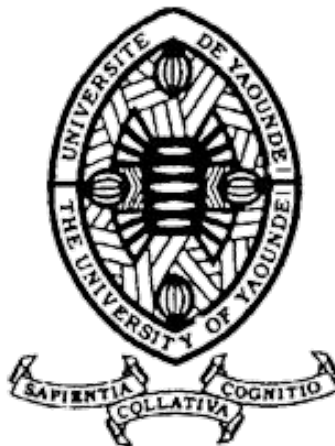


A Communication-Efficient Parallel Algorithm for Decision Tree

Fiche de lecture

Supervisé par :
Dr. MESSI



Faculté des Sciences

| | |
|-------------------|--|
| Nom : | DJIEMBOU TIENTCHEU VICTOR NICO |
| Devoir : | 2 |
| Publié à : | 30th Conference on NIPS 2016 |
| 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 | 3 |
| 2 | Problème | 3 |
| 3 | motivations | 3 |
| 4 | Arbre de décision : vue générale | 3 |
| 5 | PV-Tree | 4 |
| 6 | Conclusion | 5 |

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 alléger cette lourde tâche qu'est devenu l'entraînement 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**.

La parallélisation des attributs pense à répartir des sous ensembles distincts de données séparés verticalement sur différentes machines et ainsi à chaque itérations, elles chercherons à trouver le meilleur attribut et la bonne règle de séparation. Toutefois, la communication entre ses machines est très couteuse car il faut toujours récupérer les informations sur les meilleurs attributs dans chaque machine pour pouvoir poursuivre la repartition à l'itérations suivantes or cette tâche de repartition en elle même n'est pas parallèle.

La parallélisation des données pense à répartir des sous ensembles distincts de données séparés horizontalement sur différentes machines. Il y'a cependant le besoin de partager un histogramme de chaque attribut sur la base du sous ensembles de données à toutes les autres machines mutuellement afin de synchroniser les résultats pour récupérer le meilleur attribut global et son critère de séparation des données. Il est claire que cette communication est très couteuse.

Donc ayant prise connaissance de ses différentes réalités, les auteurs Qi Meng, Guolin Ke et Al. s'inscrivent dans le cadre de pouvoir améliorer la parallélisation des données en proposant un modèle appelé **Parallel Voting Decision tree (PV-Tree)** qui offrira un meilleur équilibre dans la communication entre machine et garantira une meilleur exactitude du modèle généré.

2 Problème

Plus le temps passe, plus les données associées à des problèmes gonflent et plus l'entraînement 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 mener un entraînement tout en garantissant d'améliorer l'exactitude de ses modèles face aux problèmes.

3 motivations

Les axes de motivations sont :

- comment entrainer sur différentes machines en parallèle ?
- comment assurer la communication synchrone entre ses différents machine ?
- comment malgré l'indépendance des entraînement assurer l'exactitude du modèle final ?

4 Arbre de décision : vue générale

La construction d'un Arbre de décision est une opération récursive de la **recherche du meilleur attribut qui sépare les données** et de **son critère de séparation** (voir Figure 1). Le critère ou encore le seuillage est différemment définit par **le gain d'information en problèmes de classification (Indice de Gini)** ou bien encore par **La variance d'information en problème de regression**.

Algorithm 1 BulidTree

```
Input: Node N, Dateset D
if StoppingCirteria(D) then
    N.output = Prediction(D)
else
    bestSplit = FindBestSplit(D)
    (DL, DR) = Split(D, N, bestSplit)
    BuildTree(N.leftChild, DL)
    BuildTree(N.rightChild, DR)
end if
```

FIGURE 1 – Algorithmme de construction d'un Arbre de Décision

5 PV-Tree

La différence majeur à noter entre les approches de parallélisation de la données conventionnelles et l'approche PV-Tree est que le modèle ne fais confiance qu'à l'histogramme d'information global. A cette fin, le présent algorithmes connait 3 trois grandes phases :

1. Vote Local : ici sur chaque machine, on sélectionne les **k premiers attributs** basé sur les données locaux et leur information de gain ou de reduction, dépendenment que nous soyons dans classification respectivement regression. puis on communique les indices de ces meilleurs aux autres machines.
2. Vote Global : Ici, on range les attributs suivant le fait que plusieurs machines l'aillent choisi comme bonne attribut (vote majoritaire) et ainsi on ne conserve que le top $2k$ après le rangement.
3. Identification du meilleur attribut : sur la la base des histogrammes des top $2k$ attributs, on sélectionne le meilleur attribut et sa condition de séparation tel qu'il soit celui possédant le meilleur gains d'information.

Algorithm 2 FindBestSplit

```
Input: DataSet D
for all X in D.Attribute do
    ▷ Construct Histogram
    H = new Histogram()
    for all x in X do
        H.binAt(x.bin).Put(x.label)
    end for
    ▷ Find Best Split
    leftSum = new HistogramSum()
    for all bin in H do
        leftSum = leftSum + H.binAt(bin)
        rightSum = H.AllSum - leftSum
        split.gain = CalSplitGain(leftSum, rightSum)
        bestSplit = ChoiceBetterOne(split, bestSplit)
    end for
end for
return bestSplit
```

Algorithm 3 PV-Tree_FindBestSplit

```
Input: Dataset D
localHistograms = ConstructHistograms(D)
▷ Local Voting
splits = []
for all H in localHistograms do
    splits.Push(H.FindBestSplit())
end for
localTop = splits.TopKByGain(K)
▷ Gather all candidates
allCandidates = AllGather(localTop)
▷ Global Voting
globalTop = allCandidates.TopKByMajority(2*K)
▷ Merge global histograms
globalHistograms = Gather(globalTop, localHistograms)
bestSplit = globalHistograms.FindBestSplit()
return bestSplit
```

FIGURE 2 – Algorithmmes PV-Tree

6 Conclusion

La parallélisation des modèles d'arbre de décision peut se faire de différente façon, notamment la parallélisation des attributs et la parallélisation des données. Les avancées les plus importantes se sont fait au niveau de la parallélisation des attributs où des nombreux auteurs ont pu proposer des frameworks qui permettant d'exécution dans un environnement distribué une version parallèle de l'algorithme d'Arbre de décision. De façon empirique, l'approche de parallélisation des attributs se veut de proposer de meilleurs speedup et métriques d'entraînement.

Références

- [BHTT10] Yael Ben-Haim and Elad Tom-Tov, *A streaming parallel decision tree algorithm.*, Journal of Machine Learning Research **11** (2010), no. 2.
- [JA03] Ruoming Jin and Gagan Agrawal, *Communication and memory efficient parallel decision tree construction*, Proceedings of the 2003 SIAM international conference on data mining, SIAM, 2003, pp. 119–129.
- [JKK98] Mahesh V Joshi, George Karypis, and Vipin Kumar, *Scalparc : A new scalable and efficient parallel classification algorithm for mining large datasets*, Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing, IEEE, 1998, pp. 573–579.
- [Kuf97] Richard Kufrin, *Decision trees on parallel processors*, 1997.
- [MKW⁺16] Qi Meng, Guolin Ke, Taifeng Wang, Wei Chen, Qiwei Ye, Zhi-Ming Ma, and Tie-Yan Liu, *A communication-efficient parallel algorithm for decision tree*, Advances in Neural Information Processing Systems **29** (2016).
- [PHBB09] Biswanath Panda, Joshua S Herbach, Sugato Basu, and Roberto J Bayardo, *Planet : massively parallel learning of tree ensembles with mapreduce*.
- [SAM⁺96] John Shafer, Rakesh Agrawal, Manish Mehta, et al., *Sprint : A scalable parallel classifier for data mining*, Vldb, vol. 96, Citeseer, 1996, pp. 544–555.
- [SB11] Krysta M Svore and CJ Burges, *Large-scale learning to rank using boosted decision trees*, Scaling Up Machine Learning : Parallel and Distributed Approaches **2** (2011), 2011.
- [TWAP11] Stephen Tyree, Kilian Q Weinberger, Kunal Agrawal, and Jennifer Paykin, *Parallel boosted regression trees for web search ranking*, Proceedings of the 20th international conference on World wide web, 2011, pp. 387–396.