Iterative Teaching by Label Synthesis

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we consider the problem of iterative machine teaching, where a teacher provides examples sequentially based on the current iterative learner. In contrast to previous methods that have to scan over the entire pool and select teaching examples from it in each iteration, we propose a label synthesis teaching framework where the teacher randomly selects input teaching examples (*e.g.*, images) and then synthesizes suitable outputs (*e.g.*, labels) for them. We show that this framework can avoid costly example selection while still provably achieving exponential teachability. We consider both omniscient and black-box teaching in this framework. Finally, we empirically demonstrate the value of our framework.

1 Introduction

Machine teaching [70, 73] studies the problem of constructing a minimal dataset for a target concept such that a learner can learn the concept based on this dataset. Machine teaching has diverse applications ranging from crowd sourcing [50, 49, 67, 68] to model robustness [2, 3, 45, 39]. Machine teaching also has nice connections with curriculum learning [5] and coresets [1, 20].

Based on the learner, machine teaching can be performed in either batch or sequential fashion. The majority of prior work studies batch machine teaching [69, 70, 32, 41], where a teacher constructs a minimal batch set of training samples and provides it to a student in one shot without further interactions. Then the student keeps learning from this batch dataset for the target concept. The size of such a minimal batch set is called *teaching dimension* [15]. Differently, sequential machine teaching [33, 34, 29, 44] bridges the gap between machine teaching and practical learning algorithms by studying the sequential (*i.e.*, iterative) learner such as prevailing neural networks. A typical example is iterative machine teaching (IMT) [33, 34] where the teacher guides a learner to a target concept by interacting with the learner (*e.g.*, feeding training samples) in every iteration. The minimum number of such iterations is called the *iterative teaching dimension*. One of the largest challenges in sequential teaching is how to effectively and efficiently provide teaching examples to the iterative learner. Usually we are mostly interested in the pool-based teaching in IMT since it well matches the setting of modern machine learning. However, exploring all the possible teaching trajectories is computationally prohibitive. For example, there are $\binom{m}{k}^n$ possible teaching trajectories (n is the number of iterations) if we select k samples per iteration from a pool of size m. Due to such a huge search space, the selection of teaching examples is a combinatorial problem that is inherently difficult to solve. IMT [33] performs the teaching sample selection with a greedy policy, but it could be substantially sub-optimal in certain cases [29] and its computational complexity scales linearly with the dataset size.

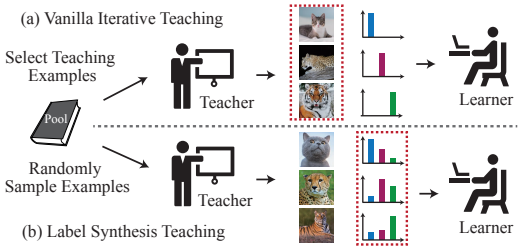


Figure 1: Comparison of vanilla iterative machine teaching and label synthesis teaching. The red dotted frames indicate the teacher's efforts.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

We propose a general teaching framework called LAbel Synthesis Teaching (LAST), which in its standard version avoids the problems of sample selection – though we will later discuss how sample selection can be combined with our approach. In the standard version of LAST, teaching examples are randomly sampled (similar to SGD) in the pool and the teacher synthesizes their labels in order to quickly guide the learner to the desired model. A brief comparison between IMT and LAST is given in Fig. 1. LAST restricts the teaching to the label space and bypasses the selection of teaching samples. Therefore, LAST avoids the high complexity of selecting teaching samples in a large pool.

Intuition for why LAST is able to achieve promising teaching performance comes from the empirical success of knowledge distillation [19] and label smoothing [51]. Knowledge distillation shows that training neural networks with soft labels from a pretrained model can generally improve generalization. Label smoothing demonstrates that the ground truth labels are not necessarily the optimal supervision for training a model. Instead, smoothing the label with uniform distribution can calibrate the model and lead to better generalization. Both methods can be viewed as providing an alternative label (instead of the ground truth) to the learner in order to improve its generalizability. Moreover, [55, 54] show that there exists privileged information beyond the ground truth labels that can significantly improve the convergence rate of learning algorithms. Therefore, now we can safely argue that the ground truth labels are not always optimal learning signals. Motivated by these work, we aim to construct a teacher model that can adaptively synthesize suitable labels for the learner (with the hope to implicitly encode priviledged information) in order to improve its convergence and generalization.

Specifically, we study LAST under two scenarios: omniscient teaching where the teacher knows everything about the learner (*e.g.*., the optimal learner parameters), and black-box teaching where the teacher knows nothing about the learner (*e.g.*., the optimal learner parameters are unavailable). We consider greedy teaching and parameterized teaching in both scenarios. In omniscient teaching, we show that greedy teaching can achieve exponential teachability (ET) [33] without selecting teaching examples. In black-box teaching, we propose two methods (gradient unrolling and reinforcement learning) to learn a parameterized teaching policy without prior knowledge of the optimal leaner.

More importantly, LAST provides a unified view for understanding knowledge distillation [19, 38], label smoothing [51, 8, 43], and self-training [71]. All these methods can be interpreted as modifying the labels to achieve desirable learning behavior and outcome. Under the umbrella of LAST, we can connect iterative machine teaching to many classic learning algorithms and shed novel light on them.

2 Related Work

Batch machine teaching [69, 70, 32, 41] has been widely studied in recent years. Particularly, different learners exhibit different behaviors during teaching. [7, 52] focus on teaching version space learners. The teaching dimension of linear learners [32], kernel learners [27] and reinforcement learner [63] is extensively studied. Teaching a forgetful learner is also explored in [21, 34]. [72, 62] consider how to teach multiple learners simultaneously. Sequential machine teaching [33, 34, 29] studies iterative learners by considering the specific optimization algorithm that the learner adopts. Unlike batch scenario, the teaching quality is evaluated by the learner's convergence. [29] connects sequential teaching to optimal control and gains interesting insights, but it can not produce a practical teaching policy. Machine teaching is shown useful in reinforcement learning [53, 24, 45, 17], human-in-the-loop learning [23, 6, 40], crowd sourcing [50, 67, 68] and cyber security [42, 2, 66, 64, 65]. [10, 14, 74, 47, 6, 26] study machine teaching from a more theoretical aspect.

3 Label Synthesis Teaching

Teaching protocol. We first consider the cleanest teaching protocol following [33]. Both the teacher and the learner observe the same sample $\mathcal{A}$ and share the same feature space by representing $\mathcal{A}$ as $\boldsymbol{x}$ with label $\boldsymbol{y}$. The teacher has access to all the information about the learner such as the model parameter $\boldsymbol{w}^i$ at the i-th iteration, the learning rate η_i, the loss function ℓ and the specific optimization algorithm (*e.g.*, SGD). For interaction, the teacher can only communicate with the learner via examples $\{(\boldsymbol{x}_j^i, \boldsymbol{y}_j^i)\}_{j=1,\cdots,m}$ in the i-th iteration, where m denotes the size of the mini-batch.

Teaching objective. The goal of the teacher is to provide examples in each iteration such that the learner parameters $\boldsymbol{w}$ converge to desired parameters $\boldsymbol{w}^*$ as fast as possible. For example, $\boldsymbol{w}^*$ typically is $\arg\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\{\ell(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{w})\}$. Therefore, a general objective for any teaching algorithm is $\min_{\{(\boldsymbol{x}^1,\boldsymbol{y}^1),\cdots,(\boldsymbol{x}^T,\boldsymbol{y}^T)\}} d(\boldsymbol{w}^T, \boldsymbol{w}^*)$ where T denotes the termination iteration for the teaching algorithm. We use batch size 1 here, but it is straightforward to extend the batch size to arbitrary m. $d(\cdot,\cdot)$ denotes some discrepancy measure (*e.g.*, Euclidean distance). This objective aims to find a

teaching trajectory that is terminated at the T-th step such that the discrepancy between the current learner parameters and the desired learner parameters is the smallest. This is very challenging given the enormous search space. To simplify the problem, [33] proposes a greedy policy to generate the teaching curriculum, but it is still computationally expensive when dealing with a large pool.

3.1 The LAST Framework

Unlike the general teaching objective that optimizes the sequential sample selection, the LAST framework instead optimizes the teaching examples in the label space, and the corresponding objective becomes $\min_{\{\boldsymbol{y}^1,\cdots,\boldsymbol{y}^T\}} d(\boldsymbol{w}^T, \boldsymbol{w}^*)$ where $\boldsymbol{x}^1, \cdots, \boldsymbol{x}^T$ are sampled uniformly from the pool (same as SGD). Such a teaching objective is general and does not make any assumption on the optimal structure in the teaching trajectory, making the problem intractable. To simplify the problem, we can assume that the teaching signal (*i.e.*, label $\boldsymbol{y}$ in LAST) only depends on the randomly sampled $\{\boldsymbol{x}, \tilde{\boldsymbol{y}}\}$ and the learner's current and desired model (*i.e.*, parameters). Therefore, we can formulate the corresponding label synthesis policy as $\boldsymbol{y} = \pi_{\boldsymbol{\theta}}(\boldsymbol{x}, \tilde{\boldsymbol{y}}, \boldsymbol{w}^t, \boldsymbol{w}^*)$ where π is optionally parameterized by $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{y}}$ denotes the original ground truth label. Then the teaching objective becomes $\arg\min_{\pi_{\boldsymbol{\theta}}} d(\boldsymbol{w}^T, \boldsymbol{w}^*)$. An overview of LAST is given in Fig. 2. Moreover, LAST provides a unified framework for knowledge distillation, label smoothing and self-training. Knowledge distillation [19] generates the label with the policy $\boldsymbol{y}_i = \frac{\exp(\boldsymbol{z}_i(\boldsymbol{x})/\psi)}{\sum_j \exp(\boldsymbol{z}_j(\boldsymbol{x})/\psi)}$ where ψ is the temperature and $\boldsymbol{z}_i(\boldsymbol{x})$ is the i-th logit output of $\boldsymbol{x}$ in a pretrained neural network. Label smoothing [51] uses $\boldsymbol{y} = \mu\tilde{\boldsymbol{y}} + \frac{1-\mu}{K}\mathbf{1}$ (K is the number of classes) as the label synthesis policy. Self-training [71] feeds the learner with the pseudo-label predicted by the learner from the last iteration. All these methods can be viewed as using a customized label synthesis policy in the black-box teaching scenario.

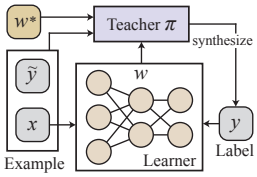


Figure 2: Overview.

3.2 Greedy Teaching in LAST

By considering $T=1$ and gradient descent in LAST, we obtain a greedy policy $\arg\min_{\boldsymbol{y}} d(\boldsymbol{w}^1, \boldsymbol{w}^*)$ where $\boldsymbol{w}^1$ is one-step update from the initial learner $\boldsymbol{w}^0$ *w.r.t.* the learner loss ℓ and example $\{\boldsymbol{x}, \boldsymbol{y}\}$. The greedy policy minimizes the Euclidean distance between the current learner parameters and the desired parameters in each iteration by learning a suitable label. We apply the greedy policy recursively to every iteration, and the label for the input example $\boldsymbol{x}$ at the t-th iteration is produced by

$$\min_{\boldsymbol{y}} \underbrace{\left\{\left\|\boldsymbol{w}^t - \eta_t \frac{\partial \ell(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t} - \boldsymbol{w}^*\right\|_2^2\right\}}_{G(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{w}^t)} \Rightarrow \min_{\boldsymbol{y}} \left\{\eta_t^2 \underbrace{\left\|\frac{\partial \ell(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t}\right\|_2^2}_{T_1(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{w}^t)} - 2\eta_t \underbrace{\langle \boldsymbol{w}^t - \boldsymbol{w}^*, \frac{\partial \ell(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t}\rangle}_{T_2(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{w}^t)}\right\} \quad (1)$$

where $\ell(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{w}^t)$ denotes the loss function of the learner and $\boldsymbol{w}^t$ is the learner parameter in the t-th iteration. For simplicity, we consider the mini-batch size as 1. It is straightforward to write Eq. (1) in a mini-batch form. According to the decomposition of the parameter discrepancy, the teacher optimizes the labels such that the parameter discrepancy can be minimized. In contrast to IMT, which aims to optimize the sample $\boldsymbol{x}$, the greedy teaching in LAST first randomly samples the teaching example and then optimize the label based on $\arg\min_{\boldsymbol{y}} \eta_t^2 T_1(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{w}^t) - 2\eta_t T_2(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{w}^t)$. The procedure is given in Algorithm 1. Depending on the teacher's capability, we can optionally consider a few constraints for the synthesized labels.

Algorithm 1 Omniscient Greedy LAST

Initialize $t=1$, $\boldsymbol{w}^0$, ϵ and T;
while $\|\boldsymbol{w}^t - \boldsymbol{w}^*\|_2 \geq \epsilon$ or $t \leq T$ **do**
  Randomly select a sample $\boldsymbol{x}^t$ from the pool;
  Solve Eq. (1) to synthesize the label $\boldsymbol{y}^t$:
  Use the synthesized label $\boldsymbol{y}^t$ for the update:
  $$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \eta_t \frac{\partial \ell(\boldsymbol{x}^t, \boldsymbol{y}^t|\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t};$$
  Set $t \leftarrow t+1$;
end

One-hot constraint. For classification tasks, we can constrain the teacher to synthesize one-hot labels for $\boldsymbol{x}$. With the one-hot constraint, label synthesis becomes an integer optimization [60].

Soft constraint. A relaxation of the one-hot constraint is to enforce the continuous soft labels, *i.e.*, $\sum_i \boldsymbol{y}_i = 1$ and $\boldsymbol{y}_i \geq 0, \forall i$. Then label synthesis becomes a constrained optimization.

Magnitude constraint. For regression, we can constrain the p-norm of the label within a constant magnitude r, *i.e.*, $\|\boldsymbol{y} - \langle \boldsymbol{w}^t, \boldsymbol{x}\rangle\|_p \leq r$. Larger r indicates a more flexible and capable teacher.

3.2.1 Teaching Linear Learners

We start by discussing how to use greedy LAST to teach representative linear learners such as linear least square regression (LSR) and logistic regression (LR), and then give theoretical analyses on their iterative teaching dimension (*i.e.*, convergence properties). Specifically, we consider the following

standard objective function: $\ell_{\text{LSR}}(\boldsymbol{x}, y|\boldsymbol{w}) = \frac{1}{2}(\langle \boldsymbol{w}, \boldsymbol{x}\rangle - y)^2$ for the LSR learner and $\ell_{\text{LR}}(\boldsymbol{x}, y|\boldsymbol{w}) = \log(1+\exp\{-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle\})$ for the LR learner. $(\boldsymbol{x}_{[i]}, y_{[i]})$ is the i-th sample in the pool. We then discuss what the synthesized label looks like. By taking LSR as an example, we have that Eq (1) becomes

$$\min_{y} \eta_t^2 \langle \boldsymbol{x}, \boldsymbol{x}\rangle y^2 + \big(2\eta_t \langle \boldsymbol{w} - \boldsymbol{w}^*, \boldsymbol{x}\rangle - 2\eta_t^2 \langle \boldsymbol{x}, \boldsymbol{x}\rangle \langle \boldsymbol{w}, \boldsymbol{x}\rangle\big) y + \eta_t^2 \langle \boldsymbol{x}, \boldsymbol{x}\rangle \langle \boldsymbol{w}, \boldsymbol{x}\rangle - 2\eta_t \langle \boldsymbol{w}, \boldsymbol{x}\rangle \langle \boldsymbol{w} - \boldsymbol{w}^*, \boldsymbol{x}\rangle \quad (2)$$

from which we can obtain the closed-form solution for the synthesized label as

$$y^* = \underbrace{\frac{2\eta_t^2 \langle \boldsymbol{x}, \boldsymbol{x}\rangle \langle \boldsymbol{w}, \boldsymbol{x}\rangle - 2\eta_t \langle \boldsymbol{w} - \boldsymbol{w}^*, \boldsymbol{x}\rangle}{2\eta_t^2 \langle \boldsymbol{x}, \boldsymbol{x}\rangle}}_{\text{Synthesized label}} = (1-\lambda) \cdot \underbrace{\langle \boldsymbol{w}, \boldsymbol{x}\rangle}_{\text{Predicted label}} + \lambda \cdot \underbrace{\langle \boldsymbol{w}^*, \boldsymbol{x}\rangle}_{\text{Optimal label}} \quad (3)$$

where $\lambda = (\eta_t \langle \boldsymbol{x}, \boldsymbol{x}\rangle)^{-1}$. λ is essentially a parameter that weights the importance between using the current prediction as the training label and using the (pseudo) ground truth label as the training label. Suppose that the input data lies on a hypersphere with radius γ, then we have $\lambda = \frac{1}{\eta_t \gamma^2}$. Typically η_t will gradually decrease during training, so λ will gradually increase. Interestingly, this implies that the training label should be gradually shifted from the current predicted label to the ground truth label during training. This result validates our argument that it is not always optimal to feed the learner with ground truth labels. The predicted label is usually easy to learn for the current learner and the optimal label is more difficult to learn, implying that the learner should be taught in an easy-to-hard fashion. This implication nicely matches the conclusion in curriculum learning [5] and IMT [33, 34].

Moreover, greedy LAST also has the property of teaching monotonicity [33] if the learner loss satisfies certain conditions (Appendix C). This nice property implies that LAST always converges to $\boldsymbol{w}^*$ faster than a random teacher (i.e., SGD) if both of them sample teaching examples uniformly from the pool. An illustration of how LAST works and why LAST yields better convergence to $\boldsymbol{w}^*$ is given in Fig. 3. If the learner loss is properly designed, LAST can converge faster than SGD. The teaching monotonicity comes from the first iteration where the gradient update in LAST is guaranteed to better minimize the discrepancy between $\boldsymbol{w}^1$ and $\boldsymbol{w}^*$.

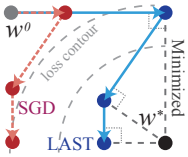


Figure 3: Example.

3.2.2 Teaching Neural Learners

We extend the greedy teaching to neural learners. As an illustrative example, we teach a two-layer multi-layer perceptron (MLP): $\ell_{\text{MLP}} = \text{CrossEntropy}(\text{SoftMax}(\boldsymbol{W}^\top \sigma(\boldsymbol{V}^\top \boldsymbol{x})), \boldsymbol{y})$ where $\boldsymbol{V} \in \mathbb{R}^{d_1 \times d_2}$ is the weight matrix of the first layer, $\boldsymbol{W} \in \mathbb{R}^{d_2 \times K}$ is the weight matrix of the output layer, $\sigma(\cdot)$ denotes an element-wise nonlinear function (*e.g.*, ReLU), $\text{CrossEntropy}(\cdot,\cdot)$ is the cross-entropy loss and $\text{SoftMax}(\cdot)$ is the softmax function. We denote the intermediate outputs as $\boldsymbol{U} = \boldsymbol{V}^\top \boldsymbol{x}$ and $\boldsymbol{P} = \sigma(\boldsymbol{U})$. Because the weights for the MLP are hierarchically structured, we propose a simple yet effective heuristic to teach MLP learners. The training label is generated by

$$\min_{\boldsymbol{y}} \underbrace{\left\| \boldsymbol{W}^t - \eta_t \frac{\partial \ell(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{V}^t, \boldsymbol{W}^t)}{\partial \boldsymbol{W}^t} - \boldsymbol{W}^* \right\|_F^2}_{\text{Discrepancy of the output layer: } \|\boldsymbol{W}^{t+1} - \boldsymbol{W}^*\|} + \beta \underbrace{\left\| \boldsymbol{V}^t - \eta_t \frac{\partial \ell(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{V}^t, \boldsymbol{W}^t)}{\partial \boldsymbol{P}^t} \circ \frac{\partial \boldsymbol{P}^t}{\partial \boldsymbol{U}^t} \circ \frac{\partial \boldsymbol{U}^t}{\partial \boldsymbol{V}^t} - \boldsymbol{V}^* \right\|_F^2}_{\text{Discrepancy of the first layer: } \|\boldsymbol{V}^{t+1} - \boldsymbol{V}^*\|} \quad (4)$$

where we are using the chain rule to compute $\frac{\partial \ell}{\partial \boldsymbol{V}^t}$. Both $\boldsymbol{P}^t$ and $\boldsymbol{U}^t$ are intermediate outputs at the t-th iteration. β is a hyperparameter that balances the importance of weights in different layers. When $\beta = 1$, it becomes $\min_{\boldsymbol{y}} \|[\boldsymbol{W}^{t+1}; \boldsymbol{V}^{t+1}] - [\boldsymbol{W}^*; \boldsymbol{V}^*]\|_F^2$ which is equivalent to minimizing the Euclidean distance between the concatenated weight matrices and the desired ones.

3.3 Learning a Parameterized Teaching Policy

Since the greedy policy only considers $T = 1$ and it is by no means an optimal teaching policy, we take a step further by considering $T > 1$. In order to solve $\min_{\pi_{\boldsymbol{\theta}}} \|\boldsymbol{w}^T - \boldsymbol{w}^*\|_2^2$, we propose to learn a parameterized teaching policy via two distinct approaches: *unrolling* and *policy gradient*.

Unrolling. We can unroll the optimization algorithm of the learner into the teaching objective. In general, we aim to solve the following bi-level nested optimization objective for the teacher:

$$\min_{\boldsymbol{\theta}} \left\| \boldsymbol{w}^T(\boldsymbol{\theta}) - \boldsymbol{w}^* \right\|_2^2 \quad \text{s.t. } \boldsymbol{w}^T(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{w}} \mathbb{E}_{\{\boldsymbol{x}, \tilde{\boldsymbol{y}}\}} \big\{ \ell(\boldsymbol{x}, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}, \tilde{\boldsymbol{y}}, \boldsymbol{w}^t, \boldsymbol{w}^*) | \boldsymbol{w}) \big\} \quad (5)$$

where ℓ is the learner loss and $\pi_{\boldsymbol{\theta}}$ is the parameterized teaching policy that generates the label. For gradient descent learners, $\boldsymbol{w}^T$ is obtained by unrolling T steps of stochastic gradient descent *w.r.t.* the learner loss. This shares similar spirits with back-propagation through time in recurrent networks and meta-learning [13]. The greedy policy is in fact the optimal solution to Eq. (5) for $T = 1$. For larger T, unrolling builds a larger computational graph for back-propagation and is more costly.

Optimizing the unrolled teaching policy is conceptually similar to optimizing recurrent neural networks with back-propagation through time, as illustrated in Fig. 4. Unrolling T steps is equivalent to seeking a teaching policy that best minimizes the weight discrepancy after T-step gradient descent. Broadly speaking, our method also has intrinsic connections to learning an optimizer [4, 30] and teaching a loss function [61] in the sense that both aim to learn better gradients.

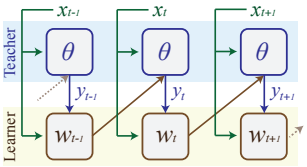


Figure 4: Unrolling the teacher.

Policy gradient. We can simply define $r_t = -\|\boldsymbol{w}^t - \boldsymbol{w}^*\|$ as the reward and directly apply policy gradient [59] to solve it. The final teaching objective is to maximize the expected reward: $\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := \mathbb{E}_{(s_t,a_t)\sim\pi_{\boldsymbol{\theta}}(a|s)}\{\sum_{t=1}^{T}\gamma^t r_t(s_t,a_t)\}$ where $\gamma\in(0,1]$ is the discounting factor, T denotes the termination iteration and (s_t,a_t) is the state-action pair at the t-the iteration. The state is characterized by the sample $\{\boldsymbol{x},\tilde{\boldsymbol{y}}\}$, current learner $\boldsymbol{w}$ and optimal learner $\boldsymbol{w}^*$. The action space is a discretized label space. The policy gradient for updating $\boldsymbol{\theta}$ is $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(a_t|s_t)}\{R\nabla_{\boldsymbol{\theta}}\sum_{t=1}^{T}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)\}$ where $R=\sum_{t=1}^{T}\gamma^t r_t(s_t,a_t)$. The details for the state features and action space are given in Appendix F.

Parameterization of the teacher. The teaching policy can be parameterized as any neural network, such as MLP, CNN [25, 48, 18], transformer [56]. It can be beneficial to use dynamic networks [16, 35, 22] as the teacher, since the optimal teacher parameters for different input may vary.

3.4 Theoretical Insights

We first study how label synthesis affects the gradients when the label is a scalar. For LSR learners, the gradient *w.r.t.* $\boldsymbol{w}$ of a single sample $(\boldsymbol{x},\tilde{y})$ is $\nabla_{\boldsymbol{w}}\ell = (\langle\boldsymbol{w},\boldsymbol{x}\rangle - \tilde{y})\boldsymbol{x}$. For LR learners, the gradient is $\nabla_{\boldsymbol{w}}\ell = \frac{-\tilde{y}\boldsymbol{x}}{1+\exp(\tilde{y}\langle\boldsymbol{w},\boldsymbol{x}\rangle)}$. Modifying the label is essentially to re-scale the gradient without changing the direction. We denote the new gradient after modifying the label to y as $\nabla'_{\boldsymbol{w}}\ell = g(y)\nabla_{\boldsymbol{w}}\ell$ where $g(y)$ is a scalar function of y. For LSR and LR learners, we have $g(y) = \frac{\langle\boldsymbol{w},\boldsymbol{x}\rangle - y}{\langle\boldsymbol{w},\boldsymbol{x}\rangle - \tilde{y}}$ and $g(y) = \frac{y(1+\exp(\tilde{y}\langle\boldsymbol{w},\boldsymbol{x}\rangle))}{\tilde{y}(1+\exp(y\langle\boldsymbol{w},\boldsymbol{x}\rangle))}$, respectively ($\tilde{y}$ is the original label for $\boldsymbol{x}$). In general, the new gradient of linear learners (*e.g.*, LSR, LR and SVM) in greedy LAST can be re-written as $g(y)\nabla_{\boldsymbol{w}}\ell$ where $g(y)$ varies for different cases. However, in the mini-batch case, modifying labels will become much more flexible than adjusting the learning rate, since $g(y)$ is generally sample-specific and learning rate is not.

We discuss how $g(y)$ determines the exponential teachability, since LAST can be viewed as modifying $g(y)$ to minimize the model parameter discrepancy. Here we consider the learner to have a loss function $f(\boldsymbol{w}) = \sum_{i=1}^{n}\ell_i(\boldsymbol{w})$ where $\ell_i(\boldsymbol{w}) = \ell(\boldsymbol{x}_{[i]}, y_{[i]}|\boldsymbol{w})$ and n is the number of samples. We show in Theorem 1 that exponential teachability [33, 34] (*i.e.*, the teacher guides the learner to converge to $\boldsymbol{w}^*$ at an exponential speed) can be achieved by intelligently tuning $g(y)$ in LAST:

Theorem 1 (Exponential teachability)**.** *Assume that the learner loss function ℓ_i has the property of interpolation, L_i-Lipschitz, and convexity. And f is order-1 μ strongly convex. Then LAST can achieve ET with $g(y) = c_1\|\boldsymbol{w}^t - \boldsymbol{w}^*\|$,* i.e., $\mathbb{E}\{\|\boldsymbol{w}^T - \boldsymbol{w}^*\|^2\} \leq (1 - c_1\eta_t\bar{\mu} + c_1^2\eta_t^2 L_{\max})^T \left\|\boldsymbol{w}^0 - \boldsymbol{w}^*\right\|^2$ *where $L_{\max} = \max_i L_i$ and $\bar{\mu} = \sum_i \mu_i/n$. It implies that $(\log\frac{1}{c_2})^{-1}\log(\frac{1}{\epsilon})$ samples are needed to achieve $\mathbb{E}\{\|\boldsymbol{w}^T - \boldsymbol{w}^*\|^2\} \leq \epsilon$. $c_2 = 1 - c_1\eta_t\bar{\mu} + c_1^2\eta_t^2 L_{\max}$ and c_1 is adjusted such that $0 < c_1\eta_t < \bar{\mu}/L_{\max}$.*

Theorem 1 shows that $g(y)$ plays a critical role, similar in spirit to the way that in line search, the step size is adaptively adjusted. We connect Theorem 1 to Armijo linear search in Appendix B. When the label y is a scalar, LAST can also be viewed as a method that intelligently controls the step size in order to improve convergence to a desired model $\boldsymbol{w}^*$. Theorem 1 is also stronger than the theoretical results in [33, 34], since it does not require to modify $\boldsymbol{x}$ to achieve ET. When y becomes a vector, then LAST essentially controls a multi-dimensional learning rate that can modify the gradient in a more fine-grained fashion. As an example, for LSR learners with a vector $\boldsymbol{y}\in\mathbb{R}^K$, we have the new gradient matrix of size $d\times K$ (after label synthesis) as $\nabla'_{\boldsymbol{w}}\ell = g(\boldsymbol{y})\cdot\nabla_{\boldsymbol{w}}\ell$ where $g(\boldsymbol{y}) = \frac{1}{d}(\boldsymbol{y} - \langle\boldsymbol{w},\boldsymbol{x}\rangle)\cdot(1\oslash(\tilde{\boldsymbol{y}} - \langle\boldsymbol{w},\boldsymbol{x}\rangle))$ ($\oslash$ denotes the Hadamard division). We show in Theorem 2 that super-exponential teachability can be achieved when $g(\boldsymbol{y})$ is properly selected:

Theorem 2 (Super-exponential teachability)**.** *Consider a learner loss $f(\boldsymbol{w}) = \sum_{i=1}^{n}\ell_i(\boldsymbol{w})$ and an optimal weight matrix $\boldsymbol{w}^*$ such that $\nabla_{\boldsymbol{w}} f(\boldsymbol{w}^*) = 0$. Assume that $\nabla_{\boldsymbol{w}} f(\boldsymbol{w})$ is L-smooth and continuously differentiable within a sufficiently small δ-neighborhood $\|\boldsymbol{w} - \boldsymbol{w}^*\| \leq \delta$. In LAST, the gradient update is $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta_t g(\boldsymbol{y})\nabla_{\boldsymbol{w}} f(\boldsymbol{w})$. If we have that $\boldsymbol{w}^0$ is in the δ-neighborhood, $|\lambda_{\min}(\nabla^2_{\boldsymbol{w}} f(\boldsymbol{w}))| \geq \mu$ and $L\mu\delta < 2$, then LAST can achieve super-ET with $g(\boldsymbol{y}) = \eta_t^{-1}(\nabla^2_{\boldsymbol{w}} f(\alpha\boldsymbol{w}^* + (1-\alpha)\boldsymbol{w}))^{-1}$ where $\alpha\in[0,1]$. When $\alpha = 0$, the result reduces to Newton's method.*

Theorem 2 can be viewed as a generalization of Newton's method. In constrast to Newton's method, LAST takes advantages of the knowledge of the optimal learner $\boldsymbol{w}^*$ and reduces the requirement of the Hessian information. If we simply set $\alpha = 1$, then LAST only needs a single static $\nabla^2_{\boldsymbol{w}} f(\boldsymbol{w}^*)$ to achieve super-ET. In contrast, Newton's method requires $\nabla^2_{\boldsymbol{w}} f(\boldsymbol{w})$ in every iteration (*i.e.*, Hessian information of a local neighborhood). If we borrow ideas from [46], we can easily extend Theorem 2 to work with sub-sampled Hessian $\frac{1}{|\mathcal{S}|}\sum_{i\in|\mathcal{S}|}\nabla^2_{\boldsymbol{w}}\ell_i(\boldsymbol{w}^*)$ where $\mathcal{S}$ is a subset of indices.

3.5 Combining Sample Selection to LAST

It is natural to further consider combining sample selection with LAST. A straightforward approach is to first perform the teaching sample selection in IMT and then run LAST to synthesize the labels for the selected samples. This can be viewed as applying alternating optimization to approximately solve $\min_{\boldsymbol{x},\boldsymbol{y}} G(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{w}^t)$. Specifically, mixed teaching first solves $\boldsymbol{x}$ with $\boldsymbol{y}$ fixed as the original ground truth label, and then solves $\boldsymbol{y}$ with $\boldsymbol{x}$ fixed. This procedure is iterated until sufficiently small error $\|\boldsymbol{w}^* - \boldsymbol{w}^t\|$ is attained. Due to stronger flexibility, this mixed greedy teaching strategy yields a more powerful policy than LAST. Based on [33], it is easy to verify that mixed teaching can achieve ET. Fig. 5 gives an illustrative comparison of SGD, LAST and mixed teaching. Unlike LAST that tends to follow the largest gradient direction (same as SGD), mixed teaching finds the closest point to $\boldsymbol{w}^*$ in the gradient space. Mixed teaching combines label synthesis in LAST with example selection in IMT, and is able to achieve faster teaching empirically.

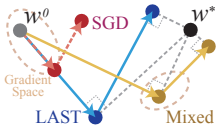


Figure 5: Comparison.

4 BLAST: Black-box Label Synthesis Teaching

The previous section considers omniscient teaching where everything about the learner is accessible. However, the optimal learner parameters ($\boldsymbol{w}^*$) are often unavailable in practice. This motivates us to study black-box teaching [34, 11, 61, 12] where the teacher knows much less information about the learner. A significant difference is that a black-box teacher has no access to $\boldsymbol{w}^*$. Therefore, we need to use a surrogate objective to efficiently guide the learner to $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\{\ell(\boldsymbol{x},\tilde{\boldsymbol{y}}|\boldsymbol{w})\}$. Such a surrogate objective can either be a differentiable criterion on a validation set (*e.g.*, some learner loss function) or a non-differentiable performance measure on a validation set (*e.g.*, validation accuracy). For the former, we propose to unroll the teacher into the optimization of the learner. For the latter, we define the performance measure as the reward signal and formulate black-box teaching as a reinforcement learning problem. Then we utilize the policy gradient method to solve it.

Unrolling. We start with a fully differentiable variant for BLAST. Because $\boldsymbol{w}^*$ is no longer available in the black-box scenario, the objective $\|\boldsymbol{w}^T - \boldsymbol{w}^*\|$ becomes infeasible. In order to conduct the black-box teaching, we propose to directly use the learner loss $\ell(\boldsymbol{x},\tilde{\boldsymbol{y}}|\boldsymbol{w})$ as the surrogate. After removing the dependence on $\boldsymbol{w}^*$ from Eq. (5), we obtain the objective function of BLAST:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\{\boldsymbol{x},\tilde{\boldsymbol{y}}\}\sim\mathcal{D}_v}\left\{\ell(\boldsymbol{x},\tilde{\boldsymbol{y}}|\boldsymbol{w}^T)\right\} \quad \text{s.t.} \quad \boldsymbol{w}^T(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{w}} \mathbb{E}_{\{\boldsymbol{x},\tilde{\boldsymbol{y}}\}\sim\mathcal{D}_r}\left\{\ell(\boldsymbol{x},\pi_{\boldsymbol{\theta}}(\boldsymbol{x},\tilde{\boldsymbol{y}},\boldsymbol{w},\boldsymbol{w}^*)|\boldsymbol{w})\right\} \tag{6}$$

where $\tilde{\boldsymbol{y}}$ denotes the ground truth label of sample $\boldsymbol{x}$ and the output of $\pi_{\boldsymbol{\theta}}$ is the synthesized label $\boldsymbol{y}$. $\mathcal{D}_v$ denotes the validation set, and $\mathcal{D}_r$ denotes the training set. Sometimes we can use the same set as $\mathcal{D}_v$ and $\mathcal{D}_r$. Here we use stochastic gradient descent to solve the inner minimization. Typically we can unroll T-step gradient updates. For example, when unrolling one-step gradient update, we have

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{w}^0}\mathbb{E}_{\{\boldsymbol{x},\tilde{\boldsymbol{y}}\}\sim\mathcal{D}_v}\left\{\ell\big(\boldsymbol{x},\tilde{\boldsymbol{y}}|\boldsymbol{w}^0 - \eta\nabla_{\boldsymbol{w}=\boldsymbol{w}^0}\mathbb{E}_{\{\boldsymbol{x},\tilde{\boldsymbol{y}}\}\sim\mathcal{D}_r}\{\ell(\boldsymbol{x},\pi_{\boldsymbol{\theta}}(\boldsymbol{x},\tilde{\boldsymbol{y}},\boldsymbol{w},\boldsymbol{w}^*)|\boldsymbol{w})\}\big)\right\} \tag{7}$$

where $\boldsymbol{w}^0$ is the random initialization of learner parameters. Eq. (7) can also be easily extended to the case where multiple steps of gradient updates are unrolled, at the cost of more computational overhead. With unrolling, the teacher can be trained in an end-to-end fashion. To reduce the difficulty of learning the teacher in practice, we can parameterize the synthesized label to be $\alpha\tilde{\boldsymbol{y}} + (1-\alpha)\boldsymbol{y}'$ where $\alpha \in (0,1)$ and $\boldsymbol{y}' = \pi_{\boldsymbol{\theta}}(\boldsymbol{x},\tilde{\boldsymbol{y}},\boldsymbol{w},\boldsymbol{w}^*)$. This is essentially to learn a residue between the synthesized label and the ground truth label. The space of the synthesized label is largely constrained such that the black-box teaching can be easier. When α is large, the black-box teaching will be closer to the standard empirical risk minimization and the gain from teaching also becomes smaller.

Policy gradient. We also explore a more general variant for BLAST. We first frame black-box teaching as a reinforcement learning task. Since we have no access to the optimal model $\boldsymbol{w}^*$, we use a hold-out set to evaluate the performance of the learner (Fig. 6), which also serves as the surrogate learning objective. Typically the performance on the hold-out set is non-differentiable *w.r.t.* $\boldsymbol{\theta}$, so we leverage the policy gradient method [59] to learn the teaching policy $\boldsymbol{\theta}$. Specifically, we use the

following objective function: $\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) := \mathbb{E}_{(s_t,a_t)\sim\pi_{\boldsymbol{\theta}}(a|s)}\{\sum_{t=1}^{T} r_t(s_t,a_t)\}$ where T denotes the termination iteration, (s_t, a_t) is the state-action pair at the t-the iteration and $r_t(s_t, a_t)$ is the reward. Since we use a terminal reward $R = r_T(s_t, a_t)$ defined on a hold-out set, the objective becomes $J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\{R\}$. Therefore, we update the teacher parameters $\boldsymbol{\theta}$ with $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ where

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \mathbb{E}_{\pi_{\boldsymbol{\theta}}(a_t|s_t)}\{\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) R\} \approx \sum_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) R. \tag{8}$$

We take classification as an illustrative example here. Specifically, there are many ways to define the terminal reward, for example, *(i)* the accuracy after T iterations on the hold-out set, and *(ii)* the number of iterations to achieve a preset accuracy ζ on the hold-out set (T is the maximal iteration number). We can parameterize the teacher as any classification model, such as linear classifier or neural network. We use a MLP as the teacher. The learner optimizes its weights by cross-entropy loss and SGD. The teaching policy aims to generate suitable labels for input samples so that the learner can converge efficiently. Fig. 6 illustrates the reinforcement learning formulation. Since there are many choices to design the state features, action space and the teaching policy, we emphasize that our goal here is to demonstrate the feasibility of applying reinforcement learning to perform black-box teaching instead of enumerating all the possible design choices.

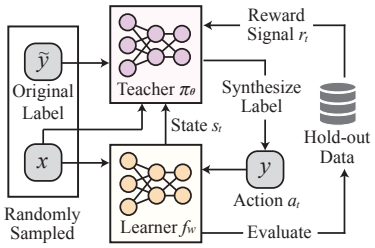


Figure 6: RL formulation for BLAST.

- **State**. State features are the input for the teacher model and are crucial for learning a good teaching policy. For simple linear learners, we construct the state features $h(s_t)$ with the current learner $\boldsymbol{w}$, the optimal learner $\boldsymbol{w}^*$, the input sample $\boldsymbol{x}$ and the original ground truth label $\tilde{\boldsymbol{y}}$. For complex nonlinear learners, we follow the practice in [28, 11, 5] and construct the state features $h(s_t) = \{\mathcal{D}_t, \mathcal{M}_t, \mathcal{C}_t\}$ where $\mathcal{D}$ is the data features that summarize input examples, such as ground truth labels and low-level generic image features, $\mathcal{M}$ is the learner model features that represent the learner status such as historical training loss and validation accuracy, and $\mathcal{C}$ summarizes the interaction between the input examples and the learner such as predicted probabilities and loss value of the input example. Details are given in Appendix F.
- **Action**. For small number of classes, we can uniformly discretize the label space (*i.e.*, simplex) into a set of vectors. For example, the binary label space can be discretized into $[0, 1]$, $[0.25, 0.75]$, $[0.5, 0.5]$, $[0.75, 0.25]$ and $[1, 0]$. For large number of classes, we need to reduce search difficulty in the label space. For example, inspired by label smoothing (LS) and knowledge distillation (KD), we design the synthesized label to be $\boldsymbol{y} = \mu\tilde{\boldsymbol{y}} + (1-\mu)\boldsymbol{p}$ where $\tilde{\boldsymbol{y}}$ is the one-hot ground truth label. $\boldsymbol{p}$ is an all-K vector in LS and is a soft label vector from a pretrained model in KD (K is the number of classes). We discretize the action space $\mu \in [0.5, 1]$ and use the reward to guide the search of state-dependent μ. The teaching policy produces $\mu_t = \pi_{\boldsymbol{\theta}}(h(s_t))$ where $\pi_{\boldsymbol{\theta}}$ is a MLP.

5 Empirical Evaluation and Discussion

This section empirically evaluates LAST in both omniscient and black-box teaching scenarios. Additional details and more results are given in Appendix F and E, respectively.

5.1 Omniscient Greedy Teaching

Least square regression learner. We evaluate LAST on linear regression where the training data is generated by $\tilde{\boldsymbol{y}} = \langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + \epsilon$ (ϵ follows the normal distribution). We compute the training objective value and the Euclidean distance between $\boldsymbol{w}^t$ and $\boldsymbol{w}^*$ in Fig. 7 to evaluate empirical convergence. For LAST, we test "no constraint" (denoted as "NC") and "magnitude constraint".

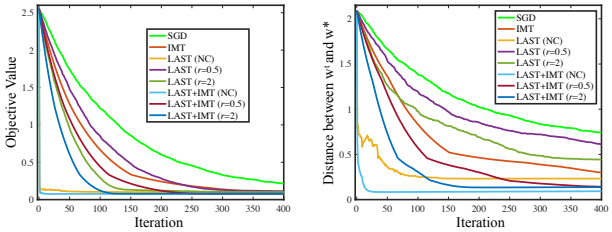


Figure 7: Convergence curves for linear regression. LAST (r=0.5) denotes LAST with magnitude constraint of $r = 0.5$.

The results show that LAST generally converges better with a more flexible teacher, and outperforms IMT with magnitude coefficient $r = 2$. This suggests that a suitable label may sometimes be more crucial than a suitable input example. We also observe that mixed teaching (LAST+IMT) converges significantly faster than IMT or LAST alone, implying that alternating optimization works well.

Logistic regression learner. Next we evaluate LAST on linear classification with logistic regression. We teach the logistic regression learner to perform binary classification on both synthetic data and real image data. For synthetic data, we use 2D half moon data and Gaussian cluster data (in Appendix E). For real image data, we use 3/5 digits in MNIST. From Fig. 8, we observe that mixed teaching converges efficiently and consistently achieves the best performance. All variants of LAST significantly outperform SGD. Without constraints, LAST can outperform IMT without the costly example selection.

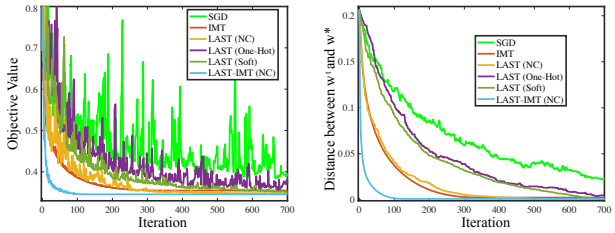


Figure 8: Binary classification for logistic regression. "One-Hot" and "Soft" denote the one-hot and soft constraints, respectively.

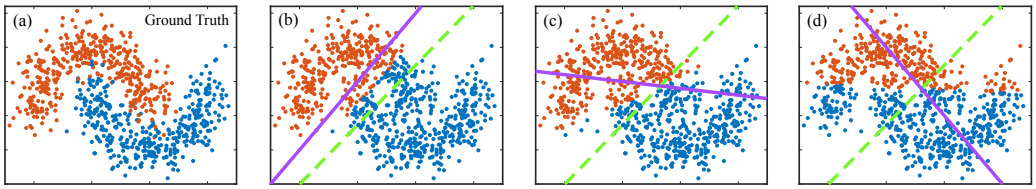


Figure 9: Synthesized labels on half-moon data with 1-hot constraint. (a) ground truth labels. (b,c,d) synthesized labels by LAST. The purple line is the current classifier and the green dotted line is the optimal classifier.

Fig. 9 shows how synthesized labels change depending on the learner's parameters. We observe that LAST synthesizes labels adaptively based on the current learner. Our results connect well to the VC theory that the convergence rate depends on how well a classifier can separate the data [54]. The synthesized labels are always linearly separable, which improves convergence.

Multi-layer perceptron learner. We apply LAST to neural networks to empirically show its effectiveness. Specifically, we use LAST ($\beta = 1$) to teach a 2-layer MLP with ReLU nonlinearity (the output is connected to a logistic regression classifier). We perform binary classification of digit 3 and 5 on MNIST. Results in Fig. 10 show that all the variants of LAST achieve significantly better convergence than SGD. With a teacher that generates labels without constraints, both LAST and mixed teaching achieve far better convergence than IMT. LAST is also empirically faster.

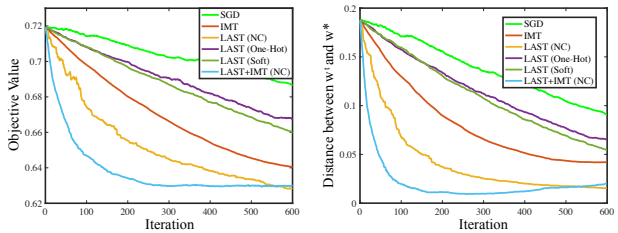


Figure 10: Convergence curves for MLP learner.

Fig. 11 presents some teaching examples in 3/5 and 7/9 classification, where the one-hot labels synthesized by LAST differ from the ground truth. The synthesized labels depend on the learner status and may vary across iterations. Most of the examples are semantically ambiguous to humans, suggesting that labels of hard samples near the decision boundary are helpful for convergence.

Figure 11: Ground truth vs. LAST.

5.2 Omniscient Parameterized Teaching

Logistic regression learner. We first learn a parameterized teaching policy for linear classifier learners. We parameterize the teacher with a MLP. Details are given in Appendix F. We conduct experiments on binary classification of digit 3/5 in MNIST. Fig. 12 shows that both unrolling and policy gradient can learn an effective teaching policy that achieves better convergence than SGD. Moreover, we evaluate two unrolling settings (20-step and 100-step) and both cases outperform policy gradient by a large margin. Due to the teaching target, the distance to $\boldsymbol{w}^*$ converges more stably.

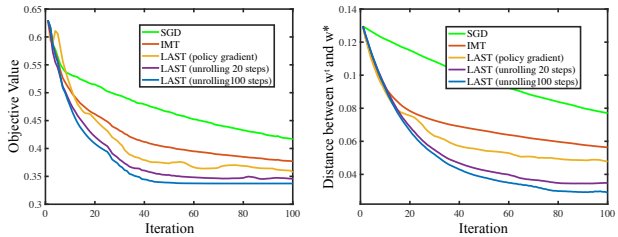


Figure 12: Parameterized teaching for linear learners.

Multi-layer perceptron learner. We then learn a parameterized teaching policy for 2-layer MLP learners. We parameterize the teacher with a MLP. Details are given in Appendix F. We conduct multi-class classification of MNIST digit 1/3/5. For the unrolling, we unroll the teaching policy into 20 steps of gradient update. The results in Fig. 13 show that both unrolling and policy gradient are powerful enough to learn an effective teaching policy that achieves faster convergence than SGD. We set the batch size as 1 for all the methods. We can observe that SGD has large variance and can barely converge. In contrast, LAST converges fast and stably to the optimal learner.

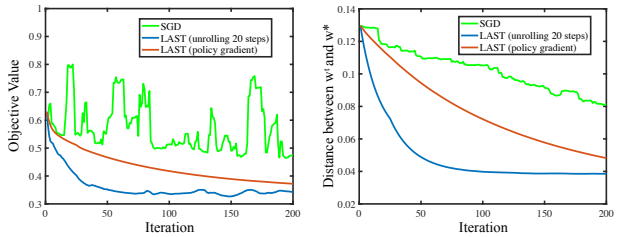


Figure 13: Parameterized teaching for MLP learners.

5.3 Black-box Parameterized Teaching

Logistic regression learner. To compare how unrolling and policy gradients perform in the black-box scenario, we first evaluate both unrolling and policy gradient on teaching a logistic regression learner on MNIST 3/5 digit classification. We parameterize the teacher with a MLP. Details are given in Appendix F. For the policy gradient, the space of synthesized label (*i.e.*, action space) includes $[0, 1]$, $[0.25, 0.75]$, $[0.5, 0.5]$, $[0.75, 0.25]$ and $[1, 0]$. For the unrolling, we unroll the teaching policy into 20 steps of the gradient update. The results in Fig. 14 demonstrate the great potential of BLAST to be able to consistently outperform SGD in the black-box scenario.

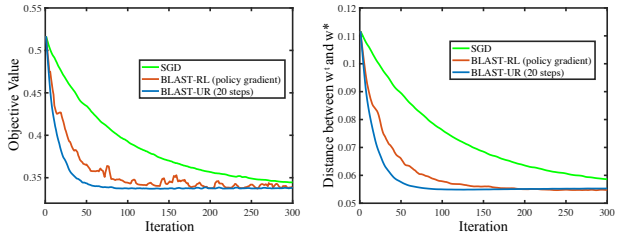
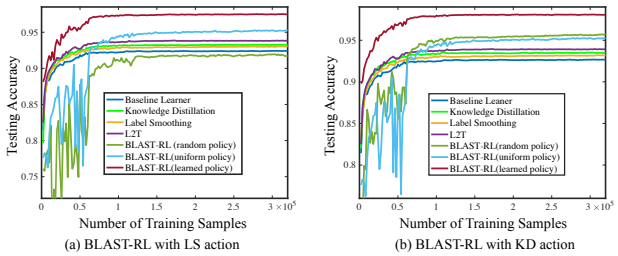


Figure 14: Black-box parameterized teaching for linear learners.

Multi-class neural learner. We study two different types of action space for the reinforcement learning framework in BLAST. As elaborated in Section 4, we consider the synthesized label as $\boldsymbol{y} = \mu\tilde{\boldsymbol{y}} + (1-\mu)\boldsymbol{p}$. For KD, $\boldsymbol{p}$ is the soft label from a pretrained model. For LS, $\boldsymbol{p}$ is simply an all-k vector $K \cdot \mathbf{1}$ where K is the number of classes. Our teaching policy aims to learn μ. Both the teacher and the learner use a MLP. Details are given in Appendix F. We use the number of iterations to achieve a preset accuracy as the terminal reward. We first train the teaching policy on the full MNIST training set, then use the learned teaching policy to synthesize labels for the input images. The learner is trained completely using the labels generated by BLAST-RL. Testing accuracy dynamics are shown in Fig. 15. Specifically, we compare our method with the baseline trained with the ground truth labels, knowledge distillation, label smoothing and L2T [11]. We choose the best hyperparameters for the compared methods via cross-validation. In order to show that BLAST-RL has indeed learned some useful teaching policy, we compare it with a learner trained by randomly initialized teacher and uniform teacher (*i.e.*, output each action with equal probability). We can observe that our BLAST-RL converges significantly faster than all the compared methods and also yields better generalization. This experiment verifies the potential for BLAST to be useful in practice.



(a) BLAST-RL with LS action (b) BLAST-RL with KD action

Figure 15: Testing accuracy of BLAST-RL on MNIST.

6 Concluding Remarks

In this paper, we present a novel iterative teaching framework via label synthesis. We consider both the omniscient scenario where the teacher has access to the optimal learner parameters and the black-box scenario where the teacher has no access to the optimal learner parameters. We propose greedy teaching and parameterized teaching, and provide theoretical insights on why they can guide the learner to converge quickly. Extensive experiments demonstrate the effectiveness of our methods. Our methods apply broadly to improve the learning rate of ML models, hence could have negative societal implications if used to train a malicious model more effectively.

References

[1] Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. Geometric approximation via coresets. *Combinatorial and computational geometry*, 2005. 1

[2] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *AAAI*, 2016. 1, 2

[3] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Explicit defense actions against test-set attacks. In *AAAI*, 2017. 1

[4] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016. 5

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009. 1, 4, 7

[6] Yuxin Chen, Oisin Mac Aodha, Shihan Su, Pietro Perona, and Yisong Yue. Near-optimal machine teaching via explanatory teaching sets. In *AISTATS*, 2018. 2

[7] Yuxin Chen, Adish Singla, Oisin Mac Aodha, Pietro Perona, and Yisong Yue. Understanding the role of adaptivity in machine teaching: The case of version space learners. In *NeurIPS*, 2018. 2

[8] Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*, 2016. 2

[9] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 22

[10] Thorsten Doliwa, Gaojian Fan, Hans Ulrich Simon, and Sandra Zilles. Recursive teaching dimension, vc-dimension and sample compression. *JMLR*, 2014. 2

[11] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *ICLR*, 2018. 6, 7, 9, 23

[12] Yang Fan, Yingce Xia, Lijun Wu, Shufang Xie, Weiqing Liu, Jiang Bian, Tao Qin, and Xiang-Yang Li. Learning to reweight with deep interactions. In *AAAI*, 2021. 6

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 4

[14] Ziyuan Gao, Hans Ulrich Simon, and Sandra Zilles. On the teaching complexity of linear sets. In *ALT*, 2015. 2

[15] S.A. Goldman and M.J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 1995. 1

[16] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *ICLR*, 2016. 5

[17] Luis Haug, Sebastian Tschiatschek, and Adish Singla. Teaching inverse reinforcement learners via features and demonstrations. In *NeurIPS*, 2018. 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 5

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 3

[20] Jonathan H Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. In *NIPS*, 2016. 1

[21] Anette Hunziker, Yuxin Chen, Oisin Mac Aodha, Manuel Gomez Rodriguez, Andreas Krause, Pietro Perona, Yisong Yue, and Adish Singla. Teaching multiple concepts to a forgetful learner. In *NeurIPS*, 2019. 2

[22] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *NIPS*, 2016. 5

[23] Edward Johns, Oisin Mac Aodha, and Gabriel J Brostow. Becoming the expert-interactive multi-class machine teaching. In *CVPR*, 2015. 2

[24] Parameswaran Kamalaruban, Rati Devidze, Volkan Cevher, and Adish Singla. Interactive teaching algorithms for inverse reinforcement learning. *arXiv preprint arXiv:1905.11867*, 2019. 2

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 5

[26] Akash Kumar, Adish Singla, Yisong Yue, and Yuxin Chen. Average-case complexity of teaching convex polytopes via halfspace queries. *arXiv preprint arXiv:2006.14677*, 2020. 2

[27] Akash Kumar, Hanqi Zhang, Adish Singla, and Yuxin Chen. The teaching dimension of kernel perceptron. *arXiv preprint arXiv:2010.14043*, 2020. 2

[28] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NeurIPS*, 2010. 7

[29] Laurent Lessard, Xuezhou Zhang, and Xiaojin Zhu. An optimal control approach to sequential machine teaching. In *AISTATS*, 2019. 1, 2

[30] Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016. 5

[31] Gui-Hua Lin and M Fukushima. Some exact penalty results for nonlinear programs and mathematical programs with equilibrium constraints. *Journal of Optimization Theory and Applications*, 2003. 15

[32] Ji Liu, Xiaojin Zhu, and H Gorune Ohannessian. The teaching dimension of linear learners. In *ICML*, 2016. 1, 2

[33] Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B. Smith, James M. Rehg, and Le Song. Iterative machine teaching. In *ICML*, 2017. 1, 2, 3, 4, 5, 6, 23

[34] Weiyang Liu, Bo Dai, Xingguo Li, Zhen Liu, James Rehg, and Le Song. Towards black-box iterative machine teaching. In *ICML*, 2018. 1, 2, 4, 5, 6

[35] Weiyang Liu, Zhen Liu, James M Rehg, and Le Song. Neural similarity learning. *NeurIPS*, 2019. 5

[36] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 22

[37] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, 2016. 21

[38] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015. 2

[39] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. In *NeurIPS*, 2019. 1

[40] Oisin Mac Aodha, Shihan Su, Yuxin Chen, Pietro Perona, and Yisong Yue. Teaching categories to human learners with visual explanations. In *CVPR*, 2018. 2

[41] Farnam Mansouri, Yuxin Chen, Ara Vartanian, Jerry Zhu, and Adish Singla. Preference-based batch and sequential teaching: Towards a unified view of models. In *NeurIPS*, 2019. 1, 2

[42] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, 2015. 2

[43] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *NeurIPS*, 2019. 2

[44] Tomi Peltola, Mustafa Mert Çelikok, Pedram Daee, and Samuel Kaski. Machine teaching of active sequential learners. In *NeurIPS*, 2019. 1

[45] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *ICML*, 2020. 1, 2

[46] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods ii: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016. 6

[47] Rahim Samei, Pavel Semukhin, Boting Yang, and Sandra Zilles. Algebraic methods proving sauer's bound for teaching complexity. *Theoretical Computer Science*, 2014. 2

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[49] Adish Singla, Ilija Bogunovic, G Bartók, A Karbasi, and A Krause. On actively teaching the crowd to classify. In *NeurIPS Workshop on Data Driven Education*, 2013. 1

[50] Adish Singla, Ilija Bogunovic, Gabor Bartok, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *ICML*, 2014. 1, 2

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 2, 3

[52] Behzad Tabibian, Utkarsh Upadhyay, Abir De, Ali Zarezade, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. Enhancing human learning via spaced repetition optimization. *Proceedings of the National Academy of Sciences*, 2019. 2

[53] Sebastian Tschiatschek, Ahana Ghosh, Luis Haug, Rati Devidze, and Adish Singla. Learner-aware teaching: Inverse reinforcement learning with preferences and constraints. In *NeurIPS*, 2019. 2

[54] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *JMLR*, 2015. 2, 8

[55] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5):544–557, 2009. 2

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 5

[57] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *NeurIPS*, 2019. 15, 16

[58] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018. 22

[59] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992. 5, 6

[60] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 1999. 3

[61] Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. Learning to teach with dynamic loss functions. In *NeurIPS*, 2010. 5, 6

[62] Teresa Yeo, Parameswaran Kamalaruban, Adish Singla, Arpit Merchant, Thibault Asselborn, Louis Faucon, Pierre Dillenbourg, and Volkan Cevher. Iterative classroom teaching. In *AAAI*, 2019. 2

[63] Xuezhou Zhang, Shubham Kumar Bharti, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. The teaching dimension of q-learning. *arXiv preprint arXiv:2006.09324*, 2020. 2

[64] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *ICML*, 2020. 2

[65] Xuezhou Zhang, Xiaojin Zhu, and Laurent Lessard. Online data poisoning attacks. In *L4DC*, 2020. 2

[66] Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. Training set debugging using trusted items. In *AAAI*, 2018. 2

[67] Yao Zhou, Arun Reddy Nelakurthi, and Jingrui He. Unlearn what you have learned: Adaptive crowd teaching with exponentially decayed memory learners. In *KDD*, 2018. 1, 2

[68] Yao Zhou, Arun Reddy Nelakurthi, Ross Maciejewski, Wei Fan, and Jingrui He. Crowd teaching with imperfect labels. In *WWW*, 2020. 1, 2

[69] Xiaojin Zhu. Machine teaching for bayesian learners in the exponential family. In *NeurIPS*, 2013. 1, 2

[70] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, 2015. 1, 2

[71] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 2009. 2, 3

[72] Xiaojin Zhu, Ji Liu, and Manuel Lopes. No learner left behind: On the complexity of teaching multiple learners simultaneously. In *IJCAI*, 2017. 2

[73] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018. 1

[74] Sandra Zilles, Steffen Lange, Robert Holte, and Martin Zinkevich. Teaching dimensions based on cooperative learning. In *COLT*, 2008. 2

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We make it clear in abstract and introduction.
   (b) Did you describe the limitations of your work? [Yes] We have discussed them in experiments.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Our approach is general but could be used to train a malicious model more effectively, as discussed in the Concluding Remarks.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read the ethics review guidelines and ensured that our paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See the beginning of every lemma and theorem.
   (b) Did you include complete proofs of all theoretical results? [Yes] They are included in the main text and appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We have included them in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We used the open-sourced datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No] We didn't use any crowdsourcing or conduct research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [No] We didn't include any human participant.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No] We didn't include any human participant.