Curs 13:

Metode de tip ansamblu
(meta-modele)

Structura

- Motivaţie
- Ideea modelelor de tip ansamblu
- Colecţii de modele (bucket of models)
- Strategii de agregare a modelelor
  - Bagging – Random forests
  - Boosting – AdaBoost, Gradient Boosting
  - Stacking

Motivaţie

O analiză a rezultatelor competițiilor de la Kaggle (https://www.kaggle.com) cele mai bune performanțe în rezolvarea unor probleme de predicție (clasificare, regresie) sunt obținute de către

- NN = Neural Networks (diferite arhitecturi, în particular deep neural networks)
- LightGBM = light Gradient Boosting Models
- XGB = XGboost = Extreme Gradient Boosting Models
- RF = Random Forest Models
- ...

... ultimele trei metode se bazează pe ideea de a combina mai multe modele pornind de la intuiția că este util să fie consultați mai mulți experți/ colectate mai multe opinii înainte de a se lua o decizie

... cum s-ar putea transpune această idee în practica construirii modelelor de clasificare/regresie?

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

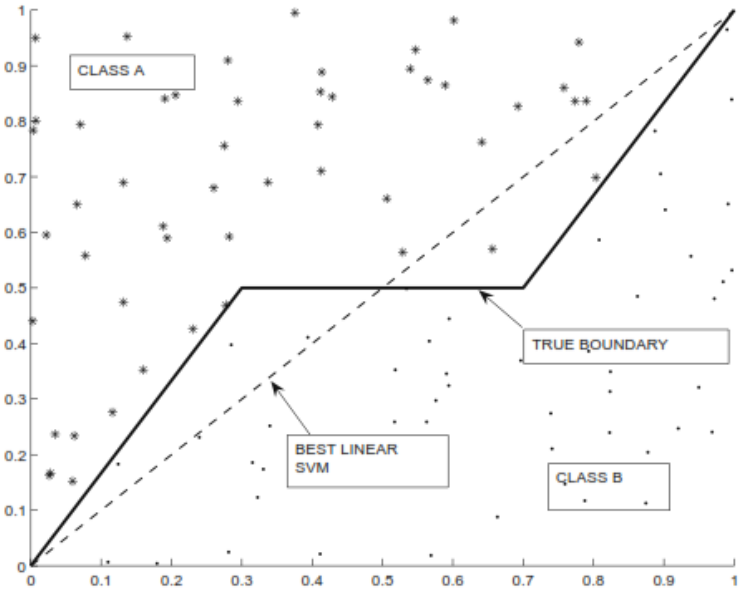
MSE = eroare medie pătratică

Extragerea modelului din date: estimarea parametrilor modelului astfel ca MSE (sau altă funcție de eroare) să fie minimizată

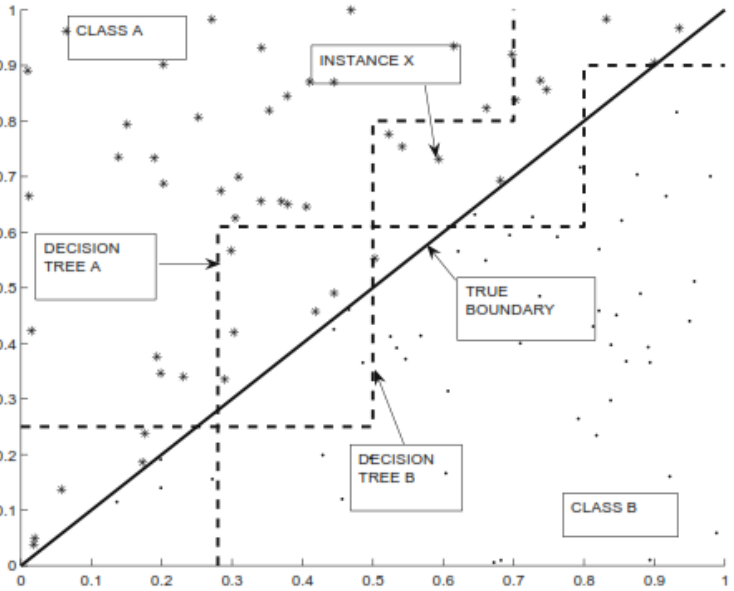
Motivaţie

Cauze ale erorilor de clasificare

- Fiecare model are o capacitate specifică de reprezentare a dependenței dintre variabila țintă și cele predictor
- Modelele sunt construite folosind un set finit de antrenare (un eșantion extras din mulțimea tuturor datelor)



(a) bias



(b) variance

Motivaţie

Exemplu: analiza proceselor de vânzare în context Bussiness-to-bussiness (produsele care trebuie vândute sunt complexe, iar procesul de negociere poate fi îndelungat)

- scop: estimarea sansei de succes a unei vânzări
- sistemele de tip CRM (Customer Relationship Management) conțin informații privind vânzări anterioare

Dataset
(date anonimizate
corespunzătoare unei companii
reale):
https://www.salvirt.com/b2bdataset/

Feature	Description	Values
Product	Offered product.	e.g. ERP, A, B, etc.
Seller	Seller's name.	Seller's name
Authority	Authority level at a client side.	Low, Mid, High
Company size	Size of a company.	Big, Mid, Small
Competitors	Do we have competitors?	No, Yes, Unknown
Purchasing depart	Is the purchasing department involved?	No, Yes, Unknown
Partnership	Selling in partnership?	No, Yes
Budget allocated	Did the client reserve the budget?	No, Yes, Unknown
Formal tender	Is a tendering procedure required?	No, Yes
...	...	...
Crosssale	Do we sell a different product to existing client?	No, Yes
Upsale	Increasing existing products?	No, Yes
Deal type	Type of a sale.	Consulting, Project, etc.
...	...	...
Status	An outcome of sales opportunity.	Lost, Won

- 448 instanțe
- 22 atribute nominale
- clasificare binară

(negociere finalizată cu succes/ insucces)

Motivaţie

Exemplu: analiza proceselor de vânzare în context Bussiness-to-bussiness (https://www.salvirt.com/b2bdataset/)

Instabilitatea arborilor de clasificare:

- Clasificatorul este sensibil la setul de date utilizat pentru antrenare
- Variabilitate mare între clasificatorii construiți folosind diferite seturi de date de antrenare

Resampling the training data set (80% of the full dataset)

Variant 1

```
Up_sale = No
|   Partnership = No: Lost
|   Partnership = Yes
|   |   Forml_tend = No: Won
|   |   Forml_tend = Yes: Lost
Up_sale = Yes: Won
Number of Leaves :     4
Size of the tree :     7
------------------------------
Accuracy : 0.7416
Kappa : 0.4822
Sensitivity : 0.8222
Specificity : 0.6591
```

Variant 2

```
Client = Current
..............
|   Scope = Low
|   |   Cross_sale = No: Won
|   |   Cross_sale = Yes: Lost
Client = New
|   Partnership = No: Lost
|   Partnership = Yes
..............
Client = Past: Lost
Number of Leaves :     42
Size of the tree :     58
------------------------------
Accuracy : 0.7528
Kappa : 0.5041
Sensitivity : 0.8889
Specificity : 0.6136
```

Motivaţie

- Structura erorii: MSE = Bias + Variance

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

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

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

⇩ deplasare

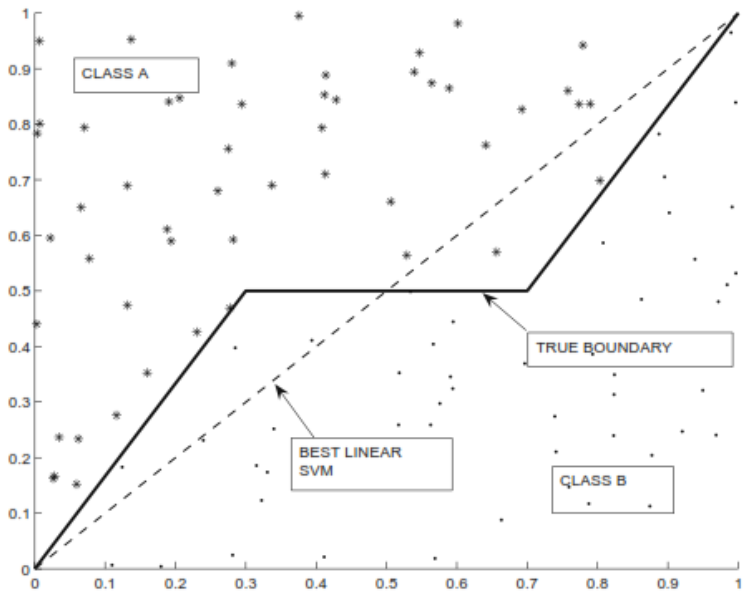
(influenţată de model)

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

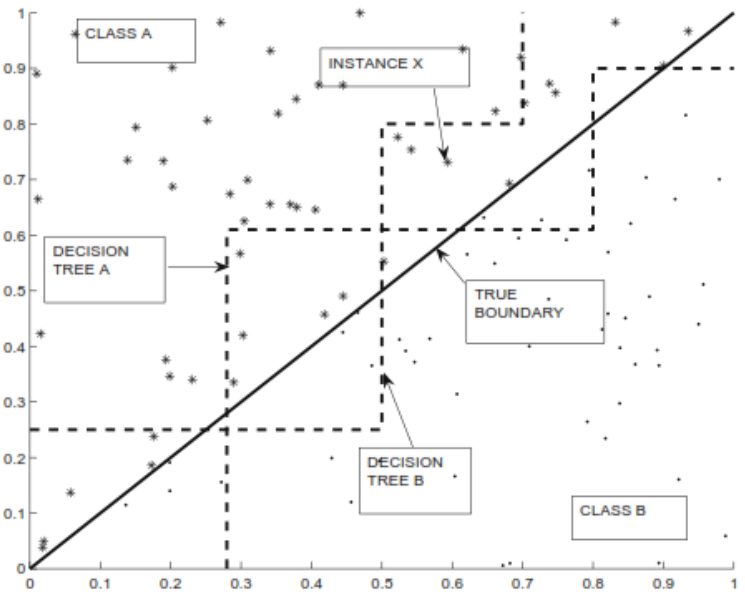
Motivaţie

Componente ale erorii:

- Deplasare (Bias) = inabilitatea clasificatorului de a clasifica corect cauzată de limitările modelului (e.g. modelul e caracterizat prin suprafețe 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 antrenați pe seturi diferite au performanţe diferite)



(a) bias



(b) variance

C. Aggarwal, Data Mining. The textbook, 2015

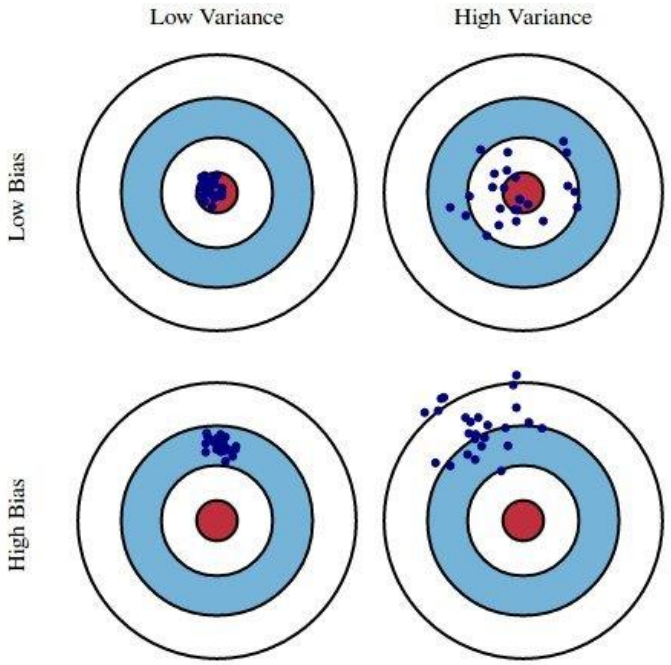
Motivaţie

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?



[Source: https://www.kdnuggets.com/2016/08/bias-variance-tradeoff-overview.html]

Motivaţie

Bias vs variance

	low variance	high variance
low bias		too complex
high bias	too simple	

- 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)

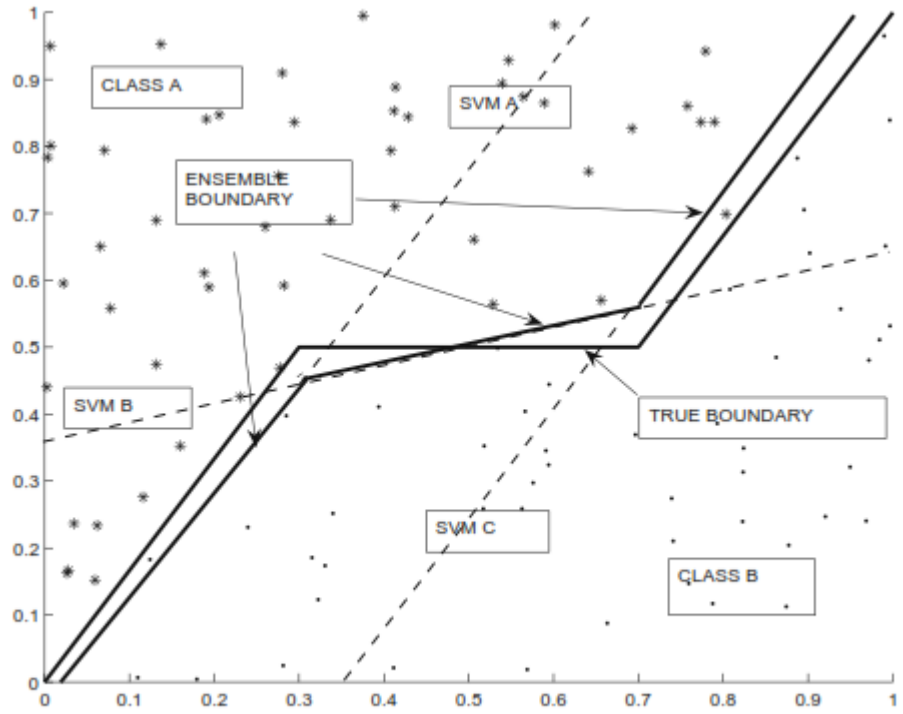
- Modele complexe (e.g. Reţele neuronale/ arbori de decizie cu multe nivele)
  - Deplasare mică (întrucât modelează bine suprafeţele de decizie)
  - Varianţă mare (senzitive la modificările în setul de date; pot fi afectate de supra-antrenare)

Obs:

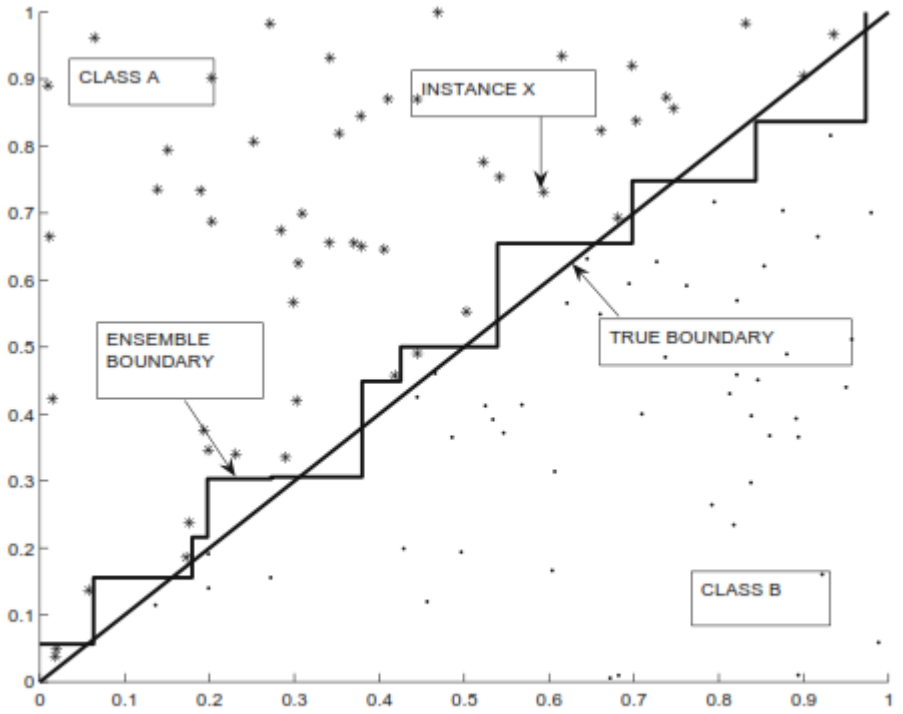
- Dacă se utilizează un singur model este necesară găsirea unui compromis între deplasare şi varianţă
- Prin combinarea mai multor modele pot fi reduse simultan atât varianţa cât şi deplasarea

Motivaţie

Combinarea mai multor modele → model de tip ansamblu



(a) bias



(b) variance

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

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
  - 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 de eroare in cazul un ansamblu constituit din cei 25 de clasificatori (presupunând că ansamblul folosește regula majorității pentru a decide răspunsul):

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

Este util să se combine modelele?

O analiza probabilistă simplă

- Caz general: ansamblu de L modele independente
- Perr(M) = rata de eroare a fiecărui model
- Pcor(M) = rata de răspunsuri corecte a fiecărui model
- Perr(E) și Pcor(E) – rate de eroare/răspunsuri corecte ale ansamblului de modele (bazat pe agregarea răspunsurilor prin votare)

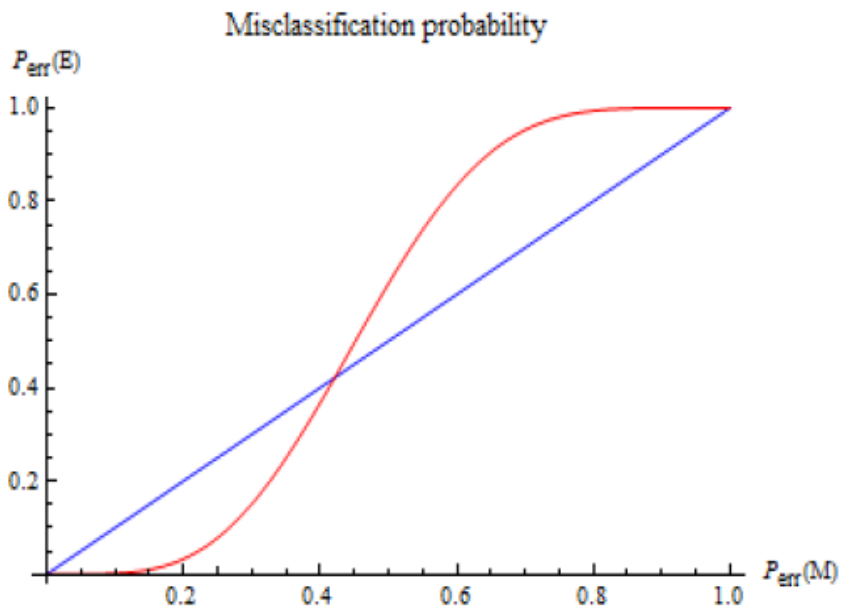
$$Perr(E) = 1 - \sum_{i=\frac{L}{2}+1}^{L} C_L^i (1 - Perr(M))^i (Perr(M))^{L-i}$$

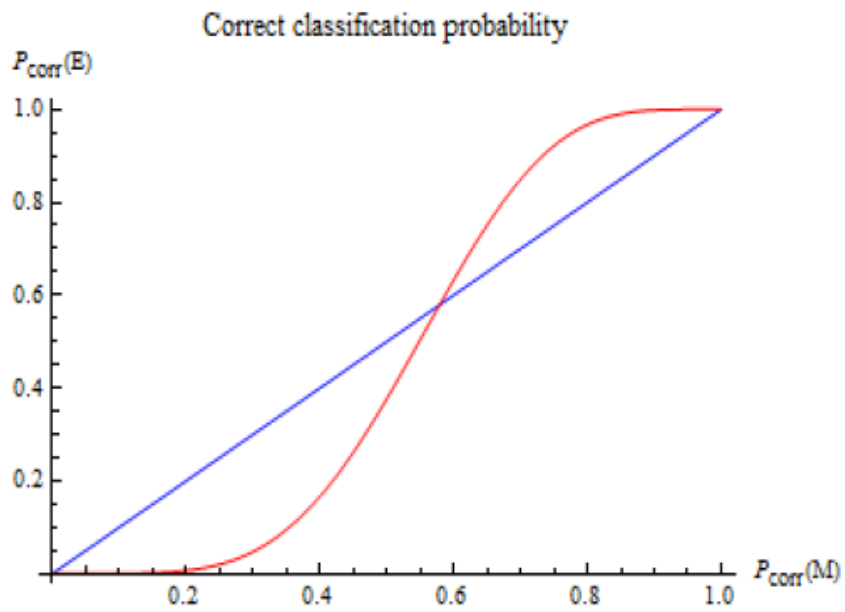
$$Pcor(E) = \sum_{i=\frac{L}{2}+1}^{L} C_L^i Pcor(M)^i (1 - Pcor(M))^{L-i}$$

Este util să se combine modelele?

O analiza probabilistă simplă – evolutia ratei de eroare/răspunsuri corecte ale unui ansamblu cu L=10 componente in functie de rata de eroare/răspunsuri corecte ale modelelor componente

- Ansamblul potențează calitatea modelelor componente bune și accentuează defectele modelelor componente slabe

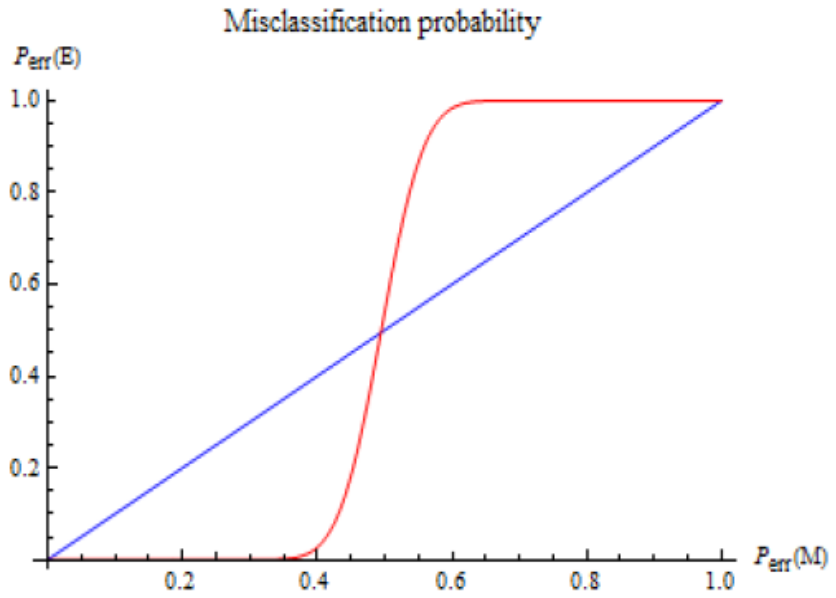


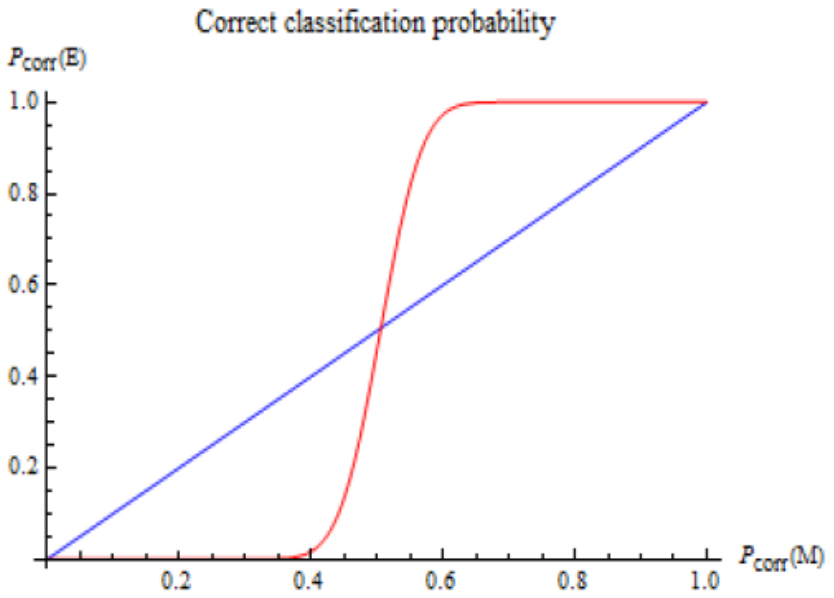


Este util să se combine modelele?

O analiza probabilistă simplă (L=100)

- Dacă numărul de modelel componente crește pragul între comportament corect/incorect se apropie de 0.5 ⇒ se pot utiliza modele componente cu o rată de succes ușor mai mare decât 0.5 (modele “slabe” – doar ușor mai bune decât un model aleator simplu bazat pe aruncarea unei monede)





Modele de tip ansamblu

Cum se construiesc modelele de tip ansamblu?

- Prin construirea mai multor modele (pe baza unor ipoteze structurale diferite) 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
- selectează un algoritm A din setul de algoritmi $\{A_1, A_2, \ldots A_r\}$
- Construiește un set de antrenare D_k (prin selecție din D)
- Construiește modelul M_k prin aplicarea algoritmului A asupra setului de date D_k
- Evaluează performanța ansamblului curent $\{M_1, M_2, \ldots, M_k\}$ (pt fiecare model se utilizează pt evaluare date care nu au fost folosite la antrenare)
- k=k+1

UNTIL se obține performanța dorită

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 (exemplu: Random forests)
  - Boosting (exemplu: adaptive boosting – AdaBoost)
- Algoritmi diferiți, seturi de date diferite
  - Stacking

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 combină prin:
  - Regula majorităţii (clasificare)
  - Medierea rezultatelor de la modelele componente (regresie)

Varianta 2:

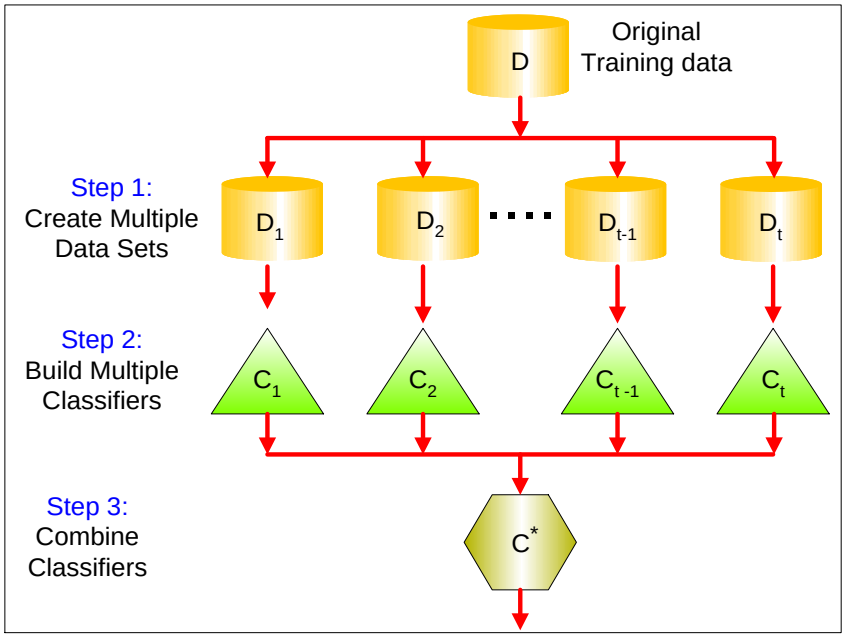
- Setul de date D se divide în două subseturi A şi B
- Se antrenează toate modelele folosind setul A
- Se selectează modelul cu cel mai bun comportament pentru subsetul B
- Se reantrenează modelul selectat pentru întregul set de date D

Obs:

- Poate conduce la reducerea deplasării dacă pt diferite părţi spațiului datelor de intrare sunt adecvate diferite modele

Bagging

- Idee de bază: un algoritm, mai multe seturi de date → mai multe modele de predicție



Seturi de antrenare:

- Obţinute prin selecţie cu revenire din setul complet de date D (nr de elemente din subset = nr elemente din 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, astfel că nu sunt eliminate limitările modelului)

Obs:

- Reducerea varianţei este asigurată doar dacă modelele din ansamblu sunt independente (seturile de antrenare sunt construite independent unele de altele)
- Limitarea corelaţiei dintre modele se poate obţine prin introducerea unor elemente aleatoare în construirea modelelor componente → random forests

Random forests

Random forest = colecţie de arbori de decizie construiţi folosind strategii de selecție aleatoare - tehnica de tip 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 (se selectează clasa cea mai frecventă)
    - medierea distribuțiilor de probabilitate (clasa se selectează pe baza distribuției mediate) – e varianta utilizată în ScikitLearn

Random forests

Random tree = arbore de decizie construit folosind random-split

Etape:

- Dacă numărul de exemple din setul de antrenare este L, atunci se selectează L cazuri aleator (prin selecție cu revenire). Acest set este folosit pt construirea arborelui

- In cazul a M variabile de intrare se fixează un m<M (dacă M este mare, m trebuie să fie semnificativ mai mic) iar la ramificarea fiecărui nod se selectează aleator m variabile 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.

[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

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

- Prin descreșterea lui m se 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"

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

Random forests

Caracteristici:

- Datele nu necesită pre-procesare
  - Valorile absente nu trebuie neapărat imputate
  - Nu e necesară selecție explicită a atributelor (permite prelucrarea seturilor de date cu număr mare de atribute
- Se bazează pe două nivele de aleatoritate:
  - La selecția setului de date
  - La selecția setului de atribute pentru a se decide condiția de partiționare
- Sunt eficiente computațional:
  - Atributul de partiționare se selectează dintr-un subset de atribute
  - Arborii componenți pot fi construiți în paralel
- Analiza performanței se bazează pe metoda Out-Of-Bag (OOB) – se utilizează datele care nu au fost folosite la construirea arborelui

Boosting

Ideea de bază:

- Fiecare instanţă din setul de antrenare are o pondere care poate fi utilizată
  - direct în cadrul modelului (dacă acesta permite utilizarea ponderilor)
  - sau în definirea unor probabilităţi de selecţie
- Fiecare model poate avea o pondere în construirea răspunsului final, în funcție de rata de succes sau calitatea modelului
- Ponderile pot fi adaptive (instanţele clasificate incorect au asociate valori mai mari ale ponderilor)

Abordare:

- La început toate instanţele au asociate aceleaşi valori ale ponderilor
- 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) folosind o distribuție de probabilitate a exemplelor care favorizează exemplele pentru care eroarea de clasificare e mare – vezi slide următor

- Clasificatorul de bază e de regulă un clasificator simplu fără putere prea mare de discriminare (weak classifier) - e suficient ca acuratețea să fie mai mare decât 50% (în cazul arborilor de decizie se folosește un singur nod de decizie – arbore de tip stump)

[Y. Freund, and R. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”, 1995]

AdaBoost

AdaBoost (A,D)

t=1;
// initializare ponderi pentru exemplele din setul de antrenare (probabilități de selecție):
w(t,i)=1/L for all i=1..L
REPEAT
  t=t+1; // construire M_t utilizând valorile curente ale ponderilor
  se calculează rata ponderată de eroare pt modelul M_t pe setul de date D ($\mathcal{E}(t)$)
  calculează ponderea modelului $\alpha(t)=\ln((1-\mathcal{E}(t))/\mathcal{E}(t))/2$
  FOR i=1,L **DO**
    IF x_i este clasificat eronat **THEN** $w(t+1,i)=w(t,i)*\exp(\alpha(t))$
                                ELSE $w(t+1,i)=w(t,i)*\exp(-\alpha(t))$
 // calcul noua distributie de probabilitate a datelor
  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 ($\mathcal{E}(t)=0$) or ($\mathcal{E}(t)>=0.5$)

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

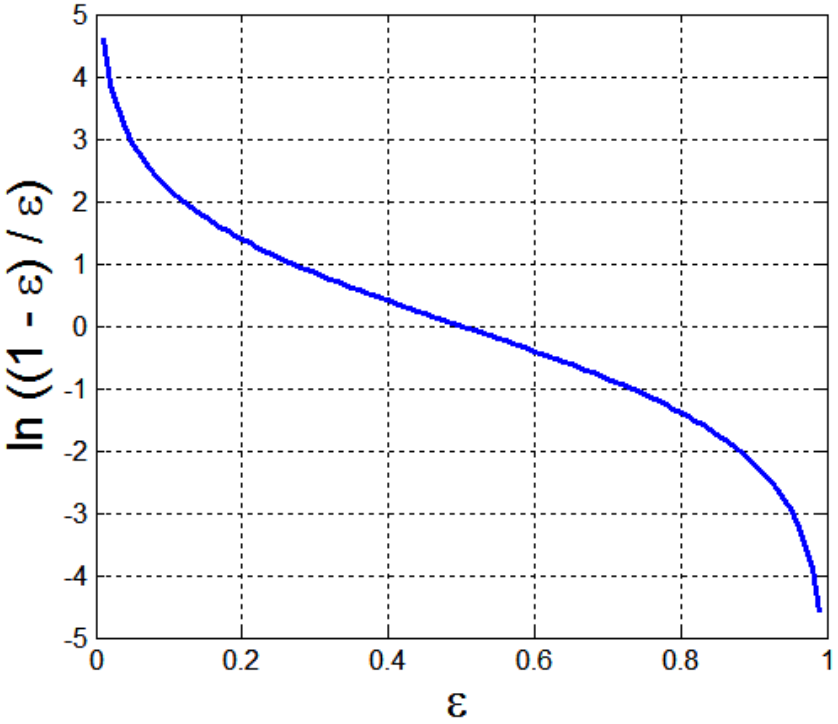
Câteva detalii:

- Rată ponderată de eroare:

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

- Importanţa (ponderea) unui model/ clasificator:

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



AdaBoost

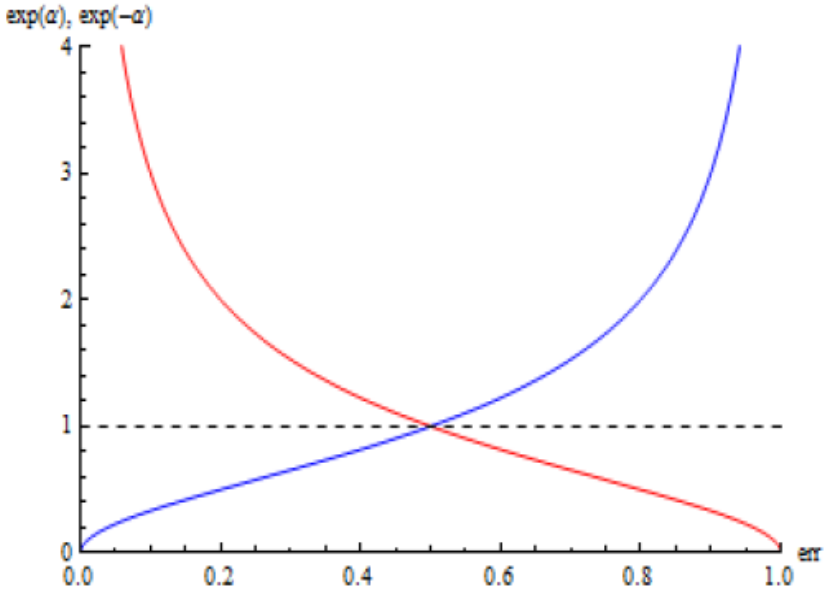
Câteva detalii:

- Rată ponderată de eroare:

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

- Importanţa (ponderea) unui model/ clasificator:

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



Roşu: exp(α(t)) – factor utilizat în ajustarea ponderii exemplelor clasificate incorect

Albastru: exp(-α(t)) – factor utilizat în ajustarea ponderii exemplelor clasificate corect

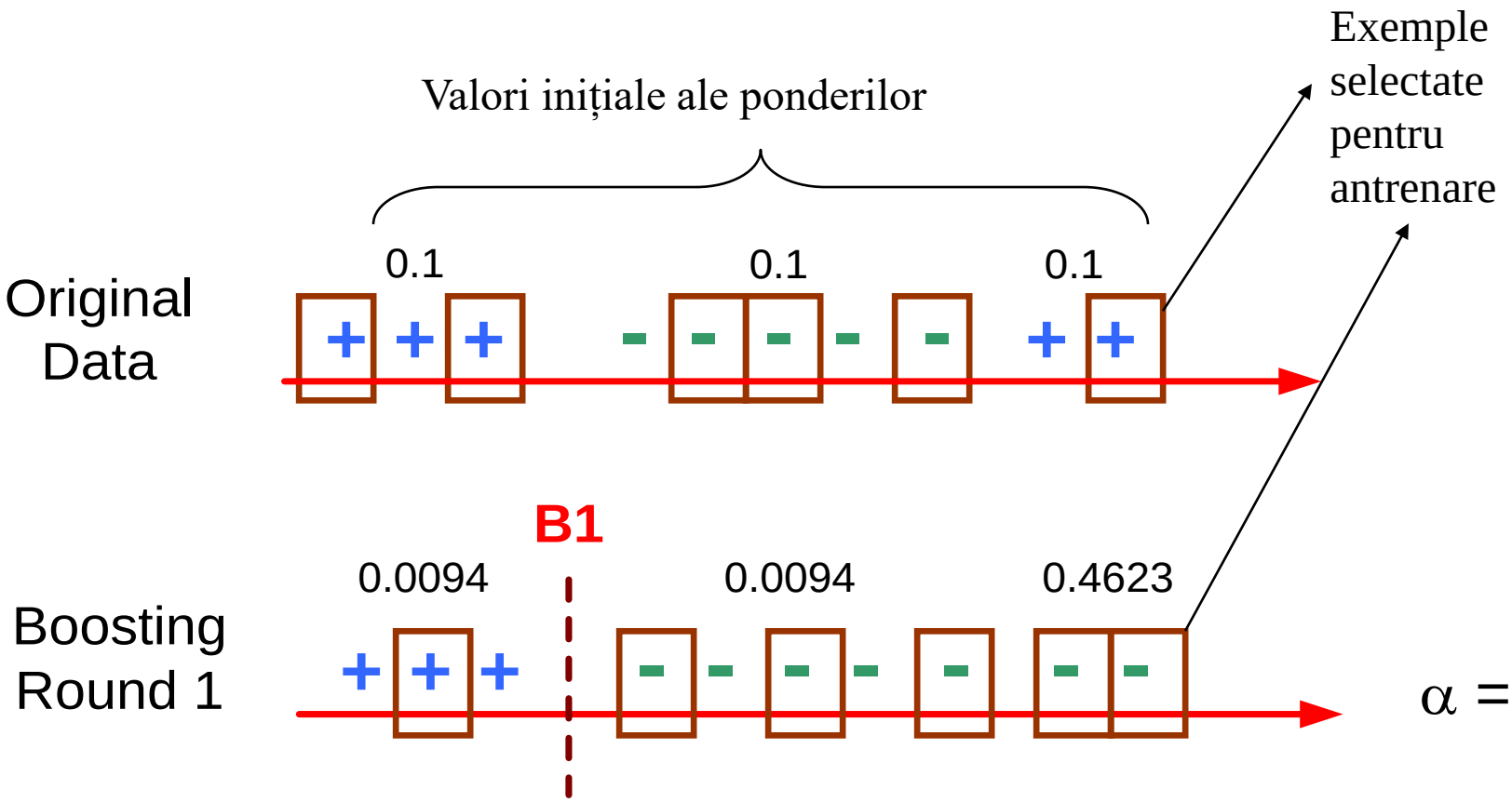
AdaBoost

Etapa de clasificare

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

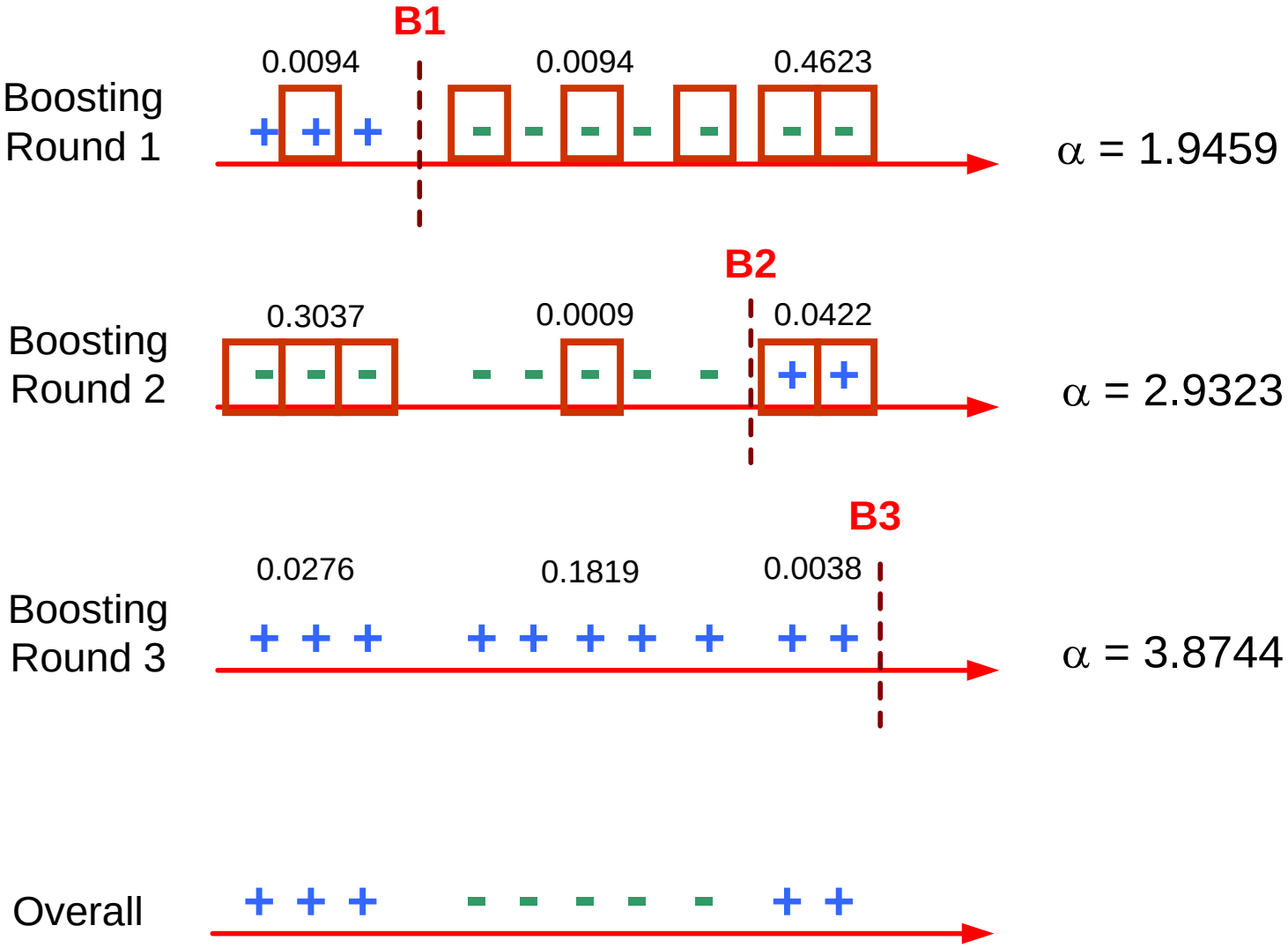
- Se aplică fiecare dintre componentele ansamblului (M_1, M_2,…, M_T) şi se colectează rezultatele (r_1,r_2,…r_T) (valori din in {-1,1})
- Se combină rezultatele ținând cont de ponderea fiecărui model:
  - calcul $r=\alpha_1 r_1 + \alpha_2 r_2 +\ldots+ \alpha_T r_T$
  - IF r<0 THEN return -1 ELSE return +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

Gradient Boosting

Idee de bază:

- generalizare AdaBoost pt regresie
- se construiește un model aditiv (constituit din modele slabe) – construirea este incrementală folosind ideea de la metoda gradientului de optimizare a funcției de eroare în raport cu modelele componente (optimizare în spațiul funcțiilor care joacă rol de modele slabe)
- Poate fi aplicat pentru diferite funcții de eroare (loss) pentru care gradientul poate fi calculat analitic

Notații:

- Model aditiv: $F_T(x)=h_1(x)+h_2(x)+\ldots+h_T(x)$
- fiecare $h_t(x)$ este un model slab (e.g. arbore cu un singur nod de decizie) selectat astfel încât să asigure descreșterea funcției de eroare
- Funcție de eroare (loss) calculată pentru exemplul (x_i,y_i): $L(y_i, F(x_i))$
- Gradientul funcției de eroare calculat în (x_i,y_i): $g_i= \frac{\partial L(y_i,F(x_i))}{\partial F(x_i)}$
- Obs: în cazul modelelor de regresie cel mai frecvent se folosește MSE iar $g_i = y_i - F(x_i)$

Gradient Boosting

Structura generală:

- Inițializare: $F_0(x)$=constant (in cazul regresiei poate fi media valorilor corespunzătoare din setul de date)

- FOR t=1,T
  - Pentru fiecare exemplu (x_i,y_i) se calculează gradientul funcției de eroare:

$$r_{it} = -\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)} = -(y_i - F_{t-1}(x_i))$$

Obs: In cazul MSE, gradientul corespunde reziduurilor

  - Se construiește un model local $h_t(x)$ pt setul de date corespunzător reziduurilor (x_i,r_{it})

  - Se adaugă modelul local la cel general: $F_t(x)= F_{t-1}(x)+ h_t(x)$

Obs: performanța poate fi îmbunătățită prin shrinkage (acționează ca o tehnică de regularizare): $F_t(x)= F_{t-1}(x) + alpha * h_t(x)$ (alpha in (0,1))

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)

- Al doilea nivel:
  - 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 pentru a construi 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

- Bagging şi random forests au fost proiectate să reducă varianţa
  - Modelele componente sunt independente și pot fi construite în paralel
  - Modelele componente ar trebui să aibă deplasare cât mai mica ➔ nu se aplică simplificare (pruning) asupra lor

- Boosting permite reducerea ambelor componente (varianța și deplasarea)
  - Modelele componente sunt construite secvențial (nu sunt independente)
  - Modelele componente sunt simple

- Stacking – cea mai flexibilă variantă (dar și mai costisitoare dpdv computațional)
  - Modelele componente pot fi construite independent
  - Agregarea rezultatelor se bazează pe un proces de învățare

Sumar

Extindere 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 – muchie care conectează mai multe noduri)

Ansamblu implicit:

- Tehnica dropout utilizată în contextul rețelelor neuronale poate fi interpretată ca o tehnică de construire a unui ansamblu implicit de rețele în care la fiecare epocă de antrenare un alt set de neuroni sunt activi