

A Multilinear Sampling Algorithm to Estimate Shapley Values

Ramin Okhrati¹ and Aldo Lipani^{2,3}

¹Institute of Finance and Technology, ²Space Time Lab and ³Web Intelligence Group - University College London



Introduction

Due to the growing complexity of machine learning models explaining them is becoming important aspect of data science.

Among many methods, Shapley values is the most popular method to measure the contribution of features.

Their popularity is due to their solid mathematical foundation, and second, the satisfaction of certain desirable properties like efficiency.

Shapely Values

Definition

Given an instance X , feature x_j , and a machine learning model ν , we define its contribution towards $\nu(X)$ as the Shapley value of the feature x_j for ν , that is:

$$S_j(\nu) = \sum_{A \subseteq X \setminus \{x_j\}} \frac{|A|! (n - |A|)!}{(n+1)!} (\nu(A \cup \{x_j\}) - \nu(A)),$$

where $|A|$ is the cardinality of the set A , and with some abuse of notation, $\nu(A \cup \{x_j\})$ and $\nu(A)$ must be understood as the evaluation of ν for the corresponding tuples obtained respectively from $A \cup \{x_j\}$ and A , through replacing a missing feature by zero in the tuples.

This formula reveals that the computational complexity of Shapley values is exponential.

Estimation Methods

Shapely estimation methods are divided into three types:

1. semi-closed form solutions, based on the central limit theorem;
2. data-driven methods such as regression and linear based techniques, quantitative input influence approaches, and DASP;
3. statistical sampling methods like the Castro's sampling algorithm that in general becomes computationally expensive as the number of feature increases, however, this has the advantage of converging to the exact Shapley values.

In this work, we provide a statistical sampling type algorithm based on a multilinear extension technique as applied in game theory.

(Halved) Owen Sampling Algorithm

Input: $\nu, X = (x_0, x_1, \dots, x_n), Q, M$
Output: $\hat{S}(\nu) = (\hat{S}_0(\nu), \hat{S}_1(\nu), \dots, \hat{S}_n(\nu))$

$\hat{S}_j(\nu) := 0$, for all j
for $q = 0, 1/Q, 2/Q, \dots, 1$ **do**
 $e_j := 0$, for all j
 for $m = 1, 2, \dots, M$ **do**
 $I_m^{(q)} \leftarrow (b_j : b_j \sim \text{Bern}(q), \text{ for all } j)$
 $h_{m,j} \leftarrow \nu(I_m^{(q)} \odot X + X^{(j)}) - \nu(I_m^{(q)} \odot X)$, for all j
 $e_j \leftarrow e_j + h_{m,j}$, for all j
 end for
 $\hat{S}_j(\nu) \leftarrow \hat{S}_j(\nu) + e_j$, for all j
end for
 $\hat{S}(\nu) \leftarrow \hat{S}(\nu)/(QM)$, for all j

Halved Version

$q = 0, 1/Q, 2/Q, \dots, 0.5$
 $I_m^{(q)} \leftarrow (b_j : b_j \sim \text{Bern}(q), \text{ for all } j)$
 $I_m^{(-q)} \leftarrow 1 - I_m^{(q)}$
 $h_{m,j}^{(q)} \leftarrow \nu(I_m^{(q)} \odot X + X^{(j)}) - \nu(I_m^{(q)} \odot X)$, for all j
 $h_{m,j}^{(-q)} \leftarrow \nu(I_m^{(-q)} \odot X + X^{(j)}) - \nu(I_m^{(-q)} \odot X)$, for all j
 $e_j \leftarrow e_j + h_{m,j}^{(q)} + h_{m,j}^{(-q)}$, for all j

Experimental Setup

Datasets

Credit Card Dataset (CC). This is a financial dataset made of a total number of 29,351 observations where each observation is composed of 23 features and a binary target variable. The features are either financial (such as pay related information) or non-financial like age. The target variable is either zero or one with one indicating the default of the credit card account.

Modified NIST (MNIST). This is a large database of handwritten digits that is commonly used for training and testing machine learning models. Each sample is a black and white image of a handwritten digit. Furthermore, the black and white images are normalized to fit into a 28x28 pixel bounding box. The MNIST dataset contains 70,000 images.

MLPs and Training

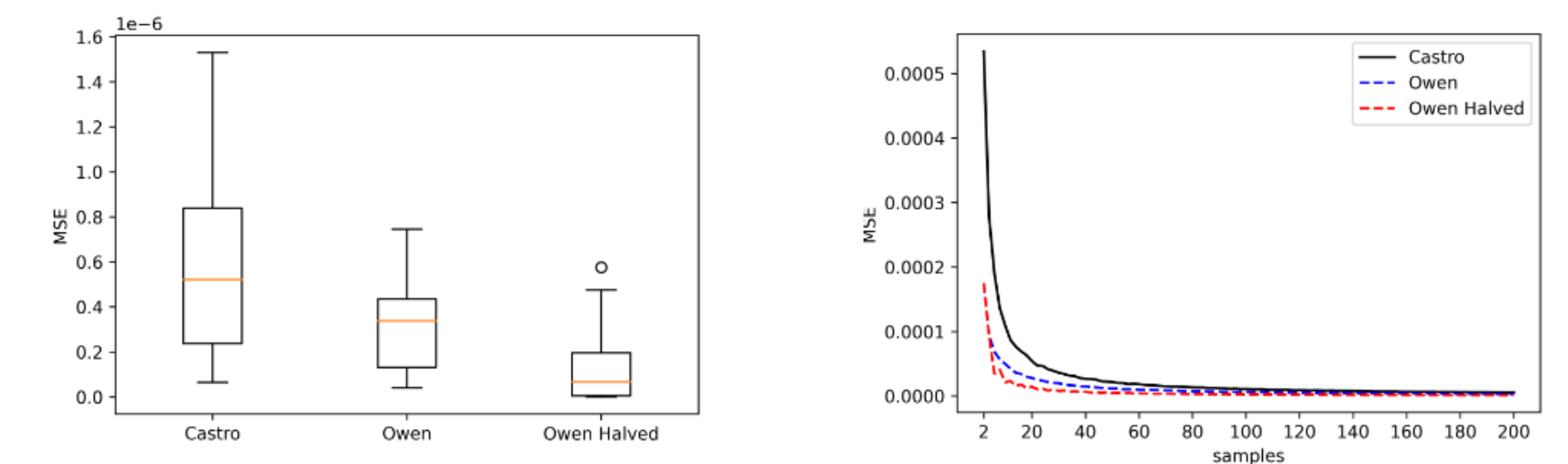
For the **CC** dataset, we sample 1000 models with a number of hidden layers from 0 to 3 and a number of neurons for each hidden layer from 1 to 15. The best model has 2-hidden layers with 13 and 9 neurons each. The accuracy of this model on the CC test set is **0.8247**.

For the **MNIST** dataset, we sample 1000 models with a number of hidden layers from 0 to 3 and a number of neurons for each hidden layer from 25 to 500 at multiples of 25. The best model has 2-hidden layers with 300 and 25 neurons each. The accuracy of this model on the MNIST test set is **0.9818**.

Experiments & Results

We compare against the Castro's Algorithm.

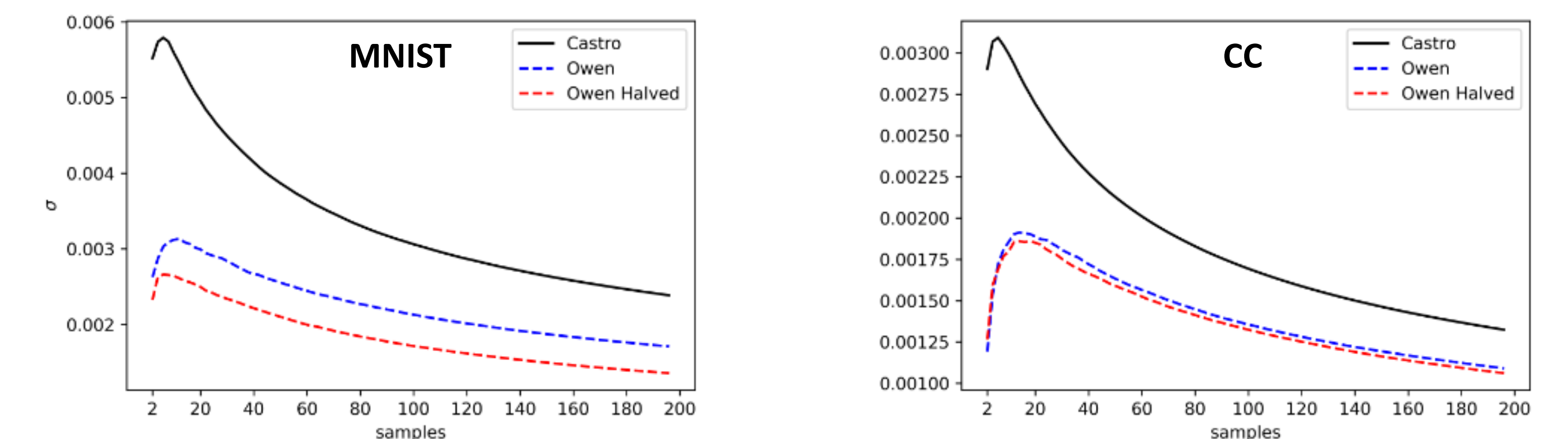
We first measure the MSE against the exact Shapley values computed on the **CC** dataset:



Summary of the results, averaged over 50 randomly selected examples:

Algorithm	Parameters	MSE (10^{-6})	Time (ms)
Castro	$M_c = 2000$	0.5575	3.004
Owen	$M = 2, Q = 1000$	0.3184	1.044
Halved Owen	$M = 2, Q = 1000$	0.1207	0.968

Then, we perform a variance analysis on both datasets. The results show that our estimators admit a lower variance comparing to that of Castro's one, hence a more accurate estimation:



Both plots are based on 50 randomly selected examples of the test sets.

Conclusion & Future Work

We have provided a sampling algorithm to efficiently estimate Shapley values that can be also used as a ground truth for comparison purposes. The method takes advantage of a variance reduction method and provides more accurate estimations for the Shapley values.

More experimental analysis on different datasets could be carried out on more complex deep learning architectures than MLPs, since our algorithm could work with any machine learning model.

The accuracy of our algorithm is controlled by two parameters, however, in our analysis, we set one of the parameters to 2. A more efficient and smart combinations of these parameters might improve the performance of the algorithm.