Chapitre 12 : Méthodes Ensemblistes

26 février 2026

Plan du chapitre

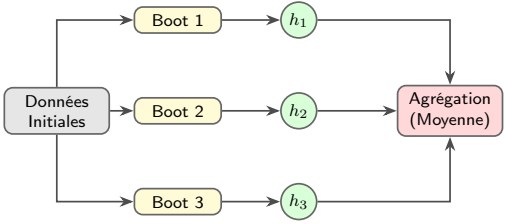
Le concept des Méthodes Ensemblistes

Idée : Combiner plusieurs modèles de base (*apprenants faibles*) pour créer un modèle global plus performant et robuste.

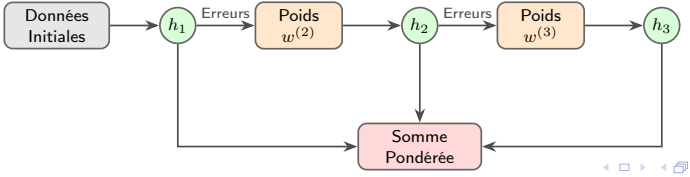
Deux grandes familles s'opposent géométriquement et statistiquement :

- **Averaging (Moyenne) :** Bagging, Forêts Aléatoires.
  - Modèles entraînés en *parallèle* sur des perturbations des données.
  - Objectif analytique : Réduire la **variance** sans trop altérer le biais.
- **Boosting (Séquentiel) :** AdaBoost, Gradient Boosting.
  - Modèles entraînés en *série*, le modèle m modélisant l'erreur résiduelle du modèle $m-1$.
  - Objectif analytique : Réduire massivement le **biais** (tout en régularisant).

1. BAGGING (Parallèle)



2. BOOSTING (Séquentiel)



Le Bagging (Bootstrap Aggregating)

Algorithme idéal pour les modèles à forte variance (ex : arbres profonds).

1. **Bootstrap :** Tirer B échantillons $\mathscr{S}_1, \ldots, \mathscr{S}_B$ de taille n, avec remise, à partir de $\mathscr{S}$.
2. **Apprentissage :** Entraîner h_b sur chaque $\mathscr{S}_b$ indépendamment.
3. **Agrégation :**
   - Classification (Vote majoritaire) :

$$h_{\textbf{bag}}(\mathbf{x}) = \text{sign}\left(\sum_{b=1}^{B} h_b(\mathbf{x})\right)$$

   - Régression (Moyenne) :

$$h_{\textbf{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} h_b(\mathbf{x})$$

Bagging et réduction de la Variance

Soient B modèles $h_1, \ldots, h_B$ identiquement distribués (car entraînés sur des échantillons Bootstrap), mais **non indépendants** (puisqu'ils proviennent du même jeu $\mathscr{S}$).

Si chaque modèle a une variance σ^2 et que la corrélation par paire entre deux modèles est ρ, la variance de l'estimateur moyen $H(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} h_b(\mathbf{x})$ s'écrit :

Décomposition de la variance

$$\mathrm{Var}(H(\mathbf{x})) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Lorsque le nombre de modèles $B \to \infty$, le second terme s'annule. La variance ne tend pas vers zéro mais vers une limite incompressible : $\rho\sigma^2$.

L'unique moyen d'améliorer asymptotiquement le Bagging est de **réduire la corrélation** ρ.

Les Forêts Aléatoires (Random Forests)

C'est du Bagging appliqué aux arbres de décision qui force une **décorrélation** des arbres par une **double randomisation** pour casser la corrélation entre arbres (réduction de ρ)

Le sous-échantillonnage des caractéristiques

À chaque nœud de chaque arbre, la recherche de la meilleure coupure ne s'optimise pas sur l'espace complet $\mathbb{R}^d$, mais sur un sous-espace aléatoire $\mathbb{R}^m$ avec typiquement $m = \lfloor \sqrt{d} \rfloor$. Ceci casse la domination d'une "caractéristique très forte" sur l'ensemble de la forêt.

Théorème Out-Of-Bag (OOB) : La probabilité qu'une observation soit exclue d'un échantillon Bootstrap de taille n (tirage avec remise) est :

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 36.8\%$$

Ces données ignorées forment un ensemble de validation gratuit.
L'erreur OOB moyenne donne un estimateur non-biaisé du risque de généralisation.

Modélisation Additive Séquentielle (FSAM)

La vaste majorité des algorithmes de Boosting s'inscrivent dans un cadre d'optimisation rigoureux : le **Forward Stagewise Additive Modeling**.
On cherche la fonction prédictive $F(\mathbf{x})$ comme combinaison linéaire d'apprenants faibles paramétrés $h(\mathbf{x}; \theta_m)$:

$$F_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h(\mathbf{x}; \theta_m)$$

Minimisation Gloutonne (Greedy Optimization)

Minimiser le risque empirique complet sur l'ensemble des paramètres est intractable. On l'optimise donc itérativement en gelant les termes précédents :

$$(\alpha_m, \theta_m) = \arg\min_{\alpha, \theta} \sum_{i=1}^{n} L\Big(y_i, F_{m-1}(\mathbf{x}_i) + \alpha h(\mathbf{x}_i; \theta)\Big)$$

Optimisation locale du Risque Exponentiel

AdaBoost : FSAM pour $y \in \{-1, 1\}$ et perte exponentielle $L(y, F) = e^{-yF}$.
À l'étape m, on cherche le classifieur h et $\alpha > 0$ qui minimisent :

$$J_m = \sum_{i=1}^{n} \overbrace{e^{-y_i F_{m-1}(\mathbf{x}_i)}}^{w_i^{(m)}} e^{-\alpha y_i h(\mathbf{x}_i)}$$

Puisque y_i et $h(\mathbf{x}_i)$ valent ± 1, leur produit vaut 1 si prédiction correcte, et -1 sinon.
On scinde la somme :

$$J_m = e^{-\alpha} \sum_{y_i = h(\mathbf{x}_i)} w_i^{(m)} + e^{+\alpha} \sum_{y_i \neq h(\mathbf{x}_i)} w_i^{(m)}$$

En ajoutant et soustrayant $e^{-\alpha} \sum_{y_i \neq h(\mathbf{x}_i)} w_i^{(m)}$:

$$J_m = e^{-\alpha} \sum_{i=1}^{n} w_i^{(m)} + \underbrace{(e^{+\alpha} - e^{-\alpha})}_{>0 \text{ si } \alpha > 0} \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}_{y_i \neq h(\mathbf{x}_i)}$$

En notant $W_m = \sum w_i^{(m)}$ et $\epsilon_m = \frac{1}{W_m} \sum w_i^{(m)} \mathbb{1}_{y_i \neq h(\mathbf{x}_i)}$ l'erreur pondérée, on a :

$$J_m = W_m \left[e^{-\alpha}(1 - \epsilon_m) + e^{+\alpha} \epsilon_m \right].$$

1. Classifieur optimal h_m

Minimiser J_m par rapport à h revient simplement à minimiser l'erreur pondérée :

$$h_m = \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}_{y_i \neq h(\mathbf{x}_i)}$$

2. Poids optimal du modèle α_m

En dérivant J_m par rapport à α et en annulant la dérivée :

$$-e^{-\alpha}(1 - \epsilon_m) + e^{+\alpha} \epsilon_m = 0 \implies e^{2\alpha} = \frac{1 - \epsilon_m}{\epsilon_m} \implies \alpha_m = \frac{1}{2} \log \left(\frac{1 - \epsilon_m}{\epsilon_m} \right)$$

Mise à jour des poids

Comment évoluent les poids des observations pour l'itération $m+1$?

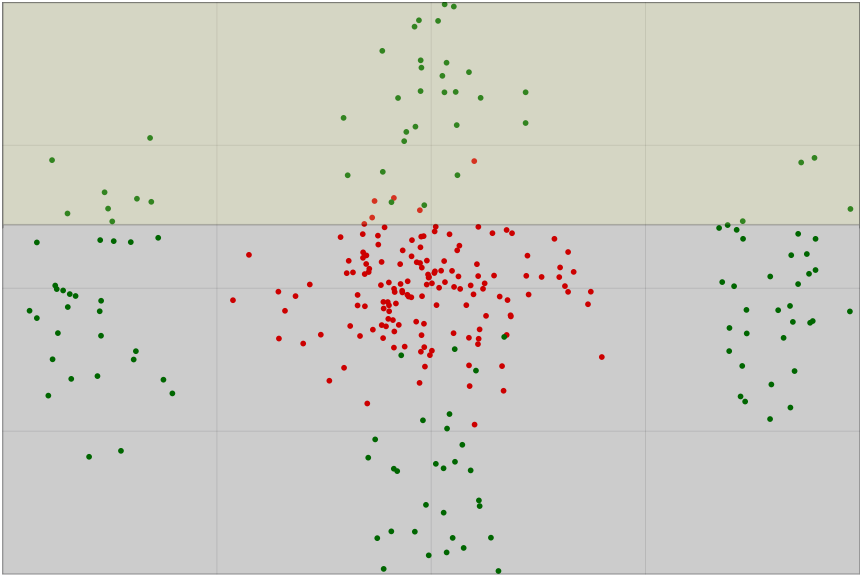
$$w_i^{(m+1)} = \exp(-y_i F_m(\mathbf{x}_i)) = w_i^{(m)} \exp\Big(- \alpha_m y_i h_m(\mathbf{x}_i)\Big)$$

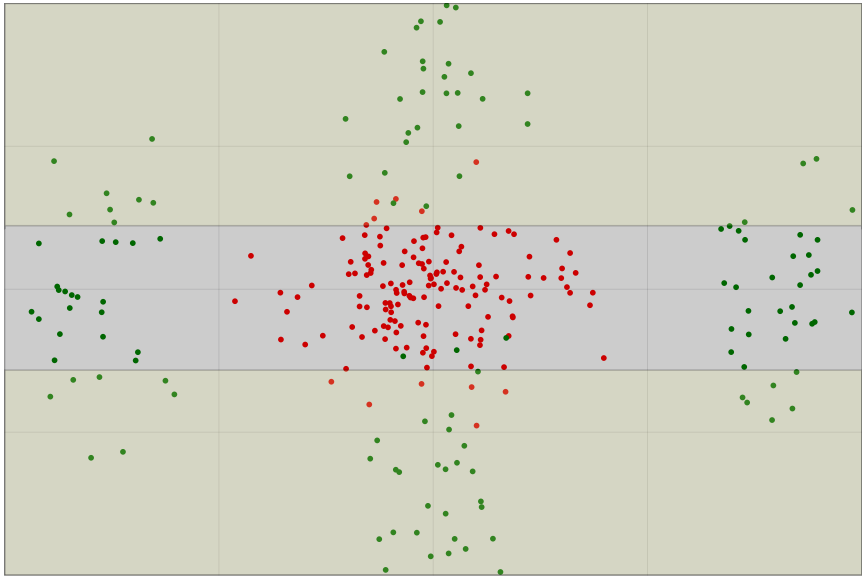
On a $-y_i h_m(\mathbf{x}_i) = 2\mathbb{1}_{y_i \neq h_m(\mathbf{x}_i)} - 1$, donc :

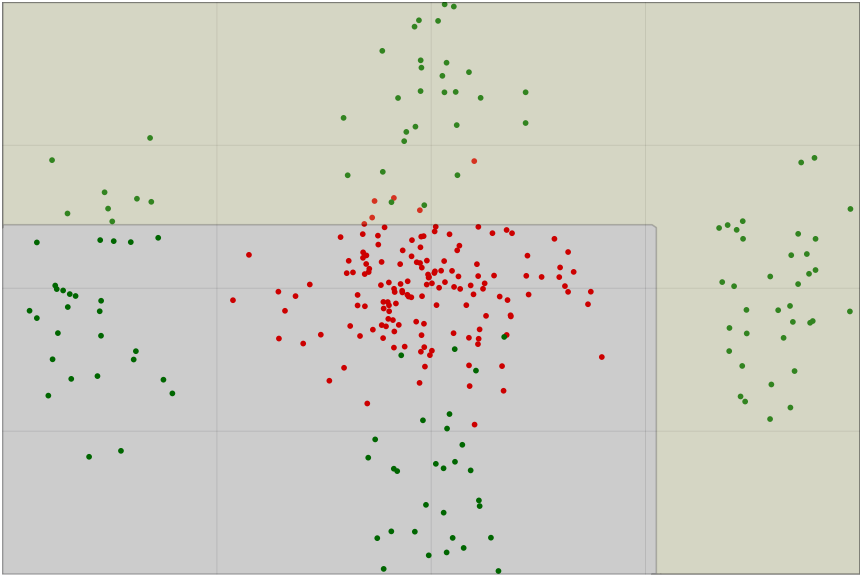
3. Mise à jour des poids

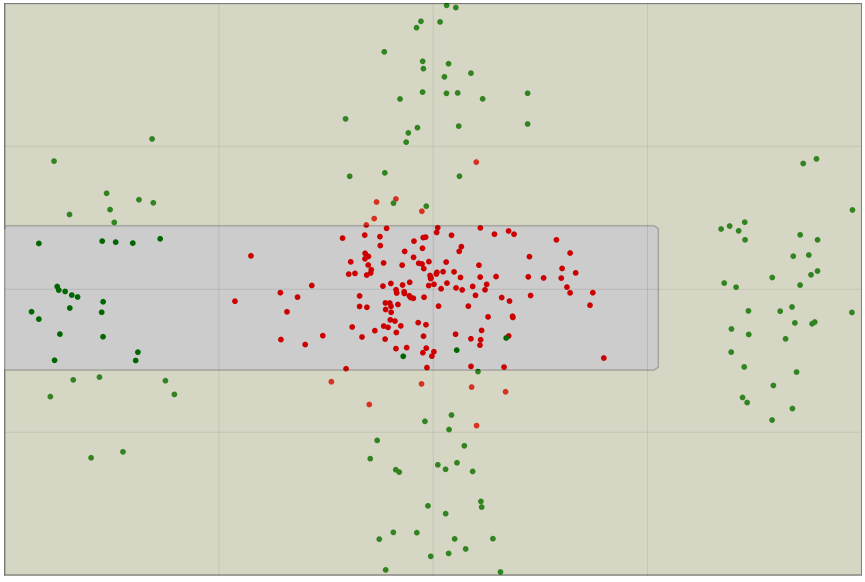
$$w_i^{(m+1)} = w_i^{(m)} \exp\Big(2\alpha_m \mathbb{1}_{y_i \neq h_m(\mathbf{x}_i)}\Big) \cdot e^{-\alpha_m}.$$

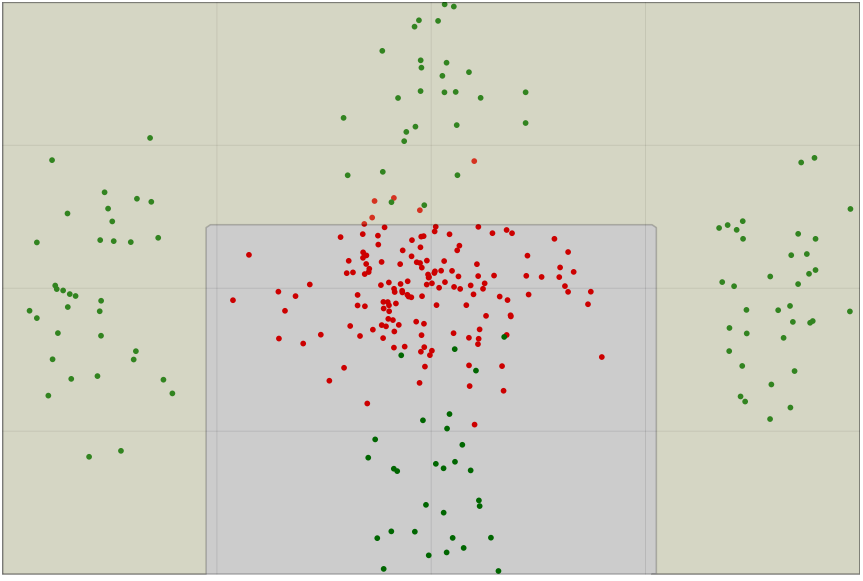
Le terme multiplicatif constant $e^{-\alpha_m}$ est ignoré en pratique, car le vecteur de poids $\mathbf{w}$ est systématiquement **renormalisé** à chaque itération pour que $\sum w_i = 1$.
L'algorithme accroît donc de manière exponentielle l'importance des erreurs.

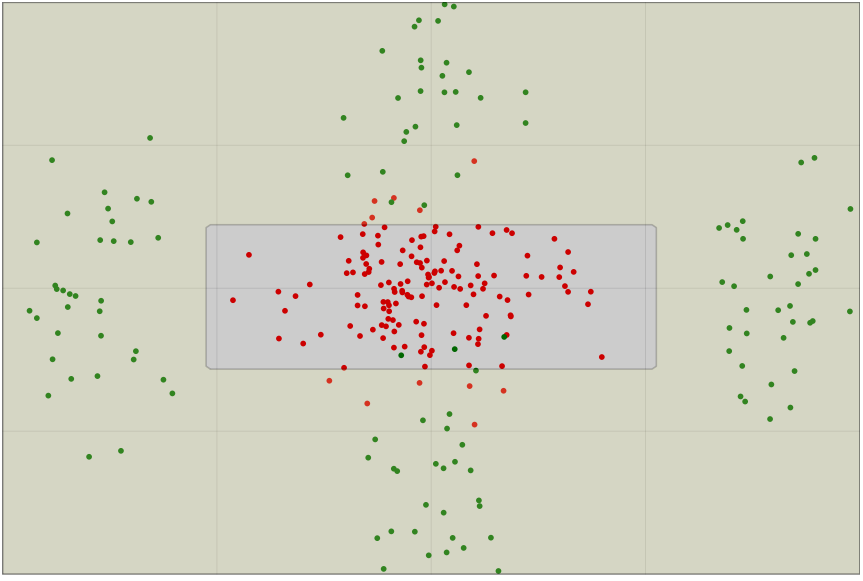


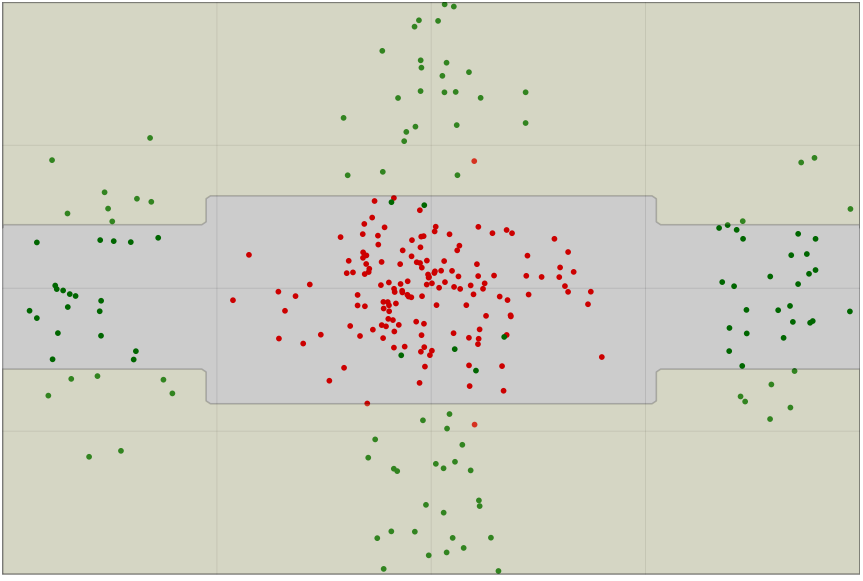


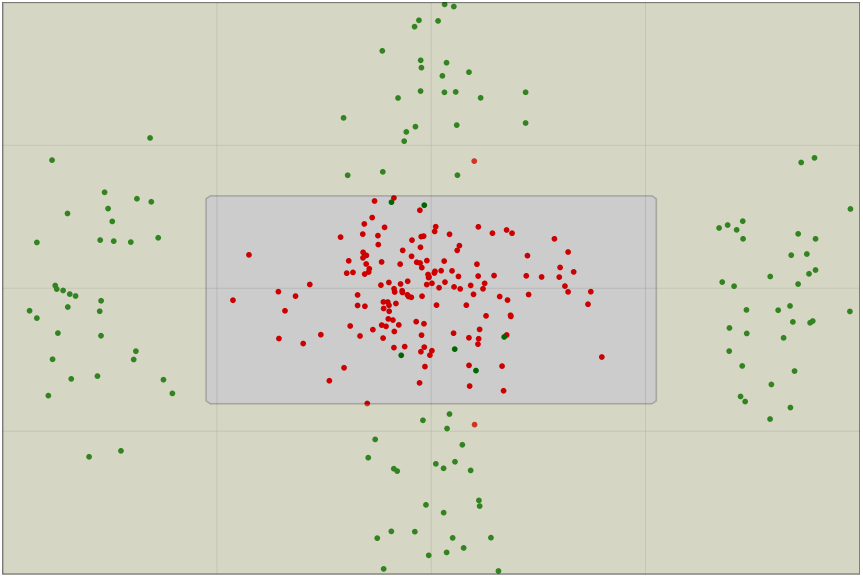


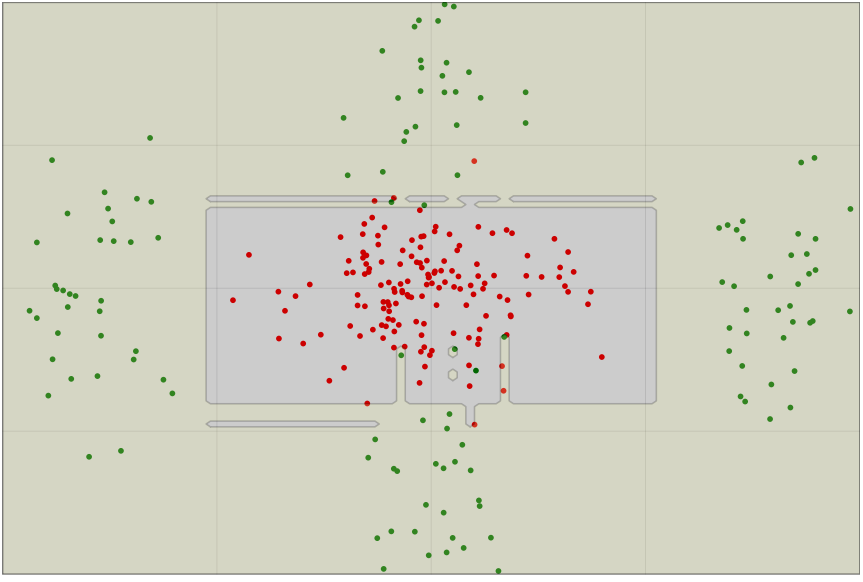


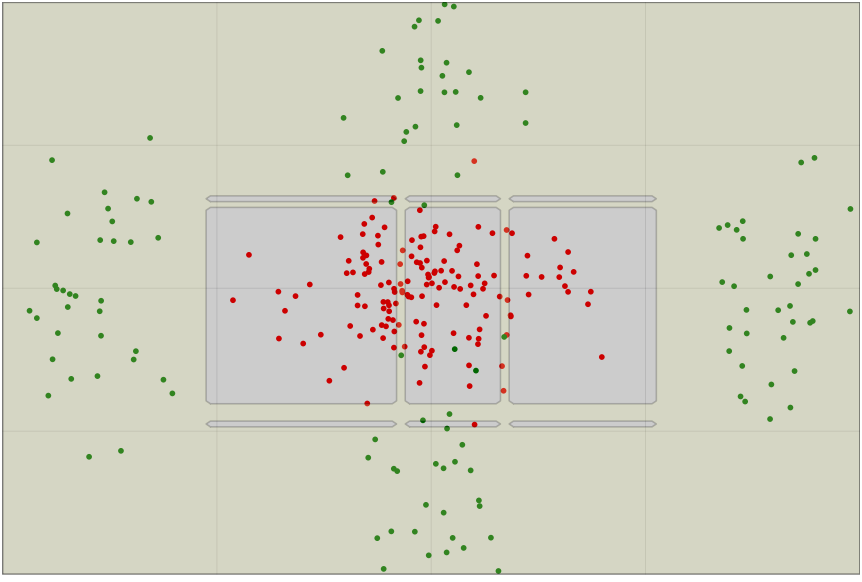


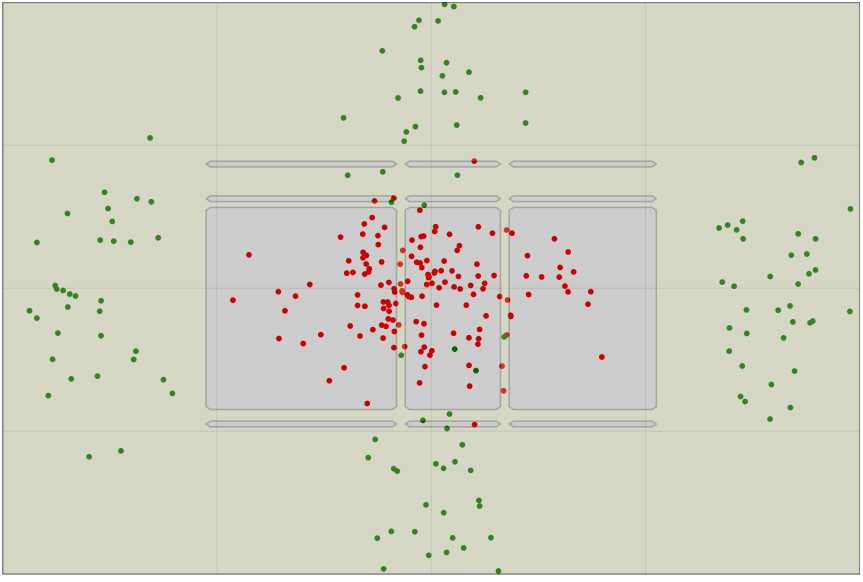




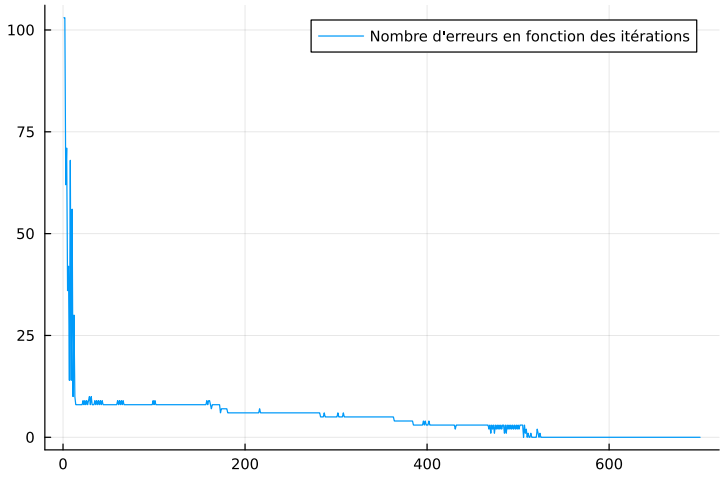








Nombre d'erreurs au fil des itérations



Limites d'AdaBoost et Descente Fonctionnelle

AdaBoost est très sensible aux valeurs aberrantes à cause de la croissance exponentielle de sa fonction de perte pour les fortes erreurs.

Le défi du Gradient Boosting : Généraliser l'algorithme à **n'importe quelle fonction de perte** $L(y, F)$ différentiable (MSE, Huber, Log-Loss).

Descente de Gradient dans l'Espace des Fonctions

J. Friedman (2001) formule l'apprentissage comme une optimisation sur la fonction continue $\mathbf{F} = (F(\mathbf{x}_1), \ldots, F(\mathbf{x}_n))^\top$. La direction de plus forte pente descendante est le gradient négatif, appelé **pseudo-résidu** :

$$r_{im} = - \left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)} \right]_{F=F_{m-1}}$$

Gradient Tree Boosting : L'Étape de Projection

On ne peut pas simplement additionner ces gradients r_{im} sous peine de surapprentissage total et d'incapacité à généraliser à des données invisibles. On "projette" ce champ de gradients sur notre espace d'hypothèses.
On entraîne un arbre h_m pour approximer ces pseudo-résidus au sens des moindres carrés :

$$h_m = \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} \left(r_{im} - h(\mathbf{x}_i) \right)^2$$

Dans un arbre de décision, l'espace est partitionné en T feuilles disjointes $R_1, \ldots, R_T$. Au lieu d'un pas global α_m, Friedman optimise une valeur γ_{jm} indépendante pour **chaque feuille** j :

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{jm}} L\Big(y_i, F_{m-1}(\mathbf{x}_i) + \gamma \Big)$$

Mise à jour : $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \sum_{j=1}^{T} \gamma_{jm} \mathbb{1}_{\mathbf{x} \in R_{jm}}$ *(ν = Shrinkage / Learning Rate).*

XGBoost : Boosting de Second Ordre

En 2014, **XGBoost** (Chen & Guestrin) accélère et stabilise la convergence du Boosting via une optimisation de **Newton** (développement de Taylor à l'ordre 2).
Notons le gradient $g_i = \partial_F L$ et le Hessien $h_i = \partial_F^2 L$ évalués en $F_{m-1}(\mathbf{x}_i)$. L'ajout d'un modèle additif $f_m(\mathbf{x})$ modifie le risque empirique approché :

Approximation de Taylor (Ordre 2)

$$\mathscr{L}^{(m)} \approx \sum_{i=1}^{n} \left[g_i f_m(\mathbf{x}_i) + \frac{1}{2} h_i f_m^2(\mathbf{x}_i) \right] + \Omega(f_m)$$

Où $\Omega(f_m)$ pénalise rigoureusement la topologie de l'arbre pour limiter la variance :

$$\Omega(f_m) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \quad (T = \text{nb feuilles, } w_j = \text{valeur de la feuille})$$

XGBoost : Solution Analytique des Feuilles

En injectant $f_m(\mathbf{x}) = \sum w_j \mathbb{1}_{\mathbf{x} \in R_j}$ et en regroupant la somme sur les observations par les T feuilles, l'objectif devient une somme de paraboles scalaires indépendantes (où I_j sont les données de la feuille j) :

$$\mathscr{L}^{(m)} \approx \sum_{j=1}^{T} \left[\underbrace{\left(\sum_{i \in I_j} g_i \right)}_{G_j} w_j + \frac{1}{2} \underbrace{\left(\sum_{i \in I_j} h_i + \lambda \right)}_{H_j + \lambda} w_j^2 \right] + \gamma T$$

En dérivant cette équation quadratique par rapport à w_j et en annulant, on obtient analytiquement **le poids optimal absolu** affecté à chaque feuille :

Poids optimal (Pas de Newton régularisé)

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

XGBoost : Split Finding

Comment l'algorithme choisit-il les caractéristiques pour construire l'arbre (le split) ?
En réinjectant $w_j^\star$ dans la perte approchée, on quantifie la "qualité" d'un nœud. Lors de l'évaluation d'une division d'un nœud Parent en fils Gauche (L) et Droit (R), la réduction du risque empirique (le **Gain**) s'écrit de manière fermée :

Score de Gain de séparation (Remplace Gini / MSE)

$$\text{Gain} = \frac{1}{2}\left[\underbrace{\frac{G_L^2}{H_L+\lambda}}_{\text{Fils Gauche}} + \underbrace{\frac{G_R^2}{H_R+\lambda}}_{\text{Fils Droit}} - \underbrace{\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}}_{\text{Nœud Parent}}\right] - \gamma$$

Implications pratiques de l'algorithme :

- C'est cette formulation exacte qui est calculée pour balayer les variables à chaque nœud.
- Si Gain < 0, la scission est rejetée. Le paramètre γ agit donc mathématiquement comme un algorithme strict d'**élagage préemptif** (pruning).

Optimisations algorithmiques de l'État de l'Art

Pour clore sur l'ingénierie, voici ce qui permet à ces mathématiques de s'exécuter sur des millions de données en quelques secondes :

- **Stochastic Gradient Boosting :** Tout comme les forêts aléatoires, les arbres sont entraînés sur des sous-échantillons aléatoires de lignes et colonnes pour faire chuter la corrélation ρ vue en section 1.
- **Histogrammes (LightGBM, HistGradientBoosting) :** Pour évaluer le Gain de tous les splits continus, les variables sont discrétisées en *bins* (ex : 256 niveaux). La complexité de recherche s'effondre de $\mathcal{O}(n \log n)$ à $\mathcal{O}(\text{bins})$.
- **Croissance Leaf-Wise (LightGBM) :** L'arbre ne grandit plus niveau par niveau, mais divise asymétriquement la feuille qui présente le plus grand Gain absolu, accélérant la minimisation de la perte.