RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] If you like Shapley, then you'll love the core

Anes Benmerzoug[1] and Miguel de Benito Delgado[2]
[1]appliedAI Initiative GmbH, Munich, Germany – [2]appliedAI Institute gGmbH, Munich, Germany

Edited by
Koustuv Sinha,
Maurits Bleeker,
Samarth Bhargav

Received
04 February 2023

Published
20 July 2023

DOI
10.5281/zenodo.8173733

Reproducibility Summary

We investigate the results of [1] in the field of data valuation. We repeat their experiments and conclude that the (Monte Carlo) Least Core is sensitive to important characteristics of the ML problem of interest, making it difficult to apply.

Scope of Reproducibility – We test all experimental claims about Monte Carlo approximations to the *Least Core* and their application to standard data valuation tasks.

Methodology – We use the open source library [2] for all valuation algorithms. We document all details on dataset choice and generation in this paper, and release all code as open source in [3].

Results – We were able to reproduce the results on Least Core approximation. For the task of low-value point identification we observed similar performance for least core and (Truncated Monte Carlo) Shapley values, whereas for high-value identification, the least core outperformed other methods. In two experiments, we must depart from the original paper and arrive at different conclusions. Overall, we find that the Least Core offers similar results to other game-theoretic approaches to data valuation, but it does not overcome the main drawbacks of computational complexity and sensitivity to randomness that such techniques have.

What was easy – Open source libraries like DVC and ray enabled efficiently designing and running the experiments.

What was difficult – Data generation was difficult for dog-vs-fish because no code was available. Computing the Monte Carlo Least Core was very sensitive to the choice of utility function. Reproducing some experiments was difficult due to lack of details.

Communication with original authors – We asked the authors for details on the experimental setup and they kindly and promptly sent us the code used for the paper. This was very useful in understanding all steps taken and in uncovering some weaknesses in the experiments.

Correspondence should be addressed to Miguel de Benito Delgado (m.debenito@appliedai-institute.de)
The authors have declared that no competing interests exist.
Code is available at https://github.com/aai-institute/mlrc22-like-shapley-love-the-core. – SWH swh:1:dir:294da04ace110a1e2944203314f968a0bbf3c0a1.
Open peer review is available at https://openreview.net/forum?id=vWzZQAahuW.

1 Introduction

Data Valuation in Machine Learning (ML) is commonly understood as the task of assigning a scalar *value* to samples in a training set $\mathcal{D}_{\text{train}}$ which reflects their usefulness for a ML algorithm. Being a form of credit (or *payoff*) assignment, a game-theoretic concept called *Shapley value* (SV), which is unique in fulfilling several reasonable axioms, has established itself as a popular means of computing such a *valuation function* $\phi : \mathcal{D}_{\text{train}} \to \mathbb{R}$. Valuation methods are interesting for applications of model debugging, data acquisition, outlier detection and active learning. We refer to [4] for a recent survey.
The **Original Paper** [1] (OP in the sequel) proposes an alternative game-theoretic solution concept, the *Least Core* (LC) for the task of valuation. The LC dispenses with some of the properties that SV fulfils and whose relevance in ML is contentious,[1] but has instead the property of *Coalitional Rationality* (see the discussion in Section 5.3).
The main obstacle with combinatorial methods stemming from game theory is the cost of exact computation, which is exponential in the number of training samples. For this reason, much work revolves around Monte Carlo approximations, as is the case of the OP.

1.1 Notation and definitions

We follow the notation of the OP and set $N = \mathcal{D}_{\text{train}}$, $n = |N|$ and $\boldsymbol{x} : N \to \mathbb{R}$ the valuation function, which we identify with a vector $x \in \mathbb{R}^n$. The utility function $v : 2^N \to \mathbb{R}$ is the *score* of a ML model trained on some $S \subseteq N$, evaluated on unseen data $\mathcal{D}_{\text{val}}$. For some tasks, an additional held-out set $\mathcal{D}_{\text{test}}$ is used. The **least core** (LC) is the set of solutions $x \in \mathbb{R}^n$ to the following minimisation problem:

$$\begin{array}{rcll} \min e & & s.t. & \\ \sum_{i \in N} x_i & = & v(N) & \\ \sum_{i \in S} x_i + e & \geqslant & v(S) & \forall S \subseteq N. \end{array} \tag{1}$$

Each of these is a complete valuation function, or **payoff**. The scalar $e > 0$ is known as **subsidy** and the set of all solutions as *e-core*. The 0-core is known simply as the core. Because (1) has $1 + 2^n$ constraints, the authors propose to solve a reduced problem with only $m \ll 2^n$:

$$\begin{array}{rcll} \min\limits_{e>0} e & & s.t. & \\ \sum_{i \in N} x_i & = & v(N) & \\ \sum_{i \in S_j} x_i + e & \geqslant & v(S) & S_j \sim \mathcal{D}, \quad j \in \{1, \ldots, m\}, \end{array} \tag{2}$$

where $\mathcal{D}$ is any distribution over the power set of N.[2] After obtaining an optimal $\hat{e}$ in (2), the so-called **egalitarian least core** is selected from the $\hat{e}$-core by minimizing the ℓ_2 norm:

$$\begin{array}{rcll} \min \|x\|_2 & & & \\ \sum_{i \in N} x_i & = & v(N) & \\ \sum_{i \in S} x_i + \hat{e} & \geqslant & v(S) & \forall S \subseteq N. \end{array} \tag{3}$$

We will denote the algorithm solving (2), then (3), Monte Carlo Least Core (MCLC). In order to connect the solution of (3) with that of (1), the authors define the following relaxation: Let $e^\star$ be the optimum in (1), a payoff x is in the δ-**approximate least core** iff

$$\mathbb{P}_{S \sim \mathcal{D}} \left(\sum_{i \in S} x_i + e^\star \geqslant v(S) \right) \geqslant 1 - \delta. \tag{4}$$

[1] In particular the axiom of *linearity*.
[2] Because N might not be among the S_j, one must enforce non-negativity of the subsidy e.

2 Scope of reproducibility

The first theoretical result (OP Theorem 1) is a sample bound for the reduced problem (2) which is polynomial in n: using $m = \mathcal{O}\left(\left(n + \log \frac{1}{\Delta}\right) \delta^{-2}\right)$ samples, there is a $(1-\Delta)$-probable guarantee of obtaining a δ-approximate least core. The experimental claim (OP Section 5.1 and Claim 1 below) is that this bound translates to an effective algorithm for computing the LC. It is substantiated by an experiment in feature valuation using three public datasets.[3]
The second theoretical result is a further relaxation adding a slack variable (*subsidy*) to the approximate constraint (4), which becomes $\mathbb{P}_{S\sim D}\left(\sum_{i\in S} x_i + e^\star + \varepsilon \geqslant v(S)\right) \geqslant 1-\delta$. The set of payoffs for which this holds is called (ε, δ)-**probably approximate least core**. For this condition a new sample bound which is *logarithmic* in n is obtained in OP, Theorem 2. There is however no corresponding experimental claim related to this result. We discuss this in Section 5.2.
The second set of experiments attempts to verify that MCLC payoffs outperform SV for data valuation tasks. These are Claims 2 and 4 below. An additional experiment checks how MCLC handles noisy data, cf. Claim 3.
The third and final theorem of the paper is a minimal sample complexity of a similar relaxation for a further solution concept, the *nucleolus*. Given the negative character of the result, which states the impracticality of the nucleolus, we did not verify it experimentally.
Finally, the authors comment on the relevance of the LC for ML applications. We discuss it in Section 5.3
To summarise, the main claims which we set to verify or refute are:

Claim 1 *Sub-sampling of constraints for* (1) *leads to a stable solution converging to the LC with high probability. See Section 4.1.1.*

Claim 2 *Under a limited computational budget, MCLC outperforms SV in best and worst sample removal. See Section 4.1.2.*

Claim 3 *MCLC assigns lower value to noisy data than to "clean" one, and the fraction of utility assigned to clean data increases as the noise level does. See Section 4.1.3.*

Claim 4 *Under a limited computational budget, MCLC outperforms SV in flipped label detection. See Section 4.1.4.*

3 Methodology

We verified all our implementation choices with the source code that the authors provided via email, to the extent that it matched the paper and to the best of our understanding. In what follows, we detail the OP's setup, and any departure will be explicitly mentioned. In particular, we remark that the OP's code for the solution of (2) includes an additional non-negativity constraint on the payoffs which we did not implement.
All experiments are repeated 10 times and 95% Normal confidence intervals estimated. Each experiment repetition uses either a new split for "natural" data or generates it anew in the case of synthetic data.

3.1 Model descriptions

We benchmark Truncated Monte Carlo Shapley (TMCS, [5]), Group Testing Shapley (GTS, [6]), Monte Carlo Least Core (MCLC), Leave One Out (LOO) and random values.

[3] Feature valuation refers to using features instead of samples as players in the coalitional game. An evaluation of the utility over a subset of features S corresponds to training the model on the whole $\mathcal{D}_{\text{train}}$ using only those features.

Experiment 1: δ-approximate least core — We use logistic regression with the LIBLINEAR solver and 100 maximum iterations. We test three scoring functions v for the utility: accuracy, average precision and F_1 score (the last two are not in the OP). We compute the MCLC with fixed constraint budgets of 1, 2, 5, 7.5, 10 and 15% of all 2^n constraints.

Experiment 2: Sample removal — We use a logistic regression model with the LIBLINEAR solver and 100 iterations. As scoring function for the utility we used accuracy. For data values we use MCLC, TMCS, GTS, LOO and random.

Experiment 3: Noisy data detection — We use logistic regression with the LIBLINEAR solver and 100 iterations. As scoring function for the utility we used accuracy. For valuation we use MCLC, random, TMCS, GTS and LOO (the last three are not in the OP).

Experiment 4: Fixing mislabeled data — We use a Gaussian Naive Bayes model and 3 scoring functions for the utility: accuracy, average precision and F_1 score. For valuation we use MCLC, random, TMCS, GTS and LOO (the last three are not in the OP).

Hyperparameter search — We did not run any.

3.2 Datasets

In all experiments we used the same datasets as in the paper or our best approximation. The dog-vs-fish dataset was unavailable. We release code to generate it as part of the experiments.

Experiment 1: δ-approximate least core — Train-test split in all three datasets was 80% / 20%. The OP's authors use only 11 features in every dataset, but we used all of them. All features were standardised.

- *House*: US Congressional Voting Dataset. 16 features. Additional pre-processing: NaN values imputed with most frequent ones, encoded categorical features and target labels.
- *Medical*: Breast Cancer Dataset. 9 features. Additional pre-processing: Dropped rows with NaNs, dropped ID column.
- *Chemical*: Wine Dataset. 13 features. Additional pre-processing: None.

Experiment 2: Sample removal —

- Synthetic Gaussian data with 2 classes, 200 training samples, 5000 testing samples. To generate, we sample from a 50-dimensional spherical Gaussian with parameters $\mu = 0$, $\Sigma = \text{Id}$. 50% of the features are selected at random and a sigmoid is applied to their sum. The 5200 resulting scalars are thresholded with as many $U(0, 1)$ values to generate the labels. All features were standardised.
- Dog vs Fish dataset with 2 classes, 600 imbalanced training samples, 600 imbalanced testing samples. Pre-processed with pre-trained INCEPTIONV3 model to produce embeddings in $\mathbb{R}^{2000}$. All features were standardised.

Experiment 3: Noisy data detection — Using the synthetic Gaussian dataset, we select 20% of the training samples and apply increasing noise to the covariates in multiples 0.5, 1, 2 and 3 of the standard deviation of the original data. This differs from the OP, where "noise level" is not clearly defined.[4]

[4] And from the original code, which *appends* data with *fixed noise* while *also flipping the labels*. We believe this transformation to confuse the experiment so we followed the procedure above.

Experiment 4: Fixing mislabeled data — We take 1000 samples from the Enron1 spam dataset [7] and pre-process with SCIKIT-LEARN's TfIdfVectorizer to convert the emails to a matrix of token counts. 30% of the data is reserved for $\mathcal{D}_{\text{test}}$, the remaining is split 70/30 into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$. We randomly flip 10%, 20% and 30% of the training set labels (as opposed to just 20% in the OP). We depart from the OPs implementation, which also adds noise to the covariates.[5]

3.3 Experimental setup and code

We used the open source library PYDVL [2] for its implementation of valuation methods and release the code for all our experiments at [3]. Environment setup is done with POETRY and easy experiment reproduction is ensured with DVC [8] pipelines. Detailed instructions are given in the repository's documentation.

3.4 Computational requirements

Single experiments without uncertainty estimates can run on a 2022 consumer laptop without a GPU in a few hours, but computing confidence intervals considerably increases cost. Because the problems are embarrasingly parallel, time scales almost linearly with the number of cores, so we used one high-cpu machine in GCP to repeat the experiments before submission.

4 Results

4.1 Results reproducing the original paper

Experiment 1: δ-approximate least core — The goal is to verify Claim 1. As in the OP, we uniformly sample $m_1 = 0.01 \times 2^n, \ldots, m_6 = 0.15 \times 2^n$ constraints and solve (2), then (3), obtaining payoffs $x^{(j)} \in \mathbb{R}^n$. The fraction of all 2^n constraints which is fulfilled by each $x^{(j)}$ is calculated. We are able to (almost) exactly reproduce OP, Figure 1, concluding that this gives credence to the procedure, see Figure 1. See however our discussion in Section 5.1.

Because we used every feature instead of just 11, we had higher scores across the board, hence lower utility variance, possibly leading to a smaller fraction of constraint violations overall wrt. the OP. In particular the maximal utility $v(N)$ is higher for all three datasets, leading to higher payoffs.

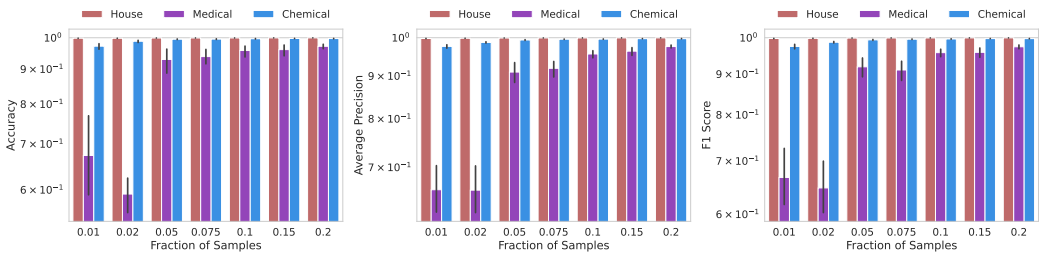

Figure 1. For all 3 utilities, as m_j grows, the approximate payoffs fulfil an increasing fraction of the whole set of constraints. δ in (4) corresponds to the distance from the top of the bars to $y = 1$. For a fixed height, Δ of the main theorem corresponds roughly to the portion of the black bar below it.

[5] We believe this to be a bug but are unable to verify it since the complete code for this experiment was unavailable.

Experiment 2: Sample removal — The goal is to verify Claim 2. Fixed budgets of 10K and 50K subsets $S \subseteq N$ are fixed for the methods TMCS, GTS and MCLC.[6] LOO requires only n evaluations of the utility and random values do not require a budget. Two sets of experiments are run after computing the payoffs. In the first, training samples are ranked from highest to lowest value, and in the second from lowest to highest. In each case, samples are successively removed from $\mathcal{D}_{\text{train}}$ in the corresponding order, the model is retrained, and its performance on $\mathcal{D}_{\text{test}}$ recorded.
Surprisingly, we observe that LOO performs comparably to the game-theoretic methods with the synthetic data, casting doubt over the adequacy of the choice of distribution.

Worst sample removal, Figure 2: For the synthetic data our results do not allow picking a winner, but GTS is clearly useless with so few subsets of N. For the natural data we observe clearer margins than the OP's (both for MCLC and TMCS) but note that theirs and ours are uninformative, representing just a 0.2% change in accuracy.

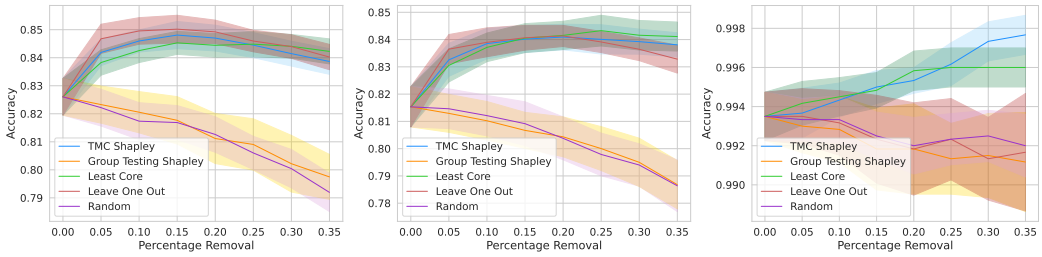

Figure 2. Worst sample removal. LTR: synthetic data 10K and 50K subsets, natural data 10k subsets.

Best sample removal, Figure 3: For the synthetic data our results qualitatively follow those of the OP, except for LOO as noted. For the natural data we observe similar trends to the OP, with MCLC marginally better than TMCS, but again by an inconclusive 0.5%.[7]

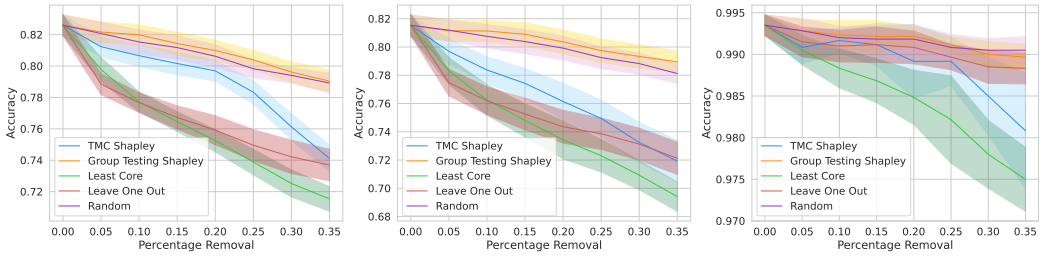

Figure 3. Best sample removal. LTR: synthethic data 10K and 50K subsets, natural data 10k subsets.

Experiment 3: Noisy data detection — The goal is to test Claim 3. We defined our own experiment:[8] For each noise level we compute: the fraction of noisy samples among those with the lowest 20% assigned payoffs, the percentage of total utility assigned to the clean data, and the sum of all values assigned to noisy, clean and all data.
MCLC and TMCS perform comparably (Figure 4), but even at 3σ noise both detect less than 50% of the noisy samples. Furthermore, the percentage of "clean" utility is approximately constant wrt. noise level, refuting Claim 3. Interestingly, the distribution of

[6] Time constraints forced us to leave the 50K experiment with dog-vs-fish out, but we were observing similar trends before interrupting it.
[7] However, at a buget of 5K, we observe that TMCS outperforms MCLC, in contrast to Claim 2.
[8] The OP lacked details, and their code seemed to lack this particular experiment.

values is insensitive to the noise (Figure 5), so that the identification of samples is due to different rankings, and not because of more extreme values.

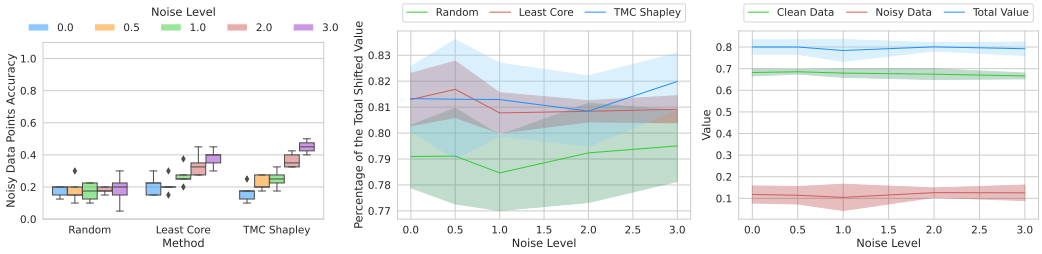

Figure 4. LTR: a) Fraction of noisy samples among the lowest 20% values; b) Percentage of value assigned to noisy data, values were shifted up to be non-negative; c) Sum of values for each subset (MCLC).

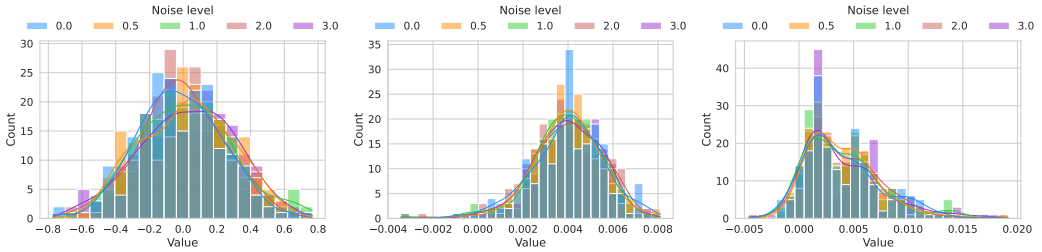

Figure 5. Histograms of values for increasing noise levels. LTR: Random, MCLC and TMCS.

Experiment 4: Fixing mislabeled data — The goal is to verify Claim 4. For each amount of flipped data, the payoffs are computed using 5K elements of 2^N. Training points are ranked by increasing value and successively removed from $\mathcal{D}_{\text{train}}$, retraining with every removal and evaluating performance on $\mathcal{D}_{\text{test}}$.
We could not reproduce the OP using accuracy as score. For average precision and F_1, we did observe the reported behaviour for MCLC, but it was matched and even outperformed by TMCS. The same happened for detection rate, with TMCS around 50% better than MCLC. Given our experience in Section 4.1.2 we believe that the choice of 5K constraints might be too low for MCLC.

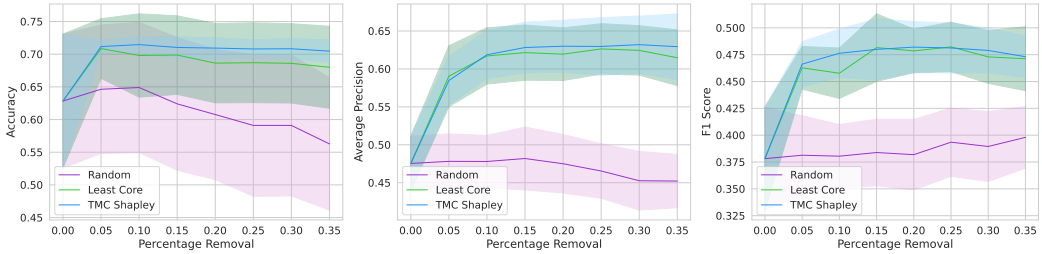

Figure 6. Performance after worst-sample removal for three scoring functions and 30% flipped labels.

4.2 Results beyond original paper

As reported above: we tested several scoring functions for the utility besides accuracy; we repeated the experiments more times and computed confidence intervals for all

quantities; we computed detection rates for noisy covariates and flipped labels in Sections 4.1.3 and 4.1.4.

5 Discussion

The major conclusion we extract from our experiments is that MCLC performs better in identifying high value points, as measured by best-sample removal tasks. In all other aspects, it performs worse or similarly to TMCS at comparable budgets. It is also more computationally intensive because of the solution of large linear and quadratic optimization problems. For this reason we would select (some variation of) SV for outlier detection, and perhaps MCLC for the selection of interesting points, but we believe that more benchmarks are required.
One major point left out by the OP and us is the effect that the variance of the utility has on the values computed. This can be measured by the concept of "rank stability", defined in [9]. In our opinion it, or some variation thereof, should be part of all benchmarks in data valuation. To isolate the effect, one should use deterministic utilities too.
Finally, we note that differences in datasets, data splits and model training have repeatedly led to different (higher in our case) model scores, and hence values. For instance, despite generating the synthetic data with what we believed to be the exact same procedure, we have had around 5 percentage points higher accuracy in several experiments.

5.1 Computational cost of the δ-approximate least core

Our main observation about the experiment in Section 4.1.1 is the following: The core (pun intended) of the OP is Claim 1, namely that it is enough to solve (2) with $m = \mathcal{O}(n/\delta^2)$ sufficiently large to fulfil to have a set of payoffs which is close to the exact $e^\star$-LC in the sense of (4). We find that their experiment only tackles this in a loose way. As one uses more constraints for (2), the fraction of *all* constraints fulfilled does increase. But no investigation is done of the constants hidden in the asymptotic bound. As the OP points out, this can be seen to be of minor relevance since it is $\mathcal{O}(n)$, but it is relevantfor low n, especially if, as we suspect, the constant hides a dependency on the range of the utility as is typically the case in such bounds, even in OP, Theorem 2.
We also note that this experiment could have been done in data valuation using a small dataset, e.g. less than 20 samples. Finally, we believe that a deterministic utility would be better suited for the verification of the $(1 - \Delta)$-probable solution, because it would avoid the additional randomness inherent to ML training.

5.2 Applicability of the (ε, δ)-approximate least core

The logarithmic sample bound is a result of relaxing (4), not of a change in the problem formulation. Values of $\varepsilon > 0$ increase the fraction of sets for which the weaker constraints are satisfied, effectively decreasing δ. This is potentially analogous to increasing δ while fixing $\varepsilon = 0$. It would be interesting to know how both are related.
But crucially, it is unclear how to use this bound in practice. What is a reasonable value for the slack ε and what is the corresponding sample size? How does this affect the ranking of payoffs and consequently sample removal, outlier detection, etc.? We believe that these questions should be addressed if the concept of (ε, δ)-least core is to have any practical application.

5.3 The least core in ML

In OP, Section 4, the authors discuss the relevance of the LC in machine learning and how it compares to SVs. Their arguments in favour of the LC are:

1. The LC fulfils the same axioms as SVs except for linearity. However this defficiency is inconsequential if one evaluates the model on the whole test set instead of sample by sample.

2. The stability property (Coalitional Rationality) aligns with human expectations of fairness better than SVs, and enables using the LC for actual economic compensation of data providers because it produces payoffs which are *plausible,* in the sense that "every coalition is compensated for at least its market value".

3. SVs suffer from complexity results: exact computation is known to be NP-complete, and there are hardness results for confidence intervals. This makes Monte Carlo approximations necessary.

We agree with the first point made and do not elaborate on it.
As to the second point, one can ask whether Coalitional Rationality really matters in ML applications. This property ensures that every coalition is credited at least as much as it is worth in terms of the given utility. This is allegedly of interest when paying multiple data vendors: as a buyer one would like a credit assignment scheme that encourages providing data. However, we see a major technical dificulty in the application of data values to payments: they tend to concentrate around 0, with only the extremes significantly different. This is enough for worst/best data identification, but assigns very uninformative values to most points. Even more so, taking into account rank instability due to the stochasticity of ML training procedures (i.e. running the valuation a second time can permute many samples).
Regarding the third item, Monte Carlo estimates of SVs are known to require $\mathcal{O}(n)$ subsets, but this is indeed not the case for the well-known TMCS, which employs heuristic stopping criteria. Nevertheless, we find that in practice this is not so much of an issue as are the stochasticity of the utility and its sensitivity to the size of the subset used. Generalizations of SV to *semi-values*, and in particular the Banzhaf index [9], exhibit provably robust behaviour wrt. these issues. In addition, the (ε, δ)-approximate algorithm of [9] includes an $\mathcal{O}(\log(n/\delta)/\varepsilon^2)$ sample bound for ℓ_∞-good approximations.

5.4 What was easy

Using open source libraries simplified and accelerated implementation.

5.5 What was difficult

With regards to the reproduction of the OP, we found several experiments difficult to interpret, even finding the need to design new ones instead. Our suggestion to the authors would be to always release code in a public repository clearly separating experiment configuration from execution, allowing reproduction, but also changes to the experiments just by modifying configuration files. There are multiple ways to achieve this, one of them being DVC [8], but there exist many others.
But perhaps the most difficult aspect of data valuation is the inherent stochasticity and lack of convexity of most ML training procedures. This yields a very noisy signal, making applications brittle. In particular, the choice of the scoring function is crucial, a fact reflected in most sample bounds in the literature, where its range typically appears as an additional quadratic factor.

5.6 Communication with original authors

The authors kindly replied to our first questions and request for code. Alas, a nearing deadline made any further communication impossible and we could not ask for feedback on our analysis.

References

1. T. Yan and A. D. Procaccia. "If You Like Shapley Then You'll Love the Core." In: **Proceedings of the 35th AAAI Conference on Artificial Intelligence, 2021**. Vol. 6. Virtual conference: Association for the Advancement of Artificial Intelligence, May 2021, pp. 5751–5759. DOI: 10.1609/aaai.v35i6.16721. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16721 (visited on 04/23/2021).
2. TransferLab Team. **pyDVL: The Python Data Valuation Library**. appliedAI Institute gGmbH. Version 0.4.0. 2022. URL: https://pypi.org/project/pyDVL/.
3. A. Benmerzoug. **Mlrc22 - If You like Shapley Then You'll Love the Core**. appliedAI Institute gGmbH. 2022. URL: https://github.com/aai-institute/mlrc22-like-shapley-love-the-core.
4. R. H. L. Sim, X. Xu, and B. K. H. Low. "Data Valuation in Machine Learning: "Ingredients", Strategies, and Open Challenges." In: **Thirty-First International Joint Conference on Artificial Intelligence**. Vol. 6. July 2022, pp. 5607–5614. DOI: 10.24963/ijcai.2022/782. URL: https://www.ijcai.org/proceedings/2022/782 (visited on 01/29/2023).
5. A. Ghorbani and J. Zou. "Data Shapley: Equitable Valuation of Data for Machine Learning." In: **Proceedings of the 36th International Conference on Machine Learning, PMLR**. PMLR, May 2019, pp. 2242–2251. arXiv:1904.02868. URL: http://proceedings.mlr.press/v97/ghorbani19c.html (visited on 11/01/2020).
6. R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gürel, B. Li, C. Zhang, D. Song, and C. J. Spanos. "Towards Efficient Data Valuation Based on the Shapley Value." In: **Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics**. PMLR, Apr. 2019, pp. 1167–1176. URL: http://proceedings.mlr.press/v89/jia19a.html (visited on 02/12/2021).
7. V. Metsis, I. Androutsopoulos, and G. Paliouras. "Spam Filtering with Naive Bayes-Which Naive Bayes?" In: **3rd Conference on Email and Anti-Spam**. Mountain View, California USA, July 2006. URL: https://cir.nii.ac.jp/crid/1571135650462485632 (visited on 01/29/2023).
8. R. Kuprieiev et al. **DVC: Data Version Control - Git for Data & Models**. Zenodo. Jan. 2023. DOI: 10.5281/zenodo.7559368. URL: https://zenodo.org/record/7559368 (visited on 01/29/2023).
9. J. T. Wang and R. Jia. **Data Banzhaf: A Robust Data Valuation Framework for Machine Learning**. Oct. 2022. DOI: 10.48550/arXiv.2205.15466. arXiv:2205.15466. URL: http://arxiv.org/abs/2205.15466 (visited on 10/28/2022).