Algoritmi de boosting. AdaBoost

Algoritmii de *boosting* constau în metode bazate pe ansambluri ce au ca scop crearea unui clasificator puternic (*strong classifier*) folosind rezultatele mai multor clasificatori slabi (*weak classifier*). Un clasificator slab generează rezultate cu acuratețe redusă (eroarea de clasificare este semnificativă, peste 20–30%), dar, prin combinarea adecvată a mai multor astfel de clasificatori, eroarea de clasificare se poate reduce semnificativ.

AdaBoost este unul din cei mai populari algoritmi de boosting. Metoda începe cu o etapă de antrenare în cadrul căreia un clasificator slab se aplică în repetate rânduri pe un set de date de antrenare (o mulțime de instanțe cărora li se cunoaște deja clasa de încadrare). Fiecărei instanțe i se asociază câte o pondere, inițial aceste ponderi fiind egale. În cadrul unei iterații a algoritmului AdaBoost se determiă eroarea clasificatorului slab și se ajustează ponderile astfel încât instanțele clasificate greșit să aibă ponderi mai mari. Astfel, la următoarea iterație, clasificatorul slab va prioritiza instanțele clasificate incorect, tendința fiind de a reduce eroarea de clasificare. Rezultatul final al algoritmului este o combinație liniară a rezultatelor individuale ale clasificatorilor slabi de la fiecare iterație, ponderate de erorile fiecărui clasificator slab.

Decision Stump

Clasificatorul slab pe care îl vom utiliza și îmbunătăți este un arbore cu un singur nivel numit Decision Stump (prescurtat DS). În cadrul unui DS se ia o singură decizie funcție de un singur atribut al setului de date vizat. Astfel, un DS este un arbore de decizie care are doar rădăcina și un singur set de noduri frunză. Decizia singulară care se adoptă în cazul unui DS este adesea insuficientă pentru a realiza o clasificare riguroasă, așadar eroarea de clasificare a unui DS este în general, semnificativă.

În continuare, vom utiliza DS binari, deoarece AdaBoost este gândit, la rândul său, în ideea de a oferi decizii binare. Un astfel de arbore binar necesită un atribut funcție de care se ia decizia și o valoare a sa care împarte spațiul de instanțe în două părți a căror dimensiune depinde de valoarea aleasă.

Considerăm setul de date din Tabelul 1. Pe baza acestuia, vom genera un DS cu două ramuri, care va subdiviza setul de date funcție de valoarea medie a unuia dintre atributele A1, A2. Pentru a determina atributul utilizat, vom calcula indicele Gini al fiecărui atribut (Ec. 1) și îl vom alege pe cel cu indicele minim.

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2 \qquad (1)$$

Unde:

S este setul de date pentru care se calculează indicele Gini

C este numărul de clase (în setul de date din tabelul anterior sunt două clase, așadar, C = 2)
p_i este probabilitatea de apariție a clasei *i* în setul de date S)

Tabelul 1. Set de date cu două atribute numerice. Instanțele se încadrează în două clase cu etichetele -1 și 1

A1	A2	Class
4.2	4	-1
2.1	2	-1
6	5	1
6	5.5	-1
1.5	6.5	1
3.7	7.5	1
1.5	3.2	1
4	3.5	-1
1.7	7.8	1
3.5	5.5	-1

Fie $\bar{A}1$ și $\bar{A}2$ valorile medii ale celor două atribute. Presupunem că A este atributul din nodul DS. Astfel, DS împarte setul de date în două partiții, care corespund $A < \bar{A}$, respectiv $A \geq \bar{A}$. Dorim să determinăm $A \in \{\}$ pentru care dintre atributele A1, A2 partiționarea generează submulțimi care minimizează indicele Gini.

Pentru A1:

$GiniA1_{A1 < \bar{A}1} = 1 - P^2(Class = -1 \mid A1 < \bar{A}1) - P^2(Class = 1 \mid A1 < \bar{A}1)$

$GiniA1_{A1 \geq \bar{A}1} = 1 - P^2(Class = -1 \mid A1 \geq \bar{A}1) - P^2(Class = 1 \mid A1 \geq \bar{A}1)$

$GiniA1 = P(A1 < \bar{A}1) * GiniA1_{A1 < \bar{A}1} + P(A1 \geq \bar{A}1) * GiniA1_{A1 \geq \bar{A}1}$

Din calcule rezultă GiniA1 = 0.416 . În mod similar, GiniA2 = 0.48 . Prin urmare, DS va fi arborele reprezentat în Fig.1.

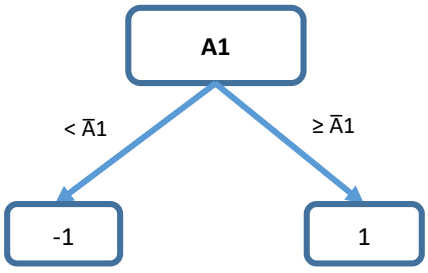

Fig 1. Decision Stump pentru datele din Tabelul 1

Eroarea de clasificare se determină aplicând DS pe datele de antrenare. Rezultatul este ilustrat în Tabelul 2, unde se observă faptul că eroarea de clasificare este 0.7 .

Tabelul 2. Rezultatul clasificării folosind decision stump-ul din Fig. 1

A1	A2	Class	DS
4.2	4	-1	1
2.1	2	-1	-1
6	5	1	1
6	5.5	-1	1
1.5	6.5	1	-1
3.7	7.5	1	1
1.5	3.2	1	-1
4	3.5	-1	1
1.7	7.8	1	-1
3.5	5.5	-1	1

AdaBoost

Pentru îmbunătățirea rezultatului anterior, vom aplica DS în repetate rânduri pe setul de date de antrenare, în cadrul mai multor iterații. Vom asigna câte o pondere pentru fiecare instanță din setul de date. Inițial, toate instanțele vor avea aceeași pondere, 1/n, unde n = nr. de instanțe (Tabelul 3).

Tabelul 3. Setul de date ponderat. Fiecărei instanțe i se asociază o pondere inițializată cu 1/nr_instanțe

A1	A2	W	Class
4.2	4	0.1	-1
2.1	2	0.1	-1
6	5	0.1	1
6	5.5	0.1	-1
1.5	6.5	0.1	1
3.7	7.5	0.1	1
1.5	3.2	0.1	1
4	3.5	0.1	-1
1.7	7.8	0.1	1
3.5	5.5	0.1	-1

La fiecare iterație, ponderile se vor modifica astfel încât instanțele clasificate greșit la iterația t-1 să aibă ponderi mai mari la iterația t. Astfel, instanțele care au generat erori la iterația t-1 for fi luate în considerare în măsură mai mare la iterația t, în ideea de a reduce eroarea de clasificare.

Considerăm următoarele notații:

T – numărul de iterații

t – o iterație oarecare, t = 1..T

x – o instanță oarecare (un rând din tabelele cu setul de date)

X – mulțimea instanțelor din setul de date

n – numărul de instanțe din setul de date de antrenare

x_i – o instanță din setul de date, $x_i \in X$, i = 1..n

y – valoarea asociată unei clase: $y \in \{-1, 1\}$

y_i – clasa instanței x_i din setul de date de antrenare

$w_t(i)$ – ponderea de la iterația t a instanței x_i

$h_t(x)$ – rezultatul clasificării instanței x folosind clasificatorul slab (altfel spus, clasa care se obține aplicând DS pe x), care poate avea valorile 1 sau -1.

Algoritmul AdaBoost poate fi descris astfel:

Datele inițiale sunt de forma $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, unde $x_i \in X$, și $y_i \in \{-1, 1\}$, i = 1..n
Se inițializează ponderile $w_1(i) = 1/n$
Se generează un prim DS conform celor descrise anterior.
Pentru t = 1..T:

- Se aplică DS pe datele de antrenare x_i și se obțin valorile pentru $h_t(x_i)$
- Se determină eroarea ponderată a clasificatorului (suma ponderilor instanțelor clasificate greșit / suma tuturor ponderilor)

  $$\varepsilon_t = P_{i \sim w_t}(h_t(x_i) \neq y_i)$$

- Se determină un coeficient α_t al iterației curente astfel:

  $$\alpha_t = \frac{1}{2} ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

- Se actualizează ponderile astfel:

  $$w_{t+1}(i) = w_t(i)\, e^{-\alpha_t\, y_i\, h_t(x_i)}$$

- Se normalizează ponderile astfel obținute (se împarte fiecare pondere la suma tuturor ponderilor)
- Se genereaza un nou DS cu noile valori ale ponderilor

Pentru o instanță x oarecare, clasa se determină ca fiind o combinație a tuturor DS-urilor generate în decursul iterațiilor, astfel:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x)\right)$$

Rezultatele aplicării AdaBoost pe datele din laborator pentru trei iterații (suficiente pentru a minimiza eroarea) se prezintă în Tabelul 4.

Tabelul 4. Rezultatul aplicării AdaBoost în decursul a trei iterații. W_0 sunt ponderile inițiale, AB_0 sunt predicțiile generate inițial (aceleași ca în Tabelul 2). W_i și AB_i , i=1..3, sunt ponderile, respectiv predicțiile aferente celor trei iterații

A1	A2	W_0	W_1	W_2	W_3	Class	AB_0	AB_1	AB_2	AB_3
4.2	4	0.1	0.084	0.065	0.044	-1	1	1	-1	-1
2.1	2	0.1	0.137	0.107	0.072	-1	-1	-1	-1	-1
6	5	0.1	0.137	0.185	0.125	1	1	1	-1	1
6	5.5	0.1	0.084	0.113	0.185	-1	1	1	1	1
1.5	6.5	0.1	0.084	0.065	0.044	1	-1	-1	1	1
3.7	7.5	0.1	0.137	0.107	0.072	1	1	1	1	1
1.5	3.2	0.1	0.084	0.113	0.185	1	-1	-1	-1	-1
4	3.5	0.1	0.084	0.065	0.044	-1	1	1	-1	-1
1.7	7.8	0.1	0.084	0.065	0.044	1	-1	-1	1	1
3.5	5.5	0.1	0.084	0.113	0.185	-1	1	1	1	1
Eroare							0.7	0.7	0.4	0.3

Cerințe

1. Implementați un arbore DS. Aplicați-l pe setul de date din laborator și determinați eroarea de clasificare.
2. Implementați AdaBoost folosind DS pe post de clasificator slab. Afișați erorile clasificatorului slab de la fiecare iterație, precum și eroarea finală de clasificare a algoritmului.

   Observație: Atunci când instanțele din setul de date sunt ponderate, toate calculele efectuate se vor realiza ponderat. Aceasta implică următoarele:

Probabilitățile se calculează astfel:

- neponderat: $P = \frac{numar\_cazuri\_favorabile}{numar\_cazuri\_posibile}$

- ponderat: $P = \frac{suma\_ponderilor\_cazurilor\_favorabile}{suma\_ponderilor\_cazurilor\_posibile}$

Media unui set de valori x_i, cu ponderile w_i, i ∈ {1, ... ,n}:

- neponderat: $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

- ponderat: $\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$