10 Clasificatori liniari şi algoritmul perceptron

10.1 Obiective

Acest laborator prezintă algoritmul de învăţare perceptron pentru clasificatori liniari. Vom aplica gradient descent şi stochastic gradient descent pentru a obţine vectorul de ponderi. Se va rezolva o problemă de clasificare binară folosind două trăsături.

10.2 Fundamente teoretice

Scopul clasificării este determinarea clasei unor exemple descrise prin trăsături. Clasa unui exemplu face parte dintr-o mulţime finită, de exemplu: roşu/albastru, pieton/fundal, o cifră din mulţimea 0, 1, ... 9. Un clasificator liniar reuşeşte acest lucru prin luarea deciziei în funcţie de o combinaţie liniară a trăsăturilor.

10.2.1 Definiţii

Definim un dataset etichetat ca şi o pereche *(X,Y)*, unde $X \in M_{n\times m}(R)$ este o matrice cu n linii şi m coloane având valori reale şi Y este un vector coloană $Y \in M_{n\times 1}(D)$, care conţine etichetele pentru fiecare instanţă, iar D este mulţimea claselor posibile. X conţine pe linii instanţele de antrenare descrise cu m trăsături. Un clasificator este o funcţie care mapează orice instanţă reprezentată printr-un vector de trăsături la o clasă: $f: R^m \rightarrow D$.

Astfel un clasificator separă spaţiul de trăsături în $|D|$ regiuni disjunte (numărul de elemente din D). Subspaţiul care separă clasele se numeşte suprafaţa de decizie (eng. decision boundary). Dacă problema de clasificare este bidimensională acest subspaţiu va fi unidimensional şi va fi o linie sau o curbă, în general. În continuare vom discuta despre cazul cu două clase, algoritmul se poate extinde uşor la mai multe clase. Pentru etichete vom folosi $D = \{-1,1\}$.

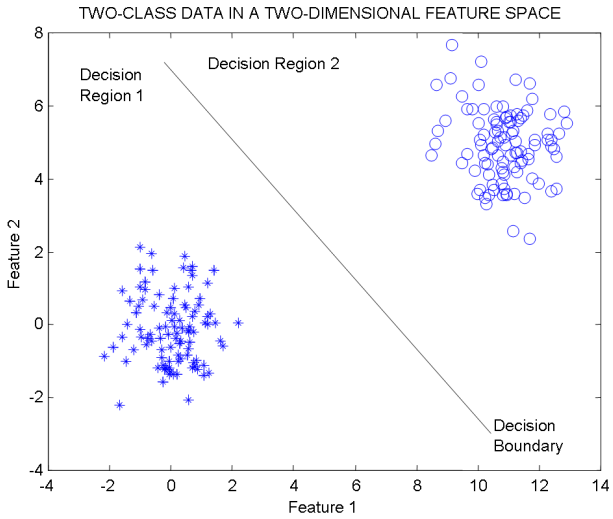


Figura 10.1. Exemplu de clasificator liniar care separă două clase şi utilizează două trăsături

10.2.2 Forma generală a unui clasificator liniar

Un clasificator liniar este o combinaţie liniară a trăsăturilor de intrare. Dacă notăm cu $\boldsymbol{x} \in M_{1\times m}(R)$ vectorul de trăsături şi cu $\boldsymbol{w} \in M_{m\times 1}$ putem exprima funcţia de clasificare ca şi:

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

Unde

- $\boldsymbol{w}$ este **vectorul de ponderi**
- w_0 este deplasamentul sau valoarea de prag.

Următoarea schemă ilustrează modul de operare a clasificatorului liniar:

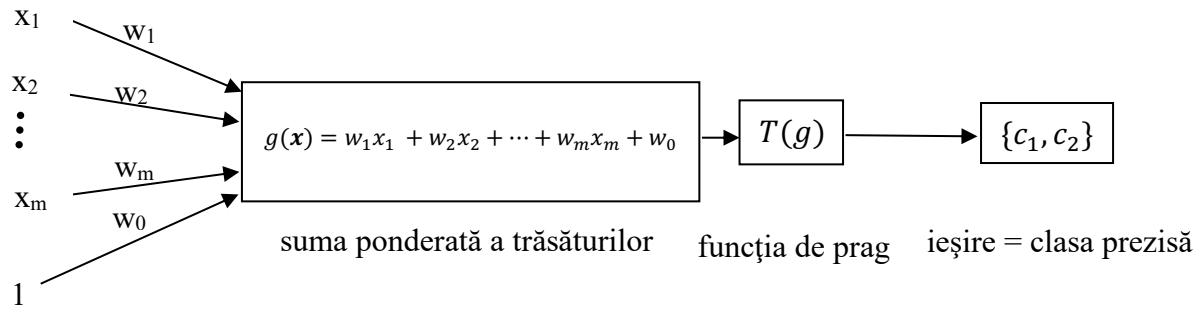


Figura 10.2 – Schema pentru un clasificator liniar

Pentru a simplifica notaţia, calculele şi pentru a permite tratarea generală vom absorbi valoarea w_0 în vectorul de ponderi şi vom augmenta vectorul de trăsături cu o componentă egală cu 1. În acest caz se simplifică expresia în:

$$w_0 + \boldsymbol{w}^T \boldsymbol{x} = [w_0 \quad \boldsymbol{w}] \begin{bmatrix} 1 \\ \boldsymbol{x}^T \end{bmatrix} = \bar{\boldsymbol{w}} \bar{\boldsymbol{x}}^T$$

Un clasificator liniar pentru două clase implementează următoarea regulă de clasificare:

- dacă $g(\boldsymbol{x}) > 0$, decidem că $\boldsymbol{x}$ aparţine clasei +1;
- dacă $g(\boldsymbol{x}) < 0$, decidem că $\boldsymbol{x}$ aparţine clasei -1;

sau echivalent:

- dacă $w^T \boldsymbol{x} > -w_0$, decidem că $\boldsymbol{x}$ aparţine clasei +1;
- dacă $w^T \boldsymbol{x} < -w_0$, decidem că $\boldsymbol{x}$ aparţine clasei -1;

Dacă $g(\boldsymbol{x}) = 0$, atunci $\boldsymbol{x}$ poate fi atribuit la oricare clasă, fiind pe suprafaţa de separare.

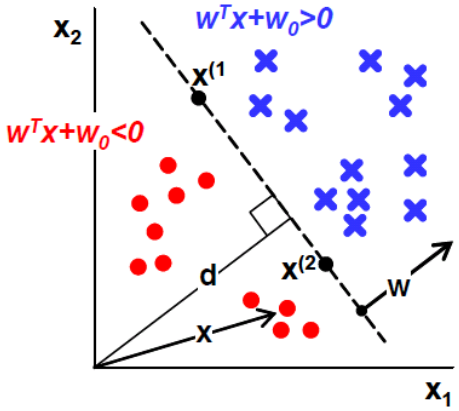


Figura 10.2. Ilustrarea componentelor din vectorul de ponderi: vectorul w este normala la dreapta de separare iar w_0 este egal cu distanţa d (cu semn) a dreptei de la origine înmulţită cu $\|w\|$ norma lui w

10.2.3 Metode de învăţare pentru clasificatori liniari

Vom prezenta două metode de învăţare pentru clasificatori liniari. Învăţarea se face prin transformarea problemei de clasificare într-o problemă de minimizare a unei funcţii de cost. Denumim funcţia L (de la loss function din engleză). Funcţia poate să aibă mai multe forme şi trebuie să penalizeze diferenţele dintre predicţia clasificatorului şi clasa adevărată. În cazul algoritmului perceptron adoptăm următoarea funcţie de cost:

$$L(\bar{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n} max(0, -y_i \bar{\boldsymbol{w}}^T \cdot \bar{\boldsymbol{x}}_i) = \frac{1}{n}\sum_{i=1}^{n} L_i(\bar{\boldsymbol{w}})$$

Dacă o instanţă este corect clasificată atunci nu se aplică nici o penalizare, în schimb, dacă instanţa se atribuie la clasa greşită atunci se penalizează proporţional cu scorul greşit $y_i \bar{\boldsymbol{w}}^T \cdot \bar{\boldsymbol{x}}_i$. Semnul expresiei $y_i \bar{\boldsymbol{w}}^T \cdot \bar{\boldsymbol{x}}_i$ este negativ dacă semnul etichetei de clasă y_i este diferit de semnul clasei prezise $g(\boldsymbol{x})$.

Pentru a minimiza acest cost vom utiliza metoda gradient descent. Vom porni de la un vector de ponderi aleator şi vom schimba acest vector bazându-ne pe valoarea gradientului în poziţia curentă. Pasul va fi în direcţia opusă a gradientului:

$$\bar{\boldsymbol{w}}_{k+1} \leftarrow \bar{\boldsymbol{w}}_k - \eta \nabla \boldsymbol{L}(\bar{\boldsymbol{w}}_k)$$

unde $\bar{\boldsymbol{w}}_k$ este vectorul de ponderi la momentul k, parametrul η controlează mărimea pasului şi se numeşte rata de învăţare (eng. learning rate), şi $\nabla \boldsymbol{L}(\bar{\boldsymbol{w}}_k)$ este vectorul de gradient calculat în punctul $\bar{\boldsymbol{w}}_k$. Vectorul de gradient are expresia:

$$\nabla \boldsymbol{L}(\bar{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n} \nabla \boldsymbol{L_i}(\bar{\boldsymbol{w}})$$

$$\nabla \boldsymbol{L_i}(\bar{\boldsymbol{w}}) = \begin{cases} 0, & \text{daca } y_i \bar{\boldsymbol{w}}^T \cdot \bar{\boldsymbol{x}}_i > 0 \\ -y_i \bar{\boldsymbol{x}}_i^T, & \text{altfel} \end{cases}$$

În metoda care foloseşte gradientul pe setul întreg ponderile se schimbă doar după ce am vizitat toate exemplele de antrenare. Această abordare se mai cheama şi batch-update. Algoritmul de perceptron din lucrarea [1] actualizează ponderile după examinarea fiecărui exemplu de antrenare. În acest caz algoritmul se numeşte online perceptron. Regula de actualizare devine:

$$\bar{\boldsymbol{w}}_{k+1} \leftarrow \bar{\boldsymbol{w}}_k - \eta \nabla \boldsymbol{L}_{\boldsymbol{i}}(\bar{\boldsymbol{w}})$$

Redăm în continuare, în paralel, cele două variante ale algoritmului de învăţare:

Algorithm:
Batch Perceptron

```
init w, η, Elimit, max_iter
for iter=1:max_iter
  E = 0, L = 0
  ∇L = [0,0,0]
  for i=1:n
    z_i = Σ_{j=0}^{d} w_j X_ij
    if z_i · y_i ≤ 0
      ∇L ← ∇L − y_i X_i
      E ← E + 1
       L ← L − y_i z_i
    endif
  endfor
  E ← E/n
  L ← L/n
  ∇L ← ∇L /n
  if E < E_limit
    break
  w ← w − η∇L
endfor
```

Algorithm:
Online Perceptron

```
init w, η, Elimit, max_iter
for iter=1:max_iter
  E = 0
  for i=1:n
    z_i = Σ_{j=0}^{d} w_j X_ij
    if z_i · y_i ≤ 0
      w ← w + η X_i y_i
      E ← E + 1
    endif
  endfor
  E ← E/n
  if E < E_limit
    break
endfor
```

10.3 Aspecte practice

Vom folosi puncte 2D care sunt împărţite în 2 clase. Punctele se citesc din imaginile de intrare şi apartenenţa lor se stabileşte pe baza culorii. Asociem clasa -1 la punctele albastre şi 1 la punctele roşii.

Fiecare punct va fi descris de două trăsături x_1 şi x_2 care coincid cu coordonatele lor x şi y, respectiv. Astfel, vectorul de trăsături augmentat are forma $\bar{\boldsymbol{x}} = [1\ x_1\ x_2]$ iar vectorul de ponderi are trei componente $\bar{\boldsymbol{w}} = [w_0\ w_1\ w_2]^T$.

10.4 Activitate practică

1. Citiţi punctele dintr-o singură imagine *test0*.bmp* şi construiţi setul de antrenare *(X,Y)*. Asociaţi -1 la punctele albastre şi +1 la punctele roşii.
2. Implementaţi algoritmul *Online Perceptron* pentru a găsi linia de separare dintre cele două clase. Sugestie pentru parametri:
   $\eta=10^{-4}$, $w_0 = [0.1, 0.1, 0.1]$, $E_{limit}=10^{-5}$, $max\_iter = 10^5$.

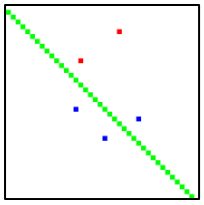
Obervaţie: Pentru a accelera convergența algoritmului se foloseşte o rata de învățare mai mare doar pentru termenul liber w_0.

3. Desenaţi linia de separare găsită de algoritm şi dată de ecuaţia:
$w_0 + w_1 x + w_2 y = 0$.
4. Implementaţi algoritmul de batch perceptron. Găsiţi parametri buni care asigură învăţarea. Se va monitoriza funcţia de cost care trebuie să descrească încet la fiecare pas.
5. Vizualizaţi linia de separare în timp ce rulează algoritmul să observaţi cum se schimbă.
6. Schimbaţi valorile de start pentru vectorul **w**, modificaţi rata de învăţare şi observaţi ce se întâmplă în fiecare caz. Ce înseamnă dacă funcţia de cost L oscilează în loc să descrească la fiecare pas?

10.5 Exemplu numeric

Fie punctele din fişierul points00 – (x,y) sau (coloana, rând):

- Roșii: (23, 5), (15,11) – clasa +1
- Albastre: (14, 21), (27,23), (20, 27) – clasa -1



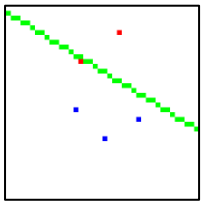
Pașii pentru algoritmul online perceptron sunt următorii:

- Rata de învățare = 0.01

Iterația 0

i=0: w=[1.000000 1.000000 -1.000000] xi=[1 23 5] yi = 1 zi=19.000000
i=1: w=[1.000000 1.000000 -1.000000] xi=[1 15 11] yi = 1 zi=5.000000
i=2: w=[1.000000 1.000000 -1.000000] xi=[1 14 21] yi = -1 zi=-6.000000
i=3: w=[1.000000 1.000000 -1.000000] xi=[1 27 23] yi = -1 zi=5.000000 greșit
  - update w0 = w0 - 0.01, w1 = w1 - 27*0.01, w2 = w2 – 23*0.01

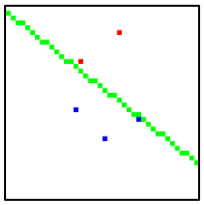
i=4: w=[0.990000 0.730000 -1.230000] xi=[1 20 27] yi = -1 zi=-17.620000



Iterația 1

i=0: w= [0.990000 0.730000 -1.230000] xi= [1 23 5] yi = 1 zi=11.630000
i=1: w= [0.990000 0.730000 -1.230000] xi= [1 15 11] yi = 1 zi=-1.590000 greșit
  - update w0 = w0 + 0.01, w1 = w1 + 15*0.01, w2 = w2 + 11*0.01

i=2: w= [1.000000 0.880000 -1.120000] xi= [1 14 21] yi = -1 zi=-10.200000
i=3: w= [1.000000 0.880000 -1.120000] xi= [1 27 23] yi = -1 zi=-1.000000
i=4: w= [1.000000 0.880000 -1.120000] xi= [1 20 27] yi = -1 zi=-11.640000



Iterația 2

i=0: w= [1.000000 0.880000 -1.120000] xi= [1 23 5] yi = 1 zi=15.640000
i=1: w= [1.000000 0.880000 -1.120000] xi= [1 15 11] yi = 1 zi=1.880000
i=2: w= [1.000000 0.880000 -1.120000] xi= [1 14 21] yi = -1 zi=-10.200000
i=3: w= [1.000000 0.880000 -1.120000] xi= [1 27 23] yi = -1 zi=-1.000000
i=4: w= [1.000000 0.880000 -1.120000] xi= [1 20 27] yi = -1 zi=-11.640000
Toate clasificate corect

10.6 Referinţe

[1] Rosenblatt, Frank (1957), The Perceptron - a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.
[2] Richard O. Duda, Peter E. Hart, David G. Stork: Pattern Classification 2nd ed.
[3] Xiaoli Z. Fern, Machine Learning Course, Oregon University -
[4] http://web.engr.oregonstate.edu/~xfern/classes/cs434/slides/perceptron-2.pdf
[5] Gradient Descent - http://en.wikipedia.org/wiki/Gradient_descent
[6] Avrim Blum, Machine Learning Theory, Carnegie Mellon University - https://www.cs.cmu.edu/~avrim/ML10/lect0125.pdf