

Curs 11:

Metode de tip ansamblu (meta-modele)

Structura

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

Motivație

Reminder:

- Scopul unui clasificator este estimarea relației dintre atributul de clasă și celelalte attribute
- construirea unui clasificator se bazează pe:
 - Un set de antrenare
 - Ipoteze asupra modelului de clasificare (de exemplu suprafața de decizie este liniară sau liniară pe porțiuni)
- **Notații**

$y=f(x)$ = clasa aferentă datei x

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

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

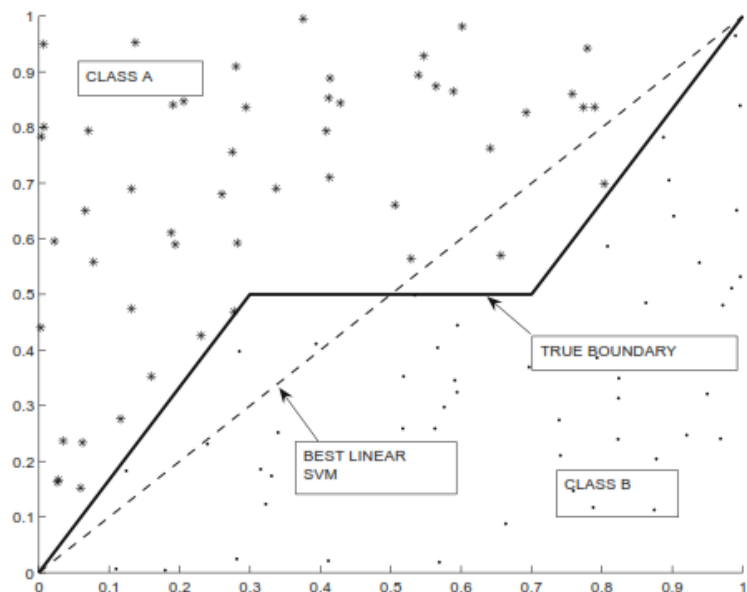
MSE = eroare medie pătratică

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

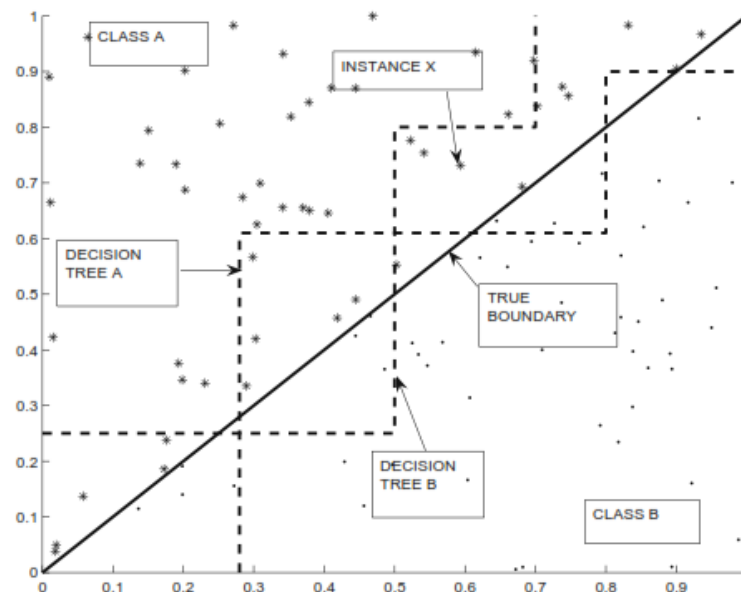
Motivație

Componente ale erorii:

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



(a) bias



(b) variance

Motivație

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

(influențată de model)



Varianța răspunsurilor produse de clasificatori
Antrenati pe diferite seturi de date
(influențată de date)

Motivație

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

Obs:

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

Cum se poate reduce eroarea?

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

Motivație

Bias vs variance

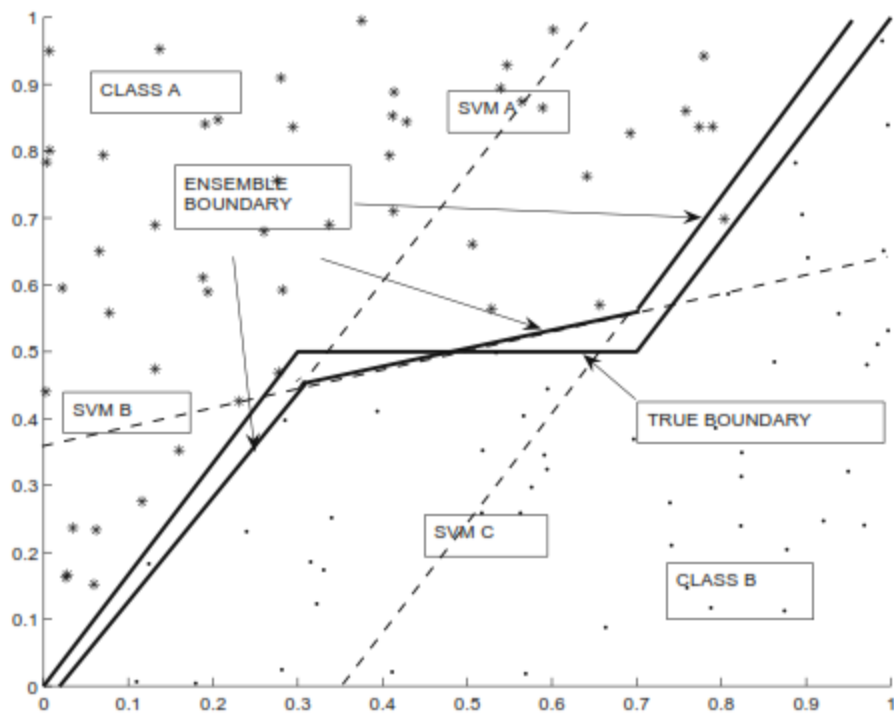
- Modele simple (e.g. Modele liniare, modele bazate pe reguli simple, arbori de decizie simpli, naïve Bayes)
 - Deplasare mare (datorită faptului că suprafața de decizie este prea simplă)
 - Varianță mică (modelele simple sunt robuste în raport cu schimbările din seturile de date; modelele simple suferă rar de supra-antrenare)
- Model complexe (e.g. Rețele neuronale/ arbori de decizie cu multe nivele)
 - Deplasare mică (întrucât modelează bine suprafețele de decizie)
 - Varianță 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 și varianță

Prin combinarea mai multor modele pot fi reduse simultan atât varianța cât și deplasarea

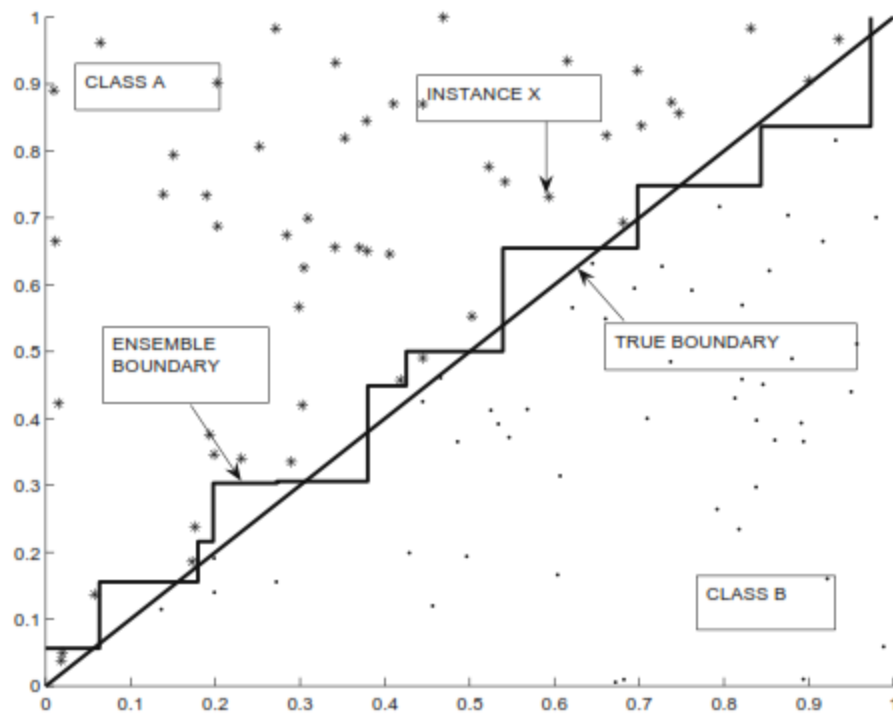
Motivație

Combinarea mai multor modele → model de tip ansamblu



(a) bias

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



(b) variance

Combinând arbori de decizie antrenati pe seturi diferite se poate reduce varianța

Este util să se combine modelele?

O analiza probabilistă simplă

- Considerăm 25 de clasificatori
 - Fiecare clasificator are o anumită rată de eroare, $\varepsilon = 0.35$ (probabilitatea să producă un răspuns eronat)
 - Presupunem că cei 25 de clasificatori sunt **independenți**
 - Probabilitatea ca un ansamblu constituit din cei 25 de clasificatori (în cazul unei reguli de agregare bazată pe regula majorității)=:

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

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 același set de date - corespunde **ansamblelor centrate pe modele**
- Prin antrenarea aceluiași 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 obține 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 $\{A_1, A_2, \dots, A_r\}$
- Output: un model de tip ansamblu constând din K modele individuale $\{M_1, M_2, \dots, M_K\}$

REPEAT

- $k=1$
- select an algorithm A from the set $\{A_1, A_2, \dots, A_r\}$
- 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 $\{M_1, M_2, \dots, M_k\}$ (for each model the data not included in the corresponding training set are used)

UNTIL desired performance

Modele de tip ansamblu

Algoritm generic pt un model de tip ansamblu

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

Cazuri particulare:

- Algoritmi diferiți, un set de antrenare (e.g. Colecție de modele - bucket of models)
- Același algoritm, diferite seturi de antrenare
 - Bagging
 - Random forests
 - Boosting

Colecții de modele – bucket of models

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

Varianta 1:

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

Varianta 2:

- Setul de date D se divide în două subseturi A și 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 părți ale setului de date ar putea fi diferite modele mai adecvate

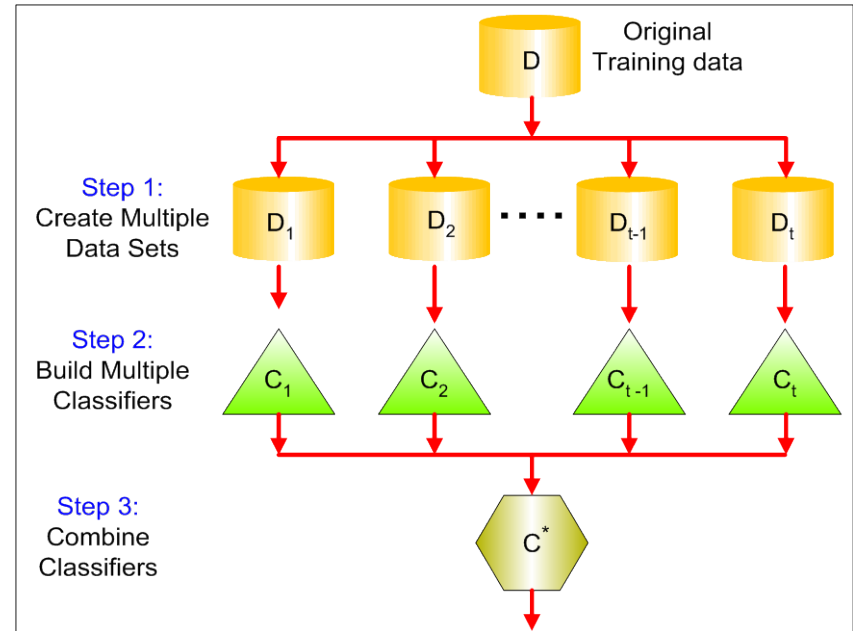
Bagging

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

Seturi de antrenare:

- Obținute prin selecție 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 varianța
- Nu reduce deplasarea (întrucât același model este folosit pt toate seturile de antrenare nu sunt eliminate limitările modelului)

Obs:

- Reducerea varianței este asigurată doar dacă modelele din ansamblu sunt independente
- Limitarea corelației dintre modele se poate obține prin introducerea unor elemente aleatoare → [random forests](#)

Random forests

Random forest = colecție de **arbori aleatori construiți** folosind tehnica de la bagging (seturile de antrenare sunt construite prin selecție aleatoare cu revenire)

Construire random forest:

- Se construiește 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 = arboire de decizie construit 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 un $m \ll M$ iar la ramificarea fiecărui nod se selectează aleator m variabile și 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 folosește **pruning**.

[Leo Breiman

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

Random forests

Obs: eroarea arborilor aleatori depinde de 2 elemente:

- **Corelația** dintre oricare 2 arbori – cu cât corelația este mai mare cu atât eroarea este mai mare
- **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

[Leo Breiman

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

Random forests

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

- Reducerea lui m reduce atât corelația cât și puterea.
- Creșterea lui m conduce la creșterea corelației și a puterii
- E necesar un compromis – se alege valoarea lui m care conduce la eroare mică

Obs: estimarea calității se face folosind datele care nu au fost selectate la construirea arborelui (nu e necesară validare încrucișată) – tehnica “out-of-bag”

[Leo Breiman

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

Boosting

Ideea de bază:

- Fiecare instanță din setul de antrenare are o pondere care poate fi utilizată
 - Direct în cadrul modelului (dacă acesta permite utilizare ponderilor)
 - sau în definirea unor probabilități de selecție
- Ponderile pot fi adaptive (instanțele clasificate incorect au asociate valori mai mari ale ponderilor)

Abordare:

- La început toate instanțele au asociate aceleași
- Pe parcursul procesului de antrenare:
 - Pt Instanțele clasificate incorect se mărește valoarea ponderii
 - Pt Instanțele clasificate corect se micșorează valoarea ponderii

Obs: Se presupune că principala componentă a erorii este deplasarea și se încearcă reducerea acesteia prin acordarea unei importanțe 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, \dots, M_T)
 - Set de ponderi corespunzătoare modelelor
- **Ideea de bază**
 - La fiecare pas t al algoritmului, se obține 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)

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 ($\epsilon(t)$)

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

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** $w(t+1,i)=w(t+1,i)/\text{sum}(w(t+1,j),j=1..L)$

UNTIL ($t \geq T$) or ($\epsilon(t)=0$) or ($\epsilon(t) \geq 0.5$)

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 $1/L$ și se repetă procedura

AdaBoost

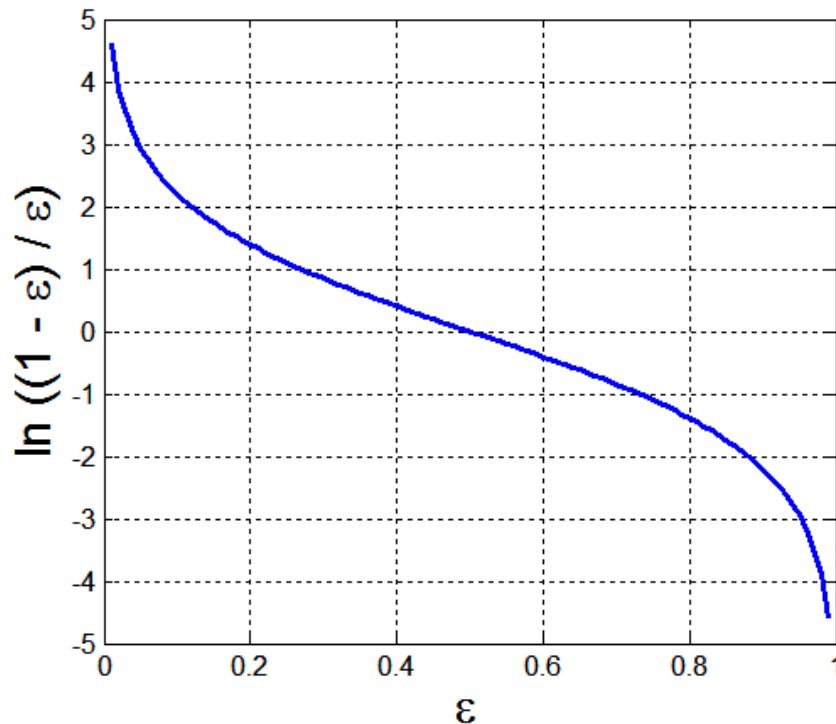
Câteva detalii:

- Rată ponderată de eroare:

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

- Importanța (pondere) unui model/ clasificator:

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



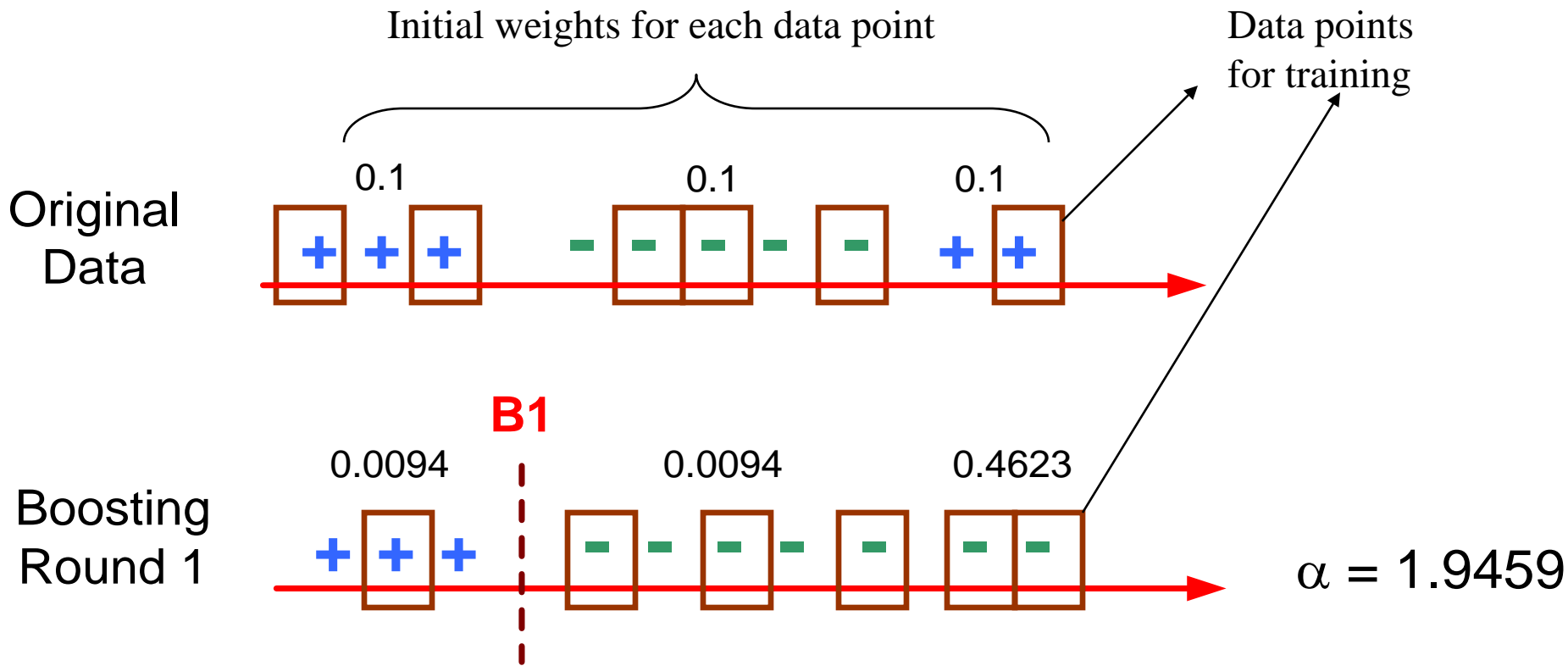
AdaBoost

Etapa de clasificare

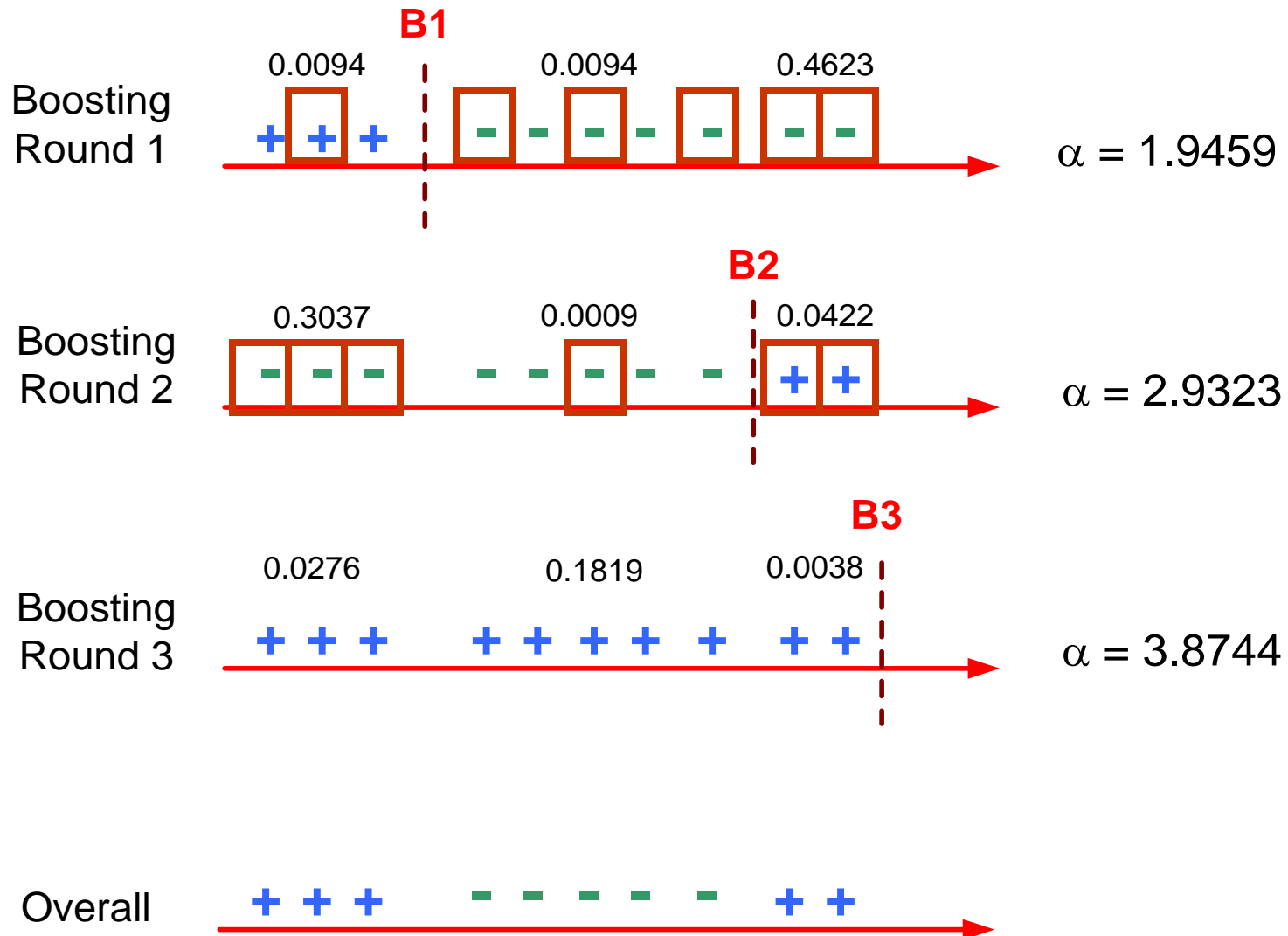
(în cazul clasificatorilor binari ce produc valori în $\{-1,1\}$)

- Se aplică fiecare componentele ansamblului (M_1, M_2, \dots, M_T) și se colectează rezultatele (r_1, r_2, \dots, r_T) (în $\{-1,1\}$)
- Se agreghează rezultatele:
 - calcul $r = \alpha_1 r_1 + \alpha_2 r_2 + \dots + \alpha_T r_T$
 - IF $r < 0$ THEN return -1 ELSE return +1

AdaBoost



AdaBoost



Stacking

Idee de bază: două nivele de clasificare

Etape principale:

- Se divide setul de date D in două subseturi A și B
- **Primul nivel:** se antrenează un ansamblu de k clasificatori bazați pe A (poate fi o colecție de clasificatori, se pot baza pe bagging sau pe k runde de boosting)
- **Second level:**
 - Se determină k ieșiri (etichete ale claselor) ale clasificatorilor antrenați la primul nivel pentru fiecare dintre instanțele subsetului B
 - Se construiește un set de date având ca atribute de intrare aceste k ieșiri și ca atribut de clasă eticheta corectă a instanței 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 și un clasificator combinat
- Pentru o instanță de test, primul nivel de clasificatori este utilizat o nouă instanță k -dimensională pe când al doilea clasificator furnizează rezultatul corespunzător instanței 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 probabilități și nu etichete de clase
- Tehnica stacking permite reducerea ambelor componente ale erorii (întrucât al doilea nivel învață din erorile diferitelor componente ale ansamblului)

Sumar

Impactul metodelor de tip ansamblu asupra componentelor erorii

- Bagging și random forests au fost proiectate să reducă varianța
- Boosting și stacking au fost proiectate să reducă ambele componente

Extindere ideii de ansamblu în contextul tehnicilor de clustering

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