SHAPR in dalex for Python

Jacek Rutkowski
Szymon Tworkowski
Jakub Walendowski
University of Warsaw, Poland

Abstract

We implement SHAPR extension to KernelSHAP method in dalex library. We evaluate the implementation on synthetic and real-world datasets, comparing to standard KernelSHAP and exploring possible optimizations to make the method more efficient in practice.

1. Introduction

The main computational cost of establishing Shapley values comes from the difficulty of computing the payoff value for coalitions. KernelSHAP (Lundberg and Lee, 2017) avoids this difficulty by assuming the independence of features. This assumption is unrealistic, as dependent features are omnipresent in real-world datasets. Therefore, reliance on it might lead to an inaccurate estimation of Shapley values. SHAPR (Aas et al., 2021) is an extension to KernelSHAP, designed to account for the dependence of features. The goal of our project was to add a SHAPR extension to the DALEX library (Baniecki et al., 2021) and benchmark the newly added method in two aspects: quality of explanations (see Sections 3.1, 3.2) and computational cost of the method, compared to the original KernelSHAP. The implementation is available under this link.

2. Methodology

2.1 SHAPR method

The Kernel SHAP method simplified the task of computing Shapley values in two ways. Firstly, it reduces the problem to a weighted linear regression problem for which Shapley values are solutions. Secondly, it avoids the computationally demanding problem of computing payoffs of coalitions by introducing the assumption that the features are independent. SHAPR (Aas et al., 2021) is a method that uses the first improvement without adopting the oversimplifying feature independence assumption.

In KernelSHAP method, the assumption about features independence was used only to simplify the integral which expresses the coalition payoff:

$$v(S) = \mathbb{E}[f(x)|x_S = x_S^*] = \int f(x_{\bar{S}}, x_S^*) p(x_{\bar{S}}|x_S = x_{\bar{S}}) dx_{\bar{S}} \tag{1}$$

The main difficulty of computing this integral lies in the conditional probabilistic measure $p(x_{\bar{S}}|x_S = x_S^*)$. In KernelSHAP, it is equated to $p(x_{\bar{S}})$ which gives unrealistic results.

SHAPR, on the other hand, determines the required conditional distributions empirically. With this approach, the integral 1 becomes a weighted sum with weights obtained by applying the Gaussian distribution kernel to the training instances. This kernel is motivated by the idea that samples $(x_{\bar{S}}, x_S)$ with x_S close to x^*_S are informative about the conditional distribution $p(x_{\bar{S}}|x^*_S)$.

2.2 Computational cost of SHAPR

In this section we compare the computational complexity of SHAPR to standard KernelSHAP. We assume the training dataset has N samples and M features. We also note that a single computation means evaluating the explained model f on a single input.

2.2.1 Sampling coalitions

Ideally, we would like to include all 2^M coalitions in our linear regression computation, to account for every coalition. This is prohibitive, as the number of features in practice is often bigger than 20. Therefore, the KernelSHAP implementation from (Lundberg and Lee, 2017) relies on sampling these subsets which we also try in our implementation.

2.2.2 Computing Shapley value for a coalition

In standard Kernel SHAP, we can randomly sample values for them with uniform distribution, as the value from each training sample is equally likely. In contrast, the goal of SHAPR is to account for the dependence of features in computation of the integral 1. To do that, we need to estimate the conditional probabilities and weigh model predictions on N samples by them (for each example in the dataset, we replace the features in the coalition and evaluate f on it). This, in practice, means evaluating the model f for each data point per each coalition, yielding the $O(N)$ factor, resulting in the total complexity $O(2^M \cdot N)$.

2.2.3 Two heuristics to speed up SHAPR

In this work, we explore two simple heuristics to improve the SHAPR computational cost and make the method practical. We name the first one the *top-k* variant, based on a heuristic where in computing the integral 1, we only include values that have k largest conditional probabilities, and ignore the rest. We only evaluate the performance of this heuristic in practice (see Sections 3.1 and 3.2). Another heuristic as we test is sampling subsets, instead of computing values on all 2^M coalitions. We find this modification to decrease performance on synthetic data, and thus we do not use it in the final implementation.

2.3 Evaluation of explanations

As one of our main goals consists in comparing KernelSHAP and SHAPR, we use several methods to check whether SHAPR indeed attains better results and whether it requires less computational cost. Among numerous methods of evaluating explanations, the most self-explanatory could rely on computing correlation with some ground truth. Although such a ground truth rarely exists and is rather clumsy to define, in our case, we regard the true Shapley values as the ground truth attribution. While Shapley values are prohibitively costly to determine for the usual setup, special synthetic benchmarks created in (Liu et al.,

2021) allow the efficient computation of conditional expected values, which appear in Shapley values definition. We check for both KernelSHAP and SHAPR if they return values close to these ground truth Shapley values.

The other metrics we use are not based on any ground truth and are established through some interactions with the model. Faithfulness is a metric that tries to answer whether features indicated by a feature attribution method are important. It is checked by obscuring or removing features in the model's input and comparing the output on changed observation (Alvarez-Melis and Jaakkola, 2018). To compute the monotonicity metric, one needs to remove features without replacement, which makes it appropriate when the main goal is to see the cumulative effect of features (Luss et al., 2021).

As it was plausibly suggested in (Alvarez-Melis and Jaakkola, 2018), the three methods mentioned above suffer from a certain inappropriateness of the removal procedure. After removing some feature (for instance, some super-pixel of the image), one has to fill the input with another feature in this place, thus changing the input distribution. It violates one of the critical machine learning assumptions that training and evaluation data come from the same distribution. Therefore, it is unclear whether the degradation in model performance comes from the distribution shift or because the removed features were indeed important. The ROAR (RemOve And Retrain) approach attempts to bypass this difficulty by retraining the model each time the features are removed. Since it requires a lot of training, faithfulness, monotonicity, and infidelity are far less computationally intensive than ROAR.

2.4 Synthetic datasets

In order to efficiently compute ground true Shapley values, in (Liu et al., 2021) were constructed special synthetic datasets which allow the computation of conditional distributions which appear in the integral 1. The generation is split into two parts: generating features and defining a function to generate labels from features. Features are based on multivariate Gaussian random vectors $X = (X_1, X_2, \dots X_n) \sim \mathcal{N}(\mu, \Sigma)$. We can partition the n-dimensional vector to a smaller number of features, say we have two features $X = (Z_1, Z_2)$, where $Z_1 = (X_1, \dots X_k)$, $Z_2 = (X_{k+1}, \dots X_n)$ for some $k < n$. We can partition μ, Σ as:

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$$

Then the conditional distribution $p(Z_1|Z_2 = z_2^*)$ is a Gaussian random variable with mean and variance:

$$\mu^* = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(z_2^* - \mu_2), \quad \Sigma^* = \Sigma_{11} + \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}.$$

Since our goal was to verify the performance of SHAPR compared to KernelSHAP for dependent features, we have chosen the same covariance matrix Σ used in (Aas et al., 2021) as a parameter to generate the dataset - we took a constant matrix with value ρ, with ones on the diagonal. As expected, SHAPR gave better results on this benchmark (see Section 3.1).

3. Experimental results

In this section we empirically evaluate variants of SHAPR, comparing it to the original KernelSHAP (Lundberg and Lee, 2017).

3.1 Results on synthetic data

We use the `gaussianPiecewiseConstant` dataset from (Liu et al., 2021) to explain a decision tree model, with 500 training and 50 validation samples. The details on the dataset are explained in Section 2.4. We find the SHAPR method to significantly outperform KernelSHAP in the groundtruth Shapley value correlation.

Method	Shapley GT corr.	Faithfulness	$\text{Faithfulness}_{\text{R}}$	$\text{Monotonicity}_{\text{R}}$
BruteKernelSHAP	0.855	0.699	0.254	0.43
KernelSHAP	0.846	0.720	**0.385**	0.475
SHAPR	0.944	0.756	0.349	**0.490**
$\text{SHAPR}_{topk=10}$	**0.954**	0.752	0.329	0.45
$\text{SHAPR}_{topk=5}$	0.928	**0.771**	0.18	0.48

Table 1: Comparison between different Shapley value estimation methods on synthetic data described in 2.4. In the first column, we measure correlation with groundtruth Shapley values. In the third and fourth columns, the "R" subscript means the ROAR version of the metric. In the two last rows we evaluate SHAPR variants with the top k heuristic described in section 2.2.3.

3.2 Results on real world data

For the real world data comparison we use metrics and data from (Agarwal et al., 2022), particularly the COMPAS dataset. We observe slightly decreased performance, compared to standard KernelSHAP.

Method	RC	FA	RA	SA
Random	0.05	0.46	0.15	0.23
KernelSHAP	**0.62**	**0.55**	**0.33**	0.3
SHAPR	0.52	0.51	0.27	0.3
$\text{SHAPR}_{topk=10}$	0.55	0.53	0.3	0.29
$\text{SHAPR}_{topk=5}$	0.53	0.52	0.25	**0.31**

Table 2: OpenXAI metrics on the COMPAS dataset (higher = better).

4. Conclusion

We aim to add the SHAPR method to the dalex library. We find the proposed implementation to yield better results on synthetic data but underperform on real-world data slightly. With the top k heuristic, we managed to optimize one runtime bottleneck of the method. We did not manage to resolve the sampling problem, which we leave to future work.

References

Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artif. Intell.*, 298:103502, 2021. doi: 10.1016/j.artint.2021.103502. URL https://doi.org/10.1016/j.artint.2021.103502.

Chirag Agarwal, Eshika Saxena, Satyapriya Krishna, Martin Pawelczyk, Nari Johnson, Isha Puri, Marinka Zitnik, and Himabindu Lakkaraju. Openxai: Towards a transparent evaluation of model explanations. *CoRR*, abs/2206.11104, 2022. doi: 10.48550/arXiv.2206.11104. URL https://doi.org/10.48550/arXiv.2206.11104.

David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7786–7795, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3e9f0fc9b2f89e043bc6233994dfcf76-Abstract.html.

Hubert Baniecki, Wojciech Kretowicz, Piotr Piatyszek, Jakub Wisniewski, and Przemyslaw Biecek. dalex: Responsible machine learning with interactive explainability and fairness in python. *J. Mach. Learn. Res.*, 22:214:1–214:7, 2021. URL http://jmlr.org/papers/v22/20-1473.html.

Yang Liu, Sujay Khandagale, Colin White, and Willie Neiswanger. Synthetic benchmarks for scientific research in explainable machine learning. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c16a5320fa475530d9583c34fd356ef5-Abstract-round2.html.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Ronny Luss, Pin-Yu Chen, Amit Dhurandhar, Prasanna Sattigeri, Yunfeng Zhang, Karthikeyan Shanmugam, and Chun-Chen Tu. Leveraging latent features for local explanations. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 1139–1149. ACM, 2021. doi: 10.1145/3447548.3467265. URL https://doi.org/10.1145/3447548.3467265.