RÉPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie

Université de Yaoundé I

École Nationale Supérieure
Polytechnique de Yaoundé (ENSPY)

BP 8390 Tel/Fax: 222 22 45 47
Yaoundé, Cameroun

REPUBLIC OF CAMEROON
Peace-Work-Fatherland

The University of Yaounde I

National Advanced School of
Engineering of Yaounde (NASEY)

PO BOX 8390 Tel/Fax: 222 22 45 47
Yaounde, Cameroon

ANI-IA 3046 : CREATION D'UNE INTELLIGENCE ARTIFICIELLE 1 :

Nom : MATEMB

Prénom : AUGUSTIN KEVIN

Matricule : 21P564

Filière : ART ET INTELLIGENCE ARTIFICIELLE 3

EXPOSEE : AdaBoost

INTRODUCTION

AdaBoost, qui signifie "Adaptive Boosting" en français, est une technique puissante d'apprentissage automatique utilisée pour les tâches de classification. En gros, il s'agit de combiner plusieurs algorithmes de classification relativement faibles (des algorithmes qui ne sont pas excellents pour classifier des éléments) en un seul algorithme beaucoup plus performant.

- **Principe de base : Renforcer du faible au fort**

Imaginez un groupe d'amis qui essaient d'identifier différents types d'oiseaux. Chaque ami peut être capable de reconnaître quelques oiseaux, mais aucun n'est infaillible. AdaBoost revient à leur faire voter tous ensemble pour déterminer le type d'oiseau. Les amis qui se sont constamment trompés lors des tours précédents auront moins d'influence sur la décision finale, tandis que ceux qui avaient raison auront plus de poids. De cette façon, les connaissances combinées du groupe deviennent bien plus précises que celles de chaque individu.

- **Fonctionnement d'AdaBoost**

1. **Attribution de poids :** AdaBoost commence par attribuer des poids égaux à tous les points de données de votre ensemble de données d'apprentissage.
2. **Entraînement d'un apprenant faible :** Ensuite, il entraîne un apprenant faible, comme un simple arbre de décision, sur les données.
3. **Évaluation de l'apprenant :** AdaBoost analyse la performance de l'apprenant faible. Les points de données que l'apprenant a mal classés se voient attribuer un poids plus important au tour suivant. Cela oblige l'apprenant à se concentrer davantage sur les exemples les plus difficiles lors de la prochaine itération.
4. **Répétition et amélioration :** Les étapes 2 et 3 sont répétées plusieurs fois. À chaque tour, les apprenants faibles se concentrent davantage sur les points de données mal classés précédemment, améliorant progressivement la précision globale.
5. **Ensemble final :** Au final, AdaBoost combine tous les apprenants faibles en un seul classificateur puissant. Ce classificateur final tient compte des poids attribués à chaque apprenant en fonction de ses performances, accordant plus de poids aux plus précis.

- **Avantages d'AdaBoost**

  - **Précision améliorée**

  AdaBoost atteint une précision de classification supérieure à celle de ses apprenants faibles individuels grâce à plusieurs mécanismes :

  Combinaison d'apprenants faibles : En combinant les prédictions de plusieurs apprenants faibles, AdaBoost peut tirer parti de leurs forces individuelles et minimiser leurs faiblesses.

  Pondération des apprenants : AdaBoost accorde plus de poids aux apprenants faibles qui ont une meilleure performance, ce qui permet de maximiser l'impact des prédictions les plus fiables.

  Focalisation sur les erreurs : AdaBoost se concentre sur les exemples mal classés par les apprenants précédents, ce qui permet d'améliorer la précision sur les cas les plus difficiles.

  Illustration :

  Supposons que nous avons 3 apprenants faibles avec des précisions de 60%, 70% et 80%. En les combinant avec AdaBoost, nous pouvons obtenir une précision globale de 90%.

  - **Polyvalence**

  AdaBoost est un algorithme flexible qui peut fonctionner avec différents types d'apprenants faibles, tels que :

  Arbres de décision : Des arbres de décision simples peuvent être utilisés comme apprenants faibles dans AdaBoost.

Régression linéaire : La régression linéaire peut également être utilisée comme apprenant faible dans AdaBoost.

K-plus proches voisins : L'algorithme K-plus proches voisins peut également être utilisé comme apprenant faible dans AdaBoost.

Cette flexibilité permet d'adapter AdaBoost à une grande variété de problèmes d'apprentissage automatique, tels que :

- ✓ **Classification d'images :** AdaBoost peut être utilisé pour classer des images en différentes catégories.
- ✓ **Détection d'anomalie : AdaBoost peut être utilisé pour détecter des anomalies dans les données.**
- ✓ **Apprentissage par renforcement :** AdaBoost peut être utilisé pour entraîner des agents intelligents dans des environnements complexes.

- **Robustesse au sur ajustement**

Le sur ajustement est un problème courant en apprentissage automatique où le modèle fonctionne bien sur les données d'entraînement mais mal sur les données invisibles. AdaBoost est relativement robuste au sur ajustement grâce à plusieurs mécanismes :

Concentration sur les erreurs : En se concentrant sur les exemples mal classés, AdaBoost encourage les apprenants faibles à se concentrer sur les aspects les plus importants des données.

Régularisation : AdaBoost peut être utilisé avec des techniques de régularisation pour limiter la complexité du modèle et réduire le risque de surajustement.

Arrêt prématuré : L'entraînement d'AdaBoost peut être arrêté avant que le sur ajustement ne se produise.

Interopérabilité : Les prédictions d'AdaBoost peuvent être interprétées en analysant les contributions des différents apprenants faibles.

Efficacité : AdaBoost peut être implémenté de manière efficace et s'exécute rapidement sur des ordinateurs modernes.

- ➢ **Limites d'AdaBoost**
  - **Sensibilité au bruit**

AdaBoost est sensible au bruit dans les données d'apprentissage. Le bruit peut provenir de différentes sources, telles que des erreurs de mesure, des valeurs aberrantes ou des données manquantes. Lorsque les données d'apprentissage sont bruyantes, les performances d'AdaBoost peuvent être dégradées pour plusieurs raisons :

Focus sur les exemples erronés : AdaBoost se concentre sur les exemples mal classés par les apprenants précédents. Si les données sont bruyantes, il est possible que de nombreux exemples soient mal classés par erreur, ce qui peut induire AdaBoost en erreur.

Amplification du bruit : Le processus d'apprentissage d'AdaBoost peut amplifier le bruit dans les données. En effet, les exemples mal classés reçoivent un poids plus important, ce qui peut propager l'erreur à d'autres exemples.

Illustration :

Supposons que nous avons un ensemble de données avec 100 exemples, dont 90 sont correctement étiquetés et 10 sont mal étiquetés. Si AdaBoost est utilisé avec cet ensemble de données, il est possible qu'il se concentre sur les 10 exemples mal étiquetés et produise des prédictions erronées.

Solutions :

Nettoyage des données : Il est important de nettoyer les données d'apprentissage et de supprimer les sources de bruit avant d'utiliser AdaBoost.

Techniques de robustesse : Il existe des techniques de robustesse qui peuvent être utilisées pour réduire la sensibilité d'AdaBoost au bruit, telles que l'échantillonnage aléatoire et la régularisation.

- **Complexité de calcul**

AdaBoost peut être plus complexe à calculer que d'autres algorithmes d'apprentissage automatique. Cela est dû au fait qu'il doit entraîner plusieurs apprenants faibles et combiner leurs prédictions. La complexité de calcul d'AdaBoost dépend de plusieurs facteurs :

Nombre d'apprenants faibles : Le nombre d'apprenants faibles utilisés par AdaBoost peut avoir un impact important sur la complexité de calcul. Plus le nombre d'apprenants est élevé, plus la complexité est grande.

Type d'apprenants faibles : Le type d'apprenants faibles utilisés par AdaBoost peut également influencer la complexité de calcul. Certains types d'apprenants, tels que les arbres de décision, peuvent être plus complexes à calculer que d'autres.

Solutions :

Implémentations optimisées : Il existe des implémentations optimisées d'AdaBoost qui peuvent améliorer la performance de l'algorithme.

Approximations : Il existe des techniques d'approximation qui peuvent réduire la complexité de calcul d'AdaBoost sans affecter significativement ses performances.

- **Difficulté d'adaptation des hyper paramètres**

AdaBoost possède plusieurs hyper paramètres qui peuvent influencer ses performances, tels que :

Nombre d'apprenants faibles : Le nombre d'apprenants faibles utilisés par AdaBoost est un hyper paramètre important qui doit être ajusté en fonction du problème à résoudre.

Taux d'apprentissage : Le taux d'apprentissage contrôle la vitesse à laquelle les poids des exemples sont mis à jour. Un taux d'apprentissage trop élevé peut entraîner un sur ajustement, tandis qu'un taux d'apprentissage trop faible peut ralentir l'apprentissage.

Fonction de perte : La fonction de perte utilisée pour évaluer les performances des apprenants faibles est un autre hyper paramètre important.

La recherche des valeurs optimales de ces hyper paramètres peut-être difficile et prendre du temps. Il n'y a pas de solution unique pour tous les problèmes, et la meilleure configuration d'hyper paramètres peut varier en fonction du type de données et du problème à résoudre.

Solutions :

Validation croisée : La validation croisée est une technique qui peut être utilisée pour trouver les valeurs optimales des hyper paramètres d'AdaBoost.

Recherche d'hyper paramètres : Il existe des outils et des bibliothèques qui peuvent automatiser la recherche des hyper paramètres d'AdaBoost.

- **Risque de sur ajustement**

Bien que AdaBoost soit relativement robuste au sur ajustement, il n'est pas garanti d'y être insensible. Le sur ajustement peut se produire si les hyper paramètres ne sont pas bien ajustés ou si l'algorithme est entraîné trop longtemps. Lorsque AdaBoost s'adapte aux particularités des données d'apprentissage, il peut ne pas généraliser correctement aux nouvelles données.

Solutions :

Validation croisée : La validation croisée est une technique qui peut être utilisée pour détecter le sur ajustement et choisir le modèle qui généralise le mieux aux nouvelles données. Elle consiste à diviser l'ensemble de données en plusieurs parties et à entraîner le modèle sur chaque partie en utilisant les autres parties comme données de test. La performance du modèle est ensuite évaluée en moyenne sur toutes les parties.

Arrêt prématuré : L'arrêt prématuré est une technique qui consiste à arrêter l'entraînement du modèle avant qu'il ne commence à s'adapter aux particularités des données d'apprentissage. Cela peut être réalisé en surveillant la performance du modèle sur un ensemble de données de validation et en arrêtant l'entraînement lorsque la performance commence à diminuer.

Régularisation : La régularisation est une technique qui consiste à ajouter des contraintes au processus d'apprentissage afin de limiter la complexité du modèle. Cela peut aider à réduire le risque de sur ajustement.

- **Early stopping (Arrêt prématuré) :**

Principe : Surveiller l'erreur sur un ensemble de validation indépendant et arrêter l'apprentissage lorsque l'erreur commence à augmenter.

Avantages : Réduit le risque de sur ajustement et permet d'obtenir un modèle plus généralisable.

Inconvénients : Nécessite un ensemble de validation indépendant et peut être difficile à implémenter.

- **Régularisation :**

Principe : Pénaliser les modèles complexes et favoriser les modèles plus simples.

Techniques : L1/L2 regularization, dropout, etc.

Avantages : Réduit le risque de surajustement et améliore la performance du modèle.

Inconvénients : Peut légèrement réduire la précision du modèle.

En plus de ces techniques, il est important de choisir judicieusement les hyper paramètres d'AdaBoost, tels que le nombre d'apprenants faibles et le taux d'apprentissage

- **AdaBoost peut être sensible à la sélection des caractéristiques.**
- **AdaBoost peut ne pas être performant sur certains types de problèmes, tels que les problèmes avec des classes très déséquilibrées.**

➢ **Apprentissage par ensemble avec AdaBoost**

AdaBoost, ou "Adaptive Boosting", est un algorithme d'apprentissage automatique puissant qui s'appuie sur la technique d'apprentissage par ensemble. L'idée est de combiner plusieurs "apprenants faibles" (des modèles simples avec une précision limitée) pour créer un "apprenant fort" plus performant.

➢ **Conception**

1. **Initialisation :** AdaBoost affecte des poids égaux à chaque instance de l'ensemble d'apprentissage.
2. **Apprentissage d'un apprenant faible :** Un apprenant faible est ensuite entraîné sur l'ensemble d'apprentissage avec les poids actuels.
3. **Évaluation de l'apprenant faible :** La performance de l'apprenant faible est ensuite évaluée.
4. **Mise à jour des poids :** Les poids des instances mal classées sont augmentés, tandis que ceux des instances bien classées sont diminués.
5. **Répétition des étapes 2 à 4 :** Les étapes 2 à 4 sont répétées plusieurs fois, en générant et en intégrant de nouveaux apprenants faibles.
6. **Combinaison des apprenants faibles :** Les prédictions des apprenants faibles sont ensuite combinées pour obtenir une prédiction finale.

➢ **Classification**

AdaBoost est souvent utilisé pour la classification, où il vise à prédire la classe d'une instance donnée. La combinaison des prédictions des apprenants faibles peut se faire par vote majoritaire ou par pondération en fonction de leur performance. AdaBoost est souvent utilisé pour la classification, où il vise à prédire la classe d'une instance donnée. La combinaison des prédictions des apprenants faibles peut se faire par vote majoritaire ou par pondération en fonction de leur performance.

Classification avec AdaBoost

Combinaison des prédictions

Lorsque AdaBoost est utilisé pour la classification, il existe plusieurs options pour combiner les prédictions des apprenants faibles :

Vote majoritaire : C'est la méthode la plus simple. Chaque apprenant faible vote pour la classe qu'il prédit, et la classe avec le plus grand nombre de votes est la classe finale prédite.

Pondération : Cette méthode prend en compte la performance de chaque apprenant faible. Les votes des apprenants faibles avec une meilleure performance sont pondérés plus fortement.

Exemple

Supposons que nous avons un ensemble d'apprentissage de 10 instances, dont 6 appartiennent à la classe A et 4 à la classe B. Nous entraînons 3 apprenants faibles sur cet ensemble d’apprentissage :

Apprenant 1 : Prédit correctement 7 instances (6 A, 1 B).

Apprenant 2 : Prédit correctement 8 instances (5 A, 3 B).

Apprenant 3 : Prédit correctement 9 instances (6 A, 3 B).

Vote majoritaire :

Apprenant 1 : A

Apprenant 2 : A

Apprenant 3 : A

Classe finale prédite : A

Pondération :

Apprenant 1 : Poids 0.7 (7/10)

Apprenant 2 : Poids 0.8 (8/10)

Apprenant 3 : Poids 0.9 (9/10)

Classe finale prédite : A (avec une probabilité plus élevée due au poids plus important de l'apprenant 3)

- **Avantages de la classification avec AdaBoost**

Amélioration de la précision : AdaBoost peut améliorer significativement la précision de la classification par rapport à un seul apprenant faible.

Robustesse au sur ajustement : AdaBoost est moins sensible au surajustement que d'autres algorithmes d'apprentissage automatique.

Flexibilité : AdaBoost peut être utilisé avec différents types d'apprenants faibles.

- **Limites de la classification avec AdaBoost**

Sensibilité au bruit : AdaBoost peut être sensible au bruit dans les données d'apprentissage.

Complexité de calcul : AdaBoost peut être plus complexe à calculer que d'autres algorithmes d'apprentissage automatique.

- **Régression**

AdaBoost peut également être utilisé pour la régression, où il vise à prédire une valeur numérique pour une instance donnée. Dans ce cas, la combinaison des prédictions des apprenants faibles se fait généralement par moyenne pondérée.

Régression avec AdaBoost

Combinaison des prédictions

Lorsque AdaBoost est utilisé pour la régression, la combinaison des prédictions des apprenants faibles se fait généralement par moyenne pondérée :

Prédiction finale = Σ (poids_i * prédiction_i) / Σ poids_i

Poids i est le poids de l'apprenant faible i

Prédiction i est la prédiction de l'apprenant faible i

Exemple

Supposons que nous avons un ensemble d'apprentissage de 10 instances avec des valeurs cibles comprises entre 0 et 100. Nous entraînons 3 apprenants faibles sur cet ensemble d’apprentissage :

Apprenant 1 : Prédit une valeur moyenne de 80 avec une erreur quadratique moyenne (MSE) de 10.

Apprenant 2 : Prédit une valeur moyenne de 85 avec une MSE de 5.

Apprenant 3 : Prédit une valeur moyenne de 90 avec une MSE de 2.

Pondération :

Apprenant 1 : Poids 0.2 (1/5)

Apprenant 2 : Poids 0.4 (2/5)

Apprenant 3 : Poids 0.4 (2/5)

Prédiction finale :

(0.2 * 80 + 0.4 * 85 + 0.4 * 90) / (0.2 + 0.4 + 0.4) = 86.67

- **Avantages de la régression avec AdaBoost**

Amélioration de la précision : AdaBoost peut améliorer significativement la précision de la régression par rapport à un seul apprenant faible.

Robustesse au sur ajustement : AdaBoost est moins sensible au surajustement que d'autres algorithmes d'apprentissage automatique.

Flexibilité : AdaBoost peut être utilisé avec différents types d'apprenants faibles.

- **Limites de la régression avec AdaBoost**

Sensibilité au bruit : AdaBoost peut être sensible au bruit dans les données d'apprentissage.

Complexité de calcul : AdaBoost peut être plus complexe à calculer que d'autres algorithmes d'apprentissage automatique.

AdaBoost : L'Adaptive Boosting pour la Classification et la Régression

CONCLUSION

AdaBoost est une technique puissante d'apprentissage automatique utilisée pour améliorer la performance des classifieurs et des régresseurs faibles. Il est largement utilisé dans les tâches de classification et de régression pour sa capacité à améliorer la précision des prédictions. Cependant, il présente des limitations, notamment sa sensibilité aux outliers et sa complexité de construction du modèle. Il est important de prendre ces facteurs en compte lors de l'utilisation d'AdaBoost dans des applications réelles.