. 1 .
INTRODUCERE ÎN REȚELE NEURONALE. PREZENTAREA PACHETULUI NEURAL NETWORK TOOLBOX

Obiectivele lucrării

- însuşirea noţiunilor de bază din domeniul reţelelor neuronale artificiale;
- prezentarea pachetului Neural Network Toolbox al mediului MATLAB.

Noțiuni teoretice

O reţea neuronală (neurală) artificiale constă dintr-o mulţime de elemente de prelucrare, denumite neuroni sau unități de calcul, puternic interconectate în diferite configurații care permit propagarea și prelucrarea de informație numerică. Individual, neuronii realizează operații relativ simple, însă numărul unităților din rețea și arhitectura acesteia conduce la realizarea unui calcul mult mai complex.

Caracteristica principală a rețelelor neuronale este capacitatea de a învăța pe bază de exemple. Învățarea determină modificarea parametrilor tuturor unităților de prelucrarea din rețea în așa fel încât valorile acestora să cuprindă informația utilă din exemplele disponibile. Informaţia este memorată difuz și abstract în toată reţeaua, adică este "conținută" în valorile numerice ale parametrilor neuronilor. De cele mai multe ori, aceste valori nu reflectă valorile numerice din exemplele pe care se bazează învățarea, iar funcționarea rețelei este influențată de parametri într-o manieră neclară.

Descrierea completă a unei reţele neuronale curpinde descrierea modelului neuronal (modelul unităților funcționale din rețea), a arhitecturii reţelei şi a algoritmului de învățare folosit.

Modelul neuronului artificial

Neuronul artificial constituie elementul fundamental din structura unei reţele neurale. Acesta este o *unitate de calcul* ce realizează o operație relativ simplă, folosind semnalele de intrare, dar care contribuie la un calcul mult mai complex realizat de întreaga rețea.

Structura generală a unui neuron artificial este prezentată în Figura 1.1. Acesta are n *semnale de intrare*, pe care le vom nota $x_i \in \mathrm{P}$, cu $i = 1,...,n$, şi un *semnal de ieşire*, notat $y \in \mathrm{P}$. Semnalele de intrare sunt conectate la neuron prin intermediul *sinapselor*, caracterizate de factorii $w_i \in \mathrm{P}$, denumiți *ponderi sinaptice* (sau simplu *ponderi*; eng. *weights*). Parametrul notat $b \in \mathrm{P}$ se numește *deplasare* sau *prag* (eng. *bias*) și este asimilat ca pondere a unei intrări suplimentare având valoarea fixată 1.

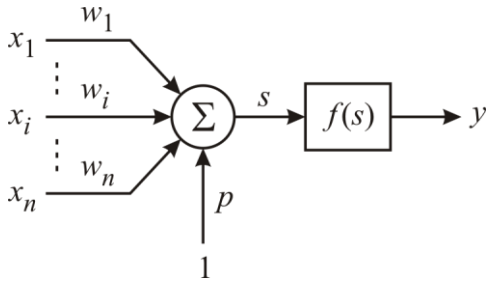


Figura 1.1. Modelul general al neuronului artificial.

Prin însumarea semnalelor de intrare ponderate, împreună cu valoarea pragului, se obţine *activarea totală* sau *starea internă* a neuronului, notată s, adică:

$$s = \sum_{i=1}^{n} w_i x_i + b \tag{1.1}$$

Termenul *parametrii neuronului* se referă la variabilele w_i și b în comun. Valorile lor sunt ajustabile pe parcursul antrenării neuronului, iar pe ajustarea adecvată a acestora se bazează capacitatea de învăţare a rețelei neuronale.

Cu valoarea activării totale, ieșirea este calculată prin *funcția de activare* (*funcția de transfer*, *funcţie de ieşire*, *funcţie neuronală*, eng. *transfer function*) a neuronului $f : \mathrm{P} \to \mathrm{P}$:

$$y = f(s) \tag{1.2}$$

Aceasta este o funcţie analitică, care poate fi liniară sau neliniară (vezi secțiunea următoare).

Relațiile (1.1) și (1.2) formează modelul neuronal și descriu calculul realizat de o unitate din rețeaua neuronală. Prima relație reprezintă *partea liniară* a neuronului, iar a doua definește *nodul* acestuia.

Deseori, în prezentarea algoritmilor de instruire, semnalele de intrare și ponderile lor sunt scrise în forma vectorială. Astfel, avem *vectorul intrărilor* $\mathbf{x} = [x_1 \quad ... \quad x_n]^{\mathrm{T}} \in \mathrm{P}^{n\times 1}$ și *vectorul ponderilor* $\mathbf{w} = [w_1 \quad ... \quad w_n] \in \mathrm{P}^{1\times n}$. Cu aceste notații, activarea totală a neuronului se scrie

$$s = \mathbf{w}\mathbf{x} + b \tag{1.3}$$

Observaţie. Intrarea suplimentară cu valoare fixă 1, ca semnal ponderat cu valoarea deplasării b, nu este o intrare efectivă, întrucât aceasta nu reprezintă o mărime concretă disponibilă în setul de date de antrenare. Pentru descrierea algoritmilor este mai convenabil să introducem convenția de notație $x_0 = 1$ și $w_0 = b$, ceea ce permite o scriere mai compactă. Astfel, vom scrie activarea totală și ieșirea neuronului:

$$s = \mathbf{w}\mathbf{x} = \sum_{i=0}^{n} w_i x_i \tag{1.4}$$

$$y = f(\mathbf{w}\mathbf{x}) \tag{1.5}$$

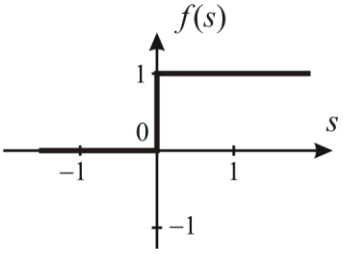
Observație. Pentru a putea efectua înmulțirea matriceală, vectorul ponderilor $\mathbf{w}$ este linie, iar vectorul intrărilor $\mathbf{x}$ este coloană.

Tipuri de funcții de activare

Funcțiile de activare uzuale sunt următoarele:

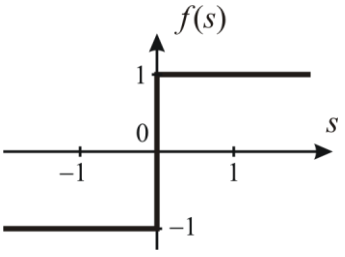
1. Funcția treaptă asimetrică (Heaviside)

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



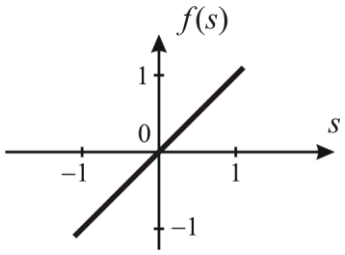
2. Funcția treaptă simetrică (signum)

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



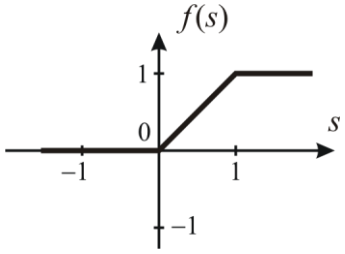
3. Funcția liniară (identitate)

$$f(s) = s$$



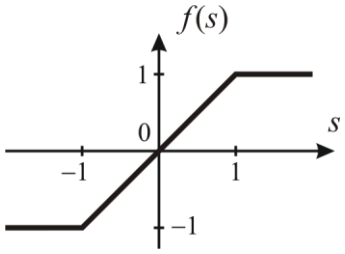
4. Funcția liniară cu saturație asimetrică

$$f(s) = \begin{cases} 0, s < 0 \\ s, 0 \le s \le 1 \\ 1, s > 1 \end{cases}$$



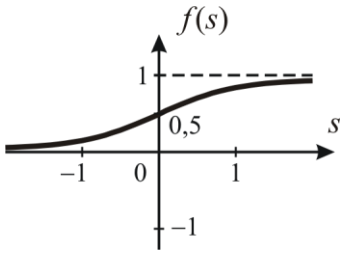
5. Funcția liniară cu saturație simetrică

$$f(s) = \begin{cases} -1, s < -1 \\ s, -1 \le s \le 1 \\ 1, s > 1 \end{cases}$$



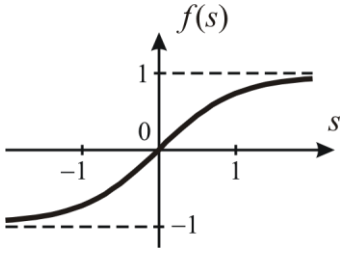
6. Funcția sigmoidală asimetrică

$$f(s) = \frac{1}{1+e^{-s}}$$



7. Funcția sigmoidală simetrică (tangentă hiperbolică)

$$f(s) = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}}$$



Arhitecturi de rețele neuronale

Modalitățile de interconectare a neuronilor artificiali într-o rețea sunt numeroase și variate, însă se pot identifica două clase:

a) *Reţele feed-forward*, unde semnalele se transmit prin rețea într-o singură direcție, de la intrare spre ieşire. În această structură, se definesc ușor straturi de neuroni, fiecare strat cuprinzând neuronii aflați la aceeași distanță de intrările rețelei. Nu există bucle și nici conexiuni între neuroni aflați pe același strat.

b) *Reţele feed-back* sau *rețele recurente,* în care semnalele se pot transmite în ambele direcţii, introducând conexiuni de reacţie în arhitectura reţelei. Aceste tipuri de rețele sunt foarte puternice şi pot fi extrem de complicate.

Există câteva structuri de reţele neuronale artificiale consacrate şi foarte des utilizate, deoarece au oferit rezultate încurajatoare în diverse aplicaţii:

- rețele feed-forward: perceptronul, rețeaua AdaLiNe, memorii asociative liniare;
- rețele recurente: maşina Boltzmann, learning vector quantization, reţele Hopfield, memorii asociative bidirecţionale, reţele Kohonen cu auto-organizare.

În multe aplicații, rețelele prezintă mai multe unități funcționale la ieșire (semnale de ieșire). Din acest motiv, introducem aici o notație suplimentară, necesară pentru o rețea neuronală cu m semnale de ieșire și anume vectorul mărimilor de ieșire $\mathbf{y} \in \mathrm{P}^{m\times 1}$.

Instruirea rețelelor neuronale

Caracteristica principală a rețelelor neuronale este capacitatea de a învăța sau de a extrage informație din date de antrenare, care pot fi incomplete, parțial eronate, afectate de perturbații etc. Cunoaşterea pe care o dobândeşte reţeaua este *memorată* în ponderile sinaptice ce caracterizează intrările rețelei și conexiunile dintre neuroni.

Datele de antrenare se prezintă reţelei în mod secvenţial și repetitiv, iar ponderile reţelei sunt ajustate în mod iterativ, conform unei proceduri prestabilite numită *algoritm de instruire* (*antrenare, învățare*). Algoritmul de instruire determină convergenţa ponderilor spre valori care să producă ieşirea dorită pentru majoritatea datelor de antrenare. O rețea neuronală este antrenată dacă funcționarea ei în faza de lucru, descrisă prin valorile calculate ale semnalelor de ieșire, este în limitele unor erori admise.

Există două tipuri fundamentale de învăţare utilizate în reţelele neuronale: *învăţarea supervizată* şi *învăţarea nesupervizată*.

În cazul instruirii supervizate, datele de antrenare conțin valori ale semnalelor de intrare și valorile corespunzătoare (impuse, corecte) ale semnalelor de ieșire. Aceste perechi formează mulțimea datelor de antrenare,

$$\mathscr{P} = \{(\mathbf{x}, \mathbf{d})^k\}\Big|_{k=\overline{1,P}} \tag{1.6}$$

în care $\mathbf{x} \in \mathrm{P}^{n\times 1}$ este vectorul celor n datelor de intrare, $\mathbf{d} \in \mathrm{P}^{m\times 1}$ este vectorul ce conține valorile corespunzătoare (impuse, considerate corecte) ale celor m semnale de ieșire ale rețelei, iar P reprezintă numărul de perechi de date.

În faza de antrenare, valorile mărimilor de ieșire ale rețelei, $\mathbf{y} \in \mathrm{P}^{m\times 1}$, obținute pe baza valorilor mărimilor de intrare, $\mathbf{x} \in \mathrm{P}^{n\times 1}$, diferă de valorile cele impuse, $\mathbf{d} \in \mathrm{P}^{m\times 1}$, diferența fiind o *eroare de antrenare*. Pentru un exemplu de antrenare, eroarea este

$$e = \sum_{j=1}^{m} (d_j - y_j) \tag{1.7}$$

iar pentru întregul set de date de antrenare

$$E = \frac{1}{2}\sum_{k=1}^{P} (e^k)^2 = \frac{1}{2}\sum_{k=1}^{P} (d_j^k - y_j^k)^2 \tag{1.8}$$

Această mărime de eroare depinde de vectorul tuturor ponderilor sinaptice, notat $\mathbf{w}$, și este o funcție criteriu pe care algoritmii de antrenare caută să o minimizeze. Așadar, algoritmii de antrenare supervizată sunt bazați pe metode de căutare a punctului de minim al unei funcții criteriu care conține eroarea de antrenare totală, calculată pentru toate exemplele de antrenare.

În instruirea nesupervizată datele de antrenare nu conțin valori impuse ale mărimilor de ieşire, iar învăţarea se desfăşoară fără o cunoaştere specifică a ceea ce ar putea să fie un răspuns corect. Datele de antrenare sunt în forma

$$\mathscr{P} = \{(\mathbf{x})^k\}\Big|_{k=\overline{1,P}} \tag{1.9}$$

În decursul instruirii, prin ajustarea ponderilor, se realizează o grupare a datelor de antrenare în clase. Răspunsul pe care îl va produce o clasă nu poate fi determinat înainte de încheierea procesului de instruire. Prin urmare, ieşirile unei reţele instruite cu o metodă nesupervizată trebuie, în general, să fie interpretate.

Desfășurarea lucrării. Exemple

Implementarea în MATLAB a rețelelor neuronale. Neural Network Toolbox

Pachetul *Neural Network Toolbox* (NN) conține un ansamblu de funcții care implementează algoritmi specifici domeniului sau care realizează operații utile în aplicații specifice. Sunt disponibile funcții pentru inițializarea rețelelor, antrenarea lor, simularea funcționării,

reprezentarea grafică a rezultatelor etc. Funcțiile sunt flexibile permițând utilizatorului să specifice arhitectura rețelei, tipul funcției de activare, modul de conectare între unități sau să aleagă între mai multe variante de algoritmi de învățare.

Lista funcțiilor cuprinse în pachet se obține cu comanda:

```
help nnet
```

Sunt incluse, de asemenea, o serie de aplicații demonstrative, accesibile prin comanda:

```
nnd
```

care prezintă principalele categorii de rețele ce pot simulate. Între aplicațiile demonstrative avem: identificarea semnalelor folosind modele liniare (applin1), predicții în serii temporale (applin2), recunoașterea șabloanelor temporale (appelm1), recunoașterea caracterelor (appcr1). Alte exemple sunt prezentate prin interfețe grafice: recunoașterea șabloanelor (nprtool), aproximarea datelor (nftool), clasificarea nesupervizată (nctool).

Funcțiile de activare

Funcțiile de activare uzuale în MATLAB sunt:

- Funcția treaptă asimetrică: **hardlim**
- Funcția treaptă simetrică: **hardlims**
- Funcția liniară: **purelin**
- Funcția liniară cu saturație asimetrică: **satlin**
- Funcția liniară cu saturație simetrică: **satlins**
- Funcția sigmoidală asimetrică: **logsig**
- Funcția sigmoidală simetrică: **tansig**

Structura de date a unei rețele neuronale

Indiferent de arhitectura ei, o rețea neuronală implementată în NN toolbox este o structură de date neomogenă ce conţine toate informaţiile privind reţeaua. Componentele ei sunt matricile parametrilor neuronali pentru toate unitățile din rețea, parametrii privind variantele de implementare ale proceselor de simulare și de antrenare, parametrii de control etc. Printre câmpurile acestei structuri se află vectorii de celule [1] ce conţin: matricea ponderilor, vectorul pragurilor, vectorul semnalelor de ieşire.

Pentru exemplificare, să considerăm o rețea formată dintr-o singură unitate de calcul, având două mărimi de intrare, inițializată prin comanda

```
net = newp([0 1;0 1], 1);
```

(funcția **newp** este prezentată în lucrarea următoare). Componentele structurii pot fi accesate în fereastra "Workspace". În "Command window" sau în programe, vectorul ponderilor semnalelor de intrare este accesibil prin:

```
net.IW{1}
```

iar deplasarea

```
net.b{1}
```

Parametrii de control se stabilesc prin atribuirea unor valori. De exemplu, pentru antrenarea rețelei, se poate defini eroarea maxim admisă a erorii de antrenare

```
net.trainParam.goal = 0.1;
```

(care trebuie să fie o valoare foarte mică în raport cu valorile din setul de date de antrenare) sau numarul maxim de parcurgeri ale setului de date de antrenare

```
net.adaptParam.passes = 10;
```

Antrenarea rețelelor neuronale în MATLAB

În MATLAB sunt disponibile două funcții pentru antrenarea unei rețele neuronale anterior inițializată: **adapt** și **train**. Funcţia **adapt** [2] realizează o antrenare serială (incrementale, eng. *sequential learning*) în sensul că ajustarea ponderilor rețelei se efectuează după prezentarea fiecărui exemplu din setul de antrenare. Funcția **train** [3] realizează o antrenare pe lot (eng. *batch learning*), adică ajustarea ponderilor rețelei se face după o parcurgere a setului de date.

Sintaxa minimală a celor două funcții este similară:

```
retea_antrenata = adapt(retea, date_de_intrare, date_de_iesire)
retea_antrenata = train(retea, date_de_intrare, date_de_iesire)
```

unde:

- "retea" este rețeaua neuronală obținută în urma inițializării,
- "date_de_intrare" este o matrice conținând valorile semnalelor de intrare din setul de date de antrenare,
- "date_de_iesire" este o matrice conținând valorile impuse semnalelor de ieșire corespunzătoare.

Evident, ambele funcții implementează algoritmi de instruire supervizată.

Simularea unei rețele

După antrenare, rețeaua neuronală este testată folosind fie setul de date de antrenare sau o parte a acestuia, fie un set de date de verificare distrinct. Simularea rețelei se face cu funcția **sim**, a cărei sintaxă este:

```
iesiri = sim(retea_antrenata, intrari)
```

unde:

- "intrari" este o matrice conținând valorile semnalelor de intrare din setul de date de testare,
- "iesiri" este o matrice conținând valorile mărimilor de ieșire obținute de rețeaua antrenată.

Dacă rețeaua este corect antrenată, atunci valorile semnalelor de ieșire obținute sunt egale cu cele impuse (sau sunt acceptabil de apropiate).

Observație. În MATLAB sunt definite mai multe funcții cu numele **sim** în diferite pachete, identificabile prin tipul argumentelor transmise. Pentru a afla detalii privind funcția care simulează funcționarea unei rețele neuronale, se scrie comanda

```
help network/sim
```

Exemple

- **Exemplul 1.1.**

  Reprezentarea grafică a funcțiilor de activare uzuale.

- *Rezolvare*

  Programul următor prezintă grafic toate cele 7 funcții de activare uzuale enumerate în tabelul 1.1.

```
% Domeniul de definitie pentru care se reprezinta grafic functiile
x = -5:0.01:5;

% Calculam valorile functiilor de activare pe domeniul stabilit
y1 = hardlim(x);
y2 = hardlims(x);
y3 = purelin(x);
y4 = satlin(x);
```

```
y5 = satlins(x);
y6 = logsig(x);
y7 = tansig(x);

% Afisam fiecare functie intr-un "subgrafic"
grid on
subplot(2,4,1); plot(x,y1); title('Treapta asimetrica (hardlim)');
subplot(2,4,2); plot(x,y2); title('Treapta simetrica (hardlims)');
subplot(2,4,3); plot(x,y3); title('Liniara (purelin)');
subplot(2,4,4); plot(x,y4); title('Liniara cu saturatie asimetrica (satlin)');
subplot(2,4,5); plot(x,y5); title('Liniara cu saturatie simetrica (satlins)');
subplot(2,4,6); plot(x,y6); title('Sigmoidala asimetrica (logsig)');
subplot(2,4,7); plot(x,y7); title('Sigmoidala simetrica (tansig)');
```

Exerciții propuse

- **Problema 1.1**

  Să se scrie câte o funcție MATLAB sau C/C++ pentru implementarea funcțiilor de activare uzuale.

- **Problema 1.2**

  Studiați influența pragului de activare (deplasării) asupra ieșirii produse de funcția de activare.

  Indicație: Se vor reprezenta pe același grafic funcțiile $f(s)$, $f(s+b_1)$, $f(s+b_2)$, ..., unde f este o funcție de activare uzuală, iar b_1, b_2, ... sunt diverse valori ale pragului de activare.

- **Problema 1.3**

  Să se scrie un program MATLAB sau C/C++ pentru implementarea modelului neuronal (descris în figura 1.1 și relațiile 1.1 - 1.2) și ilustrarea calculelor efectuate de acesta. Realizați mai multe variante folosind funcții de activare diferite.

  Indicație: Se va stabili o structura de date pentru parametrii neuronului și se va scrie o funcție pentru simularea funcționării acestuia. Funcția va primi ca argument vectorul semnalelor de intrare și va returna valoarea mărimii de ieșire.

- **Problema 1.4**

  Studiați programele demonstrative `nnd2n1` și `nnd2n2` cuprinse în pachetul Neural Network.

Referințe

[1] ***, https://www.mathworks.com/help/matlab/cell-arrays.html

[2] ***, https://www.mathworks.com/help/nnet/ref/adapt.html

[3] ***, https://www.mathworks.com/help/nnet/ref/train.html