Arbre de décision (ID3)

Fiche de compréhension

Supervisé par :
Dr. MESSI

Faculté des Sciences

Nom :	**DJIEMBOU TIENTCHEU VICTOR NICO**
Devoir :	4
Auteur :	Qi Meng, Guolin Ke et Al.
Mots clés :	Distributed and parallel ML, Decision Tree
UE :	INF5099
Matricule :	17T2051
Département :	Informatique
Niveau - Option :	M2 - Sciences des Données

Table des matières

1 Contexte

L'algorithme d'arbre de décision est très répandu de par le monde pour ses règles simplistes et son interprétabilité. Ainsi de nombre nombreuses personnes ont conçu des algorithmes sur cette base à l'instar de Random Forest (RF) ou encore Gradient Boosting Decision Tree (GBDT). Cependant, avec la vogue des technologies et la digitalisation de systèmes, les jeux de données sont devenus très volumineux ainsi la tâche d'apprentissage automatique en a pris un coup.

Comment alleger cette lourde tâche qu'est devenu l'entrainement de modèles d'arbre de décision a donc été un point important pour les chercheurs. Deux grandes approches ont été proposé pour cette fin. notamment, la **parallélisation des attributs** et **parallélisation des données**.

2 Problème

Plus le temps passe, plus les données asssociées à des problèmes gonflent et plus l'entrainement des modèles d'apprentissage automatique en particulier ceux des arbres décision deviennent difficile et coûteux. Le problème qui s'élève alors est celle de trouver un moyen de pouvoir paralléliser cette algorithme afin de réduire le temps et l'espace nécessaire pour méner un entrainement tout en garantissant d'améliorer l'exactitude de ses modèles face aux problèmes.

3 motivations

Les axes de motivations sont :
— comment entrainer sur différentes travailleurs en parallèle ?
— comment assurer la communication synchrone entre ses différents machine ?
— comment malgré l'indépendance des entrainement assurer l'exactitude du modèle final ?

4 Arbre de décision : vue générale

Il existe trois principaux algorithmes d'arbre de décision : `ID3(Iterative Dichotomiser 3)`, `CART`, `C4.5`.

ID3 (Iterative Dichotomiser 3) est l'un des premiers algorithmes d'apprentissage automatique pour la construction d'arbres de décision. Il utilise la mesure de l'entropie pour évaluer la pureté des partitions et choisir les attributs de division. ID3 ne prend pas en charge les attributs numériques et peut être sensible aux données manquantes.

CART (Classification and Regression Trees) est un algorithme qui peut être utilisé pour construire des arbres de décision à la fois pour la classification et la régression. Contrairement à ID3, CART utilise le critère de Gini pour mesurer l'impureté des partitions et choisir les attributs de division. Il peut gérer les attributs numériques et gérer les données manquantes en utilisant des techniques telles que l'imputation.

C4.5 est une extension de l'algorithme ID3 qui a été développée pour surmonter certaines de ses limitations. Il utilise également l'entropie ou le ratio gain pour évaluer la qualité des attributs, mais il peut gérer les attributs numériques en effectuant une discrétisation. De plus, C4.5 peut traiter les données manquantes en utilisant des méthodes d'imputation ou en attribuant des poids aux instances manquantes.

5 ID3 (Iterative Dichotomiser 3)

5.1 Version recursive

Algorithme : ID3 Recursive
Données : D la matrice de données
Debut

1. Calculer de l'entropie de D en utilisant la formule de l'entropie de Shannon :

$$Entropie(D) = -\sum_{i=1}^{n} P_i \log_2 P_i$$

où :
— $Entropie(D)$ est l'entropie du jeu de données D.
— P_i est la proportion d'exemples dans D appartenant à la classe i i .
— n représente le nombre total de classe.

2. Pour chaque attribut calculer le gain d'information en utiisant la formule :

$$Gain(A) = Entropie(D) - \sum_{v \in valeurs(A)} \frac{|D_v|}{|S|} \cdot Entropie(D_v)$$

où :
— $Gain(A)$ représente le gain d'information en choisissant l'attribut A pour diviser l'ensemble d'exemples.
— $Entropie(D)$ est l'entropie de l'ensemble d'exemples D.
— v parcourt les valeurs possibles de l'attribut A.
— $|D_v|$ est le nombre d'exemples dans D ayant la valeur v pour l'attribut A.
— $|D|$ est le nombre total d'exemples dans D.
— $Entropie(D_v)$ est l'entropie de l'ensemble d'exemples ayant la valeur v pour l'attribut A.

3. Slectionner l'attribut A qui le gain d'information maximal :

$$A_{best} = \max_{A \in attributs} Gain(A)$$

où :
— A_{best} représente le meilleur gain d'information.
— A parcourt tous les attributs disponibles.
— *attributs* est l'ensemble des attributs.
— $Gain(A)$ est le gain d'information pour l'attribut A calculé selon la formule précédente.

4. Créer un noued de décision correspondant à A_{best}
5. Pour chaque valeur dans A_{best}, créer un sous ensemble de D_v contenant les exemples de D ayant la valeur v pour l'attribut A_{best}
6. Pour chaque sous ensemble D_v, repeter les étapes 1 à 5 récursivement jusqu'à atteindre un critère d'arrêt, comme l'absence d'attributs restant où l'homogénéité des exemples.
7. Si tous les exemples dans un sous-ensemble D_v appartiennent à la même classe, créer une feuille correspondate à cette classe.

Fin

5.2 Version itérative

La version récursive présentante est la forme originelle dans laquelle ID3 à été proposé mais nous ne sauront nous servir de cette forme pour mener une analyse à des fins de parallélisation. Il à dont été d'une importance de dérecursiver ce dernier. Après une fine analyse du procedé, nous avons pu avoir le pseudo algorithme itératif suivant.

Algorithme : ID3 Itératif
Données : D la matrice de données
Debut

1. Créer une pile vide
2. Créer un noeud de décision racine contenant D et ajouter à la pile
3. tant que la pile n'est pas vide faire
   (a) Retirer le noeud de décision au sommet de la pile
   (b) Si tous les exemples dans le noeud de décision appartiennent à la même classe, créer une feuille correspondante à la classe et l'associer au noeud de décision
   (c) Sinon
   — choisir l'attribut A qui a le gain d'information maximal pour le noeud de décision
   — créer un noeud de décision avec comme étiquette A et l'associer au noeud racine
   — créer des noeuds de décision enfants correspondant à chaque valeur possible A_i de l'attribut et contenant un sous ensemble résultat du partitionnement des exemples du tableau présent dans le noeud de décision tels que $A = A_i$
   — ajouter les noeuds de décision enfants à la pile
4. retourner l'arbre contruit

Fin

5.3 Evaluation de la complexité de l'algorithme

La complexité de chaque grande tâche de l'algorithme ID3 peut être décomposée comme suit :

Sélection de l'attribut : La complexité de cette tâche dépend du nombre d'attributs A et du nombre d'exemples d'entraînement N. Pour chaque attribut, il faut calculer le gain d'information ou le ratio de gain, ce qui nécessite de parcourir les exemples pour calculer les distributions de classe. Par conséquent, la complexité de cette tâche est $O(A \cdot N)$.

Division des exemples : La complexité de cette tâche dépend du nombre d'exemples d'entraînement N et de la dimensionnalité des attributs A. L'algorithme doit parcourir chaque exemple et effectuer une comparaison avec la valeur de l'attribut sélectionné. Par conséquent, la complexité de cette tâche est $O(A \cdot N)$.

Répétition récursive : La répétition récursive de l'algorithme dépend de la structure de l'arbre de décision résultant. Dans le pire des cas, où chaque nœud contient un seul exemple différent, la complexité de cette tâche est $O(N)$. Cependant, dans la plupart des cas, la complexité est inférieure à $O(N)$.

La complexité totale de l'algorithme ID3 dépend du nombre d'attributs A, du nombre d'exemples d'entraînement N et de la structure de l'arbre de décision résultant. Elle peut être approximée par la somme des complexités des tâches mentionnées ci-dessus, soit $O(A \cdot N)$ dans la plupart des cas.

5.4 Tâches les plus gourmandes et parallélisation

Les deux tâches les plus gourmandes en termes de calcul dans l'algorithme ID3 sont la sélection de l'attribut et la division des exemples. Ces tâches impliquent de parcourir tous les exemples d'entraînement et peuvent bénéficier d'une parallélisation pour accélérer le processus.

Sur la base de ce que nous avons déjà fait en cours, la parallélisation de ces tâches peut être réalisée en utilisant des techniques telles que le traitement parallèle des données ou le traitement parallèle des tâches.

Toutefois, avec mon humble niveau, je trouve qu'avec la parallélisation de données, il va falloir une forte communication des travaillleurs indépendants pour garantir de ne pas perdre les performance de l'algorithme.

Alors, moi je pense plus à une parallélisation des tâches en se sens que je vais me permettre de lancer des tâches comme le calcul de gain et la séparation des exemples sur différents travailleurs de façon paralléle. Avec cette approche je pourais atteindre un speedup theorique égale à $O(\frac{A}{N_{thread}} \cdot N)$ pour ses différentes tâches.

6 Conclusion

ID3 est un algorithme à instructions fortement dépendantes produite à l'origine sous la forme recursive. La version Itérative proposé dans ce rapport est un proposition suite à des analyse de la version récursive. Il ressort que cette algorithme en version itérative est fortement sensible à la population des exemples et au nombre de variables descriptif contenant dans ses exemples. Il devient très évidement primordial de proposer une version d'algorithme parallèle capable de pouvoir veuille atténuer l'impact des gros volume de données sur l'execution de l'algorithme. La pensée a été de paralléliser les étapes de séparation d'exemples et calcul de gain d'information car étaient les plus gourmandes.