A Projected Stochastic Gradient Algorithm for Estimating Shapley Value Applied in Attribute Importance

Grah Simon, Thouvenot Vincent

A Projected Stochastic Gradient algorithm for estimating Shapley Value applied in attribute importance*

Grah Simon [1] and Thouvenot Vincent [1,2]

[1] Thales SIX GTS, France
simon.grah@thalesgroup.com
vincent.thouvenot@thalesgroup.com
[2] SINCLAIR AI Lab, France

Abstract. Machine Learning is enjoying an increasing success in many applications: medical, marketing, defence, cyber security, transportation. It is becoming a key tool in critical systems. However, models are often very complex and highly non-linear. This is problematic, especially for critical systems, because end-users need to fully understand the decisions of an algorithm (e.g. why an alert has been triggered or why a person has a high probability of cancer recurrence). One solution is to offer an interpretation for each individual prediction based on attribute relevance. Shapley Values allow to distribute fairly contributions for each attribute in order to understand the difference between a predicted value for an observation and a base value (e.g. the average prediction of a reference population). They come from cooperative game theory. While these values have many advantages, including their theoretical guarantees, they are however really hard to calculate. Indeed, the complexity increases exponentially with the dimension (the number of variables). In this article, we propose two novel methods to approximate these Shapley Values. The first one is an optimization of an already existing Monte Carlo scheme. It reduces the number of prediction function calls. The second method is based on a projected gradient stochastic algorithm. We prove for the second approach some probability bounds and convergence rates for the approximation errors according to the learning rate type used. Finally, we carry out experiments on simulated datasets for a classification and a regression task. We empirically show that these approaches outperform the classical Monte Carlo estimator in terms of convergence rate and number of prediction function calls, which is the major bottleneck in Shapley Value estimation for our application.

Keywords: Attribute Importance · Interpretability · Shapley Value · Monte Carlo · Projected Stochastic Gradient Descent.

1 Introduction

Nowadays, Machine Learning models are used for various applications with already successful or promising results. Unfortunately, a common criticism is the lack of transparency associated with these algorithm decisions. This is mainly due to a greater interest in performance (measurable on specific tasks) at the expense of a complete understanding of the model. This results in a lack of knowledge of the internal working of the algorithm by the developer and the end user. The most obvious consequences are firstly a difficulty to correct the algorithm by an expert (different assumptions, removing outliers, adding new variables or diverse samples). Secondly, limiting its adoption by operational staff. There is even an urgent need for an explainable Artificial Intelligence (AI). Indeed, beyond these first reasons, the European Commission has imposed by legal means, with the General Data Protection Regulation, this transparency constraint on companies whose algorithms learn from personal data coming from European citizens. The challenge that companies are facing today is to bring AI into production. The transition from conclusive laboratory tests (Proof of Concept) to a production environment is not easy. To ensure that the model generalizes well on new data, a good human/machine interaction is highly appreciated.

There is no single definition of interpretability or explainability concerning model prediction (e.g. see the excellent introductory book [**?**]). Therefore, there are several ways to proceed. Assessing them objectively is

* Supported by SPARTA project

a real problem because we do not have unanimous criteria. Most studies analyze the feedback from a panel of individuals, expert or not, to demonstrate the contribution of a method in terms of understanding. Methods are rarely compared directly with each other, but rather against a lack of interpretation of algorithm decisions. However, some references try to create quantitative indicators to evaluate the complexity of a model [**?**].

We can however separate methods into two dimensions [**?**]. If the method is local or global, and if its approach is model agnostic or on the contrary inherent to it. A global method aims at explaining the general behaviour of a model, whereas a local method focuses on each decision of a model. The agnostic category (also called post-hoc explanation) considers the model as a black box. On the other hand, inherent or non-agnostic methods can modify the structure of a model or the learning process to create intrinsically transparent algorithms. Naturally, the best strategy is to find a model that is both completely transparent by design and sufficient in terms of accuracy. Unfortunately, the most effective machine learning models tend to be less transparent because their degrees of complexity are high (e.g. gradient boosting trees, deep learning, kernel methods).

In this study, the method that we improve is agnostic and local. This approach makes it possible to create a reusable and generic module for different use cases (this is the agnostic aspect). On the other hand, the complexity of such approaches is more important and therefore hinders applications where transparent decision making is done in real time. The objective of this work is precisely to speed up calculation of these explanations.

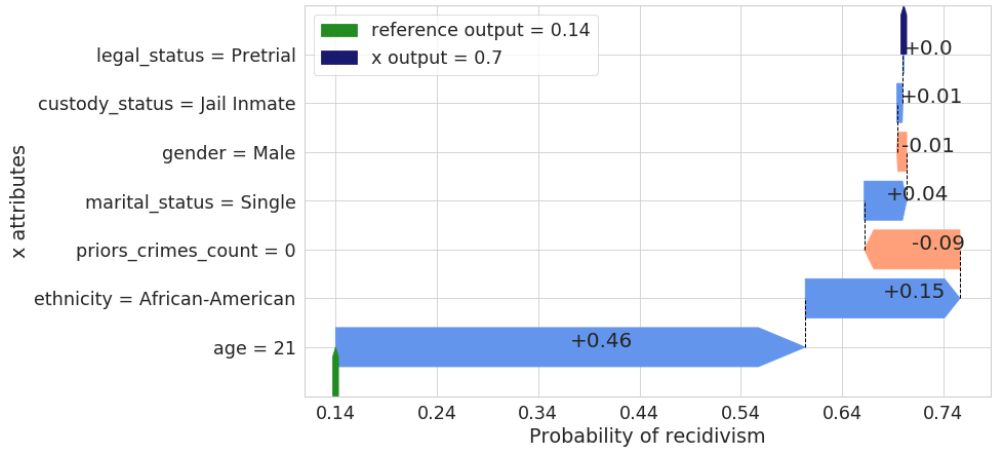


Fig. 1: Illustration of how are used Attribute Importance on a particular prediction. Here is the COMPAS dataset [**?**]. A machine learning has been trained to predict whether an individual will re-offend (giving the estimated probability associated). The importance of individual attributes give an interpretation of the difference between the probability of that individual (70% here) and the mean probability of a reference group (14%). It is relevant to choose non predicted recidivist individual references as element of comparison. We observe that the age of the individual (21 years old) has increased the probability by 46% while the absence of priors crimes has reduced it by 9%. The prediction is thus decomposed into an additive model where each attribute has a main effect.

For local explanations in Machine Learning, many methods have been proposed but the one based on Shapley Values [**?**] is becoming more and more popular. This is largely explained by the well-established theory behind these values, with the guarantee that they respect a number of interesting properties in our context, as explain in Section 2. In addition, there is no assumption made about feature independence. Shapley Values are derived from cooperative game theory and depend on the definition of a game. This is why we encounter several slightly different versions of Shapley Values in the context of attribute importance ([**?**]; [**?**]). However, Shapley Values have an unpleasant drawback: the computation cost grows exponentially

with the number of features. For instance, if the dimension is 30, that is to say 30 features, we must take into account 2^{30} coalitions and for each coalition perform 2 evaluations of the reward function. A dimension greater than 30 is common for Machine Learning problems. Several approaches have been proposed to approximate Shapley Values. Most of them are stated in articles from the Operational Research and Game Theory communities in different contexts. In this study, we will be focus only on methods related to attribute importance in Machine Learning. [**?**] worked on the estimation of Shapley Values when the data is structured (e.g. images, sentences, etc.). This relationship can look like chains or grids for example. The authors propose two approaches they call L-Shapley and C-Shapley. Because the features can be represented by a graph, we can use the notion of neighbor to limit the number of coalitions to be considered for the calculation of a Shapley Value per player. In L-Shapley, we consider all coalitions with entities whose distance is less than k from the given feature, k being a parameter chosen. With C-Shapley, coalitions are taken into account if their distance is less than k but also if they have a direct link with the player. Other authors have proposed algorithm-specific approaches. For instance, [**?**] and [**?**] present DeepSHAP, a Shapley Value approximation for Deep Learning. It is an adaptation of Deep LIFT [**?**]. In [**?**], authors propose an approximation to compute Shapley Value for tree based model (e.g. Random Forest [**?**], Gradient Tree Boosting [**?**]). In the context of Federated Learning, [**?**] give another Shapley Value approximation technique. We use Federated Learning when several entities collaborate together to train a model on distributed datasets, but do not want to share their data. In [**?**], they focus on a process where each member of the coalition train a model, and then a meta-model is learned. They are also dealing with Vertical Federated Learning, where there are many overlapped instances but few overlapped features (e.g. insurer and online retailers). Their objective is twofold: find the importance of each feature and preserve the confidential information of each partner. Their article propose an adaptation of [**?**] in this context. [**?**] formalize the use of Shapley Value for Machine Learning, in particular individual prediction explanation. All these previous approximation approaches have been used in specific contexts (Federated Learning, structured dataset) or models (Deep Learning, Tree-based model). We are interested in more generic cases. [**?**] propose an approach based on a Monte Carlo estimator which has a solid theoretical background. [**?**] and [**?**] rewrite Shapley Value computation as a weighted least square problem.

After having defined Shapley Values in Section 2, we propose two novel techniques for approximating Shapley Values, based on the works from [**?**] and [**?**]. In Section 3, we define a method based on an optimization trick of the Monte Carlo estimator of [**?**]. In Section 4, we use a projected stochastic gradient algorithm to solve the weighted least square problem of [**?**] and [**?**]. We give the theoretical properties in term of approximation errors for the second algorithm. Finally, in Section 5, we compare these two approaches with the classical Monte Carlo estimator [**?**] on simulated examples. Empirically, we observe that the time spent by calling the machine learning model prediction is often the bottleneck of estimation methods (in an agnostic approach). That is why the number of reward function evaluations, which relies on model predictions, is a good element of comparison between these methods. This point drives us for the theoretical and empirical studies. Moreover, since the true Shapley Values are intractable in many cases, theoretical guarantees such as upper bounds with high probability are essential.

2 Shapley Values in Machine Learning

2.1 Shapley Values definition

In Collaborative Game Theory, Shapley Values ([**?**]) can distribute a reward among players in a fairly way according to their contribution to the win in a cooperative game. We note $\mathcal{M}$ a set of d players. Moreover, we note $v : P(\mathcal{M}) \to R_v$ a reward function such that $v(\emptyset) = 0$. The range R_v can be $\Re$ or a subset of $\Re$. We express R_v in the context of Machine Learning in Section 2.2. $P(\mathcal{M})$ is a family of sets over $\mathcal{M}$. If $S \subset \mathcal{M}$, $v(S)$ is the amount of wealth produced by coalition S when they cooperate.
The Shapley Value of a player j is a fair share of the global wealth $v(\mathcal{M})$ produced by all players together:

$$\phi_j(\mathcal{M}, v) = \sum_{S \subset \mathcal{M} \setminus \{j\}} \frac{(d - |S| - 1)!|S|!}{d!} \left(v(S \cup \{j\}) - v(S)\right),$$

with $|S|$ = cardinal(S), i.e. the number of players in coalition S. The Shapley Values are the only values which respect the four following properties:

- Additivity: $\phi_j(\mathcal{M}, v+w) = \phi_j(\mathcal{M}, v) + \phi_j(\mathcal{M}, w)$ for all j, with $v : P(\mathcal{M}) \to R_v$ and $w : P(\mathcal{M}) \to R_v$;
- Null player: if $v(S \cup \{j\}) = v(S)$ for all $S \subset \mathcal{M} \backslash \{j\}$ then $\phi_j(\mathcal{M}, v) = 0$;
- Symmetry: $\phi_{\Pi j}(\Pi\mathcal{M}, \Pi v) = \phi_j(\mathcal{M}, v)$ for every permutation Π on $\mathcal{M}$;
- Efficiency: $\sum_{j \in \mathcal{M}} \phi_j(\mathcal{M}, v) = v(\mathcal{M})$.

When there is no risk of confusion, we will simply write ϕ_j instead of $\phi_j(\mathcal{M}, v)$ in the rest of document.

2.2 Shapley Values as contrastive local attribute importance in Machine Learning

In this study we consider the context described in [**?**] but all the estimation methods presented hereafter can be extended. Let be $X^* \subset \Re^d$ a dataset of individuals where a Machine Learning model f is trained and/or tested and d the dimension of X^*. $d > 1$ else we do not need to compute Shapley Value. We consider the attribute importance of an individual $\mathbf{x}^* = \{x_1^*, \ldots, x_d^*\} \in X^*$ according to a given reference $\mathbf{r} = \{r_1, \ldots, r_d\} \in X^*$. We're looking for $\boldsymbol{\phi} = (\phi_j)_{j \in \{1,\ldots,d\}} \in \Re^d$ such that:

$$\sum_{j=1}^{d} \phi_j = f(\mathbf{x}^*) - f(\mathbf{r}),$$

where ϕ_j is the attribute contribution of feature indexed j. We loosely identify each feature by its column number. Here the set of players $\mathcal{M} = \{1, \ldots, d\}$ is the feature set.
The local explanation is then contrastive because attribute relevance depends on a reference. For Machine Learning purposes, the range R_v of wealth v could be any real number, namely the output of a regression model, the maximum probability estimated or the individual likelihood in a classification task, anomaly score for outliers detection, and so on. It could also be a discrete number, for instance the predicted rank in a ranking problem, a one-loss function for binary or multi-class classification which means that the reward equals 1 if two instances are in the same class and 0 otherwise.

In Machine Learning, a common choice for the reward is $v(S) = \mathbb{E}[f(X)|X_S = \mathbf{x}_\mathbf{S}^*]$, where $\mathbf{x}_\mathbf{S}^* = (x_j^*)_{j \in S}$ and X_S the element of X for the coalition S.
For any $S \subset \mathcal{M}$, let's define $z(\mathbf{x}^*, \mathbf{r}, S)$ such that $z(\mathbf{x}^*, \mathbf{r}, \emptyset) = \mathbf{r}$, $z(\mathbf{x}^*, \mathbf{r}, \mathcal{M}) = \mathbf{x}^*$ and

$$z(\mathbf{x}^*, \mathbf{r}, S) = (z_1, ..., z_d) \text{ with } z_i = \begin{cases} x_i^* \text{ if } i \in S \\ r_i \text{ if } i \notin S \end{cases}.$$

As explain in [**?**], each reference $\mathbf{r}$ sets a single-game with $v(S) = f(z(\mathbf{x}^*, \mathbf{r}, S)) - f(\mathbf{r})$, $v(\emptyset) = 0$ and $v(\mathcal{M}) = f(\mathbf{x}^*) - f(\mathbf{r})$. If we sample many references from a dataset with one distribution D^{pop}, we can recover the more classical definition of attribute contributions according to a base value $\mathbb{E}_{r \sim D^{pop}}[f(r)]$ like in [**?**], [**?**] and [**?**] by averaging all the Shapley Values of each single game obtained on individuals from the reference population.

Out of direct interest, estimating $\mathbb{E}[f(X)|X_S = \mathbf{x}_\mathbf{S}^*]$ is not an easy task. In [**?**], the authors exhibit two popular ways using conditional or interventional distributions. Both techniques have their own pro and cons which lead to different definitions of what attribute relevance should be. When we use a reference $\mathbf{r}$ and create a new instance $z(\mathbf{x}^*, \mathbf{r}, S)$, we are in the second family. The main drawback is that we assume feature independence (only for the calculation of $v(S)$) which can produce out-of-distribution samples. On the other hand, the conditional approach could give non zero contribution to attributes which do not impact the reward function. It happens when an irrelevant attribute is correlated to relevant ones. Anyway, the methods that we present here are agnostic to the chosen approach, it can be conditional or interventional. Even if we use the interventional approach as an illustration, the same techniques will also work by changing the reward function or the way we estimate $v(S)$.

3 Estimation of Shapley Values by Monte Carlo

The Monte Carlo method and its variants are by far the most commonly used techniques for estimating Shapley Values. In its classical form, this approach samples random permutations between players (e.g. [**?**], [**?**], [**?**]). Our goal is to reduce the number of times the costly reward function is asked. We propose in Algorithm 1 an optimized version of the algorithm proposed by [**?**] which divides by two the use of the reward function v. To the best of our knowledge, this optimization trick has not been explicitly stated in any article dealing with Shapley Values. Moreover, this strategy can be combined with stratified sampling (e.g. [**?**], [**?**]) in order to reduce the number of iterations required. Indeed, estimated variances of each player are updated online. Let $\pi(\{1,\dots,d\})$ be the set of all ordered permutations of $\{1,\dots,d\}$ in Algorithm 1.

Data: instance $\mathbf{x}^*$ and $\mathbf{r}$, the reward function v and the number of iterations T.
Result: The Shapley Values $\widehat{\boldsymbol{\phi}} \in \Re^d$
initialization $\widehat{\boldsymbol{\phi}} = \{0,\dots,0\}$ and $\widehat{\boldsymbol{\sigma^2}} = \{0,\dots,0\}$;
for *t=1,...,T* **do**
  Choose the subset resulting of an uniform permutation $O \in \pi(\{1,\dots,d\})$ of the features values ;
  $v^{(1)} = v(\mathbf{r})$;
  $\mathbf{b} = \mathbf{r}$;
  for *j in O* **do**
    $\mathbf{b} = \begin{cases} x_i^* \text{ if } i = j \\ b_i \text{ if } i \neq j \end{cases}$, with $i \in \{1,\dots,d\}$;
    $v^{(2)} = v(\mathbf{b})$;
    $\phi_j = v^{(2)} - v^{(1)}$;
    if t > 1, $\widehat{\sigma_j^2} = \frac{t-2}{t-1}\widehat{\sigma_j^2} + (\phi_j - \widehat{\phi_j})^{2/t}$;
    $\widehat{\phi_j} = \frac{t-1}{t}\widehat{\phi_j} + \frac{1}{t}\phi_j$;
    $v^{(1)} = v^{(2)}$;
  end
end

Algorithm 1: Optimized version of Monte Carlo algorithm.

[**?**] demonstrate that the estimation error could be bounded with high probability regarding some assumptions. These results are still valid for the optimized version.

4 Estimation of Shapley Values by a Projected Stochastic Gradient algorithm

An alternative way to Monte Carlo is to use the equivalence between the initial formulation of the Shapley values and an optimization problem. Indeed, these values are the only solution of a weighted linear regression problem with an equality constraint (see [**?**], [**?**] and [**?**]). The convex optimization problem is given by Equation (1).

$$\begin{aligned} \underset{\boldsymbol{\phi}\in\Re^d}{\text{argmin}} \quad & \sum_{S\in\mathcal{M}, S\neq\{\emptyset,\mathcal{M}\}} w_S \, [v(S) - \sum_{j\in S}\phi_j]^2 \\ \text{subject to} \quad & \sum_{i=j}^{d} \phi_j = v(\mathcal{M}) \end{aligned} \tag{1}$$

where the weights $w_S = \dfrac{(d-1)}{\binom{d}{|S|}|S|(d-|S|)}$.

There are few resources that specifically address that problem. [**?**] only mention very briefly its estimation by a debiased version of Least Absolute Shrinkage and Selection Operator (LASSO, see [**?**]) without further details. [**?**] formalize more finely the resolution of this weighted linear regression. The weighted sum takes into account all the coalitions, including therefore $\mathcal{M}$ and $\emptyset$ with infinite weights $w_{\mathcal{M}} = w_\emptyset = \infty$ (in practice these weights are set by a high constant). There is no equality constraint since it is ensured by the infinite

weight $w_{\mathcal{M}}$. The function to be minimized is reformulated as a weighted least square problem, which basically has a unique solution. Because of the high dimension, exponential with the number of features, [**?**] propose to sample the coalitions used according to coalition weights to reduce the dimension. Unfortunately, we have no information on the error made by considering only a sub-sample of all coalitions. The authors recommend the largest possible value for that sub-sample size, however this results in a costly computation. The new method that we propose and detail in this paper aims to continue the work initiated by these authors on the estimation of Shapley Values by solving this weighted linear regression problem.

4.1 Solving by a Projected Stochastic Gradient algorithm

Our goal is to reduce the number of coalitions used and then the reward function evaluations while controlling the estimation error. The function we want to minimize is:

$$F(\boldsymbol{\phi}) = (X\boldsymbol{\phi} - Y)^T W (X\boldsymbol{\phi} - Y) = \frac{1}{n}\sum_{i=1}^{n} n w_i (y_i - \mathbf{x_i}^T \boldsymbol{\phi})^2 = \frac{1}{n}\sum_{i=1}^{n} g_i(\boldsymbol{\phi}),$$

where $n = 2^d - 2$ is the number of all coalitions except the full and the empty coalitions, $W = \text{diag}(w_S)$ is a diagonal matrix of size $n \times n$ whose diagonal elements are the weights w_S without $S = \emptyset$ or $\mathcal{M}$. X is a binary matrix of size $n \times d$ representing all coalitions (except null and full coalitions): each element in the column j of the row i is equal to 1 if the player j is in the coalition i, 0 otherwise. $\mathbf{x_i}$ corresponds to the i-th row of X. $\mathbf{Y} = (y_i)_{i\in\{1,\dots,n\}}$ is a vector of size n with $y_i = v(S)$ where S is the i-th coalition. We assume that for all $i \in \{1,\dots,n\}$, $|y_i| < C$, with $C > 0$ a constant. In our configuration $v(S) = v(z(\mathbf{x}^*, \mathbf{r}, S)) - v(\mathbf{r})$, each coalition S is indexed by an integer i.
We denote $K_1 = \{\boldsymbol{\phi}; \sum_{j=1}^{d} \phi_j = v(\mathcal{M})\}$ and $K_2 = \{\boldsymbol{\phi};\ \|\boldsymbol{\phi}\| \leq D\}$ two convex sets. $D > 0$ is a constant which is required to demonstrate the theoretical performance in Section 4.2, but in practice it can be large enough. This allows to avoid large gradient norm, but it can be removed by using a small learning rate. The convex set K_1 ensures that the solution respects the equality constraint $\sum_{j=1}^{d} \phi_j = v(\mathcal{M})$. F is a μ-strongly convex function defined on a convex set:

$$K = \{\boldsymbol{\phi}; \sum_{j=1}^{d} \phi_j = v(\mathcal{M})\ ;\ \|\boldsymbol{\phi}\| \leq D\} = K_1 \cap K_2.$$

This optimization problem has a unique solution $\boldsymbol{\phi}^*$ which is the Shapley Values if D is chosen such that $||\boldsymbol{\phi}^*|| \leq D$.
We denote $\boldsymbol{\phi_t} = (\phi_i^t)_{i\in\{1,\dots,d\}}$ the Shapley Values estimator at the iteration t. We also define $i_t \sim U_n(\{1,\dots,n\})$, with $U_n(\{1,\dots,n\})$ a discrete uniform distribution with support $\{1,\dots,n\}$, the coalition randomly draw for the iteration t. To find the unique minimum of F on K, the Projected Stochastic gradient algorithm at each iteration t follows the rules:

$$\begin{aligned} &i_t \sim U_n(\{1,\dots,n\}),\\ &\boldsymbol{\phi_t} = \text{Proj}_K(\boldsymbol{\phi_{t-1}} - \gamma_t \nabla g_{i_t}), \end{aligned}$$

where

- γ_t is a constant or decreasing step-size (also called the learning rate). $\forall t, \gamma_t > 0$;
- Proj_K is the orthogonal projection on K;
- $\mathbb{E}[\nabla g_{i_t} | \mathcal{F}_{t-1}]$ is the gradient of F at $\boldsymbol{\phi_{t-1}}$. $\mathcal{F}_{t-1}$ is the σ-field generated by $x_1, y_1, ..., x_{t-1}, y_{t-1}$;
- $\mathbb{E}[\|\nabla g_{i_t}\|^2] \leq B^2$ (finite variance condition). $B > 0$.

For each iteration t, we need to find the orthogonal projection of $\boldsymbol{\phi_t} \in \Re^d$ onto the convex set $K_1 \cap K_2 = K$ where K_1, K_2 are convex sets. Dykstra's algorithm ([**?**], Algorithm 2) can be used because we know how to project onto the sets K_1 and K_2 separately. For $\boldsymbol{\phi_t} \in \Re^d$, if $\boldsymbol{\phi_t}$ does not belong to either K_1 or K_2:

$$\text{Proj}_{K_1}(\boldsymbol{\phi_t}) = \boldsymbol{\phi_t} - \frac{\sum_j \phi_j^t - v(\mathcal{M})}{d},$$

$$\text{Proj}_{K_2}(\boldsymbol{\phi_t}) = \boldsymbol{\phi_t} \times \frac{D}{\|\boldsymbol{\phi_t}\|}.$$

Data: instance $\boldsymbol{\phi_t} \in \Re^d$, Proj_{K_1} and Proj_{K_2} and the maximum number of iterations L.
Result: the orthogonal projection of $\boldsymbol{\phi_t}$ onto K
initialization: $\alpha_1 = \phi$ and $p_1 = q_1 = 0$;
for *l=1,...,L* **do**
  $\beta_l = \text{Proj}_{K_2}(\alpha_l + p_l)$;
  $p_{l+1} = \alpha_l + p_l - \beta_l$;
  if $Proj_{K_1}(\beta_l + q_l) = \alpha_l$ **then**
    break;
  else
    $\alpha_l = \text{Proj}_{K_1}(\beta_l + q_l)$;
    $q_{l+1} = \beta_l + q_l - \alpha_l$;
  end
end
return α_l

Algorithm 2: Dykstra's algorithm 2.

4.2 Convergence rate and high-probability bounds

Upper bound B of stochastic gradient norm Considering $\nabla g_{i_t} = 2n(w_{i_t} < \mathbf{x_{i_t}}, \boldsymbol{\phi_{t-1}} > \mathbf{x_{i_t}} - w_{i_t} y_{i_t} \mathbf{x_{i_t}})$ and applying triangle inequality leads to:

$$\begin{aligned} \|\nabla g_{i_t}\| &\leq 2n \ (\|w_{i_t} < \mathbf{x_{i_t}}, \boldsymbol{\phi_{t-1}} > \mathbf{x_{i_t}}\| + \|w_{i_t} y_{i_t} \mathbf{x_{i_t}}\|) \\ &\leq 2n \ (|w_{i_t}| \ \|\boldsymbol{\phi_{t-1}}\| \ \|\mathbf{x_{i_t}}\|^2 + |w_{i_t}||y_{i_t}| \ \|\mathbf{x_{i_t}}\|) \\ &\leq 2n w_{i_t} \ \|\mathbf{x_{i_t}}\| \ (D \ \|\mathbf{x_{i_t}}\| + C). \end{aligned} \tag{2}$$

We introduce L_{i_t} such that $L_{i_t} = 2n w_{i_t} \ \|\mathbf{x_{i_t}}\| \ (D \ \|\mathbf{x_{i_t}}\| + C)$. L_{i_t} only depends on the chosen coalition i_t and $\|x_{i_t}\|^2$ equals the number of players for that coalition. Hence L_{i_t} is maximum when the number of players is $d-1$ with $w_{i_t} = \dfrac{1}{d}$. That is why:

$$\mathbb{E}\big[\ \|\nabla g_{i_t}\|^2\ \big] \leq \mathbb{E}[L_{i_t}^2] \leq \Big[2n \ \frac{\sqrt{d-1}}{d}(\sqrt{d-1}D + C)\Big]^2.$$

This upper bound can be improved by using Importance Sampling. If we consider a discrete distribution p on $[0,1]^n$:

$$F(\boldsymbol{\phi}) = (X\boldsymbol{\phi} - Y)^T W (X\boldsymbol{\phi} - Y) = \frac{1}{n}\sum_{i=1}^{n} g_i(\boldsymbol{\phi}) = \sum_{i=1}^{n} p_i \ (np_i)^{-1} g_i(\boldsymbol{\phi}),$$

Then, $\forall t > 0$:

$$\nabla F(\boldsymbol{\phi_{t-1}}) = \sum_{i=0}^{n} p_i \ (np_i)^{-1} \nabla g_i(\boldsymbol{\phi_{t-1}}) = \mathbb{E}_{i_t \sim p}\big[(np_{i_t})^{-1} \nabla g_{i_t}\big].$$

We want to find a distribution p such that $\mathbb{E}_{i_t \sim p}\big[\ \big\|(np_{i_t})^{-1}\nabla g_{i_t}\big\|^2\ \big]$ is better bounded (e.g.[**?**]). As we already showed in Equation (2), $\|\nabla g_{i_t}\| \leq L_{i_t}$ and a suggested distribution is:

$$p_i = \frac{L_i}{\sum_{j=1}^{n} L_j}, \forall i = 1, \ldots, n.$$

For each coalition i, $L_i = 2n w_i \sqrt{l_i}(D\sqrt{l_i} + C)$ where l_i is the number of attributes in that coalition. Then we get:

$$\begin{aligned} \mathbb{E}_{i_t \sim p}\big[\ \big\|(np_{i_t})^{-1}\nabla g_{i_t}\big\|^2\ \big] &= \frac{1}{n^2}\sum_{i=1}^{n} (p_i)^{-1} \ \|\nabla g_i\|^2 \\ &\leq \frac{1}{n^2}\sum_{i=1}^{n} \frac{\sum_j L_j}{L_i} L_i^2 \\ &\leq \frac{\big(\sum_j L_j\big)^2}{n^2}. \end{aligned}$$

Actually, we do not have to compute L_i for all coalitions because L_i only depends on the size of the i^{th} coalition . For all i^{th} coalitions whose size is $l \in \{1, ..., d-1\}$, $L_i = 2n\frac{(d-1)}{\binom{d}{l} l(d-l)}\sqrt{l}(D\sqrt{l}+C)$. By denoting $L^l = 2n\frac{(d-1)}{\binom{d}{l} l(d-l)}\sqrt{l}(D\sqrt{l}+C)$ for every size l, we get $\sum_j L_j = \sum_{l=1}^{d-1}\binom{d}{l}L^l$.

$$\sum_j L_j = 2n\sum_{l=1}^{d-1}\binom{d}{l}\frac{(d-1)}{\binom{d}{l} l(d-l)}\sqrt{l}(\sqrt{l}D+C) = 2n\sum_{l=1}^{d-1}\frac{d-1}{\sqrt{l}(d-l)}(\sqrt{l}D+C).$$

Finally, let's define B such that $B^2 = \frac{\left(\sum_j L_j\right)^2}{n^2} = 4\left[\sum_{l=1}^{d-1}\frac{d-1}{\sqrt{l}(d-l)}(\sqrt{l}D+C)\right]^2$. We have found an upper bound for the stochastic gradient norm:

$$\mathbb{E}_{i_t\sim p}\left[\left\|(np_{i_t})^{-1}\nabla g_{i_t}\right\|^2\right] \leq B^2$$

Note that we have removed n from the previous upper bound of gradient norm. We will use this new estimator of gradient F and then the Projected Stochastic gradient algorithm becomes:

$$\begin{aligned} &i_t \sim p, \\ &\boldsymbol{\phi_t} = \text{Proj}_K(\boldsymbol{\phi_{t-1}} - \gamma_t\ (np_{i_t})^{-1}\nabla g_{i_t}), \end{aligned}$$

at each iteration t with p the chosen distribution above.

Strongly convex constant $\boldsymbol{\mu}$ F is a μ-strongly convex function. We know that μ is the smallest eigenvalues of the Hessian of F: $\nabla^2 F = X^T W X$ which does not rely on $\boldsymbol{\phi}$. We can demonstrate that $\mu = 1 - 1/d$.

$$X^T W X_{(d,d)} = \begin{pmatrix} a & c & \cdots & c \\ c & a & \cdots & c \\ \vdots & \vdots & \ddots & \vdots \\ c & c & \cdots & a \end{pmatrix} = (a-c)\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} + c\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} = (a-c)I_d + cJ_d,$$

where $a = \sum_{k=1}^{d-1}\binom{d-1}{k-1}w_k$ and $c = \sum_{k=2}^{d-1}\binom{d-2}{k-2}w_k$.

The eigenvalues of J_d are 0 (order $d-1$) and d (order 1). If z is an eigenvector of J_d associated to the eigenvalue λ, then $X^T W X z = ((a-c)I_d + cJ_d)z = ((a-c) + c\lambda)z$. That is why eigenvalues of $X^T W X$ are $(a-c)$ and $(a-c)+cd$. The smallest one is $a-c$ and thus by definition $\mu = a-c$. We can then prove that $a - c = 1 - 1/d$. Full details of that demonstration are given in Appendix **??**.

Convergence rate For the sake of clarity, we will denote $\kappa = \frac{B}{\mu} = 4\ \sum_{l=1}^{d-1}\frac{d}{\sqrt{l}(d-l)}(\sqrt{l}D+C)$.

It has been proved in [**?**] that if we choose an inverse decreasing step-size $\gamma_t = \frac{2}{\mu(t+1)}$, with $t \in \{1, \ldots, T\}$, T being the number of iterations, we get the following convergence rate:

$$F(\bar{\boldsymbol{\phi}}_{\boldsymbol{T}}) - F(\boldsymbol{\phi}^*) \leq \frac{2B^2}{\mu T},$$

where $\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} = \frac{2}{(T+1)(T+2)}\sum_{t=0}^{T}(t+1)\boldsymbol{\phi_t}$. In practice, these averaging is updated online with:

$$\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} = (1-\rho_t)\bar{\boldsymbol{\phi}}_{\boldsymbol{T-1}} + \rho_t\boldsymbol{\phi_t}, \text{ where } \rho_t = \frac{2}{(t+2)}$$

As F is μ-strongly convex, we have the following inequality for all $\mathbf{x}, \mathbf{y} \in \Re^d$:

$$F(\mathbf{y}) \geq F(\mathbf{x}) + \nabla F(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \frac{\mu}{2}\left\|\mathbf{y}-\mathbf{x}\right\|^2.$$

In particular if $\mathbf{x} = \boldsymbol{\phi}^*$, the unique solution, $\nabla F(\boldsymbol{\phi}^*) = 0$ and $\|\mathbf{y} - \boldsymbol{\phi}^*\|^2 \leq \frac{2}{\mu}(F(\mathbf{y}) - F(\boldsymbol{\phi}^*))$ for any $\mathbf{y} \in \mathbf{R}^d$. That is why:

$$\mathbb{E}\Big[\, \big\|\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} - \boldsymbol{\phi}^*\big\|^2 \,\Big] \leq \frac{4B^2}{\mu^2 T} = \frac{4\kappa^2}{T} = \mathcal{O}(\frac{1}{T}).$$

With a square root decreasing step-size $\gamma_t = \dfrac{2D}{B\sqrt{t}}$ and $\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} = \dfrac{1}{T}\sum_{t=0}^{T} \boldsymbol{\phi}_{\boldsymbol{t}}$ (e.g. [**?**]): $F(\bar{\boldsymbol{\phi}}_{\boldsymbol{T}}) - F(\boldsymbol{\phi}^*) \leq \dfrac{2DB}{\sqrt{T}}$. Then

$$\mathbb{E}\Big[\, \big\|\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} - \boldsymbol{\phi}^*\big\|^2 \,\Big] \leq \frac{4DB}{\mu\sqrt{T}} = \frac{4D\kappa}{\sqrt{T}} = \mathcal{O}(\frac{1}{\sqrt{T}}).$$

For a constant step size $\gamma < \dfrac{1}{\mu} = \dfrac{d}{d-1}$ (see [**?**]), we get:

$$\mathbb{E}\Big[\, \|\boldsymbol{\phi}_{\boldsymbol{T}} - \boldsymbol{\phi}^*\|^2 \,\Big] \leq (1-\gamma\mu)^T \|\boldsymbol{\phi}_{\mathbf{0}} - \boldsymbol{\phi}^*\|^2 + \frac{\gamma B^2}{\mu} = \mathcal{O}(\rho^T) + \mathcal{O}(\gamma),$$

where $\rho = 1 - \gamma\mu$ and $\boldsymbol{\phi}_{\mathbf{0}}$ the initial value used for the stochastic gradient algorithm. It is a fast convergence towards an imprecise solution that we can control.

High-Probability bounds In order to obtain high-probability bounds for projected stochastic gradient descent algorithm (SGD), we can use either the Markov inequality: $\forall a \geq 0,\ \mathbb{P}(Z \geq a) \leq \dfrac{\mathbb{E}[Z]}{a}$, with Z being a random variable, or the deviation inequality: $\forall a, A \geq 0,\ \mathbb{E}[Z] \leq A \Rightarrow \mathbb{P}(Z \geq A(2+4a)) \leq 2e^{-a^2}$ (see [**?**],[**?**],[**?**]). In the following we use Markov inequality and let the results obtained by the deviation inequality in Appendix **??**.
For the inverse decreasing step-size, we obtain (with the appropriate $\bar{\phi}_{\mathbf{T}}$ defined before):

$$\mathbb{P}\Big(\, \big\|\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} - \boldsymbol{\phi}^*\big\|^2 \geq \epsilon \Big) \leq \frac{4\kappa^2}{\epsilon T}, \quad \text{for all } \epsilon > 0.$$

In a same manner, with the square root decreasing step-size we get:

$$\mathbb{P}\Big(\, \big\|\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} - \boldsymbol{\phi}^*\big\|^2 \geq \epsilon \Big) \leq \frac{4D\kappa}{\epsilon\sqrt{T}}, \quad \text{for all } \epsilon > 0.$$

Considering a constant step-size leads to:

$$\mathbb{P}\Big(\, \|\boldsymbol{\phi}_T - \boldsymbol{\phi}^*\|^2 \geq \epsilon \Big) \leq \frac{(1-\gamma\mu)^T}{\epsilon} \|\boldsymbol{\phi}_{\mathbf{0}} - \boldsymbol{\phi}^*\|^2 + \frac{\gamma B^2}{\epsilon\mu}, \quad \text{for all } \epsilon > 0.$$

A well suited element of comparison between this method and the Monte Carlo presented in Section 3, is to consider the number of reward function evaluations. According to this criteria, performing T iterations of Monte Carlo is equivalent to $T \times d$ iterations of stochastic gradient.

5 Experimental evaluation

5.1 Simulated dataset

Classification We will use the simulated dataset introduced [**?**], page 339. The features $X_1, \ldots, X_d$ are standard independent Gaussian, and the deterministic target Y is defined by:

$$Y = \begin{cases} 1 \text{ if } \sum_{j=1}^{d} X_j^2 > \chi_d^2(0.5) \\ 0 \text{ otherwise} \end{cases},$$

where $\chi_d^2(0.5)$ is the median of a chi-squared random variable with d degrees of freedom (the sum of d standard Gaussian squared follows a χ^2 probability law). We denote by f_{class} a classification function that returns 1 if the predicted score is greater than 0.5 and 0 otherwise. For a reference $\mathbf{r}^*$ and a target $\mathbf{x}^*$, we define the reward function $v_c^{\mathbf{r}^*,\mathbf{x}^*}$ such that for each coalition S, $v_c^{\mathbf{r}^*,\mathbf{x}^*}(S) = \mathbb{1}_{f_{class}(\mathbf{z}(\mathbf{x}^*,\mathbf{r}^*,S))=f_{class}(\mathbf{x}^*)}$. This reward function is one choice among others. Here, all the contributions sum to 1. Moreover, using an uncommon reward function in Machine Learning illustrates that the techniques used are agnostic of that reward function.

Regression We will also use a simulated dataset from the same book ([**?**], page 401, the Radial example). $X_1, \ldots, X_d$ are standard independent Gaussian. The model is determined by:

$$Y = \prod_{j=1}^{d} \rho(X_j),$$

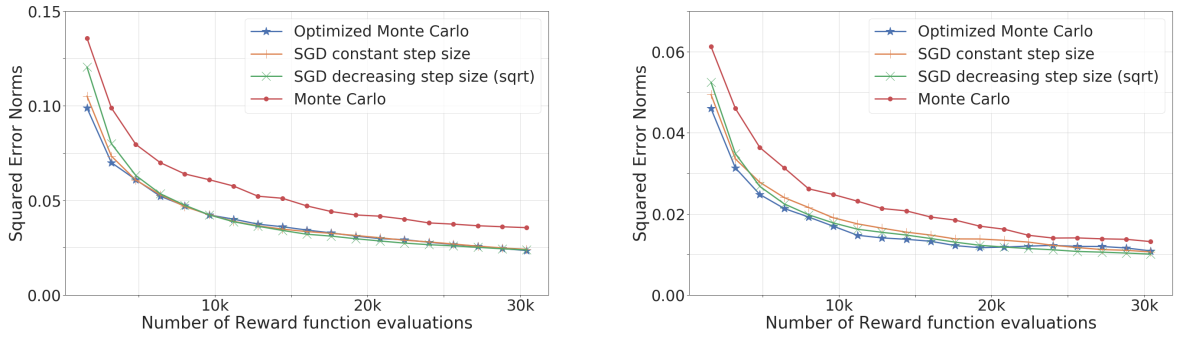
where $\rho\colon t \to \sqrt{(0.5\pi)}\exp(-t^2/2)$. The regression function f_{regr} is deterministic and simply defined by $f_r\colon \mathbf{x} \to \prod_{j=1}^{d} \rho(x_j)$. For a reference $\mathbf{r}^*$ and a target $\mathbf{x}^*$, we define the reward function $v_r^{\mathbf{r}^*,\mathbf{x}^*}$ such as for each coalition S, $v_r^{\mathbf{r}^*,\mathbf{x}^*}(S) = f_{regr}(\mathbf{z}(\mathbf{x}^*, \mathbf{r}^*, S)) - f_{regr}(\mathbf{r}^*)$.

Procedure We select at random 50 couple of instances $\mathbf{x}^*$ and $\mathbf{r}^*$. For each one, the true Shapley Values are calculated and estimation errors of the methods under study are stored for several iterations. Some graphs will sum up the experiments by displaying the mean estimation errors per iteration. For the classification problem, the instances are sampled from separate classes. Finally, let's remember that our goal is to use the fewest number of reward function evaluations as possible while having a good estimation error. The code is available here: https://github.com/ThalesGroup/shapkit.

5.2 Comparison between all methods

For the gradient based methods, the initial value $\boldsymbol{\phi_0}$ is $\mathrm{Proj}_{K_1}(\mathbf{0}) = \left\{ \frac{f_p(\mathbf{x}^*) - f_p(\mathbf{r}^*)}{d}, \ldots, \frac{f_p(\mathbf{x}^*) - f_p(\mathbf{r}^*)}{d} \right\}$, with $p = class$ or $regr$ according to the fact we work on the classification or regression task. Furthermore, we select interesting decreasing step size strategies on out-of-experiment samples $(\mathbf{x}^*, \mathbf{r}^*)$ and only display the best ones found.

Dimension 16 We choose at first a dimension d of 16 features which allows us to compute the true Shapley Values in a reasonable time. The total number of coalitions is 65 534. Figure **??** shows the results obtained for that dimension. All the proposed methods outperform the classical Monte Carlo algorithm both in classification and regression. The stochastic gradient methods displayed use a constant step size of 0.01 and a decreasing step size following $\gamma_t = 0.1/\sqrt{t}$ at each iteration t.



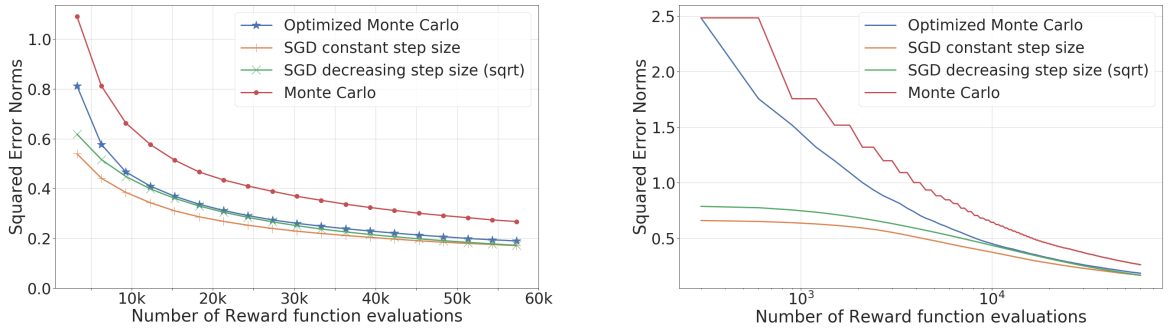
(a) Classification - dimension 16

(b) Regression - dimension 16

Fig. 2: Evolution of the mean squared error norms with the number of reward function evaluations.

Dimension 300 We would like to test the behavior of these approaches when the dimension is quite high. That is why we generate 300 features. A dimension higher than this is not so frequent for tabular

datasets. Due to computational constraints we will only study the classification problem. Indeed Shapley Values collapse towards zero in the regression settings. Because the true Shapley Values are intractable, we estimate it with the optimized Monte Carlo technique and a large number of iterations. We performed 5000 iterations which represent $5000 \times d$ reward evaluations.
The mean squared error norms can be observed on Figure **??**. The constant step size is 0.001 and the decreasing step size follows $\gamma_t = \frac{0.01}{\sqrt{t}}$. By its design, Monte Carlo techniques update each Shapley Value component sequentially during one iteration of d reward evaluations. In the meantime gradient based methods could update several Shapley Values components at each iteration. Empirically, we find that when there are strong interactions between features and when the link is strongly non-linear between the features and the target, the method of Monte Carlo approaches need more time to decrease their estimation error compared to the others.



(a) Classification - dimension 300

(b) Classification - dimension 300 - x axis log scaled

Fig. 3: Evolution of the mean squared error norms with the number of function evaluations. The right plot log scales the x axis.

6 Conclusion

In this article, we propose two novel methods for approximating Shapley Values when we want to interpret individual predictions of a Machine Learning model. The first one is based on an optimization trick applied to the classical Monte Carlo estimator of Shapley Values. The second one uses a rewrite in the form of a weighted optimization problem of the Shapley's value approximation problem, which is solved by projected stochastic gradient descent algorithm. These estimates offer theoretical guarantees on the error made. Empirically, we show that these approaches outperform the classical Monte Carlo estimator. We have observed during experiments that when features interact less with each other according to the model (that is to say the estimated model tends to an additive model), then the Monte Carlo methods are better suited while having less hyper-parameters. But when the model output depends more strongly on attribute associations (like in the previous simulated classification task), gradient based alternatives offer interesting results. It might be useful in the future to study the contribution of mini-batch approaches for the stochastic gradient. We could later study the use of Shapley Values for global explanation. Indeed, Shapley values can be estimated for a significant sub-sample of individuals and then draw several graphs: all Shapley Values as a function of one feature, mean absolute value of the shapley values for each feature, and so much more.

7 Acknowledgement

This work is supported by the SPARTA project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892.

References

1. Kjersti Aas and Martin Jullum and Anders Lland. *Explaining individual predictions when features are dependent: More accurate approximations to Shapley values*. 2019.
2. F. Bach *Adaptivity of Averaged Stochastic Gradient Descentto Local Strong Convexity for Logistic Regression*, Journal of Machine Learning Research 15 (2014) 595-627
3. F. Bach, https://www.di.ens.fr/ fbach/orsay2020.html *https://www.di.ens.fr/~fbach/fbach_orsay_2020.pdf*,
4. Barredo Arrieta, Alejandro and Diaz Rodriguez, Natalia and Del Ser, Javier and Bennetot, Adrien and Tabik, Siham and Barbado Gonzlez, Alberto and Garcia, Salvador and Gil-Lopez, Sergio and Molina, Daniel and Benjamins, V. Richard and Chatila, Raja and Herrera, Francisco. *Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI*, Journal: Information Fusion, December 2019.
5. Breiman, Leo *Random Forest*, Machine Learning 45, 532, 2001.
6. https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis
7. Boyle, J. P.; Dykstra, R. L. *A method for finding projections onto the intersection of convex sets in Hilbert spaces*, 1986, Lecture Notes in Statistics. 37. pp. 2847.
8. J. Castro, D. Gmez, E. Molina, J. Tejada *Improving polynomial estimation of the Shapley value by stratified random sampling with optimum allocation*, Computers and Operations Research Volume 82, June 2017, Pages 180-188
9. Jianbo Chen and Le Song and Martin J. Wainwright and Michael I. Jordan. *L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data*. 2018.
10. Chen, Lundberg and Lee *Explaining Models by Propagating Shapley Values*, https://arxiv.org/abs/1911.11888, 2019
11. S. Shaheen Fatima and Michael Wooldridge and Nicholas R. Jennings. *A linear approximation method for the Shapley value*. journal: Artif. Intell, volume 172, pages 1673-1699, 2008.
12. Amirata Ghorbani and James Zou. *Data Shapley: Equitable Valuation of Data for Machine Learning*. 2019.
13. Robert M. Gower *https://perso.telecom-paristech.fr/rgower/pdf/M2_statistique_optimisation/grad_conv.pdf*
14. Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, Springer Series in Statistics, 2009.
15. I. Elizabeth Kumar and Suresh Venkatasubramanian and Carlos Scheidegger and Sorelle Friedler. *Problems with Shapley-value-based explanations as feature importance measures*. 2020.
16. Simon Lacoste-Julien and Mark Schmidt and Francis Bach. *A simpler approach to obtaining an O(1/t) convergence rate for the projected stochastic subgradient method*. 2012.
17. Lundberg, Scott M., and Su-In Lee. *A unified approach to interpreting model predictions*. Advances in Neural Information Processing Systems. 2017.
18. Scott M. Lundberg and Su-In Lee *Consistent feature attribution for tree ensembles*, https://arxiv.org/abs/1706.06060, 2017
19. Scott M. Lundberg and Su-In Lee https://shap.readthedocs.io/en/latest/#, 2017
20. Sasan Maleki and Long Tran-Thanh and Greg Hines and Talal Rahwan and Alex Rogers. *Bounding the Estimation Error of Sampling-based Shapley Value Approximation With/Without Stratifying*. 2013.
21. Luke Merrick and Ankur Taly. *The Explanation Game: Explaining Machine Learning Models with Cooperative Game Theory*. 2019.
22. Christoph Molnar and Giuseppe Casalicchio and Bernd Bischl *Quantifying Model Complexity via Functional Decomposition for Better Post-Hoc Interpretability*, 2019
23. Christoph Molnar *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*, https://christophm.github.io/interpretable-ml-book/, 2019
24. A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro *Robust stochastic approximation approach to stochastic programming*, SIAM Journal on Optimization, 19(4):15741609, 2009.
25. Y. Nesterov and J. P. Vial *Confidence level solutions for stochastic programming*, Automatica, 44(6):15591568, 2008.
26. Owen, A. *Sobol indices and shapley value.*, SIAM/ASA Journal on Uncertainty Quantification 2, 1 (2014), 245251.
27. Owen, A. and Prieur, C. . *On shapley value for measuring importance of dependent inputs*, SIAM/ASA Journal on Uncertainty Quantification 5, 1 (2017), 9861002.
28. Nickalus Redell. *Shapley Decomposition of R-Squared in Machine Learning Models*. 2019.
29. Luis M. Ruiz and Federico Valenciano and Jose M. Zarzuelo *The Family of Least Square Values for Transferable Utility Games*, Games and Economic Behavior, 1998
30. Lloyd S Shapley. *"A value for n-person games". In Contributions to the Theory of Games*. 2.28 (1953), pp. 307 - 317.
31. Avanti Shrikumar and Peyton Greenside and Anna Shcherbina and Anshul Kundaje *Not Just a Black Box: Learning Important Features Through Propagating Activation Differences*, https://arxiv.org/abs/1605.01713, 2016

32. Sobol. *Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates*. Mathematics and Computers in Simulation 55, 271280, 2001.
33. Strumbelj, Erik, and Igor Kononenko. *Explaining prediction models and individual predictions with feature contributions*. Knowledge and information systems 41.3, 647-665, 2014.
34. Erik Strumbelj and Igor Kononenko. *An Efficient Explanation of Individual Classifications using Game Theory*. journal: J. Mach. Learn. Res., volume 11, pages 1-18, 2010.
35. Mukund Sundararajan and Amir Najmi. *The many Shapley values for model explanation*. 2019.
36. Tibshirani, Robert. *Regression shrinkage and selection via the lasso* . Journal of the Royal Statistical Society. Series B, 58 (1), 1996.
37. Guan Wang. *Interpret Federated Learning with Shapley Values*. 2019.
38. Peilin Zhao and Tong Zhang. *Stochastic Optimization with Importance Sampling for Regularized Loss Minimization*. Proceedings of the 32nd International Conference on Machine Learning.p 1-9, volume 37, 2015

8 Appendix

8.1 Strongly convex constant μ

F is a μ-strongy convex function. We know that μ is the smallest eigenvalues of the Hessian of F: $\nabla^2 F(\boldsymbol{\phi}) = X^T W X$ which does not rely on $\boldsymbol{\phi}$. We can prove that $\mu = 1 - 1/d$. At first, one can observe that:

$$X^T W X_{(d,d)} = \begin{pmatrix} a & c & \cdots & c \\ c & a & \cdots & c \\ \vdots & \vdots & \ddots & \vdots \\ c & c & \cdots & a \end{pmatrix} = (a-c) \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} + c \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} = (a-c) I_d + c J_d,$$

where $a = \sum_{k=1}^{d-1} \binom{d-1}{k-1} w_k$ and $c = \sum_{k=2}^{d-1} \binom{d-2}{k-2} w_k$.
Indeed, $X^T W X$ is a $d \times d$ matrix whose (i,j) term is $\sum_{l=1}^{n} \sum_{k=1}^{n} (\mathbf{x}^T)_{ik} w_{kl} x_{lj} = \sum_{l=1}^{n} x_{li} w_{ll} x_{lj}$ because $w_{kl} = 0$ if $k \neq l$. For each coalition l, value of w_{ll} only depends of the size of that coalition and the product $x_{li} \times x_{lj}$ equals 1 if features i and j are present in coalition l. Therefore if we denote k the size of one coalition l, it exists two cases: when $i = j$ we have $\binom{d-1}{k-1}$ possible coalitions of size k which contain feature i $(= j)$. And if $i \neq j$ there are $\binom{d-2}{k-2}$ coalitions. That is why $a = \sum_{k=1}^{d-1} \binom{d-1}{k-1} w_k$ and $c = \sum_{k=2}^{d-1} \binom{d-2}{k-2} w_k$.
The eigenvalues of J_d are 0 (order $d-1$) and d (order 1). If $\mathbf{z_e}$ is an eigenvector of J_d associated to the eigenvalue λ, then $X^T W X \mathbf{z_e} = ((a-c) I_d + c J_d) \mathbf{z_e} = ((a-c) + c\lambda) \mathbf{z_e}$. That is why eigenvalues of $X^T W X$ are $(a-c)$ and $(a-c) + cd$. The smallest one is $a - c$ and thus by definition $\mu = a - c$. We can then prove that $a - c = 1 - 1/d$.

$$\begin{aligned}
a - c &= \sum_{k=1}^{d-1} \binom{d-1}{k-1} w_k - \sum_{k=2}^{d-1} \binom{d-2}{k-2} w_k \\
&= w_1 + \sum_{k=2}^{d-1} [\binom{d-1}{k-1} - \binom{d-2}{k-2}] w_k \\
&= w_1 + \sum_{k=2}^{d-1} \binom{d-2}{k-1} w_k \\
&= \sum_{k=1}^{d-1} \binom{d-2}{k-1} w_k \\
&= \sum_{k=1}^{d-1} \binom{d-2}{k-1} \frac{(d-1)}{\binom{d}{k} k(d-k)} \\
&= \sum_{k=1}^{d-1} \frac{(d-2)!(d-1)}{(k-1)!(d-k-1)!\binom{d}{k} k(d-k)} \\
&= \sum_{k=1}^{d-1} \frac{(d-1)!}{(k)!(d-k)!\binom{d}{k}} \\
&= \sum_{k=1}^{d-1} \frac{(d-1)!}{d!} \\
&= \sum_{k=1}^{d-1} \frac{1}{d} \\
&= \frac{d-1}{d} \\
&= 1 - 1/d.
\end{aligned}$$

8.2 High-Probability bounds with a deviation inequality

In order to obtain high-probability bounds for projected stochastic gradient descent algorithm (SGD), we can use the deviation inequality: $\forall a, A \geq 0,\ \mathbb{E}[Z] \leq A \Rightarrow \mathbb{P}(Z \geq A(2+4a)) \leq 2e^{-a^2}$ (see [**?**],[**?**],[**?**]).
For the inverse decreasing step-size, we obtain (with the appropriate $\bar{\phi}_{\mathbf{T}}$ defined before):

$$\mathbb{P}\Big(\big\|\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} - \boldsymbol{\phi}^{*}\big\|^2 \geq \epsilon \Big) \leq 2\exp\Big(-\frac{1}{16}\big[\frac{\epsilon T}{4\kappa^2} - 2\big]^2 \Big) \text{ for all } \epsilon > 0.$$

In a same manner, with the square root decreasing step-size we get:

$$\mathbb{P}\Big(\big\|\bar{\boldsymbol{\phi}}_{\boldsymbol{T}} - \boldsymbol{\phi}^{*}\big\|^2 \geq \epsilon \Big) \leq 2\exp\Big(-\frac{1}{16}\big[\frac{\epsilon\sqrt{T}}{4D\kappa} - 2\big]^2 \Big) \text{ for all } \epsilon > 0.$$

Considering a constant step-size leads to:

$$\mathbb{P}\Big(\big\|\boldsymbol{\phi}_T - \boldsymbol{\phi}^{*}\big\|^2 \geq \epsilon \Big) \leq 2\exp\Big(-\frac{1}{16}\big[\epsilon/\big(\rho^T \big\|\boldsymbol{\phi}_{\mathbf{0}} - \boldsymbol{\phi}^{*}\big\|^2 + \frac{\alpha B^2}{\mu}\big) - 2\big]^2 \Big) \text{ for all } \epsilon > 0.$$