Shapley Performance Attribution for Least Squares

Logan Bell Nikhil Devanathan Stephen Boyd

Stanford University

October 21, 2023

Disclaimer

Shapley performance attribution (SPA) is very different from Shapley additive explanations (SHAP).

- SPA attributes the performance of a model across its features.
- SHAP attributes a specific output of a model across its features for model explainability.
- Both methods share several keywords, but they are only superficially related.
- We will only discuss SPA, and we will not discuss SHAP in this talk.

Toy example

- We generate data set $X \in \mathbf{R}^{20\times 3}$ and labels $y \in \mathbf{R}^{20}$.
- We split the dataset into 10 training and 10 testing points.
- The data set has three features x_1, x_2, and x_3 corresponding to the columns of X.

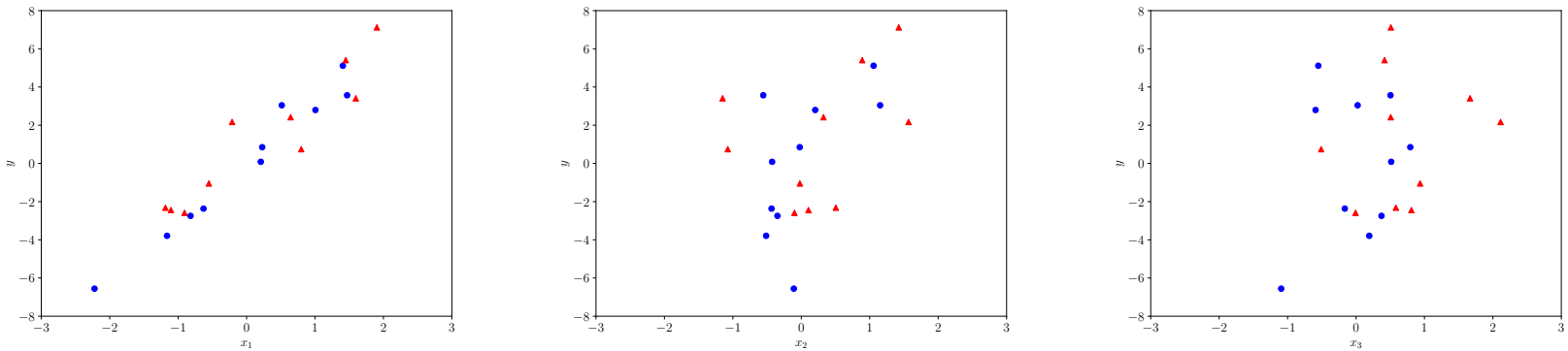

Figure: Labels y on the vertical axis plotted against features x_1, x_2, and x_3 on the horizontal axis for the generated data set. Blue points are training points, and red triangles are testing points.

Toy example

- We fit a linear model on the training data via least squares.
- Our linear model attains a coefficient of determination (R^2) of $.96$ on the testing data.
- The feature x_1 seems to correlate strongly with y, while x_2 and x_3 seem almost uncorrelated with y.
- A linear model that only considers x_1 only attains an R^2 of $.60$.
- This suggests x_2 and x_3 are still substantially important.

How can we systematically determine the importance of the features?

Least squares

Suppose we have training data $X \in \mathbf{R}^{N \times p}$ and labels $y \in \mathbf{R}^N$ with N observations and p features. We assume that $N \gg p$ and X is full rank.

- We can fit a linear model to the data by finding the $\theta^\star \in \mathbf{R}^p$ that minimizes
$$\|X\theta - y\|_2^2.$$
- The optimal parameter $\theta^\star$ is given by the formula
$$\theta^\star = (X^T X)^{-1} X^T y.$$
- How good is our model?

How good is our model?

If we have M test data points $X^{\text{tst}} \in \mathbf{R}^{M \times p}$ and labels $y^{\text{tst}} \in \mathbf{R}^{M}$, then we can measure the performance of our model by computing the R^2 statistic of our model,

$$R^2 = \frac{\|y^{\text{tst}}\|_2^2 - \|X^{\text{tst}}\theta^\star - y^{\text{tst}}\|_2^2}{\|y^{\text{tst}}\|_2^2}.$$

How valuable is each of our features?

The R^2 gives us the performance of our model.

- How do we extract feature importance from the R^2?
- R^2 is not separable in θ.

We propose a method for determining feature importance for least-squares models.

Shapley attribution

- In 1952, Lloyd Shapley introduced **Shapley attribution**, a method for splitting a reward granted to a coalition of cooperative players.
- The value that Shapley attribution assigns to each player is the player's **Shapley value**.
- Shapley attribution can also be used to attribute the R^2 of a model across the features.

Why use Shapley values?

Shapley attribution is provably the only scheme of attribution for which the following three key desiderata hold.

- **Full attribution**: The sum of the players' Shapley values is the reward granted to the coalition.
- **Fairness**: Permuting players within the coalition does not change their Shapley values.
- **Monotonicity**: If the reward is changed such that a player's marginal contribution to any coalition increases, then their Shapley value must increase.

Computing Shapley values

We define:

- A feature chain is a sequence of sets $\mathcal{S}_1, \ldots, \mathcal{S}_p$ such that $\emptyset = \mathcal{S}_0 \subset \mathcal{S}_1 \subset \cdots \subset \mathcal{S}_p = \{1, \ldots, p\}$ and $|\mathcal{S}_k| = k$.
- π is the unique permutation of $\{1, \ldots, p\}$ defined by $\mathcal{S}_1, \ldots, \mathcal{S}_p$ such that $\mathcal{S}_k = \{\pi_1, \ldots, \pi_k\}$.
- $\mathcal{P}$ is the set of all $p!$ permutations of $\{1, \ldots, p\}$.
- $R^2_{\mathcal{S}}$ is the test R^2 computed with the solution $\theta^\star_{\mathcal{S}}$ to the problem

$$\begin{array}{ll} \text{minimize} & \|X\theta - y\|_2^2 \\ \text{subject to} & \theta_j = 0, \quad j \notin \mathcal{S}. \end{array}$$

- $L : \mathcal{P} \to \mathbf{R}^p$ is the function such that $L(\pi)_j = R^2_{\mathcal{S}_j} - R^2_{\mathcal{S}_{j-1}}$.

Computing Shapley values

The vector of Shapley attributions S is given by

$$S = \frac{1}{p!} \sum_{\pi \in \mathcal{P}} L(\pi).$$

Requires retraining model 2^p times to compute!

Least-squares Shapley performance attribution

We present least-squares Shapley performance attribution (LS-SPA), an efficient method for estimating Shapley attributions for least-squares problems.

Approximating Shapley values

We can efficiently and accurately estimate Shapley values for least-squares problems. We define $\Pi \subset \mathcal{P}$ with $|\Pi| = K \ll p!$. We compute an estimate $\hat{S}$ of the Shapley attribution to be

$$\hat{S} = \frac{1}{K} \sum_{\pi \in \Pi} L(\pi).$$

We define the **relative error** of our estimatated Shapley attribution $\hat{S}$ to be

$$\frac{\|S - \hat{S}\|_2}{\|S\|_2}.$$

Sampling permutations

- If we sample Π uniformly at random from $\mathcal{P}$, then $\hat{S}$ is a **Monte Carlo** (MC) approximation of S.
- We can use a **quasi-Monte Carlo** (QMC) method to sample Π.
  - QMC methods generate a sequence of points in a manner designed to uniformly cover the sample space.
  - We generate QMC samples by generating a Sobol' sequence in $[0, 1]^p$ and, for each sample vector, choosing the permutation of indices which sorts its entries.
- MC estimation of Shapley values is not novel.

Efficiencies from least squares

For least-squares problems specifically, we can exploit the model structure to more efficiently estimate Shapley values. If $\theta^\star = \operatorname{argmin}_\theta \|X\theta - y\|_2^2$ and $X = QR$ is the QR decomposition of X, then

$$\theta^\star = R^{-1}Q^T y.$$

Initial size reduction

The problem

$$\text{minimize} \quad \|X\theta - y\|_2^2$$

has dimension $N \times p$. We observe that

$$\|X\theta - y\|_2^2 = \|R\theta - Q^T y\|_2^2 + \|y - Q(Q^T y)\|_2^2.$$

The term $\|y - Q(Q^T y)\|_2^2$ is constant, and

$$\text{minimize} \quad \|R\theta - Q^T y\|_2^2$$

is a problem with dimension $p \times p$.

QR factorization advantages

- Computing a QR decomposition is very expensive for solving a least-squares problem.
- If we expect to solve our least-squares problem more than p times, then the cost is justified.
  - $O(Np^2)$ to compute QR.
  - $O(Np^2)$ to find $\operatorname{argmin}_\theta \|X\theta - y\|_2^2$.
  - $O(p^3)$ to find $\operatorname{argmin}_\theta \|R\theta - Q^T y\|_2^2$.

Efficiently computing lifts

We consider the case where $\pi = (1, \ldots, p)$. Let $\tilde{X} \in \mathbf{R}^{p \times p}$ and $\tilde{y} \in \mathbf{R}^p$ be a reduced data matrix and labels vector. Suppose we want to find the solution $\theta_k^\star$ to

$$\begin{array}{ll} \text{minimize} & \|\tilde{X}\theta - \tilde{y}\|_2^2 \\ \text{subject to} & \theta_j = 0, \quad j > k \end{array}$$

for $k = 1, \ldots, p$.

- Solving problem p times directly costs $O(p^4)$ flops.
- We can do better.

Efficiently computing lifts

If we let $\tilde{Y} = \tilde{y}\mathbf{1}^T$, then finding $\theta_k^\star$ for $k = 1, \ldots, p$ is equal to finding $\Theta^\star \in \mathbf{R}^{p \times p}$ that is the solution to

$$\begin{array}{ll} \text{minimize} & \|\tilde{X}\Theta - \tilde{Y}\|_2^2 \\ \text{subject to} & \Theta \text{ upper triangular.} \end{array}$$

Let $\tilde{X} = \tilde{Q}\tilde{R}$ be the QR decomposition of $\tilde{X}$. Then, it is a property of the QR decomposition that

$$\Theta^\star = \tilde{R}^{-1}\mathbf{triu}(\tilde{Q}\tilde{Y}).$$

- Intuitively, this holds because $\tilde{R}\Theta^\star$ will be upper triangular, and $\mathbf{triu}(\tilde{Q}\tilde{Y})$ is the "closest" an upper triangular matrix can be to $\tilde{Q}\tilde{Y}$.
- Computing $\tilde{R}^{-1}\mathbf{triu}(\tilde{Q}\tilde{Y})$ takes $O(p^3)$ flops.
- From here, compute $L(\pi)$ using columns of $\Theta^\star$.

Performance uplift

- Direct computation of the Shapley value is simply "impossible" for $p > 20$.
- Using QR to reduce the size of the problem, we "pay" $O(Np^2)$ to reduce our per lift cost from $O(Np^3)$ to $O(p^4)$.
- We use a QR trick in computing each lift to reduce the per-lift cost from $O(p^4)$ to $O(p^3)$.
- We have to compute one lift vector for each of the K permutations we sample.
- The naïve method for estimating Shapley values costs $O(KNp^3)$ flops.
- LS-SPA takes $O(Np^2 + Kp^3)$ flops.

We assume that we have more training data than testing data ($N \geq M$). In practice, N is often very large (we have lots of data), and p is relatively small (we have much fewer features).

Using LS-SPA

```
from ls_spa import ls_spa

attrs = ls_spa(X, y, X_tst, y_tst)
```

All the math disappears!

Numerical example

We generated a dataset with $p = 100$ features, $N = 10^5$ training data points, and $M = 10^5$ testing data points. We demonstrate the rate of decay in the relative error of LS-SPA as more random samples are used.

Numerical example

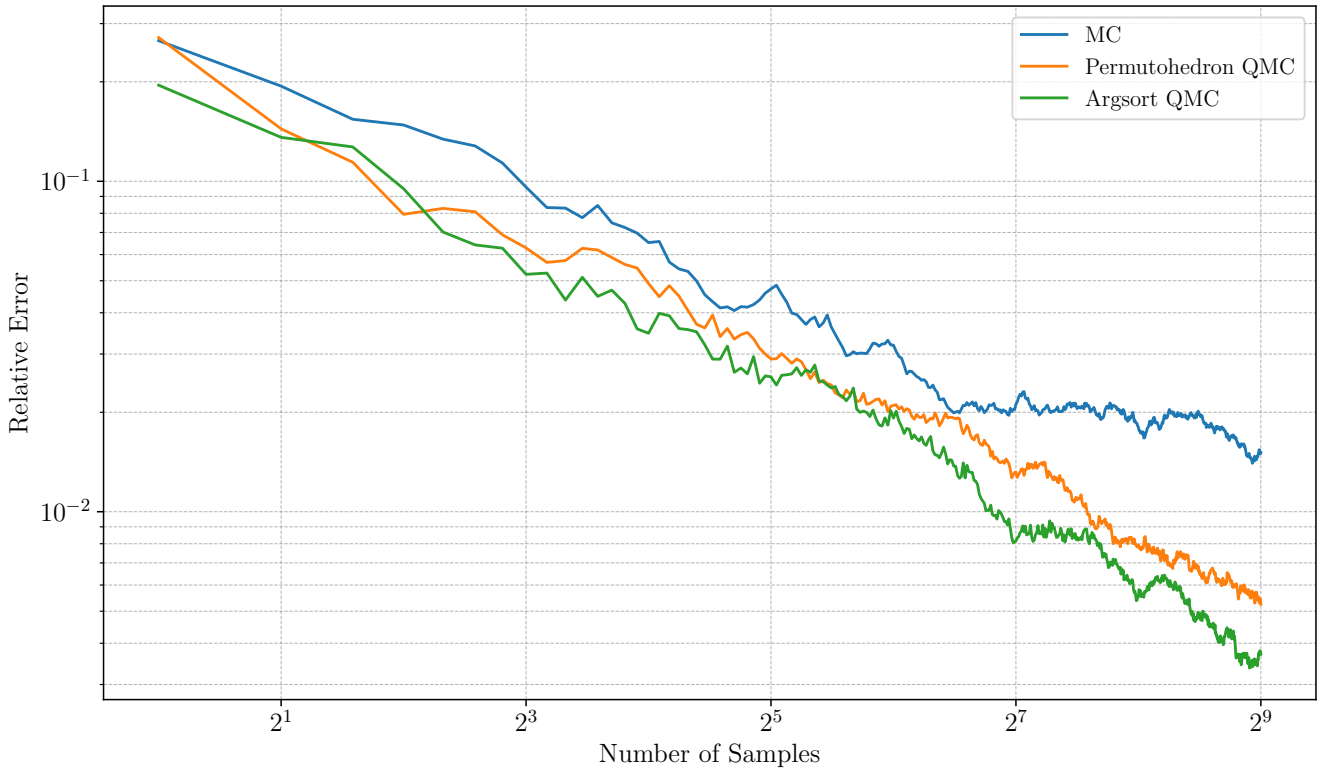

Figure: Relative error of LS-SPA versus number of samples on the generated dataset using different methods to sample feature chains.

Numerical example

We were able to execute LS-SPA on this dataset with 2^9 samples in 2.6 seconds on a laptop. Ground truth was computed by running LS-SPA with 2^{28} samples.

Acknowledgements

We thank Ron Kahn for suggesting the topic, and Kunal Menda for recommending the use of quasi-Monte Carlo. We thank Trevor Hastie and Thomas Schmelzer for useful comments and suggestions.