PROCESAREA CUNOŞTINȚELOR ŞI CALCUL INTELIGENT - PCCI

Reţele neuronale artificiale pentru învăţare supravegheată. Metode de îmbunătăţire a învăţării

Cuprins:

- Problema supraantrenării
- Metode de îmbunătăţire a învăţării
  - Optimizarea datelor de intrare
  - Optimizarea funcţiei eroare
  - Optimizarea ponderilor
  - Algoritmul Resilient Backpropagation – RProp
  - Algoritme de învăţare alternative

Recapitulare

- Neuronul elementar:

$$y = b + w_1 \cdot x_1 + \ldots + w_i \cdot x_i + \ldots + w_n \cdot x_n =$$

$$= w_0 \cdot 1 + w_1 \cdot x_1 + \ldots + w_i \cdot x_i + \ldots + w_n \cdot x_n = \sum_{i=0}^{n} w_i \cdot x_i,\ x_0 = 1$$

- Algoritmul de retropropagare după metoda gradientului
- Eroarea:

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

- Ajustarea ponderilor:

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

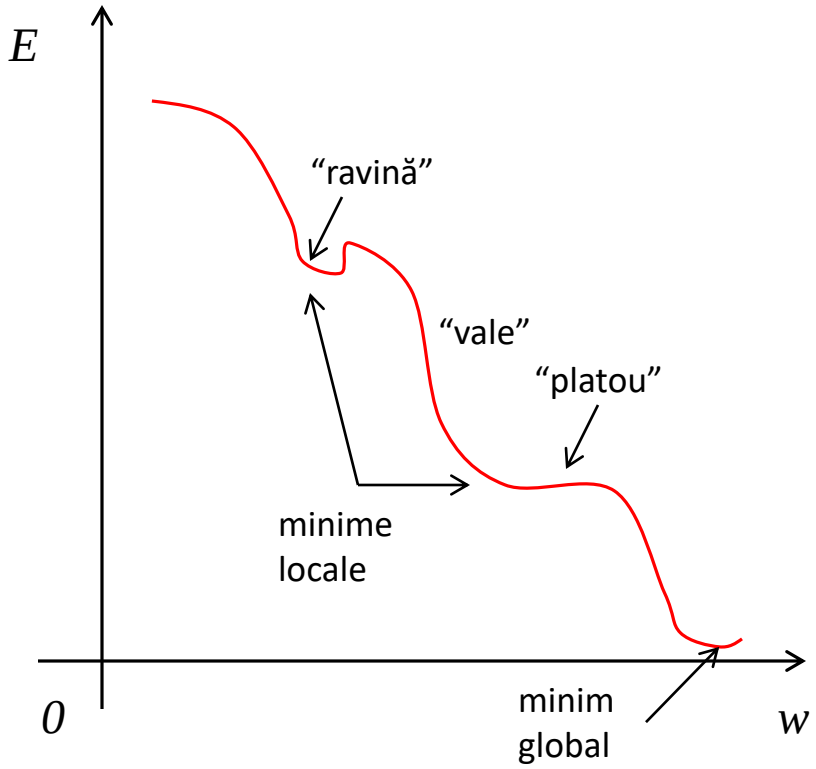
Recapitulare

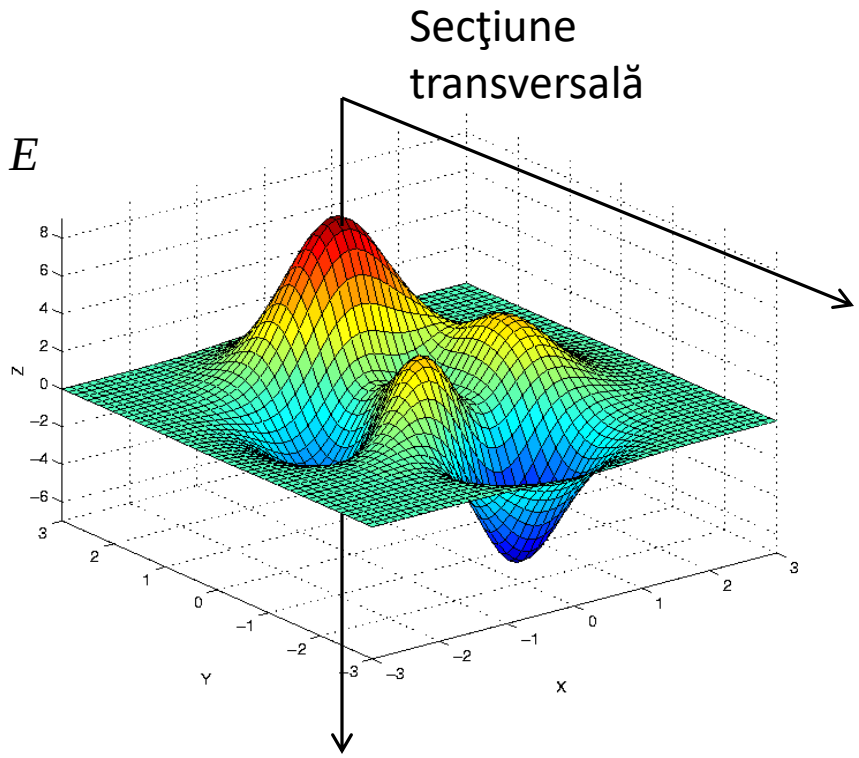
- Algoritmul de retropropagare:

  - Definire problemă şi pregătire date de intrare.
  - Definire arhitectură RNA.
  - Iniţializează ponderi şi praguri RNA.
  - Repetă până la îndeplinirea unui criteriu de oprire
    - Propagare înainte (calcul erori)
    - Propagare înapoi (corectare ponderi şi praguri în funcţie de eroare).

Recapitulare

- Folosirea unor modele de intrare cu mai mulţi parametri şi suprapunerea efectelor generate de mai mulţi neuroni interconectaţi fac suprafaţa erorii non-convexă, cu “dealuri” şi “văi”, cu “platouri” locale şi “ravine”, care pot încetini sau bloca atingerea obiectivului antrenării, a minimului global.





Antrenare şi generalizare

- În cazul unei antrenări ideale, eroarea scade repede, valorile de ieşire aproximate de RNA se apropie suficient de aproape de valorile de ieşire şi RNA este capabilă să generalizeze, adică să calculeze ieşiri rezonabil corecte pentru date de intrare noi.

- Scopul antrenării RNA este generalizarea.

- Soluţiile date de RNA nu trebuie să fie perfecte, cu rezonabil de aproape de cele exacte, cu eroare cât mai mică.

Posibile probleme de antrenare şi generalizare:

- Antrenarea are eroare mică, dar RNA nu generalizează bine, nu calculează corect ieşirea pentru modele de intrare noi.
  - RNA se află în unul din două cazuri: subantrenare (underfitting) sau supraantrenare (overffitting).
    - Datele de intrare: sunt suficiente şi relevante ?
    - Arhitectura reţelei (număr de straturi şi de neuroni) este adecvată ?
    - Funcţia de aproximare modelată de RNA este adecvată ?

- Învăţare insuficientă - eroarea de antrenare rămâne mare (blocaj în minim local)
  - Adaptarea ponderilor nu se desfăşoară corespunzător sau eroarea se blochează în minime locale sau “ravine”.
    - Ponderile şi pragurile au valori rezonabile ?
    - Rata de învăţare are o valoare adecvată ?

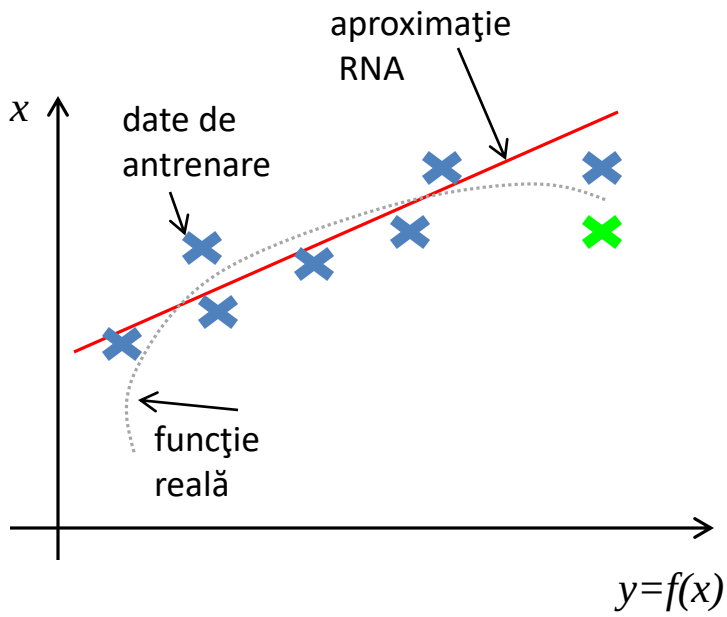
- Învăţare lentă
  - Eroarea nu trece suficient de repede peste minimele locale, “platouri” sau ”ravine”.
  - Corecţiile ponderilor sunt prea mici.
    - Prelucrarea suplimentară a datelor de antrenare (uniformizare)
    - Factori de corecţie suplimentari
    - Algoritme mai bune de adaptare a ponderilor şi pragurilor.

Subantrenarea

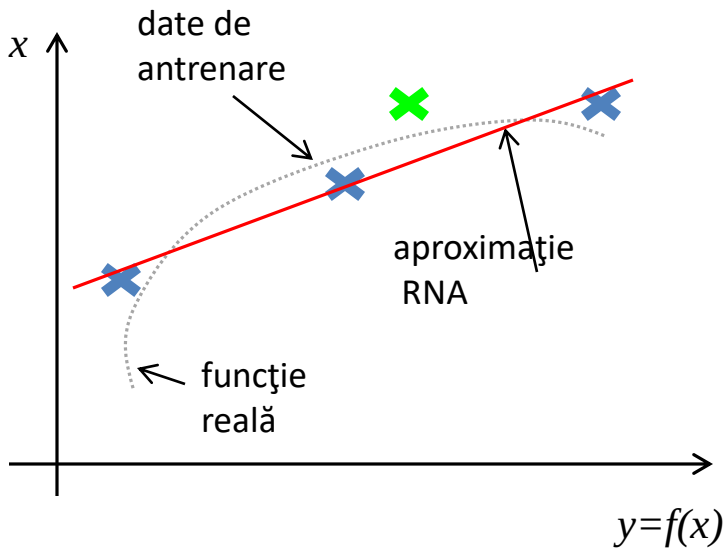
Subantrenare: funcţia cu care RNA aproximează ieşirea dorită este prea simplă

- ❖ RNA nu va generaliza corect pentru punctele de tipul ✖

– Curbă prea netedă faţă de necesar



– RNA are la dispoziţie prea puţine date, care nu reflectă variaţia reală a funcţiei

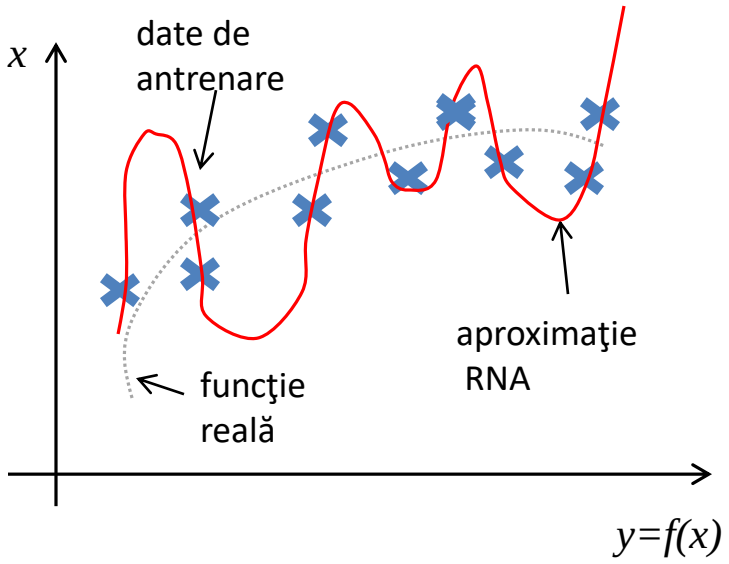


Supraantrenarea

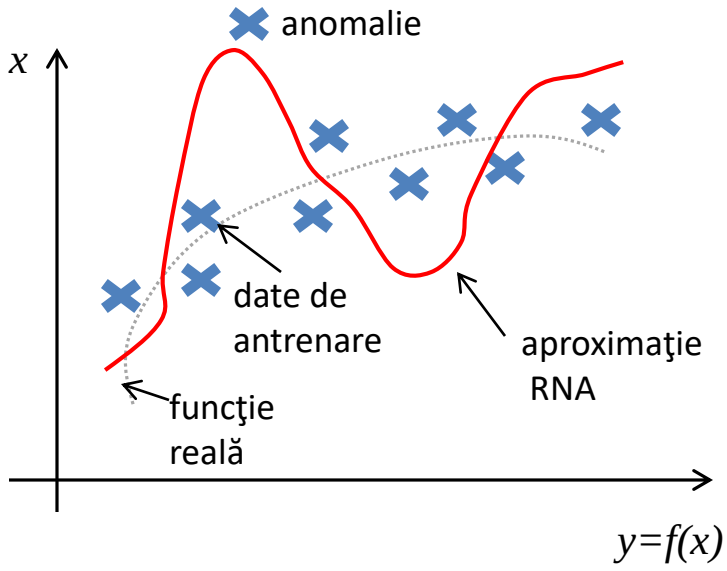
Subantrenare: funcţia cu care RNA aproximează ieşirea dorită este prea complexă.

- ❖ RNA nu va generaliza corect punctele neincluse în setul de antrenare

– Curbă prea întortocheată faţă de necesar



– Prezenţa unor anomalii în datele de intrare



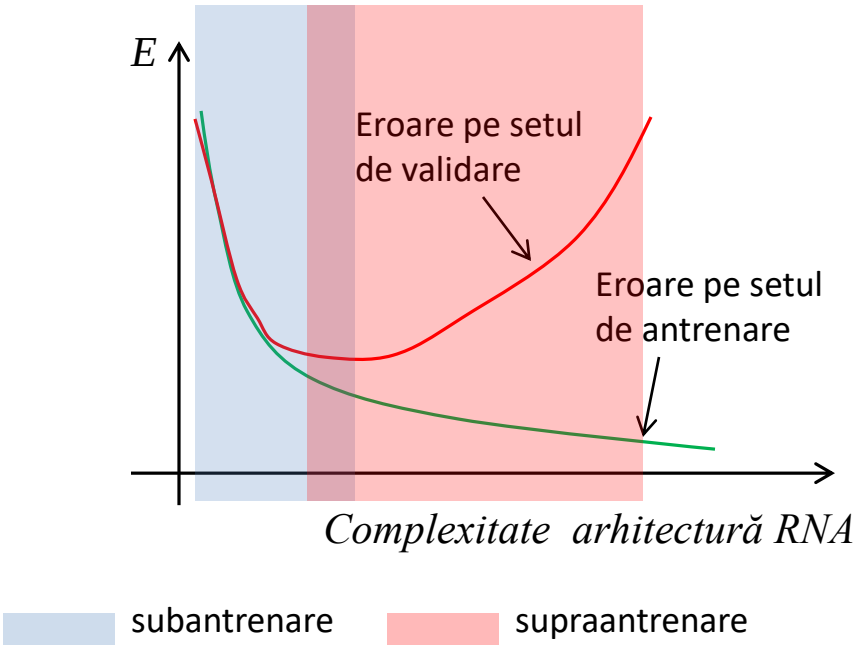
Soluții

- Pentru evitarea sau prevenirea subantrenării:
  - Antrenarea unor RNA cu arhitectură complexă (mulţi neuroni ascunşi, multe straturi ascunse) - permite aproximarea unor funcţii complexe
  - Folosirea unui număr mare de modele de antrenare – pentru a reprezenta mai fidel funcţia căutată
  - Creşterea numărului de caracteristici al unui model de intrare – creşte numărul de intrări al RNA şi numărul de conexiuni între neuroni (ponderi), permiţând definirea unor funcţii de aproximat complexe

- Pentru evitarea supraantrenării:
  - Antrenarea unor RNA cu arhitectură simplă (puţini neuroni ascunşi, un strat ascuns) - reduce complexitatea funcţiei ce poate fi aproximată
  - Eliminarea anomaliilor din datele de antrenare, prin analiză vizuală sau statistică.
  - Folosirea unor modele de intrare cu un număr mic de caracteristici - reduce numărul de ponderi.

Detectarea supraantrenării

- Datele disponibile pentru setul de antrenare se împart în trei:
  - Setul de antrenare (60%) – pentru aplicarea algoritmului de învăţare care realizează adaptarea ponderilor.
  - Setul de validare (20%) - pentru verificarea progresului antrenării
  - Setul test (20%) – pentru verificarea generalizării

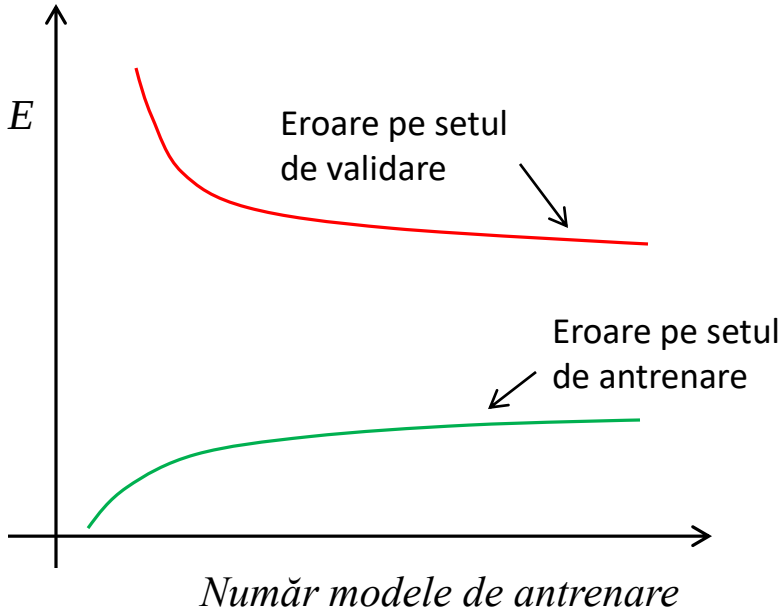


- Eroarea pe setul de antrenare scade, deoarece pe aceste date se corectează ponderile. RNA aproximează cel mai bine aceste puncte.
- Eroarea pe setul de validare creşte, deoarece ponderile (aproximarea RNA) nu se potriveşte şi pe datele noi din setul de validare, deci generalizarea nu este asigurată
- Antrenarea se opreşte când eroarea pe setul de validare începe să crească.

Detectarea supraantrenării

- Câte modele de antrenare să folosesc ? Când sunt prea multe ?

- Supraantrenare:



- Eroarea pe setul de antrenare creşte, deoarece creşte numărul de modele şi erorile se cumulează.

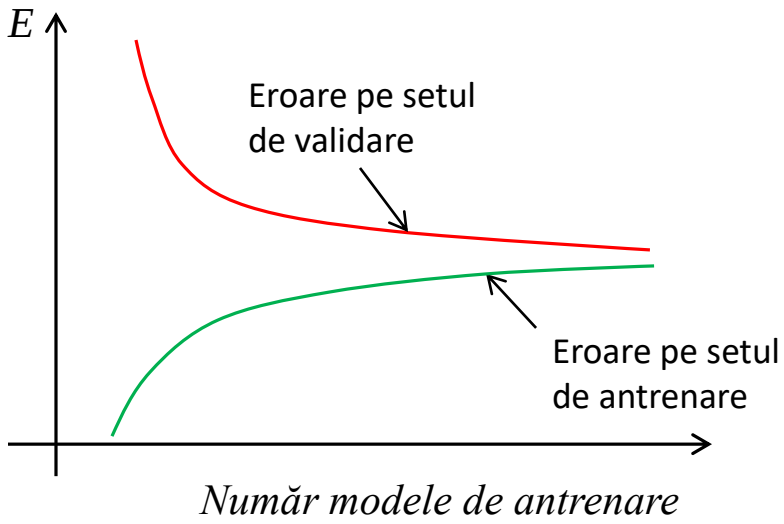
- Eroarea pe setul de validare scade, dar nu se apropie de cea obţinută pe setul de antrenare, deoarece funcţia de aproximare a RNA nu se potriveşte şi pentru modelele din setul de validare .

[Andrew NG, scikits]

Detectarea subantrenării

- Câte modele de antrenare să folosesc ? Când sunt prea puţine ?

- Subantrenare:



- Eroarea pe setul de antrenare creşte, deoarece creşte numărul de modele şi erorile se cumulează

- Eroarea pe setul de validare scade, deoarece aproximarea RNA e mai bună cu mai multe modele , şi se apropie de cea a setului de antrenare

- Curbele se întâlnesc la o valoare mare a erorii

[Andrew NG, scikits]

Influența ratei de învățare asupra antrenării

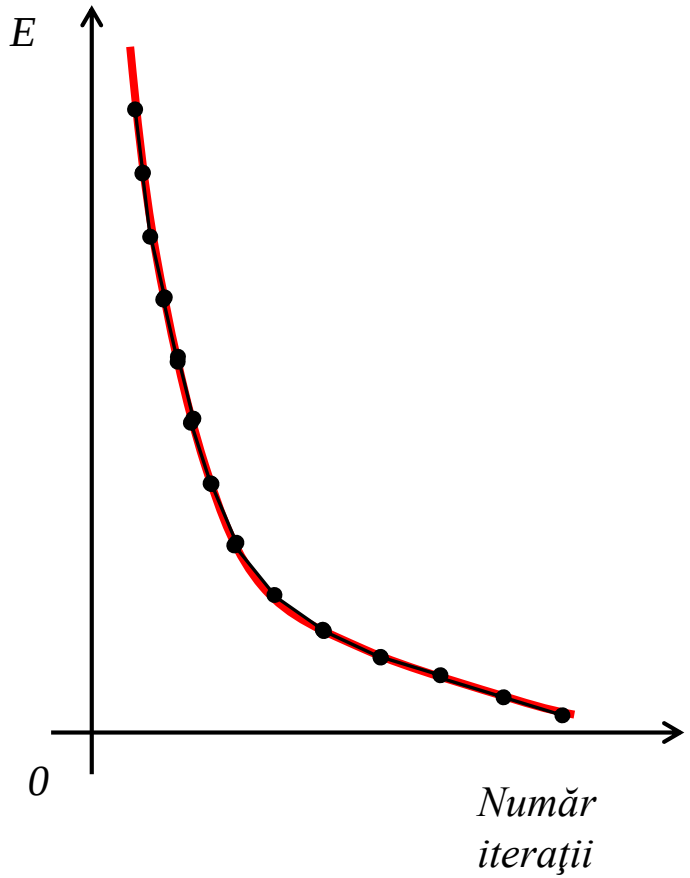
$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \cdot \frac{\partial E}{\partial w_{ij}^{(t)}}\bigg|_{w=w_{ij}^{(t)}}, \quad i = 1...I,\ j = 1...J$$

- O rată de învăţare prea mică înseamnă corecţii mici pentru ponderi
  - Antrenarea poate dura mult sau se poate bloca într-o “ravenă” sau pe un “platou” local.

- O rată de învăţare prea mare înseamnă corecţii mai pentru ponderi, care pot cauza ratarea minimului global şi, în caz extrem, divergenţă.
  - Antrenarea nu găseşte minimul global, ceea ce probabil va influenţa negativ şi generalizarea

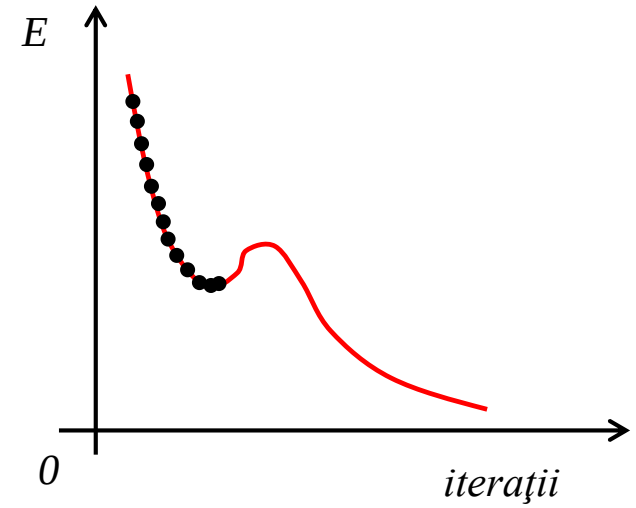
- Valori recomandate: η=0.1 … 0.2
- Nu există o reţetă garantată de succes, trebuie făcute teste

Rată de învățare prea mică = învățare lentă

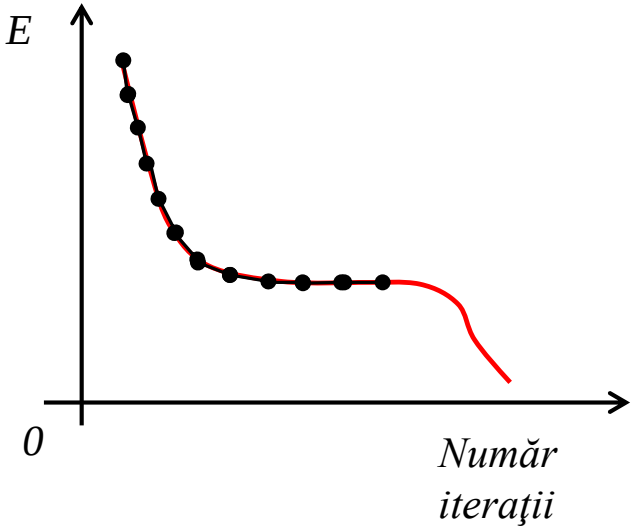


- Minimul global al erorii se atinge după multe iteraţii
- Caz fericit, când suprafaţa erorii e convexă
- Se poate ajunge la zeci de mii, chiar sute de mii de iteraţii

Rată de învățare prea mică = blocaj

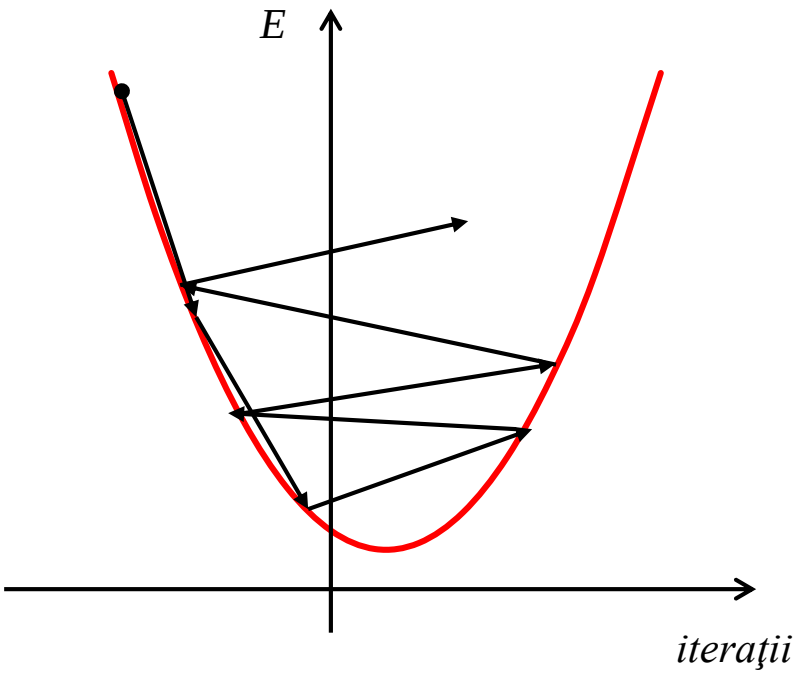


- Gradientul nu e suficient de mare pentru a ieşi din “ravenă”.



- Dacă criteriul de oprire al antrenării e un prag de variaţie a erorii pe parcursul unui număr mic de iteraţii succesive, antrenarea se poate opri înainte de ieşirea din “platou”.

Rată de învățare prea mare = divergență



- Când gradientul este prea mare, corecţia erorii este prea mare
- Antrenarea poate intra în divergenţă

Antrenarea cu rate de învățare adaptive

- Reducerea progresivă a ratei de învăţare de la o iteraţie la alta:

$$\eta^{(t+1)} = p \cdot \eta^{(t)},\ p = [0.1...1]$$

  - În prima parte a antrenării, rata de învăţare mare permite apropierea mai rapidă de minimul global, sărind peste “ravenele” mici şi minimele globale
  - La apropierea de minimul global, converge către acesta, reducând riscul depăşirii lui.

- Cu cât corecţia este mai mare (p este mai mic), cu atât mă apropii de cazul η prea mic.
- Cu cât corecţia este mai mică (p este mai mare), cu atât mă apropii de cazul η prea mare.

Antrenarea cu rate de învățare adaptive

- Viteza şi acurateţea învăţării pot fi îmbunătăţite dacă fiecărei ponderi i se ataşează o rată de învăţare distinctă, adaptată diferit, după următorul principiu [Hinton]:
  - Valoarea iniţială a ponderilor este 1
  - De la o iteraţie la alta, se calculează variaţia (gradientul) ponderii
  - Dacă, în două iteraţii succesive, gradientul îşi păstrează semnul (eroarea scade şi apoi iar scade ori creşte şi iar creşte), rata de învăţare este amplificată – sunt pe drumul cel bun.
  - Dacă, în două iteraţii succesive, gradientul îşi schimbă semnul (eroarea scade, apoi creşte ori creşte, apoi scade, adică oscilează), rata de învăţare este diminuată – sunt pe drumul greşit.

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \cdot corr_{ij} \frac{\partial E}{\partial w_{ij}^{(t)}}$$

$$test = \left(\frac{\partial E}{\partial w_{ij}} \right)^{(t+1)} \cdot \left(\frac{\partial E}{\partial w_{ij}} \right)^{(t)}$$

$$test > 0, \quad corr_{ij}^{(t+1)} = corr_{ij}^{(t)} + 0.05$$

$$test <= 0, \quad corr_{ij}^{(t+1)} = corr_{ij}^{(t)} \cdot 0.95$$

$$corr = [0..100]$$

Inițializarea ponderilor

- Ponderile nu pot fi inițializate cu 0, deoarece toate ieşirile calculate ar fi 0, deci eroarea şi corecţiile ar fi aceleaşi la fiecare iteraţie, pentru toate ponderile.

- Există numeroase metode dezvoltate pe bază empirică, care au dat rezultate bune:
  - Iniţializare aleatorie cu valori mici, în intervalul (-1, 1)
  - Regula Russo:

$$-\frac{2.4}{I} \leq w_{ij} \leq \frac{2.4}{I}$$

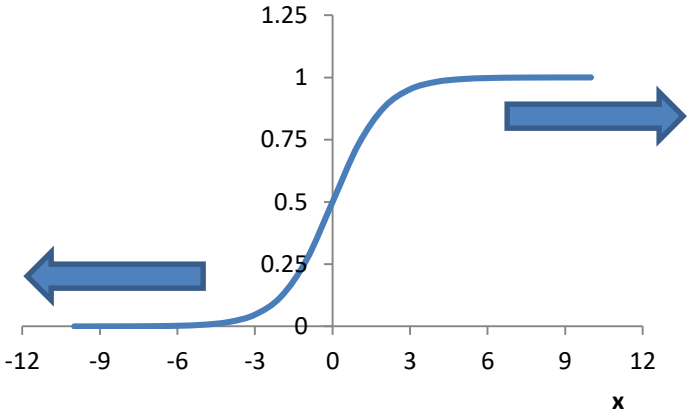
I - numărul de intrări ale RNA

  - …şi altele

Optimizarea antrenării

- Pe parcursul antrenării, prin adăugarea succesivă a corecţiilor Δw, ponderile tind să aibă valori foarte mari (pozitive) sau foarte mici (negative).
- Dacă se foloseşte funcţia de activare sigmoid logistic, valori depărtate de 0 ale intrării nete a neuronului saturează ieşirea sigmoidului către 0 sau 1, ceea ce scade performanţele antrenării

- Este de dorit menţinerea intrării nete în zona de variaţie amplă a sigmoidului, pentru diferenţierea fină în cadrul claselor identificate.
  - Se pot modifica ponderile w
  - Se pot modifica intrările x



$$net = w_1 \cdot x_1 + \ldots + w_i \cdot x_i + \ldots + w_n \cdot x_n \qquad y = \frac{1}{1+e^{-net}}$$

Optimizarea valorii ponderilor în timpul antrenării

- Deprecierea ponderilor (weight decay, regularization):

$$E = E + pen = \frac{1}{2} \cdot \sum_{m=1}^{M} \left(d^{(m)} - o^{(m)} \right)^2 + \frac{1}{2} \cdot \lambda \cdot \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} w^2$$

$$\Delta w = -\eta \cdot \left(\frac{\partial E}{\partial w} \right) - \lambda \cdot w$$

- Efect: ponderile care trebuie să scadă devin şi mai mici dacă eroarea E nu este mare, pentru că se scade din ele şi valoarea $\lambda \cdot w$. Ponderile cu valoare absolută mică (inutile, nu afectează semnificativ antrenarea) tind spre 0.
- Coeficientul λ se alege de către utilizator.

- Altă variantă: reducerea ponderilor cu o valoare scalară.

Optimizarea valorii ponderilor în timpul antrenării

- Folosirea unui termen de moment

$$w_i^{(t+1)} = w_i^{(t)} - \eta \cdot \frac{\partial E}{\partial w_i^{(t)}}\bigg|_{w=w_i^{(t)}} + \mu \cdot (w_i^{(t)} - w_i^{(t-1)}), \quad i = 0...n$$

- Efect: Corecţia ponderii ţine cont de evoluţia erorii în iteraţiile anterioare (inerţie), nu se modifică concomitent cu variaţia suprafeţei erorii, ci cu o anumită întârziere, permiţând trecerea mai rapidă peste minime locale şi “ravine”.
  - La începutul antrenării, eroarea scade mai rapid către minimul global
  - Către sfârşitul antrenării, se reduc oscilaţiile în apropierea minimului.
- Valoarea coeficientului μ (subunitară, 0.1…0.9) depinde de problemă şi se determină prin încercări.

Preprocesarea datelor de intrare

- Când se folosesc funcţii de activare care dau valori într-un interval limitat (de ex. sigmoid logistic între 0 şi 1, tangent hiperbolic între -1 şi 1), este benefică scalarea datelor de antrenare în acelaşi interval (de ex, pentru sigmoid logistic, scalarea între 0.15 şi 0.85, pentru a menţine antrenarea în zona de variaţie amplă a sigmoidului).

- În absenţa scalării, componentele $w_i \cdot x_i$ în care x are valori mai mari (de exemplu $x_i=3500$ vs $x_j=1.2$) influenţează mai mult ieşirea neuronului.

- Scalarea se face pe coloane (elemente x_i ale unui model).

Preprocesarea datelor de intrare

- Pentru un vector de date (o coloană reprezentând valoarea unui parametru x_m dintr-un set de antrenare cu M modele):

$$X = (x_1, ..., x_m, ..., x_M)^T$$

  - Scalarea min-max (între o valoare minimă şi maximă):

$$x_m^{scalat} = \frac{x_m^{nescalat} - x_{\min}^{nescalat}}{x_{\max}^{nescalat} - x_{\min}^{nescalat}} \cdot (\lim_{scalare}^{\max} - \lim_{scalare}^{\min}) + \lim_{scalare}^{\min}$$

  - Scalarea în intervalul [0,1]:

$$x_m^{scalat} = \frac{x_m^{nescalat} - x_{\min}^{nescalat}}{x_{\max}^{nescalat} - x_{\min}^{nescalat}}$$

Preprocesarea datelor de intrare

- Scalarea prin aducerea valorilor la medie 0 (centrarea pe medie)

$$x_m^{scalat} = x_m^{nescalat} - \bar{x} = x_m^{nescalat} - \frac{1}{M}\sum_{m=1}^{M} x_m^{nescalat}$$

- Standardizarea (scalarea după varianţă) transformarea astfel încât media să fie 0 şi deviaţia standard să fie 1 (transformarea z):

  - Deviaţia standard (standard deviation): rădăcina pătrată din varianţă (media pătratului diferenţelor faţă de medie): variaţia datelor faţă de medie.

$$x_m^{scalat} = \frac{x_m^{nescalat} - \bar{x}}{std} \qquad \text{unde} \qquad std = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(x_m^{nescalat} - \bar{x}\right)^2}$$

  - Caz particular - distribuţia (clopotul) Gauss, caracteristică multor serii întâlnite în practică – cu număr egal de valori de o parte şi de alta a mediei.
    - Mediile şcolare.

Exemplu de scalare

- Scalarea după varianţă:

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

Suprafaţă locuinţă [mp]	Venit lunar [lei]	Putere instalată [kW]	Consum mediu [kWh/lună]
x_1	x_2	x_3	y
0.461	-0.572	-0.361	0.608
-0.909	-0.732	-0.755	-0.779
-0.749	0.787	0.624	-0.566
0.959	0.130	1.609	0.953
-0.989	-1.115	-1.149	-1.285
1.226	1.502	0.033	1.069

Preprocesarea datelor de intrare

- Analiza statistică a setului de antrenare şi:
  - ❖ Eliminarea anomaliilor (vezi supraantrenare)
  - ❖ Eliminarea parametrilor (coloanelor din matrice) corelaţi
    - ○ Corelare – tendinţă aproximativ sincronă a valorilor

Îmbunătățirea convergenței – algoritmul RProp

- RProp corectează ponderile în funcţie de semnul erorii în două iteraţii succesive, astfel:

$$w_{ij}^{t+1} = w_{ij}^{t} + \Delta w_{ij}^{t+1}$$

unde

$$\Delta w_{ij}^{t+1} = \begin{cases} -sign\left[\frac{\partial E^{t+1}}{\partial w_{ij}}\right] \cdot \delta_{ij}^{t+1} & if \ \frac{\partial E^{t+1}}{\partial w_{ij}} \cdot \frac{\partial E^{t}}{\partial w_{ij}} \geq 0 \\ -sign\left[\Delta w_{ij}^{t}\right] \cdot \delta_{ij}^{t+1} & if \ \frac{\partial E^{t+1}}{\partial w_{ij}} \cdot \frac{\partial E^{t}}{\partial w_{ij}} < 0 \end{cases}$$

$$\delta_{ij}^{t+1} = \begin{cases} \eta^{+} \cdot \delta_{ij}^{t} & if \ \frac{\partial E^{t+1}}{\partial w_{ij}} \cdot \frac{\partial E^{t}}{\partial w_{ij}} > 0 \\ \eta^{-} \cdot \delta_{ij}^{t} & if \ \frac{\partial E^{t+1}}{\partial w_{ij}} \cdot \frac{\partial E^{t}}{\partial w_{ij}} < 0 \\ \delta_{ij}^{t} & if \ \frac{\partial E^{t+1}}{\partial w_{ij}} \cdot \frac{\partial E^{t}}{\partial w_{ij}} = 0 \end{cases}$$

sign = semn(+ sau -)

Îmbunătățirea convergenței – algoritmul RProp

- Dacă eroarea nu schimbă semnul (continuă să scadă), δ este amplificat cu un coeficient supraunitar η^+ şi corecţia ponderii este accelerată.
- Dacă eroarea schimbă semnul (oscilaţie, minim depăşit) corecţia erorii este diminuată cu coeficientul η^- pentru a se reveni în minim.
- Dacă una din derivate se anulează (am atins minimul, panta derivatei este 0), δ rămâne neschimbat.

- Valori recomandate: $\eta^+=1.2$, $\eta^-=0.5$, $\delta_{init}=0.5$.

va urma...

- Reţele neuronale artificiale pentru clasificare
  - RNA Kohonen