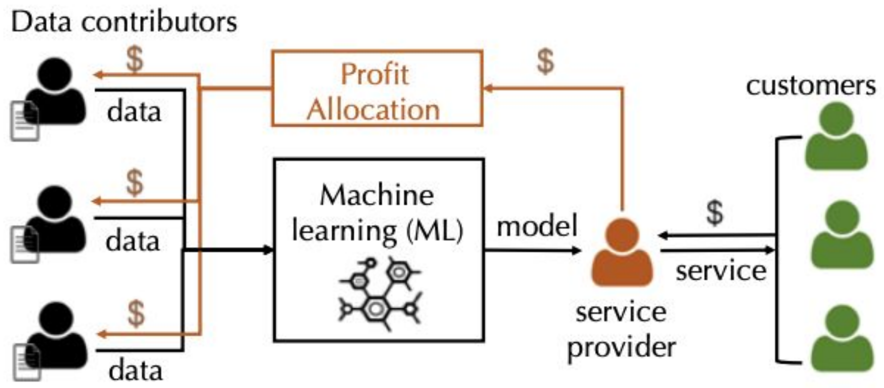
Towards Efficient Data Valuation Based on the Shapley Value

Ruoxi Jia , David Dao , Boxin Wang , Frances Ann Hubis , Nick Hynes , Nezihe Merve Gurel ,
Bo Li , Ce Zhang , Dawn Song , Costas Spanos

How do we attach value to data?

Data for ML models are often contributed by multiple parties

How do we reward the different contributors?



Background: Shapley Value

“A classic method in cooperative game theory to distribute the total gains generated by the coalition of all players”

At a high level, to determine user “u”s contribution, we consider their contribution in all subsets of all users (that include user “u”).

Utilized in multiple real-world scenarios as it addresses many desirable properties: fairness, rationality, decentralization, etc.

Can we use the Shapley Value to determine the contribution of each party’s data to the whole dataset?

Problem Formulation

For Utility function *U* and user *i*, we want to assign the user some value *s(U, i)*

Shapley Value Equation:

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

Average marginal contribution of user "i"s data to all subsets of whole dataset "D"

Problem Formulation Cont.

*“The importance of the SV stems from the fact that it is the **unique** value division scheme that satisfies the following desirable properties”*

Group Rationality: Value of entire dataset is distributed amongst all contributors

Fairness: Users with identical contributions have same SV

Additivity: $s(U, i) + s(V, i) = s(U + V, i)$

- *“The additivity property facilitates efficient value calculation when data is used for multiple applications, each of which is associated with a specific utility function”*

Efficiently Calculating Shapley Value

Computation cost of SV is very high. To calculate SV, we need to calculate marginal utility of every user to every coalition, which takes $O(2^N)$

Calculating Utility itself can be computationally expensive as well and might require an ML model

Let’s try coming up with a more efficient measure for SV by approximating. For SV approximation $\hat{\boldsymbol{s}} \in \mathbb{R}^N$, it is an ($\epsilon$, δ)-approximation to the true SV value if $P[||\hat{s}_i - s_i||_p \leq \epsilon] \geq 1 - \delta$

Critical Thinking

- Do we really need to calculate ALL subsets of the whole dataset for each user? What are some alternatives?

- If the SV measures user “u”s data contribution to the ML model, what if the data contributes negatively to the model? Will they be rewarded more, and how do we handle this?

Baseline: Permutation Sampling

Let π be a random permutation of $\boldsymbol{I}$, the set of users, and each permutation has a probability of 1/N!

Consider random variable: $\phi_i = U(P_i^\pi \cup \{i\}) - U(P_I^\pi)$

We can estimate SV for user “i” as expected value of ϕ as according to the definition earlier:

Shapley Value is measured by the average marginal contribution of user “i”s data to all subsets of whole dataset “D”

Baseline: Permutation Sampling (Runtime)

According to Hoeffding’s bound, for each user, we need “m” permutations to achieve an (ϵ, δ)-approximation is:

$$m = (2r^2N/\epsilon^2)log(2N/\delta)$$

Where “r” is the range of the utility function.

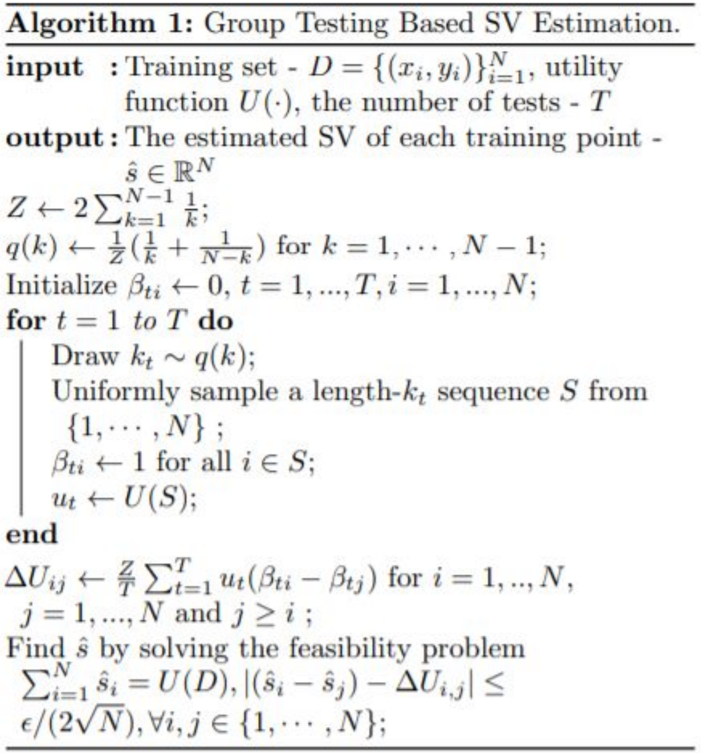
Since the runtime is N*m, we get a runtime of: $O(N^2log(N))$

Group Testing-Based Approach

Can we reduce number of utility evaluations compared to Baseline Algorithm?

We can utilize group testing, where each test we calculate the utility of randomly selected users.

Each test T only needs one utility calculation, significantly decreasing the number of utility function calls required



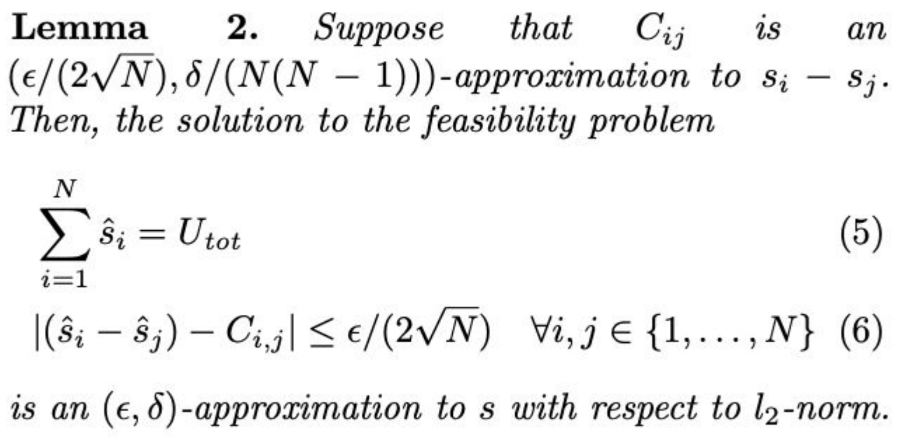
Algorithm 1: Group Testing Based SV Estimation.

```
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 $k_t \sim q(k)$;
    Uniformly sample a length-$k_t$ sequence $S$ from
     $\{1, \cdots, N\}$ ;
    $\beta_{ti} \leftarrow 1$ for all $i \in S$;
    $u_t \leftarrow U(S)$;
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
 $\sum_{i=1}^{N} \hat{s}_i = U(D), |(\hat{s}_i - \hat{s}_j) - \Delta U_{i,j}| \leq$
 $\epsilon/(2\sqrt{N}), \forall i, j \in \{1, \cdots, N\}$;
```

Group Testing-Based Approach (Finding $\hat{s}_i$)

Since we utilized group testing, how do we get individual SV for one user?

“If we can estimate the Shapley differences between all data pairs up to (ε/√ N, δ/N), then we will be able to recover the SV with the approximation error (ε, δ)”



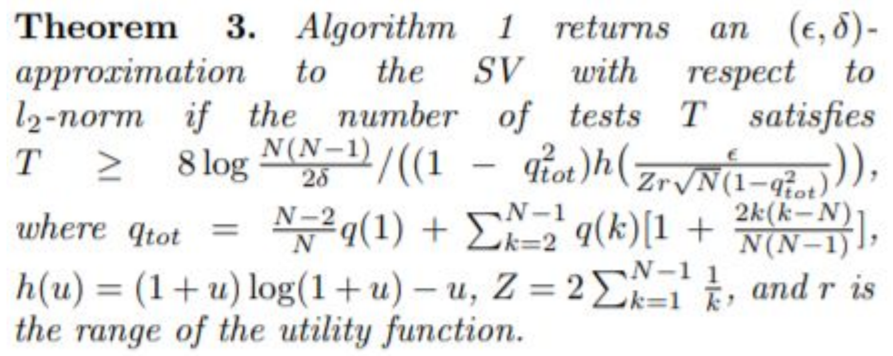
Lemma 2. *Suppose that C_{ij} is an $(\epsilon/(2\sqrt{N}), \delta/(N(N-1)))$-approximation to $s_i - s_j$. Then, the solution to the feasibility problem*

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

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

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

Group Testing-Based Approach (Runtime)



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* $T \geq 8 \log \frac{N(N-1)}{2\delta} / ((1 - q_{tot}^2) h(\frac{\epsilon}{Zr\sqrt{N}(1-q_{tot}^2)}))$, *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.*

We can use Taylor expansion to see that with large N, we have T = $O(N(log(N)^2))$

Since we only have 1 utility calculation per test, our total run time is $O(N(log(N)^2))$

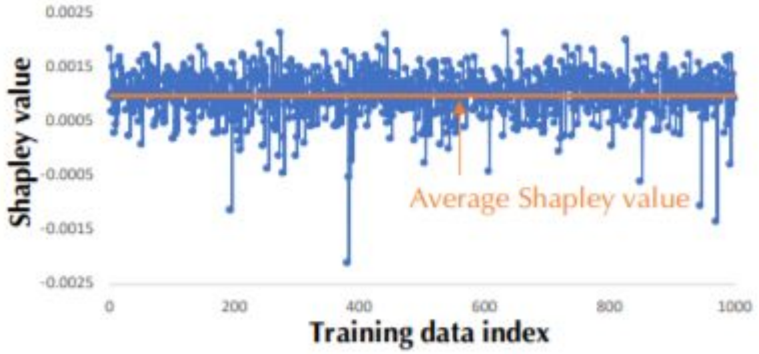
Critical Thinking

- Do we have to use random sampling for the baseline permutation algorithm? Or are there alternatives?

- What if some user “U” is never included in a test group? If this is a possibility, is random sampling still valid?

Exploiting the Sparsity of Values

Most values are concentrated around the mean and only a few data points have significant value



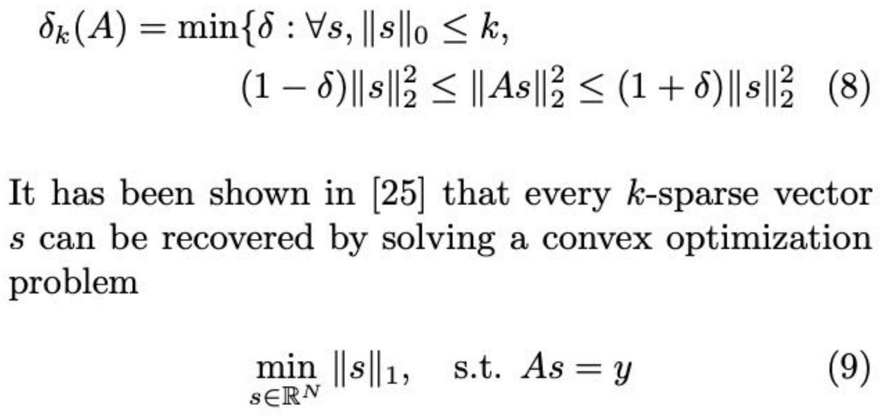
Can utilize compressive sensing, which studies recovering a sparse signal *s* with far fewer measurements *y* = *As*

Exploiting the Sparsity of Values (Choosing matrix *A*)

Matrix must satisfy *Restricted Isometry Property (RIP)*

- High Level: Sparse matrices that are nearly orthonormal

We can simply choose *A* to be a random Bernoulli matrix



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

It has been shown in [25] that every k-sparse vector s can be recovered by solving a convex optimization problem

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

Exploiting the Sparsity of Values (Runtime)

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 $m = 1, \ldots, M$;
$\bar{s} = U(D)/N$;
$\Delta s^* \leftarrow \mathrm{argmin}_{\Delta s \in \mathbb{R}^N} \|\Delta s\|_1$, s.t. $\|A(\bar{s} + \Delta s) - \bar{y}\|_2 \leq \epsilon$;

$\hat{s} = \bar{s} + \Delta s^*$;
```

For constant C', consider M and T:

$$M \geq C'(K log(N/(2K)) + log(2/\delta))$$

$$T \geq \frac{2r^2}{\epsilon^2} log \frac{4M}{\delta}$$

Runtime of evaluation is N*T:

$$O(N * log(log(N)))$$

Influence Functions

Influence Functions - analytical tools that can be used to assess the effect of removing an observation on the value of a statistic without having to recalculate that statistic

“A plain way to evaluate the difference requires training a large number of models on different subsets of data”

- Consider: $U(S \cup \{i\}) - U(S)$

By utilizing influence functions, we can avoid re-training the model often as it will be robust against parameter changes

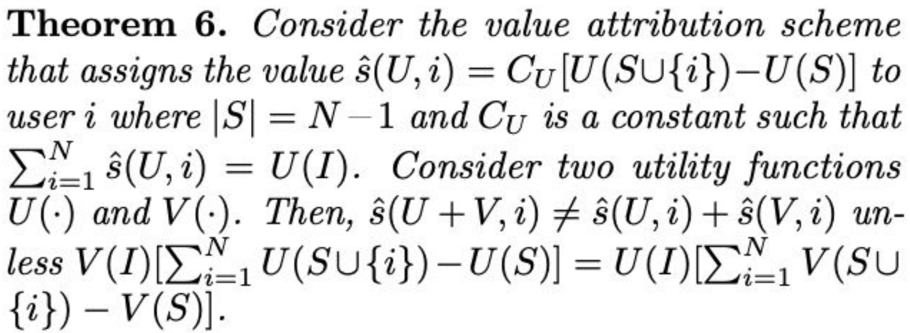
Baseline Permutation Sampling Method’s performance is greatly improved when combined with Influence Functions alongside a more sophisticated sampling technique.

Largest-S Approximation

Consider a single subset S for computing, s_i to be $I \setminus \{i\}$

With this heuristic, we can take a model trained on the whole dataset and compute the influence function for each data point rather than grabbing different subsets

Since Largest-S only considers 1 subset, it cannot satisfy both group rationality and additivity simultaneously.



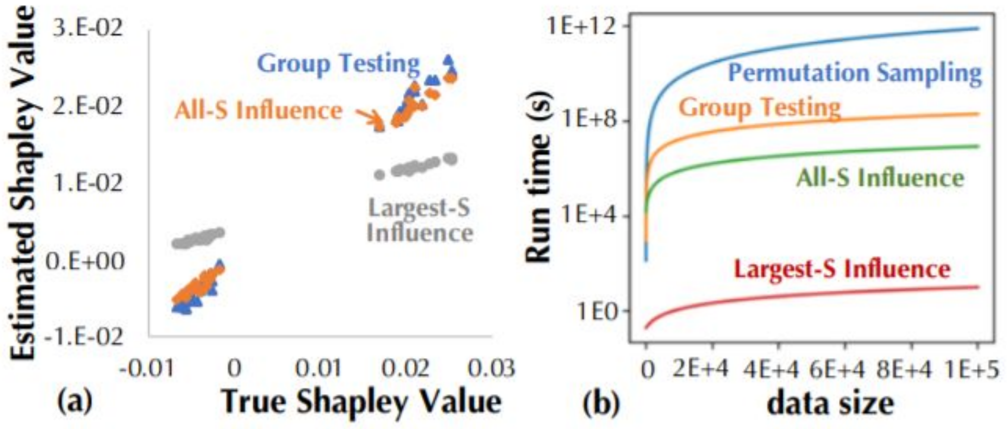
Theorem 6. *Consider the value attribution scheme that assigns the value* $\hat{s}(U,i) = C_U[U(S \cup \{i\}) - U(S)]$ *to user* i *where* $|S| = N - 1$ *and* C_U *is a constant such that* $\sum_{i=1}^{N} \hat{s}(U,i) = U(I)$. *Consider two utility functions* $U(\cdot)$ *and* $V(\cdot)$. *Then,* $\hat{s}(U+V,i)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

Experimental Results

Result Evaluation:

- Approximation Accuracy
- Runtime

How do we choose a method?



Value for Privacy-Preserving

Noise often added as a means of privacy

SV decreases with the increase of noise

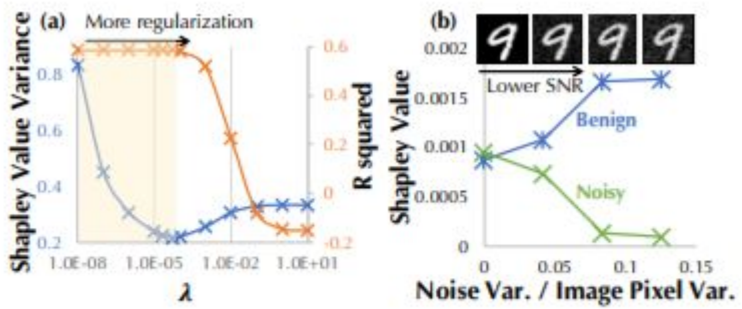
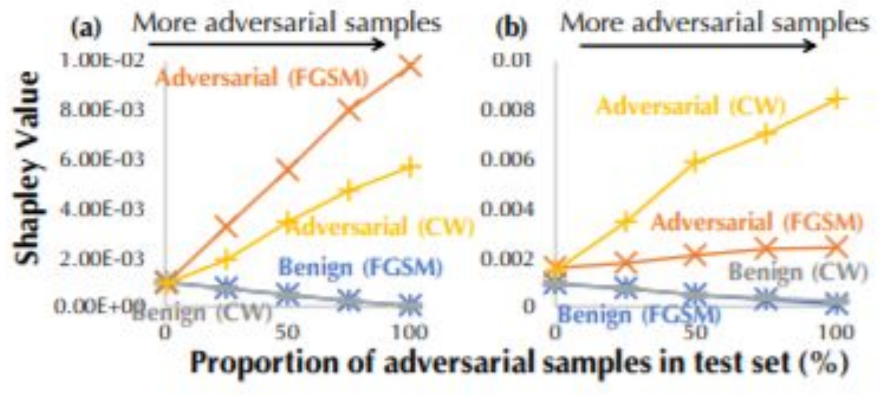


Figure 5: (a) Variance of data values for a ridge regression with different regularization strength (λ). (b) Tradeoff between data value and privacy.

Value for Adversarial Examples

SV value increases for adversarial examples

Can we potentially use this for adversarial example detection?



Critical Thinking

- Why is it an issue that Largest-S Approximation cannot satisfy both rationality and additivity?

- What are scenarios where we would want to prioritize accuracy over runtime, or vice versa?

Discussion Points

Is it possible to use this method to detect poisoned data (if submitted by same user)? Perhaps the poisoned data’s corresponding SV will stand out?

By having rewards based on data contribution, will this encourage higher quality of data in the future?

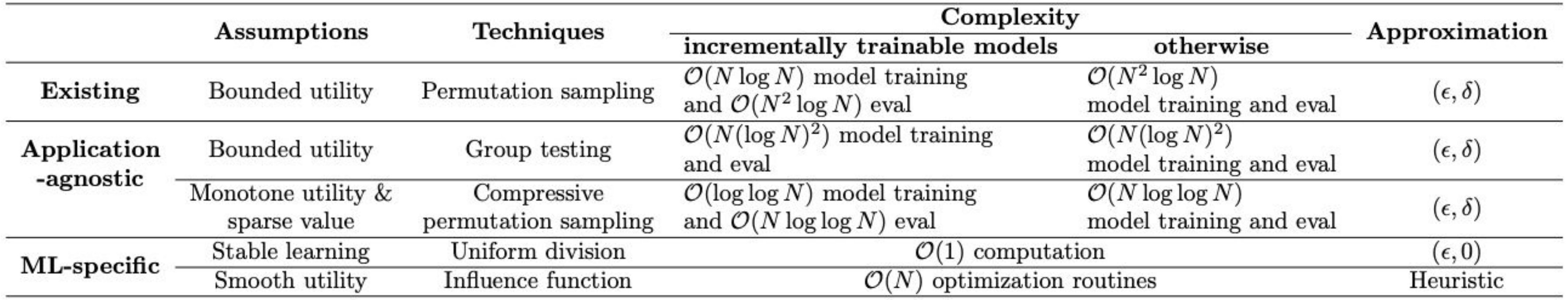
What are aspects of data that are desirable? What would a valid utility function look for in “valuable” data?

Conclusion

Shapley Value and game theory in general can contribute a lot to ML.

Valuation of data in general is an unexplored yet emerging topic.

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



	Assumptions	Techniques	Complexity: incrementally trainable models	Complexity: otherwise	Approximation
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	$\mathcal{O}(N(\log N)^2)$ model training and eval	$\mathcal{O}(N(\log N)^2)$ model training and eval	(ϵ, δ)
	Monotone utility & sparse value	Compressive permutation sampling	$\mathcal{O}(\log \log N)$ model training and $\mathcal{O}(N \log \log N)$ 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

Questions?