Techniques avancées d'apprentissage

Etude autour du Boosting

Guillaume Vaudaux-Ruth
Vincent Le Meur

17 mai 2018

Table des matières

Introduction **2**

Contexte général du Boosting **3**

Etude de l'algorithme Adaboost **4**
Présentation de l'algorithme 4
Propriétés d'Adaboost 5
Optimisation 6

Variantes du Boosting **6**
Real AdaBoost 6
LogitBoost 7
GentleAdaboost 7

Conclusion **8**

Annexe : Listing des différentes figures **9**

Introduction

En apprentissage, le boosting désigne un ensemble de méthodes permettant de prendre des décisions précises à partir d'un ensemble de règles de décisions "faibles" (simplement meilleures que le hasard).

Ses champs d'application peuvent être l'estimation de densité, la régression ou la classification. Nous allons dans notre étude nous intéresser exclusivement au cas de la classification binaire.

Nous allons détailler le contexte général du boosting. Nous effectuerons un focus particulier sur l'une de ses variantes les plus populaires : l'algorithme Adaboost. Nous détaillerons le fonctionnement et les propriétés de cet algorithme et de plusieurs de ses variantes. Nous allons mettre en pratique ces algorithmes sur des données réelles et simulées.

Les données réelles sont issues d'un dataset de classification binaire disponible sur scikit-learn portant sur la détection du cancer du sein. Il contient 569 observations pour 30 variables. Nos données simulées quant à elles sont issues d'une loi normale multivariée (on obtient une matrice de dimension 1000×50). Le label est obtenu en prenant le signe du produit de cette matrice avec un vecteur aléatoire (de taille 50).

Contexte général du Boosting

Le Boosting correspond à une méthode d'"ensemble learning" qui va donc agréger plusieurs prédicteurs simples pour effectuer une prédiction.

Nous allons étudier ici le cas de la classification binaire en utilisant les notations suivantes :
- $X \in \mathbb{R}^{n \times d}$ pour les variables explicatives
- $y_1, .., y_n \in \{-1, 1\}$ pour les labels associés
- $L(a)$ pour la fonction de perte utilisée
- $G \subset F(X, \{-1, 1\})$ pour la famille de prédicteurs faibles que l'on cherche à agréger
- $A(h)$ le risque général associé à l'estimateur h et la fonction de coût L

Ce sous ensemble G n'est typiquement pas à un sous-ensemble convexe de l'espace vectoriel $F(X, \{-1, 1\})$. Pour convexifier cet ensemble on peut considérer la λ-enveloppe convexe de G définie par :

$$H_\lambda = \left\{ \sum_{j=1}^{m} \alpha_j g_j : m \in \mathbb{N}, \alpha \in \mathbb{R}_+^m, (g_1, .., g_m) \in G^m, \sum_{j=1}^{m} \alpha_j \leq \lambda \right\} \tag{1}$$

Dans ce cadre le problème du boosting général correspond à résoudre le problème suivant :

$$h_{boost}^{\lambda} \in argmin_{h \in H_\lambda} \frac{1}{n} \sum_{i=1}^{n} L(-y_i h(X_i)) \tag{2}$$

Il est intéressant de mentionner un résultat démontré par Lugosi & Vayatis (Annals of Statistics,2004). Ils montrent sous de bonnes conditions la consistance universelle de la méthode :

Théorème : Si $\lambda = \lambda_n \to \infty$ est tel que $\lambda_n L'(\lambda_n) \sqrt{\frac{lnn}{n}} \to \infty$ et l'ensemble G vérifie :
- $lim_{\lambda \to \infty} inf_{h \in H_\lambda} A(h) = inf_{h \in F(X, \mathbb{R})} A(h)$
- G a une VC-dimension finie

alors h_{boost}^{λ} est universellement consistant.

Même si le problème (2) est convexe, résoudre le problème reste difficile en raison du caractère infini-dimensionnel de l'espace des contraintes H_λ. Pour pallier ce défaut, un algorithme itératif, Adaboost a été introduit. En revanche il n'existe aucune garantie formelle que cet algorithme Adaboost converge vers la solution du problème.

Etude de l'algorithme Adaboost

Présentation de l'algorithme

L'algorithme Adaboost (pour Adaptative Boosting) repose sur trois idées fondamentales :

1. La prise de décision se fait d'après le vote d'un comité d'experts (classifieurs faibles G_m)
2. Le vote de chaque expert est pondéré par α_m en fonction de son taux d'erreur
3. Les données d'apprentissage sont pondérées de manière adaptative afin de surpondérer les exemples mal classés aux itérations précédentes.

Ainsi, à chacune des étapes $m = 1, ..., M$, on commence par entraîner G_m, un classifieur MRE, avec les données d'entraînement pondérées avec les poids w_i. Ensuite, on calcule l'erreur normalisée du classifieur G_m afin de calculer sa pondération α_m dans le classifieur final. Pour finir, on met à jour les poids w_i pour qu'à l'itération suivante, le nouveau classifieur faible se spécialise sur ces données difficiles à classifier. On renvoie alors le classifieur final G qui est l'agrégation des classifieurs G_m.

AdaBoost.M1.

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Afin de comprendre les performances d'Adaboost, il est nécessaire de le considérer comme équivalent à un *forward stagewise additive modeling* en utilisant la fonction de perte $L(y, f(x)) = \exp(-yf(x))$.

En effet, en utilisant la définition du L-risque empirique, on cherche :

$$(\alpha_m, G_m) = \arg\min_{\alpha,G} \sum_{i=1}^{N} e^{-y_i(f_{m-1}(x_i)+\alpha G(x_i))} \tag{3}$$

$$= \arg\min_{\alpha,G} \sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha G(x_i)} \tag{4}$$

$$= \arg\min_{\alpha,G} \sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha G(x_i)} [\mathbb{I}(y_i = G(x_i)) - \mathbb{I}(y_i \neq G(x_i)] \tag{5}$$

$$= \arg\min_{\alpha,G} (e^{-\alpha} - e^{\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq G(x_i)) + e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \tag{6}$$

En commençant par minimiser par rapport à G, le point minimum de l'expression (6) peut être obtenu en résolvant le problème

$$\widehat{G} \in \arg\min_{\alpha,G} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq G(x_i)) \tag{7}$$

En insérant le prédicteur $\widehat{G}$ dans l'équation (6), alors le point minimum $\widehat{\beta}$ est solution de

$$\widehat{\alpha} \in \arg\min_{\alpha,G} \{(e^{-\alpha} - e^{\alpha}) err_m + e^{-\alpha}\} \tag{8}$$

Et ainsi, la solution du problème (3) est donnée par :

$$\widehat{G} \in \arg\min_{\alpha,G} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq G(x_i)) \tag{9}$$

$$\widehat{\alpha} = \frac{1}{2} \log(\frac{1 - err_m}{err_m}) \qquad avec \;\; err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq G(x_i))}{\sum_{i=1}^{N} w_i^{(m)}} \tag{10}$$

AdaBoost peut donc être interprété comme une procédure d'estimation pas à pas pour "fitter" un modèle de régression logistique additif. Nous représentons en annexe (1,2) les courbes obtenues sur nos données réelles et simulées pour cet algorithme. Nous obtenons de bonnes performances pour les données du cancer du sein (avec 0,04% d'erreur). Le problème simulé étant plus complexe (plus de données et dimensionalité plus importante) nous arrivons à un taux d'erreur de l'ordre de 20%. Sur les deux jeux de données l'erreur d'entraînement chute rapidement et l'erreur de généralisation se stabilise aux alentours de 200 estimateurs pour le dataset simulé.

Propriétés d'Adaboost

En théorie de l'apprentissage, en posant N le nombre d'exemples d'apprentissage, d_H la dimension de Vapnik-Chervonenkis et m le nombre d'étapes de boosting, on a :

$$R_{Reel} \leq R_{apparent} + O(\sqrt{\frac{m.d_H}{N}}) \tag{11}$$

D'après cette borne, lorsque m devient grand le boosting devrait avoir tendance à sur-apprendre. Empiriquement, il a été observé que cela ne se produit pas et donc que cette borne ne s'applique pas.

Ce résultat s'explique en fait par la théorie des marges. En définissant la marge d'un exemple (x, y) :

$$marge(x, y) = \frac{y \sum_m \alpha_m G_m(x)}{\sum_m \alpha_m} \tag{12}$$

La marge est donc positive si G classifie correctement et négative sinon. Elle peut être interpretée comme une mesure de confiance dans la prédiction. Par intuition, de larges marges devraient permettre une meilleure borne sur l'erreur réele (ce que ne prend pas en compte l'inégalité précédente). Il a été montré par Schapire, Freund, Bartlett & Lee que l'erreur réelle peut être bornée avec grande probabilité, pour tout $\theta > 0$,

$$R_{Reel} \leq P[marge(x, y) \leq \theta] + O(\sqrt{\frac{d_H}{N\theta^2}}) \tag{13}$$

En majorant la probabilité, on montre que cette borne décroit exponentiellement vite avec le nombre d'étapes de boosting. En effet, en se concentrant sur des exemples difficiles à classer, c'est à dire sur des exemples avec une faible marge, le boosting va chercher à l'augmenter.

Optimisation

Ici nous allons optimiser notre algorithme en introduisant un nouveau paramètre : le "shrinkage". Si nous reprenons l'algorithme Adaboost détaillé précédemment ce paramètre qu'on notera $\nu \in [0, 1]$ va s'insérer dans l'étape (c) de la manière suivante :

$$(c) \qquad \alpha_m = \log(\frac{1 - err_m}{err_m})^{\nu} \tag{14}$$

Cela permet de régler certains cas où Adaboost "overfite" les données. En effet cela "lisse" la pondération des données. Nous observons en annexe (4) l'influence de ce paramètre sur des données simulées. Nous observons bien qu'en diminuant ce paramètre notre algorithme va converger plus lentement vers une erreur plus faible.

Variantes du Boosting

Real AdaBoost

Real AdaBoost peut être présenté comme une version généralisée d'AdaBoost. Dans ce modèle, le prédicteur faible ne va plus retourner la classe d'appartenance mais une estimation de la probabilité d'appartenance à la classe $p_m(x) = \widehat{\mathbb{P}}_w(y = 1|x) \in [0, 1]$. La contribution au classifieur final est la moitié du logit de cette probabilité estimée.

Real AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$ using weights w_i on the training data.
   (b) Set $f_m(x) \leftarrow \frac{1}{2} \log \frac{p_m(x)}{1-p_m(x)} \in R$.
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} f_m(x)]$

LogitBoost

Le LogitBoost utilise quant à lui un autre critère de perte. On montre que la perte "log-vraisemblance binomiale négative" utilisée ici (ou *deviance*) $L(y, f(x)) = -\log(1 + e^{-2yf(x)})$, a la même population de minimiseurs que la perte exponentielle. Le LogitBoost utilise une procédure d'estimation par méthode de Newton pour "fiter" un modèle logistique additif par maximum de vraisemblance :

LogitBoost (2 classes)

1. Start with weights $w_i = 1/N$ $i = 1, 2, \ldots, N$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Compute the working response and weights

$$\begin{aligned} z_i &= \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))} \\ w_i &= p(x_i)(1 - p(x_i)) \end{aligned}$$

   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of z_i to x_i using weights w_i.
   (c) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x)$ via (28).
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

GentleAdaboost

Pour finir, le GentleBoost est un algorithme dérivé du RealAdaBoost. Au lieu d'optimiser exactement le risque associé à la perte exponentielle à chaque itération, le GentleAdaboost va approximer par la méthode de Newton comme le fait le LogitBoost. Ainsi, on montre que la principale différence avec le RealAdaboost est la manière dont sont utilisés les poids des classes de probabilité. Le GentleBoost utilise le classifieur : $f_m(x) = \mathbb{P}_w(y = 1|x) - \mathbb{P}_w(y = -1|x)$:

Gentle AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weights w_i.
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$
   (c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ and renormalize.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

Motivations

La performance des 4 algorithmes sur les données simulées est présentée dans la figure 5. Empiriquement, on montre que les algorithmes ont des performances similaires. Leur particularité réside dans leur méthode d'actualisation. Les LogitBoost et Gentle AdaBoost ont de meilleurs résultats sur des données présentant du bruit ou des outliers. La classification de ces données étant plus compliquée, les classifieurs vont se tromper plus souvent et donc attribuer un poids élevé à ces données. En se référant à la figure (3), on voit que la fonction de perte du LogitBoost pondère plus modérément les données mal classifiées, comme le fait aussi Gentle AdaBoost avec son approximation par méthode de Newton du risque associé à la perte exponentielle.

Gentle AdaBoost et LogitBoost peuvent donc sur-performer Adaboost lorsqu'il présente des problèmes de stabilité.

Conclusion

Nous avons ici réalisé une étude autour du Boosting. Après avoir fait un descriptif général de cette notion nous avons étudié en détail l'un de ses algorithmes les plus populaires : Adaboost. Nous avons discuté de ses différentes propriétés et étudié plusieurs de ses variantes dont nous avons illustré les performances sur des données réelles et simulées.

Comme souvent en Machine Learning, l'efficacité d'un algorithme dépend des données sur lesquelles on l'applique. Ici, il semble que l'algorithme Gentle Adaboost soit plus adapté à nos données.

Annexe : Listing des différentes figures

FIGURE 1: Adaboost sur données simulées

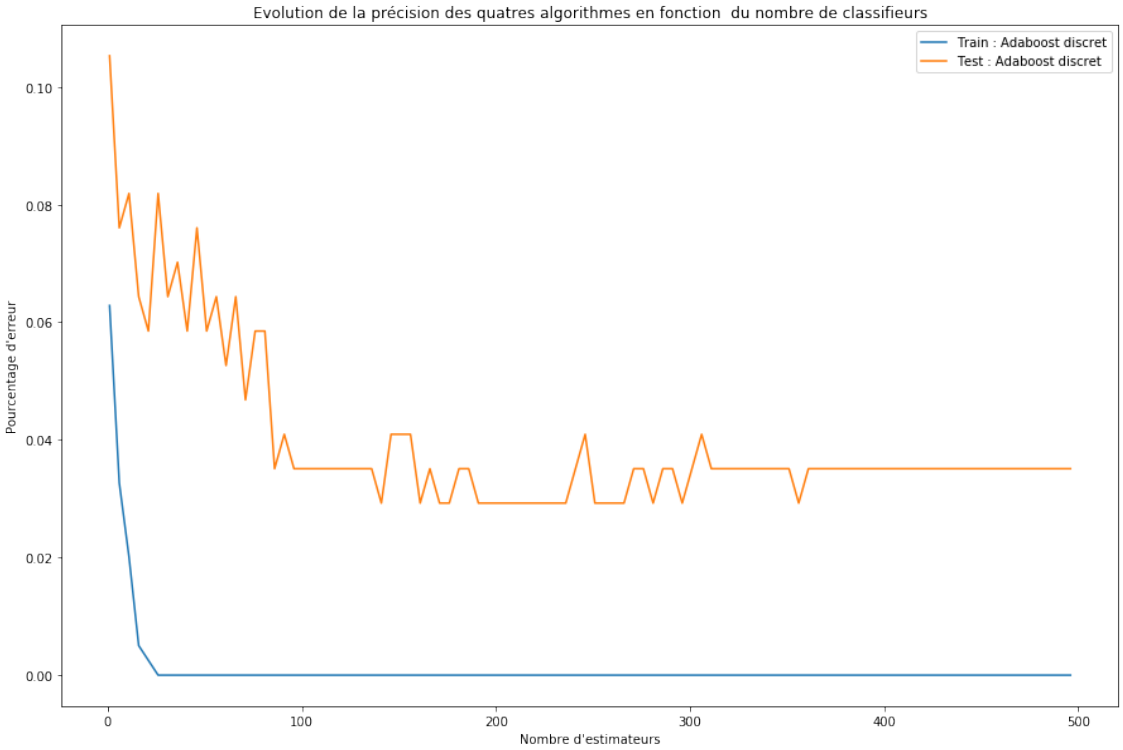

FIGURE 2: Adaboost sur données réelles

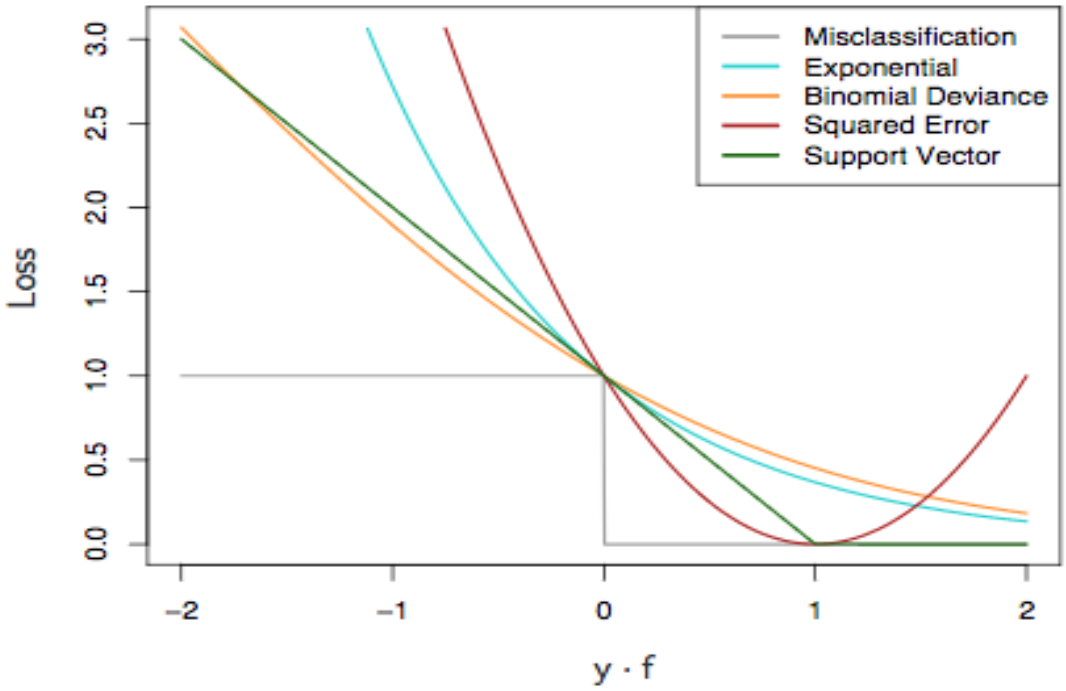

FIGURE 3: Différentes fonctions de pertes

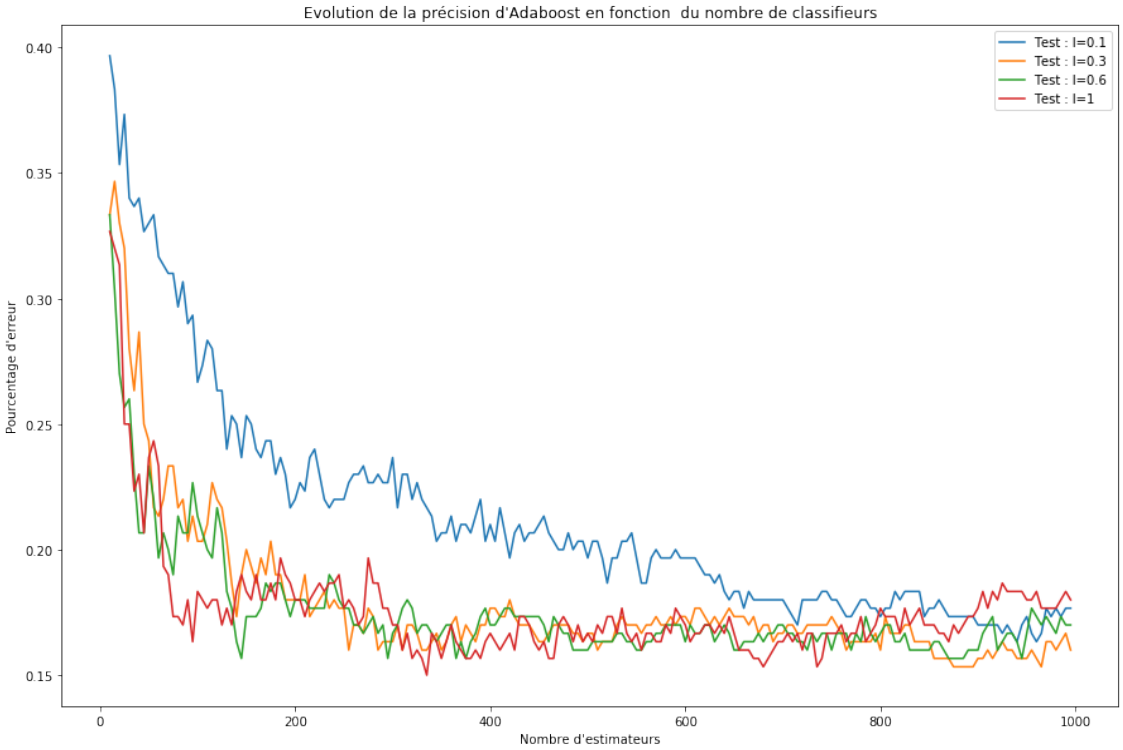

FIGURE 4: Influence du Shrinkage

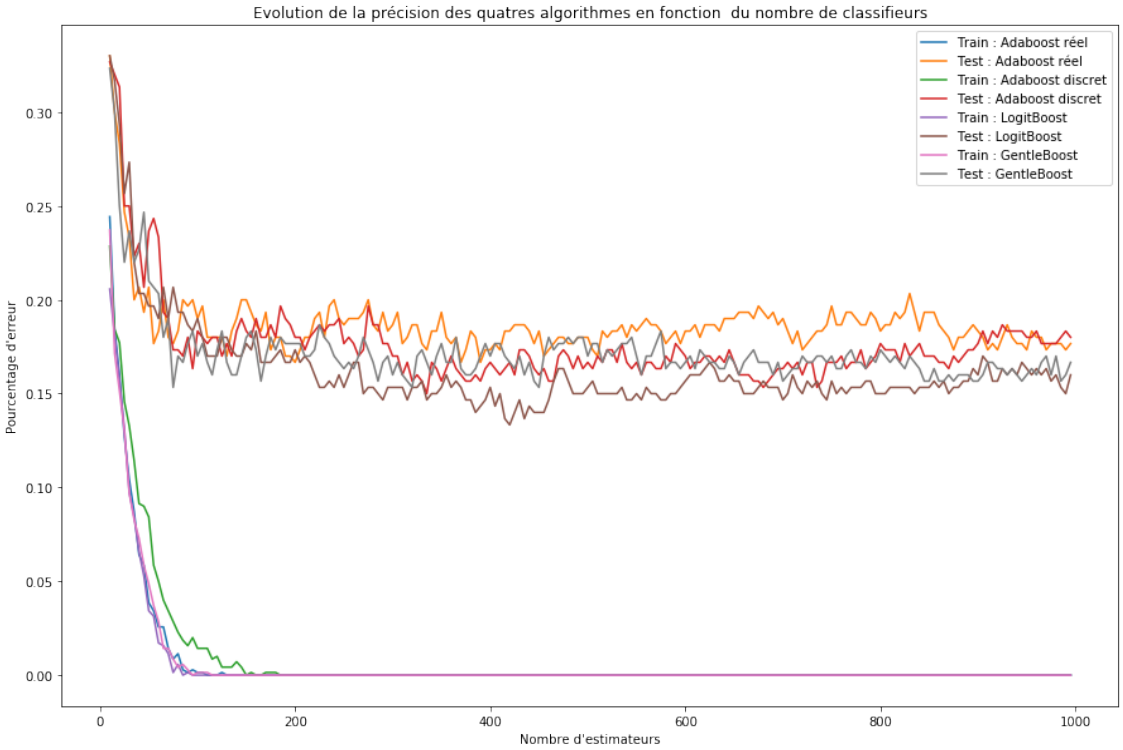

FIGURE 5: Comparaison des quatres algorithmes