Modele de tip “Ensemble” partea a II-a
Modelarea de tip boosting

ADRIAN OȚOIU

Modele de tip “Ensemble” partea II

Modelarea de tip boosting

- Modelarea de tip ensemble
- Modele de tip boosting. Elemente de bază
- Istoricul modelelor de tip boosting
- Modelul adaboost M1

Modelarea cu metoda Gradient boosting

- Particularități și forma generală
- Noțiuni elementare. Funcția de eroare, funcția de cost, optimizarea gradient descent
- Formalizarea algoritmului GBM. Elemente de optimizare
- Modele de tip XGBoost
- Optimizarea hiperparametrilor

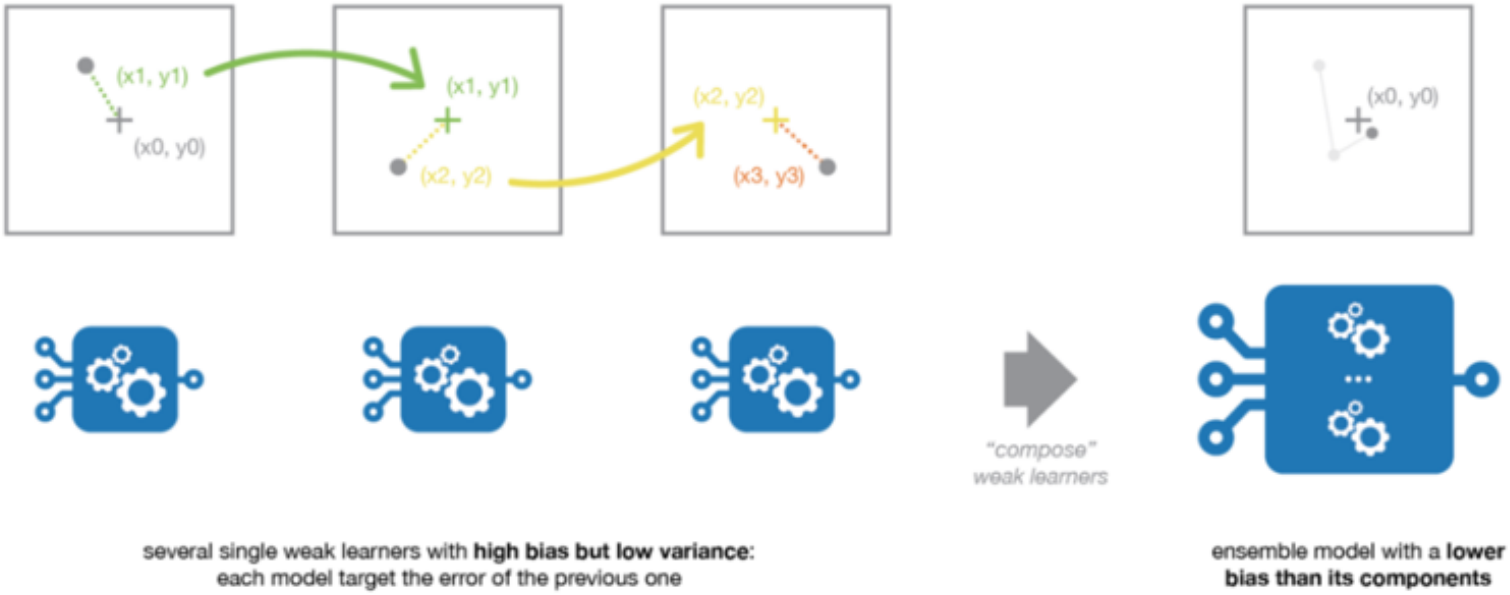
Aprecieri generale despre modelele de tip ensemble

Modelarea de tip ensemble

Obținerea unor modele optime dpdv. al compromisului bias-variance prin modelarea de tip **ensemble**.

Modelarea de tip ensemble (paradigmă de machine learning): tehnica prin care mai multe modele de bază cu performanță scăzută (weak learners), sunt antrenate pentru a rezolva **aceeași problemă**, iar rezultatele acestora sunt combinate pentru a obține modele cu performanțe superioare **(strong learners).**

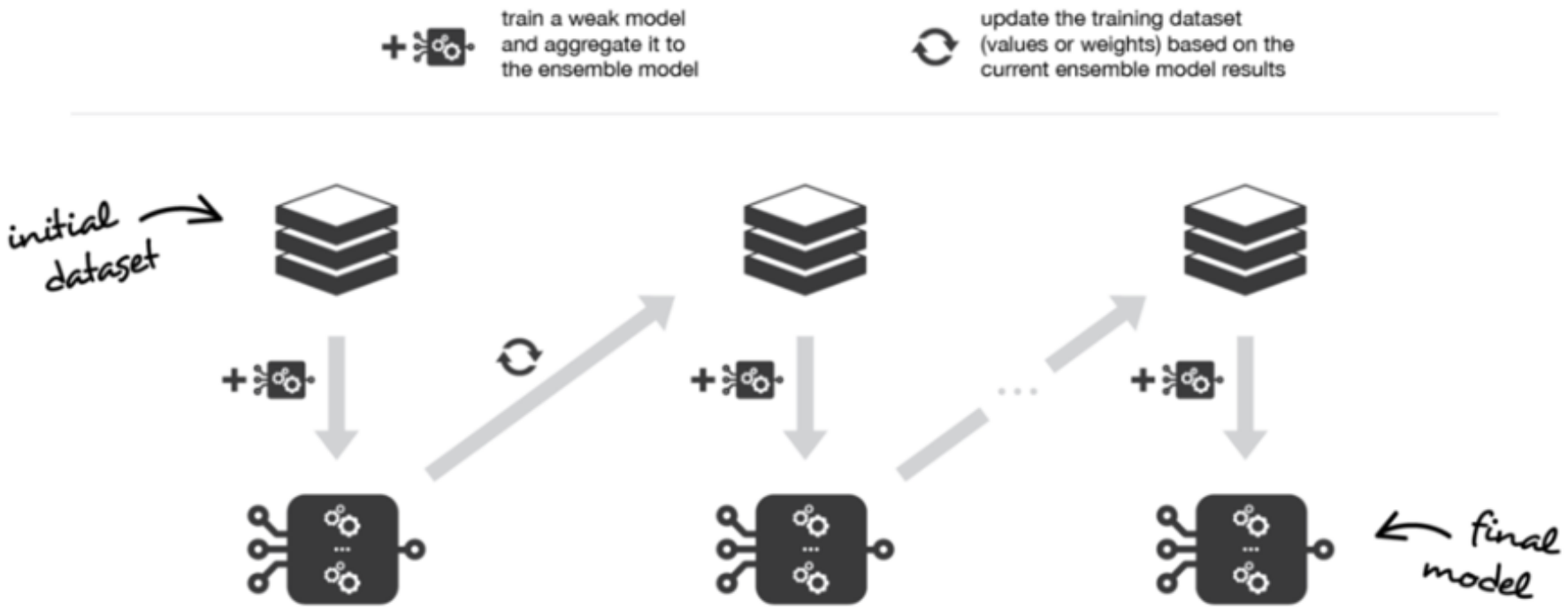
Exemplu: model ensemble de tip boosting



Sursa: Joseph Rocca (2019)

Modele de tip boosting

Modele de tip boosting: folosesc **un singur tip de model de bază.** Algoritmii sunt implementați într-o **manieră secvențială, adaptivă,** astfel încât estimările generate de către un model de bază depind de estimările modelelor de bază anterioare. Rezultatul final al estimării se obține prin combinarea rezultatelor folosind o metodă deterministă.



Forma generală a unui algoritm de tip boosting

Fie un anumit tip de model de bază I, un set de date de antrenare X, k dimensiunea eșantionului pentru primul model de bază estimat, și M_1, M_2 și M_3 rezultatele estimării a trei modele de bază.

P1. Se selectează k < m observații din X și se obține subeșantionul X_1
P2. Se estimează modelul $M_1 = I(X_1)$
P3. Se selectează X_2 prin prelevarea aleatoare din observațiile rămase ($X-X_1$) astfel încât jumătate din observații sunt incorect clasificate de modelul M_1
P4. Se estimează modelul $M_2 = I(X_2)$
P5. Se selectează X_3 din observațiile rămase ($X-X_1-X_2$) a celor pentru care există diferențe între rezultatele obținute cu modelele M_1 și M_2
P6. Se estimează modelul $M_3 = I(X_3)$
P7. Se agregă rezultatele modelelor M (de exemplu prin vot majoritar)

Kuhn și Johnson (2013), Rokach (2019)

Scurt istoric al modelelor de tip boosting

Modelarea de tip boosting a fost inventată de Freund și Schapire (1997) care au dezvoltat primul algoritm de acest tip, denumit AdaBoost (adaptive boosting).

Cele mai cunoscute modele de tip adaboost sunt AdaBoost.M1, AdaBoost.M2, Real AdaBoost (Friedman *și alții*, 2000) și SABoost (Tsao și Chang, 2007).

Friedman (2001) a dezvoltat metoda **gradient boosting (GBM)**. Metoda constă în estimarea secvențială a unor modele de bază cu metoda celor mai mici pătrate, folosind **pseudoreziduurile** obținute în ultimul model de bază ca variabilă dependentă.

Pseudoreziduurile reprezintă **gradientul funcției obiectiv (funcție diferențiabilă)**, minimizat în funcție de rezultatele obținute la nivelul fiecărei observații folosite, prin estimarea modelului pe setul de date de antrenare.

Kuhn și Johnson (2013), Rokach (2019)

Scurt istoric al modelelor de tip boosting

Pentru o estimare mai rapidă, modelele de tip gradient boosting folosesc subeșantioane extrase din setul de date de antrenare pentru estimarea modelelor de bază în fiecare iterație.

În contextul utilizării arborilor de decizie, modelele GBM au fost modificate pentru a optimiza, de o manieră separată, regiuni ale arborilor de decizie estimați ca modele de bază, și nu arborii de decizie luați în ansamblul lor.

Cele mai avansate modele de tip gradient boosting sunt bazate pe algoritmul XGBoost (Chen și Guestrin (2016). Îmbunătățirile aduse (discutate ulterior în curs) au consolidat popularitatea modelelor XGBoost ca fiind unele dintre cele mai performante. Alături de XGBoost, alte variante folosite sunt LightGBM (dezvoltat de Microsoft) și Catboost (dezvoltat de Yandex).

Kuhn și Johnson (2013), Rokach (2019)

Modele adaboost

Primul algoritm AdaBoost (adaptive boosting) dezvoltat de Freund și Schapire (1997) denumit AdaBoost.M1 are următoarea structură:

1) Estimarea unui model de bază, având ponderi egale ale observațiilor
2) Evaluarea performanțelor modelului estimat cu o funcție de eroare* err_m
3) Ponderarea observațiilor (sau a subeșantioanelor, în funcție de implementare) prin aplicarea de ponderi mai mari observațiilor clasificate incorect
4) Estimarea unui nou model de bază pe setul de date ponderat
5) Repetarea pașilor 2-4 de M ori, cu ponderile observațiilor calculate anterior
6) Agregarea rezultatelor după o formulă de forma $$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

unde sign este determinat de valorile variabilei y [-1,1], $G_m(x)$ reprezintă estimările individuale ale modelelor de bază iar α_m reprezintă ponderile corespunzătoare contribuției fiecărui model de bază la performanța de ansamblu a modelului adabost, calculată cu formula $\log((1 - err_m)/err_m)$.

* Calculată ca diferența dintre valorile estimate și cele reale ale variabilei dependente

Modelele de tip gradient boosting

Pe baza cercetărilor și a conceptualizării algoritmului AdaBoost, Friedman (2001) a definit modelul gradient boosting (GBM), pornind de la conceptele de funcție de cost (loss function) și modelare aditivă. Aceasta a permis extinderea modelelor de boosting și la modele de tip regresie.

Principiul modelelor de tip gradient boosting:

- Pe baza une funcții de eroare (de ex. abaterea standard a erorii) și a folosirii unui model de bază (de ex. arbore de regresie), algoritmul caută un model care minimizează funcția de eroare pentru estimările ulterioare.
- Estimarea se face pe subeșantioane, de regulă prelevate fără revenire, de dimensiune < N/2

Implementarea în R este realizată în pachetul GBM.

Kuhn și Johnson (2013), Bishop (2006) , Hastie și alții (2017)

Gradient boosting (I)

Pseudocodul pentru un model gradient boosting cu variabilă dependentă continuă, folosind modele de bază de tipul arborilor de regresie, este prezentat mai jos:

1 Selectarea adâncimii arborelui D* și a numărului de iterații, K

2 Calculul lui $\bar{y}$ folosit ca predicție inițială pentru fiecare subeșantion

3 for k = 1 to K do

  4 Calculul **valorii reziduale (gradient)** pentru fiecare subeșantion

  5 Estimarea unui arbore de regresie de adâncime D, folosind reziduurile ca variabilă dependentă

  6 Efectuarea de predicții pentru fiecare subeșantion folosind modelul estimat în pasul anterior

  7 Actualizarea valorilor prezise pentru fiecare subeșantion prin adunarea acestora (ajustată cu o **rată de învățare** α) la valoarea prezisă în pasul anterior

8 end

- Distanța maximă de la nodul terminal la nodul de origine.

Kuhn și Johnson (2013), Bishop (2006) , Hastie și alții (2017)

Gradient boosting. Noțiuni de bază

Funcția de eroare (error function) $L(y, f(X))$ estimează măsura în care estimările variabilei dependente y pe baza valorilor matricei variabilelor explicative X diferă de valorile observate ale acesteia. Rezultatul acesteia sunt valorile reziduale.

În contextul modelelor de tip gradient boosting această funcție **trebuie să fie diferențiabilă.** Pentru aplicarea metodei gradient descent**, funcția trebuie să fie de asemenea convexă, pentru evitarea identificării celei mai bune soluții ca fiind un optim local!**

Exemple. $L(y, f(X)) = [y - f(X)]^2$ pentru modele de regresie

Pentru un model de clasificare binară (regresia logistică)

$$L(y, f(X)) = \begin{cases} -\log(f(X)) \; dacă \; y = 1 \\ 1 - \log(1 - f(X)) \; dacă \; y = 0 \end{cases}$$

Sau în formă compactă

$$L(y, f(X)) = -y\log(f(X)) - (1-y)\log(1- f(X))$$

Kuhn și Johnson (2013), Bishop (2006) , Hastie și alții (2017), Ng (2011)

Gradient boosting. Noțiuni de bază

Pentru optimizarea modelelor și identificarea celei mai bune soluții, se folosește **funcția de cost (cost function, loss function, objective function)**. Uneori se folosește chiar și denumirea de error function!

Funcția de cost evaluează performanța modelelor ținând cont de rezultatele obținute pe tot setul de date de antrenare.

Exemple

$L(y, f(X)) = [y\text{-}f(X)]^2 \Rightarrow J(y, f(X)) = \frac{1}{2n}\sum_{i=1}^{n}[y_i - f(X_i)]^2$

$L(y, f(X)) = -y\log(f(X)) - (1-y)\log(1- f(X)) \Rightarrow$

$J(y, f(X)) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot \log(f(X_i)) + (1 - y_i) \cdot \log(1 - (f(X_i))]$

în ecuațiile de mai sus n reprezintă numărul de observații

De multe ori funcția obiectiv conține și o componentă de penalizare a complexității modelului (exemplu în funcția de cost folosită în modelul XGBoost).

Kuhn și Johnson (2013), Bishop (2006) , Hastie și alții (2017), Ng (2011)

Gradient boosting. Noțiuni de bază

Optimizarea funcției de cost se realizează adesea cu ajutorul metodei **gradient descent**. Aceasta presupune obținerea derivatelor parțiale în raport cu fiecare variabilă explicativă j.

Exemplu. Pentru j $\in (1, w)$ variabile explicative, cu X=(x_1,$x_{2,\ldots,}$ x_w) și parametrii θ_0 θ_1 … θ_w , notați cu θj , optimizarea se efectuează prin actualizarea simultană a valorii parametrilor astfel

$\theta_j : \theta_j - \propto \cdot \frac{\partial}{\partial \theta_j} J(\theta_0,\ldots \theta_w); \frac{\partial}{\partial \theta_j} J(\theta_0,\ldots \theta_w)$ notat mai simplu cu $\nabla J(\theta)$

$\propto$ reprezintă valoarea hiperparametrului **learning rate**, care asigură găsirea unei soluții de tip optim global, după un număr de iterații astfel încât soluția optimă să fie de tipul

$$\hat{\theta} = \underset{\theta}{argmin}\, J(\theta)$$

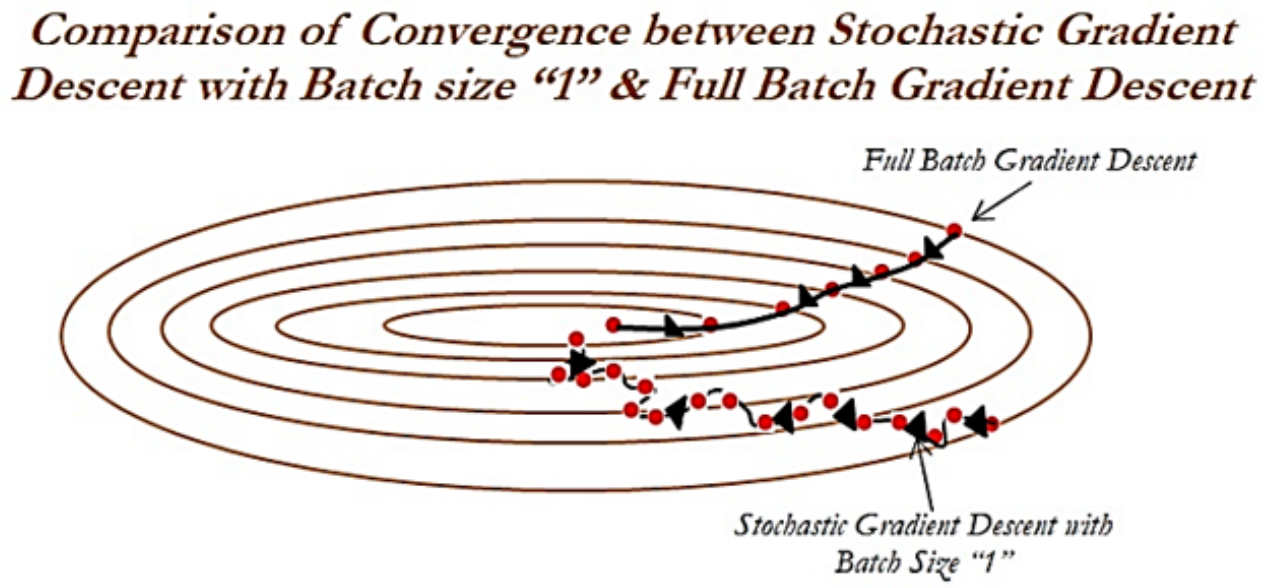
Gradient boosting. Noțiuni de bază

Soluția optimă de tipul $\hat{\theta} = \underset{\theta}{argmin}\, J(\theta)$

Se identifică prin iterații succesive, de tipul

$$\theta^{\tau+1} = \theta^{\tau} - \alpha \cdot \nabla J(\theta^{\tau})$$

Procesul de optimizare se oprește la numărul maxim de iterații T (denumite epochs) sau atunci când funcția cost nu se mai îmbunătățește



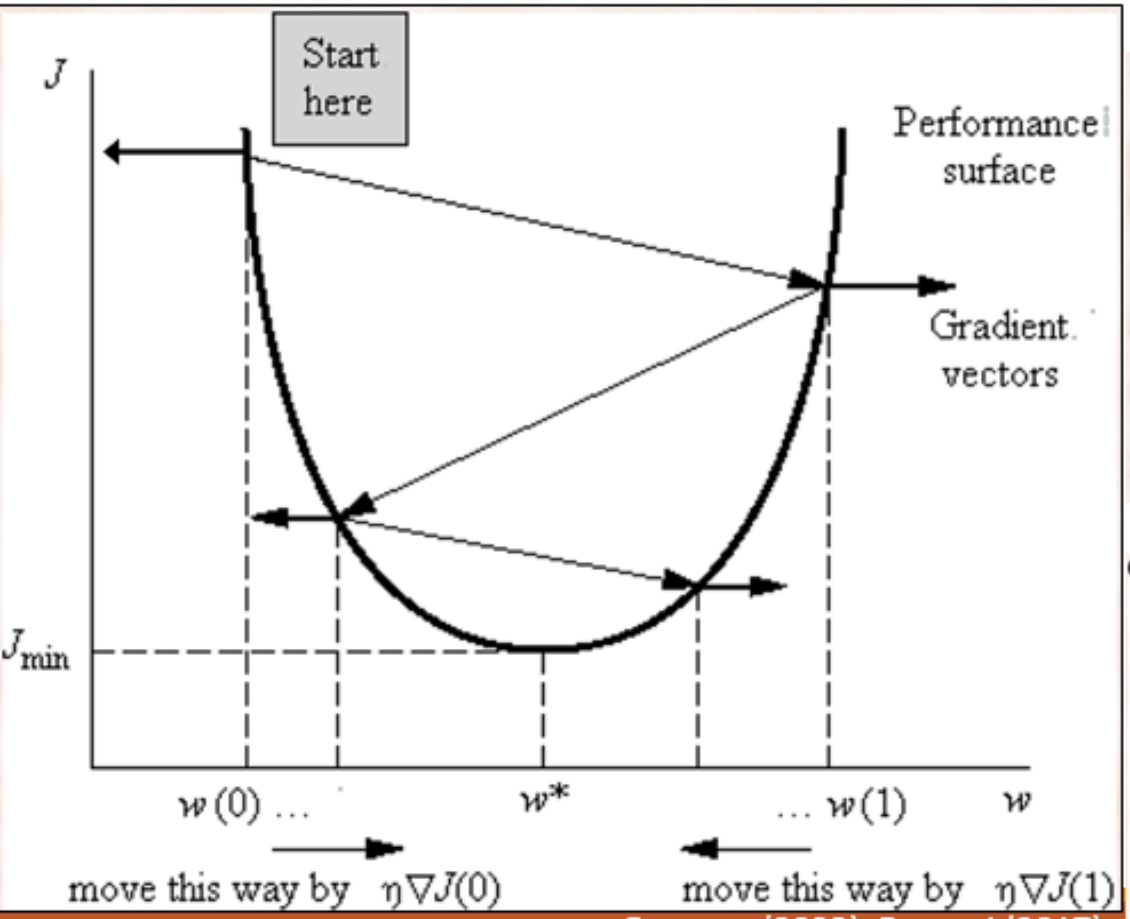
Fiecare contur interior indică o regiune mai apropiată de optimul global

Procesul de optimizare poate fi asemănat cu un joc de golf, unde prima lovitură caută sa fie cît mai aproape de destinația vizată, iar loviturile successive sunt din ce în ce mai scurte șI mai precise, corectând distanța și traiectoria loviturii în funcție de loviturile anterioare. (Parr și Howard, 2018)

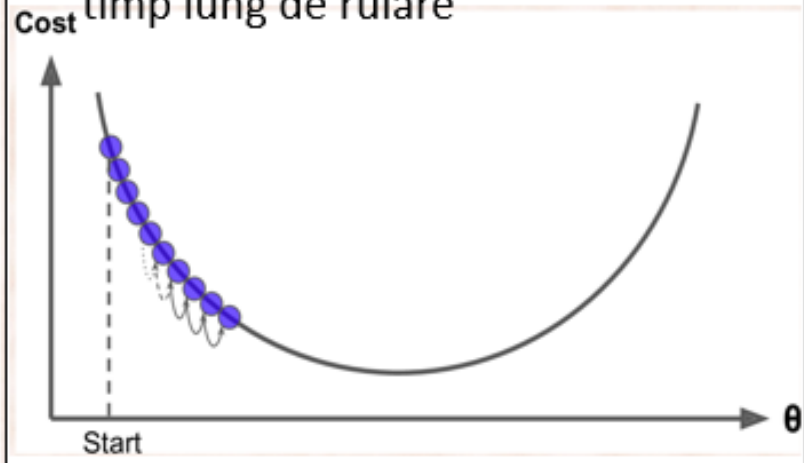
Ng (2011), Bunescu (2022), Dangeti (2017), Rokach (2019)

Gradient boosting. Noțiuni de bază

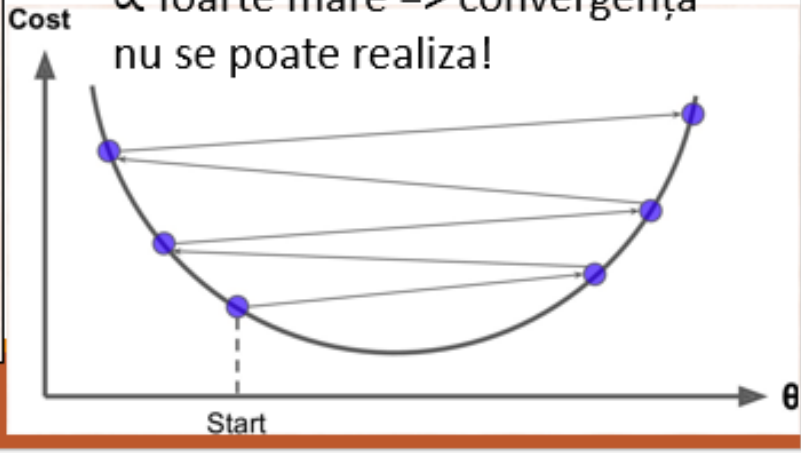
Valoarea hiperparametrului **learning rate** $\propto$ trebuie dimensionată astfel încât să nu fie prea mică sau prea mare, pentru atingerea minimului global.



$\propto$ foarte mic => convergență lentă, timp lung de rulare



$\propto$ foarte mare => convergența nu se poate realiza!



Bunescu (2022), Dangeti (2017), Ng (2011).

Gradient boosting (II)

O versiune mai formală a algoritmului gradient boosting, este prezentată mai jos

GBM Training

Input: A base regression algorithm - I, number of iterations T, the training set, $S = \{(x_i, y_i)\}_{i=1}^{m}$, and a differentiable loss function $L(y, F(x))$

1: Initialize model with a constant value: $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$

2: $j \leftarrow 1$

3: **repeat**

4: For $i = 1, \ldots, m$, compute pseudo-residuals:
$r_{ij} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{j-1}(x)}$

5: Construct regression model h_j using I using the training set $\{(x_i, r_{ij})\}_{i=1}^{m}$.

6: Find multiplier γ_j by performing line search on the following one-dimensional optimization problem:
$\gamma_j = \arg\min_{\gamma} \sum_{i=1}^{m} L\left(y_i, F_{j-1}(x_i) + \gamma h_j(x_i)\right)$.

7: Update the model: $F_j(x) = F_{j-1}(x) + \gamma_j h_j(x)$.

8: $j \leftarrow j + 1$

9: **until** $j > T$

10: Output $F_M(x)$

Notă. γ corespunde lui ∝ în acest exemplu, h(x) este notația pentru funcția de estimare f(x), iar funcția L corespunde notației din slide-urile anterioare.

Rokach (2019)

Gradient boosting (III)

Complexitatea estimării modelelor de tip gradient boosting duce în mod necesar la necesitatea optimizării implementării algoritmului.

Printre metodele folosite avem:

- **Limitarea numărului maxim de iterații T**, efectuată cu precauție deoarece poate duce la subantrenare
- Identificarea **valorii optime pentru** hiperparametrul **learning rate**. O valoare mai mică poate duce la rezultate mai bune, dar presupune un număr mai mare de iterații.
- Folosirea metodei **stochastic gradient boosting**, prin efectuarea estimărilor succesive pe subeșantioane ale setului de date de antrenare extrase fără revenire, sau prin estimarea pe o submulțime a variabilelor explicative => evitarea supraaantrenării
- Regularizarea modelelor de bază (stabilirea unui **nr maxim de noduri**, limitarea adâncimii maxime a arborilor de decizie (modelele de bază), sau folosirea unui factor de penalizare introdus în funcția de cost pentru reducerea complexității modelelor de bază).

EXtreme Gradient boosting (XGBoost) (I)

Este poate cel mai popular model de boosting. Elaborat de Chen și Guestrin (2016), a devenit cunoscut prin faptul este folosit pentru câștigarea multor competiții de machine learning.

XGBoost aduce multe optimizări algoritmului GBM.

Cea mai importantă este **introducerea unui factor de penalizare** pentru complexitatea modelelor de bază (arbori de decizie) în funcția de cost (Obj)

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training Loss measures how well model fit on training data

Regularization, measures complexity of trees

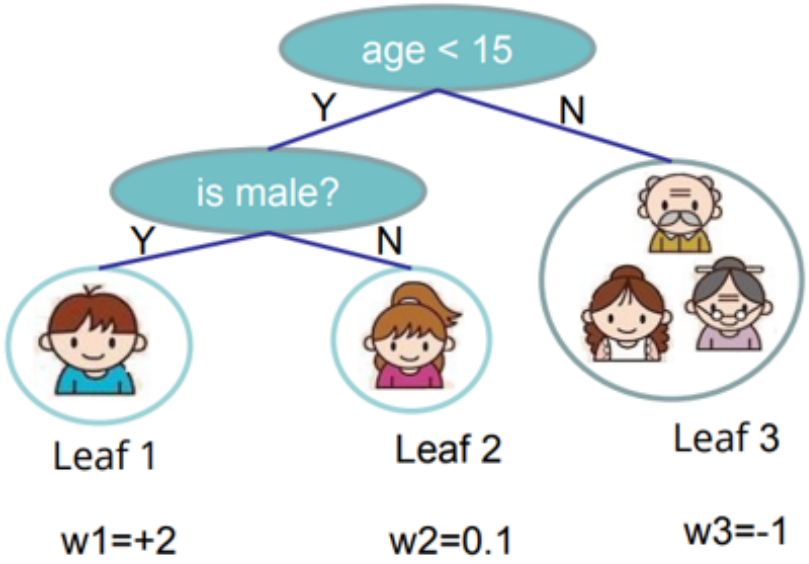
Rokach (2019), Brownlee (2018), Chen (2016)

EXtreme Gradient boosting (XGBoost) (II)

Factorul de penalizare poate fi formalizat pornind de la structura unui arbore de regresie.

Objective in XGBoost $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$

Number of leaves **L2 norm of leaf scores**



$$\Omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

Notă. γ este termenul de penalizare pentru dimensiunea arborelui de decizie T, iar λ este termenul de regularizare corespunzător rezultatelor obținute, folosit pentru prevenirea supraantrenării.
w- weights reprezintă rezultatele estimării variabilei dependente

Rokach (2019), Chen (2016)

EXtreme Gradient boosting (XGboost) (III)

Pornind de la noul tip de funcție obiectiv și ținând cont de faptul că estimarea este de tip aditiv și iterativ, funcția de cost care se estimează pentru predicția unei observații i la iterația t este

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

care se aproximează prin expansiunea seriei Taylor

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ și $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ fiind derivatele de ordinul întâi și respectiv de ordinul 2 ale funcției de cost. Eliminând termenii de tip constantă, obținem forma simplificată pentru iterația t

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

Rokach (2019), Brownlee (2018), Chen (2016)

EXtreme Gradient boosting (XGboost) (IV)

Pornind de la funcția obiectiv în formă desfășurată

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Valoarea optimă pentru estimația corespunzătoare fiecărui nod terminal j este

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

$I_j = \{i \mid q(\mathbf{x}_i) = j\}$ fiind mulțimea observațiilor clasificate ca aparținând nodului terminal j, iar $q(\mathbf{x}_i)$ funcția de mapare a observației i în nodul j.

Valoarea optimă a funcției de cost, echivalenta scorului de impuritate pentru arborii simpli de decizie, va fi calculată astfel, fiind folosită pentru dezvoltarea/estimarea modelelor de bază.

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T.$$

Rokach (2019), Brownlee (2018), Chen (2016)

EXtreme Gradient boosting (XGboost) (V)

Modul concret de estimare a câștigului de informație ca bază a deciziei de splitare/divizare a unui nod în sub-noduri este prezentat mai jos

$$gain = \frac{1}{2}[\frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} + \frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}] - \gamma$$

I_R și I_L reprezintă mulțimea observațiilor aparținînd nodului inițial, asignate nodului stâng (L) respectiv drept (R) în urma splitării.

Reprezentarea grafică a estimării unui model XGBoost este:

Rokach (2019), Chen (2016), He (2016)

XGBoost în practică I

Pachetul xgboost implementează algoritmul care, pe lângă proprietățile descrise mai sus, oferă și alte funcționalități:

- posibilitatea paralelizării estimației pe mai multe resurse informatice, putând rula de 10 ori mai rapid comparativ cu GBM
- Formate de date specifice, pentu optimizarea estimării pe matrici rare (sparse matrices) unde predomină valoarea 0.

Găsirea valorilor optime pentru hiperparametri este esențială pentru o estimare eficientă și relativ rapidă:

`eta` – learning rate, valori în intervalul [0,1]
`gamma` – valoare minimă a îmbunătățirii valorii funcție de cost necesară efectuării unei splitări, valori în intervalul [0,∞)
`max_depth`–adâncimea maximă a arborilor de decizie estimați
`subsample` – fracția de observații de eșantionat din setul de date de antrenare pentru a fi folosite în dezvoltarea modelelor de bază

https://cran.r-project.org/web//packages/xgboost/vignettes/xgboostPresentation.html

Rokach (2019), Chen și alții (2022)

XGBoost în practică II

`lambda` și `alpha` – coeficienți de regularizare a rezultatelor estimării w, folosite pentru modele de bază de tip liniar. Valori mai mari => estimări mai conservatoare

Alte opțiuni importante pentru funcția de estimare xgb.train

`objective`– modelul de bază folosit (valoare implicită: regresia liniară)

`eval_metric` – indicator de evaluare a rezultatelor estimării în funcție de modelul de bază folosit. De exemplu `eval_metric="rmse"` pentru modele de regresie, `eval_metric="auc"` pentru modele de clasificare

`maximize` – legat de valorile specificate . Valoarea TRUE înseamnă că o valoare mai mare este considerată a fi mai bună. Ex. pentru `eval_metric="rmse"`, `maximize=FALSE`

`early_stopping_rounds`- numărul de iterații după care estimarea se oprește dacă rezultatele obținute pe un set de date de validare nu se îmbunătățesc

`nrounds` – numărul maxim de iterații E

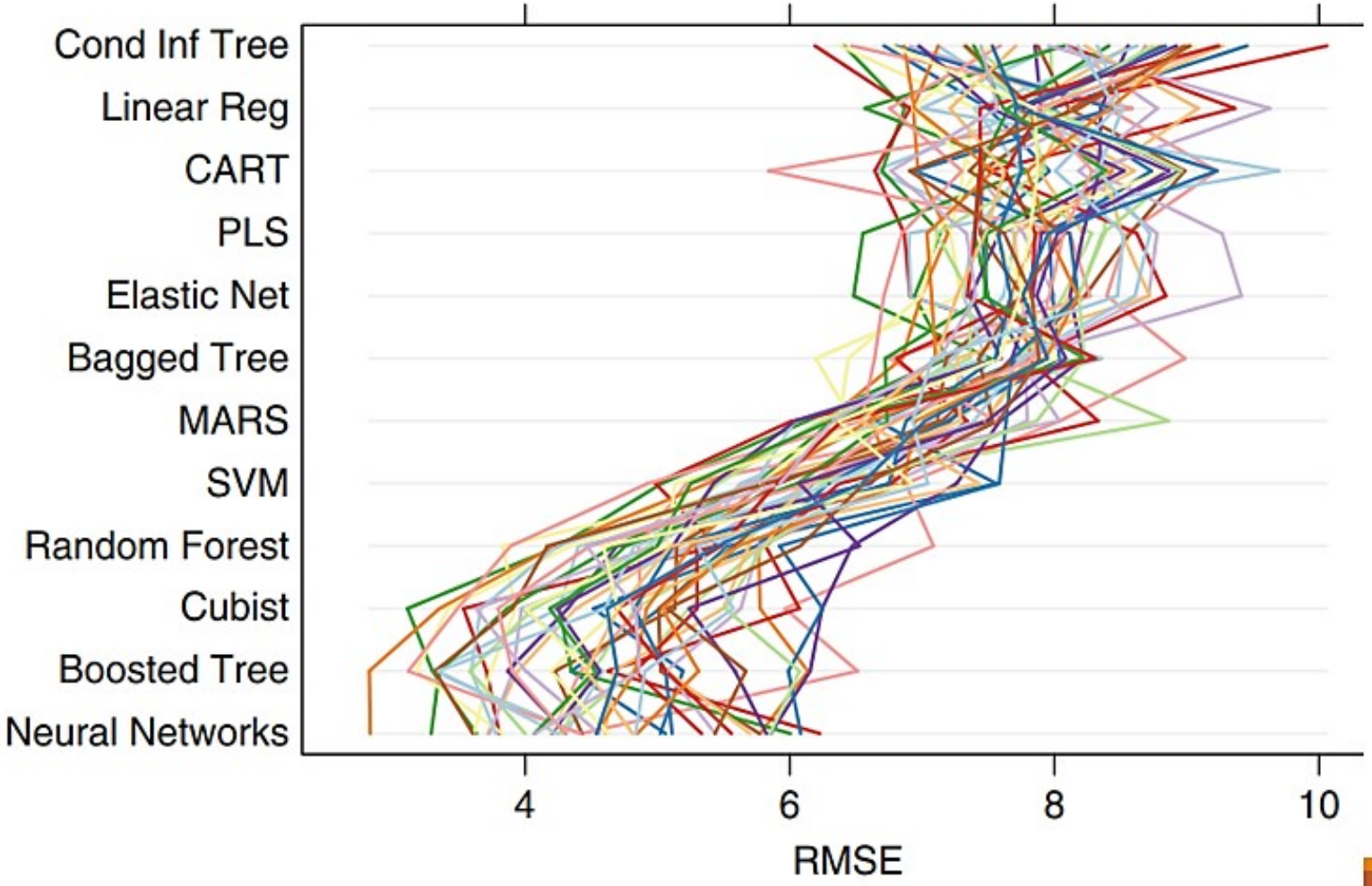
`nthread` – numărul procesărilor paralele permise. Pe un calculator personal valoarea maximă este dată de numărul de procesoare. Valoarea implicită este setată pentru numărul maxim de procesări permise

https://xgboost.readthedocs.io/en/latest/parameter.html

Rokach (2019), Chen și alții (2022)

Modelarea de tip ensemble. Aprecieri generale I

Modelele de tip ensemble s-au impus în practică datorită performanțelor superioare, datorate gestionării optime a compromisului dintre bias și variață.



Kuhn și Johnson (2013)

Modelarea de tip ensemble. Aprecieri generale II

Considerente pentru alegerea celui mai potrivit model

- **Ajustarea hiperparametrilor**. Singurul parametru important pentru modelele random forests este numărul de variabile selectate aleator pentru fiecare splitare, în timp ce modelele de tip gradient boosting folosesc mai mulți hiperparametri, dintre care **learning rate**, **numărul de iterații** și **adâncimea modelelor de bază**.
- **Supraaantrenarea**, poate apărea mai rar la modelele random forests deoarece modelele de bază sunt estimate de o manieră independentă
- **Complexitatea modelului**. Modelele random forests ating performanța maximă în urma estimării unui număr mic de arbori de complexitate mare (D>20). Cele de gradient boosting o fac prin antrenarea unui număr mare de arbori de dimensiune redusă (D<=10).
- **Performanța**. În multe cazuri, modelele de boosting au cea mai bună performanță, cu condiția setării hiperparametrilor de o manieră eficientă

Modelele random forests dau rezultate robuste într-un timp relativ scurt. Modelele de tip boosting au cea mai bună performanță, obținută cu prețul timpului necesar ajustării hiperparametrilor la valori optime.

Kuhn și Johnson (2013) Rokach (2019)

Mulțumesc pentru atenție!