Sisteme de recunoaştere a formelor – Lab 10

Clasificatori compuşi

1. Obiective

Această lucrare are ca scop studierea unui clasificator compus numit AdaBoost (Adaptive Boosting). Vom aplica acest clasificator pentru a calcula cea mai bună regulă de clasificare pentru o problemă cu două clase.

2. Fundamente teoretice

Algoritmul folosit în acest laborator este *Adaptive Boosting* (AdaBoost). Vom aplica acest algoritm pentru o problemă de clasificare ce are ca scop separarea punctelor ce aparțin a două clase diferite (separarea punctelor albastre de cele roşii, ca în figura 1).

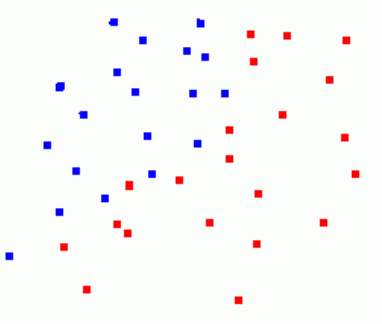

Figura 1. Exemplu de două clase (puncte roşii şi albastre)

Ideea principală a algoritmuluiAdaBoost este să se construiască un clasificator puternic f(P) folosind o combinatie liniară de clasificatori slabi $h^t(P)$.

$$f(P) = \sum_{t=1}^{T} \alpha_t h^t(P)$$

Algoritmul are T iteratii. Fiecare iteratie găseşte clasificatorul slab cel mai bun şi îl adaugă la regula de decizie finală.

2.1. AdaBoost – descrierea algoritmului

Se dă un set de exemple de antrenare de forma $\{(P_1, C_1), (P_2, C_2),\ldots, (P_m, C_m)\}$ unde C_i arată clasa exemplului (punct roşu sau albastru). Fiecare punct P_i este caracteriyat de coordonatele lui carteziene $P_i = (x_i, y_i)$, care sunt văzute ca trăsăturile lui caracteristice.

Fiecare exemplu va primi o pondere. Pentru prima iterație, ponderile sunt egale pentru fiecare punct:

$$D_1(P_i) = \frac{1}{m}, i \in \{1, \ldots, m\}$$

D_1 înseamnă că ne referim la prima iterație a algoritmului, şi $D_1(P_i)$ înseamnă ponderea exemplului P_i la iterația 1.

Algoritmul are următorul pseudocod:

- Se initializează ponderile pentru iteratia 1
- Se initializează numărul de iterații, T
- Pentru t = 1 la T execută
  - Normalizarea ponderilor

$$D_t(P_i) = \frac{D_t(P_i)}{\sum_{j=1}^{m} D_t(P_j)}$$

  - Se calculează cel mai bun clasificator, h^t şi se calculează eroarea lui care se notează ε_t.
  - Dacă $\varepsilon_t >= 0.5$ ne oprim.
  - Se calculează parametrii:

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$$

$$\alpha_t = \frac{1}{2}\log\frac{1}{\beta_t}$$

  - Se iterează pe toate punctele din setul de antrenare. Se coboară ponderea punctelor clasificate corect de h^t (DAR DOAR PENTRU ACESTE PUNCTE, lăsând ponderea punctelor clasificate incorrect nemodificată). Se foloseşte următoarea formulă pentru descreşterea ponderilor:

$$D_{t+1}(P_i) = D_t(P_i)\beta_t$$

- După T iterații, se obține regula finală de clasificare, cu forma:

$$f(P) = sign\left(\sum_{t=1}^{T} \alpha_t h^t(P)\right)$$

2.2. Calcularea clasificatorului slab

Pentru a se calcula clasificatorul slab, se foloseşte o altă regulă de clasificare. Pentru acest laborator se foloseşte cea mai simplă formă de clasificator slab, numită Decision Stump.

$$h(P, f, s, th) = \begin{cases} red_{point}, & if\ s * f(P) < s * th \\ blue_{point}, & otherwise \end{cases}$$

În ecuația de mai sus, P reprezintă un punct caracterizat prin coordonatele P(x,y), f ne arată care din cele două trăsături este luată în considerare (x sau y), s este paritatea inegalității (s poate fi 1 sau -1) şi *th* este un prag (aleasă din valorile posibile pentru trăsătura aleasă).

Configurarea datelor pentru clasificatorul slab este următoarea:

$Sample$	X_{coord}	Y_{coord}	$Weight$
P_1:	x_1	y_1	$D(P_1)$
P_2:	x_2	y_2	$D(P_2)$
P_3:	x_3	y_3	$D(P_3)$
.	.	.	.
P_m:	x_m	y_m	$D(P_m)$

Pseudocodul pentru calcularea celui mai bun clasificator slab pentru iterația curentă este următorul:

- Fie f valoarea corespunzătoare pentru coordonata x.
- Pentru th = x_1 la x_m
  - Se consideră s = 1 şi se obține următoarea regulă de clasificare:

$$h(P(x,y),x,1,th) = \begin{cases} red_{point}, & if\ x < th \\ blue_{point}, & otherwise \end{cases}$$

Se iterează pe toată mulțimea de puncte și se numără câte din acestea sunt clasificate greșit de regula de clasificare. Eroarea de clasificare este dată de suma ponderilor exemplelor clasificate greșit.

- Se consideră s = -1 și se obține regula de clasificare:

$$h(P(x,y),x,-1,th) = \begin{cases} red_{point}, & if\ x > th \\ blue_{point}, & otherwise \end{cases}$$

Se iterează pe toată mulțimea de puncte și se numără câte din acestea sunt clasificate greșit de regula de clasificare. Eroarea de clasificare este dată de suma ponderilor exemplelor clasificate greșit.

- Fie f valoarea corespunzătoare pentru coordonata y.
- Pentru th = y_1 la y_m
  - Se consideră s = 1 și se obține regula de clasificare:

$$h(P(x,y),y,1,th) = \begin{cases} red_{point}, & if\ y < th \\ blue_{point}, & otherwise \end{cases}$$

Se iterează pe toată mulțimea de puncte și se numără câte din acestea sunt clasificate greșit de regula de clasificare. Eroarea de clasificare este dată de suma ponderilor exemplelor clasificate greșit.

  - Se consideră s = -1 și se obține regula de clasificare:

$$h(P(x,y),y,-1,th) = \begin{cases} red_{point}, & if\ y > th \\ blue_{point}, & otherwise \end{cases}$$

Se iterează pe toată mulțimea de puncte și se numără câte din acestea sunt clasificate greșit de regula de clasificare. Eroarea de clasificare este dată de suma ponderilor exemplelor clasificate greșit.

- Se returnează regula de clasificare ce are valoarea minimă a erorii.

4. Exerciții

Se dă o mulțime de puncte aparținând de două clase. Să se aplice algoritmul de boosting pentru a găsi clasificatorul ce împarte cel mai bine mulțimea în două grupuri. Să se deseneze regula de clasificare finală.

Note:

- Pentru numărul de clasificatori slabi din algoritmul AdaBoost, încercați următoarele valori: 5, 15, 20, 25.
- O soluție pentru a afișa rezultatul este să se ia toate punctele din imagine care nu sunt clasificate inițial (punctele albe) și să se aplice clasificatorul pe ele. Colorați punctual în funcție de clasa lui, folosind verde și galben. Modificați paleta imaginii (index 0 este roșu, index 1 este albastru, folosiți 3 și 4 pentru noile culori). Verde este (0, 255, 0) și galben este (255, 255, 0).

5. Bibliografie

[1] Julien Meynet, Fast Face Detection Using AdaBoost 2003
[2] Robert E. Schapire, The Boosting Approach to Machine Learning, An Overview, 2001