Improving Cooperative Game Theory-based Data Valuation via Data Utility Learning

Tianhao Wang
Princeton University
tianhaowang@princeton.edu

Yu Yang
Xi'an Jiaotong University
yangyu21@xjtu.edu.cn

Ruoxi Jia
Virginia Tech
ruoxijia@vt.edu

Abstract

The Shapley value (SV) and Least core (LC) are classic methods in cooperative game theory for cost/profit sharing problems. Both methods have recently been proposed as a principled solution for *data valuation* tasks, i.e., quantifying the contribution of individual datum in machine learning. However, both SV and LC suffer computational challenges due to the need for retraining models on combinatorially many data subsets. In this work, we propose to boost the efficiency in computing Shapley value or Least core by learning to estimate the performance of a learning algorithm on unseen data combinations. Theoretically, we derive bounds relating the error in the predicted learning performance to the approximation error in SV and LC. Empirically, we show that the proposed method can significantly improve the accuracy of SV and LC estimation.

1 Introduction

Data valuation aims to quantify the usefulness of each data source for machine learning (ML) tasks, which has many use cases and has gained lots of attention recently. For instance, it can inform the implementation of policies that enable individuals to control how their data is used and monetized by third parties (Voigt & Von dem Bussche, 2017). Moreover, data value allows users to filter out poor quality data and identify data that are important to collect in the future (Jia et al., 2019c; Ghorbani & Zou, 2019). Existing data valuation approaches leverage solution concepts from cooperative game theory, specifically the Shapley value (SV) and Least core (LC), as a fair notion for data value. However, computing or even approximating these data value notions requires evaluating the learning performance (i.e., retraining) on many different combinations of data sources, which could be very computationally expensive. In this work, we develop a general framework to improve the effectiveness of sampling-based SV or LC estimation heuristics. Specifically, we propose to learn to predict the performance of a learning algorithm for an input dataset (which we refer to as *data utility learning*) and use the trained predictor to estimate the learning performance on a dataset without retraining. As the predicted learning performance may be noisy, we derive the estimation guarantee of SV/LC under a hybrid of clean and noisy samples. Moreover, we conduct extensive experiments and show that data utility learning can significantly improve SV and LC approximation accuracy.

2 Background and Related Work

The goal of data valuation is to quantify the contribution of each training data point to a learning task. Game-theoretic formulations of data valuation have become popular recently. Formally, a cooperative game considers a set of *players* $N = \{1, \ldots, n\}$, and a *characteristic function* $v : 2^N \to \mathbb{R}$ assigns a value to every subset (often referred to as a *coalition*) $S \subseteq N$. In the context of data valuation, N corresponds to the dataset and each player $i \in N$ is a data point; the characteristic function v takes a subset of data points as input, and output the performance score (e.g., test accuracy) of a learning algorithm trained on the data subset. We will refer to v as *data utility function* in data valuation context, which stresses that v reflects the utility of the input dataset for the ML algorithm. Shapley value has been widely used as a data value notion (Ghorbani & Zou, 2019; Jia et al., 2019b;a;c; Wang et al., 2020), as it uniquely satisfies a set of desirable properties for profit sharing problems. The Shapley value $\phi(v)_i$ for data point i is defined as

$$\phi(v)_i = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \frac{1}{\binom{n-1}{|S|}} \left[v(S \cup \{i\}) - v(S) \right] \tag{1}$$

In a word, Shapley value is a weighted average of the marginal contributions of a data point over all possible subsets. Recently, Yan & Procaccia (2020) propose to use the Least core, another classic solution concept in cooperative game theory, as an alternative to Shapley value for data valuation. Least core vector $\psi \in \mathbb{R}^n$ minimizes the excess of the actual utility over the assigned value for every possible subset and is computed by solving the linear programming problem below:

$$\min_{\psi} e \quad \text{s.t.} \sum_{i=1}^{n} \psi_i = v(N), \sum_{i \in S} \psi_i + e \geq v(S), \forall S \subseteq N \tag{2}$$

The fairness properties of SV and LC provide strong motivation for using them in data valuation. We point readers to the cited references for a detailed discussion on it. However, from the definitions in (1) and (2), it can be seen that the exact calculation of both SV and LC requires evaluating the data utility function, i.e., retraining, on every possible subset of the training data, which is 2^n in total. Indeed, the exact computation of SV and LC is NP-hard in general (Deng & Papadimitriou, 1994; Faigle et al., 2000), which limits their applicability in real-world data valuation applications even at the scale of hundreds of data points. Several approximation heuristics, such as TMC-Shapley (Ghorbani & Zou, 2019) and KNN-Shapley (Jia et al., 2019a), have been proposed to approximate Shapley value. Despite their computational advantage, they are biased in nature, limiting their applicability to sensitive applications such as designing monetary rewards for data sharing or assigning responsibility for ML decisions. On the other hand, unbiased Shapley estimators such as Permutation Sampling (Castro et al., 2009) and Group Testing (Jia et al., 2019b) still require a large number of learning performance evaluations for any descent approximation accuracy.

The idea of learning characteristic function was previously proposed in Balcan et al. (2015), and they studied the PAC learnability of several well-known classes of cooperative games. Yan et al. (2020) is the closest work to our paper, where they use linear regression with interaction terms to learn characteristic functions and use the learned linear model to estimate SV. However, their technique cannot be directly applicable to data valuation due to many reasons, which we discuss in Appendix A.

3 DATA VALUATION VIA DATA UTILITY LEARNING

The major idea underlying the existing data valuation heuristics is to evaluate the data utility only on some sampled subsets and then estimate the data value based on the samples of subset-utility pairs (i.e., Monte Carlo or its variants). We will refer to the samples of subset-utility pairs as *utility samples*. The key idea of our approach is that, with the utility samples, we can potentially use a parametric model $\hat{v}$ to learn and approximate the data utility function v. We can then use $\hat{v}$ to predict the utility for additional subsets that are not sampled previously.

Sampling-based SV/LC Estimation Heuristics. We first build an abstraction for sampling-based data value approximation heuristics. Sampling-based heuristics comprise of all the existing *unbiased* heuristics (Castro et al., 2009; Jia et al., 2019b; Yan & Procaccia, 2020) as well as some of the biased heuristics (e.g., TMC Shapley (Ghorbani & Zou, 2019)). A sampling-based heuristics can be characterized by two components: a sampler $\mathcal{V}_{\text{samp}}$ and an estimator $\mathcal{V}_{\text{est}}$. Under some *training budget* $m_{\text{train}} > 0$, the heuristic sampler $\mathcal{V}_{\text{samp}}$ takes a dataset N and outputs a set of utility samples $\{(S_i, v(S_i))\}_{i=1}^{m_{\text{train}}}$ where each $S_i \subseteq N$ is sampled according to certain distributions. $v(S_i)$ is usually obtained by retraining models on S_i and obtain the performance scores of trained model. The heuristic estimator $\mathcal{V}_{\text{est}}$ then takes the utility samples and computes the estimation of the corresponding solution concept (i.e., Shapley or Least core). **Example-1: Permutation Sampling estimator for Shapley value.** The definition of Shapley value in (1) can also be conveniently expressed in terms of permutations $\phi(v)_i = \frac{1}{n!} \sum_{\pi \in \Pi(N)} [v(P_i^\pi \cup \{i\}) - v(P_i^\pi)]$ where $\Pi(N)$ is the collection of $n!$ permutations of N, and P_i^π represents the set of data points ranked lower than i in the permutation π. Since this is just the expectation of $v(P_i^\pi \cup \{i\}) - v(P_i^\pi)$ when π is uniformly sampled from $\Pi(N)$, an obvious approximation is the simple Monte Carlo estimator

$$\hat{\phi}(v)_i = \frac{1}{m_{\text{perm}}} \sum_{j=1}^{m_{\text{perm}}} \left[v(P_i^{\pi_j} \cup \{i\}) - v(P_i^{\pi_j}) \right] \tag{3}$$

for a uniform of samples $\pi_1, \ldots, \pi_{m_{\text{perm}}} \in \Pi(N)$. In this case, $\mathcal{V}_{\text{samp}}$ samples $\pi_1, \ldots, \pi_{m_{\text{perm}}} \in \Pi(N)$ and compute $v(P_i^{\pi_j} \cup \{i\})$ for each $i \in N$ (so the sampling budget $m_{\text{train}} \geq n m_{\text{perm}}$ as every permutation requires n times model retraining). $\mathcal{V}_{\text{est}}$ takes utility samples and estimate SV according to (3) for each $i \in N$. **Example-2: Monte Carlo estimator for Least Core.** Given m_{train} utility samples $\{(N, v(N))\} \bigcup \{(S_j, v(S_j))\}_{j=1}^{m_{\text{train}}-1}$ where each $S_j \subset N$, it is straightforward to give a

---

Algorithm 1: Algorithm for Accelerating The Estimation of Data Valuation Solution Concepts

---

input : Data valuation heuristic $(\mathcal{V}_{\text{samp}}, \mathcal{V}_{\text{est}})$, Dataset $N = \{1, \ldots, n\}$, Training budget m_{train},
Prediction budget m_{pred}, Learning Algorithm for Data Utility Model $\mathcal{A}$.
output : Estimated solution concept $x \in \mathbb{R}^n$.

1 Sample m_{train} data subsets and their utilities $\mathcal{S}_{\text{train}} = \{(S_i, v(S_i))\}_{i=1}^{m_{\text{train}}}$ with sampler $\mathcal{V}_{\text{samp}}$.
2 Train data utility model $\hat{v} \leftarrow \mathcal{A}(\mathcal{S}_{\text{train}})$.
3 Sample m_{pred} additional data subsets $\{S_i\}$ according to $\mathcal{V}_{\text{samp}}$, and obtain their *predicted* utilities
$\mathcal{S}_{\text{pred}} = \{(S_i, \hat{v}(S_i))\}_{i=1}^{m_{\text{pred}}}$.
4 Compute the corresponding solution concept $x \leftarrow \mathcal{V}_{\text{est}}(\mathcal{S}_{\text{train}} \cup \mathcal{S}_{\text{pred}})$.
5 **return** x

Monte Carlo estimator for Least core in (2) by solving the following linear program:

$$\min_{\psi} e \quad \text{s.t.} \sum_{i=1}^{n} \psi_i = v(N), \sum_{i \in S_j} \psi_i + e \geq v(S_j), j = 1, \ldots, m_{\text{train}} - 1 \tag{4}$$

In this case, $\mathcal{V}_{\text{samp}}$ compute $v(N)$ as well as $\{(S_j, v(S_j))\}_{j=1}^{m_{\text{train}}-1}$ where $S_j \subset N$ are sampled uniformly at random, and $\mathcal{V}_{\text{est}}$ solves the linear program in (4).

Boosting Sampling-based Heuristics via Utility Function Learning. Algorithm 1 summarizes our algorithm for accelerating data valuation with data utility learning. Our algorithm leverages the set of utility samples $\mathcal{S}_{\text{train}} = \{(S_i, v(S_i))\}_{i=1}^{m_{\text{train}}}$ that need to be sampled in the existing heuristics, and uses it as the training data to learn data utility function v. In our experiment, we use a simple neural network to train on $\mathcal{S}_{\text{train}}$, where each S_i is encoded as a n-dimensional binary vector whose entry indicates the existence of a corresponding data point. Once we obtain the learned data utility model $\hat{v}$, we can use it to predict the utilities for much more subsets and obtain additional samples $\mathcal{S}_{\text{pred}} = \{(S_i, \hat{v}(S_i))\}_{i=1}^{m_{\text{pred}}}$, where we call m_{pred} the *prediction budget*. We sample the additional subsets with the *same* distribution followed by the heuristic sampler $\mathcal{V}_{\text{samp}}$. At last, we feed the combination of the original and predicted utility samples $\mathcal{S}_{\text{train}} \cup \mathcal{S}_{\text{pred}}$ to $\mathcal{V}_{\text{est}}$. Generally, querying $\hat{v}$ is usually far more efficient than retraining the ML model. Therefore, the additional predicted utility samples $\mathcal{S}_{\text{pred}}$ can almost be acquired for free! In practice, we could set $m_{\text{pred}} \gg m_{\text{train}}$.

4 THEORETICAL ANALYSIS

Since the trained parametric function $\hat{v}$ may not fully recover v, we investigate the reliability of SV and LC estimated from a hybrid of m_{train} clean samples from v and m_{pred} noisy samples from $\hat{v}$. We denote the ratio of noisy samples as $\gamma = m_{\text{pred}}/m$. Due to space constraint, we defer the worst-case, information-theoretic result to Appendix B where the noise $\hat{v} - v$ can be adversarially distributed. Below, we show the average-case guarantee in SV estimation using Permutation Sampling (Castro et al., 2009) with hybrid utility samples. Different from the worst-case analysis in Appendix B, we consider the scenario when error $\hat{v} - v$ is drawn from a *smooth* distribution, a commonly used assumption in sensitivity analysis (Yan et al., 2020; Gupta & Roughgarden, 2017) and is arguably reasonable when $\hat{v}$ is a trained on uniformly distributed samples. We say $\hat{\phi}$ is an (ε, δ)-approximation to the true SV in p-norm if $\Pr_{\hat{\phi}}\left[\left\|\phi - \hat{\phi}\right\|_p \leq \varepsilon\right] \geq 1 - \delta$. By Hoeffding, when $p = 1$ and the samples are all clean ($\gamma = 0$), we can achieve (ε, δ)-approximation with $m = O\left(\frac{n^2}{\varepsilon^2} \log(\frac{n}{\delta})\right)$ samples.

Theorem 1. *If there exists two functions κ_0, κ_1 s.t. the random noise $\hat{v} - v \sim \mathcal{D}_n$ has conditional distribution s.t. $\kappa_0(r) \leq \Pr_{\mathcal{D}_n}(x \mid \|x\|_1 = r) \leq \kappa_1(r)$ for all $r \geq 0$ and x in $\mathcal{D}_n$'s support, then with sample $m = O\left(\frac{n^2}{\varepsilon^2} \log(\frac{n}{\delta})\right)$ and noise ratio γ, permutation sampling achieves $(\varepsilon + \gamma cn, \delta)$-approximate Shapley value in ℓ_1 norm, where $c = \frac{1}{2^{n-1}} \mathbb{E}_r\left[\frac{\kappa_1(r)}{\kappa_0(r)} r\right]$.*

The above result says that when the conditional error distribution $\hat{v} - v$ is not very "peaky" anywhere over all noise $\hat{v} - v$ of the same magnitude, then the sample complexity of permutation sampling with hybrid utility samples is the same as regular permutation sampling, except that it has an extra irreducible error term γcn. As long as the error distribution is fairly smooth, i.e., $\kappa_1(r)/\kappa_0(r)$ is close to 1 for all r, this error term will be neglectable due to the 2^n in the denominator.

We also derive the guarantee for Least core estimation with Monte Carlo approach in (4). We follow Yan & Procaccia (2020) and define the (ε, δ)-probably approximate least core to be the vector $\psi \in \mathbb{R}^n$ s.t. $\Pr_{S \sim \mathcal{D}} \left[\sum_{i \in S} \psi_i + e^\star + \varepsilon \geq v(S) \right] \geq 1 - \delta$ where $e^\star$ is the minimized objective solved from the exact formula (2), so it is unknown in practice. Nevertheless, we can still derive meaningful guarantee for the estimated ψ in terms of this unknown $e^\star$.

Theorem 2. *Given distribution $\mathcal{D}$ over 2^N, and $\delta, \Delta, \varepsilon > 0$, if for every $S \subseteq N$, the error in $\hat{v}(S)$ has property $(1 - \varepsilon\sigma)v(S) \leq \hat{v}(S) \leq (1 + \varepsilon\sigma)v(S)$ with probability at least $1 - \mu$ over the learning process of $\hat{v}$, then solving the linear program in (4) over $O\left(\frac{\tau^2 (\log n + \log(1/\Delta))}{\varepsilon^2 \delta^2} \right)$ coalitions sampled from $\mathcal{D}$ with noisy sample ratio γ, where $\tau = \frac{\max_S v(S) + \varepsilon\sigma}{\min_{S \neq \emptyset} v(S)}$, gives $(\varepsilon(1 + \gamma\sigma), \delta + \gamma\sigma)$-probably approximately least core with probability at least $1 - \Delta - m_{pred}\mu$.*

When $\sigma = 0$, Theorem 2 recovers Theorem 2 in Yan & Procaccia (2020) for the case of $\hat{v} = v$. This result suggests that with number of samples polynomial in $O(\log n)$, one can still obtain a good approximation of least core with some additional irreducible error due to the error in $\hat{v}$.

5 EVALUATION

Settings. We first assess the performance of Shapley and Least core estimators on small enough datasets (# data ≤ 15); in this case, we are able to directly calculate the true data value with the exact formula in (1) and (2), then evaluate and compare the estimation error of different heuristics. Due to space constraint, we defer the results for larger datasets and data groups to the Appendix.
Baselines: For Shapley value estimation, we consider the two existing unbiased estimators as our baselines: **(1) Permutation Sampling (Perm)** (Castro et al., 2009), **(2) Group Testing (GT)** (Jia et al., 2019b), which is an improved Monte Carlo algorithm based on group testing theory. We also compare with **CGA** approach (Yan et al., 2020) mentioned in related work, which approximate v with a linear regression model with interaction terms. For LC estimation, we use **Monte Carlo (MC)** approach (Yan & Procaccia, 2020) as the baseline. For all experiments, $\hat{v}$ is a 3-layer MLP model, and the input data subsets are encoded as binary vector.

Results. We test the performance of different SV/LC estimation heuristics on tiny datasets (< 15 data points) so that it is computationally feasible to compute the *exact* SV and LC, and we can directly calculate the estimation error. We randomly sample 15 data points from the famous Iris dataset (Pedregosa et al., 2011). A support-vector machine (SVM) classifier trained on the 15 data points

Figure 1: ℓ_1 approximation error for Shapley and Least core values with the change of number of sampled utility scores. 'DUL' stands for data utility learning.

achieves around 94% test accuracy. After training the utility model, we obtain the estimated data utilities for all the subsets not sampled before ($m_{pred} = 2^{15} - m_{train}$). We then estimate SV and LC using the exact calculation formula, with the hybrid of clean and noisy data utilities. We show ℓ_1 estimation errors in Figure 1, and defer the results for ℓ_2 and ℓ_∞ errors to Appendix[1]. As we can see, with a relatively small m_{train} (e.g., 500), data utility learning can significantly reduce the estimation errors for both SV and LC. Utility prediction per se may introduce additional computational costs; yet, these costs are often negligible compared to model retraining. Moreover, CGA-based SV performs poorly, because there are high-order data interactions in data utility functions which CGA cannot successfully capture, as discussed in Appendix A.
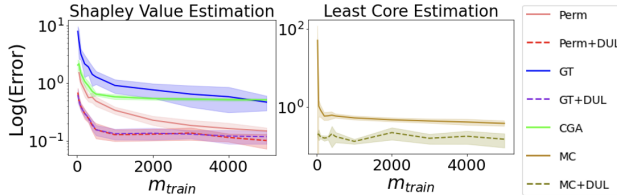
6 CONCLUSION

This work proposes a generic framework that can significantly improve the accuracy of all the existing unbiased SV/LC estimation methods through learning data utility functions. Particularly, our approach can be extended to other cooperative games where querying characteristic functions are expensive, e.g., bidders' valuation functions in combinatorial auctions (Lehmann et al., 2006). With this paper, we march one step closer towards making data valuation practical and hope to inspire more research in this direction amongst the ML community.

---

[1]It is easy to see that LC may not be unique. In the experiment, when we talk about the LC, we always refer to the vector ψ that has the smallest ℓ_2 norm, following the tie-breaking rule in Yan & Procaccia (2020). It is difficult to directly use (ε, δ)-LC definition in Section 4 to compare LC estimation as it has two error dimensions.

REFERENCES

Maria Florina Balcan, Ariel D Procaccia, and Yair Zick. Learning cooperative games. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Eric Balkanski, Umar Syed, and Sergei Vassilvitskii. Statistical cost sharing. *Advances in Neural Information Processing Systems*, 30, 2017.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

Xiaotie Deng and Christos H Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Ulrich Faigle, Walter Kern, and Daniël Paulusma. Note on the computational complexity of least core concepts for min-cost spanning tree games. *Mathematical methods of operations research*, 52 (1):23–38, 2000.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251. PMLR, 2019.

Rishi Gupta and Tim Roughgarden. A pac approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019a.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1167–1176. PMLR, 2019b.

Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? *arXiv preprint arXiv:1911.07128*, 2019c.

Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Nicolas Pinto, Zak Stone, Todd Zickler, and David Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *CVPR 2011 WORKSHOPS*, pp. 35–42. IEEE, 2011.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning. In *Federated Learning*, pp. 153–167. Springer, 2020.

Tom Yan and Ariel D Procaccia. If you like shapley then you'll love the core, 2020.

Tom Yan, Christian Kroer, and Alexander Peysakhovich. Evaluating and rewarding teamwork using cooperative game abstractions. *arXiv preprint arXiv:2006.09538*, 2020.

5

A COMPARISON WITH COOPERATIVE GAME ABSTRACTION (CGA).

Yan et al. (2020) propose a similar idea which approximates Shapley value through learning the characteristic functions in a cooperative game, where the parametric model they use to learn is called cooperative game abstraction (CGA). CGA is essentially a linear regression model where the variables are small subsets of players. The order of CGA refers to the largest size of player groups included as variables in the linear regression. One advantage of CGA is that it can recover Shapley value directly from its parameters if the function of CGA perfectly matches the characteristic function. Hence, Yan et al. (2020) propose to learn the characteristic function using CGA with certain amount of samples, and then compute Shapley value through CGA's trained parameters. For data valuation problem, the characteristic function is the data utility function v. In this sense, CGA can be viewed as a special case of data utility learning where the data utility model is linear regression. However, we argue that CGA may not be a suitable model for data utility learning for two main reasons: (i) CGA is only suitable for certain types of games where the interactions only exist among small groups of players, e.g., the team performance in basketball games. On the contrary, interactions between large groups of data points might still be strong. Thus, CGA is not a suitable model for data utility learning in nature, and we confirmed this point in Section 5. (ii) CGA has a poor scalability even for one with low order, e.g. for a third-order CGA with 2000 players, the number of total parameters is $\binom{2000}{2} + 2000 = 2,001,000$, while the number of total parameters of a $2000 \times 512 \times 1$ fully-connected neural network is only around half of it $(1,024,512)$. We also note that the parameters of CGA only has a closed form for computing Shapley value, but not the Least core.

B INFORMATION-THEORETIC RESULT FOR SV ESTIMATION

For SV, we related the error in $\hat{v}$ to the error in the final estimation. We start with an information-theoretic result when the Shapley value is computed by the exact formula in (1) with m_{train} clean samples and $m_{\text{pred}} = 2^n - m_{\text{train}}$ noisy samples. As v and $\hat{v}$ can be viewed as a 2^n-dimensional vector, the input for (1) is the hybrid data utility vector $\tilde{v} \in \mathbb{R}^{2^n}$ where the entry correspond to subset S is $v(S)$ if it is within the clean samples, and $\hat{v}(S)$ otherwise. The following result shows that, by smartly allocate the clean sample budget, one can significantly reduce the error in the final estimation.

Theorem 3. *For any integer $-1 \leq p \leq \log(n)$, if for all $|S| \leq p$ and $|S| \geq n - p$ we have clean samples $v(S)$ then the estimation error of SV is bounded by $\|\phi(v) - \phi(\tilde{v})\|_2 \leq \tilde{O}\left(\frac{1}{n^{1+p/2}}\right) \|v - \tilde{v}\|_2$.*

The above results imply that, whenever $m_{\text{train}} \geq 2\sum_{i=0}^{p}\binom{n}{i}$ for some $p \geq 0$, one can largely improve the Shapley value estimation guarantee by computing $v(S)$ for S very small or very large size. The case of $p = -1$ (i.e., $\tilde{v} = \hat{v}$) recovers Theorem 3 in Yan et al. (2020). In particular, even when $p = 0$, i.e., only two clean samples $v(\emptyset)$ and $v(N)$ are computed, the ℓ_2 error guarantee can be improved by $O\left(\frac{1}{\sqrt{n}}\right)$. In practice, to learn a good $\hat{v}$ s.t. $\|v - \hat{v}\|_2$ is small, the clean samples need to be distributed uniformly over 2^N. However, Theorem 3 still suggests that when there are spare training budget, one can distribute it to very small or very large subsets.

B.1 PROOF

The proof extends the proof of Theorem 3 in Yan et al. (2020). Like them, we use the fact that Shapley value is a linear mapping $\phi : \mathbb{R}^{2^n} \to \mathbb{R}^n$ taking v to $\phi(v)$. We can describe this map with a matrix $S_n \in \mathbb{R}^{n \times 2^n}$. For any hybrid estimation $\tilde{v}$, we have

$$\|\phi(v) - \phi(\tilde{v})\| = \|S_n v - S_n \tilde{v}\| = \|S_n(v - \tilde{v})\| \tag{5}$$

Suppose there are m_{train} clean estimations of $v(S)$ and the rest of $2^n - m_{\text{train}}$ are noisy estimation. Then $v - \tilde{v}$ has exactly m_{train} entries are 0, which means that the corresponding m_{train} columns of S_n are not used; we use $\tilde{S}_n \in \mathbb{R}^{n \times (2^n - m_{\text{train}})}$ to denote the submatrix and $\tilde{v}' \in \mathbb{R}^{n \times (2^n - m_{\text{train}})}$ to denote the compressed vector. Thus we have

$$\|S_n(v - \tilde{v})\| = \left\|\tilde{S}_n \tilde{v}'\right\| \leq \left\|\tilde{S}_n\right\|_{op} \|\tilde{v}'\| \tag{6}$$

So the question reduce to selecting $2^n - m_{\text{train}}$ columns of S_n and minimize

$$\left\|\tilde{S}_n\right\|_{op} = \sqrt{\sigma_{\max}(\tilde{S}_n^T \tilde{S}_n)} = \sqrt{\sigma_{\max}(\tilde{S}_n \tilde{S}_n^T)} \tag{7}$$

We analyze the best column subset selection approach to minimize $\sigma_{\max}\left(\tilde{S}_n \tilde{S}_n^T\right)$. Denote the selector vector $\alpha \in \{0,1\}^{2^n}$, where $\alpha_S = 1$ if the column of S is kept (noisy), and 0 otherwise (clean). So $\|\alpha\|_1 = 2^n - m_{\text{train}}$. We denote $\alpha^{(k)} = |\{\alpha_S | |S| = k, \alpha_S = 1\}|$, so $\sum_{k=0}^n \alpha^{(k)} = 2^n - m_{\text{train}}$.

For the ith row of S_n, $(S_n)_i$, we know that the entry in this row corresponding to subset S is $\frac{1}{n}\binom{n-1}{|S|-1}^{-1}$ if $i \in S$, and $-\frac{1}{n}\binom{n-1}{|S|}^{-1}$ if $i \notin S$, thus we can compute every diagonal and non-diagonal entry. The ith diagonal entry of $\tilde{S}_n \tilde{S}_n^T$ is

$$d_{ii} = (\tilde{S}_n)_i^T (\tilde{S}_n)_i \tag{8}$$

$$= \sum_{S \in 2^n, i \in S} \alpha_S \left(\frac{1}{n}\binom{n-1}{|S|-1}^{-1}\right)^2 + \sum_{S \in 2^n, i \notin S} \alpha_S \left(-\frac{1}{n}\binom{n-1}{|S|}^{-1}\right)^2 \tag{9}$$

and the (i,j) non-diagonal entry is

$$d_{ij} = (\tilde{S}_n)_i^T (\tilde{S}_n)_j \tag{10}$$

$$= \sum_{S \in 2^n, i,j \in S} \alpha_S \left(\frac{1}{n}\binom{n-1}{|S|-1}^{-1}\right)^2 + \sum_{S \in 2^n, i \notin S, j \in S} \alpha_S \left(\frac{1}{n}\binom{n-1}{|S|-1}^{-1}\right)\left(-\frac{1}{n}\binom{n-1}{|S|}^{-1}\right) \tag{11}$$

$$+ \sum_{S \in 2^n, i \in S, j \notin S} \alpha_S \left(\frac{1}{n}\binom{n-1}{|S|-1}^{-1}\right)\left(-\frac{1}{n}\binom{n-1}{|S|}^{-1}\right) + \sum_{S \in 2^n, i,j \notin S} \alpha_S \left(-\frac{1}{n}\binom{n-1}{|S|}^{-1}\right)^2 \tag{12}$$

Now if $m_{\text{train}} \geq 2\sum_{j=0}^p \binom{n}{j}$. We make $\alpha_S^{(j)} = 0$ for all $0 \leq j \leq p$ and $n - p \leq j \leq n$ (i.e., explicitly computing $v(S)$ for very small and very large S).

Under this strategy, the property of $\tilde{S}_n \tilde{S}_n^T$ is very good in the sense that every diagonal entry has the same value, and every non-diagonal entry has the same value:

$$d_1 = (\tilde{S}_n)_i^T (\tilde{S}_n)_i \tag{13}$$

$$= \sum_{S \in 2^n, i \in S, p+1 \leq |S| \leq n-p-1} \alpha_S \left(\frac{1}{n}\binom{n-1}{|S|-1}^{-1}\right)^2 + \sum_{S \in 2^n, i \notin S, p+1 \leq |S| \leq n-p-1} \alpha_S \left(-\frac{1}{n}\binom{n-1}{|S|}^{-1}\right)^2 \tag{14}$$

$$= \frac{1}{n^2} \sum_{k=p+1}^{n-p-1} \left(\binom{n-1}{k-1}^{-1} + \binom{n-1}{k}^{-1}\right) \tag{15}$$

If $p \geq 1$:

$$d_2 = \frac{1}{n^2} \sum_{k=p+1}^{n-p-1} \left[\binom{n-2}{k-2}\binom{n-1}{k-1}^{-2} - 2\binom{n-2}{k-1}\binom{n-1}{k-1}^{-1}\binom{n-1}{k}^{-1} + \binom{n-2}{k}\binom{n-1}{k}^{-2}\right] \tag{16}$$

If $p = 0$:

$$d_2 = \frac{1}{n^2} \sum_{k=p+1}^{n-p-1} \left[\binom{n-2}{k-2}\binom{n-1}{k-1}^{-2} - 2\binom{n-2}{k-1}\binom{n-1}{k-1}^{-1}\binom{n-1}{k}^{-1} + \binom{n-2}{k}\binom{n-1}{k}^{-2} \right] \tag{17}$$

$$- \frac{2}{n(n-1)^2} \tag{18}$$

$$= d_2' - \frac{2}{n(n-1)^2} \tag{19}$$

Therefore we can write

$$\tilde{S}_n \tilde{S}_n^T = (d_1 - d_2)I_n + d_2 \mathbf{1}_n \tag{20}$$

where $\mathbf{1}_n$ is the matrix with 1 in all entries, which has rank 1 and thus the only non-zero eigenvalue is n. Therefore, the eigenvalues of $\tilde{S}_n \tilde{S}_n^T$ are $(d_1 - d_2 + nd_2)$ and $(d_1 - d_2)$.

Case of $p \geq 1$:

$$d_1 - d_2 + nd_2 \tag{21}$$

$$= d_1 + (n-1)d_2 \tag{22}$$

$$= \frac{1}{n^2} \sum_{k=p+1}^{n-p-1} \left(\binom{n-1}{k-1}^{-1} + \binom{n-1}{k}^{-1} \right) \tag{23}$$

$$+ \frac{n-1}{n^2} \sum_{k=p+1}^{n-p-1} \left[\binom{n-2}{k-2}\binom{n-1}{k-1}^{-2} - 2\binom{n-2}{k-1}\binom{n-1}{k-1}^{-1}\binom{n-1}{k}^{-1} + \binom{n-2}{k}\binom{n-1}{k}^{-2} \right] \tag{24}$$

$$= \frac{1}{n^2} \sum_{k=p+1}^{n-p-1} \left[\binom{n-1}{k-1}^{-1} + \binom{n-1}{k}^{-1} + (k-1)\binom{n-1}{k-1}^{-1} - 2k\binom{n-1}{k-1}^{-1} + (n-k-1)\binom{n-1}{k}^{-1} \right] \tag{25}$$

$$= \frac{1}{n^2} \sum_{k=p+1}^{n-p-1} \left[-k\binom{n-1}{k-1}^{-1} + (n-k)\binom{n-1}{k}^{-1} \right] \tag{26}$$

$$= 0 \tag{27}$$

and

$$d_1 - d_2 \tag{28}$$

$$= -nd_2 \tag{29}$$

$$= -\frac{1}{n} \sum_{k=p+1}^{n-p-1} \left[\binom{n-2}{k-2} \binom{n-1}{k-1}^{-2} - 2 \binom{n-2}{k-1} \binom{n-1}{k-1}^{-1} \binom{n-1}{k}^{-1} + \binom{n-2}{k} \binom{n-1}{k}^{-2} \right] \tag{30}$$

$$= -\frac{1}{n(n-1)} \sum_{k=p+1}^{n-p-1} \left[(k-1)\binom{n-1}{k-1}^{-1} - 2k\binom{n-1}{k-1}^{-1} + (n-1-k)\binom{n-1}{k}^{-1} \right] \tag{31}$$

$$= -\frac{1}{n(n-1)} \sum_{k=p+1}^{n-p-1} \left[(-k-1)\binom{n-1}{k-1}^{-1} + (n-1-k)\binom{n-1}{k}^{-1} \right] \tag{32}$$

$$= -\frac{1}{n(n-1)} \sum_{k=p+1}^{n-p-1} \left[\frac{-(k+1)(k-1)!(n-k)! + (n-1-k)k!(n-1-k)!}{(n-1)!} \right] \tag{33}$$

$$= -\frac{1}{n(n-1)} \sum_{k=p+1}^{n-p-1} \left[\frac{-(k-1)!(n-k)! - k!(n-1-k)!}{(n-1)!} \right] \tag{34}$$

$$= \frac{1}{n(n-1)} \sum_{k=p+1}^{n-p-1} \left[\binom{n-1}{k-1}^{-1} + \binom{n-1}{k}^{-1} \right] \tag{35}$$

$$\leq \frac{2}{n(n-1)} \sum_{k=p}^{n-p-1} \binom{n-1}{k}^{-1} \tag{36}$$

$$\leq \frac{4}{n(n-1)} \sum_{k=p}^{n/2} \binom{n-1}{k}^{-1} \tag{37}$$

$$\leq \frac{4}{n(n-1)} \binom{n-1}{p}^{-1} \sum_{k=p}^{n/2} \left(\frac{1}{2} \right)^{k-p} \tag{38}$$

$$\leq \frac{8}{n(n-1)} \binom{n-1}{p}^{-1} \tag{39}$$

$$\leq \frac{8}{n(n-1)} \left(\frac{p}{n} \right)^p \tag{40}$$

where (38) is due to

$$\binom{n-1}{k}^{-1} \Big/ \binom{n-1}{k-1}^{-1} = \frac{k!(n-1-k)!}{(k-1)!(n-k)!} = \frac{k}{n-k} \leq 1/2 \tag{41}$$

for $k \leq n/2$. Therefore, when $1 \leq p \leq \log(n)$, we have

$$\left\| \tilde{S}_n \right\|_{op} = \sqrt{\sigma_{\max}(\tilde{S}_n \tilde{S}_n^T)} \leq \tilde{O}\left(\frac{1}{n^{1+p/2}} \right) \tag{42}$$

Case of $p = 0$: For the first eigenvalue $d_1 + (n-1)d_2$, follow a similar argument as for the case of $p \geq 1$, we derive

$$d_1 + (n-1)d_2 = 0 \tag{43}$$

For the second eigenvalue $d_1 - d_2$, we have

$$d_1 - d_2 \tag{44}$$

$$= -nd_2 \tag{45}$$

$$= -n\left(d_2' - \frac{2}{n(n-1)^2}\right) \tag{46}$$

$$= \frac{1}{n(n-1)} \sum_{k=2}^{n-2} \left[\binom{n-1}{k-1}^{-1} + \binom{n-1}{k}^{-1}\right] + \frac{2}{(n-1)^2} \tag{47}$$

$$\leq \frac{2}{(n-1)^2} + O\left(\frac{1}{n^3}\right) \tag{48}$$

Therefore, when $p = 0$, i.e., only the full set and empty set's utility $v(\emptyset)$ and $v(N)$ is explicitly calculated, the worst-case L2 error guarantee can be improved to $\left\|\tilde{S}_n\right\|_{op} = \sqrt{\sigma_{\max}(\tilde{S}_n \tilde{S}_n^T)} \leq O\left(\frac{1}{n}\right)$. We also note that when $p = -1$, i.e., there are no clean estimation and $\tilde{v} = \hat{v}$, we have $\left\|\tilde{S}_n\right\|_{op} = \|S_n\|_{op} \leq O\left(\frac{1}{\sqrt{n}}\right)$ as derived in Theorem 3 of Yan et al. (2020). More generally, we have $\left\|\tilde{S}_n\right\|_{op} \leq \tilde{O}\left(\frac{1}{n^{1+p/2}}\right)$ for $-1 \leq p \leq \log(n)$ in general.

C PROOF OF THEOREM 1

Recall that we say $\hat{\phi}$ is an (ε, δ)-approximation to the true SV in terms of p-norm if

$$\Pr_{\hat{\phi}}\left[\left\|\phi - \hat{\phi}\right\|_p \leq \varepsilon\right] \geq 1 - \delta \tag{49}$$

By Hoeffding, when $p = 1$, we can achieve (ε, δ)-approximation with $m = O\left(\frac{n^2}{\varepsilon^2} \log(\frac{n}{\delta})\right)$.

Similar to Appendix B, since v and $\hat{v}$ can be viewed as a 2^n-dimensional vector, we can construct a hybrid data utility vector $\tilde{v} \in \mathbb{R}^{2^n}$ where the entry correspond to subset S is $v(S)$ if it is within the clean samples, and $\hat{v}(S)$ otherwise. For simplicity, we assume that both m and m_{pred} can be divided by n. Now consider the situation when the characteristic function $\tilde{v}$ is noisy for m_{pred}/n permutations among the m/n total permutations. By triangle inequality, we have

$$\left|\hat{\phi}_i(\tilde{v}) - \phi_i(v)\right| \leq \left|\hat{\phi}_i(\tilde{v}) - \hat{\phi}_i(v)\right| + \left|\hat{\phi}_i(v) - \phi_i(v)\right| \tag{50}$$

The extra error term

$$\left|\hat{\phi}_i(\tilde{v}) - \hat{\phi}_i(v)\right| = \frac{1}{m/n} \sum_{j=1}^{m_{\text{pred}}/n} \left|\left[(\hat{v}(P_i^{\pi_j} \cup \{i\}) - \hat{v}(P_i^{\pi_j})) - (v(P_i^{\pi_j} \cup \{i\}) - v(P_i^{\pi_j}))\right]\right| \tag{51}$$

$$\leq \frac{1}{m/n} \sum_{j=1}^{m_{\text{pred}}/n} \left[\left|\hat{v}(P_i^{\pi_j} \cup \{i\}) - v(P_i^{\pi_j} \cup \{i\})\right| + \left|\hat{v}(P_i^{\pi_j}) - v(P_i^{\pi_j})\right|\right] \tag{52}$$

Now fix the sampled permutations $\pi_1, \ldots, \pi_j$. Suppose the error distribution follows $|\hat{v} - v|/r \sim \mathcal{D}_r$ with the support of 2^n simplex, and smooth s.t. $\kappa_0 \leq \Pr_{x \sim \mathcal{D}_r}[x] \leq \kappa_1$. With this error distribution,

the expected value of the extra error term can be bounded as

$$\mathbb{E}_{\hat{v}-v\sim\mathcal{D}_r}\left[\left|\hat{\phi}_i(\tilde{v})-\hat{\phi}_i(v)\right|\right] \leq \frac{1}{m/n}\sum_{j=1}^{m_{\text{pred}}/n}\left[\mathbb{E}_{\hat{v}-v\sim\mathcal{D}_r}\left[\left|\hat{v}(P_i^{\pi_j}\cup\{i\})-v(P_i^{\pi_j}\cup\{i\})\right|\right]\right. \tag{53}$$

$$\left.+\ \mathbb{E}_{\hat{v}-v\sim\mathcal{D}_r}\left[\left|\hat{v}(P_i^{\pi_j})-v(P_i^{\pi_j})\right|\right]\right] \tag{54}$$

$$\leq \frac{m_{\text{pred}}}{m}\frac{2\kappa_1}{\kappa_0}\frac{r}{2^n} \tag{55}$$

$$= \gamma\frac{2\kappa_1}{\kappa_0}\frac{r}{2^n} \tag{56}$$

where the inequality in (55) is due to the following[2]: let subset $C^* = \operatorname{argmax}_C \mathbb{E}_{\boldsymbol{v}-\hat{\boldsymbol{v}}\sim\mathcal{D}_{S_r}}\left[|v(C)-\hat{v}(C)|\right]$ and subset $C' = \operatorname{argmin}_C \mathbb{E}_{\boldsymbol{v}-\hat{\boldsymbol{v}}\sim\mathcal{D}_{S_r}}\left[|v(C)-\hat{v}(C)|\right]$.

$$\mathbb{E}_{\boldsymbol{v}-\hat{\boldsymbol{v}}\sim\mathcal{D}_{S_r}}\left[\left|v\left(C^*\right)-\hat{v}\left(C^*\right)\right|\right] = \int \left|v\left(C^*\right)-\hat{v}\left(C^*\right)\right|\operatorname{Pr}_{\mathcal{D}_{S_r}}(\boldsymbol{v}-\hat{\boldsymbol{v}})d(\boldsymbol{v}-\hat{\boldsymbol{v}}) \tag{57}$$

$$\leq \int \left|v\left(C^*\right)-\hat{v}\left(C^*\right)\right|\kappa_1 d(\boldsymbol{v}-\hat{\boldsymbol{v}}) \tag{58}$$

$$= \int \left|v\left(C'\right)-\hat{v}\left(C'\right)\right|\kappa_1 d(\boldsymbol{v}-\hat{\boldsymbol{v}}) \tag{59}$$

$$\leq \int \left|v\left(C'\right)-\hat{v}\left(C'\right)\right|\frac{\kappa_1}{\kappa_0}\operatorname{Pr}_{\mathcal{D}_{S_r}}(\boldsymbol{v}-\hat{\boldsymbol{v}})d(\boldsymbol{v}-\hat{\boldsymbol{v}}) \tag{60}$$

$$= \frac{\kappa_1}{\kappa_0}\mathbb{E}_{\boldsymbol{v}-\hat{\boldsymbol{v}}\sim\mathcal{D}_{S_r}}\left[\left|v\left(C'\right)-\hat{v}\left(C'\right)\right|\right] \tag{61}$$

$$\leq \frac{\kappa_1}{\kappa_0}\left(\frac{r}{2^n}\right) \tag{62}$$

Here (59) holds by symmetry as the expectation of any two vector coordinates under a uniform distribution over the simplex of vectors is the same. (62) holds because every vector in the support of $\mathcal{D}_{S_r}$ has ℓ_1 norm of r, $\sum_C \mathbb{E}_{\boldsymbol{v}-\hat{\boldsymbol{v}}\sim\mathcal{D}_{S_r}}\left[|v(C)-\hat{v}(C)|\right] = \mathbb{E}_{\boldsymbol{v}-\hat{\boldsymbol{v}}\sim\mathcal{D}_{S_r}}\left[\sum_C |v(C)-\hat{v}(C)|\right] = r$ and so by our choice of C', $\mathbb{E}_{\boldsymbol{v}-\hat{\boldsymbol{v}}\sim\mathcal{D}_{S_r}}\left[|v\left(C'\right)-\hat{v}\left(C'\right)|\right] \leq \frac{r}{2^n}$.

Integrating over r, we have

$$\mathbb{E}_{\hat{v}-v\sim\mathcal{D}}\left[\left|\hat{\phi}_i(\hat{v})-\hat{\phi}_i(v)\right|\right] \leq \frac{\gamma}{2^{n-1}}\mathbb{E}_r\left[\frac{\kappa_1(r)}{\kappa_0(r)}r\right] \tag{63}$$

Therefore

$$\mathbb{E}_{\hat{v}-v\sim\mathcal{D}}\left[\left|\hat{\phi}_i(\hat{v})-\phi_i(v)\right|\right] \leq \frac{\gamma}{2^{n-1}}\mathbb{E}_r\left[\frac{\kappa_1(r)}{\kappa_0(r)}r\right]+\left|\hat{\phi}_i(v)-\phi_i(v)\right| \tag{64}$$

Denote $c = \frac{1}{2^{n-1}}\mathbb{E}_r\left[\frac{\kappa_1(r)}{\kappa_0(r)}r\right]$.

Hence

$$\operatorname*{Pr}_{\pi_1,\ldots,\pi_m\sim Unif(\Pi)}\left[\mathbb{E}_{\hat{v}-v\sim\mathcal{D}}\left[\left\|\hat{\phi}(\hat{v})-\phi(v)\right\|_1\right]\leq\varepsilon+\gamma cn\right] \tag{65}$$

$$= \operatorname*{Pr}_{\pi_1,\ldots,\pi_m\sim Unif(\Pi)}\left[\gamma cn+\sum_{i=1}^n\left|\hat{\phi}_i(v)-\phi_i(v)\right|\leq\varepsilon+\gamma cn\right] \tag{66}$$

$$\geq \operatorname*{Pr}_{\pi_1,\ldots,\pi_m\sim Unif(\Pi)}\left[\sqcup_{i=1}^n\left|\hat{\phi}_i(v)-\phi_i(v)\right|\leq\varepsilon/n\right] \tag{67}$$

$$\geq 1-\delta \tag{68}$$

whenever

$$m = O\left(\frac{n^2}{\varepsilon^2}\log(\frac{n}{\delta})\right)$$

---

[2]This argument is inspired by Yan et al. (2020).

Even if $\gamma = \frac{m_{\text{pred}}}{m} \to 1$, as long as $c = o(\frac{1}{n})$, the irreducible error term $\gamma c n \to 0$ as $n \to \infty$. This is very possible since c has a $\frac{1}{2^n}$ as a factor.

D   PROOF OF THEOREM 2

This proof directly extends the proof of Theorem 2 of Yan & Procaccia (2020), which is based on the observation in Balcan et al. (2015) and Balkanski et al. (2017) that estimating least core from finite samples is equivalent to the problem of learning an unknown linear function (x, e) s.t. $\sum_{i \in S} x_i \geq v(S)$ for all $S \subseteq N$. Like Yan et al. (2020) and Balkanski et al. (2017), we use the following learnability result for linear classifiers Shalev-Shwartz & Ben-David (2014):

Lemma 4 (Rademacher Complexity for Halfspaces). *Let $\mathcal{H} = \{\mathbf{w} : \|\mathbf{w}\|_1 \leq B\}$ be the hypothesis class, and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be the examples domain. Suppose $\mathcal{D}_Z$ is a distribution over $\mathcal{Z}$ s.t $\|\mathbf{x}\|_\infty \leq R$. Let the loss function $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ be of the form $\ell(\mathbf{w}, (\mathbf{x}, y)) = \phi(\langle \mathbf{w}, \mathbf{x} \rangle, y)$ and $\phi : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ is such that for all $y \in \mathcal{Y}$, the scalar function $a \to \phi(a, y)$ is ρ-Lipschitz and such that $\max_{a \in [-BR, BR]} |\phi(a, y)| \leq c$. Then for any $\Delta \in (0, 1)$, with probability of at least $1 - \Delta$ over the choice of an i.i.d. sample of size $m, (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$:*

$$\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_z}[\ell(\mathbf{w}, (\mathbf{x}, y))] \leq \frac{1}{m} \sum_{i=1}^{m} \ell(\mathbf{w}, (\mathbf{x}^i, y^i)) + 2\rho BR\sqrt{\frac{2\log(2d)}{m}} + c\sqrt{\frac{2\log(2/\Delta)}{m}}$$

for all $\mathbf{w} \in \mathcal{H}$.

We assume that for every $S \subseteq N$, the error distribution has concentration property

$$(1 - \varepsilon\sigma)v(S) \leq \hat{v}(S) \leq (1 + \varepsilon\sigma)v(S) \tag{69}$$

with probability at least $1 - \mu$. If we have m_{pred} noisy samples of $\hat{v}(S)$, then with probability at least $1 - m_{\text{pred}}\mu$ that all of them satisfy (69).

We also require the following observation:

Lemma 5. *For any $\varepsilon, \sigma > 0$, $\delta < 1$, and (x, e) computed from the linear program in (4) with each $v(S)$ replaced by $\tilde{v}(S)$,*

$$\mathbb{E}_{S \sim \mathcal{D}} \left[\left[1 - \frac{\sum_{i \in S} z_i + e}{v(S)} \right]_+ \right] \leq \varepsilon(\sigma + \delta) \Rightarrow \Pr_{S \sim \mathcal{D}} \left[\sum_{i \in S} z_i + e^\star + \varepsilon(1 + \sigma) \geq v(S) \right] \geq 1 - \sigma - \delta \tag{70}$$

Proof.

$$\left[1 - \frac{\sum_{i \in S} z_i + e}{v(S)} \right]_+ \leq \varepsilon \Leftrightarrow 1 - \frac{\sum_{i \in S} z_i + e}{v(S)} \leq \varepsilon \tag{71}$$

$$\Leftrightarrow \sum_{i \in S} z_i + e + \varepsilon v(S) \geq v(S) \tag{72}$$

$$\Rightarrow \sum_{i \in S} z_i + e + \varepsilon \geq v(S) \tag{73}$$

$$\Rightarrow \sum_{i \in S} z_i + (e - \sigma\varepsilon) + \varepsilon + \sigma\varepsilon \geq v(S) \tag{74}$$

$$\Rightarrow \sum_{i \in S} z_i + e^\star + \varepsilon + \sigma\varepsilon \geq v(S) \tag{75}$$

By Markov inequality,

$$\Pr_{S \sim \mathcal{D}} \left[\sum_{i \in S} z_i + e^\star + \varepsilon(1 + \sigma) \geq v(S) \right] = \Pr_{S \sim \mathcal{D}} \left[\left[1 - \frac{\sum_{i \in S} z_i + e}{v(S)} \right]_+ \leq \varepsilon \right] \quad (76)$$

$$\geq 1 - \frac{\mathbb{E}_{S \sim \mathcal{D}} \left[\left[1 - \frac{\sum_{i \in S} z_i + e}{v(S)} \right]_+ \right]}{\varepsilon} \quad (77)$$

$$\geq 1 - \frac{\varepsilon(\sigma + \delta)}{\varepsilon} \quad (78)$$

$$\geq 1 - \sigma - \delta \quad (79)$$

$$\square$$

Turning back to the theorem's proof. First, we bound the ℓ_1 norm of the solution solved from the linear program in (4). Suppose (x, e) is the solution, then

$$\|(x, e)\|_1 = v(N) + e \leq v(N) + \max_S \hat{v}(S) \leq v(N) + \max_S v(S) + \varepsilon\sigma \leq 2\max_S v(S) + \varepsilon\sigma \quad (80)$$

where we stress that $v(N)$ is always explicitly computed in the Monte Carlo approach for Least core estimation.

We take our hypothesis class to be

$$\mathcal{H} = \{ z \in \mathbb{R}^{n+1} : \|z\|_1 \leq 2\max_S v(S) + \varepsilon\sigma \} \quad (81)$$

For any data subset S, we define the corresponding data feature as $\mathbf{x}^S = \left(\frac{\mathbb{I}[i \in S]}{v(S)}, \frac{1}{v(S)} \right)$ and the label to be $y^S = 1$. We define to $\mathcal{D}$ to be the uniform distribution over all $(\mathbf{x}^S, y^S)$ pairs. Next, suppose we obtain m samples $S_1, \ldots, S_m$ from $\mathcal{D}$, we may run the linear program in (4) on the m samples where m_{pred} of the samples are replaced to $\mathbf{x}^S = \left(\frac{\mathbb{I}[i \in S]}{\hat{v}(S)}, \frac{1}{\hat{v}(S)} \right)$, which gives us a payoff allocation $\hat{z}$ and a value $\hat{e}$. We take our classifier to be of the form $\mathbf{w} = (\hat{z}, \hat{e})$ and we define its loss ℓ to be:

$$\ell\left(\mathbf{w}, \left(\mathbf{x}^S, y^S \right) \right) = \ell\left((\hat{z}, \hat{e}), \left(\left(\frac{\mathbb{I}[i \in S]}{v(S)}, \frac{1}{v(S)} \right), y^S \right) \right)$$

$$= \left[y^S - (\hat{z}, \hat{e}) \cdot \left(\frac{\mathbb{I}[i \in S]}{v(S)}, \frac{1}{v(S)} \right) \right]_+$$

$$= \left[1 - \frac{\sum_{i \in S} \hat{z}_i + \hat{e}}{v(S)} \right]_+$$

Now, we utilize Lemma 4 with the remaining variables being $R = \frac{1}{\min_{S \neq \emptyset} v(S)}$, $B = 2\max_S v(S) + \varepsilon\sigma$, $\phi(a, y) = [y - a]_+$, $\rho = 1$ and $c = 1 + 2BR$. By definition of $\mathbf{x}^S$, $\|\mathbf{x}^S\|_\infty \leq \frac{1}{\min_{S \neq \emptyset} v(S)}$. By definition of the hypothesis class, $\|(\mathbf{z}, e)\|_1 \leq 2\max_S v(S) + \varepsilon\sigma$ for all $(\mathbf{z}, e) \in \mathcal{H}$, $\phi(a, y) = [y - a]_+$ is 1-Lipschitz. Since our example domain $\mathcal{Z}$ is such that $\mathcal{Y} = \{1\}$. We may obtain upper bound c: $c = \max_{a \in [-BR, BR]} |\phi(a, y)| = \max_{a \in [-BR, BR]} [1 - a]_+ \leq (1 - -BR) = 1 + BR$.

Now we bound the training error of (x, e) on clean samples:

$$\frac{1}{m} \sum_{j=1}^{m_{\text{pred}}} \left[1 - \frac{\sum_{i \in S_j} \hat{\mathbf{z}}_i + \hat{e}}{v(S_j)} \right]_+ \leq \frac{1}{m} \sum_{j=1}^{m_{\text{pred}}} \left[1 - \frac{\hat{v}(S_j)}{v(S_j)} \right]_+ \leq \frac{1}{m} \sum_{j=1}^{m_{\text{pred}}} [1 - (1 - \varepsilon\sigma)]_+ \leq \frac{m_{\text{pred}}}{m} \varepsilon\sigma$$

$$(82)$$

13

Therefore by Lemma 4

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[l(\mathbf{w},(\mathbf{x},y))] = \mathbb{E}_{S\sim D}\left[\left[1 - \frac{\sum_{i\in S}\hat{\mathbf{z}}_i + \hat{e}}{v(S)}\right]_+\right] \tag{83}$$

$$\leq \frac{m_{\text{pred}}}{m}\varepsilon\sigma + 2\cdot 1\cdot BR\sqrt{\frac{2\log(2(n+1))}{m}} + (1+BR)\sqrt{\frac{2\log(2/\Delta)}{m}} \tag{84}$$

$$\leq \frac{m_{\text{pred}}}{m}\varepsilon\sigma + 2BR\sqrt{\frac{2\log(2(n+1))}{m}} + (1+BR)\sqrt{\frac{2\log(2/\Delta)}{m}} \tag{85}$$

$$\leq \gamma\varepsilon\sigma + 2BR\sqrt{\frac{2\log(2(n+1))}{m}} + (1+BR)\sqrt{\frac{2\log(2/\Delta)}{m}} \tag{86}$$

Using Lemma 5, we need m samples to be such that

$$\gamma\varepsilon\sigma + 2BR\sqrt{\frac{2\log(2(n+1))}{m}} + (1+BR)\sqrt{\frac{2\log(2/\Delta)}{m}} \leq \varepsilon(\gamma\sigma + \delta) \tag{87}$$

and we get that

$$O\left(\frac{B^2 R^2\left(\log n + \log\left(\frac{1}{\Delta}\right)\right)}{\varepsilon^2\delta^2}\right)$$

samples suffice.

E ADDITIONAL EXPERIMENT DETAILS AND RESULTS

We show the implementation details (E.1) and additional results on larger datasets valuation (E.3) here. For larger datasets, since it is impractical to compute the exact data value, we compare the performance of data value estimates on *data removal* task, following existing data valuation literature (Ghorbani & Zou, 2019; Jia et al., 2019a;c; Wang et al., 2020; Yan & Procaccia, 2020). Besides, we also evaluate the performance on data *group* valuation (E.4), where the Shapley or least core values are assigned to a group of data rather than a single point.

E.1 IMPLEMENTATION DETAILS

For tiny data experiment in Section 6, we use a small MLP model with 2 hidden layers as the data utility learning model, where the number of neurons in the hidden layers are 20 and 10, respectively. For the experiment of data valuation on larger data point / data group, we use MLP models with 3 hidden layers as the utility learning model. Each fully-connected layer has LeakyReLU as the activation function and is regularized by Dropout (Srivastava et al., 2014). The input datasets are encoded as binary vector in the natural sense, where 1 indicates the data point is present, and 0 indicates the data point is missing. We use Adam optimizer with learning rate 10^{-3}, mini-batch size 32 to train all of the utility models mentioned above for up to 800 epochs.

For fair comparisons, we always fix the same training budget m_{train} for different baselines. For Group Testing, we leverage half of the training budget to estimate the Shapley value of the last data point and the other half of the training budget to estimate the differences in Shapley value between data points. We use CVXOPT[3] library to solve the constrained minimization problem in the least core approximation. We set the degree of CGA as 2 in the experiment. We use SGD optimizer with learning rate 10^{-3}, batch size 32 to train the CGA model.

For every experiment we conduct, we repeat each heuristic computation for 10 times to obtain the error bars.
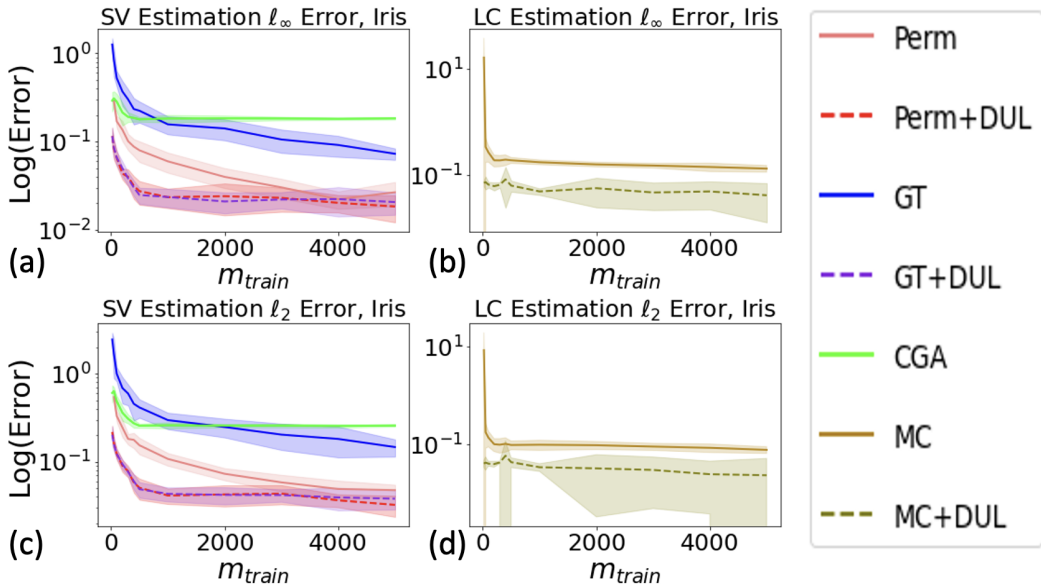
---

[3]https://cvxopt.org/

Figure 2: ℓ_2, ℓ_∞ approximation error for Shapley and Least core with the change of number of samples for Iris dataset with 15 data points. DUL stands for data utility learning.

E.2 Additional Results on Tiny Dataset

We show ℓ_2, ℓ_∞ errors in estimating SV or LC in Figure 2. For these two less stringent error metrics, we can still see that data utility learning still greatly reduces the Shapley/Least core estimation error with relatively small amount of sampled utilities.

Moreover, we show additional results on a synthetic dataset in Figure 3. To generate the synthetic dataset, we sample 10 data points from a bivariate Gaussian distribution where the means are 0.1 and -0.1 on each dimension, and covariance matrix is identity matrix. The labels are assigned based on the sign of the sum of the two features. We define the utility of a subset as the test accuracy of a logistic regression model trained on the subset. A logistic regression classifier trained on the 10 data points could achieve around 80% test accuracy. As we can see from Figure 3, data utility learning again significantly improve the SV and LC estimation performance.

E.3 Data Removal

We evaluate the Shapley/Least core value estimations on larger datasets by comparing the performance on data removal task. Specifically, we remove the most (least) valuable portion of dataset and see how the utility of the remaining dataset changes. Intuitively, a better data value estimate can better identify the importance of data points. Hence, when the data with the highest (lowest) value estimates are removed, a better data value estimation method would lead to a faster (slower) performance drop. Similar to the experiment on tiny datasets, we experiment on both synthetic and real-world datasets. For the synthetic data generation, we sample 200 data points from a 50-dimensional Gaussian distribution, where the covariance matrix is identity matrix and the 50 mean parameters are sampled uniformly from $[-1, 1]$. Each data point is labeled by the sign of the sum of the data point vector. The utility of a subset is defined by the test accuracy of a logistic regression classifier trained on the dataset. For the real-world data experiment, we select 2000 data points from PubFig83 (Pinto et al., 2011) dataset and the utility refers to the Top-5 accuracy of a simplified VGG model trained on it (for facial recognition). Since in error simulation experiment we showed that CGA does not perform well in estimating Shapley values for data valuation, and since CGA is not scalable to larger datasets as discussed in Section 3, we do not compare it as a baseline for this experiment.

We experiment with different training budget m_{train}, and we set the prediction budget $m_{\text{pred}} = 10m_{\text{train}}$ as utility model evaluation is much faster than retraining. We show the results of $m_{\text{train}} = 500, 1000, 2000$ for synthetic data in Figure 4, and $m_{\text{train}} = 2500$ for PubFig83 in Figure 5. These
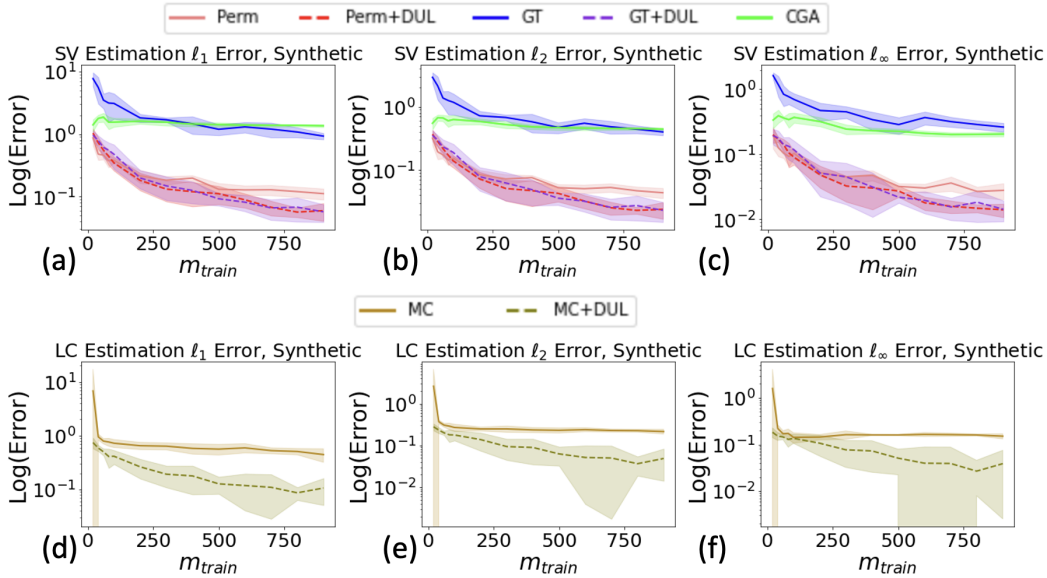
Figure 3: $\ell_1, \ell_2, \ell_\infty$ approximation error for Shapley and Least core with the change of number of samples for synthetic dataset with 10 data points. DUL stands for data utility learning.

are clearly low-resource settings, as computing the exact Shapley or least core require 2^{200} times of training on synthetic data and 2^{2000} times of training on PubFig83. As we can see, the estimation heuristics equipped with data utility learning consistently performs better on identifying the most and least valuable data points. This means that the Shapley and Least core values estimated with predicted utilities are at least more effective in predicting the most and least valuable (in a sense) data points in these settings. As a side note, the Shapley value estimated by Permutation sampling is superior to the Least core estimated by Monte-Carlo algorithm, which does not agree with the experiment results in Yan & Procaccia (2020). An interesting future work is to better understand the comparison between Shapley and Least core for data valuation.

E.4 DATA VALUATION ON GROUPS OF DATA

We also experiment on estimating Shapley and Least core values for groups of data points. This is a potentially more realistic and useful setting since in practice, more than one data records will be collected from one party. We divide Adult dataset (Dua & Graff, 2017) into 200 groups. The size of each group is varied. The proportion of data points allocated to each group is sampled from Dirichlet distribution where α is 30 in all dimensions. This design ensures that there are moderate amount of variations in group sizes. A utility sample in this setting refers to a coalition of groups and the performance of a logistic regression trained on the data points provided by the coalition.

Since Adult dataset is a highly unbalanced dataset, we use F1-score as the utility metric. We vary the training budget m_{train} from 500 to 2000, and the prediction budget $m_{\text{pred}} = 10 m_{\text{train}}$ as in the previous section. As we can see from Figure 6, the heuristics equipped with data utility learning is again favorable to both of the Shapley and Least core estimation.
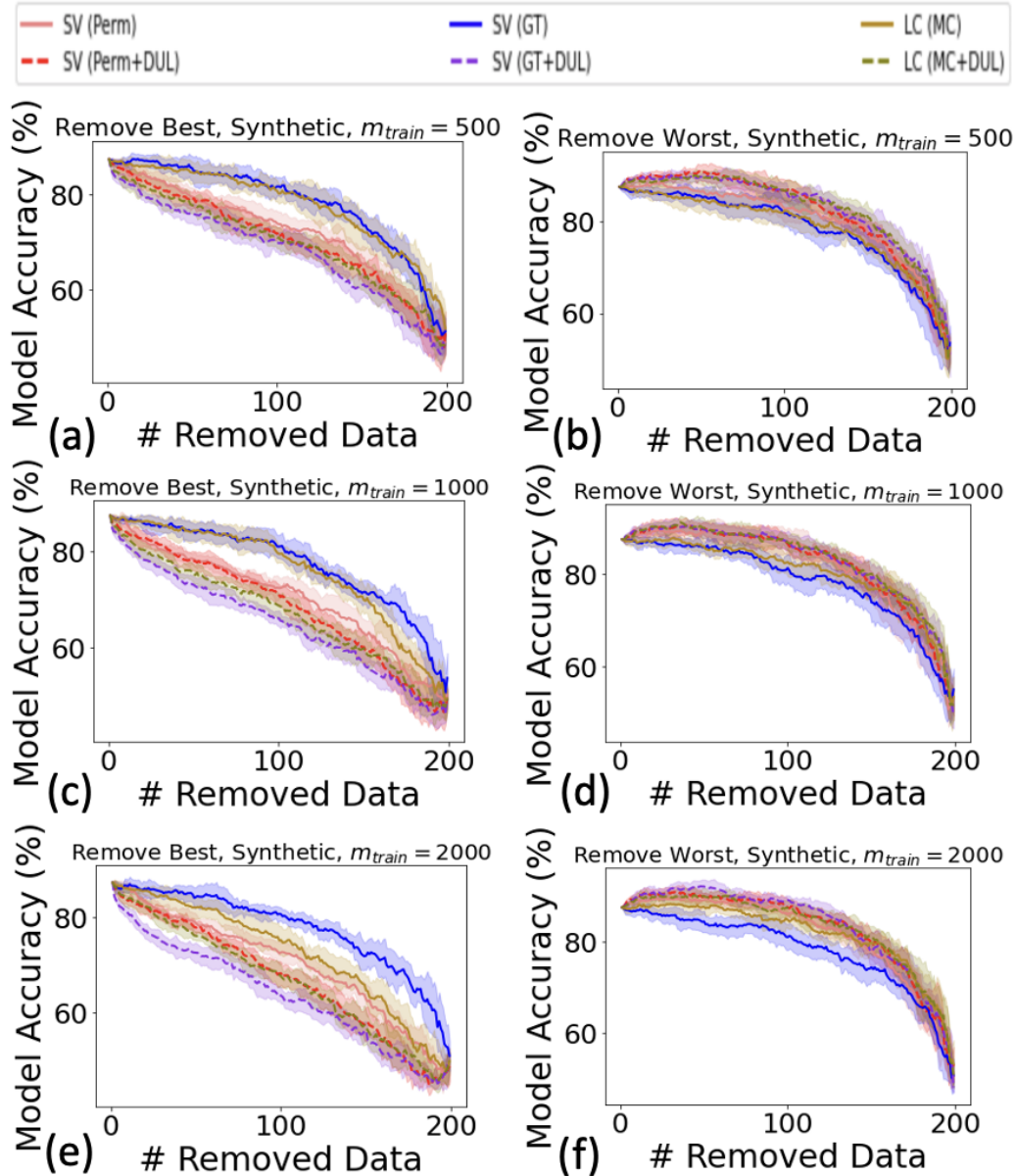
Figure 4: Curves of model test performance when the best/worst **synthetic** data points ranked according to Shapley (**SV**) or Least core (**LC**) estimations are removed. The left column ((a), (c), (e)) removes the best data points. The steeper the drop, the better. The right column ((b), (d), (f)) removes the worst data points. The sharper the rise, the better. DUL stands for data utility learning.
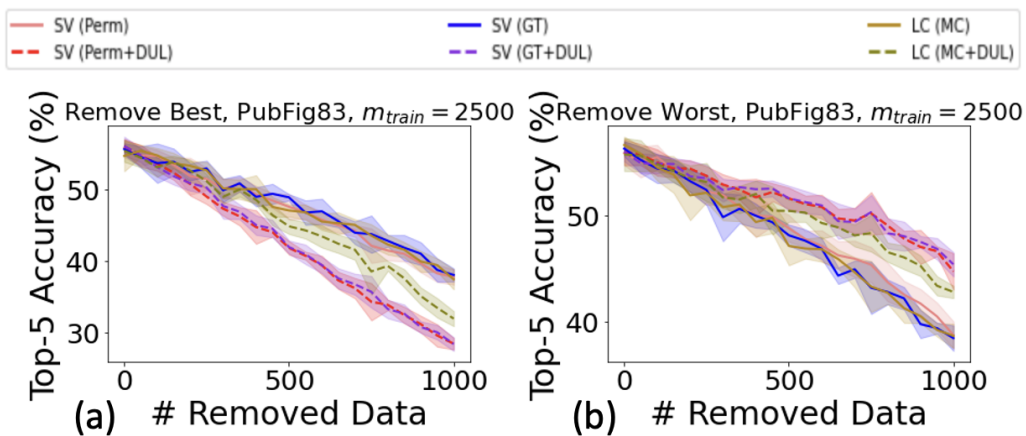
Figure 5: Curves of model test performance when the best/worst **PubFig83** data points ranked according to Shapley (**SV**) or Least core (**LC**) estimations are removed. The left column ((a), (c), (e)) removes the best data points. The steeper the drop, the better. The right column ((b), (d), (f)) removes the worst data points. The sharper the rise, the better. DUL stands for data utility learning.
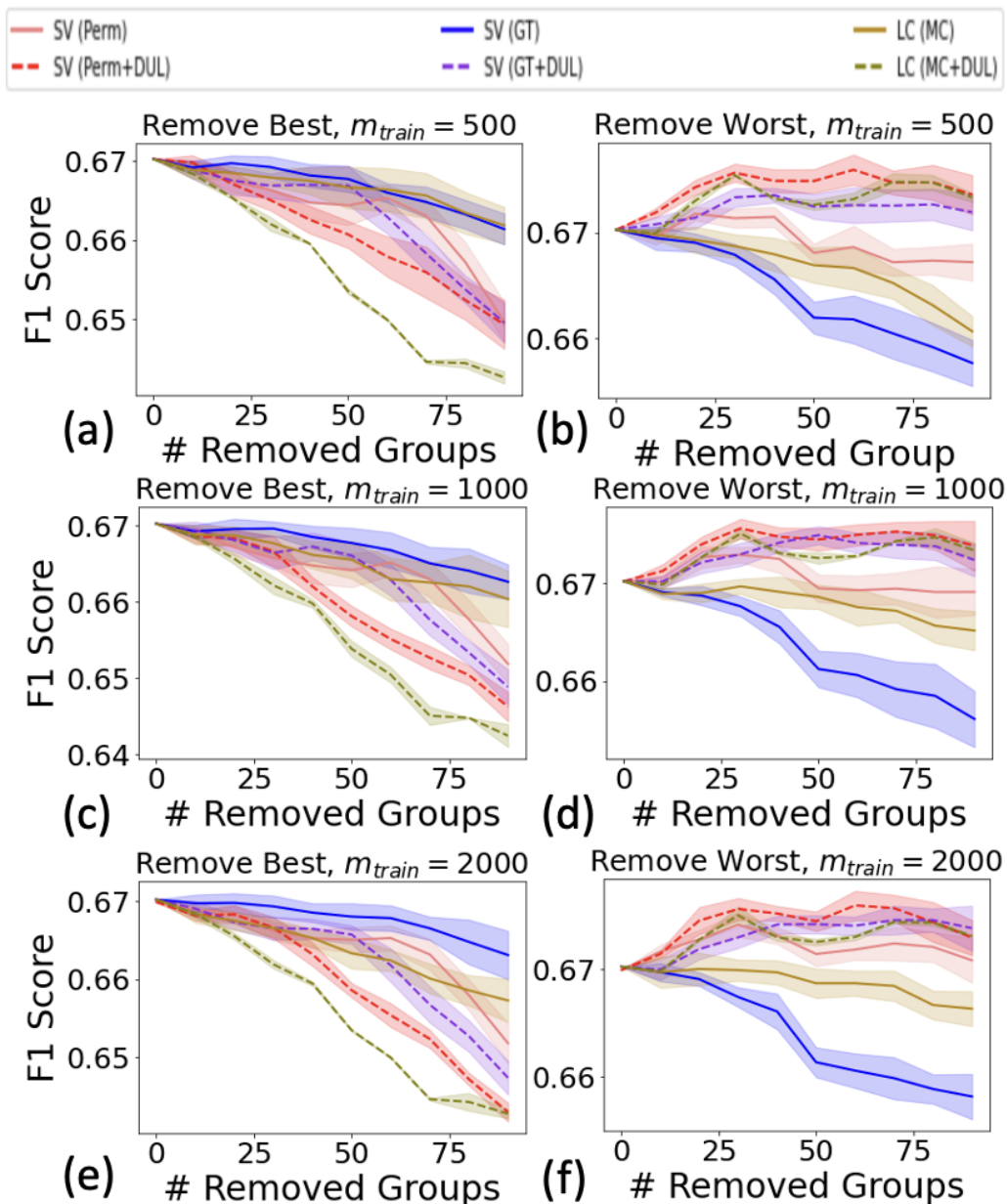
Figure 6: Curves of model test performance when the best/worst **adult** data groups ranked according to Shapley (**SV**) or Least core (**LC**) estimations are removed. The left column ((a), (c), (e)) removes the best data points (groups). The steeper the drop, the better. The right column ((b), (d), (f)) removes the worst data points (groups). The sharper the rise, the better. DUL stands for data utility learning.