Learning to Reason with Curriculum I: Provable Benefits of Autocurriculum

Nived Rajaraman NRAJARAMAN@MICROSOFT.COM
Microsoft Research

Audrey Huang AUDREYH5@ILLINOIS.EDU
University of Illinois Urbana Champaign

Miro Dudik MDUDIK@MICROSOFT.COM
Microsoft Research

Robert Schapire SCHAPIRE@MICROSOFT.COM
Microsoft Research

Dylan J. Foster DYLANFOSTER@MICROSOFT.COM
Microsoft Research

Akshay Krishnamurthy AKSHAYKR@MICROSOFT.COM
Microsoft Research

Editors: Steve Hanneke and Tor Lattimore

Abstract

Chain-of-thought reasoning, where language models expend additional computation by producing thinking tokens prior to final responses, has driven significant advances in model capabilities. However, training these reasoning models is extremely costly in terms of both data and compute, as it involves collecting long traces of reasoning behavior from humans or synthetic generators and further post-training the model via reinforcement learning. Are these costs fundamental, or can they be reduced through better algorithmic design? We show that *autocurriculum*—where the model uses its own performance to decide which problems to focus training on—provably improves upon standard training recipes for both supervised fine-tuning (SFT) and reinforcement learning (RL). For SFT, we show that autocurriculum requires *exponentially* fewer reasoning demonstrations than non-adaptive fine-tuning (Joshi et al., 2025), by focusing teacher supervision on prompts where the current model struggles. For RL fine-tuning, autocurriculum *decouples* the computational cost from the quality of the reference model, reducing the latter to a burn-in cost that is nearly independent of the target accuracy. These improvements arise purely from adaptive data selection, drawing on classical techniques from boosting (Freund and Schapire, 1997) and learning from counterexamples (Angluin, 1987), and requiring no assumption on the distribution or difficulty of prompts.

Keywords. Autocurriculum, Language model reasoning, Reinforcement Learning

1. Introduction

Recent advances in language models have demonstrated that performance on complex reasoning tasks can be substantially improved by increasing inference-time computation. In particular, multi-step chain-of-thought (CoT) reasoning (Wei et al., 2022) enables models to solve hard tasks by generating long intermediate reasoning traces before producing the final answer. Reasoning models are often trained to elicit this behavior through a combination of supervised fine-tuning and reinforcement learning (Guo et al., 2025). Supervised fine-tuning (SFT) on long CoT data has been one of the

largest drivers of reasoning capabilities in domains like code-generation or mathematical reasoning (Jaech et al., 2024; Hu et al., 2025), but collecting high-quality data for supervision relies heavily on strong teacher models or human labelers, requiring massive and concerted efforts (Guha et al., 2025). On the other hand, reinforcement learning fine-tuning (RL) does not require costly CoT data, but is computationally expensive, as it requires generating a large number of reasoning traces from the model being trained (Liu et al., 2025). To push beyond the capabilities of current frontier models, it is essential to reduce these statistical and computational costs.

A promising approach to this challenge is *curriculum design*, where reasoning problems of varied difficulty are used to guide the model toward solving progressively harder problems. Curricula can be hand-designed by humans based on intuition about problem difficulty, automatically designed by the model itself, adapting to model capabilities over the course of training, or a combination of both. The latter automatic or *autocurriculum approaches* have recently been incorporated into large-scale RL systems as a mechanism to improve compute efficiency and stabilize training (Yu et al., 2025; Khatri et al., 2025), and also to gradually scale problem difficulty (Zeng et al., 2025). Theoretically, however, the algorithmic, statistical, and computational aspects of autocurriculum are poorly understood, particularly as they pertain to LLM reasoning. With this as motivation, we ask:

How should reasoning models design their own curricula?
What are the statistical and computational benefits of autocurriculum for reasoning?

To address these questions, we study autocurriculum in a theoretical framework for learning with autoregressive models (Joshi et al., 2025), encompassing both supervised fine-tuning (SFT) and reinforcement learning with verifiable rewards (RLVR). In both settings, the learner has access to a class of models that generate reasoning traces token-by-token, and an *outcome verifier* (e.g., a unit test for code, or an answer checker for math) that checks whether the final answer is correct.

1.1. Contributions

We establish that autocurriculum, using the verifier to adaptively select which prompts to focus training on, provably and dramatically reduces the cost of training, yielding up to *exponential* improvements over non-adaptive approaches.

Section 3: Supervised fine-tuning. In this setting, the learner has interactive access to an expert that generates perfect reasoning traces, capturing SFT and distillation scenarios (Shao et al., 2024; Abdin et al., 2025; Olmo et al., 2025). We design `AutoTune` (Autocurriculum Fine-Tuning), an algorithm inspired by classical boosting (Freund, 1995; Schapire and Freund, 2013) and learning from counterexamples (Angluin, 1987; Gluch and Urbanke, 2021; Angluin and Dohrn, 2017) that iteratively queries for reasoning traces on prompts where the current model errs. We show that this learner is *exponentially* more sample-efficient than the non-adaptive approach of Joshi et al. (2025), reducing the number of reasoning demonstrations from $\widetilde{\Theta}(d/\varepsilon)$ to $\widetilde{O}(d)$, where d measures the complexity of the model class and $1-\varepsilon$ is the target accuracy. In other words, the number of teacher demonstrations becomes nearly independent of the target accuracy. Perhaps surprisingly, this holds without distributional assumptions on the prompt space or hypothesis class—unlike classical active learning, where logarithmic label complexity requires restrictive structural conditions.

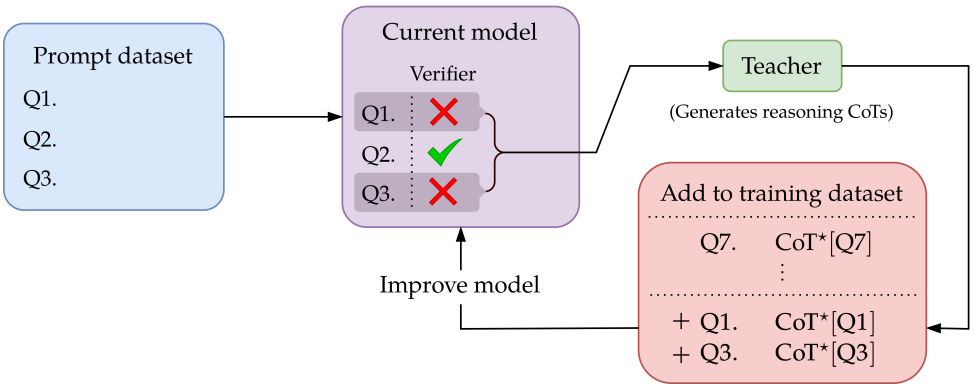


Figure 1: An example of autocurriculum for supervised fine-tuning: the learner chooses which prompts to receive teacher CoTs on, based on its accuracy. In each iteration, the learner's model is updated to digest the new supervision and improve its accuracy.

Setting	No curriculum	Autocurriculum
SFT	$n_{\texttt{prompt}} = n_{\texttt{CoT}} \sim \frac{\log(\lvert\Pi\rvert)}{\varepsilon}$	$(n_{\texttt{prompt}}, n_{\texttt{CoT}}) \sim \left(\frac{\log(\lvert\Pi\rvert)}{\varepsilon}, \log(\lvert\Pi\rvert)\log(1/\varepsilon)\right)$
RLVR	$n_{\texttt{comp}} \sim \frac{C_{\mathrm{seq}}\log(\lvert\Pi\rvert)}{\varepsilon}$	$n_{\texttt{comp}} \sim C_{\mathrm{seq}}\log(\lvert\Pi\rvert)\log(1/\varepsilon) + \frac{\log(\lvert\Pi\rvert)}{\varepsilon}$

Table 1: "$\sim$" refers to equivalence up to $\mathrm{polylog}(1/\varepsilon, 1/\delta, d, C_{\mathrm{seq}})$ factors, Π is the class of next-token models, and C_{seq} is the sequence-level coverage of the reference model in the RLVR setting (Theorem 13). Entries in the table describe the complexity of learning a model $\widehat{\pi}$ such that $\texttt{Acc}_\rho(\widehat{\pi}) \geqslant 1-\varepsilon$ when Π is finite and deterministic. $n_{\texttt{prompt}}$ denotes the number of prompts, $n_{\texttt{CoT}} \leqslant n_{\texttt{prompt}}$ is the number of CoTs queried from an expert demonstrator; for RLVR, $n_{\texttt{comp}}$ is the number of model rollouts during training.

Section 4: Fine-tuning a reference model. Our second setting captures reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2024). Here, the learner starts with a reference model and must guide it toward higher accuracy through interaction with the outcome verifier. Under a natural coverage assumption on the reference model (Zhu et al., 2023; Song et al., 2024), we show that autocurriculum *decouples* the dependence on coverage from the target accuracy, reducing the computational cost from $\widetilde{O}(dC_{\mathrm{seq}}/\varepsilon)$ to $\widetilde{O}(dC_{\mathrm{seq}} + d/\varepsilon)$, where C_{seq} measures how well the reference model covers the correct reasoning traces. The $\widetilde{O}(dC_{\mathrm{seq}})$ term reflects a coverage dependent startup cost that grows negligibly with the target accuracy; beyond this burn-in, the $\widetilde{O}(d/\varepsilon)$ cost for improving accuracy matches the cost of a reference model with perfect coverage.

In both settings, no assumption on the distribution or difficulty of prompts is required: the autocurriculum emerges entirely from the model's own training dynamics, complementing the more widely studied data-centric approaches in modern pipelines.

1.2. Organization

In Section 2 we introduce preliminaries, in Section 3 we introduce some prior results on CoT learning with distillation feedback and describe how adaptive queries can significantly reduce labeling costs, focusing on the setting of deterministic models. In Section 3.4 we extend these results to the setting where the learner operates with general models. In Section 4 we consider the RLVR setting where the learner has access to a reference model and show how autocurriculum reduces the *computational cost* of learning good policies.

2. Preliminaries

We introduce the formal setup in three parts: the autoregressive language modeling framework, the notion of accuracy with respect to an outcome verifier, and the SFT and RL settings.

Basic notation. For nonnegative quantities a, b, $a \lesssim b$ (resp. $a \gtrsim b$) if there exists a universal constant $C > 0$ such that $a \leqslant Cb$ (resp. $a \geqslant Cb$) and $a \asymp b$ if both $a \lesssim b$ and $a \gtrsim b$ hold. We use standard Big-Oh notation: $a = O(b)$ means $|a| \leqslant Cb$ for a universal constant C, and $a = \Omega(b)$ denotes the reverse inequality; we write $a = \Theta(b)$ when both bounds hold. Finally, we use the $\|$ delimiter to separate core arguments from noteworthy hyperparameters, e.g., $\mathsf{Alg}(\cdots \| \cdots)$.

Language models. Let Σ denote a finite token space and Σ^* denote the collection of (possibly infinite length) strings constructed from this vocabulary. We consider prompted language models, where $\mathcal{X} \subseteq \Sigma^*$ denotes an abstract prompt space, and $\rho \in \Delta_{\mathcal{X}}$ is a target distribution over prompts to learn under. Let $\pi : \Sigma^* \to \Delta_\Sigma$ denote a *next-token predictor model*, which takes in a prefix of tokens and returns a next-token distribution. We consider learning in the *parameter-sharing regime*: a single model π is shared across every position in the sequence (as in standard autoregressive decoding). Generating autoregressively from π induces a distribution over sequences, i.e., a chain-of-thought.

Definition 1 (Chain-of-thought (CoT) distribution) *For a model π, the chain-of-thought distribution, $\pi_{1:T}$, is a conditional distribution over length-T sequences: for a prompt $\mathbf{x} \in \mathcal{X}$, $\pi_{1:T}(\cdot|\mathbf{x})$ is the distribution over $(y_1, \cdots, y_T) \in \Sigma^T$ obtained by the iterative process: $\forall t \geqslant 1$, $y_t \sim \pi(\cdot|\mathbf{x}, y_1, \cdots, y_{t-1})$. $\pi_{1:T}$ is supported on the space of responses, $\mathcal{Y} = \Sigma^T$.*

In this abstraction of language model reasoning, we assume that the final token generated in the chain-of-thought corresponds to the "answer" of the model to a given prompt. Thus, we also introduce notation for the distribution over the final token generated by the model.

Definition 2 (Outcome distribution) *Given a prompt $\mathbf{x} \in \mathcal{X}$ and a model π, let $\pi_T(\cdot|\mathbf{x})$ denote the $\mathbf{x}$-conditional distribution of y_T for $(y_1, \cdots, y_T) \sim \pi_{1:T}(\cdot|\mathbf{x})$, supported on Σ.*

Accuracy and verification. The learner's objective is to return a model that correctly predicts the final answer with high probability. To measure correctness, we assume access to an *outcome verifier* $\mathcal{V}$ (e.g., a unit test or answer checker) that determines whether a given answer is correct for a given prompt.

Definition 3 (Outcome verifier) *An outcome verifier is a function $\mathcal{V} : \mathcal{X} \times \Sigma \to \{0, 1\}$ which takes a prompt $\mathbf{x} \in \mathcal{X}$ and a guess for the final answer $y \in \Sigma$, and returns 1 if and only if y is a correct answer for $\mathbf{x}$.*

Given a model π, its accuracy with respect to the outcome verifier $\mathcal{V}$ is defined as the average accuracy across prompts in predicting the final answer correctly,

$$\mathtt{Acc}_\rho(\pi_{1:T}\|\mathcal{V}) \triangleq \mathbb{E}_{\mathbf{x}\sim\rho}\big[\mathtt{Acc}_{\mathbf{x}}(\pi_{1:T}\|\mathcal{V})\big], \text{ where } \mathtt{Acc}_{\mathbf{x}}(\pi_{1:T}\|\mathcal{V}) = \mathbb{E}_{\mathbf{y}\sim\pi_{1:T}(\cdot|\mathbf{x})}\big[\mathcal{V}(\mathbf{x}, y_T)\big]. \quad (1)$$

When $\mathcal{V}$ and T are clear from context, we abbreviate $\mathtt{Acc}_\rho(\pi_{1:T}\|\mathcal{V})$ as $\mathtt{Acc}_\rho(\pi)$ (and similarly $\mathtt{Acc}_{\mathbf{x}}(\pi_{1:T}\|\mathcal{V})$ as $\mathtt{Acc}_{\mathbf{x}}(\pi)$).

Throughout the paper, we consider a realizable setting where the learner has access to a model class Π containing a model $\pi^\star$ which *achieves perfect accuracy*.

Assumption 2.1 (Optimal model is realizable) *There exists $\pi^\star \in \Pi$ such that $\mathtt{Acc}_\rho(\pi^\star) = 1$.*

Learning settings. We consider two learning settings. In the SFT setting, the learner can query a teacher for reasoning traces on chosen prompts. In the RLVR setting, the learner instead has access to a pre-trained reference model that can generate candidate traces, and must improve it using verifier feedback.

Definition 4 (Learning settings) *Consider a dataset of prompts, $D = \{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \sim \rho$. A learning algorithm $\mathit{Alg}_{\mathcal{Q}}(D\|\varepsilon, \delta, T)$ returns a model $\widehat{\pi}$ such that as long as $n \geqslant n_{\mathtt{prompt}}(\varepsilon, \delta, T)$, w.p. at least $1-\delta$,*

$$\mathtt{Acc}_\rho(\widehat{\pi}) \geqslant 1 - \varepsilon.$$

$n_{\mathtt{prompt}}(\varepsilon, \delta, T)$ is referred to as the sample complexity of the learning algorithm. The subscript $\mathcal{Q} \in \{\mathtt{SFT}, \mathtt{RL}\}$ indicates the learning setting and how costs are measured.

- *SFT setting. For $\mathbf{x} \in \mathcal{X}$, a CoT oracle $\mathtt{CoT}(\mathbf{x})$ returns $\mathbf{y} \sim \pi^\star_{1:T}(\cdot|\mathbf{x})$. In the SFT setting, we assume that the learner has query access to $\mathtt{CoT}(\cdot)$. The query complexity, denoted $n_{\mathtt{CoT}}(\varepsilon, \delta, T)$, measures the number of queries to $\mathtt{CoT}(\cdot)$ made by $\mathit{Alg}_{\mathtt{SFT}}(\cdot\|\varepsilon, \delta, T)$.*
- *RL setting. A reference model is a conditional distribution $\pi_{\mathrm{ref}} : \mathcal{X} \to \Delta_{\Sigma^T}$. In the RL setting, we assume query access to it: for $\mathbf{x} \in \mathcal{X}$, we obtain a response $\Sigma^T \ni \mathbf{y} \sim \pi_{\mathrm{ref}}(\cdot|\mathbf{x})$ as well as its probability $\pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x})$. The computational cost of the learner, denoted $n_{\mathtt{comp}}(\varepsilon, \delta, T)$, measures the total number of length-T responses generated from π_{ref} or any other model in Π during the execution of $\mathit{Alg}_{\mathtt{RL}}(\cdot\|\varepsilon, \delta, T)$.*

In both settings, the learner also has query access to the outcome verifier $\mathcal{V}$.

3. SFT: Fine-Tuning with Teacher Supervision

We begin by reviewing prior results on SFT without curriculum (Section 3.1), then motivate and formalize the autocurriculum problem (Section 3.2). Our main result, an exponential improvement in the number of CoT demonstrations for deterministic models, is in Section 3.3; the extension to general models is in Section 3.4.

3.1. Prior Work: SFT without Curriculum

The simplest approach to SFT is to collect CoT demonstrations from the teacher on every training prompt, then fit the model via next-token prediction on the resulting dataset. Joshi et al. (2025) and Foster et al. (2024) analyze this approach and show that it is statistically efficient: treating n reasoning traces as nT next-token examples and minimizing empirical risk yields a model with high accuracy. The following proposition, which is a corollary of their results, summarizes the baseline sample complexity.

Proposition 5 (Corollary of Foster et al. (2024); Joshi et al. (2025)) *Let next-token prediction,* NTP, *denote the CoT-supervised learning algorithm which takes a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ of CoTs with $\mathbf{x}_i \sim \rho$ and $\mathbf{y}_i \sim \pi^\star_{1:T}(\cdot|\mathbf{x}_i)$, and returning the model,*

$$\pi^{\mathsf{NTP}} \in \arg\min_{\pi \in \Pi} \frac{1}{nT} \sum_{i=1}^{n} \sum_{t=1}^{T} \ell\big(\pi(\cdot|\mathbf{x}_i, \mathbf{y}_{i,1:t-1}), y_{i,t}\big)$$

Under Assumption 2.1, with ℓ as the log-loss ($\ell(\pi, a) \equiv \log(1/\pi(a))$), NTP *has sample and query complexity,*

$$n_{\mathsf{prompt}}(\varepsilon, \delta, T) = n_{\mathsf{CoT}}(\varepsilon, \delta, T) \lesssim \frac{\log(|\Pi|/\delta)\log(1/\varepsilon)}{\varepsilon^2}$$

Furthermore, when Π is a (potentially unbounded) family of deterministic models, with ℓ as the 0-1 *loss, ($\ell(\pi, a) \equiv \mathbb{E}_{a' \sim \pi}[\mathbb{I}(a' \neq a)]$), under Assumption 2.1, the sample and query complexity of* NTP *is upper bounded by,*

$$n_{\mathsf{prompt}}(\varepsilon, \delta, T) = n_{\mathsf{CoT}}(\varepsilon, \delta, T) \lesssim \frac{d \cdot \log(dT|\Sigma|)\log(1/\varepsilon) + \log(1/\delta)}{\varepsilon} \tag{2}$$

where $d = \mathrm{Ndim}(\Pi) \leqslant \log_2(|\Pi|)$ is the Natarajan dimension of Π (Natarajan, 1989).

3.2. Reducing the Cost of Supervision: Autocurriculum

The baseline approach in Theorem 5 collects CoT demonstrations on *every* training prompt. But this is wasteful: many prompts may already be easy for the current model, and collecting expensive reasoning traces on them provides little benefit. A natural idea is to let the learner *choose* which prompts to query for CoTs, focusing supervision on the prompts where it is most needed. To formalize this, we assume that in addition to the CoT oracle, the learner has access to the outcome verifier $\mathcal{V}$, which can cheaply evaluate whether the model's current answer is correct.

Problem 6 (Autocurriculum for SFT) *The learner receives a dataset of n prompts $D = \{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i \overset{i.i.d.}{\sim} \rho$ and has query access to the outcome verifier $\mathcal{V}$ (Theorem 3), and to CoT supervision oracle $\mathsf{CoT} : \mathcal{X} \to \Delta_{\Sigma^T}$ (Theorem 4). The objective is to return a model $\widehat{\pi}$ such that $\mathsf{Acc}_\rho(\widehat{\pi}) \geqslant 1 - \varepsilon$.*

Note that this setting is *not* a special case of active learning: the verifier provides a cheap source of feedback (is the current model correct on this prompt?) that is distinct from the expensive CoT supervision. This is what enables the exponential savings we establish below.

3.3. Main Result: Exponential Improvement in CoT Supervision via Autocurriculum

We first specialize to deterministic models, i.e., each $\pi \in \Pi$ maps each token prefix to a single token. While language models trained in practice are usually not deterministic, we consider this setting as a warmup for the general case discussed in Section 3.4. The deterministic setting is also closely tied to problems like semiautomaton learning (Giapitzakis et al., 2025) which have received attention recently. We also assume that each prompt has a unique correct answer (Assumption 3.1 below), whereby accuracy reduces to the probability of matching the teacher's answer[1]

$$\mathsf{Acc}_\rho(\pi) = \mathbb{E}_{\mathbf{x}\sim\rho}\big[\mathbb{I}\big(\pi_T(\mathbf{x}) = \pi^\star_T(\mathbf{x})\big)\big]. \tag{3}$$

Assumption 3.1 (Unique answers (sparse verifier)) *The verifier $\mathcal{V}$ is supported on a unique correct answer for each prompt $\mathbf{x} \in \mathcal{X}$: $\{y \in \Sigma : \mathcal{V}(\mathbf{x}, y) = 1\}$ is a singleton set. Under Assumption 2.1, this implies that $\pi^\star_T(\cdot|\mathbf{x})$ must be supported only on the corresponding token in this set, almost surely over $\mathbf{x} \sim \rho$.*

Algorithm idea. Our algorithm, AutoTune (Algorithm 1), is inspired by classical boosting (Freund, 1995; Schapire and Freund, 2013): it iteratively trains an ensemble of models, where each new model is trained using next-token prediction (NTP) on prompts that the current ensemble gets wrong. Crucially, the verifier determines which prompts to focus on, while CoT demonstrations are only collected for those prompts. Since each new model fixes a constant fraction of the remaining errors, $k = \mathcal{O}(\log(1/\varepsilon))$ rounds suffice to reach accuracy $1 - \varepsilon$, granting an *exponential* improvement in the number of CoT demonstrations required over the non-adaptive baseline. The resulting algorithm is improper, inducing an "outcome-level" model $\widehat{f} : \mathcal{X} \to \Delta_\Sigma$ obtained by aggregating the answers of the models in the ensemble.

Algorithm 1 $\mathtt{AutoTune}\big(D_{\mathsf{prompt}} \big\| \mathsf{Alg}_{\mathcal{Q}}, \varepsilon, \delta, T\big)$

```
# Supervised fine-tuning of deterministic policies with autocurriculum.
Input: Class of models Π; target accuracy 1 − ε and failure prob. δ; reasoning steps T;
       Prompt dataset D_prompt = {x_i}_{i=1}^n where x_i ~ ρ;
       Outcome verifier V : X × Σ → {0, 1};
       Base learning algorithm Alg_Q(· ‖ ε', δ', T).
Initialize: err_⋆ ← 1/4 and k ← ⌈C log(1/ε)/err_⋆^2⌉ for a large absolute constant C > 0.
Split D_prompt into k equal parts {D^j_prompt : 0 ⩽ j ⩽ k − 1}
Π_0 ← ∅
for phase j ← 0 to k − 1 do
    D^j_out ← Sample(D^j_prompt ‖ Π_j, k)        ▶ Sample(·) (Algorithm 2) induces target learning
    π̂^j_T ← Alg_Q(D^j_out ‖ err_⋆, δ/k, T)          distribution via rejection sampling from ρ.
    Π_{j+1} ← Π_j ∪ {π̂^j}
end
return Plu({π̂_T : π̂ ∈ Π_k})
```

1. We overload notation so that for any $\pi \in \Pi$, $\pi_T(\mathbf{x})$ denotes the token $\pi_T(\cdot|\mathbf{x})$ is supported on.

Theorem 7 (Exponential improvement via autocurriculum for SFT) *Consider any $\delta \in (0, 1/2)$ and $\varepsilon \in (0, 1)$. Suppose Π is composed of deterministic models, the verifier is sparse (Assumption 3.1) and that the optimal model is realizable (Assumption 2.1). Let* AutoTune *(Algorithm 1) be instantiated with the base learner $\mathsf{Alg}_{\mathsf{SFT}}(\cdot \| \varepsilon, \delta, T)$ from Theorem 5. If the size of the prompt dataset input to* AutoTune *satisfies,*

$$n \geqslant \frac{d}{\varepsilon} \cdot \mathrm{polylog}(\varepsilon^{-1}, \delta^{-1}, T, |\Sigma|). \tag{4}$$

where $d = \mathrm{Ndim}(\Pi)$. Then, the number of times AutoTune *queries the verifier is upper bounded by n, and the number of queries to the CoT oracle* CoT$(\cdot)$ *is upper bounded by,*

$$n_{\mathsf{CoT}} \leqslant d \cdot \mathrm{polylog}(\varepsilon^{-1}, \delta^{-1}, T, |\Sigma|)$$

and the resulting outcome-level model $\widehat{f}$ satisfies with probability at least $1 - \delta$, $\mathtt{Acc}_\rho(\widehat{f}) \geqslant 1 - \varepsilon$.

Proof The proof is deferred to Section B.1. ■

Comparisons to baseline algorithms. *CoT feedback without autocurriculum.* As discussed in Section 3.1, Joshi et al. (2025) show that given an i.i.d. dataset of $\widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right)$ prompts labeled by CoTs from $\pi^\star$,[2] next-token prediction (NTP) can be used to train a model to accuracy $1 - \varepsilon$, and that this is optimal in the worst case. In Theorem 7 we show that it is possible to achieve accuracy $1 - \varepsilon$ querying only $\widetilde{\mathcal{O}}(d)$ CoTs from $\pi^\star$, using autocurriculum, improving query complexity exponentially.
End-to-end feedback. Joshi et al. (2025) also consider an "end-to-end" learning setting where the learner is given a dataset of n prompts labeled only with the final answer, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \sim \rho$ and $y_i \sim \pi^\star_T(\cdot|\mathbf{x}_i)$, and aims to learn a model with high accuracy. Here the authors show strong statistical and computational lower bounds for learning with this feedback: the worst-case sample complexity degrades to $\widetilde{\Omega}\left(\frac{dT}{\varepsilon}\right)$, compared to the setting where the learner receives prompts labeled with full CoTs (cf. Eq. (2)). While this can be attributed to the fact that CoTs are more informative, there are also computational barriers: learning a model that is much better than random guessing, from end-to-end feedback, is shown to be computationally intractable.

In contrast, Theorem 7 shows that even assuming a weaker form of supervision than end-to-end feedback (namely, access to an outcome verifier), a small amount of carefully selected CoT feedback is sufficient to restore computational tractability (Algorithm 1 is efficient with respect to an ERM oracle for Π under the 0-1 loss). Furthermore, from a statistical point of view, the prompt dataset size required by AutoTune to achieve accuracy $1 - \varepsilon$ (cf. Eq. (4)) returns to $\widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right)$.

Taken together, these results show that autocurriculum enables best-of-both-worlds guarantees: the query complexity of autocurriculum improves exponentially over learning from CoTs alone, while simultaneously avoiding the statistical and computational barriers that are inherent to learning from end-to-end feedback.

Remark 8 (Comparison to active learning) *In the active learning literature, $\log(1/\varepsilon)$-style label complexity bounds for classification are achievable under distributional assumptions such as bounded star number (Hanneke and Yang, 2015) or bounded disagreement coefficient (Hanneke, 2007)*

2. $d = \mathrm{Ndim}(\Pi) \leqslant \log_2(|\Pi|)$ denotes the Natarajan dimension of Π.

Algorithm 2 $\texttt{Sample}(D\|\Pi_j, k)$

```
# Subsampling prompts via rejection sampling.
Input: Dataset of prompts, D = {x_i}_{i=1}^n
       Set of j models Π_j;
       Outcome verifier V : X × Σ → {0,1};
       Base learning algorithm Alg_Q(·‖ε, δ, T) with sample complexity n_prompt(ε, δ, T).
Initialize: err_⋆ ← 1/4 and D_out ← ∅
for prompts x ∈ D do
    if |D_out| ⩾ n_prompt(err_⋆, δ/k, T) then      ▶ If D_out is sufficiently large that Alg_Q can train a
        Return: D_out                                  model with constant accuracy, terminate loop.
    end
    Draw η ∼ Unif([0,1]).
    if η ⩽ w_j(x)/‖w_j‖_∞ then                     ▶ w_j and ‖w_j‖_∞ are defined in Eq. (5).
        D_out ← D_out ∪ {x}
    end
end
Return: D_out
```

For $0 \leqslant r \leqslant j < k$, w_j is defined as $w_j(\mathbf{x}) = \alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}$ and $\|w_j\|_\infty = \max_{0\leqslant r\leqslant j} \alpha^{j,k}_r$, where,

$$\alpha^{j,k}_r = \beta^{j+1,k}_r - \beta^{j+1,k}_{r+1}, \text{ and, } \beta^{j,k}_r = \begin{cases} \mathbb{I}(r \leqslant k/2) & \text{if } j = k \\ \texttt{err}_\star \cdot \beta^{j+1,k}_r + (1 - \texttt{err}_\star) \cdot \beta^{j+1,k}_{r+1} & \text{if } j < k \end{cases} \tag{5}$$

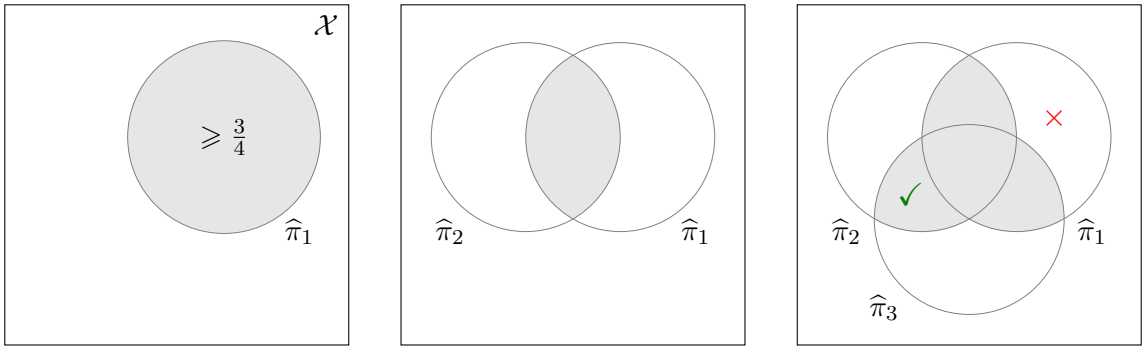
$\mathrm{rank}_j(\mathbf{x})$ counts the number of models in Π_j which guess the label on $\mathbf{x}$ correctly: for $j \geqslant 0$,

$$\mathrm{rank}_j(\mathbf{x}) = \sum\nolimits_{\pi\in\Pi_j} \mathcal{V}(\mathbf{x}, \pi_T(\mathbf{x})) \in [0, j]. \tag{6}$$

on the hypothesis class. In contrast, the CoT query complexity for AutoTune *in Theorem 7 has polylogarithmic dependence in $1/\varepsilon$ without any such assumptions on Π or the induced end-to-end class $\Pi_T = \{\pi_T : \pi \in \Pi\}$. This highlights that Problem 6 is not a special case of active learning: the learner can extract information about $\pi^\star$ by querying the verifier, which is sufficiently informative to circumvent active learning lower bounds.*

Remark 9 (The role of outcome-level accuracy) *The bound on the number of CoTs queried by Algorithm 1 in Theorem 7 (or any algorithm, for that matter) is only achievable when the accuracy $\mathsf{Acc}_\rho(\widehat{\pi})$ of a model $\widehat{\pi}$ is measured as in Eq. (3) as the correctness in predicting the final token $\pi^\star_T(\mathbf{x})$ for $\mathbf{x} \sim \rho$. In particular, if the accuracy of $\widehat{\pi}$ is instead measured by its correctness on the full CoT sequence, $\mathbb{E}_{\mathbf{x}\sim\rho}\big[\mathbb{E}_{\mathbf{y}\sim\widehat{\pi}_{1:T}(\cdot|\mathbf{x})}\big[\mathbb{I}\big(\mathbf{y} = \pi^\star_{1:T}(\mathbf{x})\big)\big]\big]$, the bound degrades to $\widetilde{\Omega}\big(\frac{d}{\varepsilon}\big)$ CoT queries. The outcome verifier cannot provide any supervision for the intermediate tokens in the teacher's CoT.*

Proof sketch for Theorem 7. The high-level idea is illustrated in Figure 2. In each of the $k = \mathcal{O}(\log(1/\varepsilon))$ phases, a new model is trained to constant accuracy on a reweighted distribution over prompts that upweights the regions where the current ensemble errs. The $\texttt{Sample}(\cdot)$ subroutine



(a) $\widehat{\pi}_1$ trained to accuracy at least $\frac{3}{4}$.

(b) Model $\widehat{\pi}_2$ is trained subsequently to accuracy $\frac{3}{4}$.

(c) $\widehat{\pi}_3$ fixes labels on some incorrectly labeled prompts.

Figure 2: An illustration of how models trained on the appropriate prompt distributions can correct errors of prior ones. The region marked in gray captures the region correctly labeled by the plurality of the ensemble of models. Comparing (a) to (c), the model $\widehat{\pi}_3$ corrects some errors made by $\widehat{\pi}_1$ (green checked region), but also introduces new errors (red crossed region). When trained under the appropriate autocurriculum, the accuracy of the ensemble improves geometrically toward 1.

(Algorithm 2) implements this reweighting using the verifier. Each new model is trained to be a *weak learner*, fixing a constant fraction of remaining errors. Thereby, the ensemble's accuracy approaches 1 at a geometric rate. A subtle point is that weak models trained to constant accuracy may introduce new errors when added to the ensemble. The key argument in the proof is that the reweighting ensures each new model fixes mistakes on a larger measure of prompts than it introduces errors on. *Distributions induced by* $\texttt{Sample}(\cdot)$. The distribution over prompts induced by $\texttt{Sample}(\cdot)$ in iteration j, $\rho_j^\star$, can be written down in terms of a reweighting function w_j applied to the original prompt distribution ρ,

$$\rho_j^\star(\cdot) \propto \rho(\cdot) w_j(\cdot), \text{ where } w_j(\mathbf{x}) = \alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})} \text{ and } \|w_j\|_\infty = \max_{0 \leqslant r \leqslant j} \alpha_r^{j,k} \tag{7}$$

where $\alpha_r^{j,k}$ and rank_j are defined in Eq. (5) and Eq. (6) respectively. The choice of this weight function is inspired from the boosting-by-filtering approach (Freund, 1995). $\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}$ has a natural interpretation: it captures the probability that more than half of the models in the final ensemble predict the correct label on $\mathbf{x}$, given that $\mathrm{rank}_j(\mathbf{x})$ among the first j predict the correct label, and assuming that every future model is independently correct on $\mathbf{x}$ with probability $1 - \texttt{err}_\star = \frac{3}{4}$.

3.4. Extension to General Models

We now extend our results to the case where Π contains stochastic models, which more accurately models the language modeling setting. We show that autocurriculum yields significant improvements in this setting as well, provided that the correctness guarantee is relaxed from *high accuracy* to *moderate accuracy on most prompts*. This can be interpreted as a bound on the model's *pass@k rate* (Chen, 2021): the probability of a positively rewarded attempt among k independently generated answers. Furthermore, these guarantees can be *sharpened* to high-accuracy ones at inference time when prompts have unique final correct answers.

Theorem 10 (Autocurriculum for SFT with general models) *Consider any $\delta \in (0, 1/2)$ and $\varepsilon \in (0, 1)$ and assume that the optimal model is realizable (Assumption 2.1). Let* AutoTune *(Algorithm 1) use the base algorithm $\mathsf{Alg}_{\mathsf{SFT}}$ as the learner from Theorem 5. Suppose the size of the prompt dataset satisfies, $n \geqslant \frac{\log(|\Pi|)}{\varepsilon} \cdot \mathrm{polylog}(\varepsilon^{-1}, \delta^{-1}, T)$. Then,* AutoTune *queries the oracle* $\mathsf{CoT}(\cdot)$ *at most,*

$$n_{\mathsf{CoT}} \leqslant \log(|\Pi|) \cdot \mathrm{polylog}(\varepsilon^{-1}, \delta^{-1}, T)$$

times to return an outcome-level model $\widehat{f}$ such that with probability $1-\delta$, $\Pr_{\mathbf{x}\sim\rho}\big(\mathsf{Acc}_{\mathbf{x}}(\widehat{f}) \geqslant \frac{3}{5}\big) \geqslant 1-\varepsilon$. Finally, AutoTune *is implementable with $\mathcal{O}(\log(1/\varepsilon))$ calls to a log-loss ERM oracle for Π.*

Proof The proof is deferred to Section B.2. ■

For the case of deterministic model classes considered earlier, Algorithm 1 constructs an ensemble of models such that at the end of training, on a $1-\varepsilon$ mass of prompts, over half of the models guess the label correctly. The proof follows a similar strategy to the deterministic case, except that the 0-1 loss is replaced by

$$\ell_{\mathbf{x}}(\pi) = \mathbb{I}\Big(\mathsf{Acc}_{\mathbf{x}}(\pi) \geqslant \frac{4}{5}\Big) \tag{8}$$

We defer a detailed discussion to the appendix, where we address the technical complication that Eq. (8) cannot be computed exactly when the model π is stochastic. This changes how the weak learning distributions are defined.

Remark 11 (Sharpening by consensus vote) *Assuming that the answers to prompts are unique (Assumption 3.1), it is possible to improve the guarantee of Theorem 10 to a high accuracy one by consensus vote. The resulting algorithm,* $\mathsf{AutoTune}_{\mathsf{cons}}$ *(inducing the outcome-level model $\widehat{f}_{\mathsf{cons}}$), is defined as follows: for $\mathbf{x} \in \mathcal{X}$, $\widehat{f}_{\mathsf{cons}}(\cdot|\mathbf{x})$ samples an answer by computing $\mathsf{Plu}\big(\{y^1, \cdots, y^N\}\big)$ for N independent answers, $y^i \sim \widehat{f}(\cdot|\mathbf{x})$. For suitably large $N = \Omega(\log(1/\varepsilon))$, the outcome-level model returned by* $\mathsf{AutoTune}_{\mathsf{cons}}$ *satisfies the guarantee,* $\mathsf{Acc}_{\rho}(\widehat{f}_{\mathsf{cons}}) \geqslant 1 - \mathcal{O}(\varepsilon)$.

4. RL: Improving a Reference Model with Verifier Guidance

In the previous section, the learner had access to an optimal teacher. We now turn to the more practical RLVR setting (Lambert et al., 2024), where the learner starts with a pre-trained reference model and must improve it using only verifier feedback. Since generation costs dominate in practice (Zhou et al., 2025), we measure computational cost by the total number of reasoning traces generated during training. We first formalize the problem and introduce a natural coverage assumption on the reference model, then establish a baseline showing that learning is tractable via rejection sampling, and finally show that autocurriculum decouples the dependence on coverage from accuracy.

Problem 12 (Fine-tuning a reference model) *The learner receives a dataset of n prompts $D = \{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i \overset{\text{i.i.d.}}{\sim} \rho$ and has query access to the outcome verifier $\mathcal{V}$, and to a reference model $\pi_{\mathrm{ref}} : \mathcal{X} \to \Delta_{\Sigma^T}$. The objective is to return a model $\widehat{\pi}$ such that $\mathsf{Acc}_{\rho}(\widehat{\pi}) \geqslant 1-\varepsilon$.*

The computational cost of the learner is measured as the total number of CoTs generated over the course of training: this includes those from π_{ref}, as well as those from the learner's own models.

Algorithm 3 $\texttt{AutoTune}_{\texttt{stoch}}\big(D_{\texttt{prompt}} \| \mathsf{Alg}_{\texttt{SFT}}, \varepsilon, \delta, T\big)$

Supervised fine-tuning of stochastic policies with autocurriculum.

Input: Class of models Π; target accuracy $1-\varepsilon$ and failure prob. δ; reasoning steps T;
Prompt dataset $D_{\texttt{prompt}} = \{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i \sim \rho$;
Outcome verifier $\mathcal{V} : \mathcal{X} \times \Sigma \to \{0,1\}$;
Base CoT-supervised learning algorithm $\mathsf{Alg}_{\texttt{SFT}}(\cdot \| \varepsilon', \delta', T)$.

Let $\widetilde{\texttt{AutoTune}}(D_{\texttt{prompt}} \| \mathsf{Alg}_{\texttt{SFT}}, \varepsilon, \delta, T)$ be a variant of $\texttt{AutoTune}$ (Algorithm 1) with $\texttt{err}_\star \leftarrow \frac{1}{400}$ and where Algorithm 1 is replaced by:

$$D^j_{\texttt{out}} \leftarrow \widetilde{\texttt{Sample}}\big(D^j_{\texttt{prompt}} \big\| \Pi_j, k\big),$$

$\widetilde{\texttt{Sample}}(D \| \Pi_j, k)$ is a variant of $\texttt{Sample}$ (Algorithm 2) with $\texttt{err}_\star \leftarrow \frac{1}{10}$ and weight function $\widehat{w}_j$ instead of w_j on Algorithm 2 of Algorithm 2. ▶ $\widehat{w}_j$ is defined in Eq. (15) to Eq. (18)

Let $\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}$ be the models trained within $\widetilde{\texttt{AutoTune}}(D_{\texttt{prompt}} \| \mathsf{Alg}_{\texttt{SFT}}, \varepsilon, \delta, T)$

Return: Uniform mixture of outcome-level models, $\frac{1}{k}\sum_{j=0}^{k-1} \widehat{\pi}^j_T$.

In the natural regimes of RL finetuning in practice, one starts from a reference model that is trained to generate correct reasoning traces with non-negligible probability. We formalize this through *sequence-level coverage* (Rashidinejad et al., 2021; Foster et al., 2025), which requires that the reference model places at least C_{seq}^{-1} probability on the correct CoT for each prompt. C_{seq} quantifies how well the reference model "covers" the target behavior; smaller C_{seq} means better coverage.

Definition 13 (Sequence-level coverage (Jiang and Xie, 2025)) *Assume the model class Π is deterministic. A reference model π_{ref} satisfies sequence-level coverage with parameter C_{seq} if,*

$$\pi_{\mathrm{ref}}(\mathbf{y}^\star_{\mathbf{x}} | \mathbf{x}) \geqslant C_{\mathrm{seq}}^{-1}, \tag{9}$$

where, $\mathbf{y}^\star_{\mathbf{x}} = \pi^\star_{1:T}(\mathbf{x})$ denotes the CoT generated by $\pi^\star$ on the prompt $\mathbf{x}$.

4.1. Baseline: Learning via Rejection Sampling

As a baseline, we consider the algorithm $\texttt{RLFineTune}$ (Algorithm 4), which follows a natural approach analogous to filtered SFT: for each prompt, generate multiple candidate CoTs from the reference model, keep only those that produce a correct answer (as judged by the verifier), and train a model on the surviving traces. Under the coverage assumption, the reference model is guaranteed to produce at least one correct trace per prompt with high probability after $\widetilde{\mathcal{O}}(C_{\mathrm{seq}})$ samples.

Proposition 14 (Learning from a reference model with coverage) *Consider any target error $\varepsilon \in (0,1)$ and failure probability $\delta \in (0,1)$. Suppose Π is deterministic, the reference model π_{ref} satisfies sequence-level coverage with parameter C_{seq} (Theorem 13), answers are unique (Assumption 3.1) and that the optimal model is realizable (Assumption 2.1). Consider the model $\widehat{\pi}$ returned by $\texttt{RLFineTune}$ (Algorithm 4). There exists an absolute constant $C_1 > 0$ such that if,*

$$n \geqslant C_1 \frac{d \log(T|\Sigma|C_{\mathrm{seq}}) \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon},$$

then, with probability at least $1-\delta$, $\texttt{Acc}_\rho(\widehat{\pi}) \geqslant 1-\varepsilon$. $\texttt{RLFineTune}$ queries the outcome verifier oracle $\mathcal{V}$, and generates CoTs from π_{ref}, no more than $m = \mathcal{O}(nC_{\mathrm{seq}} \log(nC_{\mathrm{seq}}/\delta))$ times.

Proof The proof of this result is deferred to Section B.3. ■

Proof sketch. `RLFineTune` (Algorithm 4) proceeds by drawing a set of $m = \widetilde{\mathcal{O}}(C_{\mathrm{seq}})$ responses on each prompt $\mathbf{x}_i \in D_{\texttt{prompt}}$ in the dataset, and filtering out the responses which are either, (a) duplicates, (b) have low probability under $\pi_{\mathrm{ref}}(\cdot|\mathbf{x}_i)$, or (c) terminate in an incorrect answer. For any such prompt $\mathbf{x}_i$, as long as m is sufficiently large, with high probability the learner can guarantee that the set of surviving responses, $\widetilde{D}_i$, exactly equals the set, $\mathcal{Y}^\star(\mathbf{x}) = \{\mathbf{y} \in \mathcal{Y} : \mathcal{V}(\mathbf{x}, y_T) = 1 \text{ and } \pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x}) \geqslant C_{\mathrm{seq}}^{-1}\}$. That is, $\mathcal{Y}^\star(\mathbf{x})$ is the set of all responses that have high probability under $\pi_{\mathrm{ref}}(\cdot|\mathbf{x})$ and terminate in a correct answer. Consequently, under the same high probability event, the loss optimized by `RLFineTune` equals,

$$\mathcal{L}(\pi; D_{\texttt{prompt}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\big(\pi_{1:T}(\mathbf{x}_i) \notin \widetilde{D}_i\big) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\big(\pi_{1:T}(\mathbf{x}_i) \notin \mathcal{Y}^\star(\mathbf{x}_i)\big) \tag{10}$$

Now, the key trick to show a uniform convergence bound for this loss is that if for some model $\pi \in \Pi$, we have that $\pi_{1:T}(\mathbf{x}_i) \notin \mathcal{Y}^\star(\mathbf{x}_i)$, then this event is witnessed by the first deviation point t, such that $\pi_{1:t}(\mathbf{x}_i) = \mathbf{y}_{1:t}$ for some $\mathbf{y} \in \mathcal{Y}^\star(\mathbf{x}_i)$, but $\pi_{t+1}(\mathbf{x}_i) \neq y_{t+1}$. Since there can be at most T such first deviation points, and each such deviation is realized by $\{\pi(\mathbf{x}_i, \mathbf{y}_{1:t}) \neq y_{t+1}\}$[3] for $\pi \in \Pi$, a counting argument based on the Sauer-Shelah lemma shows there are at most $(enC_{\mathrm{seq}}T)^d$ possible behaviors Π can express over the set $\big(\mathbb{I}\big(\pi_{1:T}(\mathbf{x}_i) \notin \mathcal{Y}^\star(\mathbf{x}_i)\big) : i \in [n]\big)$. This results in a bound on the growth function of the loss class corresponding to the empirical risk in Eq. (10), and thereby a uniform concentration bound for the same, which can be used to prove the statement of the theorem.

Remark 15 (Partial coverage) *In Section B.3, we prove a more general version of Theorem 14 when π_{ref} satisfies coverage on all but a small mass of prompts (Song et al., 2024; Chen et al., 2025). Namely, for some $\eta > 0$, $\Pr_{\mathbf{x}\sim\rho}\big(\pi_{\mathrm{ref}}(\mathbf{y}^\star_{\mathbf{x}}|\mathbf{x}) \geqslant C_{\mathrm{seq}}^{-1}\big) \geqslant 1 - \eta$ where $\mathbf{y}^\star_{\mathbf{x}} = \pi^\star_{1:T}(\mathbf{x})$. The implications of this guarantee for weaker notions such as L_1-coverage are discussed in the appendix.*

While the computational cost of `RLFineTune` scales linearly with the coverage coefficient as $\widetilde{\mathcal{O}}(nC_{\mathrm{seq}})$, the sample complexity only scales *logarithmically* in it. The former is a necessary cost: the learner must draw $\widetilde{\mathcal{O}}(C_{\mathrm{seq}})$ CoTs from π_{ref} per prompt to see even a single correct one.

4.2. Main Result: Autocurriculum Decouples Coverage from Accuracy

Having established that learning is tractable without a curriculum, we now ask: *can autocurriculum further reduce the computational cost?* We show that the answer is yes. `AutoTune.RL` (Algorithm 5) applies the same boosting-based autocurriculum from Section 3, using `RLFineTune` as the base learner. As in the SFT setting, the verifier identifies prompts where the current ensemble errs, and the base learner is retrained on those prompts. The key difference is that instead of querying a teacher for CoTs, each invocation of `RLFineTune` generates its own traces from π_{ref} via rejection sampling.

Theorem 16 (Autocurriculum decouples coverage from accuracy) *Consider any target error $\varepsilon \in (0, 1)$ and failure probability $\delta \in (0, 1)$. Suppose Π is deterministic, π_{ref} satisfies sequence-level coverage with parameter C_{seq} (Theorem 13), answers are unique (Assumption 3.1) and that*

3. Here, we abuse notation to let $\pi(\mathbf{x}_i, \mathbf{y}_{1:t})$ denote the deterministic next-token π would generate on the input sequence $(\mathbf{x}_i, \mathbf{y}_{1:t})$

Algorithm 4 $\texttt{RLFineTune}(D_{\texttt{prompt}} \| \varepsilon, \delta, T)$

Sharpening a reference model π_{ref} satisfying sequence-level coverage using verifier feedback.

Input: Class of models Π; target accuracy $1-\varepsilon$ and failure prob. δ; reasoning steps T;
Reference model, π_{ref} satisfying C_{seq} sequence-level coverage (Theorem 13);
Prompt dataset, $D_{\texttt{prompt}} = \{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i \sim \rho$;
Outcome verifier oracle $\mathcal{V} : \mathcal{X} \times \Sigma \to \{0,1\}$;

Initialize: $m = C_{\mathrm{seq}} \log(4nC_{\mathrm{seq}}/\delta)$

for $\mathbf{x}_i \in D_{\texttt{prompt}}$ **do**

  Draw m length-T chains-of-thought from $\pi_{\mathrm{ref}}(\cdot|\mathbf{x}_i)$ and deduplicate. Denote this set as D_i.

  Discard low-probability chains-of-thought which lead to incorrect answers,

$$\widetilde{D}_i = \left\{\mathbf{y} \in D_i : \mathcal{V}(\mathbf{x}_i, y_T) = 1 \text{ and } \pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x}_i) \geqslant C_{\mathrm{seq}}^{-1}\right\}$$

end

Return: $\widehat{\pi} \in \operatorname*{argmin}_{\pi \in \Pi} \mathcal{L}(\pi; D_{\texttt{prompt}})$ for $\mathcal{L}(\pi; D_{\texttt{prompt}}) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}\big(\pi_{1:T}(\mathbf{x}_i) \notin \widetilde{D}_i\big)$

Algorithm 5 $\texttt{AutoTune.RL}(D_{\texttt{prompt}} \| \mathsf{Alg}_{\mathsf{RL}}, \varepsilon, \delta, T)$

Input: Class of models Π, reasoning steps T, target failure prob. δ,
Prompt dataset, $D_{\texttt{prompt}} = \{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i \sim \rho$,
Outcome verifier, $\mathcal{V} : \mathcal{X} \times \Sigma \to \{0,1\}$,

Return: $\widehat{f} = \texttt{AutoTune}(D_{\texttt{prompt}} \| \mathsf{Alg}_{\mathsf{RL}}, \varepsilon, \delta, T)$, where,

$$\mathsf{Alg}_{\mathsf{RL}}(\cdot \| \varepsilon', \delta', T) \leftarrow \texttt{RLFineTune}(\cdot \| \varepsilon', \delta', T)$$

the optimal model is realizable (Assumption 2.1). AutoTune.RL *(Algorithm 5) guarantees that as long as the size of the prompt dataset is at least,* $n \geqslant \frac{d}{\varepsilon} \cdot \mathrm{polylog}(C_{\mathrm{seq}}, \varepsilon^{-1}, \delta^{-1}, T)$*, the resulting outcome-level model* $\widehat{f}$ *satisfies* $\texttt{Acc}_\rho(\widehat{f}) \geqslant 1-\varepsilon$ *with probability at least* $1-\delta$*. Furthermore, up to* $\mathrm{polylog}(C_{\mathrm{seq}}, \varepsilon^{-1}, \delta^{-1}, T)$ *factors,*

- *The number of length-T CoTs generated from any model (π_{ref}, or rollouts of the learner's own models during training),* $n_{\texttt{comp}}$ *is at most* $\widetilde{\mathcal{O}}\big(dC_{\mathrm{seq}} + \frac{d}{\varepsilon}\big)$.
- *The number of calls to the outcome verifier is also upper bounded by* $\widetilde{\mathcal{O}}\big(dC_{\mathrm{seq}} + \frac{d}{\varepsilon}\big)$.

Proof The proof of this result is deferred to Section B.4. ■

Compared to RLFineTune, which requires $\widetilde{\mathcal{O}}(dC_{\mathrm{seq}}/\varepsilon)$ total traces, AutoTune.RL reduces this to $\widetilde{\mathcal{O}}(dC_{\mathrm{seq}} + d/\varepsilon)$. The coverage-dependent cost $\widetilde{\mathcal{O}}(dC_{\mathrm{seq}})$ becomes a one-time burn-in that grows only logarithmically with $1/\varepsilon$; as the target accuracy increases, the learner pays as if coverage were $O(1)$. The key reason is that each of the $O(\log(1/\varepsilon))$ weak learners trains on only $\widetilde{\mathcal{O}}(d)$ prompts, so the per-prompt generation cost of $\widetilde{\mathcal{O}}(C_{\mathrm{seq}})$ does not multiply with $1/\varepsilon$. The remaining $\widetilde{\mathcal{O}}(d/\varepsilon)$ cost comes from evaluating the ensemble to route prompts, and is nearly independent of C_{seq}.

References

Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, et al. Phi-4-reasoning technical report. arXiv:2504.21318, 2025.

Dana Angluin. Learning regular sets from queries and counterexamples. Information and Computation, 1987.

Dana Angluin and Tyler Dohrn. The power of random counterexamples. In International Conference on Algorithmic Learning Theory, 2017.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. The Annals of Statistics, 2005.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Proceedings of the 26th International Conference on Machine Learning (ICML), 2009.

Fan Chen, Audrey Huang, Noah Golowich, Sadhika Malladi, Adam Block, Jordan T Ash, Akshay Krishnamurthy, and Dylan J Foster. The coverage principle: How pre-training enables post-training. arXiv preprint arXiv:2510.15020, 2025.

Mark Chen. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In International Conference on Learning Representations (ICLR), 2018. URL https://openreview.net/forum?id=HJewuJWCZ.

Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. Advances in Neural Information Processing Systems, 2024.

Dylan J Foster, Zakaria Mhammedi, and Dhruv Rohatgi. Is a good foundation necessary for efficient reinforcement learning? the computational role of the base model in exploration. arXiv preprint arXiv:2503.07453, 2025.

Yoav Freund. Boosting a weak learning algorithm by majority. Information and Computation, 1995.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1):119–139, 1997.

Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780):1612, 1999.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training. arXiv preprint arXiv:2510.01135, 2025.

George Giapitzakis, Kimon Fountoulakis, Eshaan Nichani, and Jason D Lee. On the statistical query complexity of learning semiautomata: a random walk approach. arXiv preprint arXiv:2510.04115, 2025.

Grzegorz Gluch and Ruediger Urbanke. Exponential separation between two learning models and adversarial robustness. In Advances in Neural Information Processing Systems, 2021.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. arXiv:2506.04178, 2025.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. arXiv preprint arXiv:2308.08998, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. Nature, 2025.

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks, 2019. URL https://arxiv.org/abs/1904.03626.

Steve Hanneke. A bound on the label complexity of agnostic active learning. In Proceedings of the 24th international conference on Machine learning, pages 353–360, 2007.

Steve Hanneke and Liu Yang. Minimax analysis of active learning. J. Mach. Learn. Res., 16(1): 3487–3602, 2015.

David Haussler and Philip M Long. A generalization of sauer's lemma. Journal of Combinatorial Theory, Series A, 71(2):219–240, 1995.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. arXiv preprint arXiv:2503.24290, 2025.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. International Conference on Machine Learning (ICML), 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. arXiv preprint arXiv:2412.16720, 2024.

Nan Jiang and Tengyang Xie. Offline reinforcement learning in large state spaces: Algorithms and guarantees. Statistical Science, 40(4):570–596, 2025.

Nirmit Joshi, Gal Vardi, Adam Block, Surbhi Goel, Zhiyuan Li, Theodor Misiakiewicz, and Nathan Srebro. A theory of learning with autoregressive chain of thought. arXiv:2503.07932, 2025.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. arXiv preprint arXiv:2510.13786, 2025.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In Advances in Neural Information Processing Systems (NeurIPS), 2010.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. arXiv preprint arXiv:2411.15124, 2024.

Nayoung Lee, Ziyang Cai, Avi Schwarzschild, Kangwook Lee, and Dimitris Papailiopoulos. Self-improving transformers overcome easy-to-hard and length generalization challenges. arXiv preprint arXiv:2502.01612, 2025.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, et al. Goedel-prover: A frontier model for open-source automated theorem proving. arXiv preprint arXiv:2502.07640, 2025.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. arXiv preprint arXiv:2505.24864, 2025.

Deyu Meng, Qian Zhao, Lu Jiang, et al. What objective does self-paced learning indeed optimize?, 2015. URL https://arxiv.org/abs/1511.06049.

Sumeet Ramesh Motwani, Alesia Ivanova, Ziyang Cai, Philip Torr, Riashat Islam, Shital Shah, Christian Schroeder de Witt, and Charles London. h1: Bootstrapping llms to reason over longer horizons via reinforcement learning. arXiv preprint arXiv:2510.07312, 2025.

Balas K Natarajan. On learning sets and functions. Machine Learning, 4(1):67–97, 1989.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. arXiv preprint arXiv:2512.13961, 2025.

Gabriel Poesia, David Broman, Nick Haber, and Noah Goodman. Learning formal mathematics from intrinsic motivation. Advances in Neural Information Processing Systems, 37:43032–43057, 2024.

Jatin Prakash and Anirudh Buvanesh. What can you do when you have zero rewards during rl? arXiv preprint arXiv:2510.03971, 2025.

Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. arXiv preprint arXiv:2503.07572, 2025.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. Advances in Neural Information Processing Systems, 34:11702–11716, 2021.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J Foster. Taming imperfect process verifiers: A sampling perspective on backtracking. arXiv preprint arXiv:2510.03149, 2025.

Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. MIT Press, 2013.

Amrith Setlur, Matthew YR Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms. arXiv preprint arXiv:2506.09026, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv:2402.03300, 2024.

Yuda Song, Gokul Swamy, Aarti Singh, J Bagnell, and Wen Sun. The importance of online data: Understanding preference fine-tuning via coverage. Advances in Neural Information Processing Systems, 37:12243–12270, 2024.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers), pages 13484–13508, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35:24824–24837, 2022.

Wei Xiong, Chenlu Ye, Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, Jiang Bian, Nan Jiang, and Tong Zhang. Reinforce-ada: An adaptive sampling framework under non-linear rl objectives. arXiv preprint arXiv:2510.04996, 2025.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In The Twelfth International Conference on Learning Representations, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476, 2025.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, et al. Rlve: Scaling up reinforcement learning for language models with adaptive verifiable environments. arXiv preprint arXiv:2511.07317, 2025.

Yuzhen Zhou, Jiajun Li, Yusheng Su, Gowtham Ramesh, Zilin Zhu, Xiang Long, Chenyang Zhao, Jin Pan, Xiaodong Yu, Ze Wang, Kangrui Du, Jialian Wu, Ximeng Sun, Jiang Liu, Qiaolin Yu, Hao Chen, Zicheng Liu, and Emad Barsoum. April: Active partial rollouts in reinforcement learning to tame long-tail generation, 2025. URL https://arxiv.org/abs/2509.18521.

Banghua Zhu, Michael Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In International Conference on Machine Learning, pages 43037–43067. PMLR, 2023.

Appendices

A Related Work **19**

B Proofs for Main Results **20**
B.1 Autocurriculum for SFT (Deterministic Π): Proof of Theorem 7 20
B.1.1 Proof of Theorem 7 23
B.2 Autocurriculum for SFT (General Π): Proof of Theorem 10 24
B.2.1 Proof of Theorem 10 27
B.3 Improving a Reference Model Satisfying Coverage: Proof of Theorem 14 28
B.3.1 Proof of Theorem 14 30
B.4 Autocurriculum for Fine-Tuning a Reference Model: Proof of Theorem 16 30

C Proofs for Supporting Lemmas **32**
C.1 Proofs for Lemmas from Theorem 7 32
C.1.1 Proof of Theorem 17 32
C.1.2 Proof of Theorem 19 33
C.1.3 Proof of Theorem 20 33
C.2 Proofs for Lemmas from Theorem 10 34
C.2.1 Proof of Theorem 22 34
C.2.2 Proof of Theorem 24 35
C.2.3 Proof of Theorem 25 36
C.2.4 Proof of Theorem 26 36
C.2.5 Proof of Theorem 27 37
C.3 Proofs for Lemmas from Theorem 14 37
C.3.1 Proof of Theorem 32 37

Appendix A. Related Work

Curriculum in deep learning and language model post-training. Curriculum learning formalizes the intuition that presenting training examples in a meaningful order, such as from easy to hard, can improve optimization and generalization (Bengio et al., 2009). Self-paced learning makes this idea algorithmic by alternating between selecting or reweighting examples based on a difficulty proxy and updating parameters (Kumar et al., 2010; Meng et al., 2015; Hacohen and Weinshall, 2019; Fan et al., 2018).

Synthetic data generation and prompt curation. Closely related to approaches like expert iteration (Wang et al., 2023; Gulcehre et al., 2023; Lin et al., 2025), models are trained on synthetically generated data at the cutting edge of their capabilities to optimize training signals (Xu et al., 2024; Motwani et al., 2025; Poesia et al., 2024). In RL settings, curriculum strategies based on prompt curation have been shown to enable models to make progress, *even when the initial pass@k performance on the target task is close to* 0 (Lee et al., 2025; Prakash and Buvanesh, 2025).

Autocurricula: Dynamic data selection and compute allocation. Autocurriculum methods, where sampling distributions are adaptively altered along the course of training, have gained traction in LLM training as a way to improve sample efficiency and stability by allowing models to guide their own data collection. In particular, self-evolving curricula, where the learner prioritizes examples

it finds challenging, have shown substantial improvements in compute efficiency (Yu et al., 2025; Khatri et al., 2025). One line of work focuses on adaptively selecting prompts so that models focus on tasks which are hard, but within the horizon of their capabilities (Gao et al., 2025; Xiong et al., 2025). Of note are approaches based on length-based curricula (Setlur et al., 2025) and adaptive compute allocation (Qu et al., 2025), where models are gradually exposed to longer thinking budgets over the course of training. In other work, adaptive verifiable environments provide an environment-based autocurriculum for scaling RL while maintaining reliable feedback (Zeng et al., 2025). Our results complement these empirical results by providing a theoretical account of how and when adaptive curriculum mechanisms can yield provable reductions in sample and compute complexity.

Theoretical frameworks for reasoning in LLMs. Several recent theory papers provide foundations for analyzing language-model post-training and reasoning. Joshi et al. (2025) analyze learnability in autoregressive models using chain-of-thought structure. In a related line of work, Foster et al. (2024) provide learning-theoretic analyses of imitation learning characterizing the effect of the sequence-length of the problem. In LLM post-training, coverage has emerged as a fundamental quantity characterizing sample and computational complexity. Chen et al. (2025) study the coverage properties that emerge from pre-training language models, while Huang et al. (2025) and Foster et al. (2025) show that it tightly quantifies post-training performance. The coverage of the reference model has also been used in the analysis of various training-time and inference-time algorithms for post-training LLMs (Zhu et al., 2023; Song et al., 2024; Rohatgi et al., 2025).

Boosting and learning from counterexample queries. Our algorithmic approaches connect to classical results for boosting, which formalize how adaptive data collection can hasten learning (Freund, 1995; Freund and Schapire, 1997; Freund et al., 1999). Boosting-by-filtering is an aggregation scheme for classification, where models are trained iteratively on a sequence of evolving distributions to correct the mistakes of the previous ones, in a manner such that the aggregated model has low error. Similarly, our work is also closely connected to the problem of learning from counterexample and equivalence oracles (Angluin, 1987; Angluin and Dohrn, 2017; Gluch and Urbanke, 2021).

Appendix B. Proofs for Main Results

B.1. Autocurriculum for SFT (Deterministic Π): Proof of Theorem 7

In this section, we prove Theorem 7 for the setting where the Π is composed of deterministic models and the outcome verifier is sparse (Assumption 3.1). First we plot the evolution of $\alpha_r^{j,k}$ as a function of r across different values of j.

Proof sketch of Theorem 7. The accuracy of the outcome-level model returned by `AutoTune` can be calculated by considering the probability mass on prompts where more than half the models in the final ensemble $\Pi_k = \{\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}\}$ predict the correct label. We begin with a decomposition of the test-error incurred by Algorithm 1 into terms which depend on the performance of the models trained on samples from $\rho_j^\star$ in each iteration.

Lemma 17 (Test-error decomposition) *Let $\widehat{f}$ denote the outcome-level model returned by Algorithm 1. Then,*

$$\Pr_{\mathbf{x}\sim\rho}\big(\widehat{f}(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\big) \leqslant \beta_0^{0,k} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_{\mathbf{x}\sim\rho}\big[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\big]$$

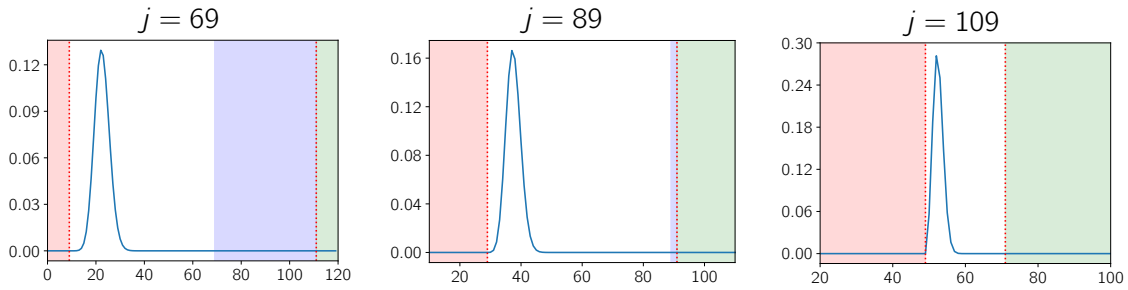


Figure 3: For $k = 120$, we plot of $\alpha_r^{j,k}$ as a function of the rank placeholder r across different values of j (number of models in the current ensemble). The shaded green region captures the values of the rank r (for some prompt $\mathbf{x}$) such that even if the remaining $k - j$ models were to all predict the wrong label on $\mathbf{x}$, the plurality vote remains robustly correct. The shaded red region captures the values of the rank r for which even if the remaining $k - j$ models were to all predict the correct label on $\mathbf{x}$, the plurality vote cannot be guaranteed to be correct. The shaded blue region plots values of the rank which are unattainable (the maximum rank achievable in iteration j is j).

Here, recall $\texttt{err}_\star = \frac{1}{4}$ *and* $\texttt{err}_j = \Pr_{\mathbf{x}\sim\rho_j^\star}\big(\widehat{\pi}_T^j(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\big) = 1 - \texttt{Acc}_{\rho_j^\star}(\widehat{\pi}^j)$.

Proof The formal proof of this result is discussed in Section C.1.1. ■

Theorem 17 decomposes the test error of $\widehat{f}$ into a sum of several terms, and the proof of this result will follow by defining a potential function Φ_j which tracks the performance of the plurality of the outcome-level models, $\{\pi_T : \pi \in \Pi_j\}$ trained until the $(j-1)^{\text{th}}$ phase,

$$\Phi_j = \mathbb{E}_{\mathbf{x}\sim\rho}\big[\beta^{j,k}_{\text{rank}_j(\mathbf{x})}\big]$$

Here, the quantity $\beta_r^{j,k}$ is defined in Eq. (5), while $\text{rank}_j(\mathbf{x})$ is defined in Eq. (6). The term $\beta_0^{0,k}$ appearing in the statement of the above lemma is the initial value of the potential function, Φ_0, while the remaining terms in the summation are precisely the differences $\Phi_j - \Phi_{j-1}$. Note that Φ_k itself takes a simple form, $\mathbb{E}_{\mathbf{x}\sim\rho}\big[\mathbb{I}(\text{rank}_k(\mathbf{x}) \leqslant k/2)\big]$, which is precisely the test error of the plurality of the ensemble induced by the models $\big\{\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}\big\}$, since it captures the event that $k/2$ or fewer models get the final label correct. This clarifies the connection between the Φ_j's and the final prediction error.

Next we discuss how to simplify Theorem 17 further. It is a short calculation to show by writing down an explicit expression for $\Phi_0 = \beta_0^{0,k}$ that this term in fact decays exponentially with k (a fact we prove formally in Theorem 18), and so, as long as $k = \Omega(\log(1/\varepsilon))$, the first term is bounded by $\varepsilon/2$, which is within the target error guarantee of Theorem 7. The remaining terms in Theorem 17 bring out a clean tradeoff to establish: in iterations where generating training examples from $\rho_j^\star$ is easy, $\texttt{err}_j$ is likely to be smaller than the threshold $\texttt{err}_\star = \frac{1}{4}$, resulting in, $(\texttt{err}_j - \texttt{err}_\star) \cdot \mathbb{E}\big[\alpha^{j,k}_{\text{rank}_j(\mathbf{x})}\big] \leqslant 0$. When sampling from $\rho_j^\star$ is hard, we will need to argue that $\alpha^{j,k}_{\text{rank}_j(\mathbf{x})}$ is also likely to be small. The interpretation of such a result would be that, in the iterations where sampling from $\rho_j^\star$ is hard, there is sufficient leeway in the plurality of the existing models that adding an arbitrary $\widehat{\pi}^j$ to the ensemble does not hurt the error significantly.

Toward establishing such a guarantee, we will first define an event which captures whether the dataset collected in iteration j, D^j_{out} is sufficiently large (a proxy for whether training the model in iteration j is feasible),

$$\mathsf{ABORT}[j] = \left\{|D^j_{\mathsf{out}}| < n_{\mathsf{prompt}}(\mathsf{err}_\star, \delta/k, T)\right\} \tag{11}$$

where $n_{\mathsf{prompt}}(\varepsilon, \delta, T)$ is the sample complexity of the base learning algorithm, $\mathsf{Alg}_{\mathsf{CoT}}(\cdot\|\varepsilon, \delta, T)$, in Algorithm 1, which is chosen as the learner from Theorem 5. We will also define,

$$p_j = \mathbb{E}_{\mathbf{x}\sim\rho}\left[\frac{w_j(\mathbf{x})}{\|w_j\|_\infty}\right] = \mathbb{E}_{\mathbf{x}\sim\rho}\left[\frac{\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}}{\alpha^{j,k}_{\max}}\right] \tag{12}$$

which is the probability that the prompt $\mathbf{x}$ accepted into D^j_{out} in Algorithm 2 in the j^{th} iteration. Intuitively, a larger value of p_j means that we will have more prompts to train the model $\widehat{\pi}^j$ on, which itself translates to a smaller probability of this iteration aborting.

Analyzing the test-error decomposition of Theorem 17. Our main argument to analyze the main summation $\sum_{j=0}^{k-1}(\mathsf{err}_j - \mathsf{err}_\star)\cdot\mathbb{E}_{\mathbf{x}\sim\rho}\left[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\right]$ in Theorem 17 will be to show that,

1. If iteration j does not abort, then with high probability, $\mathsf{err}_j \leqslant \mathsf{err}_\star$ (Theorem 19) and by extension, $(\mathsf{err}_j - \mathsf{err}_\star)\cdot\mathbb{E}_{\mathbf{x}\sim\rho}\left[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\right] \leqslant 0$.
2. If iteration j aborts, then we will argue that with high probability this implies $p_j \leqslant \mathcal{O}\left(\frac{\varepsilon}{\sqrt{k}}\right)$ must be small (Theorem 20). Notice from its definition in Eq. (12) that p_j is proportional to $\mathbb{E}_{\mathbf{x}\sim\rho}\left[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\right]$ implying that the latter will also be small in aborted iterations. Namely,

$$(\mathsf{err}_j - \mathsf{err}_\star)\cdot\mathbb{E}_{\mathbf{x}\sim\rho}\left[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\right] \leqslant \mathbb{E}_{\mathbf{x}\sim\rho}\left[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\right] = p_j\cdot\alpha^{j,k}_{\max}$$

   A short calculation shows that $\sum_{j=0}^{k-1}\alpha^{j,k}_{\max} = \mathcal{O}(\sqrt{k})$, and by choosing constants carefully, we can argue that the overall contribution of such terms can be bounded by $\varepsilon/2$.

By invoking the above arguments to simplify the summations in Theorem 17 and using the bound on $\beta^{0,k}_0 \leqslant \varepsilon/2$ (by sufficiently large choice of k), we arrive at the inequality,

$$\Pr_{\mathbf{x}\sim\rho}\left(\widehat{f}(\mathbf{x}) \neq \pi^\star_T(\mathbf{x})\right) \leqslant \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon,$$

completing the proof sketch.

Having established the high-level approach, we will introduce the lemmas mentioned above. First we will argue that $\beta^{0,k}_0$ decays exponentially fast in k.

Lemma 18 *There exists an absolute constant $c > 0$ such that, $\beta^{0,k}_0 \leqslant e^{-ck}$.*

Proof Consider a set of k biased coins, $Z_1, \cdots, Z_k$, with probability of heads equal to $1 - \mathsf{err}_\star = \frac{3}{4}$. We will show that $\beta^{0,k}_0$ equals the probability that at most $\frac{k}{2}$ of them come up heads. Since the expected number of heads is $\frac{3k}{4}$, by standard concentration arguments (say, Hoeffding's inequality), the statement of the lemma is established.

Let $S_j = \sum_{i=1}^{j} \mathbb{I}(Z_i = \mathtt{H})$. We claim that, $\beta_r^{j,k} = \Pr\left(S_k \leqslant \frac{k}{2} \mid S_j = r\right)$. By definition of $\beta_r^{j,k}$ this is true for $j = k$. For smaller j, we leave as a short exercise to the reader to show that setting $\beta_r^{j,k}$ as such, results in the equation, $\beta_r^{j,k} = \mathtt{err}_\star \cdot \beta_r^{j+1,k} + (1 - \mathtt{err}_\star) \cdot \beta_{r+1}^{j+1,k}$ being satisfied inductively. ■

Next, we will show that in any iteration j which does not abort (i.e., the dataset $D_{\mathtt{out}}^j$ is sufficiently large), with high probability the model $\widehat{\pi}^j$ achieves low test error under $\rho_j^\star$.

Lemma 19 *Suppose* $\mathtt{ABORT}[j]$ *is false in iteration* $j \geqslant 0$. *Instantiating the base learner* $\mathsf{Alg}_{\mathtt{CoT}}(\cdot \| \varepsilon, \delta, T)$ *in Algorithm 1 as the learner in Theorem 5, then with probability at least* $1 - \frac{\delta}{k}$, *the model* $\widehat{\pi}^j$ *trained in iteration* j *satisfies,*

$$\Pr_{\mathbf{x} \sim \rho_j^\star}\left(\widehat{\pi}_T^j(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\right) = \mathtt{err}_j \leqslant \mathtt{err}_\star = \frac{1}{4} \tag{13}$$

where $\rho_j^\star$ *is defined in Eq. (7).*

Proof The proof of this lemma is discussed in Section C.1.2. ■

The next lemma shows that in iterations which are likely to abort (i.e., sampling from $\rho_j^\star$ is hard), p_j must be small.

Lemma 20 *Let* $n_{\mathtt{prompt}}(\varepsilon, \delta, T)$ *denote the sample complexity of the base learner* $\mathsf{Alg}_{\mathtt{CoT}}(\cdot \| \varepsilon, \delta, T)$ *in Algorithm 1. Suppose, for a sufficiently large constant* $C > 0$,

$$|D_{\mathtt{prompt}}| \geqslant \frac{Ck^{3/2}}{\varepsilon} \times \left(n_{\mathtt{prompt}}(\mathtt{err}_\star, \delta/k, T) + \log(k/\delta)\right).$$

Then, if $p_j \geqslant \frac{\varepsilon}{4\sqrt{k}}$ *in iteration* j, *then* $\Pr(\mathtt{ABORT}[j] \mid \mathcal{H}_{j-1}) \leqslant \frac{\delta}{k}$.

Proof The proof of this lemma is discussed in Section C.1.3. ■

B.1.1. PROOF OF THEOREM 7

Bound on prediction error. By the test-error decomposition of $\widehat{f}$ in Theorem 17, the bound on $\beta_0^{0,k}$ in Theorem 18 and the choice of $k = \Omega(\log(1/\varepsilon))$,

$$\Pr_{\mathbf{x} \sim \rho}\left(\widehat{f}(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\right) \leqslant \frac{\varepsilon}{2} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_{\mathbf{x} \sim \rho}\left[\alpha_{\mathrm{rank}_j(\mathbf{x})}^{j,k}\right]$$

By Theorem 20, in any iteration j where $p_j \geqslant \frac{\varepsilon}{4\sqrt{k}}$, with probability $1 - \frac{\delta}{k}$ this iteration does not abort. Conditioned on this event by Theorem 19, $\mathtt{err}_j \leqslant \mathtt{err}_\star$. Union bounding across all k iterations, this implies that with probability at least $1 - \delta$,

$$\begin{aligned}
\Pr_{\mathbf{x} \sim \rho}\left(\widehat{f}(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\right) &\leqslant \frac{\varepsilon}{2} + \sum_{j=0}^{k-1} \mathbb{E}_{\mathbf{x} \sim \rho}\left[\alpha_{\mathrm{rank}_j(\mathbf{x})}^{j,k}\right] \cdot \mathbb{I}\left(p_j < \frac{\varepsilon}{4\sqrt{k}}\right) \\
&\overset{(a)}{\leqslant} \frac{\varepsilon}{2} + \sum_{j=0}^{k-1} \left(\alpha_{\max}^{j,k} p_j\right) \cdot \mathbb{I}\left(p_j < \frac{\varepsilon}{4\sqrt{k}}\right) \\
&\leqslant \varepsilon
\end{aligned}$$

where (a) is by definition of p_j, and the last inequality follows by invoking Theorem 21 below.

Bound on size of prompt dataset. The bound on the size of the prompt dataset required to establish the above guarantee for Algorithm 1 is demonstrated in Theorem 20 and is,

$$n = |D_{\mathtt{prompt}}| \geqslant \frac{d}{\varepsilon} \cdot \mathrm{polylog}(T, |\Sigma|, \varepsilon^{-1}, \delta^{-1})$$

when we plug in the sample complexity of the base CoT supervised learner, $n_{\mathtt{prompt}}(\mathtt{err}_\star, \delta/k, T)$, from Theorem 5.

Bound on number of $\mathtt{CoT}(\cdot)$ oracle queries. The number of calls made to $\mathtt{CoT}$ is precisely $\sum_{j=0}^{k-1} |D^j_{\mathtt{out}}|$, which by the constraint on Algorithm 2 of Algorithm 2 gives us the upper bound $k \cdot n_{\mathtt{prompt}}(\mathtt{err}_\star, \delta/k, T)$. Overall, this means that the number of CoT queries made by Algorithm 1 is upper bounded by,

$$n_{\mathtt{CoT}} \leqslant d \cdot \mathrm{polylog}(T, |\Sigma|, \varepsilon^{-1}, \delta^{-1}).$$

almost surely, where we again plug in the upper bound on $n_{\mathtt{prompt}}(\mathtt{err}_\star, \delta/k, T)$ of the base learner derived in Theorem 5.

Lemma 21 (Lemma 3.9 in Freund (1995)) *For $0 \leqslant j \leqslant k-2$, $\alpha^{j,k}_{\max} \leqslant \frac{1.1}{\sqrt{k-j-1}}$ and $\alpha^{k-1,k}_{\max} \leqslant 1$. As a consequence,*

$$\sum_{j=0}^{k-1} \alpha^{j,k}_{\max} \leqslant 2\sqrt{k}.$$

B.2. Autocurriculum for SFT (General Π): Proof of Theorem 10

We now prove Theorem 10. In contrast to Theorem 7, we use a slightly different choice of reweighting functions to define the distributions models in the ensemble are trained on, as well as a different notion of rank of a prompt. The proof goes through the intermediate step of showing that the sequence of models $\Pi_j = \{\widehat{\pi}^0, \cdots, \widehat{\pi}^{j-1}\}$ trained in Algorithm 3 satisfy,

$$\Pr_{\mathbf{x}\sim\rho}\left(\sum\nolimits_{\pi\in\Pi_k} \mathbb{I}\Big(\mathsf{Acc}_{\mathbf{x}}(\pi_T) \geqslant \frac{4}{5}\Big) > \frac{3k}{4}\right) \geqslant 1-\varepsilon \tag{14}$$

As will be discussed in more detail later, this guarantee will suffice to argue that the outcome-level mixture model $\widehat{f} = \frac{1}{k}\sum_{j=0}^{k-1} \widehat{\pi}^j_T$ satisfies $\Pr_{\mathbf{x}\sim\rho}\big(\mathsf{Acc}_{\mathbf{x}}\big(\widehat{f}\big) \geqslant \frac{3}{5}\big) \geqslant 1-\varepsilon$. In contrast to the setting where the class of models Π is deterministic, we define an approximate notion of rank, estimated via Monte-Carlo rollouts. For a model $\pi : \mathcal{X} \to \Delta_\Sigma$, define the random variable,

$$\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(\mathcal{V}(\mathbf{x}, Y_i) = 1), \text{ where, } \{Y_i\}_{i=1}^m \overset{\text{i.i.d.}}{\sim} \pi_T(\cdot|\mathbf{x}), \tag{15}$$

and define a randomized notion of rank induced by these Monte-Carlo accuracy estimates,

$$\widehat{\mathrm{rank}}_j(\mathbf{x}) = \sum\nolimits_{\pi\in\Pi_j} \mathbb{I}\Big(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\Big). \tag{16}$$

We reserve the $\widehat{\ }$ accent to denote random variables. $\widehat{\mathrm{rank}}$ is used to define the sequence of distributions under which Algorithm 3 trains models. The threshold on the per-prompt accuracy of a

model, set in $\widehat{\text{rank}}_j$ as $\frac{9}{10}$, is chosen to be higher than our target accuracy of $\frac{4}{5}$ in Eq. (14) to account for the fact that the $\widehat{\text{rank}}$ is a random estimate.

Next we define a modified version of the $\alpha(\cdot)$'s weights considered in Algorithm 1 to define the sequence of weights to be constructed. For $0 \leqslant r \leqslant j \leqslant k$, define,

$$\widetilde{\alpha}_r^{j,k} = \widetilde{\beta}_r^{j+1,k} - \widetilde{\beta}_{r+1}^{j+1,k}, \text{ where, } \widetilde{\beta}_r^{j,k} = \begin{cases} \mathbb{I}(r \leqslant 4k/5) & \text{if } j = k \\ \mathtt{err}_\star \cdot \widetilde{\beta}_r^{j+1,k} + (1 - \mathtt{err}_\star) \cdot \widetilde{\beta}_{r+1}^{j+1,k} & \text{if } j < k \end{cases}, \tag{17}$$

where, $\mathtt{err}_\star = \dfrac{1}{10}$.

Note the subtle difference compared to Eq. (5) where the threshold on the rank r is changed from $\frac{k}{2}$ to $\frac{4k}{5}$ in $\beta_r^{j,k}$. This is to ensure that we can achieve the guarantee in Theorem 10 where the targeted threshold on the per-prompt accuracy of $\frac{3}{5}$ is a constant strictly larger than $\frac{1}{2}$. We correspondingly define the target distributions under which we carry out learning to be, $\left\{\widetilde{\rho}_j^\star\right\}_{j \geqslant 0}$,

$$\widetilde{\rho}_j^\star(\mathbf{x}) \propto \rho(\mathbf{x}) \cdot \mathbb{E}[\widehat{w}_j(\mathbf{x}) \mid \mathbf{x}, \Pi_j], \text{ where } \widehat{w}_j(\mathbf{x}) = \widetilde{\alpha}^{j,k}_{\widehat{\text{rank}}_j(\mathbf{x})} \tag{18}$$

Here, the expectation is over the randomness of $\widehat{\text{rank}}$. Finally since $\widehat{\text{rank}}$ is random, we will let the notation $\Pr_\rho(\cdot)$ (resp. $\mathbb{E}_\rho[\cdot]$) to denote probabilities (resp. expectations) marginalizing over $\mathbf{x} \sim \rho$ and all other sources of randomness.

Similarly to Theorem 17, we begin with a decomposition of the test-error incurred by Algorithm 1 into terms which depend on the performance of the models trained on samples from $\widetilde{\rho}_j^\star$ in each iteration.

Lemma 22 (Loss decomposition) *Let $\widehat{\pi}^0, \cdots, \widehat{\pi}^{k-1}$ denote the sequence of models trained within Algorithm 3. Then,*

$$\Pr_\rho\left(\widehat{\text{rank}}_k(\mathbf{x}) \leqslant \frac{4k}{5} \,\middle|\, \Pi_k\right) = \widetilde{\beta}_0^{0,k} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_\rho\left[\widetilde{\alpha}^{j,k}_{\widehat{\text{rank}}_j(\mathbf{x})} \,\middle|\, \Pi_j\right] \tag{19}$$

where, $\mathtt{err}_\star = \frac{1}{10}$ and $\mathtt{err}_j \triangleq \Pr_{\widetilde{\rho}_j^\star}\left(\widehat{\mathtt{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) \leqslant \frac{9}{10} \,\middle|\, \widehat{\pi}^j\right)$.

Proof The formal proof of this result is discussed in Section C.2.1. ■

At a high level, the argument will rely on analyzing the change in a potential function, defined as,

$$\widetilde{\Phi}_j = \mathbb{E}_\rho\left[\widetilde{\beta}^{j,k}_{\widehat{\text{rank}}_j(\mathbf{x})} \,\middle|\, \Pi_j\right]$$

And noting that, $\widetilde{\Phi}_k$ equals the LHS of Eq. (19), while $\widetilde{\Phi}_0 = \widetilde{\beta}_0^{0,k}$ equals the first term on the RHS. As before, in the next lemma we will bound the initial value of the potential.

Lemma 23 *There exists an absolute constant $c > 0$ such that, $\widetilde{\beta}_0^{0,k} \leqslant e^{-ck}$. Consequently, as long as $k \geqslant C \log(1/\varepsilon)$ for a sufficiently large constant $C > 0$, $\widetilde{\beta}_0^{0,k} \leqslant \frac{\varepsilon}{4}$.*

Proof Consider a set of k biased coins, $Z_1, \cdots, Z_k$, with probability of heads equal to $1 - \mathtt{err}_\star = \frac{9}{10}$. By the same argument as in the proof of Theorem 18, $\widetilde{\beta}_0^{0,k}$ equals the probability that at most $\frac{4k}{5}$ of the coins come up heads. Since the expected number of heads is $\frac{9k}{10}$, by Hoeffding's inequality, the statement of the lemma is established. ■

Next, we will define an event which captures whether the dataset collected in iteration j, D^j_{out} is sufficiently large (a proxy for whether training the model in iteration j is feasible),

$$\widetilde{\mathtt{ABORT}}[j] = \left\{|D^j_{\mathsf{out}}| < n_{\mathtt{prompt}}(1/400, \delta/k, T)\right\} \tag{20}$$

where $n_{\mathtt{prompt}}(\varepsilon', \delta', T)$ is the sample complexity of the base learning algorithm, $\mathsf{Alg}(\cdot \| \varepsilon', \delta', \mathtt{CoT})$ used in Algorithm 3. Furthermore, we will also define,

$$\widetilde{p}_j = \frac{\mathbb{E}_\rho\left[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \,\middle|\, \Pi_j\right]}{\widetilde{\alpha}^{j,k}_{\max}} \tag{21}$$

which will turn out to equal the probability that a prompt $\mathbf{x} \sim \rho$ is accepted into the dataset D^j_{out} in Algorithm 2 in the j^{th} iteration. Intuitively, a larger value of $\widetilde{p}_j$ means that we will have more prompts to train the model $\widehat{\pi}^j$ on, which itself translates to a smaller probability of this iteration aborting.

Lemma 24 *Suppose $\widetilde{\mathtt{ABORT}}[j]$ is false in iteration $j \geqslant 0$. Then, instantiating the base learning algorithm $\mathsf{Alg}(\cdot\|\varepsilon, \delta, \mathtt{CoT})$ in Algorithm 3 as the learner in Theorem 5, then the model $\widehat{\pi}^j$ trained in iteration j satisfies with probability at least $1 - \frac{\delta}{k}$,*

$$\Pr_{\widetilde{\rho}^\star_j}\left(\widehat{\mathtt{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) < \frac{9}{10}\right) \triangleq \mathtt{err}_j \leqslant \frac{1}{10}. \tag{22}$$

Proof The proof of this lemma is discussed in Section C.2.2. ■

Finally, similar to Theorem 20, we will show that any iteration j is unlikely to abort as long as $\widetilde{p}_j$ is sufficiently large. This will use the fact that the marginal probability that $\mathbf{x} \sim \rho$ is accepted into D^j_{out} via rejection sampling equals $\widetilde{p}_j$.

Lemma 25 *Suppose, for a sufficiently large constant $C > 0$,*

$$|D_{\mathtt{prompt}}| \geqslant \frac{Ck^{3/2}}{\varepsilon} \times \left(n_{\mathtt{prompt}}(1/400, \delta/k, T) + \log(k/\delta)\right).$$

Then, if $\widetilde{p}_j \geqslant \frac{\varepsilon}{16\sqrt{k}}$ in iteration j, then $\Pr(\widetilde{\mathtt{ABORT}}[j] \mid \mathcal{H}_{j-1}) \leqslant \frac{\delta}{k}$.

Proof The proof of this lemma is discussed in Section C.2.3. ■

Finally, we introduce a lemma showing how to translate between bounds on the accuracy, and the Monte Carlo estimate of the accuracy (Eq. (15)) with some slack.

Lemma 26 *For the ensemble of models Π_k, we have,*

$$\Pr_{\mathbf{x}\sim\rho}\left(\sum\nolimits_{\pi\in\Pi_k} \mathbb{I}\left(\mathtt{Acc}_{\mathbf{x}}(\pi) \geqslant \frac{4}{5}\right) \leqslant \frac{3k}{4} \,\middle|\, \Pi_k\right) \leqslant 2\cdot\Pr_\rho\left(\sum\nolimits_{\pi\in\Pi_k} \mathbb{I}\left(\widehat{\mathtt{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\right) \leqslant \frac{4k}{5} \,\middle|\, \Pi_k\right)$$

Proof The proof of this lemma is discussed in section C.2.4. ■

Having introduced all the necessary results, we are ready to prove the main result of this section.

B.2.1. PROOF OF THEOREM 10

Bound on prediction error. By the test-error decomposition of $\widehat{f}$ in Theorem 22, the bound on $\widetilde{\beta}_0^{0,k}$ in Theorem 23 and the choice of $k = \Omega(\log(1/\varepsilon))$,

$$\Pr_\rho\left(\widehat{\mathrm{rank}}_k(\mathbf{x}) \leqslant \frac{4k}{5} \,\middle|\, \Pi_k\right) \leqslant \frac{\varepsilon}{4} + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_\rho\left[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \,\middle|\, \Pi_j\right].$$

In any iteration j where $\widetilde{p}_j \geqslant \frac{\varepsilon}{16\sqrt{k}}$, with probability $1 - \frac{\delta}{k}$ this iteration does not abort. Conditioned on this event, by Theorem 24, with probability $1 - \frac{\delta}{k}$, $\mathtt{err}_j \leqslant \mathtt{err}_\star$. This implies, with probability at least $1 - \delta$,

$$\begin{aligned}
\Pr_\rho\left(\widehat{\mathrm{rank}}_k(\mathbf{x}) \leqslant \frac{4k}{5} \,\middle|\, \Pi_k\right) &\leqslant \frac{\varepsilon}{4} + \sum_{j=0}^{k-1} \mathbb{E}_\rho\left[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \,\middle|\, \Pi_j\right] \cdot \mathbb{I}\left(\widetilde{p}_j < \frac{\varepsilon}{16\sqrt{k}}\right) \\
&\overset{(a)}{\leqslant} \frac{\varepsilon}{4} + \sum_{j=0}^{k-1} \left(\widetilde{\alpha}^{j,k}_{\max} \widetilde{p}_j\right) \cdot \mathbb{I}\left(\widetilde{p}_j < \frac{\varepsilon}{16\sqrt{k}}\right) \\
&\leqslant \frac{\varepsilon}{2},
\end{aligned} \tag{23}$$

where (a) is by definition of $\widetilde{p}_j$, and the last inequality invokes Theorem 27 introduced below. Finally, we have the following sequence of inequalities to bound the performance of the outcome-level mixture model $\widehat{f} = \frac{1}{k}\sum_{\pi \in \Pi_k} \pi_T$. First, noting that $\mathtt{Acc}_{\mathbf{x}}(\widehat{f}) = \frac{1}{k}\sum_{\pi\in\Pi_k} \mathtt{Acc}_{\mathbf{x}}(\pi)$, by an application of Markov's inequality,

$$\begin{aligned}
\Pr_{\mathbf{x}\sim\rho}\left(\mathtt{Acc}_{\mathbf{x}}(\widehat{f}) > \frac{3}{5} \,\middle|\, \Pi_k\right) &= \Pr_{\mathbf{x}\sim\rho}\left(\sum\nolimits_{\pi\in\Pi_k} \mathtt{Acc}_{\mathbf{x}}(\pi) > \frac{3k}{5} \,\middle|\, \Pi_k\right) \\
&\geqslant \Pr_{\mathbf{x}\sim\rho}\left(\sum\nolimits_{\pi\in\Pi_k} \mathbb{I}\left(\mathtt{Acc}_{\mathbf{x}}(\pi) \geqslant \frac{4}{5}\right) > \frac{3k}{4} \,\middle|\, \Pi_k\right)
\end{aligned}$$

Taking the complement on both sides and invoking Theorem 26, we get,

$$\Pr_{\mathbf{x}\sim\rho}\left(\mathtt{Acc}_{\mathbf{x}}(\widehat{f}) \leqslant \frac{3}{5} \,\middle|\, \Pi_k\right) \leqslant 2 \cdot \Pr_\rho\left(\widehat{\mathrm{rank}}_k(\mathbf{x}) \leqslant \frac{4k}{5} \,\middle|\, \Pi_k\right).$$

Plugging in the upper bound on the RHS from Eq. (23) completes the proof.

Bound on size of prompt dataset. The bound on the size of the prompt dataset required to establish the above guarantee for Algorithm 3 is demonstrated in Theorem 25 and is,

$$n = |D_{\mathtt{prompt}}| \gtrsim \frac{\log(|\Pi|)}{\varepsilon} \cdot \mathrm{polylog}(T, \varepsilon^{-1}, \delta^{-1})$$

when we plug in the sample complexity $n_{\mathtt{prompt}}(1/400, \delta/k, T)$ from Theorem 5.

Bound on number of $\mathtt{CoT}(\cdot)$ oracle queries. The number of calls made to $\mathtt{CoT}(\cdot)$ is precisely $\sum_{j=0}^{k-1} |D^j_{\mathtt{out}}|$, which by the constraint on Algorithm 2 of Algorithm 2 gives us the upper bound $k \cdot n_{\mathtt{prompt}}(1/400, \delta/k, T)$. Overall, this means that the number of CoT queries made by Algorithm 3 is a.s. upper bounded by,

$$n_{\mathtt{CoT}} \leqslant \log(|\Pi|) \cdot \mathrm{polylog}(T, \varepsilon^{-1}, \delta^{-1}).$$

Lemma 27 *For* $0 \leqslant j \leqslant k-2$, $\widetilde{\alpha}_{\max}^{j,k} \leqslant \frac{2}{\sqrt{k-j-1}}$ *and* $\widetilde{\alpha}_{\max}^{k-1,k} \leqslant 1$. *Consequently,*

$$\sum_{j=0}^{k-1} \widetilde{\alpha}_{\max}^{j,k} \leqslant 4\sqrt{k}.$$

Proof The proof of this lemma is described in Section C.2.5. ■

B.3. Improving a Reference Model Satisfying Coverage: Proof of Theorem 14

In this section, we state and prove a more general version of Theorem 14 in a setting where the reference model π_{ref} is allowed to fail to satisfy sequence-level coverage on a small mass of prompts.

Definition 28 (Partial sequence-level coverage) π_{ref} *is said to satisfy* (C_{seq}, η) *partial sequence-level coverage under the prompt distribution* ρ *if,*

$$\Pr_{\mathbf{x}\sim\rho}\left(\pi_{\mathrm{ref}}(\mathbf{y}^\star_{\mathbf{x}}|\mathbf{x}) \geqslant C_{\mathrm{seq}}^{-1}\right) \geqslant 1-\eta \tag{24}$$

where $\mathbf{y}^\star_{\mathbf{x}} = \pi^\star_{1:T}(\mathbf{x})$.

Theorem 29 (Learning with partial sequence-level coverage) *Consider any target error* $\varepsilon \in (0,1)$ *and failure probability* $\delta \in (0,1)$. *Suppose the reference model* π_{ref} *satisfies* (C_{seq}, η) *partial coverage (cf. Theorem 28). Consider the outcome-level model* $\widehat{f}$ *returned by Algorithm 4. Then, there exist absolute constants* $C_1, C_2 > 0$ *such that as long as,*

$$n \geqslant C_1 \frac{d\log(TC_{\mathrm{seq}})\log(n) + \log(1/\delta)}{\varepsilon},$$

with probability at least $1-\delta$, $\mathtt{Acc}_\rho\left(\widehat{f}\right) \geqslant 1 - C_2(\varepsilon+\eta)$. *Furthermore, Algorithm 4 queries the outcome verifier oracle* $\mathcal{V}$, *and generates traces from* π_{ref} *no more than* $\mathcal{O}(nC_{\mathrm{seq}}\log(nC_{\mathrm{seq}}/\delta))$ *times.*

Algorithm 4 trains an outcome-level model by minimizing the loss $\mathcal{L}(\pi; D_{\mathtt{prompt}})$ defined below,

$$\widehat{\pi} \in \arg\min_{\pi\in\Pi} \mathcal{L}(\pi; D_{\mathtt{prompt}}), \text{ where } \mathcal{L}(\pi; D_{\mathtt{prompt}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left(\pi_{1:T}(\mathbf{x}_i) \notin \widetilde{D}_i\right), \tag{25}$$

where the $\widetilde{D}_i$ datasets are constructed within the algorithm. For any $\mathbf{x} \in \mathcal{X}$, let,

$$\mathcal{Y}^\star(\mathbf{x}) = \left\{\mathbf{y} \in \mathcal{Y} : \mathcal{V}(\mathbf{x}, y_T) = 1 \text{ and } \pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x}) \geqslant C_{\mathrm{seq}}^{-1}\right\}$$

be the set of all sufficiently high probability strings under $\pi_{\mathrm{ref}}(\cdot|\mathbf{x})$ that also predict the correct answer on $\mathbf{x}$. By the partial sequence-level coverage assumption on π_{ref},

$$\Pr_{\mathbf{x}\sim\rho}\left(\mathcal{Y}^\star(\mathbf{x}) \neq \emptyset\right) \geqslant 1-\eta. \tag{26}$$

We first establish a high-probability event that holds when Algorithm 4 is run.

Lemma 30 *Let $\mathcal{E}$ denote the event $\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\big(\widetilde{D}_i \neq \mathcal{Y}^\star(\mathbf{x}_i)\big) \leqslant 2\eta + \varepsilon$. Then $\Pr(\mathcal{E}) \geqslant 1 - \frac{\delta}{2}$.*

Proof For $\mathbf{x} \in \mathcal{X}$, let $\mathcal{Y}(\mathbf{x}) = \big\{\mathbf{y} \in \mathcal{Y} : \pi_{\text{ref}}(\mathbf{y}|\mathbf{x}) \geqslant C_{\text{seq}}^{-1}\big\}$. Let D_{sample} be a set of $m = C_{\text{seq}}\log(4nC_{\text{seq}}/\delta)$ responses $\{\mathbf{y}^1, \cdots, \mathbf{y}^m\}$ sampled i.i.d. from $\pi_{\text{ref}}(\cdot|\mathbf{x})$, and let $D \subseteq D_{\text{sample}}$ denote the deduplicated subset of responses $\mathbf{y}^i$ such $\pi_{\text{ref}}(\mathbf{y}^i|\mathbf{x}) \geqslant C_{\text{seq}}^{-1}$. For any $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$, the probability of $\mathbf{y} \notin D_{\text{sample}}$ is upper bounded by,

$$\Pr\big(\mathbf{y} \notin D_{\text{sample}} \;\big|\; \mathbf{x}\big) \leqslant \frac{\delta}{4nC_{\text{seq}}}.$$

Union bounding over $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$ (at most C_{seq} such strings) gives us the inequality,

$$\Pr(D = \mathcal{Y}(\mathbf{x}) \;\big|\; \mathbf{x}\big) = \Pr\big(\mathcal{Y}(\mathbf{x}) \subseteq D_{\text{sample}} \;\big|\; \mathbf{x}\big) \geqslant 1 - \frac{\delta}{4n}. \tag{27}$$

Let $\mathcal{X}_{\text{cov}}$ denote the set of prompts $\big\{\mathbf{x} : \mathcal{Y}^\star(\mathbf{x}) \neq \emptyset\big\}$. Then, by Eq. (26), $\Pr_{\mathbf{x}\sim\rho}(\mathcal{X}_{\text{cov}}) \geqslant 1 - \eta$. For every $\mathbf{x} \in \mathcal{X}_{\text{cov}}$,

$$\mathcal{Y}(\mathbf{x}) = D \implies \mathcal{Y}^\star(\mathbf{x}) = \{\mathbf{y} \in D : \mathcal{V}(\mathbf{x}, y_T) = 1\} \triangleq \widetilde{D} \tag{28}$$

Furthermore, by an application of Chernoff bound to the sum of the random variables $\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(\mathbf{x}_i \notin \mathcal{X}_{\text{cov}})$ (which has expectation at most η from Eq. (26)), with probability at least $1 - \frac{\delta}{4}$,

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\big(\mathbf{x}_i \notin \mathcal{X}_{\text{cov}}\big) \leqslant \eta + 10\sqrt{\frac{\eta\log(4/\delta)}{n}} \overset{(i)}{\leqslant} 2\eta + \frac{50\log(4/\delta)}{n} \overset{(ii)}{\leqslant} 2\eta + \varepsilon,$$

where, (i) uses the AM-GM inequality, while (ii) uses the lower bound on the size of the dataset n. Combining with Eq. (27) and Eq. (28) completes the proof. ∎

Corollary 31 *Under the event $\mathcal{E}$ (cf. Theorem 30), for every model π,*

$$\left|\mathcal{L}(\pi; D_{\texttt{prompt}}) - \frac{1}{n}\sum_{i=1}^{n} g_\pi(\mathbf{x}_i)\right| \leqslant 2\eta + \varepsilon, \text{ where, } g_\pi(\mathbf{x}) = \mathbb{I}\big(\pi_{1:T}(\mathbf{x}) \notin \mathcal{Y}^\star(\mathbf{x})\big)$$

Furthermore, $\inf_{\pi\in\Pi}\mathbb{E}_{\mathbf{x}\sim\rho}\big[g_\pi(\mathbf{x})\big] \leqslant \eta$.

Proof By the assertion, $\big|\mathbb{I}\big(\mathbf{y} \in \mathcal{Y}^\star(\mathbf{x}_i)\big) - \mathbb{I}\big(\mathbf{y} \in \widetilde{D}_i\big)\big| \leqslant \mathbb{I}\big(\mathcal{Y}^\star(\mathbf{x}_i) \neq \widetilde{D}_i\big)$ for every $i \in [n]$ and upper bounding the latter summation via Theorem 30 proves the main statement of the corollary. On the other hand, $\mathbb{E}_{\mathbf{x}\sim\rho}\big[g_\pi(\mathbf{x})\big] \leqslant \eta$ at $\pi \leftarrow \pi^\star$ (by the assumption on partial sequence-level coverage of π_{ref} in theorem 28). ∎

Next we will define the loss class,

$$\mathcal{G} = \big\{g_\pi(\cdot) = \mathbb{I}\big(\pi_{1:T}(\cdot) \notin \mathcal{Y}^\star(\cdot)\big) : \pi \in \Pi\big\}$$

associated with Π. For any $\mathbf{x} \in \mathcal{X}$, $g_\pi(\mathbf{x}) = 0$ only if $\pi^\star_{1:T}(\mathbf{x})$ takes one of at most C_{seq} possible sequences, which are those in $\mathcal{Y}(\mathbf{x})$. On such points, the class of models Π is forced to have "low

complexity", suggesting that the growth function of the loss class can be bounded and enabling a uniform convergence argument to bound the variation of the loss. In order to formally prove this statement, we first introduce some notation. For a set of n prompts, $D = \{\mathbf{x}_i\}_{i=1}^n \in \mathcal{X}^n$, define the set of possible behaviors which losses in $\mathcal{G}$ can express over these points by,

$$\mathcal{B}_{\mathcal{G}}(D) = \big\{\big(g(\mathbf{x}_i) : i \in [n]\big) : g \in \mathcal{G}\big\}$$

The maximum cardinality of this set over datasets of size n is the growth function of the loss class under consideration.

Lemma 32 *For any dataset $D = \{\mathbf{x}_i\}_{i=1}^n$ of size n, $|\mathcal{B}_{\mathcal{G}}(D)| \leqslant (enT|\Sigma|C_{\mathrm{seq}})^d$. This implies that,*

$$\mathrm{VC}(\mathcal{G}) \leqslant d\log(T|\Sigma|C_{\mathrm{seq}}).$$

where $\mathrm{VC}(\cdot)$ returns the VC dimension of its argument.

Proof We will defer the proof of this lemma to Section C.3.1. ■

Finally, we also use a standard generalization bound based on uniform convergence and localization (i.e., the offset trick). The proof follows by invoking (Bartlett et al., 2005, Theorem 5.2) and an application of the AM-GM inequality.

Theorem 33 (Excess risk bound for ERM under 0-1 loss) *Consider a dataset $D = \{\mathbf{x}_i\}_{i=1}^n \overset{i.i.d.}{\sim} \rho$. With probability at least $1-\delta$, for all $\widehat{g} \in \mathcal{G}$,*

$$\mathbb{E}_{\mathbf{x}\sim\rho}[\widehat{g}(\mathbf{x})] \lesssim \left(\frac{1}{n}\sum_{i=1}\widehat{g}(\mathbf{x}_i) - \min_{g\in\mathcal{G}}\frac{1}{n}\sum_{i=1}g(\mathbf{x}_i)\right) + \inf_{g\in\mathcal{G}}\mathbb{E}_{\mathbf{x}\sim\rho}\big[g(\mathbf{x})\big] + \frac{\mathrm{VC}(\mathcal{G})\log(n/\mathrm{VC}(\mathcal{G})) + \log(1/\delta)}{n}.$$

B.3.1. PROOF OF THEOREM 14

In conjunction with Theorem 31 and Theorem 32, for the predictor $\widehat{g} \leftarrow g_{\widehat{\pi}}$ where $\widehat{\pi} \in \Pi$ is the minimizer of $\mathcal{L}(\pi; D_{\mathtt{prompt}})$, with probability at least $1-\delta$,

$$\mathbb{E}_{\mathbf{x}\sim\rho}\big[g_{\widehat{\pi}}(\mathbf{x})\big] \lesssim \frac{d\log(TC_{\mathrm{seq}})\log(n) + \log(1/\delta)}{n} + \eta + \varepsilon. \tag{29}$$

Since $g_\pi(\mathbf{x}) = \mathbb{I}\big(\pi_{1:T}(\mathbf{x}) \notin \mathcal{Y}^\star(\mathbf{x})\big) \geqslant \mathbb{I}\big(\pi_T(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\big)$, this implies that the LHS of Eq. (29) is further lower bounded by $\Pr_{\mathbf{x}\sim\rho}\big(\pi_T(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\big)$, completing the proof.

B.4. Autocurriculum for Fine-Tuning a Reference Model: Proof of Theorem 16

The proof of Theorem 16 largely follows that of Theorem 7, except where we instantiate the base learner $\mathsf{Alg}_{\mathcal{Q}}$ as `RLFineTune` (Algorithm 4). This will essentially only change the weak learning guarantee of Theorem 19. All other definitions (Π_j, $\rho_j^\star$, etc.) are kept as before.

Lemma 34 *Suppose the event $\mathtt{ABORT}[j]$ is false in iteration $j \geqslant 0$. Let the base learner $\mathsf{Alg}_{\mathsf{RL}}(\cdot\|\varepsilon', \delta', T)$ in Algorithm 5 be instantiated as `RLFineTune` (Algorithm 4). Then, the model $\widehat{\pi}^j$ trained in Algorithm 5 satisfies with probability at least $1 - \frac{\delta}{k}$,*

$$\Pr_{\mathbf{x}\sim\rho_j^\star}\left(\widehat{\pi}_T^j(\mathbf{x}) \neq \pi_T^\star(\mathbf{x})\right) = \mathtt{err}_j \leqslant \mathtt{err}_\star = \frac{1}{4}.$$

What remains is to analyze the sample and computational complexity of Algorithm 5. We begin with the cost of generating the datasets $\big\{D_{\mathtt{out}}^j : 0 \leqslant j \leqslant k-1\big\}$.

Cost of generating datasets $\left\{D^j_{\texttt{out}} : 0 \leqslant j \leqslant k-1\right\}$.

1. *Length-T CoTs generated from π_{ref}.* Across the k invocations of `RLFineTune`, the number of length-T CoTs generated from π_{ref} is $\sum_{j=0}^{k-1} m \times |D^j_{\texttt{out}}|$. With the bound on $n_{\texttt{prompt}}(\varepsilon', \delta', T)$ in Theorem 14,

$$\begin{aligned} k &= \mathcal{O}(\log(1/\varepsilon)) \\ m &= \mathcal{O}(C_{\mathrm{seq}} \log(nC_{\mathrm{seq}}/\delta)) \\ |D^j_{\texttt{out}}| &\leqslant \mathcal{O}(d \log(TC_{\mathrm{seq}}) + \log(k/\delta)), \end{aligned}$$

   the number of calls to π_{ref} is upper bounded by,

$$dC_{\mathrm{seq}} \cdot \mathrm{polylog}(C_{\mathrm{seq}}, \varepsilon^{-1}, \delta^{-1}, T)$$

2. *Length-T CoTs generated from learner's models.* The number of length-T CoTs generated from $\widehat{\pi}^j$ is at most $|D_{\texttt{prompt}}|$, which implies that the total number of CoTs generated across all models trained by the learner is upper bounded by $k \times |D_{\texttt{prompt}}|$,

$$\frac{d}{\varepsilon} \cdot \mathrm{polylog}(C_{\mathrm{seq}}, \varepsilon^{-1}, \delta^{-1}, T)$$

   Bearing only polylogarithmic dependency on the sequence-level coverage C_{seq}.

3. *Number of calls to the outcome verifier $\mathcal{V}$.* Across the k invocations of `RLFineTune`, the outcome verifier is called once for every call to π_{ref}. Furthermore, in constructing the intermediate learning distributions $\rho^\star_j$, the learner calls the verifier $m \times |D_{\texttt{prompt}}|$ times. This implies that the overall number of calls to $\mathcal{V}$ also scales as,

$$dC_{\mathrm{seq}} \cdot \mathrm{polylog}(C_{\mathrm{seq}}, \varepsilon^{-1}, \delta^{-1}, T) + \frac{d}{\varepsilon} \cdot \mathrm{polylog}(C_{\mathrm{seq}}, \varepsilon^{-1}, \delta^{-1}, T)$$

Computation spent in running `RLFineTune`. Running `RLFineTune` requires a single optimization call to minimize the loss in Eq. (10) over a dataset of size n. `AutoTune.RL` makes $k = \mathcal{O}(\log(1/\varepsilon))$ calls to this oracle. However, each call is on a much smaller dataset than `RLFineTune` would require without autocurriculum. In particular, each $\widehat{\pi}^j$ is trained on a prompt dataset $D^j_{\texttt{out}}$ of size at most $\mathcal{O}(d \log(TC_{\mathrm{seq}}) + \log(k/\delta))$.

Appendix C. Proofs for Supporting Lemmas

C.1. Proofs for Lemmas from Theorem 7

C.1.1. PROOF OF THEOREM 17

For $j \geqslant 0$, recall by definition,

$$\Phi_{j+1} = \sum_{r=0}^{j+1} \beta_r^{j+1,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_{j+1}(\mathbf{x}) = r\big).$$

Note that $\mathrm{rank}_{j+1}(\mathbf{x}) = r$ is only possible if $\mathrm{rank}_j(\mathbf{x}) = r$ or $\mathrm{rank}_j(\mathbf{x}) = r-1$. Likewise, if $\mathrm{rank}_j(\mathbf{x}) = r$, then $\mathrm{rank}_{j+1}(\mathbf{x}) \neq r \implies \mathrm{rank}_{j+1}(\mathbf{x}) = r+1$. With this, we decompose the above expression as,

$$\begin{aligned}
\Phi_{j+1} &= \sum_{r=0}^{j} \beta_r^{j+1,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_j(\mathbf{x}) = r\big) \\
&\quad - \sum_{r=0}^{j} \beta_r^{j+1,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_{j+1}(\mathbf{x}) = r+1 \text{ and } \mathrm{rank}_j(\mathbf{x}) = r\big) \\
&\quad + \sum_{r=1}^{j+1} \beta_r^{j+1,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_{j+1}(\mathbf{x}) = r \text{ and } \mathrm{rank}_j(\mathbf{x}) = r-1\big) \\
&= \sum_{r=0}^{j} \beta_r^{j+1,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_j(\mathbf{x}) = r\big) \\
&\quad - \sum_{r=0}^{j} \Big(\underbrace{\beta_r^{j+1,k} - \beta_{r+1}^{j+1,k}}_{\alpha_r^{j,k}}\Big) \cdot \Pr_{\mathbf{x}\sim\rho}\Big(\underbrace{\mathrm{rank}_{j+1}(\mathbf{x}) = r+1 \text{ and } \mathrm{rank}_j(\mathbf{x}) = r}_{\equiv\left\{\mathrm{rank}_j(\mathbf{x}) = r \text{ and } \widehat{\pi}_T^j(\mathbf{x}) = \pi_T^\star(\mathbf{x})\right\}}\Big) \qquad (30)
\end{aligned}$$

The second term on the RHS of the above equation can be further decomposed as,

$$\begin{aligned}
&\sum_{r=0}^{j} \alpha_r^{j,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_j(\mathbf{x}) = r \text{ and } \widehat{\pi}_T^j(\mathbf{x}) = \pi_T^\star(\mathbf{x})\big) \\
&= \mathbb{E}_{\mathbf{x}\sim\rho}\big[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})} \cdot \mathbb{I}\big(\widehat{\pi}_T^j(\mathbf{x}) = \pi_T^\star(\mathbf{x})\big)\big] \\
&\overset{(a)}{=} \mathbb{E}_{\mathbf{x}\sim\rho}\big[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\big] \cdot \Pr_{\mathbf{x}\sim\rho_j^\star}\big(\widehat{\pi}_T^j(\mathbf{x}) = \pi_T^\star(\mathbf{x})\big) \\
&= (1 - \mathtt{err}_j) \times \sum_{r=0}^{j} \alpha_r^{j,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_j(\mathbf{x}) = r\big)
\end{aligned}$$

where in (a), we use the definition of $\rho_j^\star$, which is the distribution proportional to $\rho(\cdot)w_j(\cdot)$. In the final equation in the sequence, we use the definition of $\mathtt{err}_j$ in Eq. (13). Combining back with Eq. (30), noting that $\alpha_r^{j,k} = \beta_r^{j+1,k} - \beta_{r+1}^{j+1,k}$ and $\beta_r^{j,k} = \mathtt{err}_\star \cdot \beta_r^{j+1,k} + (1 - \mathtt{err}_\star) \cdot \beta_{r+1}^{j+1,k}$ where $\mathtt{err}_\star = \frac{1}{4}$,

$$\Phi_{j+1} = \sum_{r=0}^{j} \Big(\mathtt{err}_\star \cdot \beta_r^{j+1,k} + (1 - \mathtt{err}_\star) \cdot \beta_{r+1}^{j+1,k}\Big) \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_j(\mathbf{x}) = r\big)$$

$$
\begin{aligned}
&+ (\mathtt{err}_j - \mathtt{err}_\star) \sum_{r=0}^{j} \alpha_r^{j,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_j(\mathbf{x}) = r\big) \\
&= \sum_{r=0}^{j} \beta_r^{j,k} \cdot \Pr_{\mathbf{x}\sim\rho}\big(\mathrm{rank}_j(\mathbf{x}) = r\big) + (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_{\mathbf{x}\sim\rho}\big[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\big] \\
&= \Phi_j + (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_{\mathbf{x}\sim\rho}\big[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\big],
\end{aligned}
$$

Summing from $j = 0$ to $j = k-1$, we arrive at the equation,

$$
\Phi_k = \Phi_0 + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_{\mathbf{x}\sim\rho}\big[\alpha^{j,k}_{\mathrm{rank}_j(\mathbf{x})}\big]
$$

Finally plugging in the explicit formula for Φ_0 and Φ_k completes the proof.

C.1.2. PROOF OF THEOREM 19

Recall from Theorem 5, that the learning algorithm $\mathsf{Alg}(\cdot\|\varepsilon', \delta', \mathtt{CoT})$ has sample complexity $n_{\mathtt{prompt}}(\varepsilon', \delta', T)$. Assuming that $\mathtt{ABORT}[j]$ is false in iteration j, the size of the dataset $D^j_{\mathtt{out}}$ is larger than $n_{\mathtt{prompt}}(\mathtt{err}_\star, \delta/k, T)$. This implies that with probability $1 - \frac{\delta}{k}$ the model $\widehat{\pi}^j$ trained in iteration j satisfies,

$$
\Pr_{\mathbf{x}\sim\rho^\star_j}\Big(\widehat{\pi}^j_T(\mathbf{x}) \neq \pi^\star_T(\mathbf{x})\Big) = \mathtt{err}_j \leqslant \mathtt{err}_\star = \frac{1}{4}. \tag{31}
$$

This equation uses the fact that prompts in $D^j_{\mathtt{out}}$ fed into the base learner $\mathsf{Alg}(\cdot\|\varepsilon', \delta', \mathtt{CoT})$ are sampled from the distribution $\rho^\star_j$ (Eq. (7)), by rejection sampling from ρ.

C.1.3. PROOF OF THEOREM 20

By definition of p_j (Eq. (12)), and by the structure of the `Sample` subroutine (Algorithm 1 in Algorithm 1), the size of the dataset $|D^j_{\mathtt{out}}|$ which the model $\widehat{\pi}^j$ trains on, can be expressed as the sum of $n' = |D^j_{\mathtt{prompt}}| = |D_{\mathtt{prompt}}|/k$ i.i.d. Bernoulli random variables, each with mean p_j. Indeed, p_j is the probability that $\mathbf{x} \sim \rho$ is accepted into the dataset $D^j_{\mathtt{out}}$. By an application of the multiplicative Chernoff bound,

$$
\Pr\left(|D^j_{\mathtt{out}}| \leqslant \frac{n'p_j}{2} \;\middle|\; \mathcal{H}_{j-1}\right) \leqslant \exp\left(-\frac{n'p_j}{8}\right)
$$

By the sufficiently large choice of $|D_{\mathtt{prompt}}| = n'k$, when $p_j \geqslant \frac{\varepsilon}{4\sqrt{k}}$, we have that,

1. $n'p_j \geqslant 8\log(k/\delta)$, and,
2. $n'p_j \geqslant 2n_{\mathtt{prompt}}(\mathtt{err}_\star, \delta/k, T)$.

Together with the definition of $\mathtt{ABORT}[j]$ in Eq. (11), these imply that,

$$
\Pr(\mathtt{ABORT}[j] \mid \mathcal{H}_{j-1}) \leqslant \frac{\delta}{k}
$$

C.2. Proofs for Lemmas from Theorem 10

C.2.1. PROOF OF THEOREM 22

For $j \geqslant 0$, recall by definition,

$$\widetilde{\Phi}_{j+1} = \sum_{r=0}^{j+1} \widetilde{\beta}_r^{j+1,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_{j+1}(\mathbf{x}) = r \;\big|\; \Pi_{j+1}\big).$$

Note that the condition $\widehat{\mathrm{rank}}_{j+1}(\mathbf{x}) = \sum_{\pi \in \Pi_{j+1}} \mathbb{I}\big(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\big) = r$ is only possible in one of two cases, either,

$$\sum_{\pi \in \Pi_j} \mathbb{I}\Big(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\Big) = r \text{ or } \sum_{\pi \in \Pi_j} \mathbb{I}\Big(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\Big) = r - 1$$

Likewise, if $\widehat{\mathrm{rank}}_j(\mathbf{x}) = r$, then $\widehat{\mathrm{rank}}_{j+1}(\mathbf{x}) = \sum_{\pi \in \Pi_{j+1}} \mathbb{I}\big(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\big) \neq r \implies \widehat{\mathrm{rank}}_{j+1}(\mathbf{x}) = r + 1$. With this, we decompose the above expression as,

$$\begin{aligned}
\widetilde{\Phi}_{j+1} &= \sum_{r=0}^{j} \widetilde{\beta}_r^{j+1,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \;\big|\; \Pi_{j+1}\big) \\
&\quad - \sum_{r=0}^{j} \widetilde{\beta}_r^{j+1,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_{j+1}(\mathbf{x}) = r+1 \text{ and } \widehat{\mathrm{rank}}_j(\mathbf{x}) = r \;\big|\; \Pi_{j+1}\big) \\
&\quad + \sum_{r=1}^{j+1} \widetilde{\beta}_r^{j+1,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_{j+1}(\mathbf{x}) = r \text{ and } \widehat{\mathrm{rank}}_j(\mathbf{x}) = r-1 \;\big|\; \Pi_{j+1}\big) \\
&= \sum_{r=0}^{j} \widetilde{\beta}_r^{j+1,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \;\big|\; \Pi_j\big) \\
&\quad - \sum_{r=0}^{j} \Big(\underbrace{\widetilde{\beta}_r^{j+1,k} - \widetilde{\beta}_{r+1}^{j+1,k}}_{\widetilde{\alpha}_r^{j,k}}\Big) \cdot \Pr_\rho\Big(\underbrace{\widehat{\mathrm{rank}}_{j+1}(\mathbf{x}) = r+1 \text{ and } \widehat{\mathrm{rank}}_j(\mathbf{x}) = r}_{\equiv \left\{\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \text{ and } \widehat{\mathsf{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{9}{10}\right\}} \;\Big|\; \Pi_{j+1}\Big) \quad (32)
\end{aligned}$$

The last equation uses the fact that $\widehat{\mathrm{rank}}_j$ only depends on the models in Π_j. The second term on the RHS of the above equation can be further decomposed as,

$$\begin{aligned}
&\sum_{r=0}^{j} \widetilde{\alpha}_r^{j,k} \cdot \Pr_\rho\Big(\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \text{ and } \widehat{\mathsf{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{9}{10} \;\Big|\; \Pi_{j+1}\Big) \\
&= \mathbb{E}_\rho\Big[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \cdot \mathbb{I}\Big(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{9}{10}\Big) \;\Big|\; \Pi_{j+1}\Big] \\
&\overset{(a)}{=} \mathbb{E}_\rho\Big[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \;\Big|\; \Pi_{j+1}\Big] \cdot \Pr_{\widetilde{\rho}_j^\star}\Big(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{9}{10} \;\Big|\; \widehat{\pi}^j\Big) \\
&\overset{(b)}{=} (1 - \mathsf{err}_j) \times \sum_{r=0}^{j} \widetilde{\alpha}_r^{j,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \;\big|\; \Pi_j\big)
\end{aligned}$$

where in (a), we use several facts. Firstly that $\widehat{\mathrm{rank}}_j$ only depends on the models in Π_j, so we can change the conditioning from $\Pi_{j+1} \to \Pi_j$. Secondly, recalling that $\widetilde{\rho}_j^\star$ (cf. Eq. (18)) is the distribution satisfying $\widetilde{\rho}_j^\star(\mathbf{x}) \propto \rho(\mathbf{x}) \cdot \mathbb{E}[\widetilde{w}_j(\mathbf{x}) \mid \mathbf{x}, \Pi_j]$, for any (possibly randomized) test function, $g(\cdot)$, by definition of $\widetilde{w}_j$,

$$\mathbb{E}_{\mathbf{x}\sim\widetilde{\rho}_j^\star}[g(\mathbf{x}) \mid g] = \mathbb{E}_{\mathbf{x}\sim\rho}[g(\mathbf{x}) \cdot \mathbb{E}[\widetilde{w}_j(\mathbf{x}) \mid \mathbf{x}, \Pi_j] \mid g] = \mathbb{E}_\rho\Big[g(\mathbf{x}) \cdot \widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \;\Big|\; g, \Pi_j\Big]$$

which is also used within (a). In (b) we use the definition of $\mathtt{err}_j$ from theorem 24. Combining back with Eq. (32), noting that $\widetilde{\alpha}_r^{j,k} = \widetilde{\beta}_r^{j+1,k} - \widetilde{\beta}_{r+1}^{j+1,k}$, and the recursion for $\widetilde{\beta}_r^{j,k}$ in Eq. (17) and rearranging,

$$\begin{aligned}
\widetilde{\Phi}_{j+1} &= \sum_{r=0}^{j} \Big(\mathtt{err}_\star \cdot \widetilde{\beta}_r^{j+1,k} + (1-\mathtt{err}_\star) \cdot \widetilde{\beta}_{r+1}^{j+1,k}\Big) \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \;\big|\; \Pi_j\big) \\
&\qquad + (\mathtt{err}_j - \mathtt{err}_\star) \sum_{r=0}^{j} \widetilde{\alpha}_r^{j,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \;\big|\; \Pi_j\big) \\
&= \sum_{r=0}^{j} \widetilde{\beta}_r^{j,k} \cdot \Pr_\rho\big(\widehat{\mathrm{rank}}_j(\mathbf{x}) = r \;\big|\; \Pi_j\big) + (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_\rho\Big[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \;\Big|\; \Pi_j\Big] \\
&= \widetilde{\Phi}_j + (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_\rho\Big[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \;\Big|\; \Pi_j\Big],
\end{aligned}$$

Summing from $j = 0$ to $j = k-1$, we arrive at the equation,

$$\widetilde{\Phi}_k = \widetilde{\Phi}_0 + \sum_{j=0}^{k-1} (\mathtt{err}_j - \mathtt{err}_\star) \cdot \mathbb{E}_\rho\Big[\widetilde{\alpha}^{j,k}_{\widehat{\mathrm{rank}}_j(\mathbf{x})} \;\Big|\; \Pi_j\Big]$$

Finally plugging in the explicit formula for $\widetilde{\Phi}_0$ and $\widetilde{\Phi}_k$ completes the proof.

C.2.2. PROOF OF THEOREM 24

Recall from Theorem 5, that the learning rule $\mathsf{Alg}(\cdot\|\varepsilon', \delta', \mathtt{CoT})$ has sample complexity $n_{\mathtt{prompt}}(\varepsilon', \delta', T)$. Assuming that $\mathtt{ABORT}[j]$ is false in iteration j, the size of the dataset D^j_{out} is larger than $n_{\mathtt{prompt}}(1/400, \delta/k, T)$. This implies that with probability $1 - \frac{\delta}{k}$ the model $\widehat{\pi}^j$ trained in iteration j satisfies,

$$\mathtt{Acc}_{\widetilde{\rho}_j^\star}\big(\widehat{\pi}^j_T\big) \geqslant \frac{399}{400}. \tag{33}$$

This inequality uses the fact that prompts in the dataset D^j_{out} fed into the base learner $\mathsf{Alg}(\cdot\|\varepsilon', \delta', \mathtt{CoT})$ are drawn from the distribution $\widetilde{\rho}_j^\star$ (Eq. (18)) via rejection sampling from ρ. By an application of Markov's inequality, Eq. (33) translates into the following guarantee on $\widehat{\pi}^j$,

$$\Pr_{\mathbf{x}\sim\widetilde{\rho}_j^\star}\Big(\mathtt{Acc}_{\mathbf{x}}\big(\widehat{\pi}^j\big) \geqslant \frac{19}{20}\Big) \geqslant \frac{19}{20}. \tag{34}$$

Finally, we will use this to show that with some slack across both constants in the above inequality, we have,

$$\Pr_{\mathbf{x}\sim\widetilde{\rho}_j^\star}\Big(\widehat{\mathtt{Acc}}_{\mathbf{x}}\big(\widehat{\pi}^j\big) \geqslant \frac{9}{10}\Big) \geqslant \frac{9}{10},$$

which is the statement of the lemma. In order to show this, we first argue that for any $\mathbf{x} \in \mathcal{X}$,

$$\Pr\left(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{9}{10} \;\middle|\; \mathbf{x}, \widehat{\pi}^j,\ \mathsf{Acc}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{19}{20}\right) \geqslant \frac{19}{20} \tag{35}$$

Note that $\widehat{\mathsf{Acc}}_{\mathbf{x}}$ is computed as a Monte Carlo estimate, and is thereby an average of m Bernoulli random variables each having mean $\mathsf{Acc}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{19}{20}$. By Chernoff bound, as long as m is a sufficiently large constant (which it is chosen to satisfy within Algorithm 3), Eq. (35) holds with probability at least $\frac{19}{20}$. Taking an expectation on both sides of Eq. (35),

$$\Pr_{\mathbf{x}\sim\widetilde{\rho}^\star_j}\left(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{9}{10}\right) \geqslant \frac{19}{20} \times \Pr_{\mathbf{x}\sim\widetilde{\rho}^\star_j}\left(\mathsf{Acc}_{\mathbf{x}}(\widehat{\pi}^j) \geqslant \frac{19}{20}\right) \overset{(a)}{\geqslant} \left(\frac{19}{20}\right)^2 \geqslant \frac{9}{10}$$

where (a) follows from Eq. (34). This completes the proof.

C.2.3. PROOF OF THEOREM 25

By definition of $\widetilde{p}_j$ (Eq. (21)), and by the structure of the `Sample` subroutine (Algorithm 1 in Algorithm 1), the size of the dataset $|D^j_{\mathsf{out}}|$ which the model $\widehat{\pi}^j$ is trained on, can be expressed as the sum of $n' = |D^j_{\mathsf{prompt}}| = |D_{\mathsf{prompt}}|/k$ i.i.d. Bernoulli random variables, each with mean $\widetilde{p}_j$. Indeed, $\widetilde{p}_j$ is the probability that $\mathbf{x} \sim \rho$ is accepted into the dataset D^j_{out} (cf. Algorithm 2 of Algorithm 2). By an application of the multiplicative Chernoff bound,

$$\Pr\left(|D^j_{\mathsf{out}}| \leqslant \frac{n'\widetilde{p}_j}{2} \;\middle|\; \mathcal{H}_{j-1}\right) \leqslant \exp\left(-\frac{n'\widetilde{p}_j}{8}\right)$$

By the sufficiently large choice of $|D_{\mathsf{prompt}}| = n'k$ in the statement of this lemma, when $\widetilde{p}_j \geqslant \frac{\varepsilon}{16\sqrt{k}}$,

1. $n'\widetilde{p}_j \geqslant 8\log(k/\delta)$, and,
2. $n'\widetilde{p}_j \geqslant n_{\mathsf{prompt}}(1/400, \delta/k, T, \mathsf{CoT})$.

Together with the definition of $\widetilde{\mathsf{ABORT}}[j]$ in Eq. (20), these inequalities imply,

$$\Pr(\widetilde{\mathsf{ABORT}}[j] \mid \mathcal{H}_{j-1}) \leqslant \frac{\delta}{k}$$

C.2.4. PROOF OF THEOREM 26

Let $\mathcal{E}$ denote the event that $\sum_{\pi\in\Pi_k} \mathbb{I}(\mathsf{Acc}_{\mathbf{x}}(\pi) \geqslant \frac{4}{5}) \leqslant \frac{3k}{4}$. Let $Z = \sum_{\pi\in\Pi_k} \mathbb{I}(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10})$. Under the event $\mathcal{E}$, for at most $\frac{3k}{4}$ choices of $\pi \in \Pi_k$, we have that $\mathsf{Acc}_{\mathbf{x}}(\pi) \geqslant \frac{4}{5}$. This implies that,

$$\mathbb{E}[Z \mid \mathbf{x}, \Pi_k, \mathcal{E}] = \mathbb{E}\left[\sum\nolimits_{\pi\in\Pi_k} \mathbb{I}\left(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\right) \;\middle|\; \mathbf{x}, \Pi_k, \mathcal{E}\right] \leqslant \frac{3k}{4} + \frac{k}{4} \times \frac{1}{10} = \frac{31k}{40}$$

where the last equation follows from the choice of m within the Monte Carlo estimate in Algorithm 3 being a sufficiently large constant, so that $\Pr(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10} | \mathbf{x}, f, \mathsf{Acc}_{\mathbf{x}}(\pi) \leqslant \frac{4}{5}) \leqslant \frac{1}{10}$. Furthermore, note that $\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi)$ is independent across $\pi \in \Pi_k$ conditioned on $\mathbf{x}$ and Π_k, which implies that,

$$\mathbf{Var}[Z \mid \mathbf{x}, \Pi_k, \mathcal{E}] = \mathbf{Var}\left[\sum\nolimits_{\pi\in\Pi_k} \mathbb{I}\left(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\right) \;\middle|\; \mathbf{x}, \Pi_k, \mathcal{E}\right] \leqslant \frac{31k}{40}$$

Noting that the variance of a sum of independent Bernoulli random variables is upper bound by its mean. Therefore, by an application of Chebyshev's inequality,

$$\Pr\left(Z \leqslant \mathbb{E}[Z] + 2\sqrt{\mathbf{Var}[Z]} \;\middle|\; \mathbf{x}, \Pi_k, \mathcal{E}\right) \geqslant \frac{1}{2}$$

Plugging in the upper bound on $\mathbb{E}[Z]$ and $\mathbf{Var}[Z]$, and choosing k to be at least a sufficiently large absolute constant so that $\mathbb{E}[Z] + 2\sqrt{\mathbf{Var}[Z]} \leqslant \frac{31k}{40} + 2\sqrt{\frac{31k}{40}} \leqslant \frac{4k}{5}$, we have that,

$$\Pr\left(\sum\nolimits_{\pi \in \Pi_k} \mathbb{I}\left(\widehat{\mathsf{Acc}}_{\mathbf{x}}(f) \geqslant \frac{9}{10}\right) \leqslant \frac{4k}{5} \;\middle|\; \mathbf{x}, \Pi_k, \mathcal{E}\right) \geqslant \frac{1}{2}$$

Finally, multiplying both sides by $\Pr(\mathcal{E}|\mathbf{x}, \Pi_k)$ and taking an expectation over $\mathbf{x} \sim \rho$, we have that,

$$\Pr\nolimits_{\mathbf{x}\sim\rho}(\mathcal{E}|\Pi_k) \leqslant 2 \cdot \Pr\nolimits_{\rho}\left(\sum\nolimits_{\pi \in \Pi_k} \mathbb{I}\left(\widehat{\mathsf{Acc}}_{\mathbf{x}}(\pi) \geqslant \frac{9}{10}\right) \leqslant \frac{4k}{5} \;\middle|\; \Pi_k\right)$$

Completing the proof of the result.

C.2.5. PROOF OF THEOREM 27

Let $Z_1, \cdots, Z_k$ denote a sequence of k biased coins with probability of heads equal to $1 - \mathsf{err}_\star = \frac{9}{10}$. By following the same argument as in Theorem 18, we have an explicit form for $\widetilde{\beta}_r^{j,k}$ as equal to $\Pr\left(S_k \leqslant \frac{4k}{5} \;\middle|\; S_j = r\right)$ where $S_j = \sum_{i=1}^{j} \mathbb{I}(Z_i = \mathsf{H})$. Then,

$$\begin{aligned}
\widetilde{\alpha}_{\max}^{j,k} &= \max_{0 \leqslant r \leqslant j} \left(\widetilde{\beta}_r^{j+1,k} - \widetilde{\beta}_{r+1}^{j+1,k}\right) \\
&\leqslant \max_{0 \leqslant r \leqslant j} \Pr\left(S_k \leqslant \frac{4k}{5} \;\middle|\; S_{j+1} = r\right) - \Pr\left(S_k \leqslant \frac{4k}{5} \;\middle|\; S_{j+1} = r+1\right) \\
&\overset{(a)}{=} \max_{0 \leqslant r \leqslant j} \Pr\left(S_k - S_{j+1} \leqslant \frac{4k}{5} - r\right) - \Pr\left(S_k - S_{j+1} \leqslant \frac{4k}{5} - (r+1)\right) \\
&= \max_{0 \leqslant r \leqslant j} \Pr\left(S_k - S_{j+1} = \frac{4k}{5} - r\right) \qquad (36) \\
&\overset{(b)}{\leqslant} \frac{1}{\sqrt{2\pi \cdot \mathsf{err}_\star(1 - \mathsf{err}_\star) \cdot (k - j - 1)}} \\
&\leqslant \frac{2}{\sqrt{k - j - 1}}
\end{aligned}$$

where (a) uses the fact that $S_k - S_{j+1}$ and S_{j+1} are independent, while (b) uses a standard upper bound on the Binomial PMF using the Stirling approximation for $j < k - 1$. When $j = k - 1$, Eq. (36) gives us an upper bound of 1.

C.3. Proofs for Lemmas from Theorem 14

C.3.1. PROOF OF THEOREM 32

Recall the definition, $\mathcal{Y}(\mathbf{x}) = \left\{\mathbf{y} \in \mathcal{Y} : \pi_{\text{ref}}(\mathbf{y}|\mathbf{x}) \geqslant C_{\text{seq}}^{-1}\right\}$. By the pigeonhole principle, $|\mathcal{Y}(\mathbf{x})| \leqslant C_{\text{seq}}$. With this, we may rewrite $\mathcal{B}_{\mathcal{G}}(D)$ as,

$$\mathcal{B}_{\mathcal{G}}(D) = \left\{\left(\mathbb{I}\left(\pi_T(\mathbf{x}_i) = \pi_T^\star(\mathbf{x}_i)\right) \cdot \mathbb{I}\left(\pi_{1:T}(\mathbf{x}_i) \in \mathcal{Y}(\mathbf{x}_i)\right) : i \in [n]\right) : \pi \in \Pi\right\}$$

Now, define the following list of tables, $\mathcal{T}(D)$:

$$\mathcal{T}(D) = \left\{ \left(\pi(\mathbf{x}_i, \mathbf{y}^i_{1:t-1}) : t \in [T], i \in [n], \mathbf{y}^i \in \mathcal{Y}(\mathbf{x}_i) \right) : \pi \in \Pi \right\}$$

We will prove two claims:

Claim C.1 *$|\mathcal{T}(D)| \leqslant (enT|\Sigma|C_{\text{seq}})^d$ where $d = \text{Ndim}(\Pi)$ is the Natarajan dimension of Π.*

Proof $\mathcal{T}(D)$ captures the number of ways in which Π labels a fixed set of nTC_{seq} prefixes. The proof of this claim follows by a generalization of the Sauer-Shelah lemma to multiclass predictors (Haussler and Long, 1995). ■

Claim C.2 *For any fixed D, there exists a surjection from $\mathcal{T}(D) \to \mathcal{B}_{\mathcal{G}}(D)$.*

Proof We will argue that if for any $\pi \in \Pi$, we are given the corresponding table $\left(\pi(\mathbf{x}_i, \mathbf{y}^i_{1:t-1}) : t \in [T], i \in [n], \mathbf{y}^i \in \mathcal{Y}(\mathbf{x}_i) \right) \in \mathcal{T}(D)$ and also $\{\mathcal{Y}(\mathbf{x}_i) : i \in [n]\}$ and $\{\pi^\star_T(\mathbf{x}_i) : i \in [n]\}$ (but there is no explicit identification of π itself), we can compute $\mathbb{I}\left(\pi_T(\mathbf{x}_i) = \pi^\star_T(\mathbf{x}_i)\right) \cdot \mathbb{I}\left(\pi_{1:T}(\mathbf{x}_i) \in \mathcal{Y}(\mathbf{x}_i)\right)$ for this π. The procedure is as follows.

- First note that we can infer $y^i_1 = \pi(\mathbf{x}_i)$ from the table we are given. Looking at the table $\mathcal{Y}(\mathbf{x}_i)$, we can identify if there exists a $\mathbf{z}^{i,1} \in \mathcal{Y}(\mathbf{x}_i)$ such that $\mathbf{z}^{i,1}_1 = y_1$.
- If no such $\mathbf{z}^{i,1}$ exists, the procedure terminates, and we assert that $\mathbb{I}\left(\pi_{1:T}(\mathbf{x}_i) \in \mathcal{Y}(\mathbf{x}_i)\right) = 0$, since the partial CoT generated by π does not belong to the set of prefixes realized by strings in $\mathcal{Y}(\mathbf{x}_i)$.
- If some such $\mathbf{z}^{i,1}$ exists, we proceed by computing $y^i_2 = \pi(\mathbf{x}_i, \mathbf{z}^{i,1}_1)$, which is also present in the table. We again check if there exists a $\mathbf{z}^{i,2} \in \mathcal{Y}(\mathbf{x}_i)$ such that $\mathbf{z}^{i,2}_{1:2} = \mathbf{y}^i_{1:2}$. If no such $\mathbf{z}^{i,2}$ exists, we terminate and return 0. If it exists, we proceed to the next step.
- In any iteration t, if the procedure has not yet terminated, we have a candidate sequence $(^i_1, \cdots, y^i_{t-1})$, which is inductively assumed to compute the first $t-1$ symbols of $\pi_{1:T}(\mathbf{x}_i)$, and is also the prefix of some string $\mathbf{z}^{i,t-1} \in \mathcal{Y}(\mathbf{x}_i)$. We compute $y^i_t = \pi\left(\mathbf{x}_i, \mathbf{z}^{i,t-1}_{1:t-1}\right)$ by looking at the corresponding entry in $\mathcal{T}$. If $\mathbf{y}^i_{1:t} = \mathbf{z}^{i,t}_{1:t}$ for some $\mathbf{z}^{i,t} \in \mathcal{Y}(\mathbf{x}_i)$, we proceed to iteration $t+1$. If not, we terminate the procedure.

By the end of this process, we can compute $\pi_{1:T}(\mathbf{x}_i)$ if $\pi_{1:T}(\mathbf{x}_i) \in \mathcal{Y}(\mathbf{x}_i)$, or certify that $\pi_{1:T}(\mathbf{x}_i) \notin \mathcal{Y}(\mathbf{x}_i)$. In the former case, we may check whether $\mathcal{V}(\mathbf{x}_i, y) = 1$ for $y = \pi^\star_T(\mathbf{x}_i)$ (which is also fixed and does not depend on the π under consideration). Thus, regardless of which case we are in, it is possible to compute $\mathbb{I}\left(\pi_T(\mathbf{x}_i) = \pi^\star_T(\mathbf{x}_i)\right) \cdot \mathbb{I}\left(\pi_{1:T}(\mathbf{x}_i) \in \mathcal{Y}(\mathbf{x}_i)\right)$ given the table in $\mathcal{T}(D)$ corresponding to some π. ■

As a consequence of Claims C.1 and C.2, we arrive at the statement $|\mathcal{B}_{\mathcal{G}}(D)| \leqslant |\mathcal{T}(D)| \leqslant (enT|\Sigma|C_{\text{seq}})^d$, proving the lemma.