Curs 6:

Clasificarea datelor (III)

Structura

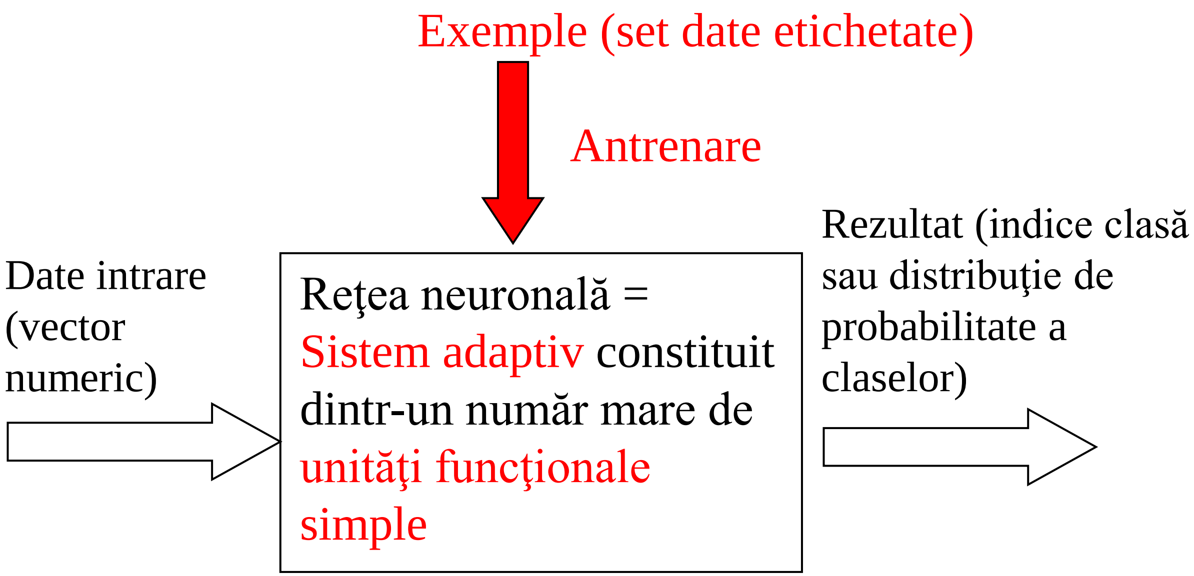
- Clasificatori bazaţi pe reţele neuronale
  - Rețele neuronale feedforward (Multilayer Perceptrons)

- Clasificatori bazaţi pe vectori suport (Support Vector Machines)

Rețele neuronale artificiale

Particularităţi:

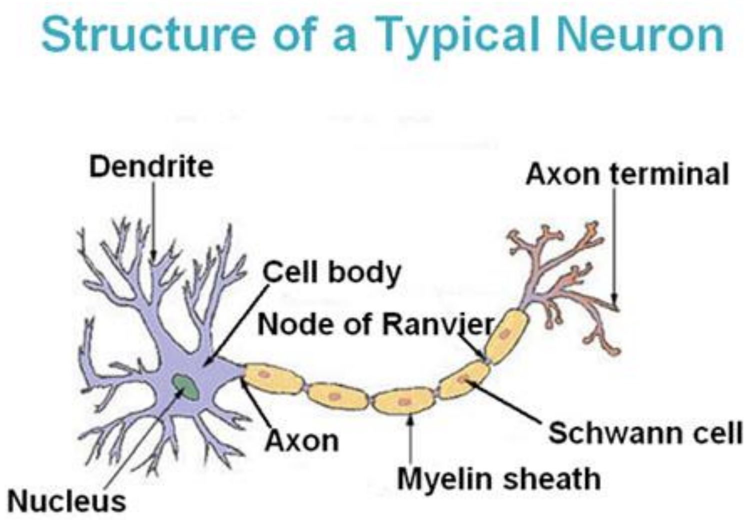
- Sunt clasificatori de tip black-box = permit predicţia clasei dar nu furnizează direct reguli explicite de clasificare (nu posedă modul explicativ)

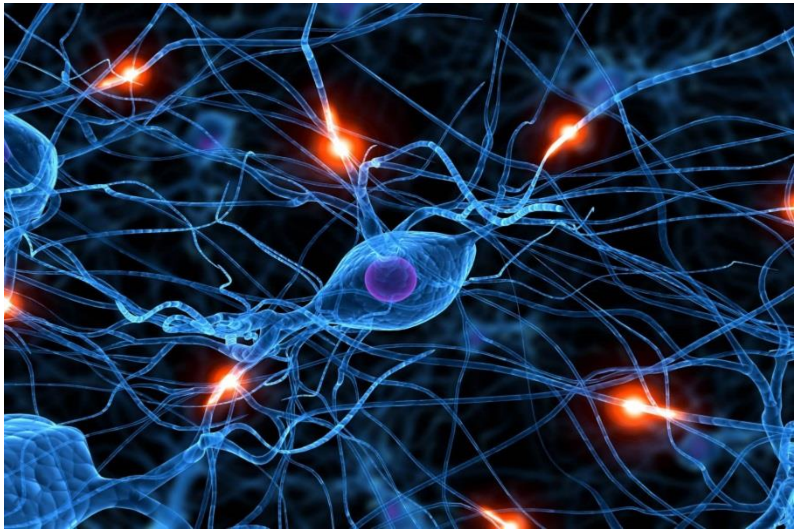


Rețele neuronale – modelul biologic

Particularităţi:

- Inspirate iniţial de structura şi funcţionarea creierului = sistem de neuroni interconectaţi
- Creier = cca 10^{10} neuroni și 10^{14} sinapse





Rețele neuronale artificiale

- RNA = set de neuroni artificiali (unităţi functionale) interconectaţi
  - Fiecare neuron primeşte mai multe semnale de intrare și produce un semnal de ieşire
  - RNA primeşte un vector de intrare (prin neuronii de intrare) şi produce un vector de ieşire (prin neuronii de ieşire)

- Aspecte principale ale unei RNA:
  - Arhitectura = graf orientat etichetat; fiecare arc are asociată o pondere numerică care modelează permeabilitatea sinaptică

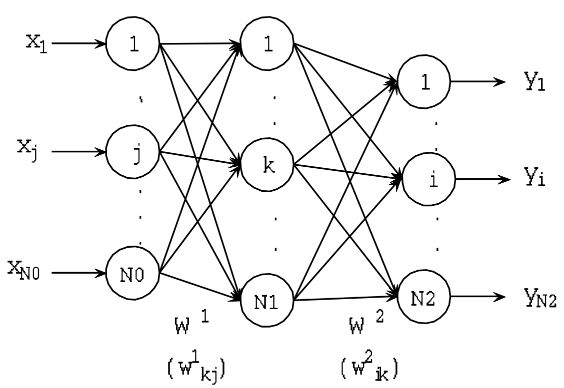
  - Funcţionare = procesul prin care RNA transformă un vector de intrare într-un vector de ieşire

  - Antrenare = procesul prin care sunt stabilite valorile ponderilor sinaptice şi ale altor parametri ai reţelei (de exemplu, praguri de activare – vezi slide-uri următoare)

Rețele neuronale artificiale

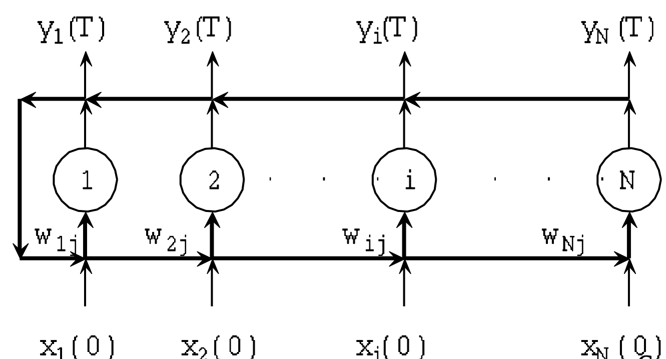
Principalele tipuri de arhitecturi:

- Unidirecțională (Feed-forward):
  - Graful suport nu conţine cicluri (neuronii sunt de obicei plasaţi pe mai multe nivele)
  - Semnalul de ieşite poate fi calculat prin compunerea unor funcţii de agregare şi de activare (transfer)
- Recurentă (Recurrent):
  - Graful suport conţine cicluri
  - Semnalul de ieşire este calculat prin simularea unui sistem dinamic (proces iterativ)



Feed-forward (multilayer perceptron)

RNA recurentă (reţea complet interconectată)



Rețele neuronale artificiale

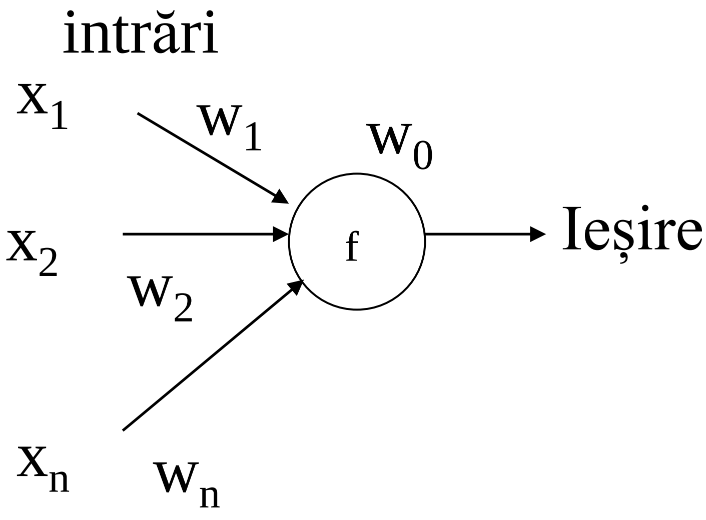
Proiectarea unei RNA:

- Alegerea arhitecturii: număr de nivele, număr de unităţi pe fiecare nivel, funcţii de activare, tip interconectare
- Antrenare: determinarea valorilor ponderilor folosind un set de antrenare şi un algoritm de învăţare
- Validare/testare: analiza comportamentului reţelei pentru exemple care nu fac parte din setul de antrenare

Obs:

- Pt o problemă de clasificare a unor date N-dimensionale în M clase reţeaua ar trebui să aibă:
  - N unităţi de intrare
  - M unităţi de ieşire
- Modelul de clasificare este încorporat în ponderile sinaptice (ponderile asociate conexiunilor dintre neuroni)

Rețele neuronale artificiale



Rețea neuronală artificială = ansamblu de unități simple de prelucrare (neuroni) interconectate

Unitate funcțională: mai multe intrări, o ieșire (model computațional simplificat al neuronului)

w_1, w_2, ...: Ponderi numerice atașate conexiunilor

Notații:

semnale de intrare: $x_1, x_2, \ldots, x_n$

ponderi sinaptice: $w_1, w_2, \ldots, w_n$

(modelează permeabilitatea sinaptică)

prag (bias): b (sau w_0)

(modelează pragul de activare al neuronului)

ieșire: y

$$y = f\left(\sum_{j=1}^{n} w_j x_j - w_0\right)$$

Funcție activare

prag

Obs: Toate valorile sunt numere reale

Unități funcționale

Generarea semnalului de ieșire:

- Se “combină” semnalele de intrare utilizând ponderile sinaptice și pragul de activare
  - Valoarea obținută modelează potențialul local al neuronului
  - Combinarea semnalelor de intrare în unitate se realizează printr-o funcție de agregare (integrare)
- Se generează semnalul de ieșire aplicand o funcție de activare (transfer)
  - corespunde generării impulsurilor de-a lungul axonului

Semnale de intrare ($y_1,\ldots,y_n$)

Funcție de agregare

Starea neuronului (u)

Funcție de activare

Semnal de ieșire (y)

Unități funcționale

Exemple de funcții clasice de agregare

Suma ponderată

$$u = \sum_{j=1}^{n} w_j \, x_j - w_0$$

$$u = \prod_{j=1}^{n} x_j^{w_j}$$

Neuron multiplicativ

Distanța euclidiană

$$u = \sqrt{\sum_{j=1}^{n} (w_j - x_j)^2}$$

$$u = \sum_{j=1}^{n} w_j \, x_j + \sum_{i,j=1}^{n} w_{ij} \, x_i x_j + \ldots$$

Conexiuni de ordin superior

Observatie: pentru varianta cu suma ponderată se poate asimila pragul cu o pondere sinaptică corespunzătoare unei intrări fictive (cu valoare -1) astfel că starea neuronului poate fi exprimată prin suma ponderată:

$$u = \sum_{j=0}^{n} w_j \, x_j$$

Unități funcționale

Exemple de funcții de activare (transfer)

$$f(u) = \operatorname{sgn}(u) = \begin{cases} -1 & u \leq 0 \\ 1 & u > 0 \end{cases}$$ signum

$$f(u) = H(u) = \begin{cases} 0 & u \leq 0 \\ 1 & u > 0 \end{cases}$$ Heaviside

$$f(u) = \begin{cases} -1 & u < -1 \\ u & -1 \leq u \leq 1 \\ 1 & u > 1 \end{cases}$$ rampă

$$f(u) = u$$ liniară

Unități funcționale

Exemple de funcții de activare (transfer)

$$f(u) = \max\{0, u\}$$

ReLU

Semi-liniară (rectified linear unit - ReLU)

$$f(u) = \begin{cases} a(\exp(u) - 1) & u \leq 0 \\ 1 & u > 0 \end{cases}$$

ELU

Obs: utilizate în rețelele cu structură adâncă (Deep NN)

$$softmax(y_i) = \frac{\exp(y_i)}{\exp(y_1)+\exp(y_2)+\cdots+\exp(y_M)}$$

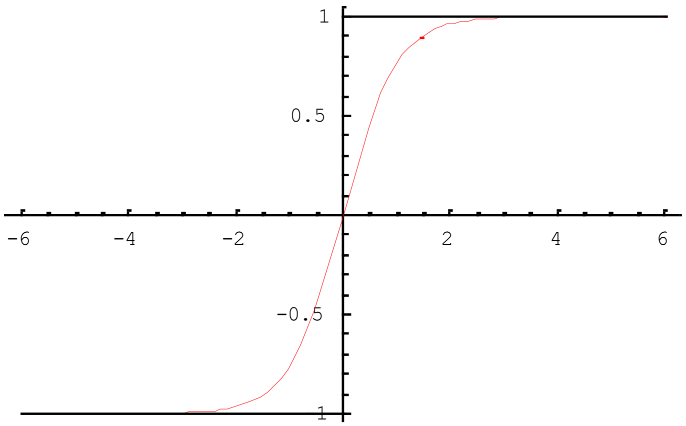
Softmax – se utilizează doar pentru nivelul de ieșire în cazul rețelelor neuronale utilizate pentru clasificare – vectorul de ieșire poate fi interpretat ca o distribuție de probabilitate (Obs: trebuie să se țină totuși cont de faptul că valorile de ieșire nu sunt neapărat calibrate)

Unități funcționale

Exemple de funcții de activare (funcții sigmoidale)

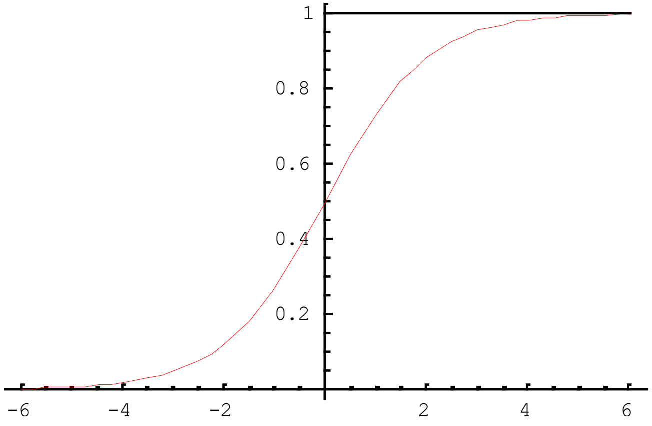
(tangenta hiperbolică)

$$f(u) = \tanh(u) = \frac{\exp(2u) - 1}{\exp(2u) + 1}$$



$$f(u) = \frac{1}{1 + \exp(-u)}$$

(logistică)



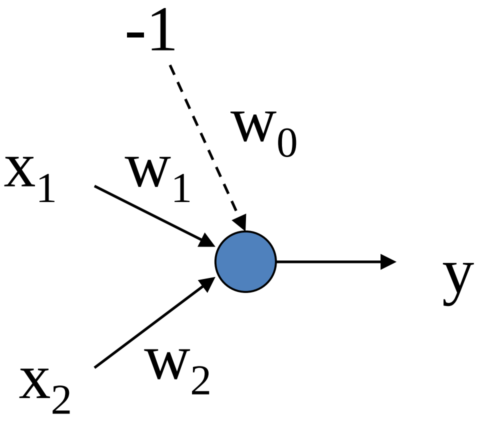
Observație: uneori se folosește un parametru numit pantă (slope) care multiplică argumentul funcției de activare: y=f(p*u)

Unități funcționale

- Ce se poate face cu un singur neuron ?

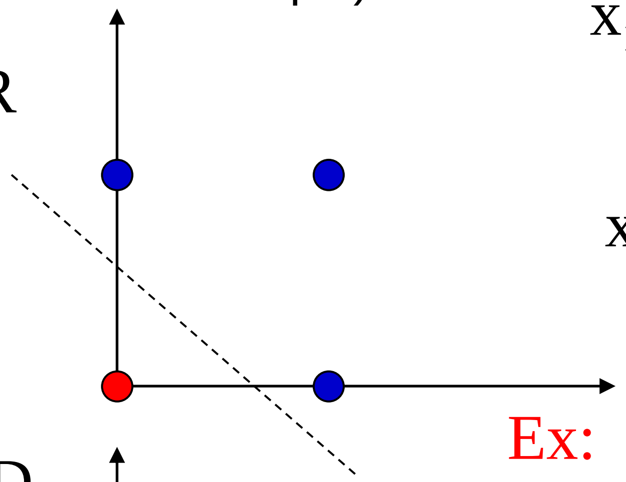
Se pot rezolva probleme simple de clasificare

(ex: se pot reprezenta funcții booleene simple)



OR

	0	1
0	0	1
1	1	1

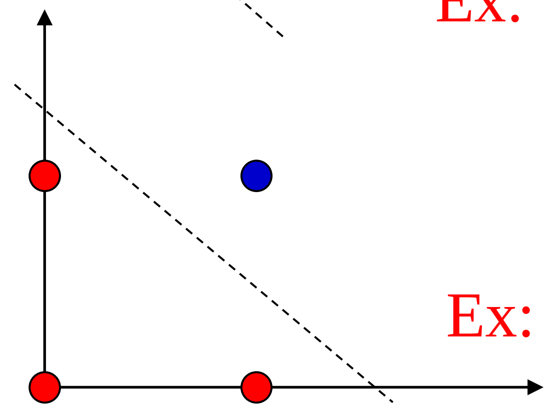


$$y=H(w_1x_1+w_2x_2-w_0)$$

Ex: $w_1=w_2=1,\ w_0=0.5$

AND

	0	1
0	0	0
1	0	1

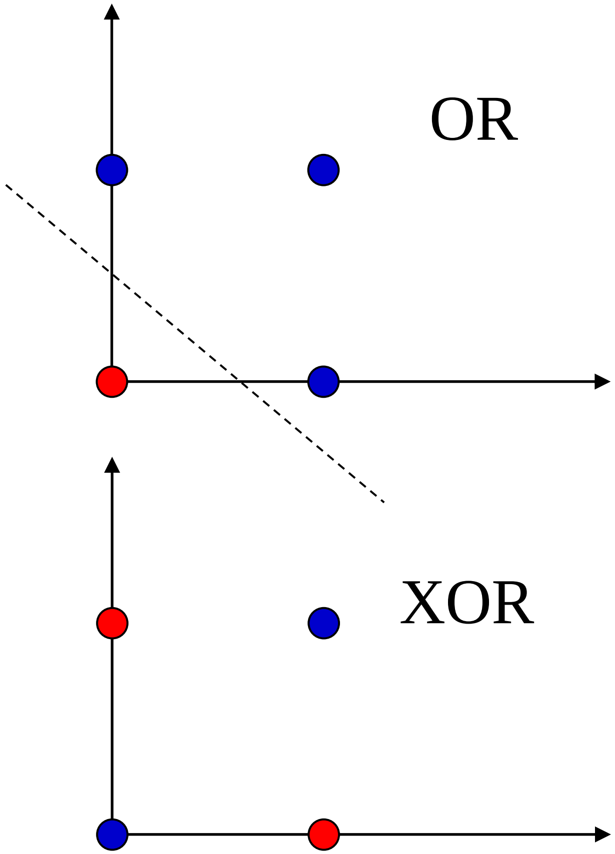


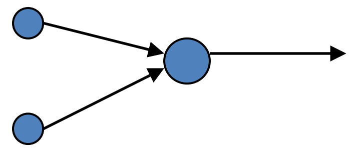
$$y=H(w_1x_1+w_2x_2-w_0)$$

Ex: $w_1=w_2=1,\ w_0=1.5$

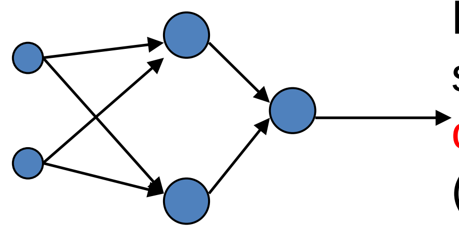
Liniar/neliniar separabilitate

Reprezentarea unor funcții booleene: $f:\{0,1\}^N \to \{0,1\}$





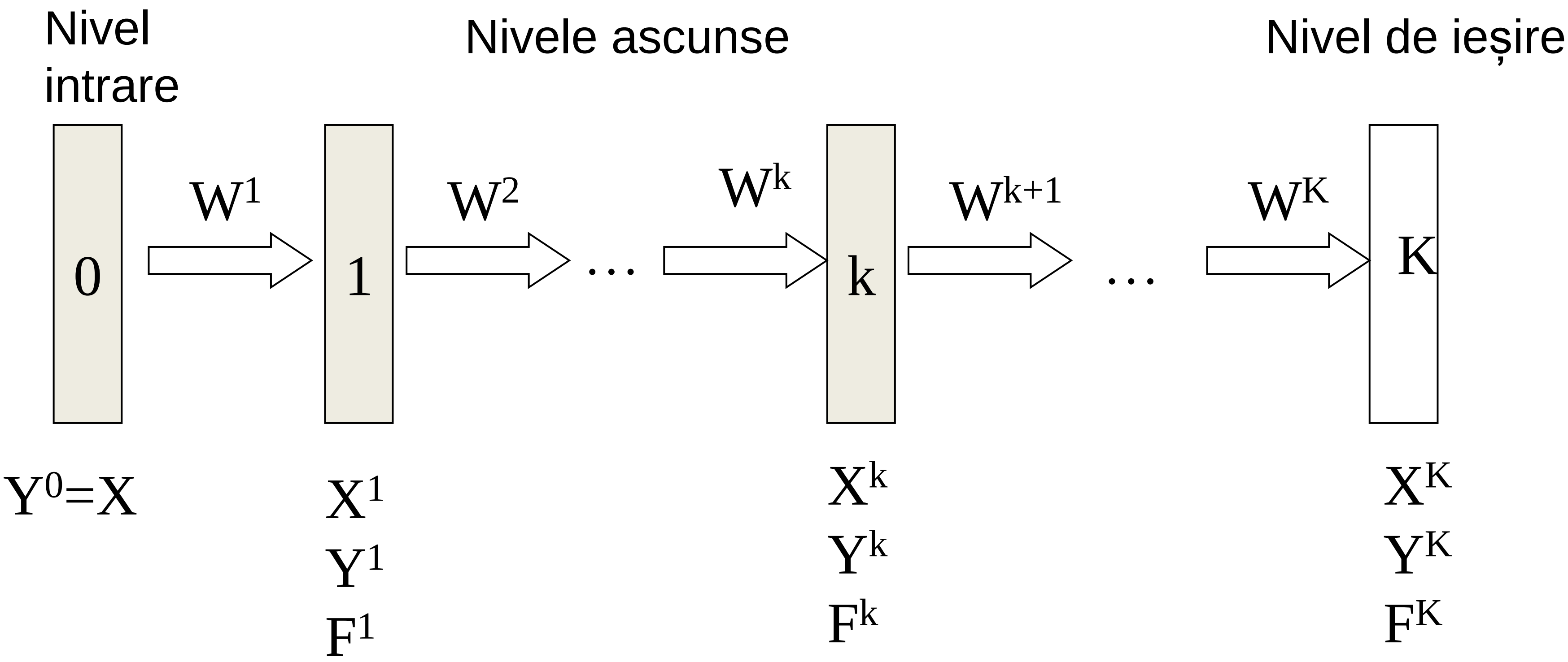
Problema liniar separabilă – e suficientă o rețea uninivel



Problema neliniar separabilă – e necesară o rețea multinivel (cel puţin un nivel ascuns – hidden layer)

Rețele feedforward - arhitectura

Arhitectura și funcționare (K nivele funcționale)

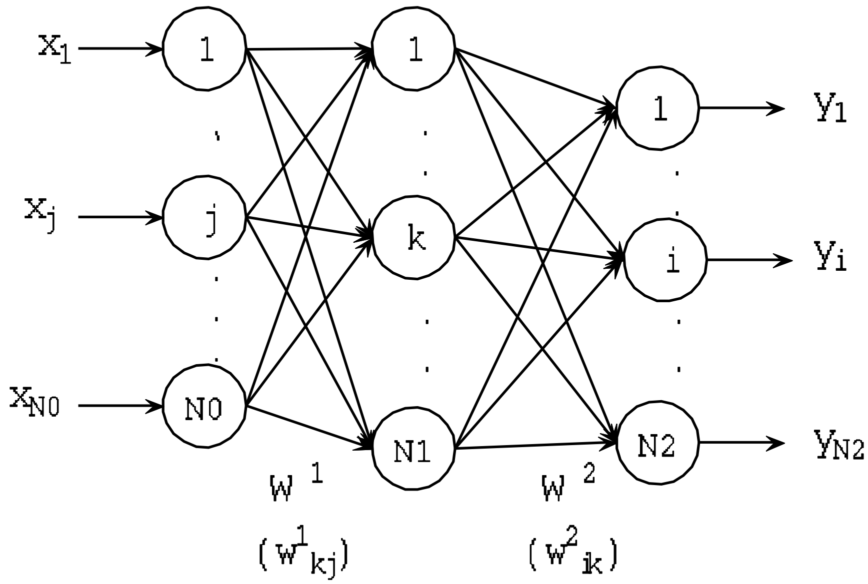


X = vector intrare, Y= vector ieșire, F=funcție vectorială de activare

Calcul vector de ieșire: $Y=F^K(W^K*F^{K-1}(W^{K-1}*F^{K-2}(.....F^1(W^1*X))))$

Rețele feedforward – funcționare

Arhitectura și funcționare
(caz particular: un nivel ascuns)



Parametrii modelului: matricile cu ponderi W^1 si W^2 (setul tuturor ponderilor e notat cu W)

$$y_i = f_2\left(\sum_{k=0}^{N_1} w^{(2)}{}_{ik} f_1\left(\sum_{j=0}^{N_0} w^{(1)}{}_{kj} x_j\right)\right), \qquad i = 1..N_2$$

Obs:

- în mod tradițional se lucrează cu unul sau două nivele ascunse
- rețelele cu număr mare de nivele sau cu structură adâncă (Deep Neural Networks) sunt folosite frecvent în particular pentru recunoașterea imaginilor și a vorbirii (https://www.deeplearningbook.org/)

Rețele feedforward - antrenare

Antrenare (supervizată):

- Set de antrenare: $\{(x^1,d^1), \ldots, (x^L,d^L)\}$

  (x^l = vector intrare, d^l = vector de ieșire corect)

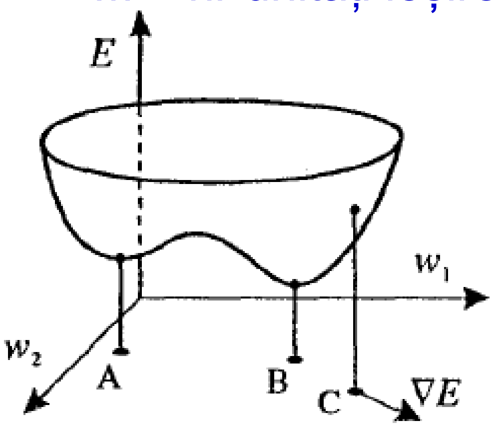
- Funcție de eroare (suma pătratelor erorilor):

$$E(W) = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{M}\left(d_i^l - f_2\left(\sum_{k=0}^{K} w_{ik} f_1\left(\sum_{j=0}^{N} w_{kj} x_j^l\right)\right)\right)^2$$

Notații:

- N = nr unități intrare
- K = nr unități ascunse
- M = nr unități ieșire

- Scopul antrenării: minimizarea funcției de eroare
- Metoda de minimizare: metoda gradientului (gradient descent)



Rețele feedforward - antrenare

Relația de ajustare (metoda gradientului):

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

Pas descreștere
=
Rata de învățare

Functia de eroare:

$$E(W) = \frac{1}{2}\sum_{l=1}^{L} \sum_{i=1}^{M} \left(d_i^l - f_2\left(\sum_{k=0}^{K} w_{ik} f_1\left(\sum_{j=0}^{N} w_{kj} x_j^l\right)\right)\right)^2$$

Notații:

$$\frac{1}{2}\sum_{l=1}^{L} \underbrace{\sum_{i=1}^{M} \left(d_i^l - \underbrace{f_2\left(\underbrace{\sum_{k=0}^{K} w_{ik} \underbrace{f_1\left(\underbrace{\sum_{j=0}^{N} w_{kj} x_j^l}_{x_k}\right)}_{y_k}}_{x_i}\right)}_{y_i}\right)^2}_{E_l(W)\ \text{(eroarea corespunzatoare exemplului } l\text{)}}$$

Rețele feedforward - antrenare

- Calculul derivatelor partiale

$$E(W)=\frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{M}\left(d_i^l-\underbrace{f_2\left(\underbrace{\sum_{k=0}^{K}w_{ik}\underbrace{f_1\left(\underbrace{\sum_{j=0}^{N}w_{kj}x_j^l}_{x_k}\right)}_{y_k}}_{x_i}\right)}_{y_i}\right)^2$$

$$\frac{\partial E_l(W)}{\partial w_{ik}}=-(d_i^l-y_i)f_2'(x_i)y_k=-\delta_i^l y_k$$

$$\frac{\partial E_l(W)}{\partial w_{kj}}=-\sum_{i=1}^{M}w_{ik}\,(d_i^l-y_i)f_2'(x_i)f_1'(x_k)x_j^l=-\left(f_1'(x_k)\sum_{i=1}^{M}w_{ik}\delta_i^l\right)x_j=-\delta_k^l x_j^l$$

Obs: δ_i reprezintă o măsură a erorii corespunzătoare unității de ieșire i iar δ_k reprezintă eroarea de la nivelul unității ascuns k (obținut prin propagarea înapoi in rețea a erorii de la nivelul de ieșire)

Rețele feedforward - antrenare

$$\frac{\partial E_l(W)}{\partial w_{ik}} = -(d_i^l - y_i) f_2'(x_i) y_k = -\delta_i^l y_k$$

$$\frac{\partial E_l(W)}{\partial w_{kj}} = -\sum_{i=1}^{M} w_{ik} (d_i^l - y_i) f_2'(x_i) f_1'(x_k) x_j^l = -\left(f_1'(x_k) \sum_{i=1}^{M} w_{ik} \delta_i^l \right) x_j = -\delta_k^l x_j^l$$

Obs: derivatele funcțiilor tradiționale de activare (logistica și tanh) pot fi calculate simplu folosind următoarele proprietăți:

Logistica: f'(x)=f(x)(1-f(x)) => f'(x)=y(1-y)

Tanh: f'(x)=1-f(x)^2 => f'(x)=1-y^2

ReLU: f'(x)=0 pt x<0, f'(x)=1 pt x>0

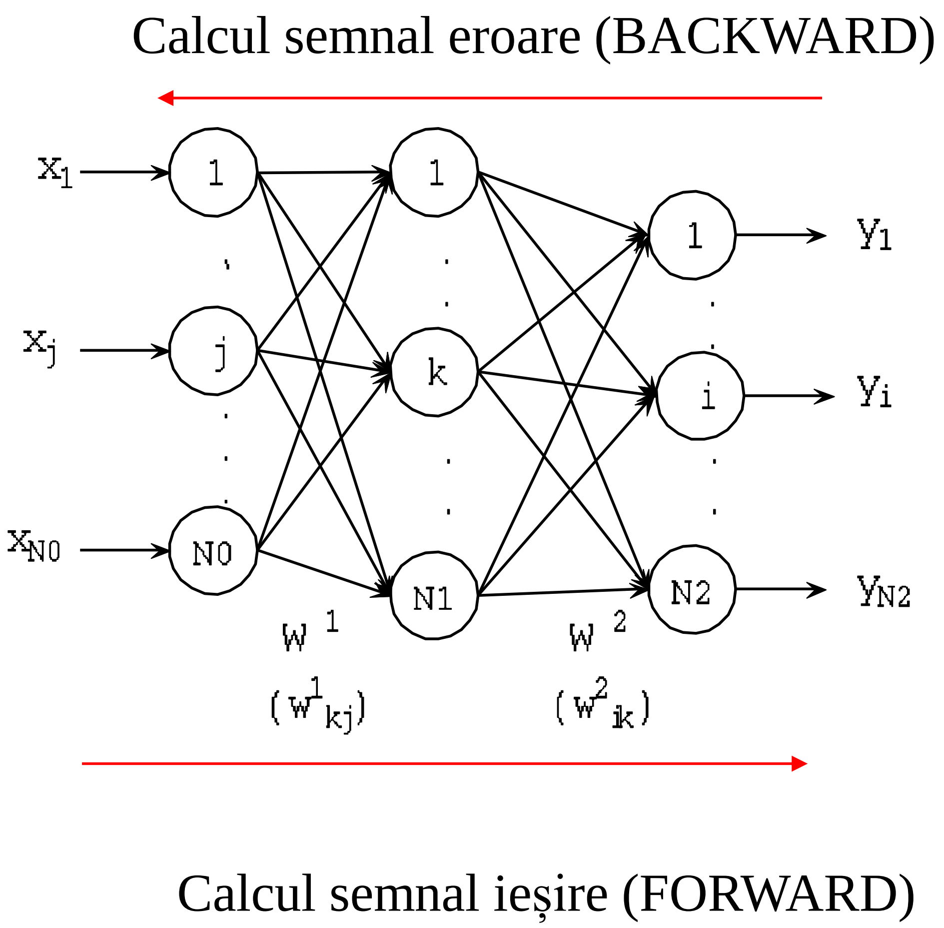
ELU: f'(x)=f(x)+a pt x<0, f'(x)=1 pt x>0

Algoritmul BackPropagation

Idee:

Pentru fiecare exemplu din setul de antrenare (sau din subset – minibatch):

- se determină semnalul de ieșire
- se calculează eroarea la nivelul de ieșire
- se propagă eroarea înapoi în rețea și se reține factorul delta corespunzător fiecărei ponderi
- se aplică ajustarea corespunzătoare fiecărei ponderi



Algoritmul BackPropagation

Inițializarea aleatoare a ponderilor
REPEAT
    FOR l=1,L DO
        etapa FORWARD
        etapa BACKWARD
        ajustare ponderi
    Recalcularea erorii
UNTIL <condiție oprire>

epoca

Obs.

- Valorile inițiale se aleg aleator in [0,1] sau [-1,1] (preferabil)
- La ajustare se ține cont de rata de învățare (parametrul eta)
- Recalcularea erorii presupune determinarea semnalului de ieșire pentru fiecare dată de intrare
- Condiția de oprire depinde de valoarea erorii și/sau numărul de epoci de antrenare

Algoritmul BackPropagation

Varianta serială

$$w_{kj}^{(1)} = rand(-1,1), w_{ik}^{(2)} = rand(-1,1)$$

$p = 0$

REPEAT

FOR $l = 1, L$ DO

/* Etapa FORWARD */

$$x_k^l = \sum_{j=0}^{N} w_{kj}^{(1)} x_j^l \, , y_k^l = f_1(x_k^l), x_i^l = \sum_{k=0}^{K} w_{ik}^{(2)} y_k^l \, , y_i^l = f_2(x_i^l)$$

/* Etapa BACKWARD */

$$\delta_i^l = f_2'(x_i^l)(d_i^l - y_i^l), \delta_k^l = f_1'(x_k^l) \sum_{i=1}^{M} w_{ik}^2 \delta_i^l$$

/* Etapa de ajustare */

$$w_{kj}^{(1)} = w_{kj}^{(1)} + \eta \delta_k^l x_j^l, \qquad w_{ik}^{(2)} = w_{ik}^{(2)} + \eta \delta_i^l y_k^l$$

ENDFOR

Obs. varianta “stochastic gradient descent” se caracterizează prin selectia, aleatoare, la fiecare epocă, a unui subset din setul de antrenare și parcurgerea acestuia

Algoritmul BackPropagation

/* Calculul erorii */

$E = 0$

FOR $l = 1, L$ DO

/* Etapa FORWARD (cu noile valori ale ponderilor)*/

$$x_k^l = \sum_{j=0}^{N} w_{kj}^{(1)} x_j^l \,, y_k^l = f_1(x_k^l), x_i^l = \sum_{k=0}^{K} w_{ik}^{(2)} y_k^l \,, y_i^l = f_2(x_i^l)$$

/* Sumarea erorii */

$$E = E + \sum_{l=1}^{L} (d_i^l - y_i^l)^2$$

ENDFOR

$E = E/(2L)$

$p = p + 1$

UNTIL $p > p_{\max}$ OR E<E*

E* reprezintă toleranța la erori a rețelei

p_{max} reprezintă numărul maxim de epoci de antrenare

Algoritmul BackPropagation

Varianta pe blocuri (se bazează pe cumularea ajustarilor) – batch variant

$$w_{kj}^{(1)} = rand(-1,1), w_{ik}^{(2)} = rand(-1,1), \qquad i = 1..M, k = 0..K, j = 0..N$$

$p = 0$

REPEAT

$$\boxed{\Delta_{kj}^{1} = 0, \Delta_{ik}^{2} = 0}$$

FOR $l = 1, L$ DO

/* Etapa FORWARD */

$$x_k^l = \sum_{j=0}^{N} w_{kj}^{(1)} x_j^l, y_k^l = f_1(x_k^l), x_i^l = \sum_{k=0}^{K} w_{ik}^{(2)} y_k^l, y_i^l = f_2(x_i^l)$$

/* Etapa BACKWARD */

$$\delta_i^l = f_2'(x_i^l)(d_i^l - y_i^l), \delta_k^l = f_1'(x_k^l) \sum_{i=1}^{M} w_{ik}^{(2)} \delta_i^l$$

/* Etapa de ajustare */

$$\boxed{\Delta_{kj}^{1} = \Delta_{kj}^{1} + \eta \delta_k^l x_j^l, \qquad \Delta_{ik}^{2} = \Delta_{ik}^{2} + \eta \delta_i^l y_k^l}$$

ENDFOR

$$\boxed{w_{kj}^{(1)} = w_{kj}^{(1)} + \Delta_{kj}^{1}, \qquad w_{ik}^{(2)} = w_{ik}^{(2)} + \Delta_{ik}^{2}}$$

Algoritmul BackPropagation

/∗ Calculul erorii ∗/

$E = 0$

FOR $l = 1, L$ DO

/∗ Etapa FORWARD (cu noile valori ale ponderilor)∗/

$$x_k^l = \sum_{j=0}^{N} w_{kj}^{(1)} x_j^l \,, y_k^l = f_1\left(x_k^l\right), \quad x_i^l = \sum_{k=0}^{K} w_{ik}^{(2)} y_k^l \,, \quad y_i^l = f_2(x_i^l)$$

/∗ Sumarea erorii ∗/

$$E = E + \sum_{l=1}^{L} (d_i^l - y_i^l)^2$$

ENDFOR

$E = E/(2L)$

$p = p + 1$

UNTIL $p > p_{\max}$ OR E>E*

Variante

Altă funcţie de eroare:

- MSE (eroarea medie pătratică) este mai potrivită pentru problemele de regresie

- In cazul problemelor de clasificare o variantă mai adecvată este entropia încrucişată (cross-entropy error)
- Caz particular: clasificare binară (un neuron de ieşire):
  - d_l aparţine lui {0,1} (0 corespunde clasei 0 şi 1 corespunde clasei 1)
  - y_l aparţine lui (0,1) (pentru a decide clasa este necesară utilizarea unei valori prag: daca y_l < prag atunci clasa este 0, altfel clasa este 1

$$CE(W) = -\sum_{l=1}^{L} (d_l \log y_l + (1 - d_l) \log(1 - y_l))$$

Obs: forma derivatelor parţiale se schimbă, deci şi termenii utilizaţi în ajustarea ponderilor – principiul general al propagării înapoi a erorii rămâne însă valabil;

Variante

Entropia incrucișată – caz particular: clasificare binară cu funcție de activare sigmoidală

Calcul derivate pentru CE

$$CE(W) = -\sum_{l=1}^{L}(d_l \log y_l + (1-d_l)\log(1-y_l))$$

Calcul factor de ajustare pentru ponderile conexiunilor dintre nivelul ascuns și neuronul de ieșire ($w_k^{(2)}$)

$$\delta_l = (\frac{d_l}{y_l} - \frac{1-d_l}{1-y_l})f_2'(x^{(2)}) = \frac{d_l(1-y_l) - y_l(1-d_l)}{y_l(1-y_l)} \cdot y_l(1-y_l)$$

$$= d_l(1-y_l) - y_l(1-d_l) = d_l - y_l$$

Regula de ajustare: $w_k^{(2)}(p+1) = w_k^{(2)}(p) + \eta \delta_l y_k^l$

Variante

Clasificare multiplă (M clase):

- se utilizează softmax pentru calculul valorii produse de către fiecare dintre unitățile de ieșire
- răspunsurile corecte sunt vectori binari cu M componente: (0,0,...,0,1,0,...,0) cu valoarea 1 plasată pe poziția m (corespunzătoare clasei corecte)

$$CE(W) = -\sum_{l=1}^{L}\sum_{i=1}^{M} d_i^l \log\left(y_i^l\right) = -\sum_{l=1}^{L} \log\left(y_{m(l)}^l\right)$$

Regula de ajustare pentru ponderea care conectează unitatea ascunsă k de unitatea de ieșire m(l):

$$w_{k,m(l)}^{(2)}(p+1) = w_{k,m(l)}^{(2)}(p) + \eta(1 - y_{m(l)}^l)y_k^l$$

$y_{m(l)}^l$ = ieșire produsă de unitatea m(l) aplicând softmax
y_k^l = ieșire produsă de unitatea k de pe nivelul ascuns

Probleme ale algoritmului Backpropagation

P1. Viteza mică de convergență (eroarea descrește prea încet)

P2. Oscilații (valoarea erorii oscilează în loc să descrească în mod continuu)

P3. Problema minimelor locale (procesul de învățare se blochează într-un minim local al funcției de eroare)

P4. Stagnare (procesul de învățare stagnează chiar dacă nu s-a ajuns într-un minim local)

P5. Supraantrenarea și capacitatea limitată de generalizare

Probleme ale algoritmului BP

P1-P2: Eroarea descrește prea încet sau oscilează în loc să descrească

Cauze:

- Valoare inadecvată a ratei de învățare (valori prea mici conduc la convergența lentă iar valori prea mari conduc la oscilații)

  Soluție: adaptarea ratei de învățare

- Metoda de minimizare are convergență lentă

  Soluții:

  - modificarea euristică a variantei standard (varianta cu moment)
  - utilizarea unei alte metode de minimizare (Newton, gradient conjugat)

Probleme ale algoritmului BP

- Rata adaptivă de învățare:
  - Dacă eroarea crește semnificativ atunci rata de învățare trebuie redusă (ajustările obținute pentru valoarea curentă a ratei sunt ignorate)
  - Daca eroarea descrește semnificativ atunci rata de învățare poate fi mărită (ajustările sunt acceptate)
  - In toate celelalte cazuri rata de învățare rămâne neschimbată

$$E(p) > (1+\gamma)E(p-1) \Rightarrow \eta(p) = a\eta(p-1), 0 < a < 1$$
$$E(p) < (1-\gamma)E(p-1) \Rightarrow \eta(p) = b\eta(p-1), 1 < b < 2$$
$$(1-\gamma)E(p-1) \leq E(p) \leq (1+\gamma)E(p-1) \Rightarrow \eta(p) = \eta(p-1)$$

Exemplu: γ=0.05

Probleme ale algoritmului BP

- Varianta cu “moment” (termen de inerție):
  - Se introduce o “inerție” în calculul ponderilor:
    - termenul de ajustare a ponderilor de la epoca curentă se calculează pe baza semnalului de eroare precum și a ajustărilor de la epoca anterioară
  - Acționează ca o adaptare a ratei de învățare: ajustările sunt mai mari în porțiunile plate ale funcției de eroare și mai mici în cele abrupte
  - Se combină cu varianta pe blocuri (batch)

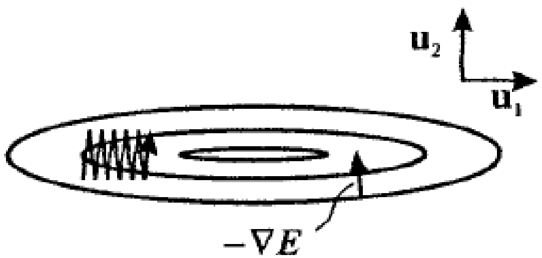
$$\Delta w_{ij}(p+1) = \eta \delta_i y_j + \alpha \Delta w_{ij}(p)$$

Exemplu de elegere a coeficientului termenului moment:

$\alpha = 0.9$

Probleme ale algoritmului BP

- Varianta cu “moment” (termen de inerție):
  - Se introduce o “inerție” în calculul ponderilor:
    - termenul de ajustare a ponderilor de la epoca curentă se calculează pe baza semnalului de eroare precum și a ajustărilor de la epoca anterioară



Metoda clasică

Utilizarea unui termen de inerţie

Probleme ale algoritmului BP

Alte metode de minimizare (mai rapide însă mai complexe):

- Metoda gradientului conjugat (și variante ale ei)
- Metoda lui Newton (caz particular: Levenberg Marquardt)

Particularități ale acestor metode:

- Convergența rapidă (ex: metoda gradientului conjugat converge în n iterații pentru funcții pătratice cu n variabile)

- Necesită calculul matricii hessiene (matrice conținând derivatele de ordin doi ale funcției de eroare) și uneori a inversei acesteia

Probleme ale algoritmului BP

- Exemplu: metoda lui Newton

$E: R^n \to R, \qquad w \in R^n$ (vectorul ce contine toate ponderile)

Prin dezvoltare in serie Taylor in $w(p)$ (estimarea corespunzatoare epocii p)

$$E(w) \cong E(w(p)) + (\nabla E(w(p)))^T (w - w(p)) + \frac{1}{2}(w - w(p))^T H(w(p))(w - w(p))$$

$$H(w(p))_{ij} = \frac{\partial^2 E(w(p))}{\partial w_i \partial w_j}$$

Derivand dezvoltarea in serie Taylor in raport cu w si punand conditia de punct critic noua aproximare pentru w se va obtine ca solutie a ecuatiei:

$$H(w(p))w - H(w(p))w(p) + \nabla E(w(p)) = 0$$

Noua estimare a lui w va fi:

$$w(p+1) = w(p) - H^{-1}(w(p)) \cdot \nabla E(w(p))$$

Probleme ale algoritmului BP

Caz particular: metoda Levenberg-Marquardt

- Metoda lui Newton adaptată pentru cazul în care eroarea este o sumă de pătrate de diferențe (cum este eroarea medie patratică)

$$E(w)=\sum_{l=1}^{L}E_l(w), \qquad e: R^n \to R^L, e(w)=(E_1(w),\ldots,E_L(w))^T$$

$$w(p+1)=w(p)-(J^T(w(p))\cdot J(w(p))+\mu_p I)^{-1}J^T(w(p))e(w(p))$$

$J(w)=$ jacobianul lui $e(w)=$ matricea derivatelor lui e in raport cu toate argumentele

$$J_{ij}(w)=\frac{\partial E_i(w)}{\partial w_j}$$

Termen de perturbare care elimină cazurile singulare (cand matricea este neinversabilă)

Avantaje:

- Nu necesită calculul hessianei
- Pentru valori mari ale factorului de atenuare ajustarea devine similară celei de la metoda gradientului

Probleme ale algoritmului BP

P3: Problema minimelor locale (procesul de învățare se blochează într-un minim local al funcției de eroare)

Cauza: metoda gradientului este o metodă de minimizare locală

Soluții:

- Se restartează antrenarea de la alte valori inițiale ale ponderilor
- Se introduc perturbații aleatoare (se adaugă la ponderi după aplicarea ajustărilor):

$$w_{ij} := w_{ij} + \xi_{ij}, \quad \xi_{ij} = \text{valori aleatoare uniform sau normal distribuite}$$

Probleme ale algoritmului BP

Soluție:

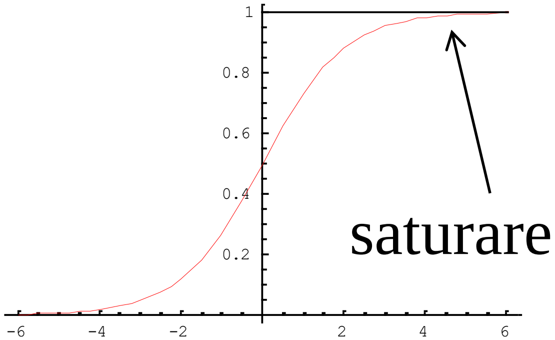
- Inlocuirea metodei gradientului cu o metodă aleatoare de optimizare
- Inseamnă utilizarea unei perturbații aleatoare în locul celei calculate pe baza gradientului
- Ajustările pot conduce la creșterea valorii erorii

$$\Delta_{ij} := \text{valori aleatoare}$$
$$\text{IF } E(W + \Delta) < E(W) \text{ THEN se accepta ajustare } (\text{W} := \text{W} + \Delta)$$

Obs:

- Ajustările sunt de regulă generate în conformitate cu repartiția normală de medie 0 și dispersie adaptivă
- Daca ajustarea nu conduce la o descreștere a valorii erorii atunci nu se acceptă deloc sau se acceptă cu o probabilitate mică
- Algoritmii aleatori de minimizare nu garanteaza obținerea minimului dar unii dintre ei satisfac proprietăți de convergență în sens probabilist.

Probleme ale algoritmului BP



- Pb 4: Stagnare

(procesul de învățare stagnează chiar dacă nu s-a ajuns într-un minim local)

- Cauza: ajustările sunt foarte mici întrucât se ajunge la argumente mari ale funcțiilor sigmoidale ceea ce conduce la valori foarte mici ale derivatelor; argumentele sunt mari fie datorită faptului ca datele de intrare nu sunt normalizate fie întrucât valorile ponderilor sunt prea mari
- Soluții:
  - Se "penalizează" valorile mari ale ponderilor prin regularizare (weight decay)
  - Se utilizeaza doar semnele derivatelor nu și valorile lor
  - Se normalizează datele de intrare (valori în apropierea intervalului (-1,1))
  - Se utilizează funcții de activare de tip ReLU

Probleme ale algoritmului BP

Penalizarea valorilor mari ale ponderilor: se adaugă un termen de penalizare la funcția de eroare (similar cu tehnicile de regularizare folosite în metodele de optimizare)

$$E_{(r)}(W) = E(W) + \lambda \sum_{i,j} w_{ij}^2$$

Ajustarea va fi:

$$\Delta^{(r)} w_{ij} = \Delta w_{ij} - 2\lambda w_{ij}$$

Obs: o altă variantă de regularizare este cea în care în loc de pătrate ale valorii ponderilor se consideră valoarea absolută ($|w_{ij}|$) – regularizare de tip Lasso (favorizeaza creșterea numărului de ponderi nule)

Probleme ale algoritmului BP

Utilizarea semnului derivatei nu și a valorii

(Resilient BackPropagation – RPROP)

$$\Delta w_{ij}(p) = \begin{cases} -\Delta_{ij}(p) & \text{if } \dfrac{\partial E(W(p-1))}{\partial w_{ij}} > 0 \\ \Delta_{ij}(p) & \text{if } \dfrac{\partial E(W(p-1))}{\partial w_{ij}} < 0 \end{cases}$$

$$\Delta_{ij}(p) = \begin{cases} a\Delta_{ij}(p-1) & \text{if } \dfrac{\partial E(W(p-1))}{\partial w_{ij}} \cdot \dfrac{\partial E(W(p-2))}{\partial w_{ij}} > 0 \\ b\Delta_{ij}(p-1) & \text{if } \dfrac{\partial E(W(p-1))}{\partial w_{ij}} \cdot \dfrac{\partial E(W(p-2))}{\partial w_{ij}} < 0 \end{cases}$$

$$0 < b < 1 < a$$

Probleme ale algoritmului BP

Pb 5: Supraantrenare și capacitate limitată de generalizare

Cauze:

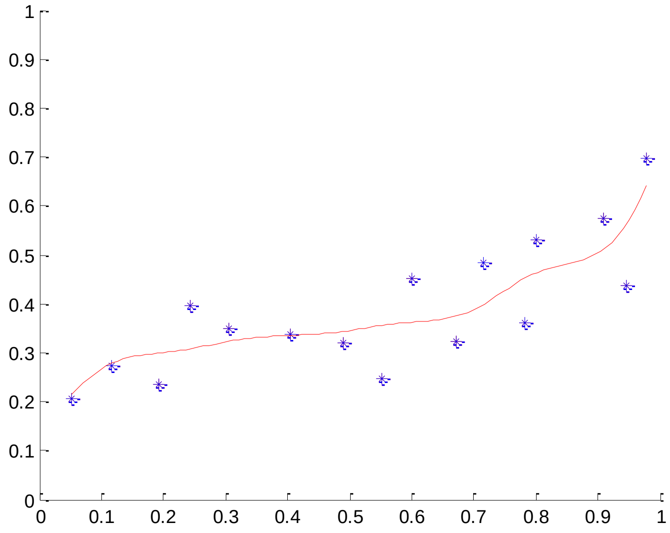
- Arhitectura rețelei (numărul de unitați ascunse)
  - Un număr prea mare de unități ascunse poate provoca supraantrenare (rețeaua extrage nu doar informațiile utile din setul de antrenare ci și zgomotul)
- Dimensiunea setului de antrenare
  - Prea puține exemple nu permit antrenarea și asigurarea capacității de generalizare
- Numărul de epoci (toleranța la antrenare)
  - Prea multe epoci pot conduce la supraantrenare

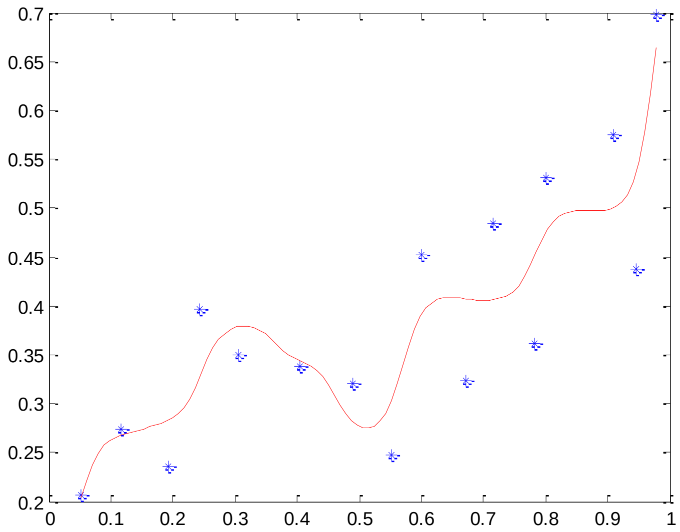
Soluții:

- Modificarea dinamică a arhitecturii
- Criteriul de oprire se bazează nu pe eroarea calculată pentru setul de antrenare ci pentru un set de validare

Probleme ale algoritmului BP

Supraantrenare – influența numărului de unități ascunse



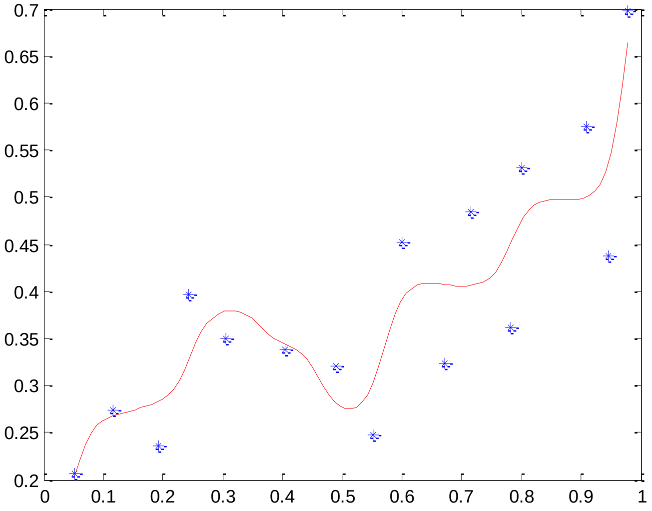


5 unități ascunse

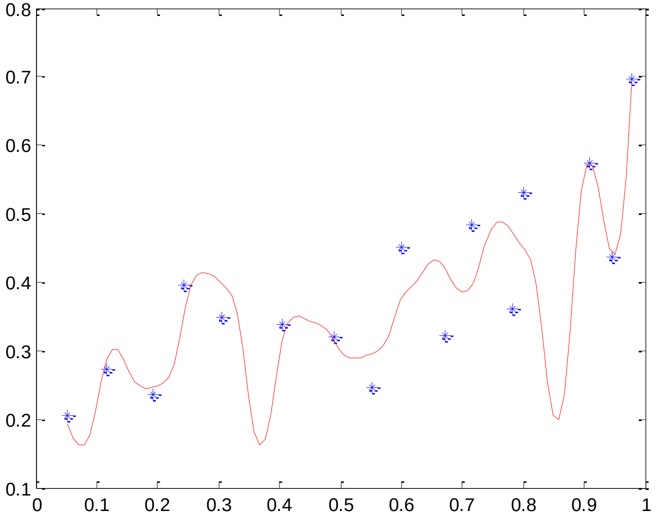
10 unități ascunse

Probleme ale algoritmului BP

Supraantrenare – influența numărului de unități ascunse



10 unități ascunse



20 unități ascunse

Probleme ale algoritmului BP

Modificarea dinamică a arhitecturii:

- Strategie incrementală:
  - Se pornește cu un număr mic de unități ascunse
  - Dacă antrenarea nu progresează se adaugă succesiv unități; pentru asimilarea lor se ajustează în câteva epoci doar ponderile corespunzătoare

- Strategie decrementală:
  - Se pornește cu un număr mare de unități
  - Dacă există unități care au impact mic asupra semnalului de ieșire atunci acestea se elimină

Probleme ale algoritmului BP

Criteriu de oprire bazat pe eroarea pe setul de validare :

- Se imparte setul de antrenare în m părți: (m-1) sunt folosite pentru antrenare și una pentru validare
- Ajustarea se aplică până când eroarea pe setul de validare începe să crească (sugerează că rețeaua începe să piardă din abilitatea de generalizare)

Validare încrucișată (reminder):

- Algoritmul de învățare se aplică de m ori pentru cele m variante posibile de selecție a subsetului de validare

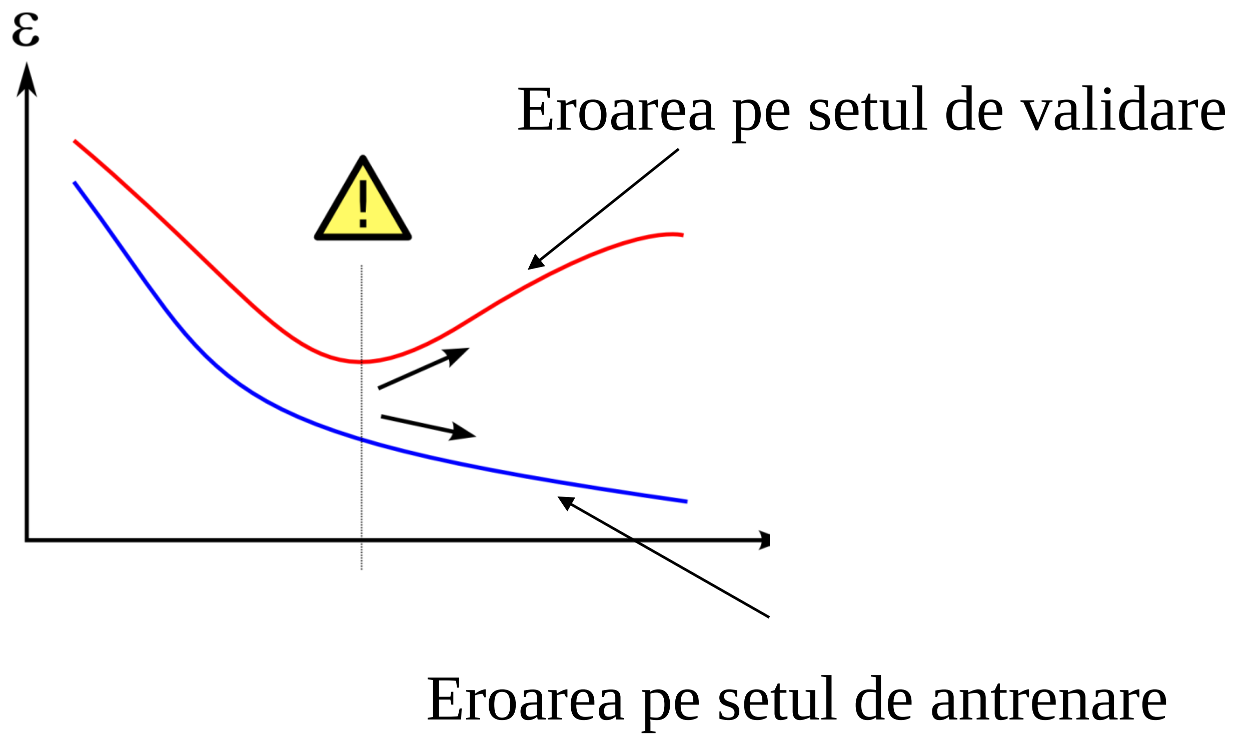
1: S=(S1,S2, ...,Sm)

2: S=(S1,S2, ...,Sm)

....

m: S=(S1,S2, ...,Sm)

Probleme ale algoritmului BP



Support Vector Machines

Support Vector Machine (SVM) = tehnică de clasificare caracterizată prin:

- Antrenare bazată pe o metodă de optimizare cu restricţii şi funcţie obectiv pătratică.

Obs: se evită problemele ce apar la antrenarea de tip Backpropagation (blocarea în minime locale si supraantrenarea)

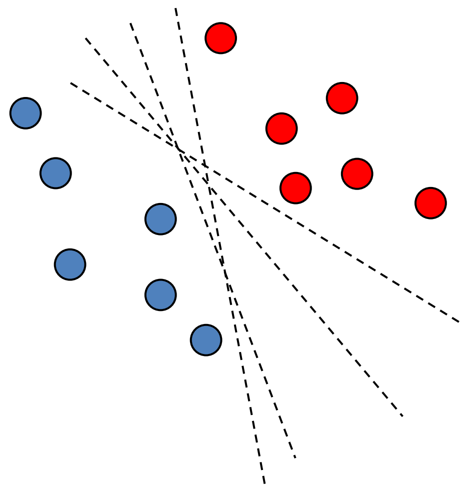
- Asigură o bună capacitate de generalizare

- Se bazează pe rezultate teoretice din domeniul analizei statistice a metodelor de învăţare (principalii contributori: Vapnik și Chervonenkis)

- Aplicații: recunoaștere scris, identificarea vorbitorului, recunoaștere obiecte etc
- Bibliografie: C.Burges – A Tutorial on SVM for Pattern Recognition, Data Mining and Knowledge Discovery, 2, 121–167 (1998)

Support Vector Machines

Considerăm o problemă simplă de clasificare binară



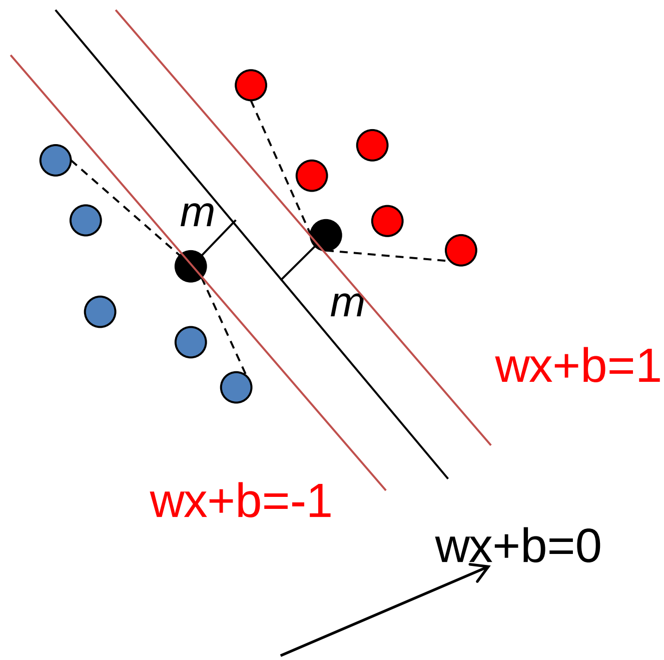
Problema e liniar separabilă și se observă că există o infinitate de drepte (hiperplane, în cazul general) care permit separarea celor două clase

Care dintre hiperplanele separatoare este mai bun ?

Cel care ar conduce la o bună capacitate de generalizare = clasificare corectă nu doar pentru datele din setul de antrenare ci și pentru potențialele date de test

Support Vector Machines

Care e cea mai bună dreaptă (hiperplan) separatoare ?



Ecuația dreptei (hiperplanului) separatoare

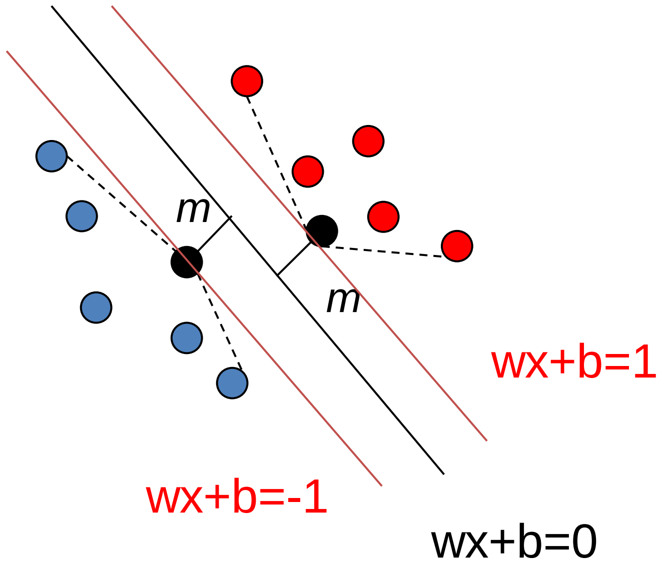
Cea pentru care distanța minimă față de punctele aflate pe înfășurătoarea convexă a setului de puncte corespunzător fiecărei clase este maximă

Dreptele care trec prin punctele marginale sunt considerate drepte canonice

Distanța dintre dreptele canonice este 2/||w||, deci a maximiza lărgimea zonei separatoare este echivalent cu a minimiza norma lui w

Support Vector Machines

Cum se poate determina hiperplanul separator ?



Se determină w și b care

Minimizează $||w||^2$

(maximizează marginea separatoare)

și satisface

$(wx_i+b)d_i-1>=0$

pentru toate elementele setului de antrenare $\{(x_1,d_1),(x_2,d_2),\ldots,(x_L,d_L)\}$

$d_i=-1$ pentru clasa albastră

$d_i=1$ pentru clasa roșie

(clasifică corect exemplele din setul de antrenare)

Support Vector Machines

Problema de minimizare cu restricții se poate rezolva folosind metoda multiplicatorilor lui Lagrange:

Problema inițială:

Minimizează $||w||^2$ astfel încât $(wx_i+b)d_i-1>=0$ pentru i=1..L

Introducerea multiplicatorilor lui Lagrange transformă problema în determinarea punctului șa (saddle point) pentru V:

$$V(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \alpha_i \left(d_i(w \cdot x_i + b) - 1\right), \qquad \alpha_i \geq 0$$

(w^*, b^*, α^*) este punct sa daca: $V(w^*, b^*, \alpha^*) = \max_\alpha \min_{w,b} V(w,b,\alpha)$

Construirea funcției duale:

$$W(\alpha) = \min_{w,b} V(w,b,\alpha)$$

$$\frac{\partial V(w,b,\alpha)}{\partial w} = 0 \Rightarrow w = \sum_{j=1}^{L} \alpha_j\, d_j x_j \qquad \frac{\partial V(w,b,\alpha)}{\partial b} = 0 \Rightarrow 0 = \sum_{j=1}^{L} \alpha_j\, d_j$$

Support Vector Machines

Se ajunge astfel la problema maximizării funcției duale (în raport cu α):

$$W(\alpha) = \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{L} \alpha_i \, \alpha_j d_i d_j (x_i \cdot x_j)$$

(cunoscute din setul de antrenare)

Cu restricțiile:

$$\alpha_i \geq 0, \qquad \sum_{i=1}^{L} \alpha_i \, d_i = 0$$

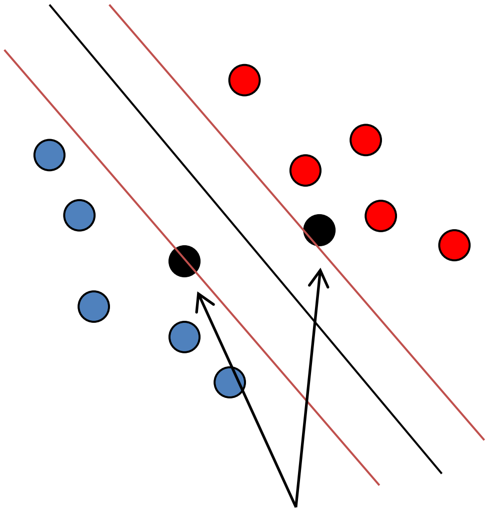
După rezolvarea problemei de mai sus (în raport cu multiplicatorii α) se calculează elementele hiperplanului separator astfel:

$$w^* = \sum_{i=1}^{L} \alpha_i d_i \, x_i, \qquad b^* = 1 - w^* \cdot x_k$$

unde k este indicele unui multiplicator nenul iar x_k este exemplul corespunzător ce aparține clasei de etichetă +1

Support Vector Machines

Observații:



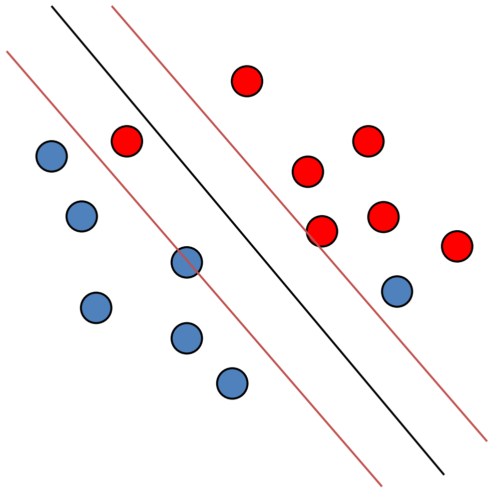
- Multiplicatorii nenuli corespund exemplelor pentru care restricțiile sunt active (w x+b=1 sau w x+b=-1). Aceste exemple sunt denumite vectori suport și sunt singurele care influențează ecuația hiperplanului separator (celelalte exemple din setul de antrenare pot fi modificate fără a influența hiperplanul separator)

- Multiplicatorii nuli corespund elementelor din setul de antrenare care nu influențează hiperplanul separator
- Funcția de decizie obținută după rezolvarea problemei de optimizare pătratică este:

$$D(z) = \mathrm{sgn}\left(\sum_{i=1}^{L} \alpha_i d_i \,(x_i \cdot z) + b^*\right)$$

Support Vector Machines

Ce se întâmplă în cazul în care datele nu sunt foarte bine separate ?



Se relaxează condiția de apartenență la o clasă:

$$w \cdot x_i + b \geq 1 - \xi_i, \qquad \text{daca } d_i = 1$$
$$w \cdot x_i + b \leq -1 + \xi_i, \qquad \text{daca } d_i = -1$$

Funcția de minimizat devine:

$$V(w, b, \alpha, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} \xi_i - \sum_{i=1}^{L} \alpha_i \left(d_i(w \cdot x_i + b) - 1\right)$$

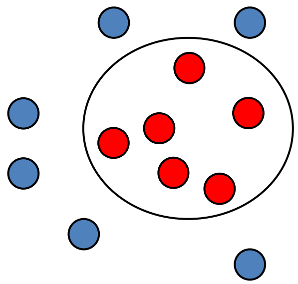
Ceea ce schimbă restricțiile din problema duală astfel:

$$\text{in loc de } \alpha_i \geq 0 \text{ se introduce } 0 \leq \alpha_i \leq C$$

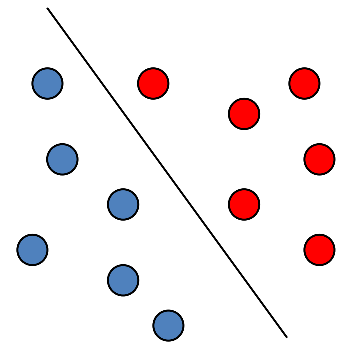
Obs: Parametrul C controlează compromisul între a accepta erori pe setul de antrenare și a avea margine largă (abilitate de generalizare). Cu cât C e mai mare cu atât restricția este mai puternică (erorile sunt mai puternic penalizate)

Support Vector Machines

Ce se întâmplă în cazul in care problema NU este liniar separabilă?



$x_1^2 + x_2^2 - R^2 = 0$



$w \cdot z + b = 0,\quad z_1 = x_1^2,\ z_2 = x_2^2$

$w_1 = w_2 = 1,\quad b = -R^2$

$x_1 \rightarrow \theta(x_1) = x_1^2$

$x_2 \rightarrow \theta(x_2) = x_2^2$

Support Vector Machines

In cazul general se aplică transformarea:

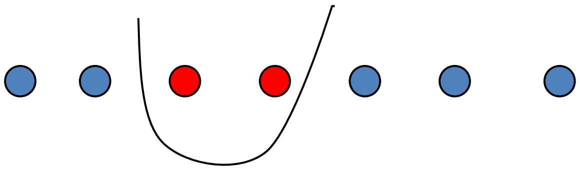
$x \to \theta(x)$ iar produsul scalar al vectorilor transformati este
$\theta(x) \cdot \theta(x') = K(x, x')$

Intrucât în rezolvarea problemei de optimizare intervin doar produsele scalare nu este necesară cunoașterea expresiei explicite a funcției de transformare θ ci este suficient să se cunoască doar funcția nucleu K

Support Vector Machines

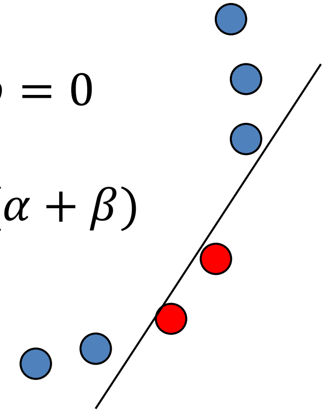
Exemplu 1: Transformarea unei probleme neliniar separabile într-una liniar separabilă prin trecerea la o dimensiune mai mare

$$(x-\alpha)(x-\beta) = x^2 - (\alpha+\beta)x + \alpha\beta$$



Pb. 1-dimensională neliniar separabilă

$$w_1 z_1 + w_2 z_2 + b = 0$$
$$z_1 = x^2, z_2 = x$$
$$w_1 = 1, w_2 = -(\alpha+\beta)$$
$$b = \alpha\beta$$



Pb. 2-dimensională liniar separabilă

Exemplu 2: Deducerea unei funcții nucleu în cazul în care suprafața de decizie este dată de o funcție pătratică oarecare (se trece de la dimensiunea 2 la dimensiunea 5)

$$\theta(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

$$K(x, x') = \theta(x_1, x_2) \cdot \theta(x'_1, x'_2) = (x^T \cdot x' + 1)^2$$

Support Vector Machines

Exemple de functii nucleu:

$$K(x, x') = (x^T \cdot x' + 1)^d$$

$$K(x, x') = \exp(-\frac{\|x - x'\|^2}{2\sigma^2})$$

$$K(x, x') = \tanh(kx^T \cdot x' + b)$$

Functia de decizie devine:

$$D(z) = \operatorname{sgn}(\sum_{i=1}^{L} \alpha_i y_i K(x_i, z) + b^*)$$

Support Vector Machines

Adaptare SVM pentru clasificare multiplă (M clase)

Idee de bază: reducerea construirii unui clasificator multiplu la construirea mai multor clasificatori binari

Variante:

"one vs all": pentru fiecare clasă se construiesc M-1 clasificatori binari al căror scop este să separe clasa țintă de celelalte

- Antrenare: pentru fiecare clasă se pregătește un set de antrenare în care clasa țintă este considerată clasa pozitivă iar toate celelalte clase reprezintă clasa negativă
- Clasificare: se alege clasa pentru care valoarea corespunzătoare funcției de decizie înainte de aplicarea funcției signum ($\sum_{i=1}^{L} \alpha_i y_i K(x_i, z) + b^*$) este cea mai mare

Support Vector Machines

Adaptare SVM pentru clasificare multiplă (M clase)

Idee de bază: reducerea construirii unui clasificator multiplu la construirea mai multor clasificatori binari

Variante:

"one vs one": se construiesc M(M-1)/2 clasificatori binari (câte unul pentru fiecare pereche de clase)

- Antrenare: pentru fiecare pereche se utilizează subset de antrenare ce conține doar exemple din cele două clase

- Clasificare: pentru o nouă instanță se alege clasa dominantă obținută prin aplicarea celor M(M-1)/2 clasificatori

Support Vector Machines

Implementări

LibSVM [http://www.csie.ntu.edu.tw/~cjlin/libsvm/]: (+ link-uri catre implementari in Java, Matlab, R, C#, Python, Ruby)

SVM-Light [http://www.cs.cornell.edu/People/tj/svm_light/]: implementare in C

SciKit-learn – implementări în Python

R – pachet caret

Sumar

Rețele neuronale feedforword (modele de tip perceptron)

- Modele flexibile pentru clasificare neliniară într-un număr arbitrar de clase
- Proiectare:
  - Arhitectura (graf orientat aciclic)
  - Funcționare (funcții de activare)
  - Antrenare supervizată (estimarea parametrilor algoritmi de optimizare neliniară – implică derivate de ordinul 1 sau de ordinul 2)

- Dezavantaje:
  - Model de tip black box (nu e interpretabil)
  - Pot fi utilizate doar pentru date numerice
  - Alegerea arhitecturii (numărul și dimensiunea nivelelor ascunse) și a valorilor hiperparametrilor (rata de învățare, coeficient moment) poate fi dificilă
  - Funcția de eroare nu e neapărat pătratică ➔ dificultăți în procesul de antrenare (ex: blocare în minime locale, stagnare)

- Recomandări practice:
  - Se recomandă standardizarea datelor de intrare
  - Se recomandă utilizarea unor strategii de regularizare (ex: weight decay)

Sumar

Support Vector Machines (clasificatori bazați pe vectori suport)

- Modele flexibile pentru clasificare neliniară într-un număr arbitrar de clase

- Proiectare:
  - Alegerea funcției nucleu (kernel)
  - Alegerea valorii parametrului care controlează toleranța la erori pe setul de antrenare (C)

- Avantaje (în raport cu rețelele neuronale):
  - Antrenarea este mai rapidă și nu există risc de blocare în optime locale (antrenarea se bazează pe metode de optimizare pătratică)
  - Nu necesită un număr mare de exemple în setul de antrenare

- Dezavantaje:
  - Poate fi utilizat doar pentru date numerice
  - Nu permite aplicare directă în cazul clasificării multiple

Curs următor

Gruparea datelor

- Concepte de bază
- Evaluarea calităţii grupării
- Algoritmi partiţionali
- Algoritmi ierarhici