模式分析与机器智能
工业和信息化部重点实验室
MIIT Key Laboratory of
Pattern Analysis & Machine Intelligence

ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

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

ICML 2020

Outline

- Background
  - REINFORCE algorithm
  - Problem setting
- The method
- Experiments
  - Removing high/low value samples
  - Noisy setting
  - Transfer setting
- Conclusions

REINFORCE

- The REINFORCE algorithm is a policy gradient method, it directly update the policy weights, without learning a value function.

$$E_{\tau\sim p(\tau)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right]$$

- τ is the trajectory
- π is the policy model parameterized by θ

$$\nabla_\theta E_{\tau\sim p(\tau)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right]$$

The likelihood ratio trick

$$= \int \nabla_\theta p(\tau)\left(\sum_{t=1}^{T} r(s_t, a_t)\right) d\tau$$

$$= \int p(\tau)\nabla_\theta \log p(\tau)\left(\sum_{t=1}^{T} r(s_t, a_t)\right) d\tau$$

$$= E_{\tau\sim p(\tau)}\left[\nabla_\theta \log p(\tau) \sum_{t=1}^{T} r(s_t, a_t)\right]$$

$$\nabla_\theta \log p(\tau) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

- Quantifying the **VALUE** of data is important.

1. Incorrect label (e.g. human labeling errors).
2. Input comes from a different distribution (e.g. different location or time).
3. Input is noisy or low quality (e.g. noisy capturing hardware).
4. Usefulness for target task

Background

- Related works:

1. Leave-One-Out

- Computational expensive
- Fail when there are two equal data

2. Data Shapley

- Computational expensive
- Less effective with Monte Carlo approximation

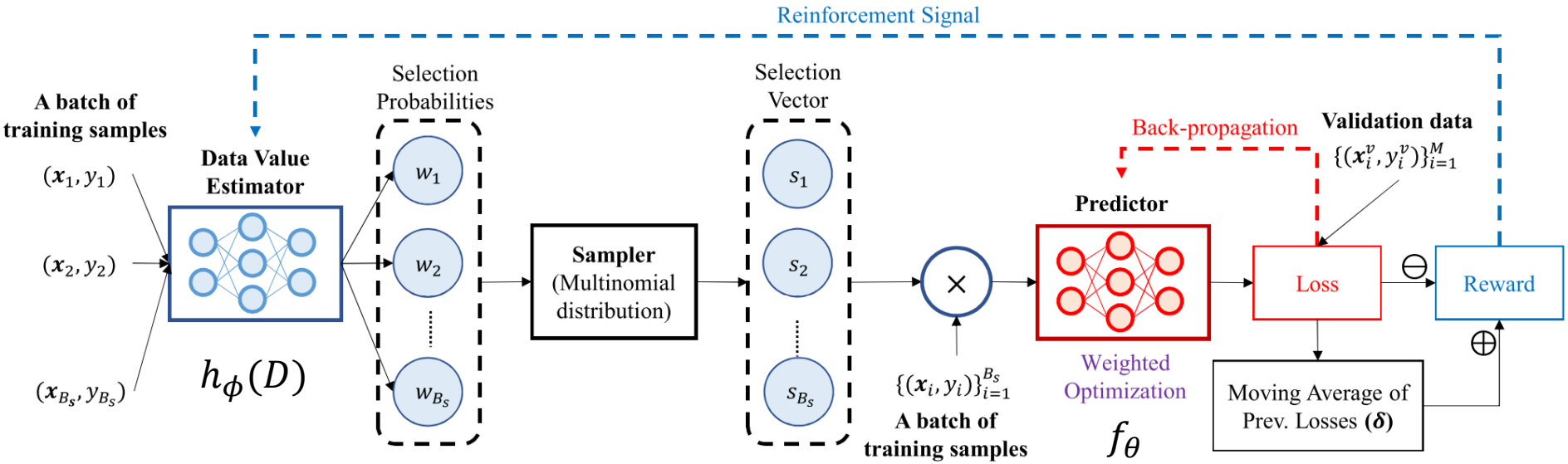
3. Meta-Learning Based Methods

- Model dependent
- Decoupled from predictor model training

1. Meta-Weight-Net
2. Learning to Reweight
3. MentorNet

The method

- Overall framework



$w_i \in [0,1]$ is the importance score

$s_i \in \{0,1\}$ is the selection vector

$\boldsymbol{s}$ is sampled from $\pi_\phi(D,s) = \prod_{i=1}^{N} h_\phi(x_i,y_i)^{s_i} \cdot (1 - h_\phi(x_i,y_i))^{1-s_i}$

The method

- Overall framework

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 $\delta = 0$

Outer loop

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: $s_j \sim Ber(w_j)$

h_ϕ predict scores

Inner loop

  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

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

Update prediction model with sampled data

  Update the data value estimator model network parameters ϕ

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

Update policy model h_ϕ with validation set

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

Update policy

- Target of the policy model

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

- Apply the REINFORCE method

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

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

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

The likelihood ratio trick

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

Update policy

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

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

Experiments

- **Compared methods**
  - Random assigned weights
  - Leave-One-Out (LOO)
  - Data Shapley Value (ICML'19)
  - Learning to Reweight (ICML'18)
  - MentorNet (ICML'18)
  - Influence Function (ICML'17)
- **Datasets**
  - Blog
  - Adult
  - Rossmann
  - HAM 10000
  - MNIST
  - USPS
  - Flower
  - Fashion-MNIST
  - CIFAR-10
  - CIFAR-100
  - Email Spam
  - SMS Spam
- **Settings**
  - Removing high/low value samples
  - Noisy setting
  - Transfer setting

项目代码：https://github.com/google-research/ google-research/tree/master/dvrl

Removing samples

- Clean data

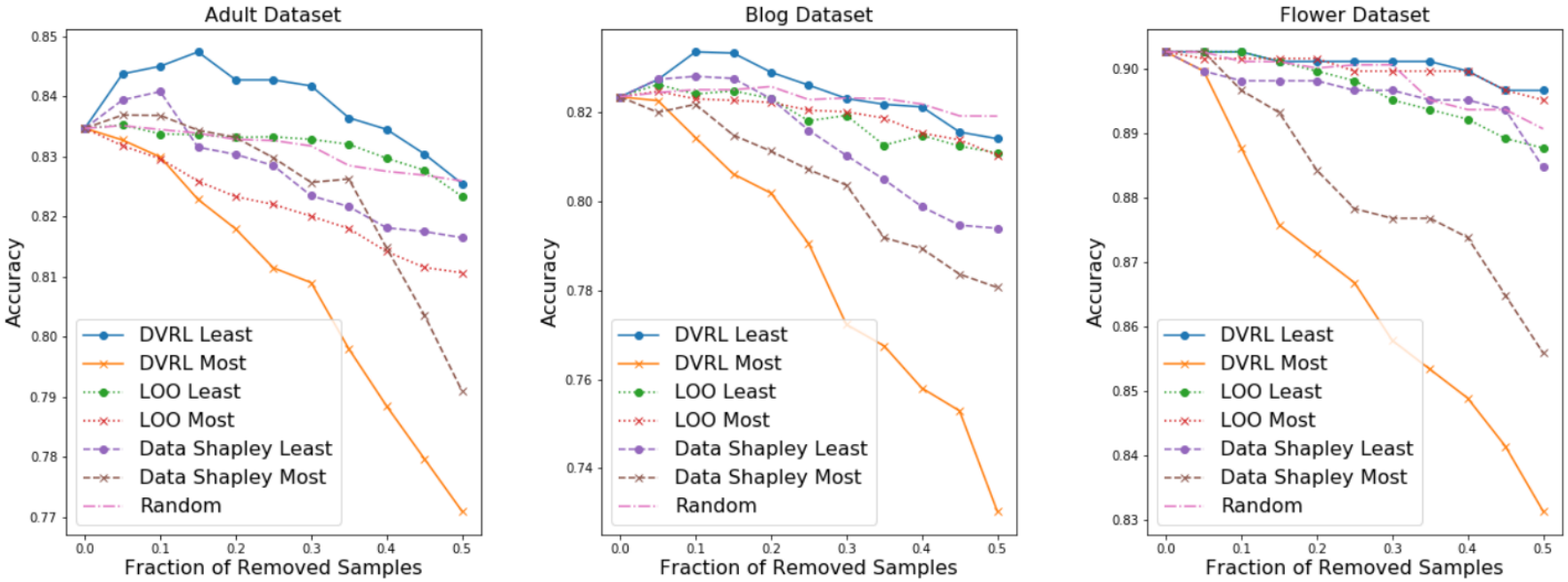
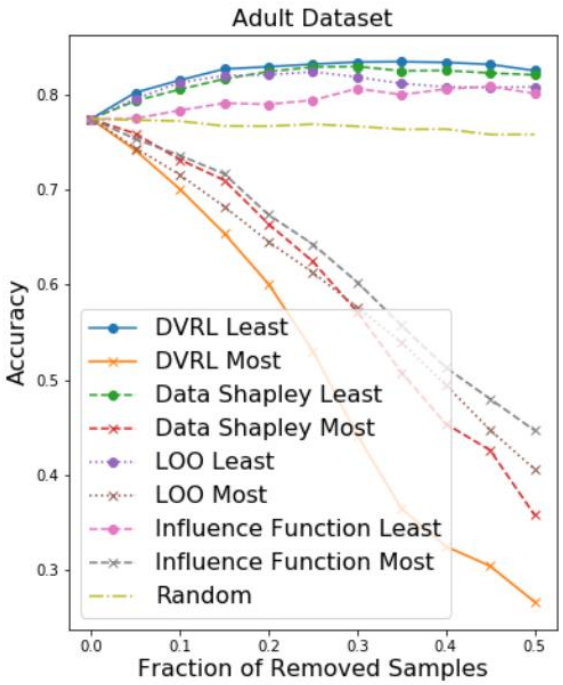
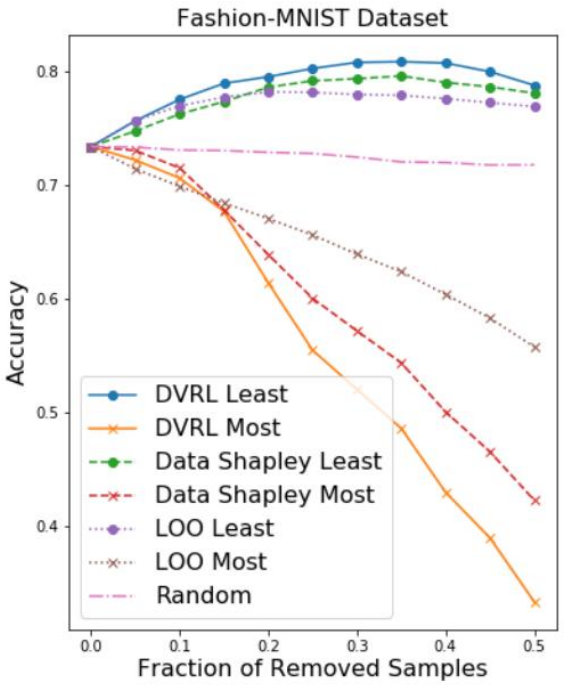


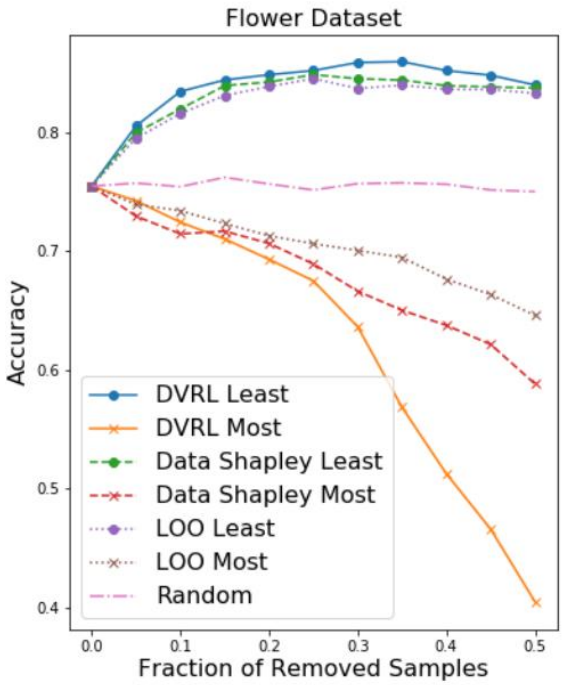
Figure 2: Performance after removing the most (marked with ◯) and least (marked with ×) important samples according to the estimated data values in a conventional supervised learning setting.

Removing samples

- Noisy data (20%)



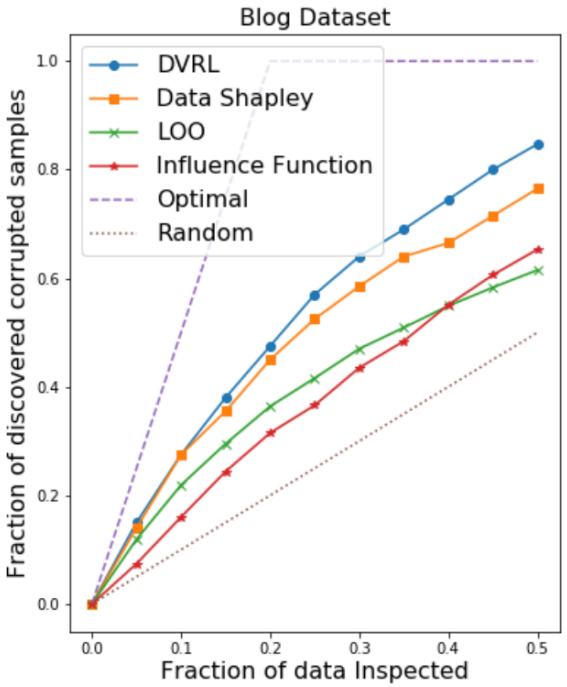


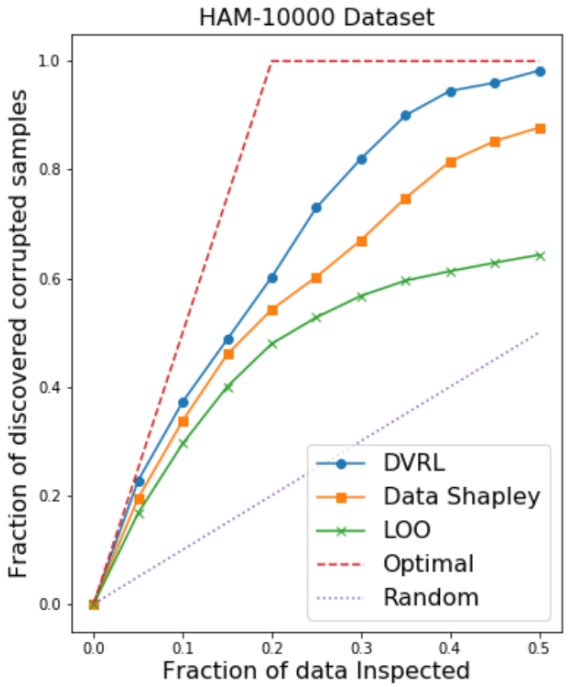


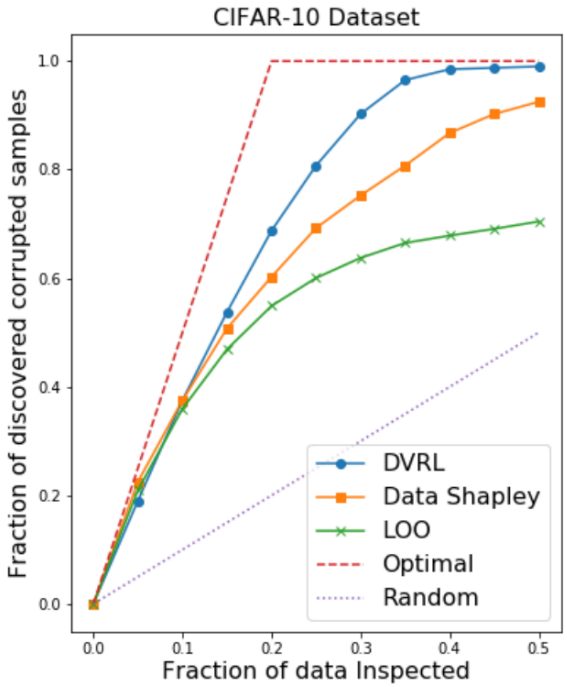
Note that Influence Function is a computationally-efficient approximation of LOO; thus, the performance of Influence Function is similar or slightly worse than LOO in most cases

Noise discovering

- Noisy data (20%)







X-axis: The fraction of data with the lowest scores

Y-axis: How many noisy examples are discovered

Transfer setting

- **Learning from a different domain**

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**

Predictor Model	Store	*Train on All*		*Train on Rest*		*Train on Specific*	
(Metric: RMSPE)	Type	*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**

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

Size of validation set

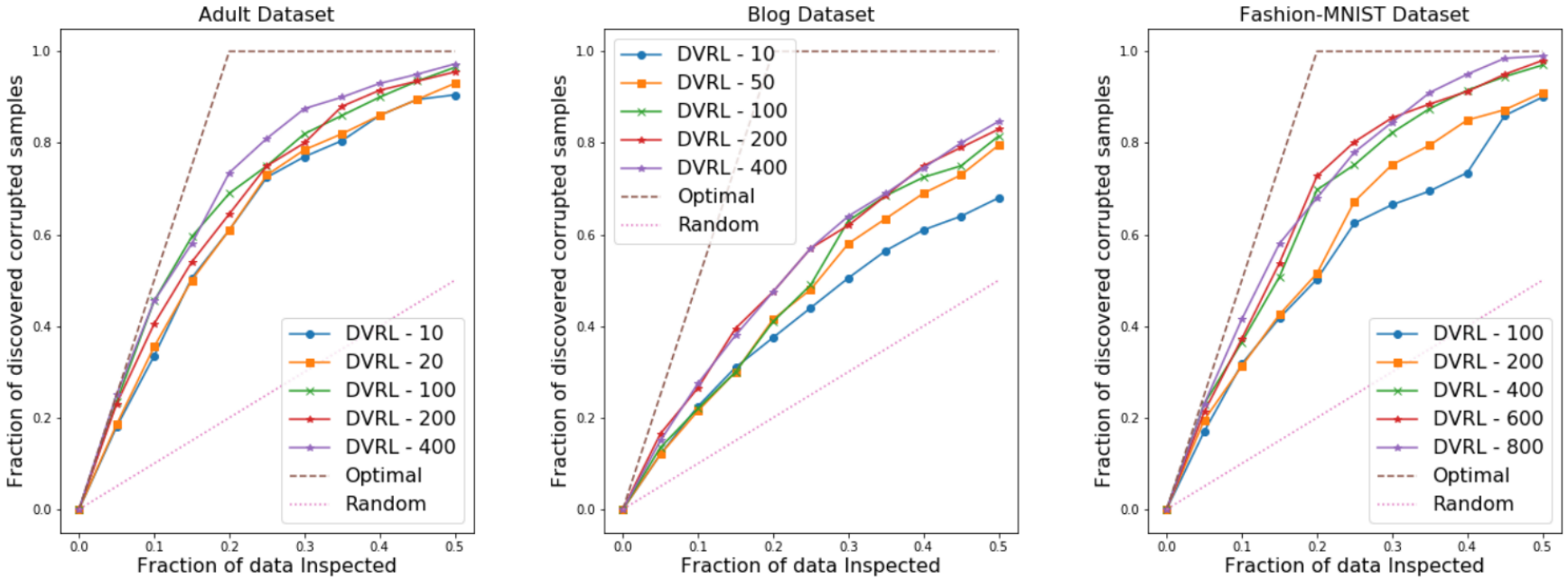


Figure 5: Number of validation samples needed for DVRL. Discovering corrupted samples in three datasets (Adult, Blog and Fashion MNIST) with various number of validation samples.

DVRL achieves reasonable performance with 100 to 400 validation samples. In the Adult dataset, even 10 validation samples are sufficient

Conclusion

- This work propose a meta learning framework, named DVRL, that adaptively learns data values jointly with a target task predictor model.

- DVRL uses REINFORCE method to update the policy model, and validate the effectiveness.

- DVRL is model agnostic, and even applicable to non-differentiable target objectives.