Curs 12:

Metode de tip ansamblu
(meta-modele)

Structura

- Motivaţie
- Ideea modelelor de tip ansamblu
- Colecţii de modele (bucket of models)
- Colecţii de arbori aleatori (random forests)
- Strategii de agregare a modelelor
  - Bagging
  - Boosting
  - Stacking

Motivaţie

Reminder:

- Scopul unui clasificator este estimarea relaţiei dintre atributul de clasă şi celelalte atribute
- construirea unui clasificator se bazează pe:
  - Un set de antrenare
  - Ipoteze asupra modelului de clasificare (de exemplu suprafaţa de decizie este liniară sau liniară pe porţiuni)
- Notaţii

y=f(x)= clasa aferentă datei x

D={$(x_1,y_1),(x_2,y_2),\ldots,(x_L,y_L)$} = setul de antrenare

g(x;D) = răspunsul estimat de către modelul construit pe baza setului de antrenare

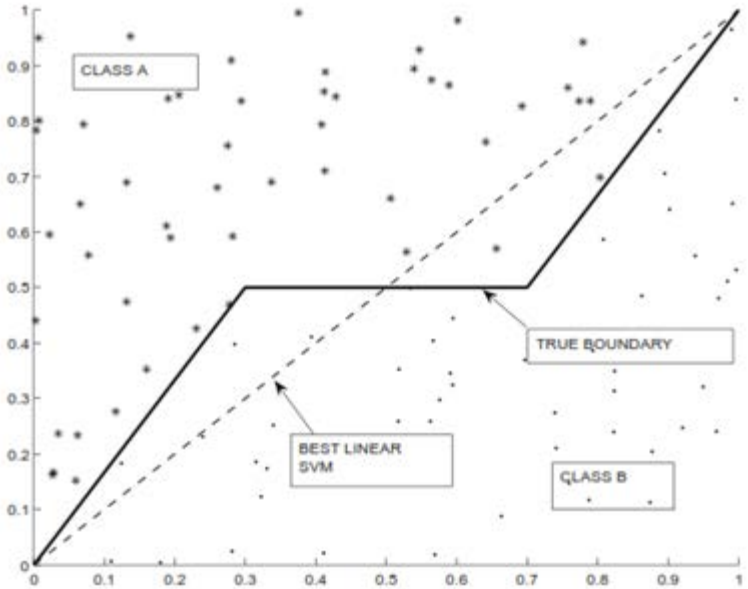
MSE = eroare medie pătratică

Extragerea modelului din date: estimarea parametrilor modelului astfel ca MSE să fie minimizată

Motivaţie

Componente ale erorii:

- Deplasare (Bias) = inabilitatea clasificatorului de a clasifica corect cauzată de limitările modelului (e.g. modelul are suprafete de decizie liniare iar cele reale sunt neliniare)



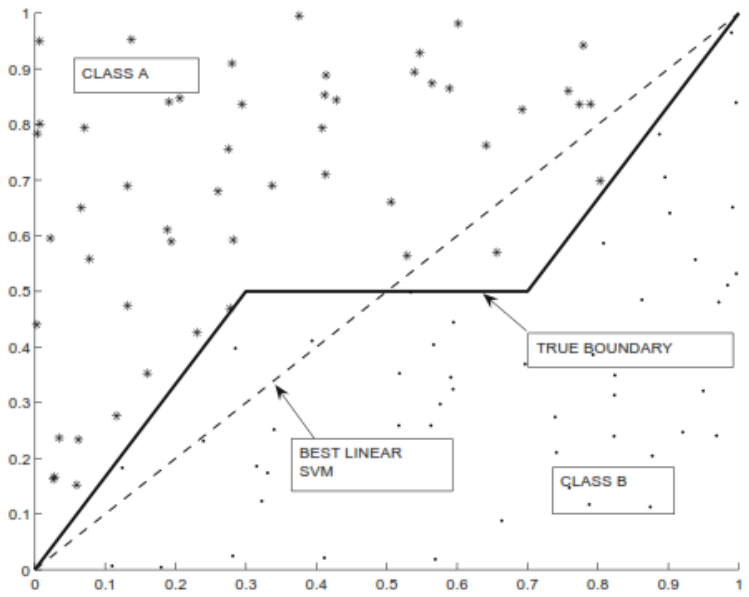
(a) bias

Aggarwal, Data Mining. The textbook, 2015

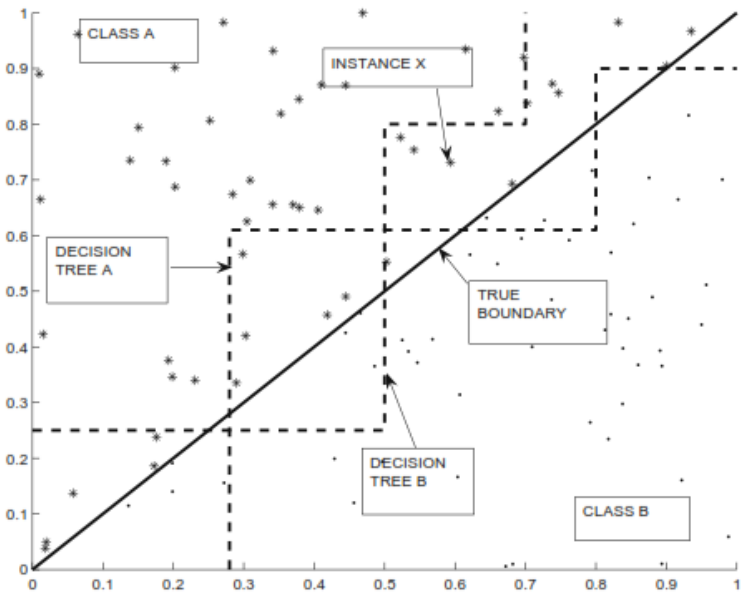
Motivaţie

Componente ale erorii:

- Deplasare (Bias) = inabilitatea clasificatorului de a clasifica corect cauzată de limitările modelului (e.g. modelul are suprafete de decizie liniare iar cele reale sunt neliniare)
- Varianţa (Variance) = cauzată de volumul limitat de date de antrenare (e.g. doi clasificatori bazati pe acelaşi model dar antrenate pe seturi diferite au performanţe diferite)



(a) bias



(b) variance

C. Aggarwal, Data Mining. The textbook, 2015

Motivaţie

- Deplasare (Bias) = inabilitatea clasificatorului de a clasifica corect cauzată de limitările modelului (e.g. modelul are suprafete de decizie liniare iar cele reale sunt neliniare)
- Varianţa (Variance) = cauzată de volumul limitat de date de antrenare (e.g. doi clasificatori bazati pe acelaşi model dar antrenate pe seturi diferite au performanţe diferite)

$$MSE = \frac{1}{L}\sum_{i=1}^{L}(y_i - g(x_i;D))^2 = \frac{1}{L}\sum_{i=1}^{L}(y_i^2 - 2y_i g(x_i;D) + g(x_i;D)^2)$$

$$E_D(MSE) = \frac{1}{L}\sum_{i=1}^{L}(y_i^2 - 2y_i E_D(g(x_i;D)) + E_D(g(x_i;D)^2) + (E_D(g(x_i;D)))^2 - (E_D(g(x_i;D)))^2)$$

$$= \frac{1}{L}\sum_{i=1}^{L}((y_i - E_D(g(x_i;D)))^2 + (E_D(g(x_i;D)^2) - (E_D(g(x_i;D)))^2)$$

deplasare

(influenţată de model)

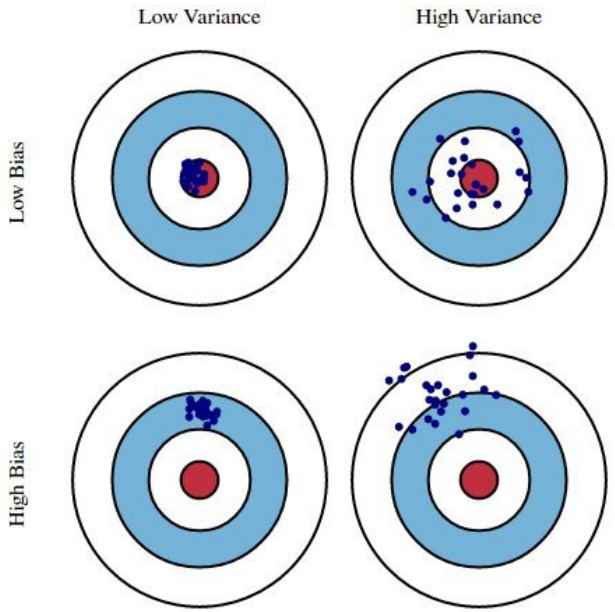
Varianţa răspunsurilor produse de clasificatori antrenati pe diferite seturi de date (influenţată de date)

Motivaţie

- Deplasare (Bias) = inabilitatea clasificatorului de a clasifica corect cauzată de limitările modelului (e.g. modelul are suprafete de decizie liniare iar cele reale sunt neliniare)
- Varianţa (Variance) = cauzată de volumul limitat de date de antrenare (e.g. doi clasificatori bazati pe acelaşi model dar antrenate pe seturi diferite au performanţe diferite)

Obs:

- Un model cu valoare mare pt bias va genera erori chiar dacă se modifică setul de antrenare
- Un model cu varianţă mare va produce rezultate inconsistente când este antrenat pe diferite seturi de date



Cum se poate reduce eroarea?

- Reducând deplasarea sau reducând varianţa
- Este posibil să se reducă ambele? Cum?

Motivaţie

Bias vs variance

- Modele simple (e.g. Modele liniare, modele bazate pe reguli simple, arbori de decizie simpli, naïve Bayes)
  - Deplasare mare (datorită faptului că suprafaţa de decizie este prea simplă
  - Varianţă mică (modelele simple sunt robuste în raport cu schimbările din seturile de date; modelele simple suferă rar de supra-antrenare)

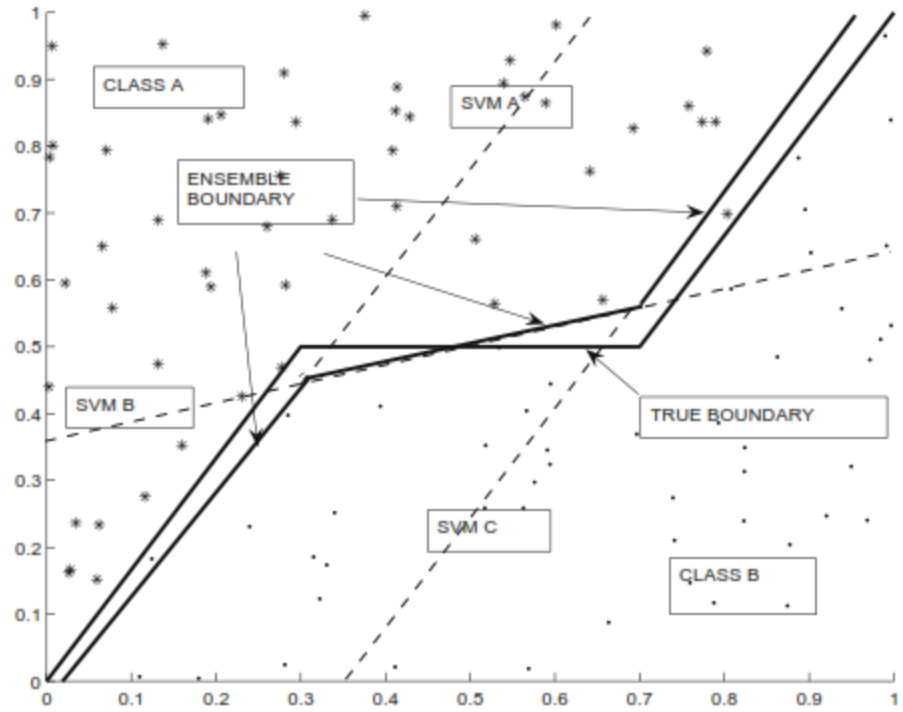
- Model complexe (e.g. Reţele neuronale/ arbori de decizie cu multe nivele)
  - Deplasare mică (întrucât modelează bine suprafeţele de decizie)
  - Varianţă mare (senzitive la modificările în setul de date; pot fi afectate de supra-antrenare)

Dacă se utilizează un singur model este necesară găsirea unui compromis între deplasare şi varianţă

Prin combinarea mai multor modele pot fi reduse simultan atât varianţa cât şi deplasarea

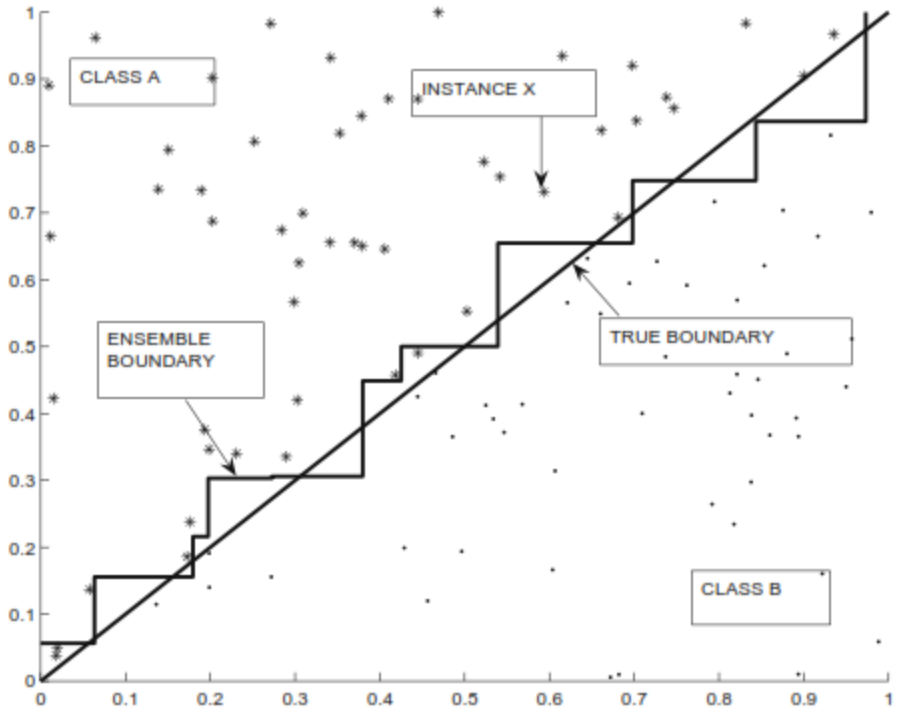
Motivaţie

Combinarea mai multor modele → model de tip ansamblu



(a) bias

Combinând 3 SVM-uri liniare se poate ajunge la suprafaţă neliniară



(b) variance

Combinând arbori de decizie antrenaţi pe seturi diferite se poate reduce varianţa

C. Aggarwal, Data Mining. The textbook, 2015

Este util să se combine modelele?

O analiza probabilistă simplă

- Considerăm 25 de clasificatori binari
  - Fiecare clasificator are o anumită rată de eroare, ε = 0.35 (probabilitatea să producă un răspuns eronat)
  - Presupunem că cei 25 de clasificatori sunt independenţi
  - Probabilitatea ca un ansamblu constituit din cei 25 de clasificatori (în cazul unei reguli de agregare bazată pe regula majorităţii = răspunsul dat de cel puțin 13 clasificatori):

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Modele de tip ansamblu

Cum se construiesc modelele de tip ansamblu?

- Prin construirea mai multor modele (pe baza unor diferite ipoteze structurale) pornind de la acelaşi set de date - corespunde ansamblelor centrate pe modele

- Prin antrenarea aceluiaşi model folosind diferite seturi de date (extrase aleator dintr-un set global de date) – aceasta corespunde ansamblelor centrate pe date

Cum se pot utiliza modelele de tip ansamblu?

- Pentru o anumită dată de intrare se aplică toate modelele din ansamblu iar rezultatul final se obţine prin agregarea rezultatelor prin:
  - Votare (cel mai frecvent răspuns) – în cazul problemelor de clasificare
  - Mediere – în cazul problemelor de regresie

Modele de tip ansamblu

Algoritm generic pt un model de tip ansamblu

- Input: set de date D; set de metode/ algoritmi $\{A_1, A_2, \ldots A_r\}$
- Output: un model de tip ansamblu constând din K modele individuale $\{M_1, M_2, \ldots, M_K\}$

REPEAT

- k=1
- select an algorithm A from the set $\{A_1, A_2, \ldots A_r\}$
- create a training dataset D_k (by sampling from D)
- construct the model M_k by applying algorithm A to the dataset D_k
- k=k+1
- Evaluate the performance of the current ensemble $\{M_1, M_2, \ldots, M_k\}$ (pt fiecare model se utilizează datele care nu au fost utilizate în faza de antrenare)

UNTIL desired performance

Modele de tip ansamblu

Algoritm generic pt un model de tip ansamblu

- Input: set de date D; set de metode/ algoritmi $\{A_1, A_2, \ldots A_r\}$
- Output: un model de tip ansamblu constând din K modele individuale $\{M_1, M_2, \ldots, M_K\}$

Cazuri particulare:

- Algoritmi diferiţi, un set de antrenare (e.g. colecţie de modele - bucket of models)

- Acelaşi algoritm, diferite seturi de antrenare
  - Bagging (bootstrap aggregation)
  - Random forests
  - Boosting

Colecţii de modele – bucket of models

Idee de bază: mai mulţi algoritmi, un set de date → un meta-model agregat

Varianta 1:

- Se antrenează diferite modele utilizând acelaşi set de date
- Rezultatele produse de modelele componente se agreghează prin:
  - Regula majorităţii
  - Medierea rezultatelor de la modelele componente

Varianta 2:

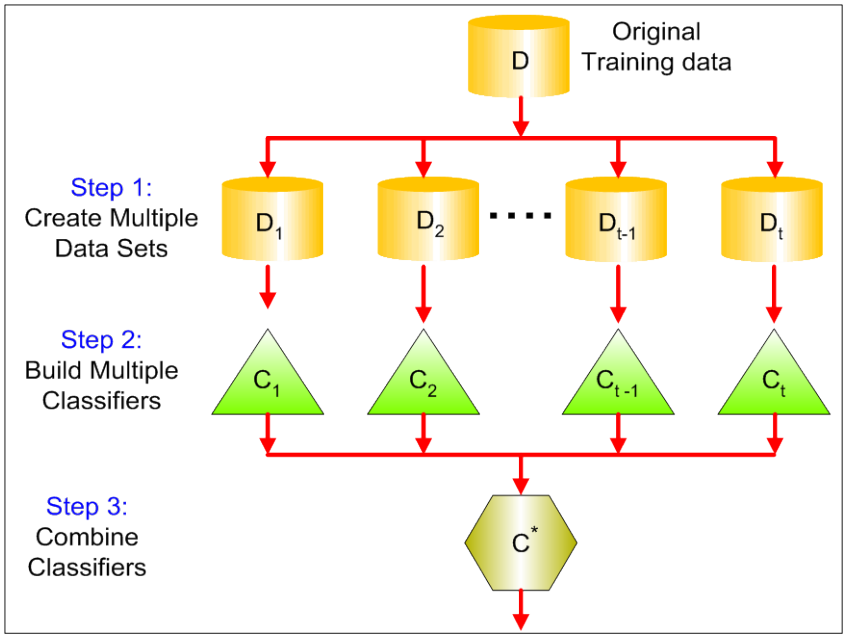
- Setul de date D se divide în două subseturi A şi B
- Se antrenează toate modelele folosind setul A
- Se selectează modelul cu cel mai bun comportament pentru subsetul B
- Se reantrenează modelul selectat pentru întregul set de date D

Obs:

- Reduce deplasarea întrucât pt diferite părţi ale setului de date ar putea fi diferite modele mai adecvate

Bagging

- Idee de bază: un algoritm, mai multe seturi de date → mai mulţi clasificatori



Seturi de antrenare:

- Obţinute prin selecţie cu revenire din setul complet de date D
- Dacă setul complet are L elemente atunci probabilitatea ca un anumit element să fie selectat este $1-(1 - 1/L)^L$;

- Exemplu:

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

[Slides by Kumar/ Introduction to Data Mining, 2004]

Bagging

Impact bagging:

- Reduce varianţa

- Nu reduce deplasarea (întrucât acelaşi model este folosit pt toate seturile de antrenare nu sunt eliminate limitările modelului)

Obs:

- Reducerea varianţei este asigurată doar dacă modelele din ansamblu sunt independente
- Limitarea corelaţiei dintre modele se poate obţine prin introducerea unor elemente aleatoare în construirea modelelor (pe lângă aleatoritatea din selecția datelor de antrenare) → random forests

Random forests

Random forest = colecţie de arbori aleatori construiţi folosind tehnica de la bagging (seturile de antrenare sunt construite prin selecţie aleatoare cu revenire)

Construire random forest:

- Se construieşte un random tree pt fiecare set de antrenare

Utilizare random forest:

- Se aplică fiecare arbore datei de intrare
- Se selectează răspunsul dominant (schemă simplă de votare)

Random forests

Random tree = arbore de decizie construt folosind random-split

Etape:

- Dacă numărul de exemple din setul de antrenare este L, atunci se selectează L cazuri aleator – dar cu revenire. Acest set este folosit pt construirea arborelui
- In cazul a M variabile de intrare se fixează o valoare m<<M iar la ramificarea fiecărui nod se selectează aleator doar m variable şi cea mai bună dintre ele este folosită pt a ramifica nodul. Valoarea lui m este păstrată constantă până la construirea arborelui
- Fiecare arbore este extins cât de mult se poate pt a asigura o valoare mică a deplasării. Nu se foloseşte pruning (ceea ce favorizează reducerea deplasării).

[Leo Breiman https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm]

Random forests

Obs: eroarea arborilor aleatori depinde de 2 elemente:

- Corelaţia dintre oricare 2 arbori – cu cât corelaţia este mai mare cu atât eroarea este mai mare – asigurată de selecția aleatoare a variabilelor candidat pt splitting

- Puterea fiecărui arbore din pădure. Un arbore cu eroare mică are putere mare. Cu cât este mai mare puterea arborilor din pădure cu atât eroarea pe ansamblu este mai mică - asigurată de absența etapei de pruning

[Leo Breiman https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm]

Random forests

Obs: Influenţa lui m (numărul de atribute selectate în procesul de ramificare)

- Reducerea lui m reduce atât corelaţia cât şi puterea.
- Creşterea lui m conduce la creşterea corelaţiei şi a puterii
- E necesar un compromis – se alege valoarea lui m care conduce la eroare mică

Obs: estimarea calităţii se face folosind datele care nu au fost selectate la construirea arborelui (nu e necesară validare încrucişată) – tehnica "out-of-bag" (OOB)

[Leo Breiman https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm]

Boosting

Ideea de bază:

- Fiecărei instanţe din setul de antrenare i se asociază o pondere care poate fi utilizată
    - direct în cadrul modelului (dacă acesta permite utilizare ponderilor)
    - sau în definirea unor probabilităţi de selecţie
- Ponderile pot fi adaptive (instanţele clasificate incorect au asociate valori mai mari ale ponderilor)

Abordare:

- La început toate instanţele au asociate aceleaşi ponderi
- Pe parcursul procesului de antrenare:
    - Pt instanţele clasificate incorect se măreşte valoarea ponderii
    - Pt instanţele clasificate corect se micşorează valoarea ponderii

Obs: Se presupune că principala componentă a erorii este deplasarea şi se încearcă reducerea acesteia prin acordarea unei importanţe mai mari datelor clasificate incorect

AdaBoost

Algoritm de antrenare:

- Input: algoritm de clasificare de bază: A; set de date: D
- Output:
  - Set de modele de clasificare (M_1,…, M_T)
  - Set de ponderi corespunzătoare modelelor

- Ideea de bază

- La fiecare pas t al algoritmului, se obţine o componentă a ansamblului (M_t) – vezi slide următor
- Clasificatorul de bază e de regulă un clasificator simplu fără putere prea mare de discriminare (weak classifier) - e suficient ca performanța lui să fie ușor mai bună ca cea a unui clasificator care ia decizii aleator (echivalent cu aruncarea unei monede); poate fi un arbore binar cu un nivel

[Recommended: R. Schapire, Explaining AdaBoost, http://rob.schapire.net/papers/explaining-adaboost.pdf]

AdaBoost

AdaBoost (A,D)

t=1; initialize the weights of the training instances: w(t,i)=1/L for all i=1..L

REPEAT

t=t+1; construct M_t using the current values of the instances' weights

compute the weighted error rate of model M_t on D ($\varepsilon(t)$)

compute the model weight $\alpha(t)=\ln((1-\varepsilon(t))/\varepsilon(t))/2$

FOR i=1,L **DO**

IF x_i is wrongly classified **THEN** $w(t+1,i)=w(t,i)*\exp(\alpha(t))$

ELSE $w(t+1,i)=w(t,i)*\exp(-\alpha(t))$

FOR i=1,L **DO** w(t+1,i)=w(t+1,i)/sum(w(t+1,j),j=1..L)

UNTIL (t>=T) or ($\varepsilon(t)$=0) or ($\varepsilon(t)$>=0.5)

Obs: dacă la oricare dintre rundele intermediare rata de eroare este mai mare decât 50% în loc să se oprească algoritmul se readuc ponderile la 1/L şi se repetă procedura

AdaBoost

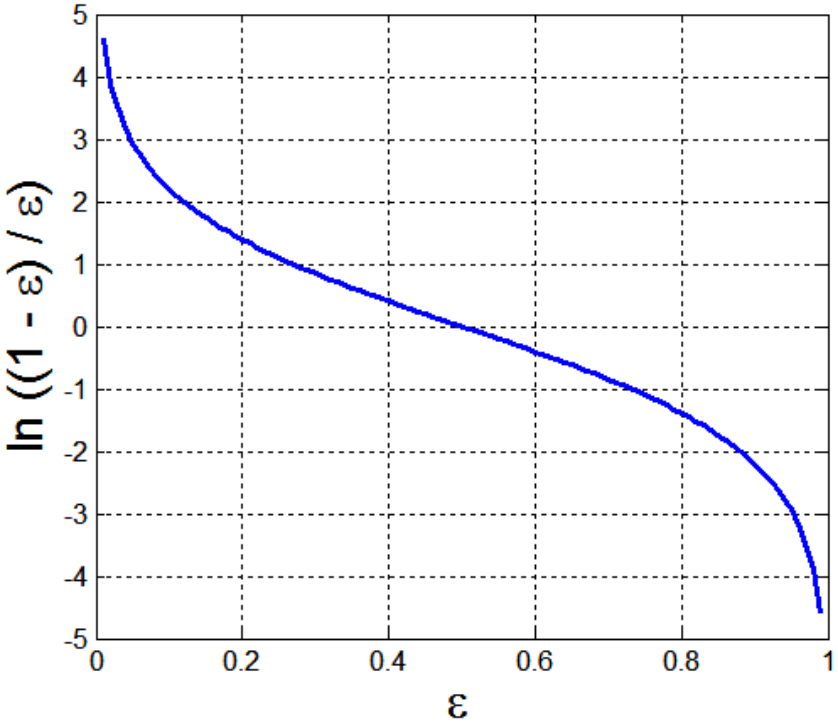
Câteva detalii:

- Rată ponderată de eroare:

$$\varepsilon_t = \frac{1}{L}\sum_{i=1}^{L} w_i \delta\left(M_t(x_i) \neq y_i\right)$$

- Importanţa (pondere) unui model/ clasificator:

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



AdaBoost

Etapa de clasificare

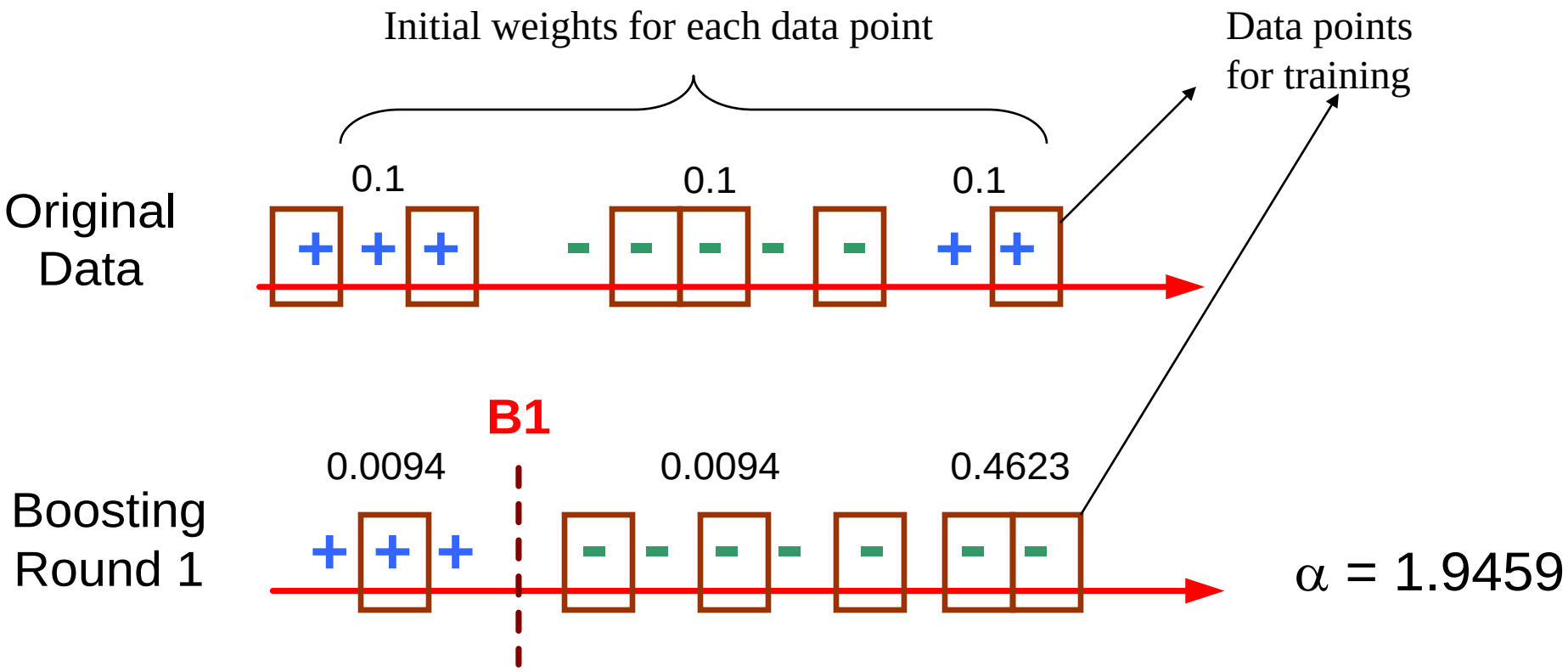
(în cazul clasificatorilor binari ce produc valori în {-1,1})

- Se aplică fiecare dintre componentele ansamblului (M_1, M_2,…, M_T) şi se colectează rezultatele (r_1,r_2,…r_T) (in {-1,1})

- Se combină rezultatele:

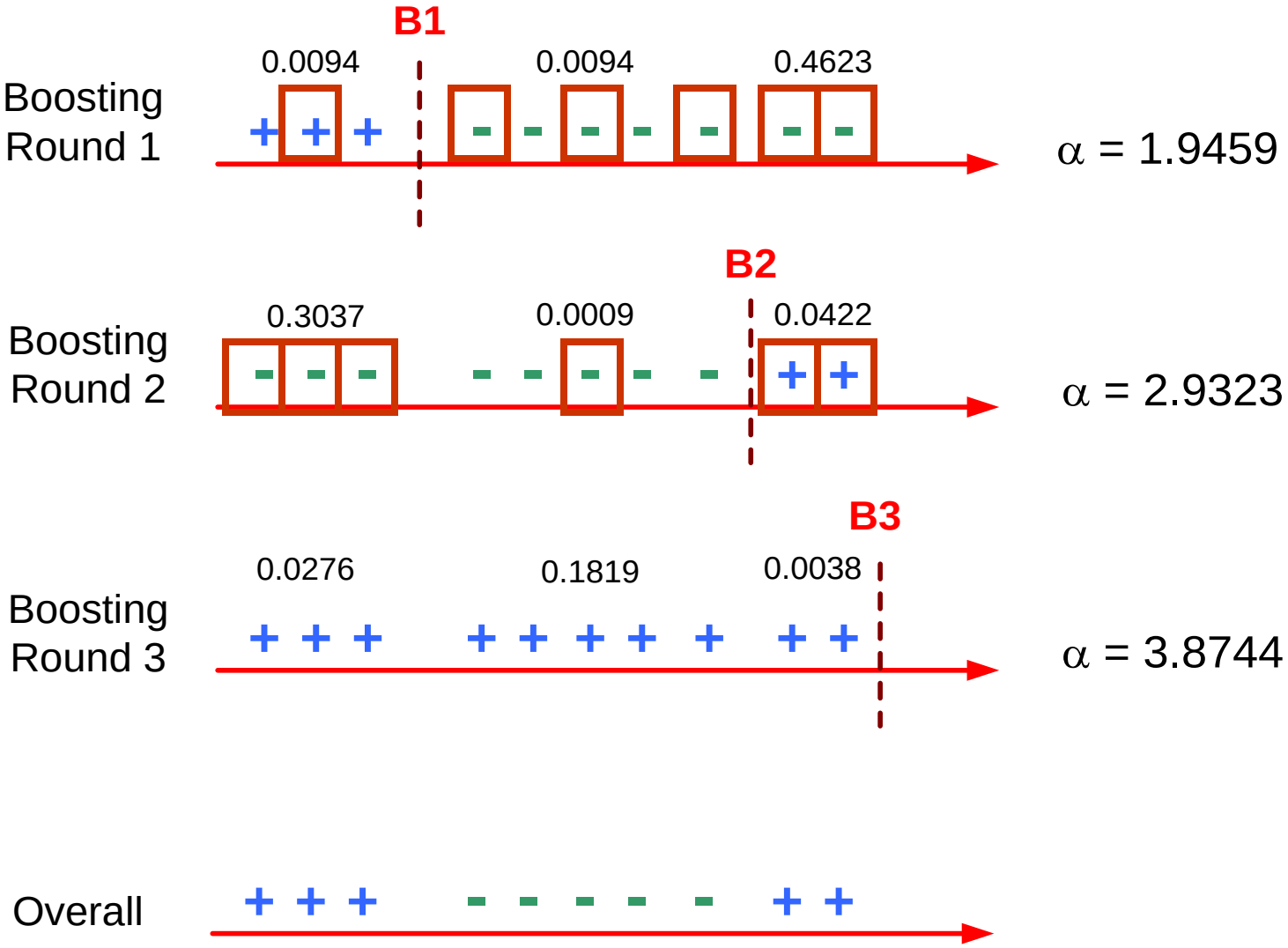
  - calcul $r=\alpha_1 r_1 + \alpha_2 r_2 +\ldots+ \alpha_T r_T$
  - IF r<0 THEN return -1 ELSE return +1

AdaBoost



[Slides by Kumar/ Introduction to Data Mining, 2004]

AdaBoost


B1
0.0094
0.0094
0.4623
Boosting Round 1
α = 1.9459
B2
0.3037
0.0009
0.0422
Boosting Round 2
α = 2.9323
B3
0.0276
0.1819
0.0038
Boosting Round 3
α = 3.8744
Overall

Stacking

Idee de bază: două nivele de clasificare

Etape principale:

- Se divide setul de date D in două subseturi A şi B

- Primul nivel: se antrenează un ansamblu de k clasificatori bazaţi pe A (poate fi o colecţie de clasificatori, se pot baza pe bagging sau pe k runde de boosting)

- Second level:
  - Se determină k ieşiri (etichete ale claselor) ale clasificatorilor antrenaţi la primul nivel pentru fiecare dintre instanţele subsetului B
  - Se construieşte un set de date având ca atribute de intrare aceste k ieşiri şi ca atribut de clasă eticheta corectă a instanţei corespunzătoare din subsetul B
  - Se antrenează un clasificator pe acest set nou de date

Stacking

Obs:

- Rezultatul de la stacking este un set de k clasificatori de prim nivel şi un clasificator combinat
- Pentru o instanţă de test, primul nivel de clasificatori este utilizat o nouă instanţă k-dimensională pe când al doilea clasificator furnizează rezultatul corespunzător instanţei transformate
- Atributele originale pot fi combinate cu noile atribute la construirea clasificatorului corespunzător celui de al doilea nivel; este de asemenea posibil ca cele k atribute noi să fie probabilităţi şi nu etichete de clase
- Tehnica stacking permite reducerea ambelor componente ale erorii (întrucât al doilea nivel învaţă din erorile diferitelor componente ale ansamblului)

Sumar

Impactul metodelor de tip ansamblu asupra componentelor erorii

- Bagging şi random forests au fost proiectate să reducă varianţa
- Boosting şi stacking au fost proiectate să reducă ambele componente

Extinderea ideii de ansamblu în contextul tehnicilor de clustering

- Aceeaşi idee ca şi la clasificare:
  - Aplică diferite metode de clustering (sau aceeaşi metodă dar folosind diferite valori ale parametrilor)
  - Agregare rezultate folosind algoritmi de clustering pt hipergrafuri (datele reprezintă noduri şi fiecare cluster reprezintă o hipermuchie)