Rețele neuronale. Rezolvarea problemelor de asociere

- Probleme de asociere = clasificare, aproximare, predicție
- Rețele feed-forward uninivel si multinivel (multilayer perceptrons)
- Rețele cu funcții radiale (RBF networks)
- Clasificatori bazați pe vectori suport (Support Vector Machines)

Probleme de asociere

Exemplu 1: identificarea speciei din care face parte o floare de iris

- Atribute: lungime sepale / petale, lățime petale/ sepale
- Clase: Iris setosa, Iris versicolor, Iris virginica

Probleme de asociere

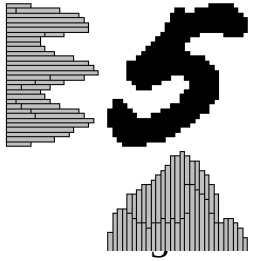
Exemplu 1: identificarea speciei din care face parte o floare de iris

- Atribute: lungime sepale / petale, lățime petale/ sepale
- Clase: Iris setosa, Iris versicolor, Iris virginica

Exemplu 2: recunoașterea caracterelor

- Atribute: caracteristici statistice, caracteristici geometrice
- Clase: corespund setului de caractere care urmeaza a fi recunoscute

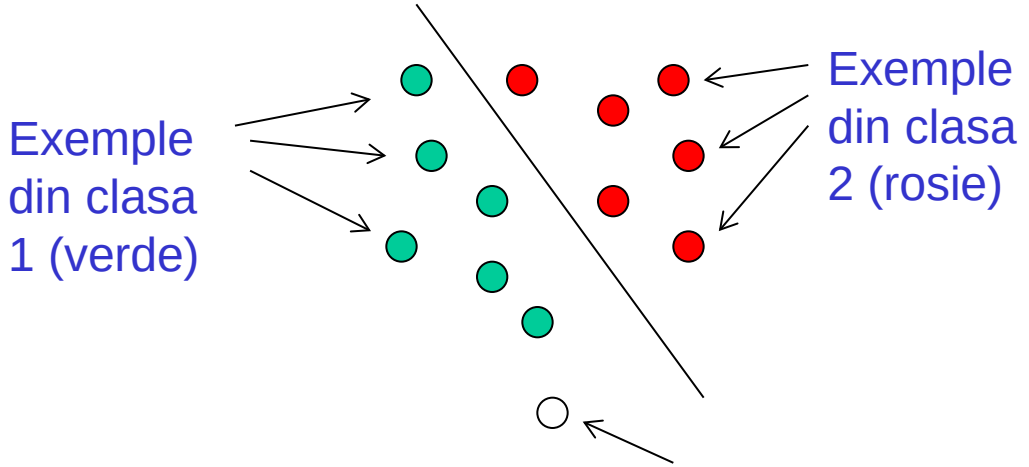
$\Rightarrow$ Probleme de clasificare = asociere între vectori de atribute și indici de clase



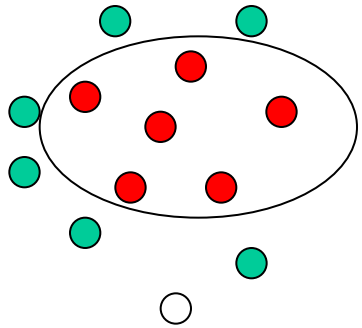
(Du Trier et al; Feature extraction methods for character Recognition. A Survey. Pattern Recognition, 1996)

Probleme de asociere

- Clasificare:
  - Problema: identificarea clasei căreia îi aparține un obiect descris printr-un set de atribute;
  - Se cunosc: exemple de clasificare corectă, numărul claselor și etichetele acestora (clasificare supervizată)
  - Grad dificultate: de la scăzut (clase clar delimitate) la ridicat (clase delimitate de suprafețe cu structură complexă)



Problemă liniar separabilă



Problemă neliniar separabilă

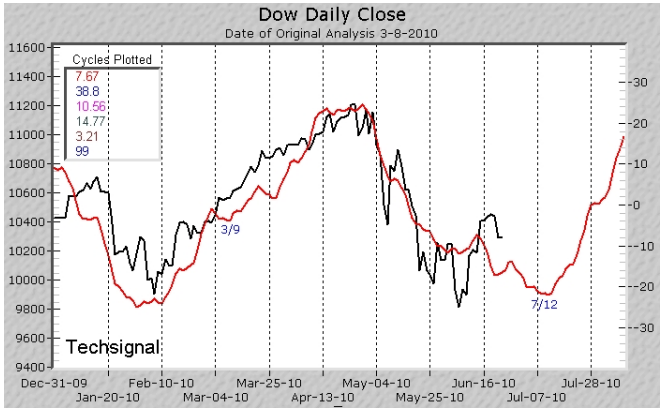
Probleme de asociere

- Estimarea prețului unei case cunoscând:
  - Suprafața totală,
  - Numărul de camere,
  - Dimensiunea grădinii,
  - Zona, etc

  => problemă de aproximare

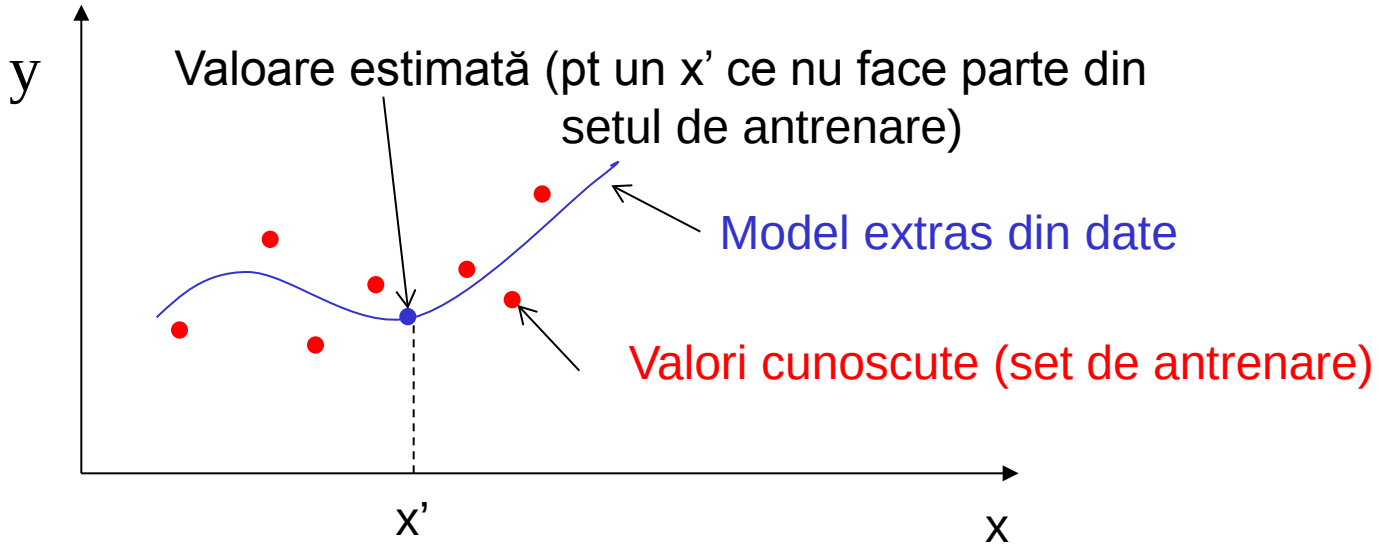
- Estimarea numărului de clienți ai unui serviciu web cunoscând evoluția acestora de-a lungul unei perioade de timp sau estimarea prețului unor active financiare ...

  => problemă de predicție= asociere între valori anterioare si valoarea sau tendința următoare



Probleme de asociere

- Regresie (aproximare, predicție):
  - Problema: determinarea relației dintre două sau mai multe mărimi (aproximarea unei funcții pornind de la cunoașterea valorilor pentru un set de argumente)
  - Se cunosc: perechi de valori corespondente sau succesiune de valori anterioare (set de antrenare)



Probleme de asociere

Reformulare ca problemă de optimizare:

Pornind de la un set de date (X^i,Y^i), X^i din R^N iar Y^i din R^M se caută o funcție F:R^N -> R^M , F=(F_1,...,F_M), care să minimizeze un criteriu de eroare (de exemplu, suma pătratelor erorilor corespunzătoare exemplelor din set):

$$\sum_i \sum_{j=1}^{M} (y_j^i - F_j(X^i))^2$$

Scopul urmărit este să se determine funcția F (având componentele (F_1,...,F_M)) care aproximează cât mai bine datele din set (explică cât mai bine dependența dintre Y și X)

Intrebări:

- Ce forma are F ?
- Cum se determină parametrii corespunzători lui F ?

Probleme de asociere

Poate fi rezolvată o astfel de problemă utilizând rețele neuronale ?

Da, dpdv teoretic rețelele neuronale sunt "aproximatori universali" [Hornik, 1985]:

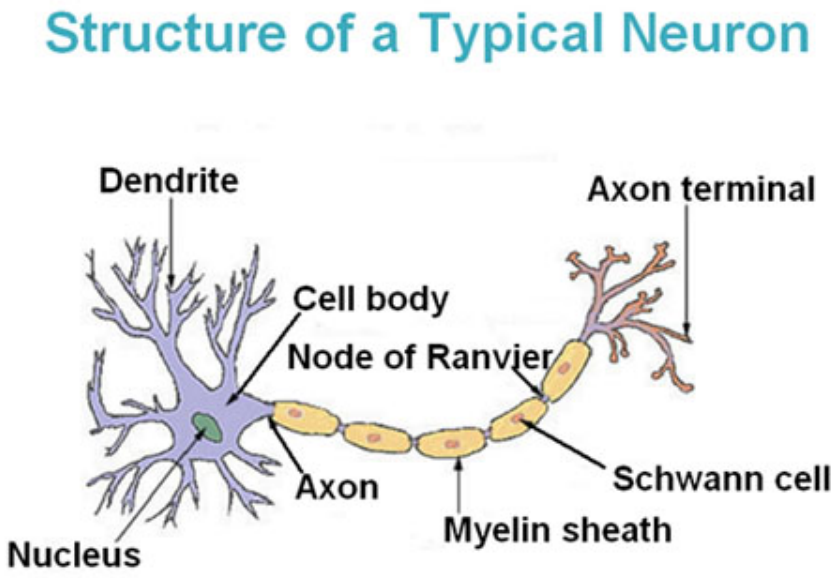
" Orice funcție continuă poate fi aproximată de o rețea feed-forward având cel puțin un nivel ascuns. Acuratețea aproximării depinde de numărul de unități ascunse"

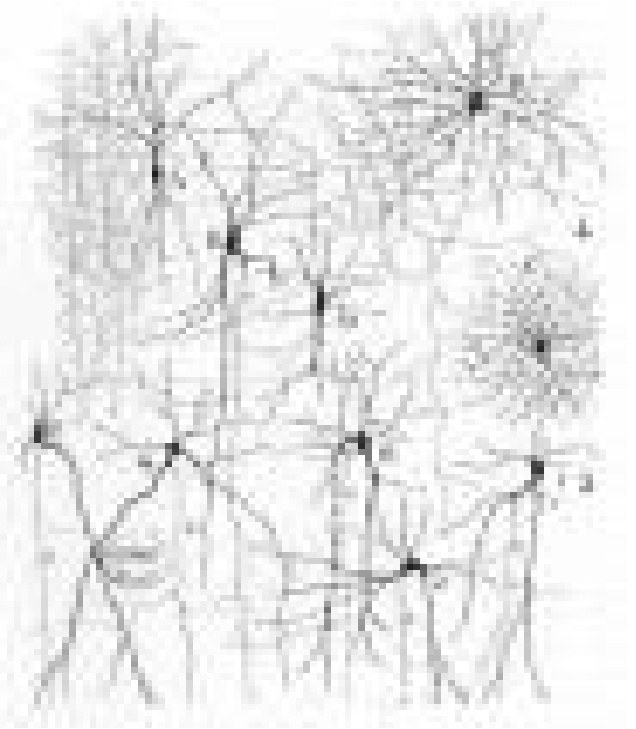
- Forma funcției este determinată de arhitectura rețelei și de proprietățile funcției de activare (transfer)
- Parametrii sunt ponderile asociate conexiunilor dintre unități
- Pentru unele probleme numărul de unități ascunse necesare pentru a obține o reprezentare acceptabilă poate fi inacceptabil de mare; acest fapt este unul dintre motivele dezvoltării arhitecturilor adânci (cu multe nivele)

Retele neuronale – modelul biologic

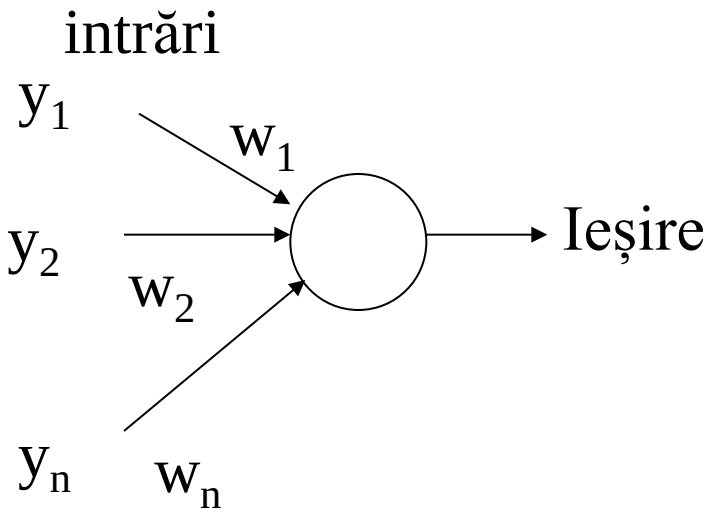
Creierul uman

cca 10^{10} neuroni, cca 10^{14} interconexiuni





Rețele neuronale artificiale



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

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

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

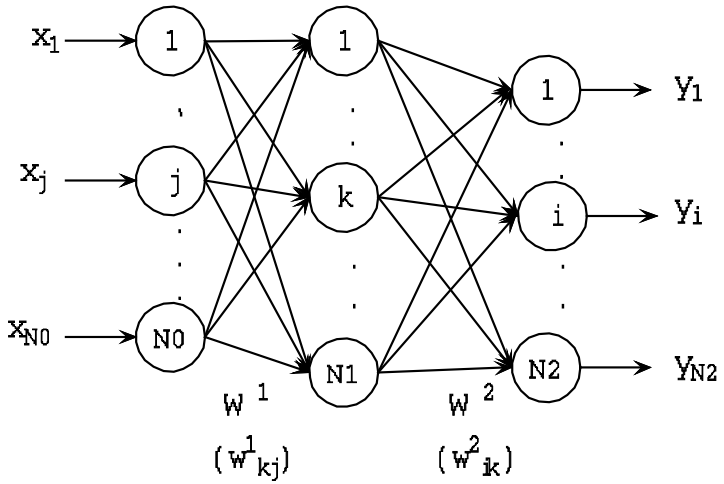
Notații:

- semnale de intrare: $y_1,y_2,\ldots,y_n$
- ponderi sinaptice: $w_1,w_2,\ldots,w_n$ (modelează permeabilitatea sinaptică)
- prag: b (sau w_0) (modelează pragul de activare al neuronului)
- ieșire: y

Obs: Toate valorile sunt numere reale

Rețele neuronale artificiale

Structura unei rețele neuronale artificiale:



- Arhitectura: modul de interconectare între unități
- Funcționare: modul de calcul al semnalului de ieșire

$$y_i = f\left(\sum_{k=0}^{N1} w_{ik}^2 f\left(\sum_{j=0}^{N0} w_{kj}^1 x_j\right)\right),\ i = \overline{1, N2}$$

- Antrenare: modul de determinare a parametrilor ajustabili

Rețea feedforward cu un nivel ascuns

Rețele neuronale artificiale

Proiectarea unei rețele pentru rezolvarea unei probleme de asociere presupune:

- Alegerea arhitecturii:
  - număr de nivele
  - număr de unități pe fiecare nivel
  - mod de interconectare (topologie)
  - funcții de activare

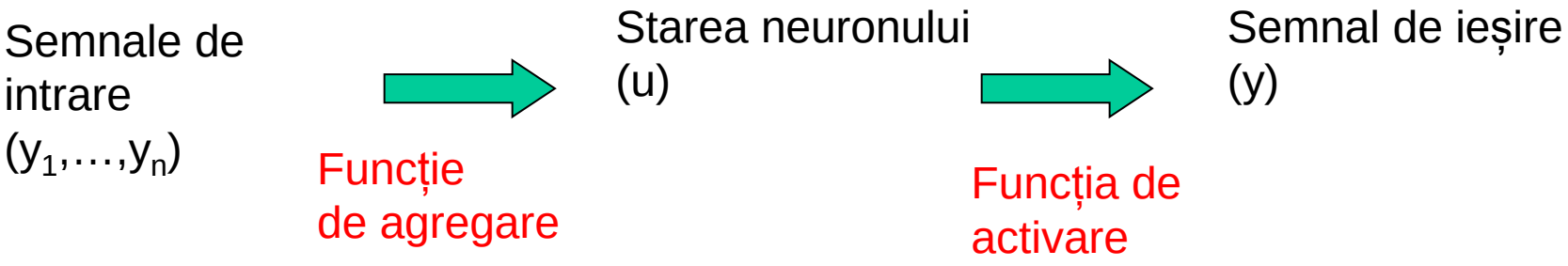
- Antrenarea: determinarea valorilor ponderilor pornind de la setul de antrenare folosind un algoritm de învățare

- Validarea rețelei: analiza comportării rețelei pentru date ce nu fac parte din setul de antrenare

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



Unități funcționale

Exemple de funcții clasice de agregare

Suma ponderată

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

Distanța euclidiană

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

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

Neuron multiplicativ

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

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:

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

Unități funcționale

Exemple de funcții de activare (transfer)

$$f(u) = \text{sgn}(u) = \begin{cases} -1 & u \le 0 \\ 1 & u > 0 \end{cases}$$

signum

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

Heaviside

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

rampă

$$f(u) = u$$

liniară

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

Semi-liniară (rectified linear unit - ReLU)

Obs: utilizate în rețelele cu structură adâncă

Unități funcționale

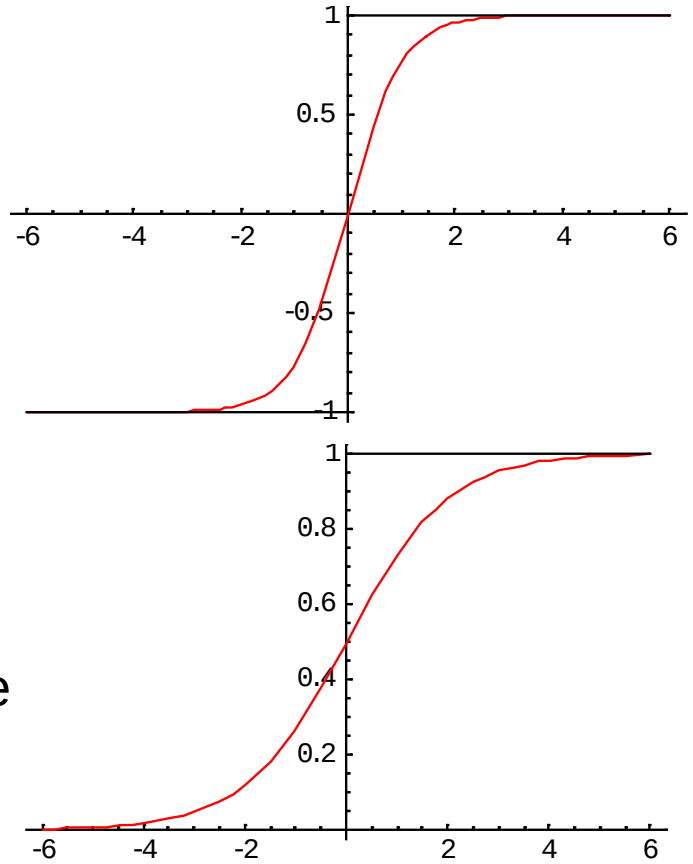
Exemple de funcții de activare (funcții sigmoidale)

(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

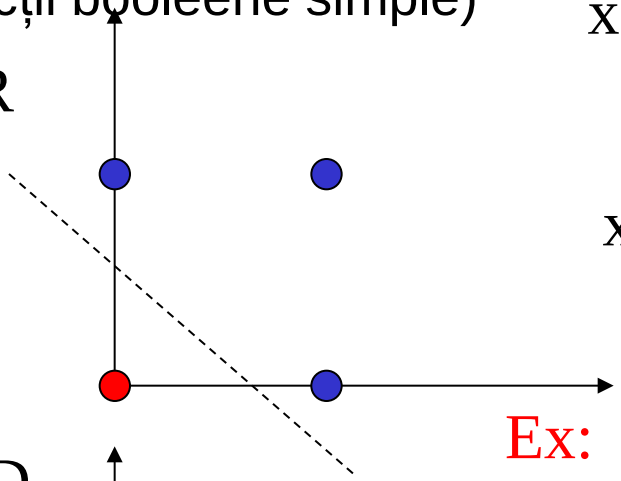
- Ce se poate face cu un singur neuron ?

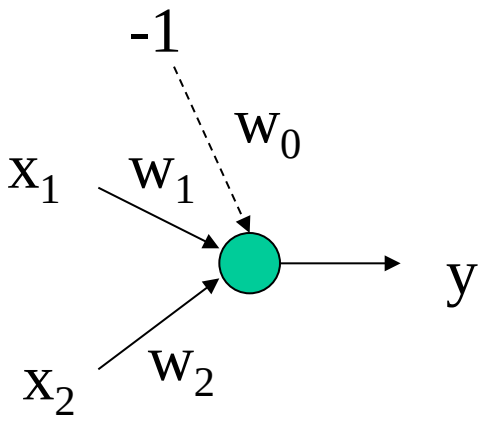
  Se pot rezolva probleme simple de clasificare

(ex: se pot reprezenta funcții booleene simple)

	0	1
0	0	1
1	1	1

OR



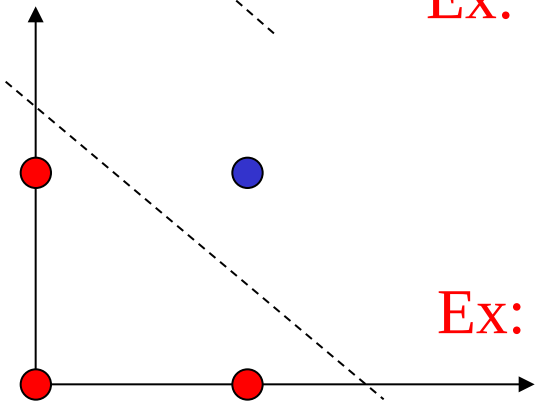


$y=H(w_1x_1+w_2x_2-w_0)$

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

	0	1
0	0	0
1	0	1

AND

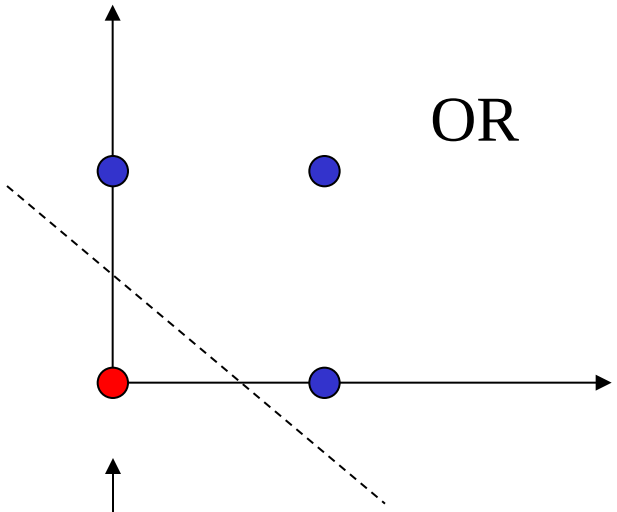


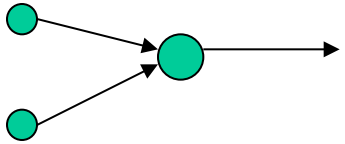
$y=H(w_1x_1+w_2x_2-w_0)$

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

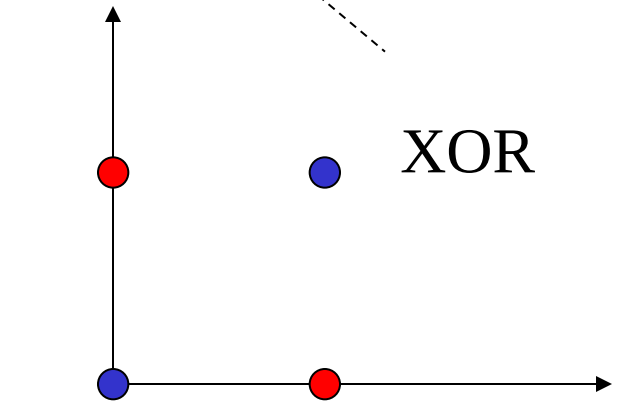
Liniar/neliniar separabilitate

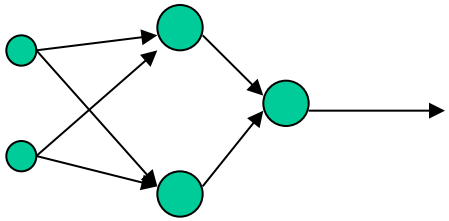
Reprezentarea unor funcții booleene: $f:\{0,1\}^N \rightarrow \{0,1\}$





Problema liniar separabilă – e suficientă o rețea uninivel

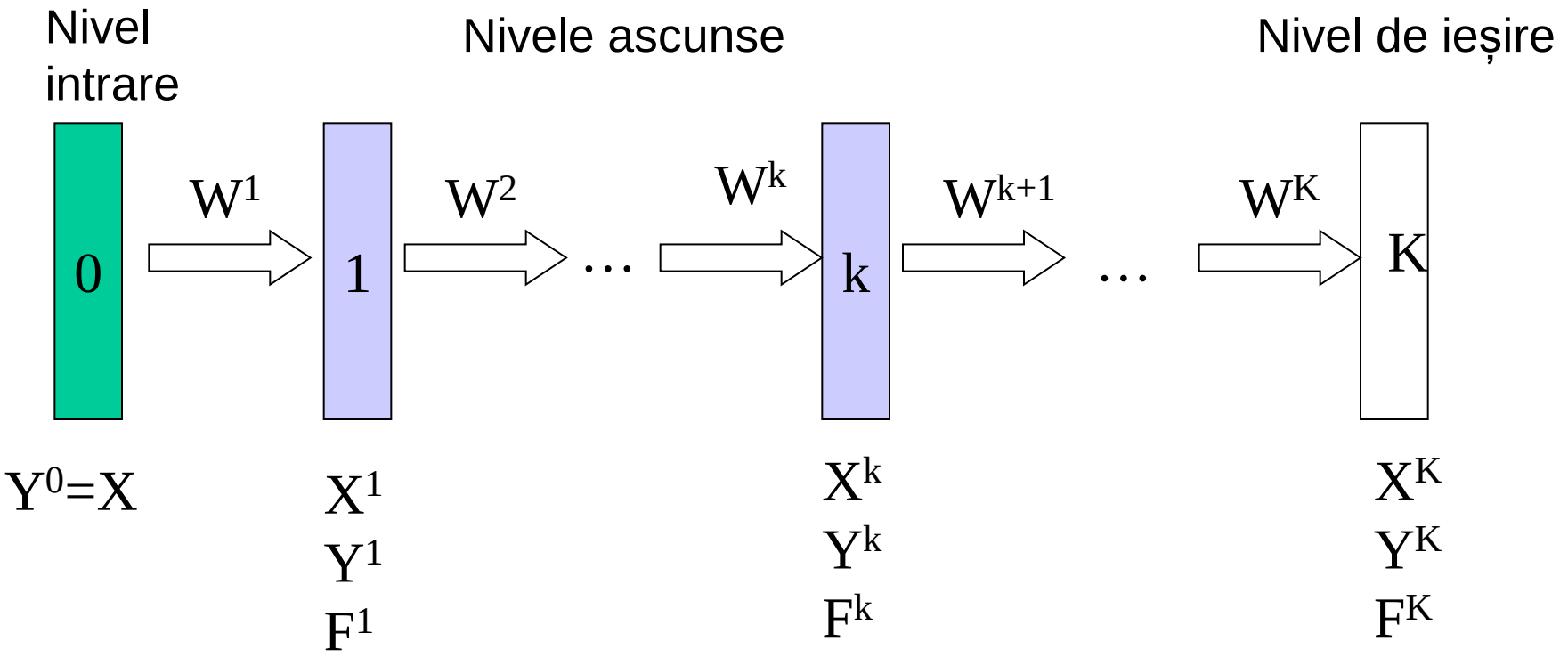




Problema neliniar separabilă – e necesară o rețea multinivel

Rețele feedforward - arhitectura

Arhitectura și funcționare (K nivele funcționale)



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

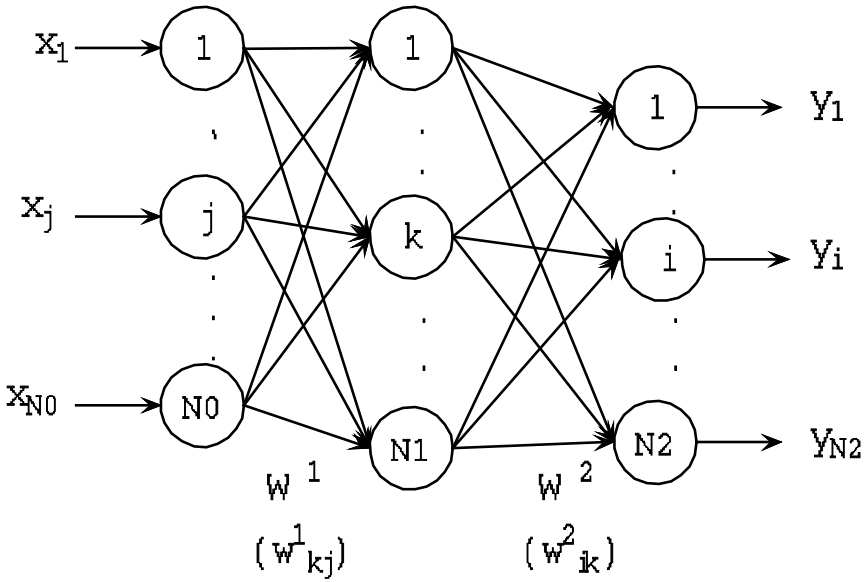
Calcul vector de ieșire: $Y=F^K(W^K*F^{K-1}(W^{K-1}*F^{K-2}(.....F^1(W^1*X))))$

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)



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

Obs:

- în mod tradițional se lucrează cu unul sau două nivele ascunse
- În ultimii ani au devenit din ce în ce mai folosite rețelele cu număr mare de nivele (Deep Neural Networks) folosite în particular pentru recunoașterea imaginilor și a vorbirii (http://deeplearning.net)

Rețele feedforward - antrenare

Antrenare (supervizată):

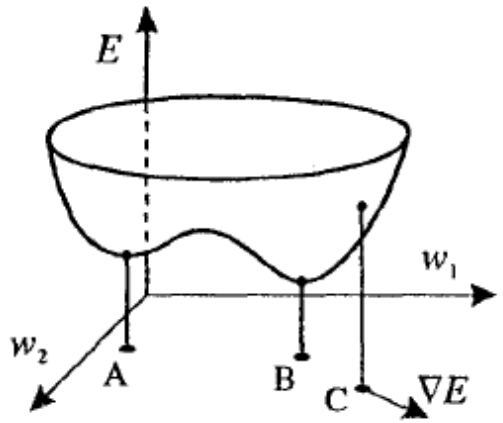
- Set de antrenare: $\{(x^1,d^1), \ldots, (x^L,d^L)\}$

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

- Funcție de eroare (suma pătratelor erorilor):

$$E(W) = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{N2}\left(d_i^l - f_2\left(\sum_{k=0}^{N1} w_{ik} f_1\left(\sum_{j=0}^{N0} w_{kj} x_j^l\right)\right)\right)^2$$

- Scopul antrenării: minimizarea funcției de eroare
- Metoda de minimizare: metoda gradientului



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:

$$E(W) = \frac{1}{2} \sum_{l=1}^{L} \sum_{i=1}^{N2} \left(d_i^l - f_2 \left(\sum_{k=0}^{N1} w_{ik} f_1 \underbrace{\left(\sum_{j=0}^{N0} w_{kj} x_j^l \right)}_{x_k} \right) \right)^2$$

y_k

x_i

y_i

Notații:

$E_l(W)$ (eroarea corespunzatoare exemplului l)

Rețele feedforward - antrenare

- Calculul derivatelor partiale

$$E(W) = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{N2}\left(d_i^l - f_2\left(\sum_{k=0}^{N1} w_{ik}\, f_1\left(\underbrace{\sum_{j=0}^{N0} w_{kj} x_j^l}_{x_k} \right) \right) \right)^2$$

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

$$\frac{\partial E_l(W)}{\partial w_{kj}} = -\sum_{i=1}^{N2} w_{ik}(d_i^l - y_i)\, f_2'(x_i)\, f_1'(x_k) x_j^l = -\left(f_1'(x_k) \sum_{i=1}^{N2} w_{ik}\delta_i^l \right) x_j = -\delta_k^l x_j^l$$

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

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

$$\frac{\partial E_l(W)}{\partial w_{kj}} = -\sum_{i=1}^{N2} w_{ik}(d_i^l - y_i) f_2^{'}(x_i) f_1^{'}(x_k) x_j^l = -\left(f_1^{'}(x_k) \sum_{i=1}^{N2} w_{ik} \delta_i^l \right) x_j = -\delta_k^l x_j^l$$

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

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

Tanh: $f'(x)=1-f(x)^2 \Rightarrow f'(x)=1-y^2$

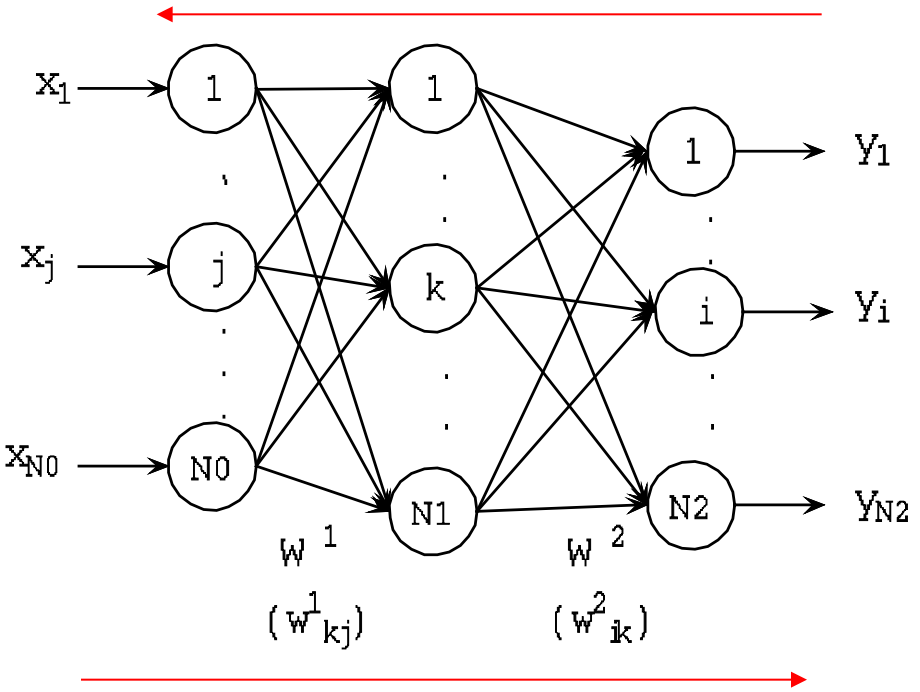
Algoritmul BackPropagation

Idee:

Pentru fiecare exemplu din setul de antrenare:

- 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

Calcul semnal eroare (BACKWARD)



Calcul semnal ieșire (FORWARD)

Algoritmul BackPropagation

Inițializarea aleatoare a ponderilor

REPEAT

epoca

- FOR l=1,L DO
  - etapa FORWARD
  - etapa BACKWARD
  - ajustare ponderi
- Recalcularea erorii

UNTIL <condiție oprire>

Obs.

- Valorile inițiale se aleg aleator in [0,1] sau [-1,1]
- La ajustare se ține cont de rata de învățare
- 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 * /

$$x_k^l = \sum_{j=0}^{N0} w_{kj}^1 x_j^l,\ y_k^l = f_1(x_k^l),\ x_i^l = \sum_{k=0}^{N1} w_{ik}^2 y_k^l,\ y_i^l = f_2(x_i^l)$$

/ * Etapa BACKWARD * /

$$\delta_i^l = f_2^{'}(x_i^l)(d_i^l - y_i^l),\ \delta_k^l = f_1^{'}(x_k^l)\sum_{i=1}^{N2} w_{ik}^2 \delta_i^l$$

/ * Etapa de ajustare * /

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

ENDFOR

Algoritmul BackPropagation

/ * Calculul erorii * /

$E = 0$

FOR $l = 1, L$ DO

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

$$x_k^l = \sum_{j=0}^{N0} w_{kj}^1 x_j^l,\ y_k^l = f_1(x_k^l),\ x_i^l = \sum_{k=0}^{N1} w_{ik}^2 y_k^l,\ y_i^l = f_2(x_i^l)$$

/ * 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)

$w_{kj}^1 = rand(-1,1), w_{ik}^2 = rand(-1,1),\ i = 1..N2, k = 0..N1,\ j = 0..N0$

$p = 0$

REPEAT

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

FOR $l = 1, L$ DO

/ * Etapa FORWARD * /

$$x_k^l = \sum_{j=0}^{N0} w_{kj}^1 x_j^l,\ y_k^l = f_1(x_k^l),\ x_i^l = \sum_{k=0}^{N1} w_{ik}^2 y_k^l,\ y_i^l = f_2(x_i^l)$$

/ * Etapa BACKWARD * /

$$\delta_i^l = f_2^{'}(x_i^l)(d_i^l - y_i^l),\ \delta_k^l = f_1^{'}(x_k^l)\sum_{i=1}^{N2} w_{ik}^2 \delta_i^l$$

/ * Etapa de ajustare * /

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

ENDFOR

$$w_{kj}^1 = w_{kj}^1 + \Delta_{kj}^1,\ w_{ik}^2 = w_{ik}^2 + \Delta_{ik}^2$$

Algoritmul BackPropagation

/ * Calculul erorii * /

$E = 0$

FOR $l = 1, L$ DO

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

$$x_k^l = \sum_{j=0}^{N0} w_{kj}^1 x_j^l,\ y_k^l = f_1(x_k^l),\ x_i^l = \sum_{k=0}^{N1} w_{ik}^2 y_k^l,\ y_i^l = f_2(x_i^l)$$

/ * 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 $E < E^*$

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: Eroarea descreşte 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

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

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

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

$$\alpha = 0.9$$

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

$E : R^n \to R,\ w \in R^n$ (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 ec :

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

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)

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

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

$J(w) =$ 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 elimina cazurile singulare (cand matricea este neinversabila)

Avantaje:

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

Probleme ale algoritmului BP

P2: 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

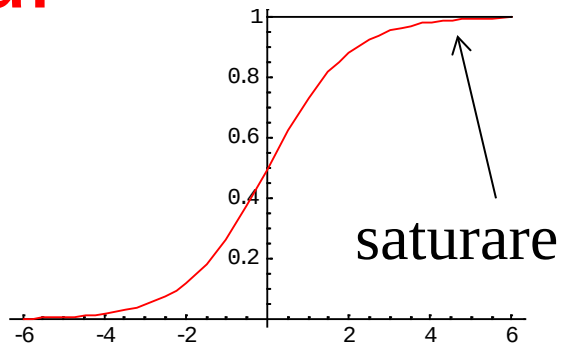
$$\Delta_{ij} := \text{valori aleatoare}$$

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

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

Probleme ale algoritmului BP



- Pb 3: 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
  - Se utilizeaza doar semnele derivatelor nu și valorile lor
  - Se normalizează datele de intrare (valori în apropierea intervalului (-1,1)

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:

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

Probleme ale algoritmului BP

Utilizarea semnului derivatei nu și a valorii

(Resilient BackPropagation – RPROP)

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

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

$0 < b < 1 < a$

Probleme ale algoritmului BP

Pb 4: Supraantrenare și capacitate limitată de generalizare

Cauze:

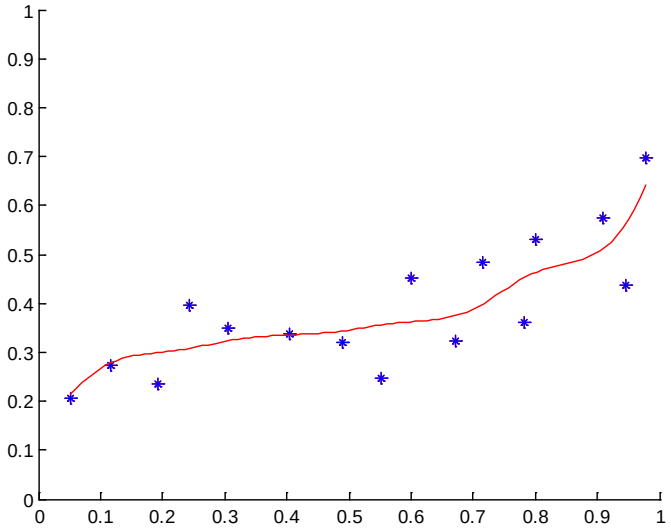
- Arhitectura rețelei (numărul de unitați ascunse)
  - Un număr prea mare de unități ascunse poate provoca supraantrenare (rețeaua extrage nu doar informațiile utile din setul de antrenare ci și zgomotul)
- Dimensiunea setului de antrenare
  - Prea puține exemple nu permit antrenarea și asigurarea capacității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

Soluții:

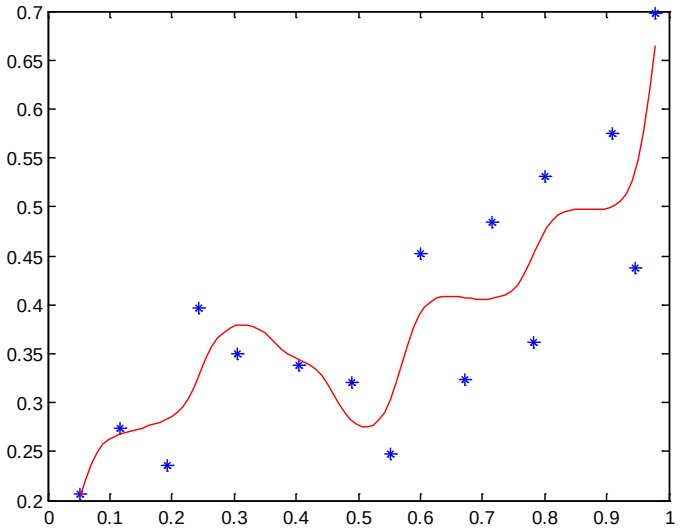
- Modificarea dinamică a arhitecturii
- Criteriul de oprire se bazează nu pe eroarea calculată pentru setul de antrenare ci pentru un set de validare

Probleme ale algoritmului BP

Supraantrenare – influența numărului de unități ascunse



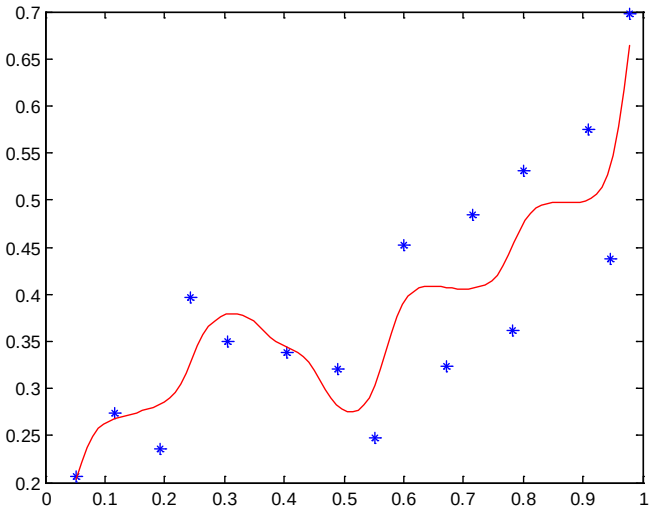
5 unități ascunse



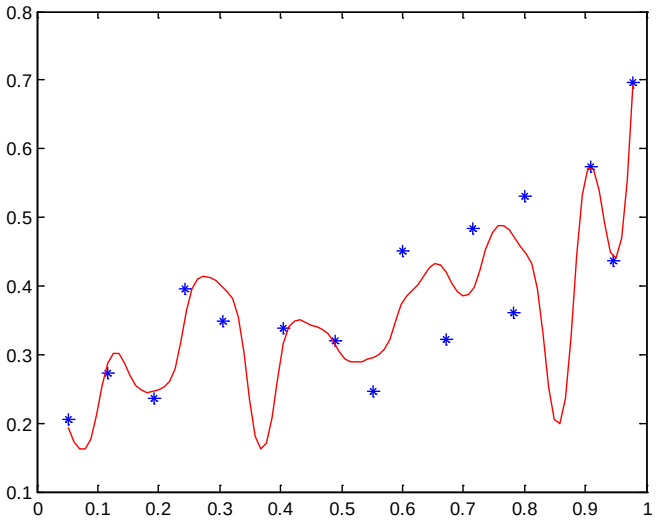
10 unități ascunse

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:

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

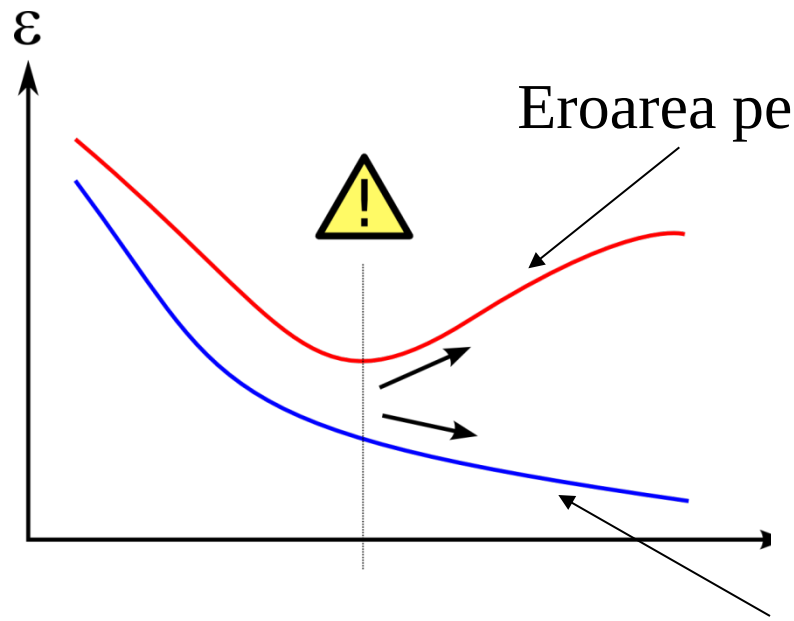
1: S=(S1,S2, ...,Sm)

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

....

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

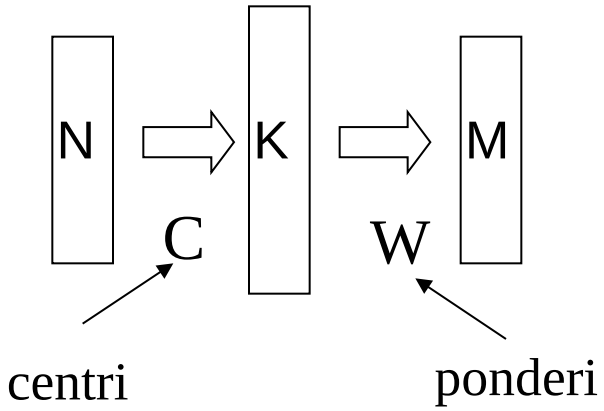
Probleme ale algoritmului BP



Rețele cu funcții radiale

- RBF - “Radial Basis Function”:

- Arhitectura:
  - Două nivele de unități funcționale



  - Funcții de agregare:
    - Unități ascunse: distanța dintre vectorul de intrare și cel al ponderilor corespunzătoare unității ascunse
    - Unități de ieșire: suma ponderată

$$G(X, C^k) = \left\| X - C^k \right\| = \sqrt{\sum_{i=1}^{N} (x_i - c_i^k)^2}$$

Funcții de transfer (activare):

- nivelul ascuns: funcții cu simetrie radială

- nivelul de ieșire: funcții liniare

Rețele cu funcții radiale

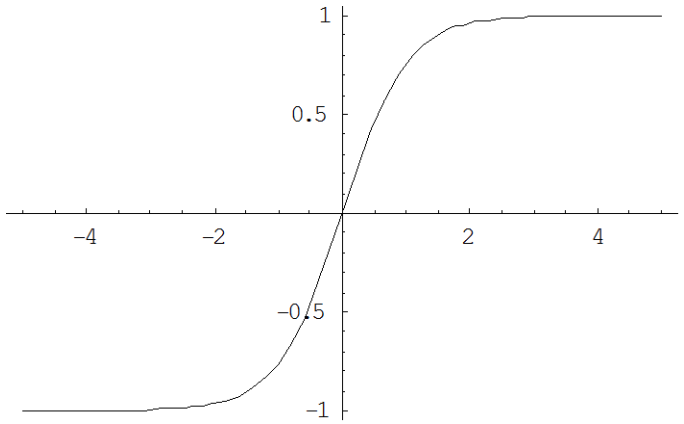
Diferența față de rețelele feedforward clasice:

Funcții de transfer:

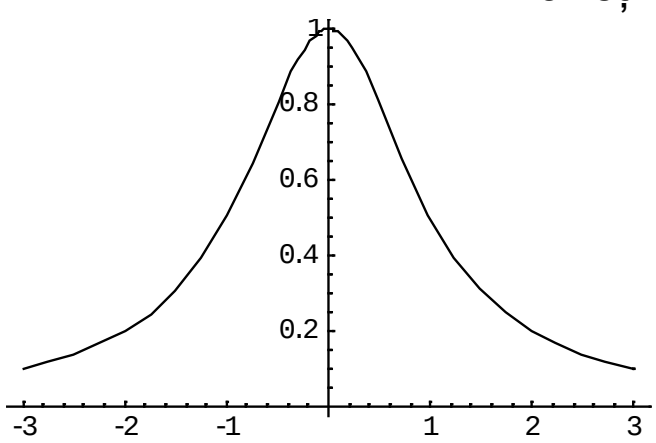
FF: funcții sigmoidale

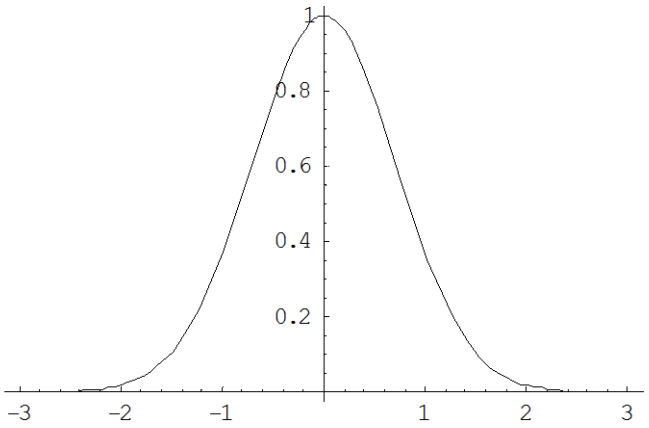
RBF: funcții cu simetrie radială

Funcție sigmoidală



Funcții cu simetrie radială



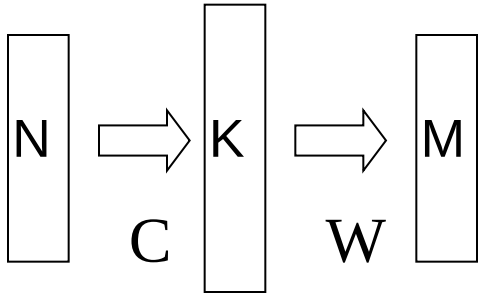


Rețele cu funcții radiale

Funcționare:

$$y_i = \sum_{k=1}^{K} w_{ik}\, g(\|X - C^k\|) - w_{i0},\ i = \overline{1, M}$$

$$y_i = \sum_{k=1}^{K} w_{ik} z_k - w_{i0},\ z_k = g(\|X - C^k\|)$$



Matrice centri Matrice ponderi

Parametrii C^k pot fi interpretați ca prototipuri (centri) asociați unităților ascunse: vectorii de intrare X apropiați lui C^k vor conduce la o valoarea de ieșire semnificativă pe când cei îndepărtați vor conduce la o valoare de ieșire nesemnificativă; la construirea răspunsului rețelei vor contribui doar unitățile a căror centri sunt suficient de similari cu data de intrare

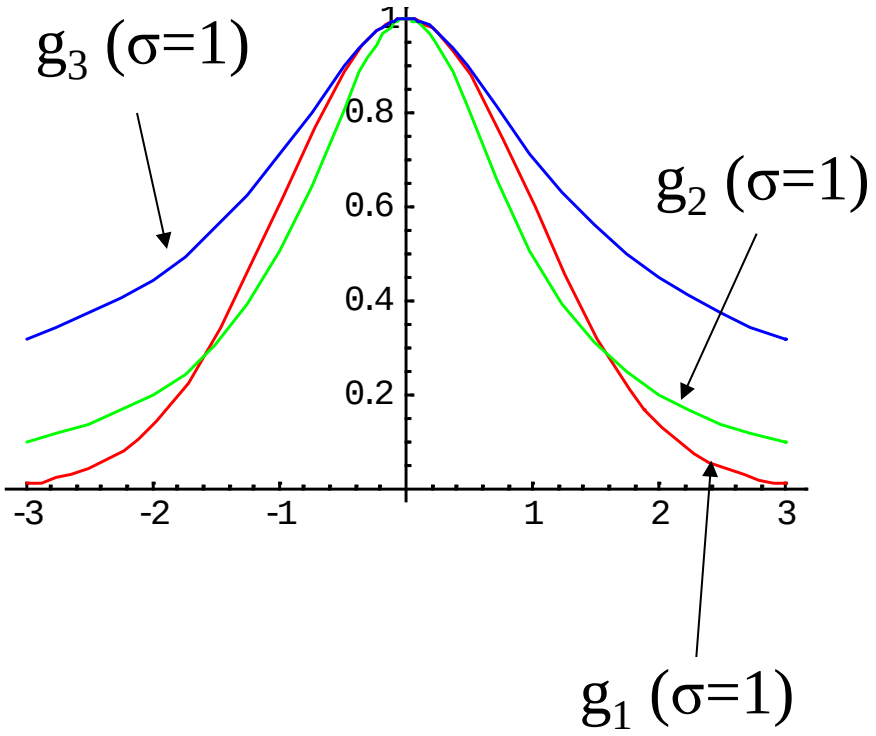
Rețele cu funcții radiale

Exemple de funcții radiale:

$$g_1(u) = \exp(-u^2/(2\sigma^2))$$

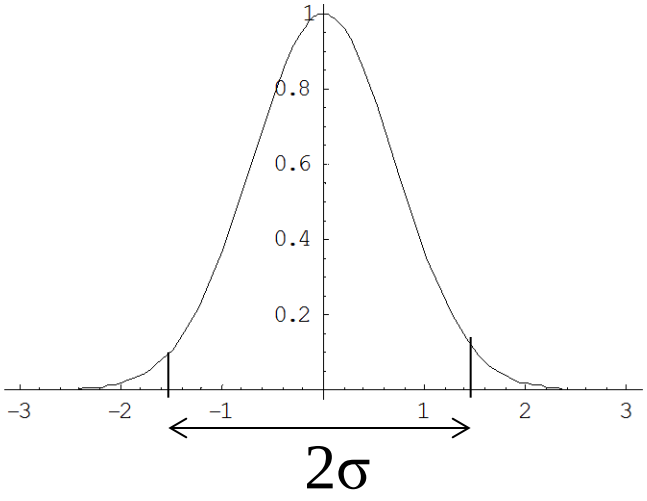
$$g_2(u) = 1/(u^2+\sigma^2)$$

$$g_3(u) = 1/\sqrt{u^2+\sigma^2}$$

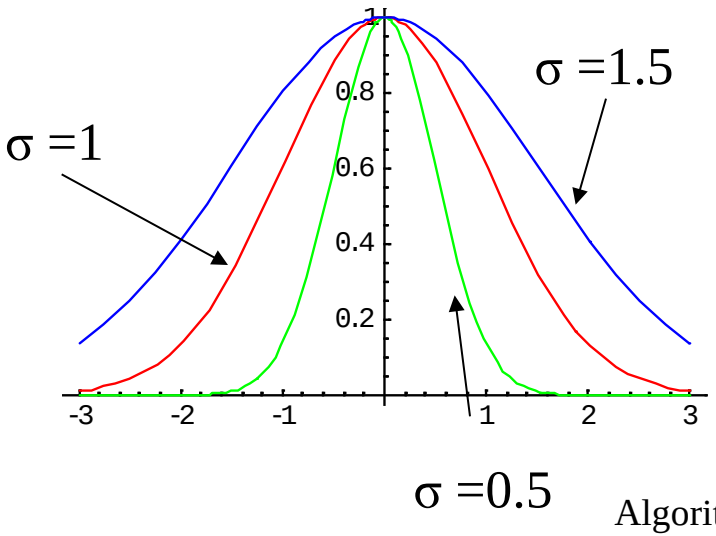


Rețele cu funcții radiale

- Fiecare unitate ascunsă este "sensibilă" la semnalele de intrare provenite dintr-o regiune a spațiului de intrare aflată în vecinatatea centrului. Aceasta regiune este denumită câmp receptiv
- Dimensiunea câmpului receptiv depinde de σ

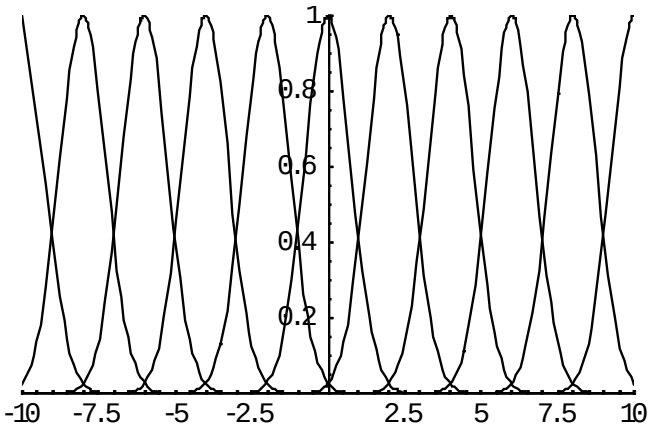


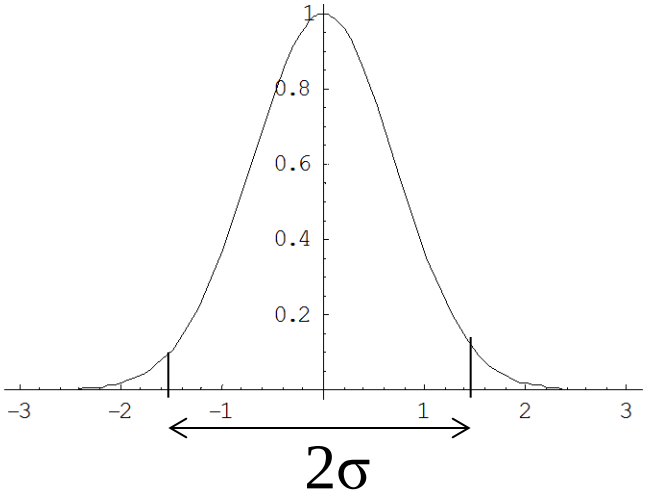
$$g(u) = \exp\left(-\frac{u^2}{2\sigma^2}\right)$$

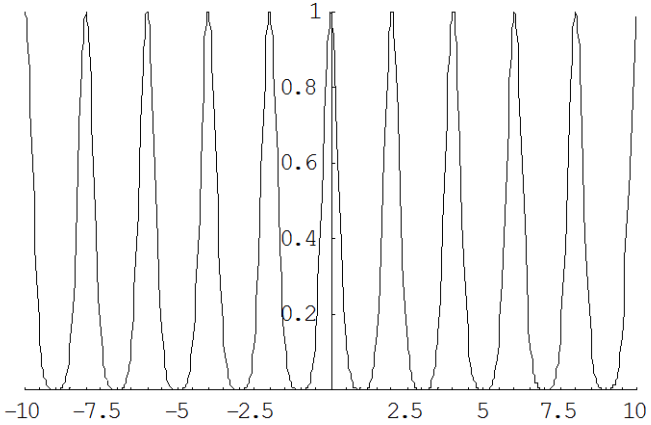


Rețele cu funcții radiale

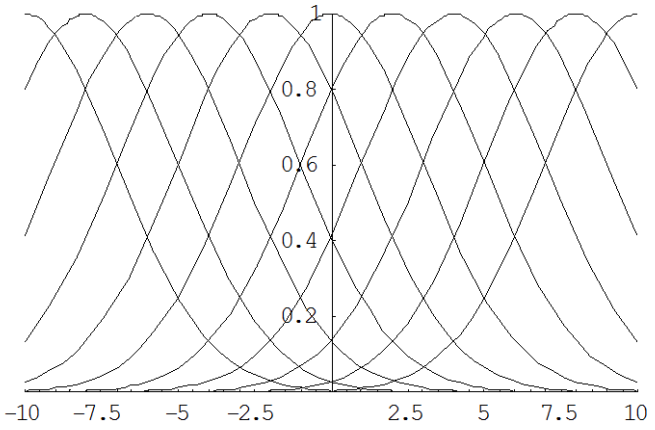
Influența lui σ: $g(u) = \exp\left(-\frac{u^2}{2\sigma^2}\right)$







subacoperire

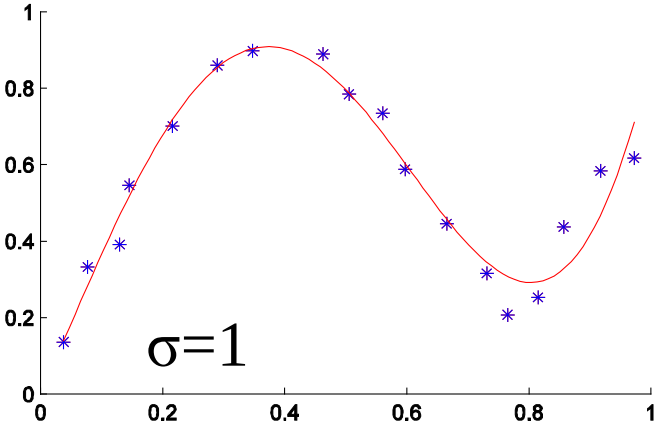


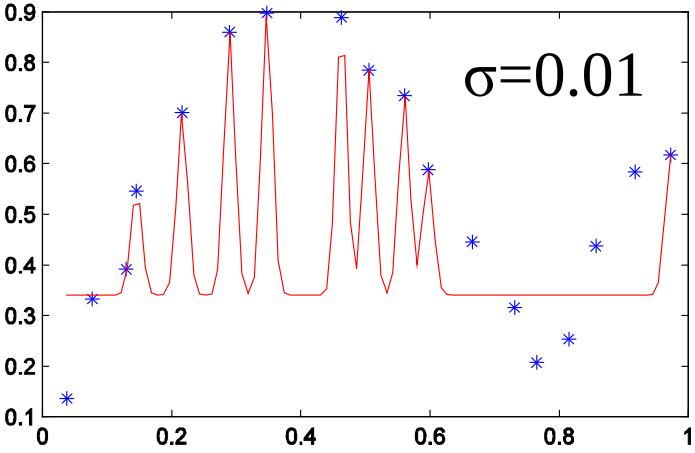
supraacoperire

Rețele cu funcții radiale

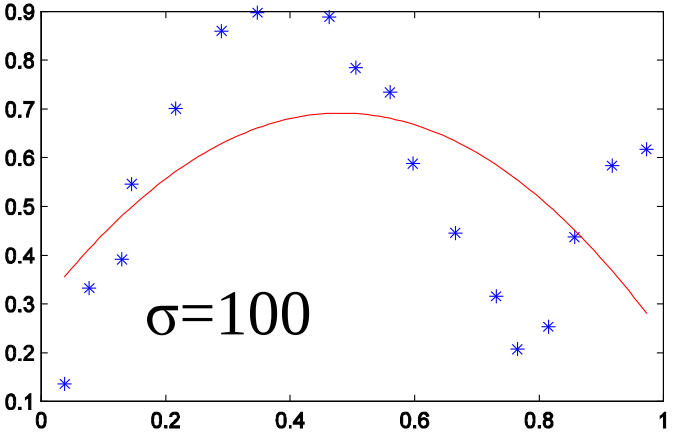
- O bună acoperire a domeniului datelor de intrare de către câmpurile receptive ale funcțiilor radiale de transfer este esențială pentru calitatea aproximării
- Valori prea mici conduc la incapacitatea de a produce rezultate pentru întreg domeniul datelor
- Valori prea mari nu surprind variabilitatea datelor

Acoperire adecvată





Subacoperire

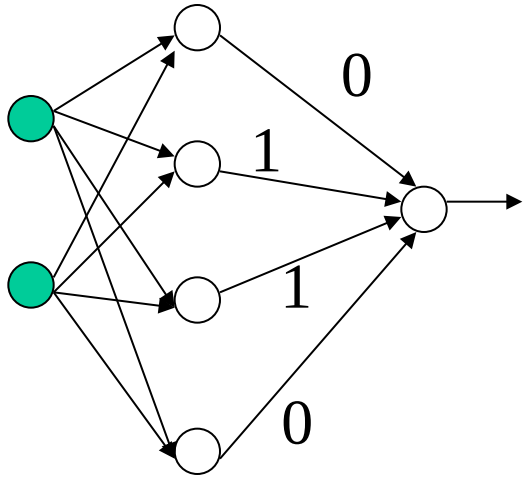


Supraacoperire

Rețele cu funcții radiale

Exemplu (caz particular) : rețea RBF pentru reprezentarea lui XOR

- 2 unități de intrare
- 4 unități ascunse
- 1 unitate de ieșire



Centrii:

u.a. 1: (0,0)

u.a. 2: (1,0)

u.a. 3: (0,1)

u.a. 4: (1,1)

Ponderi:

w1: 0

w2: 1

w3: 1

w4: 0

Funcție de activare:

g(u)=1 if u=0

g(u)=0 if u<>0

Aceasta abordare nu poate fi aplicată pentru probleme generale de aproximare

Rețele cu funcții radiale

Invățare:

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

Etape:

(a) Stabilirea parametrilor corespunzatori nivelului ascuns: centrii C și parametrii σ

(b) Determinarea parametrilor W (problemă de optimizare liniară)

Obs: Invățarea de tip RBF elimină o parte dintre dezavantajele algoritmului BP: convergența lentă, blocarea în minime locale (întrucât se ajunge la rezolvarea unei probleme mai simple de optimizare) etc.

Rețele cu funcții radiale

Invățare:

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

(a) Stabilirea parametrilor corespunzători nivelului ascuns: centrii C și parametrii σ

(a) K=L (nr centri = nr exemple), $C^k=x^k$

(b) K<L : centrii se stabilesc

(a) prin selecție aleatoare dintre exemplele din setul de antrenare

(b) prin selecție sistematică dintre exemplele din setul de antrenare (Orthogonal Least Squares)

(c) prin utilizarea unui algoritm de grupare (poate permite și estimarea numărului de centri) – in acest caz centrii nu vor face neapărat parte din setul de antrenare

Rețele cu funcții radiale

Orthogonal Least Squares:

- Selecție incrementală a centrilor astfel încât eroarea să fie micșorată cât mai mult
- Noul centru este ales astfel încât să fie ortogonal pe spațiul generat de către centrii deja selectați (procesul este bazat pe metoda de ortogonalizare Gram-Schmidt)
- Abordarea este corelată cu regresia de tip “ridge”

Rețele cu funcții radiale

Grupare (clustering):

•Se urmărește identificarea a K clase în setul de date de antrenare $\{X_1,\ldots,X_L\}$ astfel încât datele din fiecare clasă să fie suficient de similare pe când datele din clase diferite să fie suficient de diferite

•Fiecare clasă va avea un reprezentant (e.g. media datelor din clasă) care va fi considerat centrul clasei

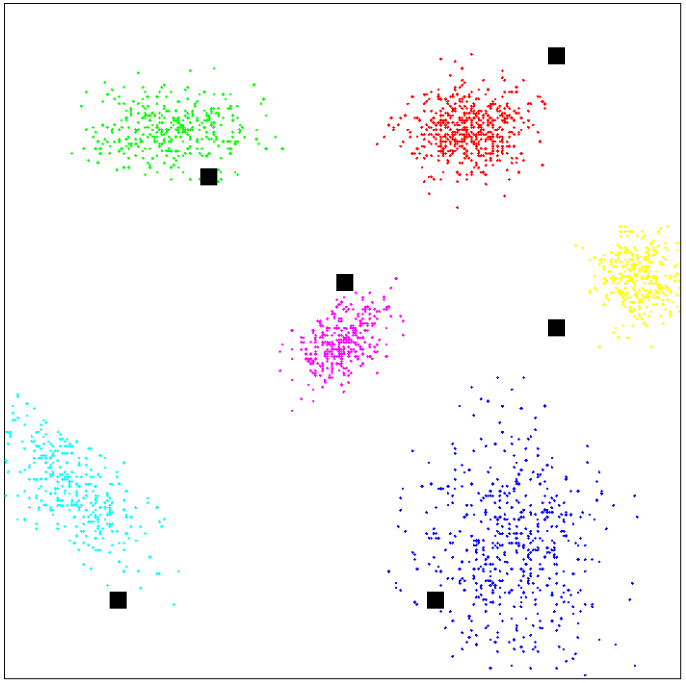
•Algoritmii pentru determinarea reprezentanților clasei sunt cunoscuți sub numele de algoritmi partiționali (realizează o partiționare a spațiului de intrare)

Algoritm clasic: K-means

Rețele cu funcții radiale

Algortitmul K-means:

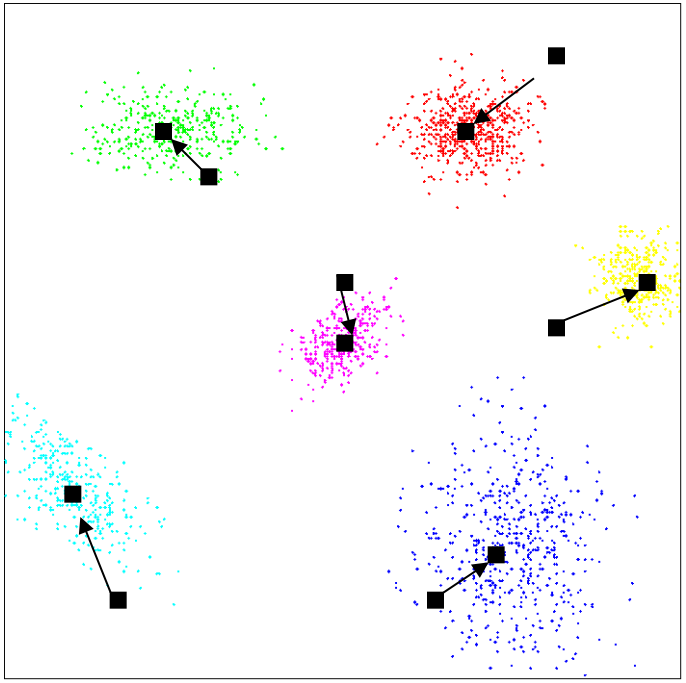
- Se pornește de la centri inițializați aleator
- Proces iterativ:
  - Se asignează datele la clase folosind criteriul distanței minime (sau a celui mai apropiat centru)
  - Se recalculează centrii ca fiind medii ale elementelor asignate fiecarei clase



Rețele cu funcții radiale

Algortitmul K-means:

- Se porneste de la centri initializati aleator
- Proces iterativ:
  - Se asigneaza datele la clase folosind criteriul distantei minime (sau a celui mai apropiat centru)
  - Se recalculeaza centrii ca fiind medii ale elementelor asignate fiecarei clase



Rețele cu funcții radiale

Algoritmul K-means:

C_k:=select({X_1,…,X_L}), k=1..K (centrii sunt initializati cu exemple selectate aleator din setul de antrenare)

REPEAT
  FOR l:=1,L
    Determină k(l) astfel încât d(X_l,Ck(l)) <=d(X_l,Ck)
    Asignează X_l clasei k(l)
  ENDFOR
  FOR k:=1,K
    C_k: = media elementelor ce au fost asignate clasei k
  ENDFOR
UNTIL “nu s-au mai efectuat modificări ale centrilor”

Obs:

- Centrii nu vor fi neapărat vectori din setul de antrenare
- Numărul de clase trebuie cunoscut de la început.

Rețele cu funcții radiale

Varianta incrementală:

- Se pornește cu un număr mic de centri inițializați aleator

- Se parcurge setul de antrenare:
  - Dacă există un centru suficient de similar cu data de intrare atunci componentele centrului respectiv se modifică pentru a asigura asimilarea datei de intrare în clasa aferentă centrului.
  - Dacă data de intrare este diferită semnificativ de toți centrii atunci este adăugat un nou centru (echivalent cu adăugarea unei noi unități ascunse) care este inițializat chiar cu data de intrare analizată

Obs: necesită definirea unor valor prag care sa permită cuantificarea pt suficient de similar/diferit

Rețele cu funcții radiale

Antrenare incrementală pentru rețele RBF

$K = K_0$

$C_i^k = select(\{X_1^i, \ldots, X_L^i\}), i = 1..N; k = 1..K$

$t = 0$

REPEAT

  FOR l = 1, L DO

    determina $k^* \in \{1, ..., K\}$ astfel incat $d(X^l, C^{k^*}) \leq d(X^l, C^k)$ pt orice k

    IF $d(X^l, C^{k^*}) < \delta$ THEN $C^{k^*} := C^{k^*} + \eta \cdot (X^l - C^{k^*})$

    ELSE $K = K + 1; C^K = X^l$

$t = t + 1$

$\eta = \eta_0 t^{-\alpha}$

UNTIL $t > t_{\max}$ OR $\eta < \varepsilon$

Rețele cu funcții radiale

Estimarea lărgimilor câmpurilor receptive.

- Reguli euristice:

$$\sigma = \frac{d_{\max}}{\sqrt{2K}},\ d_{\max} = \text{distanta maxima dintre centri}$$

$$\sigma_k = \gamma d(C^k, C^j), C^j = \text{centrul cel mai apropiat de } C^k, \gamma \in [0.5,1]$$

$$\sigma_k = \frac{1}{m}\sum_{j=1}^{m} d(C^k, C^j), C^1, ..., C^m : \text{cei mai apropiati } m \text{ centri de } C^j$$

  - Proces iterativ intercalat:
    - Fixează valorile σ și optimizează valorile centrilor
    - Fixează valorile centrilor și optimizează valorile σ

Rețele cu funcții radiale

Determinarea ponderilor conexiunilor dintre nivelul ascuns și nivelul de ieșire:

- Problema este similară cu cea a antrenării unei rețele cu un singur nivel de unități cu funcții liniare de activare

$$E(W) = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{M}\left(d_i^l - \sum_{k=1}^{K} w_{ik} g_k^l\right)^2, \quad g_k^l = g(\left\| x^l - C^k \right\|)$$

$$E(W) = \frac{1}{2}\sum_{l=1}^{L}\left\| d^l - Wg^l \right\|^2 = \frac{1}{2}\sum_{l=1}^{L}(d^l - Wg^l)^T(d^l - Wg^l)$$

$$\nabla E(W) = -\sum_{l=1}^{L}(g^l)^T(d^l - Wg^l) = 0$$

$$G^T G W = G^T d$$

$$W = (G^T G)^{-1} G^T d$$

Rețele cu funcții radiale

Determinarea ponderilor conexiunilor dintre nivelul ascuns și nivelul de ieșire:

- Problema este similară cu cea a antrenării unei rețele cu un singur nivel de unități cu funcții liniare de activare
- Algoritm: Widrow-Hoff (caz particular al algoritmului BackPropagation)

- Initializare:

  $w_{ij}(0):=rand(-1,1)$ (ponderile sunt inițializate aleator în [-1,1]),

  p:=0 (contor de iterații)

- Proces iterativ

```
REPEAT
    FOR l:=1,L DO
        Calculeaza y_i(l) si delta_i(l)=d_i(l)-y_i(l),  i=1,M
        Ajusteaza ponderile: w_ik:=w_ik+eta*delta_i(l)*z_k(l)
    ENDFOR
    Calculează E(W) pentru noile valori ale ponderilor
    p:=p+1
UNTIL E(W)<E* OR p>pmax
```

Comparație: RBF vs. BP

Rețele de tip RBF:

- 1 nivel ascuns
- Funcții de agregare bazate pe distanțe (pt. nivelul ascuns)
- Funcții de activare cu simetrie radială (pt. nivelul ascuns)
- Unități de ieșire cu funcție liniară
- Antrenare separată a parametrilor adaptivi
- Similare cu tehnicile de aproximare locală

Rețele de tip BackPropagation(BP):

- Mai multe nivele ascunse
- Funcții de agregare bazate pe suma ponderată
- Funcții de activare sigmoidale (pt. nivelul ascuns)
- Unități de ieșire liniare sau neliniare
- Antrenare simultană a parametrilor adaptivi
- Similare cu tehnicile de aproximare globală

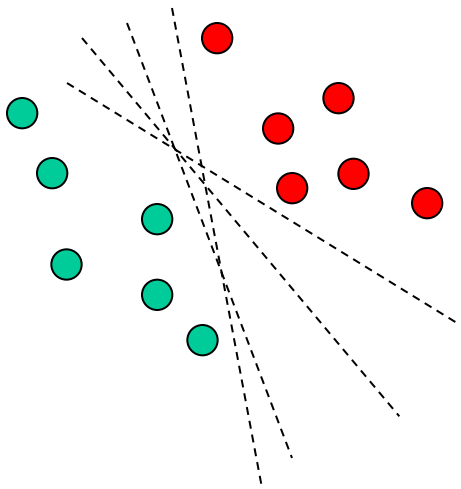
Support Vector Machines

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

- Antrenare bazată pe o metodă de optimizare a funcțiilor pătratice care 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 învățare (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



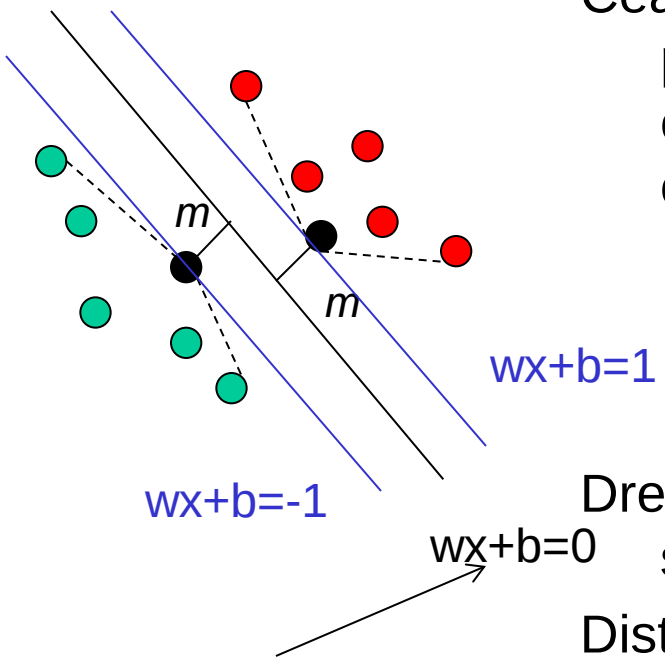
Problema e liniar separabilă și se observă că există o infinitate de drepte (hiperplane, în cazul general) care permit separarea celor două clase

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 ?



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

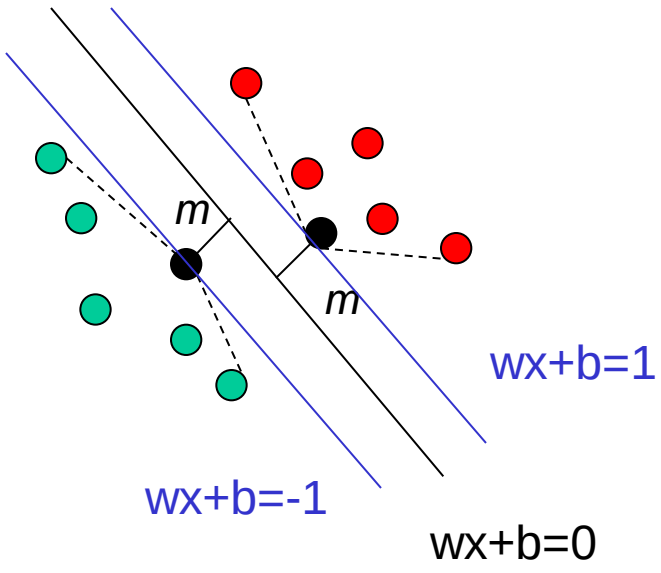
Dreptele care trec prin punctele marginale sunt considerate drepte canonice

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

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

pentru toate elementele setului de antrenare $\{(x_1,y_1),(x_2,y_2),\ldots,(x_L,y_L)\}$

$y_i=-1$ pentru clasa verde

$y_i=1$ 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 $(wx_i+b)y_i-1>=0$ pentru i=1..L

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

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

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

Construirea funcției duale:

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

$$\frac{\partial V(w,b,\alpha)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i \qquad \frac{\partial V(w,b,\alpha)}{\partial b} = 0 \Rightarrow 0 = \sum_{i=1}^{L} \alpha_i y_i$$

Support Vector Machines

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

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

Cu restricțiile: (cunoscute din setul de antrenare)

$$\alpha_i \geq 0, \quad \sum_{i=1}^{L} \alpha_i y_i = 0$$

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

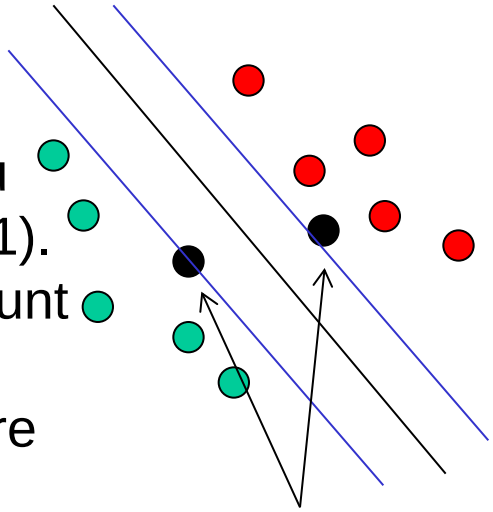
$$w^* = \sum_{i=1}^{L} \alpha_i y_i x_i, \quad b^* = 1 - w \cdot x_k$$

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

Support Vector Machines

Observații:

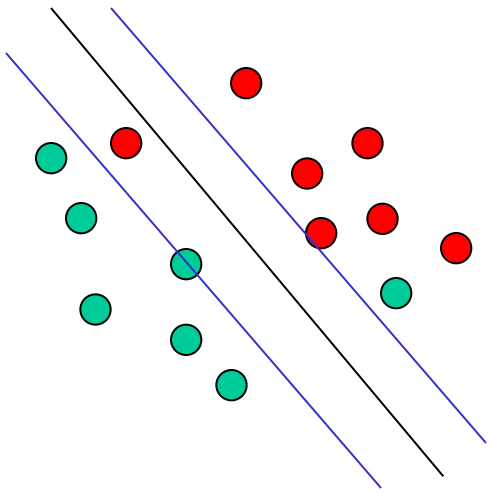
- Multiplicatorii nenuli corespund exemplelor pentru care restricțiile sunt active (w x+b=1 sau w x+b=-1). Aceste exemple sunt denumite vectori suport și sunt singurele care influențează ecuația hiperplanului separator (celelalte exemple din setul de antrenare pot fi modificate fără a influența hiperplanul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



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

Support Vector Machines

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



Se relaxează condiția de apartenență la o clasă:

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

Funcția de minimizat devine:

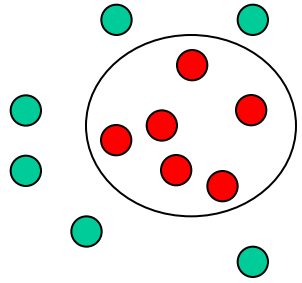
$$V(w, b, \alpha, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\alpha_i(y_i(w \cdot x_i + b) - 1)$$

Ceea ce schimbă restricțiile din problema duală astfel:

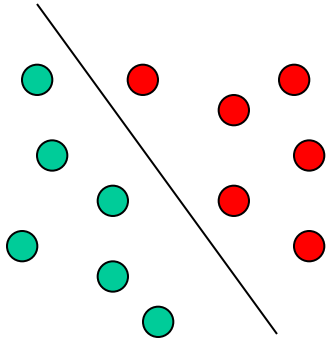
$$\text{in loc de } \alpha_i \geq 0 \text{ se introduce } 0 \leq \alpha_i \leq C$$

Support Vector Machines

Ce se întâmplă în cazul in care problema este neliniar separabilă?



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



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

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

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

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

Support Vector Machines

In cazul general se aplică transformarea:

$x \to \theta(x)$ iar produsul scalar al vectorilor transformati este

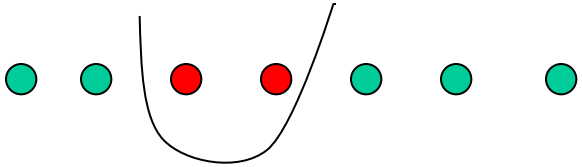
$\theta(x) \cdot \theta(x') = K(x, x')$

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

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



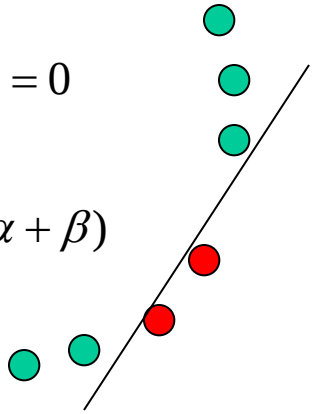
Pb. 1-dimensională neliniar separabilă

$$w_1 z_1 + w_2 z_2 + b = 0$$

$$z_1 = x^2, z_2 = x$$

$$w_1 = 1, w_2 = -(\alpha+\beta)$$

$$b = \alpha\beta$$



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)

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

$$K(x, x') = \theta(x_1, x_2) \cdot \theta(x'_1, x'_2) = (x \cdot x' + 1)^2$$

Support Vector Machines

Exemple de functii nucleu:

$$K(x, x') = (x \cdot x' + 1)^d$$

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

$$K(x, x') = \tanh(kx \cdot x' + b)$$

Functia de decizie devine:

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

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

Spider [http://www.kyb.tue.mpg.de/bs/people/spider/tutorial.html]: implementare Matlab

Interfață SciLab pt LibSVM (http://atoms.scilab.org/toolboxes/libsvm)