Active Coalition of Variable (ACV) optimized in C/C++

Salim Ibrahim Amoukou [1] & Nicolas Brunel [2]

[1] *Stellantis & Université Paris Saclay, CNRS, Laboratoire de Mathématiques et Modélisation d'Evry, salim.ibrahim-amoukou@universite-paris-saclay.fr*
[2] *Quantmetry & Université Paris Saclay, CNRS, ENSIIE Laboratoire de Mathématiques et Modélisation d'Evry, nicolas.brunel@ensiie.fr*

1 Goal

The objective of this work is to develop tools that could provide better insights in the important variables, at a global or at a local level, in Tree-based model.

- Compute properly Shapley values (SV)
- Compute Same Decision Probability (SDP)
- Compute various SV with different Value functions

The package directory:

```
ACV-master
 |__ notebook
 |__ docs
 |__ cext_shap
 |    |__ acv_tree.h
 |    |__ _cext.cc
 |__ acv_explainers
 |    |__ base_tree.py
 |    |__ acv_tree.py
 |    |__ py_acv.py
 |    |__ utils.py
 |__ experiments
 |__ requirements.txt
 |__ README.md
 |__ setup.py
```

- The main functions to be implemented in `acv_tree.h` are:

Listing 1: Functions to be optimize in C++

```
inline void leaves_partition(const TreeEnsemble& trees,
const ExplanationDataset &data) {}

inline void multi_game_sv(tfloat *out_contribs, const TreeEnsemble& trees
const ExplanationDataset &data) {}

inline void multi_game_sv_acv(tfloat *out_contribs, const TreeEnsemble& t

inline void cond_exp_tree(tfloat *out_contribs, const tfloat *x,
const int *S, const TreeEnsemble& trees, const ExplanationDataset &data)

inline void cond_sdp_clf(tfloat *out_contribs, const tfloat *x,
const int *S, const TreeEnsemble& trees, const ExplanationDataset &data)

inline void cond_sdp_reg(tfloat *out_contribs, const tfloat *x,
const int *S, const TreeEnsemble& trees, const ExplanationDataset &data)

inline void brute_force_tree_sv(tfloat *out_contribs, const tfloat *x,
tfloat (*value_func)(tfloat, int), const TreeEnsemble& trees,
const ExplanationDataset &data, const int *S_star, const int *N) {}

inline void swing_sv(tfloat *out_contribs, const tfloat *x,
tfloat (*value_func)(tfloat, int), const TreeEnsemble& trees,
const ExplanationDataset &data) {}

inline void local_sdp(tfloat *out_contribs, const tfloat *x,
const TreeEnsemble& trees, const ExplanationDataset &data) {}

inline void global_sdp_importance(tfloat *out_contribs, const tfloat *x,
const TreeEnsemble& trees, const ExplanationDataset &data) {}
```

- Interface between C++ and Python is implemented in `_cext.cc`
- Execution in Python for different models of trees is in `acv_tree.py`

2 Shapley values

We assume that we have a regression tree with M leafs $L_1, \dots, L_M$ based on the variables $X_1, \dots, X_p$ (continuous or qualitative), the function f can be a predictor or any function estimated with a tree.

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} f_m \mathbb{1}_{L_m}(\boldsymbol{x}).$$

For a regression, we have $f_m = \frac{1}{N(L_m)} \sum_{i \in L_m} z_i$ where $N(L_m)$ is the number of observations that fall in leaf L_m. We can have also the class probabilities for classification, or Cox regression, survival functions [1].

For any group of variables $\boldsymbol{X}_S = (X_i)_{i \in S}$, with any subset $S \subseteq [\![1, p]\!]$, we define the reduced predictors as

$$f_S(\boldsymbol{x}_S) \triangleq E\left[f\left(\boldsymbol{X}\right) | \boldsymbol{X}_S = \boldsymbol{x}_S\right]. \tag{2.1}$$

The SV for local interpretability at $\boldsymbol{x}$ are based on a cooperative game with the value function $v(f; S) \triangleq f_S(\boldsymbol{x}_S)$ (a value function is a function from 2^p set to $\mathbb{R}$). For any coalition of variables $C \subseteq [\![1, p]\!]$ and $k \in [\![1, p - |C|]\!]$, we denote the set $\mathcal{S}_k(C) = \{S \subseteq [\![1, p]\!] \setminus C | \, |S| = k\}$: the Shapley Value (SV) of the coalition C is defined as

$$\phi_C(f; \boldsymbol{x}) = \frac{1}{p - |C| + 1} \sum_{k=0}^{p-|C|} \frac{1}{\binom{p-|C|}{k}} \sum_{S \in \mathcal{S}_k(C)} \left(f_{S \cup C}(\boldsymbol{x}_{S \cup C}) - f_S(\boldsymbol{x}_S)\right)) \tag{2.2}$$

The definition (2.2) of the SV is a straightforward extension of the standard SV of a single variable (or player) to a group of variables. The standard SV is recovered with $C = \{i\}$ for $i \in [\![1, p]\!]$.

2.1 Estimation of the reduced predictor and SV

The reduced predictor of tree-based model f is : $f_S(\boldsymbol{x}_S) = \sum_{m=1}^{M} f_m P_X\left(L_m \left| \boldsymbol{X}_S = \boldsymbol{x}_S\right.\right)$ showing the main challenge is the computation of probability $P_X\left(L_m \left| \boldsymbol{X}_S = \boldsymbol{x}_S\right.\right)$. We approximate this probability with a Plug-In estimator:

$$P_X\left(L_m \left| \boldsymbol{X}_S = \boldsymbol{x}_S\right.\right) \approx \frac{N(L_m)}{N(L_m^S)}$$

where

- $N(L_m)$ is the number of observations in the leaf L_m
- $N(L_m^S)$ is the number of observations satisfying the conditions $\boldsymbol{x}_S \in L_m^S$ across all the leaves of the tree.

2.2 Estimation of SV

A straightforward algorithm for computing SV has a complexity $\mathcal{O}(p \times \texttt{tree-depth} \times 2^p)$ (called *Brute Force Algorithm*): we have p variables, 2^p groups of variables to consider each time, and we need to go down into the tree. Instead, we suggest to compute SV leaf by leaf thanks to equation (eq.4.5 in the supplementary materials of the ICML papers). In that case, the computation of the SV for the p variables is done by summing over M games (leafs), each of them having a number of variables $|S_m|$ lower than `tree-depth`. Consequently, the complexity is $\mathcal{O}(p \times M \times 2^{\texttt{tree-depth}})$ in worst cases.
The *Multi-Games algorithm* improves dramatically the computational complexity as `tree-depth` is often much lower than p. Moreover, the algorithm is linear in the number of observations where we want to compute the SV.

The algorithm is describes below, we use the following notations $N(L_m^{\emptyset}) = \sum_{m=1}^{M} N(L_m)$ and $\mathbb{1}_{L_m^{\emptyset}}(\boldsymbol{x}_{\emptyset}) = 1$ and is implemented in acv_tree.py\shap_values.

Algorithm 1: *Multi-Games Algorithm* - Compute SV in the worst case in $\mathcal{O}(p \times M \times 2^{\texttt{tree-depth}})$

```
Inputs: $\boldsymbol{x}, f(\boldsymbol{x}) = \sum_{m=0}^{M} f_m \mathbb{1}_{L_m}(\boldsymbol{x})$;
$p = length(\boldsymbol{x})$;
$\phi = zeros(p)$;
for $m = 1$ to $M$ do
    for $i$ in $[p]$ do
        if $i$ not in $S_m$ then
            continue ;                              /* skip to next variable */
        end
        for $S \subseteq S_m$ do
            $\phi[i] +=$
            $\left( \binom{p-1}{|S|}^{-1} + \sum_{k=1}^{p-|S_m|} \binom{p-1}{k+|S|}^{-1} \right) \left( \mathbb{1}_{L_m^{S\cup i}}(\boldsymbol{x}_{S\cup i}) \frac{N(L_m)}{N(L_m^{S\cup i})} - \mathbb{1}_{L_m^{S}}(\boldsymbol{x}_S) \frac{N(L_m)}{N(L_m^{S})} \right)$
        end
        end
    end
    return $\phi$
```

Remark 2.1. The algorithm can be vectorized in order to compute SV of several observations at the same time.

3 Same decision Probability

Definition 3.1. ***(Same Decision Probability of a classifier).*** *Let $f : \mathcal{X} \longrightarrow [0, 1]$ a probabilistic predictor and its classifier $C(\boldsymbol{x}) = \mathbb{1}_{f(\boldsymbol{x}) \geq T}$ with threshold T, the Same*

Decision Probability of coalition $S \subset [\![1, p]\!]$, w.r.t $\boldsymbol{x} = (\boldsymbol{x}_S, \boldsymbol{x}_{\bar{S}})$ is

$$SDP_S\left(C; \boldsymbol{x}\right) = P\left(C(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}) = C(\boldsymbol{x})\,|\boldsymbol{X}_S = \boldsymbol{x}_S\right)$$

SDP gives the probability to keep the same decision $C(\boldsymbol{x})$ when we do not observe the variables $\boldsymbol{X}_{\bar{S}}$. The higher is the probability, the better is the explanation based on S. Therefore, we want to identify the **minimal** subset of features such that the classifier makes the same decision with high probability π, given only them. More formally:

Definition 3.2. *(**Sufficient Coalition**). Given C a binary classifier, an observation $\boldsymbol{x} = (\boldsymbol{x}_S, \boldsymbol{x}_{\bar{S}})$, $S \triangleq S^\star_\pi(\boldsymbol{x})$ is a Sufficient Coalition for probability π if:*

1. *$SDP_{S^\star_\pi(\boldsymbol{x})}\left(C; \boldsymbol{x}\right) \geq \pi$*
2. *No subset Z of $S^\star_\pi(\boldsymbol{x})$ satisfies $SDP_Z\left(f; \boldsymbol{x}\right) \geq \pi$*

Proposition 3.1. *Let f a probabilistic predictor and its binary classifier C with threshold T, $Q_{S,\boldsymbol{x}}$ the law of $\boldsymbol{X}_{\bar{S}}|\boldsymbol{X}_S = \boldsymbol{x}_S$, then the $SDP_S\left(f; \boldsymbol{x}\right)$ can be written explicitly with the reduced predictor:*

$$SDP_{S,\boldsymbol{x}} = \frac{d - d^-}{d^+ - d^-} \tag{3.1}$$

Where $d = E_{Q_{S,\boldsymbol{x}}}\left[f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}})\right]$, $d^+ = E_{Q_{S,\boldsymbol{x}}}\left[f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}})|f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}) \geq T]\right]$ and
$d^- = E_{Q_{S,\boldsymbol{x}}}\left[f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}})|f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}) < T\right]$.

Proposition 3.2. *Let f be a regressor, $Q_{S,\boldsymbol{x}}$ the law $\boldsymbol{X}_{\bar{S}}|\boldsymbol{X}_S = \boldsymbol{x}_S$, then the $SDP_S\left(f; \boldsymbol{x}, t\right)$ can be written explicitly as:*

$$SDP_S\left(f; \boldsymbol{x}, t\right) = \frac{d^+ - d}{d^+ - d^-} \tag{3.2}$$

Where $d = E_{Q_{S,\boldsymbol{x}}}\left[d(f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}), f)\right]$, $d^+ = E_{Q_{S,\boldsymbol{x}}}\left[d(f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}), f) \mid d(f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}), f) > t\right]$ and $d^- = E_{Q_{S,\boldsymbol{x}}}\left[d(f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}), f) \mid d(f(\boldsymbol{x}_S, \boldsymbol{X}_{\bar{S}}), f) \leq t\right]$.

Proposition 3.3. *Let a tree function define as $f(\boldsymbol{x}) = \sum_{m=1}^M f_m \mathbb{1}_{L_m}(\boldsymbol{x})$, $Q_{S,x}$ the law of $X_{\bar{S}}|X_S = x_S$. We can compute the conditional expectation of the euclidean distance with our Plug-In estimator:*

$$\begin{aligned}\mathbb{E}_{Q_{S,x}}\left[d(f(x_S, X_{\bar{S}}), f)\right] &= \mathbb{E}_{Q_{S,x}}\Big[(\sum_m f_m \mathbb{1}_{L_m}(x) - f)^2\Big] \\ &= \sum_m f_m^2 \mathbb{P}_{Q_{S,x}}\Big[L_m(x)\Big] + f^2 - 2f \sum_m f_m \mathbb{P}_{Q_{S,x}}\Big[L_m(x)\Big]\end{aligned}$$

Hence, we can approximate $\mathbb{P}_{Q_{S,x}}\Big[L_m(x)\Big] \simeq \frac{N(L_m)}{N(L_m^S)}$ as in Equation 3.1.

Proposition 3.4. *For any Random Forest regressor $F(\boldsymbol{x}) = \sum_{b=1}^{B} \frac{f^b(\boldsymbol{x})}{B}$ with $f^b(\boldsymbol{x}) = \sum_{m=1}^{M_b} f_{m,b} \mathbb{1}_{L_{m,b}}(\boldsymbol{x})$, the conditional expectation of the Euclidean distance is equal to*

$$\mathbb{E}_{X_{\bar{S}}|X_S=x_S}\Big[(\frac{\sum_b f_b(x)}{B} - f)^2\Big] = f^2 + \sum_{b=1}^{B}\sum_{m=1}^{M_b}(\frac{1}{M^2} f_{m,b}^2 - \frac{2f}{M} f_{m,b})\mathbb{P}_{X_{\bar{S}}|X_S=x_S}(L_{m,b})$$
$$+ \frac{1}{M^2}\sum_{b,l=1 b\neq l}^{B}\sum_{i=1}^{M_b}\sum_{j=1}^{M_l} f_{m,b} f_{b,l} \mathbb{P}_{X_{\bar{S}}|X_S=x_S}(L_{i,b}(x) \cap L_{j,l}(x))$$

Remark 3.1. We have an additional term that we need to estimate and approximate $\mathbb{P}_{X_{\bar{S}}|X_S=x_S}(L_{i,b}(x) \cap L_{j,l}(x))$, but we can still use the Plug-In estimator to compute it. Indeed,

$$\begin{aligned}\mathbb{P}_{X_{\bar{S}}|X_S=x_S}(L_{i,b}(x) \cap L_{j,l}(x)) &\approx \mathbb{P}(L_{i,b}(x) \cap L_{j,l}(x) | x_S \in L_{i,b}^S \cap L_{j,l}^S) \\ &\approx \frac{N(L_{i,b}(x), L_{j,l}(x))}{N(L_{i,b}^S, L_{j,l}^S)} \end{aligned} \tag{3.3}$$

Where

- $N(L_{i,b}(x), L_{j,l}(x))$ is the number of observations in the leaf $L_{i,b}(x)$ and $L_{j,l}(x)$
- $N(L_{i,b}^S, L_{j,l}^S)$ is the number of observations satisfying the conditions $\boldsymbol{x}_S \in L_{i,b}^S \cap L_{j,l}^S$ across all the leaves of the tree.

Hence, we can approximate the SDP with the Plug-In estimator. The implementation for classifier and regressor trees are in acv_tree.py\cond_sdp_clf, acv_tree.py\cond_sdp_reg respectively.

Based on the computation of the SDP of any coalition given by the previous propositions, we can derive an algorithm that finds the Sufficient Coalitions for probability π i.e $S_\pi^\star(\boldsymbol{x})$. Unlike SV computation, we don't have to compute all the conditional expectations for all subsets in order to find the coalition $S_\pi^\star$. We use a greedy algorithm that computes the SDPs for subsets of increasing sizes (starting from 1) until we find a minimal subset satisfying the Sufficient Coalition conditions. The algorithm is described in Algorithm 1 and defines the function *returnSubsets($\boldsymbol{x}$, size)* that returns all subsets of length *size* of $\boldsymbol{x}$. We can find it implementation in acv_tree.py\compute_local_sdp_clf, acv_tree.py\compute_local_sdp_reg.

3.1 Global SV

In this section, we show that the SDP gives a natural way to go from local to global variable importance scores: we interpret the task of computing a global score as the stability across the data set of the selection process of the Sufficient Coalition. By analogy with the stability selection process introduced in [2] and advocated in [3], we introduce the following definition:

```
Algorithm 2: Find Sufficient Coalition
 Input : $\boldsymbol{x}, \pi$
 n = $length(\boldsymbol{x})$
 find = False
 bestSdp = $-1$
 for $size = 1$ to $n$ do
     $S \subset returnSubsets(\boldsymbol{x}, \text{size})$
     sdp = $SDP_S(\boldsymbol{x}, f)$
     if $sdp \geq \pi$ and $sdp \geq$ bestSdp then
         bestSdp = sdp
         $S^\star_\pi = S$
         find = True
     if find = True then
         return $S^\star_\pi$
```

Definition 3.3. ***(SDP-Global).*** *Let $X = (X_1, ..., X_p)$ be features with law P and a predictor f. The Global SDP importance of X_i is defined as:*

$$SDP_{global}(X_i) = \mathbb{E}_P\left[\mathbb{1}_{X_i \in S^\star_\pi(\boldsymbol{x})}\right]$$

It is implemented in acv_tree.py\compute_global_importance_sdp_clf, acv_tree.py\compute_global_importance_sdp_reg.

4 Active SV and Swing SV (Brute force)

Definition 4.1. ***(Active Shapley values).*** *Let f a model, $\boldsymbol{x}$ an instance, $S^\star_\pi(x)$ the Sufficient Coalition of level π of $\boldsymbol{x}$, $N_\pi(\boldsymbol{x})$ is the set of remaining variables (the Null Coalition). We define the new cooperative game with value function $v^\star$ defined for all S in $S^\star_\pi(x)$, such that*

$$v^\star(f; S) \triangleq f_{S\cup N_\pi(\boldsymbol{x})}(\boldsymbol{x}_{S\cup N_\pi(\boldsymbol{x})})$$

and $v^\star(f; \emptyset) = E\left[f(X)\right]$. For all the variables X_i in $S^\star_\pi$, we define the new Shapley value as

$$\phi^\star_i(f; \boldsymbol{x}) = \frac{1}{|S^\star_\pi|} \sum_{k=0}^{|S^\star_\pi|-1} \frac{1}{\binom{|S^\star_\pi|-1}{k}} \sum_{S \in \mathcal{S}_k(S^\star_\pi(\boldsymbol{x})))} v^\star(S \cup i) - v^\star(S)$$

Definition 4.2 (**Swing Shapley Values**). *Let f a model, $\boldsymbol{x}$ an instance, $SDP_S(f; \boldsymbol{x})$ the same decision probability of coalition S. We define the new cooperative game with value* *function:*

$$v_{SDP,\pi}(f;S) = \begin{cases} 1, & \textit{if } SDP_S(f;\boldsymbol{x}) \geq \pi \\ 0, & \textit{otherwise} \end{cases} \tag{4.1}$$

Active SV is implemented in acv_tree.py\shap_values_acv. and Swing SV in acv_tree.py\shap_values_swing_clf, acv_tree.py\shap_values_swing_reg.

References

[1] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, M. S. Lauer, *et al.*, "Random survival forests," *Annals of Applied Statistics*, vol. 2, no. 3, pp. 841–860, 2008.

[2] N. Meinshausen and P. Bühlmann, "Stability selection," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 4, pp. 417–473, 2010.

[3] B. Yu and K. Kumbier, "Veridical data science," *Proceedings of the National Academy of Sciences*, vol. 117, no. 8, pp. 3920–3929, 2020.