shapr: An R-package for explaining machine learning models with dependence-aware Shapley values

Nikolai Sellereite[1] **and Martin Jullum**[1]

1 Norwegian Computing Center

Summary

A common task within machine learning is to train a model to predict an unknown outcome (response variable) based on a set of known input variables/features. When using such models for real life applications, it is often crucial to understand why a certain set of features lead to a specific prediction. Most machine learning models are, however, complicated and hard to understand, so that they are often viewed as "black-boxes", that produce some output from some input.

Shapley values (Shapley, 1953) is a concept from cooperative game theory used to distribute fairly a joint payoff among the cooperating players. Štrumbelj & Kononenko (2010) and later Lundberg & Lee (2017) proposed to use the Shapley value framework to explain predictions by distributing the prediction value on the input features. Established methods and implementations for explaining predictions with Shapley values like Shapley Sampling Values (Štrumbelj & Kononenko, 2014), SHAP/Kernel SHAP (Lundberg & Lee, 2017), and to some extent TreeSHAP/TreeExplainer (Lundberg et al., 2020; Lundberg, Erion, & Lee, 2018), assume that the features are independent when approximating the Shapley values. The R-package shapr, however, implements the methodology proposed by Aas, Jullum, & Løland (2019), where predictions are explained while accounting for the dependence between the features, resulting in significantly more accurate approximations to the Shapley values.

Implementation

shapr implements a variant of the Kernel SHAP methodology (Lundberg & Lee, 2017) for efficiently dealing with the combinatorial problem related to the Shapley value formula. The main methodological contribution of Aas et al. (2019) is three different methods to estimate certain conditional expectation quantities, referred to as the *empirical*, *Gaussian* and *copula* approach. Additionally, the user has the option of combining the three approaches. The implementation supports explanation of models fitted with the functions natively: `stats::lm`, `stats::glm`, `ranger::ranger`, `mgcv::gam` and `xgboost::xgboost`/`xgboost::xgb.train`. Moreover, the package supports explanation of custom models by supplying two simple additional class functions.

For reference, the package also includes a benchmark implementation of the original (independence assuming) version of Kernel SHAP (Lundberg & Lee, 2017), providing identical results to the "official" Kernel SHAP `Python` package `shap`. This allows the user to easily see the effect and importance of accounting for the feature dependence.

The user interface in the package has largely been adopted from the `R`-package `lime` (Pedersen & Benesty, 2019). The user first sets up the explainability framework with the

shapr function. Then the output from shapr is provided to the explain function, along with the data to explain the prediction and the method that should be used to estimate the aforementioned conditional expectations.

The majority of the code is in plain R (R Core Team, 2019), while the most time consuming operations are coded in C++ through the Rcpp package (Eddelbuettel & François, 2011) for computational speed up. For RAM efficiency and computational speed up of typical bookeeping operations, we utilize the data.table package (Dowle & Srinivasan, 2019) which does operations "by reference", i.e. without memory copies.

For a detailed description of the underlying methodology that the package implements, we refer to the paper (Aas et al., 2019) which uses the package in examples and simulation studies. To get started with the package, we recommend going through the package vignette and introductory examples available at the package's pkgdown site.

Acknowledgement

References

Aas, K., Jullum, M., & Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *arXiv preprint arXiv:1903.10464*.

Dowle, M., & Srinivasan, A. (2019). *data.table: Extension of 'data.frame'*. Retrieved from https://CRAN.R-project.org/package=data.table

Eddelbuettel, D., & François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*(8), 1–18. doi:10.18637/jss.v040.i08

Lundberg, S. M., Erion, G. G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., et al. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*, 56–67.

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems* (pp. 4765–4774).

Pedersen, T. L., & Benesty, M. (2019). *lime: Local interpretable model-agnostic explanations*. Retrieved from https://CRAN.R-project.org/package=lime

R Core Team. (2019). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Shapley, L. S. (1953). A value for n-person games. *Contributions to the theory of games*, *2*, 307–317.

Štrumbelj, E., & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, *11*(Jan), 1–18.

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, *41*(3), 647–665. doi:10.1007/s10115-013-0679-x