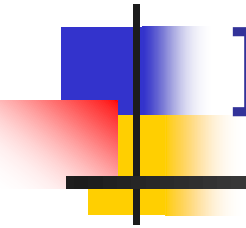


Inteligență artificială

12. Rețele neuronale (II)

Florin Leon

Universitatea Tehnică „Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare

http://florinleon.byethost24.com/curs_ia.html

Rețele neuronale (II)

1. Mașini cu vectori suport
2. Metodologia de antrenare
3. Învățarea hebbiană
4. Concluzii

Rețele neuronale (II)

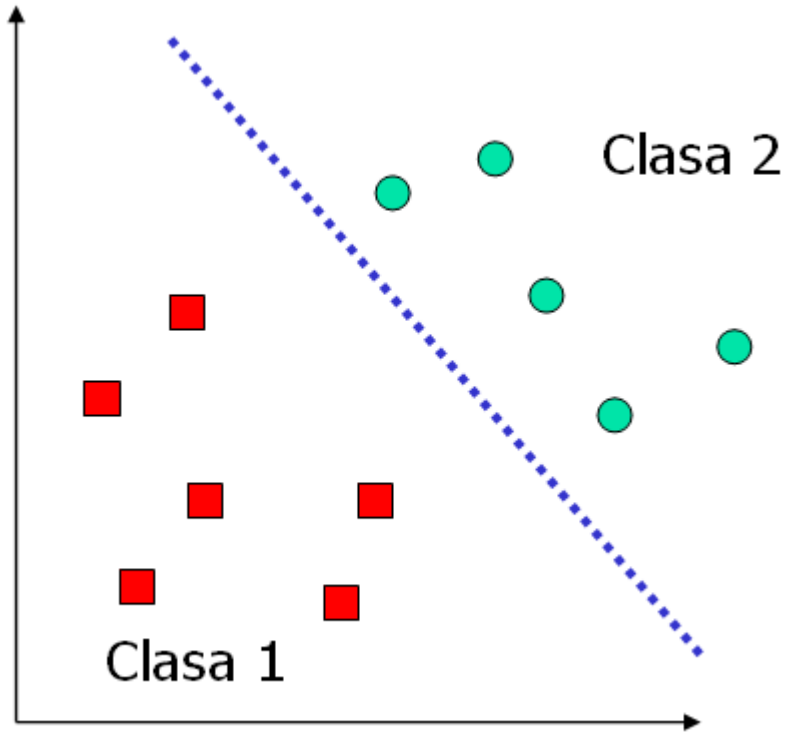
1. Mașini cu vectori suport
2. Metodologia de antrenare
3. Învățarea hebbiană
4. Concluzii

Mașini cu vectori suport

- engl. “Support Vector Machines”, SVM
  - Boser, Guyon & Vapnik (1992)
  - Vapnik (1995)
- Devenite populare datorită succesului în recunoașterea cifrelor scrise de mână
  - Eroare de 1.1% pe mulțimea de test
- Fundament teoretic solid
- Performanțe experimentale foarte bune
  - Mașinile cu vectori suport reprezintă una dintre cele mai bune metode de învățare din generația anterioară învățării profunde

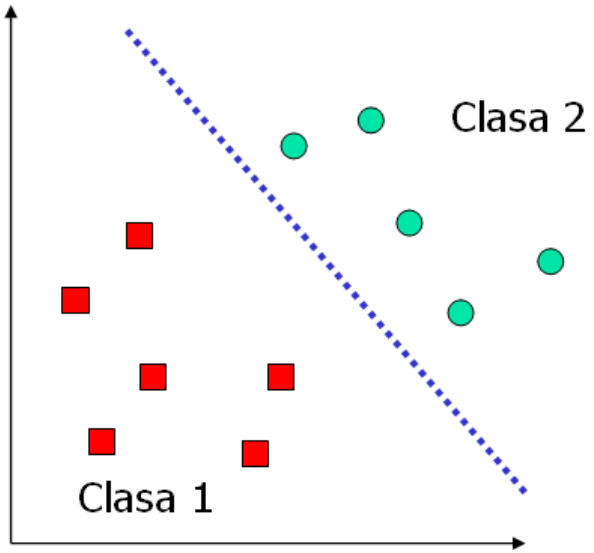
Limitele suprafețelor de decizie

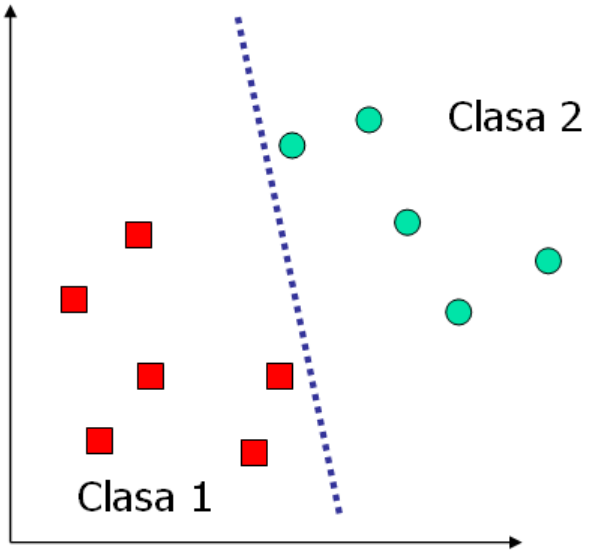
- Să considerăm o problemă de clasificare liniar separabilă cu două clase
  - Există multe limite de separație posibile
  - Algoritmul de învățare al perceptronului poate găsi astfel de limite



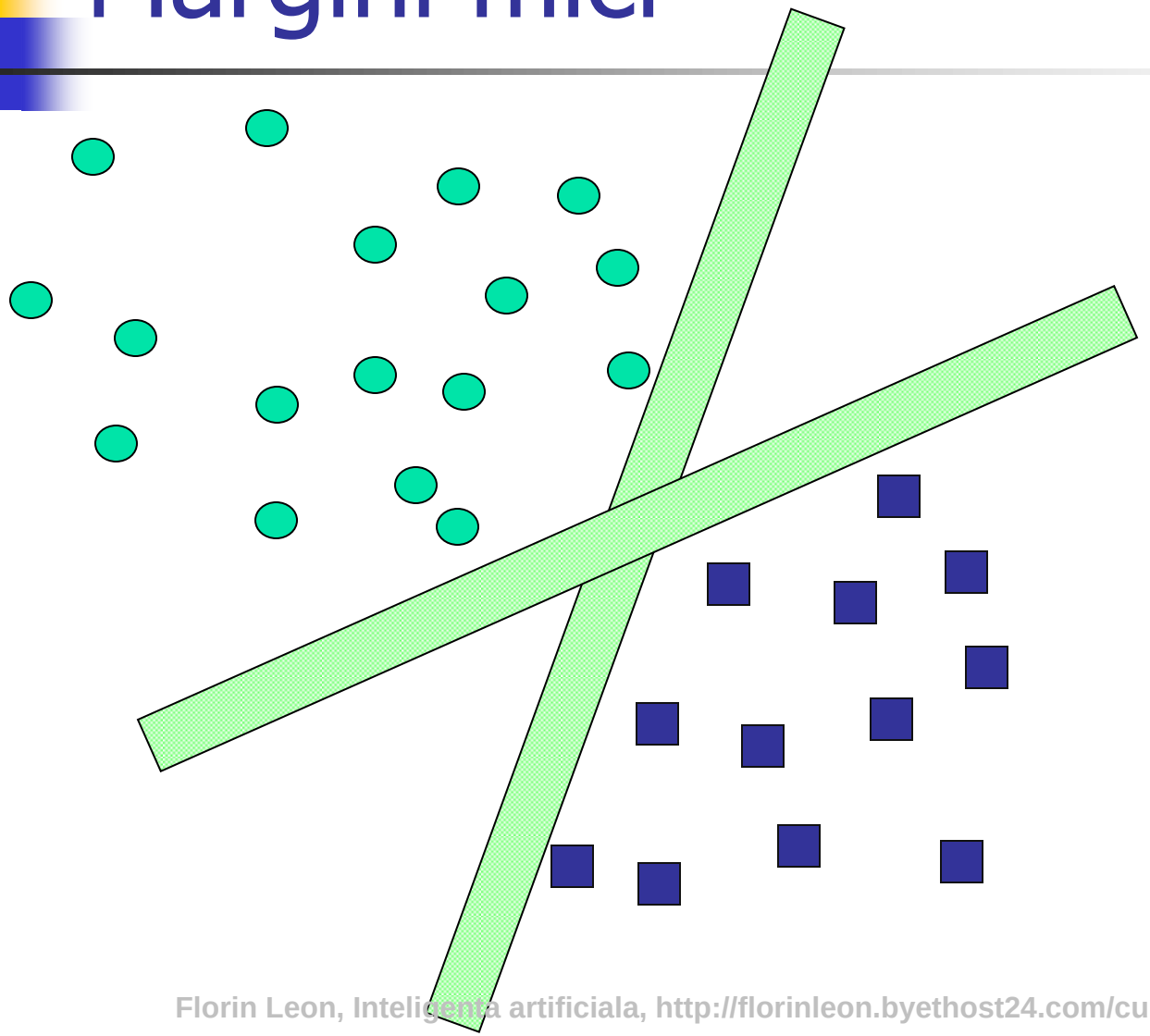
Limitele suprafețelor de decizie

- Pot exista mai multe limite de separație
- Care este cea mai bună?

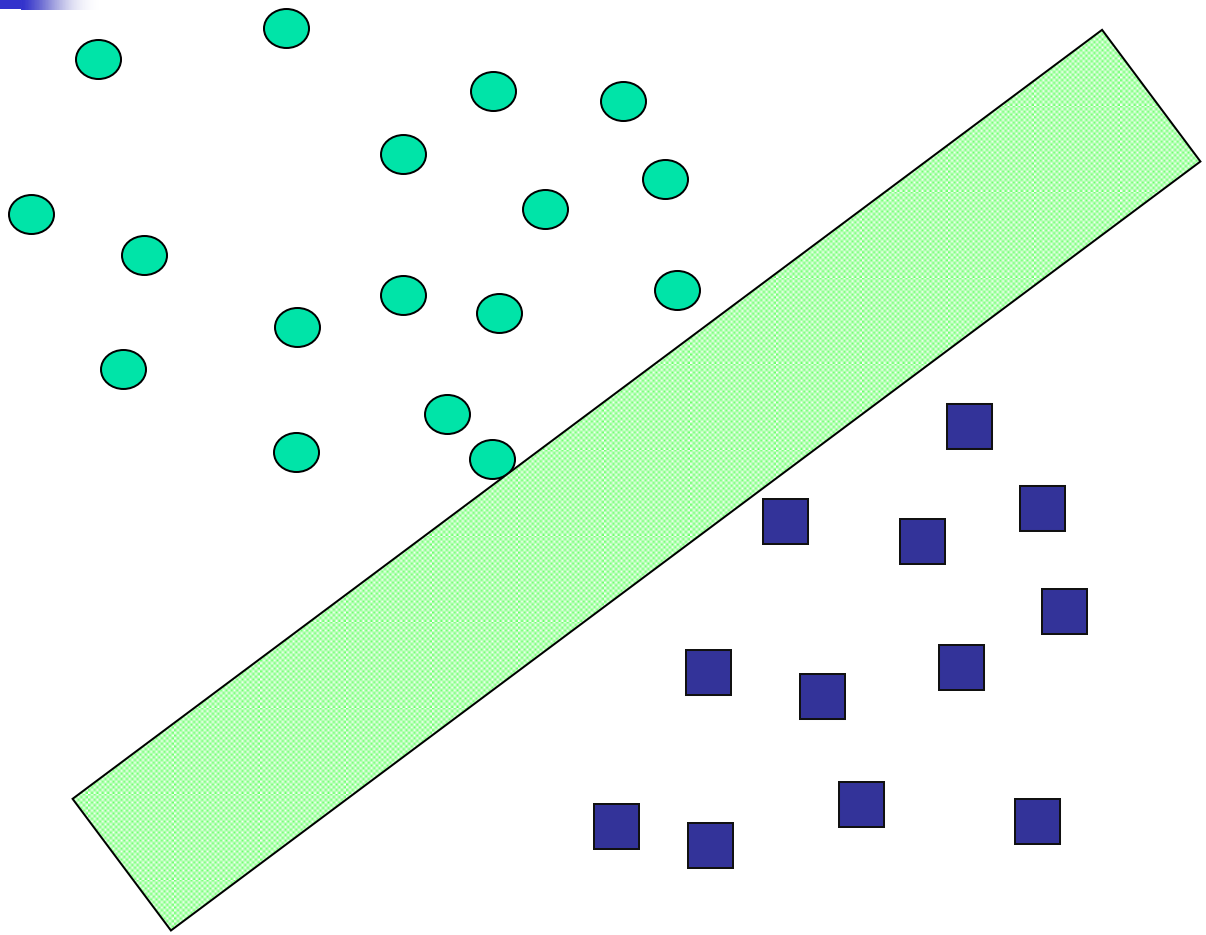




Margini mici



Margine mare

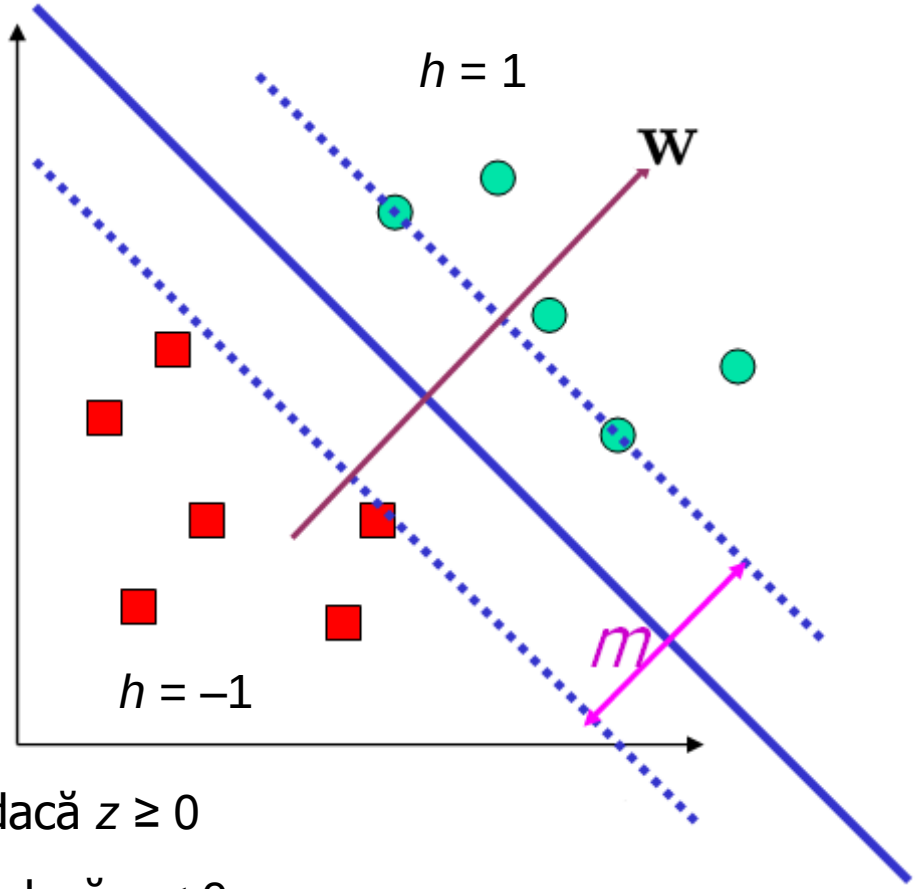


Noțiuni importante

- Fie vectorii $\mathbf{x} = (x_1, ..., x_n)$ și $\mathbf{y} = (y_1, ..., y_n)$
- Produs scalar, notat $\mathbf{x}^T\mathbf{y}$, $\mathbf{x} \cdot \mathbf{y}$ sau $\langle \mathbf{x}, \mathbf{y} \rangle$
  - $\mathbf{x}^T\mathbf{y} = x_1 \cdot y_1 + x_2 \cdot y_2 + ... + x_n \cdot y_n$
- Normă euclidiană
  - $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$

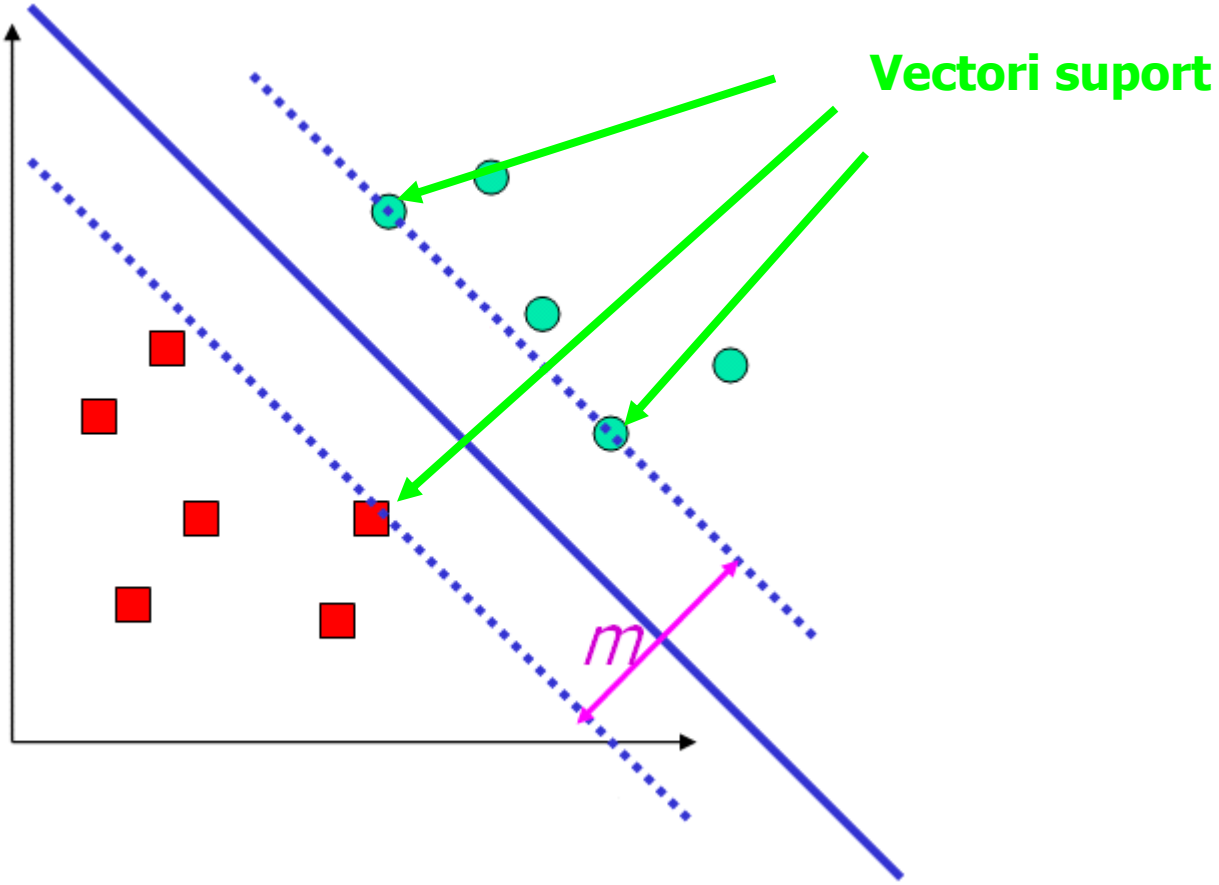
Limite de decizie cu margini mari

- Limita de decizie trebuie să fie cât mai departe de datele din ambele clase
- Marginea *m* trebuie maximizată



$$h(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + b) \qquad g(z) = \begin{cases} 1, \text{ dacă } z \geq 0 \\ -1, \text{ dacă } z < 0 \end{cases}$$

Vectori suport



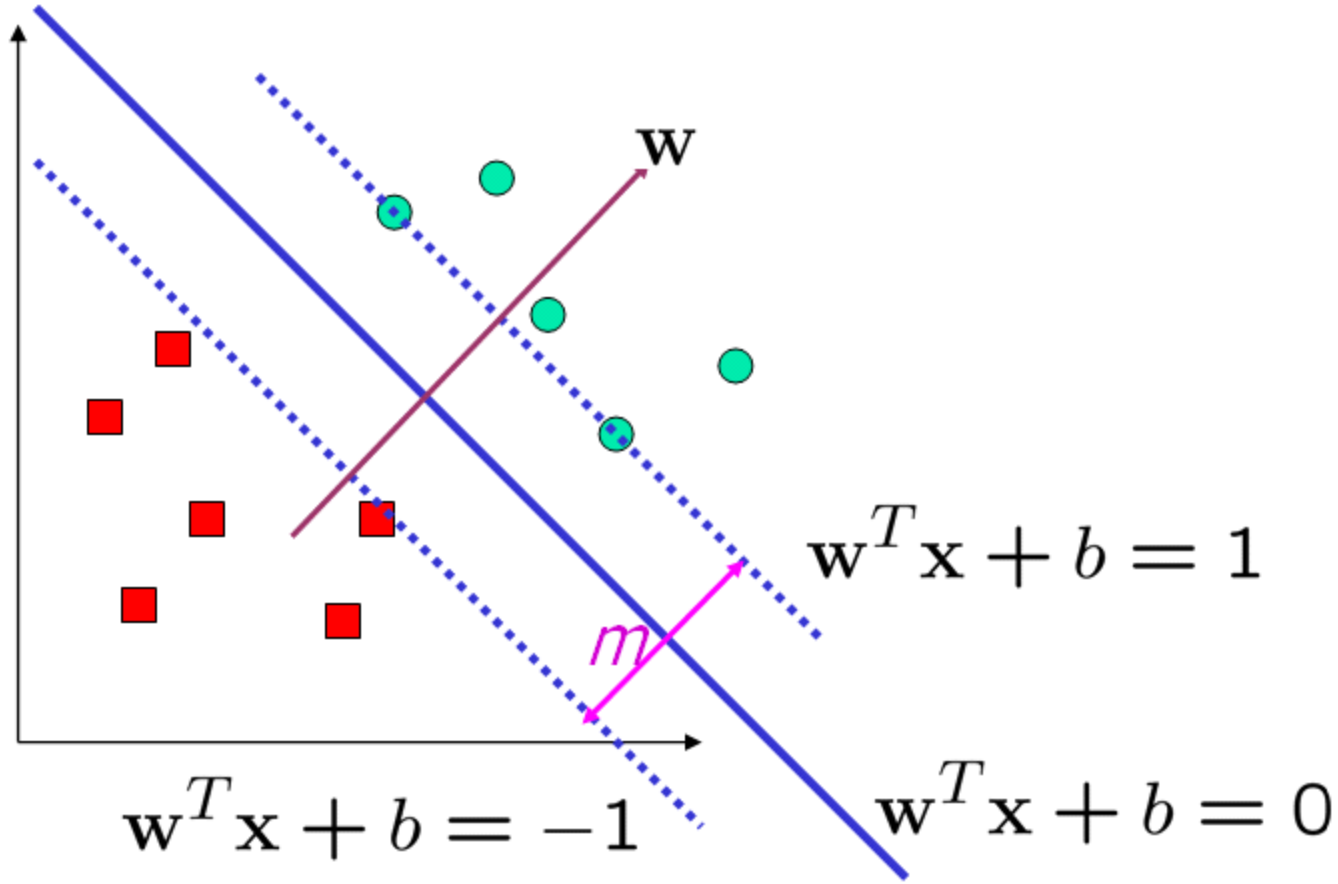
Normalizarea

de exemplu: $w_1 \cdot x_1 + w_2 \cdot x_2 + b$

- $h(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + b)$; $g(z)$ este –1 sau 1
- Doar semnul lui $(\mathbf{w}^T\mathbf{x} + b)$ contează, nu și valoarea
- Putem normaliza ecuațiile (constrângerile) astfel ca în limitele care trec prin vectorii suport ai celor două clase, $\mathbf{w}^T\mathbf{x} + b$ să fie 1, respectiv –1
- Pentru toate punctele celor două clase:
  - $\mathbf{w}^T\mathbf{x}_i + b \geq 1$ dacă $y_i = 1$
  - $\mathbf{w}^T\mathbf{x}_i + b \leq -1$ dacă $y_i = -1$

y_i este clasa instanței i

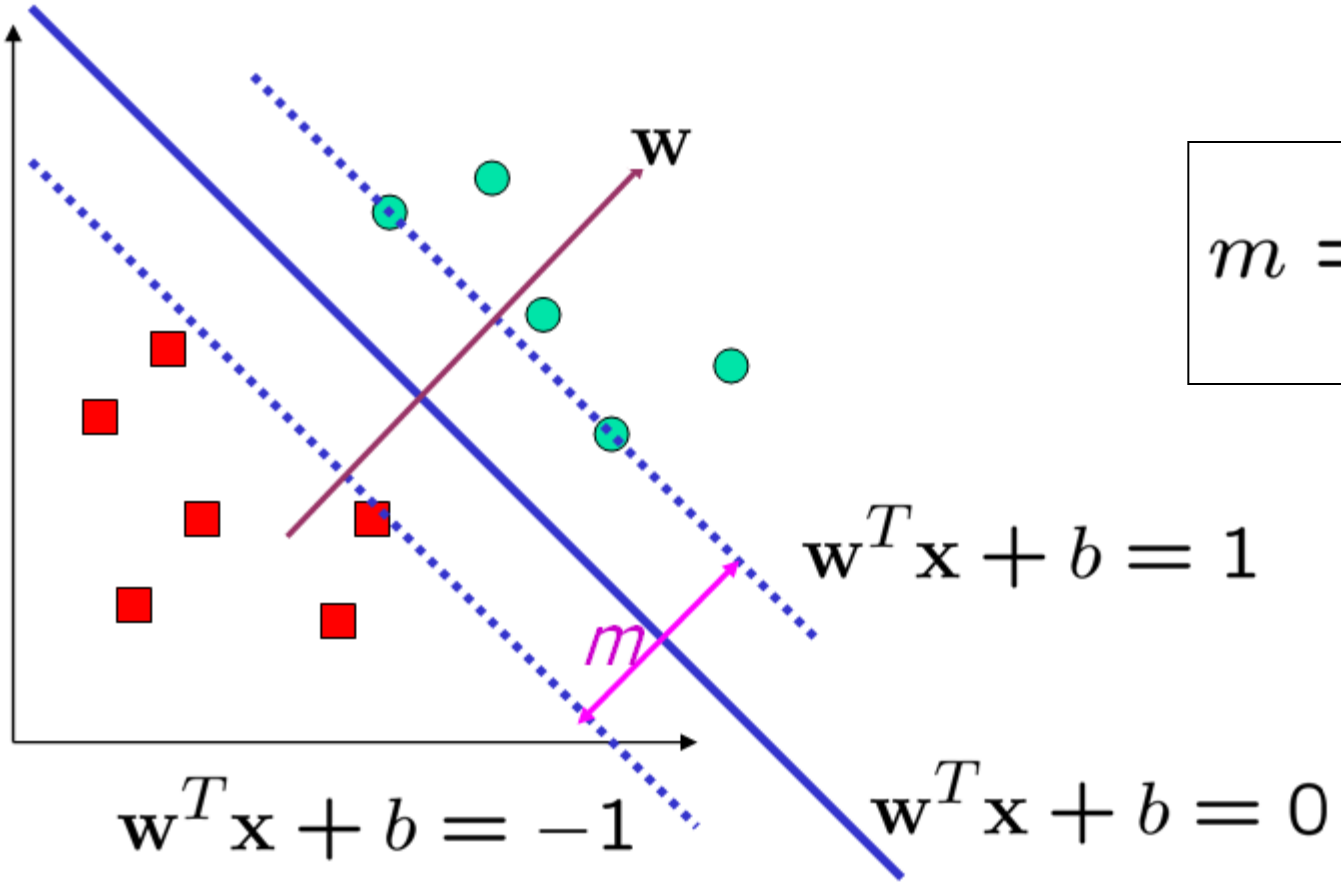
Normalizarea



Marginea

- Marginea *m* este proiecția distanței dintre doi vectori suport $\mathbf{x}_1$ și $\mathbf{x}_2$ pe direcția vectorului $\mathbf{w}$
- Valoarea lui *m* se poate obține, de exemplu, scăzând ecuațiile lui $\mathbf{x}_1$ și $\mathbf{x}_2$
- $\mathbf{w} \cdot \mathbf{x}_1 + b = -1$
- $\mathbf{w} \cdot \mathbf{x}_2 + b = 1$
- $\mathbf{w} \cdot (\mathbf{x}_2 - \mathbf{x}_1) = 2$
- $m = \mathbf{w} \cdot (\mathbf{x}_2 - \mathbf{x}_1) / \|\mathbf{w}\| = 2 / \|\mathbf{w}\|$

Marginea



$$m = \frac{2}{||\mathbf{w}||}$$

Determinarea marginii

- Trebuie determinați **w** și *b* astfel încât marginea $m = 2 / \|\mathbf{w}\|$ să fie maximă pentru toate instanțele $\{ \mathbf{x}_i , y_i \}$
- respectând constrângerile:
  - $\mathbf{w}^T\mathbf{x}_i + b \geq 1$ dacă $y_i = 1$
  - $\mathbf{w}^T\mathbf{x}_i + b \leq -1$ dacă $y_i = -1$
- adică, mai concis:
  - $y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Determinarea marginii

- O formulare mai bună a problemei de optimizare:
  - Minimizarea lui ½ $\|\mathbf{w}\|^2$, respectând aceleași constrângeri
- Problemă de optimizare cuadratică

$$\min_{\gamma,w,b} \quad \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m$$

- S-ar putea rezolva cu metode tipice de optimizare, inclusiv cu algoritmi evolutivi, deși constrângerile necesită atenție suplimentară

Determinarea marginii

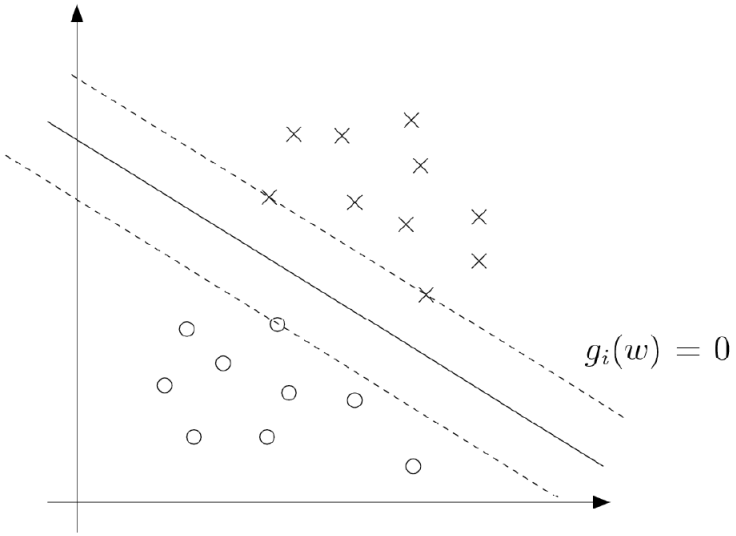
- Dar...
- SVM este în general utilizat pentru clasificarea unor date cu foarte multe dimensiuni
  - De exemplu, în clasificarea textelor sau detecția *spam*-ului, frecvențele de apariție ale cuvintelor sunt atributele
  - Posibil mii de atribute
  - Teoretic, **w** poate fi infinit dimensional

Problema primară

$$\min_{\gamma,w,b} \quad \frac{1}{2}||w||^2$$

$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m$$

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$$



Dualitatea Lagrange

- Rezolvarea problemei anterioare de optimizare (problema primară) este echivalentă cu rezolvarea problemei duale

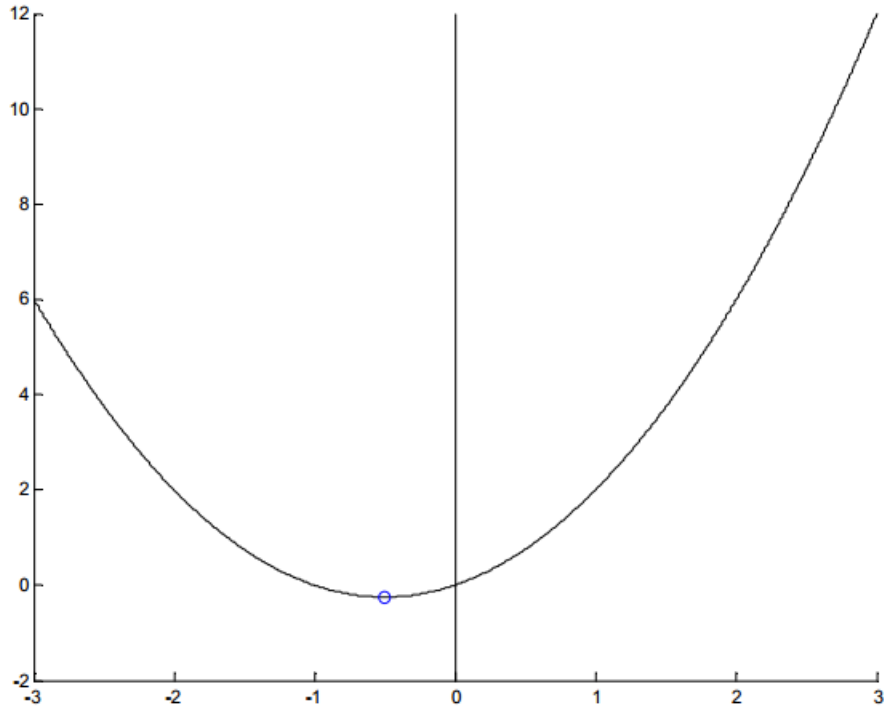
$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \ldots, k \\ & h_i(w) = 0, \quad i = 1, \ldots, l. \end{aligned}$$

$$\Rightarrow \quad \mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

Lagrangian

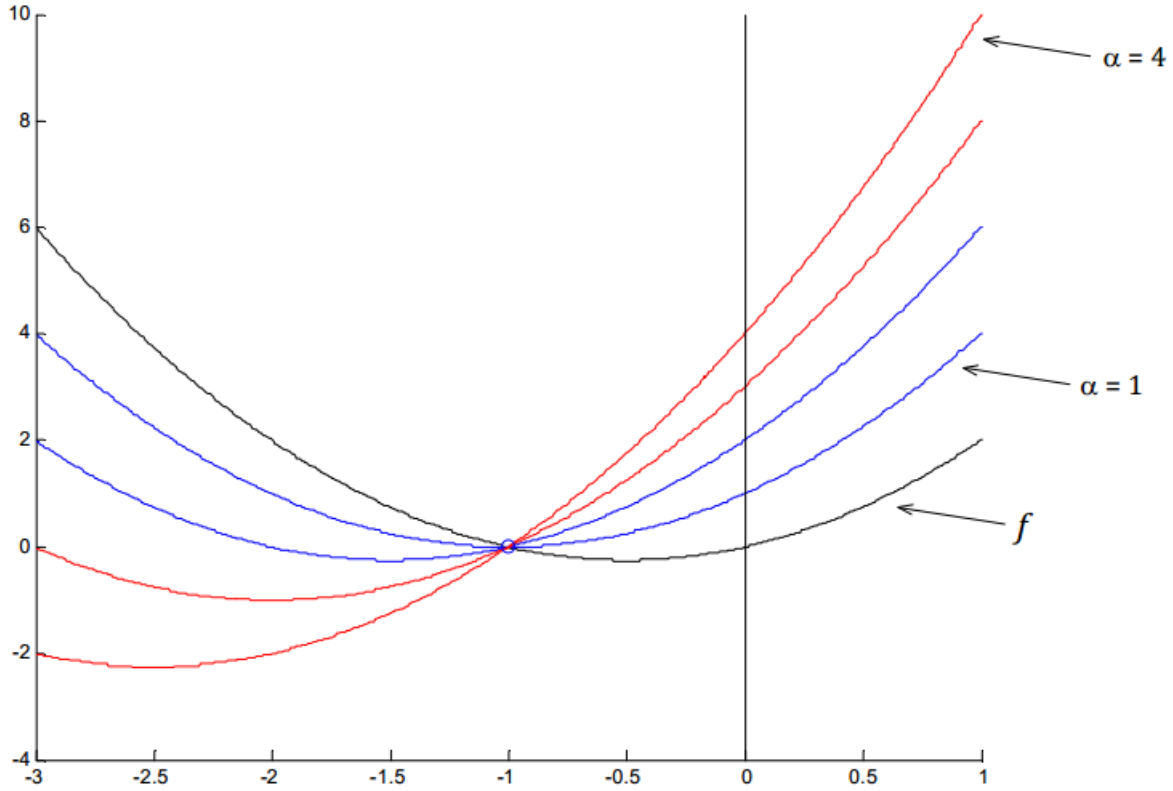
multiplicatori lagrangieni

Exemplu



Minimul funcţiei $f(x) = x^2 + x$ *în absenţa constrângerilor*

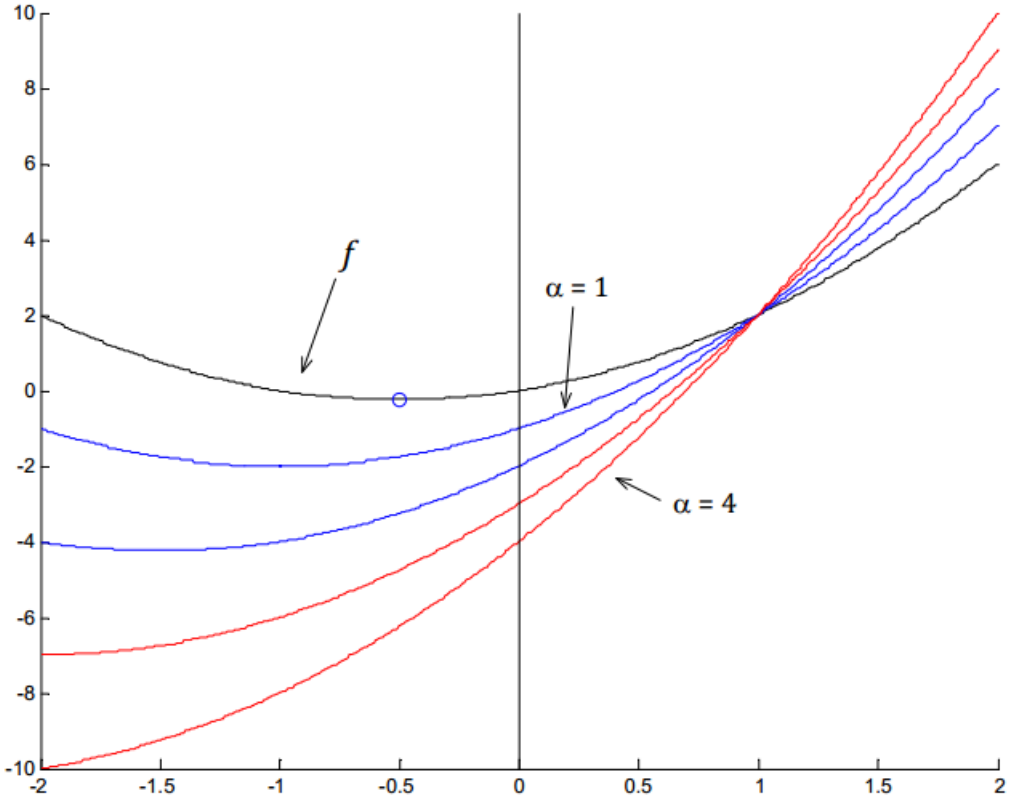
Constrângere activă: $\alpha > 0$



Familie de funcţii lagrangiene pentru funcţia f şi constrângerea $x + 1 \leq 0$

$$\min_x \Theta_P = \min_x \max_\alpha L(x, \alpha)$$

Constrângere inactivă: $\alpha = 0$



Familie de funcţii lagrangiene pentru funcţia f

şi constrângerea $x - 1 \leq 0$

Problema duală

- Problema duală este în general mai ușor de rezolvat decât problema primară

$$\min_x \Theta_P = \min_x \max_\alpha L(x, \alpha)$$

$$\max_\alpha \Theta_D = \max_\alpha \min_x L(x, \alpha)$$

$$\max_\alpha \min_x L(x, \alpha) \leq \min_x \max_\alpha L(x, \alpha)$$

$$\max_\alpha \Theta_D = \max_\alpha \min_x L(x, \alpha) = \min_x \max_\alpha L(x, \alpha) = \min_x \Theta_P$$

dualitate puternică dacă funcția este convexă

Condițiile Karush-Kuhn-Tucker

- Există **w***, **α*** și **β*** (soluțiile problemei primare, respectiv duale), astfel încât:

$$\begin{aligned}
\frac{\partial}{\partial w_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \ldots, n \\
\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \ldots, l \\
\alpha_i^* g_i(w^*) &= 0, \quad i = 1, \ldots, k \\
g_i(w^*) &\leq 0, \quad i = 1, \ldots, k \\
\alpha^* &\geq 0, \quad i = 1, \ldots, k
\end{aligned}$$

Formularea problemei duale

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)}(w^T x^{(i)} + b) - 1 \right]$$

- Aplicăm condițiile Karush-Kuhn-Tucker (diferențialele în raport cu **w** și *b* să fie 0):

$$\frac{\partial}{\partial w}\mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0 \Rightarrow w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

$$\frac{\partial}{\partial b}\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Formularea problemei duale

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)}(w^T x^{(i)} + b) - 1 \right]$$

- Înlocuind aceste relații în formula de mai sus, vom avea:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

sumă dublă: după *i*, respectiv după *j*

Determinarea parametrilor **w** și *b*

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

- Rezolvând problema duală, se determină α_i iar apoi se calculează **w** și *b*

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

Determinarea parametrilor **w** și *b*

- Pentru probleme separabile liniar

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + \min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}$$

- Pentru cazul general (situațiile pe care le vom prezenta în continuare)

$$b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - \sum_{t \in S} \alpha_t y_t (\mathbf{x}_t \cdot \mathbf{x}_s) \right)$$

unde S este mulţimea vectorilor suport iar $|S|$ este numărul lor

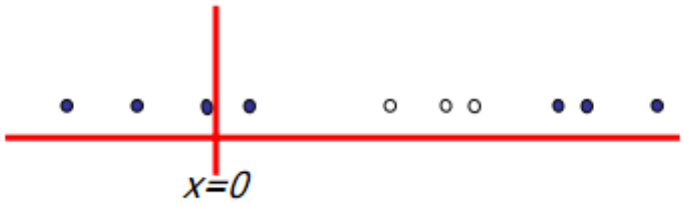
Avantaj

- Prin rezolvarea problemei duale, se determină $\alpha_i \geq 0$
- De fapt, toți α_i sunt 0 cu excepția multiplicatorilor corespunzători vectorilor suport
- Numărul vectorilor suport este mult mai mic decât numărul instanțelor: $n_{vs} \ll n$

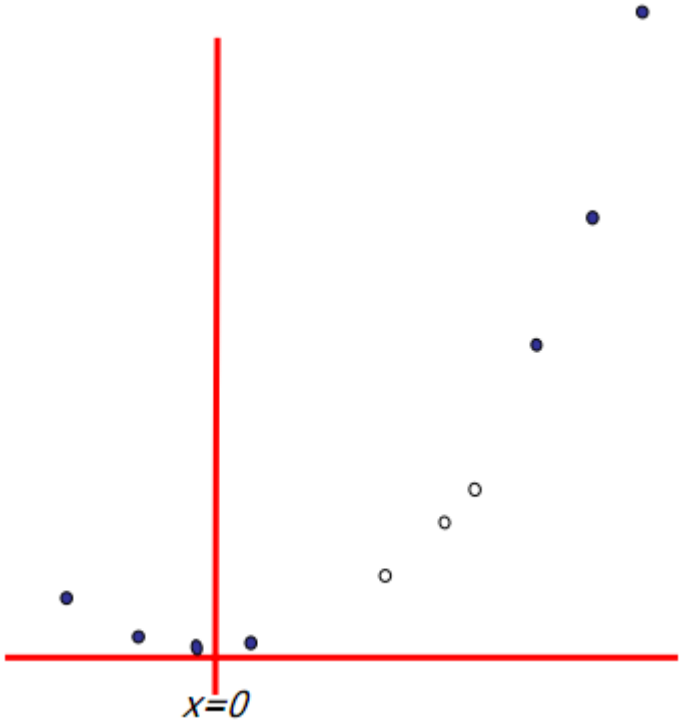
Dimensiuni

- **w** are o dimensiune egală cu numărul de atribute ale problemei (asemănător ponderilor unui perceptron)
- *b* este un număr real (asemănător pragului)
- Numărul de multiplicatori α_i este egal cu numărul instanțelor de antrenare, la fel ca $\mathbf{x}_i$ și y_i
- α_i este un număr real pozitiv; majoritatea α_i sunt 0
- $\mathbf{x}_i$ este un vector de numere reale, cu dimensiunea egală cu numărul de atribute ale problemei
- y_i este 1 sau -1 (clasa instanței *i*)

Transformarea datelor



$z_k = (x_k, x_k^2)$

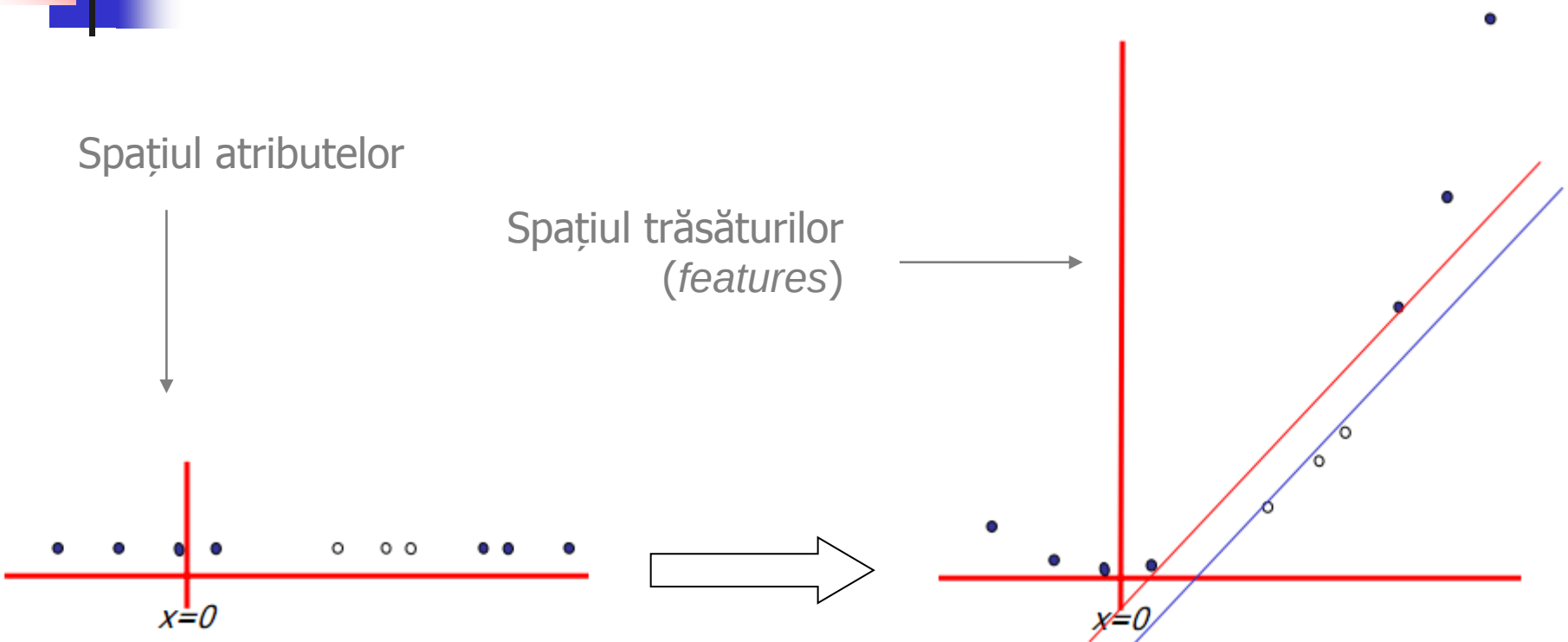


Clasele nu sunt liniar separabile

Transformarea datelor

Spațiul atributelor

Spațiul trăsăturilor
(*features*)



Clasele nu sunt liniar separabile

Clasele devin liniar separabile într-un spațiu cu mai multe dimensiuni

Nuclee

- Transformarea în trăsături (*feature mapping*)

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

- Nucleu (*kernel*)

$$K(x, z) = \phi(x)^T \phi(z)$$

Trucul nucleului

- engl. “kernel trick”
- Cantitatea necesară pentru clasificare este:

$$w^T x + b = \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b$$

$$= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b.$$

- Folosind un nucleu:

Calculele depind doar de perechi ($\mathbf{x}_1$, $\mathbf{x}_2$)

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}\right) + b$$

Trucul nucleului

$$K(x,z) = (x^T z)^2.$$

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_i z_i\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

Trucul nucleului

$$
\begin{aligned}
K(x,z) &= (x^T z + c)^2 \\
&= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) + \sum_{i=1}^{n} (\sqrt{2c}x_i)(\sqrt{2c}z_i) + c^2.
\end{aligned}
$$

$$
\phi(x) = \begin{bmatrix}
x_1 x_1 \\
x_1 x_2 \\
x_1 x_3 \\
x_2 x_1 \\
x_2 x_2 \\
x_2 x_3 \\
x_3 x_1 \\
x_3 x_2 \\
x_3 x_3 \\
\sqrt{2c}x_1 \\
\sqrt{2c}x_2 \\
\sqrt{2c}x_3 \\
c
\end{bmatrix}
$$

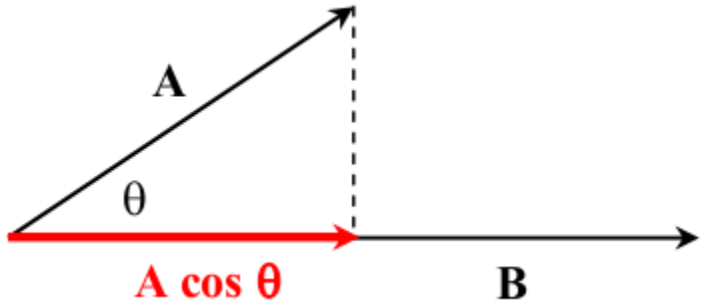
- Deoarece calculele se fac doar în perechi, nu este nevoie să calculăm explicit trăsăturile instanțelor
- Calcularea nucleului unei perechi de instanțe este de obicei mult mai simplă

Nuclee uzuale

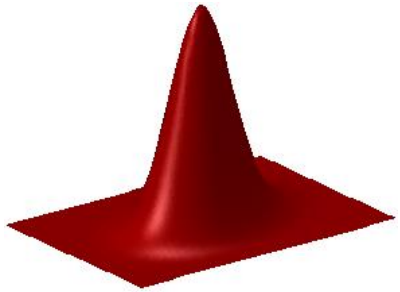
- Nucleul liniar $K(x, z) = x^T z$
- Nucleul polinomial $K(x, z) = (\gamma x^T z + r)^d$
- Nucleul gaussian $K(x, z) = \mathrm{e}^{-\gamma \|x-z\|^2}$
- Nucleul sigmoid $K(x, z) = \tanh(\gamma x^T z + r)$
  - Uneori nu este valid (vezi teorema lui Mercer)
- Parametri: $\gamma > 0$, d, r

Interpretare

- Intuitiv, un nucleu poate fi interpretat ca o măsură a similarității dintre cele două argumente
- Nucleul liniar
(produsul scalar)



- Nucleul gaussian



Exemplu

- Clasificarea proteinelor: șiruri de aminoacizi codificați prin caractere
- Fie $\boldsymbol{\phi}(x)$ o funcție care numără aparițiile fiecărei secvențe de lungime k în șirul x
- $\boldsymbol{\phi}(x)$ este un vector cu 20^k dimensiuni (există 20 de aminoacizi standard)
- Pentru subșiruri mici de 5 caractere, există aproximativ 3 milioane de dimensiuni
- Totuși, nucleul poate implementa un algoritm eficient de *string matching*

Nuclee valide

- Folosind doar nucleul, nu mai este nevoie să calculăm funcția ϕ
- Nucleul ar putea avea orice formă
  - Trebuie să respecte însă condiția:
    $K(\mathbf{x},\mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{x})$

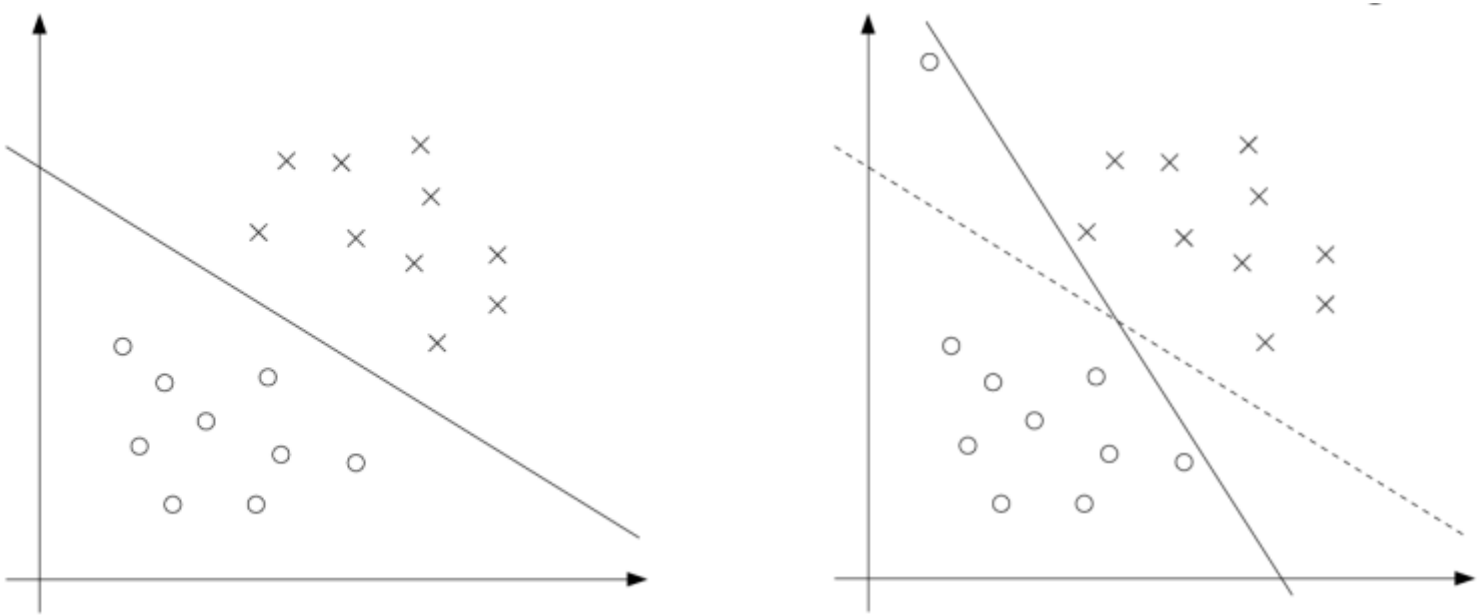
    chiar dacă nu cunoaștem sau nu ne interesează forma lui ϕ
- Fie matricea nucleului $\mathbf{K}$ o matrice în care elementele $\mathbf{K}_{ij} = K(x^{(i)}, x^{(j)})$

Teorema lui Mercer

- Condiția necesară și suficientă pentru ca un nucleu real să fie valid este ca matricea nucleului să fie simetrică și pozitiv semidefinită:
  - $\mathbf{K}_{ij} = \mathbf{K}_{ji}$
  - $\mathbf{z}^T \cdot \mathbf{K} \cdot \mathbf{z} \geq 0, \forall \mathbf{z}$
- Intuitiv, un nucleu pozitiv semidefinit doar deformează, nu împătureşte spațiul

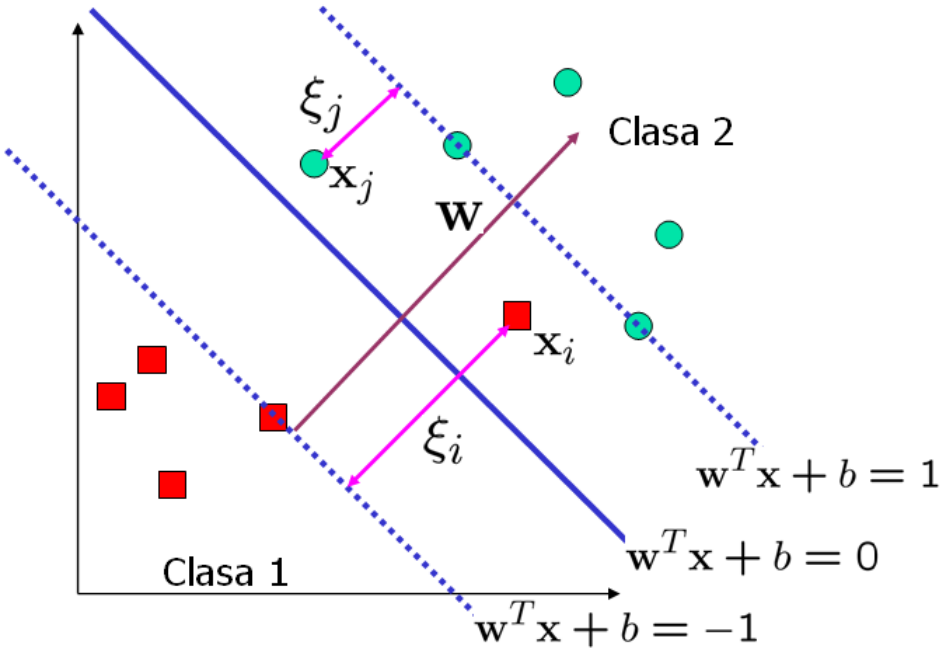
Cazurile neseparabile liniar

- În unele cazuri, datele nu sunt separabile liniar nici după transformările prezentate anterior
- sau prezintă valori extreme (*outliers*), care influențează marginea optimă



Probleme neseparabile liniar

- Putem permite existența unor erori ξ_i în clasificare
- Hiperplanul are acum o margine flexibilă (*soft margin*)
- ξ_i se numesc variabile de decalaj (*slack variables*)



Probleme neseparabile liniar

- Problema de optimizare primară devine:

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m. \end{aligned}$$

Parametrul *C*

- Parametrul de cost *C* este o măsură a erorii admise în clasificare
- *C* controlează compromisul dintre a permite erori pe mulțimea de antrenare și a forța margini stricte
- Creșterea valorii lui *C* mărește costul clasificării greșite a instanțelor și determină crearea unui model mai precis, dar care poate să nu generalizeze bine
- Dacă *C* este mare, marginea de separare va fi mai mică
- Dacă *C* este mic, marginea de separare va fi mai mare
- Valoarea lui *C* trebuie aleasă de utilizator și depinde de problemă
- De multe ori, *C* se alege prin încercări repetate, de exemplu: 10^{-5}, 10^{-3}, 0.1, 1, 10, 100, 10^{3}, 10^{5}

Problema duală

$$\max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0,$$

- Oarecum surprinzător, termenii ξ_i nu apar în problema duală
- Contează numai parametrul *C*, care limitează multiplicatorii α_i
- Pentru instanțele neclasificabile, multiplicatorii ar crește foarte mult și ar determina și apariția unor vectori suport (cu $\alpha_i > 0$) suplimentari

Algoritmul SMO

- Sequential Minimal Optimization (Platt, 1999)
- Scopul său este rezolvarea problemei duale de optimizare (determinarea α_i)
- Este rapid și deci foarte potrivit pentru optimizarea problemelor de dimensiuni mari cu care lucrează de obicei SVM

- Biblioteci pentru SVM: SVM-light, LibSVM

Ideea de bază a SMO

$$\max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m$$

$$\boxed{\sum_{i=1}^{m} \alpha_i y^{(i)} = 0}$$

- Optimizarea este asemănătoare metodei *hill climbing*
- Se ajustează succesiv câte 2 multiplicatori α_i

$$\alpha_1 y_1 + \alpha_2 y_2 = \text{const}$$

Exemplu

- Să presupunem că avem 5 instanțe unidimensionale
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, cu 1, 2, 6 din clasa 1 și 4, 5 din clasa 2 ⇒ y_1=1, y_2=1, y_3=–1, y_4=–1, y_5=1
- Folosim nucleul polinomial de grad 2
  - $K(x, z) = (x \cdot z + 1)^2$
- Setăm $C = 100$
- Mai întâi determinăm α_i (i = 1, ..., 5) prin

$$\max. \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

și constrângerile: $100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$

Exemplu

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}\right) + b$$

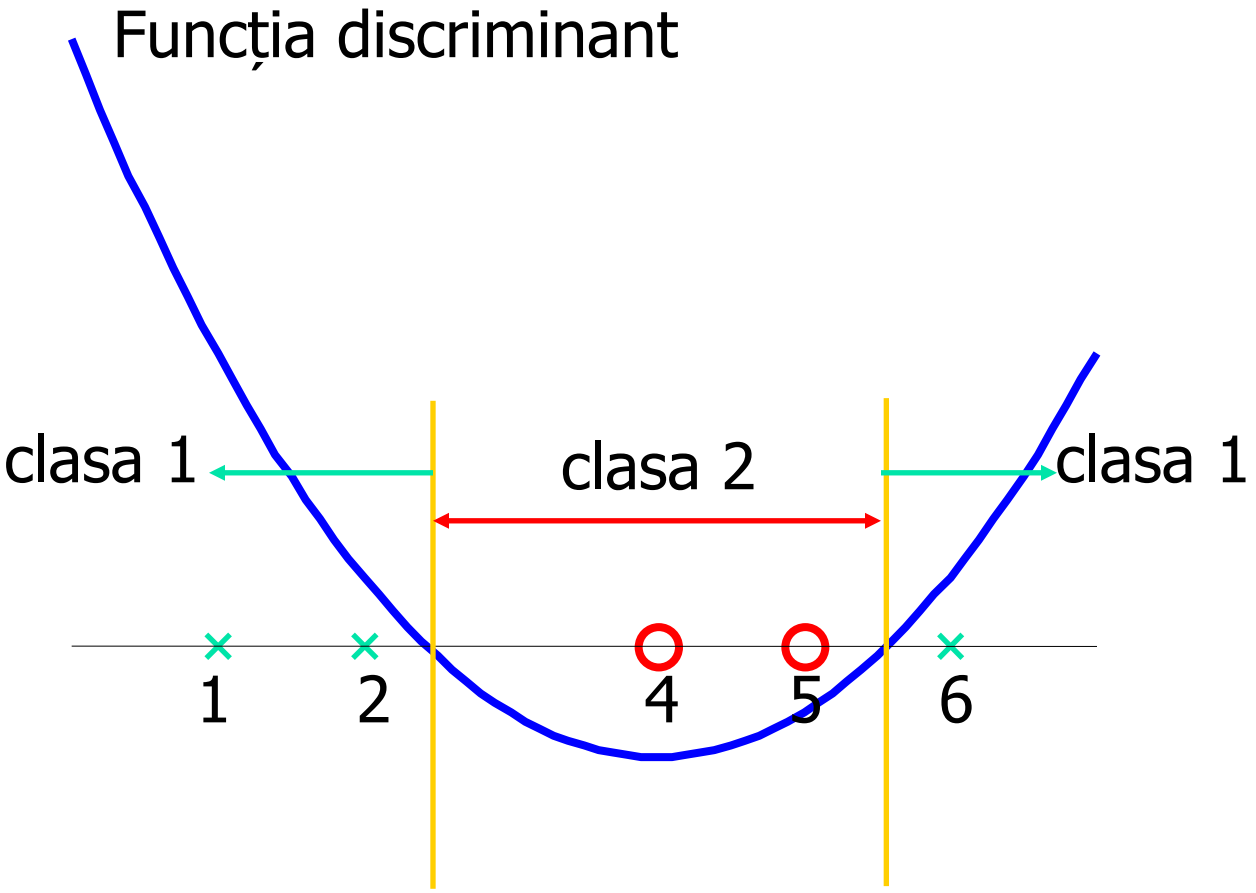
- Rezolvând problema de optimizare, obținem:
  - $\alpha_1 = 0$, $\alpha_2 = 2.5$, $\alpha_3 = 0$, $\alpha_4 = 7.333$, $\alpha_5 = 4.833$
  - Constrângerile sunt satisfăcute
  - Vectorii suport sunt: { $x_2 = 2$, $x_4 = 5$, $x_5 = 6$ }
- Funcția discriminant este:

α_5 y_5 $K(z, x_5)$

$$f(z)$$
$$= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b$$
$$= 0.6667z^2 - 5.333z + b$$

- b se determină prin rezolvarea $f(2) = 1$ sau $f(5) = -1$ sau $f(6) = 1$, deoarece x_2 și x_5 sunt pe linia $\phi(\mathbf{w})^T \phi(\mathbf{x}) + b = 1$ iar x_4 este pe linia $\phi(\mathbf{w})^T \phi(\mathbf{x}) + b = -1$
- Toate trei dau $b = 9 \Rightarrow f(z) = 0.6667z^2 - 5.333z + 9$

Exemplu



Exemplu

- Clasificarea instanțelor:

$$h(\mathbf{x}) = g(f(\mathbf{x})) \qquad g(z) = \begin{cases} 1, \text{ dacă } z \geq 0 \\ -1, \text{ dacă } z < 0 \end{cases}$$

- $f(1) = 4.3337 \Rightarrow h(1) = 1$
- $f(2) = 1 \Rightarrow h(1) = 1$ (vector suport)
- $f(4) = -1.6648 \Rightarrow h(1) = -1$
- $f(5) = -1 \Rightarrow h(1) = -1$ (vector suport)
- $f(6) = 1 \Rightarrow h(1) = 1$ (vector suport)

Clasificarea cu clase multiple

- Mașinile cu vectori suport clasice permit rezolvarea problemelor binare, cu numai două clase
- Pentru a trata mai multe clase există mai multe metode, dintre care cele mai des folosite sunt:
  - Abordarea „una versus toate”
  - Abordarea „una versus una”

Una versus toate

- *one-versus-all* / *one-against-all*
- Pentru *k* clase, se creează *k* modele
- În modelul cu numărul *i*, SVM este antrenat cu instanțele din clasa *i* ca exemple pozitive și toate celelalte instanțe (din celelalte clase) ca exemple negative
- Pentru a clasifica o nouă instanță, se apelează toate cele *k* modele și se alege clasa cu funcția de decizie maximă:

$$C = \underset{i=1..k}{\operatorname{argmax}} \left(f(\mathbf{x}) \right)$$

Una versus una

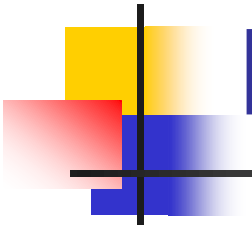
- *one-versus-one* / *one-against-one*
- Pentru k clase, se creează $k \cdot (k - 1) / 2$ modele corespunzătoare tuturor perechilor de clase
- Pentru a clasifica o nouă instanță, se apelează toate modelele și fiecare model dă un vot pentru apartenența la o clasă
- Se alege în final clasa care are cele mai multe voturi
- Antrenarea poate fi mai rapidă pentru că mulțimile de antrenare sunt mai mici

Generalizarea

- S-a demonstrat că probabilitatea de eroare așteptată pentru mulțimea de test este mărginită:

$$E\left[P(e)\right] \leq \frac{E\left[n_{vs}\right]}{l}$$

  - $E[x]$ = valoarea așteptată pentru x
  - n_{vs} = numărul de vectori suport
  - $P(e)$ = probabilitatea erorii
  - l = dimensiunea mulțimii de antrenare
- Capacitatea de generalizare crește cu cât numărul de vectori suport este mai mic și mulțimea de antrenare mai mare
- Rezultatul este independent de numărul de atribute al instanțelor, adică de dimensiunea spațiului problemei

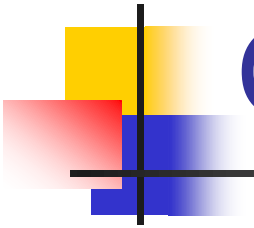


Discuție

- Principalele probleme care apar la lucrul cu SVM sunt legate de:
  - Stabilirea nucleului și a parametrilor săi
  - Stabilirea parametrului de cost *C*

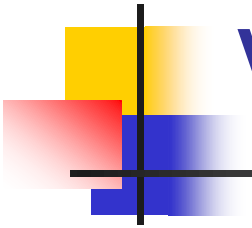
Rețele neuronale (II)

1. Mașini cu vectori suport
2. Metodologia de antrenare
3. Învățarea hebbiană
4. Concluzii



Generalizarea

- Dorim să găsim un model cu eroare cât mai mică
- Modelul trebuie să aibă capacitate de generalizare, care arată cât de bun este modelul pentru date noi
- De obicei există o mulțime de antrenare, pentru crearea modelului și o mulțime de test pentru verificarea capacității de generalizare



Validarea

- Uneori, se folosește și o mulțime de validare, pentru a decide când să se oprească antrenarea sau pentru a alege arhitectura cea mai potrivită a modelului
  - Mulțimea de validare nu este folosită pentru determinarea parametrilor modelului, ci pentru estimarea calității sale
- De exemplu:
  - Antrenăm o rețea neuronală doar cu mulțimea de antrenare
  - Observăm eroarea pe mulțimea de validare și oprim antrenarea când aceasta începe să crească
  - Observăm erorile pentru un număr diferit de neuroni în stratul ascuns și alegem numărul cel mai potrivit

Generarea mulțimii de test

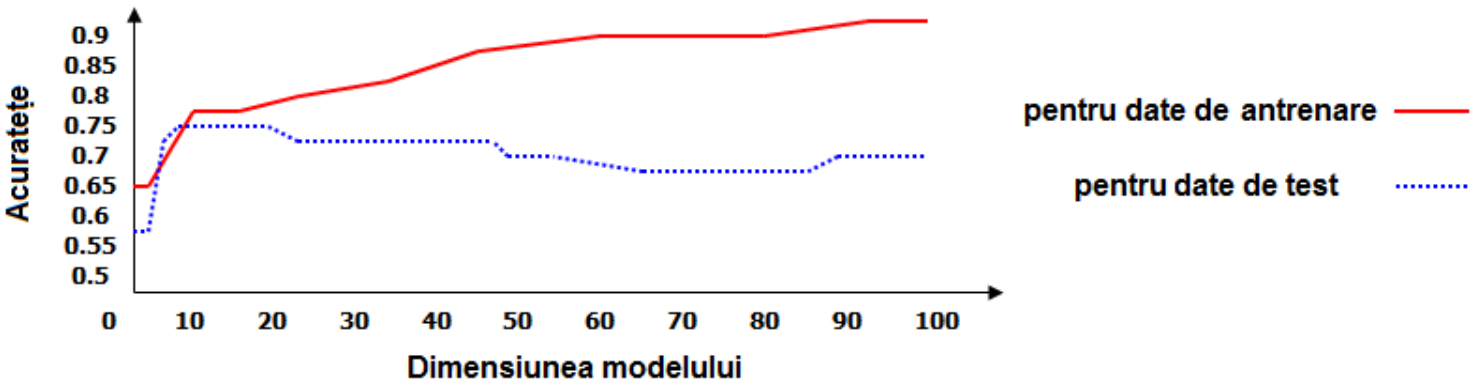
- Împărțirea 2/3 – 1/3
  - 2/3 pentru antrenare, 1/3 pentru testare
- Validarea încrucișată (*cross-validation*)
  - *n* grupuri, *n* – 1 pentru antrenare, al *n*-lea pentru testare, se repetă de *n* ori
  - De obicei, *n* = 10
- *Leave one out*
  - *n* – 1 instanțe pentru antrenare, a *n*-a pentru testare, se repetă de *n* ori

Exemplu: validarea încrucișată

$n = 5$

Generalitatea unui model

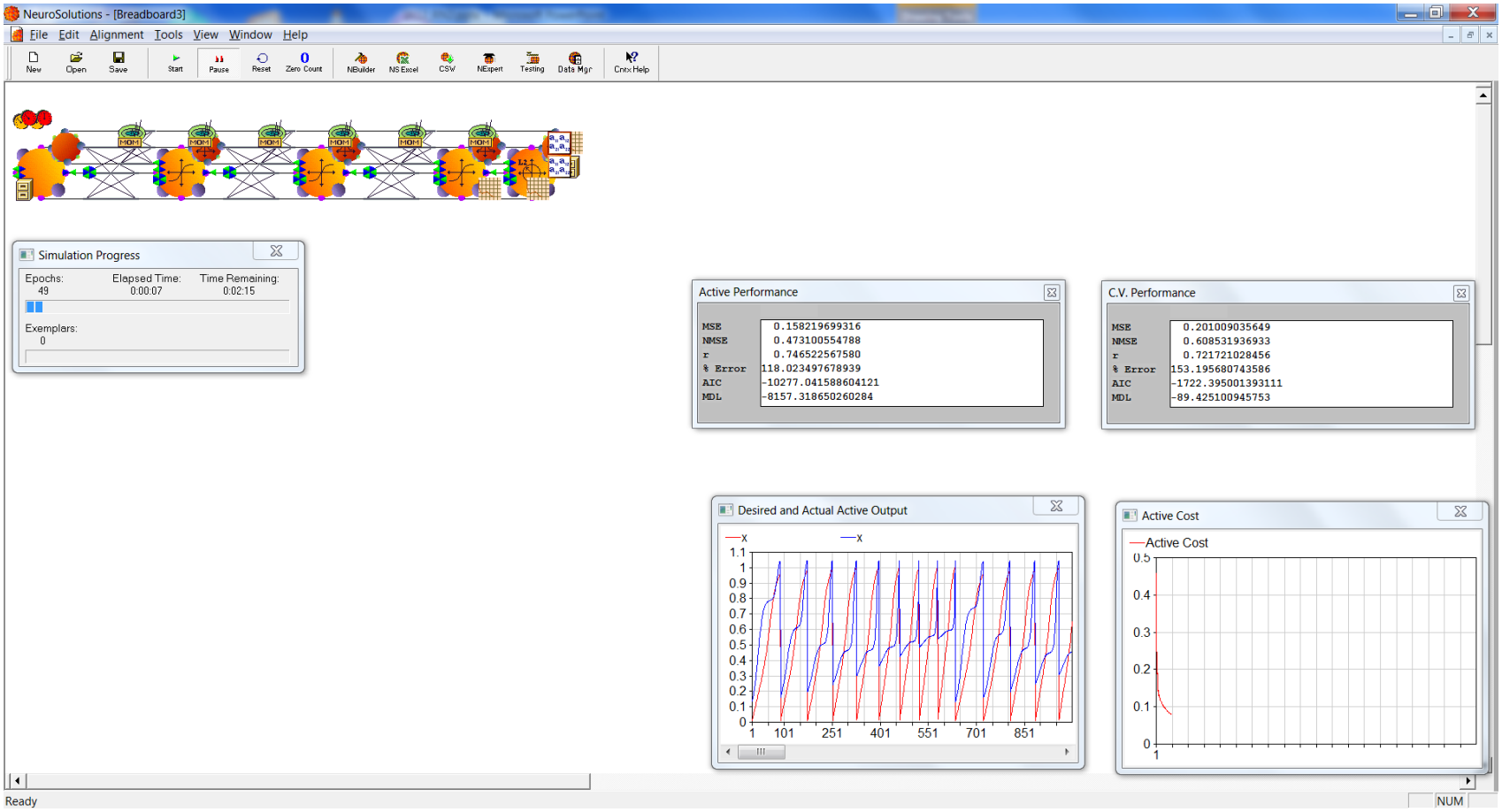
- Subpotrivirea (*underfitting*): modelul este prea simplu și nu poate învăța distribuția datelor
- Suprapotrivirea (*overfitting*): modelul este prea complex și poate fi influențat de zgomot și date irelevante
- Un model suprapotrivit are performanțe foarte bune pe mulțimea de antrenare, dar performanțe slabe pe mulțimea de validare/test



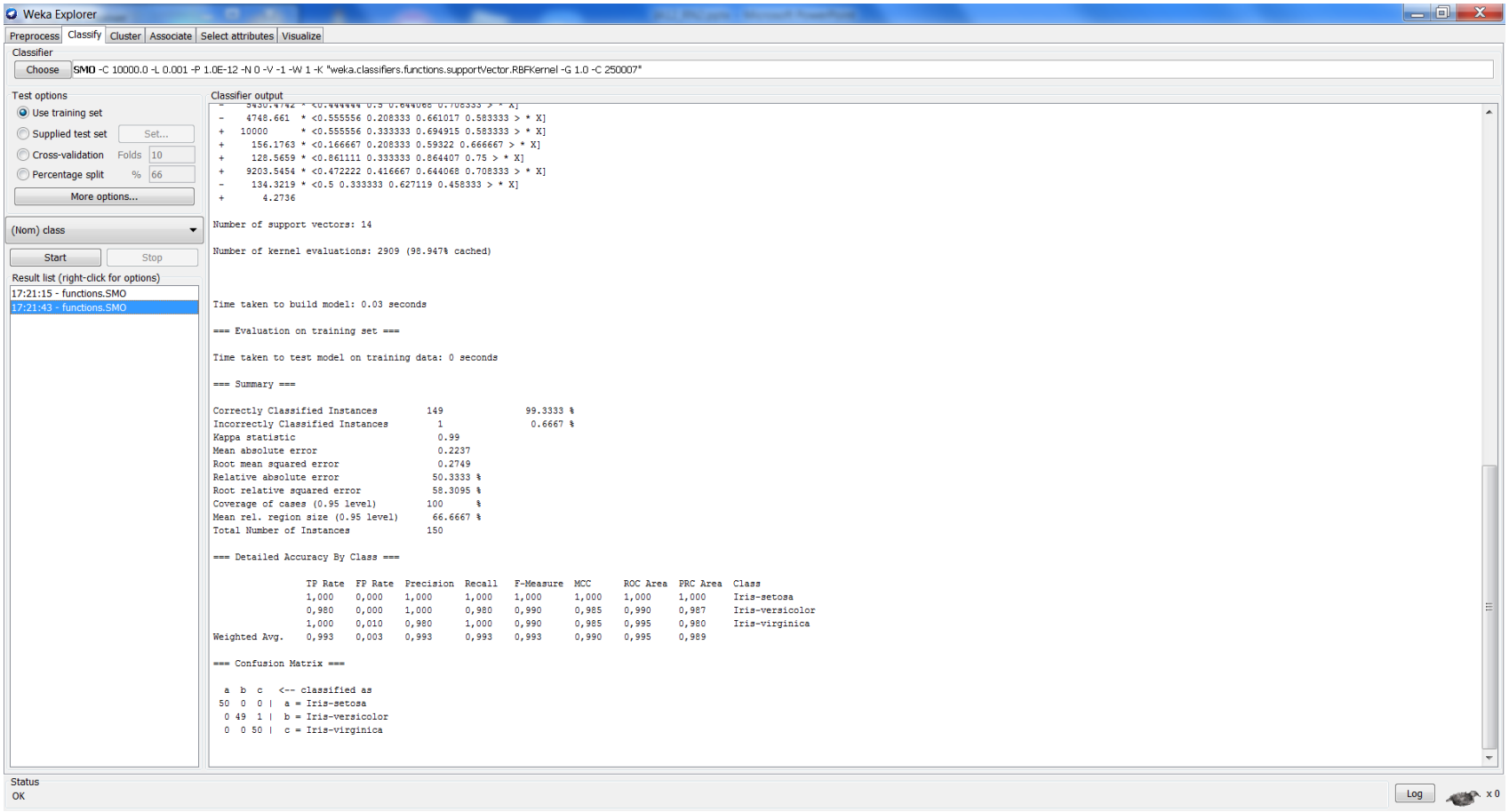
Metodologia de antrenare a unui model supervizat

- Metodologia este aceeași indiferent de metoda folosită
  - Rețele neuronale
  - Mașini cu vectori suport
  - Alți algoritmi

Neurosolutions

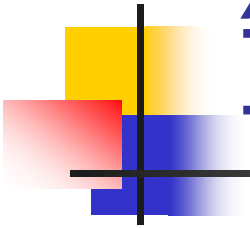


Weka


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
SMO -C 10000.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.RBFKernel -G 1.0 -C 250007"
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
17:21:15 - functions.SMO
17:21:43 - functions.SMO
Classifier output
- 4748.661 * <0.555556 0.208333 0.661017 0.583333 > * X]
+ 10000 * <0.555556 0.333333 0.694915 0.583333 > * X]
+ 156.1763 * <0.166667 0.208333 0.59322 0.666667 > * X]
+ 128.5659 * <0.861111 0.333333 0.864407 0.75 > * X]
+ 9203.5454 * <0.472222 0.416667 0.644068 0.708333 > * X]
- 134.3219 * <0.5 0.333333 0.627119 0.458333 > * X]
+ 4.2736
Number of support vectors: 14
Number of kernel evaluations: 2909 (98.947% cached)
Time taken to build model: 0.03 seconds
=== Evaluation on training set ===
Time taken to test model on training data: 0 seconds
=== Summary ===
Correctly Classified Instances 149 99.3333 %
Incorrectly Classified Instances 1 0.6667 %
Kappa statistic 0.99
Mean absolute error 0.2237
Root mean squared error 0.2749
Relative absolute error 50.3333 %
Root relative squared error 58.3095 %
Coverage of cases (0.95 level) 100 %
Mean rel. region size (0.95 level) 66.6667 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1,000 0,000 1,000 1,000 1,000 1,000 1,000 1,000 Iris-setosa
0,980 0,000 1,000 0,980 0,990 0,985 0,990 0,987 Iris-versicolor
1,000 0,010 0,980 1,000 0,990 0,985 0,995 0,980 Iris-virginica
Weighted Avg. 0,993 0,003 0,993 0,993 0,993 0,990 0,995 0,989
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 49 1 | b = Iris-versicolor
0 0 50 | c = Iris-virginica
Status
OK
Log
x 0

Rețele neuronale (II)

1. Mașini cu vectori suport
2. Metodologia de antrenare
3. Învățarea hebbiană
4. Concluzii



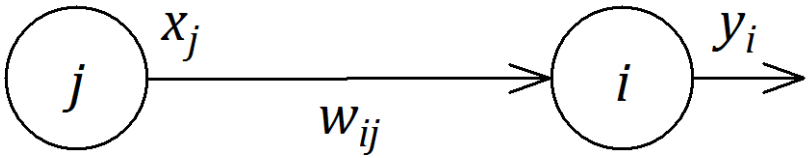
Învățarea nesupervizată

- Nu există nicio informație despre ieșirea dorită
- Algoritmul trebuie să descopere singur relațiile de interes din datele de intrare
  - Modele, regularități, corelații
- Relațiile descoperite se regăsesc în ieșire

Învățarea hebbiană

- Legea lui Hebb (1949)
  - Dacă doi neuroni conectați sunt activați în același timp, ponderea conexiunii dintre ei crește
  - Dacă doi neuroni sunt activați în contratimp, ponderea conexiunii dintre ei scade
  - *“Neurons that fire together, wire together”*

Învățarea hebbiană



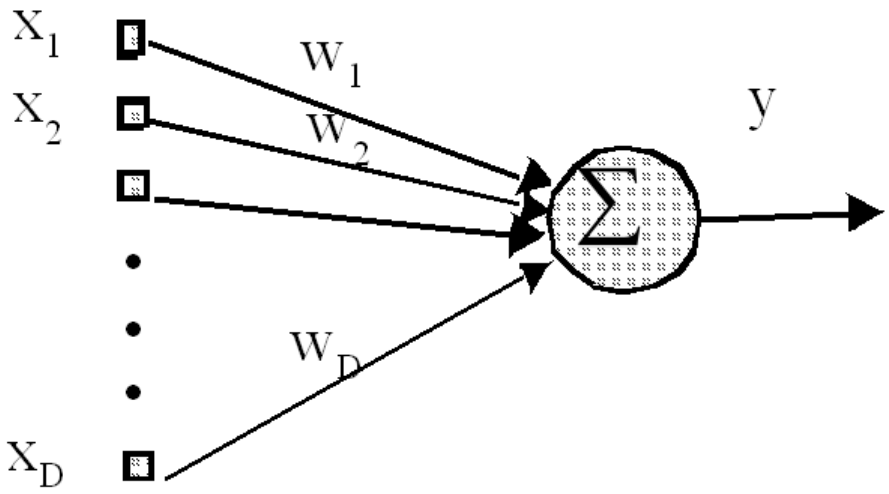
$$\Delta w_{ij} = \alpha \cdot x_j \cdot y_i \quad \leftarrow \quad \alpha \text{ este rata de învățare}$$

De obicei, *x* este 0 sau 1

$$y = \begin{cases} 0, & wx < 0.5 \\ 1, & wx \geq 0.5 \end{cases} \qquad \text{sau} \qquad y = wx$$

Intrări multiple

Modelele liniare sunt des întâlnite la învățarea hebbiană



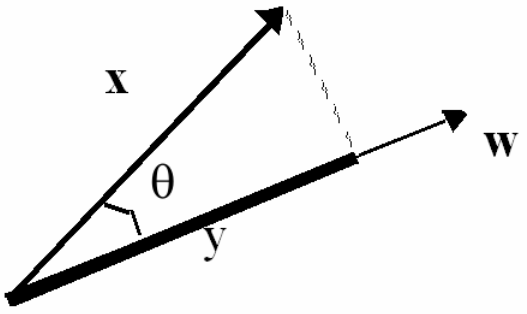
$$y = \sum_{i=1}^{D} w_i x_i \qquad \Delta w_i = \alpha \cdot x_i \cdot y$$

$$y = \mathbf{w}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{w}$$

Neuronul hebbian implementează o măsură de similaritate în spațiul de intrare

O ieșire *y* mare înseamnă că intrarea curentă este similară cu vectorul de antrenare **x** care a creat ponderile

Rețeaua își amintește vectorul de antrenare, deci se comportă ca o memorie asociativă



Instabilitatea

$$w \leftarrow w + \alpha \cdot x \cdot y$$

$$y = w \cdot x$$

$$w \leftarrow w \cdot \left(1 + \underbrace{\alpha \cdot x^2}_{\geq 0}\right)$$

$\Rightarrow$ ponderile vor crește nelimitat

Regula lui Oja

$$w_i \leftarrow \frac{w_i + \alpha \cdot x_i \cdot y}{\sqrt{\sum_i \left(w_i + \alpha \cdot x_i \cdot y\right)^2}}$$

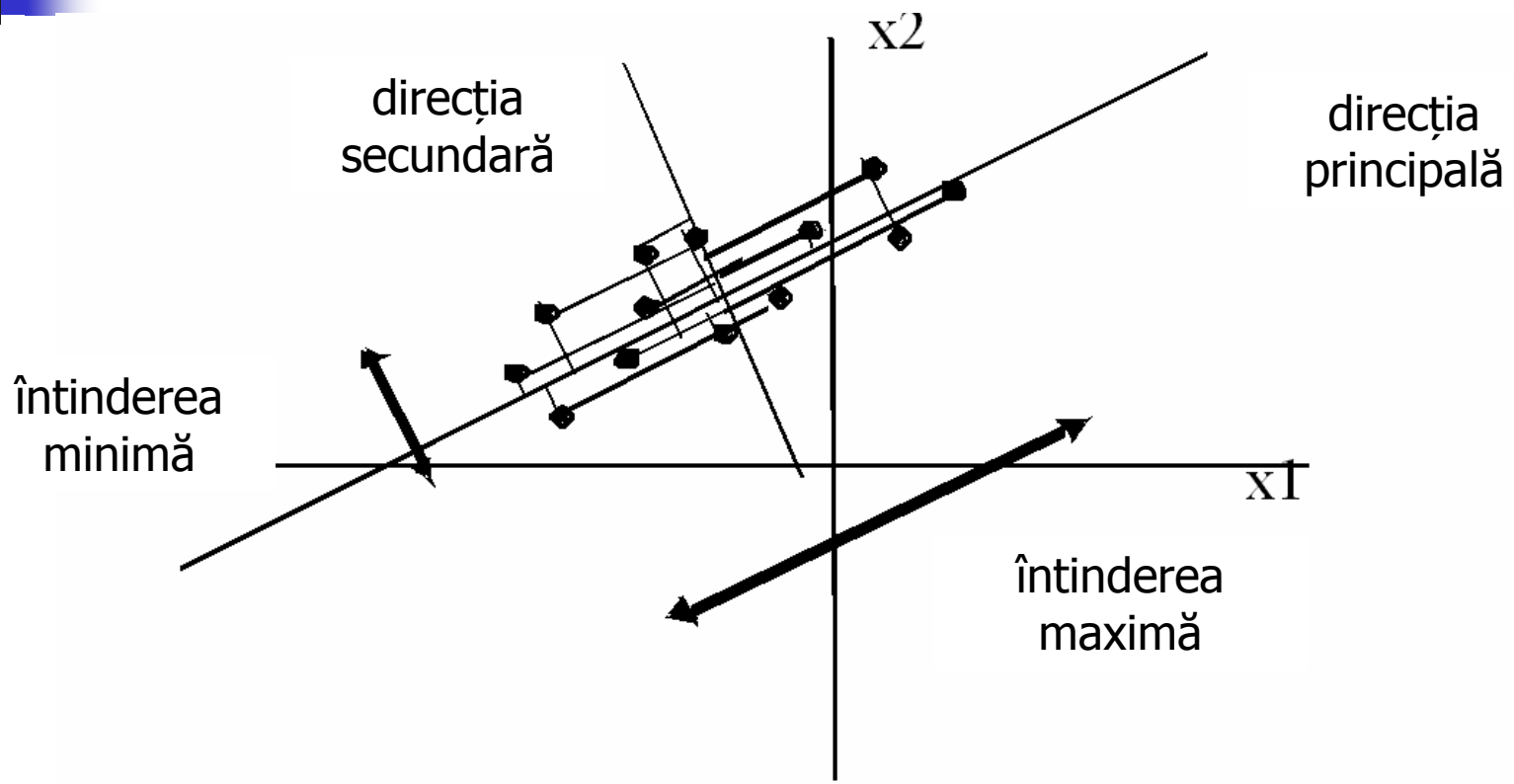
Normalizarea regulii hebbiene: noua valoare a unei ponderi se împarte la norma vectorului de ponderi. Dacă o pondere crește, celelalte trebuie să scadă.

Regula lui Oja este o formulă care aproximează normalizarea:

$$\Delta \mathbf{w} = \alpha \cdot y \cdot (\mathbf{x} - y \cdot \mathbf{w})$$

Ponderile nu mai cresc nelimitat, dar rețeaua poate uita asocierile vechi.
Dacă un vector de antrenare nu este prezentat frecvent, el poate fi uitat.

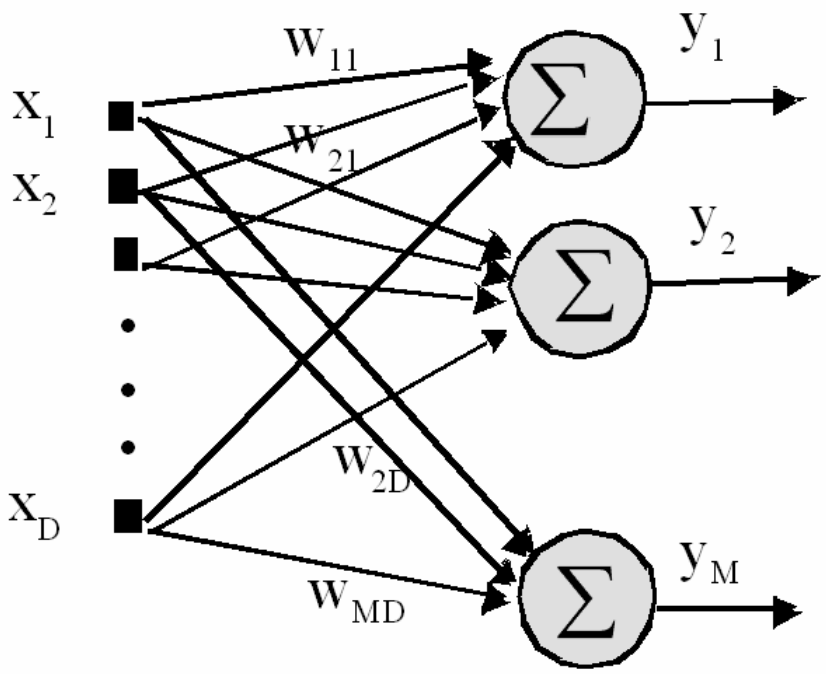
Regula lui Oja și analiza componentelor principale



Regula lui Oja determină un versor de ponderi **w** coliniar cu componenta principală a datelor de intrare

Rețea PCA

engl. "Principal Component Analysis", PCA



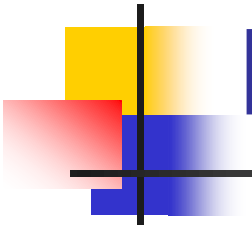
Ponderile reprezintă cei *M* vectori de dimensiune *D*

M ≤ *D*, dar de obicei *M* ≪ *D*

$$y_i = \sum_{j=1}^{D} w_{ij} x_j$$

Regula lui Sanger determină toate componentele principale, care converg succesiv:

$$\Delta w_{ij} = \alpha \cdot y_i \cdot \left(x_j - \sum_{k=1}^{i} w_{kj} y_k \right)$$



Discuție

- PCA este o metodă de reducere a dimensionalității sau de compresie a datelor
- Prin selectarea proiecțiilor pe cele mai importante M dimensiuni, din cele D ale spațiului de intrare, cu $M \leq D$, se păstrează cele mai importante caracteristici ale datelor

Abordări recente

- Modelul hebbian generalizat *ABCD*:

$$\Delta w_{ij} = \eta_w \cdot (A_w o_i o_j + B_w o_i + C_w o_j + D_w)$$

  - η este rata de învățare, o_i și o_j sunt ieșirile neuronilor *i* și *j*
  - Parametri diferiți pentru fiecare pondere *w*
  - S-a folosit pentru a evolua strategii de învățare cu întărire (spre deosebire de antrenarea bazată pe gradienți a rețelelor din învățarea cu întărire profundă)
- *Hierarchical Temporal Memory*, arhitectură bazată pe coloanele de neuroni din neocortex

Concluzii

- Mașinile cu vectori suport sunt una dintre cele mai eficiente metode de clasificare la ora actuală și au o fundamentare matematică solidă
- Învățarea hebbiană modelează dinamica sinapselor din creierele biologice