptronului care demonstra că acesta nu este capabil să rezolve multe probleme importante, precum XOR, deoarece clasele nu sunt linear separabile, ceea ce a dus la o supresie în dezvoltarea rețelelor și la stagnarea cercetărilor timp de aproximativ 15 ani.

# 2. Perceptronul simplu

## 2.1. Arhitectura perceptronului

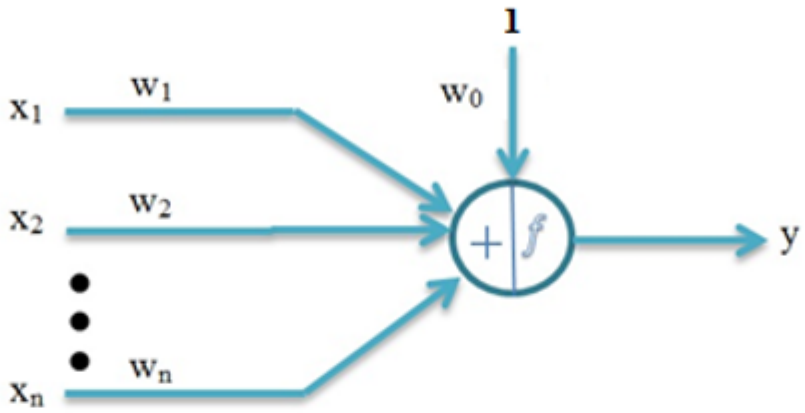


Fig. 2.1 – Arhitectura perceptronului simplu

În Fig. 2.1 este reprezentată arhitectura perceptronului simplu, unde $\boldsymbol{X} = (x_1, x_2, \dots, x_n)$ , $x_i \in \mathbb{R}$, reprezintă intrările în neuron, iar $\boldsymbol{W} = (w_0, w_1, \dots, w_n)$, $w_i \in \mathbb{R}$, reprezintă ponderile neuronului.

Ieșirea neuronului reprezentată în Fig. 2.1 cu litera y, este dată de ecuația:

$$y = f\left(w_0 + \sum w_i x_i\right) \tag{2.1}$$

Unde f(x) reprezintă funcția de activare a neuronului și este de forma:

$$f(k) = \begin{cases} 0, dacă\ \left(w_0 + \sum w_i x_i\right) < 0 \\ 1, dacă\ \left(w_0 + \sum w_i x_i\right) > 0 \end{cases} \tag{2.2}$$

Ieșirea neuronului mai poate fi scrisă și matriceal:

$$y = f(W^T X) \tag{2.3}$$

Unde:

$$X = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \qquad W = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}$$

## 2.2. Algoritmul perceptronului simplu

1. Inițializarea aleatoare a ponderilor la momentul t=0
2. Evaluarea ieșirii reale $y_j = f(W^T X_j)$, unde j=1:N, N – numărul de vectori de intrare
3. Compararea ieșirii reale $y_j$ cu ieșirea dorită $d_j$

$$e_j(t) = d_j - y_j(t) \tag{2.4}$$

În acest caz, există 3 valori posibile pentru eroarea de la ieșire: $\{0,1,-1\}$

4. Ajustarea ponderilor cu o valoare care micșorează eroarea:

$$\Delta w_i(t) = \sum_{j=1}^{N} \eta e_j(t) x_{ij} \tag{2.5}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \tag{2.6}$$

Unde $\eta$ reprezintărata de învățare.

5. Revenire la pasul 2 până când toate corespondentele $\{(X_j, d_j)\}$ sunt corecte

## 2.3. Teorema de convergență

Se poate demonstra că:

Dacă se aplică pentru două seturi de vectori liniari separabili, algoritmul perceptron converge către o soluție stabilă într-un număr finit de pași.

Suprafața de separație între clase este un hiperplan de ecuație:

$$w_0 + \sum w_i x_i = 0 \tag{2.7}$$

Două mulțimi de vectori se numesc liniar separabile dacă există cel puțin o suprafață liniară (de gradul I) care separă spațiul în două semispații în care clasele sunt disjuncte (nu se intersectează).

Una dintre problemele perceptonului este reprezentată de imposibilitatea găsirii unei soluții pentru probleme neseparabile liniar (XOR, par-impar). Mai mult, NU există o soluție de STOP în astfel de situații.

# 3. ADALINE (Adaptive Linear Neuron)

## 3.1. Arhitectura neuronului ADALINE

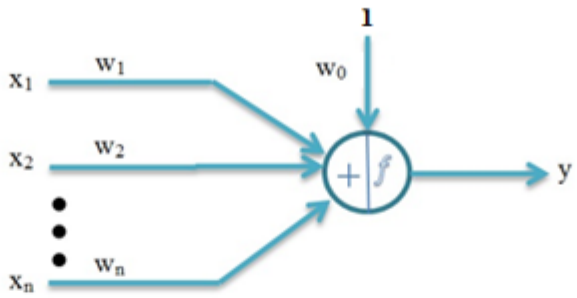


Fig. 3.1 – Arhitectura neuronului ADALINE

Structura în etapa de funcționare a neuronului ADALINE este identică cu cea a perceptronului. Diferența față de perceptron este modificarea funcției de activare în etapa de învățare:

$$f(x) = x \tag{3.1}$$

Noutatea ADALINE-ului a constat în dezvoltarea unei noi filozofii în antrenarea rețelei neuronale. S-a introdus o funcție cost care măsoară performanța funcționării neuronului. Funcția cost pentru ADALINE este eroarea pătratică între ieșirea dorită și cea reală.

## 3.2. Algoritm ADALINE

1. Inițializarea aleatoare a ponderilor la momentul t=0
2. Evaluarea ieșirii reale $y_j = f(W^T X_j)$, unde j=1:N, N – numărul de vectori de intrare
3. Calcularea erorii

$$E(W) = \frac{1}{N}\sum_{j=1}^{N} E_j(W) \tag{3.2}$$

$$E_j(W) = \frac{1}{2}\left(d_j - y_j\right)^2 \tag{3.3}$$

4. Ajustarea ponderilor cu o valoare care micșorează eroarea:

$$\Delta w_i(t) = \eta \frac{\partial E}{\partial w_i} \tag{3.4}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \tag{3.5}$$

Unde $\eta$ reprezintărata de învățare.

5. Revenire la pasul 2 până când eroarea medie pătratică pe setul de date scade sub o anumita valoare dorită.

# 4. MADALINE (Multiple ADALINE)

## 4.1. Arhitectura MADALINE

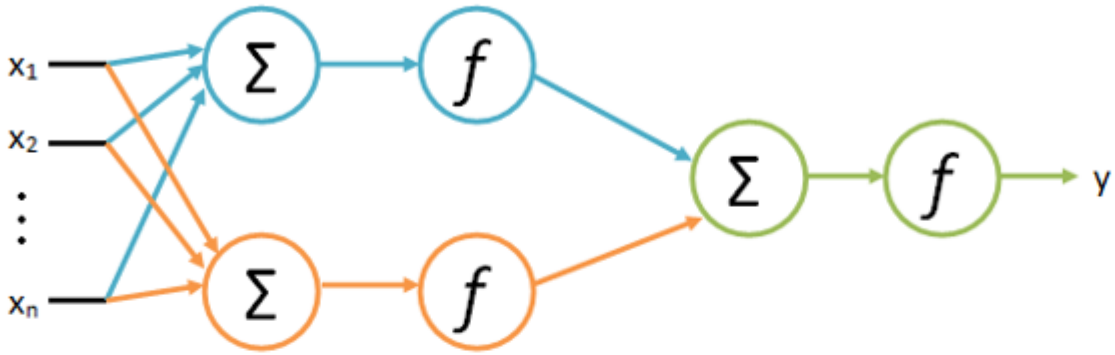


Fig. 4.1 – Arhitectura rețelei MADALINE

## 5. Probleme

1. Să se proiecteze o rețea pentru funcțiile NOR și NAND cu două intrări.
2. Să se proiecteze o rețea pentru funcțiile AND, OR, NOR și NAND cu trei intrări.
3. Se dau perechile de vectori de forma: $\{(X_i,d_i)\}$, unde $X_i$ reprezintă vectorul de intrare, iar $d_i$ reprezintă ieșirea dorită. Să se proiecteze o rețea care să învețe vectorii de intrare.

$$X = \left\{\left(\begin{bmatrix}0\\0\end{bmatrix},0\right),\left(\begin{bmatrix}0\\1\end{bmatrix},1\right),\left(\begin{bmatrix}1\\0\end{bmatrix},1\right),\left(\begin{bmatrix}2\\0\end{bmatrix},0\right),\left(\begin{bmatrix}1\\1\end{bmatrix},0\right),\left(\begin{bmatrix}3\\0\end{bmatrix},1\right),\left(\begin{bmatrix}2\\1\end{bmatrix},1\right)\right\}$$

4. Să se implementeze în Octave/MatLab algoritmul rețelei Perceptron pentru funcția logică AND cu două intrări.
5. Să se implementeze în Octave/MatLab algoritmul rețelei ADALINE utilizând datele prezente în fișierul RNSF_Lab3: „X.mat” și „y.mat”.