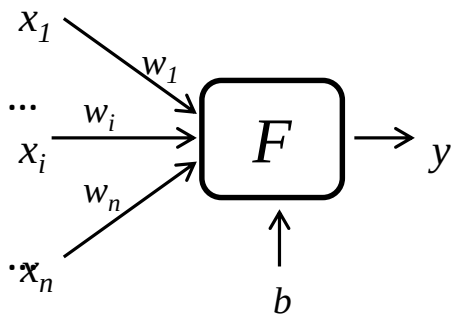
PROCESAREA CUNOȘTINȚELOR ȘI CALCUL INTELIGENT - PCCI

Reţele neuronale artificiale pentru învăţare supravegheată. Perceptronul multistrat.

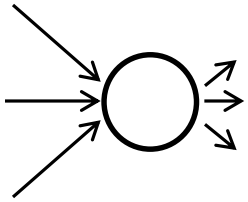
Cuprins:

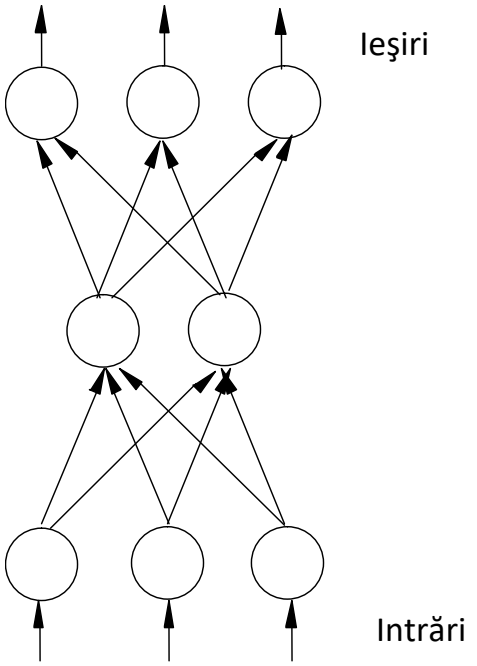
Rețele neuronale artificiale

- Mai mulţi neuroni interconectaţi formează o reţea neuronală artificială (RNA).



- un singur neuron

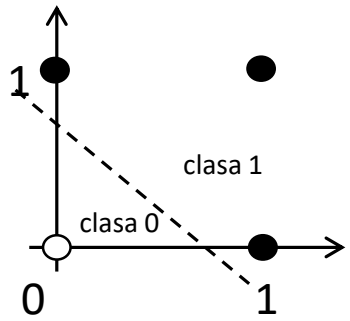




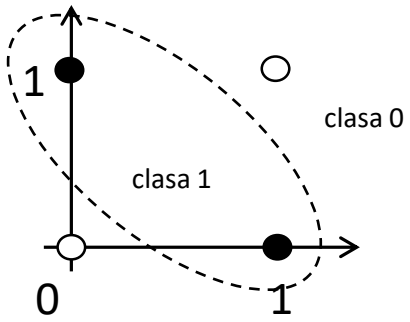
- O reţea neuronală

Necesitatea formării RNA

- Un singur neuron poate învăţa o problemă liniar separabilă, dar eşuează în cazul funcţiei simple XOR:

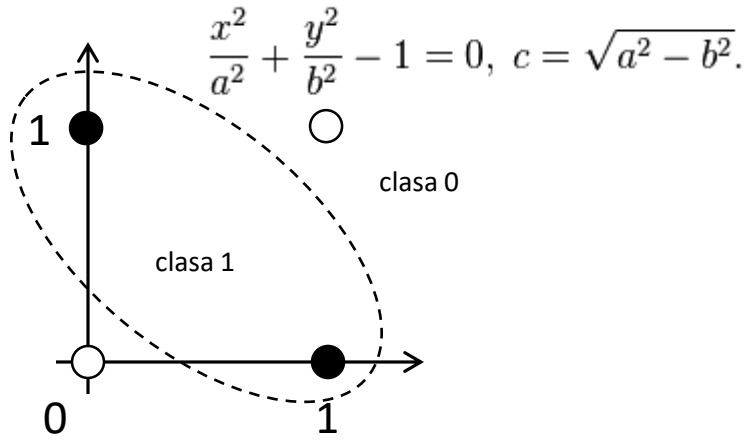


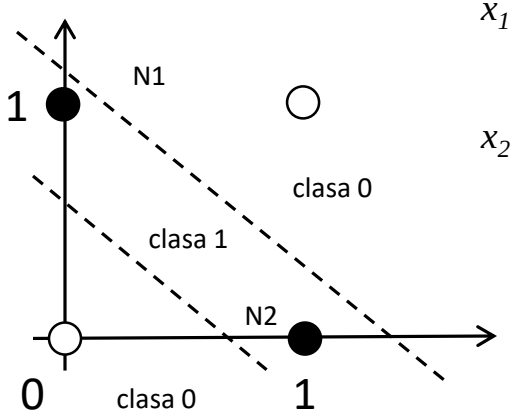
SAU, OR – funcţie liniar separabilă

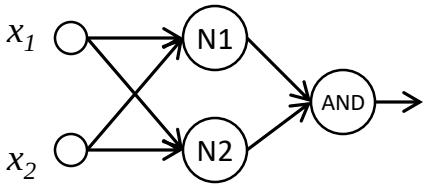


SAUE, XOR – funcţia nu este liniar separabilă

- Pentru învăţarea funcţiei XOR, e nevoie de trei straturi neuronale:

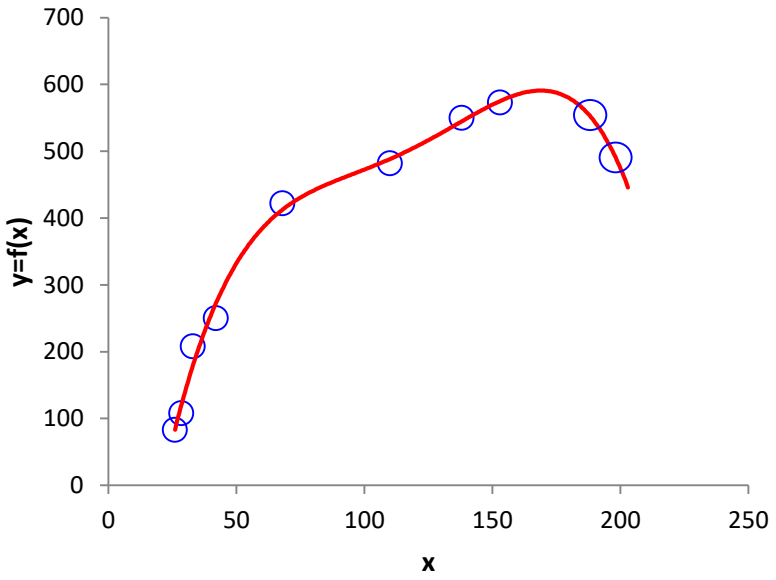






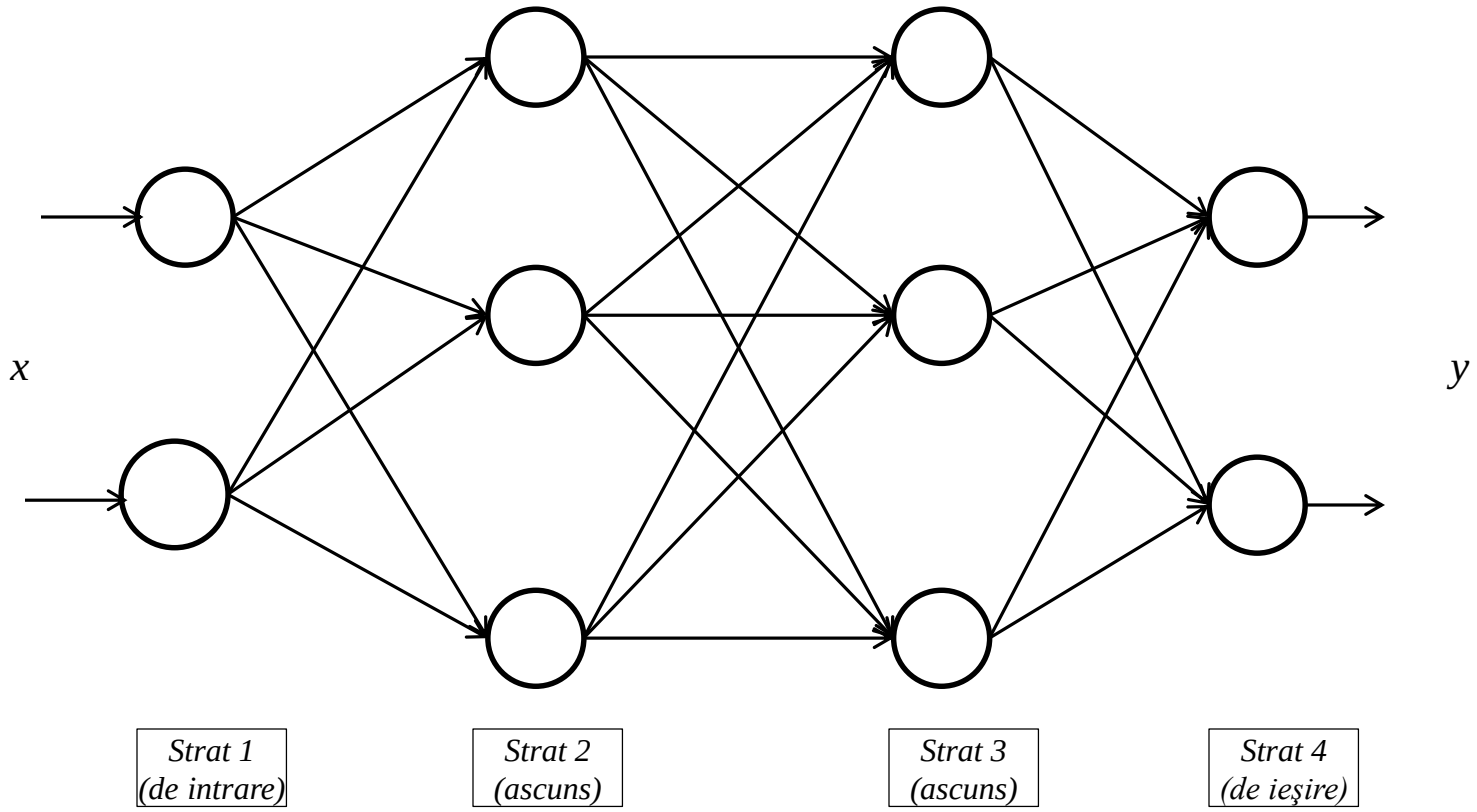
Necesitatea formării RNA pentru regresie

- O combinaţie de neuroni (unul sau mai multe straturi ascunse) poate modela funcţii complexe, de grad superior, care modelează mai bine un set de date caracterizat de o funcţie de variaţie $y=f(x)$ complexă.



$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3 + a_4 \cdot x^4$$

Necesitatea formării RNA pentru regresie

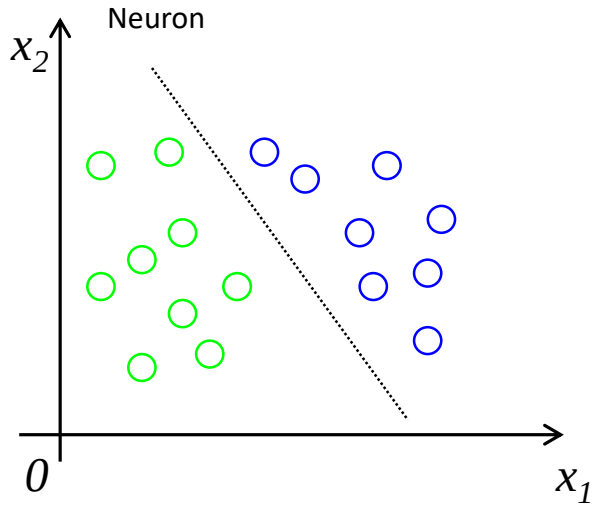


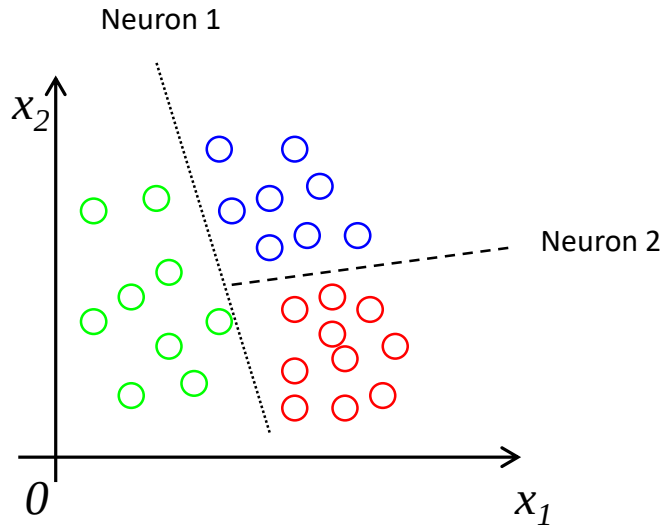
x	Distribuţie	Determinare fragmente f(x)	xompunere f(x)	$y=f(x)$

- RNA îşi alege singură felul în care reprezintă intern pe $f(x)$, prin mecanismul ajustării ponderilor şi pragurilor.
- Pentru două seturi de antrenare diferite, aceeaşi funcţie originală x-$f(x)$ va fi reprezentată diferit, cu alte ponderi şi praguri.

Necesitatea formării RNA pentru clasificare

- Problemele de clasificare de tip 0/1 pot fi rezolvate cu un neuron de tip sigmoid logistic, care determină o frontieră.
- Problemele de clasificare cu mai multe clase trebuie să determine mai multe frontiere, deci folosesc mai mulţi neuroni.

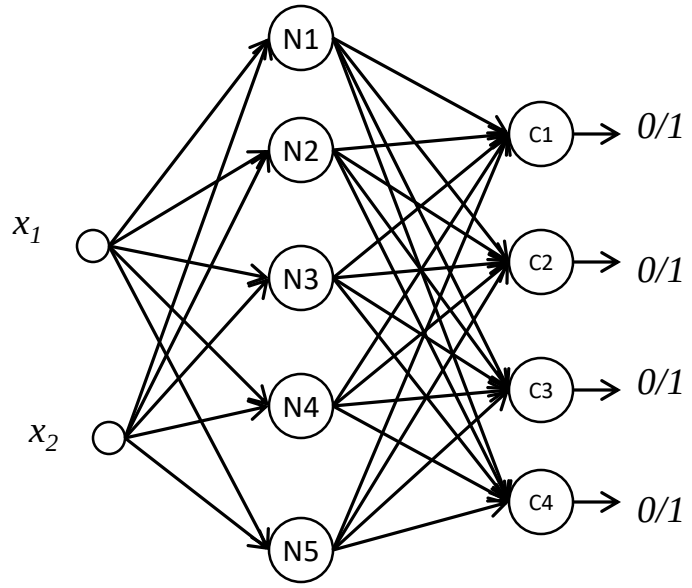


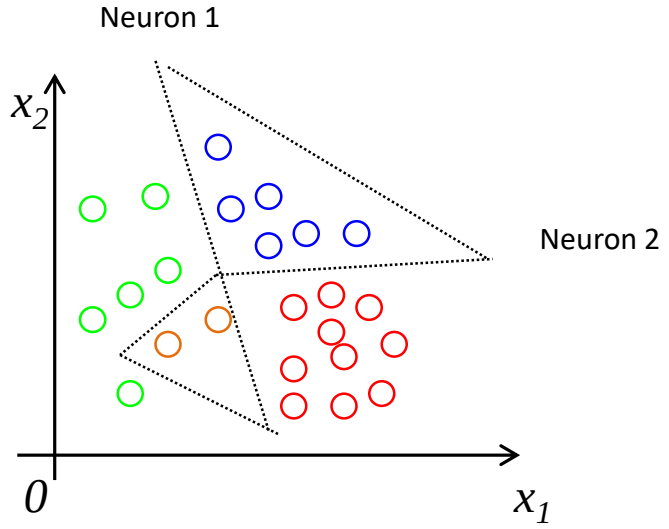


Justificarea matematică

- O RNA cu minim 3 straturi poate separa orice spaţiu, dacă are suficienţi neuroni pe stratul ascuns (teorema lui Kolmogorov [MacLeod]).

- Reţeaua "regionalizează" spaţiul variabilelor de intrare.
- Exemplu: o RNA care identifică 4 clase folosind 5 neuroni ascunşi. Un model de intrare poate să aparţină unei singure clase (valoare ieşire neuron aproximativ 1). Restul neuronilor vor calcula la ieşire valoarea aproximativ 0.





Utilizarea RNA

- Definirea problemei şi alegerea tipului de reţea corespunzător.
  - Arhitectura RNA pentru învăţare supravegheată diferă substanţial de arhitectura RNA pentru învăţare nesupravegheată
- Pregătirea datelor pentru învăţare/antrenare.
  - Scalare, eliminarea redundanţelor
- Structurarea reţelei, în funcţie de problemă (număr de neuroni, de straturi etc).
  - De obicei, se stabilesc prin încercări
- Antrenarea.
  - Învăţarea datelor din setul de antrenare
- Recunoaşterea (generalizarea).
  - Utilizarea RNA antrenate pentru a rezolva cazuri noi ale problemei, neincluse în datele de antrenare.

Definirea problemei

- Învăţarea supravegheată: perechi de date de intrare-ieşire cu valori cunoscute.

Nr.	Titular contract	Persoane	Suprafaţă locuinţă [mp]	Studii	Venit lunar [lei]	Consum mediu [kWh /lună]	Putere instalată [kW]
			x_1		x_2	y	x_3
1	Nicolae Popa	3	110	superioare	3350	482	10
2	Vasile Ionescu	5	33	liceu	3100	208	8
3	Maria Popovici	2	42	superioare	5480	250	15
4	Adriana Elisei	3	138	superioare	4450	550	20
-	...	...	...	...	...	...	...
7	Paul Irimciuc	4	28.5	liceu	2500	108	6
9	Florin Mihalcea	2	153	doctorat	6600	573	12

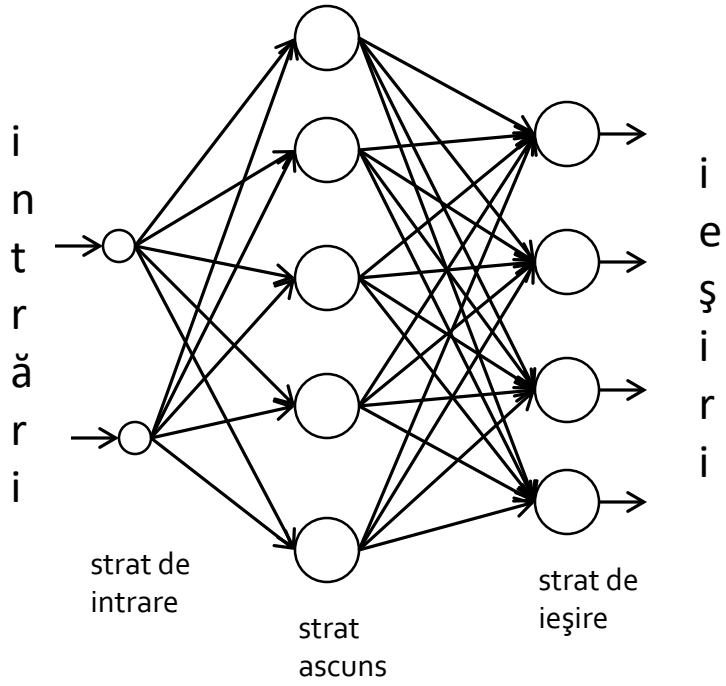
intrări (3) ieşiri (1)

Nr.	Suprafaţă locuinţă [mp]	Venit lunar [lei]	Putere instalată [kW]	Consum mediu [kWh /lună]
	x_1	x_2	x_3	y
1	110	3350	10	482
2	33	3100	8	208
3	42	5480	15	250
4	138	4450	20	550
-	...	...	...	...
7	28.5	2500	6	108
9	153	6600	12	573

- Exemplu: estimarea consumului unui client nou în funcţie de suprafaţa locuinţei, venitul lunar şi puterea totală instalată a receptoarelor electrice din casă.
- RNA trebuie să înveţe dependenţa *consum=f(suprafaţă, venit, putere instalată)* folosind datele clienţilor deja aflaţi sub contract cu furnizorul şi să fie capabilă să prognozeze corect consumul unor clienţi noi, pentru care se cunosc suprafaţa casei, venitul lunar şi puterea receptoarelor folosite în casă.

Arhitectura RNA

- Perceptronul multistrat (PMS) (multilayer perceptron, MLP):
  - ❖ Reţea feedforward: informaţia curculă doar într-un sens, de la intrare către ieşire
  - ❖ Fiecare neuron este conectat cu fiecare neuron din straturile adiacente.
  - ❖ Nu există conexiuni între neuronii de pe acelaşi strat
  - ❖ Are minim trei straturi: unul de intrare, cel puţin unul ascuns şi unul de ieşire.

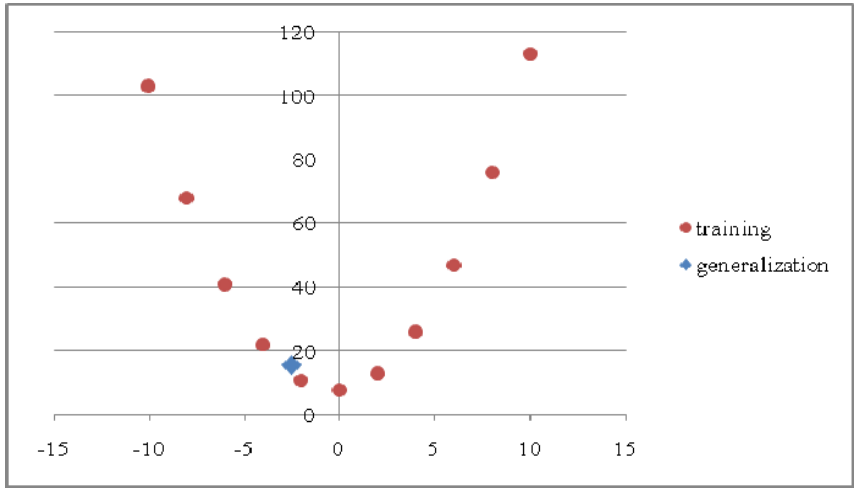


Exemplu: RNA PMS cu 3 straturi:
- •2 neuroni pe stratul de intrare
- •5 neuroni pe stratul ascuns
- •4 neuroni pe stratul de ieşire

Pregătirea datelor de intrare

- Datele de antrenare trebuie să fie suficiente pentru a ilustra variaţia reală a ieşirilor în funcţie de intrări.
  - O parabolă nu poate fi aproximată corect prin trei puncte.
- Datele de antrenare trebuie să fie relevante.
  - RNA nu va aproxima corect zona x=[-15,0] a parabolei dacă e antrenată doar cu date din zona x=[0,15]

- Uneori, este necesară scalarea datelor, adică aducerea lor la aproximativ acelaşi ordin de mărime.



Structurarea rețelelor PMS

- Numărul de neuroni de pe stratul de intrare şi de pe stratul ascuns depinde de problemă. Pentru problema

  consum=f(suprafaţă, venit, putere instalată)

  sunt necesari 3 neuroni pe stratul de intrare şi 1 neuron pe stratul de ieşire.

- Numărul de neuroni pe stratul ascuns nu e fixat.
  - Dacă se alege mic, poate fi insuficient pentru complexitatea problemei.
  - Dacă e prea mare, poate apărea supraantrenarea (specializarea).
  - Se preferă nr neuroni ascunşi = între 1 şi 2* nr intrări.
  - De obicei, se stabileşte prin încercări.

Structurarea rețelelor PMS

- Funcţia de activare a stratului de intrare este întotdeauna liniară. De multe ori, stratul de intrare nici măcar nu mai este reprezentat grafic.

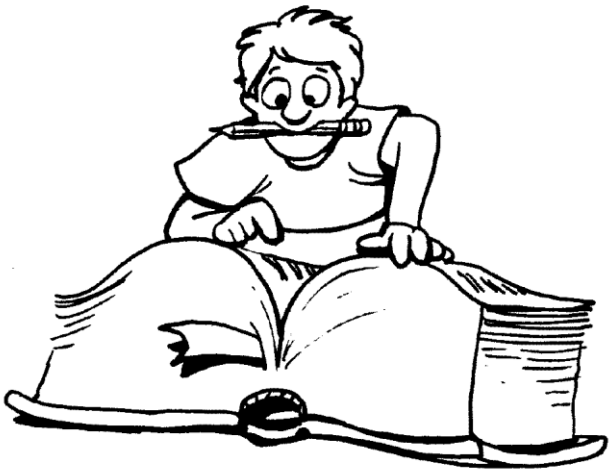
- Pe straturile ascunse, de obicei se folosesc funcţia sigmoid logistic, tangent hiperbolic sau liniară, în funcţie de tipul de problemă (regresie, prognoză, clasificare).

- RNA cu mai mult de un strat ascuns se numesc reţele adânci (deep neural networks).

- Cel mai frecvent întâlnite sunt RNA cu un singur strat ascuns.

Antrenarea rețelelor PMS

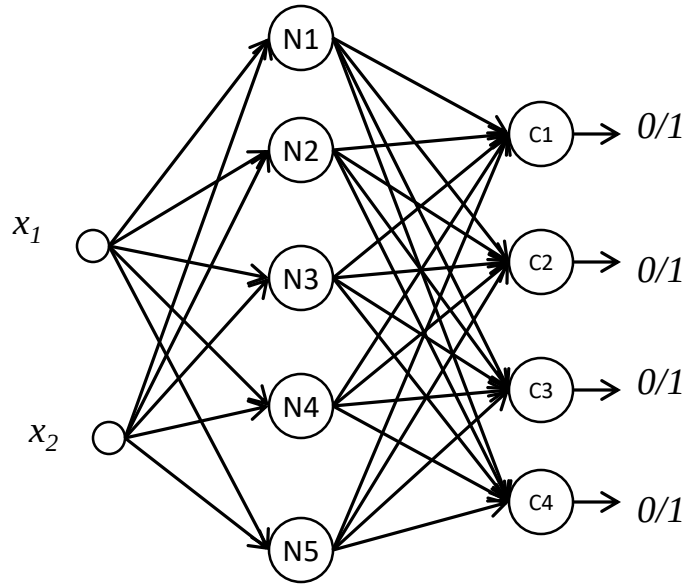
- Antrenarea (învăţarea) înseamnă determinarea valorilor ponderilor conexiunilor sinaptice dintre neuroni w_{ij} şi a pragurilor b_i ale fiecărui neuron care asigură performanţele optime ale reţelei (rezolvă problema), folosind un algoritm specific.
  - Cel mai cunoscut algoritm de antrenare pentru PMS este retropropagarea (backpropagation).
  - Generalizare a algoritmului aplicat pentru neuronul elementar.

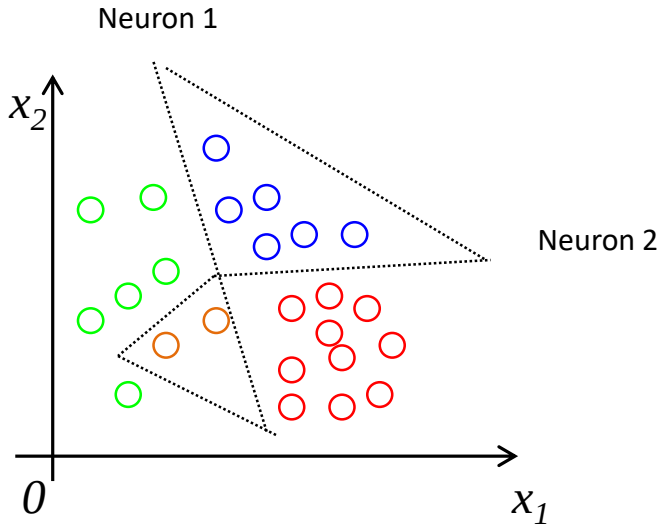


Antrenarea RNA

- Felul în care RNA "cartografiază" intrările către ieşiri nu poate fi controlat de către utilizator, RNA "decide" singură, în funcţie de modelele existente în setul de antrenare şi de valorile iniţiale ale ponderilor şi pragurilor, care de obicei sunt iniaţializate aleatoriu.

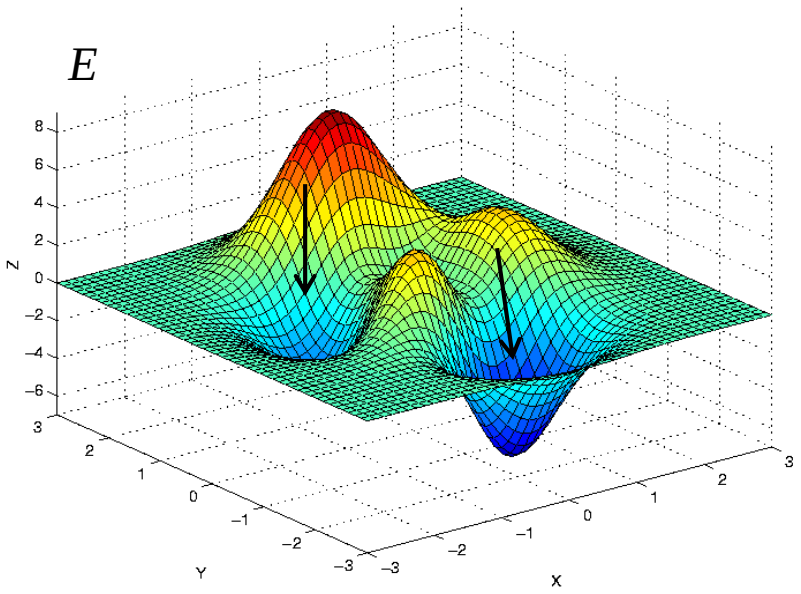
- RNA sunt considerate "cutii negre" (black boxes)





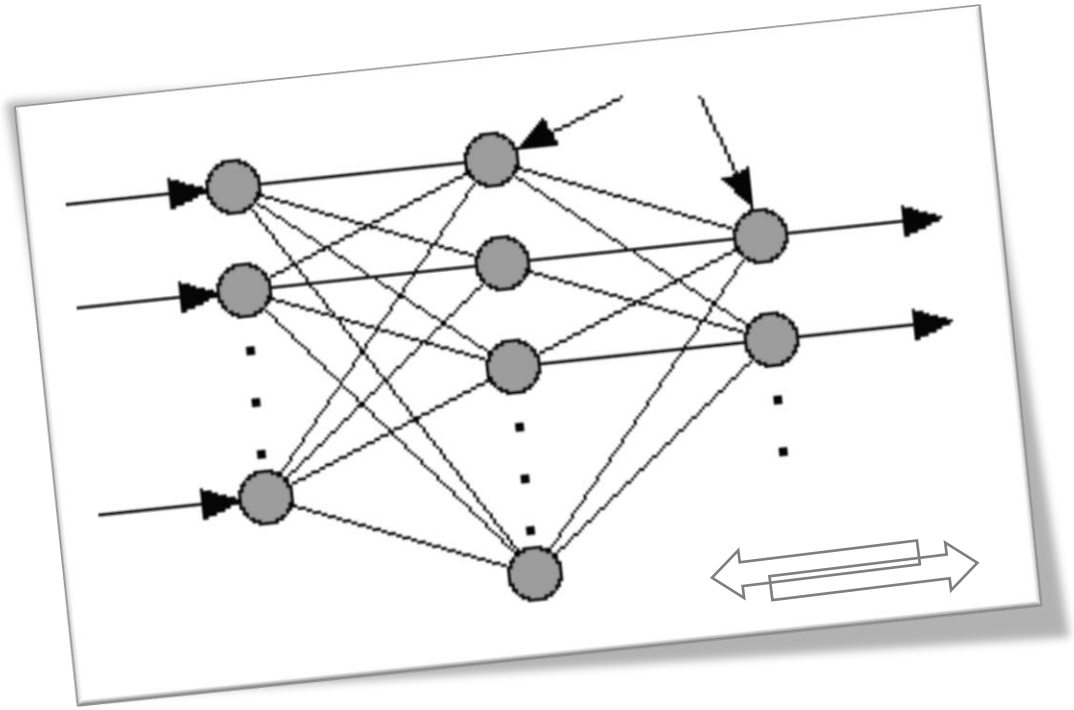
Antrenarea RNA

- Cu acelaşi set de date şi folosind acelaşi algoritm de antrenare , performanţele reţelei pot fi diferite de la o antrenare la alta, deoarece ponderile şi pragurile iniţiale sunt diferite şi antrenarea porneşte din alt punct al suprafeţei erorii, care nu mai este convexă din cauza suprapunerii efectelor mai multor neuroni.



Generalizarea

- Folosind ponderile şi pragurile corectate în procesul de antrenare, RNA trebuie să calculeze valori corecte de ieşire ale problemei pentru oricare model de intrare neinclus în setul de antrenare.
  - “Corecte” înseamnă cu o anumită aproximaţie considerată satisfăcătoare.
- Astfel RNA



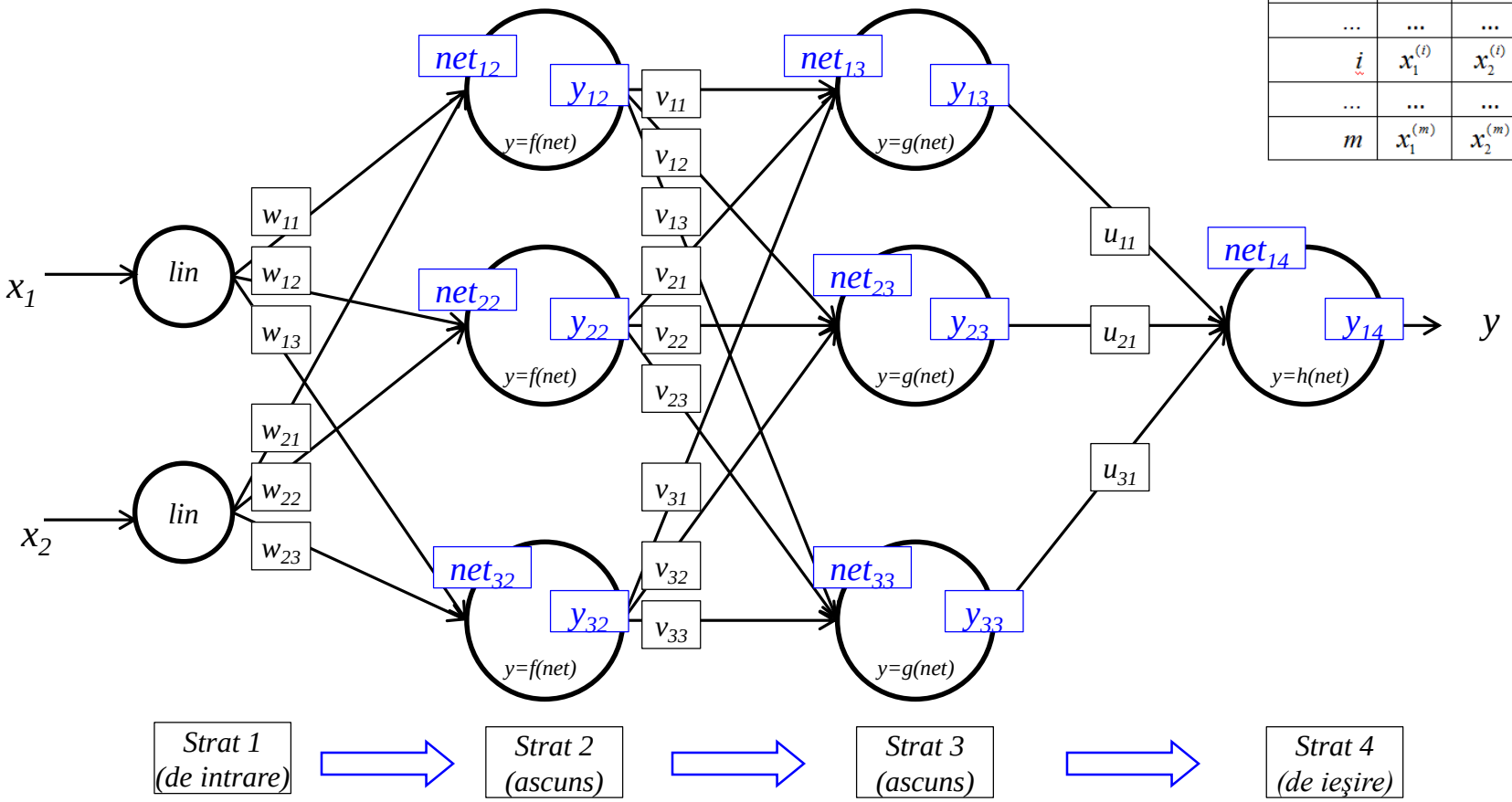
Algoritmul de retropropagare pentru perceptronul multistrat

Paşii algoritmului de retropropagare

- Pregătire date de intrare sub forma de perechi intrare-ieşire.
- Iniţializare arhitectură RNA: număr de straturi, număr de neuroni pe fiecare strat, inițializare aleatoare ponderi şi praguri.
- repetă
  - Propagare înainte pentru toate modelele de antrenare.
  - Calcul eroare între rezultatele obţinute şi cele dorite.
  - Propagare înapoi: corecţie ponderi şi praguri în funcţie de eroare.
  - Verifică criteriul de oprire.
- Dacă s-a îndeplinit criteriul de oprire, antrenarea s-a încheiat.

Propagare înainte intrare

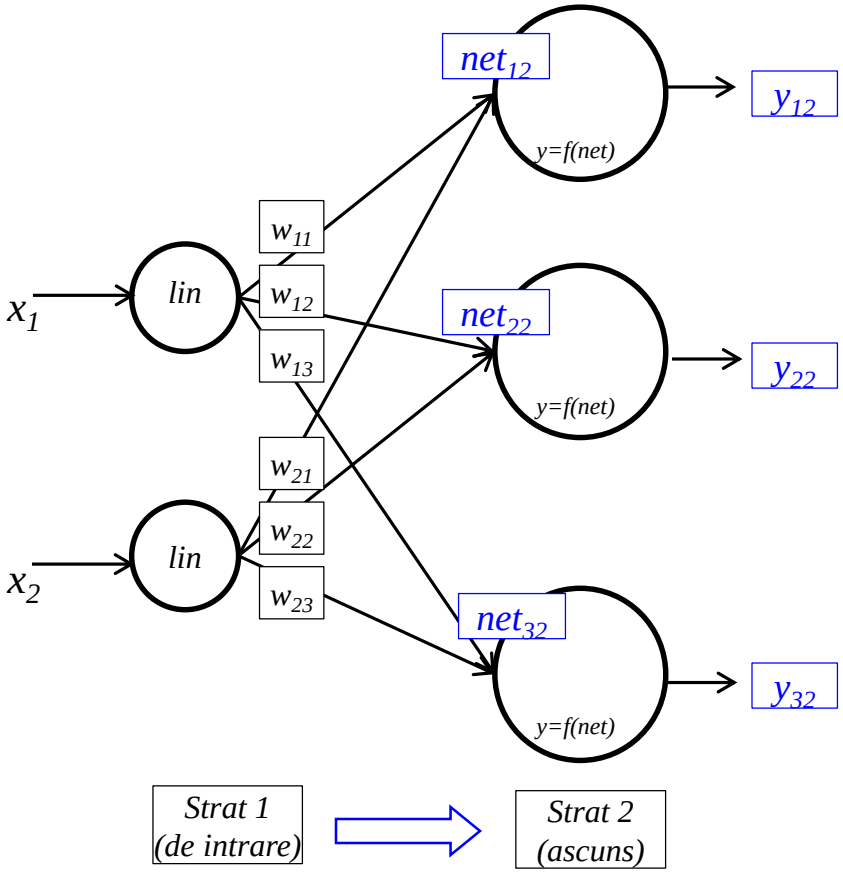
model	$y=f(x_1, x_2)$		
	x_1	x_2	y
1	$x_1^{(1)}$	$x_2^{(1)}$	$y^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$	$y^{(2)}$
...	...	...	...
i	$x_1^{(i)}$	$x_2^{(i)}$	$y^{(i)}$
...	...	...	...
m	$x_1^{(m)}$	$x_2^{(m)}$	$y^{(m)}$

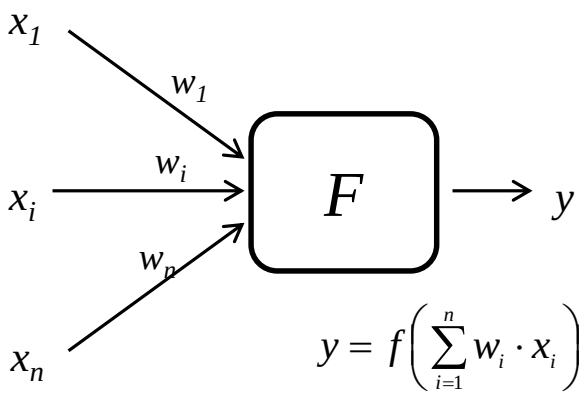


- Exemplu: RNA cu 4 straturi, 2 intrări şi 1 ieşire.
- Se calculează ieşirea y, pentru toate modelele i ($i=1 \ldots m$), în funcţie de valorile curente ale ponderilor.
- Pentru simplificare, pragurile se consideră 0 şi nu intervin în calcule.

Propagare înainte intrare

- Propagare înainte între straturile 1-2, un model de intrare

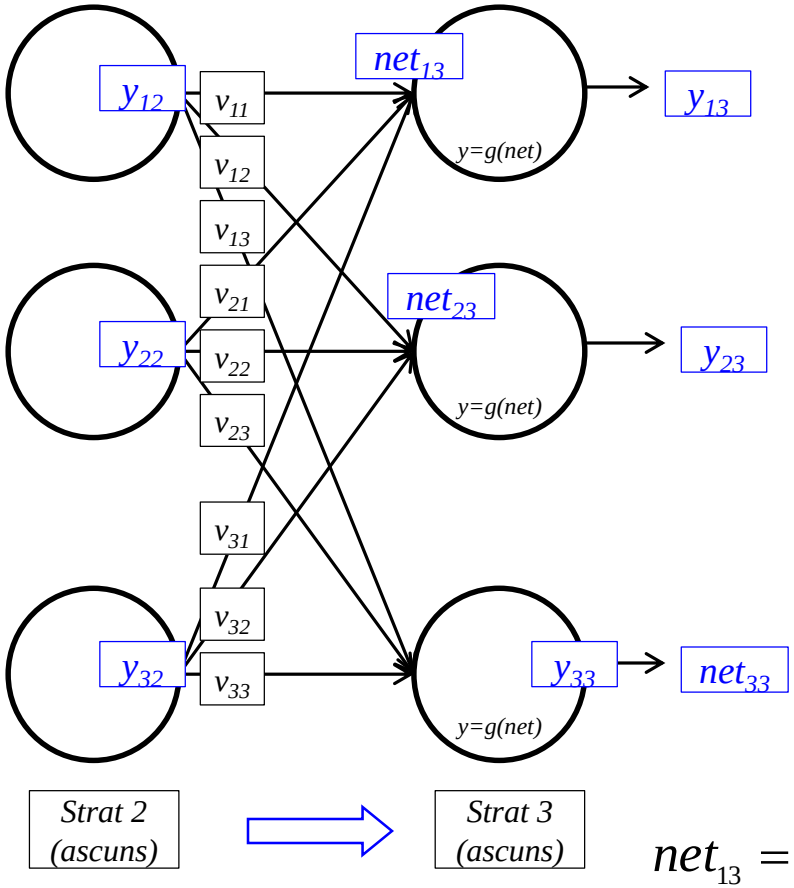




$$net_{12} = w_{11} \cdot x_1 + w_{21} \cdot x_2 \qquad y_{12} = f(net_{12})$$
$$net_{22} = w_{12} \cdot x_1 + w_{22} \cdot x_2 \qquad y_{22} = f(net_{22})$$
$$net_{32} = w_{13} \cdot x_1 + w_{23} \cdot x_2 \qquad y_{32} = f(net_{23})$$

Propagare înainte intrare

- Propagare înainte între straturile 2-3



$$net_{13} = v_{11} \cdot y_{12} + v_{21} \cdot y_{22} + v_{31} \cdot y_{32} \qquad y_{13} = g(net_{13})$$

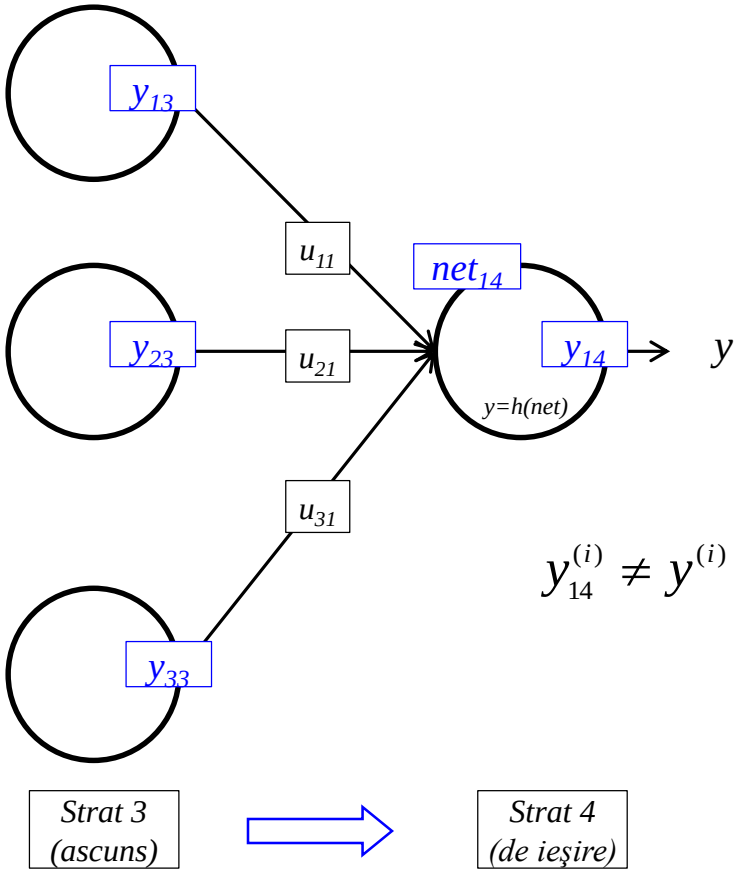
$$net_{23} = v_{12} \cdot y_{12} + v_{22} \cdot y_{22} + v_{32} \cdot y_{32} \qquad y_{23} = g(net_{23})$$

$$net_{33} = v_{13} \cdot y_{12} + v_{23} \cdot y_{22} + v_{33} \cdot y_{32} \qquad y_{33} = g(net_{33})$$

Propagare înainte intrare

model	$y=f(x_1, x_2)$		
	x_1	x_2	y
1	$x_1^{(1)}$	$x_2^{(1)}$	$y^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$	$y^{(2)}$
...	...	...	...
i	$x_1^{(i)}$	$x_2^{(i)}$	$y^{(i)}$
...	...	...	...
m	$x_1^{(m)}$	$x_2^{(m)}$	$y^{(m)}$

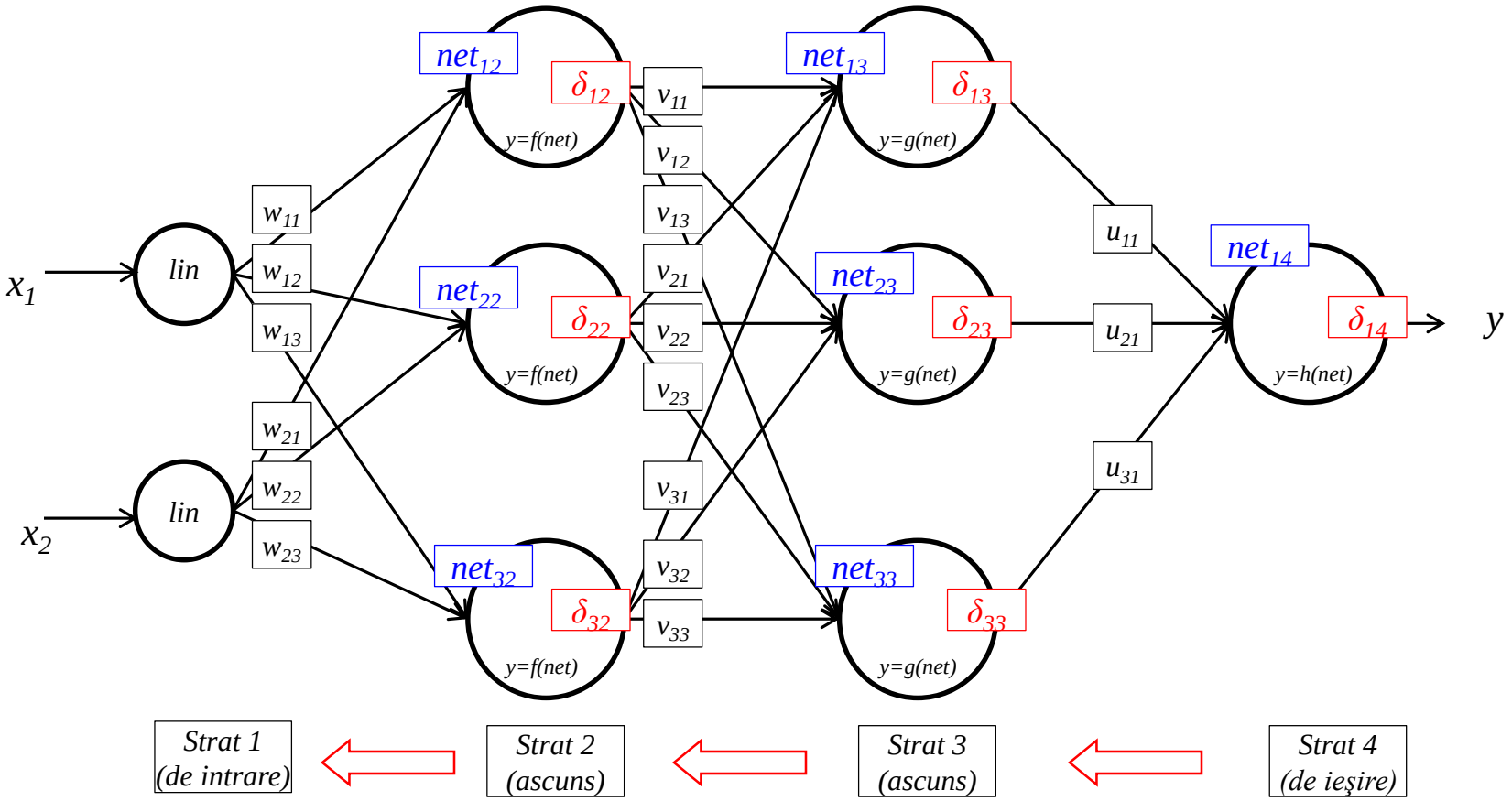
- Propagare înainte între straturile 3-4



$$net_{14} = u_{11} \cdot y_{13} + u_{21} \cdot y_{23} + u_{31} \cdot y_{33}$$
$$y_{14} = h(net_{14})$$

- Deoarece ponderile au fost iniţializate aleatoriu şi foarte probabil nu sunt cele care rezolvă corect problema, ieşirea y_{14}, cea obţinută pentru un model oarecare i, $o^{(i)}$, va diferi de ieşirea dorită $d^{(i)}$.
- Dorim să apropiem ponderile de cele corecte.
- Se calculează eroarea $\delta^{(i)}$ între ieşirea obţinută $o^{(i)}$ şi cea dorită $d^{(i)}$.
- Se repetă propagarea înainte pentru toate modelele, se calculează de fiecare dată eroarea şi se realizează suma erorilor pe toate modelele de antrenare.
- Se face propagarea înapoi a erorii, pentru corectarea ponderilor.

Propagare înapoi erori

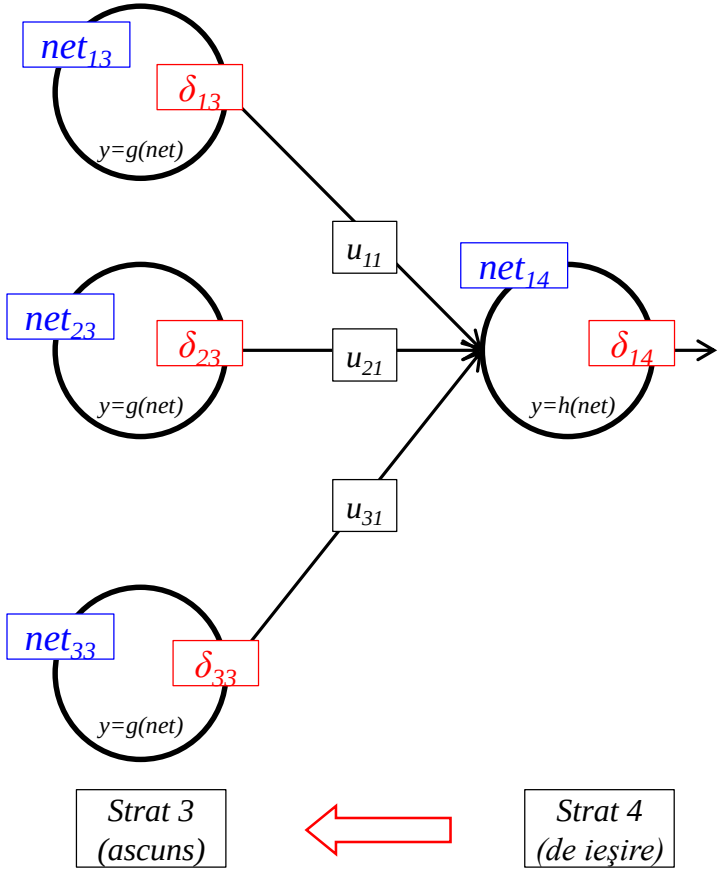


$$\delta_{ij} = \frac{\partial}{\partial net_{ij}} (\textit{functie eroare})$$

Propagare înapoi erori

- Propagare eroare înapoi între straturile 4 şi 3

$$\delta_{14} = y^{(i)} - y_{14}^{(y)} = y^{(i)} - h(net_{14})$$
$$\delta_{13} = (u_{11} \cdot \delta_{14}) \cdot g'(net_{13})$$
$$\delta_{23} = (u_{21} \cdot \delta_{14}) \cdot g'(net_{23})$$
$$\delta_{23} = (u_{31} \cdot \delta_{14}) \cdot g'(net_{33})$$



Propagare înapoi erori

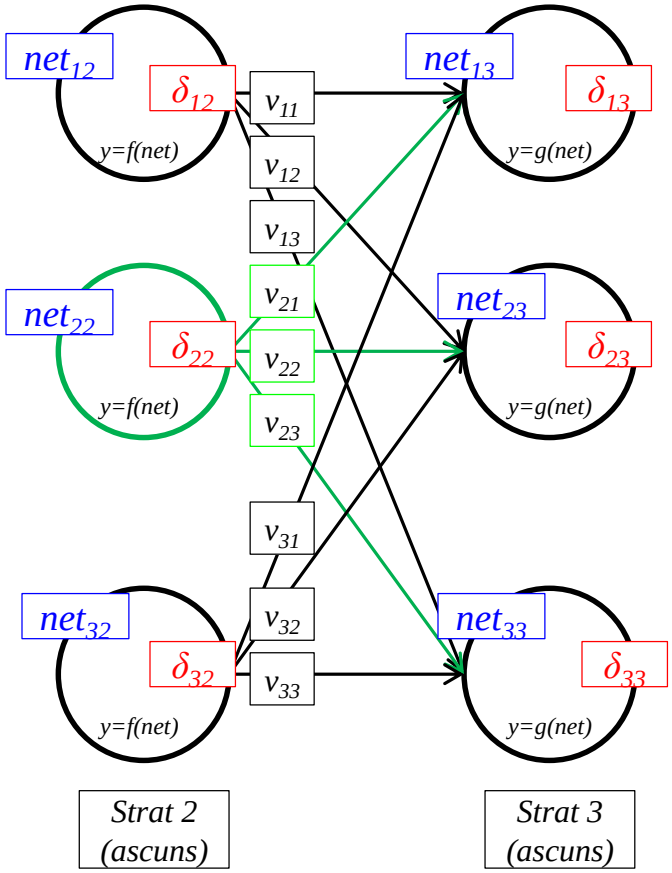
- Propagare eroare înapoi între straturile 3 şi 2

$$\delta_{12} = (v_{11} \cdot \delta_{13} + v_{12} \cdot \delta_{23} + v_{13} \cdot \delta_{33}) \cdot f'(net_{12})$$

$$\delta_{22} = (v_{21} \cdot \delta_{13} + v_{22} \cdot \delta_{23} + v_{23} \cdot \delta_{33}) \cdot f'(net_{22})$$

$$\delta_{13} = (v_{31} \cdot \delta_{13} + v_{32} \cdot \delta_{23} + v_{33} \cdot \delta_{33}) \cdot f'(net_{13})$$

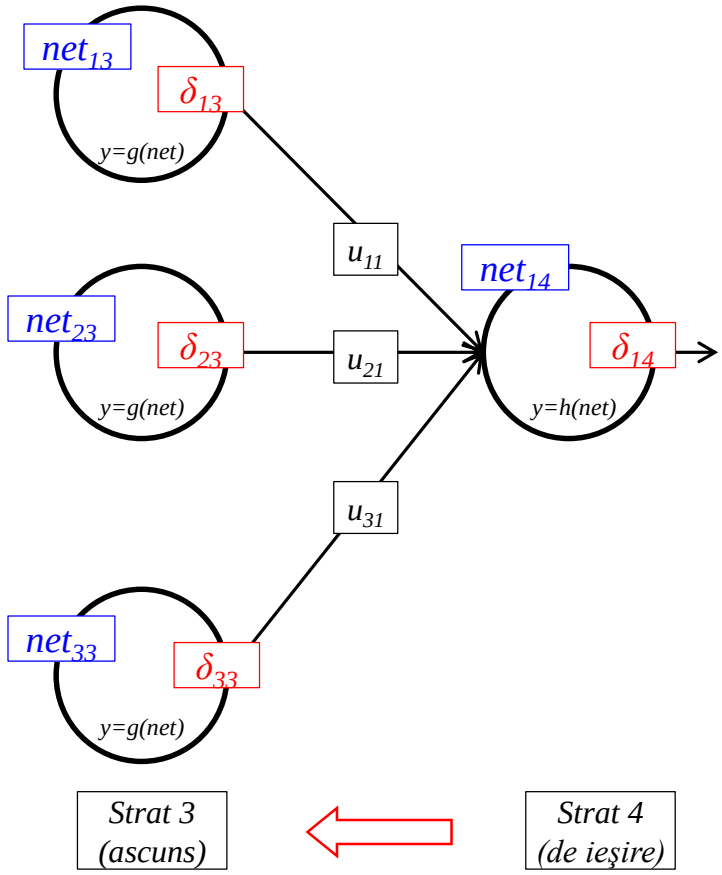
- Între straturile 2 şi 1 nu mai putem calcula erori



Corecție ponderi

- Între straturile 3-4

$$u_{11} = u_{11} + \Delta u_{11} = u_{11} + \eta \cdot \delta_{14} \cdot net_{13}$$
$$u_{12} = u_{12} + \Delta u_{12} = u_{12} + \eta \cdot \delta_{14} \cdot net_{23}$$
$$u_{13} = u_{13} + \Delta u_{13} = u_{13} + \eta \cdot \delta_{14} \cdot net_{33}$$



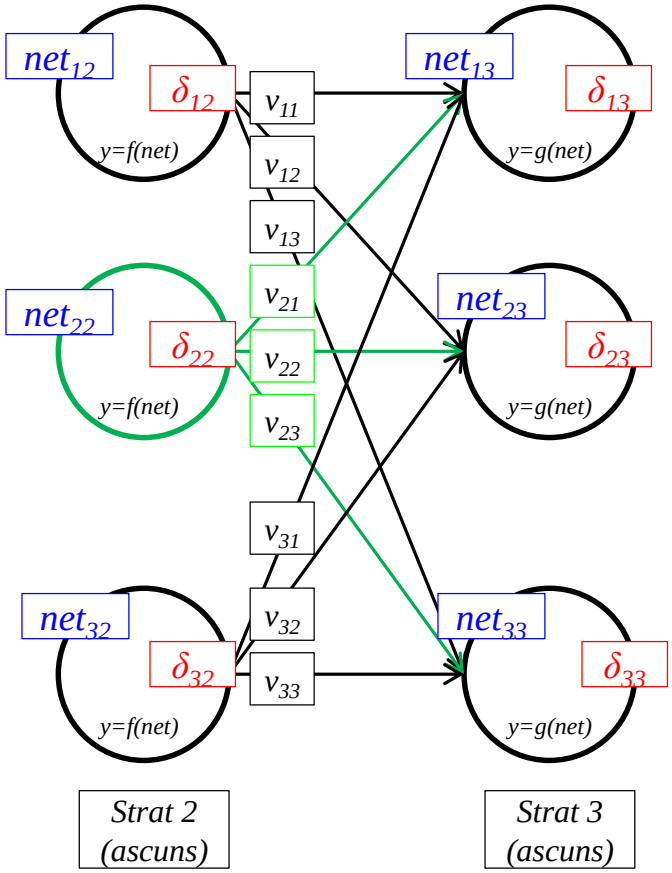
Corecție ponderi

- Între straturile 2-3

$$v_{11} = v_{11} + \Delta v_{11} = v_{11} + \eta \cdot \delta_{13} \cdot net_{12}$$
$$v_{12} = v_{12} + \Delta v_{12} = v_{12} + \eta \cdot \delta_{23} \cdot net_{12}$$
$$v_{13} = v_{13} + \Delta v_{13} = v_{13} + \eta \cdot \delta_{33} \cdot net_{12}$$

$$v_{21} = v_{21} + \Delta v_{21} = v_{21} + \eta \cdot \delta_{13} \cdot net_{22}$$
$$v_{22} = v_{22} + \Delta v_{22} = v_{22} + \eta \cdot \delta_{23} \cdot net_{22}$$
$$v_{23} = v_{23} + \Delta v_{23} = v_{23} + \eta \cdot \delta_{33} \cdot net_{22}$$

$$v_{31} = v_{31} + \Delta v_{31} = v_{31} + \eta \cdot \delta_{13} \cdot net_{32}$$
$$v_{32} = v_{32} + \Delta v_{32} = v_{32} + \eta \cdot \delta_{23} \cdot net_{32}$$
$$v_{33} = v_{33} + \Delta v_{33} = v_{33} + \eta \cdot \delta_{33} \cdot net_{32}$$

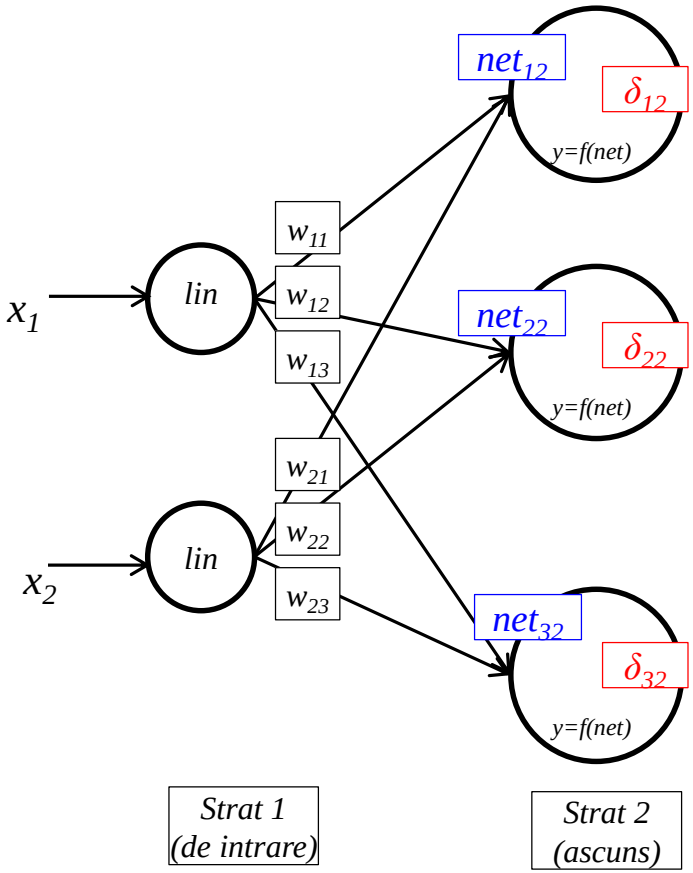


Corecție ponderi

- Între straturile 1-2

$$w_{11} = w_{11} + \Delta w_{11} = w_{11} + \eta \cdot \delta_{12} \cdot x_1$$
$$w_{12} = w_{12} + \Delta w_{12} = w_{12} + \eta \cdot \delta_{22} \cdot x_1$$
$$w_{13} = w_{13} + \Delta w_{13} = w_{13} + \eta \cdot \delta_{32} \cdot x_1$$

$$w_{21} = w_{21} + \Delta w_{21} = w_{21} + \eta \cdot \delta_{12} \cdot x_2$$
$$w_{22} = w_{22} + \Delta w_{22} = w_{22} + \eta \cdot \delta_{22} \cdot x_2$$
$$w_{23} = w_{23} + \Delta w_{23} = w_{23} + \eta \cdot \delta_{32} \cdot x_2$$



Algoritmul de retropropagare

- În acest moment, sunt recalculate toate ponderile.
- Dacă nu s-a îndeplinit criteriul de oprire, se trece la iteraţia următoare, cu o nouă propagare înainte.

- În continuare, se przintă generalizarea algoritmului de retropropagare pentru o RNA PMS de orice dimensiune.

Regula delta generalizată (1 din 2)

- Pentru fiecare model de intrare – ieşire m din setul de antrenare, corecţia unei ponderi w_{ij} – notată $\Delta^{(m)}w_{ij}$ – pentru conexiunea dintre neuronul j şi neuronul i din stratul inferior (vezi Fig. 4.4.a) este proporţională cu un termen de eroare $\delta_j^{(m)}$ asociat neuronului j:

$$\Delta^{(m)}w_{ij} = \eta \cdot \delta_j^{(m)} \cdot o_i^{(m)}$$

unde $o_i^{(m)}$ este ieşirea neuronului i din stratul inferior, pentru modelul m, iar η este un factor de proporţionalitate, numit rată de învăţare.

Regula delta generalizată (2 din 2)

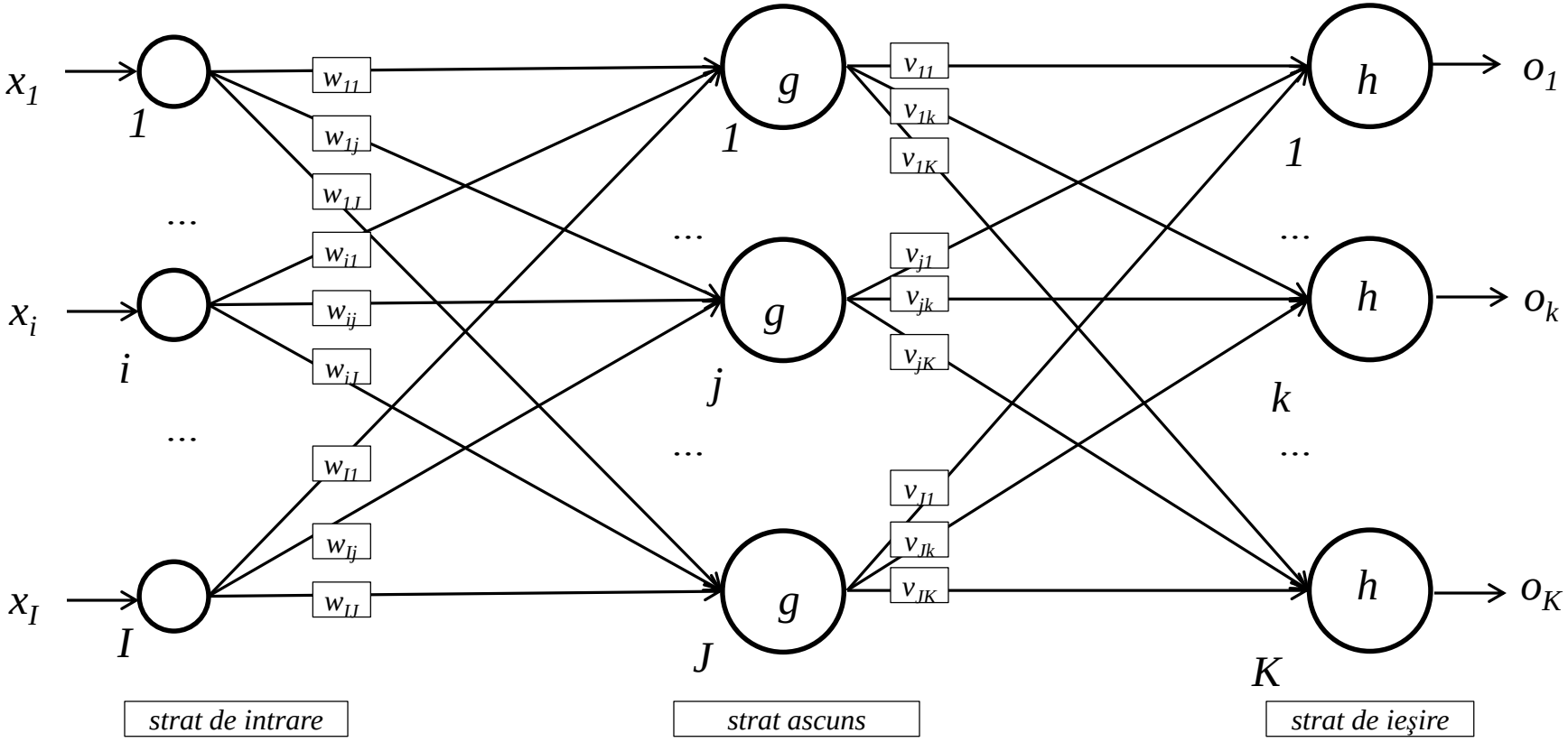
- Dacă neuronul j se află în stratul de ieşire, termenul de eroare $\delta_j^{(m)}$ se calculează în funcţie de abaterea între valoarea reală $o_j^{(m)}$ şi cea dorită $d_j^{(m)}$ şi derivata funcţiei de activare f a neuronului j în raport cu intrarea netă corespunzătoare modelului m, $net_j^{(m)}$:

$$\delta_j^{(m)} = \left(d_j^{(m)} - o_j^{(m)}\right) \cdot f'\left(net_j^{(m)}\right)$$

- Dacă neuronul j se află în stratul ascuns, fiind legat prin conexiuni sinaptice cu neuronii k din stratul de ieşire, termenul de eroare $\delta_j^{(m)}$ este proporţional cu suma tuturor termenilor de eroare asociaţi neuronilor de ieşire k, modificaţi de ponderile conexiunilor respective w_{jk} şi cu derivata funcţiei de activare în raport cu intrarea netă $net_j^{(m)}$:

$$\delta_j^{(m)} = \left(\sum_k \delta_k^{(m)} \cdot w_{jk}\right) \cdot f'\left(net_j^{(m)}\right)$$

Algoritmul de retropropagare pentru PMS cu un strat ascuns



RNA PMS cu 3 straturi

- I neuroni pe stratul de intrare, neuron generic i
- J neuroni pe stratul ascuns, neuron generic j
- K neuroni pe stratul de ieşire, neuron generic k

Algoritmul de retropropagare

- RNA este antrenată cu un set de M modele de antrenare, perechi de intrare-ieşire $(x_1, .., x_i, \ldots, x_I)^{(m)} - (d_1, \ldots, d_k, \ldots, d_K)^{(m)}$, $m=1 \ldots M$.
- Funcţia de activare a primului strat este identitatea; funcţiile de activare ale celorlalte două straturi sunt g (pe stratul ascuns) şi h (pe stratul de ieşire).
- Ponderile dintre stratul de intrare şi cel ascuns se notează cu w_{ij}, cu $i=1 \ldots I$, respectiv $j=1 \ldots J$.
- Ponderile dintre stratul de intrare şi cel ascuns se notează cu v_{jk}, cu $j=1 \ldots J$, respectiv $k=1 \ldots K$.
- Intrările nete ale neuronilor pentru un model m
  - pe stratul ascuns:

$$r_j^m = \sum_{i=1}^{i} w_{ij} \cdot x_i^{(m)}$$

  - Pe stratul de ieşire:

$$q_k^m = \sum_{j=1}^{J} v_{jk} \cdot y_j^{(m)}$$

unde $y_j^{(m)} = g(r_j^{(m)})$

şi $o_k^{(m)} = h(q_k^{(m)})$

Algoritmul de retropropagare

- Formula erorii – abaterea pătratică totală:

$$E(w,v) = \frac{1}{2} \cdot \sum_{m=1}^{M} \left(d^{(m)} - o(w,v,x)^{(m)} \right)^2$$

- Scrisă desfăşurat:

$$E(w,v) = \frac{1}{2} \sum_{m=1}^{M} \left\| d^{(m)} - o^{(m)} \right\| = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left(d_k^{(m)} - o_k^{(m)} \right)^2 =$$

$$= \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(q_k^{(m)} \right) \right]^2 = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(\sum_{j=1}^{J} v_{jk} \cdot g\left(r_j^{(m)} \right) \right) \right]^2 =$$

$$= \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(\sum_{j=1}^{J} v_{jk} \cdot y_j^{(m)} \right) \right]^2 = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \left[d_k^{(m)} - h\left(\sum_{j=1}^{J} v_{jk} \cdot g\left(\sum_{i=1}^{I} w_{ij} \cdot x_i^{(m)} \right) \right) \right]^2$$

Algoritmul de retropropagare

- Dacă se foloseşte metoda gradientului, în iteraţia *t* ponderile se vor corecta în fiecare iteraţie folosind formulele:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \cdot \left. \frac{\partial E}{\partial w_{ij}^{(t)}} \right|_{w=w_{ij}^{(t)}}, \quad i = 1...I,\ j = 1...J$$

$$v_{jk}^{(t+1)} = v_{jk}^{(t)} - \eta \cdot \left. \frac{\partial E}{\partial v_{ijk}^{(t)}} \right|_{v=v_{jk}^{(t)}}, \quad j = 1...J, k = 1...K$$

Algoritmul de retropropagare după metoda gradientului

- Pentru ponderile dintre stratul ascuns şi cel de ieşire:

$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial v_{jk}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial q_k} \cdot \frac{\partial q_k}{\partial v_{jk}} =$$

$$= 2 \cdot (d_k - o_k) \cdot (-1) \cdot h'(q_k) \cdot y_j = -2 \cdot (d_k - o_k) \cdot h'(q_k) \cdot y_j$$

- Pentru ponderile dintre stratul de intrare şi cel ascuns:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial r_j} \cdot \frac{\partial r_j}{\partial w_{ij}} =$$

$$= -2 \cdot \left[\sum_{k=1}^{K} (d_k - o_k) \cdot h'(q_k) \cdot v_{jk} \right] \cdot g'(r_j) \cdot x_i$$

Algoritmul de retropropagare după metoda gradientului

- Dacă se foloseşte ca funcţie de activare sigmoidul logistic:

$$g(p) = h(p) = \frac{1}{1+e^{-p}}, \quad g' = h' = p \cdot (1-p)$$

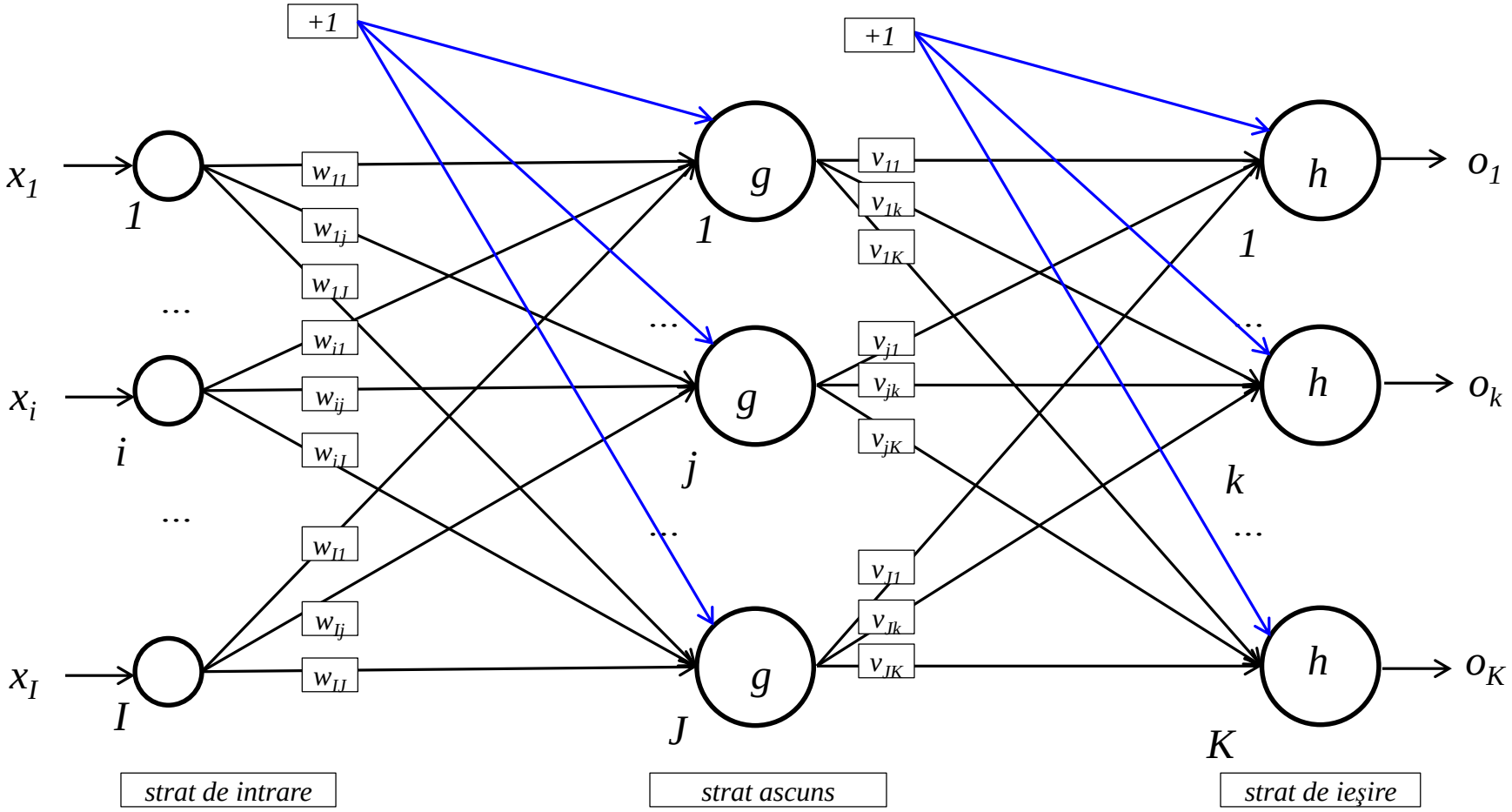
- Derivatele erorilor se vor calcula:

$$\frac{\partial E}{\partial v_{jk}} = -2 \cdot (d_k - o_k) \cdot o_k \cdot (1-o_k) \cdot y_j$$

$$\frac{\partial E}{\partial w_{ij}} = -2 \cdot \left[\sum_{k=1}^{K} (d_k - o_k) \cdot o_k \cdot (1-o_k) \cdot v_{jk} \right] \cdot y_j \cdot (1-y_j) \cdot x_i$$

Algoritmul de retropropagare

- Pragurile pot fi incluse în algoritmul de antrenare considerându-le drept ponderea unei mărimi suplimentare de valoare + 1, actualizate cu aceleaşi formule generale de calcul.



Tipuri de învățare pentru RNA PMS

- Cu adaptare în bloc a ponderilor (batch learning)
- Cu adaptare model cu model (online, stochastic)
- Cu adaptare în miniblocuri (mini-batch learning)
  - Algoritmul mini-batch este folosit în principal la antrenarea cu seturi de date foarte mari, cu sute de mii sau milioane de modele.

Algoritmul de retropropagare batch

- Adaptarea ponderilor se face o singură dată, la capătul iteraţiei, după propagarea înainte a tuturor modelelor de intrare şi cumularea erorii.

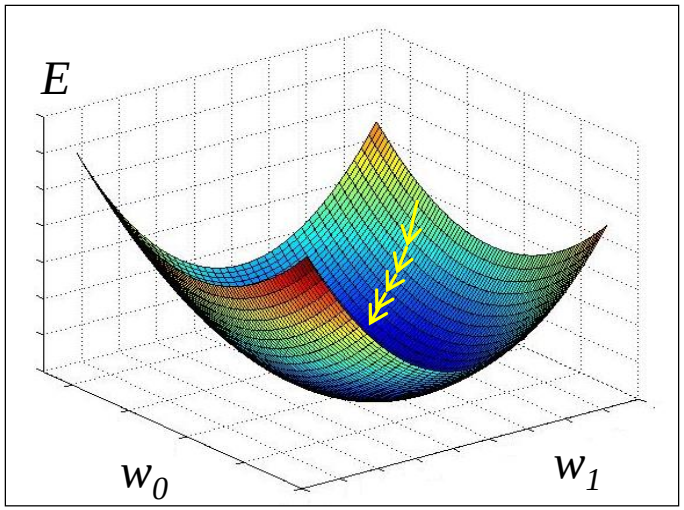
- Pregătire date de intrare sub forma de perechi intrare-ieşire.
- Iniţializare arhitectură RNA: număr de straturi, număr de neuroni pe fiecare strat, inițializare aleatoare ponderi şi praguri.
- repetă
  - ❖ Pentru fiecare model de antrenare
    - Propagare înainte model
      - Stratul 1-2
      - Stratul 2-3
    - Calcul eroare între rezultatele obţinute şi cele dorite şi sumarea erorii modelului individual la eroarea globală a setului de antrenare.
  - ❖ Propagare înapoi:
    - Calcul corecţii ponderi în funcţie de eroarea globală.
      - Stratul 2-3
      - Stratul 1-2
    - Aplicare corecţii ponderi
  - ❖ Verifică criteriul de oprire.
- Dacă s-a îndeplinit criteriul de oprire, antrenarea s-a încheiat.

Algoritmul de retropropagare online

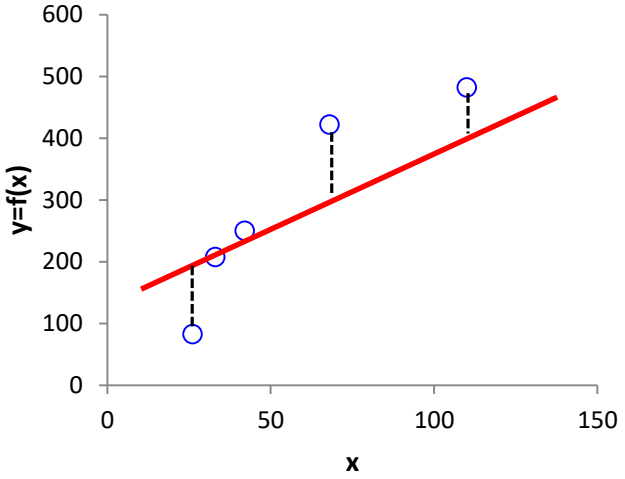
- Adaptarea ponderilor se face după propagarea înainte-înapoi a fiecărui model din setul de antrenare.

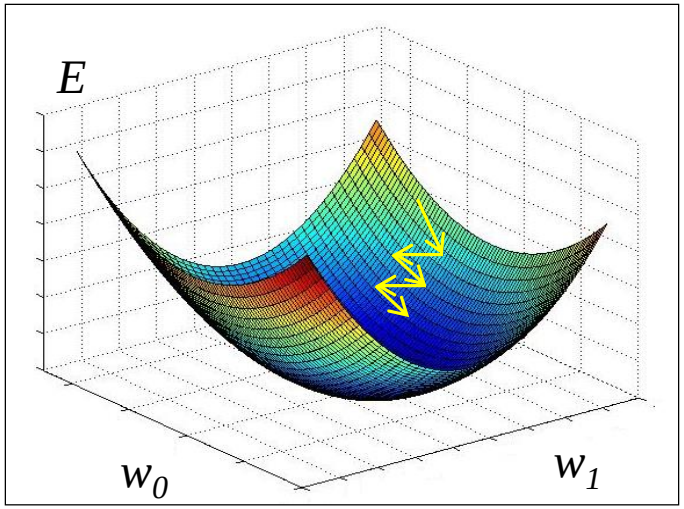
- Pregătire date de intrare sub forma de perechi intrare-ieşire.
- Iniţializare arhitectură RNA: număr de straturi, număr de neuroni pe fiecare strat, inițializare aleatoare ponderi şi praguri.
- repetă
- pentru fiecare model de intrare, ales aleatoriu din setul de antrenare
  - ❖ Propagare înainte.
    - Stratul 1-2
    - Stratul 2-3
  - ❖ Calcul eroare între rezultatele obţinute şi cele dorite.
  - ❖ Propagare înapoi:
    - Calcul corecţii ponderi.
      - Stratul 2-3
      - Stratul 1-2
    - Aplicare corecţii ponderi
  - ❖ Verifică criteriul de oprire.
- Dacă s-a îndeplinit criteriul de oprire, antrenarea s-a încheiat.

Învățarea în bloc vs. model cu model



în bloc





model cu model

Algoritmul de retropropagare mini-batch

- Adaptarea ponderilor se face după propagarea înainte-înapoi a unui bloc parţial din setul de antrenare (100-1000 de modele de intrare).

- Pregătire date de intrare sub forma de perechi intrare-ieşire.
- Iniţializare arhitectură RNA: număr de straturi, număr de neuroni pe fiecare strat, inițializare aleatoare ponderi şi praguri.
- repetă
  - ❖ pentru fiecare bloc de modele de intrare
    - Pentru fiecare model din bloc
    - Propagare înainte
      - » Stratul 1-2
      - » Stratul 2-3
      - – Calcul eroare între rezultatele obţinute şi cele dorite şi sumare eroare la eroarea globală a blocului.
  - ❖ Propagare înapoi
    - Calcul corecţii ponderi.
      - – Stratul 2-3
      - – Stratul 1-2
    - Aplicare corecţii ponderi
  - ❖ Verifică criteriul de oprire.
- Dacă s-a îndeplinit criteriul de oprire, antrenarea s-a încheiat.

Criterii de oprire a învățării

- Efectuarea unui număr fixat de iteraţii.
- Coborârea corecţiei erorii într-o iteraţie sub un prag limită.
- Stagnarea erorii pe parcursul unui număr de iteraţii.

- Metoda setului de validare:
  - ❖ Datele disponibile pentru setul de antrenare se împart în trei:
    - Setul de antrenare (60%)
    - Setul de validare (20%)
    - Setul test (20%)
  - ❖ Setul test rămâne pentru verificarea puterii de generalizare a RNA antrenate.
  - ❖ În fiecare iteraţie, datele de validare se folosesc ca un pseudo-test: dacă pe parcursul câtorva iteraţii eroarea pe setul de validare creşte, antrenarea se întrerupe (apare supraantrenarea sau specializarea).

Algoritme de antrenare pentru RNA PMS

- Există o varietate de algoritme de antrenare, nu doar retropropagarea simplă:

Function	Algorithm
trainlm	Levenberg-Marquardt
trainbr	Bayesian Regularization
trainbfg	BFGS Quasi-Newton
trainrp	Resilient Backpropagation
trainscg	Scaled Conjugate Gradient
traincgb	Conjugate Gradient with Powell/Beale Restarts
traincgf	Fletcher-Powell Conjugate Gradient
traincgp	Polak-Ribiére Conjugate Gradient
traingdx	Variable Learning Rate Gradient Descent
traingdm	Gradient Descent with Momentum
traingd	Gradient Descent

va urma...

- Metode de îmbunătăţire şi accelerare a învăţării pentru RNA cu învăţare supravegheată