Roadmap Julia

Bérénice-Alexia Jocteur

Université Claude Bernard Lyon 1

1er février 2022

Sommaire

Utilisation du vrai critère de split plutôt que des pseudo outcomes :

$$\frac{N_L N_R}{N_P^2}\left(\frac{\sum_L (W_i - \bar{W}_L)(Y_i - \bar{Y}_L)}{\sum_L (W_i - \bar{W}_L)^2} - \frac{\sum_R (W_i - \bar{W}_R)(Y_i - \bar{Y}_R)}{\sum_R (W_i - \bar{W}_R)^2}\right)$$

En effet on peut faire la même astuce rsum et lsum actualisés à la fois pour Y et W. Pas de gain de rapidité à utiliser les ρ_i dans ce cas.

Procedure 1 Code used in GRF paper : S = data, x = test point

$\alpha = zeros(|S|)$
for $b = 1$ to B **do**
  $I = Subsample(S, s)$ où s hyperparamètre proportion de la taille initiale
  $I_1, I_2 = SplitSample(I)$ divise I en deux parties égales sans chevauchement
  $T = BuildTree(I_1)$ Construit arbre avec les Specification 1 : proportion de treat et ctrl dans chaque child + proba de split sur feature donnée > 0 voir next slide implem
  $N = Neighbours(x, T, I_2)$ Pas be soin de recalculer T à chaque x : faire 2 fonctions buildforest(data) et predict(forest, setI2 ,x)
  for $e \in N$ **do**
    $\alpha[e] += \frac{1}{|N|}$
  end for
end for
return $\hat{\theta}(x) = \frac{\sum_{i=1}^{n} \alpha_i(x)(W_i - \bar{W}_\alpha)(Y_i - \bar{Y}_\alpha)}{\sum_{i=1}^{n} \alpha_i(x)(W_i - \bar{W}_\alpha)^2}$ où $\alpha = \frac{\alpha}{B}$ et $W_\alpha = \sum \alpha_i(x) W_i$

SPECIFICATION 1. All trees are symmetric, in that their output is invariant to permuting the indices of training examples; make balanced splits, in the sense that every split puts at least a fraction ω of the observations in the parent node into each child, for some $\omega > 0$; and are randomized in such a way that, at every split, the probability that the tree splits on the j-th feature is bounded from below by some $\pi > 0$. The forest is honest and built via subsampling with subsample size s satisfying $s/n \to 0$ and $s \to \infty$, as described in Section 2.4.

Respect de Specification 1 de GRF :

- Proba de choisir une feature donnée soit $\pi > 0$. On utilise la stratégie suivante :
  1. Le nombre de features testé pour un split donné (mtry) est random : $h = \min(\max(Poisson(m), 1), p)$ où m hyperparameter fixé par defaut à $\min(\sqrt{p} + 20, p)$.
  2. Tirer h features et splitter sur celles-ci.
- Avoir des noeuds équilibrés entre chaque groupe : avoir minnodesize treat et control dans chaque noeud. Possibilité de rajouter condition pour avoir au moins un proportion w de treat et ctrl dans chaque noeud. Ce n'est pas implémenté par défaut dans GRF, mais possibilité de le faire pour remplir Specification 1.

Bootstrap

Changer le subsampling par du bootstraping.
I = Bootstrap(S)
Question : Honesty toujours pertinente ? ? Si je bootstrap et je divise en 2 j'aurai peut être un même élément dans I1 et I2 : faire un split spécial bootstrap qui fait attention à cela ? Mais on perd peut être un peu l'intérêt du bootstrap ? ?
Dans le cas bootstrap il faudra aussi modifier l'expression de α : dans la la boucle il faudrait multiplier par le nombre de fois que l'item à été tiré par bootstrap (comme dans l'article cond rf)

No honesty

Enlever l'hypothèse d'honesty
Ca simplifie
Initialement doutes sur nécessité d'une telle hypothèse.

Misc

- Finalement ça ferait 4 cas avec 2 facteurs discriminants : Subsampling vs Bootstrap ET Honesty vs No-honesty
- Supprimer les poids (appelés W dans le code julia de base)
- Considérer que traitement binaire pour le moment : on fait les preuves sur ca et c'est plus compliqué pour certaines choses (condition sur nb de treat et de ctrl)

Notes pour moi

- load_data.jl useless
- measures.jl surement useless aussi ? ?
- scikitlearnAPI.jl c'est pour utiliser les fonctionnalités de scikit learn julia : pipeline, gridsearch. A garder ? ?
- util.jl oui à garder et on pourra rajouter la nos fonctions techniques et surement faire du tri pour virer les func qui servent plus à rien + modifier check input etc...
- DecisionTree.jl voir pour virer integration sklearn
- garder que le dossier régression car c'est le seul qui fat sens pour forêt causale !
- Classification/tree.jl c'est la qu'on construit l'arbre ! !
- Git : je vais fork ou clone le DecisionTree.jl et travailler la dessus. Outils à installer : Julia, Juno, Gitkraken je déposerai sur mon github perso

Mail sur webmail ICJ -> contrat CIFRE pas sûre de quoi répondre à certaines questions