Optimization by Gradient Boosting

Gerard Biau et Benoît Cadre

Thomas Gravier - Theotime Le Hellard

08 Janvier 2025

Contexte

- **Contexte** : on a des données
  $\mathscr{D}_n = \{(X_1, Y_1), ..., (X_2, Y_2)\}$ iid, on cherche une fonction $F : \mathcal{X} \mapsto \mathbb{R}$ minimisant un risque empirique :
  $C_n(F) = \frac{1}{n} \sum_{i=1}^{n} \psi(F(X_i), Y_i)$
  En passant sur une distribution : $C(F) = \mathbb{E}\psi(F(X), Y)$

Contexte

- **Contexte** : on a des données $\mathscr{D}_n = \{(X_1, Y_1), ..., (X_2, Y_2)\}$ iid, on cherche une fonction $F : \mathcal{X} \mapsto \mathbb{R}$ minimisant un risque empirique : $C_n(F) = \frac{1}{n} \sum_{i=1}^{n} \psi(F(X_i), Y_i)$
  En passant sur une distribution : $C(F) = \mathbb{E}\psi(F(X), Y)$
- **Gradient boosting** : construit progressivement un F par somme de termes provenant d'un $\mathscr{F}$, de sorte qu'on cherche $inf_{F \in lin(\mathscr{F})}\ C(F)$
  Par exemple en prenant $\mathscr{F}$ les arbres de décisions, on additionne à F un $f = \sum_{j=1}^{k} \beta_j \mathbb{1}_{A_j}$.

Contexte

- **Contexte** : on a des données $\mathscr{D}_n = \{(X_1, Y_1), ..., (X_2, Y_2)\}$ iid, on cherche une fonction $F : \mathcal{X} \mapsto \mathbb{R}$ minimisant un risque empirique : $C_n(F) = \frac{1}{n} \sum_{i=1}^{n} \psi(F(X_i), Y_i)$
  En passant sur une distribution : $C(F) = \mathbb{E}\psi(F(X), Y)$
- **Gradient boosting** : construit progressivement un F par somme de termes provenant d'un $\mathscr{F}$, de sorte qu'on cherche $inf_{F \in lin(\mathscr{F})}\ C(F)$
  Par exemple en prenant $\mathscr{F}$ les arbres de décisions, on additionne à F un $f = \sum_{j=1}^{k} \beta_j \mathbb{1}_{A_j}$.
- $\Rightarrow$ En notant ξ une sub diff de ψ, on est tenté de faire une descente de gradient stochastique en prenant $f_t = -\xi(F_t(X), Y)$, sauf qu'on est contraint à chercher dans $\mathscr{F}$.

Algorithmes

Algorithme 1 : par dévelopement de Taylor à l'ordre 1, $C(F) - C(F + wf) \simeq -w\langle \nabla C(F), f \rangle_{\mu_X}$; ainsi pour maximiser la descente on prend :

$$f_{t+1} \in arg\, max_{f \in \mathscr{F}} - \mathbb{E}[\xi(F_t(X), Y) f(X)]$$

$$F_{t+1} = F_t + w_{t+1} f_{t+1}$$

Algorithmes

Algorithme 1 : par dévelopement de Taylor à l'ordre 1, $C(F) - C(F + wf) \simeq -w\langle \nabla C(F), f \rangle_{\mu_X}$; ainsi pour maximiser la descente on prend :

$$f_{t+1} \in arg\, max_{f \in \mathscr{F}} - \mathbb{E}[\xi(F_t(X), Y) f(X)]$$
$$F_{t+1} = F_t + w_{t+1} f_{t+1}$$

Algorithme 2 : directement en cherchant le $f \in \mathscr{F}$ le plus proche du pas de gradient :

$$f_{t+1} \in arg\, min_{f \in \mathscr{P}} \mathbb{E}[-\xi(F_t(X), Y) - f(X)]^2$$

qui devient :

$$f_{t+1} \in arg\, min_{f \in \mathscr{P}} [2\mathbb{E}(\xi(F_t(X), Y) f(X)) + ||f||^2_{\mu_X}]$$
$$F_{t+1} = F_t + \nu f_{t+1}$$

Hypothèses

But : Prouver que ces algorithmes convergent vers l'optimal : $lim_{t \to \infty} C(F_t) = inf_{F \in lin(\mathscr{F})} C(F)$

Hypothèses

But : Prouver que ces algorithmes convergent vers l'optimal : $lim_{t\to\infty} C(F_t) = inf_{F\in lin(\mathscr{F})}\ C(F)$

- $\mathbf{A_1}$: Les dérivées de ψ sont localement bornées et donc C l'est : $E(\psi(0, Y)) < \infty$; et

$$sup_{G\in L^2(\mu_X):||G-F||_{\mu_X}\leq\delta}(E|\partial_x^- \psi(G(X),Y)|^2+E|\partial_x^+ \psi(G(X),Y)|^2)<\infty$$

Hypothèses

But : Prouver que ces algorithmes convergent vers l'optimal : $lim_{t\to\infty} C(F_t) = inf_{F\in lin(\mathscr{F})}\ C(F)$

- $\mathbf{A_1}$: Les dérivées de ψ sont localement bornées et donc C l'est : $E(\psi(0,Y)) < \infty$; et

  $sup_{G\in L^2(\mu_X):||G-F||_{\mu_X}\leq\delta}(E|\partial_x^- \psi(G(X),Y)|^2+E|\partial_x^+ \psi(G(X),Y)|^2)<\infty$
- $\mathbf{A_2}$: $\psi(.,y)$ est α-Strongly Convex (et donc C l'est).

Hypothèses

But : Prouver que ces algorithmes convergent vers l'optimal : $lim_{t\to\infty} C(F_t) = inf_{F\in lin(\mathscr{F})}\ C(F)$

- $\mathbf{A_1}$: Les dérivées de ψ sont localement bornées et donc C l'est : $E(\psi(0,Y)) < \infty$; et

  $sup_{G\in L^2(\mu_X):||G-F||_{\mu_X}\leq\delta}(E|\partial_x^- \psi(G(X),Y)|^2+E|\partial_x^+ \psi(G(X),Y)|^2)<\infty$
- $\mathbf{A_2}$: $\psi(.,y)$ est α-Strongly Convex (et donc C l'est).
- $\mathbf{A_3}$: Le gradient de ψ est L-Lipschitz

  version faible : $\forall x_1,x_2 \quad |\mathbb{E}(\xi(x_1,Y)-\xi(x_2,Y)\ |X)|\leq L|x_1-x_2|$

  version forte : $|\partial_x\psi(x_1,y)-\partial_x\psi(x_2,y)|\leq L|x_1-x_2|$

Hypothèses

But : Prouver que ces algorithmes convergent vers l'optimal : $lim_{t\to\infty} C(F_t) = inf_{F\in lin(\mathscr{F})}\ C(F)$

- $\mathbf{A_1}$: Les dérivées de ψ sont localement bornées et donc C l'est : $E(\psi(0,Y)) < \infty$; et

  $sup_{G\in L^2(\mu_X):||G-F||_{\mu_X}\leq\delta}(E|\partial_x^- \psi(G(X),Y)|^2+E|\partial_x^+ \psi(G(X),Y)|^2)<\infty$
- $\mathbf{A_2}$: $\psi(.,y)$ est α-Strongly Convex (et donc C l'est).
- $\mathbf{A_3}$: Le gradient de ψ est L-Lipschitz

  version faible : $\forall x_1,x_2 \quad |\mathbb{E}(\xi(x_1,Y)-\xi(x_2,Y)\ |X)|\leq L|x_1-x_2|$

  version forte : $|\partial_x\psi(x_1,y)-\partial_x\psi(x_2,y)|\leq L|x_1-x_2|$
- $\mathbf{A_4}$: Dans le cas d'un $\psi(x,y) = \phi(x,y) + \gamma x^2$; avoir ϕ localement Lipschitz en x :

  $\forall p\geq 0,\ \exists\zeta(p)>0,\ \forall x_1,x_2,y\in\mathbb{R}^2\times\mathscr{Y}$ avec $max(|x_1|,|x_2|)\leq p$

  $\Rightarrow |\phi(x_1,y)-\phi(x_2,y)| \leq \zeta(p)|x_1-x_2|$

Les théorèmes de convergence

A1 dérivées bornées, A2 Strongly convex, A3 gradient ψ L-Lipschitz, A4 ϕ localement Lipschitz.

- **Théorème 3.1** : $\mathbf{A_1}$ et $\mathbf{A_3}$ (faible), alors l'Algo 1 converge vers $C(F^*)$ optimal.
  En rajoutant $\mathbf{A_2}$ CV dans la closure et $F_t \to F^*$.
  Note : dans la preuve il faut
  $w_{t+1} = min(w_t, -(2L)^{-1}\mathbb{E}[\xi(F_t(X), Y)f_{t+1}(X)])$

Les théorèmes de convergence

A1 dérivées bornées, A2 Strongly convex, A3 gradient ψ L-Lipschitz, A4 ϕ localement Lipschitz.

- **Théorème 3.1** : $\mathbf{A_1}$ et $\mathbf{A_3}$ (faible), alors l'Algo 1 converge vers $C(F^*)$ optimal.
  En rajoutant $\mathbf{A_2}$ CV dans la closure et $F_t \to F^*$.
  Note : dans la preuve il faut
  $w_{t+1} = min(w_t, -(2L)^{-1}\mathbb{E}[\xi(F_t(X), Y)f_{t+1}(X)])$

- **Théorème 3.2** : $\mathbf{A_1}$, $\mathbf{A_2}$ et $\mathbf{A_3}$ (faible), et $0 < \nu < 1/2L$, alors l'Algo 2 converge idem.
  Pour la convergence $F_t \to F^*$ il faut $\mathbf{A_3}$ (fort).

Théorème pour éviter l'overfit

Théorème 4.1 : conditions pour ne pas *overfit*. Dans le contexte empirique, pour n données, on aboutit à un $\hat{F}_n$ optimal sur ces n données, mais donc risque d'overfit. But : pour n grandissant $C(\hat{F}_n) \to C(F^*)$ sur la vraie distribution. Idée : limiter la taille de $\mathscr{F}_n$ disponible quand on a n données.

Théorème pour éviter l'overfit

Théorème 4.1 :
Contexte : $\psi(x,y) = \phi(x,y) + \gamma_n x^2$ avec $\mathscr{F}$ des decision trees et $\bar{\phi} = sup_y \phi(0,y) < \infty$.
$\mathbf{A_1}$ et $\mathbf{A_2}$ sont vérifiées ; requière en plus $\mathbf{A_3}$ et $\mathbf{A_4}$; et le controle sur la taille des decision trees :
$\gamma_n \to 0$; $N \to \infty$; $\frac{logN}{n \cdot \nu_n} \to 0$; une densité g pour X avec $0 < inf_X g \leq sup_X g < \infty$; et $\frac{1}{\sqrt{n\nu_n\gamma_n}} \zeta\left(\sqrt{\frac{2\bar{\phi}}{\nu_n \gamma_n \, inf_X g}}\right) \to 0$
Pour aboutir à $lim_{n\to\infty} A(\bar{F}_n) = A(F^*)$;
où A est l'objectif sans la régularisation :
$A_n(F) = \frac{1}{n}\sum_i^n \phi(F(X_i), Y_i)$; $A(F) = \mathbb{E}(\phi(F(X), Y))$

Étude des hypothèses

- $\mathbf{A_1}$ (i.e. dérivées bornées) est raisonnable : le gradient ne doit pas exploser localement (sup sur une boule $< \infty$).
- $\mathbf{A_2}$ (i.e. strongly convex) très commun dans les preuves d'optimisation, donne l'unicité et la convergence des suites (ici F_n) en plus de celle des valeurs (ici $C(F_n)$). Passe au besoin par une régularisation.

Étude des hypothèses

- **A_3** idem dans les cours d'optimisation, commun d'avoir cette hypothèse de *smoothness* : le gradient est L-Lipschitz.
  Par ailleurs, l'idée d'une inégalité faible, avec juste l'espérance, et une forte en tout point, nous rappelle la démonstration de la descente de gradient stochastique : On avait : $\nabla J(\theta) = \mathbb{E}_w(g(\theta, w))$; donc le g est notre ξ ici ; et forte $||g(\theta, w)||_2^2 \leq B^2$; faible $E(||g(\theta, w)||_2^2) \leq B^2$

Étude des hypothèses

- $\mathbf{A_3}$ idem dans les cours d'optimisation, commun d'avoir cette hypothèse de *smoothness* : le gradient est L-Lipschitz.
  Par ailleurs, l'idée d'une inégalité faible, avec juste l'espérance, et une forte en tout point, nous rappelle la démonstration de la descente de gradient stochastique : On avait : $\nabla J(\theta) = \mathbb{E}_w(g(\theta, w))$; donc le g est notre ξ ici ; et forte $||g(\theta, w)||_2^2 \leq B^2$; faible $E(||g(\theta, w)||_2^2) \leq B^2$
- $\mathbf{A_4}$ va avec toutes les hypothèses qui l'accompagne, mais est satisfait par toutes les loss convexes standards.

Remarques / critiques

- Les iterations des algos sont théoriques, avec $f_{t+1} \in arg\ max_{f \in \mathcal{F}} \mathbb{E}...$; $w_{t+1} = min(w_t, -(2L)^{-1}\mathbb{E}[\xi(F_t(X), Y) f_{t+1}(X)])$, là où les algos pratiques sont heuristiques
- Le théorème 4.1 est clé : sous couvert que les algos convergent, et des hypothèses, pas d'overfit.

Remarques / critiques

- Pour la régularisation, ils proposent $\psi(x, y) = \phi(xy) + \gamma x^2$, mais en pratique la régularisation porte généralement sur la structure / les poids de la fonction (le w) ; pas les valeurs de $F(x)$. D'ailleurs XGBoost introduit le regularized objective comme :

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

Remarques / critiques

- Pour la régularisation, ils proposent $\psi(x, y) = \phi(xy) + \gamma x^2$, mais en pratique la régularisation porte généralement sur la structure / les poids de la fonction (le w) ; pas les valeurs de $F(x)$. D'ailleurs XGBoost introduit le regularized objective comme :

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$
$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

- Sur les notations, entre A, A_j^n, A_n, $A^n(x)$... qui désigne d'un coté des sets de l'arbre, de l'autre l'objectif non régularisé.

Idea of Gradient Boosting Decision Tree

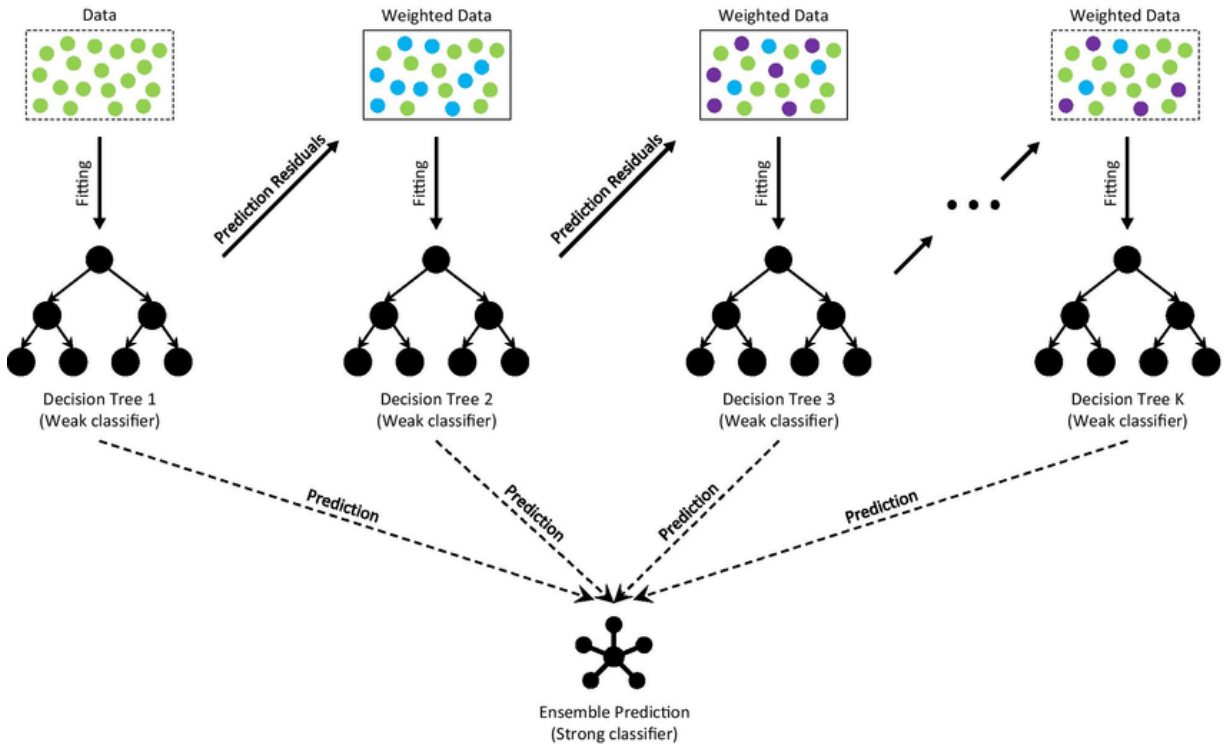

Figure – Architecture of the Gradient Boosting Decision Tree (Source : Deng et Al)

Project Architecture

The project was implemented as follows with CART weak learner :

- `boosting_fromscratch/`
  - `deep_learning/`
    - `loss_functions.py` : Contains loss functions used for model optimization, such as Mean Squared Error and Log-Loss.
  - `supervised_learning/`
    - `decision_tree.py` : Implements decision trees for classification and regression.
    - `gradient_boosting.py` : Contains the implementation of Gradient Boosting.
  - `utils/`
    - `data_manipulation.py` : Functions for data preprocessing.
    - `data_operation.py` : Mathematical functions used in the algorithms.

Gradient Boosting Classifier Test

Dataset	Implementation	Accuracy (%)	Training Time (s)	Inference Time (s)
Iris	Boosting from scratch	92.3	9.2	0.01
Iris	Scikit-learn	94.0	0.22	0.001
Wine Quality	Boosting from scratch	83.5	90.0	0.02
Wine Quality	Scikit-learn	87.5	0.31	0.001

Table – Comparison of Gradient Boosting implementations on Iris and Wine Quality datasets.

Performance Comparison : Custom vs Scikit-learn

- `Decision Tree:`
  - **Threshold Calculation :** Scikit-learn uses histogram-based methods, and not all treshold
  - **Data Manipulation :** Concatenates `X` and `y` (memory-intensive), Scikit-learn manipulates data indices directly.
  - **Stopping Criteria :** Scikit-learn adds impurity-based early stopping to limit complexity
- `Gradient Boosting:`
  - **Gradient Updates :** Scikit-learn fully vectorizes this process.
  - **Tree Training :** Multi-threading
  - **Memory Management :** Scikit-learn optimizes memory via index-based manipulation.

Modifications to Gradient Boosting

Objective : Adapt the Gradient Boosting algorithm to satisfy the theorem's conditions and avoid overfitting.

Key Changes :

- **Tree complexity :** Maximum tree depth is scaled as max_depth $\propto \log_2(n)$ to prevent excessive complexity for small datasets.
- **Dynamic regularization :** A regularization term $\gamma_n = \frac{1}{n}$ is added to the gradient update.
- **Gradient stabilization :** The gradient includes a regularization term $\gamma_n x^2$ to control convergence and avoid divergence.

Impact : Ensures convergence to F^* while maintaining model generalization.

Evaluating Overfitting in Gradient Boosting (Wine Quality Dataset)

Dataset Size (%)	Model	Train Error	Test Error	Gap (Test - Train)
Small (10%)	Gradient Boosting (Base)	Low	High	Large
Small (10%)	Gradient Boosting (Modified)	Slightly Higher	Lower	Small
Medium (30%)	Gradient Boosting (Base)	Very Low	Moderate	Moderate
Medium (30%)	Gradient Boosting (Modified)	Low	Slightly Lower	Small
Large (100%)	Gradient Boosting (Base)	Very Low	Low	Small
Large (100%)	Gradient Boosting (Modified)	Low	Low	Very Small

Table – Comparison of overfitting with and without modifications to Gradient Boosting on the Wine Quality dataset.

Conclusion