Arhitectura Multistrat a Rețelelor Neurale. Propagarea înapoi a erorii

Seminarul Republican Logica și
Inteligența Artificială

Bumbu Tudor, Ștefan Stratulat

- ARHITECTURĂ
  - Felul în care este construit sau alcătuit ceva; aspectul compozițional al unei opere artistice; **structură**. – Din *fr. architecture, lat. architectura*.

Arhitectura Multistrat a RN

- Neuronii pot fi conectați în diferite moduri pentru a forma o rețea neurală

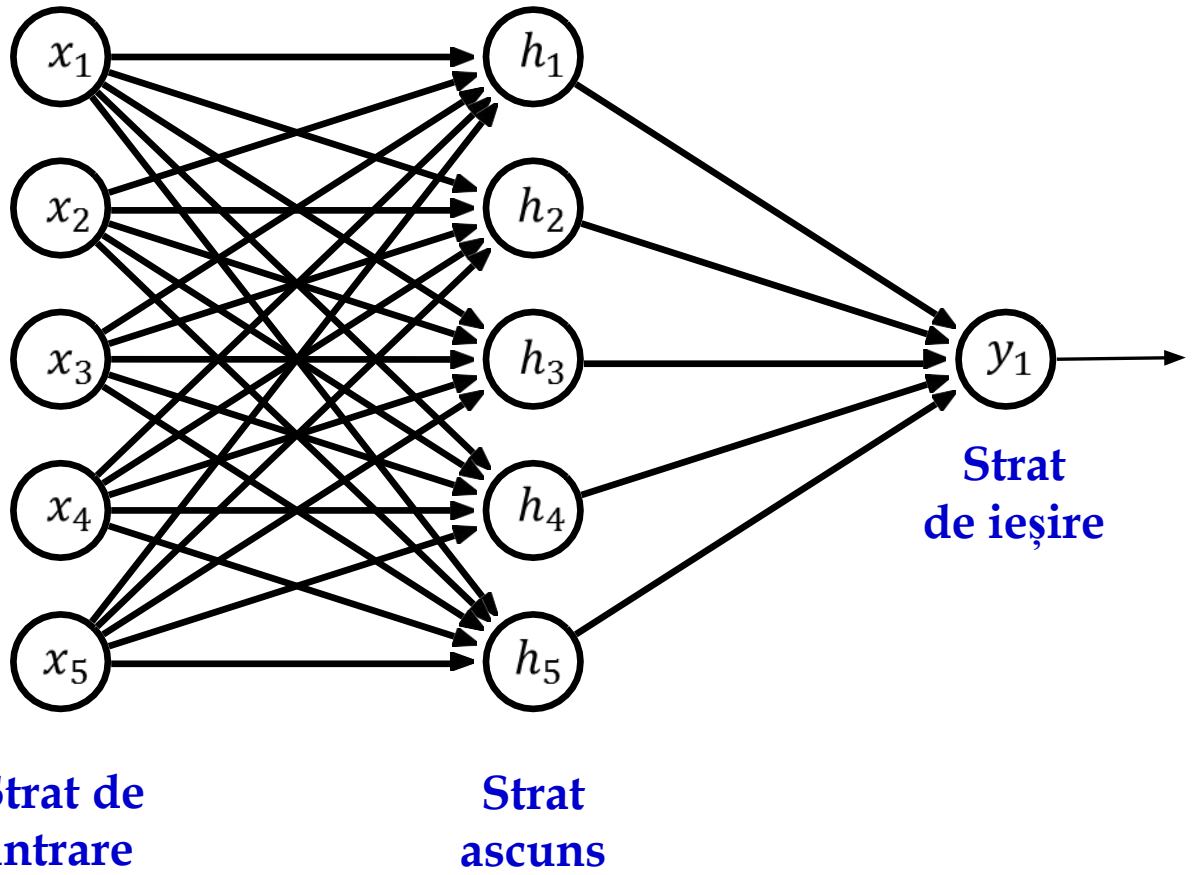
•O *rețea neurală multistrat* conține:

  - **două** sau **mai multe** straturi de neuroni
    - Primul strat primește intrările din mediu
    - Ieșirile din primul strat sunt intrări pentru stratul următor
    - Ieșirea rețelei este formată din ieșirile neuronilor din ultimul strat

- Straturile situate între primul și ultimul strat se numesc *straturi ascunse ale rețelei*

O rețea neurală cu un strat ascuns

- Fiecare săgeată este o pondere



3 concepte importante

- Transformare
- Instruire
- Asociere

Transformare

- Rețelele neuronale transformă o anumită configurație de intrare într-o configurație de ieșire

- Ieșirile rețelei pot corespunde unei acțiuni pe care rețeaua trebuie să o realizeze
  - Ieșirile rețelei pot prezenta o clasificare a obiectelor prezentate la intrare

- În general, ieșirea poate fi orice transformare a spațiului de intrare, deci funcționarea rețelei reprezintă o ***aplicație*** (proiecție, transformare) a spațiului vectorilor de intrare în spațiul vectorilor de ieșire

Instruire

- Prin *instruire* sau *învățare* înțelegem o metodă prin care este construită o astfel de aplicație

- În mod obișnuit instruirea presupune că se furnizează rețelei suficient de multe exemple de ***perechi intrare – ieșire***

- Folosind aceste exemple, rețeaua este forțată să se adapteze în sensul generalizării unei aplicații intrare – ieșire
  - Această aplicație îi va permite rețelei să trateze corect și perechi intrare – ieșire care nu i-au mai fost prezentate niciodată

Asociere

Prin *formă* înțelegem un vector

$$\mathbf{R} = (\mathbf{x_1}\ \mathbf{x_2}\ \ldots \mathbf{x_m})^{\mathbf{T}}$$

ce are componente reale.

În unele situații se poate impune componentelor acestui vector condiția

$$x_i \in [-1, 1].$$

O *asociere* este o pereche ordonată (**R, S**), unde **R** este forma – cheie și

$$\mathbf{S} = (\boldsymbol{s_1}\ \boldsymbol{s_2}\ \ldots \boldsymbol{s_n})$$

este *forma asociată* vectorului de intrare.

$(\mathbf{R}, \mathbf{S})$ - asociere

$\mathbf{R} = (\mathbf{x_1}\ \mathbf{x_2}\ \ldots \mathbf{x_m})^{\mathbf{T}}$

$\mathbf{S} = (s_1\ s_2\ \ldots s_n)$

Forma cheie		Forma asociată
x1	x2	S
1	0	1
0	0	0
0	1	1
1	1	1

Liste de asociere

Numim *listă de asociere* de lungime $\boldsymbol{p}$ sau *mulțime de instruire* o mulțime LA de $\boldsymbol{p}$ asocieri,

$$\mathbf{LA} = \left\{\left(\boldsymbol{R^1}, \boldsymbol{S^1}\right), \ldots, \left(\boldsymbol{R^p}, \boldsymbol{S^p}\right)\right\}, \qquad \boldsymbol{p} \geq \mathbf{1},$$

unde toate formele – cheie cu aceeași dimensiune, adică

$$\dim R^j = m\,, \qquad j = 1, \ldots, p.$$

Se admite că și formele asociate au, toate, aceeași dimensiune, adică

$$\dim S^j = n\,, \qquad j = 1, 2, \ldots, p.$$

$$\mathbf{LA} = \{(R^1, S^1), \ldots, (R^p, S^p)\}$$

LA

x1	x2	S
1	0	1
0	0	0
0	1	1
0	0	0
1	1	1
0	1	1
0	1	1

- În *Rețelele Neurale* asocierile se memorează în ponderile sinapselor neuronilor

- Reprezentarea unei asociații este distribuită peste mai multe conexiuni

- Reciproc, fiecare conexiune este implicată în memorarea mai multor asocieri

- Această reprezentare distribuită înzestrează sistemul cu un anumit grad de robustețe față de deteriorări fizice minore,
  - În plus, permite sistemul să „observe” (să detecteze) anumite regularități în mulțimea de instruire

- Faza în care toate asocierile sunt memorate sau *învățate* reprezintă *faza de instruire* a rețelei

- *Faza de lucru* - este caracterizată printr-un proces de regăsire
  - În această fază se prezintă rețelei diferite forme-cheie

Faza de lucru și de instruire

- Semnalul se propagă în rețea dinspre stratul de intrare spre cel de ieșire
  - În **faza de lucru** avem o ***propagare înainte a semnalului***
  - În **faza de instruire** avem o ***propagare înapoi a erorii***
  - În continuare vom prezenta algoritmul de propagare înapoi

Algoritmul de propagare înapoi (I)

- Algoritmul de propagare înapoi – retro-propagare (**Back Propagation**, BP):
  - cel mai important şi mai utilizat algoritm pentru instruirea rețelelor neuronale multistrat
- Popularitatea sa a constituit un factor important în reînvierea direcției conexioniste, în Inteligența Artificială

Algoritmul de propagare înapoi (II)

- **Algoritmul de propagare înapoi** este o metodă de instruire în rețelele neuronale multistrat cu transmitere înainte (rețele unidirecționale),
  - în care se urmăreşte minimizarea erorii medii pătratice printr-o metodă de gradient.

Caracteristici

- BP acționează asupra unei arhitecturi cu mai mulți neuroni organizați în straturi multiple, dintre care unele straturi sunt ascunse iar neuronii au funcții de ieşire sigmoidale.

Scurt istoric

- Ideea propagării înapoi a fost redescoperită de mai multe ori
- **1974**: Paul Werbos
- **1982:** David Parker și Rumelhart („Parallel Distributed Processing")
  - autorii au sugerat că algoritmul de propagare înapoi sau regula delta generalizată (cum au numit-o ei) poate depăşi limitările algoritmului perceptronului, enumerate de Minsky şi Papert (1969).

Primele practici

- În cadrul unui experiment pentru a sintetiza vocea dintr-un text scris, fără a face apel la reguli ale unui sistem expert (Sejnowski şi Grossberg, 1987)
- O bandă magnetică conținea o bogată experiență de instruire
  - Răsunetul pe care acest experiment l-a avut, a determinat popularizarea sa rapidă ca „un nou algoritm de instruire care învață din experiență”.

Aplicații

- S-a apreciat că algoritmul de propagare înapoi ar putea avea numeroase aplicații potențiale:
  - recunoaşterea în timp real a vorbirii,
  - traducerea automată,
  - vedere artificială

Analogie

- Psihologii şi neurofiziologii au căutat să identifice algoritmul de propagare înapoi în funcționarea creierului uman şi a sistemului nervos central
- Cel puțin până acum, nu s-a descoperit analogia biologică a algoritmului de retro-propagare

Algoritmul de propagare înapoi pentru RN cu un singur strat ascuns

P0. Inițializare

- Se inițializează ponderile rețelei cu valori mici, generate aleator.
- Se inițializează constanta de instruire $\boldsymbol{c}$ ($\boldsymbol{c} \in (\mathbf{0}, \mathbf{1})$).

P1. Un ciclu de instruire

- Se prezintă rețelei o mulțime de exemple de instruire $\boldsymbol{k}$.
  - Pentru fiecare exemplu de instruire se execută secvența **P2 ... P12**.

P2. Se prezintă rețelei vectorul $\boldsymbol{x^k} = (x_1^k, x_2^k, x_3^k, \dots, x_n^k)^T$ și vectorul ieșirilor dorite $\boldsymbol{d^k}$.

P3. Se calculează valorile de activare ale neuronilor din stratul ascuns:

$$h_q^k = \sum_{i=1}^{n} x_i^k \mathrm{v}_{iq}^h(k)\,,$$

unde $\boldsymbol{q}$ este neuronul și $\mathbf{v}^{\boldsymbol{h}}$ – mulțimea de ponderi dintre stratul de intrare și stratul ascuns.

P4. Se calculează ieșirile neuronilor din stratul ascuns**:**

$$i_q^k = S_q^h(h_q^k),$$

unde S_q^h $- functia\ de\ raspuns\ a\ neuronului\ q.$

($S(x) = \frac{1}{1+e^{-x}}$ - funcția logistică)

P5. Se calculează valorile de activare ale neuronilor din stratul de ieșire:

$$y_j^k = \sum_q i_q^k \mathrm{v}_{iq}^o\ (k),$$

unde $\mathbf{v}^o$ este mulțimea ponderilor dintre stratul ascuns și stratul de ieșire a rețelei.

P6. Se calculează ieșirile rețelei utilizând formula**:**

$$o_j^k = S_j^o(y_j^k),$$

unde $\boldsymbol{S_j^o}$ $- functia\ de\ raspuns\ a\ neuronului\ \boldsymbol{q}.$

($S(x) = \frac{1}{1+e^{-x}}$ - funcția logistică)

P7. Se calculează semnalul de eroare pentru neuronii din stratul de ieșire:

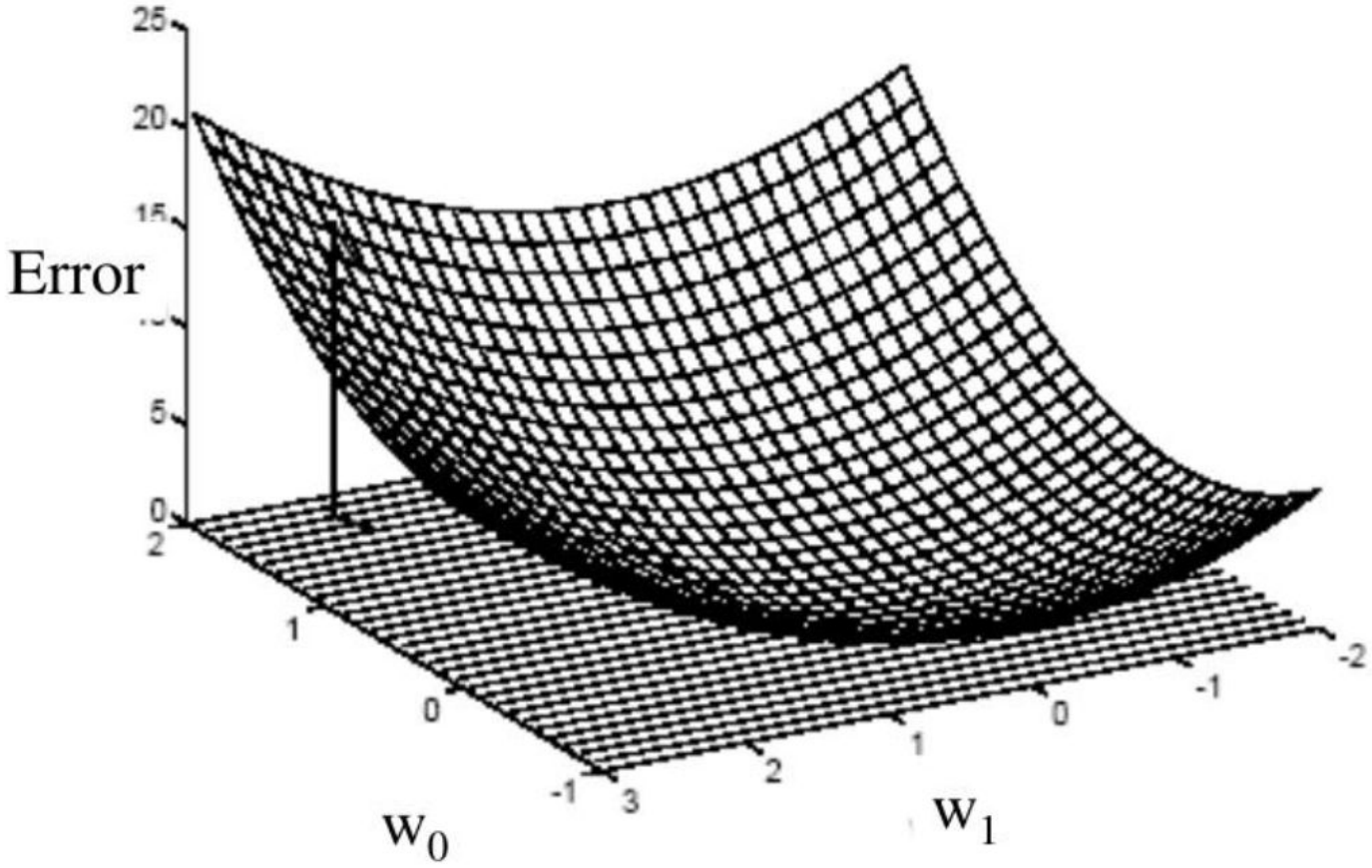
$$\delta_{kj} = (d_j^k - o_j^k) \frac{\partial S_j^o(y_j^k)}{\partial y_j^k} \quad .$$

P8. Se calculează semnalul de eroare pentru neuronii din stratul ascuns**:**

$$\delta_{kq}^{h} = \frac{\partial S_q^h(h_q^k)}{\partial h_q^k} \sum_j \delta_{kj} \mathrm{v}_{qj}^{o}(k) \quad .$$

(Acest semnal se calculează *înainte* de actualizarea ponderilor din stratul de ieșire).

Coborârea gradientului (gradient descent) în spațiul ponderilor



Din cartea **Machine Learning,** de Tom Mitchel.

http://profsite.um.ac.ir/~monsefi/machine-learning/pdf/Machine-Learning-Tom-Mitchell.pdf

P9. Se actualizează ponderile neuronilor din stratul de ieșire:

$$\mathrm{v}_{qj}^{o}(k+1) = \mathrm{v}_{qj}^{o}(k) \; + c \; \delta_{kj} \; S_q^h\left(h_q^k\right),$$

unde $\boldsymbol{q}$ este un neuron al stratului ascuns și $\boldsymbol{j}$ este un neuron de ieșire

P10. Se actualizează ponderile neuronilor din stratul ascuns**:**

$$\mathrm{v}_{iq}^{h}(k+1) = \mathrm{v}_{iq}^{h}(k) \; + c \; \delta_{kq}^{h} \; x_i^k,$$

unde $\boldsymbol{i}$ este un neuron al stratului de intrare.

P11. Se calculează eroarea:

$$E_k = \frac{1}{2}\sum_{j=1}^{p}(d_j^k - o_j^k)^2 ,$$

unde $\boldsymbol{p}$ este numărul numărul de exemple de instruire.

Eroarea este o măsură a calității învățării.

P12. Iterarea

– Dacă eroarea este acceptabilă pentru toate perechile $(\boldsymbol{x^k}, \boldsymbol{d^k})$ din mulțimea de instruire, atunci învățarea se poate considera încheiată.

– În caz contrar, vectorii de instruire sunt prezentați din nou rețelei în cadrul unui nou ciclu.

De ce avem nevoie de mai multe straturi?

- Incapacitatea arhitecturilor simple de a rezolva o problemă sau o clasă de probleme
- Uneori este sufiecient să mărim numărul de neuroni din rețea
  - **fără de a schimba arhitectura**
- În alte situații este necesară modificarea arhitecturii rețelei
  - **introducând unul sau mai multe straturi neurale noi**

Next

Tema: **Modele Secvențiale**