Iterative Teaching by Label Synthesis

Weiyang Liu[1,2,*] **Zhen Liu**[3,*] **Hanchen Wang**[1,*] **Liam Paull**[3,4] **Bernhard Schölkopf**[2] **Adrian Weller**[1,5]
[1]University of Cambridge [2]MPI for Intelligent Systems, Tübingen [3]Mila, Université de Montréal [4]CIFAR AI Chair
[5]The Alan Turing Institute [*]Equal Contribution ✉{wl396@cam.ac.uk, zhen.liu.2@umontreal.ca}

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 propose multiple novel teaching algorithms in this framework. Finally, we empirically demonstrate the value of our framework.

1 Introduction

Machine teaching [102, 105] 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 [72, 71, 99, 100] to model robustness [2, 3, 63, 54]. Machine teaching also has nice connections with curriculum learning [7] and coresets [1, 28].

Based on the learner, machine teaching can be performed in either batch or sequential fashion. The majority of prior work studies batch machine teaching [101, 102, 42, 56], 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* [22]. Differently, sequential machine teaching [43, 44, 38, 61] bridges the gap between machine teaching and practical learning algorithms by studying the sequential (*i.e.*, iterative) learner such as neural networks. A typical example is iterative machine teaching (IMT) [43, 44] 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 [43] performs the teaching sample selection with a greedy policy, but it could be substantially sub-optimal in certain cases [38] and its computational complexity also scales linearly with the size of the dataset.

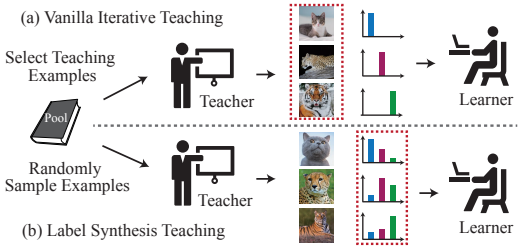

Figure 1: Comparison of vanilla iterative machine teaching and label synthesis teaching. The red dotted frames indicate the teacher's efforts.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

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 [26] and label smoothing [73]. 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, [78, 77] 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 a learner (with the hope to implicitly encode priviledged information) in order to improve the learner's convergence.

Specifically, we study LAST primarily under the omniscient scenario where the teacher knows everything about the learner (*e.g.*, the optimal learner parameters). To perform omniscient teaching, we consider a greedy teacher and a parameterized teacher. We show that greedy teaching can achieve exponential teachability (ET) [43] without selecting teaching examples. Additionally, we touch upon the black-box teaching scenario where the teacher knows less about the learner (*e.g.*, the optimal learner parameters are unavailable), and discuss how to perform LAST in this case.

LAST provides a unified view for understanding soft label methods, *e.g.*, knowledge distillation [26, 53], label smoothing [73, 11, 59], and self-training [103]. All these methods can be interpreted as modifying the labels to achieve desirable learning behavior and outcome. With LAST, we can connect iterative machine teaching to many classic learning algorithms and shed novel light on them.

2 Related Work

Machine teaching. Batch machine teaching [101, 102, 42, 56] has drawn increasing more attention recently. Different learners typically exhibit distinct behaviors during teaching. [10, 74] focus on how to teach version space learners. The teaching dimension of linear learners [42], kernel learners [36] and reinforcement learner [95] is extensively studied. Teaching a forgetful learner is explored in [29, 44]. [104, 94] consider how to teach multiple learners simultaneously. Sequential machine teaching [43, 44, 38, 91] 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. [38] connects sequential teaching to optimal control and gains interesting insights, but it can not produce a practical teaching policy. [91] uses locality-sensitive sampling to scale IMT to large-scale problems. Machine teaching is shown useful in reinforcement learning [76, 32, 63, 24], human-in-the-loop learning [31, 9, 55], crowd sourcing [72, 99, 100] and cyber security [57, 2, 98, 96, 97]. [13, 21, 107, 65, 9, 35] study machine teaching from a more theoretical point of view.

Cooperative communication. Cooperative communication [69] is a mutual theory of mind reasoning between a teacher and a learner. The teacher selects data to convey a hypothesis and the learner infers a hypothesis given the selected data. Both agents have the shared goal of successfully transmitting beliefs. [84] formulates cooperative communication with the framework of optimal transport. In this framework, pedagogical reasoning [66, 67, 68], cooperative inference [93, 83, 82] and Bayesian teaching [16, 15, 92] can all be viewed as special cases. Among this line of research, cooperative inference [82] shares many similarities to our work and also adopts an iterative teaching paradigm. While [82] formulates the teaching problem with a probabilistic inference framework, LAST addresses the iterative teaching with a nested optimization framework.

Soft-supervised learning. It has been shown that supervising the model with soft labels can be beneficial to generalization [73, 89] and robustness [106, 75]. Label smoothing [73] replaces one-hot label vector with a mixture of itself and the uniform distribution, achieving stronger generalization in training neural networks. Knowledge distillation [26] uses the soft predicted label from a pretrained model to supervise the training of a student model, which can make the student model generalize

better than a trained-from-scratch one. [89] perturbs the one-hot label vector with random noise and also observes better generalization. Large-margin softmax [51, 50, 52, 45] can be approximately viewed as training learners with dynamic soft labels (Appendix E). Self-training [103, 90, 8] also uses dynamic soft labels to iteratively train the learner. In constrast to the empirical nature of soft-supervised learning, LAST formulates it as a unified and principled iterative teaching framework.

3 Label Synthesis Teaching

Teaching protocol. We first consider the simplest teaching protocol following [43]. 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 the 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 batch size.

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, [43] proposes a greedy policy to generate the teaching curriculum, but it is 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 [26] 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 $\boldsymbol{x}$ from a pretrained neural network. Label smoothing [73] 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 [103] feeds the learner with the pseudo-label predicted by the learner from the last iteration. All these methods can be viewed as using some form of customized label synthesis policies.

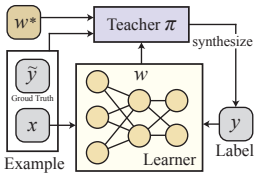

Figure 2: Overview.

3.2 Greedy Teaching Policy

For the t-th iteration, we can approximately solve the original teaching problem with the optimization $\min_{\{\boldsymbol{y}^{t+1},\cdots,\boldsymbol{y}^{t+v}\}} d(\boldsymbol{w}^{t+v}, \boldsymbol{w}^*)$ where $v \leq T$ is the number of steps being considered. By using $v = 1$ and the gradient descent update in LAST, we obtain a greedy policy $\arg\min_{\boldsymbol{y}^t} d(\boldsymbol{w}^{t+1}, \boldsymbol{w}^*)$ where $\boldsymbol{w}^{t+1}$ is one-step update from the current learner $\boldsymbol{w}^t$ *w.r.t.* the learner loss ℓ and example $\{\boldsymbol{x}^t, \boldsymbol{y}^t\}$. 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 to each iteration for generating labels. The label for the example $\boldsymbol{x}^t$ at the t-th iteration is produced by

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

where $\ell(\boldsymbol{x}^t, \boldsymbol{y}^t|\boldsymbol{w}^t)$ 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 $\boldsymbol{x}^t$, the greedy teaching in LAST first randomly samples the teaching example and then optimize the label based on $\arg\min_{\boldsymbol{y}^t} \eta_t^2 T_1(\boldsymbol{x}^t, \boldsymbol{y}^t|\boldsymbol{w}^t) - 2\eta_t T_2(\boldsymbol{x}^t, \boldsymbol{y}^t|\boldsymbol{w}^t)$. The overall 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 a standard integer programming problem [87].

Soft constraint. A straightforward relaxation of the one-hot constraint is to constrain the labels on a probability simplex, *i.e.*, $\sum_i \boldsymbol{y}_i = 1$ and $\boldsymbol{y}_i \geq 0, \forall i$, and it becomes a constrained optimization problem.

Magnitude constraint. For regression, we can constrain the p-norm of the label vector to be smaller than a positive constant r, *i.e.*, $\|\boldsymbol{y} - \langle \boldsymbol{w}^t, \boldsymbol{x}\rangle\|_p \leq r$ (for constraining the synthesized label to be around the current prediction) or $\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|_p \leq r$ (for constraining the synthesized label to be around the ground truth label), where we usually use $p=2$. A larger magnitude r indicates a more flexible and powerful teacher. For classification, we can use the same magnitude constraints (optionally) along with soft constraints $\sum_i \boldsymbol{y}_i = 1$ and $\boldsymbol{y}_i \geq 0, \forall i$.

No constraint. The most powerful teacher has no constraint on the synthesized labels and can synthesize any label vector, making it easy to solve the unconstrained optimization in Eq. (1).

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 gain intuitions by looking at the synthesized label. Taking LSR as an concrete example, 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

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

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 trivial to learn for the current learner (since the learner can output this same prediction without any update) and the optimal label is the most difficult to learn, revealing a principle that a learner should be taught in an easy-to-hard fashion in order to learn a concept efficiently. This implication nicely matches the conclusions drawn in curriculum learning [7], self-paced learning [37], cognitive shaping [17, 62, 34], continuation methods [4] and IMT [43, 44].

Moreover, greedy LAST has the property of teaching monotonicity [43] if the learner loss satisfies certain conditions (See Proposition 1 for a formal argument). This nice property implies that LAST can always converge 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 why LAST yields better convergence to $\boldsymbol{w}^*$ is given in Fig. 3. We can observe that LAST is guaranteed to better minimize the discrepancy between $\boldsymbol{w}^1$ and $\boldsymbol{w}^*$ for the first iteration if both LAST and SGD use the same initialization and training examples. Such a guarantee leads to the teaching monotonicity under a few conditions. If the learner loss is properly designed (*e.g.*, least square loss [43]) such that teaching monotonicity can be satisfied, then LAST will always converge faster than SGD.

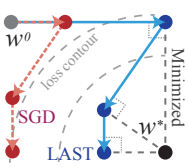

Figure 3: Example.

Proposition 1 (Teaching Monotonicity). *Given a learner ℓ and a randomly sampled teaching data point $(\boldsymbol{x}, \tilde{y}) \sim P$ for both LAST and a random teacher (SGD), if $\mathbb{E}\{G(\boldsymbol{x}, y|\boldsymbol{w}_1)\} \leq \mathbb{E}\{G(\boldsymbol{x}, y|\boldsymbol{w}_2)\}$ holds for any $\boldsymbol{w}_1, \boldsymbol{w}_2$ that satisfy $\mathbb{E}\{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|\} \leq \mathbb{E}\{\|\boldsymbol{w}_2 - \boldsymbol{w}^*\|\}$, then with the same initialization and learning rate, in expectation LAST can converge not slower than the random teacher.*

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

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

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 $v = 1$ and it is by no means an optimal teaching policy, we take a step further by considering $v > 1$. In order to approximately solve $\min_{\boldsymbol{y}^{t+1}, \dots, \boldsymbol{y}^{t+v}} \|\boldsymbol{w}^{t+v} - \boldsymbol{w}^*\|_2^2$, we propose to learn a parameterized teaching policy $\pi_{\boldsymbol{\theta}}$ through $\min_{\pi(\boldsymbol{\theta})} \|\boldsymbol{w}^{t+v} - \boldsymbol{w}^*\|_2^2$.

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:

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

where ℓ is the learner loss and $\pi_{\boldsymbol{\theta}}$ is the parameterized teaching policy that generates the label. For gradient descent learners, $\boldsymbol{w}^v(\boldsymbol{\theta})$ is obtained by unrolling v steps of stochastic gradient descent *w.r.t.* ℓ. Unrolling is shown useful in [12, 41, 58, 47, 46] and shares similar spirits with back-propagation through time in recurrent networks [85] and meta-learning [20]. The greedy policy is the optimal solution to Eq. (5) for $v = 1$, and the parameterize policy aims to approximate the solution to $v > 1$. For large v, unrolling builds a large computational graph for back-propagation and is very 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 v steps is equivalent to seeking a teaching policy that best minimizes the weight discrepancy after v-step gradient descent. Broadly speaking, our method also has intrinsic connections to learning an optimizer [5, 39] and teaching a loss function [88] in the sense that both aim to learn better gradients.

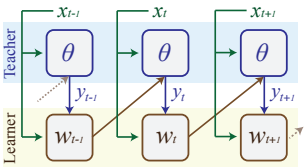

Figure 4: Unrolling the teacher.

Policy gradient. We can define $r_t = -\|\boldsymbol{w}^t - \boldsymbol{w}^*\|$ as the reward signal and directly apply policy gradient [86] to maximize 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 is the termination iteration and (s_t, a_t) is the state-action pair at the t-th 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 given by $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(a_t|s_t)} \{R \nabla_{\boldsymbol{\theta}} \sum_{t=1}^T \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\}$ where $R = \sum_{t=1}^T \gamma^t r_t(s_t, a_t)$. Appendix G gives the details for the state and action. Note that we stick to the simplest policy gradient for demonstration.

Parameterization of the teacher. The teaching policy can be parameterized as any neural network, such as MLP, CNN [33, 70, 25, 27, 52, 49] and transformer [79, 14]. It may also be beneficial to use dynamic networks [23, 48, 30], since the optimal teacher parameters for different input may vary.

3.4 Theoretical Guarantees and Justifications

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 [43, 44] (*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 the learner loss ℓ_i has the property of interpolation, L_i-Lipschitz, and convexity. 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 $\mathcal{O}((\log\frac{1}{c_0})^{-1}\log(\frac{1}{\epsilon}))$ samples are needed to achieve $\mathbb{E}\{\|\boldsymbol{w}^T - \boldsymbol{w}^*\|^2\} \leq \epsilon$. $c_0 = 1 - c_1\eta_t\bar{\mu} + c_1^2\eta_t^2 L_{\max}$ 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 emphasize that a random teacher (*i.e.*, vanilla SGD) cannot achieve ET and yields an optimal rate of $\mathcal{O}(\frac{1}{\epsilon})$ [60] (in order to reach ϵ-approximation of loss value). Additionally, we connect Theorem 1 to Armijo linear search [6, 80] by showing that ET can also be achieved by tuning $g(y)$ in a more fine-grained way under a different set of conditions:

Theorem 2 (Exponential teachability without target parameters)**.** *Assume that the learner loss ℓ_i has the property of interpolation, L_i-smoothness, convexity and f has the property of μ strong-convexity of order 2. If we adjust $g(y)$ such that the following condition is always satisfied in each iteration: $\ell_{i_t}(\boldsymbol{w}^t - \eta_t g(y)\nabla\ell_{i_t}(\boldsymbol{w}^t)) \leq \ell_{i_t}(\boldsymbol{w}^t) - c_2\eta_t g(y)\left\|\nabla\ell_{i_t}(\boldsymbol{w}^t)\right\|^2$, then LAST can achieve ET for iterative learners with $c_2 = \frac{1}{2}$: $\mathbb{E}\{\|\boldsymbol{w}^T - \boldsymbol{w}^*\|^2\} \leq \max\left\{(1 - \frac{\bar{\mu}}{L_{\max}}), (1 - \bar{\mu}\eta'_{\max})\right\}^T \left\|\boldsymbol{w}^0 - \boldsymbol{w}^*\right\|^2$ which indicates that $\mathcal{O}(c_3\log(\frac{1}{\epsilon}))$ samples are needed to achieve $\mathbb{E}\{\|\boldsymbol{w}^T - \boldsymbol{w}^*\|^2\} \leq \epsilon$.*

Remark 1 (Meaningfulness of omniscient teaching)**.** *A natural question arises: what is the point of the target parameters $\boldsymbol{w}^*$ if we can achieve ET without it? First of all, both Theorem 1 and Theorem 2 merely consider the case where the synthesized label is a scaler, while labels are usually high-dimensional vectors in practice. When labels are vectors, LAST is much more flexible than tuning the learning rate. Second, the advantages of LAST is reflected in the easiness of setting $g(y)$. Because there exists a universal constant c_1 for $g(y) = c_1\|\boldsymbol{w}^t - \boldsymbol{w}^*\|$, $g(y)$ is much easier to set in Theorem 1. In contrast, $g(y)$ in Theorem 2 has to satisfy a dynamically changing inequality, which is highly nontrivial. Third, Theorem 2 also shows that Theorem 1 can still be substantially improved and the current lower bound on iterative teaching dimension is not necessarily tight.*

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 reasonably comparable to the theoretical results in [43, 44], but it does not require to modify $\boldsymbol{x}$ to achieve ET. In contrast to deterministic ET in [43], LAST achieves probabilistic ET due to the randomness in data sampling. 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). More interestingly, Theorem 2 also validates the feasibility of black-box LAST teaching, since the convergence can be improved even without target parameters. We show in Theorem 3 that super-exponential teachability can be achieved under certain conditions and a carefully chosen $g(\boldsymbol{y})$:

Theorem 3 (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 3 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). In the light of [64], we can easily extend Theorem 3 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 The Best of Both Worlds? Combining Sample Selection to LAST

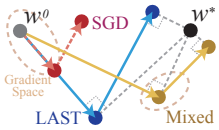

Figure 5: Comparison.

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 [43], 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.

4 A Distant Look at 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 [44, 18, 88, 19] where the teacher knows much less information about the learner. This problem is extremely challenging, but for completeness, we briefly discuss some potential strategies for black-box LAST (BLAST) to tackle this difficult problem and also provide some preliminary experiments in Appendix F.8. Because a black-box teacher has no access to $\boldsymbol{w}^*$, 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 metric on a validation set (*e.g.*, validation accuracy). For the former, we can unroll the teacher into the optimization of the learner. For the latter, we can define the performance metric 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 $\|\boldsymbol{w}^v-\boldsymbol{w}^*\|$ 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:

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

where $\tilde{\boldsymbol{y}}$ denotes the ground truth label of sample $\boldsymbol{x}$ and the output of $\pi_{\boldsymbol{\theta}}$ is the synthesized label $\boldsymbol{y}$. $\mathcal{D}_a$ denotes the validation set, and $\mathcal{D}_r$ denotes the training set. Sometimes we can use the same set as $\mathcal{D}_a$ and $\mathcal{D}_r$. Here we use stochastic gradient descent to solve the inner minimization. Typically we can unroll v-step gradient updates and train the teacher in an end-to-end fashion. To reduce the difficulty of learning the teacher in practice, it can be beneficial to 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 intuition behind is to constrain the space of synthesized labels such that the black-box teaching can be easier.

Policy gradient. We also discuss 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 [86] 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 $\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})=\sum_t\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)R$. 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 approximation method, such as linear classifier or neural network. 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 state features, action space and teaching policy, we aim to demonstrate the feasibility of applying reinforcement learning to perform black-box teaching instead of enumerating all the design choices.

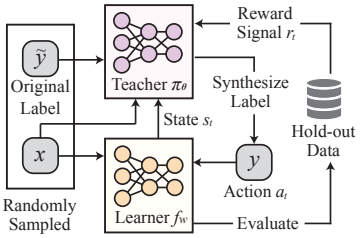

Figure 6: RL formulation for BLAST.

- **State**. State features are the input for the teacher model and are crucial for learning a good teaching policy. As an example for linear learners, we can 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 nonlinear learners, it remains an open problem to design informative state features [7, 37, 18].
- **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 can 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-$\frac{1}{K}$ 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 $\pi_{\boldsymbol{\theta}}$ can be a MLP.

5 Empirical Evaluation and Discussion

This section comprehensively evaluates LAST in the omniscient teaching scenario. Experimental details and more results (including BLAST) are given in Appendix G and Appendix F, 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". 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.

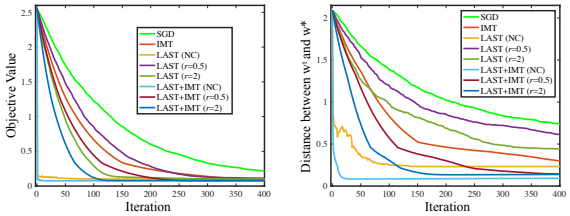

Figure 7: Convergence curves for linear regression. LAST (r=0.5) denotes LAST with magnitude constraint of $r=0.5$.

Logistic regression learner. Next we evaluate LAST on linear classification for logistic regression learners. 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 F). 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 is comparable to IMT in terms of convergence but does not require any costly example selection.

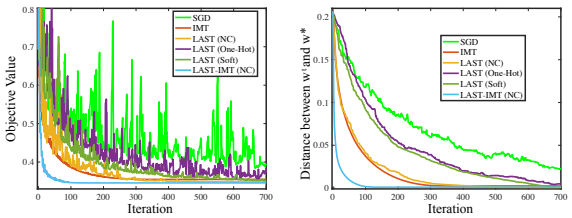

Figure 8: Binary classification for logistic regression. One-Hot: one-hot constraint. Soft: soft constraints.

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

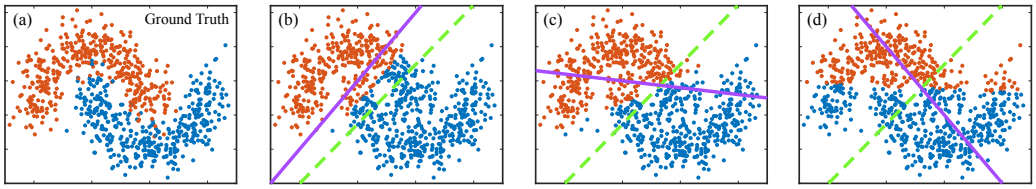

Figure 9: Synthesized labels on half-moon data with 1-hot constraint. (a) ground truth labels. (b,c,d) synthesized labels by LAST. The purple line is the current classifier and the green dotted line is the optimal classifier.

theory that the convergence rate depends on how well a classifier can separate the data [77]. The synthesized labels are always linearly separable, which effectively improves convergence.

Multi-layer perceptron learner. We apply LAST to neural networks in order to empirically show its effectiveness for nonlinear learners. Specifically, we use LAST ($\beta = 1$) to teach a 2-layer MLP with ReLU nonlinearity (the output layer is essentially 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 much more efficient than IMT.

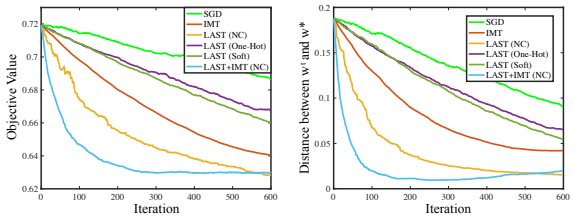

Figure 10: Binary classification for MLP learners.

Fig. 11 presents some teaching examples for MLP learners in 3/5 MNIST digit 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 different iterations. We can observe that most of the examples whose synthesized labels are different from the ground truth ones are semantically ambiguous to humans, suggesting that labels of hard samples near the decision boundary are very crucial for convergence. Moreover, such a observation is consistent for linear learners (see Appendix F.4).

Figure 11: Ground truth vs. LAST.

5.2 Omniscient Parameterized Teaching

We learn a parameterized teaching policy for logistic regression classifier learners using both unrolling and policy gradient. We parameterize the teacher model with a MLP. Experimental details are given in Appendix G. We evaluate the learned policies 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 steps and 100 steps) and unrolling more steps show marginal benefits on convergence. On time-wise convergence, we can observe that all variants of LAST performs significantly faster than IMT, because no costly example selection is needed. On iteration-wise convergence, we can see that IMT still achieves the fastest convergence due to its strong flexibility from example selection. LAST with either unrolling or policy gradient converges much faster than SGD, and policy gradient yields slightly better convergence than unrolling in the experiments.

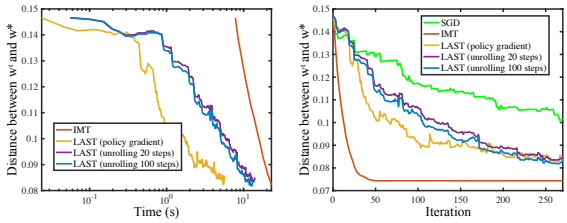

Figure 12: Omniscient parameterized teaching on Binary classification for logistic regression learners. Left: time-wise convergence. Right: iteration-wise convergence.

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.

Acknowledgments

WL is supported by a Cambridge-Tübingen Fellowship, an NVIDIA GPU grant, DeepMind and the Leverhulme Trust via CFI. AW acknowledges support from a Turing AI Fellowship under grant EP/V025379/1, The Alan Turing Institute, and the Leverhulme Trust via CFI. LP is supported by CIFAR AI Chair Program and acknowledges support from Samsung Electronics Co., Ldt. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039B, and by the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645.

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] Eugene L Allgower and Kurt Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 2012. 4

[5] 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

[6] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 1966. 6

[7] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009. 1, 4, 8

[8] Beidi Chen, Weiyang Liu, Zhiding Yu, Jan Kautz, Anshumali Shrivastava, Animesh Garg, and Animashree Anandkumar. Angular visual hardness. In *ICML*, 2020. 3

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

[10] 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

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

[12] Bo Dai, Hanjun Dai, Niao He, Weiyang Liu, Zhen Liu, Jianshu Chen, Lin Xiao, and Le Song. Coupled variational bayes via optimization embedding. In *NeurIPS*, 2018. 5

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

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 5

[15] Baxter S Eaves Jr, Naomi H Feldman, Thomas L Griffiths, and Patrick Shafto. Infant-directed speech is consistent with teaching. *Psychological review*, 2016. 2

[16] Baxter S Eaves Jr and Patrick Shafto. Toward a general, scaleable framework for bayesian teaching with applications to topic models. *arXiv preprint arXiv:1605.07999*, 2016. 2

[17] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 1993. 4

[18] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *ICLR*, 2018. 7, 8

[19] 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. 7

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 5

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

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

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

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

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

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

[27] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 5

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

[29] 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

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

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

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

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

[34] Kai A Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 2009. 4

[35] 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

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

[37] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NeurIPS*, 2010. 4, 8

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

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

[40] 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. 17

[41] Rongmei Lin, Weiyang Liu, Zhen Liu, Chen Feng, Zhiding Yu, James M Rehg, Li Xiong, and Le Song. Regularizing neural networks via minimizing hyperspherical energy. In *CVPR*, 2020. 5

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

[43] 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, 6, 7, 25

[44] 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, 6, 7

[45] Weiyang Liu, Rongmei Lin, Zhen Liu, Lixin Liu, Zhiding Yu, Bo Dai, and Le Song. Learning towards minimum hyperspherical energy. In *NeurIPS*, 2018. 3

[46] Weiyang Liu, Rongmei Lin, Zhen Liu, James M Rehg, Liam Paull, Li Xiong, Le Song, and Adrian Weller. Orthogonal over-parameterized training. In *CVPR*, 2021. 5

[47] Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Schölkopf, and Adrian Weller. Learning with hyperspherical uniformity. In *AISTATS*, 2021. 5

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

[49] Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M Rehg, and Le Song. Decoupled networks. In *CVPR*, 2018. 5

[50] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 3, 21

[51] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, 2016. 3, 21

[52] Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhen Liu, Bo Dai, Tuo Zhao, and Le Song. Deep hyperspherical learning. In *NIPS*, 2017. 3, 5

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

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

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

[56] 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

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

[58] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 2021. 5

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

[60] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 2009. 6

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

[62] Gail B Peterson. A day of great illumination: Bf skinner's discovery of shaping. *Journal of the experimental analysis of behavior*, 2004. 4

[63] 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

[64] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods ii: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016. 7

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

[66] Patrick Shafto and Noah Goodman. Teaching games: Statistical sampling assumptions for learning in pedagogical situations. In *CogSci*, 2008. 2

[67] Patrick Shafto, Noah D Goodman, and Michael C Frank. Learning from others: The consequences of psychological reasoning for human learning. *Perspectives on Psychological Science*, 2012. 2

[68] Patrick Shafto, Noah D Goodman, and Thomas L Griffiths. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive psychology*, 2014. 2

[69] Patrick Shafto, Junqi Wang, and Pei Wang. Cooperative communication as belief transport. *Trends in Cognitive Sciences*, 2021. 2

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

[71] 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

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

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

[74] 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

[75] Christian Thiel. Classification on soft labels is robust against label noise. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 65–73. Springer, 2008. 2

[76] 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

[77] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *JMLR*, 2015. 2, 9

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

[79] 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

[80] 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. 6, 17

[81] Feng Wang, Weiyang Liu, Haijun Liu, and Jian Cheng. Additive margin softmax for face verification. *arXiv preprint arXiv:1801.05599*, 2018. 21

[82] Junqi Wang, Pei Wang, and Patrick Shafto. Sequential cooperative bayesian inference. In *ICML*, 2020. 2

[83] Pei Wang, Pushpi Paranamana, and Patrick Shafto. Generalizing the theory of cooperative inference. In *AISTATS*, 2019. 2

[84] Pei Wang, Junqi Wang, Pushpi Paranamana, and Patrick Shafto. A mathematical theory of cooperative communication. In *NeurIPS*, 2020. 2

[85] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1988. 5

[86] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992. 5, 7

[87] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 1999. 4

[88] 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, 7

[89] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *CVPR*, 2016. 2, 3

[90] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020. 3

[91] Zhaozhuo Xu, Beidi Chen, Chaojian Li, Weiyang Liu, Le Song, Yingyan Lin, and Anshumali Shrivastava. Locality sensitive teaching. In *NeurIPS*, 2021. 2

[92] Scott Cheng-Hsin Yang and Patrick Shafto. Explainable artificial intelligence via bayesian teaching. In *NIPS 2017 workshop on Teaching Machines, Robots, and Humans*, 2017. 2

[93] Scott Cheng-Hsin Yang, Yue Yu, Pei Wang, Wai Keen Vong, Patrick Shafto, et al. Optimal cooperative inference. In *AISTATS*, 2018. 2

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

[95] 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

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

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

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

[99] 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

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

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

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

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

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

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

[106] Bojia Zi, Shihao Zhao, Xingjun Ma, and Yu-Gang Jiang. Revisiting adversarial robustness distillation: Robust soft labels make student better. In *ICCV*, 2021. 2

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