[Paper]

[Blog]

[Code]

DATA VALUATION USING REINFORCEMENT LEARNING

Jinsung Yoon*
Department of Electrical and Computer Engineering, UCLA, CA
jsyoon0823@g.ucla.edu

Sercan Ö. Arık
Google Cloud AI
Sunnyvale, CA
soarik@google.com

Tomas Pfister
Google Cloud AI
Sunnyvale, CA
tpfister@google.com

ABSTRACT

Quantifying the value of data is a fundamental problem in machine learning. Data valuation has multiple important use cases: (1) building insights about the learning task, (2) domain adaptation, (3) corrupted sample discovery, and (4) robust learning. To adaptively learn data values jointly with the target task predictor model, we propose a meta learning framework which we name Data Valuation using Reinforcement Learning (DVRL). We employ a data value estimator (modeled by a deep neural network) to learn how likely each datum is used in training of the predictor model. We train the data value estimator using a reinforcement signal of the reward obtained on a small validation set that reflects performance on the target task. We demonstrate that DVRL yields superior data value estimates compared to alternative methods across different types of datasets and in a diverse set of application scenarios. The corrupted sample discovery performance of DVRL is close to optimal in many regimes (i.e. as if the noisy samples were known apriori), and for domain adaptation and robust learning DVRL significantly outperforms state-of-the-art by 14.6% and 10.8%, respectively.

Presenter: Onur Aydın

Data Valuation

Business Side

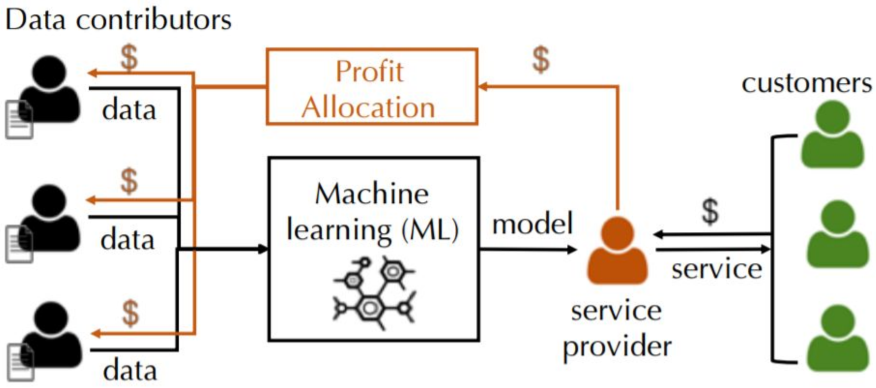


Figure 1: Overview of the data valuation problem.

“How much is my data worth?”

Jia, Ruoxi, et al. "Towards efficient data valuation based on the shapley value." *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

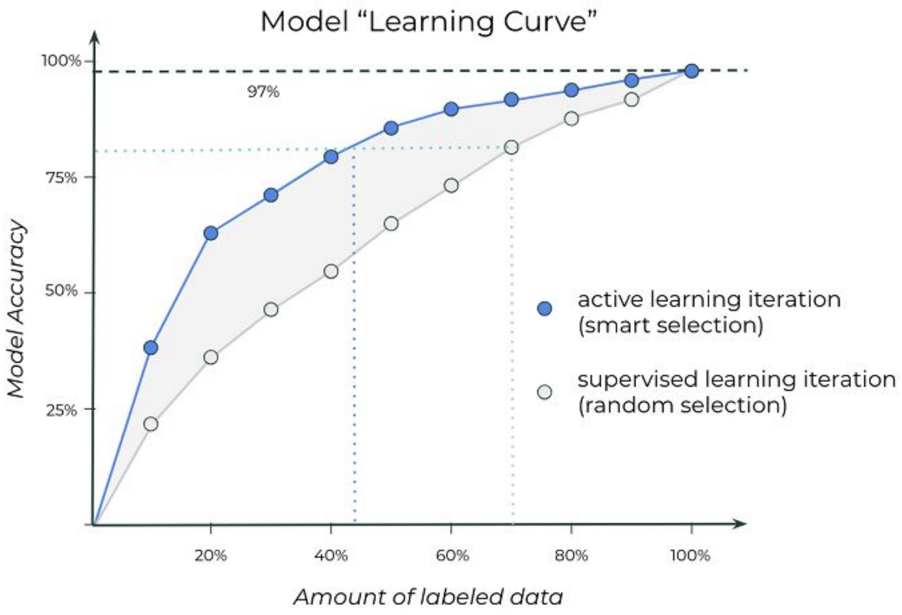
Data Valuation

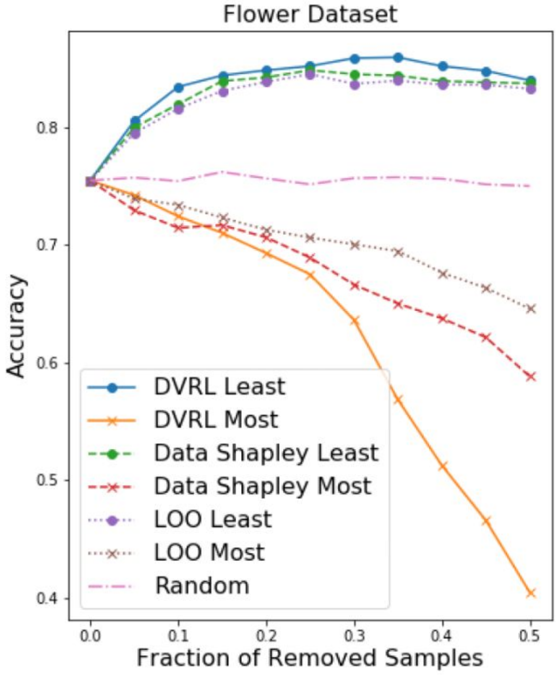
Technical Side

- Incorrect label
- Input comes from different distribution
- Input is noisy or low quality
- Usefulness for target task

Train dataset → **Test dataset**

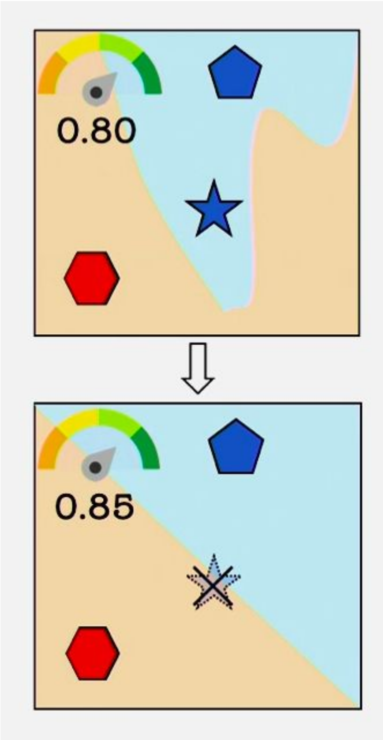
Active Learning vs. Data Valuation





Data Valuation

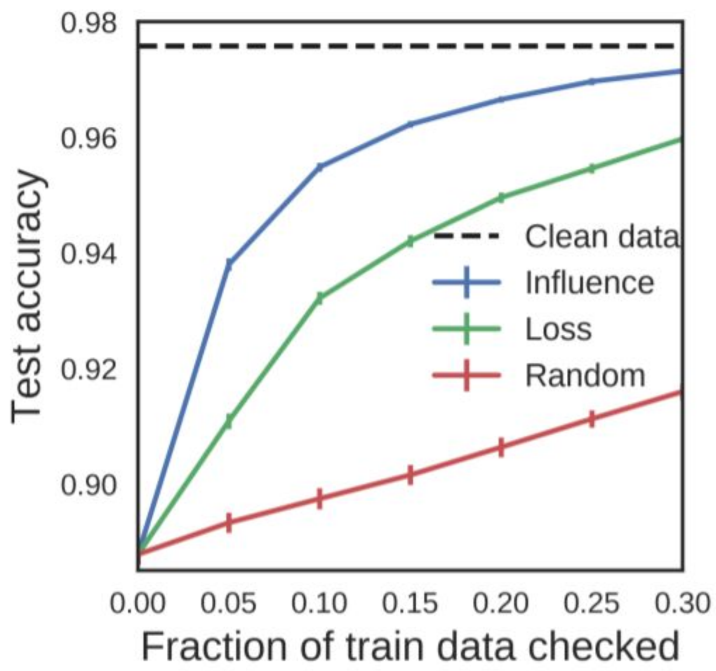
- **Leave-one-out (LOO)**
- Influence Function
- Data Shapley



Example: value (★) = 0.80 – 0.85 = -0.05

Data Valuation

- Leave-one-out (LOO)
- **Influence Function**
- Data Shapley

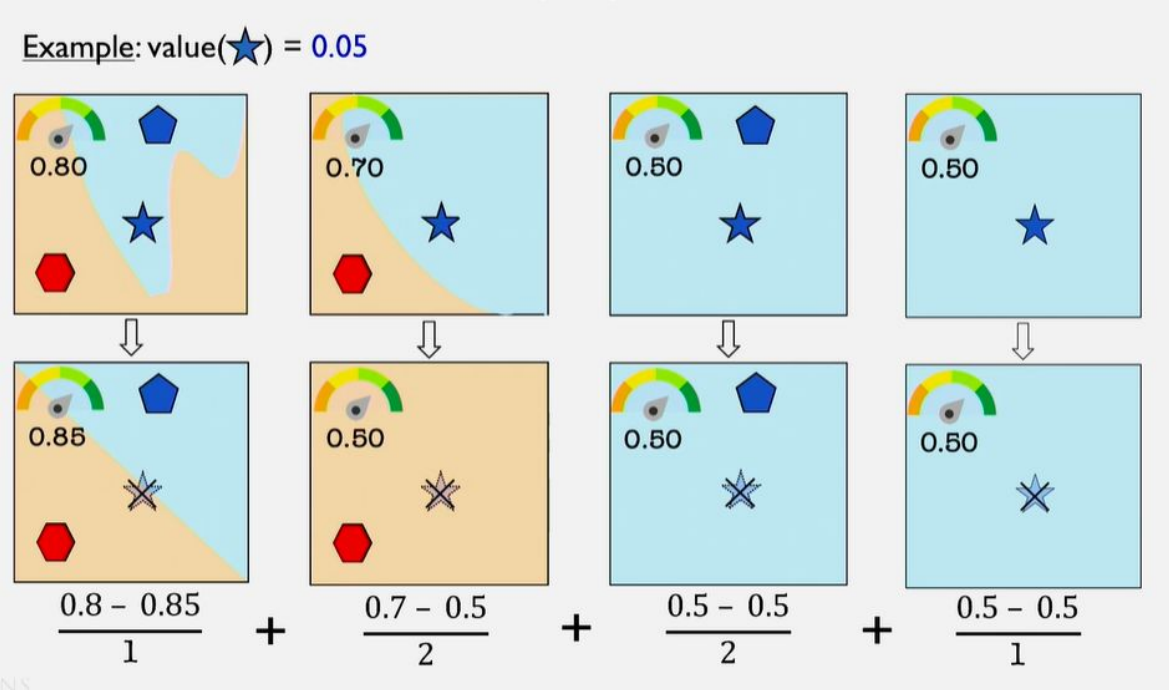


Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In ICML, pp. 1885–1894, 2017. [Link]

Data Valuation

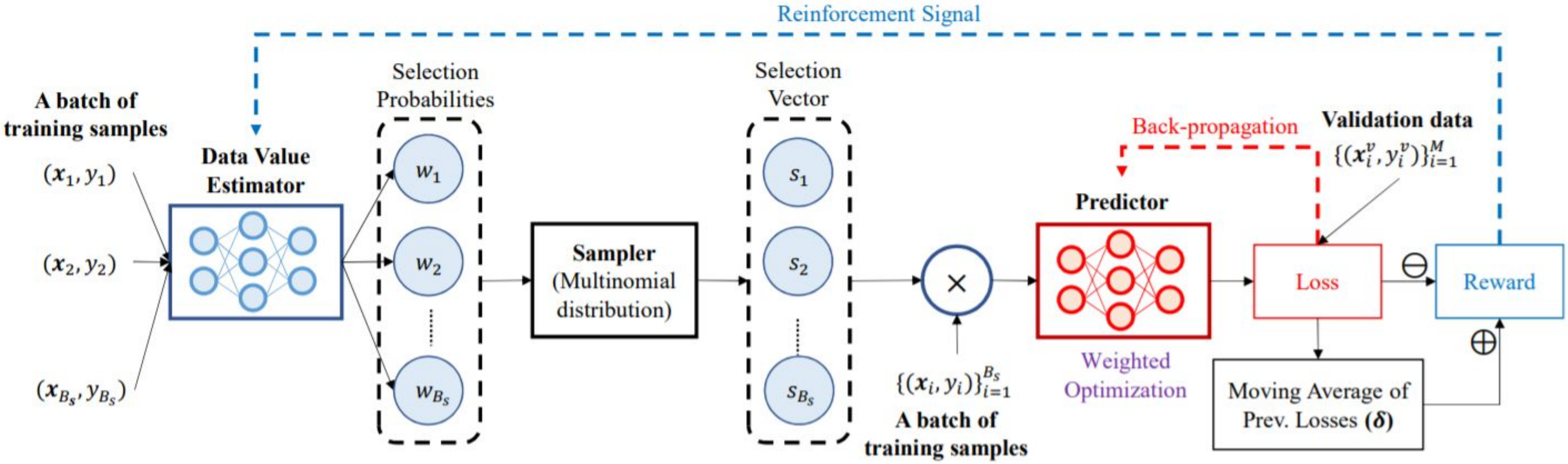
Game Theory

- Leave-one-out (LOO)
- Influence Function
- **Data Shapley**

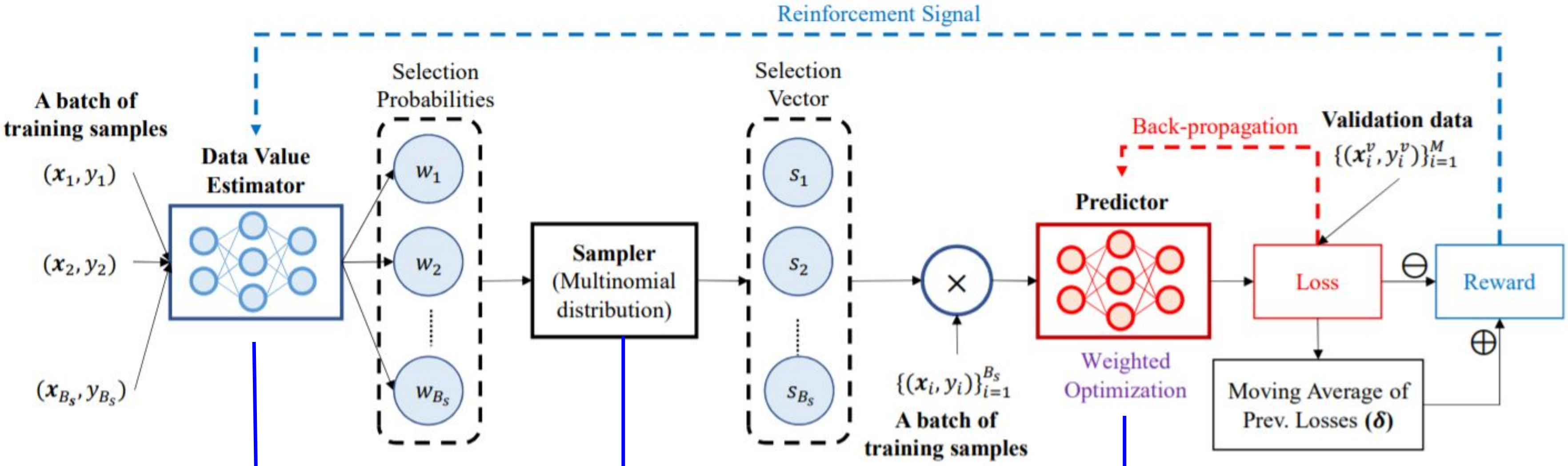


Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In ICML, pp. 2242–2251, 2019. [Link]

Architecture



Architecture

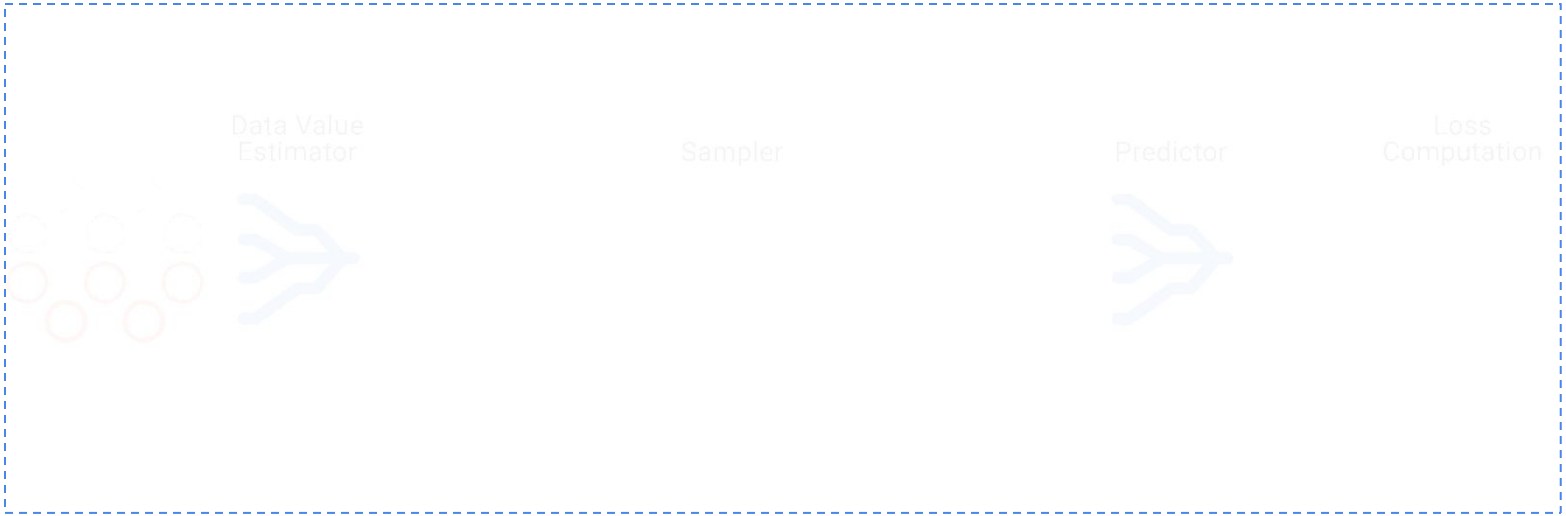


To improve the stability of the training

- Not Differentiable
- Use REINFORCE

- Differentiable
- Use Standard Stochastic Gradient Descent

Architecture



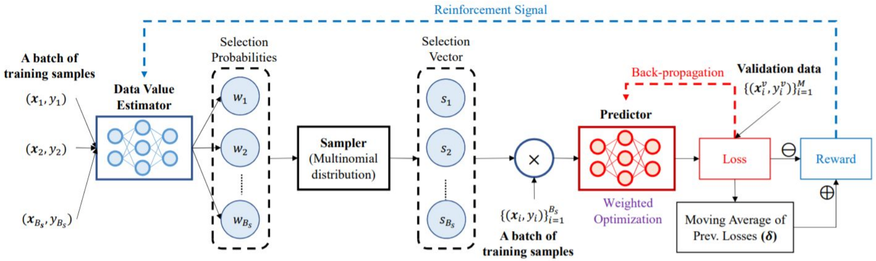
Architecture

$f_\theta \longrightarrow$ Predictor Model

$h_\phi \longrightarrow$ Data Value Estimator

Predictor Model

$$f_\theta = \arg\min_{\hat{f}\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} h_\phi(\mathbf{x}_i, y_i)\cdot \mathcal{L}_f(\hat{f}(\mathbf{x}_i), y_i)$$



Overall Optimization

$$\begin{aligned} \min_{h_\phi} \quad & \mathbb{E}_{(\mathbf{x}^v, y^v)\sim P^t}\left[\mathcal{L}_h(f_\theta(\mathbf{x}^v), y^v)\right] \\ \text{s.t.} \quad & f_\theta = \arg\min_{\hat{f}\in\mathcal{F}} \mathbb{E}_{(\mathbf{x},y)\sim P}\left[h_\phi(\mathbf{x}, y)\cdot \mathcal{L}_f(\hat{f}(\mathbf{x}), y)\right] \end{aligned}$$

Architecture

REINFORCE

$$\hat{l}(\phi) = \mathbb{E}_{(\mathbf{x}^v, y^v)\sim P^t}\left[\mathbb{E}_{\mathbf{s}\sim\pi_\phi(\mathcal{D},\cdot)}\left[\mathcal{L}_h(f_\theta(\mathbf{x}^v), y^v)\right]\right]$$
$$= \int P^t(\mathbf{x}^v)\Big[\sum_{\mathbf{s}\in[0,1]^N} \pi_\phi(\mathcal{D}, \mathbf{s}) \cdot \left[\mathcal{L}_h(f_\theta(\mathbf{x}^v), y^v)\right]\Big] d\mathbf{x}^v$$

where

$$\pi_\phi(\mathcal{D}, \mathbf{s}) = \prod_{i=1}^{N}\left[h_\phi(\mathbf{x}_i, y_i)^{s_i} \cdot (1 - h_\phi(\mathbf{x}_i, y_i))^{1-s_i}\right]$$

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3-4):229–256, 1992. [Link]

Architecture

REINFORCE

we directly compute the gradient $\nabla_\phi \hat{l}(\phi)$ as:

$$\begin{aligned}
\nabla_\phi \hat{l}(\phi) &= \int P^t(\mathbf{x}^v)\Big[\sum_{\mathbf{s}\in[0,1]^N} \nabla_\phi \pi_\phi(\mathcal{D},\mathbf{s}) \cdot \big[\mathcal{L}_h(f_\theta(\mathbf{x}^v), y^v)\big]\Big] d\mathbf{x}^v \\
&= \int P^t(\mathbf{x}^v)\Big[\sum_{\mathbf{s}\in[0,1]^N} \nabla_\phi \log(\pi_\phi(\mathcal{D},\mathbf{s})) \cdot \pi_\phi(\mathcal{D},\mathbf{s}) \cdot \big[\mathcal{L}_h(f_\theta(\mathbf{x}^v), y^v)\big]\Big] d\mathbf{x}^v \\
&= \mathbb{E}_{(\mathbf{x}^v, y^v)\sim P^t}\Big[\mathbb{E}_{\mathbf{s}\sim\pi_\phi(\mathcal{D},\cdot)}\big[\mathcal{L}_h(f_\theta(\mathbf{x}^v), y^v)\big]\nabla_\phi \log(\pi_\phi(\mathcal{D},\mathbf{s}))\Big],
\end{aligned}$$

where $\nabla_\phi \log(\pi_\phi(\mathcal{D},\mathbf{s}))$ is

$$\begin{aligned}
\nabla_\phi \log(\pi_\phi(\mathcal{D},\mathbf{s})) &= \nabla_\phi \sum_{i=1}^N \log\Big[h_\phi(\mathbf{x}_i, y_i)^{s_i} \cdot (1 - h_\phi(\mathbf{x}_i, y_i))^{1-s_i}\Big] \\
&= \sum_{i=1}^N s_i \nabla_\phi \log\big[h_\phi(\mathbf{x}_i, y_i)\big] + (1 - s_i)\nabla_\phi \log\big[(1 - h_\phi(\mathbf{x}_i, y_i))\big].
\end{aligned}$$

Algorithm 1 Pseudo-code of DVRL training

Inputs: Learning rates $\alpha, \beta > 0$, mini-batch size $B_p, B_s > 0$, inner iteration count $N_I > 0$, moving average window $T > 0$, training dataset $\mathcal{D}$, validation dataset $\mathcal{D}^v = \{(\mathbf{x}_k^v, y_k^v)\}_{k=1}^L$

Initialize parameters θ, ϕ, moving average $\delta = 0$

while until convergence **do**

  Sample a mini-batch from the entire training dataset: $\mathcal{D}_B = (\mathbf{x}_j, y_j)_{j=1}^{B_s} \sim \mathcal{D}$

  for $j = 1, ..., B_s$ **do**

    Calculate selection probabilities: $w_j = h_\phi(\mathbf{x}_j, y_j)$

    Sample selection vector: $s_j \sim Ber(w_j)$

  for $t = 1, ..., N_I$ **do**

    Sample a mini-batch $(\tilde{\mathbf{x}}_m, \tilde{y}_m, \tilde{s}_m)_{m=1}^{B_p} \sim (\mathbf{x}_j, y_j, s_j)_{j=1}^{B_s}$

    Update the predictor model network parameters θ

$$\boxed{\theta \leftarrow \theta - \alpha \frac{1}{B_p} \sum_{m=1}^{B_p} \tilde{s}_m \cdot \nabla_\theta \mathcal{L}_f(f_\theta(\tilde{\mathbf{x}}_m), \tilde{y}_m))}$$

  Update the data value estimator model network parameters ϕ

$$\boxed{\phi \leftarrow \phi - \beta \Big[\frac{1}{L} \sum_{k=1}^{L} [\mathcal{L}_h(f_\theta(\mathbf{x}_k^v), y_k^v)] - \delta \Big] \nabla_\phi \log \pi_\phi(\mathcal{D}_B, (s_1, ..., s_{B_s}))}$$

  Update the moving average baseline (δ): $\delta \leftarrow \frac{T-1}{T}\delta + \frac{1}{LT} \sum_{k=1}^{L} [\mathcal{L}_h(f_\theta(\mathbf{x}_k^v), y_k^v)]$

Datasets

Datasets	Type	Baseline Predictor Model
Blog, Adult	**Tabular**	**LightGBM**
Rossmann	**Tabular**	**XGBoost and MLP**
MNIST, Fashion-MNIST, USPS	Image	Multinomial Logistic Regression
HAM 10000, Flower, CIFAR-10/100	Image	Inception-v3
Email Spam, SMS Spam	**Language**	**Naive Bayes**

Experiments - Removing High/Low Value Samples

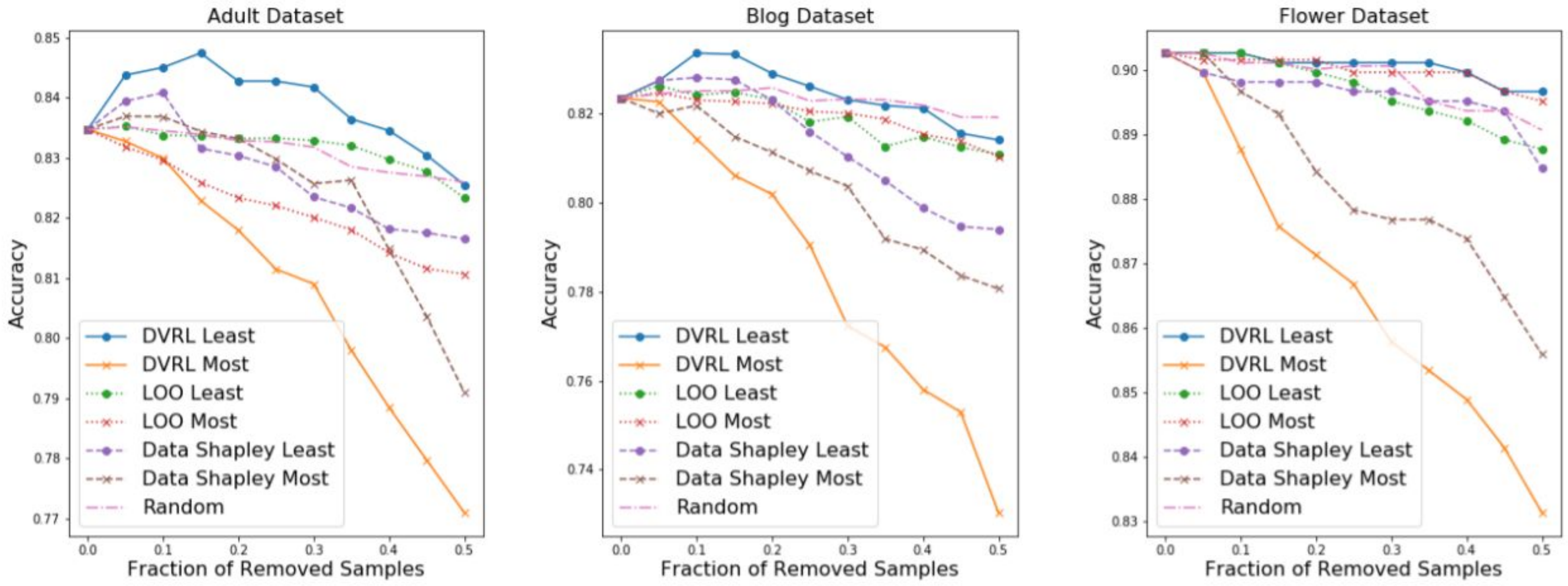


Figure 2: Performance after removing the most and least important samples according to the estimated data values in a conventional supervised learning setting.

Experiments - Removing High/Low Value Samples

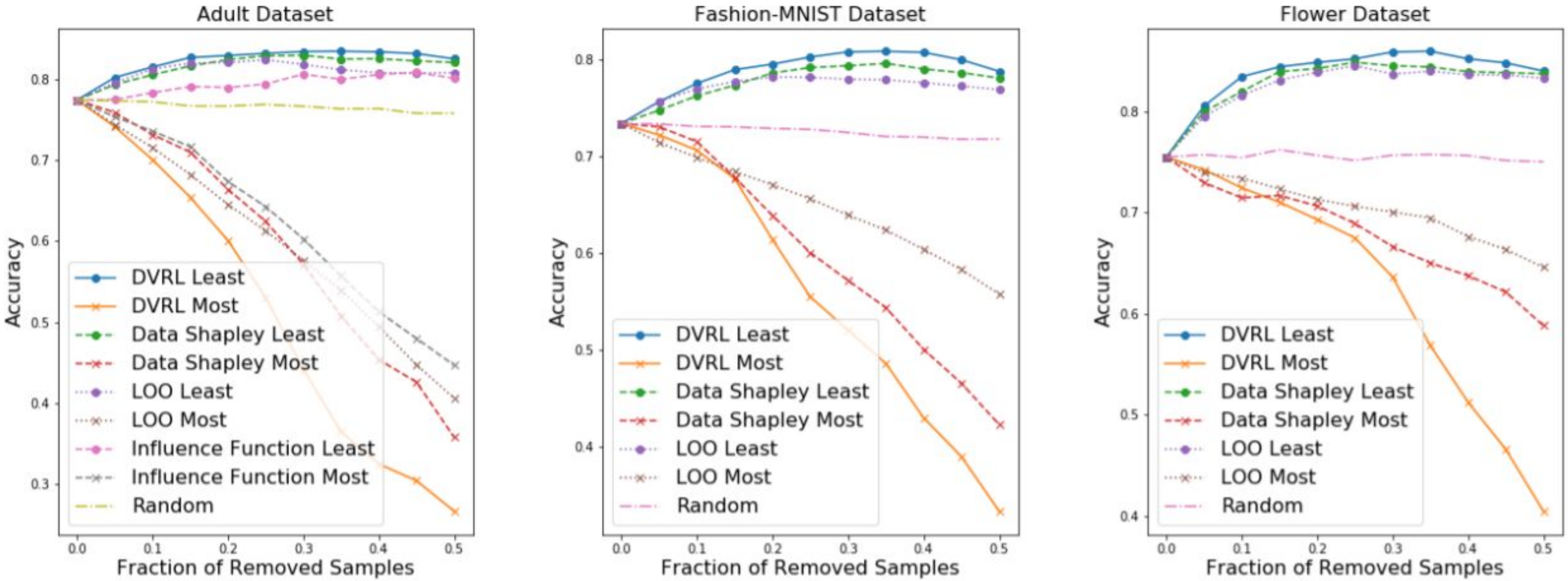


Figure 3: Prediction performance after removing the most and least important samples according to the estimated data values **with 20% noisy label ratio**.

Experiments - Corrupted Sample Discovery

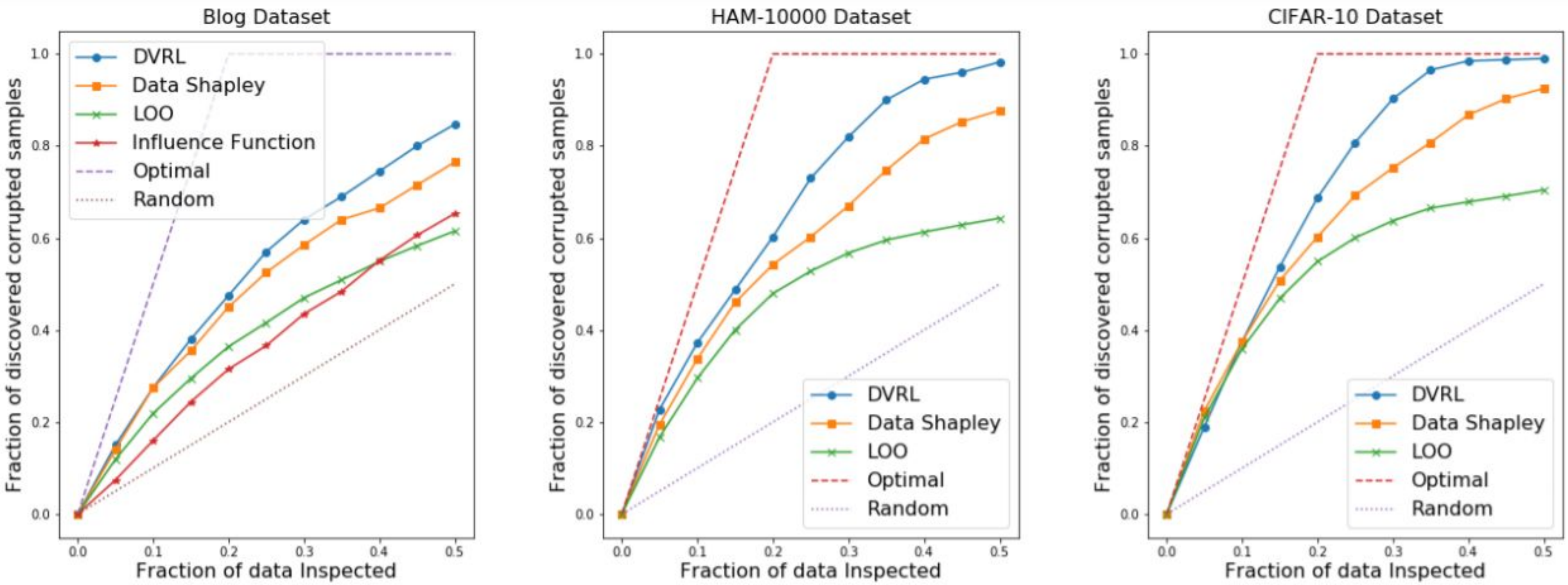


Figure 4: Discovering corrupted samples in three datasets **with 20% noisy label ratio**. ‘Optimal’ saturates at 20%, perfectly assigning the lowest data value scores to the samples with noisy labels.

Experiments - Robust Learning with Noisy Labels

Noise (predictor model)	**Uniform** (WideResNet-28-10)		**Background** (ResNet-32)	
Datasets	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Validation Set Only	46.64 ± 3.90	9.94 ± 0.82	15.90 ± 3.32	8.06 ± 0.76
Baseline	67.97 ± 0.62	50.66 ± 0.24	59.54 ± 2.16	37.82 ± 0.69
Baseline + Fine-tuning	78.66 ± 0.44	54.52 ± 0.40	82.82 ± 0.93	54.23 ± 1.75
MentorNet + Fine-tuning	78.00	59.00	-	-
Learning to Reweight	86.92 ± 0.19	61.34 ± 2.06	86.73 ± 0.48	59.30 ± 0.60
DVRL	$\mathbf{89.02 \pm 0.27}$	$\mathbf{66.56 \pm 1.27}$	$\mathbf{88.07 \pm 0.35}$	$\mathbf{60.77 \pm 0.57}$
Clean Only (60% Data)	94.08 ± 0.23	74.55 ± 0.53	90.66 ± 0.27	63.50 ± 0.33
Zero Noise	95.78 ± 0.21	78.32 ± 0.45	92.68 ± 0.22	68.12 ± 0.21

Table 1: Robust learning with noisy labels. Test accuracy for ResNet-32 and WideResNet-28-10 on CIFAR-10 and CIFAR-100 datasets with 40% of Uniform and Background noise on labels.

Experiments - Domain Adaptation

Source	Target	Task	*Baseline*	Data Shapley	**DVRL**
Google	HAM10000	Skin Lesion Classification	.296	.378	**.448**
MNIST	USPS	Digit Recognition	.308	.391	**.472**
Email	SMS	Spam Detection	.684	.864	**.903**

Table 2: **Domain adaptation** setting showing target accuracy. Baseline represents the predictor model which is naively trained on the training set with equal treatment of all training samples.

Experiments - Domain Adaptation

Predictor Model (Metric: RMSPE)	Store Type	*Train on All*		*Train on Rest*		*Train on Specific*	
		Baseline	DVRL	*Baseline*	DVRL	*Baseline*	DVRL
XGBoost	A	0.1736	**0.1594**	0.2369	**0.2109**	0.1454	**0.1430**
	B	0.1996	**0.1422**	0.7716	**0.3607**	0.0880	**0.0824**
	C	0.1839	**0.1502**	0.2083	**0.1551**	0.1186	**0.1170**
	D	0.1504	**0.1441**	0.1922	**0.1535**	0.1349	**0.1221**
Neural Networks	A	0.1531	**0.1428**	0.3124	**0.2014**	0.1181	**0.1066**
	B	0.1529	**0.0979**	0.8072	**0.5461**	0.0683	**0.0682**
	C	0.1620	**0.1437**	0.2153	**0.1804**	0.0682	**0.0677**
	D	0.1459	**0.1295**	0.2625	**0.1624**	0.0759	**0.0708**

Table 3: Performance of Baseline and DVRL in 3 different settings with 2 different predictor models on the Rossmann Store Sales dataset. Metric is Root Mean Squared Percentage Error (RMSPE, lower the better). We use 79% of the data as training, 1% as validation, and 20% as testing. DVRL outperforms Baseline in all settings.

Experiments - How many validation samples are needed?

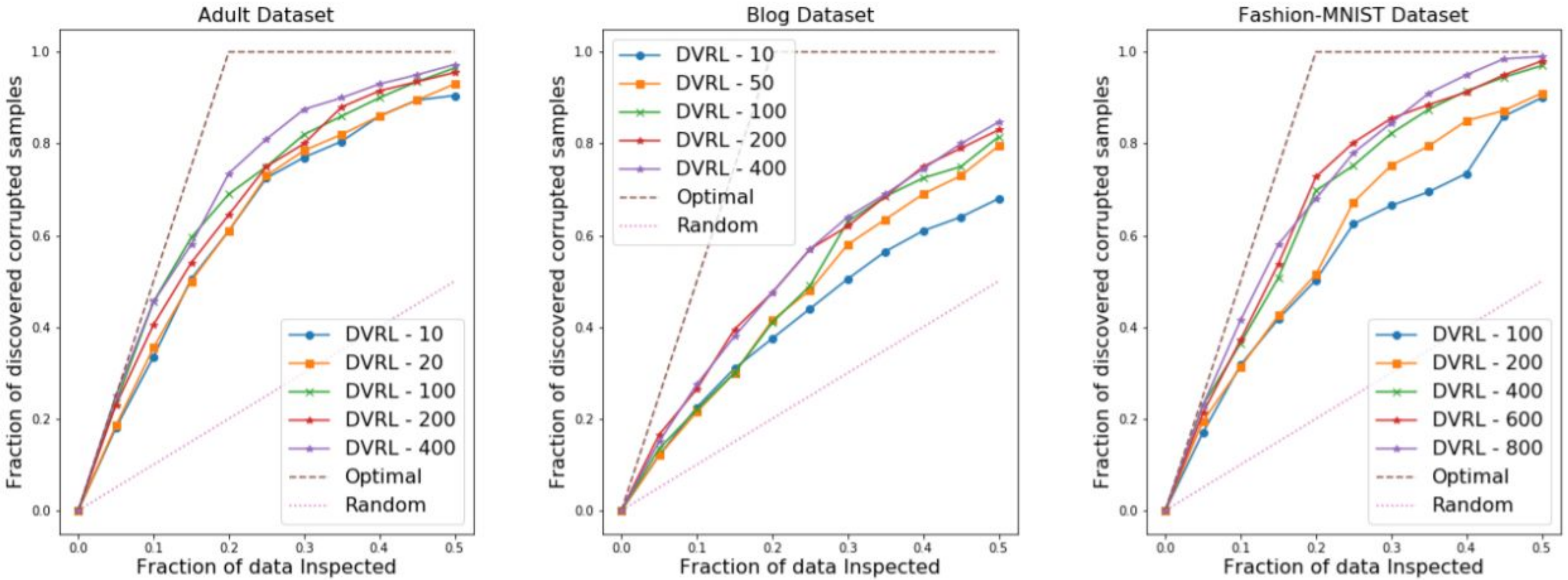


Figure 5: Number of validation samples needed for DVRL. Discovering corrupted samples in three datasets (Adult, Blog and Fashion MNIST) with various number of validation samples.

The End