Towards Efficient Data Valuation Based on the Shapley Value

Ruoxi Jia[1*], **David Dao**[2*], **Boxin Wang**[3], **Frances Ann Hubis**[2], **Nick Hynes**[1], **Nezihe Merve Gurel**[2], **Bo Li**[4], **Ce Zhang**[2], **Dawn Song**[1], **Costas Spanos**[1]
[1]University of California at Berkeley, [2]ETH, Zurich
[3]Zhejiang University, [4]University of Illinois at Urbana-Champaign

Abstract

"How much is my data worth?" is an increasingly common question posed by organizations and individuals alike. An answer to this question could allow, for instance, fairly distributing profits among multiple data contributors and determining prospective compensation when data breaches happen. In this paper, we study the problem of *data valuation* by utilizing the Shapley value, a popular notion of value which originated in coopoerative game theory. The Shapley value defines a unique payoff scheme that satisfies many desiderata for the notion of data value. However, the Shapley value often requires *exponential* time to compute. To meet this challenge, we propose a repertoire of efficient algorithms for approximating the Shapley value. We also demonstrate the value of each training instance for various benchmark datasets.

1 Introduction

Data analytics using machine learning (ML) is an increasingly common practice in modern science and business. The data for building an ML model are often provided by multiple entities. For instance, Internet enterprises analyze various users' data to improve product design, customer retention, and initiatives that help them earn revenue. Furthermore, the quality of the data from different entities may vary widely. Therefore, a key question often asked by stakeholders of a ML system is how to fairly allocate the revenue generated by a ML model to the data contributors.

This question is also motivated by a system we are building together with one of the largest hospital in the US. In the system, patients submit part of their medical records onto a "data market," and analysts pay a certain amount of money to train a ML model on patients' data. One of the challenges in such data markets is how to distribute the payment from analysts back to the patients.

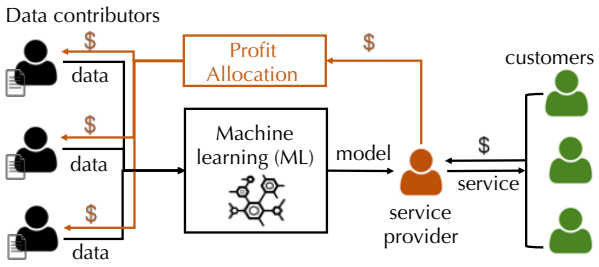

Figure 1: Overview of the data valuation problem.

A natural way of tackling the data valuation problem is to adopt a game-theoretic viewpoint, where each data contributor is modeled as a player in a coaltional game and the usefulness of data from any subset of contributors is characterized via a utility function. The Shapley value (SV) is a classic method in cooperative game theory to distribute the total gains generated by the coalition of all players, and has been applied to problems in various domains, ranging from economics [13], counter-terrorism [20, 16], environmental science [23], to ML [6]. The reason for its broad adoption is that the SV defines a unique profit allocation scheme that satisfies a set of properties with appealing real-world interpretations, such as fairness, rationality, and decentralizability.

Despite the desirable properties of the SV, computing the SV is known to be expensive; the number of utility function evaluations required by the exact SV calculation grows exponentially in the number of players. This poses a radical challenge to using the SV in the context of data valuation—how to calculate, or approximate the SV over millions or even billions of

*Equal contribution. Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

data points, a scale that is rare in previous applications of the SV, but not uncommon for real-world data valuation tasks. Even worse, for ML tasks, evaluating the utility function itself (e.g., testing accuracy) is already computationally expensive, as it requires to train a model. Due to the computational challenge, the application of the SV to data valuation has thus far been limited to stylized examples, in which the underlying utility function of the game is simple and the resulting SV can be represented as a closed-form expression [14, 5]. The state-of-the-art method to estimate the SV for a black-box utility function is based on Monte Carlo simulations [19], which still requires re-training ML models for superlinearly many times and is thus clearly impracticable. In this paper, we attempt to answer the question on whether it is possible to efficiently estimate the SV while achieving the same performance guarantee as the state-of-the-art method.

Theoretical Contribution We first study this question from a theoretical perspective. We show that, to approximate the SV of N data points with provable error guarantees, it is possible to design an algorithm with a *sublinear* amount of model evaluations—$\mathcal{O}(\sqrt{N}\log(N)^2)$. We achieve this by enabling proper information sharing between different model evaluations. Moreover, if it is reasonable to assume that the SV is "sparse" in the sense that only few data points have significant values, then we are able to further reduce the number of model training to $\mathcal{O}(\log\log(N))$, when the model can be incrementally maintained. It is worth noting that these two algorithms are agnostic to the context wherein the SV is computed; hence, they are also useful for the applications beyond data valuation.

Practical Contribution Despite the improvements from a theoretical perspective, retraining models for multiple times may still be unaffordable for large datasets and ML models. We then introduce two practical SV estimation algorithms specific to ML tasks by introducing various assumptions on the utility function. We show that if a learning algorithm is uniformly stable [2], then uniform value division produces a fairly good approximation to the true SV. In addition, for a ML model with smooth loss functions, we propose to use the influence function [15] to accelerate the data valuation process. However, the efficiency does not come for free. The first algorithm relies on the stability of a learning algorithm, which is difficult to prove for complex ML models, such as deep neural networks. The compromise that we have to make in the second algorithm is that the resulting SV estimates no longer have provable guarantees on the approximation error. Filling the gap between theoretical soundness and practicality is important future work.

Table 1 summarizes the contributions of this paper. In the rest of the paper, we will elaborate on the idea and analysis of these algorithms, and further use them to compute the data values for various benchmark datasets.

2 Related Work

Originated from game theory, the SV, in its most general form, can be #P-complete to compute [8]. Efficiently estimating SV has been studied extensively for decades. For bounded utility functions, Maleki et al. [19] described a sampling-based approach that requires $\mathcal{O}(N\log N)$ samples to achieve a desired approximation error in l_∞ norm and $\mathcal{O}(N^2\log N)$ in l_2 norm. Bachrach et al. [1] also leveraged a similar approach but focused on the case where the utility function has binary outputs. By taking into account special properties of the utility function, one can derive more efficient approximation algorithms. For instance, Fatima et al. [11] proposed a probabilistic approximation algorithm with $\mathcal{O}(N)$ complexity for weighted voting games. The game-theoretic analysis of the value of personal data has been explored in [5, 14], which proposed a fair compensation mechanism based on the SV like ours. They derived the SV under simple data utility models abstracted from network games or recommendation systems, while our work focuses on more complex utility functions derived from ML applications. In our case, the SV on longer has closed-form expressions. We develop novel and efficient approximation algorithms to overcome this hurdle.

Using the SV in the context of ML is not new. For instance, the SV has been applied to feature selection [6, 28, 21, 25, 17]. While their contributions have inspired this paper, many assumptions made for feature "valuation" do not hold for data valuation. As we will see, by studying the SV tailored to data valuation, we can develop novel algorithms that are more efficient than the previous approaches [19].

Despite not being used for data valuation, ranking the importance of training data points has been used for understanding model behaviors, detecting dataset errors, etc. Existing methods include using the influence function [15] for smooth parametric models and a variant [27] for non-parametric ones. Ogawa et al. [22] proposed rules to identify and remove the least influential data in order to reduce the computation cost when training support vector machines (SVM). One can also construct coresets—weighted data subsets—such that models trained on these coresets are provably competitive with models trained on the full dataset [7]. These approaches could potentially be used for valuing data; however, it is not clear whether they satisfy the

Table 1: Summary of Technical Results. N is the number of data points.

	Assumptions	Techniques	Complexity		Approximation
			incrementally trainable models	otherwise	
Existing	Bounded utility	Permutation sampling	$\mathcal{O}(N\log(N))$ model training and $\mathcal{O}(N^2\log(N))$ eval	$\mathcal{O}(N^2\log(N))$ model training and eval	(ϵ,δ)
Application -agnostic	Bounded utility	Group testing	$\mathcal{O}(\sqrt{N}\log(N)^2)$ model training and eval	$\mathcal{O}(\sqrt{N}\log(N)^2)$ model training and eval	(ϵ,δ)
	Sparse value	Compressive permutation sampling	$\mathcal{O}(\log\log(N))$ model training $\mathcal{O}(N\log\log(N))$ and eval	$\mathcal{O}(N\log\log(N))$ model training and eval	(ϵ,δ)
ML-specific	Stable learning	Uniform division	$\mathcal{O}(1)$ computation		$(\epsilon,0)$
	Smooth utility	Influence function	$\mathcal{O}(N)$ optimization routines		Heuristic

properties desired by data valuation, such as fairness. We leave it for future work to understand these distinct approaches for data valuation.

3 Problem Formulation

Consider a dataset $D=\{z_i\}_{i=1}^N$ containing data from N users. Let $U(S)$ be the utility function, representing the value calculated by the additive aggregation of $\{z_i\}_{i\in S}$ and $S\subseteq I=\{1,\cdots,N\}$. Without loss of generality, we assume throughout that $U(\emptyset)=0$. Our goal is to partition $U_{\text{tot}}\triangleq U(I)$, the utility of the entire dataset, to the individual users; more formally, we want to find a function that assigns to user i a number $s(U,i)$ for a given utility function U. We suppress the dependency on U when the utility is self-evident and use s_i to represent the value allocated to user i.

The SV [26] is a classic concept in cooperative game theory to attribute the total gains generated by the coalition of all players. Given a utility function $U(\cdot)$, the SV for user i is defined as the average marginal contribution of z_i to all possible subsets of $D=\{z_i\}_{i\in I}$ formed by other users:

$$s_i=\sum_{S\subseteq I\setminus\{i\}}\frac{1}{N\binom{N-1}{|S|}}\big[U(S\cup\{i\})-U(S)\big] \quad (1)$$

The formula in (1) can also be stated in the equivalent form:

$$s_i=\frac{1}{N!}\sum_{\pi\in\Pi(D)}\big[U(P_i^\pi\cup\{i\})-U(P_i^\pi)\big] \quad (2)$$

where $\pi\in\Pi(D)$ is a permutation of users and P_i^π is the set of users which precede user i in π. Intuitively, imagine all users' data are to be collected in a random order, and that every user i receives his data's marginal contribution that would bring to those whose data are already collected. If we average these contributions over all the possible orders of users, we obtain s_i. The importance of the SV stems from the fact that it is the *unique* value division scheme that satisfies the following desirable properties.

1. Group Rationality: The value of the entire dataset is completely distributed among all users, i.e., $U(I)=\sum_{i\in I}s_i$.

2. Fairness: (1) Two users who are identical with respect to what they contribute to a dataset's utility should have the same value. That is, if user i and j are equivalent in the sense that $U(S\cup\{i\})=U(S\cup\{j\}),\forall S\subseteq I\setminus\{i,j\}$, then $s_i=s_j$. (2) Users with zero marginal contributions to all subsets of the dataset receive zero payoff, i.e., $s_i=0$ if $U(S\cup\{i\})=0$ for all $S\subseteq I\setminus\{i\}$.

3. Additivity: The values under multiple utilities sum up to the value under a utility that is the sum of all these utilities: $s(U,i)+s(V,i)=s(U+V,i)$ for $i\in I$.

The *group rationality* property states that any rational group of users would expect to distribute the full yield of their coalition. The *fairness* property requires that the names of the users play no role in determining the value, which should be sensitive only to how the utility function responds to the presence of a user's data. The *additivity* property facilitates efficient value calculation when data is used for multiple applications, each of which is associated with a specific utility function. With additivity, one can decompose a given utility function into an arbitrary sum of utility functions and compute utility shares separately, resulting in transparency and decentralizability. The fact that the SV uniquely possesses these properties, combined with its flexibility to support different utility functions, leads us to employ the SV to attribute the total gains generated from a dataset to each user.

4 Efficient SV Estimation

The challenge in adopting the SV lies in its computational cost. Evaluating the exact SV using Eq. (1) involves computing the marginal utility of every user to every coalition, which is $\mathcal{O}(2^N)$. Even worse, in many ML tasks, evaluating utility *per se* (e.g., testing accuracy) is computationally expensive as it requires training a ML model. In this section, we present vari-

ous efficient algorithms for approximating the SV. We say that $\hat{s} \in \mathbb{R}^N$ is a (ϵ, δ)-approximation to the true SV $s = [s_1, \cdots, s_N]^T \in \mathbb{R}^N$ with respect to l_p-norm if $P[||\hat{s}_i - s_i||_p \leq \epsilon] \geq 1 - \delta$. Throughout this paper, we will measure the approximation error in terms of l_2 norm.

4.1 Baseline: Permutation Sampling

We start by describing a baseline algorithm [18] that approximates the SV for any bounded utility functions with provable guarantees. Let π be a random permutation of I and each permutation has a probability of $1/N!$. Consider the random variable $\phi_i = U(P_i^\pi \cup \{i\}) - U(P_i^\pi)$. According to (2), $s_i = \mathbb{E}[\phi_i]$. Thus, we can estimate s_i by the sample mean. An application of Hoeffding's bound indicates that the number of permutations needed to achieve an (ϵ, δ)-approximation is $m_{\text{perm}} = (2r^2 N/\epsilon^2) \log(2N/\delta)$, where r is the range of the utility function. For each permutation, the utility function is evaluated N times in order to compute the marginal contribution for all N users; therefore, the number of utility evaluations involved in the baseline approach is $m_{\text{eval}} = N m_{\text{perm}} = \mathcal{O}(N^2 \log N)$.

Note that for an ML task, we can write the utility function $U(S) = U_m(A(S))$, where $A(\cdot)$ represents a learning algorithm that maps a dataset S onto a model and $U_m(\cdot)$ is some measure of model performance, such as test accuracy. Typically, a substantial part of computational costs associated with the utility evaluation lies in $A(\cdot)$. Hence, it is useful to examine the efficiency of an approximation algorithm in terms of the number of model training required. In general, one utility evaluation would need to re-train a model. Particularly, when $A(\cdot)$ is incrementally trainable, one pass over the entire training set allows us to evaluate ϕ_i for all $i = 1, \cdots, N$. Hence, in this case, the number of model training needed achieving an (ϵ, δ)-approximation is the same as $m_{\text{perm}} = \mathcal{O}(N \log N)$.

4.2 Group Testing-Based Approach

We now describe an algorithm that makes the same assumption of bounded utility as the baseline algorithm, but requires significantly fewer utility evaluations than the baseline.

Our proposed approximation algorithm is inspired by previous work applying the group testing theory to feature selection [29]. Recall the group testing is a combinatorial search paradigm [9], in which one wants to determine whether each item in a set is "good" or "defective" by performing a sequence of tests. The result of a test may be positive, indicating that at least one of the items of that subset is defective, or negative, indicating that all items in that subset are good. Each test is performed on a pool of different items and the number of tests can be made significantly smaller than the number of items by smartly distributing items into pools. Hence, the group testing is particularly useful when testing an individual item's quality is expensive. Analogously, we can think of SV calculation as a group testing problem with continuous quality measure. Each user's data is an "item" and the data utility corresponds to the item's quality. Each "test" in our scenario corresponds to evaluating the utility of a subset of users and is expensive. Drawing on the idea of group testing, we hope to recover the utility of all user subsets from a small amount of customized tests.

Let T be the total number of tests. At test t, a random set of users is drawn from I and we evaluate the utility of the selected set of users. If we model the appearance of user i and j's data in a test as Boolean random variables β_i and β_j, respectively, then the difference between the utility of user i and that of user j is

$$(\beta_i - \beta_j) U(\beta_1, \cdots, \beta_N) \tag{3}$$

where $U(\beta_1, \cdots, \beta_N)$ is the utility evaluated on the users with the Boolean appearance random variable equal to 1.

Using the definition of the SV, one can derive the following formula of the SV difference between any pair of users.

Lemma 1. *For any $i, j \in I$, the difference in SVs between i and j is*

$$s_i - s_j = \frac{1}{N-1} \sum_{S \subseteq I \setminus \{i,j\}} \frac{U(S \cup \{i\}) - U(S \cup \{j\})}{\binom{N-2}{|S|}} \tag{4}$$

Due to the space limitation, we omit all the proofs of the paper to our supplemental materials. The key idea of the proposed algorithm is to smartly design the sampling distribution of $\beta_1, \cdots, \beta_N$ such that the expectation of (3) mirrors the Shapley difference in (4). This will enable us to calculate the Shapely differences from the test results with a high-probability error bound. The following Lemma states that if we can estimate the Shapley differences between all data pairs up to $(\epsilon/\sqrt{N}, \delta/N)$, then we will be able to recover the SV with the approximation error (ϵ, δ).

Lemma 2. *Suppose that C_{ij} is an $(\epsilon/(2\sqrt{N}), \delta/(N(N-1)))$-approximation to $s_i - s_j$. Then, the solution to the feasibility problem*

$$\sum_{i=1}^{N} \hat{s}_i = U_{tot} \tag{5}$$

$$|(\hat{s}_i - \hat{s}_j) - C_{i,j}| \leq \epsilon/(2\sqrt{N}) \quad \forall i, j \in \{1, \ldots, N\} \tag{6}$$

is an (ϵ, δ)-approximation to s with respect to l_2-norm.

Algorithm 1 presents the pseudo-code of the group testing-based algorithm, which first estimates the Shapley differences and then derives the SV from the Shapley differences by solving a feasibility problem.

Algorithm 1: Group Testing Based SV Estimation.

input : Training set - $D = \{(x_i, y_i)\}_{i=1}^N$, utility function $U(\cdot)$, the number of tests - T

output : The estimated SV of each training point - $\hat{s} \in \mathbb{R}^N$

$Z \leftarrow 2\sum_{k=1}^{N-1} \frac{1}{k}$;

$q(k) \leftarrow \frac{1}{Z}(\frac{1}{k} + \frac{1}{N-k})$ for $k = 1, \cdots, N-1$;

Initialize $\beta_{ti} \leftarrow 0$, $t = 1, ..., T, i = 1, ..., N$;

for $t = 1$ *to* T **do**

  Draw $k \sim q(k)$;

  for $j = 1$ *to* k_t **do**

    Uniformly sample a length-k sequence S from $\{1, \cdots, N\}$;

    $\beta_{ti} \leftarrow 1$ for all $i \in S$;

  end

  $u_t \leftarrow U(\{i : \beta_{ti} = 1\})$;

end

$\Delta U_{ij} \leftarrow \frac{Z}{T}\sum_{t=1}^{T} u_t(\beta_{ti} - \beta_{tj})$ for $i = 1, .., N$, $j = 1, ..., N$ and $j \geq i$;

Find $\hat{s}$ by solving the feasibility problem $\sum_{i=1}^N \hat{s}_i = U(D), |(\hat{s}_i - \hat{s}_j) - \Delta U_{i,j}| \leq \epsilon/(2\sqrt{N}), \forall i, j \in \{1, \cdots, N\}$;

The following theorem provides a lower bound on the number of tests T needed to achieve an (ϵ, δ)-approximation.

Theorem 3. *Algorithm 1 returns an (ϵ, δ)-approximation to the SV with respect to l_2-norm if the number of tests T satisfies $T \geq 8\log\frac{N(N-1)}{2\delta}/\big((1 - q_{tot}^2)h\big(\frac{\epsilon}{Zr\sqrt{N}(1-q_{tot}^2)}\big)\big)$, where $q_{tot} = \frac{N-2}{N}q(1) + \sum_{k=2}^{N-1} q(k)[1 + \frac{2k(k-N)}{N(N-1)}]$, $h(u) = (1+u)\log(1+u) - u$, $Z = 2\sum_{k=1}^{N-1}\frac{1}{k}$, and r is the range of the utility function.*

Note that $Z = 2\sum_{k=1}^{N-1}\frac{1}{k} \leq 2(\log(N-1)+1)$ and $1/h(1/(Z\sqrt{N})) \leq 1/\log(1 + 1/(Z\sqrt{N})) \leq Z\sqrt{N} + 1$. Since only one utility evaluation is required for a single test, the number of utility evaluations is at most $\mathcal{O}(\sqrt{N}(\log N)^2)$. On the other hand, in the baseline approach, the number of utility evaluations is $\mathcal{O}(N^2 \log N)$. Hence, the group testing requires significantly fewer model evaluations than the baseline.

4.3 Exploiting the Sparsity of Values

We now present an algorithm inspired by our empirical observations of the SV for large datasets. This algorithm can produce an (ϵ, δ)-approximation to the SV with only $\mathcal{O}(N \log(N) \log(\log(N)))$ utility evaluations.

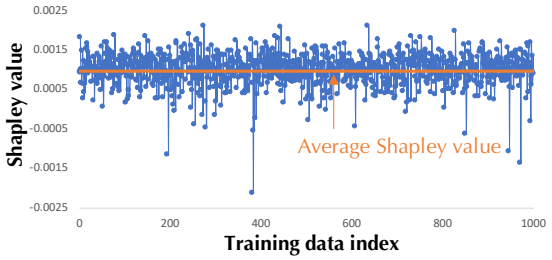

Figure 2: The distribution of the SV of a size-1000 training set randomly sampled from MNIST. $\sigma_{367}(s)/(\sum_{i=1}^N s_i) = 0.5$. The utility function is the test accuracy.

Figure 2 illustrates the distribution of the SV of the MNIST dataset, from which we observed that the SV is "approximately sparse"—most of values are concentrated around its mean and only a few data points have significant values. In the literature, the "approximate sparsity" of a vector s is characterized by a small error of its best K-term approximation:

$$\sigma_K(s) = \inf\{\|s - z\|_1, \text{ z is } K\text{-sparse}\} \tag{7}$$

This observation opens up a vast collection of tools from compressive sensing for the purpose of calculating the SV.

Compressive sensing studies the problem of recovering a sparse signal s with far fewer measurements $y = As$ than the length of the signal. A sufficient condition for recovery is that the measurement matrix $A \in \mathbb{R}^{M \times N}$ satisfies a key property, the *Restricted Isometry Property (RIP)*. In order to ensure that A satisfies this property, we simply choose A to be a random Bernoulli matrix. The results in random matrix theory implies that A satisfies RIP with high probability. Define the kth restricted isometry constant δ_k for a matrix A as

$$\delta_k(A) = \min\{\delta : \forall s, \|s\|_0 \leq k, \\ (1-\delta)\|s\|_2^2 \leq \|As\|_2^2 \leq (1+\delta)\|s\|_2^2 \tag{8}$$

It has been shown in [24] that every k-sparse vector s can be recovered by solving a convex optimization problem

$$\min_{s \in \mathbb{R}^N} \|s\|_1, \quad \text{s.t. } As = y \tag{9}$$

if $\delta_{2s}(A) < 1/3$. This result can also be generalized to noisy measurements [3]. Drawing on the ideas of compressed sensing, we present Algorithm 2, termed compressive permutation sampling.

Theorem 4. *There exists some constant C' such that if $M \geq C'(K\log(N/(2K)) + \log(2/\delta))$ and $T \geq$*

Algorithm 2: Compressive Permutation Sampling.

input : Training set - $D = \{(x_i, y_i)\}_{i=1}^N$, utility function $U(\cdot)$, the number of measurements - M, the number of permutations - T
output : The SV of each training point - $\hat{s} \in \mathbb{R}^N$
Sample a Bernoulli matrix A, where $A_{m,i} \in \{-1/\sqrt{M}, 1/\sqrt{M}\}$ with equal probability;
for $t \leftarrow 1$ **to** T **do**
  $\pi_t \leftarrow$ GenerateUniformRandomPermutation(D);
  $\phi_i^t \leftarrow U(P_i^{\pi_t} \cup \{i\}) - U(P_i^{\pi_t})$ for $i = 1, \ldots, N$;
  for $m \leftarrow 1$ **to** M **do**
    $\hat{y}_{m,t} \leftarrow \sum_{i=1}^N A_{m,i}\phi_i^t$;
  end
end
$\bar{y}_m = \frac{1}{T}\sum_{t=1}^T \hat{y}_{m,t}$ for $m = 1, \ldots, M$;
$\bar{s} = U(D)/N$;
$\Delta s^* \leftarrow \operatorname{argmin}_{\Delta s \in \mathbb{R}^N} \|\Delta s\|_1, \text{s.t. } \|A(\bar{s} + \Delta s) - \bar{y}\|_2 \leq \epsilon$;
$\hat{s} = \bar{s} + \Delta s^*$;

$\frac{2r^2}{\epsilon^2}\log\frac{4M}{\delta}$, *except for an event of probability no more than* δ, *the output of Algorithm 2 obeys*

$$\|\hat{s} - s\|_2 \leq C_{1,K}\epsilon + C_{2,K}\frac{\sigma_K(s)}{\sqrt{K}} \tag{10}$$

for some constants $C_{1,K}$ *and* $C_{2,K}$.

Therefore, the number of utility evaluations (and model training) required for achieving the approximation error guarantee in Theorem 4 is $NT = \mathcal{O}(N\log(\log(N)))$. Particularly, when the utility function is defined with respect to an incrementally trainable model, only $\log\log(N)$ full model training is needed for achieving the error guarantee.

4.4 Stable Learning Algorithms

A learning algorithm is *stable* if the model learned by the algorithm is insensitive to the removal of an arbitrary point in the training dataset [2]. More specifically, an algorithm G has uniform stability γ with respect to the loss function l if $\|l(G(S),\cdot) - l(G(S^{\setminus i}),\cdot)\|_\infty \leq \gamma$ for all $i \in \{1, \cdots, |S|\}$, where S denotes the training set and $S^{\setminus i}$ denotes the one by removing ith element of S. Indeed, a broad variety of learning algorithms are stable, including all learning algorithms with Tikhonov regularization. Stable learning algorithms are appealing as they enjoy provable generalization error bounds [2]. Assume that the model is trained via a stable learning algorithm and training data's utility is measured in terms of the testing loss. Due to the inherent insensitivity of a stable learning algorithm to the training data, we expect that the SV of each training point is similar to one another. The following theorem confirms our intuition and provides an upper bound on the SV difference between any pair of training data points.

Theorem 5. *For a learning algorithm* $A(\cdot)$ *with uniform stability* $\beta = \frac{C_{stab}}{|S|}$, *where* $|S|$ *is the size of the training set and* C_{stab} *is some constant. Let the utility of* D *be* $U(D) = M - L_{test}(A(D), D_{test})$, *where* $L_{test}(A(D), D_{test}) = \frac{1}{N}\sum_{i=1}^N l(A(D), z_{test,i})$ *and* $0 \leq l(\cdot,\cdot) \leq M$. *Then,* $s_i - s_j \leq 2C_{stab}\frac{1+\log(N-1)}{N-1}$ *and the Shapley difference vanishes as* $N \to \infty$.

By Lemma 2, if $2C_{\text{stab}}\frac{1+\log(N-1)}{N-1}$ is less than $\epsilon/(2\sqrt{N})$, uniformly assigning $\frac{U_{\text{tot}}}{N}$ to each data contributor provides an $(\epsilon, 0)$-approximation to the SV.

4.5 Heuristic Based on Influence Functions

Computing the SV involves evaluating the change in utility of all possible sets of data points after adding one more point. A plain way to evaluate the difference requires training a large number of models on different subsets of data. Koh et al. [15] show that influence functions can be used as an efficient approximation of parameter changes after adding or removing one point. Therefore, the need for re-training models is circumvented. Assume that model parameters are obtained by solving an empirical risk minimization problem $\hat{\theta}^m = \operatorname{argmin}_\theta \frac{1}{m}\sum_{i=1}^m l(z_i, \theta)$. Applying the result in [15], we can approximate the parameters learned after adding z by using the relation $\hat{\theta}_z^{m+1} = \hat{\theta}^m - \frac{1}{m}H_{\hat{\theta}^m}^{-1}\nabla_\theta L(z, \hat{\theta}^m)$ where $H_{\hat{\theta}^m} = \frac{1}{m}\sum_{i=1}^m \nabla_\theta^2 L(z_i, \hat{\theta}^m)$ is the Hessian. The parameter change after removing z can be approximated similarly, except replacing the $-$ by $+$ in the above formula. The efficiency of the baseline permutation sampling method can be significantly improved by combining it with influence functions. Moreover, we can employ a more sophisticated sampling scheme to reduce the variance of the result. Indeed, we can re-write the SV as $s_i = \frac{1}{N}\sum_{k=1}^N \mathbb{E}[X_i^k]$, where $X_i^k = U(S \cup \{i\}) - U(S)$ is the marginal contribution of user i to a size-k subset that is randomly selected with probability $1/\binom{N-1}{k}$. This suggests that stratified sampling can be used to approximate the SV, which customizes the number of samples for estimating each expectation term according to the variance of X_i^k.

Largest-S Approximation. One practical heuristic of using influence functions is to consider a single subset S for computing s_i, namely, $I \setminus \{i\}$. With this heuristic, we can simply take a trained model on the whole dataset, and calculate the influence function for each data point. For logistic regression models, the first and second derivations enjoy closed-form expressions and the change in parameters after removing one point $z = (x, y)$ can be approximated by

$-\left(\sum_{i=1}^{N}\sigma(x_i^T\hat{\theta}^N)\sigma(-x_i^T\hat{\theta}^N)x_ix_i^T\right)^{-1}\sigma(-yx_i^T\hat{\theta}^N)yx$ where $\sigma(u) = 1/(1+\exp(-u))$ and $y \in \{-1, 1\}$. The fact that largest-S influence only considers a single subset makes it impossible to satisfy the *group rationality* and *additivity* properties simultaneously.

Theorem 6. *Consider the value attribution scheme that assigns the value* $\hat{s}(U,i) = C_U[U(S\cup\{i\})-U(S)]$ *to user* i *where* $|S| = N-1$ *and* C_U *is a constant such that* $\sum_{i=1}^{N}\hat{s}(U,i) = U(I)$. *Consider two utility functions* $U(\cdot)$ *and* $V(\cdot)$. *Then,* $\hat{s}(U+V,i) \neq \hat{s}(U,i)+\hat{s}(V,i)$ *unless* $V(I)[\sum_{i=1}^{N}U(S\cup\{i\})-U(S)] = U(I)[\sum_{i=1}^{N}V(S\cup\{i\})-V(S)]$.

5 Experimental Results

Comparing Approximation Accuracy. We first compare the proposed approximation methods that only require mild assumptions on the ML models (e.g., bounded or differentiable utility), including (a) the permutation sampling baseline, (b) the group testing-based method, (c) using influence functions to approximate all marginal contributions, and (d) approximating the SV with only the influence function to the largest subset. The last two methods are hereinafter referred to as *all-S influence* and *largest-S influence*, respectively. We use a small-scale dataset, `iris`, and use (a) to estimate the true SV for a regularized logistic regression up to $\epsilon = 1/N$. Figure 6(d) shows that the approximations produced by (a)-(c) are closest to each other. The result of the largest-S influence are correlated with that of the other techniques, although it cannot recover the true SV.

Runtime comparison. We implement the SV calculation techniques on a machine with 16 cores (Intel Xeon CPU E5-2620 v4 @ 2.10GHz) and compare the runtime of different techniques on a two-class `dog-vs-fish` dataset [15] of size 900 constructed from the ImageNet dataset. To evaluate the runtime for training sizes above 900, we concatenate duplicate copies of the dog-vs-fish dataset. For each training data point, we first pre-compute the 2048-dimensional inception features and then train a logistic regression using the stochastic gradient descent for 150 epochs. The utility function is the negative testing loss of the logistic regression model. For the largest-S influence and the all-S influence, we use the method in [15] to compute the influence function. The runtime of different techniques in logarithmic scale is displayed in Figure 3 (b). We can see that the group testing-based method outperforms the permutation sampling baseline by several orders of magnitude for a large number of data points. By exploiting influence function heuristics and the stratified sampling trick in Section 4.5, the computational costs can be further reduced. Due to the fact that the largest-S influence heuristic only focuses on the marginal contribution of each training data point to a single subset, it is much more efficient than the permutation sampling, group testing and the all-S influence, which compute the marginal contributions to a large number of subsets.

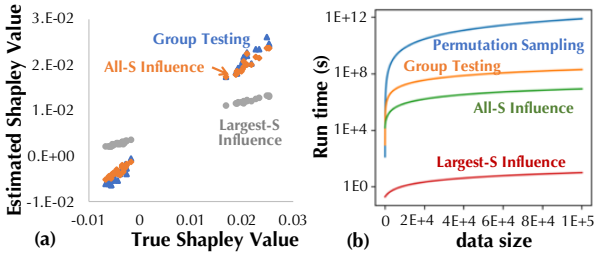

Figure 3: Consider the SV approximation methods that do not rely on specific assumptions on the underlying learning algorithms and compare the (a) data values produced by them for training a logistic regression model and (b) their runtime.

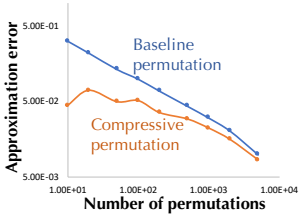

Figure 4: Comparison of approximation errors with different number of permutations for the baseline permutation sampling and the compressive permutation sampling method.

Approximation under sparsity assumptions. When it is plausible to assume the SV of a training set is sparse, we could employ the idea of compressive sensing to recover the SV with fewer samples. Figure 4 compares the sample efficiency of the baseline permutation sampling and the compressive permutation sampling method on a size-1000 dataset sampled randomly from `MNIST`. For a given approximation error, the compressive permutation requires significantly fewer samples and model valuations than the baseline approach. The superiority of the compressive permutation becomes less evident at the large sample regime.

Stable learning algorithms. Our theoretical result in Section 4.4 shows that the SV of training data tends to be uniform for a stable learning algorithm, which has a small stability parameter β. We empirically validate this result by training a ridge regression on the `diabetes` dataset and varying the strength of its regularization term. In [2], it is shown

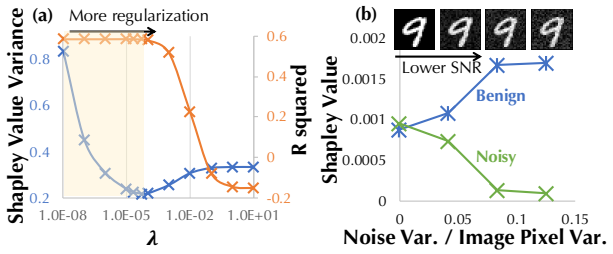

Figure 5: (a) Variance of data values for a ridge regression with different regularization strength (λ). (b) Tradeoff between data value and privacy.

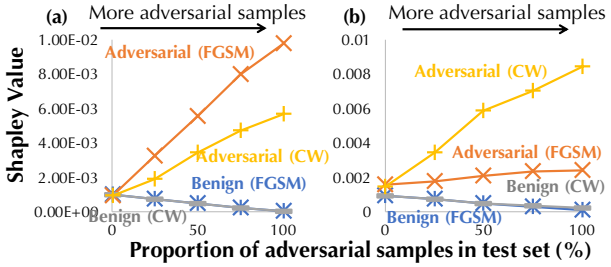

Figure 6: (a, b) Comparison of SV of benign and adversarial examples. FGSM and CW are different attack algorithms used for generating adversarial examples in the testing dataset: (a) (resp. (b)) is trained on Benign + FGSM (resp. CW) adversarial examples.

that the stability parameter β of the ridge regression $\min_\theta \frac{1}{N}\sum_{i=1}^{N} l(\theta, z_i) + \lambda\|\theta\|^2$ is proportional to σ^2/λ, where σ is the Lipschitz constant of the loss function with respect to the model parameter θ and equal to $2|x_i^T\theta - y_i| \cdot |x_i|$. When the model fits the training data well, the change in σ is small; therefore, applying more regularization leads to a more stable learning algorithm, which has lower variance in the training data values as illustrated in the shaded area of Figure 5. On the other hand, if the model no longer fits the data well due to excessive regularization, then σ will dominate the stability parameter. In this case, since σ increases with the regularization strength, β and thereby the variance of the SV also increase. Note that the variance of the SV is identical to the approximation error of a uniform value division scheme.

Value for Privacy-Preserving Data. Differential privacy [10] has emerged as a standard privacy notation and is often achieved by adding noise that has a magnitude proportional to the desired privacy level. On the other hand, noise diminishes the usefulness of data and thereby degrades the value of data. We construct a training set using the `MNIST`, and divide the training dataset into two halves, one half containing normal images and the other half containing noisy ones. The testing accuracy on normal images is used as the utility function. Figure 5(b) illustrates a clear tradeoff between privacy and data value - the SV decreases as data becomes noisier.

Value for Adversarial Examples. Mixing adversarial examples with benign examples in the training dataset, or adversarial training, is an effective method to improve the adversarial robustness of a model. In practice, we measure the robustness in terms of the testing accuracy on a dataset containing adversarial examples. We expect that the adversarial examples in the training dataset become more valuable as more adversarial examples are added into the testing dataset. Based on the `MNIST`, we construct a training dataset that contains both benign and adversarial examples and synthesize testing datasets with different adversarial-benign mixing ratios. Two popular attack algorithms, namely, Fast Gradient Sign Method (FGSM) [12] and the Carlini and Wagner (CW) attack [4] are used to generate adversarial examples. Figure 6(a, b) compares the average SV for adversarial examples and for benign examples in the training dataset. The negative testing loss for logistic regression is used as the utility function. We see that the SV of adversarial examples increases as the testing data becomes more adversarial and contrariwise for benign examples. This is consistent with our expectation. In addition, the adversarial examples in the training set are more valuable if they are generated from the same attack algorithm for testing adversarial examples.

6 Conclusion

ML has opened up exciting opportunities to tackle a wide variety of problems; nevertheless, very few works have attempted to understand the value of data used for training models. A principled way of data valuation is the key to stimulate data exchange, enabling the development of more sophisticated and robust ML models. We adopt the SV, a classic concept from cooperative game theory, for data valuation. The SV has many unique properties appealing to data valuation. However, the lack of efficient methods to compute the SV has prevented it from being adopted in the past. We develop a repertoire of techniques for estimating the SV in different scenarios.

For future work, We wish to continue exploring the connection between ML and game theory and develop efficient valuation methods for ML models. It is also critical to understand other concepts from cooperative game theory (e.g., stable coalition) in the context of data valuation. Last but not least, we hope to apply the techniques to real-world applications and revolutionize the way of data collection and dissemination.

Acknowledgements

This work is supported in part by the Republic of Singapores National Research Foundation through a grant to the Berkeley Education Alliance for Research in Singapore (BEARS) for the Singapore-Berkeley Building Efficiency and Sustainability in the Tropics (SinBerBEST) Program. This work is also supported in part by the CLTC (Center for Long-Term Cybersecurity); FORCES (Foundations Of Resilient CybEr-Physical Systems), which receives support from the National Science Foundation (NSF award numbers CNS-1238959, CNS-1238962, CNS-1239054, CNS1239166); and the National Science Foundation under Grant No. TWC-1518899. CZ and the DS3Lab gratefully acknowledge the support from Mercedes-Benz Research & Development NA, MeteoSwiss, Oracle Labs, Swiss Data Science Center, Swisscom, Zurich Insurance, Chinese Scholarship Council, and the Department of Computer Science at ETH Zurich.

References

[1] Y. Bachrach, E. Markakis, A. D. Procaccia, J. S. Rosenschein, and A. Saberi. Approximating power indices. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 943–950. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[2] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[3] E. J. Candes, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[4] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[5] M. Chessa and P. Loiseau. A cooperative game-theoretic approach to quantify the value of personal data in networks. In *Proceedings of the 12th workshop on the Economics of Networks, Systems and Computation*, page 9. ACM, 2017.

[6] S. Cohen, E. Ruppin, and G. Dror. Feature selection based on the shapley value. *In other words*, 1:98Eqr, 2005.

[7] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney. Sampling algorithms and coresets for \ell_p regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.

[8] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.

[9] D. Du, F. K. Hwang, and F. Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

[10] C. Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[11] S. S. Fatima, M. Wooldridge, and N. R. Jennings. A linear approximation method for the shapley value. *Artificial Intelligence*, 172(14):1673–1699, 2008.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[13] F. Gul. Bargaining foundations of shapley value. *Econometrica: Journal of the Econometric Society*, pages 81–95, 1989.

[14] J. Kleinberg, C. H. Papadimitriou, and P. Raghavan. On the value of private information. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, pages 249–257. Morgan Kaufmann Publishers Inc., 2001.

[15] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894, 2017.

[16] R. Lindelauf, H. Hamers, and B. Husslage. Cooperative game theoretic centrality analysis of terrorist networks: The cases of jemaah islamiyah and al qaeda. *European Journal of Operational Research*, 229(1):230–238, 2013.

[17] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4768–4777, 2017.

[18] S. Maleki. *Addressing the computational issues of the Shapley value with applications in the smart grid*. PhD thesis, University of Southampton, 2015.

[19] S. Maleki, L. Tran-Thanh, G. Hines, T. Rahwan, and A. Rogers. Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

[20] T. Michalak, T. Rahwan, P. L. Szczepanski, O. Skibski, R. Narayanam, M. Wooldridge, and N. R. Jennings. Computational analysis of connectivity games with applications to the investigation of terrorist networks. 2013.

[21] F. Mokdad, D. Bouchaffra, N. Zerrouki, and A. Touazi. Determination of an optimal feature selection method based on maximum shapley value. In *Intelligent Systems Design and Applications (ISDA), 2015 15th International Conference on*, pages 116–121. IEEE, 2015.

[22] K. Ogawa, Y. Suzuki, and I. Takeuchi. Safe screening of non-support vectors in pathwise svm computation. In *International Conference on Machine Learning*, pages 1382–1390, 2013.

[23] L. Petrosjan and G. Zaccour. Time-consistent shapley value allocation of pollution cost reduction. *Journal of economic dynamics and control*, 27(3):381–398, 2003.

[24] H. Rauhut. Compressive sensing and structured random matrices. *Theoretical foundations and numerical methods for sparse recovery*, 9:1–92, 2010.

[25] S. Sasikala, S. A. alias Balamurugan, and S. Geetha. A novel feature selection technique for improved survivability diagnosis of breast cancer. *Procedia Computer Science*, 50:16–23, 2015.

[26] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[27] B. Sharchilev, Y. Ustinovsky, P. Serdyukov, and M. de Rijke. Finding influential training samples for gradient boosted decision trees. *arXiv preprint arXiv:1802.06640*, 2018.

[28] X. Sun, Y. Liu, J. Li, J. Zhu, X. Liu, and H. Chen. Using cooperative game theory to optimize the feature selection problem. *Neurocomputing*, 97:86–93, 2012.

[29] Y. Zhou, U. Porwal, C. Zhang, H. Q. Ngo, X. Nguyen, C. Ré, and V. Govindaraju. Parallel feature selection inspired by group testing. In *Advances in Neural Information Processing Systems*, pages 3554–3562, 2014.