Sisteme de Recunoaştere a Formelor – Lab 9

Clasificatori Liniari

1. Obiective

În acest laborator vom studia funcţii de discriminare liniară, exemplificate de perceptron, şi vom aplica metoda "gradient descent" pentru a calcula cele mai bune reguli de clasificare pentru două clase.

2. Fundamente teoretice

Scopul clasificării este de a grupa elemente cu trăsături similare în clase sau grupuri. Un clasificator liniar reuşeşte acest lucru prin luarea deciziei în funcţie de o combinaţie liniară a trăsăturilor.

Definiţia unui clasificator
Fie S o mulţime de exemple $\{S_1, S_2,\ldots, S_n\}$ ce aparţin unor clase diferite $\{c_1, c_2, \ldots, c_m\}$. Fiecare exemplu are *d* trăsături associate $x = \{x_1, x_2,\ldots, x_d\}$.
Un clasificator este o corespondenţă între spaţiul trăsăturilor şi etichetele claselor, $\{c_1, c_2,\ldots,c_m\}$.

Astfel un clasificator partiţionează spaţiul în m **regiuni de decizie**. O dreaptă sau o curbă ce separă clasele se numeşte limită de decizie. Dacă avem mai mult de două trăsături, această limită devine o suprafaţă (e.g. un hiperplan).
În continuare ne vom referi la cazul cu două categorii, în care exemplele din S aparţin de două clase $\{c_1, c_2\}$.

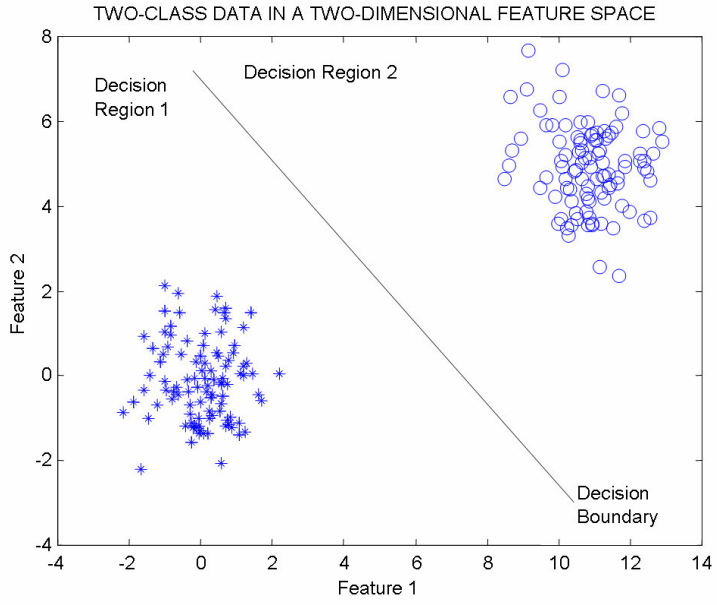


Figura 2.1: Exemplu de clasificator liniar pentru două clase. Fiecare exemplu are două trăsături.

Figura 2.1 arată un exemplu de limită de decizie liniară, între două categorii ale căror exemple sunt caracterizate de două trăsături.

2.1. Forma generală a unui clasificator liniar

Un clasificator liniar este o combinație liniară de trăsături, care sunt componentele unui vector **x**. Clasificatorul poate fi scris ca:

$$g(x) = w^T x + w_0 = \sum_{i=1}^{d} w_i x_i + w_0$$

Unde

- w este **vectorul de ponderi**
- w_0 este deplasamentul (**bias**), sau **ponderea pragului**.

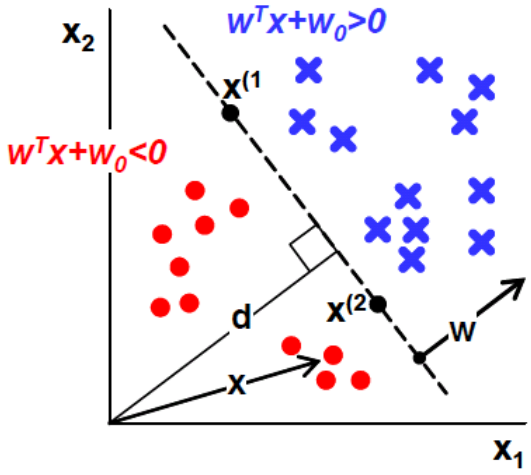
Un clasificator cu liniar două categorii (clasificator binar) implementează următoarea regulă de decizie:

$$\begin{cases} daca\ g(x) > 0\ decide\ ca\ exemplul\ x\ apartine\ clasei\ c_1 \\ daca\ g(x) < 0\ decide\ ca\ exemplul\ x\ apartine\ clasei\ c_2 \end{cases}$$

sau

$$\begin{cases} daca\ w^T x > -w_0\ decide\ ca\ exemplul\ x\ apartine\ clasei\ c_1 \\ daca\ w^T x < -w_0\ decide\ ca\ exemplul\ x\ apartine\ clasei\ c_2 \end{cases}$$

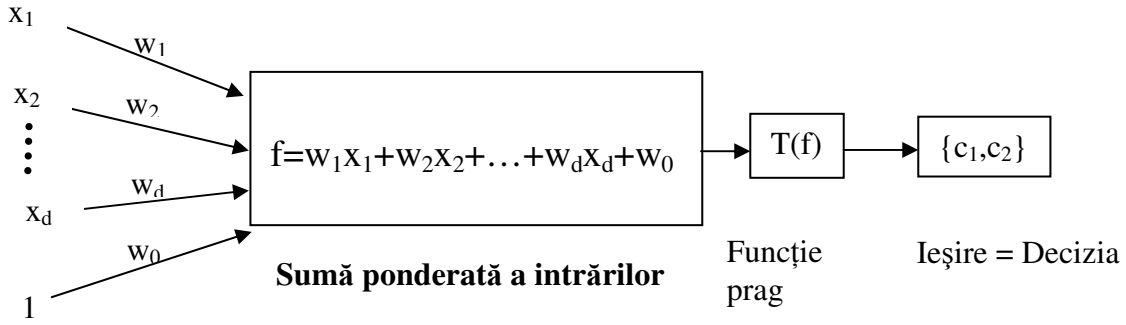
Daca g(x) = 0, x poate fi clasificat oricum.



2.2. Învățarea perceptron

Vom considera metoda învățării unei probleme de clasificare binară folosind un clasificator liniar. Presupunem că avem o mulțime de date $X=\{x^{(1}, x^{(2}, \dots, x^{(n}\}$ conținând exemple din două clase. Fiecare exemplu $x^{(i}$ din mulțime este caracterizat de un vector de trasaturi $\{x_1, x_2, \dots, x_d\}$.

O vedere schematică a perceptronului este prezentată în figura următoare.



Pentru convenienţă vom absorbi factorul liber w_0 prin introducerea în vectorul de trăsături **x** a unei dimensiuni suplimentare.

$$w^T x + w_0 = \begin{bmatrix} w_0 & w^T \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = a^T y$$

Obiectivul nostrum este de a găsi un vector **a** în aşa fel încât :

$$g(x) = a^T y \begin{cases} > 0, & x \in c_1 \\ < 0, & x \in c_2 \end{cases}$$

Pentru a simplifica calculele, vom "normalize" setul de antrenare prin înlocuirea exemplelor din clasa c_2 prin negativul lor: $y \leftarrow [-y], \forall y \in c_2$.

Acest lucru ne permite să ignorăm clasele şi să căutăm un vector de ponderi astfel ca

$$a^T y > 0, \quad \forall y$$

Pentru a găsi acest vector, definim o **functie obiectiv J(a).** Functia obiectiv pentru perceptron este:

$$J_p(a) = \sum_{y \in Y_M} (-a^T y)$$

Unde:

- Y_M este mulţimea exemplelor clasificate greşit de **a**
- Se poate observa că $J_p(a)$ este non-negativă, deoarece $a^T y<0$ pentru toate exemplele clasificate greşit.

Pentru a găsi minimul acestei funcţii, vom folosi metoda "gradient descent". În cazul nostru, gradientul este definit ca:

$$\nabla_a J_p(a) = \sum_{y \in Y_M} (-y)$$

Regula de actualizare este:

$$a(k+1) = a(k) + \eta \sum_{y \in Y_M(k)} y$$

Această regulă este numită regula de actualizare în bloc (batch update) a perceptronului. Vectorul de ponderi se poate actualiza şi pas cu pas, prin prezentarea exemplelor individuale ("on-line"):

$a(k+1) = a(k) + \eta y^{(i}$, unde $y^{(i}$ este un exemplu clasificat greşit de a(k)

Algorithm: Batch Perceptron

Begin

- **Initializare** a, η, criteriul θ, k=0
- "normalizare" a setului de antrenare prin înlocuirea tuturor exemplelor din c_2 prin negativul lor: $y \leftarrow [-y], \forall y \in c_2$
- **Do**
  - k = k+1
  - $a = a + \eta \sum_{y \in Y_M(k)} y$
- **Until** $\eta \sum_{y \in Y_M(k)} y < \theta$
- **Return** a

End

2.2.1. Exemplu numeric

Să se antreneze perceptronul pe următoarele date:
X_1 = [(1,6), (7,2), (8,9), (9,9)]
X_2 = [(2,1), (2,2), (2,4), (7,1)]

Algoritmul de învățare perceptron:

- Considerăm η=0.1
- Considerăm a(0)=[0.1, 0.1, 0.1]
- Folosim metoda de actualizare pas cu pas
- Se normalizează setul de date

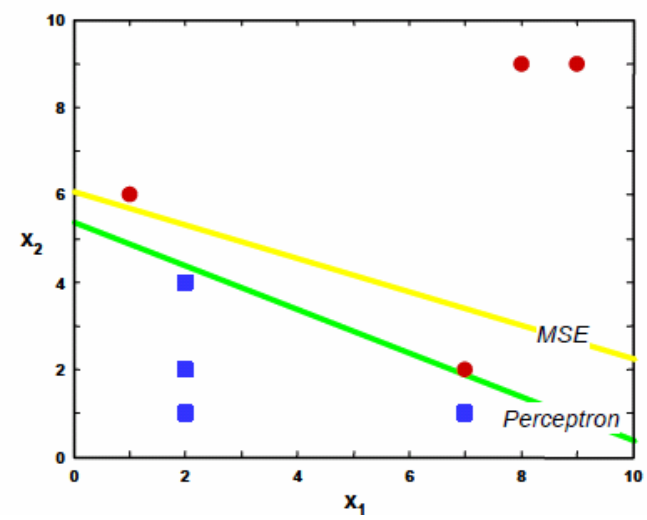
$$Y = \begin{pmatrix} 1 & 1 & 6 \\ 1 & 7 & 2 \\ 1 & 8 & 9 \\ 1 & 9 & 9 \\ -1 & -2 & -1 \\ -1 & -2 & -2 \\ -1 & -2 & -4 \\ -1 & -7 & -1 \end{pmatrix}$$

- Se parcurg exemplele şi se actualizează *a(k)* atunci când apare clasificare greşită.
  1. Y(1) ⇒ [1 1 6]*[0.1 0.1 0.1]^T>0 ⇒ ok
  2. Y(2) ⇒ [1 7 2]*[0.1 0.1 0.1]^T>0 ⇒ ok
  3. …

4. …
5. Y(5) ⇒ [-1 -2 -1]*[0.1 0.1 0.1]^T<0 ⇒ actualizare a(1) = [0.1 0.1 0.1] + η*[-1 -2 -1] = [0 -0.1 0]
6. Y(6) ⇒ [-1 -2 -2]*[0 -0.1 0]^T>0 ⇒ ok
7. …
8. Y(1) ⇒ [1 1 6]*[0 -0.1 0]^T<0 ⇒ actualizare a(2) = [0 -0.1 0] +η*[1 1 6] = [0.1 0 0.6]
9. Y(2) ⇒ [1 7 2]*[0.1 0 0.6]^T>0 ⇒ ok
10. …

Oprire când $\eta \sum_{y \in Y_M(k)} y < \theta$

- În acest exemplu regula converge după 175 iterații la a=[-3.5 0.3 0.7]



3. Clasificator perceptron bazat pe două trăsături

În sesiunea de laborator vom crea un clasificator perceptron care va discrimina intre două mulțimi de puncte. Punctele din clasa 1 sunt colorate rosu, iar cele din clasa 2 in albastru, ca in figura de mai jos.

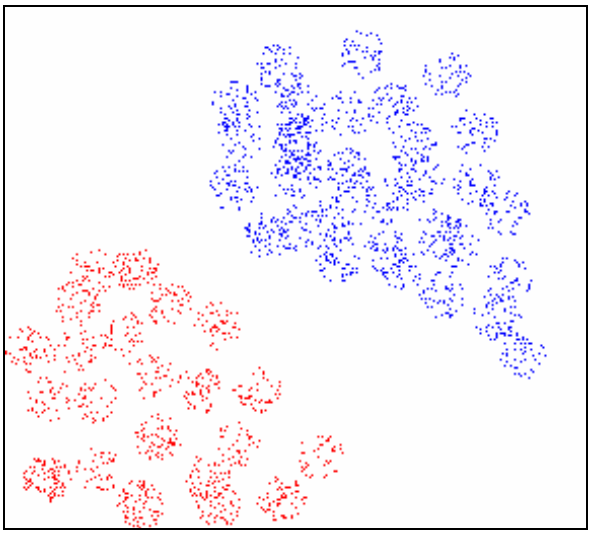


Figure 3.1: Punctele de intrare pentru clasificator

Fiecare punct este descries prin culoare (ce indica clasa) si prin cele doua coordinate ale lui, care sunt trasaturile x_1 si x_2.
Vectorul de ponderi are forma a = [w_0, w_1, w_2].
Vectorul y = [1 x_1 x_2].

Algoritmul devine:
Algorithm: Batch Perceptron pentru doua clase, doua trasaturi
Begin

- **Initialize**
  - a = [0.1 0.1 0.1]
  - η=1
  - prag oprire θ=0.00001
  - k=0
  - vector_oprire b=[0 0 0]

- "normalizare" a setului de antrenare, înlocuirea exemplelor din clasa c_2 cu negativul lor: $y \leftarrow [-y], \forall y \in c_2$, astfel ca y=[–1 $-x_1$ $-x_2$]

- **Do**
  - b = [0 0 0]
  - k = k+1
  - $a = a + \eta \sum_{y \in Y_M(k)} y$
  - $b = b + \eta \sum_{y \in Y_M(k)} y$
- **Until** $\|b\| < \theta$
- **Return** a

End

Rezultatul clasificarii este:

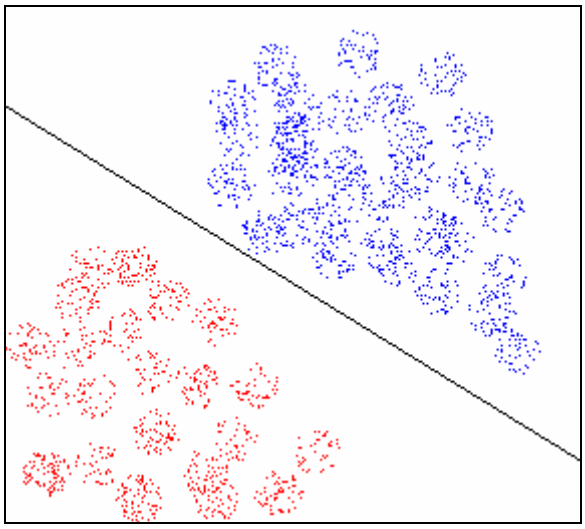


Figure 3.2: Rezultatul antrenarii perceptronului

4. Exercitii

Dându-se un set de puncte aparţinând celor două clase, roşi şi albastru:

Aplicați algoritmul “batch perceptron” pentru a găsi clasificatorul liniar ce le împarte în două grupuri.

Considerați $\eta=1$, initializati $a(0) = [0.1, 0.1, 0.1]$ si $\theta=0.0001$.

Paleta imaginilor de intrare este modificata, astfel ca pe pozitia 1 avem culoarea rosue iar pe pozitia 2 avem culoarea albastra.

5. Bibliografie

[1] Richard O. Duda, Peter E. Hart, David G. Stork: *Pattern Classification 2nd ed.*