Université Libre de Bruxelles

Faculté de Lettres, Traduction et Communication

Le Boosting en Machine Learning

Metango Kenfack Cathy

Travail réalisé sous la direction de DE VALERIOLA Sébastien dans le cadre du cours d'Architecture des Systèmes d'information (M-STIC-B540)

Année académique 2022-2023

Table des matières

1 Introduction

Le thème soumis à notre étude est celui du Boosting en machine Learning. L'apprentissage automatique est une discipline qui permet de donner à un ordinateur la capacité d'apprendre à partir des données pour résoudre une tache qui implique généralement les prédictions, Il s'agit d'un sous-domaine de l'intelligence artificielle, qui est définie au sens large comme la capacité d'une machine à imiter le comportement humain intelligent. Les systèmes d'intelligence artificielle sont utilisés pour exécuter des tâches complexes d'une manière similaire à la façon dont les humains résolvent les problèmes. L'apprentissage automatique utilisent pour ce fait des méthodes pour apprendre des informations à partir des données sans modèle de référencé prédéterminé, Il existe de ce faites plusieurs méthodes de machine Learning parmi lesquelles l'algorithme de boosting. Le boosting est un algorithme de machine Learning permettant de réduire les erreurs dans l'analyse prédictive des données. C'est une méthode d'ensemble Learning c'est a dire elle crée un modèle fort a partir d'un certain nombre de modèles faibles cela en demandant a chaque model de corriger les erreurs de son prédécesseur : les faiblesses des uns sont compensées par les forces des autres. L étude de ce projet porte sur cette notion qu'est le BOOSTING. Pour mener a bien cette étude nous aborderons plusieurs points à savoir la Définition du machine Learning et du boosting d'une part et d'autre part donner l'importance du boosting, présenter le principe et fonctionnement du boosting et enfin donner un exemple d'algorithme du boosting à savoir Adaboost.

2 Machine Learning

2.1 Définition

Notre camarade de classe a du mal à faire une différence entre le citron et la mandarine avec suffisamment de précision pour la reconnaitre dans un supermarché. Mais si nous lui montrons quelques-unes de ses photos, il repérera immédiatement les traits révélateurs dont il a besoin. Comme on dit, une image, un exemple, vaut mille mots supposons ici que votre ami c'est votre machine et qu'on aimerait apprendre à la machine à distinguer les citrons des mandarines. Plutôt que de coder une machine avec des images représentatifs des citrons et mandarines, nous pouvons plutôt programmer la machine pour apprendre à les distinguer grâce à une expérience répétée avec les citrons et les mandarines réelles. C'est ça le machine Learning donner la capacite à un ordinateur d'apprendre à partir des données.

Incapables de définir certains objets ou concepts avec suffisamment de précision, nous voulons les transmettre à la machine au moyen d'exemples. Pour que cela fonctionne, cependant, l'ordinateur doit avoir la capacite de convertir les exemples fournis par l'humain pendant l'apprentissage en connaissances. Une fois que cette phase d'apprentissage est faite, on peut faire des prédictions c'est à dire produire une réponse correcte sur les données similaires mais sur lesquelles le programme n'a pas été entrainée. D'où l'intérêt pour les algorithmes d'apprentissage automatique, le sujet de ce manuel.

Le machine Learning est donc une branche évolutive des algorithmes ayant pour but d'imiter la capacite humaine. Elle s'appuie sur notions de déférentes disciplines telles que informatique, l'intelligence artificielle, les probabilités et statistiques, la psychologie. Elle a montré son importance et elle est applique dans plusieurs domaines varie de la vie tel qu'en finance pour la prédiction du chiffre d'affaires, la détection de fraude, en transport pour la prédiction du nombre d'accident en politique pour la prédiction de résultat, en biologie, en divertissement, en marketing , en médecine , en biomédecine, en physique médicale et bien encore.

Nous allons donc pour notre travail montrer comment le machine Learning grâce aux algorithmes pour faire des prédictions. Plusieurs algorithmes apprentissage automatique existe, mais dans notre cas nous parlerons de l'algorithme de Boosting.

2.2 fonctionnements du machine Learning : apprendre à partir des données

Les machines pour qu'elles apprennent ont besoin des données (images, sons, textes, vidéos, prix d'une maison). Ce processus en apprentissage automatique est appelé expérience. Après avoir donné ces donnes à la machine on doit par suite préciser la tâche quelle doit réaliser (catégoriser les images, prédire le prix d'une maison, prédire le chiffre d'affaires). En fonction de la tache quelle a appris nous devons à la fin évaluer son apprentissage pour mesurer sa performance. Il existe donc pour ce fait des algorithmes d'apprentissages. Le machine Learning regroupe trois grandes familles d'algorithme d'apprentissage : apprentissage supervisé, apprentissage non supervisé et apprentissage semi supervisé. En fonction du type d'apprentissage nous avons plusieurs algorithmes, mais comme énoncé plus haut nous parlerons d'algorithme de boosting qui est un algorithme d'apprentissage supervisé[El Naqa and Murphy, 2015].

3 Algorithme de Boosting

Nous présentons dans cette section le boosting et son fonctionnement et par la suite nous illustrons les differents types d'algorithme du boosting.

3.1 Enjeux : Pourquoi le boosting est t'il utilisé

Avec le development de la technologie , le progrès dans les divers domaines tel que le marketing , la santé , les finance , etc. . ., il est necessaires de développer des techniques d'apprentissage automatique plus complexes et plus avancées[edu, 2019].

Le boosting est une technique qui peut etre utilisée pour resoudre des problemes complexes , basés sur des données et le monde réel. Supposons qu'on veux créer un modèle pour un ensemble de données d'images contenant des images de chats et de chiens et que nous voulons classer ces images en deux classes différentes. Nous allons pour cela commencer par identifier les images en fonction de certaines règles, telles que les éléments suivants :

1. La photo a des oreilles pointues : chat
2. L'image est un œil en forme de chat : chat
3. L'image a de grands membres : chien
4. L'image a des griffes aiguisees : chat
5. L'image a une structure de bouche plus large : chien

En effet , les prédictions de chacun de ces apprenants faibles peuvent être combinées avec une règle de majorité ou une moyenne pondérée pour rendre les prédictions plus précises. Cela en fait de solides modèles d'apprentissage. Dans lexemple ci dessus, nous avons defini 5 classificateurs faibles et la majorite de ces regles (3 classificateurs sur 5 predisent limage comme un chat) nous donne la prediction que l'image est un chat. On conclus que notre sortie est un chat. Ceci nous amene a nous pose la question ci après.

3.2 Définition

Le boosting est une technique d'ensemble general qui cree un apprenant fort a partir d'un certains nombre d'apprenant faibles. Cela se fait en construisant un modele à partir des donnees d'apprentissage puis en creant un deuxieme modele qui tente de corriger les erreurs du model precedent.

3.3 Les techniques d'ensemble learning

En regroupant un ensemble de modeles , on obtiens ume meilleurs performance que lorsqu'on utilise nos modèles chacun de leur cote. Cette methode est base sur le concept The wisdom of the crowd qui est un phenomene lie a la loi des grands nombres qui fait qune foule d'individus a souvent plus raison qu'un expert tout seul a condition que la foule soit grande et diversifieée (les modèles doivent etres différent, construit sur des données différents ou avec des algorithms d'apprentissage différents)et compétente. En apprentissage automatique on peu utiliser ce concept pour creer un ensemble de modeles qui surpassent les performances des meilleures modeles de machine d'apprentissage au monde[ens]. Pour cela on distingue 3 grandes techniques qui sont : le Bagging , le Boosting et le Stacking.

le Boosting Ici, les apprenants faibles sont générés séquentiellement , en demandant à chaque modèle de corriger les erreurs de son predécesseur, on n'a donc toujours un nouvel algorithme formé sur les mauvaises predictions faites précédemment[ens]. En attribuant des poids plus élevés aux échantillons précédemment mal classés, les performances du modèle sont améliorées. Un exemple de boosting est l'algorithme AdaBoost.

Le Bagging les apprenants faibles sont produits parallèlement. Ici , les données sont echantillonnées (le boosstrapping) et on obtenir un ensemble de models diversivié en entrenant sur

une portion aleatoire de données chaque modèle. La prediction finale est la moyenne des predictions de chaque modèles qui ont été prealablement obtenus.

Le Stacking cette technique consiste à entrainer plusieurs modèles différents avec les meme données puis utiliser les resultats des différents modèles sur un modèle final qui donne la prediction finale. Elle permet entrainer un modèle d 'apprentissage à reconnaitre qui a tort et qui a raison dans un ensembles de modèles, ceci améliore encore plus la performance générale.

Bien que ces trois modèles sont très important pour la prédiction des problèmes complexes, nous allons nous focalisé sur les techniques du boosting.

3.4 Fonctionnement du boosting

Le principe de base de algorithmes de boosting est de générer plusieurs apprenants faibles et de combiner leurs prédictions pour former un apprenant fort. Ces apprenants faibles sont générés en appliquant des algorithmes d'apprentissage automatique de base à différentes distributions de l'ensemble de données.[edu, 2019] Ces algorithmes génèrent des règles faibles à chaque itération. Après quelques itérations, les apprenants faibles sont combinés en apprenants forts qui prédisent des résultats plus précis.

étape 1 L'algorithme de base lit les données et attribue des poids égaux à chaque observation d'échantillon de données. Puis algorithme de base faire des predictions pour chacun de ces échantillons.

étape 2 L'algorithme identifie les prédictions incorrectes faites par les apprenants de base. À l'itération suivante, ces prédictions erronées sont attribuées à l'apprenant de base suivant avec des poids plus élevés pour ces prédictions erronées.

étape 3 on répète l'étape 2 jusqu'à ce que l'algorithme puisse apprendre correctement. Par conséquent, l'objectif principal du boosting est de se concentrer davantage sur les prédictions erronées.

Maintenant que nous connaissons comment fonctionne l'algorithme de boosting, intéressons-nous maintenant aux différents types de algorithme de boosting. Mais avant nous présenterons des Forces et Faiblesses de celui ci.

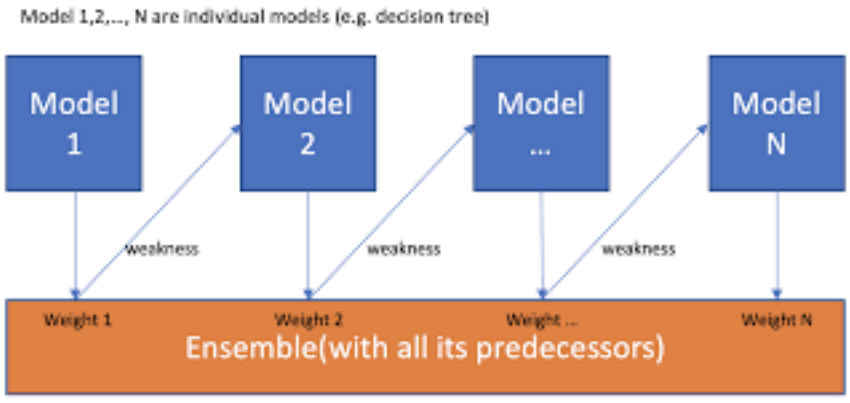

FIGURE 1 – Présentation de la technique du Boosting

3.5 Les forces et Faiblesses du Boosting

3.5.1 Les forces de l'algorithme de Boosting

Le boosting offrent plusieurs avantages.

Facile à implémenter

grâce a ces algorithmes faciles à comprendre et à interpréter qui apprennent de leurs erreurs, Ces algorithmes ne nécessitent aucun prétraitement des données et disposent de routines intégrées pour gérer les données manquantes[AWS]. De plus, la plupart des langages ont des bibliothèques intégrées pour implémenter des algorithmes de boosting avec de nombreux paramètres pour affiner les performances.

Reduction des biais

Le biais est la présence d'incertitude ou d'imprécision dans les résultats d'apprentissage automatique. Un algorithme de renforcement combine plusieurs apprenants faibles de manière itérative pour améliorer itérativement une observation. Cette approche permet d'atténuer les biais élevés communs aux modèles d'apprentissage automatique.

Efficacité de calcul

Les algorithmes de boosting donnent la priorité aux fonctionnalités qui augmentent la précision des prédictions pendant l'entraînement. Ils permettent de réduire les attributs de données et de traiter efficacement de grands ensembles de données. Les algorithmes de boost donnent la priorité aux fonctionnalités qui augmentent la précision des prédictions pendant l'entraînement. Ils permettent de réduire les attributs de données et de traiter efficacement de grands ensembles de données.

Malgré les forces de cet algorithme, elle présente néanmoins des faiblesses.

3.5.2 Les faiblesses de l'algorithme de boosting

Vulnérabilité aux données aberrantes

Les modèles de boost sont sujets aux valeurs aberrantes ou aux valeurs de données qui diffèrent du reste de l'ensemble de données. Comme chaque modèle tente de corriger les bogues du modèle précédent, les valeurs aberrantes peuvent grandement fausser les résultats[AWS].

Implémentation en temps réel

Le boosting peut être difficile à utiliser pour les implémentations en temps réel car l'algorithme est plus complexe que les autres processus. Les techniques de boosting sont très adaptables. Cela vous permet d'utiliser différents paramètres de modèle qui affectent directement les performances du modèle.

3.6 Les différents types d'algorithme de boosting

Il existe trois façons principales d'effectuer le boosting.

Boosting adaptatif

Adaptative Boosting (AdaBoost) a été l'un des premiers modèles de boosting développés. Il essaie de s'adapter et de s'autocorriger à chaque itération du processus de boosting.

AdaBoost donne initialement à chaque ensemble de données un poids égal. Ensuite, ajustez automatiquement les poids des points de données après chaque arbre de décision. Pesez les

articles mal classés et corrigez-les pour le tour suivant. Ce processus est répété jusqu'à ce que le résidu, la différence entre la valeur réelle et la valeur prédite, tombe en dessous du seuil acceptable[Saini, 2021].

AdaBoost peut être utilisé avec de nombreux prédicteurs, mais il n'est généralement pas aussi sensible que d'autres algorithmes de boosting. Cette approche ne fonctionne pas bien lorsqu'il existe une corrélation entre les caractéristiques ou lorsque les données sont très dimensionnelles.

Dans l'ensemble, AdaBoost est un type de boost approprié pour les problèmes de classification.

Le boosting de gradient

Gradient Boosting(GB), également connu sous le nom d'amplificateur de gradient, est similaire à AdaBoost en ce sens qu'il s'agit également d'une technique d'entraînement séquentielle.

Contrairement à AdaBoost GB ne donne pas de poids aux éléments mal classés, le logiciel GB optimise la fonction de perte en générant des apprenants de base de manière séquentielle, de sorte que l'apprenant de base actuel est toujours plus efficace que le précédent. Cette méthode essaie d'obtenir des résultats précis la première fois au lieu de corriger les erreurs en cours de route comme le fait AdaBoost[edu, 2019]. Cela permet au logiciel GB de produire des résultats plus précis.

Le boosting de gradient est utile pour les problèmes de classification et de régression.

Le boosting de gradient extreme

Extreme Gradient Boosting (XGBoost) améliore le boosting de gradient de plusieurs manières en termes de vitesse de calcul[Mousa et al., 2019]. XGBoost utilise plusieurs cœurs du CPU, permettant un apprentissage parallèle pendant l'entrainement. Il s'agit d'un algorithme de boosting qui peut gérer de grands ensembles de données, ce qui le rend attractif pour les applications de Big Data.

Les principales fonctionnalités de XGBoost sont la parallélisation, le calcul distribuée, l'optimisation du cache et le traitement hors cœur.

Bien que ces trois algorithmes sont très important , nous allons nous focalisé sur algorithme Adaboost.

4 Algorithme Adaboost

4.1 Présentation de l'algorithme.

4.2 Explication détaillée de l'algorithme.

[Schapire and Freund, 2012]

comme tout algorithme d'apprentissage, l'algorithme de boosting prend en entrée un ensemble d'exemple d'apprentissage D = $(\mathrm{x}_1, y_1), \ldots, (x_m, y_m)$ où $x_i \in X$ et $y_i \in [-1, +1]$

chaque x_i est une instance de X et chaque y_i est l'etiquette ou la classe associée.

le seul moyen dont dispose un algorithme de boosting pour apprendre des donées est d'appeler l'algorithme d'apprentisage de base. Cependant, si l'apprenant de base est simplement appélé de façon répétée, toujours avec le meme jeu de données d'apprentissage, nous ne pouvons pas

Algorithm 1 algorithme AdaBoost.

Require: D = $(x_1, y_1), \ldots, (x_m, y_m) ou x_i \in X$, $y_i \in [-1, +1]$

Initialisation : $D_1(i) = \frac{1}{m}$ (initialisation de la repartision du poids) **for** $i = (1), \ldots, (m)$ **do**

for i=(1),. . .,(T) : ou T est le nombre de tours d'apprentissage

1. Entraîner l'apprenant faible en utilisant la distribution D_t
2. Obtenir l'hypothèse faible h_t :X → [-1,+1]
3. Objectif : sélectionner h_t pour minimiser l'erreur
   $\epsilon_t = \mathbf{Pr}_i \sim D_t[h_t(x_i) \neq y_i] = \sum (i : h_t(x_i) \neq y_i) D_t(i)$ (il s'agit du % de mesure de l'erreur h_t)
4. calcul du poids $\alpha_t = (\frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ (il s'agit du % qui determine le poids h_t)
5. Mise à jour, **for** i= (1),. . .,(m) :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} * \begin{cases} e\text{-}\alpha t & if \quad h_t(x_i) = y_i (si) \text{ si instance i est correctement classée} \\ e^{\alpha} t & if \quad h_t(x_i) \neq y_i \text{ si instance i n'est pas correctement classée} \end{cases}$$

$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ (% de mise à jour de la distribution.)

où Zt est un facteur de normalisation ($\sum_{t=1}^{m} D_{t+1} = 1$)

Sortie : H(x) = sign $(\sum_{t=1}^{T} \alpha_t \ h_t(x))$

nous attendre a ce que quelque chose d'intéressant se produise ; par contre, nous nous attendons à ce que quelque le meme classificateur de base, ou presque, soit produit encore et encore, de sorte que peu de choses sont gagnées par rapport à l'éxecution de l'apprenant de base une seule fois.

Cela montre que si l'algorithme veut s'améliorer l'apprenant de base, doit d'une certaine manière manipuler les données qu'il lui sont fournit.

Nous décrivons ici en détail l'algorithme boosting Adaboost dont le pseudocode est présenté plus haut. Adaboost procède par cycles ou appels itératifs à l'apprenant de base.

Pour choisir les ensembles de formation fournis à l'apprenant de base à chaque tour, AdaBoost maintient une distribution sur les exemples de formation. La distribution utilisée au tième tour est désignée par Dt , et le poids qu'elle attribue à l'exemple d'apprentissage i est désigné par Dt(i). Intuitivement, ce poids est une mesure de l'importance de la classification correcte de l'exemple i au tour actuel. Initialement, tous les poids sont égaux, mais à chaque tour, les poids des exemples incorrectement classés sont augmentés de sorte que, effectivement, les exemples difficiles obtiennent successivement un poids plus élevé, forçant l'apprenant de base à concentrer son attention sur eux.

Le travail de l'apprenant de base est de trouver un classificateur de base h_t : X → [-1, +1] approprié pour la distribution D_t . Conformément à la discussion précédente, la qualité d'un classificateur de base est mesurée par son erreur pondérée par la distribution D_t : $\epsilon_t = \mathbf{Pr}_i \sim D_t[h_t(x_i) \neq y_i] = \sum (i : h_t(x_i) \neq y_i) D_t(i)$.

ici , $\epsilon_t$désigne la probabilité que l'exemple soit choisi au hazard . **Pr**_i∼$D_t[h_t(x_i) \neq y_i]$désigne la probabilité que l'exemple soit choisi au hasard (elle suit la distribution D_t donnée par l'indice i). Par conséquent,l'erreur pondérée ϵ_t est La probabilité que h_t classe mal un échantillon aléatoire s'il est choisi en fonction de Dt. En conséquent , il s'agit de la somme des poids des

exemples mal classés. Notons que les erreurs sont mesurées par rapport à la même distribution D_t. Le classificateur de base à été entrainé.

L'apprenant faible tente de choisir une hypothèse faible h_t avec une faible erreur pondérée ϵ_t . Dans ce contexte, cependant, nous ne nous attendons pas à ce que cette erreur soit particulièrement faible dans un sens absolu, mais seulement dans un sens plus général et relatif ; en particulier, nous nous attendons à ce qu'elle soit seulement un peu meilleure que le hasard, et typiquement loin de zéro. Pour souligner le manque de rigueur dans ce que nous exigeons de l'apprenant faible, nous disons que le but de l'apprenant faible est de minimiser l'erreur pondérée, en utilisant ce mot pour signifier une diminution plus vague et moins stricte que celle connotée par minimiser.

une fois que l'apprenant de base h_t est reçu , Adaboost choisit un paramètre α_t comme dans l'algorithme. Intuitivement, α_t mesure l'importance qui est attribuée à h_t. Notons que α_t devient plus grand lorsque ϵ_t devient plus petit. Ainsi, plus le classificateur de base h_t est precis, plus l'importance lui est accordée. La distribution D_t est ensuite mise à jour en utilisant la règle indiquée dans l'algorithme. Tout d'abord, tous les poids sont multipliés soit par $e^- < 1$ pour les exemples correctement classés par h_t , soit par $e^\alpha > 1$ pour les exemples incorrectement classés .

De meme, étant donné que nous utilisons des étiquettes et des prédictions dans -1 , +1 , cette mise à jour peut etre exprimée de manière plus concise comme une mise à l'échelle de chaque instance i par $\exp(-\alpha_t y_t h_t(x_i))$.Ensuite, l'ensemble de valeurs résultant est ensuite renormalisé en divisant par le facteur z_t afin de s'assurer que la nouvelle distribution D_{t+1} a bien une somme égale à 1.

Cette règle a pour effet d'augmenter le poids des classificateurs mal classés par h_t et diminuer le poids des exemples correctement clasés. ainsi le poids a tendance se concentrer sur les exemples "durs". Adaboot, pour etre plus precis , choisit une nouvelle distribution D_{t+1} sur laquelle le dernier classifieur de h_t est sur de faire extremement mal. De cette façon, comme nous l'avons vu plus haut, Adaboot essaie à chaque itération de forcer l'apprenant de base à apprendre quelque chose de nouveau sur les données.

Après plusieurs appels de l'apprenant de base, Adaboost combine les nombreux classificateurs de base en un seul classificateur final H. Ceci est réalisé par un simple vote pondéré des classificateurs de base. Autrement dit, étant donné une nouvelle instance x, le classificateur combiné évalue tous les classificateurs de base et prédit avec la majorité pondérée des prédictions des classificateurs de base. Ici, le vote du t-ième classificateur de base ht est pondéré par le paramètre α_t choisi précédemment. La formule résultante pour la prédiction de H est comme indiqué dans l'algorithme .

Résumé :

1. Au départ, toutes les observations ont le même poids.
2. Un modèle est construit sur un sous-ensemble de données.
3. En utilisant ce modèle, des prédictions sont faites sur l'ensemble des données.
4. Les erreurs sont calculées en comparant les prédictions et les valeurs réelles.
5. Lors de la création du modèle suivant, des poids plus élevés sont attribués aux points de données qui ont été prédits de manière incorrecte.

6. Les pondérations peuvent être déterminées en utilisant la valeur de l'erreur. Par exemple, plus l'erreur est élevée, plus le poids attribué à l'observation est important.
7. Ce processus est répété jusqu'à ce que la fonction d'erreur ne change pas, ou que la limite maximale du nombre d'estimateurs soit atteinte.

4.3 Cas pratique

Pour illustrer le fonctionnement d'AdaBoost, regardons le petit problème d'apprentissage des jouets illustré dans l'image ci-dessus [Schapire and Freund, 2012]. Une instance ici est un point dans le plan marqué par + ou -. Dans ce cas, nous avons m = 10 exemples de formation comme indiqué. 5 sont positifs et 5 sont négatifs. Supposons que notre apprenant de base trouve des classificateurs définis par des lignes verticales ou horizontales traversant le plan. Par exemple, un tel classificateur de base défini par une ligne verticale pourrait classer tous les points à droite de la ligne comme positifs, et tous les points à gauche comme négatifs.
On peut vérifier qu'aucun classificateur de base de cette forme ne classe correctement plus de sept des dix exemples d'apprentissage, ce qui signifie qu'aucun n'a une erreur d'apprentissage non pondérée inférieure à 30%. A chaque tour t, nous supposons que l'apprenant de base trouve toujours l'hypothèse de base de cette forme qui a une erreur pondérée minimale par rapport à la distribution D_t (brisée arbitrairement).. Nous supposons qu'à chaque tour t, l'apprenant de base trouve toujours une hypothèse de base (arbitrairement brisée) de cette forme avec la plus petite erreur pondérée par rapport à la distribution Dt. Nous verrons dans cet exemple comment, en utilisant un tel apprenant de base pour trouver des classificateurs de base aussi faibles, AdaBoost est capable de construire un classificateur combiné qui classe correctement tous les exemples d'apprentissage en seulement T = 3 tours de boosting.

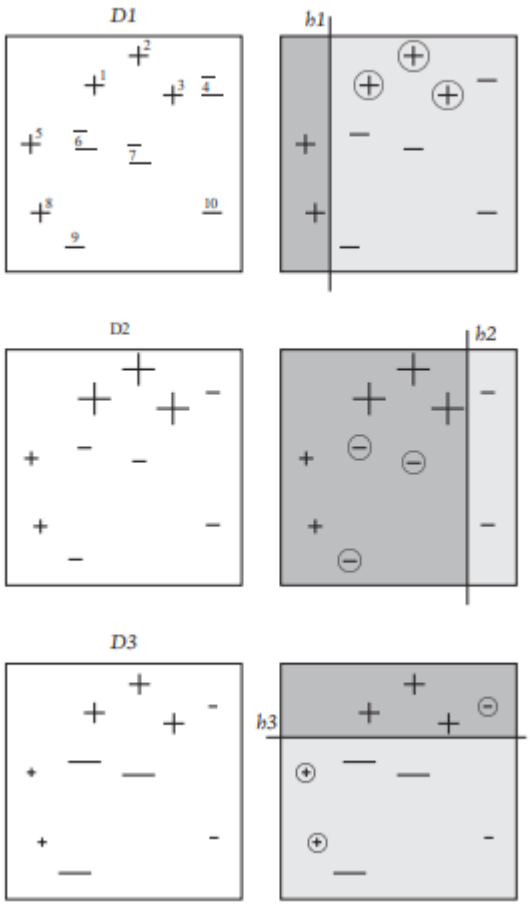

FIGURE 2 – Probleme d'apprentissage

Illustration d'AdaBoost à l'aide d'un petit problème fictif avec m = 10 exemples. Chaque ligne représente t = 1, 2, 3 tours. La case à gauche de chaque ligne représente la distribution D_t à l'échelle de la taille pour chaque exemple Proportionnel au poids dans cette distribution. Chaque case à droite indique une hypothèse faible h_t et un ombrage foncé indique une région d'hypothèse faible. L'ombrage foncé indique la partie du domaine prédite comme étant positive. Les exemples mal classés par h_t sont encerclés.

Le calcul numérique correspond à l'exemple du jouet.

	1	2	3	4	5	6	7	8	9	10	
D_1 (i)	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	$E_1 = 0{,}30, \alpha_1 \approx 0{,}42$
$e\text{-}\alpha 1^{\,yi\,h1\,(xi)}$	1.53	1.53	1.53	0.65	0.65	0.65	0.65	0.65	0.65	0.65	
D_1 (i) $e^{-\alpha_1 yi\,h1\,(xi)}$	0,15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$Z_1 \approx 0.92$
D_2 (i)	0.17	0.17	0.17	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$E_2 \approx 0{,}21, \alpha_2 \approx 0{,}65$
$e\text{-}\alpha 2^{\,yi\,h2\,(xi)}$	0.52	0.52	0.52	0.52	0.52	1.91	1.91	0.52	1.91	0.52	
D_2 (i) $e^{-\alpha_2 yi\,h2\,(xi)}$	0,09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04	$Z_2 \approx 0.82$
D_3 (i)	0.11	0.11	0.11	0.05	0.05	0.17	0.17	0.05	0.17	0.05	$E_3 \approx 0{,}14, \alpha_3 \approx 0{,}92$
$e\text{-}\alpha 3^{\,yi\,h5\,(xi)}$	0.40	0.40	0.40	2.52	2.52	0.40	0.40	2.52	0.40	0.40	
D_3 (i) $e^{-\alpha_3 yi\,h3\,(xi)}$	0,04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02	$Z_3 \approx 0.69$

FIGURE 3 – Calcul numérique de l'algorithme

Les calculs sont présentés pour les 10 exemples numérotés de la figure. Les exemples où l'hypothèse h_t échoue sont indiqués par des nombres soulignés dans les lignes marquées D_t.

Au premier tour, AdaBoost attribue un poids égal à tous les exemples en les dessinant tous de la même taille dans la case intitulée D_1, comme indiqué sur la figure. A partir de ces exemples de pondération, l'apprenant de base choisit l'hypothèse de base marquée par h1 dans le schéma. Cela classe un point comme positif uniquement s'il se trouve à gauche de cette ligne.

Cette hypothèse classe incorrectement trois points, à savoir les trois points positifs encerclés, de sorte que son erreur 1 est de 0,30. En se branchant sur la formule de l'algorithme , on obtient $\alpha_1 \sim 0,42$.

En construisant D_2, on augmente le poids des 3 points mal classés par h_1 et on diminue le poids de tous les autres points. Ceci est indiqué par la taille des points dans la case étiquetée D_2.

Au deuxième tour, l'apprenant de base selectionne la ligne marquée h_2. Ce classificateur de base classe correctement 3 élements de poids relativement élevé manqués par h_1, mais au prix de l'absence de trois autres points à poids comparativement faible qui ont été correctement classés par h_1.

Sous la distribution D_2, ces trois points n'ont un poids que d'environ 0,07, donc l'erreur de h_2 par rapport à D_2 est $\epsilon_2 \sim 0,21$ce qui donne $\alpha_2 \sim 0,65$.Lors de la construction de D_3, les poids de ces trois points mal classés sont augmentés tandis que les poids des autres points sont diminués.

Au troisième tour, le classificateur h_3 est choisi. Ce classificateur ne rate jamais les points mal classés par h_1 et h_2. En effet, ces points ont des poids relativement élevés dans D3. En revanche, 3 points de très faible poids sous D3 sont mal classés car ils ne sont pas mal classés par h_1 ou h_2. Au troisième tour, $\epsilon_3 \sim 0,14, \alpha_3 \sim 0,92$.

Le classificateur combiné H est un vote pondéré de h_1, h_2 et h_3, comme le montre la figure où les poids des classifieurs respectifs sont $\alpha_1, \alpha_2, \alpha_3$.Bien que chacun des classificateurs faibles composites classe mal trois des dix exemples, le classificateur combiné.

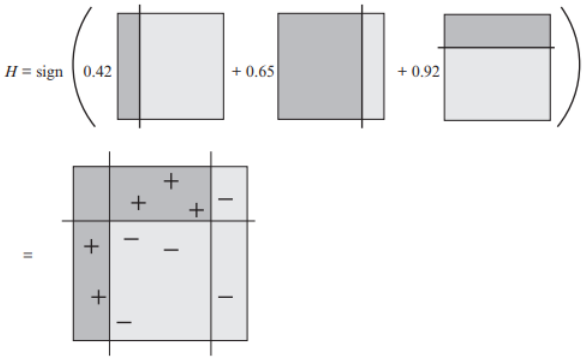

FIGURE 4 – Classificateur final H

Le classificateur combiné dans l'exemple de jouet illustré est calculé comme le signe de la somme pondérée des trois hypothèses faibles. $\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$comme ci-dessus. Cela correspond au classificateur ci-dessous. (Comme indiqué, les régions où le classificateur prédit positivement sont indiquées par un ombrage plus foncé.)

comme le montre la figure, classe correctement tous les exemples de formation. Par exemple, la classification de l'exemple négatif dans le coin supérieur droit (instance 4), qui est classifié classée négative par h_1 et h_2, mais positive par h_3, d'où on n'a.

$\text{sign}(-\alpha_1 - \alpha_2 + \alpha_3) = sign(-0.15) = -1$

5 Conclusion

En réalisant ce travail, nous souhaitions approfondir la matière vue au cours d'architecture de système d'information sur l'introduction au machine learning. Nous rétenons de cette étude que le Boosting, technique ensemble learning est un domaine du machine learning (une branche de l'intelligence artificielle). cet algorithme combine plusieurs modèles (apprenants) faible dans une methode séquentielle ce qui améliore les observations de mannière itérative.

Centrer sur la résolution des problèmes de classification, Le boosting pésente trois algoithmes puissants à savoir Adaboost, le boosting de gradiant et le boosting de gradiant extreme. De cette trois algorithme Adaboost. Il est appelé Adaptive Boosting car les poids sont réaffectés à chaque instance et les instances mal classées se voient attribuer des poids plus élevés. Le renforcement est utilisé pour réduire les biais et la variance dans l'apprentissage supervisé. À l'exception du premier apprenant, tous les apprenants suivants proviennent d'apprenants déjà adultes. En termes simples, les apprenants faibles deviennent des apprenants forts.

Durant ce projet, nouus avons rencontré plusieurs difficultés tels que la compréhension des concepts mathématiques présent dans l'algorithme, lutilisation de l'outil latex pour la rédaction des équations mathématiques, mais avec beaucoup de recherche et détermination nous avons pu atteindre nos objectifs.

Table des figures

Bibliographie

Issam El Naqa and Martin J. Murphy. What Is Machine Learning ? pages 3–11, 2015.

A Beginners Guide To Boosting Machine Learning Algorithms. *Edureka*, June 2019. URL `https://www.edureka.co/blog/boosting-machine-learning/`. Section : Data Science.

Ensemble Learning | Ensemble Techniques. URL `https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/`.

Qu'est-ce que le boosting ? Guide du boosting en machine learning – AWS. URL `https://aws.amazon.com/fr/what-is/boosting/`.

Anshul Saini. AdaBoost Algorithm - A Complete Guide for Beginners. *Analytics Vidhya*, September 2021.

Saleh R. Mousa, Peter R. Bakhit, and Sherif Ishak. An extreme gradient boosting method for identifying the factors contributing to crash/near-crash events : a naturalistic driving study. *Canadian Journal of Civil Engineering*, 46(8) :712–721, August 2019. ISSN 0315-1468, 1208-6029.

Robert E. Schapire and Yoav Freund. Boosting : foundations and algorithms. 2012.