# AdaBoost analytique

Olivier LAFITTE[1] Jean-Marc BROSSIER[2]

[1] Université Sorbonne Paris Nord, LAGA, UMR 7539. IRL CNRS-CRM 3457. Université de Montréal. Canada

[2] CNRS, Univ. Grenoble Alpes, Grenoble-INP, GIPSA-lab, Grenoble, France

**Résumé –** AdaBoost est un algorithme qui, à chaque étape, renvoie un classifieur faible en appelant itérativement un WeakLearner, calcule le coefficient de ce nouveau classifieur et renvoie de nouveaux poids pour les exemples de l'ensemble d'entraînement.
Cette étude apporte un éclairage nouveau sur la procédure AdaBoost dans le cas de deux classes, en prouvant que tous les coefficients et poids mentionnés ci-dessus sont donnés par des formules explicites.
Pour ce faire, nous utilisons une analyse logique qui construit une table de vérité qui repose sur l'ensemble d'entraînement et l'ensemble des classifieurs faibles. Cette table de vérité est une partition de l'ensemble d'entraînement qui fournit des informations sur la décision des classifieurs sur chaque sous-ensemble et permet de réécrire le risque associé.

**Abstract –** AdaBoost is an algorithm which, at each step, returns a weak classifier by calling iteratively a WeakLearner, calculates the coefficient of this new classifier and returns new weights for examples in the training set.
This study sets a new light on the AdaBoost procedure in the case of two classes, by proving that all the weights mentioned above are given by explicit formulae.
To do this, we use a logical analysis that constructs a truth table based on the training set and the set of weak classifiers. This truth table is a partition of the training set which provides the information about the decision of the classifiers on each subset and allows to rewrite the associated risk.

## 1 Position par rapport à l'état de l'art

De nombreux algorithmes utilisent des classifieurs faibles pour construire un classifieur fort qui améliore la précision et la robustesse en tirant parti de la diversité des classifieurs faibles. Les deux principales catégories sont le Bagging et le Boosting.

Le Bagging (Bootstrap Aggregating) construit plusieurs instances du même classifieur sur des échantillons bootstrap de l'ensemble d'entraînement et les combine. L. Breiman a introduit le Bagging (Bagging Predictors) [1], les forêts aléatoires (Random Forest)[2] en sont un exemple célèbre.

Le boosting combine plusieurs classifieurs faibles de manière séquentielle, chaque nouveau classifieur se concentrant sur les erreurs des précédents. La méthode la plus connue et la première mise en œuvre pratique est AdaBoost (Adaptive Boosting), proposée par [4].

Cet article se concentre sur le Boosting, voir le livre de référence sur les techniques de boosting [7], et plus particulièrement sur AdaBoost qui a la réputation d'être un parfait exemple de cette approche.

Considérons un WeakLearner, qui renvoie, pour chaque ensemble d'entraînement composés d'exemples pondérés, un classifieur faible.

L'algorithme AdaBoost appelle itérativement le WeakLearner pour déduire un nouveau classifieur faible, puis calcule de nouveaux poids pour les exemples ainsi que le coefficient de ce classifieur faible supplémentaire dans la combinaison linéaire qui est le classifieur résultant.

Le rôle principal de l'algorithme AdaBoost est donc de construire des classifieurs faibles successifs.

Ce que nous proposons ici : une fois $k$ classifieurs construits, tous les coefficients et poids sont exprimés analytiquement à partir des cardinaux de la partition des exemples à l'aide de ces $k$ classifieurs.

Cette démarche ne se limite pas à l'algorithme AdaBoost, elle s'applique à n'importe quelle famille de $p$ classifieurs, indépendamment de la manière dont ils sont obtenus.

Bien que de nombreuses études aient été consacrées à AdaBoost (par exemple [5, 3, 9, 6]), le présent document est le premier, à notre connaissance, à donner des formules analytiques pour tous les poids des exemples et tous les coefficients des classifieurs retournés par l'algorithme Adaboost.

## 2 Introduction

Nous présentons l'approche, qui ne semble pas avoir été utilisée auparavant à notre connaissance, que nous adoptons pour AdaBoost en utilisant une analyse logique de l'ensemble d'entraînement basée sur un ensemble de classifieurs faibles. L'intérêt de cette étude est principalement théorique. Les principales contributions de cette étude sont les suivantes :

- définir la table de vérité pour un ensemble d'entraînement $\mathcal{S}$ et une liste non ordonnée de $p$ classifieurs. La table de vérité est une partition de $\mathcal{S}$ en $2^p$ sous-ensembles, appelés classes d'exemples dans la suite. La table de vérité encode l'ensemble des informations logiques sur le comportement de la liste des classifieurs sur l'ensemble d'entraînement $\mathcal{S}$.
- Une fois les tables de vérité déterminées, déduire des formules analytiques pour les coefficients $\beta_k$ de chaque classifieur $G_k$ dans le classifieur résultant $h = \text{sign}\left(\sum \beta_j G_j\right)$ ainsi que des formules analytiques pour les poids de chaque classe d'exemples dans la table de vérité. Tous les poids ne dépendent que des cardinaux des sous-ensembles.

- Donner une formule analytique pour le classifieur résultant et pour les poids des exemples dans le cas $p = 3$ et des formules d'induction dans le cas général de $p > 3$ (pour lequel une formule analytique explicite pourrait également être obtenue, mais ce n'est pas le but du présent document).

Nous prouvons que cette procédure correspond à la première étape de l'algorithme de descente par coordonnées pour le risque exponentiellement convexe.

Nous présentons les résultats sous la forme d'un algorithme dont on peut facilement vérifier qu'il produit les mêmes résultats numériques que l'algorithme AdaBoost.

Cette contribution est organisée comme suit.

La section 3 expose le problème et le principe de l'analyse logique de l'ensemble d'entraînement à l'aide d'un ensemble de classifieurs. Une illustration élémentaire est donnée dans le cas de trois classifieurs.

La section 4 étend cette analyse à $p > 3$ classifieurs et définit les notations générales.

La section 5 applique cette décomposition de l'ensemble d'entraînement à la déduction de formules pour les coefficients des classifieurs.

La section 6 effectue le calcul explicite exact effectué par l'algorithme AdaBoost pour le cas de 3 classifieurs (qui est le premier cas de combinaison non trivial).

La section 7 résume les résultats en tant qu'algorithme et souligne que la validité des formules obtenues peut être facilement vérifiée numériquement.

# 3 Définition du problème

Soit $\mathcal{S} = \{(x_i, y_i)\}_{i=1..n} \subset \mathcal{X} \times \mathcal{Y}$ un ensemble d'entraînement, où $\mathcal{X} = \mathbb{R}^d$ est l'espace de caractéristiques et $\mathcal{Y} = \{-1, +1\}$ celui des étiquettes pour deux classes. Identifier un classifieur revient à identifier une fonction $h : \mathcal{X} \to \mathcal{Y}$ qui associe chaque caractéristique $x_i$ à son étiquette $y_i$ avec le moins d'erreurs possible.

Dans cet article, ce classifieur $h$ est obtenu en combinant linéairement $p$ classifieurs faibles $G_k : \mathcal{X} \to \mathcal{Y}$.

Par exemple, pour $p = 3$ classifieurs faibles $\mathbf{G} = (G_1, G_2, G_3)$, nous introduisons leurs coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3) \in \mathbb{R}^3$ de sorte que le classifieur résultant est donné par $h = \text{sign}\,(\boldsymbol{\beta} \cdot \mathbf{G})$ avec $\boldsymbol{\beta} \cdot \mathbf{G} = \sum \beta_j G_j$.

Étant donné que $(y_i, G_k(x_i)) \in \mathcal{Y}^2$, pour tout $k \in [\![1; p]\!]$ et $i \in [\![1; n]\!]$, le produit $y_i G_k(x_i)$ et soit égal à $+1$ si $y_i$ et $G_k(x_i)$ sont de même signe, c'est-à-dire $G_k(x_i)$ est vrai ($G_k(x_i) = y_i$), ou égal à $-1$ si $y_i$ et $G_k(x_i)$ ne sont pas du même signe, c'est-à-dire $G_k(x_i)$ est faux ($G_k(x_i) \neq y_i$).

Pour une liste de $p$ classifieurs faibles, cela conduit à $2^p$ configurations possibles et à une partition en $2^p$ sous-ensembles de $\{x_i, i = 1 \cdots n\}$.

Nous créons ainsi une table de vérité composée de $p$ lignes (une ligne pour chaque classifieur) et $2^p$ colonnes (une colonne pour chaque configuration possible des classifieurs) englobant toutes les valeurs de $y_i G_k(x_i)$.

Pour $p = 3$, nous avons la table de vérité suivante (en étiquetant le signe de $y_i G_k(x_i)$ par $G_k$ pour des raisons de simplicité et en désignant par $\sharp$ le cardinal de chaque configuration dans l'ensemble d'entraînement $\mathcal{S}$) :

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $G_1$ | -1 | 1 | 1 | -1 | -1 | 1 | -1 | 1 |
| $G_2$ | -1 | 1 | -1 | 1 | 1 | -1 | -1 | 1 |
| $G_3$ | -1 | 1 | -1 | 1 | -1 | 1 | 1 | -1 |
| $\sharp$ | $c_8$ | $c_{15}$ | $c_{12}$ | $c_{11}$ | $c_{10}$ | $c_{13}$ | $c_9$ | $c_{14}$ |

# 4 Structuration de l'ensemble d'entraînement pour $p$ classifieurs

Cette structure peut être généralisée à $p$ classifieurs :
$\rightarrow$ pour un classifieur (On note $\bigsqcup$ l'union disjointe.), on a :
$\mathcal{S} = \mathcal{S}_1 = \mathcal{S}_2 \bigsqcup \mathcal{S}_3$, $\mathcal{S}_2 = \{(x_i, y_i) : y_i G_1(x_i) = -1\}$, $\mathcal{S}_3 = \{(x_i, y_i) : y_i G_1(x_i) = +1\}$,
$\rightarrow$ pour deux classifieurs :
$\mathcal{S}_1 = \mathcal{S}_2 \bigsqcup \mathcal{S}_3 = (\mathcal{S}_4 \bigsqcup \mathcal{S}_5) \bigsqcup (\mathcal{S}_6 \bigsqcup \mathcal{S}_7)$ où
$\mathcal{S}_4 = \{(x_i, y_i) : y_i G_1(x_i) = -1, y_i G_2(x_i) = -1\}$,
$\mathcal{S}_5 = \{(x_i, y_i) : y_i G_1(x_i) = -1, y_i G_2(x_i) = +1\}$,
$\mathcal{S}_6 = \{(x_i, y_i) : y_i G_1(x_i) = +1, y_i G_2(x_i) = -1\}$,
$\mathcal{S}_7 = \{(x_i, y_i) : y_i G_1(x_i) = +1, y_i G_2(x_i) = +1\}$ :
on a ainsi divisé $S_2$ en $S_4$ et $S_5$ selon la validité de la prédiction de $G_2$, de même on divise $S_3$ en $S_6$ et $S_7$,
$\rightarrow$ et plus généralement pour $k$ classifieurs :
$\mathcal{S}_l = \mathcal{S}_{2l} \bigsqcup \mathcal{S}_{2l+1}$, pour $l \in [\![2^k; 2^{k+1} - 1]\!]$ où
$\mathcal{S}_{2l} = \mathcal{S}_l \cap \{(x_i, y_i) : y_i G_k(x_i) = -1\}$.

Définissons les coefficients $c_l = \sharp \mathcal{S}_l$, $c_1 = n$. Cela définit une structure arborescente de sous-ensembles disjoints de $\mathcal{S} = \mathcal{S}_1$ telle que,

$\forall k < p$, $\mathcal{S} = \bigsqcup_{j=2^k}^{2^{k+1}-1} \mathcal{S}_j$ et $n = \sum_{j=2^k}^{2^{k+1}-1} c_j$. C'est la numération Sosa-Stradonitz d'un arbre généalogique [8].

| | $c_2$ | $c_3$ |
|---|---|---|
| $G_1$ | -1 | 1 |

| | $c_4$ | $c_5$ | $c_6$ | $c_7$ |
|---|---|---|---|---|
| $G_1$ | -1 | | 1 | |
| $G_2$ | -1 | 1 | -1 | 1 |

| | $c_8$ | $c_9$ | $c_{10}$ | $c_{11}$ | $c_{12}$ | $c_{13}$ | $c_{14}$ | $c_{15}$ |
|---|---|---|---|---|---|---|---|---|
| $G_1$ | -1 | | | | 1 | | | |
| $G_2$ | -1 | | 1 | | -1 | | 1 | |
| $G_3$ | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 |

Pour chaque $c_j$, $j \in [\![2^{k-1}; 2^k - 1]\!]$, nous construisons $\boldsymbol{\epsilon}(j) \in \mathcal{Y}^{k-1}$ qui retrace la généalogie de $c_j$ grâce à la table de vérité au niveau $(k-1)$. Par exemple, on a $\boldsymbol{\epsilon}(5) = (-1, 1)$ et $\boldsymbol{\epsilon}(13) = (1, -1, 1)$.

# 5 Calcul des coefficients des classifieurs

Une façon de déterminer les coefficients $\boldsymbol{\beta}$ nécessaires à la construction du classifieur résultant $\text{sign}(\sum \beta_j G_j) = \text{sign}\,(\boldsymbol{\beta} \cdot \mathbf{G})$ est de définir le risque empirique convexifié

$$\mathcal{R}(\boldsymbol{\beta}, \mathcal{S}) = \sum_{i=1}^{n} \exp\left(-y_i \boldsymbol{\beta} \cdot \mathbf{G}(x_i)\right)$$

de sorte qu'un calcul de minimisation de ce risque conduit à la solution optimale, si elle existe.

Nous utilisons la méthode de descente par coordonnées en considérant successivement

$$\begin{aligned}\beta_1 &= \arg\min_\beta \mathcal{R}\left((\beta, 0, \cdots, 0), \mathcal{S}\right),\\ \beta_2 &= \arg\min_\beta \mathcal{R}\left((\beta_1, \beta, 0, \cdots, 0), \mathcal{S}\right),\\ &\vdots\\ \beta_k &= \arg\min_\beta \mathcal{R}\left((\boldsymbol{\beta}^{(k-1)}, \beta, 0, \cdots, 0), \mathcal{S}\right).\end{aligned}$$

et $\boldsymbol{\beta}^{(k)} = (\boldsymbol{\beta}^{(k-1)}, \beta_k)$. Comme cet algorithme a exactement $k$ étapes, il ne peut pas donner les coefficients optimaux pour $G_1, \cdots G_k$ contrairement à l'algorithme de relaxation.

Le risque $\mathcal{R}\left(\boldsymbol{\beta}^{(k-1)}, \mathcal{S}\right)$ à $\boldsymbol{\beta}^{(k-1)}$ est donc

$$\sum_{i=1}^{n} e^{-y_i \boldsymbol{\beta}^{(k-1)} \cdot \mathbf{G}(x_i)} = \sum_{j=2^{k-1}}^{2^k-1} c_j e^{-\boldsymbol{\epsilon}(j)\cdot\boldsymbol{\beta}^{(k-1)}}.$$

Tous les exemples dans $\mathcal{S}_j$ ont pour poids $e^{-\boldsymbol{\epsilon}(j)\cdot\boldsymbol{\beta}^{(k-1)}}$.

L'ajout du classifieur $G_k$ à la liste partitionne chaque $\mathcal{S}_j, j \in [\![2^{k-1}; 2^k - 1]\!]$ en $\mathcal{S}_{2j}$ et $\mathcal{S}_{2j+1}$ et $c_j = c_{2j} + c_{2j+1}$. Avec un coefficients $\beta$ pour le classifieur $G_k$, le risque devient

$$\sum_{j=2^{k-1}}^{2^k-1} \left(c_{2j} e^{\beta} + c_{2j+1} e^{-\beta}\right) e^{-\boldsymbol{\epsilon}(j).\boldsymbol{\beta}^{(k-1)}}. \tag{1}$$

Pour $(\sum_{j=2^{k-1}}^{2^k-1} c_{2j})(\sum_{j=2^{k-1}}^{2^k-1} c_{2j+1}) > 0$, le point de minimum de cette fonction est

$$\beta_k = \frac{1}{2} \ln \frac{\sum_{j=2^{k-1}}^{2^k-1} c_{2j+1} e^{-\boldsymbol{\epsilon}(j).\boldsymbol{\beta}^{(k-1)}}}{\sum_{j=2^{k-1}}^{2^k-1} c_{2j} e^{-\boldsymbol{\epsilon}(j).\boldsymbol{\beta}^{(k-1)}}}. \tag{2}$$

Cela donne le coefficient du classifieur $G_k$.

Notons $\tau_k = e^{\beta_k}$, la valeur du minimum est $\sum_{j=2^{k-1}}^{2^k-1} \left(c_{2j}\tau_k + c_{2j+1}\tau_k^{-1}\right) e^{-\boldsymbol{\epsilon}(j).\boldsymbol{\beta}^{(k-1)}}$. Cela définit $\tau_k$ pour tout $k \leq p$.

Observons que $e^{-\boldsymbol{\epsilon}(j).\boldsymbol{\beta}^{(k-1)}} = \prod_{p=1}^{k-1} \tau_p^{-\boldsymbol{\epsilon}(j)_p} := \boldsymbol{\tau}_{k-1}^{-\boldsymbol{\epsilon}(j)}$.

Cela prouve par induction que le poids de tous les exemples dans $\mathcal{S}_j, j \in [\![2^{k-1}; 2^k - 1]\!]$ est $\boldsymbol{\tau}_{k-1}^{-\boldsymbol{\epsilon}(j)}$ et que, pour $k \leq p$

$$\tau_k^2 = \frac{\sum_{j=2^{k-1}}^{2^k-1} c_{2j+1} \boldsymbol{\tau}_{k-1}^{-\boldsymbol{\epsilon}(j)}}{\sum_{j=2^{k-1}}^{2^k-1} c_{2j} \boldsymbol{\tau}_{k-1}^{-\boldsymbol{\epsilon}(j)}} = \frac{\sum_{j=2^{k-1}}^{2^k-1} c_{2j+1} \boldsymbol{\tau}_{k-1}^{\mathbf{1}-\boldsymbol{\epsilon}(j)}}{\sum_{j=2^{k-1}}^{2^k-1} c_{2j} \boldsymbol{\tau}_{k-1}^{\mathbf{1}-\boldsymbol{\epsilon}(j)}}. \tag{3}$$

La deuxième égalité s'obtient en multipliant le numérateur et le dénominateur par $\boldsymbol{\tau}_{k-1} = \tau_1 \cdots \tau_{k-1}$. Comme tous les éléments de $\mathbf{1} - \boldsymbol{\epsilon}(j)$ sont 0 ou 2, $\boldsymbol{\tau}_{k-1}^{\mathbf{1}-\boldsymbol{\epsilon}(j)}$ est rationnel et par induction $\tau_k^2$ est rationnel.

L'ajout d'un classifieur $G_{p+1}$ à l'ensemble $\{G_1, \cdots, G_p\}$ divise $\mathcal{S}_j$ en $\mathcal{S}_{2j}$ et $\mathcal{S}_{2j+1}$ comme précédemment et multiplie les poids des exemples dans $\mathcal{S}_{2j}$ par $\tau_{p+1}^{-1}$ et les poids des exemples dans $\mathcal{S}_{2j+1}$ par $\tau_{p+1}$.

De manière incrémentale, nous calculons le coefficient $\beta_k$ à l'étape $k \leq p$ en utilisant uniquement les tables de vérité et les coefficients $\boldsymbol{\beta}^{(k-1)} = (\beta_1, \ldots, \beta_{k-1})$ obtenus lors des étapes précédentes. La construction de la ligne suivante de la table de vérité nécessite la nouvelle pondération des exemples et l'exécution du WeakLearner choisi.

La table de vérité finale à l'étape $p$, illustrée dans l'introduction dans le cas $p = 3$, est construite de manière incrémentale à partir des tables d'ordre inférieur, ce qui implique la construction de tables de vérité à chaque étape : à l'étape $k$, le coefficient $\beta_k$ du classifieur $G_k$ est calculé en utilisant la table de vérité à l'étape $k-1$ et l'action du classifieur $G_k$ renvoyée par le WeakLearner.

# 6 Le cas particulier de trois classifieurs faibles

Les calculs explicites dans le cas $p = 3$ sont les suivants : en utilisant la formule 3, on trouve

$\tau_1^2 = \frac{c_3}{c_2}$, en utilisant à nouveau 3, on trouve

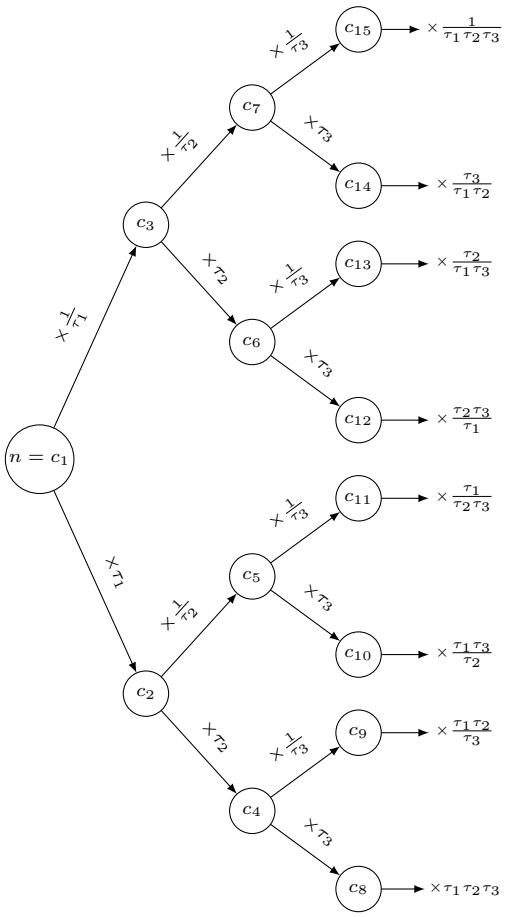
$\tau_2^2 = \frac{c_5\tau_1 + c_7/\tau_1}{c_4\tau_1 + c_6/\tau_1} = \frac{c_5\tau_1^2 + c_7}{c_4\tau_1^2 + c_6}$, et en utilisant à nouveau 3

$\tau_3^2 = \frac{c_9\tau_1\tau_2 + c_{11}\tau_1/\tau_2 + c_{13}\tau_2/\tau_1 + c_{15}/(\tau_1\tau_2)}{c_8\tau_1\tau_2 + c_{10}\tau_1/\tau_2 + c_{12}\tau_2/\tau_1 + c_{14}/(\tau_1\tau_2)}$

$= \frac{c_9\tau_1^2\tau_2^2 + c_{11}\tau_1^2 + c_{13}\tau_2^2 + c_{15}}{c_8\tau_1^2\tau_2^2 + c_{10}\tau_1^2 + c_{12}\tau_2^2 + c_{14}}$.

Notons que $\tau_1^2$, $\tau_2^2$, $\tau_3^2$ sont complètement connus par $c_8, \cdots, c_{15}$ en utilisant $c_2 = c_8 + c_9 + c_{10} + c_{11}$, $c_3 = c_{12} + c_{13} + c_{14} + c_{15}$, $c_4 = c_8 + c_9$, $c_5 = c_{10} + c_{11}$, $c_6 = c_{12} + c_{13}$, $c_7 = c_{14} + c_{15}$.

La construction de $\beta_k$ est résumée dans l'arborescence :




Par conséquent, nous avons calculé les valeurs de tous les coefficients $(\beta_1, \beta_2, \beta_3) \in \mathbb{R}^3$, et nous déduisons directement le classifieur résultant renvoyé par l'algorithme AdaBoost.

Désignons par WL($list$) l'action du WeakLearner sur la liste d'exemples caractérisée par la partition et les poids.

Nous construisons la partition de $\mathcal{S}_1$ en $2^p$ sous-ensembles d'exemples à l'étape $p$, d'où nous déduisons la liste $\{(\mathcal{S}_j)_{2^p \leq j \leq 2^{p+1}-1}, \boldsymbol{\tau}_p^{-\boldsymbol{\epsilon}(j)}\}$, chaque élément de $\mathcal{S}_j$ est pondéré par un poids construit à partir de $\tau_1, ...\tau_p$ comme suit :

WL($\{\mathcal{S}_1, 1\}$)

$$\rightarrow \{\mathcal{S}_2, \mathcal{S}_3; \tau_1, 1/\tau_1\}$$
$$\mathrm{WL}(\{\mathcal{S}_2, \mathcal{S}_3; \tau_1, 1/\tau_1\})$$
$$\rightarrow \{\mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7; \tau_1\tau_2, \tau_1/\tau_2, \tau_2/\tau_1, 1/(\tau_1\tau_2)\}$$
$$\mathrm{WL}(\{\mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7; \tau_1\tau_2, \tau_1/\tau_2, \tau_2/\tau_1, 1/(\tau_1\tau_2)\})$$
$$\rightarrow \{\mathcal{S}_8, \cdots, \mathcal{S}_{15}, \boldsymbol{\tau}_3^{-\boldsymbol{\epsilon}(l)}, l = 8 \cdots 15\}.$$

Ces opérations correspondent aux opérations suivantes sur l'ensemble d'entraînement :

1. remplacer $\mathcal{S}_1$ par l'union de $\mathcal{S}_2$ (où chaque exemple est remplacé par $c_3$ copies identiques) et $\mathcal{S}_3$ (où chaque exemple est remplacé par $c_2$ copies identiques), notées $\mathcal{S}_2^{\tau_1}, \mathcal{S}_3^{\frac{1}{\tau_1}}$ ; l'union est un ensemble d'entraînement abstrait sur lequel on exécute l'algorithme WL.

2. remplacer $\mathcal{S}_2^{\tau_1}$ par $\mathcal{S}_4^{\tau_1\tau_2} \cup \mathcal{S}_5^{\tau_1/\tau_2}$ et $\mathcal{S}_3^{1/\tau_1}$ par $\mathcal{S}_6^{\tau_2/\tau_1} \cup \mathcal{S}_7^{1/(\tau_1\tau_2)}$, ce qui correspond à remplacer chacun des éléments de $\mathcal{S}_4$ par $c_3(c_3c_5 + c_2c_7)$ copies identiques et ceux de $\mathcal{S}_5$ par $c_3(c_3c_4 + c_2c_6)$ copies identiques, puis remplacer chaque ensemble par l'union de ces copies.

# 7 Algorithme

Cette procédure logique peut être écrite sous la forme de l'algorithme 1

**Algorithme 1 :** Analytical AdaBoost

**Input :**
$\mathcal{S}_1 = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i \in [\![1, n]\!]\}$, l'ensemble d'entraînement,
$p \leftarrow$ le nombre d'itérations,

$\tau_0 = 1$,
$L_0 = \{\mathcal{S}_1; \tau_0\}$

**for** $k \in [\![1, p]\!]$ **do**
Entraîner un classifieur $G_k = \mathrm{WL}(L_{k-1})$
à la partition $\{\mathcal{S}_{2^{k-1}}, \cdots, \mathcal{S}_{2^k-1}\}$, chaque ensemble $\mathcal{S}_l$ étant pondéré par $\boldsymbol{\tau}_{k-1}^{-\boldsymbol{\epsilon}(l)}$, alors

$$\tau_k^2 \leftarrow \frac{\sum_{j=2^{k-1}}^{2^k-1} c_{2j+1} \boldsymbol{\tau}_{k-1}^{\mathbf{1}-\boldsymbol{\epsilon}(j)}}{\sum_{j=2^{k-1}}^{2^k-1} c_{2j} \boldsymbol{\tau}_{k-1}^{\mathbf{1}-\boldsymbol{\epsilon}(j)}}$$

$\beta_k \leftarrow \ln \tau_k$

$L_k \leftarrow \{\mathcal{S}_{2^k}, \cdots, \mathcal{S}_{2^{k+1}-1}, \boldsymbol{\tau}_k^{-\boldsymbol{\epsilon}(l)}, l = 2^k \cdots 2^{k+1} - 1\}$
**end for**

**Output :** $h(x_i) = \mathrm{sign}\Big(\sum_{k=1}^{p} \beta_k G_k(x_i)\Big)$

Les résultats ci-dessus sont théoriques et des simulations numériques ne sont pas nécessaires. AdaBoost fournit itérativement une liste de classifieurs faibles. Ces classifieurs faibles, agissant sur $\mathcal{S}$, fournissent des tables de vérité qui sont des partitions imbriquées de $\mathcal{S}$ et les formules analytiques sont pertinentes dans ce cas.

# 8 Conclusion et perspectives

L'approche logique présentée dans cet article permet d'obtenir le classifieur résultant de la procédure AdaBoost au moyen de formules (3) utilisant les éléments des tables de vérité : la procédure AdaBoost, d'un point de vue informatique, n'est qu'une simple formule, mais elle construit les classifieurs successifs en faisant appel au WeakLearner.

La procédure décrite ici ne remplace pas le calcul d'un nouveau classifieur faible à chaque étape réalisé par AdaBoost. Les formules permettent de calculer les coefficients des classifieurs lorsque ceux-ci sont fixés ; donc, en particulier, de retrouver *a posteriori* le classifieur résultant calculé par AdaBoost en utilisant la liste des classifieurs faibles qu'il a produit.

La structuration donnée par cette approche logique fournit aussi une prédiction des cas où le risque convexifié n'a pas de point de minimum (comme mentionné par [7], Section 7.2.2 cette situation peut se produire), en s'appuyant uniquement sur les éléments de la table de vérité $c_j, j \in 2^p \cdots 2^{p+1} - 1$.

Elle pourrait donner un meilleur éclairage à l'algorithme 7.2 de [7] (minimisation gloutonne du risque exponentiel, qui s'écrit $\sum_{j=2^p}^{2^{p+1}-1} c_j e^{-\boldsymbol{\beta}.\boldsymbol{\epsilon}(j)}$). Cet algorithme pourrait être facilement remplacé par une version non gloutonne, qui est également une formule de récurrence, qui calcule successivement $\tau_1^m, ..., \tau_p^m$ en calculant (par des formules identiques à (2) et (3)) le point de minimum en chaque coordonnée $\beta_k$. Remarquons que c'est le premier parcours de toutes les coordonnées par un algorithme de relaxation. Rappelons que l'algorithme de relaxation converge vers le minimum du risque convexifié, s'il existe, et ne converge pas s'il y a un infimum ou un minimum non unique même si la valeur du coût convexifié converge vers une limite.

# Références


[1] Leo BREIMAN : Bagging predictors. *Machine learning*, 24:123–140, 1996.

[2] Leo BREIMAN : Random forests. *Machine learning*, 45:5–32, 2001.

[3] Leo BREIMAN : The 2002 wald memorial lectures. population theory for boosting ensembles 2004. *The Annals of Statistics*, 32(1):1–11, 2004.

[4] Yoav FREUND et Robert E SCHAPIRE : A desicion-theoretic generalization of on-line learning and an application to boosting. *In European conference on computational learning theory*, pages 23–37. Springer, 1995.

[5] J. FRIEDMAN, T. HASTIE et R. TIBSHIRANI : Additive logistic regression : a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[6] Indraneel MUKHERJEE, Cynthia RUDIN et Robert E. SCHAPIRE : The rate of convergence of adaboost. *JMLR : Workshop and Conference Proceedings 19 24th Annual Conference on Learning Theory*, pages 537–557, 2011.

[7] Robert E SCHAPIRE et Yoav FREUND : Boosting : Foundations and algorithms. *Kybernetes*, 42(1):164–166, 2013.

[8] Michael von AITZING : *Thesaurus principum hac aetate in Europa viventium...* Kempen, 1590.

[9] Tong ZHANG et Bin YU : Boosting with early stopping : Convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579, 2005.