Sisteme de recunoaştere a formelor – Lab 10

Clasificatori liniari şi algoritmul perceptron

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.

2. Funamente teoretice

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

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. *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. Supspaţiul care separă clasele se numeşte bordura 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, algoritmii fiind uşor extensibili la mai multe clase. Pentru etichete vom folosi *D={-1,1}*.

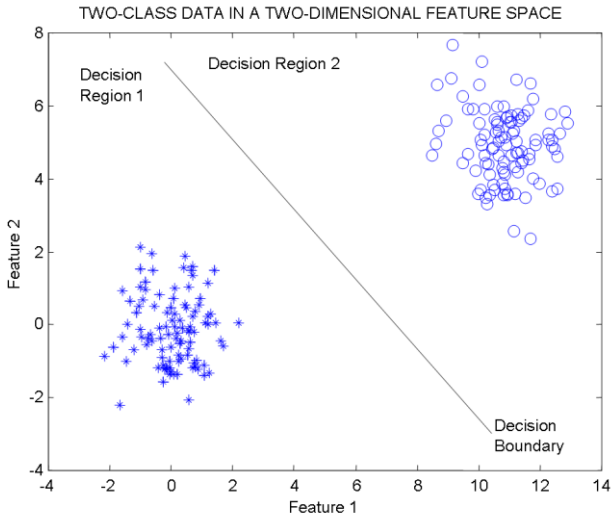


Figure 1. Exemplu de clasificator liniar care separă două clase şi utilizează două trăsături

2.1. Forma generală a unui clasificator liniar

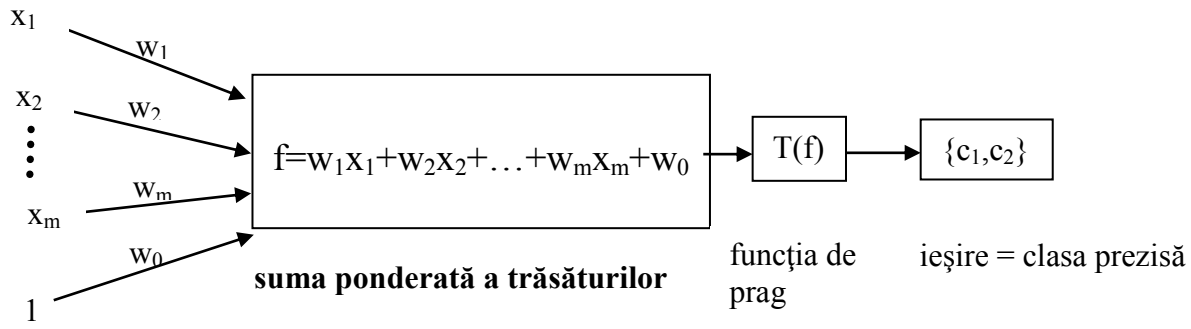
Un clasificator liniar este o combinaţie liniară a trăsăturilor de intrare. Considerând $x \in M_{n\times 1}(R)$ ca vectorul de trăsături putem exprima funcţia de clasificare ca şi:

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

Unde

- w este **vectorul de ponderi**
- w_0 este deplasamentul sau valoarea de prag.

Următoarea schemă ilustrează modul de operare a clasificatorului 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:

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

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

- daca g(**x**) > 0, decidem că **x** aparţine clasei +1;
- daca g(**x**) < 0, decidem că **x** aparţine clasei -1;

sau echivalent:

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

Dacă *g(**x**) = 0*, atunci **x** poate fi atribuit la oricare clasă.

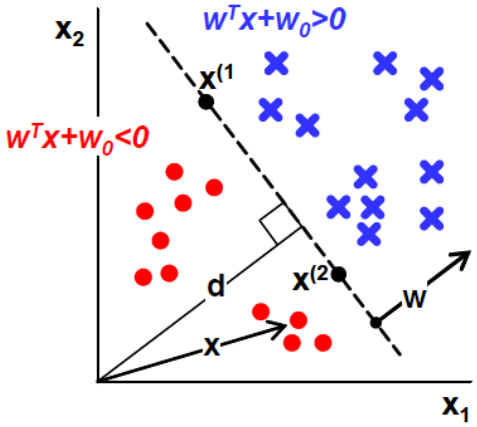


Figure 2. Ilustrarea componentelor din vectorul de ponderi: vectorul w este normala la dreapta de separare iar w_0 este distanţa (cu semn) a dreptei de la origine

2.2. 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 aiba mai multe forme şi trebuie să penalizeze diferenţele dintre predicţie şi clasa adevărată. În cazul algoritmului perceptron adoptă următoarea funcţie de cost:

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

Dacă o instanţă este corect clasificată atunci nu se aplică nici o penalizare, în schimb, dacă se atribuie clasa greşită atunci se penalizează proporţional cu scorul greşit. Semnul expresiei $y_i \bar{\boldsymbol{w}} \cdot \bar{\boldsymbol{x}}_i^T$ este negativ dacă semnul etichetei de clasă este diferit de semnul clasei prezise. Funcţia de cost se mai poate scrie şi ca:

$$L(\bar{\boldsymbol{w}}) = \frac{1}{n} \sum_{i\ clasificat\ greşit} -y_i \bar{\boldsymbol{w}} \cdot \bar{\boldsymbol{x}}_i^T$$

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 $\nabla \boldsymbol{L}(\bar{\boldsymbol{w}})$ este vectorul de gradient calculat în punctul $\bar{\boldsymbol{w}}_k$. Vectorul de gradient are expresia:

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

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

În metoda standard ponderile se schimbă doar după ce am vizitat toate exemplele de antrenare. Această abordare se mai cheama şi batch-update. Putem sa actualizăm ponderile după examinarea fiecărui exemplu de antrenare. În acest algoritmul se numeşte online perceptron. Regula de actualizare devine:

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

Algoritm:
Batch Perceptron

initializare **w**, η, θ, k=0
for *k=1:K*
  L = 0
  for *i=1:n*
    $z_i = \sum_{j=0}^{d} w_j X_{ij}$
    if $z_i \cdot y_i < 0$
      $\frac{\partial L}{\partial \boldsymbol{w}} \leftarrow \frac{\partial L}{\partial \boldsymbol{w}} - y_i \boldsymbol{X}_i$
      $L \leftarrow L - z_i \cdot y_i$
    endif
  endfor
  $L \leftarrow L/n$
  $\frac{\partial L}{\partial \boldsymbol{w}} \leftarrow \frac{\partial L}{\partial \boldsymbol{w}}/n$
  if $\left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\| < L_{limit}$
    break
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial L}{\partial \boldsymbol{w}}$
endfor

Algoritm:
Online Perceptron

initializare **w**, η, θ, k=0
for *k=1:K*
  E = 0
  for *i=1:n*
    $z_i = \sum_{j=0}^{d} w_j X_{ij}$
    if $z_i \cdot y_i < 0$
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta X_i y_i$
      $E \leftarrow E + 1$
    endif
  endfor
  $E \leftarrow E/n$
  if $E < E_{limit}$
    break
endfor

3. Detalii de implementare

Vom folosi puncte 2D care sunt împărţite în 2 clase. Punctele se citesc din imaginile de intrare şi apartenenţa se stabileşte pe baza culorii. Asociem clasa -1 la punctele albastre şi 1 la punctele roşii.
Fiecare punct va fi descris de două trăsături x1 şi x2 care coincid cu coordonatele lor. Astfel, vectorul de trăsătură augmentată are forma $\bar{x}$ = [1 x1 x2] iar vectorul de ponderi are trei componente $\bar{w}$ = [w_0 w_1 w_2].

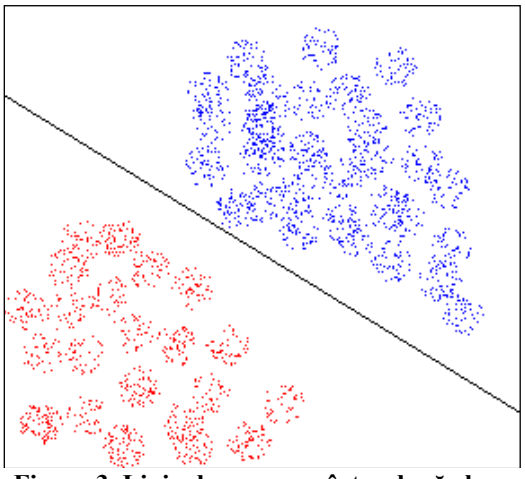


Figure 3. Linia de separare între două clase

4. Lucrare practică

1. Citiţi punctele din imaginea *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:
   $\eta=10^{-4}$, $w_0 = [0.1, 0.1, 0.1]$ şi $E_{limit}=10^{-4}$.
3. Desenaţi linia de separare găsită de algoritm şi dată de ecuaţia:
   $$w_0+w_1x+w_2y = 0.$$
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. De ce nu este corect să alegem vectorul zero la început?

5. References

[1] Richard O. Duda, Peter E. Hart, David G. Stork: *Pattern Classification 2nd ed.*
[2] http://web.engr.oregonstate.edu/~xfern/classes/cs534/notes/perceptron-4-11.pdf
[3] http://en.wikipedia.org/wiki/Gradient_descent