Leveraging Structural Uncertainty for Decision Making: From Classical Methods to Foundation Model Agents

Thesis by
Hao Liu

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended May 21, 2026

Hao Liu
ORCID: 0009-0005-4006-2585

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Yisong Yue. Thank you for your continuous support throughout this long journey, and for giving me the freedom to pursue the research directions that I am most passionate about. Your guidance, patience, and trust have shaped not only this thesis but also the way I think about research. The culture of curiosity and openness you fostered has made my PhD experience deeply meaningful.

I would also like to thank Yixin Wang for the many long discussions we shared. Your insights, encouragement, and perspective have been invaluable to my growth as a researcher. Our conversations shaped this thesis deeply.

My sincere thanks go to Anqi Liu for your encouragement and kindness. Your warmth and support made a real difference.

I am deeply grateful to my committee members, Katie Bouman and Adam Wierman. You have both been role models to me for a long time, and I am fortunate to have had your guidance and feedback for improving this thesis and my research.

I would like to thank all members of the YueCrew, past and present. It has been incredible to witness the growth and evolution of the group over the years, and it has been an honor to be part of this journey alongside such talented and inspiring people. The discussions, collaborations, and friendships formed within the group have made this experience unforgettable.

I am also grateful to all of my collaborators. The discussions and shared work we have had together are at the heart of this thesis, and I have learned tremendously from each of you. It has also been a privilege to engage with my current and former colleagues at Caltech. Together, you helped create a supportive, collaborative, and intellectually exciting environment that made these years deeply rewarding.

A special thank you to Maria Lopez of the CMS department for your professionalism and for all the help in coordinating the many procedures and administrative processes throughout my PhD journey.

Finally, my deepest gratitude goes to my parents and family. None of this would have been possible without your unconditional love and steadfast support. You have stood by my side through every stage of this journey, and this thesis is as much yours as it is mine.

ABSTRACT

Uncertainty is fundamental to machine learning and decision-making, but the appropriate way to model it depends on the type of risk or randomness being considered, which in turn is governed by the structure of the specific problem at hand. This thesis explores structural uncertainty across diverse decision-making settings, from classical frameworks such as contextual bandits and linear quadratic control to modern foundation model systems including tool-using LLM agents and automated scientific discovery, arguing that effective decision-making requires identifying and exploiting the structure underlying uncertainty. We begin with off-policy evaluation in contextual bandits, where distributional uncertainty has factorization structure. We introduce a distributionally robust approach that models the factorization of the data-generating process for improved uncertainty quantification and robustness under general covariate shift. Next, we study linear quadratic control with untrusted ML predictions, where system disturbances have compositional structure arising from heterogeneous latent sources. We introduce DISC, an online policy that simultaneously disentangles latent disturbance structure and learns per-component confidence parameters, achieving near-optimal consistency when predictions are accurate while maintaining worst-case robustness guarantees when they fail. For tool-using LLM agents, standard logit-based uncertainty estimation fails to capture agent-environment interaction structure. We develop PROBECAL, which probes LLM hidden states to extract structural features of execution traces, yielding calibrated probabilities over execution correctness and improving existing LLM reasoning techniques. For LLM-based symbolic regression, uncertainty structure lies in the environment feedback mechanism, where scalar metrics like MSE compress rich statistical structure into a single number. We develop PROAUG, enabling code-based data analysis that lets agents actively extract structural signals during evolutionary search, significantly improving discovery efficiency. Across these settings, a common principle emerges: uncertainty is inherently structured, and decision-making systems that explicitly model this structure achieve superior robustness, efficiency, and reliability. This thesis demonstrates that structural uncertainty provides a unifying foundation for trustworthy machine learning across classical and modern AI paradigms.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Hao Liu*, Xiao-Wen Yang*, Atharva Sehgal, Yixin Wang, Lan-Zhe Guo, Yu-Feng Li, and Yisong Yue. "Programmatic Context Augmentation for LLM-based Symbolic Regression". In: *arXiv preprint arXiv:2605.03101* (2026). URL: https://arxiv.org/pdf/2605.03101.
H.L. participated in the conception of the project, formulated and implemented the method, conducted experiments and analyzed results, and participated in the writing of the manuscript.

[2] Yihong Guo*, Hao Liu*, Yisong Yue, and Anqi Liu. "Distributionally Robust Policy Evaluation under General Covariate Shift in Contextual Bandits". In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856. URL: https://arxiv.org/pdf/2401.11353.
H.L. participated in the conception of the project, the formulation and implementation of the method, analyzed experiment results, and participated in the writing of the manuscript.

[3] Tongxin Li*, Hao Liu*, and Yisong Yue. "Disentangling linear quadratic control with untrusted ml predictions". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 86860–86898. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/9dff3b83d463fab213941bfee23341ba-Paper-Conference.pdf.
H.L. participated in the conception of the project, implemented the method, conducted experiments and analyzed results, and participated in the writing of the manuscript.

[4] Hao Liu, Zi-Yi Dou, Yixin Wang, Nanyun Peng, and Yisong Yue. "Uncertainty calibration for tool-using language agents". In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. 2024, pp. 16781–16805. URL: https://aclanthology.org/2024.findings-emnlp.978.pdf.
H.L. participated in the conception of the project, formulated and implemented the method, conducted experiments and analyzed results, and participated in the writing of the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Number *Page*

LIST OF TABLES

Chapter 1

INTRODUCTION

Uncertainty is a core component of machine learning and decision-making. Whether the task is evaluating a policy from historical data, controlling a physical system with imperfect ML predictions, choosing how to use an external tool, or searching for a hypothesis that best fits observed scientific data, an agent must act under incomplete information about the world. How the agent represents and manages this uncertainty determines whether it can perform in a reliable, effective, and principled way.

Crucially, uncertainty is not a single, uniform object—it is fundamentally shaped by the type of randomness or risk being modeled. The relevant risks and the appropriate ways to address them depend intimately on the structure of the specific problem at hand. A policy evaluation task faces distributional shift between logging and deployment environments. A control system faces disturbances drawn from heterogeneous latent sources whose composition is unknown. A tool-using language agent faces a miscalibration problem where interacting with external tools introduces misalignments between the agent's internal confidence and actual execution outcomes. An LLM agent performing evolutionary search for scientific equation discovery faces a bottleneck where rich dataset structure is compressed into scalar feedback, starving the search process of the signals it needs. In each of these settings, exploiting the specific structural form of uncertainty is the key to building systems that handle decision-making both effectively and reliably.

This motivates the central theme of this thesis: structural uncertainty. We argue that effective decision-making requires identifying and leveraging the structural properties that govern uncertainty in each problem setting. This thesis explores structural uncertainty across a range of decision-making problems, spanning classical machine learning frameworks and modern foundation model agents, united by a common methodology: identify the specific structural form of uncertainty in the problem, build methods that model and leverage that structure, and demonstrate through theory and experiments that the structural approach outperforms unstructured alternatives.

The four chapters of this thesis fall into two methodological paradigms, summarized in Table 1.1. Chapters 2 and 3 address classical decision-making settings where

uncertainty can be tackled through probabilistic modeling of the data-generating process–specifically, by identifying and exploiting distributional structure in how data is generated. Chapter 2 models the factorization of the data-generating process under distribution shift for policy evaluation, while Chapter 3 models the disentanglement of latent variables in system disturbances for online control. Chapters 4 and 5 focus on LLM agent settings, where formal data-generating models are typically unavailable. Here, we inject and exploit structure in the interaction between the LLM agent and its environment: Chapter 4 utilizes probing of LLM internal states to model agent-environment interaction for improved uncertainty calibration and integration into LLM reasoning, while Chapter 5 augments the environment feedback mechanism with programmatically extracted statistical structural context.

Table 1.1: Structural uncertainty across decision-making settings.

Chapter	**Task**	**Structure Setting**	**Method**
Classical Methods: Modeling Data-Generating Process Structure			
2	Offline Policy Evaluation	Factorization of distribution shift	Distributionally Robust Regression
3	Online Control	Compositional latent disturbances	Disentangled Confidence Policy (DISC)
LLM Agents: Injecting Structure into Agent-Environment Interaction			
4	Tool-Using LLM Agents	LLM internal states for interaction modeling	Probing Calibration (PROBECAL)
5	Symbolic Regression	Statistical structure in environment feedback	Programmatic Context Augmentation (PROAUG)

1.1 Factorization of Distribution Shift in Offline Policy Evaluation

Off-policy evaluation (OPE) is fundamentally a problem of reasoning under distributional uncertainty between the logging and deployment environments. Distributional uncertainty in OPE has factorization structure: it decomposes into shift in the context distribution and shift in the policy distribution. Standard approaches address only policy shift through inverse propensity scoring, implicitly assuming contexts are drawn from the same distribution during logging and deployment. This accounts for only half of the distributional discrepancy. In many real deployments, the context distribution also shifts between environments—for example, when user populations change. This creates general covariate shift, where modeling the compound effect of both shifts becomes necessary.

In Chapter 2, we identify and address this structural uncertainty by analyzing the factorization of the data-generating process that underlies counterfactual reasoning.

In the contextual bandit setting, observations are generated by sampling a context, then sampling an action given that context, then receiving a reward. We reframe policy evaluation through the lens of covariate shift, which models the distribution shift of a dependent variable when the marginal distribution of the covariates differs between training and test time, but the conditional output distribution given the covariates remains the same. Policy evaluation can be interpreted as a setting where the dependent variable is the reward and the covariates are the contexts and actions. Based on this formulation, we develop a novel family of distributionally robust policy evaluation estimators built on robust regression techniques, covering both policy shift and general covariate shift settings. By integrating our robust reward model into established evaluation frameworks, including direct methods and doubly robust methods, we obtain estimators with finite-sample bias bounds that provide clear advantages when shifts are large. Comprehensive empirical evaluation across diverse shift magnitudes and policy configurations demonstrates that our structured approach significantly and consistently outperforms baseline methods.

1.2 Disentanglement of Latent Disturbances in Online Control

Disturbance uncertainty in control systems often originates from compositional structure: multiple heterogeneous latent variables combine to produce observed perturbations. Each latent component represents a distinct source of disturbance from often independent resources or factors. For example, a drone navigating outdoors faces external perturbations caused by a mixture of predictable elements like airflows and less predictable factors like raindrops. When machine learning prediction models are applied to model these disturbances, their reliability differs across latent sources: stable components may be accurately predicted while volatile ones remain unpredictable. This compositional structure creates a challenge: without knowing which latent variables generated each observation, a controller cannot appropriately calibrate its trust in predictions.

In Chapter 3, we develop DISC (Disentangled Confidence policy), an online learning policy that tackles this challenge by explicitly modeling the latent structure of disturbances for more precise uncertainty estimation. DISC simultaneously disentangles the disturbance signal and adaptively learns per-component confidence parameters for the corresponding ML predictions. This allows the controller to calibrate its reliance on predictions at the component level. DISC achieves a near-optimal consistency-robustness tradeoff, which provably improves what can be obtained without learning the confidence parameters. We demonstrate the

effectiveness and reliability of DISC through competitive ratio theoretical analysis, as well as two real-world applications: a drone navigation problem with mixed external disturbances and voltage control in a power grid with heterogeneous power injections, where it outperforms baselines that do not distinguish between underlying latent variables.

1.3 Calibration through Internal State Probing in Tool-Using Language Agents

Tool-using language agents solve complex problems by interacting with external environments through tools that provide capabilities such as code execution and API calls. Uncertainty in tool-using settings arises not just from the model's internal knowledge limitations, but from the complex interplay between the agent's internal confidence and its interactions with the external environment. Standard approaches typically use the LLM's token logits as the uncertainty signal. While logit-based signals carry information about the model's next-token predictions, they fail to capture whether a full tool-call execution path is correct—correctness can hinge on a single specific token or on the precise sequence of interactions with the external environment. Two rollout trajectories can be textually nearly identical, with highly similar token logits, yet one correctly invokes tools at a critical position while the other does not. Distinguishing between such trajectories requires a structural approach to extracting trajectory-level features that model the interaction between the LLM and its environment.

In Chapter 4, we develop PROBECAL, a probing-based calibration framework for tool-using language agents. Drawing on techniques from the interpretability literature for understanding LLMs' internal structure, PROBECAL trains lightweight probe classifiers on hidden states at intermediate layers of the LLM, capturing structural features of the rollout and producing calibrated probability estimates that serve as uncertainty quantification over execution trace correctness. We further develop uncertainty-aware mechanisms that integrate our calibration method into prompt selection and execution trace ranking, improving upon traditional LLM reasoning approaches such as self-consistency. PROBECAL delivers significant and consistent improvements across diverse tool-using benchmarks.

1.4 Structured Context in LLM-based Symbolic Regression

Symbolic regression studies the task of inferring mathematical equations that describe observed data. Recent work has shown that LLMs can serve as effective hypothesis generators in evolutionary search for symbolic regression, reflecting a long-horizon

decision-making under uncertainty task in which LLMs must iteratively search hypotheses in a high-dimensional, combinatorial space with feedback signals from the environment throughout the search process. The structure of uncertainty here is in the environment feedback mechanism. When an LLM agent proposes candidate equations, each is evaluated against the dataset and receives only a scalar loss (typically mean squared error) as feedback. This compresses rich statistical structure in the dataset, such as variable correlations, functional behaviors, and power-law relationships, into a single number. This deficiency of structure in the environment feedback mechanism creates a fundamental information constraint that limits the agent's ability to reason about the dataset and guide its search effectively.

In Chapter 5, we develop PROAUG (Programmatic Context Augmentation), a framework that augments the feedback mechanism with dataset structure. PROAUG assigns the LLM agent a dual role: in addition to generating candidate equations, the agent generates and executes data analysis code that actively extracts structural signals from the dataset. These extracted signals–such as statistical properties and variable correlations–are assembled into rich contextual augmentation that is fed back into the hypothesis generation prompt. This transforms the agent from a passive recipient of scalar feedback into an active explorer of dataset structure, achieving substantially improved discovery efficiency and accuracy over strong baselines on challenging benchmarks.

Chapter 2

DISTRIBUTIONALLY ROBUST POLICY EVALUATION UNDER GENERAL COVARIATE SHIFT

This chapter is based on the paper:
Yihong Guo*, Hao Liu*, Yisong Yue, Anqi Liu. "Distributionally Robust Policy Evaluation under General Covariate Shift in Contextual Bandits." In: *Transactions on Machine Learning Research (2024)*.

2.1 Introduction

Contextual bandits are online learning algorithms where a policy learner repeatedly observes a context, takes an action, and receives a reward only for the selected action [22]. It has a wide range of real-world applications, including recommender systems [16, 24, 52], online advertising [6, 45], experimental design [21], and medical interventions [23, 46]. Evaluating policy performance is essential for these applications, but assessing the value through direct online deployment can be costly and impractical. This motivates the important task of policy evaluation from historical, pre-collected data.

The key challenge in policy evaluation using pre-collected data lies in addressing the counterfactual reasoning that arises from the divergence between the distributions of the logging and target data. Specifically, the actions selected by the target policy may not be covered in the pre-collected logging data, and shifts in the context distribution can further complicate policy evaluation. When the distribution shift only occurs in the policy distribution, we recover the classic setting of off-policy evaluation (OPE). Existing OPE approaches can be divided into three main categories: (1) inverse propensity scoring (IPS), which uses importance weights adjustment [15, 43]; (2) direct methods (DM), which directly regress the value of a target policy by learning a conditional reward model; and (3) doubly robust (DR) methods, which integrate DM and IPS [2, 10, 11, 39, 50], achieving asymptotic consistency if either DM or IPS is consistent. In both DM and DR methods, a high-quality reward estimation for each context-action pair in the target data can be beneficial. However, compared to the rich literature on improving the IPS component [39, 43, 50], less work has explored how to integrate a carefully designed conditional reward model in OPE [12].

Beyond the standard setting of policy shift discussed above, very few studies have explored distribution shift in the context variables. [48] suggested incorporating the context distribution into IPS estimation for both IPS and DR methods when a context shift occurs and may suffer from similar issues like high variance as traditional OPE methods. However, their approach does not address how a shift in context distribution might influence reward estimation. Recently, distributionally robust methods have been introduced to enhance the robustness of the evaluation by constructing KL-divergence uncertainty sets to capture potential distribution shifts [18, 36] in the joint distribution of context, action, and reward. However, the substantially large uncertainty set that characterizes the shift may lead to over-conservativeness in practice.

In this paper, we tackle the problem of policy evaluation through the lens of covariate shift [35]. Covariate shift refers to the modeling of a dependent variable when the marginal distribution of the covariates differs between training and test time but the conditional output distribution on the covariates stays the same. Policy evaluation can be interpreted as learning under the covariate shift, where the dependent variable is the reward, the covariates are the contexts and actions, and we are interested in learning a robust conditional reward model $\hat{r}(\boldsymbol{x}, a) \sim P(r|\boldsymbol{x}, a)$ that performs well under distribution shift. Specifically, we study the settings where shifts can occur in policy distribution alone (the standard setting), and both policy and context distributions, which we refer to as *Policy Shift* (PS) and *General Covariate Shift* (GCS), respectively. Building on recent progress in robust regression under covariate shift [7, 27], we introduce a novel family of distributionally robust policy evaluation estimators, which is created by seamlessly integrating the conditional reward model, obtained from robust regression, into existing DM and DR methods.

Our contributions. We formulate the task of policy evaluation using the framework of covariate shift and incorporate a robust regression method into the policy evaluation setting where shifts can happen in both policy and context distribution. Leveraging a base distribution, the resulting shift-aware conditional reward model allows for a conservative evaluation of policy value in scenarios where the context-action pair in target data is not encompassed by the pre-collected logging data.

We develop direct methods based on the proposed shift-aware conditional reward model, which can be easily integrated into various doubly robust methods. We establish the finite sample upper bound results for the bias of the proposed method,

demonstrating the proposed method's robustness under large distribution shifts, which is further validated in the experiments.

We conduct extensive experiments spanning 360 policy shift conditions and 1260 total conditions for various combinations of logging, target policies, and covariate shifts on data. In our comparison against strong baseline methods on standard benchmark datasets, our method demonstrates superior performance under distribution shifts across various conditions, outperforming the baseline methods in over 90% of cases under policy shifts, and over 72% of the instances in general covariate shift settings.

2.2 Problem Formulation

Policy Evaluation for Contextual Bandits. In contextual bandits, a policy π iteratively observes a context $\boldsymbol{x}$, takes an action a, and receives a scalar reward $r_{\boldsymbol{x},a}$. Assuming that the contexts $\boldsymbol{x}$ are independent and identically distributed (i.i.d.), the target object, the value of the target policy, can be expressed as:

$$V^T = \mathbb{E}_{\boldsymbol{x}\sim P_t(\boldsymbol{x}),a\sim\pi(a|\boldsymbol{x})}\left[r_{\boldsymbol{x},a}\right], \tag{2.1}$$

where $\boldsymbol{x} \sim P_t(\boldsymbol{x})$ denotes some known exogenous target context distribution (e.g., profiles of users), and $a \sim \pi(a|\boldsymbol{x})$ denotes the stochasticity of the known target policy. Ususally, we assume access to a pre-collected dataset of n tuples of the form: $S = \{(\boldsymbol{x}, a, r)\}$, where $\boldsymbol{x} \sim P_s(\boldsymbol{x})$, $a \sim \beta(a|\boldsymbol{x})$, and r is the reward observed. Then given S and π, the goal of policy evaluation is to reliably estimate $\hat{V}_S^T \equiv \hat{V}$ of V^T in Equation 2.1. In traditional OPE, the goal is to estimate V^T offline using pre-collected historical data from some other (possibly unknown) logging policy $\beta(a|\boldsymbol{x})$ on the context distribution $\boldsymbol{x} \sim P_s(\boldsymbol{x})$, with the assumption that $P_s(\boldsymbol{x}) = P_t(\boldsymbol{x})$. When we consider context distribution shift, we further assume $P_s(\boldsymbol{x}) \neq P_t(\boldsymbol{x})$. Here, s and t stand for source and target distribution, respectively.

Policy Evaluation as Covariate Shift.

Covariate shift is a special case of distribution shift where only the distribution of the covariates changes, but the conditional output distribution remains unaltered [35]. In the setting of this paper, our goal is to evaluate policy π over a potentially different context distribution $P_t(\boldsymbol{x})$, as depicted in the data-generating process in Figure 2.1. The assumption can be expressed as $P_s(\boldsymbol{x}, a) \neq P_t(\boldsymbol{x}, a)$ and $P_s(r|\boldsymbol{x}, a) = P_t(r|\boldsymbol{x}, a)$. Note we use "logging" and "source" distribution interchangeably. Based on this data-generating process, the joint distribution of covariates ($\boldsymbol{x}$ and a in this case) can be described as shifting from $P_s(\boldsymbol{x}, a) = P_s(\boldsymbol{x})\beta(a|\boldsymbol{x})$ to $P_t(\boldsymbol{x}, a) = P_t(\boldsymbol{x})\pi(a|\boldsymbol{x})$. We

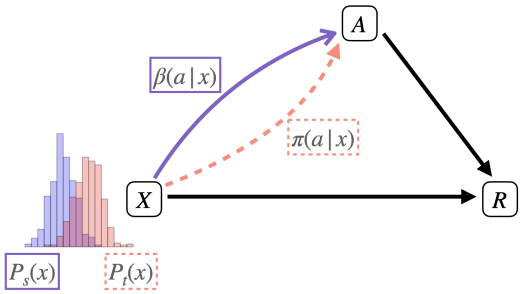


Figure 2.1: Data generating process for policy evaluation under general covariate shift. X, A, and R represent context, action, and reward variables, respectively. Logging data distribution $P_s(x)\beta(a|\boldsymbol{x})$ is marked with solid purple lines, and target data distribution $P_t(x)\pi(a|\boldsymbol{x})$ is marked with dashed red lines. Black lines indicate reward is generated by both contexts and actions.

can further assume that the distribution shift problem of policy evaluation can be decomposed into two different mechanisms [31, 33]: (1) Context distribution shift, and (2) Policy shift. General covariate shift (GCS) refers to the setting when both context distribution shift and policy shift occur. As the ground truth conditional reward distribution $P^*(r|\boldsymbol{x}, a)$, that we aim to learn, remains unchanged, we can formulate the reward estimation problem in policy evaluation as a covariate shift problem. To simplify the notations, We omit $\boldsymbol{x} \sim P(\boldsymbol{x})$ sometimes in the OPE setting. We further define $P_s(\boldsymbol{x}, a, r) = P_s(\boldsymbol{x}, a)P^*(r|\boldsymbol{x}, a)$ and $P_t(\boldsymbol{x}, a, r) = P_t(\boldsymbol{x}, a)P^*(r|\boldsymbol{x}, a)$.

2.3 Distributionally Robust Policy Evaluation under General Covariate Shift

In this section, we formally present our robust regression approach for reward prediction from logging data in policy evaluation, which is constructed from a distributionally robust learning framework. The high-level goal is to estimate a reward model $\hat{f}(\boldsymbol{x}, a)$ that can be robust to the worst-case data-generating distribution that can occur to maximize a loss function on the target data. We present the formulation and derived predictive form in Section 2.3 and the parameter learning algorithm in Section 2.3.

The Distributionally Robust Learning Formulation

ERM v.s. DRL. Traditional supervised learning frames reward estimation as an empirical risk minimization problem (ERM). ERM seeks to minimize the following

empirical risk on the logging data:

$$\hat{f}_{\text{sup}}(\boldsymbol{x}, a) = \arg\min \mathbb{E}_{P_s(\boldsymbol{x},a,r)} \left[\mathcal{L}(r, \hat{f}(\boldsymbol{x}, a))\right] \approx \arg\min \frac{1}{|S|} \sum_{i \in S} \mathcal{L}(r_i, \hat{f}(\boldsymbol{x}_i, a_i)), \tag{2.2}$$

where $\mathcal{L}$ is a loss function that measures the disparity between the actual reward and prediction from the model. However, this approach is susceptible to bias under covariate shift, as the logging data reflect reward information solely within the scope of the logging context and policy. To address this, the Distributionally Robust Learning (DRL) framework provides an alternative perspective by modeling the problem as a two-player minimax game over the target data, which involves a predictor player minimizing and an adversarial player maximizing the expected target loss as follows: $\hat{f}(\boldsymbol{x}, a) = \arg\min_{\hat{f}(\boldsymbol{x},a)} \max_{g(\boldsymbol{x},a) \in \Sigma_S} \mathbb{E}_{\boldsymbol{x},a \sim P_t(\boldsymbol{x},a), r \sim g} \left[\mathcal{L}(r, \hat{f}(\boldsymbol{x}, a))\right]$, where $\hat{r} \sim \hat{f}(\boldsymbol{x}, a)$ is the predictor player and $r \sim g(\boldsymbol{x}, a)$ is the adversarial player. The adversarial player is confined to reside in a constrained set Σ_S, defined on the training data S. Covariate shift results in a mismatch between the expected loss we are optimizing (defined on $P_t(x, a)$) and the constrained set for the adversary (defined on $P_s(x, a)$). The choice of the loss function $\mathcal{L}$ and the constrained set Σ_S shape the specific derived form of the predictor $\hat{f}(\boldsymbol{x}, a)$. Both f and g are valid conditional distributions of r given the context and the action $(\boldsymbol{x}, a)$.

DRL Formulation for Robust Regression. In this paper, we incorporate the following relative loss function for the reward estimation:

$$\mathcal{L}(r, \hat{f}) \coloneqq \mathbb{E}_{\boldsymbol{x},a \sim P_t(\boldsymbol{x},a), r \sim g} \left[-\log \hat{f} + \log f_0\right], \tag{2.3}$$

where f_0 is a predefined base predictor. This loss function represents the conditional log-loss difference between predictor $\hat{f}$ and a base conditional distribution f_0 with respect to g on the target data distribution. Given a fixed g, $\hat{f}$ aims to be as close as to g if possible. If this is unachievable, $\hat{f}$ will be equal to f_0. The motivation for introducing the base distribution f_0 is that the logging data is not always informative in minimizing the loss function on the target data. Therefore, there should be a default choice of prediction when logging data does not cover the target data distribution well enough. To further shape this process, we incorporate the following constrain Σ_S (used in the definition of $\hat{f}(\boldsymbol{x}, a)$) to regulate the adversary g:

$$\Sigma_S \coloneqq \left\{ g \left| \left| \mathbb{E}_{\boldsymbol{x},a\sim P_s(\boldsymbol{x},a),r\sim g}[r(\boldsymbol{x},a)^2] - \frac{1}{|S|}\sum_{i\in S} r_i^2 \right| \le \eta, \right. \right.$$
$$\left. \left| \mathbb{E}_{\boldsymbol{x},a\sim P_s(\boldsymbol{x},a),r\sim g}[r(\boldsymbol{x},a)\boldsymbol{\phi}(\boldsymbol{x},a)] - \frac{1}{|S|}\sum_{i\in S} r_i\boldsymbol{\phi}(\boldsymbol{x},a) \right| \le \eta \right\}, \tag{2.4}$$

where $\boldsymbol{\phi}(\boldsymbol{x},a)$ is a feature vector and η is the slack term. In general, $\boldsymbol{\phi}(\boldsymbol{x},a)$ can be any feature functions. In this work, we set it to be the concatenation of $\boldsymbol{x}$ and a. For simplicity, we use the same slack for the two constraints in Equation 2.4. This constraint indicates that the expectation of a quadratic function in r under the adversary conditional probability distribution g must be close to the empirical expectation (i.e., the average) of that function within the logging data. We choose this quadratic function because it leads to a Gaussian distribution predictor, which is an exponential family distribution with a quadratic form.

Derived Predictive Form. If we also choose the base distribution to be Gaussian as $f_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$, we have the following form for the predictor $\hat{f}$: $\hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x},a) \sim \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{x},a), \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x},a))$ (The full derivation can be found in Appendix A.2):

$$\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x},a) = \left(2\mathcal{W}(\boldsymbol{x},a)\theta_r + \sigma_0^{-2}\right)^{-1},$$
$$\mu_{\boldsymbol{\theta}}(\boldsymbol{x},a) = \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x},a)\left(-2\mathcal{W}(\boldsymbol{x},a)\boldsymbol{\theta}_{\boldsymbol{x}}\boldsymbol{\phi}(\boldsymbol{x},a) + \mu_0\sigma_0^{-2}\right), \tag{2.5}$$

where $\boldsymbol{\theta}_{\boldsymbol{x}}$ and θ_r are components in the full parameter $\boldsymbol{\theta} = [\theta_r, \boldsymbol{\theta}_{\boldsymbol{x}}]$, and $\boldsymbol{\theta}_{\boldsymbol{x}}$ is a vector with the same dimension as $\boldsymbol{\phi}$. $\mathcal{W}(\boldsymbol{x},a)$ is the density ratio, which equals to $\frac{P_s(\boldsymbol{x},a)}{P_t(\boldsymbol{x},a)}$ under general covariate shift, and reduces to $\frac{\beta(a|\boldsymbol{x})}{\pi(a|\boldsymbol{x})}$ in the OPE setting as the context distribution does not shift. The derivation involves using strong duality to switch the min and max problem and applying the Lagrangian multiplier $\boldsymbol{\theta}$ to convert the constrained problem of Equation 2.3 and Equation 2.4 into an unconstrained one.

The Role of the Density Ratio $\mathcal{W}(x,a)$ and the Base Distribution f_0. The density ratio $\mathcal{W}(\boldsymbol{x},a)$ is defined as the ratio of the source density $P_s(\boldsymbol{x},a)$ over the target density $P_t(\boldsymbol{x},a)$. For cases where the magnitude of $P_s(\boldsymbol{x},a)$ is significantly smaller than $P_t(\boldsymbol{x},a)$, the corresponding $(\boldsymbol{x},a)$ occurs infrequently in the logging data but frequently in the target data. Under such cases, the estimator will rely more on the base distribution f_0 rather than the logging data, as $\mathcal{W}(\boldsymbol{x},a)$ becomes closer to zero in Equation 2.5. In other words, the estimator heavily depends on the base distribution when there is a large discrepancy between the logging and target data. On the other hand, if the logging data is well-covered, the estimator becomes more

Algorithm 1: Stochastic Gradient Descent for Robust Regression under General Covariate Shift

Input: Training data points $\{(\boldsymbol{x}_i, a_i, r_i)\}$, logging policy $\beta(a|\boldsymbol{x})$, target policy $\pi(a|\boldsymbol{x})$, source data distribution $P_s(\boldsymbol{x})$, target data distribution $P_t(\boldsymbol{x})$, $\boldsymbol{\theta}$ with initialization, SGD optimizer Opt, learning rate lr, epoch number T.
$\boldsymbol{\theta} \leftarrow$ random initialization, epoch $\leftarrow 0$;
While epoch $< T$
  For each mini-batch
    Compute $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}, a)$ and $\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}, a)$ following Equation 2.5;
    Compute gradients for $\boldsymbol{\theta}$ following Equation 2.6 and Equation 2.7;
    Stochastic Gradient descent on $\boldsymbol{\theta}$ using Opt.step(lr);
Output: Trained model parameters $\boldsymbol{\theta}$.

confident in its predictions, with μ_θ relying heavily on the learned parameters $\boldsymbol{\theta_x}$ and θ_r. Intuitively, the base distribution will affect the result more when the distribution shift is considerably large.

The Role of Gaussian Predictive Form. We assume a Gaussian base distribution and a quadratic feature constraint, which results in a Gaussian conditional target distribution $p(r|\boldsymbol{x}, a)$. While this Gaussian assumption is generally valid for real-world examples and widely applied in ML literature, we can construct synthetic examples to violate such assumption. For instance, we can define the conditional target distribution as an exponential family with cubic terms over the exponential, corresponding to a cubic function feature constraint. In such cases, the standard bias and variance tradeoff analysis applies: using a model with Gaussian assumption in this situation leads to an overly simplified model, resulting in high bias and low variance. However, in practice, when little is known about the underlying data generation process, the robust regression model under Gaussian assumption is preferred due to its nice implementation properties and broad applicability. Further discussions on the Gaussian assumption is provided in Appendix A.2.

The Proposed Algorithm for Parameter Learning

We summarize the proposed robust regression approach for reward prediction in Figure 2.2 (a). After deriving the predictive form, we here present how to learn model parameters $\boldsymbol{\theta}$. With the predictive form of $\hat{f}$ in Equation 2.5, we can plug it back into the Lagrangian form of the constrained optimization problem in Equation 2.3 and Equation 2.4, and obtain the target loss to be equivalent as maximizing the log-likelihood over the target distribution: $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{P_t(\boldsymbol{x},a,r)}\left[\log \hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, a)\right]$. Given

the form of $\hat{f}$, this objective is convex w.r.t $\boldsymbol{\theta}$. By taking the derivative of $\mathcal{L}(\boldsymbol{\theta})$ w.r.t the θ_r and $\boldsymbol{\theta}_{\boldsymbol{x}}$, we can obtain the following gradient form, where S_{batch} is the set for data mini-batch. Standard gradient-based learning can then be applied for learning the parameters. The training procedure is summarized in Algorithm 1. Detailed derivation are in Appendix A.2.

$$\nabla_{\theta_r}\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|S_{\text{batch}}|}\sum_{(\boldsymbol{x},a,r)\in S_{\text{batch}}}(\mu^2_{\boldsymbol{\theta}}(\boldsymbol{x},a)+\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x},a)-r^2), \tag{2.6}$$

$$\nabla_{\boldsymbol{\theta}_{\boldsymbol{x}}}\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|S_{\text{batch}}|}\sum_{(\boldsymbol{x},a,r)\in S_{\text{batch}}}(\mu_{\boldsymbol{\theta}}(\boldsymbol{x},a)-r)\boldsymbol{\phi}(\boldsymbol{x},a). \tag{2.7}$$

2.4 Methods

In this section, we first provide an overview of how our robust regression approach can be integrated into existing policy evaluation methods as a reward model. A detailed comparison between classical OPE and the setting of policy evaluation under general covariate shift considered in this paper is illustrated in Figure 2.2 (b). We then establish upper bounds for the bias of these methods. We will refer to our methods using the notation **PS** and **GCS** as suffixes, representing policy shift and general covariate shift, respectively. Specifically, the proposed methods **DM-PS**, **DR-PS**, **DM-GCS**, **DR-GCS**, **SnDR-PS** and **SnDR-GCS** are formulated as follows. We also investigate the integration of our robust regression approach into other OPE methods including SWITCH [50] and DRoS [40] in Appendix A.4.

DM-PS. In the OPE setting, the context distribution $P(\boldsymbol{x})$ does not shift, reducing the density ratio $\mathcal{W}(\boldsymbol{x},a)$ to $\frac{\beta(a|\boldsymbol{x})}{\pi(a|\boldsymbol{x})}$. We enhance standard DMs by incorporating the mean value predictor estimated from robust regression, denoted as $\mu_{\boldsymbol{\theta}}(\boldsymbol{x},a)$, yielding the "Direct Method with Policy Shift" (DM-PS) estimator.

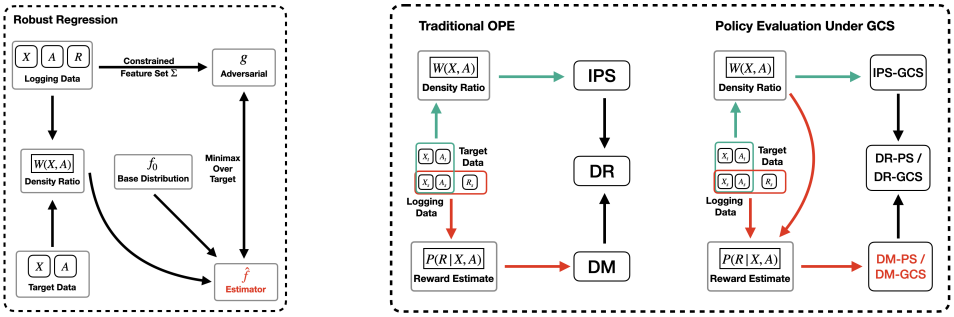
$$\hat{V}_{\text{DM-PS}} = \tfrac{1}{|S|}\textstyle\sum_{\boldsymbol{x}\in S}\mathbb{E}_{a\sim\pi}\left[\mu^{\text{PS}}_{\boldsymbol{\theta}}(\boldsymbol{x},a)\right]. \tag{2.8}$$

DR-PS. We apply DM-PS to be the reward model part in DR framework, yielding the "Doubly Robust estimator with Policy Shift" (DR-PS) estimator.

$$\hat{V}_{\text{DR-PS}} = \tfrac{1}{|S|}\textstyle\sum_{(\boldsymbol{x},a,r)\in S}\left[\frac{(r-\mu^{\text{PS}}_{\boldsymbol{\theta}}(\boldsymbol{x},a))\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})}\right]+\hat{V}_{\text{DM-PS}}. \tag{2.9}$$

DM-GCS. Under general covariate shift, we set $\mathcal{W}(\boldsymbol{x},a)$ to be $\frac{P_s(\boldsymbol{x},a)}{P_t(\boldsymbol{x},a)}$ in robust regression and plug it into the standard DM to get "Direct Method with General Covariate Shift" (DM-GCS) estimator. We also plug $\mathcal{W}(\boldsymbol{x},a)=\frac{P_s(\boldsymbol{x},a)}{P_t(\boldsymbol{x},a)}$ into IPS to implement IPS-GCS as a baseline. More details can be found in Appendix A.1.

$$\hat{V}_{\text{DM-GCS}} = \tfrac{1}{|S|}\textstyle\sum_{\boldsymbol{x}\in S}\mathbb{E}_{a\sim\pi}\left[\mu^{\text{GCS}}_{\boldsymbol{\theta}}(\boldsymbol{x},a)\right], \tag{2.10}$$



(a) The Robust Regression Method (b) Comparison between OPE and the proposed approach

Figure 2.2: **(a):** Description of the robust regression method. The density ratio $\mathcal{W}(\boldsymbol{x}, a)$ is estimated from the logging data and the target data. The feature set Σ is obtained from logging data to constrain the adversarial player g. The estimator $\hat{f}$ is estimated by taking minimax loss over target data distribution, with density ratio $\mathcal{W}(\boldsymbol{x}, a)$, base distribution f_0, and adversarial player g. **(b):** Comparision between the setting of classical off-policy evaluation (OPE) and policy evaluation under general covariate shift. There are two types of information derived from the data: (1) density ratio information, obtained from the covariate (context and action) distribution from both logging and target data and used by the inverse propensity score (IPS) method (green arrow), and (2) pairing information between covariate and reward from the logging data, used by the direct method (DM) (red arrow) to estimate the reward conditional distribution. Traditional DM relies solely on the logging data, whereas the proposed DM-PS and DM-GCS methods also utilize the density ratio information in the robust regression.

DR-GCS. We apply DM-GCS to be the reward model part in DR and apply IPS-GCS to the IPS part of DR to obtain the "Doubly Robust estimator with General Covariate Shift" (DR-GCS) estimator.

$$\hat{V}_{\text{DR-GCS}} = \frac{1}{|S|} \sum_{(\boldsymbol{x},a,r)\in S} \left[\frac{(r-\mu_{\boldsymbol{\theta}}^{\text{GCS}}(\boldsymbol{x},a))P_t(\boldsymbol{x},a)}{P_s(\boldsymbol{x},a)} \right] + \hat{V}_{\text{DM-GCS}}. \tag{2.11}$$

SnDR-PS/SnDR-GCS. Similar to standard DR, following the idea of SnIPS, we can further control the variance of the IPS component and obtain the self-normalized DR-PS (SnDR-PS) and self-normalized DR-GCS (SnDR-GCS), by normalizing over the sum of the importance weights.

$$\hat{V}_{\text{SnDR-PS}} = \frac{1}{|S|} \sum_{(\boldsymbol{x},a,r)\in S} \left[\frac{(r - \mu_{\boldsymbol{\theta}}^{\text{PS}}(\boldsymbol{x}, a))\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x}) \sum_{(\boldsymbol{x},a,r)\in S} \frac{\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})}} \right] + \hat{V}_{\text{DM-PS}}, \tag{2.12}$$

$$\hat{V}_{\text{SnDR-GCS}} = \frac{1}{|S|} \sum_{(\boldsymbol{x},a,r)\in S} \left[\frac{(r - \mu_{\boldsymbol{\theta}}^{\text{GCS}}(\boldsymbol{x}, a))P_t(\boldsymbol{x}, a)}{P_s(\boldsymbol{x}, a) \sum_{(\boldsymbol{x},a,r)\in S} \frac{P_t(\boldsymbol{x},a)}{P_s(\boldsymbol{x},a)}} \right] + \hat{V}_{\text{DM-GCS}}. \tag{2.13}$$

Theoretical Results

Bias Analysis

Expected Error. We first establish the upper bound for the robust regression-based reward estimation's expected squared error on the target distribution as follows:

Theorem 2.1. *Assuming the density ratio of the target over the source distribution* $\frac{P_t(\boldsymbol{x},a)}{P_s(\boldsymbol{x},a)}$ *is upper bounded by a constant C for* $(\boldsymbol{x}, a)$ *in the source distribution, the base distribution is chosen in the way that* $\mathbb{E}_{P_t(\boldsymbol{x},a,r)}[(r-\mu_0)^2]$ *over the target distribution is upper bounded by a constant H, and source distribution has n i.i.d. data samples. Assuming there are m percentage of samples in the target distribution that have a non-zero density ratio* $\mathcal{W}(\boldsymbol{x}, a) = \frac{P_s(\boldsymbol{x},a)}{P_t(\boldsymbol{x},a)}$*, the expected squared error in the target data distribution for the robust regression estimator is upper bounded as follows with probability at least* $1-\delta$*:*

$$\begin{aligned}&\mathbb{E}_{P_t(\boldsymbol{x},a,r)}\left[(r-\hat{\mu}(\boldsymbol{x},a))^2\right]\\&\qquad\leq mC\left[\frac{1}{n}\sum_{i=1}^{n}\sigma^2(\boldsymbol{x}_i,a_i)+\eta_1+4M\hat{\mathfrak{R}}(\mathcal{F})+3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right]+(1-m)H,\end{aligned}\tag{2.14}$$

where $\mathcal{F}$ *is the function class of* f *with* $\sup_{f,f'\in\mathcal{F},\boldsymbol{x},a}|f(\boldsymbol{x},a)-f'(\boldsymbol{x},a)|\leq M$ *and a Rademacher complexity of* $\hat{\mathfrak{R}}$*, and* η_1 *upper bounds the gradient of optimization algorithm when it converges, as shown in Equation A.18.*

Bias Analysis for DM Methods. Based on the analysis of the expected error as above, we obtain the following upper bound for the bias of the proposed direct method estimators, DM-PS/DM-GCS.

Corollary 2.2. *Under the same assumptions as Theorem 2.1, the bias of the robust regression direct method estimators over the target data distribution are upper bounded as follows with probability at least* $1-\delta$*:*

$$\begin{aligned}&\mathbb{E}_{P_t(\boldsymbol{x},a,r)}\left[(r-\hat{\mu}(\boldsymbol{x},a))\right]\\&\qquad\leq\left[mC\left[\frac{1}{n}\sum_{i=1}^{n}\sigma^2(\boldsymbol{x}_i,a_i)+\eta_1+4M\hat{\mathfrak{R}}(\mathcal{F})+3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right]+(1-m)H\right]^{1/2}:=\epsilon.\end{aligned}\tag{2.15}$$

Bias Analysis for DR Methods. We further analyze the bias of the proposed doubly robust estimators, DR-PS/DR-GCS. We have the following upper bound.

Theorem 2.3. *Under the same assumptions as Theorem 2.1, the bias of the robust regression doubly robust estimators over the target distribution are upper bounded as follows with probability at least* $1-\delta$*:*

$$\left|\mathbb{E}_{P_t(\boldsymbol{x},a,r)}\left[\hat{V}_{\mathrm{DR-PS/DR-GCS}}-V^T\right]\right| \leq \frac{C\eta_1}{l} + 2CM\hat{\mathfrak{R}}(\mathcal{F}) + 3CM\sqrt{\frac{\log\frac{2}{\delta}}{2n}} + \epsilon. \quad (2.16)$$

The proofs of Theorem 2.1 and 2.3 can be found in Appendix A.3.

Robustness under Large Shift. The above bias analysis shows that the proposed methods remain robust under large distribution shifts. Our bias analysis results consist of two parts, the first term is based on data with a non-zero density ratio where we reweight the target distribution into source distribution and use standard Rademacher complexity. The second term refers to the bounded error of the base distribution for target data that do not share support with the source data. In this case, the m value, which is the percentage of target data points well-supported by the source distribution, represents the magnitude of the distribution shift. When the shift becomes larger, more proportion of the data will have a density ratio close to 0, and m becomes smaller, corresponding to cases where the prediction relies more heavily on the base predictor f_0. Note that traditional ERM methods can have an arbitrarily bad performance when there is a severe non-overlapping issue between the source and target data distribution, such as large part of the target distribution is uncovered by the source distribution. Reweighting-based methods also become ineffective as the importance weights will have a zero denominator in a subset of data points. Our methods can effectively alleviate the issue as the bias of the proposed methods, which is shown in Equation 2.15 and Equation 2.16, will lean towards utilizing the base predictor f_0 and have a bounded error as m becomes closer to 0. This makes our method more favorable than other methods under large shift settings, as further illustrated in the experiments section. In particular, we demonstrate how our method performs compared with other methods under different scales of shifting in Section 2.5. The experimental results show that our method remains robust and outperforms other methods, especially when the shift becomes large, as shown in Figure 2.4. Besides, we also provide a consistency analysis result for the settings where the density ratio is bounded away from 0 in Appendix A.3.

2.5 Experiments

Experiment Settings

Datasets. In line with the experimental settings employed in previous studies [11, 12, 38, 39, 50], we conduct experiments[1] on 9 UCI datasets by transforming the classification problems to the contextual bandits setting. Specifically, the context is set to be the input feature of the dataset, while the policy is a classification model and the action is the predicted class from the model. The reward is 1 if the predicted class matches the ground truth label and 0 otherwise. More details about how the dataset is created can be found in Appendix A.4, and the statistics of these datasets are summarized in Table 2.1 as follows.

Table 2.1: Dataset Statistics.

Dataset	Glass	Ecoli	Vehicle	Yeast	PageBlok	OptDigits	SatImage	PenDigits	Letter
Actions	6	8	4	10	5	10	6	10	26
Features	9	7	18	103	10	64	36	16	16
Instances	214	336	846	1484	5473	5620	6435	10992	20000

Logging Policies. We conducted experiments using three distinct categories of logging policies, encompassing 10 different policies. Specifically, we employed the softened policy, which is in line with previous off-policy evaluation (OPE) work [12, 39], as well as Tweak-1 (ρ) policy and Dirichlet (γ) policy, which were inspired by prior research on label shift [25] and offline contextual bandit optimization [51]. **1)** Softened policy: Softened policy is defined as $\pi_{(\lambda,\zeta)}(a|\boldsymbol{x}) = \lambda + \zeta u$ if $a = \hat{\psi}(x)$, where $u \sim \text{Uniform}(-0.5, 0.5)$ and $\hat{\psi}$ is a deterministic policy trained by logistic regression. The remaining probability mass, $1 - (\lambda + \zeta u)$, is evenly distributed among classes $a \neq \hat{\psi}(x)$. **2)** Tweak-1 (ρ): One class accounts for ρ probability, and the other classes evenly share the remaining $1 - \rho$. **3)** Dirichlet (γ): A fixed probability distribution generated by Dirichlet distribution parameterized by γ. Compared with the softened policy, the Tweak-1 and Dirichlet policies create more probability values close to 0 or 1 among the arms, resulting in a larger distribution shift between the logging policy and the target policy.

Target Policies. We evaluate our method across various combinations of logging policies and three distinct types of target policies. **1)** Softened target policy: We use the softened policy with parameters $(\lambda, \zeta) = (0.9, 0)$. **2)** Softened perfect policy: We further apply the softened policy techniques to modify a perfect policy, which consists of a 100% accuracy classification model. **3)** Softened diverse perfect policy:

[1]The code for the experiments is available at https://github.com/guoyihonggyh/Distributionally-Robust-Policy-Evaluation-under-General-Covariate-Shift-in-Contextual-Bandits.

We set the target policy value for each class as a random permutation of the sequence $\left[\frac{1}{n}, \frac{2}{n}, \ldots, 1\right]$, where n represents the number of classes. This can be viewed as a discrete version of uniform sampling from the interval [0,1].

Covariate Shift on the Context. In our study, we examine two different types of context shifts within the data. We uniformly sample context for the target data, while manipulating the distribution of the context in the logging data in the following two ways to create covariate shift. **1)** Gaussian covariate shift: We first apply Principal Component Analysis (PCA) to reduce the dimension. We then define a Gaussian distribution on the first principal component as the new data distribution. **2)** Tweak-1 covariate shift (ω): Following the logic of the Tweak-1 (ρ) logging policy, we assign a specific class of data a higher weight ω and all other classes with weight 1, and then do weighted sampling. Detailed information about the policies and data generation process can be found in Appendix A.4.

Estimation of the Density Ratio. We consider both the density ratio $\mathcal{W}$ known and unknown conditions in our experiments. For $\mathcal{W}$ unknown cases, we need to estimate them. Specifically, in policy shift settings, we estimate the logging policy β with a logistic regression model, where we use the context $\boldsymbol{x}$ in the logging data as features and the corresponding action a as labels. For general covariate shift case, we also need to estimate $\frac{P_s(\boldsymbol{x})}{P_t(\boldsymbol{x})}$. We leverage a discriminative classifier to obtain $\frac{P(\text{source}|\boldsymbol{x})}{P(\text{target}|\boldsymbol{x})}$ using both the logging and the target context. For instance, we train a logistic regression model to classify whether the context $\boldsymbol{x}$ comes from the logging data or the target data. Then, according to Bayes' theorem, we can obtain the $\frac{P_s(\boldsymbol{x})}{P_t(\boldsymbol{x})}$ by $\frac{P(\text{source}|\boldsymbol{x})}{P(\text{target}|\boldsymbol{x})} = \frac{P_s(\boldsymbol{x})P(\text{source})}{P(\boldsymbol{x})} / \frac{P_t(\boldsymbol{x})P(\text{target})}{P(\boldsymbol{x})} = \frac{P_s(\boldsymbol{x})}{P_t(\boldsymbol{x})}$ (when the source and target data has the same size, i.e. $P(\text{source}) = P(\text{target})$) [4].

Baselines. For our experimental comparisons, we consider several baseline methods, including Direct Method (DM), Inverse Propensity Score (IPS), Doubly Robust (DR) [2, 10, 32], SnIPS, Self-Normalized Doubly Robust (SnDR) [43], More Robust Doubly Robust Estimator (MRDR) [12], and Distributionally Robust Policy Evaluation (DRPE) [36]. We include a variant of our proposed methods DM (R), shown in Equation A.8, as a baseline by setting the density ratio in the robust regression to 1. More details about the baselines, including SnIPS, SnDR, IPS-GCS, and DM(R), can be found in Appendix A.1. Additionally, we provide the further comparison with other OPE methods in Appendix A.4, including SWITCH [50] and Doubly robust estimator with Shrinkage (DRoS) [40], demonstrating the advantage of our method when integrated with various kinds of OPE approaches.

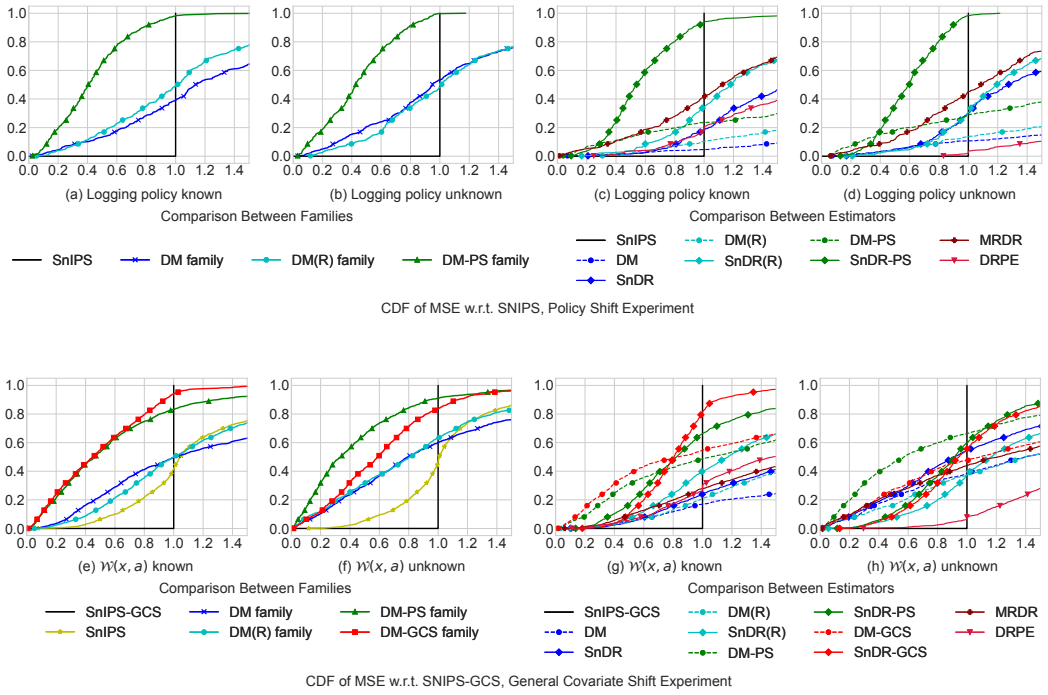


Figure 2.3: First row: Cumulative Distribution Function (CDF) of relative MSE w.r.t SnIPS when only policy shifts exist across 360 conditions. Second row: CDF of relative MSE w.r.t SnIPS-GCS when general covariate shifts exist across 1260 conditions. Subfigures (a)/(e) and (b)/(f) are comparisons between different families of methods, while (c)/(g) and (d)/(h) are comparisons between individual estimators. Better performance is indicated by CDF curves positioned in the top-left corner, signifying a relatively lower MSE. In the first row, among the evaluated methods, the DM-PS family is the best performer in known and unknown logging policy cases. In the second row, in the context of known $\mathcal{W}(x,a)$, the DM-GCS family performs best, while in scenarios with unknown $\mathcal{W}(x,a)$, the DM-PS family prevails.

To better compare our methods with baselines, we consider the corresponding families. Each DM estimator family comprises DM itself, DR, and SnDR employing the corresponding DM. Each family contains the same number of estimators for fair comparison. For instance, the DM-PS family includes DM-PS, DR-PS, and SnDR-PS. We select the best-performing method from each family for comparison in our experiments.

Evaluation Metric. We measure the performance of each estimator using Mean Squared Error (MSE), which is defined as $\mathrm{E}[(\hat{V} - V(\pi))^2]$, where $\hat{V}$ is the value calculated from the estimator and $V(\pi)$ is the ground truth value of the target policy on the target data distribution, both computed on a test set which consists of 25% of the dataset.

Experiment Results

We present the results for various combinations of logging policies, target policies, and covariate shifts on data, encompassing 360 conditions for the policy shift-only experiments and 1260 conditions for the general covariate shift experiments. Following the established setting in previous studies [39, 50], we show the Cumulative Distribution Function (CDF) of relative MSE with respect to SnIPS for the policy shift and CDF of relative MSE with respect to SnIPS-GCS for the general covariate shift as the evaluation metric in Figure 2.3. Our analysis covers both cases where the logging policy is known (Subfigures (a), (c), (e), and (g)) and unknown (Subfigures (b), (d), (f) and (h)).

General Performance Comparison: From Figure 2.3 (a) and (b), which compare the performance across different method families under policy shift, we can see that the DM-PS family consistently outperforms other methods substantially. In Table A.1 in Appendix A.4, we further present the number of conditions where our proposed method families perform optimally. The DM-PS family emerges as the top performer in both known and unknown logging policies, outperforming other baseline methods in over 90% of the conditions. For general covariate shift settings, when the density ratio $\mathcal{W}(\boldsymbol{x}, a)$ is known (Figure 2.3 (e)), the DM-GCS family exhibits the best performance. When $\mathcal{W}(\boldsymbol{x}, a)$ is unknown (Figure 2.3 (f)), the DM-PS family leads in performance. In both scenarios (Figure 2.3 (e) and (f)), both DM-GCS and DM-PS families outperform SnIPS-GCS in over 70% of the examined conditions. In (c), (d), (g), and (h), DM-PS and DM-GCS significantly outperform MRDR and DRPE. Further comparisons with advanced OPE methods such as SWITCH and DRoS are provided in Appendix A.4.

Comparison between DM-PS and DM-GCS: When comparing DM-PS and DM-GCS in Figure 2.3 (g) and (h) as well as in Table 2.2, we observe that DM-PS and its family perform better when the density ratio $\mathcal{W}(\boldsymbol{x}, a)$ is unknown, while DM-GCS and its family excels when $\mathcal{W}(\boldsymbol{x}, a)$ is known. These results can be attributed to that the GCS setting introduces the disparity between $P_s(\boldsymbol{x})$ and $P_t(\boldsymbol{x})$, creates larger distribution shifts and harder-to-estimate density ratios, and destabilizes the training of DM-GCS and the IPS component within DR methods. We further demonstrate this by comparing SnIPS and SnIPS-GCS in Table A.2 in Appendix A.4. Also, the relatively weaker performance of DM-GCS when $\mathcal{W}(\boldsymbol{x}, a)$ is unknown can be attributed to the estimation of $P_s(\boldsymbol{x}, a)$, which requires a concurrent estimation of both $P_s(\boldsymbol{x})$ and $\beta(a|\boldsymbol{x})$. The inherent difficulty in accurately estimating $P_s(\boldsymbol{x})$, which

Table 2.2: The number of conditions where each family is the best under general covariate shift. DM-PS and DM-GCS families outperform baseline methods in over 72% of the conditions. When comparing the settings of $P_s(\boldsymbol{x}, a)$ known with $P_s(\boldsymbol{x}, a)$ unknown, the DM-GCS family outperforms DM-GCS family among known settings while the DM-PS family excels among unknown settings.

	SnIPS-GCS	DM	DM(R)	DM-PS	DM-GCS
$P_s(\boldsymbol{x}, a)$ known	40	148	93	372	607
$P_s(\boldsymbol{x}, a)$ unknown	60	265	84	690	161

Table 2.3: The number of conditions where DM-PS or SnDR-PS is the best under different Tweak-1 logging policies. From left to right, the shift becomes larger. This suggests that SnDR-PS's advantage can be weakened by the worsened performance of the IPS component and disappear, as SnDR-PS consists of the DM-PS and the IPS component.

	Tweak-1 ($\rho = 0.91$)	Tweak-1 ($\rho = 0.95$)	Tweak-1 ($\rho = 0.99$)
DM-PS	6	8	18
SnDR-PS	30	28	18

is typically more complex than estimating the logging policy, negatively affects the performance of the DM-GCS family.

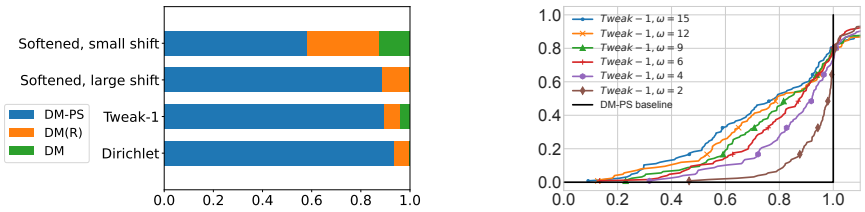
Comparison between DM Methods and DR Methods: In Figure 2.3 (c) and (d), by comparing the DM-PS with DM(R) and DM, we see that our method shows the best performance, highlighting the advantages of accommodating covariate shifts in reward estimation. We also observe that the SnDR-PS has the best performance. However, in the left corner, we see instances where the DM-PS outperforms the SnDR-PS, indicating that although SnDR-PS generally outperforms the DM-PS, as the shift becomes larger, its advantage is weakened by the worsened performance of IPS component. This is further demonstrated in Table 2.3, where we show the number of conditions where DM-PS and SnDR-PS are the best under different Tweak-1 logging policies. A larger ρ represents a larger shift. One can see that when the shift becomes larger, the SnDR-PS's advantage becomes smaller. Furthermore, in the general covariate shift setting, we can still observe that DM-PS outperforms SnDR-PS in many conditions shown in Figure 2.3 (c) and (d), a pattern that is also seen between DM-GCS and SnDR-GCS. This can be explained by the negative effect on performance when incorporating the unreliable IPS into the SnDR method when the distribution shift is large. Additionally, when the distribution shift of $P_s(\boldsymbol{x})$ occurs, obtaining a reliable estimation of $\beta(a|\boldsymbol{x})$ becomes increasingly challenging, consequently affecting the estimation of the density ratio $\mathcal{W}(\boldsymbol{x}, a)$, and harming the performance of SnDR methods.

The Effect of Different Shift Magnitudes: We analyze the performance of DM-PS and DM-GCS under various scales of policy and covariate shifts to show how shifts affect the performance of the DM methods.

1. Policy Shift. Consider two types of softened shift logging policies: small shifts, such as $\pi_{(0.95,0)}(a|\boldsymbol{x})$ and $\pi_{(0.7,0.1)}(a|\boldsymbol{x})$, where logging policy deviations from the target policy are modest; and large shifts, represented by $\pi_{(0.5,0.1)}(a|\boldsymbol{x})$ and $\pi_{(0.1,0)}(a|\boldsymbol{x})$, that exhibit more substantial deviations. The Tweak-1 and Dirichlet policies are treated as even larger shift logging policies due to more probability values close to 0 or 1. Figure 2.4 (a) illustrates the proportion of each DM method achieving the best performance across these conditions, revealing that DM-PS consistently surpasses other methods across diverse logging policy scenarios. DM-PS's advantage grows as the shift enlarges, demonstrating its excellence in accommodating policy shifts in reward estimation and aligning well with our theoretical results in Section 2.4.

2. General Covariate Shift. Figure 2.4 (b) showcases the CDF of the relative MSE of DM-GCS with respect to DM-PS in Gaussian covariate shift scenarios, with a larger ω indicating a larger covariate shift. As the covariate shift increases, DM-GCS exhibits widening advantages over DM-PS, indicating the benefits of accommodating covariate shifts in reward estimation.

Collectively, these analyses highlight the superior performance of DM-PS and DM-GCS in handling varying scales of covariate shifts in reward estimation. They consistently exceed other methods, particularly as shifts become more pronounced.



(a) Policy shift (b) General covariate shift

Figure 2.4: **(a)** The proportion of each DM method achieving the best performance across conditions. As the covariate shift increases, the advantages of DM-PS become more pronounced, signifying the importance of considering covariate shifts in reward estimation. **(b)** The CDF of the relative MSE of DM-GCS with respect to DM-PS in Gaussian covariate shift cases. As the Gaussian covariate shifts on context increase, DM-GCS demonstrates more advantages.

2.6 Related Work

Off-Policy Evaluation for Contextual Bandits. Classic approaches for OPE (which only handle the policy shift setting) can be summarized into three categories: direct methods (DM), inverse propensity scoring (IPS), and doubly robust (DR) methods that combine DM and IPS. DMs aim to directly learn the mapping from context and action to reward, but they can suffer from significant bias [10] due to the divergence between the target and logging policies. IPS tackles the problem using the idea of important weighting between policy distribution. While unbiased, they exhibit high variance, especially under large policy shifts, due to unstable estimations of the density ratio [10]. A more recently proposed approach is the self-normalized IPS estimator (SnIPS) [42], which offers improved accuracy over standard IPS when fluctuations in weights dominate that of the rewards.

DR methods [10–12, 50] strike a balance between the biased, low-variance DM and the unbiased, high-variance IPS by incorporating both. It can also be interpreted as using control variates within an IPS framework [47, 49]. If either IPS or DM is unbiased, DR methods are guaranteed to be unbiased [10]. Much of the follow-up work on DR methods has focused on mitigating the detrimental effects of variance or extreme probabilities from the IPS component. For example, SnDR utilizes SnIPS [43], SWITCH estimator truncates IPS using a threshold [50], and DR with Optimistic Shrinkage (DRoS) finds a weighting strategy by optimizing a sharp bound on mean squared error [39]. While much of the prior work has emphasized improving the IPS component of DR estimators and their asymptotical guarantees, our research targets enhancing the robustness of conditional reward estimators, especially under large shifts and limited data. Our key insight is to view this problem as a covariate shift problem and utilize robust regression to train the conditional reward estimator. Our proposed method can also be naturally incorporated into existing methods like DMs and DRs by offering a distributionally robust reward estimator.

Covariate Shift and Distributional Robustness. Covariate shift is a specific type of distribution shift, and has been extensively studied in the machine learning community [35]. Various methods have been developed to mitigate this challenge, including importance weighting [35, 41] and kernel mean matching [14]. In the context of policy evaluation, [48] introduced a DR-based estimator under covariate shift that uses a density ratio estimator between the source and target distributions. However, their method focuses on the propensity score weighting formulation, and the IPS component in the estimator introduces high variance when extreme weights

exist. In contrast, we tackle the problem orthogonally in this paper, utilizing a distributional robust direct method that can be effortlessly combined with their method.

Distributionally robust optimization (DRO) [3, 37], a prevalent approach to handling distribution shifts, introduces the idea of optimizing with respect to the most adverse scenarios over a defined family of distributions. This has led to different variants such as Wasserstein DRO [5]. A few studies have explored distributional robustness in policy evaluation. [36] and [18] augmented evaluation robustness by conservatively estimating reward mapping with KL-divergence uncertainty sets. However, in these methods, the uncertainty set covers all possible shifts in the joint distribution of context, action, and reward, which does not characterize the covariate shift specifically. As a result, the estimator can be overly robust and less effective. Another line of work utilize DRO for estimating confidence interval in the off-policy setting [8, 13, 20], following a generalized empirical likelihood approach [9]. The differences from our approach lie in the formulation of the DRO problem. Specifically, they do not use the conditional distribution of the target variable (the reward in our case) as the adversarial player, and the uncertainty set constraint is formulated by the perturbation of empirical distribution of the data with distance measured using f-divergence, rather than using feature matching as in our case. Robust regression [7, 26] tailored distributional robustness methods specifically for the setting of covariate shift. It constructs a predictor that approximately matches training data statistics but is otherwise the most uncertain on the testing distribution by minimizing the worst-case expected target loss and obtaining a parametric form of the predicted output labels' probability distributions. We are the first to employ robust regression methods for policy evaluation, enabling the refinement of direct methods and the further enhancement of doubly robust estimators.

Causal Inference. The task explored in this paper has close ties to causal inference [1, 16]. A core challenge in evaluating the individual treatment effect (ITE) and the average treatment effect (ATE) in causal inference is the evaluation of a counterfactual policy. IPS and DR have been investigated for ATE estimation using linear models [19, 44] and recent advancements have leveraged methods from domain adaptation and deep representation learning [17]. In particular, ITE estimation is challenged by the specific type of distribution shift where the context distribution changes between treatment and control groups [34]. There has also been work on using causal models to enhance off-policy evaluation results [28, 29]. Besides, the setting of covariate

shift has been studied in the task of randomized controlled trials (RCT) in causal inference as an issue of lacking external validity [30]. However, these works mainly focus on devising new identification strategies of causal estimands. In this work, we focus on developing robust learning methods for counterfactual reasoning from the perspective of covariate shift.

2.7 Conclusion

We introduced a distributionally robust approach for policy evaluation under general covariate shifts in contextual bandits using robust regression. Utilizing this approach, we developed a novel family of policy evaluation methods for not only the policy shift setting but also the general covariate shift setting. We showed how the reward model from such distributionally robust reward estimators can be integrated into direct methods and doubly robust policy estimators. Theoretically, we established the finite sample upper bounds on the bias of the proposed methods, providing advantages in large shift situations. Empirically, we conducted extensive studies across a wide range of varying distribution shift conditions, demonstrating the superior robustness of our method. Future work includes the investigation of applying the distributional robustness idea to the OPE in reinforcement learning, offline, and safe reinforcement learning.

References

[1] Susan Athey. "Machine learning and causal inference for policy evaluation". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2015, pp. 5–6.

[2] Heejung Bang and James M Robins. "Doubly robust estimation in missing data and causal inference models". In: *Biometrics* 61.4 (2005), pp. 962–973.

[3] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Vol. 28. Princeton university press, 2009.

[4] Steffen Bickel, Michael Brückner, and Tobias Scheffer. "Discriminative learning for differing training and test distributions". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 81–88.

[5] Jose Blanchet, Karthyek Murthy, and Viet Anh Nguyen. "Statistical analysis of Wasserstein distributionally robust estimators". In: *Tutorials in Operations Research: Emerging Optimization Methods and Modeling Techniques with Applications*. INFORMS, 2021, pp. 227–254.

[6] Léon Bottou et al. “Counterfactual reasoning and learning systems: The example of computational advertising”. In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 3207–3260.

[7] Xiangli Chen et al. “Robust covariate shift regression”. In: *Artificial Intelligence and Statistics*. 2016.

[8] Bo Dai et al. “Coindice: Off-policy confidence interval estimation”. In: *Advances in neural information processing systems* 33 (2020), pp. 9398–9411.

[9] John C Duchi, Peter W Glynn, and Hongseok Namkoong. “Statistics of robust optimization: A generalized empirical likelihood approach”. In: *Mathematics of Operations Research* 46.3 (2021), pp. 946–969.

[10] Miroslav Dudik, John Langford, and Lihong Li. “Doubly robust policy evaluation and learning”. In: *International Conference on Machine Learning (ICML)*. 2011.

[11] Miroslav Dudik et al. “Doubly robust policy evaluation and optimization”. In: *Statistical Science* 29.4 (2014), pp. 485–511.

[12] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. “More robust doubly robust off-policy evaluation”. In: *International Conference on Machine Learning (ICML)*. 2018.

[13] Louis Faury et al. “Distributionally robust counterfactual risk minimization”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 3850–3857.

[14] Arthur Gretton et al. “Covariate shift by kernel mean matching”. In: *Dataset shift in machine learning* 3.4 (2009), p. 5.

[15] Daniel G Horvitz and Donovan J Thompson. “A generalization of sampling without replacement from a finite universe”. In: *Journal of the American statistical Association* 47.260 (1952), pp. 663–685.

[16] Thorsten Joachims et al. “Recommendations as treatments”. In: *AI Magazine* 42.3 (2021), pp. 19–30.

[17] Fredrik Johansson, Uri Shalit, and David Sontag. “Learning representations for counterfactual inference”. In: *International conference on machine learning (ICML)*. 2016.

[18] Nathan Kallus et al. “Doubly robust distributionally robust off-policy evaluation and learning”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 10598–10632.

[19] Joseph DY Kang, Joseph L Schafer, et al. “Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data”. In: *Statistical science* 22.4 (2007), pp. 523–539.

[20] Nikos Karampatziakis, John Langford, and Paul Mineiro. "Empirical likelihood for contextual bandits". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9597–9607.

[21] Andreas Krause and Cheng S Ong. "Contextual gaussian process bandit optimization". In: *Neural Information Processing Systems (NeurIPS)*. 2011.

[22] John Langford and Tong Zhang. "The epoch-greedy algorithm for contextual multi-armed bandits". In: *Neural Information Processing Systems (NeurIPS)*. 2007.

[23] Huitian Lei, Ambuj Tewari, and Susan Murphy. "An actor-critic contextual bandit algorithm for personalized interventions using mobile devices". In: *Neural Information Processing Systems (NeurIPS)* (2014).

[24] Lihong Li et al. "A contextual-bandit approach to personalized news article recommendation". In: *The Web Conference (WWW)*. 2010.

[25] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. "Detecting and correcting for label shift with black box predictors". In: *International conference on machine learning*. PMLR. 2018, pp. 3122–3130.

[26] Anqi Liu and Brian D Ziebart. "Robust covariate shift prediction with general losses and feature views". In: *arXiv preprint arXiv:1712.10043* (2017).

[27] Anqi Liu et al. "Robust Regression for Safe Exploration in Control". In: *arXiv preprint arXiv:1906.05819* (2019).

[28] Hongseok Namkoong et al. "Off-policy policy evaluation for sequential decisions under unobserved confounding". In: *arXiv preprint arXiv:2003.05623* (2020).

[29] Michael Oberst and David Sontag. "Counterfactual Off-Policy Evaluation with Gumbel-Max Structural Causal Models". In: *arXiv preprint arXiv:1905.05824* (2019).

[30] Judea Pearl and Elias Bareinboim. "External validity: From do-calculus to transportability across populations". In: *Probabilistic and causal inference: The works of Judea Pearl*. ACM, 2022, pp. 451–482.

[31] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[32] James M Robins and Andrea Rotnitzky. "Semiparametric efficiency in multivariate regression models with missing data". In: *Journal of the American Statistical Association* 90.429 (1995), pp. 122–129.

[33] Bernhard Schölkopf et al. "On causal and anticausal learning". In: *arXiv preprint arXiv:1206.6471* (2012).

[34] Uri Shalit, Fredrik D Johansson, and David Sontag. “Estimating individual treatment effect: generalization bounds and algorithms”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3076–3085.

[35] Hidetoshi Shimodaira. “Improving predictive inference under covariate shift by weighting the log-likelihood function”. In: *Journal of statistical planning and inference* 90.2 (2000), pp. 227–244.

[36] Nian Si et al. “Distributionally robust policy evaluation and learning in offline contextual bandits”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8884–8894.

[37] Agnieszka Słowik and Léon Bottou. “On distributionally robust optimization and data rebalancing”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 1283–1297.

[38] Yi Su et al. “Cab: Continuous adaptive blending for policy evaluation and learning”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6005–6014.

[39] Yi Su et al. “Doubly robust off-policy evaluation with shrinkage”. In: *arXiv preprint arXiv:1907.09623* (2019).

[40] Yi Su et al. “Doubly robust off-policy evaluation with shrinkage”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9167–9176.

[41] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert MÃžller. “Covariate shift adaptation by importance weighted cross validation”. In: *Journal of Machine Learning Research* 8.May (2007), pp. 985–1005.

[42] Adith Swaminathan and Thorsten Joachims. “Batch learning from logged bandit feedback through counterfactual risk minimization”. In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1731–1755.

[43] Adith Swaminathan and Thorsten Joachims. “The self-normalized estimator for counterfactual learning”. In: *Neural Information Processing Systems (NeurIPS)*. 2015.

[44] Zhiqiang Tan. “A distributional approach for causal inference using propensity scores”. In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1619–1637.

[45] Liang Tang et al. “Automatic ad format selection via contextual bandits”. In: *ACM International Conference on Information and Knowledge Management (CIKM)*. 2013.

[46] Ambuj Tewari and Susan A Murphy. “From ads to interventions: Contextual bandits in mobile health”. In: *Mobile health: sensors, analytic methods, and applications* (2017), pp. 495–517.

[47] Philip Thomas and Emma Brunskill. “Data-efficient off-policy policy evaluation for reinforcement learning”. In: *International Conference on Machine Learning (ICML)*. 2016.

[48] Masatoshi Uehara, Masahiro Kato, and Shota Yasui. “Off-policy evaluation and learning for external validity under a covariate shift”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 49–61.

[49] Joel Veness, Marc Lanctot, and Michael Bowling. “Variance reduction in monte-carlo tree search”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 1836–1844.

[50] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudik. “Optimal and adaptive off-policy evaluation in contextual bandits”. In: *International Conference on Machine Learning (ICML)*. 2017.

[51] Zhouhao Yang et al. “Distributionally Robust Policy Gradient for Offline Contextual Bandits”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 6443–6462.

[52] Yisong Yue, Sue Ann Hong, and Carlos Guestrin. “Hierarchical exploration for accelerating contextual bandits”. In: *International Conference on Machine Learning (ICML)*. 2012.

Chapter 3

DISENTANGLING LINEAR QUADRATIC CONTROL WITH UNTRUSTED ML PREDICTIONS

This chapter is based on the paper:
Tongxin Li*, Hao Liu*, Yisong Yue. "Disentangling Linear Quadratic Control with Untrusted ML Predictions." In: *Advances in Neural Information Processing Systems 37 (2024)*.

3.1 Introduction

We study the problem of online decision-making with predictions. Such settings are increasingly popular with the use of machine learning for predictive modeling, with applications to power grids and robotics [35, 37, 49, 60]. However, in many real-world settings, such black-box predictors can be unreliable due to practical problems such as high model variability [24, 45], and out-of-distribution generalization issues [40, 55]. In settings that require making reliable and high-quality decisions, recent research has been working on designing the decision-making policy that can be adaptively in response to receiving low-quality predictions [37].

In this paper, we consider the setting where there are different sources of disturbance, each representing distinct (and often independent) resources or factors. For instance, in electricity grids, machine learning (ML) forecasts of variables like power consumption/generation fluctuations, electricity prices, and battery states can facilitate near-optimal operational decisions on the one hand, but on the other hand, latent variables such as power injections on certain loads can be highly unpredictable due to their volatility [48]. Another example is the drone navigation task [43], where the challenge lies in managing external perturbations caused by a mixture of predictable elements like air flows and less predictable factors like raindrops. When performing online decision-making in this setting, one important technical challenge is to be able to combine disentanglement with an adaptive online learning algorithm to result in a more exact estimation of the trustfulness of the disturbance. To address this challenge, in this work, we focus on a linear quadratic control problem where system perturbations originate from unidentified and possibly heterogeneous latent resources/components.

Our contributions. We develop a novel policy Disc that does the disentanglement and learns the adaptive parameter online simultaneously. We first introduce a policy, λ-Con, which extends the λ-confident approach in [37] by adapting a vectorized confidence parameter $\lambda \in [0,1]^k$, where each $\lambda(i)$ represents the estimated trustworthiness of the ML prediction for the i-th latent variable in the dynamical system. We then show that a static λ cannot guarantee an optimal consistency and robustness tradeoff (see Definition 3.3 for a formal description) for λ-Con: if λ-Con is $(1 + o(1))$-*consistent*, then it is at least $\omega(1)$-*robust*.

To circumvent this limitation, we propose the dynamic policy Disc (Section 3.3). This algorithm leverages online learning to optimize the confidence parameter λ_t at each time t. We establish competitive ratio guarantees for Disc in both linear and general mixing scenarios, as shown in Theorems 3.7 and 3.8 (Section 3.4) respectively. Under Assumption 3.1 and 3.4. The competitive ratio bound for Disc, outlined informally as follows, incorporates a term that embodies our "*best-of-both-worlds utilization*" of untrusted ML predictions:

$$\mathbb{E}\left[\mathsf{CR}(\textsc{Disc})\right] \leq 1 + o(1) + O\left(\rho^{2w}\right) + \underbrace{O\left(\sum_{i=1}^{k} \frac{\overline{\varepsilon}(i)}{\Omega(T/w) + \overline{\varepsilon}(i)}\right)}_{\textit{Best-of-both-worlds utilization}}, \tag{3.1}$$

where the $o(1)$ term hides quantities that vanish when the total number of steps T increases; k is the number of latent variables generating the perturbations; $\rho \in (0,1)$; w denotes the prediction window size and each $\overline{\varepsilon}(i)$ (for $i = 1, \ldots, k$) denotes the prediction error corresponding to each latent component. The last term in this result highlights the desired performance guarantee: When a component-wise prediction error $\overline{\varepsilon}(i)$ is small, the individual term $\overline{\varepsilon}(i)/(\Omega(T/w) + \overline{\varepsilon}(i))$ will be negligible; otherwise, it always holds that $\overline{\varepsilon}(i)/(\Omega(T/w) + \overline{\varepsilon}(i)) \leq 1$, regardless of how high the prediction error becomes. Our proposed Disc is both $(1 + o(1))$-consistent and $O(1)$-robust with a sufficiently large prediction window size. We offer the first bound of this nature, grounded in a term as a function of the prediction error, without restrictive assumptions on the errors $(\overline{\varepsilon}(1), \ldots, \overline{\varepsilon}(k))$.

Proving our main result above is nontrivial due to the fact that despite it is known that an input-disturbed linear system can be reduced to an online convex optimization (OCO) with structured memory [49], the connection between the problem with λ-Con and a memoryless online optimization is not previously discovered. In Lemma 3.5,

we provide a result that decouples the loss terms in the total regret in equation 3.9 and future perturbations, thereby reducing the problem of choosing λ_t to an online optimization instance. Then we use a two-stage analysis as depicted in Figure B.4 that combines a dynamic regret bound for our control policy and static regret bounds induced by online learning algorithms to derive the main result.

We demonstrate the practicality of Disc through two real-world examples (Section 3.5): a drone navigation problem with mixed external disturbances and voltage control in a power grid with heterogeneous power injections. We demonstrate that Disc, when applied with disentangled ML predictions, outperforms baselines that do not distinguish between underlying latent variables. Additionally, Disc shows remarkable adaptability to rapid changes in various scenarios, such as those involving ML models trained with out-of-distribution data in a non-stationary environment.

3.2 Problem Formulation

Notational conventions. Throughout this chapter, $\|\cdot\|$ denotes the ℓ_2-norm for vectors and the matrix norm induced by the ℓ_2-norm. We use a subscript $(\cdot)_t$ to represent a length-k vector $s_t = (s_t(1), \ldots, s_t(k))$ at time t whose i-th coordinate is written as $s_t(i)$. We use $\lambda \circ s \coloneqq (\lambda(i)s(i), i = 1, \ldots, k) \in \mathbb{R}^k$ to denote the Hadamard product between vectors $\lambda, s \in \mathbb{R}^k$.

Linear Quadratic Control with Latent Perturbations

We consider a finite-time linear dynamical system with *latent perturbations*. Let $[T] \coloneqq \{0, \ldots, T-1\}$ be the set of time steps. Denote by $x_t \in \mathbb{R}^n$ a system state and $u_t \in \mathbb{R}^m$ an action from a state feedback policy π_t at time $t \in [T]$. The system update rule is given by

$$x_{t+1} = Ax_t + Bu_t + f(s_t; \theta), \quad t \in [T] \tag{3.2}$$

where $s_t \coloneqq (s_t(1), \ldots, s_t(k)) \in \mathbb{R}^k$ ($k \leq n$) contains k latent variables (components) that together generate a perturbation at time $t \in [T]$ via a mixing function $f : \mathbb{R}^k \times \Theta \to \mathbb{R}^n$ parameterized by $\theta \in \Theta \subseteq \mathbb{R}^d$. The states and the actions are observed after being generated, while the mixing function f transforms the unobservable latent variables in s_t to an additive system perturbation. Throughout this paper, we focus on the nontrivial regime that $f(s_t; \theta) \neq \mathbf{0}_n$ for all $t \in [T]$.

In equation 3.2, $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are system matrices. We consider the standard regime that the pair (A, B) is stabilizable [37, 50]. Without loss of generality, we also assume the system is initialized with some fixed $x_0 \in \mathbb{R}^n$. The goal of control

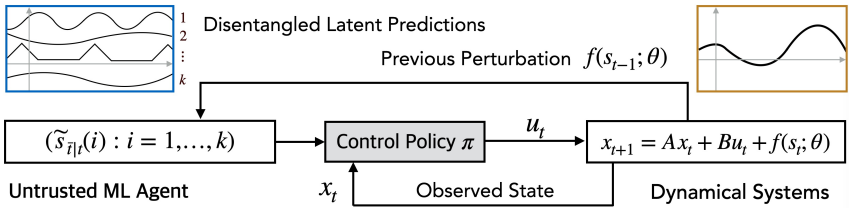


Figure 3.1: Overview of the system model considered in this work. **Left** time series: Disentangled latent variable time series predictions provided by an untrusted ML agent at time $t \in [T]$, represented by $(\widetilde{s}_{\bar{t}|t}(i) : i = 1, \ldots, k)$; **Right** time series: Observed mixed perturbations $(f(s_\tau;\theta) : \tau < t)$ at time $t-1$.

is to minimize the following quadratic costs given matrices A, B, Q, R:

$$J(\pi) \coloneqq \sum_{t=0}^{T-1} \left(x_t^\top Q x_t + u_t^\top R u_t\right) + x_T^\top P x_T, \tag{3.3}$$

where $\pi \coloneqq (\pi_t : t \in [T])$; $Q \in \mathbb{R}^{n\times n}$, $R \in \mathbb{R}^{m\times m}$ are positive definite matrices, and P is a symmetric positive definite cost-to-go solution of the following discrete algebraic Riccati equation (DARE), which must exist because (A, B) is stabilizable and Q, R are positive definite [14]:

$$P = Q + A^\top P A - A^\top P B(R + B^\top P B)^{-1} B^\top P A.$$

Given P, we define $K \coloneqq (R + B^\top P B)^{-1} B^\top P A$, and $u_t = -K x_t$ is the feedback control law of a linear quadratic regulator (LQR), which is optimal in the case of zero disturbances ($w_t = 0$ for all $t \in [T]$). Further, let $F \coloneqq A - BK$ be the closed-loop system matrix. The Gelfand's formula implies that there must exist a constant $C_F > 0$ and a spectral radius $\rho_F \in (0, 1)$ such that $\|F^t\| \leq C_F \rho_F^t$ for all $t \in [T]$.

In this work, we focus on designing a near-optimal policy $\pi = (\pi_t : t \in [T])$ that minimizes the total quadratic cost in equation 3.3 subject to the linear dynamics in equation 3.2. At time $t \in [T]$, each $\pi_t : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^m$ is a state-feedback policy that produces an action $u_t \in \mathbb{R}^m$ with an observed state x_{t-1}, and ML predictions $(\widetilde{s}_{\bar{t}|t}(i) : i = 1, \ldots, k)$ of the latent variables $s_t \in \mathbb{R}^k$ for future w time steps with a prediction window size $w > 0$ (denoting $\bar{t} \coloneqq \min\{t + w - 1, T - 1\}$). Figure 3.1 above summarizes our system model. We defer a detailed introduction of the ML predictions in Section 3.2, together with the performance benchmarks considered in this paper. We relegate concrete latent perturbation modelling examples and real-world applications to Appendix B.1.

In light of the existing nonlinear ICA models summarized in Appendix B.1, throughout this paper, we assume the continuity and invertibility conditions hold for the mixing

function f to guarantee our theoretical results. This assumption is weaker compared to those required in the nonlinear ICA literature [17, 18, 26, 58, 62] to guarantee identifiability, as summarized in Table B.1 (see Appendix B.1), noting that our goal is instead to provide a near-optimal competitive ratio bound regardless of the error of disentanglement and latent variable predictions (see our performance benchmarks defined in Section 3.2).

Assumption 3.1. *The mixing function $f : \mathbb{R}^k \times \Theta \to \mathbb{R}^n$ is Lipschitz continuous and bijective with respect to $s \in \mathbb{R}^k$.*

Performance Benchmark

Let $\widetilde{s}_{\tau|t}$ be the latent variable time series value at time τ, predicted by an untrusted ML agent at time $t \leq \tau$. The prediction window size is an integer $w > 0$. We write $\bar{t} \coloneqq \min\{t + w - 1, T - 1\}$. The estimated mixing parameter at each time $t \in [T]$ is denoted by $\widetilde{\theta}_t$. Define the following error terms[1] for all $t \leq \tau \leq \bar{t}$ and $t \in [T]$:

$$\big(\varepsilon_{\tau|t}(1), \ldots, \varepsilon_{\tau|t}(k)\big) = \varepsilon_{\tau|t} \coloneqq s_\tau - \widetilde{s}_{\tau|t}, \quad \eta_t \coloneqq \widetilde{\theta}_t - \theta. \tag{3.4}$$

To assess the cumulative impact of prediction error over all T time steps, we define the total prediction error corresponding to the component-wise latent variable time-series prediction and the mixing parameter estimation as follows:

$$\overline{\varepsilon}(i) \coloneqq \sum_{t=0}^{T-1} \sum_{\tau=t}^{\overline{T}} \left(\rho_F^{\tau-t} \varepsilon_{\tau|t}(i)\right)^2, \quad \overline{\eta} \coloneqq \sum_{t=0}^{T-1} \|\eta_t\|^2, \tag{3.5}$$

where $\rho_F \in (0, 1)$ represents the spectral radius of the matrix F. The term $\rho_F^{\tau-t}$ in equation 3.5 accounts for the exponential decay phenomenon in the impact of component-wise errors, implying that predictions further into the future contribute less to the total error. This approach to error measurement is analogous to methods employed in linear quadratic control models with adaptive offline adversarial perturbations, as discussed in [37], which provides a foundational understanding of error evaluation in our context.

Our performance benchmark is the competitive ratio for a given prediction error ε, defined as follows. To be more precise, we focus on the competitive ratio

[1] We define two pairs (s, θ) and (s', θ') as *consistent* if they yield identical outputs in the mixing function f, i.e., $f(s; \theta) = f(s'; \theta')$. For simplicity in our notation, we will treat consistent pairs (s, θ) and (s', θ') as indistinguishable when they produce the same perturbation effect in the system. Here, at each time step $t \in [T]$, the estimated mixing parameter and the time series predictions are compared against the consistent pairs θ and s_τ with the smallest prediction error.

defined for deterministic online algorithms with an adaptive offline adversary that selects the system parameters A, B, Q, R, f, θ and latent time series $(s_t : t \in [T])$. Write $\varepsilon \coloneqq (\overline{\varepsilon}(i) : i = 1, \ldots, k)$ and denote by $J(\pi; \varepsilon)$ the total cost obtained by implementing $\pi = (\pi_t : t \in [T])$ with some fixed prediction error ε.

Definition 3.2. Fix some prediction error ε. The *competitive ratio* $\mathsf{CR}(\pi; \varepsilon)$ is defined as the smallest constant $C \geq 1$ such that $J(\pi; \varepsilon) \leq C \cdot J^\star$ for all A, B, Q, R, f, θ, and $(s_t : t \in [T])$ satisfying the model assumptions.

The following notions of consistency and robustness with respect to the competitive ratio offer a concise characterization to measure the algorithmic performance in the presence of prediction error ε. It aligns with the benchmarks used in the growing literature on algorithms with predictions [7, 9, 44, 54] (see further discussions provided in Section 3.6).

Definition 3.3. A policy π is γ-*consistent* if its competitive ratio satisfies $\mathsf{CR}(\pi; \varepsilon) \leq \gamma$ for $\varepsilon = 0$ and κ-*robust* if $\mathsf{CR}(\pi; \varepsilon) \leq \kappa$ for all ε.

3.3 Disentangled Confident Policy

In this section, we present our policy, Disc, which takes disentangled and untrusted ML predictions and achieves near-optimal consistency and robustness tradeoff (see Definition 3.3).

Warm-Up: Disentangled λ-Confident Policy (λ-Con)

Before proceeding to introduce Disc, we first consider a policy that balances consistency and robustness by combining an MPC policy that fully utilizes the untrusted ML predictions, and an LQR without considering any predictions.

Denote by $\bar{t} \coloneqq \min\{t + w - 1, T - 1\}$ and let $Y \coloneqq (R + B^\top P B)^{-1} B^\top$. Recall that as defined in Section 3.2, at each time $t \in [T]$, $\left(\widetilde{s}_{\tau|t} : t \leq \tau \leq \bar{t}\right)$ represents a sequence of predictions for the future values of the latent variables $(s_\tau : t \leq \tau \leq \bar{t})$, with these predictions spanning a predetermined window size $w > 0$. Similarly, the sequence $(\widetilde{\theta}_t : t \in [T])$ corresponds to the estimated mixing parameters. At each time $t \in [T]$, an estimate $\widetilde{\theta}_t$ is obtained by applying some disentanglement algorithm (see those examples in Section B.1). With this setup, we proceed to define an MPC-type action as follows:

$$u_t = -K x_t - Y \sum_{\tau=t}^{\bar{t}} \left(F^\top\right)^{\tau - t} P f\left(\lambda \circ \widetilde{s}_{\tau|t}; \widetilde{\theta}_t\right), \quad (\lambda\text{-Con Policy}) \tag{3.6}$$

Algorithm 2: Disentangled confidence (Disc) policy

for $t = 0, \ldots, T-1$ **do**

$\bar{t} \leftarrow \min\{t + w - 1, T - 1\}$

/* Online learning of λ_t */

if $t = 0$ **then** Initialize $\lambda_0 \in \mathcal{I}$

else Set λ_t = Online-Procedure $(\zeta_\ell : \ell \in [t])$ through equation 3.12 or equation 3.13 with ζ_ℓ defined in equation 3.11

/* Obtain disentangled predictions */

Get $(\widetilde{s}_{\tau|t} : t \le \tau \le \bar{t})$ and an estimated mixing parameter $\widetilde{\theta}_t$

/* Implement disentangled λ_t-confident policy */

Take u_t as in equation 3.6 with a learned λ_t such that

$$u_t = -Kx_t - Y \sum_{\tau=t}^{\bar{t}} \left(F^\top\right)^{\tau-t} Pf\left(\lambda_t \circ \widetilde{s}_{\tau|t}; \widetilde{\theta}_t\right), \quad (\lambda_t\text{-Con Policy})$$

Update and observe x_{t+1} following equation 3.2

end

which specifies a *disentangled* λ-confident policy, denoted by λ-Con, where $\lambda \in \mathcal{I} \coloneqq [0,1]^k$ is a fixed *trust parameter*. The term $\lambda \circ \widetilde{s}_{\tau|t}$ denotes the Hadamard product of λ and $\widetilde{s}_{\tau|t}$. It is worth noting that it is well known that equation 3.6 is an optimal solution of the following MPC scheme [37, 60]:

$$\min_{u \in \mathbb{R}^m} \sum_{\tau=t}^{\bar{t}} \left(x_\tau^\top Q x_\tau + u_\tau^\top R u_\tau\right) + x_{\bar{t}+1}^\top P x_{\bar{t}+1} \tag{3.7}$$

$$\text{s.t. } x_{\tau+1} = Ax_\tau + Bu_\tau + f\left(\lambda \circ \widetilde{s}_{\tau|t}; \widetilde{\theta}_t\right), \tau \in \left[t, \bar{t}\right]. \tag{3.8}$$

In particular, if λ is an all-one vector $\mathbf{1}_k$ in equation 3.7, it trusts the ML predictions. Note that it generalizes the λ-confident policy in [37], which linearly combines actions from an MPC policy and an LQR. In Section 3.4, we present a negative result for λ-Con, indicating that it cannot achieve the optimal consistency and robustness tradeoff (see Definition 3.3). This motivates the disentangled confident policy discussed in the next section.

Disentangled Confidence Policy (Disc)

At each time $t \in [T]$, the algorithm adaptively learns a *trust parameter* $\lambda_t \in \mathcal{I} \coloneqq [0,1]^k$, and uses a Hadamard product of λ_t and the ML learned latent variable $\lambda_t \circ \widetilde{s}_{\tau|t}$ to estimate future perturbations. The update rule of the trust parameter λ_t follows an Online-Procedure, which can be constructed by the following explicit form of the

overall dynamic regret (see [37, 59]) with a fixed $\lambda \in \mathcal{I}$:

$$J(\pi(\lambda)) - J^\star = \sum_{\ell=0}^{T-1} \psi_{\ell,T}^\top(\lambda) H \psi_{\ell,T}(\lambda), \tag{3.9}$$

where $H \coloneqq B(R + B^\top P B)^{-1} B^\top$, and $\psi_{\ell,T}(\lambda)$ is defined as

$$\psi_{\ell,T}(\lambda) \coloneqq \sum_{\tau=\ell}^{\min\{\ell+w-1,T-1\}} \left(F^\top\right)^{\tau-\ell} P \left(f\left(s_\tau; \theta_\tau\right) - f\left(\lambda \circ \widetilde{s}_{\tau|\ell}; \widetilde{\theta}_\ell\right)\right). \tag{3.10}$$

Consider the application of online optimization strategies to minimize total regret as depicted in equation 3.9. At each time t, we obtain $\psi_{\ell,0}(\lambda), \ldots, \psi_{\ell,t}(\lambda)$. For $\psi_{\ell,t}(\lambda)$, since the summation is over $\tau \in \{\ell, \min\{\ell+w-1, t-1\}\}$, $\psi_{\ell,t} : \mathcal{I} \to \mathbb{R}^n$ is a function that depends on $t \in [T]$. Thus, the formulation of the offline optimization equation 3.9 does not conform to a canonical framework suitable for online optimization. This necessitates a tailored approach for online learning, which is facilitated by Lemma 1, allowing for the online learning of λ. Below we assume the following to regulate the learned sequence $(\lambda_t : t \in [T])$, which holds for typical online learning algorithms with stationary environments [16, 25, 53]. Note that when $t - \tau < 0$, we consider $\lambda_{t-\tau} = \lambda_0$.

Assumption 3.4. *If $\tau > 0$ is a constant, then $(\lambda_t : t \in [T])$ satisfies that $\sum_{t=0}^{T-1} |\lambda_t - \lambda_{t-\tau}| = o(T)$.*

Define $\underline{\ell} \coloneqq \max\{\ell - w + 1, 0\}$. The following lemma helps convert equation 3.9 to a form that can be reduced to an online optimization problem. The proof can be found in Appendix B.3.

Lemma 3.5. *Define a function $\zeta_\ell : \mathcal{I} \to \mathbb{R}^n$ as $\zeta_\ell(\lambda) \coloneqq \sum_{\tau=\underline{\ell}}^{\ell} g^\top(\tau, \ell) H g(\tau, \ell)$, where*

$$g(\tau, \ell) \coloneqq \left(F^\top\right)^{\tau-\ell} P \left(f\left(s_\tau; \theta_\tau\right) - f\left(\lambda \circ \widetilde{s}_{\tau|\ell}; \widetilde{\theta}_\ell\right)\right). \tag{3.11}$$

Then for any $t \in [T]$, it follows that $\sum_{\ell=0}^{t-1} \psi_{\ell,t}^\top(\lambda) H \psi_{\ell,t}(\lambda) \le w \sum_{\ell=0}^{t-1} \zeta_\ell$.

Note that each $\zeta_\ell(\lambda)$ defines a convex function of λ, as verified in Appendix B.3. Therefore, the selection of λ_t at each time $t \in [T]$ follows from an online learning procedure Online-Procedure, whose concrete realizations include the follow-the-regularized-leader (FTRL) approach with an ℓ_2-norm regularizer [16, 25], which is

equivalent to online mirror descent (OMD) when f is convex; and the follow-the-perturbed-leader (FTPL) [2, 15, 28, 53] for online non-convex learning, taken previous quantities $(\zeta_\ell : \ell \in [t])$ as the input. Next, we will discuss theoretical guarantees for linear and general mixing models with specified online learning procedures, as detailed later in Section 3.4. Our policy is summarized in Algorithm 2.

3.4 Main Results

In this section, we present our technical results, proving that the scheme in Disc achieves near-optimal competitive ratio bounds for both linear and general mixing cases. Proving our main result above is nontrivial due to the fact that despite it is known that an input-disturbed linear system can be reduced to an online convex optimization (OCO) with structured memory [49], the connection between the problem with λ-Con and a memoryless online optimization is not previously discovered. In Lemma 3.5, we provide a result that decouples the dependency between cost functions in our problem that depends on previous actions via a linear dynamical system (see the dynamical system defined in equation 3.2), thereby reducing the problem of choosing λ_t to an online optimization instance, then use a two-stage analysis (see Figure B.4) that combines the dynamic regret analysis of λ-Con and static regret bounds corresponding to applying online optimization algorithms to learn a confidence parameter.

A Fundamental Gap

First, we motivate the necessity of learning $(\lambda_t : t \in [T])$ online. Based on Definition 3.3, the following theorem reveals a negative result of λ-Con, since for any fixed non-zero λ, there always exists predicted time series $\left(\widetilde{s}_{\tau|t} : t \le \tau \le \bar{t}, t \in [T]\right)$ such that the competitive ratio $\mathsf{CR}(\lambda\text{-Con})$ for λ-Con can be unbounded (when T goes to infinity).

Theorem 3.6 (Consistency-Robustness Impossibility for $\mathsf{CR}(\lambda\text{-Con})$)**.** *If the λ-confident policy λ-Con is $(1 + o(1))$-consistent, then it is at least $\omega(1)$-robust, even if the mixing parameter estimate is perfect, i.e., $\overline{\eta} = 0$.*

Here, $o(\cdot)$ and $\omega(\cdot)$ characterize the asymptotic asymptotic growth rates with respect to the time horizon length T. Next, we show in both the linear and general mixing settings, $\mathsf{CR}(\text{Disc})$ can be bounded as previewed in Section 3.1, thus the gap between $\mathsf{CR}(\lambda\text{-Con})$ and $\mathsf{CR}(\text{Disc})$ can be arbitrarily large as implied by Theorem 3.6 above. Especially, Corollary B.3 in Section 3.4 implies that Disc is both $(1 + o(1))$-consistent

and $O(1)$-robust, with a sufficiently large prediction window size $w = \omega(1)$, revealing that online learning of the confidence parameter provides a significant improvement of the consistency and robustness tradeoff.

Linear Mixing

We consider a linear mixing setting, where the mixing function f is linear such that $f(s;\theta) = \theta s$ for all $s \in \mathbb{R}^k$ where $\theta = (\theta_{ij})$ denotes an $n \times k$ full column rank mixing matrix. Suppose $(s_t : t \in [T])$ and θ are bounded such that $\|s_t\| \leq \overline{s}$ for all $t \in [T]$ and $\|\theta\| \leq \overline{\theta}$. For this particular instance, we implement a (follow-the-regularized-leader) FTRL procedure that sets

$$\lambda_t \in \operatorname*{arg\,min}_{\lambda \in \mathcal{I}} \left(\sum_{\ell=0}^{t-1} \nabla_\lambda^\top \left(\zeta_\ell(\lambda)\right) \lambda + \frac{1}{\beta} \|\lambda - \lambda_0\|^2 \right) \quad \text{(FTRL for } \lambda\text{-learning)} \quad (3.12)$$

at each time $t \in [T]$ for some $\beta > 0$ that can be optimized. The Disc policy satisfies the following competitive ratio bound, whose proof can be found in Appendix B.4.

Theorem 3.7 (Disc for Linear Mixing)**.** *With our model assumptions, the competitive ratio of Disc with a linear mixing function satisfies*

$$\mathsf{CR}(\textit{Disc}) \leq 1 + O\left(\sum_{i=1}^{k} \frac{\overline{\varepsilon}(i)}{\Omega(T/w) + \overline{\varepsilon}(i)} \right) + O\left(w\sqrt{\frac{k}{T}} \right) + O\left(\rho_F^{2w} + \frac{\overline{\eta}}{T} \right),$$

where k is the number of latent variables; T is the time horizon length; $\rho_F \in (0,1)$ is the spectral radius of F; $(\overline{\varepsilon}(i) : i = 1, \ldots, k)$ and $\overline{\eta}$ are defined in equation 3.5; $O(\cdot)$ and $\Omega(\cdot)$ hide multiplicative constants (see the details in Appendix B.4).

This result highlights a consistency and robustness tradeoff of Disc, as claimed in Section 3.1, which offers a dual advantage: it exploits accurate predictions when available, and safeguards against the ramifications of trusting inaccurate forecasts, thus ensuring system performance reliability. Besides, our analysis can be carried out to convex mixing functions, as the online learning of λ_t in equation 3.12 via FTRL guarantees a sub-linear static regret as long as $\zeta_\ell(\lambda)$ is convex in λ for all $\ell \in [T]$. We further provide a bound for sample efficient ICA in the appendix.

General Mixing

In this section, we extend the linear assumption on f by considering a general setting where the mixing function f is an arbitrary Lipschitz continuous function (see Assumption 3.1). To deal with the non-convexity of f, we revisit the (follow-the-perturbed-leader) FTPL approach (see [2, 53]) and consider the following online

learning procedure for tuning $(\lambda_t : t \in [T])$:

$$\lambda_t \in \arg\min_{\lambda \in \mathcal{I}} \left(\sum_{\ell=0}^{t-1} \zeta_\ell(\lambda) + \sigma_t^\top \lambda \right) \quad \text{(FTPL for } \lambda\text{-learning)}. \tag{3.13}$$

Here, σ_t is a length-k random vector with each coordinate $\sigma_t(i)$ $(i = 1, \ldots, k)$ being an IID random variable from an exponential distribution. The Disc policy satisfies the following competitive ratio bound.

Theorem 3.8 (Disc for General Mixing)**.** *With our model assumptions, the expected competitive ratio of Disc satisfies*

$$\mathbb{E}\left[\mathsf{CR}(\mathit{Disc})\right] \leq 1 + O\left(\sum_{i=1}^{k} \frac{\overline{\varepsilon}(i)}{\Omega(T/w) + \overline{\varepsilon}(i)} \right) + O\left(\overline{s} \rho_F^{2w} + \frac{\overline{\eta}}{T} \right) + O\left(\frac{w k^2}{\sqrt{T}} \right), \tag{3.14}$$

where the parameters $\rho_F, w, k, T, \overline{\eta}$, and $(\overline{\varepsilon}(i) : i = 1, \ldots, k)$ are the same as in Theorem 3.7 and $\|s_t\| \leq \overline{s}$ for all $t \in [T]$. The expectation is taken over the randomly sampled $(\sigma_t : t \in [T])$.

The notation $O(\cdot)$ and $\Omega(\cdot)$ above in Theorem 3.8 hide multiplicative constants such as the Lipschitz constant of the mixing function f (details are provided in Appendix B.7). The result above implies the best-of-both-worlds prediction utilization as previewed in Section 3.1, except that the online learning of λ_t converges slower with a rate k^2, causing the term $O(wk^2/\sqrt{T})$ in the competitive ratio bound equation 3.14. Unlike the linear setting, to ensure a near-optimal expected competitive ratio, the latent time series $(s_t : t \in [T])$ is not necessarily bounded, as long as the prediction window size w is sufficiently large (e.g., $w = \Theta(\log T)$) so that the term $\overline{s}\rho_F^{2w}$ vanishes. Finally, note that the online learning procedure of λ_t is not limited to FTPL. For example, the non-convex online optimization algorithm in [57] can be used to optimize $(\lambda_t : t \in [T])$ online and leads to a similar guarantee. The proof of Theorem 3.8 can be found in Appendix B.7.

3.5 Experiments

Experimental Setup. To generate ML predictions, we use the FastICA method in [21] to decompose the mixed perturbations and train a multi-layer perceptron neural network with 4 hidden layers to predict the future latent variables. FastICA is a simple and efficient linear ICA method that aims to find an orthogonal rotation of the observation that maximizes the rotated components' non-Gaussianity using fixed-point iteration. Furthermore, in our experiments, we implement a follow-the-regularized-leader (FTRL) optimization with a ℓ_2-regularizer $\|\lambda - \lambda_0\|^2$, which is

equivalent to the following equivalent online mirror descent (OMD) implementation [16]: $\tau_t = \tau_{t-1} - \frac{\beta}{2}\nabla_\lambda \left(\zeta_t^\top H \zeta_t\right)$, $\lambda_t = \mathsf{Proj}_{\mathcal{I}}(\tau_t)$. The detailed parameters and hyper-parameters used in our experiments can be found in Appendix B.2. The code is available at https://github.com/tspbfs/DisentangleControl.

Experiment A: Drone Navigation with Mixed Disturbances

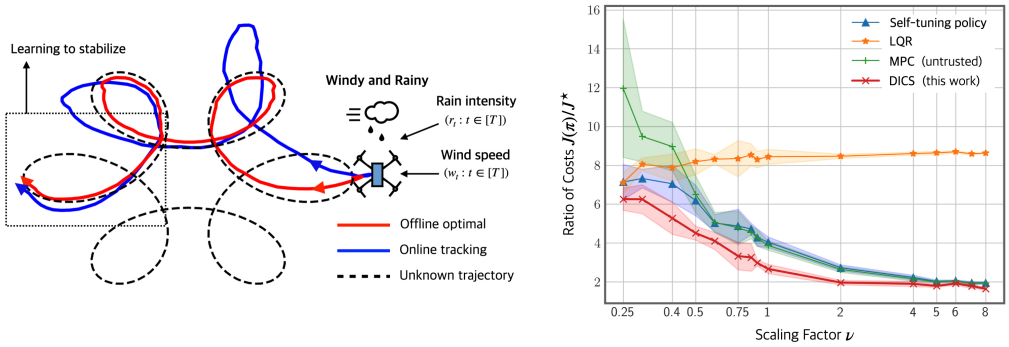


Figure 3.2: **Drone Navigation** under challenging windy and rainy weather conditions, with the drone's target path illustrated by a dotted curve. The external perturbations impacting the drone's flight are modeled by two independent, time-varying latent variables: w_t for wind speed and r_t for rain intensity at time $t \in [T]$. The trajectory produced by the Disc policy (Algorithm 2 in Section 3.3) is represented by the blue curve, while that from the offline optimal policy is traced in red. More comparison results with other baseline policies can be found in Appendix B.2. Right: **Ratio of Costs** $J(\pi)/J^\star$ (y-axis) between Disc (red), linear quadratic regulator (LQR, orange), model predictive control with untrusted ML predictions (MPC (Untrusted), green), and the self-tuning policy (blue), with a varying scaling factor ν from 0.25 to 8. Shadow area depicts the range of standard deviations for 5 random tests.

We first consider the drone piloting task described in Example B.1 (a detailed description is delegated to Appendix B.1) to track an unknown trajectory, shown on the left of Figure 3.2.

Competitive Ratio Analysis. We present a comparative analysis of the ratio of costs achieved by our Disc policy (detailed in Algorithm 2 in Section 3.3) on the right of Figure 3.2, against several benchmarks: the Linear Quadratic Regulator (LQR), Model Predictive Control (MPC) using untrusted ML predictions (i.e., setting $\lambda = \mathbf{1}_k$ in Equation equation 3.7), and the self-tuning policy outlined in [37] and [39].

We examine the impact of varying a scaling factor ν in the range 0.25 to 8, adjusting the additive perturbation $r_t c_2$ in equation B.2 to $(r_t/\nu)c_2$. This scaling factor ν serves as a proxy for different rain intensities, influencing the level of environmental unpredictability—the larger ν is, the more predictable the environment. With the

theoretically guaranteed best-of-both-worlds utilization of disentangled and untrusted ML predictions, Disc consistently outperforms existing baselines in terms of cost ratios across varying levels of environmental predictability.

Example B: Voltage Control with Heterogeneous Power Injections

We consider the voltage control task outlined in Example B.2 for a distribution power grid with 11 nodes, as depicted in Figure 3.3. Specifically, nodes 3, 5, and 8 are active, each supplying dynamic active power injections $(p_t(i) : i = 3, 5, 8)$ at every time step $t \in [T]$. Collectively, these nodes induce the composite perturbation $Gp_t + v_0\mathbf{1}_n$. The objective is to apply the Disc algorithm to determine near-optimal controllable reactive power injections that will effectively mitigate voltage fluctuations across the grid. More details of the problem setting can be found in Appendix B.2.

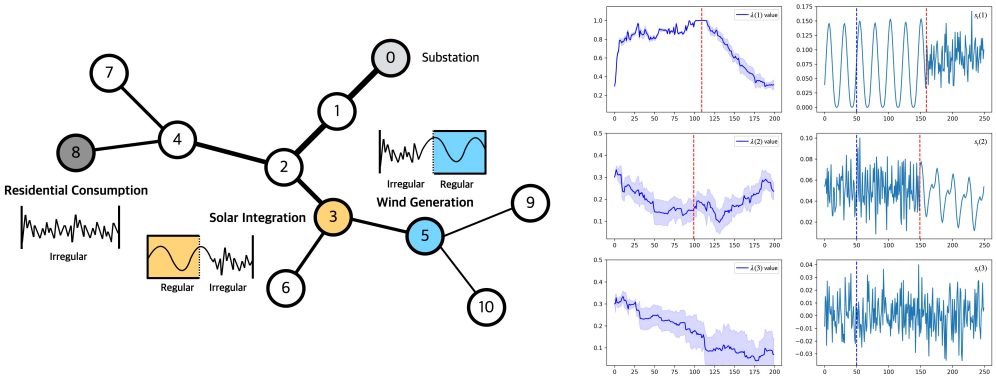


Figure 3.3: **Voltage Control** with in a dynamic environment. This figure illustrates the heterogeneity and variability of power injections at different nodes within an electrical grid, including a residential area, a solar photovoltaic (PV) system, and a wind turbine. Right: **Example B.** *Voltage Control (Section 3.5).* **Left column**: *Convergence of confidence parameters* $\lambda(1), \lambda(2), \lambda(3)$ *corresponding to* 3 *latent components*. **Right column**: *Temporal dynamics of latent time series* s_t *in* $\mathbb{R}$. We illustrate the time series data for three key latent variables in the energy system (see Figure 3.3): Gaussian-distributed residential consumption (bottom), real-world photovoltaic (PV) integration (top), and wind generation (middle). The red dotted line marks a critical transition point where there is a notable shift in the power generation patterns. We use time series before the blue dotted line as the buffered data to warm start the ML model, learn the mixing matrix, and obtain disentangled predictions. The x-axis in each graph marks the time steps, while the y-axis denotes the values of the time series.

We consider a realistic scenario when and use this to demonstrate the adaptivity of Disc in changing environments. Figure 3.3 shows the change of patterns corresponding to the solar integration and wind generation. We set $T = 200$. At

time $t = 110$, the solar generation switches from a regular pattern to generating random Gaussian noise, representing miscommunication or system faults. Similarly, at time $t = 100$, the wind generation is recovered from the irregular mode to a regular mode. For the regular solar and wind generation time series, we use the real solar PV generation time series data from the DTU-Data in 2021 [27] and wind generation from the U.S. Virgin Islands Wind Resources from the National Renewable Energy Lab (NREL) [46].

Adaptability Amidst Environmental Variability. Figure 3.3 exhibits the adaptability of our approach in response to environmental shifts in renewable energy generation and consumption patterns. Specifically, the figure presents the time series for solar generation ($s_t(1) : t \in [T]$), wind generation ($s_t(2) : t \in [T]$), and residential power consumption ($s_t(3) : t \in [T]$). The transition points in the time series patterns are marked by dotted red lines in the figure.

A notable observation is the reactive behavior of the confidence parameters to the changes in predictability within each time series. For instance, as the solar generation transitions to an irregular Gaussian noise profile at $t \geq 110$, this unpredictability precipitates a decline in the corresponding confidence parameter $\lambda_t(1)$. Conversely, the wind generation time series exhibits a move towards a more predictable pattern past $t \geq 100$, resulting in a progressive increase in the confidence parameter $\lambda_t(2)$. In the case of residential consumption, which is consistently modeled as Gaussian noise, the algorithm adaptively learns to reduce the confidence parameter $\lambda(3)$. The resilience of the Disc algorithm in dynamically adjusting confidence levels demonstrates its robustness against environmental shifts, a critical feature for real-world applications where conditions are subject to sudden and unpredictable changes. This further highlights the efficacy of Disc in maintaining system stability and performance through intelligent adaptability to the reliability of input data, as verified by the theoretical results in Section 3.4.

3.6 Related Work

Online Decision-Making with Predictions. The challenge posed by untrusted ML advice in online decision-making processes has attracted increasing attention, as evidenced by recent studies (e.g., [42, 44]) for various models such as ski rental [7, 44, 54], caching [23, 41, 47], bipartite matching [5], online covering [4, 6], online optimization [9, 30], reinforcement learning [33, 56], control [34, 35, 37, 38], and real-world applications [10, 31, 32, 36] with focuses on exploring the implications

and potential solutions for handling unreliable ML predictions. The most relevant result to this work is [37], which establishes a consistency-robustness tradeoff for the linear quadratic control problem with untrusted predictions. However, the self-tuning algorithm proposed in [37] has a competitive ratio that suffers from high variability of system perturbations and predictions, yielding a sub-optimal consistency and robustness tradeoff provided there.

Linear Quadratic Control Our work intersects with and extends upon established research in the area of linear quadratic control (LQC) systems and the analysis of the linear quadratic regulator (LQR), particularly in the context of integrating predictions with control strategies. The existing literature primarily explores the balance between achieving optimal control and managing the uncertainties inherent in predictive models. Significant work has been conducted in the realm of LQR systems, especially regarding model predictive control (MPC). For instance, Yu et al. in [60] have analyzed LQR regret in MPC under the assumption of accurate perturbation predictions. The implications of imprecise predictions are explored in [59]. In addition, there are notable contributions to the discourse on regret and competitive ratio in MPC, as evidenced by works like [37, 49, 61]. Our work is closely related to [37], which reveals a consistency and robustness tradeoff in LQC, with a self-tuning policy that updates a 1-dimensional confidence parameter based on past observations.

Adaptive and Robust Control Moreover, our approach also aligns with the adaptive control paradigm, especially in its recent intersection with learning theory to address non-asymptotic metrics. While the adaptive control community has traditionally concentrated on Lyapunov stability and asymptotic convergence [52], the newer trend represented in [1, 3, 12, 51] employs learning-theoretic measures such as regret and dynamic regret for finite-time horizon analysis. Our results contribute to this evolving field by presenting a adaptive control policy that achieves the first of-its-kind near-optimal consistency and robustness tradeoff, as a novel endeavor in the context of LQR systems that also incorporates the unique challenge raised by untrusted predictions. Besides, robust control is a large area that concerns the design of controllers with performance guarantees that are robust against model uncertainty or adversarial disturbances [14]. Tools of robust control include H_∞ synthesis [13, 63] and robust MPC [8]. Our work diverges from these existing approaches by considering an LQC model that employs disentangled predictions. While it adheres to

the robust control principle of resilience to uncertainties and perturbations, our work also aims to achieve near-optimal performance when predictions become accurate. This best-of-both-worlds strategy in utilizing untrusted predictions introduces a novel dimension to the robustness-consistency tradeoff, a concept not extensively explored in the existing robust control literature.

Learning Disentangled Latent Variables Disentanglement, a key goal of which is to separate the typically independent effects of latent variables on observations, is a well-established concept within the field of representation learning. For instance, when the underlying mixing function that mixes latent independent signals to observations is linear with respect to s_t, i.e., $f(s_t; \theta) = \theta s_t$, for a matrix $\theta \in \mathbb{R}^{n \times k}$, linear independent component analysis (ICA) [11, 21] can be effectively employed to identify the k latent components. This linear approach, while powerful in its simplicity, is often insufficient for more complex, real-world data structures. In contrast, advanced nonlinear ICA methods [22] have been developed to address these complexities, providing more nuanced tools for disentangling latent perturbations. These methods extend the traditional ICA framework to accommodate nonlinear mixing functions, thereby enhancing its applicability to a wider range of scenarios. State-of-the-art techniques in nonlinear ICA have introduced novel algorithms that can effectively manage the intricacies of nonlinear relationships between observed and latent variables. Furthermore, nonlinear ICA has been explored under various settings, with multiple assumptions ensuring the identifiability of the model, such as sparsity [29, 62], auxiliary information [19, 26], and others. Recent work by Hyvärinen et al. [20] has utilized contrastive learning to identify the nonlinear ICA model in a time-series context, demonstrating the evolving capabilities of these methods. Table B.1 in Section B.1 provides a summary of key assumptions of recent nonlinear ICA methods. Note that these methods often incorporate deep neural networks as critical black-box elements within their algorithmic structures. We focus on optimally harnessing the predictions of latent variables derived from those aforementioned disentanglement methods, regardless of their trustworthiness. Therefore, our work uniquely represents a strategic shift towards effectively utilizing uncertain or unverified predictions in downstream decision-making tasks, a challenge not directly addressed by existing representation learning techniques.

3.7 Conclusion

Our work combines online control and the concept of learning disentangled predictions. In this paper, we have introduced a novel policy Dısc that achieves the best-of-both-world utilization of untrusted ML predictions of latent variables for a linear control problem, where the advantage of disentanglement is theoretically validated. The practicality of our method has been validated through two real-world applications, which exemplify its relevance and adaptability to complex, real-world scenarios. Looking ahead, our results open several intriguing paths. An immediate extension of particular interest is adapting our methodology to nonlinear dynamical systems, which could strengthen our current results. Furthermore, our work contributes to the growing community of algorithms with predictions, a compelling question arises: can the best-of-both-worlds competitive ratio bounds we have achieved be replicated across different online decision-making problems? Obtaining either positive or negative results would potentially lead to broad implications for the field of robust control and online learning.

References

[1] Yasin Abbasi-Yadkori and Csaba Szepesvári. “Regret bounds for the adaptive control of linear quadratic systems”. In: *Proceedings of the 24th Annual Conference on Learning Theory*. JMLR Workshop and Conference Proceedings. 2011, pp. 1–26.

[2] Naman Agarwal, Alon Gonen, and Elad Hazan. “Learning in non-convex games with an optimization oracle”. In: *Conference on Learning Theory*. PMLR. 2019, pp. 18–29.

[3] Naman Agarwal et al. “Online control with adversarial disturbances”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 111–119.

[4] Keerti Anand et al. “Online algorithms with multiple predictions”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 582–598.

[5] Antonios Antoniadis et al. “Online metric algorithms with untrusted predictions”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 345–355.

[6] Etienne Bamas, Andreas Maggiori, and Ola Svensson. “The Primal-Dual method for Learning Augmented Algorithms”. In: *arXiv preprint arXiv:2010.11632* (2020).

[7] Shom Banerjee. “Improving online rent-or-buy algorithms with sequential decision making and ML predictions”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21072–21080.

[8] Alberto Bemporad and Manfred Morari. “Robust model predictive control: A survey”. In: *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[9] Nicolas Christianson, Tinashe Handina, and Adam Wierman. “Chasing convex bodies and functions with black-box advice”. In: *Conference on Learning Theory*. PMLR. 2022, pp. 867–908.

[10] Nicolas Christianson et al. “Robustifying machine-learned algorithms for efficient grid operation”. In: *NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning*. 2022.

[11] Pierre Comon. “Independent component analysis, a new concept?” In: *Signal processing* 36.3 (1994), pp. 287–314.

[12] Sarah Dean et al. “On the sample complexity of the linear quadratic regulator”. In: *Foundations of Computational Mathematics* (2019), pp. 1–47.

[13] John Doyle et al. “State-space solutions to standard H_2 and H_∞ control problems”. In: *1988 American Control Conference*. IEEE. 1988, pp. 1691–1696.

[14] Geir E Dullerud and Fernando Paganini. *A course in robust control theory: a convex approach*. Vol. 36. Springer Science & Business Media, 2013.

[15] James Hannan. “Approximation to Bayes risk in repeated play”. In: *Contributions to the Theory of Games* 3 (1957), pp. 97–139.

[16] Elad Hazan and Satyen Kale. “Extracting certainty from uncertainty: Regret bounded by variation in costs”. In: *Machine learning* 80 (2010), pp. 165–188.

[17] Aapo Hyvarinen and Hiroshi Morioka. “Nonlinear ICA of temporally dependent stationary sources”. In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 460–469.

[18] Aapo Hyvarinen and Hiroshi Morioka. “Unsupervised feature extraction by time-contrastive learning and nonlinear ica”. In: *Advances in neural information processing systems* 29 (2016).

[19] Aapo Hyvarinen, Hiroaki Sasaki, and Richard Turner. “Nonlinear ICA using auxiliary variables and generalized contrastive learning”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 859–868.

[20] Aapo Hyvärinen, Ilyes Khemakhem, and Hiroshi Morioka. “Nonlinear independent component analysis for principled disentanglement in unsupervised deep learning”. In: *Patterns* 4.10 (2023).

[21] Aapo Hyvärinen and Erkki Oja. “Independent component analysis: algorithms and applications”. In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[22] Aapo Hyvärinen and Petteri Pajunen. “Nonlinear independent component analysis: Existence and uniqueness results”. In: *Neural networks* 12.3 (1999), pp. 429–439.

[23] Sungjin Im et al. “Parsimonious Learning-Augmented Caching”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 9588–9601.

[24] Tobias Johannink et al. “Residual reinforcement learning for robot control”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6023–6029.

[25] Adam Kalai and Santosh Vempala. “Efficient algorithms for online decision problems”. In: *Journal of Computer and System Sciences* 71.3 (2005), pp. 291–307.

[26] Ilyes Khemakhem et al. “Variational autoencoders and nonlinear ica: A unifying framework”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2207–2217.

[27] Matti Juhani Koivisto and Juan Pablo Murcia Leon. “Solar PV generation time series (PECD 2021 update)”. In: *DTU-data* (May 2022). DOI: 10.11583/DTU.19727239.v1. URL: https://data.dtu.dk/articles/dataset/Solar_PV_generation_time_series_PECD_2021_update_/19727239.

[28] Walid Krichene et al. “The hedge algorithm on a continuum”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 824–832.

[29] Sébastien Lachapelle et al. “Disentanglement via mechanism sparsity regularization: A new principle for nonlinear ICA”. In: *Conference on Causal Learning and Reasoning*. PMLR. 2022, pp. 428–484.

[30] Pengfei Li et al. “Robust learning for smoothed online convex optimization with feedback delay”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[31] Tongxin Li. “Learning-Augmented Control and Decision-Making: Theory and Applications in Smart Grids”. PhD thesis. California Institute of Technology, 2023.

[32] Tongxin Li and Chenxi Sun. “Out-of-Distribution-Aware Electric Vehicle Charging”. In: *IEEE Transactions on Transportation Electrification* (2024).

[33] Tongxin Li et al. “Beyond black-box advice: learning-augmented algorithms for MDPs with Q-value predictions”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[34] Tongxin Li et al. “Certifying black-box policies with stability for nonlinear control”. In: *IEEE Open Journal of Control Systems* 2 (2023), pp. 49–62.

[35] Tongxin Li et al. “Information Aggregation for Constrained Online Control”. In: *ACM SIGMETRICS Performance Evaluation Review* 49.1 (2021), pp. 7–8.

[36] Tongxin Li et al. “Learning-based predictive control via real-time aggregate flexibility”. In: *IEEE Transactions on Smart Grid* 12.6 (2021), pp. 4897–4913.

[37] Tongxin Li et al. “Robustness and Consistency in Linear Quadratic Control with Untrusted Predictions”. In: *ACM SIGMETRICS Performance Evaluation Review* 50.1 (2022), pp. 107–108.

[38] Yiheng Lin et al. “Bounded-regret mpc via perturbation analysis: Prediction error, constraints, and nonlinearity”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 36174–36187.

[39] Yiheng Lin et al. “Online Adaptive Policy Selection in Time-Varying Systems: No-Regret via Contractive Perturbations”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[40] Chaochao Lu et al. “Invariant causal representation learning for out-of-distribution generalization”. In: *International Conference on Learning Representations*. 2021.

[41] Thodoris Lykouris and Sergei Vassilvitskii. “Competitive caching with machine learned advice”. In: *Journal of the ACM (JACM)* 68.4 (2021), pp. 1–25.

[42] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. “Online optimization with uncertain information”. In: *ACM Transactions on Algorithms (TALG)* 8.1 (2012), pp. 1–29.

[43] Tiago Oliveira, A Pedro Aguiar, and Pedro Encarnacao. “Moving path following for unmanned aerial vehicles with applications to single and multiple target tracking problems”. In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1062–1078.

[44] Manish Purohit, Zoya Svitkina, and Ravi Kumar. “Improving online algorithms via ml predictions”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 9661–9670.

[45] Benjamin Recht. “A tour of reinforcement learning: The view from continuous control”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 2 (2019), pp. 253–279.

[46] Joseph Owen Roberts and Adam Warren. *US Virgin Islands Wind Resources Update 2014*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2014.

[47] Dhruv Rohatgi. “Near-optimal bounds for online caching with machine learned advice”. In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2020, pp. 1834–1845.

[48] Evan D Sherwin, Max Henrion, and Inês ML Azevedo. “Estimation of the year-on-year volatility and the unpredictability of the United States energy system”. In: *Nature Energy* 3.4 (2018), pp. 341–346.

[49] Guanya Shi et al. “Online optimization with memory and competitive control”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20636–20647.

[50] Sungho Shin et al. “Near-optimal distributed linear-quadratic regulator for networked systems”. In: *SIAM Journal on Control and Optimization* 61.3 (2023), pp. 1113–1135.

[51] Max Simchowitz and Dylan Foster. “Naive exploration is optimal for online lqr”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8937–8948.

[52] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. Vol. 199. Prentice hall Englewood Cliffs, NJ, 1991.

[53] Arun Sai Suggala and Praneeth Netrapalli. “Online non-convex learning: Following the perturbed leader is optimal”. In: *Algorithmic Learning Theory*. PMLR. 2020, pp. 845–861.

[54] Alexander Wei and Fred Zhang. “Optimal Robustness-Consistency Trade-offs for Learning-Augmented Online Algorithms”. In: *arXiv preprint arXiv:2010.11443* (2020).

[55] Florian Wenzel et al. “Assaying out-of-distribution generalization in transfer learning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 7181–7198.

[56] Jianyi Yang et al. “Anytime-competitive reinforcement learning with policy prior”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[57] Lin Yang et al. “An optimal algorithm for online non-convex learning”. In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2.2 (2018), pp. 1–25.

[58] Xiaojiang Yang et al. “Nonlinear ica using volume-preserving transformations”. In: *International Conference on Learning Representations*. 2021.

[59] Chenkai Yu et al. “Competitive control with delayed imperfect information”. In: *2022 American Control Conference (ACC)*. IEEE. 2022, pp. 2604–2610.

[60] Chenkai Yu et al. “The Power of Predictions in Online Control”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[61] Runyu Zhang, Yingying Li, and Na Li. “On the Regret Analysis of Online LQR Control with Predictions”. In: *arXiv preprint arXiv:2102.01309* (2021).

[62] Yujia Zheng, Ignavier Ng, and Kun Zhang. “On the identifiability of nonlinear ica: Sparsity and beyond”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16411–16422.

[63] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*. Vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

Chapter 4

UNCERTAINTY CALIBRATION FOR TOOL-USING LANGUAGE AGENTS

This chapter is based on the paper:
Hao Liu, Zi-Yi Dou, Yixin Wang, Nanyun Peng, Yisong Yue. "Uncertainty Calibration for Tool-Using Language Agents." In: *Findings of the Association for ComputationalLinguistics: EMNLP 2024*.

4.1 Introduction

Language agents are increasingly used to perform tasks and interact with a variety of external tools to achieve specific, goal-oriented objectives [16, 36]. Recent advancements have focused on designing agents to more effectively utilize a diverse set of tools for different tasks [27, 37, 41]. Tool-using agents have also been applied to many domains, such as employing retrieval tools for multi-hop reasoning [43], leveraging visual perception models for visual question answering [20], and utilizing calculators for mathematical reasoning [17].

Despite these advances, tool use poses inherent uncertainties for language agents. Tools may be imperfect and produce erroneous outputs, leading to suboptimal interactions and outcomes. For instance, in tasks requiring knowledge retrieval, the quality of the retrieval tool can significantly affect the agent's performance. If the retrieval tool fails to fetch accurate or relevant information, the agent's ability to reason and generate correct responses is compromised. In visual question answering tasks, Surís, Menon, and Vondrick [37] demonstrated that even with the correct algorithms, the performance bottleneck often lies in imperfect visual perception.

The above challenges highlight the need for systematically modeling and managing the uncertainties associated with tool-use in language agents. Existing approaches generally rely on the agent's internal probabilities to choose which tools to use and how to use them (e.g., Lu et al. [27] and Wang, Fried, and Neubig [41]). However, because these tools are typically not involved during the agent's training, their operation and outputs can be misaligned with how the agent reasons using the tools' outputs.

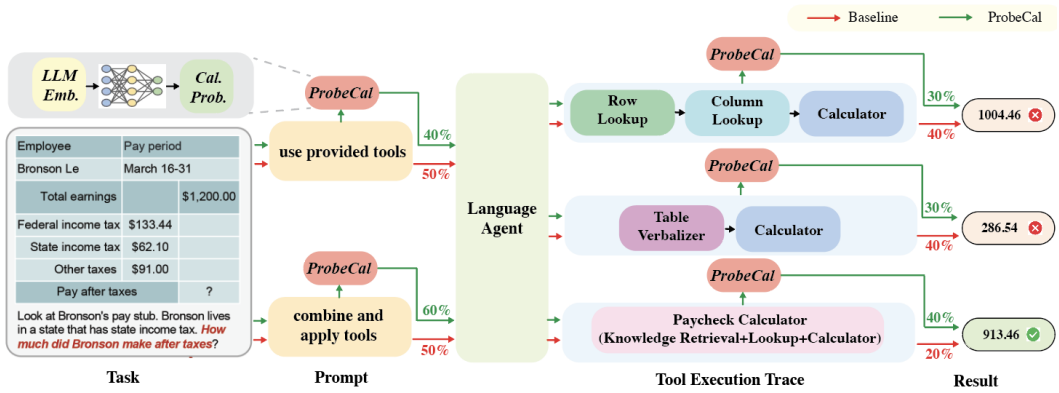


Figure 4.1: Overview of our ProbeCal framework for calibrating tool-using language agents. The process begins with a given task, exemplified by calculating pay after taxes from a pay stub. The agent is provided with a user prompt to utilize its tool-using capabilities, generating multiple execution traces. These traces involve different combinations of tool applications, such as row lookup, column lookup, and calculations. However, integrating user prompts and external tools introduces uncertainties, often resulting in misalignment between the agent's internal confidence levels and actual success rates. The framework addresses these issues by calibrating the agent's confidence levels with actual performance, based on which we select the most suitable prompt and execution trace. The calibration model is implemented as an MLP with LLM embeddings as inputs and calibrated probabilities as outputs, and is trained with the execution result supervision. The figure is adapted from Lu et al. [27].

In this paper, we study this problem of tool-use calibration, where the goal is to calibrate a language agent's internal probability estimates to more accurately reflect the actual success rates of tool-using decisions. As illustrated in Figure 4.1, we identify two primary scenarios where such misalignment can be significantly mitigated through uncertainty calibration: *language agent prompts* and *tool execution traces*.

Our key technical contribution is a set of calibration methods called Probing Calibration (ProbCal), which draws inspiration from probing literature [4, 10], involves training a classifier using large language model embeddings to predict the expected reward for a given prompt or execution trace.

We also explore other design choices such as reweighting during training and temperature scaling [13] during inference to further refine the calibration of language agents, and provide a systematic analysis of different techniques.

We conduct experiments using various types of tool-using language agents, including both *static* and *dynamic* agents, depending on whether a fixed or dynamic set of tools is available. Our results on the MATH dataset [17] and the TabMWP dataset [28]

demonstrate that off-the-shelf language models are not well-calibrated for tool-using applications. Our ProbeCal approach significantly improves both the calibration and performance of these models consistently across different tasks. For example, applying ProbeCal results in over 7% improvement on TabMWP in dynamic tool-using settings compared with strong baselines. Furthermore, our analyses reveal that our approach can generally perform well without the need for the LLM logit or posthoc calibration, such as temperature scaling, which can translate into simpler overall frameworks for tool-using language agents.

Our contributions. We proposed a lightweight calibration framework for tool-using LLM agents via probing-based uncertainty estimation. We identified and addressed the absence of systematically integrated uncertainty modeling in LLM reasoning (e.g., self-consistency), improving tabular code-based mathematical reasoning by 7%. We showed that explicit uncertainty modeling enhances the performance and reliability of tool-augmented LLM reasoning systems.

4.2 Tool-Use Calibration for Language Agents

Tool-Using Language Agents. We define a *tool-using language model* as an agent that can interact with external environment such as program compilers via a defined set of function interfaces $\mathcal{F} = \{f_1, \ldots, f_k\}$. Each function f_i in $\mathcal{F}$ represents a callable program that the LM can utilize to perform specific tasks or computations external to the LM itself. Given a textual query q and the toolbox $\mathcal{F}$, the language model synthesizes a program $z = \text{LM}(\approx(q, \mathcal{F}))$, where $\approx(q, \mathcal{F})$ is the prompt generated from both q and $\mathcal{F}$, and fed to the language agent; LM is the language agent being able to interpret natural language queries and generate corresponding executable code. The generated program z is then executed in the given environment e by an execution engine ϕ, such that the result of the execution is given by $r = \phi(e, z)$.

Following previous work [37, 41], the programs $z \in \mathcal{Z}$ are directly represented as Python code. The execution engine ϕ is capable of handling various types of inputs x (e.g., images, videos) and producing outputs r of different types (e.g., text, image crops). In practice, multiple plausible candidate programs $\{z_1, \ldots, z_N\}$ are generated from the language agent, with each program z_i being a tool execution trace, and decoding strategies such as beam search and self-consistency [7, 40] are used to select the best program from this candidate set.

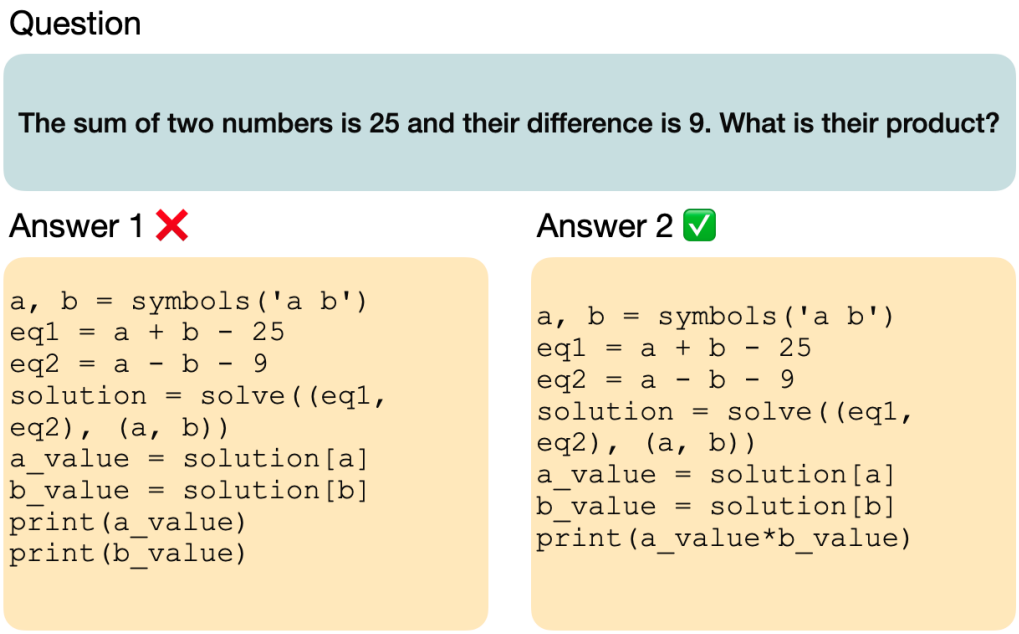


Figure 4.2: The LLM generates two different code sequences to answer the question, with the first one being incorrect and second one being correct. However, their textual formats are similar, resulting in similar uncertainty estimations with the uncalibrated LMs.

Misalignment Scenarios in Tool-Using Language Agents

The central challenge that we study is when language agents do not have a calibrated or aligned internal model of the effectiveness of tools, thereby leading to suboptimal tool-use.

Because the model is typically not trained with tools, their estimation of whether a tool-using trajectory is correct is often mis-calibrated. Consider the following question from Math Algebra [17]: "The sum of two numbers is 25 and their difference is 9. What is their product?", the LLM (`CodeLlama-7B-Instruct-hf`) generates two different code sequences to answer this question, with the first one being incorrect and second one being correct, as shown in Figure 4.2. However, their textual formats are rather similar, resulting in similar uncertainty estimations with the uncalibrated LMs. We identify two primary scenarios where uncertainty misalignment can be alleviated through calibration: *language agent prompts* and *tool execution traces*. The calibration in these scenarios can significantly impact the performance and reliability of the language agents.

Language Agent Prompts

Language models are often equipped with tool-using capabilities through the provision of carefully designed prompts. These prompts, denoted as $\approx$, include information about the tools, such as task instructions, example pairs of queries and their corresponding solutions, and comprehensive documentation detailing the tools' functionalities. The variability in the design and content of these prompts introduces uncertainty in the performance of the language models. Formally, let $\mathcal{P}$ represent the space of possible prompts. The prompt $\approx \in \mathcal{P}$ for a given query q can vary based on factors such as the specificity of the examples, the clarity of the instructions, and the comprehensiveness of the documentation. This variability can be an area of miscalibration, as different prompts may lead to different levels of tool utilization proficiency.

Prompt Design. These prompts can include information such as task instructions, example pairs of queries and solutions involving the tools, and documentation detailing the tools' functionality. In addition, the prompts can specify if the agents (1) only use *primitive functions*, which are basic, low-level operations within the task environment, such as retrieving data from a database; (2) can induce and use composite functions, such as data aggregation or filtering, integrate multiple primitive functions into a single, cohesive operation. By defining whether the use of composite functions is allowed, prompts can guide the agents to either rely on simpler, more granular steps or leverage more advanced, integrated actions that align better with the overall task objectives.

Variability in prompt design can significantly affect the agent's performance, as different prompts may result in varying levels of understanding and application of the tools. Effective prompt design should balance between providing sufficient detail to guide the agent while avoiding overwhelming it with unnecessary complexity. Clear and concise prompts help minimize misunderstandings and reduce the likelihood of errors in task execution. Therefore, prompt design is a crucial factor in managing uncertainty and enhancing the reliability of tool-using language agents.

Tool Execution Traces

Another significant area of miscalibration arises from variations in tool sequencing and the availability of multiple tools that can perform the same task. For any given task, there can be numerous plausible execution traces, each represented as a sequence of tool usages. Let $\mathcal{T}$ denote the set of all possible execution traces, where each

trace $z \in \mathcal{T}$ is a sequence of function calls from the toolbox $\mathcal{F}$. The challenge lies in selecting the optimal execution trace that leads to the correct solution.

Given a set of candidate execution traces $\{z_1, \ldots, z_N\}$ generated by the language model, decoding strategies such as beam search and self-consistency are employed to select the best trace [41]. However, the presence of multiple valid but suboptimal traces introduces uncertainty in the model's performance. The correct trace z^* that yields the desired output r is not always apparent, making it difficult for the language agent to consistently produce the correct solution.

Trace Selection. The process of selecting the optimal execution trace involves evaluating the plausibility and effectiveness of different sequences of tool usages. Each candidate trace must be assessed for its potential to achieve the desired result. This evaluation can be challenging due to the inherent complexity and variability of the tasks. The language agent must not only generate multiple potential traces but also rank and choose the most promising one. This adds a layer of complexity and potential for error, as even minor deviations in the trace can lead to incorrect outcomes.

4.3 Probing Calibration (ProbeCal)

As discussed, the performance of tool-using language agents is influenced by the variability in tool information prompts and the complexity of determining the correct tool execution traces. In this part, we introduce our proposed methods to calibrate these uncertainties and use the calibrated scores to enhance the performance of language agents. Inspired by the literature on probing, where a separate diagnostic model is trained on top of a pretrained language model's internal representations to study linguistically-informed properties [4, 23], we employ the training of a classification model that takes inputs as the embeddings from a large language model and to predict the expected reward given a prompt or execution trace.

Reward Estimation. Given a dataset containing questions and their corresponding answers, we feed models with different candidate prompts or sample multiple execution traces to collect the corresponding embeddings from a language agent and the final reward r for each prompt or trace. For example, in question-answering settings, the reward r equals 1 if the final result matches the ground truth answer and 0 otherwise. Formally, given the embedding ϕ from the language model and the corresponding reward r, reward calibration involves training a reward classification calibration model to learn the mapping $P(r|\phi)$ from ϕ to r. In our setting, we use a

three-layer MLP as the model.

We propose to use the embedding-based reward estimation to calibrate the language model, which we call Probing Calibration (ProbCal). We apply the embedding-based reward estimation to the prompt design setting, (Prompt Calibration (ProbeCal-Prompt)), the trace selection setting (Trace Calibration (ProbeCal-Trace)), and both of the settings (Prompt and Trace Calibration (ProbeCal-Prompt&Trace)) as follows.

Prompt Selection Calibration. Under our framework, we consider the problem of calibrating the probability of selecting which prompt to choose for different questions. Formally, for each question q, we want to select each prompt $\pi \in \mathcal{P}$ proportional to the estimated reward $P(\hat{r}|\phi)$ obtained from a calibration method, which we will illustrate in the next section. This is related to a classic method for decision-making under uncertainty called Boltzmann exploration, where each action is selected proportional to the exponential of the expected reward divided by a temperature parameter [6].

Execution Trace Selection Calibration. Many existing strategies rely on language agents to estimate the success probability, which can be unreliable since these tools are external to the agents. We explicitly calibrate the success probability and compute the answer by estimating the expected reward $P(r|a, z_i, q)$ using a calibration model: $\arg\max_a \mathcal{P}(a|q) = \arg\max_a \sum_{z_i} \mathcal{P}(\hat{r}|a, z_i, q)$. This approach is related to the literature on confidence-weighted majority voting [12, 30, 31], which posits that the theoretically optimal method for aggregating individual confidences is confidence-weighted majority voting, where more reliable individual responses are given greater weight.

Design Choices

Weighted Training. Reweighting has been a crucial technique in various machine learning domains, particularly for enhancing the robustness of training in the presence of significant class imbalances [3, 9, 18, 32]. This technique is well-suited to our setting, as our data samples typically include a few successful instances alongside a large number of failure cases. Specifically, for each question, we generate multiple answers from the training set, assigning a weight of 1.0 to the negative samples. The weight of the positive samples is then set as the ratio of the number of negative samples to the number of positive samples. This approach ensures that the positive samples are appropriately emphasized, addressing the imbalance and improving

overall model performance.

LLM Logits. We further study whether to incorporate the logit generated from the LLM alongside the LLM embedding. This transforms the learning problem into mapping the LLM embedding to the scaling factor necessary to adjust the LLM logit to match the ground truth reward. Previous literature has explored the extent to which the LLM-generated logit can accurately represent uncertainty [26, 29, 38]. Given token probabilities $\{p_1, ..., p_K\}$ from a token sequence of length K, we adopt two different ways to compute the sequence-level logit. The first is to compute the final logit as $\exp \sum_i \log(p_i), i = 1, ..., K$, which we denoted as Exp Sum Log (E.S.L.). As suggested by recent literature [14], we also adopt a method where $\{p_1, ..., p_K\}$ are first sorted, and the second smallest value is then used to be the final sequence logit. We denote this as Sort.

Temperature Scaling. Temperature scaling is a widely used post-processing calibration method for neural networks [13]. This technique is particularly effective for adjusting the confidence of predictions without altering the model's accuracy. Specifically, the calibration model produces logits z. The softmax function is applied to these logits, resulting in the original output probabilities $p = \text{softmax}(z)$. In temperature scaling, the logits z are divided by a temperature parameter $T > 0$ before applying the softmax function. This process modifies the distribution of the probabilities, yielding calibrated probabilities $q = \text{softmax}(z/T)$. The temperature parameter T is optimized using the negative log-likelihood (NLL) loss on the validation set. Note that while temperature scaling adjusts the confidence scores, the parameter T does not change the class with the highest probability. Therefore, the final accuracy of the model remains unaffected, but the output probabilities become better calibrated. By appropriately scaling the logits, the model's predicted probabilities become more aligned with the true likelihoods, improving the overall reliability of the predictions.

4.4 Experiments

Datasets

We evaluate our method on widely-used code-generation benchmarks, specifically MATH [17] and TabMWP [28]. The task statistics for our experiments are detailed in Table C.1 in Appendix C.1. **MATH** [17] is a benchmark comprising challenging competition mathematics problems. Each problem in MATH has a complete step-by-step solution and allows the use of code to obtain the final answer. **TabMWP**

[28] is a math reasoning dataset featuring relatively straightforward questions such as numerical calculations based on relational tables. We employ 500 questions from the training set to train the calibration model and 500 questions from the test set for evaluating the results for TabMWP.

Language Agents

We study applications of our framework to both *static* and *dynamic* tool-using agents.

Static Tool-Using Agents. We first consider static tool-using language agents that utilize a fixed set of tools. We feed agents with the Primitive and Instance prompts in [41]. The Primitive instructs agents to generate programs using primitive functions, which is the standard approach for program-aided problem solving without tool induction [8, 11]. The Instance prompt allows agents to compose high-level functions to solve tasks accordingly [33]. We perform sampling with both of the prompts to collect multiple tool execution traces.

Dynamic Tool-Using Agents. In addition to static tool-using agents, our experiments extend to dynamic language agents capable of adaptively constructing and utilizing tools tailored to specific inputs. One notable example is the TroVE framework [41]. In TroVE, three distinct prompts are employed for each problem instance, offering choices to import pre-existing tools, create new tools, or opt for a simpler approach by relying solely on primitive functions. For every problem, each prompt is sampled identically K times, and the final solution is determined through self-consistency via a majority vote from the resulting pool of $3K$ answers. TroVE can deliver solutions that are simpler compared to static tool-using language agents.

Settings

Implementation Details. We primarily use `CODELLAMA-7B-INSTRUCT-hf` in our experiments without extra notice. The embedding dimension used in our experiments is 4096. We implement the calibration model as a multi-layer perceptron (MLP) consisting of three hidden layers with 256, 256, and 80 units, respectively. The classifier is trained for 10 epochs using Adam optimizer with a learning rate 1e-4, which can be completed within minutes given the training set. For model selection, we split the training dataset into training and validation sets with a ratio of 9:1. The final model is selected based on the minimum loss achieved on the validation set. The temperature parameter is trained on the validation set when applying temperature scaling.

Table 4.1: Experiment results for static and dynamic tool-using in TabMWP and Algebra. Our approach, despite being simple, outperforms baseline methods consistently and substantially. Specifically, ProbeCal-Prompt&Trace achieves the best performance in all settings. The ECE for Instance, Instance-Sample, TroVE, and TroVE-Sample are computed between all one vectors and the label, reflecting each prompt and trace is treated equally. The ECE for ProbeCal-Prompt&Trace is computed between the average of ProbeCal-Prompt and ProbeCal-Trace logits, and the label. Acc@k refers to the accuracy with the number of samples being k.

Method	**TabMWP**				**Algebra**			
	Acc@1	Acc@5	Acc@10	ECE	Acc@1	Acc@5	Acc@10	ECE
Static Tool-Using								
Instance	32.23	48.94	51.83	0.675	18.13	26.58	29.28	0.819
Instance-Sample	35.78	50.40	52.77	0.675	18.73	27.19	29.72	0.819
Instance-Sort	32.23	49.00	52.19	0.295	18.13	27.03	29.47	0.405
Instance-E.S.L.	32.23	49.94	53.26	0.170	18.13	26.59	28.44	0.051
ProbeCal-Prompt	**42.06**	51.68	54.04	0.075	**20.27**	28.06	30.49	0.051
ProbeCal-Trace	32.23	52.66	57.02	0.046	18.13	28.29	31.18	0.048
ProbeCal-Prompt&Trace	**42.06**	**54.95**	**58.11**	**0.037**	**20.27**	**29.28**	**32.04**	**0.026**
Dynamic Tool-Using								
TroVE	22.32	38.48	44.00	0.780	15.41	28.76	32.27	0.847
TroVE-Sample	23.78	39.50	44.80	0.780	16.77	29.37	32.35	0.847
TroVE-Sort	22.32	39.08	44.18	0.380	15.41	28.62	32.09	0.404
TroVE-E.S.L.	22.32	38.77	43.23	0.163	15.41	28.17	31.46	0.097
ProbeCal-Prompt	**29.52**	42.99	47.67	0.082	**20.23**	30.25	33.12	0.055
ProbeCal-Trace	22.32	42.77	49.96	0.081	15.41	29.77	33.54	0.051
ProbeCal-Prompt&Trace	**29.52**	**46.54**	**52.15**	**0.040**	**20.23**	**31.27**	**34.05**	**0.027**

Metric. We use the task accuracy and Expected Calibration Error (ECE) [13] as the main metrics. ECE measures the calibration of a model's predicted probabilities by approximating the difference in expectation between confidence and accuracy. It is defined as $\sum_{m=1}^{M} \frac{B}{n} |\text{acc}(B_m) - \text{conf}(B_m)|$, where M is the number of bins and n is the number of samples, B_m is the set of samples whose predicted probabilities fall into the m-th bin, $\text{acc}(B_m)$ is the accuracy and $\text{conf}(B_m)$ is the average predicted confidence in m-th bin. We choose bin numbers to be 15 in the experiments.

Baselines. Our first baseline is denoted as Instance in the static setting and TroVE in the dynamic setting, where each prompt is sampled uniformly and the final trace is selected based on self-consistency without confidence-weighting [40]. We further compare with the baseline where each prompt is sampled proportional to the total number of positive answers generated by that prompt in the training set, with the final result selected using self-consistency without confidence-weighting. We denote it as Instance-Sample and TroVE-Sample in the static and dynamic

settings, respectively. We also compare with baselines where each prompt is selected uniformly, and the trace is selected based on a confidence-weighted majority vote using LLM logits, resulting in methods with suffixes E.S.L. and SORT.

Table 4.2: Experimental results for Count, Geometry, TabMWP, and Algebra for static and dynamic tool-using settings with different variants.

Method	**Count**		**Geometry**		**TabMWP**		**Algebra**	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
Static Tool-Using								
PROBECAL-TRACE	**27.88**	0.083	11.38	0.045	57.02	0.046	31.18	0.048
PROBECAL-TRACE-TS	27.81	0.090	11.37	0.046	57.10	0.050	31.18	0.056
PROBECAL-TRACE-WEIGHT	27.69	**0.079**	**11.54**	**0.042**	**57.18**	0.047	**31.49**	0.046
PROBECAL-TRACE-SORT	27.31	0.083	10.77	0.048	56.97	**0.041**	31.27	0.046
PROBECAL-TRACE-E.S.L.	25.48	0.127	9.40	0.187	53.33	0.152	30.27	**0.041**
Dynamic Tool-Using								
PROBECAL-TRACE	27.30	0.068	9.23	0.023	54.89	0.081	**35.77**	0.051
PROBECAL-TRACE-TS	27.33	0.069	9.21	0.023	**55.00**	0.086	35.75	0.057
PROBECAL-TRACE-WEIGHT	27.89	**0.060**	**9.47**	**0.016**	54.92	**0.079**	35.33	**0.050**
PROBECAL-TRACE-SORT	**28.25**	0.065	8.84	0.025	54.64	0.094	34.99	0.053
PROBECAL-TRACE-E.S.L.	26.58	0.092	6.65	0.126	53.74	0.094	34.71	0.099

Main Results

We begin by applying our embedding-based reward estimation method to prompt calibration (PROBECAL-PROMPT), trace calibration (PROBECAL-TRACE), and a combination of both prompt and trace calibration (PROBECAL-PROMPT&TRACE), and compare them against our baselines. To evaluate prompt selection calibration performance, we initially sample T answers from various prompts to create a subset of samples, with and without prompt calibration. Subsequently, we select the final answer from this pool of T samples, with (self-consistency with confidence-weighting) and without trace calibration (self-consistency without confidence-weighting). We vary T from 1 to 10 for dynamic and static tool-using cases. The final results are averaged over 5 runs. Within each run, performance is averaged across 10 instances of sampling T answers.

We present a subset of results in Table 4.1, focusing on tasks including TabMWP and Algebra in both static and dynamic tool-using contexts. Despite its simplicity, our method consistently outperforms baseline approaches in both settings. Notably, PROBECAL-PROMPT&TRACE achieves the highest performance across all datasets in all tool-using scenarios, surpassing the baseline by over 7% in the 5 and 10 sample

settings for TabMWP in the dynamic tool-using settings. While E.S.L. can produce a relatively calibrated logit, it does not always lead to performance improvement when being used in trace selection. We empirically find the average of the logits from ProbeCal-Prompt and ProbeCal-Trace produce the most calibrated logits measured with ECE, which is shown as the ECE score for ProbeCal-Prompt&Trace. Full results for each dataset are provided in Appendix C.2 and Appendix for static and dynamic settings respectively. Full results on the ECE loss are provided in Table C.9 and C.10 in Appendix C.2 and the calibration curves for ProbeCal and LLM logit are provided in Appendix C.4.

Ablation Studies

Design Choice Variants. We compare the performance of our method with different variants of design choices. As discussed in Section 4.3, we consider variants of ProbeCal including whether to apply temperature scaling, utilize weighted training, and incorporate the logit generated from the LLM alongside the LLM embedding as the input to ProbeCal, with suffixes being TS, Weight, Sort and E.S.L. respectively. We summarize a subset of results of ProbeCal-Trace variants on tasks including Count, Geometry, TabMWP, and Algebra in Table 4.2 and the full results can be found in Appendix C.2. As seen in the tables, TS does not impact the model accuracy and calibration results significantly, which is expected considering it only multiplies all the LLM logits with a single constant. Weight, on the other hand, can generally improve the model marginally and achieve the most robust performance compared to other designs. For the Sort and E.S.L. models, their performance can fluctuate, and can occasionally degrade the model performance significantly.

Effect of Temperature Scaling on ECE. We further investigate the effect of temperature scaling by investigating the ECE before and after temperature scaling on both the training and testing datasets, as shown in Table C.9 and C.10 in Appendix C.2. We observe that the ProbeCal model without temperature scaling can already produce a well-calibrated logit with a low ECE. Furthermore, we note a notable distribution shift between the training and test sets, resulting in a higher ECE loss on the test set compared to the training set. Moreover, a low train ECE loss does not always correspond to a low test ECE loss. We find that applying temperature scaling to the proposed calibration model method does not significantly improve either the ECE loss or task accuracy. This observation may stem from the calibration model already being generally well-calibrated and the distribution shift between the train and test sets.

Different Base Language Models. We also evaluate on other representative language models, including `Mistral-7B` [21], `CODELLAMA-13B-INSTRUCT-hf` and `Llama3-8b-Instruct`. Results of `Mistral-7B` in dynamic tool-using setting are summarized in Table 4.3. Further results can be found in Table C.2 and C.12 in the Appendix. We find that our proposed method can still deliver improvement when applied to a different base language model. For example, our method can improve the baseline accuracy by up to 4% on TabMWP as shown in Table 4.3, demonstrating the generality of our method to different LLM-based language agents.

Table 4.3: Results for Count and TabMWP in Dynamic Tool-Using setting using Mistral-7B. Our approach can still deliver improvement when applied to a different base language model. The full results can be found in Table C.2. Acc@k refers to the accuracy with the number of samples being k.

Method	**Count**				**TabMWP**			
	Acc@1	Acc@5	Acc@20	ECE	Acc@1	Acc@5	Acc@20	ECE
TROVE	17.46	25.70	29.27	0.826	33.02	56.60	61.92	0.663
PROBECAL-PROMPT	**20.10**	27.04	**30.02**	0.062	**48.26**	60.70	63.14	0.078
PROBECAL-TRACE	17.46	27.06	29.05	0.065	33.02	59.08	65.29	0.071
PROBECAL-PROMPT&TRACE	**20.10**	**27.66**	29.67	**0.036**	**48.26**	**62.87**	**65.94**	**0.041**

4.5 Additional Experiment Results

Effect of the Size of the Training Set

To further analyze the generalization ability of our method, we conducted experiments with different sizes of training sets ranging from 50 to 500 questions for TabMWP. The results for `CODELLAMA-13B-INSTRUCT-hf` on TabMWP in the static tool-using setting are summarized in Table C.11 in Appendix C.3. Our results show that our method PROBECAL can still deliver performance improvement even with only 50 questions from the training set.

Verbal Confidence

We further compare the proposed method with verbal confidence [38], where we instructed the LLM to output the probability between 0 to 1 to represent the uncertainty itself. We summarize the results of using `CODELLAMA-13B-INSTRUCT-hf` on TabMWP in the static tool-using setting in Table C.13 in Appendix C.3. We find that verbal confidence performs badly for the tool-using reasoning tasks we considered, and fails to bring improvement to the original baseline.

Closed-source LLMs

While we focus on the experimentes with open-source LLMs due to their transparency in this paper, we also conducted experiments with closed-source LLMs. For instance, We find that GPT-4o-mini (500 output length) results in an accuracy of 75.80%, but with a ECE of 0.5149 for E.S.L. and 0.1183 for SORT on Math Count & prob. This demonstrates that our claim "LLMs are not well-calibrated for tool-using scenarios" still holds for these models.

4.6 Related Work

Tool-Using Language Agents. Tools serve as external functional interfaces that significantly enhance and extend the task-solving capabilities of language models [42]. Recently, there has been a surge of interest in equipping language models with various tools, such as calculators, search engines, and general code programs. A notable example in this line of research is Toolformer [36], which integrates tools like Wikipedia search and machine translation systems. Toolformer utilizes GPT-J models [39] as a baseline, fine-tuning the model on self-supervised, model-synthesized examples. Another significant development is ToolLLM [34], which fine-tunes LLaMA using instructions generated from ChatGPT. Additionally, ToolkenGPT [16] represents each tool as a token, learning their embeddings and augmenting the original vocabulary, thus eliminating the need for fine-tuning. TroVE [41] prompts code language models to curate reusable high-level functions for writing solutions. However, addressing the uncertainty involved in using these tool-using agents has been rarely studied.

Uncertainty in Large Language Models. Various works have explored different methods to quantify the uncertainty within language models. These methods include the utilization of token-level logit [14, 29], multiple response generation as a proxy [26], and enabling the language model itself to express confidence verbally [38]. However, only a few of them study uncertainty calibrations for language or embodied agents[1, 35]. Among them, Han et al. [15] propose the UALA framework, which uses uncertainty as a metric to switch between the language model's own reasoning trajectory and the use of external tools, highlighting the importance of incorporating uncertainty in the design of language model agents. While in this paper we focus on the task of tool-using where the output is completely operated by external tool execution, such as code generation, it would be an interesting direction to further integrate our calibration method with reasoning frameworks that utilize uncertainty such as UALA. Additionally, research has been conducted on investigating uncertainty

calibration in few-shot learning [45], long-form generation [19], and knowledge discovery settings [5, 22]. Distinct from these works, our focus is on language agents where external tools are involved.

Probing. Probing is a growing area within machine learning interpretability [4, 10]. Similar to our approach, probing in large language models involves training a small classification model, often linear, on the internal representations of the language model. This technique is used to uncover various types of information, including linguistic properties and significant behaviors of the LLM, such as truthfulness [2, 24, 25]. Different from this line of work, our focus is to align the confidence level of language agents with their actual success rates instead of interpreting the model behaviors.

Voting Theory and Majority Voting. Our work on aggregating multiple tool execution traces with calibrated confidence scores connects to a rich literature on voting theory and collective decision-making. The classical Condorcet Jury Theorem [12] establishes that when each voter's competence exceeds random guessing, majority voting converges to the correct outcome as the number of voters grows, providing theoretical grounding for self-consistency in language model reasoning [40]. While our self-consistency setting shares this core motivation, it differs from traditional voting scenarios in several important ways. First, our "voters" are not independent human decision-makers but execution traces sampled from the same language model, which means their errors can be positively correlated through shared model weights and biases. Second, whereas voter confidence is difficult to directly estimate in traditional settings, our framework estimates it explicitly via the output of a learned calibration model. Despite these differences, our trace selection framework shares the same motivation as competence-weighted voting in the classical literature: work showed that when voters exhibit heterogeneous reliability, weighting votes by competence can substantially outperform simple majority voting [12, 30, 31], with more reliable responses receiving greater influence over the final decision. Beyond this, traditional voting theory has developed a wide variety of aggregation mechanisms for richer settings, including preference-ranking methods such as Borda-style scoring and Condorcet pairwise comparison procedures [44]. These mechanisms are less directly applicable to our setting, where each execution trace produces exactly one candidate answer together with its associated calibrated confidence score, making confidence-weighted summation a natural and well-suited aggregation rule. How best to incorporate richer voting-theoretic mechanisms into LLM reasoning and

realize their potential benefits remains an interesting direction for future work.

4.7 Conclusion

This paper addresses the inherent uncertainties in tool-using language agents, crucial for performing complex tasks. Despite advancements, challenges persist due to variability and errors from integrating external tools. Our framework recalibrates probability estimates for tool-using decisions to improve robustness and effectiveness. We focused on two primary areas of miscalibration: language agent prompts and tool execution traces. We systematically addressed these uncertainties by training a classification model using embeddings from a large language model and employing complementary methods like reweighting during training and temperature scaling during inference. Experiments on the MATH and TabMWP datasets demonstrate improvements in calibration and performance, underscoring the effectiveness of our approach. Our work fills a critical research gap, enhancing the reliability and accuracy of tool-using language agents.

References

[1] Michael Ahn et al. “Do as i can, not as i say: Grounding language in robotic affordances”. In: *arXiv preprint arXiv:2204.01691* (2022).

[2] Amos Azaria and Tom Mitchell. “The internal state of an llm knows when its lying”. In: *arXiv preprint arXiv:2304.13734* (2023).

[3] William Bankes et al. “REDUCR: Robust Data Downsampling Using Class Priority Reweighting”. In: *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*. 2023.

[4] Yonatan Belinkov. “Probing classifiers: Promises, shortcomings, and advances”. In: *Computational Linguistics* 48.1 (2022), pp. 207–219.

[5] Collin Burns et al. “Discovering latent knowledge in language models without supervision”. In: *arXiv preprint arXiv:2212.03827* (2022).

[6] Nicolò Cesa-Bianchi et al. “Boltzmann exploration done right”. In: *Advances in neural information processing systems* 30 (2017).

[7] Xinyun Chen et al. “Universal self-consistency for large language model generation”. In: *arXiv preprint arXiv:2311.17311* (2023).

[8] Zhoujun Cheng et al. “Binding Language Models in Symbolic Languages”. In: *The Eleventh International Conference on Learning Representations*. 2022.

[9] Yin Cui et al. “Class-balanced loss based on effective number of samples”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9268–9277.

[10] Yanai Elazar et al. “Amnesic probing: Behavioral explanation with amnesic counterfactuals”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 160–175.

[11] Luyu Gao et al. “Pal: Program-aided language models”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 10764–10799.

[12] Bernard Grofman, Guillermo Owen, and Scott L Feld. “Thirteen theorems in search of the truth”. In: *Theory and decision* 15.3 (1983), pp. 261–278.

[13] Chuan Guo et al. “On calibration of modern neural networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[14] Neha Gupta et al. “Language Model Cascades: Token-level uncertainty and beyond”. In: *arXiv preprint arXiv:2404.10136* (2024).

[15] Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. “Towards Uncertainty-Aware Language Agent”. In: *arXiv preprint arXiv:2401.14016* (2024).

[16] Shibo Hao et al. “Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings”. In: *Advances in neural information processing systems* 36 (2024).

[17] Dan Hendrycks et al. “Measuring mathematical problem solving with the math dataset”. In: *arXiv preprint arXiv:2103.03874* (2021).

[18] Chen Huang et al. “Learning deep representation for imbalanced classification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5375–5384.

[19] Yukun Huang et al. “Calibrating Long-form Generations from Large Language Models”. In: *arXiv preprint arXiv:2402.06544* (2024).

[20] Drew A Hudson and Christopher D Manning. “Gqa: A new dataset for real-world visual reasoning and compositional question answering”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6700–6709.

[21] Albert Q Jiang et al. “Mistral 7B”. In: *arXiv preprint arXiv:2310.06825* (2023).

[22] Zhengbao Jiang et al. “How can we know when language models know? on the calibration of language models for question answering”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 962–977.

[23] Karim Lasri et al. “Probing for the usage of grammatical number”. In: *arXiv preprint arXiv:2204.08831* (2022).

[24] Kenneth Li et al. “Inference-time intervention: Eliciting truthful answers from a language model”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[25] Kenneth Li et al. “Q-Probe: A Lightweight Approach to Reward Maximization for Language Models”. In: *arXiv preprint arXiv:2402.14688* (2024).

[26] Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. “Generating with confidence: Uncertainty quantification for black-box large language models”. In: *arXiv preprint arXiv:2305.19187* (2023).

[27] Pan Lu et al. “Chameleon: Plug-and-play compositional reasoning with large language models”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[28] Pan Lu et al. “Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning”. In: *arXiv preprint arXiv:2209.14610* (2022).

[29] Andrey Malinin and Mark Gales. “Uncertainty estimation in autoregressive structured prediction”. In: *arXiv preprint arXiv:2002.07650* (2020).

[30] Sascha Meyen et al. “Group decisions based on confidence weighted majority voting”. In: *Cognitive research: principles and implications* 6 (2021), pp. 1–13.

[31] Shmuel Nitzan and Jacob Paroush. “Optimal decision rules in uncertain dichotomous choice situations”. In: *International Economic Review* (1982), pp. 289–297.

[32] K Philip and SJS Chan. “Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection”. In: *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*. AAAI Press Manchester, UK. 1998, pp. 164–168.

[33] Cheng Qian et al. “Creator: Disentangling abstract and concrete reasonings of large language models through tool creation”. In: *arXiv preprint arXiv:2305.14318* (2023).

[34] Yujia Qin et al. “Toolllm: Facilitating large language models to master 16000+ real-world apis”. In: *arXiv preprint arXiv:2307.16789* (2023).

[35] Allen Z Ren et al. “Robots that ask for help: Uncertainty alignment for large language model planners”. In: *arXiv preprint arXiv:2307.01928* (2023).

[36] Timo Schick et al. “Toolformer: Language models can teach themselves to use tools”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[37] Dídac Surís, Sachit Menon, and Carl Vondrick. “Vipergpt: Visual inference via python execution for reasoning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 11888–11898.

[38] Katherine Tian et al. “Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback”. In: *arXiv preprint arXiv:2305.14975* (2023).

[39] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 billion parameter autoregressive language model*. 2021.

[40] Xuezhi Wang et al. “Self-consistency improves chain of thought reasoning in language models”. In: *arXiv preprint arXiv:2203.11171* (2022).

[41] Zhiruo Wang, Daniel Fried, and Graham Neubig. “TroVE: Inducing Verifiable and Efficient Toolboxes for Solving Programmatic Tasks”. In: *arXiv preprint arXiv:2401.12869* (2024).

[42] Zhiruo Wang et al. “What Are Tools Anyway? A Survey from the Language Model Perspective”. In: *arXiv preprint arXiv:2403.15452* (2024).

[43] Zhilin Yang et al. “HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 2369–2380.

[44] H Peyton Young. “Condorcet’s theory of voting”. In: *American Political science review* 82.4 (1988), pp. 1231–1244.

[45] Zihao Zhao et al. “Calibrate before use: Improving few-shot performance of language models”. In: *International conference on machine learning*. PMLR. 2021, pp. 12697–12706.

Chapter 5

PROGRAMMATIC CONTEXT AUGMENTATION FOR LLM-BASED SYMBOLIC REGRESSION

This chapter is based on the paper:
Hao Liu*, Xiao-Wen Yang*, Atharva Sehgal, Yixin Wang, Lan-Zhe Guo, Yu-Feng Li, Yisong Yue. "Programmatic Context Augmentation for LLM-based Symbolic Regression." In: *arXiv preprint arXiv:2605.03101 (2026)*.

5.1 Introduction

Symbolic regression (SR) [10, 16] is a central task across many scientific domains. Given a dataset collected from observations or experiments, the goal of SR is to discover concise and interpretable mathematical equations that can best explain the underlying structure of the data. Unlike purely predictive models, SR offers human-readable equations that provide interoperability and scientific insight, making it a powerful tool for discovery.

The key challenges of SR lie in efficiently searching the vast combinatorial space of candidate equations. Traditional approaches focus on using genetic programming [10], which iteratively generate populations of equations, evaluate them using fitness scores derived from the data, and refine them through mutation and crossover. Other approaches include neural-guided search [3, 22] and reinforcement learning [19]. While these methods have shown success in small-scale settings, they remain limited in efficiency and scalability, largely due to their reliance on random mutations and their inability to leverage prior experience. To improve upon these limitations, neural network–based SR methods trained on datasets of equation–data pairs have been proposed [9].

More recently, large language models (LLMs) have been introduced into SR through evolutionary search frameworks [23]. At each iteration, a textual prompt that combines the problem description with insights from past iterations is provided to the LLM, which then generates new candidate equations. This paradigm leverages the domain knowledge encoded in pretrained LLMs and has demonstrated promising results. Methods such as concept-library learning [6] have further enhanced LLMs' ability to reason symbolically in SR.

Despite these advances, current LLM-based SR methods remain constrained by their feedback mechanisms. Specifically, they rely almost exclusively on scalar evaluation metrics—most commonly mean squared error (MSE)—as the sole interaction with the dataset during the search process. That is, while candidate equations are evaluated against the dataset, the only signal returned to the model is a single numerical score. This stands in stark contrast to how human scientists approach the same task: rather than relying only on scalar evaluations, they actively conduct exploratory data analysis, examining relationships, distributions, and trends to guide the design of hypothesis equations. In this work, we bridge this gap by introducing **PROgrammatic context AUGmentation** (ProAug). Our framework empowers the LLM not only to propose candidate equations but also to generate code for analyzing the underlying dataset. By executing such code, the model gains access to richer signals—such as statistical properties and variable correlations—that can be used to refine its evolutionary search.

Concretely, we propose a new LLM-based SR framework in which the model is tasked with two complementary objectives: (1) proposing potential equation candidates, and (2) generating data-analysis code that extracts informative signals from the dataset. To rigorously assess our method, we benchmark it on LLM-SRBench [24], a comprehensive and recently proposed evaluation suite that mitigates concerns about LLMs leveraging memorized knowledge. Across both zero-shot and supervised fine-tuning settings, our method consistently outperforms strong baselines in terms of both efficiency and accuracy.

Our contributions. We introduce **programmatic context augmentation**, a novel mechanism that enables LLMs to actively interact with the dataset during evolutionary SR. We design a dual-task framework in which the LLM generates both candidate equations and code for data-driven analysis. We evaluate our approach on LLM-SRBench and demonstrate consistent improvements over state-of-the-art baselines across multiple settings. For instance, we see a 3x error reduction on LSR-Transform when using programmatic context augmention with DeepSeek-V3.1 as the base model.

5.2 Preliminaries

Symbolic Regression. Using the framework of Empirical Risk Minimization (ERM), the problem of symbolic regression can be formalized as follows [10, 26]. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the input features and $y_i \in \mathbb{R}$

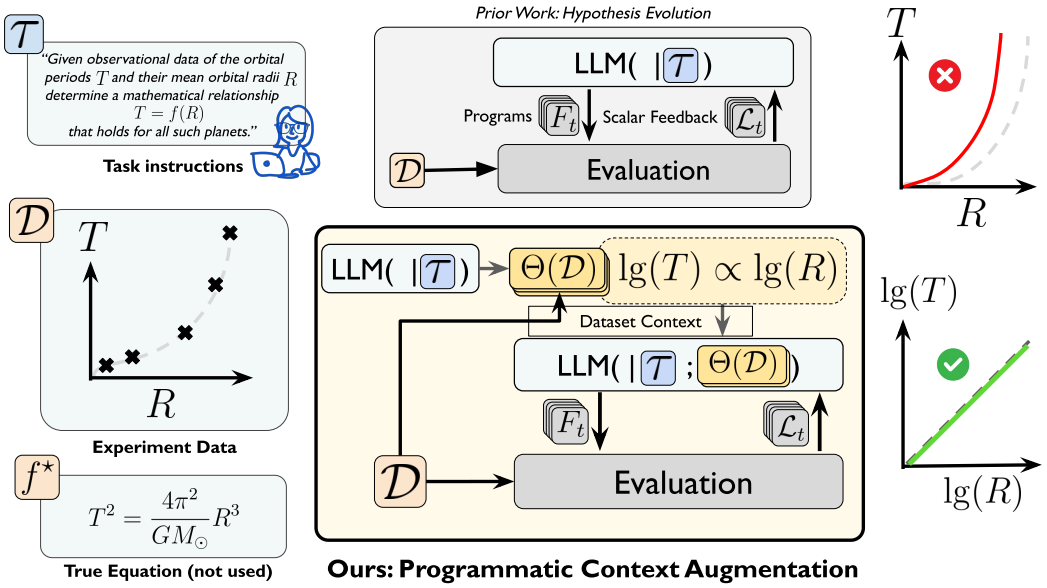


Figure 5.1: An overview of ProAug, with τ being the task instructions, D being the dataset, Θ being the dataset context, F_t being the generated program and L_t being the scalar feedback. Prior work in LLM guided symbolic regression [23] (in gray) rely on simplistic scalar evaluation metrics, such as MSE as feedback, ignoring rich information contained in dataset. Intead, ProAug enriches the model's interaction with data by assigning the LLM a dual role-to generate data analysis code and extract informative signals for context augmentation. Considering the classical problem of deriving Kepler's third law from planetary motion data as a example. The law states that the square of a planet's orbital period (T) is proportional to the cube of its semi-major axis (R), i.e., $T^2 \propto R^3$. While standard method might struggle to identify this nonlinear relationship directly, ProAug leverages the LLM's ability to reason by data analysis. Specifically, the LLM generates code to apply a logarithmic transformation to both T and R, yielding $\log(T)$ and $\log(R)$. When plotted, these transformed variables exhibit a clear linear relationship: $\log(T) \approx \frac{3}{2}\log(R) + \text{constant}$. This linear trend immediately suggests a power-law relationship between the original variables. Our method demonstrates how data-driven statistics complement structural constraints in equation discovery.

the scalar target, the objective of SR is to find a function $f^* \in \mathcal{F}$ that minimizes the empirical loss: $f^* = \arg\min_{f \in \mathcal{F}} \mathcal{L}(f)$, where $\mathcal{F}$ is the discrete hypothesis class of functions mapping $\mathbb{R}^d$ to $\mathbb{R}$, and $\mathcal{L}$ is a loss function measuring predictive accuracy. A common choice is the mean squared error (MSE): $\mathcal{L}(f) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2$.

Since the hypothesis class $\mathcal{F}$ is constrained by Python code, one can also view this problem as an instance of program synthesis [1].

LLM evolutionary search for SR. Recent work has integrated large language models (LLMs) into the framework of evolutionary algorithms, yielding LLM agents that iteratively propose solutions to tasks such as coding or scientific discovery. We focus on their application to SR, as exemplified by LLM-SR [23], which serves as a main baseline in this work. The framework consists of three key components:

(i) Hypothesis generation. At each iteration, the LLM is prompted with a structured input that includes: task instructions, problem specifications (e.g., the physical meaning of variables), the optmization and evaluation function, and demonstrations of promising prior hypotheses and their improvement trajectories from an experience buffer. Conditioned on this prompt, the LLM generates a candidate equation in the form of a Python function skeleton with learnable parameters.

(ii) Hypothesis optimization and assessment. The free parameters of the generated equation skeletons are optimized using external solvers (e.g., NumPy with BFGS). The resulting hypothesis is then evaluated on the dataset to produce a fitness score. In LLM-SR, this score is defined as the negative MSE on the training set. However, we observed that this definition can lead to generalization mismatches between training and test performance. To mitigate this, we split the training set into a *train-train* (tr-tr) and *train-val* (tr-val) subset. Parameters are optimized on the train-train set, and the fitness score is computed on the train-val set using the negative normalized MSE.

(iii) Experience management. Inspired by mechanisms in evolutionary algorithms (e.g., island models [2, 21]), LLM-SR maintains an experience buffer of past hypotheses. Candidate hypotheses are sampled from this buffer according to a distribution weighted by fitness scores, and incorporated into future prompts as demonstrations. This balances diversity and efficiency during the evolutionary search.

5.3 Overall Framework

Although LLM-based methods have demonstrated effectiveness in symbolic regression tasks, they still exhibit key limitations. As discussed earlier, existing approaches primarily rely on scalar evaluation metrics derived from raw data samples, overlooking the rich statistical characteristics of the dataset that can be extracted through program. These characteristics can provide valuable inductive biases, helping to constrain the search space and guide the discovery of correct symbolic expressions. This naturally raises the question: *Can statistical information enhance LLM-based SR?* In this section, we address this question through controlled analysis and then introduce our proposed framework, ProAug.

Can statistical information enhance LLM-based SR?

Analysis Setup. To systematically examine the role of statistical information in LLM-based SR, we conduct a controlled case study on a representative instance, `II.6.15b_1_0`, from the LLM-SRBench benchmark. The ground-truth expression for this case is: $A_{vec} = \frac{j \cdot m}{q \cdot \rho_{c_0}}$, which corresponds to a physical relationship involving the vector potential, current density, charge carrier density, elementary charge, and mass of a charge carrier. To eliminate confounding effects from prior knowledge, we anonymize all variables and remove the original physical background in the experimental conditions here. Specifically, inputs and outputs are replaced with x_i and y respectively. We then compare the following three settings:

- **Zero-shot**: The model receives no data samples or additional information.
- **Few-shot**: The model is provided with random 12 raw data samples without any additional background or statistical information.
- **Statistical Hint**: In addition to the 12 raw samples, the model is provided with a structured dictionary that summarizes dataset-level statistics. This dictionary includes (i) basic statistical measures for each variable and the output—such as mean, standard deviation, minimum, and maximum, and (ii) feature-level correlations, such as R^2 values between the logarithm of the output and simple transformations of the inputs (e.g., log, exp, sin, cos).

Prompts used here for all settings are provided in Appendix D.3. We use DeepSeek-V3.1 [12] as the base model within the LLM-SR framework, conducting a total of 100 evolutionary steps for each experiment.

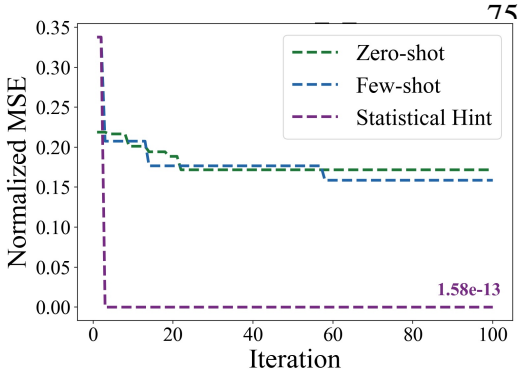


Figure 5.2: NMSE trajectories under three settings with varying amounts of background information.

Results. Figure 5.2 shows the best normalized mean squared error (NMSE) score trajectory under all settings. Both the zero-shot and few-shot settings perform poorly. The 12 randomly sampled data points provided in the few-shot condition are insufficient for the model to infer the true underlying expression. As a result, the NMSE decreases only slowly over the 100 evolutionary iterations. In contrast, the inclusion of statistical hints significantly accelerates convergence, ultimately yielding an NMSE on the order of 10^{-13}. To understand the reason for this improvement, we look into the model's behavior under the statistical hint setting. We find that the high R^2 value between $\log y$ and $\log x$ provided leads the model to infer that y is likely proportional to certain powers of the four input variables—suggesting a functional form of $y = x_0^a \cdot x_1^b \cdot x_2^c \cdot x_3^d$, where a, b, c, d are parameters can be optimized. This inferred form aligns perfectly with the ground-truth expression. These results demonstrate that even basic statistical information can provide substantial guidance in the LLM-SR process for the specific case. By constraining the search space and suggesting plausible functional forms, statistical cues can improve both the efficiency and the accuracy of symbolic regression. This mirrors how human scientists often practice exploratory data analysis before formulating hypotheses in the process of scientific discovery.

Limitations. Despite its utility, the design of the statistical context here has certain limitations. The correlation metrics are manually designed and confined to a small predefined set of transformations (e.g., exp, sin, cos). Consequently, complex nonlinear or multivariate interactions may remain undetected. While effective for highlighting simple algebraic relationships, this approach may fail with more intricate structures.

Programmatic context augmentation (ProAug)

To fully leverage statistical context, it is essential to move beyond manually engineered heuristics. In the next section, we introduce **Programmatic Context Augmentation (ProAug)**, in which LLMs—guided by physical knowledge—automatically generate dataset-analysis code. This enables the model to extract richer and more adaptive

Algorithm 3: Pseudocode for ProAug.

Input: Pretrained LLM $\mathcal{M}$, dataset $\mathcal{D}$, number of iterations T
Output: Best symbolic expression $f^\star$
for $t = 0, \ldots, T-1$ **do**
    `// Programmatic context augmentation`
    Generate dataset-analysis prompt and sample analysis code
    Execute code on $\mathcal{D}$ to obtain informative signals
    `// Context construction`
    Construct enriched prompt using task instruction, problem specification, experience demonstrations, and extracted signals
    `// Hypothesis generation`
    Sample candidate equation from $\mathcal{M}$ given enriched prompt
    `// Optimization and evaluation`
    Optimize free parameters on $\mathcal{D}_{\text{tr-tr}}$
    Evaluate fitness score on $\mathcal{D}_{\text{tr-val}}$
    `// Experience management`
    Update population buffer with candidate and score
    Update best solution $f^\star$
end

signals, preserving the benefits of inductive bias while broadening coverage.

The goal of ProAug is to empower the LLM to actively interact with the dataset during evolutionary search by first proposing *data analysis programs*. These programs are executed to extract informative signals from the dataset, which are then incorporated into the prompt context. The enriched context prompt is subsequently used by the LLM to generate candidate symbolic expressions. In contrast, the original LLMSR pipeline (see Section 5.2) bypasses this step: it directly prompts the LLM to generate hypotheses based only on task instructions, problem specifications, and past experience, without any dataset-level analysis.

Formally, ProAug extends the LLMSR pipeline by introducing an additional *dataset analysis phase* before hypothesis generation. The new pipeline proceeds as follows: 1. In the *data analysis phase*, The LLM is instructed with a structured prompt to generate Python code that extracts statistical or structural information from the dataset. 2. The generated code is executed, and the extracted signals (e.g., summary statistics, correlations, transformation relationships) are fed back as part of an enriched context prompt. 3. Using this enriched prompt, the LLM generates candidate equations within an evolutionary search framework similar as LLMSR.

The structured prompt in the *data analysis phase* includes both a task description and a lightweight code template of basic analysis tools, which includes computations

such as (i) descriptive statistics of each variable (mean, variance, range), (ii) linear correlations, and R^2 tests between transformed features (e.g., $\log, \exp, \sin, \cos$) or feature combinations and the target variable or the transformation of it. A brief overview of ProAug is shown in Algorithm 3, along with a pipeline illustration and an example of a ProAug prompt, is provided in Figure 5.1 and Appendix D.3.

5.4 Experiments

Experimental Setup

Datasets Benchmarking LLMs in symbolic regression is challenging due to concerns that pretrained models may exploit memorized knowledge rather than demonstrating genuine reasoning and search capabilities. We evaluate ProAug on LLM-SRBench [24], a recently proposed benchmark designed to mitigate this issue. LLM-SRBench consists of two primary subsets: **LSR-Transform** and **LSR-Synth**.

LSR-Transform includes 111 target functions adapted from Feynman equations [25], further modified through variable substitutions and symbolic rewrites to increase structural complexity. Due to computational constraints, we randomly sample 15% of the tasks (17 instances) for evaluation. LSR-Synth consists of symbolic regression tasks that integrate canonical scientific terms with synthetically constructed expressions, spanning four domains—chemistry, physics, biology, and materials science.From this subset, we randomly select 16 tasks for testing. The complete list of tasks used in our experiments is provided in the Appendix D.2.

Baselines We adopt LLM-SR [23] as our primary baseline and evaluate three backbone models: `Qwen3-4B-Instruct-2507` [27], `Qwen3-8B` [27], and `DeepSeek-V3.1` [12], covering a range of model scales. In addition to LLM-SR, we also compare against a hand-crafted statistical hint baseline, which incorporates both the problem specifications (physical meaning of variables) as well as the statistical hint. All methods are evolved for 150 iterations per task, with 2 samples generated per prompt. To reduce the effect of randomness, each experiments is repeated three times with different random seed, and we report the mean results.

Evaluation Metrics Follow prior work, we primilary evaluate performance using the normalized mean squared error (NMSE). NMSE quantifies prediction error normalized by target variance: $\text{NMSE} = \frac{\sum_{i=1}^{N_{\text{test}}}(f(x_i)-y_i)^2}{\sum_{i=1}^{N_{\text{test}}}(y_i-\bar{y})^2}$, where $\bar{y}$ denotes the mean of the true values. By emphasizing large deviations, NMSE is well-suited for assessing numeric precision in symbolic regression.

Table 5.1: Comparison of different methods on LLM-SRBench, evaluated using normalized mean squared error (NMSE). The benchmark subsets contain 17 tasks from LSR-Transform, and 5, 3, 5, and 3 tasks from Chemistry, Biology, Physics, and Materials Science, respectively. Each entry reports the average NMSE across all tasks within the corresponding subset. The final column (Average) reports the mean NMSE over all LSR-Synth tasks (16 instances in total). Bold values indicate the best performance for each model, while underlined values denote the best overall performance across models.

Models	LSR-Transform	LSR-Syn				
		Chemistry	Biology	Physics	Material Science	Average
`Qwen3-4B-Instruct-2507` [27]						
LLM-SR	0.258	**7.55e-6**	2.48	**0.048**	**1.60e-6**	0.480
Statistical Hint	0.225	0.032	0.095	0.22	0.012	0.099
ProAug	**0.145**	4.73e-3	**0.036**	0.060	9.62e-6	**0.027**
`Qwen3-8B` [27]						
LLM-SR	0.199	0.012	0.041	0.077	2.05e-4	0.036
Statistical Hint	0.279	4.11e-3	0.019	**0.019**	0.034	0.017
ProAug	**0.148**	**3.10e-3**	**1.08e-3**	0.021	**1.67e-7**	**7.75e-3**
`Deepseek-V3.1` [12]						
LLM-SR	0.230	0.080	**0.060**	0.110	8.47e-7	0.071
Statistical Hint	0.188	0.022	1.112	**0.034**	0.333	0.288
ProAug	**0.067**	**1.02e-05**	0.094	0.058	**1.82e-9**	**0.036**

Results

As shown in Table 5.1, our method achieves lower NMSE on most datasets, consistently outperforming the LLM-SR baseline. For instance, in the LSR-Transform subset with DeepSeek-V3.1 as the backbone, our method attains an NMSE of 0.067 compared to 0.23 for LLM-SR, which is a 3x reduction in error. Moreover, we observe that the performance advantage of our method over LLM-SR becomes more pronounced as the capability of the underlying model improves. In particular, with DeepSeek models, our approach yields significant improvements across all tasks. We attribute this trend to the nature of programmatic context augmentation, which depends on the model's semantic understanding: weaker models are more likely to be overwhelmed by the extended context, while stronger models can effectively leverage it. Therefore, as base models continue to grow stronger in semantic understanding and context management, we expect our method to become even more effective in the near future.

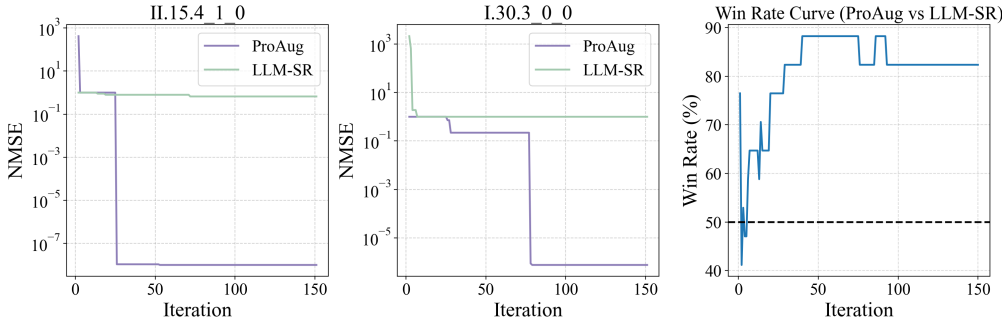


Figure 5.3: Analysis of convergence efficiency on LSR-Transform for DeepSeek-V3.1.

Table 5.2: SFT results using NMSE on LSR-Transform. best performance for each model in bold.

LSR-Transform	`Qwen3-4B-Instruct-2507`	`Qwen3-8B`
LLM-SR	0.258	0.199
LLM-SR-SFT	0.110	0.131
ProAug	0.145	0.148
ProAug-SFT	**0.107**	**0.106**

Finetuning

Setting Furthermore, we conduct supervised fine-tuning (SFT) for the Qwen3-4B-Instruct-2507 and Qwen3-8B models using samples drawn from the remaining portion of LSR-Transform. The full training set consists of 77 problems, and we employ DeepSeek-V3.1 for data distillation. The distillation data generation procedure is divided into two stages. In the first stage, the model is provided with the ground-truth expression and prompted to generate a data analysis program based on background knowledge and the correct answer. The generated program is then executed, and if execution fails, the generation process is retried up to ten attempts. In the second stage, we concatenate the results of the data analysis and prompt the model to produce the final expression together with its reasoning steps. The SFT model was trained on two NVIDIA A800 GPUs with a batch size of 16, a learning rate of 5e-6, and ZeRO Stage 2 for parallel acceleration.

We observe that the model occasionally generates non-executable code. To mitigate this issue, we augment the training data with additional examples where the data analysis code fails, but the model is still required to generate the correct expression. For comparison, we also distill outputs from DeepSeek-V3.1 for the LLM-SR method as an additional baseline. Finally, we omit SFT on LSR-Synth, as in this dataset subset, distinct problems share identical input prompts, leading to a one-to-many mapping issue that makes SFT ill-posed and counterproductive.

Results Table 5.2 reports the SFT results on LSR-Transform. We observe that supervised fine-tuning consistently improves performance for both LLM-SR and ProAug, indicating that even standard fine-tuning provides clear benefits in this setting. For example, with Qwen3-4B-Instruct-2507, NMSE decreases from 0.258 to 0.110 for LLM-SR after SFT. Similarly, ProAug also benefits from fine-tuning, with NMSE reduced from 0.145 to 0.107. Importantly, ProAug-SFT achieves the lowest errors overall across both backbones, highlighting that our method remains effective and complementary to supervised fine-tuning.

Method Behavior Analysis

Convergence Efficiency

We evaluated the convergence efficiency of ProAug relative to LLM-SR on the LSR-Transform subset from the DeepSeek-V3.1 model, with results shown in Figure 5.3. The two plots on the left provide representative examples, illustrating that our method achieves a more rapid reduction in NMSE on train-val set during the iterative search process and ultimately converges to a lower error. The plot on the right displays the win rate of ProAug over LLM-SR across 17 LSR-Transform test cases as a function of the number of iterations. We observe that ProAug starts with a high win rate (above 50%) early in the process, indicating fast convergence of scores in the initial stages. As the iterations progress, the win rate continues to increase, eventually exceeding 80%. These results demonstrate that ProAug maintains superior performance while also converges efficiently.

Variance Analysis

To rigorously evaluate the stability and reproducibility of LLM-SR and ProAug, we conducted multiple independent runs under controlled hyperparameters and random seeds. Figure 5.4 presents boxplots summarizing the distribution of results, including the median and interquartile range (IQR) for each method.

Overall, ProAug exhibits substantially lower variance across runs. LLM-SR shows high sensitivity to randomness, with outcomes occasionally differing by several orders of magnitude. For example, on the Chemistry task with Qwen3-8B, the NMSE varied from 10^{-4} to 10^{-9} across runs. Such findings highlight a critical gap in current LLM-based SR research, where multi-run evaluations are rarely reported. We therefore recommend that future studies adopt repeated trials to mitigate randomness and enhance reliability.

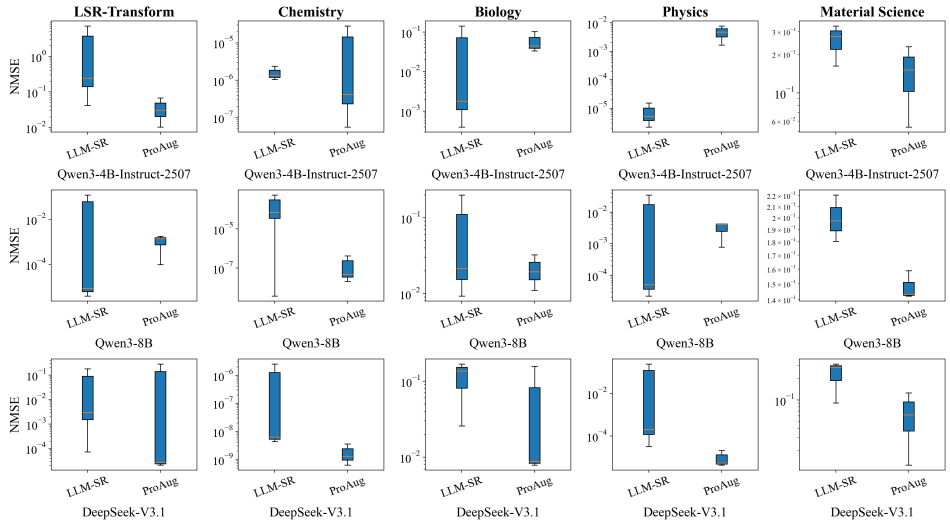


Figure 5.4: Boxplot comparison of tri-run variance for LLM-SR vs. ProAug.

5.5 Related Work

Symbolic Regression. Symbolic regression (SR) is core to scientific discovery. Interest began in the 1970s [4, 11] and SR has recently become prominent in AI-for-science [16, 17, 21]. SR methods follow three algorithmic themes: search-based, learning-based, and LLM-based approaches (we discuss search-based and learning-based methods in Appendix D.1).

LLM based methods. Recent works leverage large language models for symbolic regression [6, 14, 18, 21, 23]. These methods rely on LLMs' world knowledge to propose and mutate programs conditioned on natural language instructions and execution feedback. Specifically, Funsearch [21] and AlphaEvolve [18] demonstrate that LLMs as mutation operators within evolutionary algorithms can discover novel heuristics for long-standing problems in mathematics, hardware design, and algorithms. Many exciting works augment the evolutionary process with sketching [23], library learning [6], and tool use [14]. ProAug leverages pretrained LLMs similarly but treats the scientific dataset as a first-class citizen in the optimization process rather than merely a scoring mechanism. Consecutively, ProAug's primary contribution is *complementary* to that of these methods, and it is technically possible to employ programmatic context augmentation to enhance other SR algorithms.

Scientific Discovery with Foundation Models. Modern scientific breakthroughs increasingly demand both mastery of individual fields and trans-disciplinary insights [5, 8]. As such, many LLM frameworks have emerged that assist scientists in various

stages of the scientific process – from hypothesis falsification [7] and idea generation [20] to data-driven discovery [15], heuristic design [18, 21], and multi-purpose systems spanning hypothesis generation to experiment design [5, 13]. While these systems primarily interface with knowledge sources and scientists through natural language, ProAug (like [18, 21]) interfaces with these sources through code – leveraging the precise semantics and deterministic execution of code to rigorously analyze scientific data, in addition to using textual reasoning.

5.6 Conclusion

In this work, we address a key limitation of existing LLM-based symbolic regression methods: their reliance on simplistic scalar feedback such as mean squared error. We propose ProAug, a novel framework that enriches the model's interaction with data by assigning the LLM a dual role—generating both candidate equations and data-analysis code. This design fosters a more active, human-like discovery process that leverages rich statistical signals and variable relationships. Experimental results on the challenging LLM-SRBench benchmark highlight the clear advantages of our approach. We believe this work represents a step toward more intelligent and interactive AI systems with stronger capabilities for scientific reasoning and discovery. The current framework, while leveraging richer statistical feedback than MSE-based methods, still relies on a predefined set of data analysis workflows. This may restrict its ability to discover equations that depend on highly complex or non-standard data relationships not captured by our initial code-generation template. Future work could explore enabling the LLM to propose entirely novel analytical procedures.

References

[1] Swarat Chaudhuri et al. "Neurosymbolic programming". In: *Foundations and Trends® in Programming Languages* 7.3 (2021), pp. 158–243.

[2] Miles Cranmer. "Interpretable machine learning for science with PySR and SymbolicRegression. jl". In: *arXiv preprint arXiv:2305.01582* (2023).

[3] Miles Cranmer et al. "Discovering symbolic models from deep learning with inductive biases". In: *Advances in neural information processing systems* 33 (2020), pp. 17429–17442.

[4] Donald Gerwin. "Information processing, data inferences, and scientific generalization". In: *Behavioral Science* 19.5 (1974), pp. 314–325.

[5] Juraj Gottweis et al. *Towards an AI co-scientist*. 2025. arXiv: 2502.18864 [cs.AI]. URL: https://arxiv.org/abs/2502.18864.

[6] Arya Grayeli et al. “Symbolic regression with a learned concept library”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 44678–44709.

[7] Kexin Huang et al. *Automated Hypothesis Validation with Agentic Sequential Falsifications*. 2025. arXiv: 2502.09858 [cs.LG]. URL: https://arxiv.org/abs/2502.09858.

[8] Martin Jinek et al. “A programmable dual-RNA–guided DNA endonuclease in adaptive bacterial immunity”. In: *science* 337.6096 (2012), pp. 816–821.

[9] Pierre-Alexandre Kamienny et al. “End-to-end symbolic regression with transformers”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 10269–10281.

[10] Gabriel Kronberger et al. *Symbolic regression*. Chapman and Hall/CRC, 2024.

[11] Pat Langley. “BACON: A Production System That Discovers Empirical Laws”. In: *International Joint Conference on Artificial Intelligence*. 1977. URL: https://api.semanticscholar.org/CorpusID:2320342.

[12] Aixin Liu et al. “Deepseek-v3 technical report”. In: *arXiv preprint arXiv:2412.19437* (2024).

[13] Chris Lu et al. *The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery*. 2024. arXiv: 2408.06292 [cs.AI]. URL: https://arxiv.org/abs/2408.06292.

[14] Pingchuan Ma et al. *LLM and Simulation as Bilevel Optimizers: A New Paradigm to Advance Physical Scientific Discovery*. 2024. arXiv: 2405.09783 [cs.LG]. URL: https://arxiv.org/abs/2405.09783.

[15] Bodhisattwa Prasad Majumder et al. *DiscoveryBench: Towards Data-Driven Discovery with Large Language Models*. 2024. arXiv: 2407.01725 [cs.CL]. URL: https://arxiv.org/abs/2407.01725.

[16] Nour Makke and Sanjay Chawla. “Interpretable scientific discovery with symbolic regression: a review”. In: *Artificial Intelligence Review* 57.1 (2024), p. 2.

[17] Matteo Merler, Nicola Dainese, and Katsiaryna Haitsiukevich. “In-Context Symbolic Regression: Leveraging Language Models for Function Discovery”. In: *arXiv preprint arXiv:2404.19094* (2024).

[18] Alexander Novikov et al. “AlphaEvolve: A coding agent for scientific and algorithmic discovery”. In: *arXiv preprint arXiv:2506.13131* (2025).

[19] Brenden K Petersen et al. “Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients”. In: *arXiv preprint arXiv:1912.04871* (2019).

[20] Marissa Radensky et al. *Scideator: Human-LLM Scientific Idea Generation Grounded in Research-Paper Facet Recombination*. 2025. arXiv: 2409.14634 [cs.HC]. URL: https://arxiv.org/abs/2409.14634.

[21] Bernardino Romera-Paredes et al. "Mathematical discoveries from program search with large language models". In: *Nature* 625.7995 (2024), pp. 468–475.

[22] Ameesh Shah et al. "Learning differentiable programs with admissible neural heuristics". In: *Advances in neural information processing systems* 33 (2020), pp. 4940–4952.

[23] Parshin Shojaee et al. "Llm-sr: Scientific equation discovery via programming with large language models". In: *arXiv preprint arXiv:2404.18400* (2024).

[24] Parshin Shojaee et al. "Llm-srbench: A new benchmark for scientific equation discovery with large language models". In: *arXiv preprint arXiv:2504.10415* (2025).

[25] Silviu-Marian Udrescu and Max Tegmark. "AI Feynman: A physics-inspired method for symbolic regression". In: *Science advances* 6.16 (2020), eaay2631.

[26] Vladimir Vapnik. "Principles of risk minimization for learning theory". In: *Advances in neural information processing systems* 4 (1991).

[27] An Yang et al. "Qwen3 technical report". In: *arXiv preprint arXiv:2505.09388* (2025).

Appendix A

APPENDIX TO CHAPTER 2

A.1 Description of More Baseline Methods

DM. Given reward predictor $\hat{r}$, direct method estimate $\hat{V}_{\mathrm{DM}}$ as:

$$\hat{V}_{\mathrm{DM}} = \frac{1}{|S|} \sum_{\boldsymbol{x} \in S} \mathbb{E}_{a \sim \pi} \left[\hat{r}(\boldsymbol{x}, a) \right] . \tag{A.1}$$

IPS. Inverse propensity scoring (IPS) uses important weighting on the entries in S to reflect the relative probabilities of choosing some action a by the target policy π versus the logging policy β:

$$\hat{V}_{\mathrm{IPS}} = \frac{1}{|S|} \sum_{(\boldsymbol{x},a,r) \in S} \frac{\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})} r, \tag{A.2}$$

where $\hat{\beta}$ is the logging policy (estimated if β is unknown).

DR. The basic formulation of the doubly robust estimator is:

$$\hat{V}_{\mathrm{DR}} = \hat{V}_{\mathrm{DM}} + \frac{1}{|S|} \sum_{(\boldsymbol{x},a,r) \in S} \left[\frac{(r - \hat{r}(\boldsymbol{x}, a))\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})} \right]. \tag{A.3}$$

SnIPS. The self-normalized IPS [8] is normalized by the sum of the importance weights. It follows the form as below:

$$\hat{V}_{\mathrm{SnIPS}} = \frac{\sum_{(\boldsymbol{x},a,r) \in S} r \frac{\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})}}{\sum_{(\boldsymbol{x},a,r) \in S} \frac{\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})}}. \tag{A.4}$$

SnDR. The self-normalized doubly robust estimator normalizes the importance weights following the idea of SnIPS. The formulation of the SnDR is:

$$\hat{V}_{\mathrm{SnDR}} = \frac{\sum_{(\boldsymbol{x},a,r) \in S} \frac{(r - \hat{r}(\boldsymbol{x},a))\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})}}{\sum_{(\boldsymbol{x},a,r) \in S} \frac{\pi(a|\boldsymbol{x})}{\hat{\beta}(a|\boldsymbol{x})}} + \hat{V}_{\mathrm{DM}}. \tag{A.5}$$

IPS-GCS. When general covariate shift exists, the density ratio of IPS is $\frac{P_t(\boldsymbol{x},a)}{P_s(\boldsymbol{x},a)}$. And the "IPS under general covariate shift" (IPS-GCS) is calculated by:

$$\hat{V}_{\text{IPS-GCS}} = \frac{1}{|S|} \sum_{(\boldsymbol{x},a,r)\in S} r \frac{P_t(\boldsymbol{x},a)}{P_s(\boldsymbol{x},a)}, \tag{A.6}$$

where $P_t(\boldsymbol{x},a) = P_t(\boldsymbol{x})\pi(a|\boldsymbol{x})$ and $P_s(\boldsymbol{x},a) = P_s(\boldsymbol{x})\hat{\beta}(a|\boldsymbol{x})$.

DM (R). We further obtain the baseline method DM (R) by setting $\mathcal{W}(\boldsymbol{x},a) = 1$ in the robust regression framework. Such a framework does not consider covariate shift information. The solution of DM (R) is formulated as follows:

$$\begin{aligned}
\mu_{\boldsymbol{\theta}}^{(\text{R})}(\boldsymbol{x},a) &= \sigma^2(\boldsymbol{x},a)\left(-2\boldsymbol{\theta}_{\boldsymbol{x}}\boldsymbol{\phi}(\boldsymbol{x},a) + \mu_0\sigma_0^{-2}\right), \\
\sigma_{\boldsymbol{\theta}}^{2(R)}(\boldsymbol{x},a) &= \left(2\theta_r + \sigma_0^{-2}\right)^{-1}.
\end{aligned} \tag{A.7}$$

We then plug the mean estimates from Equation A.7 into the direct method and obtain:

$$\hat{V}_{\text{DM (R)}} = \frac{1}{|S|} \sum_{\boldsymbol{x}\in S} \mathbb{E}_{a\sim\pi}\left[\mu_{\boldsymbol{\theta}}^{(\text{R})}(\boldsymbol{x},a)\right]. \tag{A.8}$$

A.2 Derivation of Robust Regression Model

We here show the derivation of the robust regression model. According to [1], if we have the constraints for the adversary g as follows,

$$\Sigma_S \coloneqq \left\{ g \middle| \left| \mathbb{E}_{\boldsymbol{x},a\sim P_s(\boldsymbol{x},a),r\sim g}[\boldsymbol{\delta}(r,\boldsymbol{x},a)] - \frac{1}{|S|}\sum_{i\in S}\boldsymbol{\delta}(r,\boldsymbol{x},a) \right| \le \eta \right\}, \tag{A.9}$$

then the solution of the minimax formulation has the parametric form:

$$\begin{aligned}
\hat{f}_{\boldsymbol{\theta}}(r|\boldsymbol{x},a) &\propto f_0(r|\boldsymbol{x},a)\exp\left(-\frac{P_s(\boldsymbol{x})\beta(a|\boldsymbol{x})}{P_t(\boldsymbol{x})\pi(a|\boldsymbol{x})}\boldsymbol{\theta}^T\boldsymbol{\delta}(r,\boldsymbol{x},a)\right) \\
&= f_0(r|\boldsymbol{x},a)\exp\left(-\frac{P_s(\boldsymbol{x},a)}{P_t(\boldsymbol{x},a)}\boldsymbol{\theta}^T\boldsymbol{\delta}(r,\boldsymbol{x},a)\right).
\end{aligned} \tag{A.10}$$

If we set the function $\boldsymbol{\delta}$ in a specific way such that it has a quadratic form and assuming the base distribution $f_0 \sim \mathcal{N}(\mu_0,\sigma_0)$, we obtain a Gaussian distribution. We provide the necessary details here for the derivation of the mean and variance from this special form and refer more details to [1].

If our $\boldsymbol{\delta} \coloneqq \text{vector}\left([r, \boldsymbol{\phi}(\boldsymbol{x}, a)]^T [r, \boldsymbol{\phi}(\boldsymbol{x}, a)]\right)$, it corresponds to the following constraints in Σ:

$$\Sigma_S \coloneqq \left\{ g \left| \left| \mathbb{E}_{\boldsymbol{x}, a \sim P_s(\boldsymbol{x}, a), r \sim g}[r(\boldsymbol{x}, a)^2] - \frac{1}{|S|} \sum_{i \in S} r_i^2 \right| \leq \eta, \right. \right.$$
$$\left. \left| \mathbb{E}_{\boldsymbol{x} \sim P_s, a \sim \beta, r \sim g}[r(\boldsymbol{x}, a)\boldsymbol{\phi}(\boldsymbol{x}, a)] - \frac{1}{|S|} \sum_{i \in S} r_i \boldsymbol{\phi}(\boldsymbol{x}, a) \right| \leq \eta \right\}, \tag{A.11}$$

With strong duality and the constraint set for the adversary g in Equation A.9, the solution of loss function Equation 2.3 is equivalent to the solution of the following minimization problem:

$$\min_{\hat{f}} D_{\boldsymbol{x}, a \sim P_t(\boldsymbol{x}, a), r \sim \hat{f}}(\hat{f} \| f_0),$$
$$\text{such that: } \mathbb{E}_{\boldsymbol{x}, a \sim P_s(\boldsymbol{x}, a), r \sim \hat{f}}[\delta(r, \boldsymbol{x}, a)] = c,$$

where $c = \frac{1}{n} \sum_i \delta(r_i, \boldsymbol{x}_i, a_i)$. Let $\hat{f}_{\boldsymbol{\theta}}$ be defined as:

$$\hat{f}_{\boldsymbol{\theta}} = \frac{f_0(r|\boldsymbol{x}, a) \exp\left(-\mathcal{W}(\boldsymbol{x}, a)\boldsymbol{\theta}^T \delta(r, \boldsymbol{x}, a)\right)}{Z(\boldsymbol{x}, a, \boldsymbol{\theta})},$$

where $Z(\boldsymbol{x}, a, \theta)$ denote the normalization term. Moreover, let

$$\Phi(r, \boldsymbol{x}, a) = -\mathcal{W}(\boldsymbol{x}, a)\boldsymbol{\theta}^T \delta(r, \boldsymbol{x}, a) - \log Z(\boldsymbol{x}, a, \boldsymbol{\theta}),$$

and use the $\boldsymbol{\theta}$ as the Lagrangian parameters, the Lagrangian of the optimization problem is written as:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= D_{\boldsymbol{x}, a \sim P_t(\boldsymbol{x}, a), r \sim \hat{f}}(\hat{f} \| f_0) + \boldsymbol{\theta}^T \left(\mathbb{E}_{P_s(\boldsymbol{x}, a, r)}[\delta(r, \boldsymbol{x}, a)] - c\right) \\
&= \int_{\boldsymbol{x}, a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \int_{r|\boldsymbol{x}, a} \hat{f}(r|\boldsymbol{x}, a) \log(\frac{\hat{f}}{f_0}) + \boldsymbol{\theta}^T \left(\mathbb{E}_{P_s(\boldsymbol{x}, a, r)}[\delta(r, \boldsymbol{x}, a)] - c\right) \\
&= \int_{\boldsymbol{x}, a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \int_{r|\boldsymbol{x}, a} \hat{f}(r|\boldsymbol{x}, a)(-\mathcal{W}(\boldsymbol{x}, a)\boldsymbol{\theta}^T \delta(r, \boldsymbol{x}, a)) \\
&- \int_{\boldsymbol{x}, a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \int_{r|\boldsymbol{x}, a} \hat{f}(r|\boldsymbol{x}, a) \log Z(\boldsymbol{x}, a, \boldsymbol{\theta}) \\
&+ \boldsymbol{\theta}^T \left(\int_{\boldsymbol{x}, a} P_s(\boldsymbol{x})\beta(a|\boldsymbol{x}) \int_{r|\boldsymbol{x}, a} \hat{f}(r|\boldsymbol{x}, a)\delta(r, \boldsymbol{x}, a) - c \right) \\
&= -\int_{\boldsymbol{x}, a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \log Z(\boldsymbol{x}, a, \boldsymbol{\theta}) - \boldsymbol{\theta}^T c.
\end{aligned}$$

If we take the maximum of $\mathcal{L}(\boldsymbol{\theta})$, we have the following result:

$$
\begin{aligned}
\arg\max_{\boldsymbol{\theta}} \mathcal{L} &= \arg\max_{\boldsymbol{\theta}} - \int_{\boldsymbol{x},a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \log Z(\boldsymbol{x}, a, \boldsymbol{\theta}) - \boldsymbol{\theta}^T c \\
&= \arg\max_{\boldsymbol{\theta}} - \int_{\boldsymbol{x},a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \log Z(\boldsymbol{x}, a, \boldsymbol{\theta}) \\
&\qquad - \boldsymbol{\theta}^T \int_{\boldsymbol{x},a} P_s(\boldsymbol{x})\beta(a|\boldsymbol{x}) \int_{r|\boldsymbol{x},a} P^*(r|\boldsymbol{x}, a)\delta(r, \boldsymbol{x}, a) \\
&= \arg\max_{\boldsymbol{\theta}} - \int_{\boldsymbol{x},a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \log Z(\boldsymbol{x}, a, \boldsymbol{\theta}) \\
&\qquad - \boldsymbol{\theta}^T \int_{\boldsymbol{x},a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x})\mathcal{W}(\boldsymbol{x}, a) \int_{r|\boldsymbol{x},a} P^*(r|\boldsymbol{x}, a)\delta(r, \boldsymbol{x}, a) \\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{x},a} P_t(\boldsymbol{x})\pi(a|\boldsymbol{x}) \\
&\qquad \int_{r|\boldsymbol{x},a} P^*(r|\boldsymbol{x}, a) \left(\log \frac{1}{Z(\boldsymbol{x}, a, \boldsymbol{\theta})} - \mathcal{W}(\boldsymbol{x}, a)\boldsymbol{\theta}^T \delta(r, \boldsymbol{x}, a) \right) \\
&= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},a\sim P_t(\boldsymbol{x},a), r\sim P*} \left[\log \hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, a) \right].
\end{aligned}
$$

If we use parameters $\boldsymbol{\theta} \coloneqq [\theta_r, \boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{\theta}_{\boldsymbol{xx}}]$ as the Lagrangian parameters (here $\boldsymbol{\theta}_{\boldsymbol{x}}$ is a vector with the same dimension as $\boldsymbol{\phi}$), we can derive the following predictive form for $\hat{f}_{\boldsymbol{\theta}}(r|\boldsymbol{x}, a)$ (note that $\boldsymbol{\theta}_{\boldsymbol{xx}}$ is only relevant to $\boldsymbol{\phi}(\boldsymbol{x}, a)$ but not r, so we do not need to learn it explicitly):

$$
\hat{f}_{\boldsymbol{\theta}}(r|\boldsymbol{x}, a) \propto f_0 \exp\left(-\frac{P_s(\boldsymbol{x}, a)}{P_t(\boldsymbol{x}, a)} \boldsymbol{\theta}^T \boldsymbol{\delta}(r, \boldsymbol{x}, a) \right) \tag{A.12}
$$

$$
= f_0 \exp\left(-\frac{P_s(\boldsymbol{x}, a)}{P_t(\boldsymbol{x}, a)} (\theta_r r^2 + 2\boldsymbol{\theta}_{\boldsymbol{x}} r \boldsymbol{\phi}(\boldsymbol{x}, a) + \boldsymbol{\theta}_{\boldsymbol{xx}} \boldsymbol{\phi}(\boldsymbol{x}, a)^2) \right)
$$

$$
= \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left(-\frac{1}{2} \frac{(r - \mu_0)^2}{\sigma_0^2} \right) \cdot \exp\left(-\frac{P_s(\boldsymbol{x}, a)}{P_t(\boldsymbol{x}, a)} (\theta_r r^2 + 2\boldsymbol{\theta}_x r \boldsymbol{\phi}(\boldsymbol{x}, a) + \boldsymbol{\theta}_{\boldsymbol{xx}} \boldsymbol{\phi}(\boldsymbol{x}, a)^2) \right) \tag{A.13}
$$

$$
\propto \exp\left(-\left(\frac{1}{2\sigma_0^2} + \frac{P_s(\boldsymbol{x}, a)}{P_t(\boldsymbol{x}, a)} \theta_r \right) r^2 + (-2\frac{P_s(\boldsymbol{x}, a)}{P_t(\boldsymbol{x}, a)} \boldsymbol{\theta}_{\boldsymbol{x}} \boldsymbol{\phi}(\boldsymbol{x}, a) + \frac{\mu_0}{\sigma_0^2}) r \right) \tag{A.14}
$$

$$
\propto \exp\left(-\frac{1}{2} \frac{(r - \mu_{\boldsymbol{\theta}})^2}{\sigma_{\boldsymbol{\theta}}^2} \right), \tag{A.15}
$$

where

$$\sigma_{\boldsymbol{\theta}}^2(\boldsymbol{x}, a) = \left(2\frac{P_s(\boldsymbol{x}, a)}{P_t(\boldsymbol{x}, a)}\theta_r + \sigma_0^{-2}\right)^{-1}, \tag{A.16}$$

$$\mu_{\boldsymbol{\theta}}(\boldsymbol{x}, a) = \sigma_{\boldsymbol{\theta}}^2(\boldsymbol{x}, a)\left(-2\frac{P_s(\boldsymbol{x}, a)}{P_t(\boldsymbol{x}, a)}\boldsymbol{\theta}_{\boldsymbol{x}}\boldsymbol{\phi}(\boldsymbol{x}, a) + \mu_0\sigma_0^{-2}\right). \tag{A.17}$$

Gaussian Predictive Form

Here we provide an example that the data is generated from cubic form distribution instead of Gaussian distribution. In this scenario, we can adjust the robust regression model by incorporating the cubic function feature constraint instead of the quadratic constraint. Denote x as the covariate variable, y as the target variable, $f_0(y|x)$ as the prior and $\phi(x, y)$ to be the cubic function feature constraint. We can obtain the derived predictive model form of $p(y|x) \propto f_0(y|x)\exp(-w(x)\theta\phi(x, y))$, where θ is the model parameter and $w(x)$ is the density ratio. This estimator has a hard-to-track normalization denominator for the predictive model distribution, so we need to sample from this distribution to obtain the prediction of y from such a model, which will sacrifice the nice implementation convenience of our robust regression model, and we may need to tailor specific sampling methods to efficiently obtain the prediction from such an estimator.

We conduct experiments under settings where $p(y|x)$ is not Gaussian by generating synthetic data (x, y) such that $p(y|x) \propto \exp(y^3-2y^2x-3yx^2-4y^2+5yx+6y)$ following a cubic feature constraint assumption. Specifically, we created the distribution shift by generating x following $x_{\text{train}} \sim N(0, 1)$ and $x_{\text{test}} \sim N(1, 2)$. There are 500 data points in both the train and test datasets. Since the cubic feature constraint does not result in a Gaussian distribution predictor model like the quadratic feature setting and the denominator of such distribution is intractable, we use discrete target y by sampling $y \in \{0,1\}$ from the aforementioned prediction distribution in this experiment, for both the data generation and getting predictions from the robust regression model.

The cubic form robust regression achieved a mean squared error (MSE) of 0.31 ($\pm$0.00029). The quadratic form robust regression received an MSE of 0.42($\pm$0.0003), and the linear regression model received an MSE of 0.49($\pm 1.2 \times 10^{-10}$). These results are averaged over 10 runs. These results suggest that a more tailored feature set could improve the performance in synthetic settings and using a simpler hypothesis class could induce bias. However, given the original model for regression without Gaussian assumption, $p(y|x) \propto \exp(-w(x)\theta\phi(x, y))$, can be less convenient for making predictions (which is the reason we need to use sampling to get the y in this

synthetic experiments) and the broad applicability of Gaussian data assumptions, a robust regression model with Gaussian assumptions is still preferred in practice.

A.3 Proof of Bias Analysis

Expected Error. We prove the bias analysis results of the proposed robust regression estimators under large distributional shift, building upon established results from [5]. We assume the selection of a base distribution $\mathcal{N}(\mu_0, \sigma_0^2)$, such that the expected squared error over the target distribution $\mathbb{E}_{P_t(\boldsymbol{x},a,r)}[(r-\mu_0)^2]$ is upper bounded by some constant H. Typically, this condition is achieved in practice by appropriately setting the base distribution, such as to be the empirical mean of the dataset.

Our analysis begins with establishing a generalization bound for the expected squared error of the proposed robust regression-based direct method over the target distribution. We divide the target distribution data into two disjoint subsets based on whether it shares support with the source distribution, which is indicated by the value of its density ratio between the source and target distribution. Specifically, for the target distribution data points $(\boldsymbol{x}, a)$ with $\mathcal{W}(\boldsymbol{x}, a) \to 0$, which means they do not share support with the source distribution, the robust regression estimator is reduced to the base estimator based on the formulation. Consequently, under the aforementioned assumption, the following inequality holds:

$$\mathbb{E}_{P_t(\boldsymbol{x},a,r)}\left[(r-\mu_0)^2\right] \leq H.$$

For the remaining data points in the target distribution with $\mathcal{W}(\boldsymbol{x}, a) > 0$, we assume that $\frac{P_t(\boldsymbol{x},a)}{P_s(\boldsymbol{x},a)}$, the density ratio of target over the source distribution, is upper bounded by some constant C. Drawing upon Theorem 1 from [5], we derive the following upper bound for the expected squared error over the target distribution:

$$\mathbb{E}_{P_t(\boldsymbol{x},a,r)}[(r-\hat{\mu}(\boldsymbol{x},a))^2] \leq C\left[\frac{1}{n}\sum_{i=1}^{n}|r^2-\hat{\mu}(\boldsymbol{x},a)^2| + 4M\hat{\mathfrak{R}}(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right],$$

where $\mathcal{F}$ is the function class of f with $\sup_{f,f'\in\mathcal{F},\boldsymbol{x},a}|f(\boldsymbol{x},a)-f'(\boldsymbol{x},a)| \leq M$, and with a Rademacher complexity of $\hat{\mathfrak{R}}(\mathcal{F})$. Assuming the training process converges satisfactorily, we can upper bound the training gradient and derive the following

results:

$$r^2 - (\mu(\boldsymbol{x}, a)^2 + \sigma^2(\boldsymbol{x}, a)) \leq \eta_1;$$
$$r - \mu(\boldsymbol{x}, a) \leq \frac{\eta_1}{l}, \tag{A.18}$$

where η_1 is some small constant and l is the lower bound of all the features in the context and action pair representation $f(x, a)$. Without loss of generality, we assume that there are n i.i.d sampled data samples in training and m percentage of samples have a non-zero density ratio $\mathcal{W}(x, a)$ in the target data distribution. We have the following upper bound for the expected error on the target distribution holds with probability at least $1 - \delta$:

$$\begin{aligned}
&\mathbb{E}_{P_t(\boldsymbol{x},a,r)}[(r - \hat{\mu}(\boldsymbol{x}, a))^2] \\
&\leq mC \left[\frac{1}{n} \sum_{i=1}^{n} \sigma^2(\boldsymbol{x}_i, a_i) + \eta_1 + 4M\hat{\mathfrak{R}}(\mathcal{F}) + 3M^2 \sqrt{\frac{\log \frac{2}{\delta}}{2n}} \right] + (1 - m)H.
\end{aligned} \tag{A.19}$$

Bias Analysis of DM. Building upon our analysis of the expected squared error of the proposed direct method using robust regression as above, we establish an upper bound for the bias of the robust regression-based direct method estimator. The following bound for the bias of DM-PS/DM-GCS holds with probability at least $1 - \delta$:

$$\begin{aligned}
&\mathbb{E}_{P_t(\boldsymbol{x},a,r)}[(r - \hat{\mu}(\boldsymbol{x}, a))] \\
&\leq \left[mC \left[\frac{1}{n} \sum_{i=1}^{n} \sigma^2(\boldsymbol{x}_i, a_i) + \eta_1 + 4M\hat{\mathfrak{R}}(\mathcal{F}) + 3M^2 \sqrt{\frac{\log \frac{2}{\delta}}{2n}} \right] + (1 - m)H \right]^{1/2} \\
&:= \epsilon.
\end{aligned}$$

Bias Analysis of DR. We then analyze the bias of the proposed DR-PS and DR-GCS estimators. Similarly, assuming the training process converges satisfactorily, we have the following lemma by upper bounding the training gradient and using Rademacher Complexity.

Lemma A.1. *The expectation of the L_1 distance between the reward and the mean estimator from robust regression over the source distribution satisfies the following*

with probability at least $1-\delta$*:*

$$\mathbb{E}_{P_s(\boldsymbol{x},a,r)}\left[|r-\mu(\boldsymbol{x},a)|\right] \leq \frac{\eta_1}{l} + 2M\hat{\mathfrak{R}}(\mathcal{F}) + 3M\sqrt{\frac{\log\frac{2}{\delta}}{2n}}, \tag{A.20}$$

where l *is the lower bound of all the features in the context and action pair representation* $f(x,a)$*.*

Given that Equation A.20 in Lemma A.1 holds, we establish the following bound for the bias of DR-PS/DR-GCS:

$$\left|\mathbb{E}_{P_t(\boldsymbol{x},a,r)}[\hat{V}-V_\pi]\right| \leq \left|\mathbb{E}_{P_s(\boldsymbol{x},a,r)}\left[C(r-\hat{\mu}(\boldsymbol{x},a))\right]\right| + \left|\mathbb{E}_{P_t(\boldsymbol{x},a,r)}\left[r-\hat{\mu}(\boldsymbol{x},a)\right]\right| \tag{A.21}$$

$$\leq \frac{C\eta_1}{l} + 2CM\hat{\mathfrak{R}}(\mathcal{F}) + 3CM\sqrt{\frac{\log\frac{2}{\delta}}{2n}} + \epsilon \tag{A.22}$$

Consistency

We here analyze the asymptotic convergence of the proposed DM-PS/DM-GCS and DR-PS/DR-GCS estimators using the asymptotic performance of robust regression method [1, 3]. We have the following conclusions.

Proposition A.2. *Assuming* $\mathbb{E}_{P_t(\boldsymbol{x},a)}[\mathcal{W}(\boldsymbol{x},a)]$ *is bounded away from zero,* $\mathcal{W}(\boldsymbol{x},a)$ *is accurately estimated in training,* $\boldsymbol{\phi}$ *is chosen to be a universal kernel, then DM-PS/DM-GCS is consistent.*

Remarks. This proposition can be proved by directly applying the consistency results of the distributionally robust learning [3]. Here, the estimator will converge to zero error over the target distribution under the distribution shift. Note that the result is applicable for all loss functions.

We then analyze the asymptotic behavior of DR-PS/DR-GCS estimators.

Proposition A.3. *If DM-PS/DM-GCS is consistent, then DR-PS/DR-GCS is consistent.*

Remarks. This proposition is obvious since we formulate the DR-PS/DR-GCS method as doubly robust methods with robust regression-based reward models. It demonstrates that using robust regression as a reward model does not hurt the asymptotic consistency of the doubly robust methods. Despite we need more assumptions for DM-PS/DM-GCS to be consistent, such as accurate $\mathcal{W}(x,a)$ and expressive feature representation $\boldsymbol{\phi}$, it does not mean that DR-PS/DR-GCS is weaker

than general DR since we use a specific distributionally robust model for the reward model, while traditional DR analysis treats the reward model as a black box.

A.4 Experimental details

We here describe the logging policy, target policy, and covariate we mentioned in Section 2.5 in detail and the specific parameters we used. We also provide experiment results on each dataset. The code for the experiments can be found at https://github.com/guoyihonggyh/Distributionally-Robust-Policy-Evaluation-under-General-Co–variate-Shift-in-Contextual-Bandits

Data Generation

Logging Policies. We further describe our logging policy here. Firstly, we employed the softened policy, which is in line with previous work of OPE [2, 6]. The softened policy is defined by two parameters, denoted as (λ, ζ), and a trained deterministic policy using logistic regression. Additionally, inspired by previous research on label shift [4] and offline contextual bandit optimization [10], we generated the other two types of logging policies, Tweak-1 (ρ) and Dirichlet (γ), by creating biased "action label distributions".

- Softened policy, $\pi_{(\lambda,\zeta)}(a|\boldsymbol{x})$. Given a deterministic policy $\hat{\psi}$ and parameters (λ, ζ), the softened policy is defined as $\pi_{(\lambda,\zeta)}(a|\boldsymbol{x}) = \lambda + \zeta u$ if $a = \hat{\psi}(\boldsymbol{x})$ and other classes evenly share the $1 - (\lambda + \zeta u)$, where $u \sim \text{Uniform}(-0.5, 0.5)$. The deterministic policy $\hat{\psi}(x)$ is trained by logistic regression on 10% of training data. In our paper, we consider four different softened policies, which are $\pi_{(0.95,0)}(a|\boldsymbol{x})$, $\pi_{(0.7,0.1)}(a|\boldsymbol{x})$, $\pi_{(0.5,0.1)}(a|\boldsymbol{x})$ and $\pi_{(0.1,0)}(a|\boldsymbol{x})$.
- Tweak-1 (ρ). We allocate a probability of ρ to one class while the remaining classes equally divide the residual probability of $1 - \rho$. A higher ρ indicates a larger distribution shift of the logging dataset since more probability values close to 0 or 1 exist when ρ is large. We evaluate this setting with ρ values of $0.91, 0.95, 0.99$.
- Dirichlet (γ). Dirichlet (γ) represents a Dirichlet probability distribution parameterized by γ. A smaller γ implies a larger shift of the logging dataset since there are more probability values close to 0 or 1. We generate one logging policy randomly and fix that across all experiments. Our study considers γ values of $1.0, 0.5$, and 0.1. Note that with Dirichlet (0.1), the probability for some classes becomes so low that they may not be sampled. Hence, for Dirichlet (0.1), we utilize a logging policy of $0.95 \times \text{Dirichlet}(0.1) + 0.05 \times \text{Uniform}$.

Target Policies. We test our method on various combinations of logging policies and target policies. The target policies are as follows:

- Softened policy, $\pi_{(0.9,0)}(a|\boldsymbol{x})$. We first obtain a deterministic policy ψ by training a logistic regression on the entire training set. Then we set $(\lambda = 0.9, \zeta = 0)$ and obtain the softened policy.
- Softened perfect policy, $\pi_{(\text{perfect},\lambda)}$. Given access to the ground label when generating the policy, we can have the *perfect policy*, π_{perfect}, which means that for every x, the policy always chooses the action that receives the highest reward. Following the idea of softened policy, we can soften the π_{perfect} with $(\lambda, \zeta) = (\lambda, 0)$ to obtain $\pi_{(\text{perfect},\lambda)}$. Our paper considers $\lambda = 0.9, 0.7, 0.5$, respectively.
- Diverse softened perfect policy. We create a case where the performance of policy evaluation will be substantially affected by the extent of the general covariate shift. As the covariate shift of the data we consider would generally make the target data only focus on a subset of all the classes in a dataset, we make the target policy value vary across different classes. Specifically, we set the target policy value for each class as a random permutation of $[\frac{1}{n}, \frac{2}{n}, ..., 1]$, where n is the number of arms. This can be viewed as a discrete version of uniform sampling from [0,1].

Covariate Shift on the Data. We consider various context shifts in the experiments. We uniformly sample context for the target data while we consider the following choices for the scale of the distribution shift of the logging data distribution.

- Gaussian covariate shift. We first perform PCA on the dataset and select the first component c. We then define a Gaussian distribution with mean as $\min(c) + \frac{\min(c)-\text{mean}(c)}{a}$ and standard deviation as $\frac{\text{std}(c)}{b}$, where $a \in (0, \text{inf})$ and $b \in (0, 1)$. We then sample logging data based on the generated Gaussian distribution. Smaller b means a smaller variance in the generated Gaussian distribution and a larger shift. Also, when $a > 1$, larger a means a larger shift, when $a < 1$, smaller a means a small shift. In our experiments, we consider the settings of $(a, b) = (1.5, 3), (2, 2), (1.5, 2), (0.6, 2)$, respectively.
- Tweak-1 covariate (ω). Following the idea of the $Tweak-1(\rho)$ logging policy, we assign one class of data with a higher weight ω and other classes with weight 1. We can obtain a probability distribution for each data by normalizing the weight to 1. A larger ω means a larger shift. Our paper considers the setting of $\omega = 15, 12, 9, 6, 4, 2$, respectively.

When only policy shifts exist, we consider $9 \times 10 \times 4 = 360$ conditions derived from 9 datasets, 10 logging policies, and 4 target policies. When general covariate shifts exist, we consider $9 \times 7 \times 10 \times 2 = 1260$ conditions derived from 9 datasets, 7 logging policies, 10 covariate shifts, and 2 target policies. Specifically, the 7 logging policy are $\pi_{(0.95,0.1)}(a|\boldsymbol{x}), \pi_{(0.7,0.1)}(a|\boldsymbol{x})$, Tweak-1 ($\rho = 0.99$),Tweak-1 ($\rho = 0.95$), Tweak-1 ($\rho = 0.91$), Dirichlet (1.0) and Dirichlet (0.1). The two target policies are $\pi_{(0.9,0)}(a|\boldsymbol{x})$ and softened perfect policy $\pi_{(\text{perfect},0.7)}(a|\boldsymbol{x})$.

Dataset Generation. Based on the aforementioned settings, the dataset generation and evaluation process is as follows:

1. **Data Split:** We randomly split the original dataset into 75% training set $\mathcal{D}_{\text{TR}}$ and 25% test set $\mathcal{D}_{\text{TS}}$.
2. **Context and Policy Distribution Generation:** We generate logging policy β and target policy π given the policy parameters. If under the general covariate shift (GCS) setting, we further generate a logging context distribution $P_s(\boldsymbol{x})$.
3. **Target Policy Value Calculation:** We compute the ground truth of the target policy π's value V^T on the entire test set $\mathcal{D}_{\text{TS}}$, by sample the action a for each context $\boldsymbol{x}$ in $\mathcal{D}_{\text{TS}}$ and get the corresponding reward r.
4. **Logging Dataset Generation:** If under the GCS setting, we sample context $\boldsymbol{x}$ from the training set $\mathcal{D}_{\text{TR}}$ based on the logging context distribution $P_s(\boldsymbol{x})$. Otherwise, we sample the logging context $\boldsymbol{x}$ uniformly. We then sample the action a for each context $\boldsymbol{x}$ using the logging policy β and obtain the corresponding reward r to create the logging dataset $\mathcal{D}_{\text{TR-Logging}}$.
5. **Model Training:** We train the reward predictors, including DM, DM(R), DM-PS, and DM-GCS, using $(\boldsymbol{x}, a, r)$ in the logging dataset $\mathcal{D}_{\text{TR-Logging}}$.
6. **Evaluation:** We sample another logging dataset $\mathcal{D}_{\text{TS-Logging}}$ following the Logging Dataset Generation process in step 4 from the test set $\mathcal{D}_{\text{TS}}$ for estimating the importance sampling part of the estimators. All methods' final estimated policy value $\hat{V}$ is calculated on $\mathcal{D}_{\text{TS-Logging}}$.

Hyperparameters. The base distribution for robust regression is a Gaussian distribution with mean = 0.6 and variance = 1. θ is updated with SGD. We tune the hyperparameters with a grid search on the learning rate in [0.001,0.0005]. We also set the learning rate decay for the learning of θ, where the learning rate is multiplied by $\frac{10}{10+\sqrt{i-1}}$ at i-th epoch. The batch size is searched in [8, 32, 64, 256].

Comparison of DM-PS Family and Other Baselines under Policy Shift

	SnIPS	DM	DM(R)	DM-PS
$\beta(a\|\boldsymbol{x})$ known	9	7	10	334
$\beta(a\|\boldsymbol{x})$ unknown	3	17	13	327

Table A.1: The number of conditions where each family is the best under policy shift. DM-PS family outperforms baseline methods in over 90% of the conditions.

Comparison of SnIPS and SnIPS-GCS under different covariate shift

We also analyze the impact of various types of covariate shifts on the performance of SnIPS and SnIPS-GCS in Table A.2 as below. Specifically, SnIPS exhibits superior performance under the Gaussian covariate shift, which generally creates a larger covariate shift compared to the Tweak-1 covariate shift, outperforming SnIPS-GCS across an extensive range of conditions. While under the Tweak-1 shift, SnIPS-GCS demonstrates enhanced performance in a larger number of scenarios than SnIPS. This observation demonstrates that large covariate shifts create harder-to-estimate density ratios and negatively affect the performance of SnIPS-GCS, compared to SnIPS.

Table A.2: The number of conditions where each estimator is the best when $P_s(\boldsymbol{x}, a)$ is known.

covariate shift	Target policy	SnIPS	SnIPS-GCS
Gaussian covariate	$\pi_{(0.9,0)}(a\|\boldsymbol{x})$	870	390
	Diverse Softened Perfect policy	945	315
Tweak-1 covariate	$\pi_{(0.9,0)}(a\|\boldsymbol{x})$	404	856
	Diverse Softened Perfect policy	240	1020

Further Comparison with Advanced OPE Methods

We present the result of our method applied to and compared with other advanced OPE approaches such as SWITCH [9] and DRoS [7] in Figure A.1. Our method can be integrated into SWITCH and DRoS by replacing their reward estimator with our estimators, DM-PS and DM-GCS. Incorporating DM-PS into SWITCH and DRoS results in SWITCH-PS and DRoS-PS, respectively. Similarly, applying DM-GCS to SWITCH and DRoS results in SWITCH-GCS and DRoS-GCS. As shown in Figure A.1, SWITCH-PS and DRoS-PS consistently outperform the original SWITCH and DRoS methods, respectively, across all cases, including both policy shift and general covariate shift cases. Additionally, in the general covariate shift setting, the performance of DRoS is also further improved by DM-GCS. These results

demonstrates the advantage of integrating DM-PS and DM-GCS into SWITCH and DRoS.

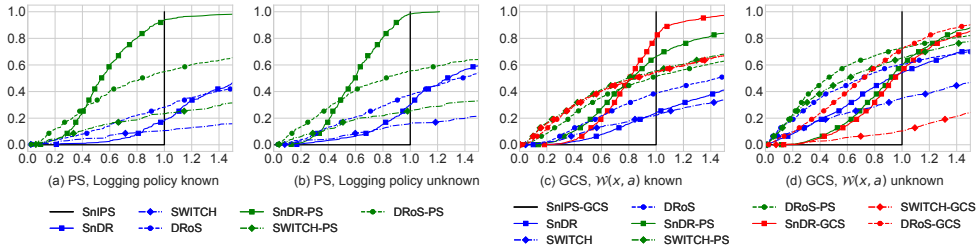
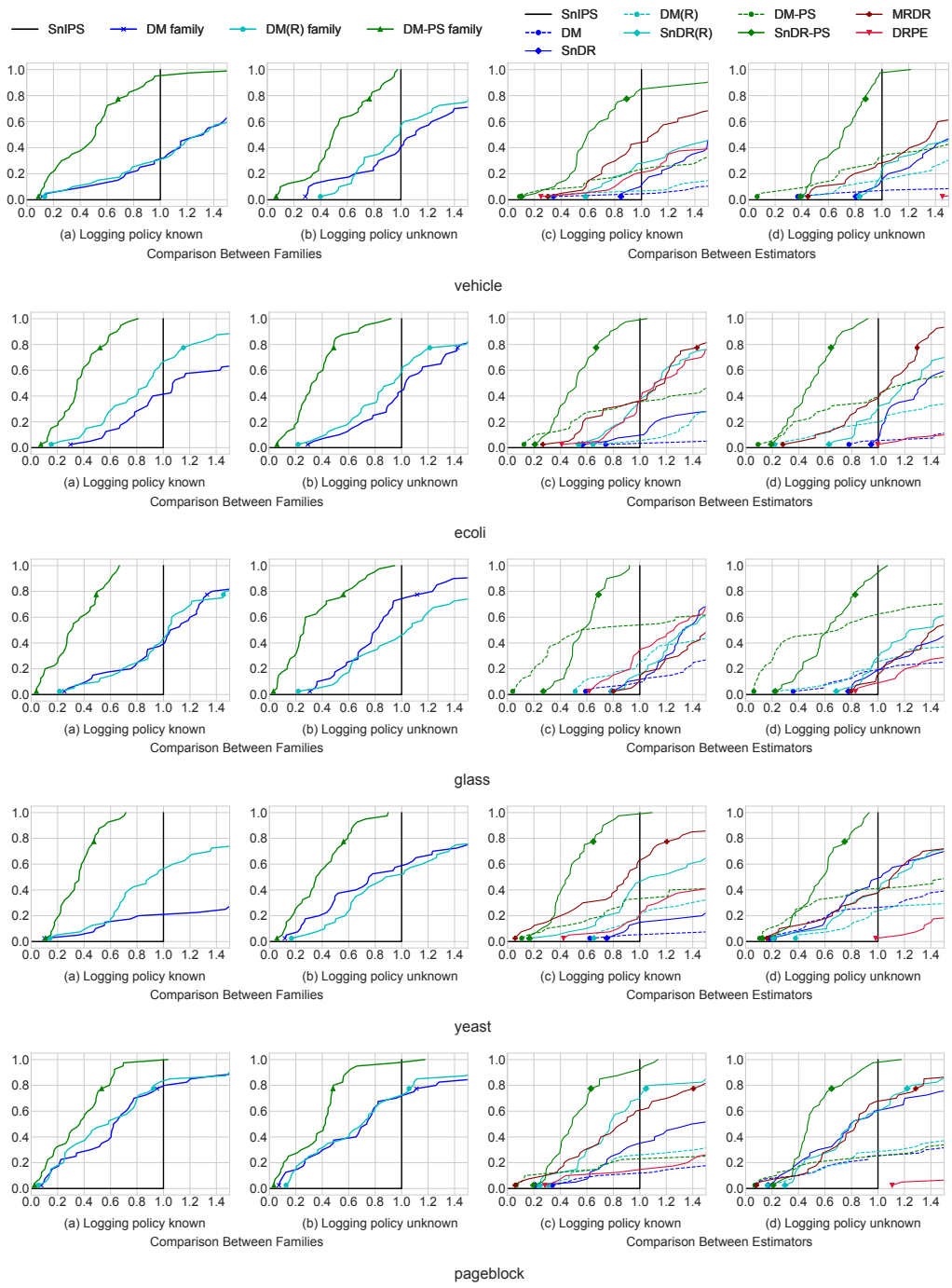


Figure A.1: Subfigures (a) and (b) present the experiment results under policy shift. Subfigures (c) and (d) present the experiment results under general covariate shift. DM-PS and DM-GCS enhance the performance of SWITCH and DRoS.

Experiments Results on Each Dataset

We show the experiment results on each dataset under policy shift and general covariate shift in Figure A.2 and Figure A.3 as below. Specifically, each row corresponds to the result of each dataset. In each row, subfigures (a) and (b) are comparisons between different families of methods, while (c) and (d) are comparisons between individual estimators. Subfigures (a) and (c) display the results when the $\beta(a|\boldsymbol{x})$ is known in policy shift conditions or $\mathcal{W}(x,a)$ is known in general covariate shift conditions, while (b) and (d) present the results when they are unknown.


SnIPS
DM family
DM(R) family
DM-PS family
SnIPS
DM
SnDR
DM(R)
SnDR(R)
DM-PS
SnDR-PS
MRDR
DRPE
(a) Logging policy known
(b) Logging policy unknown
(c) Logging policy known
(d) Logging policy unknown
Comparison Between Families
Comparison Between Estimators
vehicle
ecoli
glass
yeast
pageblock

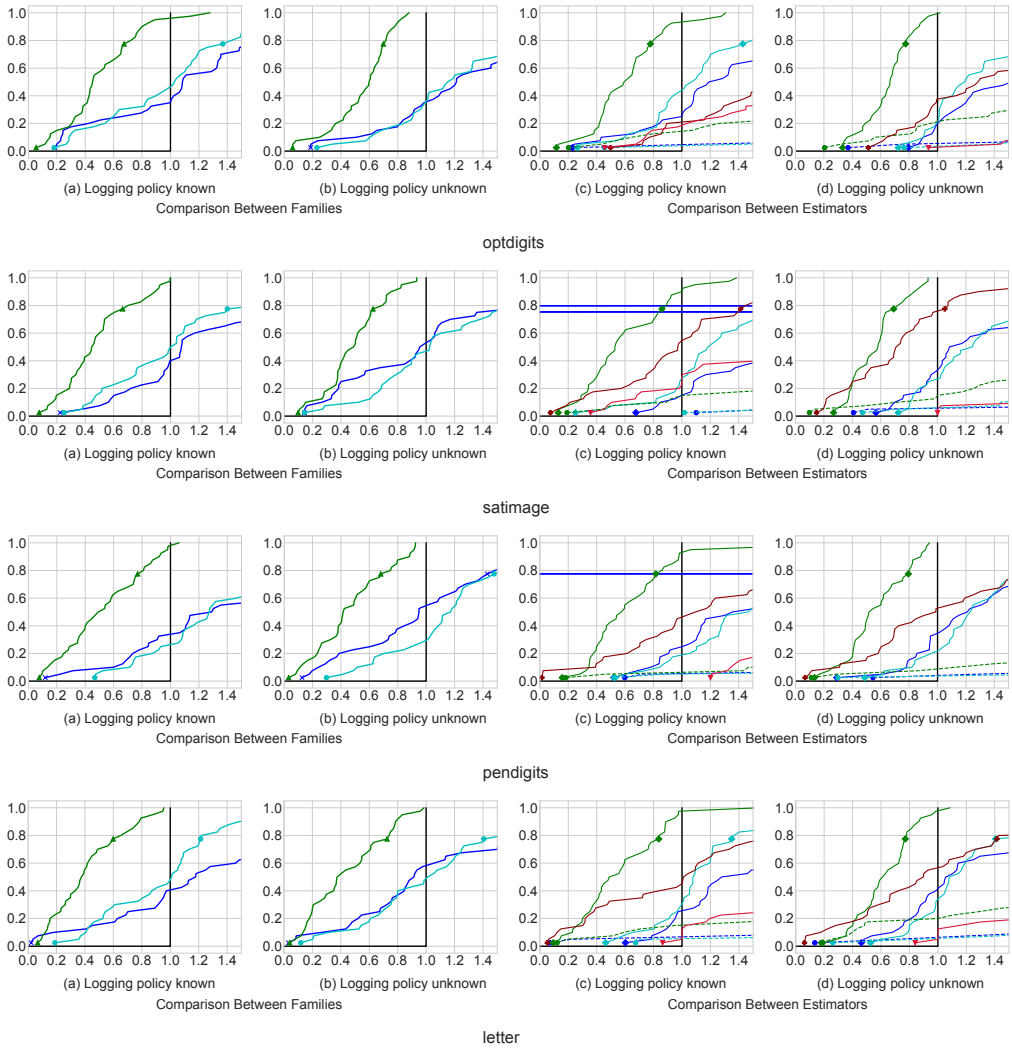
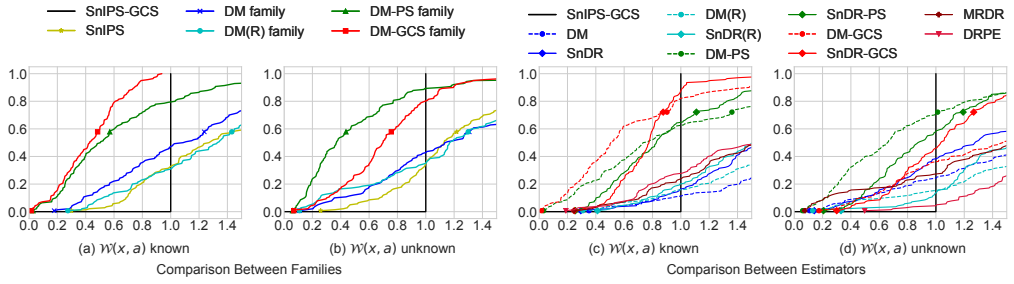
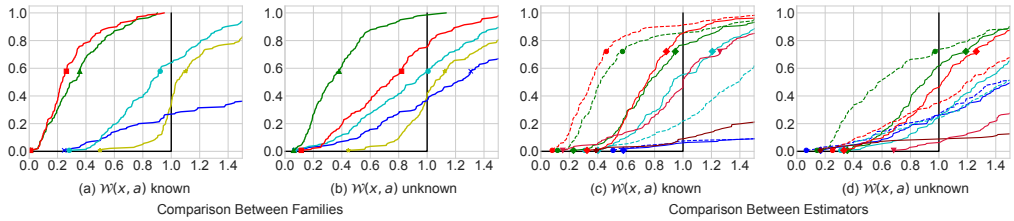


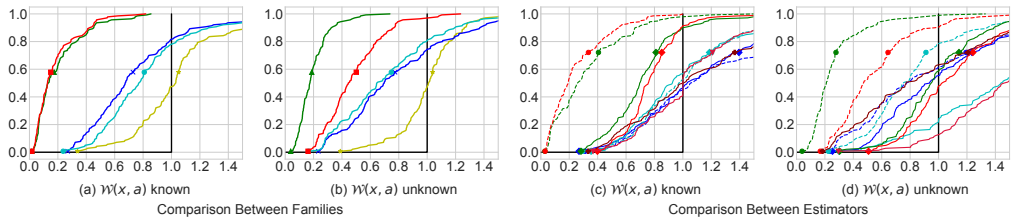
Figure A.2: Experimental results of different datasets under policy shifts.



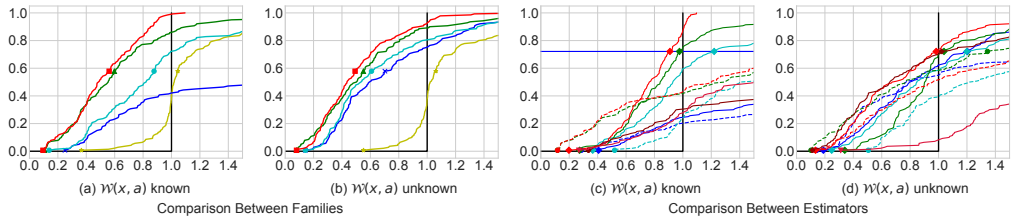
vehicle



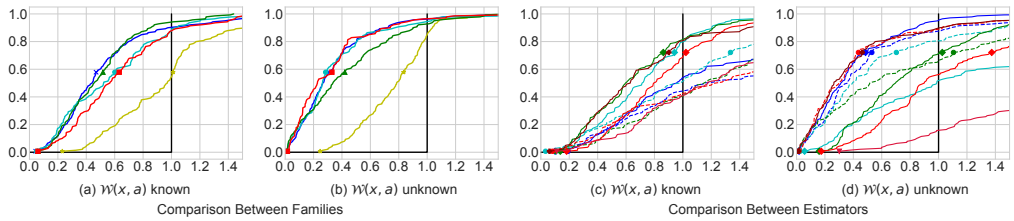
ecoli



glass



yeast



pageblock

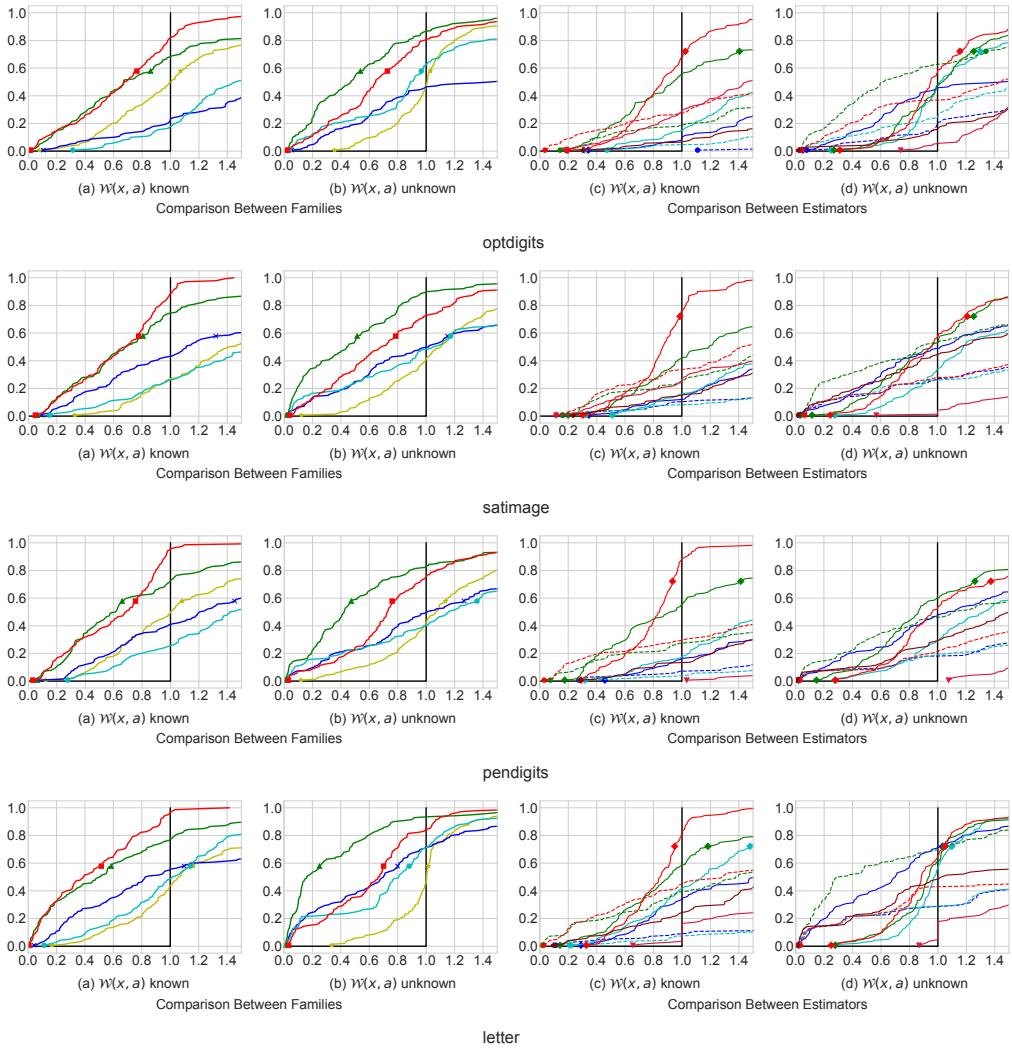


Figure A.3: Experimental results of different datasets under general covariate shifts.

References

[1] Xiangli Chen et al. “Robust covariate shift regression”. In: *Artificial Intelligence and Statistics*. 2016.

[2] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. “More robust doubly robust off-policy evaluation”. In: *International Conference on Machine Learning (ICML)*. 2018.

[3] Rizal Fathony et al. “Adversarial multiclass classification: A risk minimization perspective”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 559–567.

[4] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. “Detecting and correcting for label shift with black box predictors”. In: *International conference on machine learning*. PMLR. 2018, pp. 3122–3130.

[5] Anqi Liu et al. “Robust Regression for Safe Exploration in Control”. In: *arXiv preprint arXiv:1906.05819* (2019).

[6] Yi Su et al. “Doubly robust off-policy evaluation with shrinkage”. In: *arXiv preprint arXiv:1907.09623* (2019).

[7] Yi Su et al. “Doubly robust off-policy evaluation with shrinkage”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9167–9176.

[8] Adith Swaminathan and Thorsten Joachims. “The self-normalized estimator for counterfactual learning”. In: *Neural Information Processing Systems (NeurIPS)*. 2015.

[9] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudik. “Optimal and adaptive off-policy evaluation in contextual bandits”. In: *International Conference on Machine Learning (ICML)*. 2017.

[10] Zhouhao Yang et al. “Distributionally Robust Policy Gradient for Offline Contextual Bandits”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 6443–6462.

Appendix B

APPENDIX TO CHAPTER 3

B.1 Examples and Applications

Latent Perturbation Modelling

Understanding how the observed variables relate to a set of latent variables $s_t = (s_t(1), \ldots, s_t(k))$ has been extensively studied in machine learning. When the mixing function f is linear, standard statistical assumptions (see Corollary B.3) on s_t can guarantee the desired identifiability of f, up to multiplicative scaling and permutation. For the case when f is nonlinear, recent advances such as [10, 11, 14, 23, 25] in nonlinear independent component analysis (ICA) leverage additional assumptions on the latent variables in s_t and the mixing function f, to make the model identifiable. Below in Table B.1 we summarize the key assumptions (besides the common assumptions of mutual independence between latent variables and at most one latent variable is Gaussian) made in those models.

Table B.1: Standard assumptions on f and s for a subset of nonlinear ICA models.

Nonlinear ICA Models	Key Assumptions on f
Identifiable VAE [14]	Mixing function f is bijective and smooth
Contrastive learning [10, 11]	Mixing function f is bijective and smooth
Structural sparsity model [25]	Support of the Jacobian $\mathsf{J}_f(s)$ of f is sparse
Volume-preserving model [23]	Mixing function f is bijective and $\lvert\det\mathsf{J}_f(s)\rvert = 1$
Nonlinear ICA Models	**Key Assumptions on s**
Identifiable VAE [14]	$(s_t(1), \ldots, s_t(k))$ are conditionally independent given a variable u
Contrastive learning [10, 11]	$(s_t : t \in [T])$ is nonstationary or has temporal dependencies

Real-World Applications

Our model represents a classical control framework with extensive applicability across a variety of real-world decision-making problems. In the following section, we illustrate the versatility of our model through a selection of examples that demonstrate our dynamical model in equation 3.2 (see Section 3.2), subject to latent perturbations. These applications not only exemplify our model, but also set the stage for the comprehensive experimental results we discuss later in Section B.2, where we validate the model's practical efficacy in real-world scenarios.

Example B.1 (Piloting a Drone with mixed disturbances)**.** *Imagine the challenge of piloting a drone to follow an unpredictable path* $(y_t \in \mathbb{R}^2 : t \in [T])$ *on a windy and rainy day. This dynamic scenario can be mathematically represented in a discrete-time counterpart of the kinematic model [18, 24]:*

$$\begin{bmatrix} \Delta x_{t+1} \\ v_{t+1} \end{bmatrix} = A \begin{bmatrix} \Delta x_t \\ v_t \end{bmatrix} + Bu_t + Cw_t,$$

for some matrices $A \in \mathbb{R}^{4\times4}$*,* $B \in \mathbb{R}^{4\times2}$*, and* $C \in \mathbb{R}^{4\times4}$*. The column vector* $w_t = (w_t(1), w_t(2), w_t(3), w_t(4))$ *contains* 4 *latent variables where* $w_t(1) \coloneqq y_t(1) - y_{t+1}(1)$*;* $w_t(2) \coloneqq y_t(2) - y_{t+1}(2)$*;* $w_t(3)$ *is the environmental wind speed and* $w_t(4)$ *is the rainfall intensity at time* $t \in [T]$*. More details are discussed in Section 3.5 and Appendix B.2.*

Besides, it is well-known that any controllable system can be transformed into a canonical linear input-disturbed systems with dynamics described by $x_{t+1} = Ax_t + B(u_t + w_t)$ [6, 20] that can be used in many real-world applications with disturbances added to the control inputs. Now, we introduce another example in an electricity grid, with latent variables representing power injections from distinct buses of a distribution network.

Example B.2 (Heterogeneous Voltage Control)**.** *Consider a Volt/Var control task with the Simplified Distflow model in a distribution power network consisting of n branch buses and a substation bus [4, 16]:*

$$v_{t+1} = Xq_t^c + Gp_t + Xq_t^e + v_0\mathbf{1}_n, \tag{B.1}$$

wherein imaginary constants of the complex power injection at time $t \in [T]$ *are separated into two parts* $q_t^c \in \mathbb{R}^n$ *and* $q_t^e \in \mathbb{R}^n$*, representing the controllable reactive power injection and the uncontrollable external injection (perturbation) respectively. The fixed and known matrices* $X, G \in \mathbb{R}^{n\times n}$ *are positive definite and positive (Lemma 1 in [7]) with their entries depend on the line impedance and the grid topology. Our goal is to design a Volt/Var controller to regulate squared voltage magnitudes* $v_{t+1} = (v_{t+1}(1), \ldots, v_{t+1}(n))$ *to their nominal values by provisioning the reactive power injection* q_t^c *for each time step* $t \in [T]$*. More details are discussed in Section 3.5 and Appendix B.2.*

For the above example, we focus on a scenario with a set of n heterogeneous users corresponding to n branch nodes that generate a sequence of time-varying

active and reactive injections $(p_t(i) : i = 1, \ldots, n)$ and $(q_t^c(i) : i = 1, \ldots, n)$. The injection at the branch nodes $i = 1, \ldots, n$ may be generated by various types of renewable resources and power consumption patterns, e.g., solar/wind generations, residential, and commercial activities, energy storage management, or charging EVs, etc. Together they form a set of latent variables s_t in equation 3.2 with $k = n$ and in this special case, the mixing function $f(s_t; \theta) = Gp_t + Xq_t^e + v_0\mathbf{1}_n$ is affine.

In Section 3.5, 3.5, and Appendix B.2, we explore in greater detail the two examples previously discussed, and demonstrate the efficacy of our proposed method through case studies using real-world data.

Sample efficient ICA

There are sample efficient ICA algorithms for independent latent variables [5, 8, 22]. The following corollary can be derived from Theorem 3.7 and the sample complexity result in [22], which shows that $\overline{\eta}$ is sub-linear in T, thereby concluding the best-of-both-worlds prediction utilization claimed in Section 3.1.

Corollary B.3. *Suppose the mixing matrix θ is full column rank, and the latent variables $(s_t(i) : i = 1, \ldots, k)$ are mutually independent for all $t \in [T]$. Let $w = \Theta(\log T)$. Under the classic assumptions such that $\mathbb{E}[s_t(i)] = 0$, $\mathbb{E}\left[s_t^2(i)\right] = 1$, $\mathbb{E}\left[\left|s_t^5(i)\right|\right] \leq M$, and the fourth cumulant $|\mathrm{cum}_4(s_t(i))| \geq \Delta$ for all $t \in [T]$ and $i = 1, \ldots, k$, there exists a polynomial-time ICA algorithm that provides a sequence of mixing matrix estimates $(\widetilde{\theta}_t : t \in [T])$ of θ such that with high probability, the competitive ratio of Disc satisfies*

$$\mathsf{CR}(\textit{Disc}) \leq 1 + O\left(\sum_{i=1}^{k} \frac{\overline{\varepsilon}(i)}{T/\log T + \overline{\varepsilon}(i)}\right) + O\left(\log T\sqrt{\frac{k}{T}}\right) + O\left(\rho_F^{2w} + (\log n)^{\frac{7}{2}} k^{\frac{1}{2}} \frac{\log T}{T}\right).$$

Therefore, if for all $i \in \{1, \ldots, k\}$, $\overline{\varepsilon}(i)$ is negligible, CR(Disc) will converge to the optimal competitive ratio 1 with an increasing time horizon length T and a sufficiently large prediction window w (e.g., $w = \Theta(\log T)$); otherwise, for any $\overline{\varepsilon}(i) > 0$, CR(Disc) will be asymptotically bounded from above by $O(k) + 1$ where k is the number of latent variables. It is worth noting that the statistical assumptions on the latent variables $(s_t(i) : i = 1, \ldots, k)$ are classic in the ICA analysis [5, 8, 22].

B.2 Case Studies

In this section, we delve deeper into the practical applications of our main results by using two real-world examples (see Example B.1 and B.2) to demonstrate the

efficacy of DISC. We first describe our experimental setup in Section 3.5. We then demonstrate the effectiveness of the proposed method in Section 3.5, and compare DISC with several baselines. In Section 3.5, we show the desired resilience with respect to a non-stationary environment in a voltage control task.

Online Learning of Confidence Parameters. We illustrate the convergence of the confidence parameter ($\lambda_t : t \in [T]$) with respect the latent dimensions. In Figure B.1, we exemplify the convergence behavior of the confidence parameters associated with each of the four latent components shown in equation B.2 on the left column, together the time series for the four latent components on the right.

Notably, the first three components display learnable patterns while the last component, representing rainfall intensity is some unpredictable Gaussian noise. This distinction is reflected in the convergence behaviors of the corresponding confidence parameters: $\lambda(1), \lambda(2)$, and $\lambda(3)$, associated with the more predictable components, quickly converge to the optimal value of 1. Conversely, the last parameter $\lambda(4)$ associated with the erratic rainfall component exhibits a decreasing trend due to the unpredictability of the fourth component, highlighting the resilience of the proposed DISC policy against the prediction error.

Implementation Details of Experiment A

The drone's location at each moment, $p_{t+1} \in \mathbb{R}^2$, is determined by its current position and velocity $v_t \in \mathbb{R}^2$, evolving as $p_{t+1} = p_t + c_p v_t$. The drone controller can adjust its velocity at every time step through a control input u_t, leading to the updated velocity $v_{t+1} = v_t + c_v u_t$. The coefficients $c_p, c_v > 0$ are determined by control time intervals and physical parameters like the drone's weight and the drag coefficient. Defining $\Delta x_t \coloneqq p_t - y_t$ the location mismatch at time t, this piloting task can be mathematically represented in a linear control framework as shown in equation 3.2:

$$\mathbb{R}^4 \ni \begin{bmatrix} \Delta x_{t+1} \\ v_{t+1} \end{bmatrix} = A \begin{bmatrix} \Delta x_t \\ v_t \end{bmatrix} + Bu_t + \underbrace{\begin{bmatrix} y_t \\ \mathbf{0}_{2\times 1} \end{bmatrix} - \begin{bmatrix} y_{t+1} \\ \mathbf{0}_{2\times 1} \end{bmatrix} + w_t c_1 + r_t c_2}_{\text{4 latent components}}, \tag{B.2}$$

with system matrices $A \coloneqq \begin{bmatrix} 1 & 0 & c_p & 0 \\ 0 & 1 & 0 & c_p \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B \coloneqq \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ c_v & 0 \\ 0 & c_v \end{bmatrix}$.

The term $w_t c_1 + r_t c_2$ represents external perturbations from the environment. The

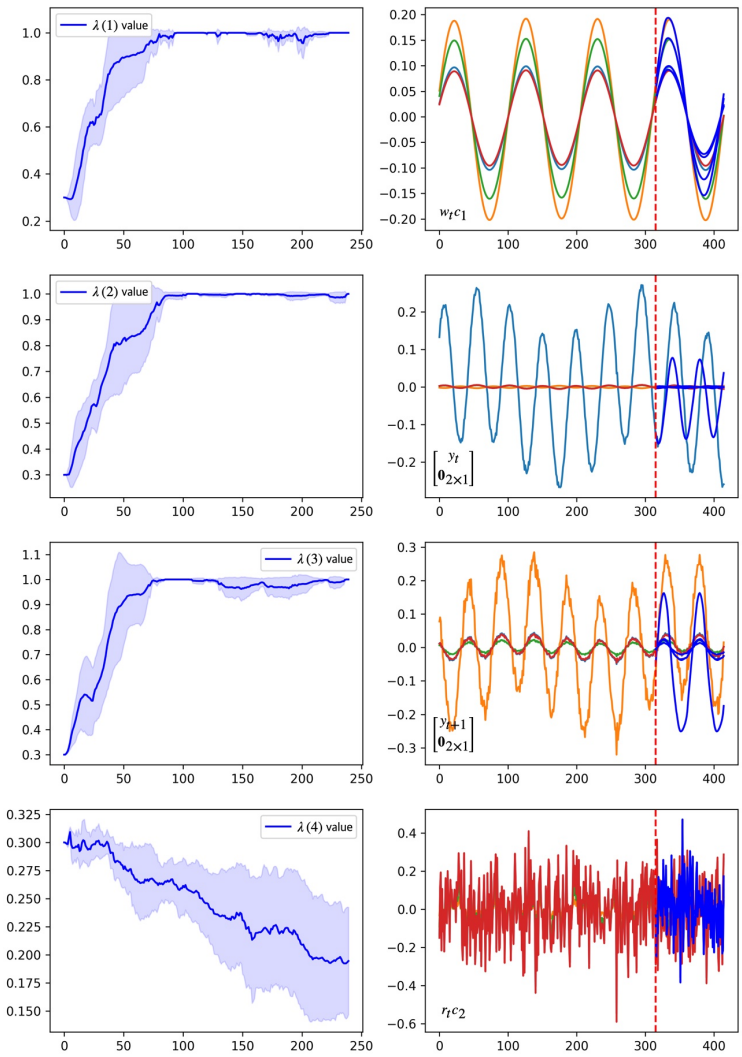


Figure B.1: **Example A.** *Drone Navigation (Section 3.5).* **Left column**: *Convergence of confidence parameters* $\lambda(1), \lambda(2), \lambda(3), \lambda(4)$ *corresponding to* 4 *latent components.* **Right column**: *Temporal dynamics of latent components* θs_t *in* $\mathbb{R}^4$. We illustrate the time series for the latent components in equation B.2, with each sub-figure featuring four distinct curves representing the dimensions of the respective 4-dimensional component. Post the dotted red line, the additional blue curves visualize imperfect ML predictions from the multi-layer neural network, elaborated in Appendix B.2. The x-axis in each graph marks the time steps, while the y-axis denotes the values of the time series.

column vectors $c_1, c_2 \in \mathbb{R}^4$ quantify the effects of the stochastic time-varying wind speed and rainfall intensity, represented by two time series $(w_t : t \in [T])$ and $(r_t : t \in [T])$. We assume the wind speed time series forms a sinusoidal function and the rain intensity time series is Gaussian, which are visualized together with the ML predictions in Figure B.1. Table B.2 in Appendix B.2 presents detailed choices of

the matrices A, B, Q, R and vectors c_1, c_2.

Additional Experimental Results

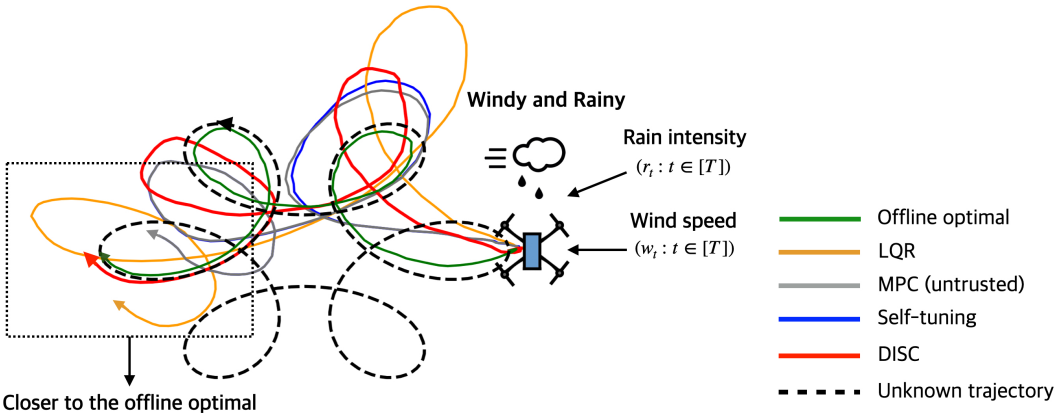


Figure B.2: Supplementary results showcasing drone navigation in windy and rainy conditions, further illustrating the scenarios discussed in Section 3.5 depicted in Figure 3.2. The external perturbations impacting the drone's flight are modeled by two independent, time-varying latent variables: w_t for wind speed and r_t for rain intensity at time $t \in [T]$. The path navigated by the drone using the Disc policy (detailed in Algorithm 2 in Section 3.3) is traced in red. Notably, this trajectory closely aligns the best with the offline optimal trajectory upon convergence, demonstrating the effectiveness of the Disc policy in adapting to complex environmental conditions.

We provide further experimental results that complement and extend the results presented in Section B.2.

In Figure B.2, we display the trajectories in the drone navigation task (Section 3.5) corresponding to Disc and the three baseline policies: the LQR, the self-tuning policy in [17], and the MPC with untrusted predictions (setting $\lambda = \mathbf{1}_k$ in equation 3.7). The trajectory for Disc closely aligns the best with the offline optimal trajectory upon convergence.

In Figure B.3, for the voltage control task (see Section 3.5), we compare the average total costs $J(\pi)$ (defined in equation 3.3) corresponding to Disc and the three baseline policies. Note that Disc consistently achieves the lowest average cost with the smallest variance computed from 5 random tests. The scaling parameter ν of the Gaussian component at node 8 is set as 1.25.

Detailed Setup and Hyper-Parameters

ML Model Setup We train an ML model online to generate predictions of $(s_\tau(i)\theta(i) : t \le \tau \le \bar{t})$ at each time $t \in [T]$. Here, $\theta(i)$ denotes the i-th column of the

true mixing parameter matrix θ. At each time step, the prediction model is trained and updated using previously collected disturbances. The warm-start buffer size is set as 100 and 50 respectively for the drone navigation and voltage control applications. In particular, we update the ML prediction model at each time step $t \in [T]$ and use a length-b subsequence of the collected time series $(s_\tau(i)\widetilde{\theta}_t(i) : \max\{0, \tau - b\} \leq \tau < t)$ as the input to predict the future $w = 5$ (w is the prediction window size defined in Section 3.2) steps of perturbations, as the output. Note that $\widetilde{\theta}_t(i)$ is the i-th column of the estimated mixing parameter matrix $\widetilde{\theta}_{t-1}$ provided by the FastICA method in [12] at time $t \in [T]$. All prediction models are formed via a fully-connected neural network with 4 hidden layers with a width 80 and LeakyReLU as the activation function except the final layer, and are trained using Adam [15] as the optimizer with a learning rate 1e$-$3 for 500 epochs.

Disentanglement Implementation A well-recognized issue in independent component analysis (ICA) and its nonlinear variants is the permutation indeterminacy of sources. This implies that the ordering of disentangled components can vary, such that the i-th component identified at one time step may not correspond to the i-th component in subsequent algorithm outputs. To mitigate this "permutation unidentifiability", our methodology involves pre-processing the time series data by ranking the components according to their sample entropy, as suggested in [19]. This step precedes the application of the ML prediction model, thereby enhancing the consistency of component identification across time steps.

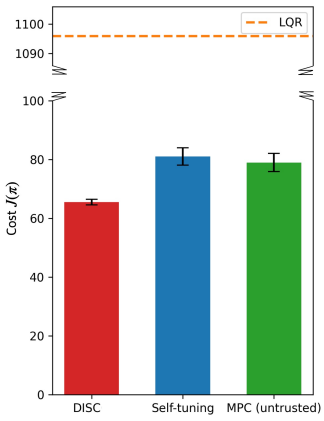


Figure B.3: Average total cost $J(\pi)$ with error bars for four policies in the voltage control task (Section 3.5).

Table B.2 below summarizes the detailed parameters used in Section B.2. In our experiments, the parameters are not subject to extensive optimization.

Detailed Setup in Section 3.5

The matrices X and G are generated from the following an impedance matrix $Z \in \mathbb{C}^{11\times 11}$ with the following non-zero entries:

$$
\begin{aligned}
&Z_{0,1} = Z_{1,0} = Z_{1,2} = Z_{2,1} = 1 + i,\\
&Z_{2,3} = Z_{3,2} = Z_{2,4} = Z_{4,2} = 1 + 2i,\\
&Z_{5,9} = Z_{9,5} = Z_{5,10} = Z_{10,5} = 2 + 4i,\\
&Z_{3,5} = Z_{5,3} = Z_{3,6} = Z_{6,3} = Z_{4,7} = Z_{7,4} = Z_{4,8} = Z_{8,4} = 2 + 2i,
\end{aligned}
$$

where i denotes the imaginary unit.

B.3 Useful Lemmas

We first present some preliminary results that will be used when proving our main theorems.

Parameter	Value
Time horizon length T	240 (Section 3.5), 200 (Section 3.5)
Warm start buffer length	100 (Section 3.5), 50 (Section 3.5)
State dimension n	4 (Section 3.5), 10 (Section 3.5)
Action dimension m	2 (Section 3.5), 10 (Section 3.5)
Prediction window size w	5
Coefficients c_v, c_p	0.2
Mixing Column c_1	$\frac{1}{8}(0.8, 1.6, 1.25, 0.75)^\top$
Mixing Column c_2	$\frac{1}{8}(1.2, 0.4, 0.75, 1.25)^\top$
Q, R (Section 3.5)	$\begin{bmatrix} I_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix}, I_{2\times2}$
Q, R (Section 3.5)	$I_{10\times10}, 0.1 \times I_{10\times10}$
Initial confidence parameter λ_0	0.3

(a) Basic Control Problem Setup in Section 3.5 and 3.5.

Hyper-Parameter	Value
Number of hidden layers	4
Hidden layer width	80
Covariate size b	5
Input dimension	$n \times b$
Output dimension	$n \times w$
Activation function	LeakyReLU
Optimizer	Adam [15]
Learning rate	1e−3

(b) Neural Network Hyper-parameters.

Table B.2: Parameters and hyper-parameters used in Section B.2.

Convexity of $\zeta_\ell(\lambda)$

Below we verify that $\zeta_\ell(\lambda)$ is a convex function of $\lambda \in [0,1]^k$. By definition,

$$\left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\left(\theta s_\ell - \widetilde{\theta}_\tau(\lambda \circ \widetilde{s}_{\ell|\tau})\right)\right]^\top H \left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\left(\theta s_\ell - \widetilde{\theta}_\tau(\lambda \circ \widetilde{s}_{\ell|\tau})\right)\right] \quad \text{(B.3)}$$

equals to (since H is symmetric)

$$\underbrace{\left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\theta s_\ell\right]^\top H \left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\theta s_\ell\right]}_{\text{Constant independent of } \lambda} + \left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\widetilde{\theta}_\tau(\lambda \circ \widetilde{s}_{\ell|\tau})\right]^\top$$

$$H\left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\widetilde{\theta}_\tau(\lambda \circ \widetilde{s}_{\ell|\tau})\right] - 2\left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\theta s_\ell\right]^\top H\left[\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\widetilde{\theta}_\tau(\lambda \circ \widetilde{s}_{\ell|\tau})\right].$$

Now, we simplify the remaining two terms. We first notice that $\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\widetilde{\theta}_\tau(\lambda \circ \widetilde{s}_{\ell|\tau}) = \Lambda\lambda$ for some matrix $\Lambda \in \mathbb{R}^{n\times k}$. To see this, denote $\phi_{\ell,\tau} := (F^\top)^{\ell-\tau}P\widetilde{\theta}_\tau \in \mathbb{R}^{n\times k}$. We get

$$\sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\widetilde{\theta}_\tau(\lambda \circ \widetilde{s}_{\ell|\tau}) = \sum_{\tau=\underline{\ell}}^{\ell}\phi_{\ell,\tau}(\lambda \circ \widetilde{s}_{\ell|\tau}) = \begin{bmatrix} \vdots & & \vdots \\ \phi_{\ell,\tau}(1) & \cdots & \phi_{\ell,\tau}(k) \\ \vdots & & \vdots \end{bmatrix}\begin{bmatrix} s_1\lambda_1 \\ \vdots \\ s_k\lambda_k \end{bmatrix}$$
$$= \underbrace{\begin{bmatrix} \vdots & & \vdots \\ s_1\phi_{\ell,\tau}(1) & \cdots & s_k\phi_{\ell,\tau}(k) \\ \vdots & & \vdots \end{bmatrix}}_{=:\Lambda}\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_k \end{bmatrix}.$$

Therefore, it suffices to validate that the matrix H is positive semi-definite to show $(\Lambda\lambda)^\top H(\Lambda\lambda) - 2\vartheta^\top H(\Lambda\lambda)$ is convex, where we denote $\vartheta := \sum_{\tau=\underline{\ell}}^{\ell}(F^\top)^{\ell-\tau}P\theta s_\ell \in \mathbb{R}^n$. Note that $H = B(R + B^\top PB)^{-1}B^\top$. Since R is positive definite by assumption and the DARE solution P is also positive definite, $(\Lambda\lambda)^\top H(\Lambda\lambda) - 2\vartheta^\top H(\Lambda\lambda)$ is a summation of a quadratic form and a linear function of λ, validating the convexity of the term in equation B.3. Following similar arguments, $\zeta_\ell(\lambda) = \sum_{\tau=\underline{\ell}}^{\ell} g(\tau,\ell)(\lambda)^\top Hg(\tau,\ell)(\lambda)$ is also convex.

A Generalized Lower Bound on $J^\star$

The following lemma is a generalized result from [17], which provides a lower bound on the offline optimal cost $J^\star$, as a function of system perturbations.

Lemma B.4 (Generalized Lower Bound on $J^\star$ [17])**.** *Let $\lambda_{\min}(Q)$ and $\lambda_{\min}(P)$ denote the smallest eigenvalues of positive definite matrices Q and P, respectively. It follows that*

$$J^\star \geq C_0 \sum_{t=0}^{T-1}\left(\sum_{\tau=t}^{T-1}\rho_F^{\tau-t}\,\|f(s_\tau;\theta)\|\right)^2 \tag{B.4}$$

for some constant $0 < C_0 \leq \frac{(1-\rho_F)^2}{2}\min\{\lambda_{\min}(P), \lambda_{\min}(R)/\|B\|, \lambda_{\min}(Q)/\max\{2,\|A\|\}\}$.

Next, we present the proof of Lemma 3.5.

Proof of Lemma 3.5

We restate the lemma below, which reduces the online learning objective $\sum_{\ell=0}^{t-1}\psi_{\ell,t}^\top(\lambda)H\psi_{\ell,t}(\lambda)$ to a canonical online optimization formulation.

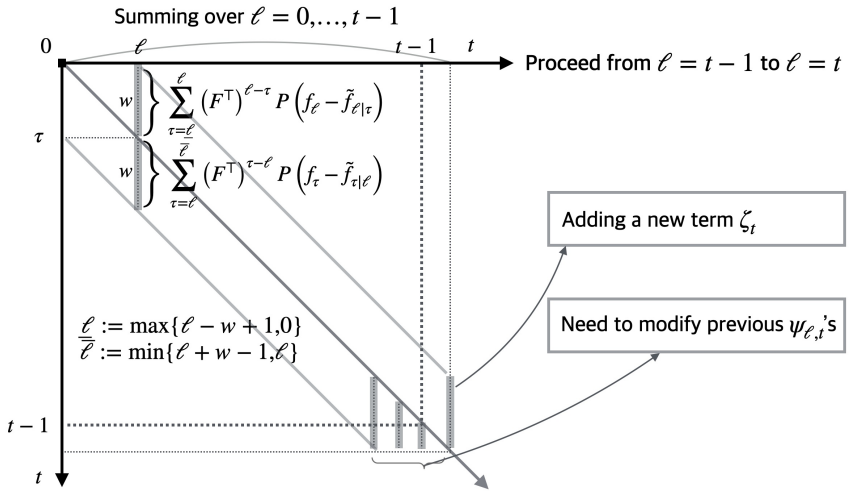


Figure B.4: Illustration of the equivalence in equation B.5. When ℓ increases from $t-1$ to t, the reduced form adds a new term ζ_t while the original form needs to update w previous functions $\psi_{\ell,t}$ for $\ell = t-w,\ldots,t-1$.

Lemma B.5. *Define $\zeta_\ell : \mathcal{I} \to \mathbb{R}^n$ as in equation 3.11. Then for any $t \in [T]$, it follows that*

$$\sum_{\ell=0}^{t-1} \psi_{\ell,t}^\top(\lambda) H \psi_{\ell,t}(\lambda) \le w \sum_{\ell=0}^{t-1} \zeta_\ell(\lambda).$$

Denote by $g(\tau,\ell) \coloneqq (F^\top)^{\tau-\ell} P\left(f\left(s_\tau;\theta_\tau\right) - f\left(\lambda \circ \widetilde{s}_{\tau|\ell};\widetilde{\theta}_\ell\right)\right)$. We first note that

$$\sum_{\ell=0}^{t-1} \sum_{\tau=\ell}^{\bar{\ell}} g(\tau,\ell)^\top H g(\tau,\ell) = \sum_{\tau=0}^{t-1} \sum_{\ell=\tau}^{\bar{\tau}} g(\ell,\tau)^\top H g(\ell,\tau) = \sum_{\tau=0}^{t-1} \zeta_\tau(\lambda). \tag{B.5}$$

Since $H = B(R + B^\top P B)^{-1} B^\top$ is positive semi-definite as we have shown in Appendix B.3, the function $h(x) \coloneqq x^\top H x$ is convex. Applying the Jensen's inequality,

$$h\left(\frac{1}{w}\sum_{\tau=\ell}^{\bar{\ell}} g(\tau,\ell)\right) \le \frac{1}{w}\sum_{\tau=\ell}^{\bar{\ell}} h\left(g(\tau,\ell)\right).$$

Therefore, equation B.5 implies

$$\sum_{\ell=0}^{t-1} \psi_{\ell,t}^\top(\lambda) H \psi_{\ell,t}(\lambda) \le w \sum_{\ell=0}^{t-1} \sum_{\tau=\ell}^{\bar{\ell}} g^\top(\tau,\ell) H g(\tau,\ell) = w \sum_{\tau=0}^{t-1} \zeta_\tau(\lambda).$$

The proof is illustrated in Figure B.4. We now proceed to show our main results. The key structure of the lemmas and theorems is outlined in Figure B.5.

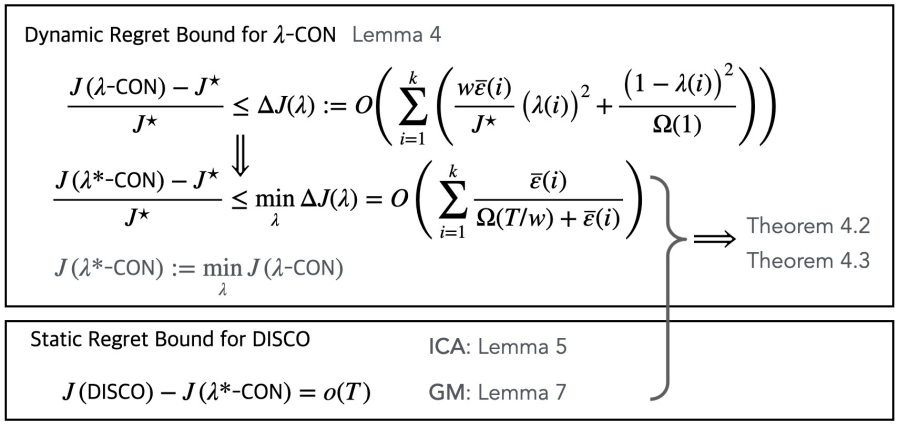


Figure B.5: Outline of key steps in the proofs of Theorem 3.7 and 3.8. Arrows denote implications.

B.4 Proof of Theorem 3.7

We first state a detailed version of Theorem 3.7 below with explicit multiplicative constants.

Theorem B.6 (Disc for Linear Mixing)**.** *Let $H \coloneqq B(R + B^\top P B)^{-1}B^\top$. With our model assumptions, the competitive ratio of Disc with a linear mixing function satisfies*

$$\mathsf{CR}(\textit{Disc}) \le 1 + \frac{C_1}{J^\star}\left(2\overline{\eta} + \left(\overline{\theta}\rho_F^w\right)^2 T\right) + 8C_2 w\left(\sum_{i=1}^{k}\left(\frac{\bar{\varepsilon}(i)}{C_0\sigma_{\min}^2(\theta)\bar{\varepsilon}(i) + J^\star}\right) + \left(\frac{\overline{s}^2}{J^\star}\sqrt{kT}\right)\right),$$

where $\sigma_{\min}(\theta) > 0$ is the smallest singular value of θ; $\overline{s}$ and $\overline{\theta}$ are defined in Section 3.4; C_0 is defined in Lemma B.4; $C_1 \coloneqq 2\|H\|\left(\frac{C_F}{1-\rho_F}\|P\|\overline{s}\right)^2$, and $C_2 \coloneqq \|H\|\left(C_F\|P\|\overline{\theta}\right)^2$.

We now proceed to prove Theorem B.6.

Step 1: Consistency-Robustness Analysis of λ-Con We first show the following lemma, which highlights a tradeoff between consistency and robustness. The following definition of dynamic regret will be used as an alternative worst-case performance metric in our main results. Dynamic regret is a more general (and often more challenging to analyze) measure than classical static regret, which has been mostly used for stationary environments [2, 3].

Definition B.7 (Dynamic regret)**.** The *dynamic regret* of a policy $\pi = (\pi_t : t \in [T])$ is defined as the difference between the quadratic cost induced by the policy π, $J(\pi)$ in equation 3.3, and the offline optimal cost $J^\star \coloneqq \inf_\pi J(\pi)$ subject to equation 3.2, i.e., $\mathsf{DR}(\pi) \coloneqq J(\pi) - J^\star$.

Lemma B.8 (Consistency-Robustness Lemma)**.** *With a fixed trust parameter $\lambda \in \mathcal{I}$, the disentangled λ-confident policy in Section 3.3 has a dynamic regret of at most*

$$2\|H\|(2\|P\|C_F\overline{s})^2\left(\overline{\eta}+\left(\overline{\theta}\rho_F^w\right)^2 T\right)+8\|H\|\sum_{t=0}^{T-1}\left(g_t^{con}+g_t^{rob}\right), \tag{B.6}$$

where $P, C_F, \overline{\eta}, \rho_F$ are defined in Section 3.2; $\overline{s}$ and $\overline{\theta}$ are defined in Section 3.4. The terms g_t^{con} and g_t^{rob} are functions of λ, defined as

$$g_t^{con} \coloneqq \left\|\sum_{\tau=t}^{\overline{T}}\left(F^\top\right)^{\tau-t} P\left(\widetilde{\theta}_t\left(\lambda\circ\varepsilon_{\tau|t}\right)\right)\right\|^2, \quad g_t^{rob} \coloneqq \left\|\sum_{\tau=t}^{\overline{T}}\left(F^\top\right)^{\tau-t} P\left(\widetilde{\theta}_t\left((\mathbf{1}_k-\lambda)\circ s_{\tau|t}\right)\right)\right\|^2.$$

Proof of Lemma B.8. Denote by λ-Con the λ-confident policy, and $J\left(\lambda\text{-Con}\right)$ the corresponding total cost induced by taking actions $(u_0,\ldots,u_{T-1})$ generated by λ-Con. Similarly, denote by $J^\star$ the optimal total cost. Denote $\overline{T} \coloneqq \min\{t+w-1, T-1\}$. Lemma 3 in [17] implies the following explicit form of $J\left(\lambda\text{-Con}\right) - J^\star$:

$$\begin{aligned}&\sum_{t=0}^{T-1}\left(\sum_{\tau=t}^{T-1}\left(F^\top\right)^{\tau-t} P\left(\theta_\tau s_\tau-\kappa(\tau)\widetilde{\theta}_t\left(\lambda\circ\widetilde{s}_{\tau|t}\right)\right)\right)^\top H\\ &\quad\cdot\left(\sum_{\tau=t}^{T-1}\left(F^\top\right)^{\tau-t} P\left(\theta_\tau s_\tau-\kappa(\tau)\widetilde{\theta}_t\left(\lambda\circ\widetilde{s}_{\tau|t}\right)\right)\right),\end{aligned} \tag{B.7}$$

where $\kappa(\tau)\equiv 0$ when $\tau > \overline{T}$. The sum of quadratic costs in equation B.7 can be further bounded by

$$\begin{aligned}&J\left(\lambda\text{-Con}\right)-J^\star\\ \le&\|H\|\sum_{t=0}^{T-1}\left\|\sum_{\tau=t}^{T-1}\left(F^\top\right)^{\tau-t} P\left(\theta s_\tau-\kappa(\tau)\widetilde{\theta}_t\left(\lambda\circ\widetilde{s}_{\tau|t}\right)\right)\right\|^2\\ \le&2\|H\|\sum_{t=0}^{T-1}\Bigg(\underbrace{\left\|\sum_{\tau=t}^{\overline{T}}\left(F^\top\right)^{\tau-t} P\left(\theta s_\tau-\widetilde{\theta}_t\left(\lambda\circ\widetilde{s}_{\tau|t}\right)\right)\right\|^2}_{\text{short-term error } g_t^{\text{error}} \text{ (linear setting)}}+\underbrace{\left\|\sum_{\tau=t+w}^{T-1}\left(F^\top\right)^{\tau-t} P\theta s_\tau\right\|^2}_{\text{long-term error } h_t^{\text{error}} \text{ (linear setting)}}\Bigg).\end{aligned} \tag{B.8}$$

In above, the first term characterizes a total cost gap induced by inaccrurate ICA identifiability and predictions. The second term in equation B.8 is the long-term error for disentangled times series predictions that are at least w steps away from the current time steps, denoted by h_t^{error}, which becomes 0 when $t+w \ge T-1$. It follows that

$$h_t^{\text{error}} \le \left(\sum_{\tau=t+w}^{T-1} C_F\rho_F^{\tau-t}\|P\|\overline{\theta}\overline{s}\right)^2 \le \left(\frac{C_F}{1-\rho_F}\|P\|\overline{\theta}\overline{s}\right)^2\rho_F^{2w}, \tag{B.9}$$

since by our assumption on F, the Gelfand's formula implies that there must exist a constant $C_F > 0$, $\rho_F \in (0,1)$ s.t. $\|F^t\| \leq C_F \rho_F^t$ for all $t \geq 0$. Moreover, for g_t^{error}, we can bound it from above by

$$\begin{aligned} g_t^{\text{error}} =& \left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \left(\theta s_\tau - \widetilde{\theta}_t s_\tau + \widetilde{\theta}_t s_\tau - \widetilde{\theta}_t \left(\lambda \circ \widetilde{s}_{\tau|t}\right) \right) \right\|^2 \\ \leq& 2 \left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \eta_t s_\tau \right\|^2 + 2 \left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \left(\widetilde{\theta}_t \left(s_\tau - \lambda \circ \widetilde{s}_{\tau|t} \right) \right) \right\|^2, \end{aligned} \tag{B.10}$$

where $\eta_t \coloneqq \widetilde{\theta}_t - \theta$ is the mixing parameter error in equation 3.4 defined Section 3.2. Furthermore, noting $\left(\varepsilon_{\tau|t}(1), \ldots, \varepsilon_{\tau|t}(k)\right) = \varepsilon_{\tau|t} \coloneqq s_\tau - \widetilde{s}_{\tau|t}$, we obtain

$$\begin{aligned} g_t^{\text{error}} \leq& 2 \left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \eta_t s_\tau \right\|^2 + 2 \left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \left(\widetilde{\theta}_t \left(\lambda \circ \varepsilon_{\tau|t} + (\mathbf{1}_k - \lambda) \circ s_\tau \right) \right) \right\|^2 \\ \leq& 2 \left(\sum_{\tau=t}^{\overline{T}} C_F \rho_F^{\tau-t} \left\| P \eta_t s_\tau \right\| \right)^2 + 4 \underbrace{\left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \left(\widetilde{\theta}_t \left(\lambda \circ \varepsilon_{\tau|t} \right) \right) \right\|^2}_{\text{consistency error } g_t^{\text{con}} \text{ (ICA setting)}} \\ & + 4 \underbrace{\left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \left(\widetilde{\theta}_t \left((\mathbf{1}_k - \lambda) \circ s_\tau \right) \right) \right\|^2}_{\text{robustness error } g_t^{\text{rob}} \text{ (ICA setting)}}. \end{aligned} \tag{B.11}$$

The right hand side of equation B.11 contains two types of errors - the consistency error, denoted by g_t^{con} that occurs due to the estimated time series; and the robustness error, denoted by $g^{\text{rob}}(t)$ that arises when λ becomes small. Therefore, combining equation B.8, equation B.9, and equation B.11 implies the following upper bound on $J\left(\lambda\text{-Con}\right) - J^\star$:

$$\begin{aligned} & J\left(\lambda\text{-Con}\right) - J^\star \leq 2\|H\| \sum_{t=0}^{T-1} \left(g_t^{\text{error}} + h_t^{\text{error}} \right) \\ \leq& 4\|H\| \left(\frac{C_F}{1-\rho_F} \|P\| \overline{s} \right)^2 \overline{\eta} + 2\|H\| T \left(\frac{C_F}{1-\rho_F} \|P\| \overline{\theta} \overline{s} \right)^2 \rho_F^{2w} + 8\|H\| \sum_{t=0}^{T-1} \left(g_t^{\text{con}} + g_t^{\text{rob}} \right), \end{aligned}$$

where $\overline{\eta}$ is defined in equation 3.4.

□

Step 2: Component-Wise Consistency Bound on g_t^{con} In the sequel, we bound g_t^{con} and $g^{\text{rob}}(t)$ respectively. First, the consistency error caused by inaccurate time

series predictions can be bounded by

$$g_t^{\text{con}} \leq \left(\sum_{\tau=t}^{\overline{T}} C_F \rho_F^{\tau-t} \overline{\theta} \|P\| \left\| \lambda \circ \varepsilon_{\tau|t} \right\| \right)^2 \leq w \left(C_F \|P\| \overline{\theta} \right)^2 \sum_{\tau=t}^{\overline{T}} \rho_F^{2(\tau-t)} \left\| \lambda \circ \varepsilon_{\tau|t} \right\|^2,$$

which can be further decomposed into a summation of component-wise latent variable time-series prediction errors by denoting $\overline{\varepsilon}(i) \coloneqq \sum_{t=0}^{T-1} \sum_{\tau=t}^{\overline{T}} (\rho_F^{(\tau-t)} \varepsilon_{\tau|t}(i))^2$ where $\varepsilon_{\tau|t} = (\varepsilon_{\tau|t}(i) : i = 1, \ldots, k)$, as defined in Section 3.2. Considering $\lambda = (\lambda(1), \ldots, \lambda(k))$, it follows that

$$\begin{aligned} \sum_{t=0}^{T-1} g_t^{\text{con}} \leq & w \left(C_F \|P\| \overline{\theta} \right)^2 \sum_{t=0}^{T-1} \sum_{\tau=t}^{\overline{T}} \rho_F^{2(\tau-t)} \left\| \lambda \circ \varepsilon_{\tau|t} \right\|^2 \\ \leq & w \left(C_F \|P\| \overline{\theta} \right)^2 \sum_{t=0}^{T-1} \sum_{\tau=t}^{\overline{T}} \rho_F^{2(\tau-t)} \left\| \lambda \circ \varepsilon_{\tau|t} \right\|^2 \\ = & w \left(C_F \|P\| \overline{\theta} \right)^2 \sum_{i=1}^{k} (\lambda(i))^2 \, \overline{\varepsilon}(i). \end{aligned} \tag{B.12}$$

Step 3: Robustness bound on g_t^{rob} Next, we consider deriving a bound on g_t^{rob}. We obatin

$$g_t^{\text{rob}} = \left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top \right)^{\tau-t} P \left(\widetilde{\theta}_\tau \left((\mathbf{1}_k - \lambda) \circ s_\tau \right) \right) \right\|^2 \leq \left(C_F \|P\| \overline{\theta} \right)^2 \left(\sum_{\tau=t}^{\overline{T}} \rho_F^{\tau-t} \left\| (\mathbf{1}_k - \lambda) \circ s_\tau \right\| \right)^2,$$

which leads to

$$g_t^{\text{rob}} \leq \left(C_F \|P\| \overline{\theta} \right)^2 \|(\mathbf{1}_k - \lambda)\|_4^2 \left(\sum_{\tau=t}^{\overline{T}} \rho_F^{\tau-t} \|s_\tau\|_4 \right)^2 \tag{B.13}$$

by applying the Cauchy–Schwarz inequality where $\|\cdot\|_4$ denotes the ℓ_4-norm. Denote by $\sigma_{\min}(\theta) > 0$ the smallest singular value of the full column rank mixing matrix θ. Noting that $\|s_\tau\|_4 \leq \|s_\tau\| \leq \frac{1}{\sigma_{\min}(\theta)} \|\theta s_\tau\|$ for all τ, and $f(s; \theta) = \theta s$ in this linear mixing setting, applying Lemma B.4,

$$\frac{1}{J^\star} \sum_{t=0}^{T-1} g_t^{\text{rob}} \leq \frac{\left(C_F \|P\| \overline{\theta} \right)^2 \|(\mathbf{1}_k - \lambda)\|_4^2}{C_0 \sigma_{\min}^2(\theta)} \tag{B.14}$$

for some constant $0 < C_0 \leq \frac{(1-\rho_F)^2}{2} \min\{\lambda_{\min}(P), \lambda_{\min}(R)/\|B\|, \lambda_{\min}(Q)/\max\{2, \|A\|\}\}$.

Step 4: Competitive Analysis of λ-Con Putting together Lemma B.8, equation B.12, and equation B.14, since $J^\star > 0$,

$$\frac{J\left(\lambda\text{-Con}\right) - J^\star}{J^\star} \leq \frac{1}{J^\star}\left(2\|H\|\left(2\|P\|C_f\overline{s}\right)^2\left(\overline{\eta} + \left(\overline{\theta}\rho_F^w\right)^2 T\right)\right) + \frac{8\|H\|}{J^\star}\sum_{t=0}^{T-1}\left(g_t^{\text{con}} + g_t^{\text{rob}}\right), \tag{B.15}$$

where the last term can be further bounded from above by

$$\begin{aligned}\frac{8\|H\|}{J^\star}\sum_{t=0}^{T-1}\left(g_t^{\text{con}} + g_t^{\text{rob}}\right) \leq & 8\|H\|\left(C_F\|P\|\overline{\theta}\right)^2\left(\frac{w}{J^\star}\sum_{i=1}^{k}\left(\lambda(i)\right)^2\overline{\varepsilon}(i) + \frac{\|(\mathbf{1}_k - \lambda)\|_4^2}{C_0\sigma_{\min}^2(\theta)}\right) \\ \leq & 8\|H\|\left(C_F\|P\|\overline{\theta}\right)^2\left(\sum_{i=1}^{k}\left(\frac{w\overline{\varepsilon}(i)}{J^\star}\left(\lambda(i)\right)^2 + \frac{\left(1-\lambda(i)\right)^2}{C_0\sigma_{\min}^2(\theta)}\right)\right).\end{aligned} \tag{B.16}$$

Step 5: Online Learning of λ_t via FTRL Consider a follow-the-regularized-leader (FTRL) optimization with an ℓ_2-regularizer $\|\lambda - \lambda_0\|^2$. The following lemma can be proved based on the equivalent online convex optimization (OCO) form of the online mirror descent (OMD) (see the implementation described in Section 3.5) above as shown in Lemma 3.5. We reprise the OMD steps below.

$$\tau_t = \tau_{t-1} - \frac{\beta}{2}\nabla_\lambda\left(\zeta_t\right), \tag{B.17a}$$

$$\lambda_t = \mathsf{Proj}_{\mathcal{I}}(\tau_t). \tag{B.17b}$$

Now, fix $\lambda = (\lambda_t : t \in [T])$ generated by Disc. Let us consider a pseudo-cost $\Delta J^\dagger(\lambda)$ defined below, which is compared with the true objective $\Delta J\left(\lambda\right)$ of Disc.

$$\Delta J^\dagger(\lambda) \coloneqq \sum_{\ell=0}^{T-1}\sum_{\tau=\underline{\ell}}^{\ell}\left[(F^\top)^{\ell-\tau}P\left(\theta s_\ell - \widetilde{\theta}_\tau(\lambda_\ell \circ \widetilde{s}_{\ell|\tau})\right)\right]^\top H\left[(F^\top)^{\ell-\tau}P\left(\theta s_\ell - \widetilde{\theta}_\tau(\lambda_\ell \circ \widetilde{s}_{\ell|\tau})\right)\right]$$

$$\Delta J\left(\lambda\right) = \sum_{\ell=0}^{T-1}\sum_{\tau=\underline{\ell}}^{\ell}\left[(F^\top)^{\ell-\tau}P\left(\theta s_\ell - \widetilde{\theta}_\tau(\lambda_\tau \circ \widetilde{s}_{\ell|\tau})\right)\right]^\top H\left[(F^\top)^{\ell-\tau}P\left(\theta s_\ell - \widetilde{\theta}_\tau(\lambda_\tau \circ \widetilde{s}_{\ell|\tau})\right)\right]$$

The lemma in the following holds.

Lemma B.9. *The online mirror descent (OMD) equation B.17a-equation B.17b generates a sequence $\lambda = (\lambda_0, \ldots, \lambda_{T-1})$ in Disc such that*

$$\Delta J^\dagger - \min_{\lambda\in\mathcal{I}}\Delta J\left(\lambda\right) \leq 8\|H\|\left(\overline{s}\overline{\theta}\|P\|\frac{C_f}{1-\rho_F}\right)^2\sqrt{kT}.$$

Proof of Lemma B.9. The gradient of the cost gap at time $t \in [T]$ satisfies

$$\begin{aligned}\nabla_\lambda\left(\zeta_t\right) =& \sum_{t=0}^{T-1}\sum_{\tau=\underline{\ell}}^{\ell} \mathsf{J}^\top(\tau,t)\left(H+H^\top\right) g(\tau,t)\\ =& 2\sum_{t=0}^{T-1}\sum_{\tau=\underline{\ell}}^{\ell} \mathsf{J}^\top\left(\tau,t\right) H\left(F^\top\right)^{t-\tau} P\left(\theta s_t - \widetilde{\theta}_\tau\left(\lambda\circ\widetilde{s}_{t|\tau}\right)\right), \end{aligned} \tag{B.18}$$

where in equation B.18 we have used the fact that $H \coloneqq B(R+B^\top PB)^{-1}B^\top$ is symmetric. Moreover, noting that $\widetilde{s}_{t|\tau} = \left(\widetilde{s}_{t|\tau}(1),\ldots,\widetilde{s}_{t|\tau}(k)\right)$, in equation B.18 above,

$$\mathsf{J}\left(\tau,t\right) \coloneqq -\left(F^\top\right)^{t-\tau} P\widetilde{\theta}_\tau \begin{bmatrix} \widetilde{s}_{t|\tau}(1) & & \\ & \ddots & \\ & & \widetilde{s}_{t|\tau}(k)\end{bmatrix}$$

denotes an $n\times k$ Jacobian matrix $\mathsf{J}\left(\tau,t\right)$ of $g(\tau,t)$ with respect to λ whose (i,j)-th entry is $\frac{\partial g(\tau,t)^{(i)}}{\partial\lambda^{(j)}}$. Normalizing the predicted time series yields that

$$\|\nabla_\lambda\left(\zeta_t\right)\| \le 4\|H\|\left(\overline{s}\overline{\theta}\|P\|\frac{C_f}{1-\rho_F}\right)^2.$$

As we have verified in Appendix B.3, each $g(\tau,t)^\top(\lambda)Hg(\tau,t)$ is a convex function of $\lambda\in\mathcal{I}$. Therefore, based on the static regret analysis for OCO [9, 13], FTRL with an ℓ_2-regularizer guarantees (after optimizing β) that

$$\begin{aligned}&\Delta J^\dagger(\lambda) - \min_{\lambda\in\mathcal{I}}\Delta J\left(\lambda\right)\\ \le& 2\left\|\nabla_\lambda\left(\zeta_t\right)\right\|\max_{\lambda\in\mathcal{I}}\|\lambda-\lambda_0\|\sqrt{T}\\ \le& 8\|H\|\left(\overline{s}\overline{\theta}\|P\|\frac{C_f}{1-\rho_F}\right)^2\sqrt{kT}. \end{aligned} \tag{B.19}$$

□

Applying Assumption 3.4 and noting that $\lambda_\ell^2-\lambda_{\ell-\tau}^2 \le O(|\lambda_\ell-\lambda_{\ell-\tau}|)$ for all $\ell\in[T]$, we conclude that $\Delta J\left(\lambda\right) - \Delta J^\dagger(\lambda) = o(T)$, since for τ that is not a constant, the exponent $\rho^\tau = o(1)$.

Denote by λ^* the optimal solution that minimizes J (λ-Con). Using B.15 and equation B.19 above, we conclude that

$$\frac{J(\textsc{Disc})}{J^\star} = 1 + \underbrace{\frac{J\left(\lambda^*\text{-Con}\right)-J^\star}{J^\star}}_{\text{Lemma B.8 and equation B.16}} + \underbrace{\frac{J(\textsc{Disc})-J\left(\lambda^*\text{-Con}\right)}{J^\star}}_{\text{Lemma B.9}},$$

whose right hand side can be bounded respectively via Lemma B.8 and Lemma B.9. Furthermore, noting that $f(s_t;\theta) \neq 0$ for all $t \in [T]$, hence $\|\theta s_t\| > 0$. As a result, Lemma B.4 implies $J^\star = \Omega(T)$. Therefore, using Lemma 3.5,

$$\begin{aligned}\frac{J(\textsc{Disc})}{J^\star} \leq 1 &+ 2\|H\|\left(2\|P\|\frac{C_f}{1-\rho_F}\overline{s}\right)^2 \frac{1}{J^\star}\left(\overline{\eta} + \left(\overline{\theta}\rho_F^w\right)^2 T\right) \\ &+ 8\|H\|\left(C_F\|P\|\overline{\theta}\right)^2 \min_{\lambda\in\mathcal{I}}\left(\sum_{i=1}^{k}\left(\frac{w\overline{\varepsilon}(i)}{J^\star}(\lambda(i))^2 + \frac{(1-\lambda(i))^2}{C_0\sigma_{\min}^2(\theta)}\right)\right) \\ &+ 8\|H\|\left(\|P\|\frac{C_f}{1-\rho_F}\overline{s}\overline{\theta}\right)^2 \frac{w\sqrt{kT}}{J^\star} + o(1).\end{aligned}$$

After optimizing over $\lambda \in \mathcal{I}$,

$$\begin{aligned}\frac{J(\textsc{Disc})}{J^\star} \leq 1 &+ 2\|H\|\left(2\|P\|C_f\overline{s}\right)^2 \frac{1}{J^\star}\left(\overline{\eta} + \left(\overline{\theta}\rho_F^w\right)^2 T\right) \\ &+ 8\|H\|\left(C_F\|P\|\overline{\theta}\right)^2 \left(\sum_{i=1}^{k}\left(\frac{w\overline{\varepsilon}(i)}{\sigma_{\min}^2(\theta)C_0 w\overline{\varepsilon}(i) + J^\star}\right)\right) \\ &+ 8\|H\|\left(\|P\|\frac{C_f}{1-\rho_F}\overline{s}\overline{\theta}\right)^2 \frac{w\sqrt{kT}}{J^\star} + o(1).\end{aligned}$$

B.5 Proof of Corollary B.3

Under the classic assumptions such that $\mathbb{E}[s_t(i)] = 0$, $\mathbb{E}\left[s_t^2(i)\right] = 1$, $\mathbb{E}\left[\left|s_t^5(i)\right|\right] \leq M$, and the fourth cumulant $|\mathrm{cum}_4(s_t(i))| \geq \Delta$ for all $t \in [T]$ and $i = 1,\ldots,k$, the Recursive Fourier PCA algorithm in [3] is a polynomial-time and it guarantees (c.f. Theorem 1 in [3]) that for all $t \in [T]$,

$$\|\eta_t\| = \|\theta - \theta_t\| \leq \|\theta - \theta_t\|_F \leq \frac{\overline{\theta}\overline{s}(\log n)^{7/2}M^{1/2}k^{1/2}}{\Delta^3 t^{1/2}},$$

with high probability, which yields $\overline{\eta} = O\left((\log n)^{7/2}k^{1/2}\sum_{t\in[T]}\frac{1}{t}\right) = O\left((\log n)^{7/2}k^{1/2}\log T\right)$. Therefore, with high probability, the competitive ratio bound in the corollary holds.

B.6 Proof of Theorem 3.6

Denote $\overline{T} \coloneqq \min\{\ell + w - 1, T-1\}$. Similar to the analysis in the proof of Theorem 3.7 in Appendix B.4, we obtain

$$\begin{aligned}&J\,(\lambda\text{-}\textsc{Con}) - J^\star \\ &= \sum_{\ell=0}^{T-1}\left(\sum_{\tau=\ell}^{\overline{T}}\left(F^\top\right)^{\tau-\ell}P\left(f_\tau - \widetilde{f}_{\tau|\ell}(\lambda)\right)\right)^\top H\left(\sum_{\tau=\ell}^{\overline{T}}\left(F^\top\right)^{\tau-\ell}P\left(f_\tau - \widetilde{f}_{\tau|\ell}(\lambda)\right)\right). \quad \text{(B.20)}\end{aligned}$$

WLOG, we assume f is linear such that $f(s;\theta) = \theta s$ for some $n \times k$ matrix $\theta = (\theta_{ij})$ and the estimated mixing matrix is accurate so that $\overline{\eta} = 0$. Then it follows that

$$J\left(\lambda\text{-Con}\right) - J^\star \geq \lambda_{\min}\left((R + B^\top P B)^{-1}\right) \sum_{\ell=0}^{T-1} \left\| B^\top \psi_{\ell,T} \right\|^2,$$

where

$$\psi_{\ell,T} \coloneqq \sum_{\tau=\ell}^{\overline{T}} \left(F^\top\right)^{\tau-\ell} P \theta_\tau \left(\lambda \circ \varepsilon_{\tau|\ell} - (\mathbf{1}_k - \lambda) \circ s_{\tau|\ell}\right). \tag{B.21}$$

Since λ-Con is $(1 + o(1))$-consistent, we must have $J\left(\lambda\text{-Con}\right) - J^\star \leq o(1)J^\star$ (by our model assumption that $s_t \neq 0$ for all $t \in [T]$, $J^\star > 0$). Combining this with equation B.21 above, it is necessary to have

$$o(1)J^\star \geq \lambda_{\min}\left((R + B^\top P B)^{-1}\right) \sum_{t=0}^{T-1} \left\| B^\top \sum_{\tau=\ell}^{\overline{T}} \left(F^\top\right)^{\tau-\ell} P \theta_\tau \left((\mathbf{1}_k - \lambda) \circ s_{\tau|\ell}\right) \right\|^2. \tag{B.22}$$

Since the competitive ratio is the worst-case ratio for all A, B, Q, R, f, θ, and $(s_t : t \in [T])$, we must have $\lambda = \mathbf{1}_k$ to guarantee equation B.22 above.

Now, consider an adaptive offline adversary that generates $\varepsilon_{\tau|\ell} = \mu s_{\tau|\ell}$ for all $\ell \leq \tau \leq \overline{\ell}$. Since $\lambda = \mathbf{1}_k$, we obtain

$$J\left(\lambda\text{-Con}\right) - J^\star \geq \mu^2 \lambda_{\min}\left((R + B^\top P B)^{-1}\right) \sum_{t=0}^{T-1} \left\| B^\top \sum_{\tau=\ell}^{\overline{T}} \left(F^\top\right)^{\tau-\ell} P \theta_\tau s_{\tau|\ell} \right\|^2.$$

Setting $\mu = \log T$, above implies $\mathsf{CR}\left(\lambda\text{-Con}\right) = \omega(1)$.

B.7 Proof of Theorem 3.8

We first state a detailed version of Theorem 3.7 below with explicit multiplicative constants.

Step 1: Consistency-Robustness Analysis of λ-Con We first show the following lemma, which highlights a tradeoff between consistency and robustness.

Lemma B.10. *With a fixed trust parameter $\lambda \in \mathcal{I}$, the disentangled λ-confident control (Disc) has a worst-case dynamic regret of at most*

$$4\|H\|\left(\frac{C_F}{1-\rho_F} C_f \|P\|\right)^2 \overline{\eta} + 2\|H\| T \left(\frac{C_F}{1-\rho_F} C_f \|P\| f_{\max}\right)^2 \rho_F^{2w} + 8\|H\| \sum_{t=0}^{T-1} \left(g_t^{con} + g_t^{rob}\right) \tag{B.23}$$

where $H \coloneqq B(R+B^\top PB)^{-1}B^\top$*;* $f_{\max} \coloneqq f_0 + C_f\overline{s}$ *with* $f_0 \coloneqq |f(0)|$*;* P, C_F, ρ_F, w *are defined in Section 3.2;* C_f *denotes the Lipschitz constant of* f*. The terms* g_t^{con} *and* g_t^{rob} *are functions of* λ*, defined as*

$$g_t^{con}(\lambda) \coloneqq \left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\left\|\lambda \circ \varepsilon_{\tau|t}\right\|\right)^2,$$

$$g_t^{rob}(\lambda) \coloneqq \left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\|(\mathbf{1}_k-\lambda)\circ s_\tau\|\right)^2.$$

Proof of Lemma B.10. Similar to the analysis in the proof of Theorem 3.7 in Appendix B.4, we obtain

$$\begin{aligned}
&J\left(\lambda\text{-Con}\right) - J^\star \\
=&\sum_{t=0}^{T-1}\left(\sum_{\tau=t}^{T-1}\left(F^\top\right)^{\tau-t} P\left(f_\tau - \widetilde{f}_{\tau|t}(\lambda)\right)\right)^\top H\left(\sum_{\tau=t}^{T-1}\left(F^\top\right)^{\tau-t} P\left(f_\tau - \widetilde{f}_{\tau|t}(\lambda)\right)\right),
\end{aligned} \tag{B.24}$$

where we simplify the latent perturbations and the corresponding predictions as $f_\tau \coloneqq f\left(s_\tau;\theta\right)$ and $\widetilde{f}_{\tau|t}(\lambda) \coloneqq f\left(\lambda\circ\widetilde{s}_{\tau|t};\widetilde{\theta}_t\right)$. Especially, $\widetilde{f}_{\tau|t}(\lambda) \coloneqq 0$ when $\tau > \overline{T}$. Thus, the total cost gap can be bounded by

$$\begin{aligned}
&J\left(\lambda\text{-Con}\right) - J^\star \\
\le&\|H\|\sum_{t=0}^{T-1}\left\|\sum_{\tau=t}^{T-1}\left(F^\top\right)^{\tau-t} P\left(f_\tau - \widetilde{f}_{\tau|t}(\lambda)\right)\right\|^2 \\
\le&2\|H\|\sum_{t=0}^{T-1}\Bigg(\underbrace{\left\|\sum_{\tau=t}^{\overline{T}}\left(F^\top\right)^{\tau-t} P\left(f_\tau - \widetilde{f}_{\tau|t}(\lambda)\right)\right\|^2}_{\text{short-term error } g_t^{\text{error}} \text{ (general mixing)}} + \underbrace{\left\|\sum_{\tau=t+w}^{T-1}\left(F^\top\right)^{\tau-t} P f_\tau\right\|^2}_{\text{long-term error } h_t^{\text{error}} \text{ (general mixing)}}\Bigg).
\end{aligned} \tag{B.25}$$

Note that by Assumption 3.1 f is continuous with a Lipschitz constant $C_f > 0$, hence, f is bounded, since $\|s_\tau\| \le \overline{s}$. Denote by $f_{\max} \le f_0 + C_f\overline{s}$ the largest value of f where $f_0 \coloneqq |f(0)|$. The second term h_t^{error} in equation B.8 can be bounded similarly to equation B.9 in the proof of Theorem 3.7 (see Appendix B.4):

$$h_t^{\text{error}} \le \left(\sum_{\tau=t+w}^{T-1} C_F\rho_F^{\tau-t}C_f\|P\|f_{\max}\right)^2 \le \left(\frac{C_F}{1-\rho_F}C_f\|P\|f_{\max}\right)^2\rho_F^{2w}, \tag{B.26}$$

The first term g_t^{error} in equation B.8 characterizes a total cost gap induced by inaccurate mixing matrix estimation and time series prediction errors. It follows

that

$$g_t^{\text{error}} = \left\| \sum_{\tau=t}^{\overline{T}} \left(F^\top\right)^{\tau-t} P \left(f\left(s_\tau;\theta\right) - f\left(s_\tau;\widetilde{\theta}_t\right) + f\left(s_\tau;\widetilde{\theta}_t\right) - f\left(\lambda \circ \widetilde{s}_{\tau|t};\widetilde{\theta}_t\right)\right)\right\|^2$$
$$\leq 2\left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\left\|\theta - \widetilde{\theta}_t\right\|\right)^2 + 2\left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\left\|s_\tau - \lambda \circ \widetilde{s}_{\tau|t}\right\|\right)^2, \tag{B.27}$$

where we have used the fact that $\left\|F^t\right\| \leq C_F \rho_F^t$ for all $t \in [T]$ with a spectral radius $\rho_F \in (0,1)$.

Recall that $\eta_t \coloneqq \widetilde{\theta}_t - \theta$ is the mixing parameter error in equation 3.4 defined Section 3.2. Furthermore, noting $\left(\varepsilon_{\tau|t}(1),\ldots,\varepsilon_{\tau|t}(k)\right) = \varepsilon_{\tau|t} \coloneqq s_\tau - \widetilde{s}_{\tau|t}$, continuing from equation B.27, we get

$$g_t^{\text{error}} \leq 2\left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\eta_t\right)^2 + 2\left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\left\|s_\tau - \lambda \circ \widetilde{s}_{\tau|t}\right\|\right)^2$$
$$\leq 2\left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\eta_t\right)^2 + \underbrace{4\left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\left\|\lambda \circ \varepsilon_{\tau|t}\right\|\right)^2}_{\text{consistency error } g_t^{\text{con}} \text{ (general mixing)}}$$
$$+ \underbrace{4\left(\sum_{\tau=t}^{\overline{T}} C_F C_f \rho_F^{\tau-t}\|P\|\|(\mathbf{1}_k - \lambda) \circ s_\tau\|\right)^2}_{\text{robustness error } g_t^{\text{rob}} \text{ (general mixing)}}. \tag{B.28}$$

The right hand side of equation B.28 contains two types of errors - the consistency error, denoted by g_t^{con} that occurs due to the estimated time series; and the robustness error, denoted by $g^{\text{rob}}(t)$ that arises when λ becomes small. Therefore, rearranging the terms and combining equation B.25, equation B.26, and equation B.28 imply the following upper bound on $J\left(\lambda\text{-Con}\right) - J^\star$:

$$J\left(\lambda\text{-Con}\right) - J^\star \leq 2\|H\|\sum_{t=0}^{T-1}\left(g_t^{\text{error}} + h_t^{\text{error}}\right)$$
$$\leq 4\|H\|\left(\frac{C_F}{1-\rho_F} C_f \|P\|\right)^2 \overline{\eta} + 2\|H\| T\left(\frac{C_F}{1-\rho_F} C_f \|P\| f_{\max}\right)^2 \rho_F^{2w} + 8\|H\|\sum_{t=0}^{T-1}\left(g_t^{\text{con}} + g_t^{\text{rob}}\right), \tag{B.29}$$

with $\overline{\eta}$ defined in equation 3.4. □

Furthermore, we can bound the terms g_t^{error} and h_t^{error} separately as:

$$\sum_{t=0}^{T-1} g_t^{\text{con}} \leq w \left(C_F C_f \|P\|\right)^2 \sum_{i=1}^{k} \left(\lambda(i)\right)^2 \overline{\varepsilon}(i), \tag{B.30}$$

and

$$\frac{1}{J^\star} \sum_{t=0}^{T-1} g_t^{\text{rob}} \leq \frac{\left(C_F C_f \|P\|\right)^2 \|(\mathbf{1}_k - \lambda)\|_4^2}{C_0 \left\|f^{-1}\right\|^2}, \tag{B.31}$$

where f^{-1} is the inverse function of the mixing function f. Since by Assumption 3.1, f is bijective, f^{-1} must exist. Thus, equation B.30 and equation B.31 together imply

$$\frac{1}{J^\star} \sum_{t=0}^{T-1} \left(g_t^{\text{con}} + g_t^{\text{rob}}\right) \leq \left(C_F C_f \|P\|\right)^2 \left(\sum_{i=1}^{k} \left(\frac{w\overline{\varepsilon}(i)}{J^\star} \left(\lambda(i)\right)^2 + \frac{(1-\lambda(i))^2}{C_0 \left\|f^{-1}\right\|^2}\right)\right). \tag{B.32}$$

Step 2: Online Learning of λ_t via FTPL Consider a follow-the-perturbed-leader (FTPL) optimization [1, 21], with:

$$\lambda_t \in \arg\min_{\lambda \in \mathcal{I}} \left(\sum_{\ell=0}^{t-1} \zeta_\ell(\lambda) + \sigma_t^\top \lambda\right) \quad \text{(FTPL for } \lambda\text{-learning)} \tag{B.33}$$

The following lemma is a result of [21] by fixing each coordinate $\sigma_t(i)$ $(i = 1, \ldots, k)$ an IID random variable drawn from an exponential distribution with some parameter $\alpha > 0$.

Lemma B.11. *The FTPL procedure equation B.33 above generates a sequence $\lambda = (\lambda_0, \ldots, \lambda_{T-1})$ in Disc such that*

$$\mathbb{E}\left[\frac{1}{T}\left(\Delta J^\dagger(\lambda) - \min_{\lambda \in \mathcal{I}} \Delta J\left((\lambda)\right)\right)\right] \leq O\left(\alpha k^{5/2} C_f^2 + \frac{k^{3/2}}{\alpha T}\right). \tag{B.34}$$

Similarly to the proof of Theorem 3.7, applying Assumption 3.4, $J\left(\text{Disc}\right) - J^\dagger = o(T)$. Denote by λ^* the optimal solution that minimizes $J\left(\lambda\text{-Con}\right)$. Putting together equation B.29, equation B.32, and equation B.34, we conclude

$$\mathbb{E}\left[\frac{J\left(\text{Disc}\right)}{J^\star}\right] = 1 + \underbrace{\mathbb{E}\left[\frac{J\left(\lambda^*\text{-Con}\right) - J^\star}{J^\star}\right]}_{\text{Lemma B.10 and Equation equation B.32}} + \underbrace{\mathbb{E}\left[\frac{J\left(\text{Disc}\right) - J\left(\lambda^*\text{-Con}\right)}{J^\star}\right]}_{\text{Lemma B.11}},$$

yielding the following after optimizing over $\alpha > 0$:

$$\mathbb{E}\left[\frac{J\,(\textsc{Disc})}{J^\star}\right] \leq 1+\frac{C_3}{J^\star}\left(4\overline{\eta}+2T\left(f_{\max}\rho_F^w\right)^2\right)+8C_4\min_{\lambda\in\mathcal{I}}\left(\sum_{i=1}^{k}\left(\frac{w\overline{\varepsilon}(i)}{J^\star}\left(\lambda(i)\right)^2+\frac{\left(1-\lambda(i)\right)^2}{C_0\left\|f^{-1}\right\|^2}\right)\right)$$
$$+O\left(k^2C_f\frac{w\sqrt{T}}{J^\star}\right) \tag{B.35}$$
$$\leq 1+O\left(\rho_F^{2w}+\frac{\overline{\eta}}{T}\right)+8C_4\left(\sum_{i=1}^{k}\left(\frac{w\overline{\varepsilon}(i)}{C_0\left\|f^{-1}\right\|^2 w\overline{\varepsilon}(i)+J^\star}\right)\right)+O\left(\frac{wk^2}{\sqrt{T}}\right), \tag{B.36}$$

where we denote $C_3 \coloneqq \|H\|\left(\frac{C_F}{1-\rho_F}C_f\|P\|\right)^2$, $C_4 \coloneqq \|H\|\left(C_FC_f\|P\|\right)^2$ and have used $J^\star = \Omega(T)$ in equation B.35 (implied by Lemma B.4 and the assumption that f is non-zero and Lipschitz continuous, similar to the proof of Theorem 3.7 in Appendix B.4) to derive equation B.36.

References

[1] Naman Agarwal, Alon Gonen, and Elad Hazan. "Learning in non-convex games with an optimization oracle". In: *Conference on Learning Theory*. PMLR. 2019, pp. 18–29.

[2] Peter Auer, Thomas Jaksch, and Ronald Ortner. "Near-optimal regret bounds for reinforcement learning". In: *Advances in neural information processing systems* 21 (2008).

[3] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. "Minimax regret bounds for reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 263–272.

[4] Mesut E Baran and Felix F Wu. "Network reconfiguration in distribution systems for loss reduction and load balancing". In: *IEEE Transactions on Power delivery* 4.2 (1989), pp. 1401–1407.

[5] Mikhail Belkin, Luis Rademacher, and James Voss. "Blind signal separation in the presence of Gaussian noise". In: *Conference on Learning Theory*. PMLR. 2013, pp. 270–287.

[6] Wen-Hua Chen et al. "A nonlinear disturbance observer for robotic manipulators". In: *IEEE Transactions on industrial Electronics* 47.4 (2000), pp. 932–938.

[7] Masoud Farivar, Lijun Chen, and Steven Low. "Equilibrium and dynamics of local voltage control in distribution systems". In: *52nd IEEE Conference on Decision and Control*. IEEE. 2013, pp. 4329–4334.

[8] Navin Goyal, Santosh Vempala, and Ying Xiao. "Fourier PCA and robust tensor decomposition". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 2014, pp. 584–593.

[9] Elad Hazan. *Introduction to online convex optimization*. MIT Press, 2022.

[10] Aapo Hyvarinen and Hiroshi Morioka. "Nonlinear ICA of temporally dependent stationary sources". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 460–469.

[11] Aapo Hyvarinen and Hiroshi Morioka. "Unsupervised feature extraction by time-contrastive learning and nonlinear ica". In: *Advances in neural information processing systems* 29 (2016).

[12] Aapo Hyvärinen and Erkki Oja. "Independent component analysis: algorithms and applications". In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[13] Adam Kalai and Santosh Vempala. "Efficient algorithms for online decision problems". In: *Journal of Computer and System Sciences* 71.3 (2005), pp. 291–307.

[14] Ilyes Khemakhem et al. "Variational autoencoders and nonlinear ica: A unifying framework". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2207–2217.

[15] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[16] Na Li, Guannan Qu, and Munther Dahleh. "Real-time decentralized voltage control in distribution networks". In: *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2014, pp. 582–588.

[17] Tongxin Li et al. "Robustness and Consistency in Linear Quadratic Control with Untrusted Predictions". In: *ACM SIGMETRICS Performance Evaluation Review* 50.1 (2022), pp. 107–108.

[18] Yingying Li, Xin Chen, and Na Li. "Online optimal control with linear dynamics and predictions: algorithms and regret analysis". In: *NeurIPS*. 2019, pp. 14858–14870.

[19] Joshua S Richman and J Randall Moorman. "Physiological time-series analysis using approximate entropy and sample entropy". In: *American journal of physiology-heart and circulatory physiology* 278.6 (2000), H2039–H2049.

[20] Guanya Shi et al. "Online optimization with memory and competitive control". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20636–20647.

[21] Arun Sai Suggala and Praneeth Netrapalli. "Online non-convex learning: Following the perturbed leader is optimal". In: *Algorithmic Learning Theory*. PMLR. 2020, pp. 845–861.

[22] Santosh S Vempala and Ying Xiao. “Max vs min: Tensor decomposition and ICA with nearly linear sample complexity”. In: *Conference on Learning Theory*. PMLR. 2015, pp. 1710–1723.

[23] Xiaojiang Yang et al. “Nonlinear ica using volume-preserving transformations”. In: *International Conference on Learning Representations*. 2021.

[24] Chenkai Yu et al. “The Power of Predictions in Online Control”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[25] Yujia Zheng, Ignavier Ng, and Kun Zhang. “On the identifiability of nonlinear ica: Sparsity and beyond”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16411–16422.

Appendix C

APPENDIX TO CHAPTER 4

C.1 Task statistics

We show the statistics of the dataset used in our experiments in Table.

Table C.1: Task Statistics

Task	Dataset	# of train	# of test
Math	Precalculus	198	156
	Geometry	474	237
	Count & prob.	483	291
	Number	780	497
	Algebra	1065	881
	Prealgebra	814	636
	Intermediate	607	503
TableQA	TabMWP	500	500

C.2 Experimental Results

Different base language model

We here show the full results for Count and TabMWP in the Dynamic Tool-Using setting using Mistral-7B in Table C.2. Our approach can still deliver improvement when applied to a different base language model.

Static Tool-Using results

We here show the full results of Static Tool-Using settings on each dataset in Table C.3.

Static Tool-Using results of different variants

We here show the full results of Static Tool-Using settings of different variants on each dataset in Table C.4 and C.5.

Dynamic Tool-Using results on each dataset

We here show the full experimental results on each dataset in the dynamic tool-using setting in Table C.6.

Table C.2: Results for Count and TabMWP in Dynamic Tool-Using setting using Mistral-7B. Our approach can still deliver improvement when applied to a different base language model.

Method	**Count**							
#Samples	1	2	3	4	5	10	15	20
TROVE	17.46	21.64	23.90	25.20	25.70	28.14	28.87	29.27
PROBECAL-PROMPT	20.10	23.57	25.13	26.36	27.04	28.75	29.80	30.02
PROBECAL-TRACE	17.46	22.73	25.17	26.49	27.06	28.51	28.99	29.05
PROBECAL-PROMPT&TRACE	20.10	24.63	26.57	27.57	27.66	28.93	29.61	29.67
Method	**TabMWP**							
#Samples	1	2	3	4	5	10	15	20
TROVE	33.02	45.36	51.28	54.41	56.60	60.69	61.89	61.92
PROBECAL-PROMPT	48.26	55.60	58.48	59.98	60.70	62.33	63.08	63.14
PROBECAL-TRACE	33.02	46.63	53.23	56.73	59.08	63.83	65.13	65.29
PROBECAL-PROMPT&TRACE	48.26	56.97	60.38	61.98	62.87	64.90	65.68	65.94

Dynamic Tool-Using results of different variants on each dataset

We here show the full experimental results of different variants on each dataset in the dynamic tool-using setting in Table C.7 and C.8.

Static Tool-Using ECE results

We here show the full experimental results of ECE loss on each dataset in the static tool-using settings in Table C.9

Dynamic Tool-Using ECE results

We here show the full experimental results of ECE loss on each dataset in the static tool-using settings in Table C.10.

C.3 More Experimental Results

We here show additional experimental results on the effect of the size of the training set, the performance of other language models and verbal confidence in Table C.11, C.12 and C.13.

C.4 Calibration Curve

We show examples of the calibration curve of PROBECAL and the LLM logits for each dataset in both static and dynamic settings. The calibration curve for PROBECAL-PROMPT&TRACE is computed between the average of the PROBECAL-PROMPT and PROBECAL-TRACE logits, and the label.

In each figure of calibration curve, the X-axis represents the predicted probability

(model confidence), the Y axis represents the accuracy, the red line (y=x) demonstrates perfect calibration, the blue curve is the model's calibration curve, and the grey histogram shows the number of instances within each confidence bin.

Static tool-using calibration curve

We here show the calibration curve of ProbeCal and LLM logit including Sort and E.S.L. for each dataset in the static tool-using setting.

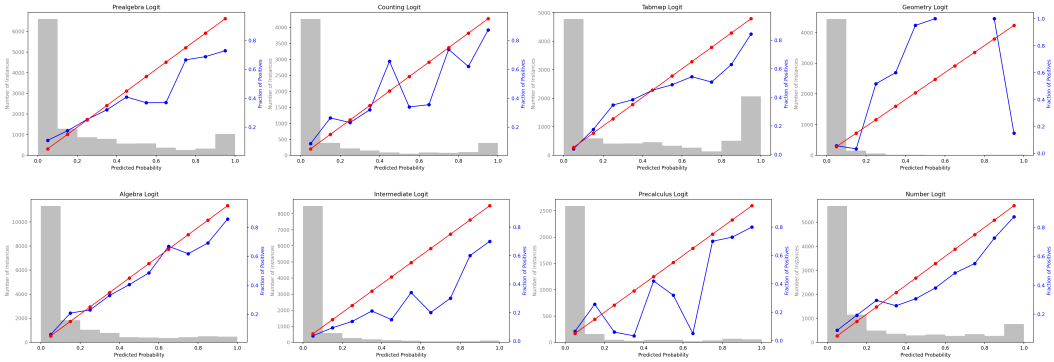


Figure C.1: Calibration curve of ProbeCal-Prompt logits of LLM in Static Tool-Using

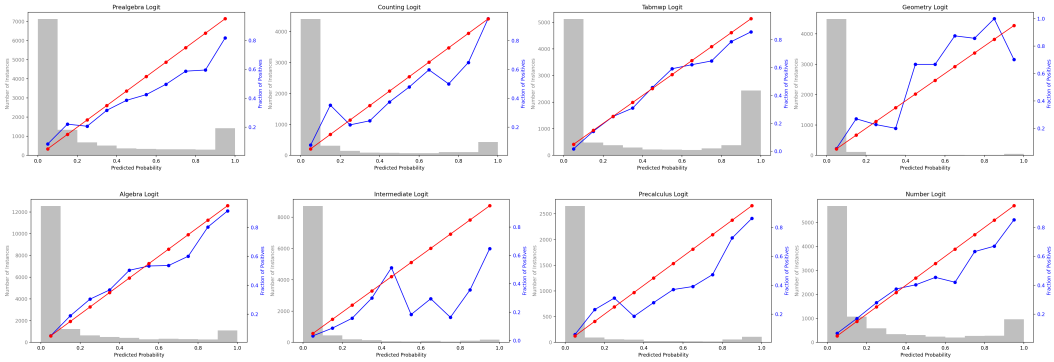


Figure C.2: Calibration curve of ProbeCal-Trace logits of LLM in Static Tool-Using

Dynamic tool-using calibration curve

We here show the calibration curve of ProbeCal and LLM logit including Sort and E.S.L. for each dataset in the dynamic tool-using setting.

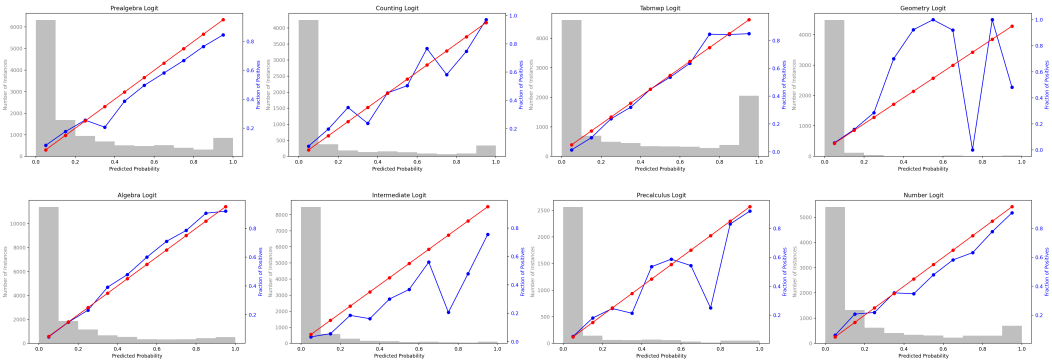


Figure C.3: Calibration curve of PROBECAL-PROMPT&TRACE logits of LLM in Static Tool-Using

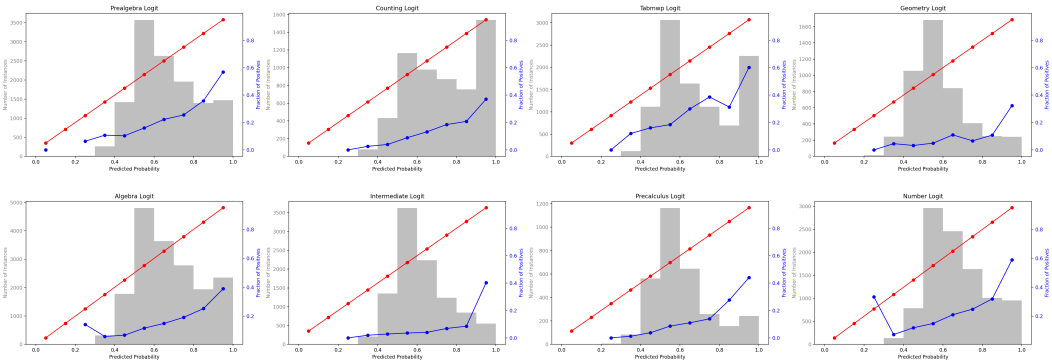


Figure C.4: Calibration curve of SORT logits of LLM in Static Tool-Using

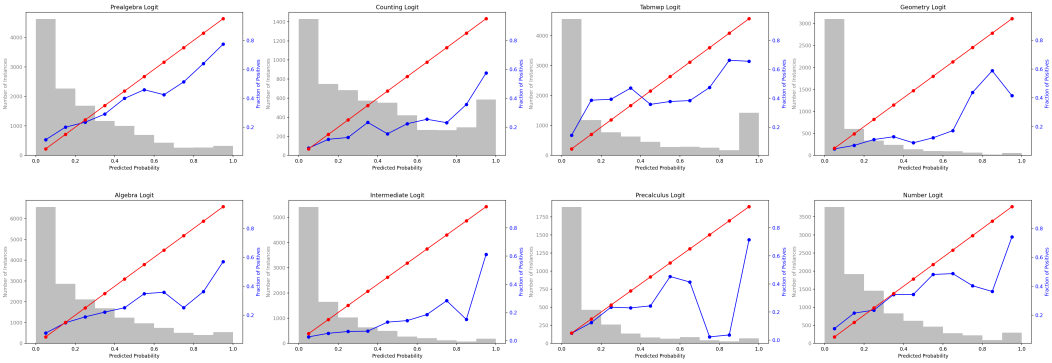


Figure C.5: Calibration curve of E.S.L. logits of LLM in Static Tool-Using

Table C.3: Static Tool-Using results on each dataset

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	7	9	10
Instance	24.67	27.88	29.96	31.00	31.71	32.74	32.87	33.15
Instance-Sample	25.30	27.68	30.09	31.12	31.92	32.46	32.94	33.14
Instance-Sort	24.67	28.20	30.09	31.08	31.92	32.52	32.72	32.97
Instance-E.S.L.	24.67	28.44	29.72	30.55	31.04	31.57	31.57	31.78
ProbeCal-Prompt	26.07	28.71	30.38	31.37	31.89	32.58	33.19	33.34
ProbeCal-Trace	24.67	28.82	30.90	32.03	32.91	33.65	34.04	34.19
ProbeCal-Prompt&Trace	26.07	29.65	31.36	32.33	32.95	33.67	34.21	34.50
Dataset	**Count**							
# Samples	1	2	3	4	5	7	9	10
Instance	19.66	21.34	23.46	24.47	25.13	26.11	26.63	26.98
Instance-Sample	20.50	21.72	23.81	24.18	25.26	25.97	26.83	26.50
Instance-Sort	19.66	22.45	24.27	24.93	25.59	26.23	26.64	27.05
Instance-E.S.L.	19.66	23.60	24.65	25.79	26.08	26.61	26.84	27.42
ProbeCal-Prompt	23.16	24.71	26.62	27.15	27.59	28.49	29.22	29.13
ProbeCal-Trace	19.66	23.66	25.20	26.54	26.76	27.51	27.81	27.88
ProbeCal-Prompt&Trace	23.16	25.53	26.62	27.26	27.39	27.88	28.08	28.16
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	7	9	10
Instance	32.23	41.47	45.22	47.56	48.94	50.64	51.41	51.83
Instance-Sample	35.78	44.42	47.47	49.36	50.40	51.76	52.40	52.77
Instance-Sort	32.23	41.32	45.16	47.38	49.00	50.76	51.59	52.19
Instance-E.S.L.	32.23	41.49	45.94	48.41	49.94	51.84	52.70	53.26
ProbeCal-Prompt	42.06	47.52	49.56	50.85	51.68	53.02	53.80	54.04
ProbeCal-Trace	32.23	42.36	47.50	50.64	52.66	55.06	56.30	57.02
ProbeCal-Prompt&Trace	42.06	49.18	51.76	53.58	54.95	56.79	57.73	58.11
Dataset	**Geometry**							
# Samples	1	2	3	4	5	7	9	10
Instance	7.34	8.43	9.38	9.79	10.30	10.74	11.22	11.28
Instance-Sample	7.41	8.95	9.54	9.89	10.44	10.51	11.12	11.22
Instance-Sort	7.34	8.64	9.65	10.03	10.30	10.67	10.96	11.15
Instance-E.S.L.	7.34	8.53	9.42	9.60	9.72	9.69	9.83	9.95
ProbeCal-Prompt	8.64	9.60	10.24	10.51	10.85	11.10	11.29	11.47
ProbeCal-Trace	7.34	8.97	10.07	10.54	10.79	11.29	11.30	11.38
ProbeCal-Prompt&Trace	8.64	9.79	10.69	10.78	11.19	11.01	11.23	11.15
Dataset	**Algebra**							
# Samples	1	2	3	4	5	7	9	10
Instance	18.13	21.96	24.47	25.99	26.58	28.01	28.96	29.28
Instance-Sample	18.73	22.52	24.84	26.25	27.19	28.50	29.28	29.72
Instance-Sort	18.13	22.22	24.77	26.24	27.03	28.28	29.18	29.47
Instance-E.S.L.	18.13	22.32	24.71	26.00	26.59	27.67	28.28	28.44
ProbeCal-Prompt	20.27	23.78	25.98	27.29	28.06	29.21	30.02	30.49
ProbeCal-Trace	18.13	23.09	25.80	27.54	28.29	29.79	30.89	31.18
ProbeCal-Prompt&Trace	20.27	24.89	27.02	28.42	29.28	30.54	31.64	32.04
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	7	9	10
Instance	6.34	9.84	11.29	12.79	13.37	14.47	14.93	15.23
Instance-Sample	6.27	9.71	11.42	12.70	13.46	14.37	15.03	15.17
Instance-Sort	6.34	9.91	11.44	12.97	13.48	14.61	15.03	15.44
Instance-E.S.L.	6.34	9.93	11.53	13.19	13.74	14.78	15.14	15.55
ProbeCal-Prompt	7.86	10.53	12.22	13.08	13.78	14.96	15.73	15.85
ProbeCal-Trace	6.34	9.96	11.52	13.00	13.81	14.84	15.61	15.99
ProbeCal-Prompt&Trace	7.86	10.56	12.34	13.28	14.02	15.22	16.22	16.32
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	7	9	10
Instance	12.14	16.00	18.33	19.99	20.86	22.08	23.10	23.18
Instance-Sample	13.50	17.67	19.83	20.55	22.18	23.13	24.00	24.32
Instance-Sort	12.14	16.32	19.17	21.21	21.97	23.04	23.47	23.60
Instance-E.S.L.	12.14	16.94	19.21	21.29	21.63	22.96	23.22	23.58
ProbeCal-Prompt	15.05	18.09	19.78	21.14	22.17	23.05	24.47	24.69
ProbeCal-Trace	12.14	16.86	19.12	20.85	21.36	22.46	22.77	22.86
ProbeCal-Prompt&Trace	15.05	18.50	19.85	20.68	21.62	21.86	22.81	22.76
Dataset	**Number**							
# Samples	1	2	3	4	5	7	9	10
Instance	23.76	25.08	27.46	28.55	29.51	30.31	31.33	31.50
Instance-Sample	24.03	25.70	27.90	29.16	30.19	30.89	31.78	31.73
Instance-Sort	23.76	26.58	28.54	29.37	30.07	30.70	31.45	31.61
Instance-E.S.L.	23.76	26.77	28.43	29.06	29.59	30.11	30.61	30.88
ProbeCal-Prompt	24.72	26.32	28.36	29.31	30.11	30.99	31.69	31.73
ProbeCal-Trace	23.76	27.84	29.89	30.92	31.64	32.62	33.42	33.56
ProbeCal-Prompt&Trace	24.72	28.58	30.16	31.31	31.80	32.92	33.42	33.65

Table C.4: Static Tool-Using results of different variants on each dataset - I

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	26.07	28.71	30.38	31.37	31.89	32.58	33.19	33.34
ProbeCal-Trace	24.67	28.82	30.90	32.03	32.91	33.65	34.04	34.19
ProbeCal-Prompt&Trace	26.07	29.65	31.36	32.33	32.95	33.67	34.21	34.50
ProbeCal-Prompt-TS	25.91	28.36	30.40	31.38	31.97	32.70	33.18	33.49
ProbeCal-Trace-TS	24.67	28.82	30.89	32.06	32.87	33.64	34.02	34.19
ProbeCal-Prompt&Trace-TS	25.91	29.38	31.26	32.02	32.78	33.68	34.07	34.32
ProbeCal-Prompt-Weight	26.15	29.04	30.61	31.58	32.01	32.83	33.31	33.47
ProbeCal-Trace-Weight	24.67	28.85	31.02	32.16	32.99	33.77	34.09	34.34
ProbeCal-Prompt&Trace-Weight	26.15	29.95	31.51	32.33	32.91	33.64	34.26	34.58
ProbeCal-Prompt-Sort	25.95	28.64	30.32	31.31	31.95	32.60	33.21	33.31
ProbeCal-Trace-Sort	24.67	28.64	30.84	31.97	32.91	33.84	34.28	34.46
ProbeCal-Prompt&Trace-Sort	25.95	29.55	31.13	32.32	32.70	33.69	34.13	34.22
ProbeCal-Prompt-E.S.L.	25.26	28.35	29.98	30.96	31.38	32.35	32.85	32.81
ProbeCal-Trace-E.S.L.	24.67	28.40	30.19	31.30	32.04	33.14	33.17	33.64
ProbeCal-Prompt&Trace-E.S.L.	25.26	29.26	30.58	31.69	32.16	32.94	33.46	33.60
Dataset	**Count**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	23.16	24.71	26.62	27.15	27.59	28.49	29.22	29.13
ProbeCal-Trace	19.66	23.66	25.20	26.54	26.76	27.51	27.81	27.88
ProbeCal-Prompt&Trace	23.16	25.53	26.62	27.26	27.39	27.88	28.08	28.16
ProbeCal-Prompt-TS	22.96	24.63	26.41	26.96	27.80	28.38	28.97	29.09
ProbeCal-Trace-TS	19.66	23.66	25.21	26.49	26.72	27.42	27.70	27.81
ProbeCal-Prompt&Trace-TS	22.96	25.67	26.84	26.88	27.42	28.09	28.26	28.03
ProbeCal-Prompt-Weight	23.03	24.05	26.22	27.12	28.00	28.38	28.87	29.10
ProbeCal-Trace-Weight	19.66	23.62	25.13	26.30	26.49	27.53	27.59	27.69
ProbeCal-Prompt&Trace-Weight	23.03	25.51	26.62	27.41	27.81	27.86	28.12	28.18
ProbeCal-Prompt-Sort	22.92	23.78	25.94	26.56	27.20	27.66	28.25	28.47
ProbeCal-Trace-Sort	19.66	23.49	24.94	25.99	26.14	26.99	27.30	27.31
ProbeCal-Prompt&Trace-Sort	22.92	25.26	25.95	26.03	26.74	27.17	27.13	27.06
ProbeCal-Prompt-E.S.L.	22.16	22.87	25.44	25.83	26.42	27.18	28.01	28.37
ProbeCal-Trace-E.S.L.	19.66	22.99	24.05	24.42	24.74	25.01	25.13	25.48
ProbeCal-Prompt&Trace-E.S.L.	22.16	23.86	24.72	24.94	24.85	25.15	25.25	25.62
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	42.06	47.52	49.56	50.85	51.68	53.02	53.80	54.04
ProbeCal-Trace	32.23	42.36	47.50	50.64	52.66	55.06	56.30	57.02
ProbeCal-Prompt&Trace	42.06	49.18	51.76	53.58	54.95	56.79	57.73	58.11
ProbeCal-Prompt-TS	42.33	47.04	49.44	51.00	51.44	52.84	53.50	54.00
ProbeCal-Trace-TS	32.23	42.36	47.51	50.66	52.70	55.16	56.36	57.10
ProbeCal-Prompt&Trace-TS	42.33	48.78	51.59	53.67	54.77	56.73	57.49	58.32
ProbeCal-Prompt-Weight	41.84	47.00	49.22	50.55	51.46	52.72	53.36	53.90
ProbeCal-Trace-Weight	32.23	42.38	47.63	50.73	52.84	55.17	56.37	57.18
ProbeCal-Prompt&Trace-Weight	41.84	48.78	51.32	53.26	54.59	56.43	57.19	57.76
ProbeCal-Prompt-Sort	41.91	47.32	49.06	50.66	51.49	52.70	53.48	53.79
ProbeCal-Trace-Sort	32.23	42.37	47.53	50.58	52.64	55.13	56.14	56.97
ProbeCal-Prompt&Trace-Sort	41.91	49.01	50.94	53.25	54.02	55.89	56.79	57.06
ProbeCal-Prompt-E.S.L.	37.96	44.35	46.77	48.35	49.55	50.75	51.59	51.93
ProbeCal-Trace-E.S.L.	32.23	41.86	45.85	48.27	49.95	52.02	52.88	53.33
ProbeCal-Prompt&Trace-E.S.L.	37.96	44.75	47.12	48.93	50.30	51.51	52.62	52.76
Dataset	**Geometry**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	8.64	9.60	10.24	10.51	10.85	11.10	11.29	11.47
ProbeCal-Trace	7.34	8.97	10.07	10.54	10.79	11.29	11.30	11.38
ProbeCal-Prompt&Trace	8.64	9.79	10.69	10.78	11.19	11.01	11.23	11.15
ProbeCal-Prompt-TS	8.44	9.78	9.80	10.59	10.84	11.37	11.39	11.33
ProbeCal-Trace-TS	7.34	8.97	10.08	10.53	10.82	11.27	11.29	11.37
ProbeCal-Prompt&Trace-TS	8.44	10.12	10.70	11.00	11.24	10.96	11.33	11.15
ProbeCal-Prompt-Weight	8.31	9.17	10.44	10.48	11.01	11.10	11.55	11.87
ProbeCal-Trace-Weight	7.34	9.11	10.22	10.61	10.96	11.27	11.48	11.54
ProbeCal-Prompt&Trace-Weight	8.31	10.04	11.27	11.29	11.40	11.53	11.74	11.86
ProbeCal-Prompt-Sort	7.68	8.91	9.44	9.83	10.36	10.82	11.32	11.27
ProbeCal-Trace-Sort	7.34	8.70	9.54	10.21	10.29	10.83	10.62	10.77
ProbeCal-Prompt&Trace-Sort	7.68	9.11	9.53	10.08	10.18	10.20	9.95	10.11
ProbeCal-Prompt-E.S.L.	6.08	7.24	7.66	8.26	8.68	9.06	9.59	9.63
ProbeCal-Trace-E.S.L.	7.34	8.42	8.67	9.09	9.12	9.22	9.33	9.40
ProbeCal-Prompt&Trace-E.S.L.	6.08	6.90	7.49	7.98	8.35	8.77	9.02	9.15

Table C.5: Static Tool-Using results of different variants on each dataset - II

Dataset	**Algebra**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	20.27	23.78	25.98	27.29	28.06	29.21	30.02	30.49
ProbeCal-Trace	18.13	23.09	25.80	27.54	28.29	29.79	30.89	31.18
ProbeCal-Prompt&Trace	20.27	24.89	27.02	28.42	29.28	30.54	31.64	32.04
ProbeCal-Prompt-TS	20.15	23.68	25.98	27.14	28.16	29.23	30.08	30.37
ProbeCal-Trace-TS	18.13	23.09	25.80	27.53	28.28	29.77	30.87	31.18
ProbeCal-Prompt&Trace-TS	20.15	24.77	27.08	28.33	29.39	30.56	31.48	31.95
ProbeCal-Prompt-Weight	20.05	23.49	25.65	27.06	27.85	29.20	30.08	30.37
ProbeCal-Trace-Weight	18.13	23.18	25.97	27.68	28.54	30.05	31.18	31.49
ProbeCal-Prompt&Trace-Weight	20.05	24.72	26.94	28.32	29.30	30.78	31.78	32.02
ProbeCal-Prompt-Sort	20.07	23.56	25.73	27.12	28.16	29.07	30.11	30.38
ProbeCal-Trace-Sort	18.13	23.02	25.80	27.50	28.21	29.74	30.92	31.27
ProbeCal-Prompt&Trace-Sort	20.07	24.68	26.72	28.38	29.27	30.40	31.53	31.96
ProbeCal-Prompt-E.S.L.	19.42	22.68	25.03	26.15	27.03	28.30	28.95	29.35
ProbeCal-Trace-E.S.L.	18.13	22.74	25.48	26.89	27.63	28.93	29.93	30.27
ProbeCal-Prompt&Trace-E.S.L.	19.42	23.57	25.65	26.67	27.52	28.86	29.83	30.10
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	7.86	10.53	12.22	13.08	13.78	14.96	15.73	15.85
ProbeCal-Trace	6.34	9.96	11.52	13.00	13.81	14.84	15.61	15.99
ProbeCal-Prompt&Trace	7.86	10.56	12.34	13.28	14.02	15.22	16.22	16.32
ProbeCal-Prompt-TS	7.77	10.66	12.11	13.24	13.93	14.82	15.55	16.05
ProbeCal-Trace-TS	6.34	9.96	11.51	12.99	13.79	14.82	15.57	15.94
ProbeCal-Prompt&Trace-TS	7.77	10.75	12.14	13.42	14.17	15.16	15.99	16.56
ProbeCal-Prompt-Weight	8.00	10.68	12.25	13.14	13.97	14.98	15.72	16.11
ProbeCal-Trace-Weight	6.34	9.93	11.46	13.03	13.70	14.75	15.56	15.82
ProbeCal-Prompt&Trace-Weight	8.00	10.85	12.46	13.26	14.11	15.28	16.23	16.57
ProbeCal-Prompt-Sort	7.76	10.16	11.85	12.62	13.76	14.82	15.06	15.77
ProbeCal-Trace-Sort	6.34	9.82	11.33	12.82	13.55	14.60	15.26	15.50
ProbeCal-Prompt&Trace-Sort	7.76	10.10	11.85	12.60	13.74	14.95	15.48	15.94
ProbeCal-Prompt-E.S.L.	6.30	8.93	10.22	11.17	11.93	12.91	13.23	13.57
ProbeCal-Trace-E.S.L.	6.34	9.62	11.00	12.33	12.83	13.90	14.35	14.59
ProbeCal-Prompt&Trace-E.S.L.	6.30	8.80	9.96	10.84	11.49	12.56	12.97	13.34
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	15.05	18.09	19.78	21.14	22.17	23.05	24.47	24.69
ProbeCal-Trace	12.14	16.86	19.12	20.85	21.36	22.46	22.77	22.86
ProbeCal-Prompt&Trace	15.05	18.50	19.85	20.68	21.62	21.86	22.81	22.76
ProbeCal-Prompt-TS	14.88	18.15	19.64	21.44	22.08	23.01	24.12	24.73
ProbeCal-Trace-TS	12.14	16.86	19.10	20.87	21.26	22.27	22.60	22.74
ProbeCal-Prompt&Trace-TS	14.88	18.56	19.51	21.12	21.49	22.22	22.47	22.45
ProbeCal-Prompt-Weight	14.83	18.28	20.27	21.51	21.68	22.82	24.14	24.86
ProbeCal-Trace-Weight	12.14	16.92	19.23	20.92	21.26	22.45	22.64	22.55
ProbeCal-Prompt&Trace-Weight	14.83	18.50	20.51	21.08	21.17	21.83	22.36	22.73
ProbeCal-Prompt-Sort	13.94	16.86	19.26	20.01	21.28	22.17	22.85	23.19
ProbeCal-Trace-Sort	12.14	16.94	19.35	20.72	21.27	22.19	22.69	22.65
ProbeCal-Prompt&Trace-Sort	13.94	17.19	19.21	19.85	20.29	20.76	21.01	21.10
ProbeCal-Prompt-E.S.L.	13.21	16.00	17.95	19.09	20.19	20.85	21.79	21.97
ProbeCal-Trace-E.S.L.	12.14	16.81	18.78	20.27	20.76	21.51	21.85	21.83
ProbeCal-Prompt&Trace-E.S.L.	13.21	15.96	17.17	18.40	18.72	19.45	20.08	20.09
Dataset	**Number**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	24.72	26.32	28.36	29.31	30.11	30.99	31.69	31.73
ProbeCal-Trace	23.76	27.84	29.89	30.92	31.64	32.62	33.42	33.56
ProbeCal-Prompt&Trace	24.72	28.58	30.16	31.31	31.80	32.92	33.42	33.65
ProbeCal-Prompt-TS	24.70	25.96	28.54	29.19	30.03	30.91	31.67	31.94
ProbeCal-Trace-TS	23.76	27.84	29.88	30.94	31.67	32.64	33.47	33.62
ProbeCal-Prompt&Trace-TS	24.70	28.33	30.16	30.99	31.81	32.78	33.43	33.86
ProbeCal-Prompt-Weight	24.81	26.65	28.72	29.20	30.20	30.84	31.67	31.83
ProbeCal-Trace-Weight	23.76	27.78	29.92	30.97	31.72	32.54	33.49	33.60
ProbeCal-Prompt&Trace-Weight	24.81	28.47	30.52	31.16	32.24	32.98	33.48	33.72
ProbeCal-Prompt-Sort	24.87	26.56	28.54	29.56	30.51	30.92	31.57	31.80
ProbeCal-Trace-Sort	23.76	27.62	29.76	30.78	31.51	32.42	33.35	33.35
ProbeCal-Prompt&Trace-Sort	24.87	28.55	30.15	31.32	31.90	32.65	33.15	33.38
ProbeCal-Prompt-E.S.L.	24.56	26.24	28.28	28.85	29.59	30.31	30.93	30.99
ProbeCal-Trace-E.S.L.	23.76	27.10	29.02	30.05	30.54	31.14	32.12	32.25
ProbeCal-Prompt&Trace-E.S.L.	24.56	27.40	28.73	29.51	30.34	30.97	31.54	31.54

Table C.6: Dynamic Tool-Using results on each dataset

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	10	15	20
TroVE	18.81	23.74	26.47	27.88	29.34	31.78	32.33	32.98
TroVE-Sample	19.64	23.80	26.83	28.32	29.38	31.61	32.53	32.87
TroVE-Sort	18.81	24.13	26.78	28.39	29.48	31.60	32.36	33.01
TroVE-E.S.L.	18.81	24.27	26.78	28.11	28.99	30.89	31.59	32.06
ProbeCal-Prompt	18.77	23.53	25.93	27.80	28.65	31.29	32.03	32.48
ProbeCal-Trace	18.81	25.12	28.08	29.53	30.69	32.99	34.07	34.51
ProbeCal-Prompt&Trace	18.77	24.77	27.23	28.89	30.33	32.62	33.64	34.20
Dataset	**Count**							
# Samples	1	2	3	4	5	10	15	20
TroVE	17.43	20.98	22.34	23.55	23.86	25.70	26.23	26.82
TroVE-Sample	18.71	22.00	23.32	24.20	24.63	26.27	26.52	26.82
TroVE-Sort	17.43	20.98	22.53	23.26	23.72	25.21	25.66	26.05
TroVE-E.S.L.	17.43	20.52	22.12	22.43	22.76	24.00	24.13	24.30
ProbeCal-Prompt	20.36	23.24	24.44	25.30	25.82	27.00	27.52	27.70
ProbeCal-Trace	17.43	21.60	23.57	24.60	25.17	26.88	27.07	27.30
ProbeCal-Prompt&Trace	20.36	24.09	25.31	25.73	26.43	27.11	27.71	28.39
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	10	15	20
TroVE	22.32	29.54	33.84	36.47	38.48	44.00	45.88	47.62
TroVE-Sample	23.78	30.55	34.81	37.55	39.50	44.80	46.60	48.00
TroVE-Sort	22.32	30.25	34.49	37.08	39.08	44.18	46.04	47.81
TroVE-E.S.L.	22.32	30.29	34.56	37.11	38.77	43.23	44.56	45.94
ProbeCal-Prompt	29.52	35.82	39.03	41.54	42.99	47.67	49.15	50.44
ProbeCal-Trace	22.32	31.32	36.74	40.18	42.77	49.96	52.66	54.89
ProbeCal-Prompt&Trace	29.52	37.60	41.48	44.56	46.54	52.15	54.23	55.40
Dataset	**Geometry**							
# Samples	1	2	3	4	5	10	15	20
TroVE	3.98	5.13	5.87	6.80	7.05	8.84	9.20	9.82
TroVE-Sample	3.61	5.44	6.27	6.78	7.25	8.64	9.62	9.59
TroVE-Sort	3.98	5.16	6.16	6.84	7.22	8.57	9.14	9.47
TroVE-E.S.L.	3.98	5.25	6.33	6.95	7.22	8.19	8.24	8.77
ProbeCal-Prompt	4.50	6.23	6.68	7.54	8.16	8.96	9.90	10.37
ProbeCal-Trace	3.98	5.16	6.02	7.10	7.27	8.54	8.84	9.23
ProbeCal-Prompt&Trace	4.50	6.48	6.84	7.66	8.05	8.89	9.55	10.04
Dataset	**Algebra**							
# Samples	1	2	3	4	5	10	15	20
TroVE	15.41	21.39	25.15	27.30	28.76	32.27	33.59	34.38
TroVE-Sample	16.77	22.96	25.97	28.39	29.37	32.35	33.66	34.25
TroVE-Sort	15.41	21.45	25.11	27.21	28.62	32.09	33.37	34.15
TroVE-E.S.L.	15.41	21.45	25.01	26.94	28.17	31.46	32.72	33.29
ProbeCal-Prompt	20.23	25.28	28.17	29.55	30.25	33.12	34.18	34.86
ProbeCal-Trace	15.41	21.94	25.92	28.18	29.77	33.54	34.83	35.77
ProbeCal-Prompt&Trace	20.23	25.96	28.96	30.36	31.27	34.05	35.22	36.18
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	10	15	20
TroVE	6.54	9.62	11.62	12.97	13.94	16.56	17.94	18.34
TroVE-Sample	6.47	9.44	11.51	12.60	13.88	16.27	17.33	18.22
TroVE-Sort	6.54	9.74	11.65	12.94	13.80	16.30	17.49	17.93
TroVE-E.S.L.	6.54	9.75	11.73	12.99	13.83	16.37	17.50	17.97
ProbeCal-Prompt	7.98	10.50	12.35	13.26	14.23	16.33	17.38	17.94
ProbeCal-Trace	6.54	9.67	11.67	12.97	13.94	16.58	17.75	18.34
ProbeCal-Prompt&Trace	7.98	10.62	12.43	13.30	14.12	16.42	17.52	18.27
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	10	15	20
TroVE	11.01	15.03	16.99	18.33	18.67	20.90	21.99	22.45
TroVE-Sample	11.24	15.69	17.65	18.08	18.42	21.42	21.53	22.18
TroVE-Sort	11.01	14.87	16.81	17.77	18.44	20.17	21.17	21.60
TroVE-E.S.L.	11.01	14.91	16.54	17.68	18.36	19.68	20.42	20.97
ProbeCal-Prompt	13.14	17.22	18.46	19.64	20.17	21.83	22.76	23.14
ProbeCal-Trace	11.01	15.09	17.13	18.32	18.72	20.88	22.18	22.77
ProbeCal-Prompt&Trace	13.14	17.37	18.76	19.82	20.32	22.10	23.21	23.49
Dataset	**Number**							
# Samples	1	2	3	4	5	10	15	20
TroVE	14.39	19.96	22.57	24.31	25.34	27.44	27.97	28.63
TroVE-Sample	17.19	21.80	24.31	25.26	25.97	27.69	28.43	28.81
TroVE-Sort	14.39	20.03	22.81	24.38	25.31	27.20	27.46	28.26
TroVE-E.S.L.	14.39	20.06	22.76	24.29	24.89	26.35	26.66	27.16
ProbeCal-Prompt	17.20	21.65	23.91	25.20	26.18	27.94	28.56	29.15
ProbeCal-Trace	14.39	20.67	23.73	25.69	26.76	28.99	30.10	30.89
ProbeCal-Prompt&Trace	17.20	22.53	24.98	26.65	27.43	29.65	30.57	31.24

Table C.7: Dynamic Tool-Using results of different variants on each dataset I

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	18.77	23.53	25.93	27.80	28.65	31.29	32.03	32.48
ProbeCal-Trace	18.81	25.12	28.08	29.53	30.69	32.99	34.07	34.51
ProbeCal-Prompt&Trace	18.77	24.77	27.23	28.89	30.33	32.62	33.64	34.20
ProbeCal-Prompt-TS	18.59	23.15	26.21	27.74	28.85	31.40	32.11	32.65
ProbeCal-Trace-TS	18.81	25.12	28.08	29.53	30.70	32.97	34.05	34.48
ProbeCal-Prompt&Trace-TS	18.59	24.38	27.48	29.20	30.06	32.69	33.70	34.27
ProbeCal-Prompt-Weight	19.03	23.46	26.27	27.69	28.82	31.11	31.93	32.47
ProbeCal-Trace-Weight	18.81	25.14	27.97	29.43	30.69	33.17	34.03	34.74
ProbeCal-Prompt&Trace-Weight	19.03	24.72	27.39	29.03	30.17	32.42	33.70	33.96
ProbeCal-Prompt-Sort	18.85	23.80	26.37	27.74	28.93	31.18	31.86	32.33
ProbeCal-Trace-Sort	18.81	25.06	27.80	29.25	30.40	32.86	33.59	34.10
ProbeCal-Prompt&Trace-Sort	18.85	25.05	27.60	28.93	30.04	32.43	33.47	33.96
ProbeCal-Prompt-E.S.L.	18.18	22.56	25.28	26.53	27.57	29.84	30.57	31.08
ProbeCal-Trace-E.S.L.	18.81	24.79	27.37	28.77	29.65	31.97	33.05	33.52
ProbeCal-Prompt&Trace-E.S.L.	18.18	23.48	25.88	27.04	28.16	30.47	31.43	32.02
Dataset	**Count**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	20.36	23.24	24.44	25.30	25.82	27.00	27.52	27.70
ProbeCal-Trace	17.43	21.60	23.57	24.60	25.17	26.88	27.07	27.30
ProbeCal-Prompt&Trace	20.36	24.09	25.31	25.73	26.43	27.11	27.71	28.39
ProbeCal-Prompt-TS	20.51	22.91	24.70	25.49	25.90	27.20	27.57	27.80
ProbeCal-Trace-TS	17.43	21.60	23.57	24.63	25.16	26.87	27.11	27.33
ProbeCal-Prompt&Trace-TS	20.51	23.77	25.51	25.75	26.47	27.32	27.90	28.08
ProbeCal-Prompt-Weight	20.59	22.99	24.89	25.35	26.05	27.37	27.44	27.85
ProbeCal-Trace-Weight	17.43	21.80	23.77	25.11	25.54	27.27	27.55	27.89
ProbeCal-Prompt&Trace-Weight	20.59	24.40	25.64	26.40	26.72	27.53	27.84	28.27
ProbeCal-Prompt-Sort	21.36	23.81	24.95	26.12	26.64	28.20	28.58	28.84
ProbeCal-Trace-Sort	17.43	21.75	23.61	24.94	25.41	26.95	27.71	28.25
ProbeCal-Prompt&Trace-Sort	21.36	24.73	25.81	26.63	26.85	27.99	28.32	28.64
ProbeCal-Prompt-E.S.L.	20.61	22.82	24.52	24.97	25.66	27.03	27.40	28.04
ProbeCal-Trace-E.S.L.	17.43	21.64	23.47	24.42	24.93	25.95	26.39	26.58
ProbeCal-Prompt&Trace-E.S.L.	20.61	23.45	25.18	25.24	25.53	26.74	26.93	27.11
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	29.52	35.82	39.03	41.54	42.99	47.67	49.15	50.44
ProbeCal-Trace	22.32	31.32	36.74	40.18	42.77	49.96	52.66	54.89
ProbeCal-Prompt&Trace	29.52	37.60	41.48	44.56	46.54	52.15	54.23	55.40
ProbeCal-Prompt-TS	29.39	35.47	39.77	41.25	43.10	47.49	49.48	50.55
ProbeCal-Trace-TS	22.32	31.32	36.74	40.19	42.77	50.00	52.71	55.00
ProbeCal-Prompt&Trace-TS	29.39	37.45	42.10	44.22	46.60	51.78	54.32	55.45
ProbeCal-Prompt-Weight	29.28	35.65	39.10	41.79	43.20	47.15	49.26	50.14
ProbeCal-Trace-Weight	22.32	31.32	36.72	40.04	42.85	49.86	52.88	54.92
ProbeCal-Prompt&Trace-Weight	29.28	37.68	41.63	44.74	46.68	51.63	54.17	55.48
ProbeCal-Prompt-Sort	29.27	35.82	39.38	42.08	43.22	47.84	49.66	50.50
ProbeCal-Trace-Sort	22.32	31.29	36.76	40.18	42.74	49.78	52.62	54.64
ProbeCal-Prompt&Trace-Sort	29.27	37.67	41.90	45.26	46.70	52.04	54.57	55.58
ProbeCal-Prompt-E.S.L.	29.62	36.01	39.57	41.98	43.81	47.64	49.17	49.80
ProbeCal-Trace-E.S.L.	22.32	31.22	36.64	39.91	42.49	48.98	51.73	53.74
ProbeCal-Prompt&Trace-E.S.L.	29.62	37.68	41.83	44.58	46.70	51.09	53.15	53.78
Dataset	**Geometry**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	4.50	6.23	6.68	7.54	8.16	8.96	9.90	10.37
ProbeCal-Trace	3.98	5.16	6.02	7.10	7.27	8.54	8.84	9.23
ProbeCal-Prompt&Trace	4.50	6.48	6.84	7.66	8.05	8.89	9.55	10.04
ProbeCal-Prompt-TS	4.76	5.84	6.95	7.48	7.80	9.22	9.84	10.27
ProbeCal-Trace-TS	3.98	5.16	6.02	7.11	7.26	8.53	8.89	9.21
ProbeCal-Prompt&Trace-TS	4.76	6.08	7.02	7.53	7.83	9.08	9.70	9.69
ProbeCal-Prompt-Weight	4.60	5.89	6.90	6.80	7.64	9.01	9.50	10.30
ProbeCal-Trace-Weight	3.98	5.10	6.04	7.05	7.06	8.46	8.87	9.47
ProbeCal-Prompt&Trace-Weight	4.60	6.08	6.78	7.08	7.56	8.33	9.11	9.48
ProbeCal-Prompt-Sort	4.99	6.44	7.73	7.84	8.19	9.48	10.09	10.58
ProbeCal-Trace-Sort	3.98	5.11	5.95	6.95	7.24	8.14	8.50	8.84
ProbeCal-Prompt&Trace-Sort	4.99	6.46	7.66	8.04	8.02	8.86	9.44	9.86
ProbeCal-Prompt-E.S.L.	3.98	5.17	5.41	5.86	6.11	6.68	6.91	7.03
ProbeCal-Trace-E.S.L.	3.98	4.88	5.50	6.46	6.43	6.80	6.58	6.65
ProbeCal-Prompt&Trace-E.S.L.	3.98	5.05	5.27	5.47	5.85	6.39	6.46	6.47

Table C.8: Dynamic Tool-Using results of different variants on each dataset II

Dataset	**Algebra**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	20.23	25.28	28.17	29.55	30.25	33.12	34.18	34.86
ProbeCal-Trace	15.41	21.94	25.92	28.18	29.77	33.54	34.83	35.77
ProbeCal-Prompt&Trace	20.23	25.96	28.96	30.36	31.27	34.05	35.22	36.18
ProbeCal-Prompt-TS	20.09	25.47	27.88	29.55	30.54	33.04	34.21	34.86
ProbeCal-Trace-TS	15.41	21.94	25.92	28.17	29.78	33.53	34.80	35.75
ProbeCal-Prompt&Trace-TS	20.09	25.99	28.79	30.35	31.33	34.15	35.21	36.06
ProbeCal-Prompt-Weight	20.46	25.32	27.86	29.40	30.39	32.91	34.04	34.85
ProbeCal-Trace-Weight	15.41	21.93	25.91	28.05	29.65	33.23	34.56	35.33
ProbeCal-Prompt&Trace-Weight	20.46	26.01	28.73	30.22	31.27	33.61	34.79	35.70
ProbeCal-Prompt-Sort	20.33	25.22	27.56	29.23	30.02	32.39	33.72	34.39
ProbeCal-Trace-Sort	15.41	21.86	25.80	27.85	29.35	32.90	34.15	34.99
ProbeCal-Prompt&Trace-Sort	20.33	25.90	28.16	29.79	30.83	33.16	34.63	35.31
ProbeCal-Prompt-E.S.L.	19.26	24.00	26.35	28.00	28.91	31.38	32.60	33.07
ProbeCal-Trace-E.S.L.	15.41	21.72	25.42	27.55	28.90	32.27	33.69	34.71
ProbeCal-Prompt&Trace-E.S.L.	19.26	24.67	26.93	28.61	29.40	32.08	32.98	33.38
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	7.98	10.50	12.35	13.26	14.23	16.33	17.38	17.94
ProbeCal-Trace	6.54	9.67	11.67	12.97	13.94	16.58	17.75	18.34
ProbeCal-Prompt&Trace	7.98	10.62	12.43	13.30	14.12	16.42	17.52	18.27
ProbeCal-Prompt-TS	7.98	10.61	12.47	13.44	14.30	16.31	17.55	18.21
ProbeCal-Trace-TS	6.54	9.67	11.67	12.97	13.94	16.57	17.77	18.33
ProbeCal-Prompt&Trace-TS	7.98	10.82	12.47	13.50	14.42	16.58	17.66	18.19
ProbeCal-Prompt-Weight	7.95	10.37	12.41	13.32	14.04	16.36	17.53	18.02
ProbeCal-Trace-Weight	6.54	9.68	11.65	13.00	13.90	16.76	17.83	18.52
ProbeCal-Prompt&Trace-Weight	7.95	10.54	12.47	13.42	14.13	16.47	17.79	18.43
ProbeCal-Prompt-Sort	7.64	10.12	11.54	12.99	13.79	16.20	17.63	18.17
ProbeCal-Trace-Sort	6.54	9.65	11.66	12.95	13.88	16.52	17.75	18.20
ProbeCal-Prompt&Trace-Sort	7.64	10.18	11.80	13.14	13.94	16.27	17.58	18.28
ProbeCal-Prompt-E.S.L.	7.58	10.00	11.64	12.62	13.24	15.82	16.93	17.71
ProbeCal-Trace-E.S.L.	6.54	9.68	11.55	12.97	13.77	16.29	17.23	17.60
ProbeCal-Prompt&Trace-E.S.L.	7.58	10.11	11.67	12.52	13.22	15.60	16.70	17.40
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	13.14	17.22	18.46	19.64	20.17	21.83	22.76	23.14
ProbeCal-Trace	11.01	15.09	17.13	18.32	18.72	20.88	22.18	22.77
ProbeCal-Prompt&Trace	13.14	17.37	18.76	19.82	20.32	22.10	23.21	23.49
ProbeCal-Prompt-TS	13.85	16.59	18.23	19.42	20.01	22.10	22.86	23.38
ProbeCal-Trace-TS	11.01	15.09	17.13	18.32	18.72	20.91	22.10	22.81
ProbeCal-Prompt&Trace-TS	13.85	16.92	18.49	19.69	20.23	22.24	23.06	23.56
ProbeCal-Prompt-Weight	13.35	16.76	18.28	19.40	20.17	21.33	21.86	22.55
ProbeCal-Trace-Weight	11.01	15.06	17.00	18.23	18.81	20.81	21.97	22.59
ProbeCal-Prompt&Trace-Weight	13.35	17.08	18.62	19.95	20.49	21.62	22.36	23.12
ProbeCal-Prompt-Sort	13.19	16.41	17.56	19.65	19.76	21.37	22.29	22.78
ProbeCal-Trace-Sort	11.01	15.09	17.03	18.27	18.63	20.90	22.21	23.01
ProbeCal-Prompt&Trace-Sort	13.19	16.74	17.94	19.99	19.68	21.33	22.65	22.96
ProbeCal-Prompt-E.S.L.	12.08	14.68	15.96	16.85	17.42	18.18	18.67	18.32
ProbeCal-Trace-E.S.L.	11.01	14.73	16.59	17.26	17.68	19.24	20.12	20.54
ProbeCal-Prompt&Trace-E.S.L.	12.08	14.86	15.85	16.62	16.83	17.65	18.18	18.13
Dataset	**Number**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	17.20	21.65	23.91	25.20	26.18	27.94	28.56	29.15
ProbeCal-Trace	14.39	20.67	23.73	25.69	26.76	28.99	30.10	30.89
ProbeCal-Prompt&Trace	17.20	22.53	24.98	26.65	27.43	29.65	30.57	31.24
ProbeCal-Prompt-TS	16.87	21.57	23.91	25.30	26.12	28.04	28.73	29.20
ProbeCal-Trace-TS	14.39	20.67	23.74	25.69	26.74	28.97	30.06	30.90
ProbeCal-Prompt&Trace-TS	16.87	22.41	25.05	26.60	27.68	29.81	30.53	31.19
ProbeCal-Prompt-Weight	16.95	21.79	23.99	25.35	26.23	28.11	28.73	29.36
ProbeCal-Trace-Weight	14.39	20.62	23.75	25.74	26.78	29.16	30.16	31.16
ProbeCal-Prompt&Trace-Weight	16.95	22.69	25.17	26.64	27.71	29.77	30.66	31.51
ProbeCal-Prompt-Sort	16.40	21.12	23.66	24.85	25.78	27.79	28.63	29.44
ProbeCal-Trace-Sort	14.39	20.61	23.69	25.60	26.59	28.94	30.02	30.87
ProbeCal-Prompt&Trace-Sort	16.40	22.00	24.69	26.02	27.22	29.30	30.14	30.94
ProbeCal-Prompt-E.S.L.	16.43	20.51	22.61	23.71	24.40	26.07	26.93	27.54
ProbeCal-Trace-E.S.L.	14.39	20.41	23.19	25.15	25.86	27.91	28.87	29.37
ProbeCal-Prompt&Trace-E.S.L.	16.43	21.11	23.30	24.60	25.11	27.00	27.93	28.75

Table C.9: ECE loss on the static tool using settings on each dataset. The settings are denoted using two letters, the first letter corresponds to whether to use weighted training, with 'W' representing yes and 'N' representing no. The second letter corresponds to whether and in which way using LLM logit as the input or not, with 'S' representing using SORT to get the logit, 'E' representing using E.S.L. to get the logit, and 'N' representing not using LLM logit. When there are two numbers for the result of a setting, the first number denotes the training ECE loss and the second number denotes the testing ECE loss. When there is one number for the result of a setting, the number refers to the testing ECE loss.

Settings	NN	WN	NS	WS	NE	WE
Dataset			**Prealgebra**			
LLM Logit	0.725, 0.751	0.725, 0.751	0.342, 0.328	0.340, 0.328	0.152, 0.054	0.151, 0.054
PROBECAL-PROMPT	0.012, 0.092	0.015, 0.091	0.013, 0.097	0.018, 0.090	0.094, 0.150	0.103, 0.144
PROBECAL-PROMPT(TS)	0.019, 0.083	0.024, 0.082	0.022, 0.088	0.024, 0.084	0.105, 0.147	0.122, 0.138
PROBECAL-TRACE	0.010, 0.090	0.014, 0.087	0.017, 0.081	0.018, 0.081	0.104, 0.138	0.104, 0.137
PROBECAL-TRACE(TS)	0.008, 0.099	0.008, 0.097	0.008, 0.091	0.008, 0.094	0.108, 0.139	0.097, 0.140
PROBECAL-PROMPT&TRACE	0.053	0.050	0.049	0.045	0.129	0.129
Dataset			**Count**			
LLM Logit	0.842, 0.804	0.840, 0.804	0.512, 0.470	0.512, 0.470	0.161, 0.096	0.161, 0.096
PROBECAL-PROMPT	0.007, 0.086	0.010, 0.082	0.006, 0.088	0.010, 0.087	0.065, 0.128	0.072, 0.131
PROBECAL-PROMPT(TS)	0.008, 0.082	0.011, 0.080	0.013, 0.079	0.018, 0.079	0.077, 0.130	0.082, 0.133
PROBECAL-TRACE	0.009, 0.083	0.013, 0.079	0.013, 0.083	0.017, 0.073	0.075, 0.127	0.077, 0.123
PROBECAL-TRACE(TS)	0.005, 0.090	0.005, 0.090	0.007, 0.090	0.009, 0.082	0.067, 0.129	0.075, 0.125
PROBECAL-PROMPT&TRACE	0.065	0.061	0.064	0.060	0.123	0.121
Dataset			**TabMWP**			
LLM Logit	0.637, 0.675	0.638, 0.675	0.308, 0.295	0.308, 0.295	0.308, 0.170	0.310, 0.170
PROBECAL-PROMPT	0.008, 0.075	0.010, 0.070	0.010, 0.077	0.014, 0.072	0.137, 0.166	0.134, 0.160
PROBECAL-PROMPT(TS)	0.015, 0.065	0.019, 0.059	0.015, 0.067	0.022, 0.061	0.143, 0.168	0.143, 0.163
PROBECAL-TRACE	0.011, 0.046	0.012, 0.047	0.012, 0.041	0.014, 0.039	0.150, 0.152	0.150, 0.147
PROBECAL-TRACE(TS)	0.009, 0.050	0.011, 0.051	0.011, 0.047	0.010, 0.045	0.166, 0.161	0.178, 0.165
PROBECAL-PROMPT&TRACE	0.037	0.035	0.039	0.040	0.156	0.153
Dataset			**Geometry**			
LLM Logit	0.957, 0.926	0.957, 0.926	0.188, 0.289	0.188, 0.289	0.048, 0.048	0.048, 0.048
PROBECAL-PROMPT	0.008, 0.047	0.009, 0.045	0.006, 0.056	0.009, 0.054	0.218, 0.184	0.245, 0.205
PROBECAL-PROMPT(TS)	0.009, 0.043	0.010, 0.044	0.013, 0.047	0.014, 0.049	0.229, 0.189	0.261, 0.217
PROBECAL-TRACE	0.007, 0.045	0.009, 0.042	0.010, 0.048	0.010, 0.049	0.229, 0.187	0.225, 0.187
PROBECAL-TRACE(TS)	0.005, 0.046	0.008, 0.045	0.008, 0.048	0.009, 0.049	0.242, 0.198	0.233, 0.193
PROBECAL-PROMPT&TRACE	0.043	0.039	0.049	0.050	0.176	0.185
Dataset			**Algebra**			
LLM Logit	0.818, 0.819	0.818, 0.819	0.432, 0.405	0.432, 0.405	0.119, 0.051	0.119, 0.051
PROBECAL-PROMPT	0.010, 0.051	0.014, 0.046	0.009, 0.057	0.021, 0.045	0.095, 0.121	0.100, 0.125
PROBECAL-PROMPT(TS)	0.018, 0.044	0.020, 0.039	0.021, 0.046	0.025, 0.043	0.108, 0.126	0.109, 0.127
PROBECAL-TRACE	0.010, 0.048	0.011, 0.046	0.013, 0.046	0.016, 0.041	0.097, 0.116	0.100, 0.122
PROBECAL-TRACE(TS)	0.004, 0.056	0.008, 0.051	0.008, 0.052	0.007, 0.051	0.087, 0.114	0.105, 0.124
PROBECAL-PROMPT&TRACE	0.027	0.026	0.030	0.028	0.120	0.125
Dataset			**Intermediate**			
LLM Logit	0.915, 0.936	0.915, 0.936	0.454, 0.430	0.455, 0.430	0.067, 0.044	0.067, 0.044
PROBECAL-PROMPT	0.008, 0.030	0.010, 0.030	0.008, 0.036	0.014, 0.032	0.108, 0.143	0.125, 0.158
PROBECAL-PROMPT(TS)	0.012, 0.027	0.012, 0.027	0.017, 0.031	0.016, 0.031	0.116, 0.149	0.142, 0.172
PROBECAL-TRACE	0.007, 0.039	0.007, 0.038	0.007, 0.041	0.010, 0.038	0.109, 0.139	0.136, 0.167
PROBECAL-TRACE(TS)	0.007, 0.044	0.004, 0.043	0.006, 0.041	0.007, 0.043	0.113, 0.143	0.143, 0.172
PROBECAL-PROMPT&TRACE	0.023	0.022	0.028	0.024	0.141	0.162
Dataset			**Precalculus**			
LLM Logit	0.893, 0.879	0.894, 0.879	0.343, 0.315	0.342, 0.315	0.087, 0.045	0.089, 0.045
PROBECAL-PROMPT	0.011, 0.063	0.016, 0.056	0.013, 0.068	0.011, 0.068	0.131, 0.176	0.171, 0.192
PROBECAL-PROMPT(TS)	0.017, 0.059	0.018, 0.056	0.017, 0.062	0.016, 0.066	0.143, 0.178	0.194, 0.203
PROBECAL-TRACE	0.011, 0.064	0.011, 0.058	0.010, 0.063	0.013, 0.060	0.158, 0.197	0.204, 0.237
PROBECAL-TRACE(TS)	0.008, 0.070	0.011, 0.061	0.011, 0.067	0.007, 0.068	0.178, 0.208	0.210, 0.239
PROBECAL-PROMPT&TRACE	0.048	0.043	0.046	0.046	0.170	0.201
Dataset			**Number**			
LLM Logit	0.758, 0.763	0.758, 0.763	0.370, 0.368	0.368, 0.368	0.136, 0.064	0.136, 0.064
PROBECAL-PROMPT	0.012, 0.074	0.012, 0.076	0.012, 0.076	0.016, 0.073	0.091, 0.112	0.104, 0.120
PROBECAL-PROMPT(TS)	0.018, 0.065	0.019, 0.067	0.019, 0.066	0.022, 0.065	0.107, 0.109	0.115, 0.121
PROBECAL-TRACE	0.014, 0.069	0.011, 0.063	0.010, 0.069	0.019, 0.053	0.099, 0.115	0.105, 0.118
PROBECAL-TRACE(TS)	0.008, 0.077	0.009, 0.069	0.009, 0.072	0.010, 0.063	0.104, 0.116	0.116, 0.123
PROBECAL-PROMPT&TRACE	0.042	0.042	0.044	0.035	0.102	0.112

Table C.10: ECE loss on the dynamic tool using settings on each dataset. The settings are denoted using two letters, the first letter corresponds to whether to use weighted training, with 'W' representing yes and 'N' representing no. The second letter corresponds to whether and in which way using LLM logit as the input or not, with 'S' representing using SORT to get the logit, 'E' representing using E.S.L. to get the logit, and 'N' representing not using LLM logit. When there are two numbers for the result of a setting, the first number denotes the training ECE loss and the second number denotes the testing ECE loss. When there is one number for the result of a setting, the number refers to the testing ECE loss.

Settings	NN	WN	NS	WS	NE	WE
Dataset			**Prealgebra**			
LLM Logit	0.767, 0.812	0.767, 0.812	0.285, 0.342	0.285, 0.342	0.092, 0.049	0.092, 0.049
PROBECAL-PROMPT	0.008, 0.084	0.014, 0.072	0.006, 0.090	0.018, 0.073	0.081, 0.134	0.091, 0.138
PROBECAL-PROMPT(TS)	0.013, 0.077	0.013, 0.073	0.012, 0.081	0.014, 0.078	0.086, 0.133	0.094, 0.137
PROBECAL-TRACE	0.007, 0.094	0.013, 0.091	0.009, 0.092	0.013, 0.085	0.083, 0.125	0.090, 0.125
PROBECAL-TRACE(TS)	0.005, 0.097	0.005, 0.102	0.005, 0.095	0.006, 0.094	0.076, 0.129	0.089, 0.127
PROBECAL-PROMPT&TRACE	0.050	0.045	0.052	0.042	0.117	0.113
Dataset			**Count**			
LLM Logit	0.848, 0.826	0.848, 0.826	0.436, 0.437	0.436, 0.437	0.131, 0.131	0.131, 0.131
PROBECAL-PROMPT	0.006, 0.057	0.011, 0.056	0.006, 0.060	0.009, 0.053	0.079, 0.112	0.081, 0.115
PROBECAL-PROMPT(TS)	0.009, 0.053	0.011, 0.057	0.007, 0.060	0.008, 0.055	0.084, 0.111	0.083, 0.115
PROBECAL-TRACE	0.005, 0.068	0.009, 0.060	0.006, 0.065	0.009, 0.066	0.071, 0.092	0.079, 0.094
PROBECAL-TRACE(TS)	0.004, 0.069	0.004, 0.067	0.004, 0.070	0.004, 0.072	0.064, 0.093	0.078, 0.093
PROBECAL-PROMPT&TRACE	0.042	0.035	0.042	0.040	0.086	0.088
Dataset			**TabMWP**			
LLM Logit	0.767, 0.780	0.767, 0.780	0.330, 0.380	0.329, 0.380	0.173, 0.163	0.173, 0.163
PROBECAL-PROMPT	0.006, 0.082	0.007, 0.079	0.009, 0.083	0.012, 0.078	0.048, 0.114	0.051, 0.107
PROBECAL-PROMPT(TS)	0.009, 0.075	0.012, 0.073	0.012, 0.080	0.014, 0.076	0.058, 0.111	0.050, 0.108
PROBECAL-TRACE	0.010, 0.081	0.012, 0.079	0.006, 0.094	0.010, 0.085	0.051, 0.094	0.055, 0.089
PROBECAL-TRACE(TS)	0.007, 0.086	0.007, 0.086	0.005, 0.099	0.007, 0.089	0.053, 0.096	0.053, 0.091
PROBECAL-PROMPT&TRACE	0.040	0.042	0.047	0.046	0.093	0.088
Dataset			**Geometry**			
LLM Logit	0.975, 0.962	0.975, 0.962	0.257, 0.322	0.257, 0.322	0.072, 0.052	0.071, 0.052
PROBECAL-PROMPT	0.003, 0.028	0.007, 0.025	0.003, 0.030	0.008, 0.024	0.131, 0.096	0.169, 0.128
PROBECAL-PROMPT(TS)	0.005, 0.025	0.005, 0.027	0.002, 0.028	0.005, 0.026	0.155, 0.113	0.165, 0.124
PROBECAL-TRACE	0.005, 0.023	0.008, 0.016	0.002, 0.025	0.007, 0.019	0.165, 0.126	0.190, 0.147
PROBECAL-TRACE(TS)	0.004, 0.023	0.004, 0.021	0.002, 0.024	0.003, 0.024	0.178, 0.138	0.204, 0.161
PROBECAL-PROMPT&TRACE	0.022	0.017	0.024	0.017	0.108	0.135
Dataset			**Algebra**			
LLM Logit	0.841, 0.847	0.840, 0.847	0.398, 0.404	0.398, 0.404	0.105, 0.097	0.106, 0.097
PROBECAL-PROMPT	0.008, 0.055	0.010, 0.052	0.006, 0.056	0.011, 0.052	0.065, 0.106	0.078, 0.113
PROBECAL-PROMPT(TS)	0.011, 0.051	0.010, 0.051	0.008, 0.052	0.010, 0.053	0.078, 0.104	0.082, 0.113
PROBECAL-TRACE	0.008, 0.051	0.008, 0.050	0.006, 0.053	0.013, 0.044	0.071, 0.099	0.076, 0.101
PROBECAL-TRACE(TS)	0.003, 0.057	0.004, 0.057	0.005, 0.053	0.004, 0.055	0.065, 0.098	0.068, 0.102
PROBECAL-PROMPT&TRACE	0.027	0.026	0.032	0.026	0.096	0.100
Dataset			**Intermediate**			
LLM Logit	0.928, 0.934	0.928, 0.934	0.339, 0.371	0.340, 0.371	0.026, 0.036	0.026, 0.036
PROBECAL-PROMPT	0.005, 0.031	0.008, 0.029	0.004, 0.033	0.006, 0.031	0.101, 0.109	0.101, 0.110
PROBECAL-PROMPT(TS)	0.009, 0.026	0.008, 0.028	0.006, 0.031	0.007, 0.030	0.118, 0.119	0.110, 0.115
PROBECAL-TRACE	0.006, 0.027	0.007, 0.028	0.006, 0.030	0.008, 0.028	0.084, 0.094	0.102, 0.104
PROBECAL-TRACE(TS)	0.004, 0.030	0.004, 0.033	0.003, 0.033	0.003, 0.035	0.104, 0.102	0.109, 0.108
PROBECAL-PROMPT&TRACE	0.023	0.021	0.024	0.024	0.100	0.107
Dataset			**Precalculus**			
LLM Logit	0.905, 0.891	0.905, 0.891	0.276, 0.306	0.277, 0.306	0.042, 0.049	0.042, 0.049
PROBECAL-PROMPT	0.004, 0.049	0.008, 0.045	0.006, 0.050	0.009, 0.044	0.119, 0.122	0.116, 0.126
PROBECAL-PROMPT(TS)	0.009, 0.042	0.009, 0.046	0.009, 0.045	0.011, 0.042	0.125, 0.125	0.120, 0.126
PROBECAL-TRACE	0.009, 0.048	0.009, 0.048	0.009, 0.049	0.007, 0.050	0.142, 0.143	0.152, 0.147
PROBECAL-TRACE(TS)	0.005, 0.052	0.006, 0.053	0.009, 0.050	0.007, 0.048	0.140, 0.142	0.174, 0.152
PROBECAL-PROMPT&TRACE	0.032	0.033	0.036	0.035	0.127	0.133
Dataset			**Number**			
LLM Logit	0.804, 0.856	0.804, 0.856	0.297, 0.314	0.296, 0.314	0.066, 0.035	0.066, 0.035
PROBECAL-PROMPT	0.007, 0.040	0.014, 0.029	0.005, 0.045	0.013, 0.033	0.079, 0.121	0.100, 0.132
PROBECAL-PROMPT(TS)	0.012, 0.033	0.012, 0.031	0.012, 0.035	0.012, 0.034	0.084, 0.122	0.099, 0.132
PROBECAL-TRACE	0.009, 0.037	0.010, 0.035	0.010, 0.038	0.013, 0.034	0.083, 0.118	0.091, 0.120
PROBECAL-TRACE(TS)	0.007, 0.037	0.004, 0.039	0.008, 0.040	0.007, 0.041	0.079, 0.117	0.079, 0.117
PROBECAL-PROMPT&TRACE	0.025	0.020	0.029	0.025	0.126	0.136

Table C.11: Experiment results for TabMWP in static tool-using settings using `CODELLAMA-13B-INSTRUCT-hf` with different size of training set from 50 to 500.

Method	**Size 50**		**Size 100**		**Size 200**		**Size 500**	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
Static Tool-Using								
Instance	65.66	0.571	65.66	0.571	65.66	0.571	65.66	0.571
ProbeCal-Prompt	66.19	0.117	67.60	0.088	68.06	0.084	68.11	0.077
ProbeCal-Trace	67.72	0.076	68.50	0.071	69.55	0.055	70.46	0.034
ProbeCal-Prompt&Trace	67.07	0.064	69.10	0.042	70.28	0.038	71.14	0.034

Table C.12: Experiment results for TabMWP in static tool-using setting using `CODELLAMA-13B-INSTRUCT-hf` and `Llama3-8b-Instruct` (Size 500).

Method	**`CODELLAMA-13B-Inst`**			**`Llama3-8b-Inst`**		
	Acc@5	Acc@10	ECE	Acc@5	Acc@10	ECE
Instance	62.94	65.66	0.571	73.07	75.20	0.404
ProbeCal-Prompt	66.59	68.11	0.077	74.79	76.53	0.080
ProbeCal-Trace	67.09	70.46	0.034	76.05	78.25	0.035
ProbeCal-Prompt&Trace	69.01	71.14	0.034	76.79	78.48	0.038

Table C.13: Experiment results for TabMWP in static tool-using setting with `CODELLAMA-13B-INSTRUCT-hf` using verbal confidence.

Method	**`CODELLAMA-13B-Inst` (Size 50)**		
	Acc@5	Acc@10	ECE
Instance	62.94	65.66	0.571
ProbeCal-Prompt	64.27	66.19	0.117
ProbeCal-Trace	64.81	67.72	0.076
ProbeCal-Prompt&Trace	65.23	67.07	0.064
Verbal Confidence	61.52	64.68	0.212

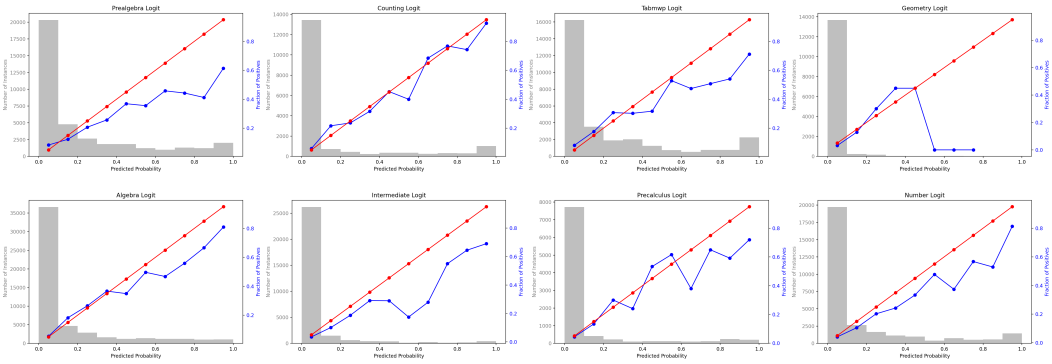


Figure C.6: Calibration curve of ProbeCal-Prompt logits of LLM in Dynamic Tool-Using

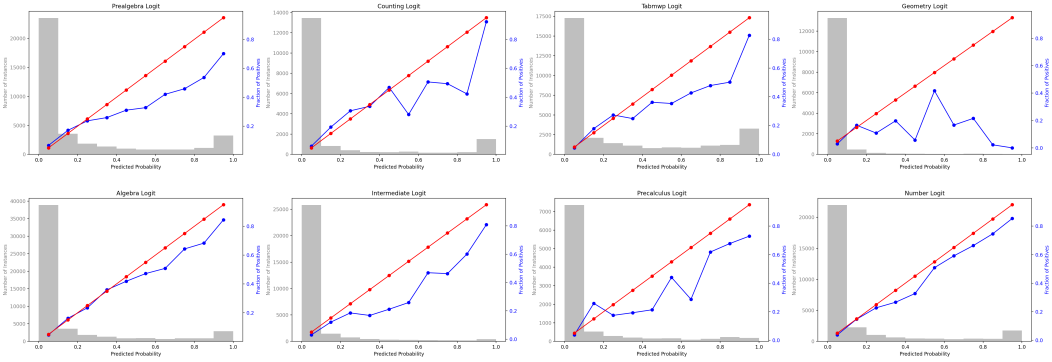


Figure C.7: Calibration curve of ProbeCal-Trace logits of LLM in Dynamic Tool-Using

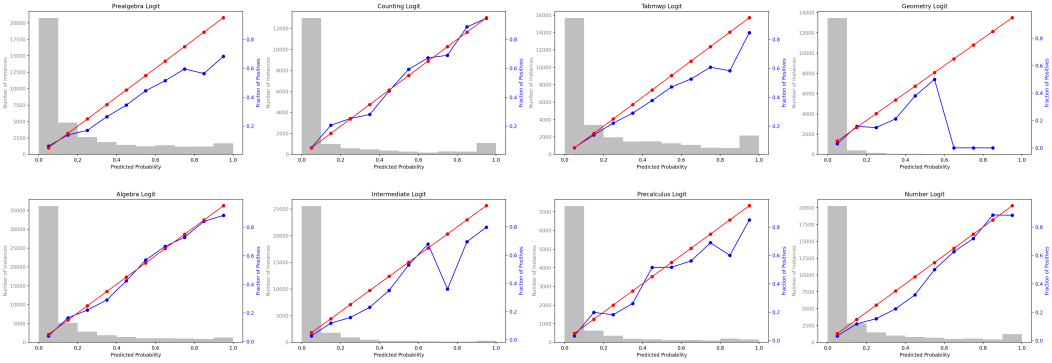


Figure C.8: Calibration curve of ProbeCal-Prompt&Trace logits of LLM in Dynamic Tool-Using

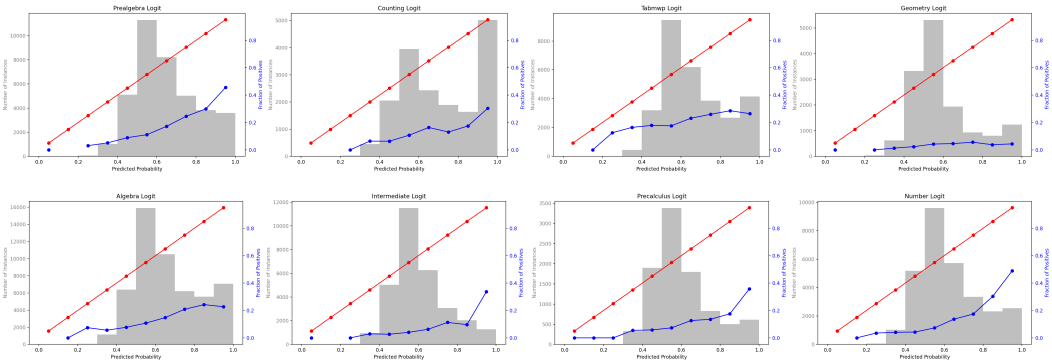


Figure C.9: Calibration curve of SORT logits of LLM in Dynamic Tool-Using

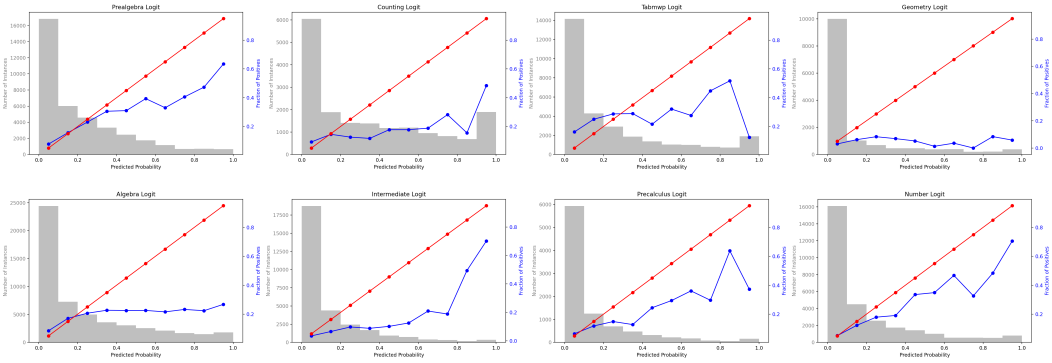


Figure C.10: Calibration curve of E.S.L. logits of LLM in Dynamic Tool-Using

Appendix D

APPENDIX TO CHAPTER 5

D.1 Additional Related Work

Here we provide additional discussion of search-based and learning-based symbolic regression methods.

Search based methods. SR is a challenging combinatorial optimization task. These methods search the space of possible equations while minimizing an objective function, relying on custom heuristics and parallelization to efficiently hypothesize and evaluate candidates [2, 8, 11, 13]. Notably, PySR [2] presents a multi-population evolutionary algorithm that has found practical utility in cosmology [3], international economics [14], and climate modeling [4]. While such approaches work well out-of-the-box, they are difficult to steer towards specific class of equations. ProAug extends such approaches by providing two flexible steering mechanisms: scientists can either manually steer search through natural language priors or such priors can be automatically extracted using dataset statistics.

Learning based methods. Another approach uses neural networks to accelerate SR [1, 5, 12]. These methods train networks on experimental data to either directly induce target expressions [1, 6, 7, 9] or to accelerate the search for target expressions [9, 10, 12, 15]. However, neural networks require retraining when datasets change significantly, necessitating scientists to monitor data distribution shifts and repeatedly retrain models – both computationally expensive tasks. ProAug follows prior work in primarily being a data-driven SR algorithm, but avoids the computational overheads by replacing learned neural representations with programmatically extracted insights, greatly increasing practical utility.

D.2 Problem details

Details of the problems we used in our experiments are shown in Table D.1.

Table D.1: Test data we use from LLM-SRBench.

Discipline	Name
Chemistry	CRK13, CRK12, CRK35, CRK31, CRK8
Biology	BPG16, BPG8, BPG9
Physics	PO12, PO37, PO40, PO24, PO8
Material Science	MatSci9, MatSci18, MatSci8
LSR-Transform	I.29.16_1_0, II.6.15b_1_0, II.11.27_1_0, I.34.1_2_0, I.30.3_0_0, II.24.17_1_1, III.15.27_2_0, I.12.2_3_0, III.10.19_2_1, I.37.4_0_1, II.11.17_4_0, II.6.15b_3_0, II.15.4_1_0, II.13.23_1_0, II.34.29b_3_0, I.11.19_2_0, I.32.5_1_1

D.3 Prompt Design

Prompts for `II.6.15b_1_0`

Zero-shot

You are a helpful assistant tasked with discovering mathematical function structures for scientific data science tasks. Complete the 'equation' function below, considering the dataset information.

```
"""
### Your Task:
FIRST, provide **BRIEF** reasoning inside a <thought>...</thought>
↪  block.

THEN, AFTER THE </thought> BLOCK, output your evolved mathematical
↪  function skeleton in Python. The function should begin with a
↪  'def' line and follow standard Python formatting.

Note: DO NOT use more than 10 params
"""




import numpy as np

#Initialize parameters
MAX_NPARAMS = 10
params = [1.0]*MAX_NPARAMS
```

```
def equation_v0(x_0: np.ndarray, x_1: np.ndarray, x_2: np.ndarray,
↪  x_3: np.ndarray, params: np.ndarray) -> np.ndarray:
    """ Mathematical function

    Args:
        x_0: A numpy array.
        x_1: A numpy array.
        x_2: A numpy array.
        x_3: A numpy array.

        params: Array of numeric constants or parameters to be
        ↪  optimized

    Return:
        A numpy array as the result of applying the mathematical
        ↪  function to the inputs.
    """
    output = params[0] * x_0 + params[1] * x_1 + params[2] * x_2 +
    ↪  params[3] * x_3 + params[4]
    return output


def equation_v1(x_0: np.ndarray, x_1: np.ndarray, x_2: np.ndarray,
↪  x_3: np.ndarray, params: np.ndarray) -> np.ndarray:
    """Improved version of `equation_v0`."""
```

Few-shot

You are a helpful assistant tasked with discovering mathematical function structures for scientific data science tasks. Complete the 'equation' function below, considering the dataset information.

```
"""
Find the mathematical function skeleton that represents y, given data
↪  on x_0, x_1, x_2, x_3.
```

```
### 12 Random Samples (X, Y) (Sorted by Y from small to large):
X[0] = [-32.193, 4.357, 4.774, 1.672], Y[0] = 2.588
X[1] = [-8.799, 2.543, 2.522, 2.127], Y[1] = 2.918
X[2] = [-45.200, 4.476, 3.700, 2.379], Y[2] = 6.494
X[3] = [-64.941, 3.223, 3.793, 1.486], Y[3] = 7.892
X[4] = [-46.401, 3.874, 4.408, 4.287], Y[4] = 11.650
X[5] = [-76.242, 2.469, 2.298, 1.248], Y[5] = 16.766
X[6] = [-84.225, 4.124, 1.593, 1.798], Y[6] = 23.048
X[7] = [-72.211, 3.124, 3.184, 3.350], Y[7] = 24.319
X[8] = [-43.300, 1.639, 4.213, 4.297], Y[8] = 26.954
X[9] = [-41.009, 1.516, 3.111, 3.784], Y[9] = 32.907
X[10] = [-83.370, 3.689, 2.829, 4.502], Y[10] = 35.962
X[11] = [-43.438, 1.281, 2.523, 3.718], Y[11] = 49.963


### Your Task:
FIRST, provide **BRIEF** reasoning inside a <thought>...</thought>
↪  block.

THEN, AFTER THE </thought> BLOCK, output your evolved mathematical
↪  function skeleton in Python. The function should begin with a
↪  'def' line and follow standard Python formatting.

Note: DO NOT use more than 10 params
"""



import numpy as np

#Initialize parameters
MAX_NPARAMS = 10
params = [1.0]*MAX_NPARAMS


def equation_v0(x_0: np.ndarray, x_1: np.ndarray, x_2: np.ndarray,
↪  x_3: np.ndarray, params: np.ndarray) -> np.ndarray:
```

```
    """ Mathematical function

    Args:
        x_0: A numpy array.
        x_1: A numpy array.
        x_2: A numpy array.
        x_3: A numpy array.

        params: Array of numeric constants or parameters to be
        ↪  optimized

    Return:
        A numpy array as the result of applying the mathematical
        ↪  function to the inputs.
    """
    output = params[0] * x_0 + params[1] * x_1 + params[2] * x_2 +
    ↪  params[3] * x_3 + params[4]
    return output


def equation_v1(x_0: np.ndarray, x_1: np.ndarray, x_2: np.ndarray,
↪  x_3: np.ndarray, params: np.ndarray) -> np.ndarray:
    """Improved version of `equation_v0`."""
```

Statistical Hint

You are a helpful assistant tasked with discovering mathematical function structures for scientific data science tasks. Complete the 'equation' function below, considering the dataset information.

```
"""
Find the mathematical function skeleton that represents y, given data
↪  on x_0, x_1, x_2, x_3.
The information of (X, Y) dataset including random sample points,
↪  smoothness estimation and simple basis fit scores are as follows:
```

```
### 12 Random Samples (X, Y) (Sorted by Y from small to large):
X[0] = [-32.193, 4.357, 4.774, 1.672], Y[0] = 2.588
X[1] = [-8.799, 2.543, 2.522, 2.127], Y[1] = 2.918
X[2] = [-45.200, 4.476, 3.700, 2.379], Y[2] = 6.494
X[3] = [-64.941, 3.223, 3.793, 1.486], Y[3] = 7.892
X[4] = [-46.401, 3.874, 4.408, 4.287], Y[4] = 11.650
X[5] = [-76.242, 2.469, 2.298, 1.248], Y[5] = 16.766
X[6] = [-84.225, 4.124, 1.593, 1.798], Y[6] = 23.048
X[7] = [-72.211, 3.124, 3.184, 3.350], Y[7] = 24.319
X[8] = [-43.300, 1.639, 4.213, 4.297], Y[8] = 26.954
X[9] = [-41.009, 1.516, 3.111, 3.784], Y[9] = 32.907
X[10] = [-83.370, 3.689, 2.829, 4.502], Y[10] = 35.962
X[11] = [-43.438, 1.281, 2.523, 3.718], Y[11] = 49.963
Statistics: {'mean_Y': 21.331, 'std_Y': 24.473, 'min_Y': 0.03,
↪  'max_Y': 335.661, 'mean_X_0': -44.025, 'std_X_0': 25.099,
↪  'min_X_0': -87.583, 'max_X_0': -0.436, 'mean_X_1': 3.011,
↪  'std_X_1': 1.151, 'min_X_1': 1.0, 'max_X_1': 5.0, 'mean_X_2':
↪  2.995, 'std_X_2': 1.156, 'min_X_2': 1.0, 'max_X_2': 5.0,
↪  'mean_X_3': 3.007, 'std_X_3': 1.156, 'min_X_3': 1.0, 'max_X_3':
↪  5.0, 'r2_Y_X_0': 0.245, 'r2_Y_X_1': 0.151, 'r2_Y_X_2': 0.15,
↪  'r2_Y_X_3': 0.111, 'r2_log(Y)_log(X_0)': 0.595,
↪  'r2_log(Y)_log(X_1)': 0.133, 'r2_log(Y)_log(X_2)': 0.133,
↪  'r2_log(Y)_log(X_3)': 0.134, 'r2_log(Y)_log(sin(x_0))': 0.0,
↪  'r2_log(Y)_log(cos(x_0))': 0.0, 'r2_log(Y)_log(exp(x_0))': 0.433,
↪  'r2_log(Y)_log(sqrt(x_0))': 0.595, 'r2_log(Y)_log(sin(x_1))':
↪  0.003, 'r2_log(Y)_log(cos(x_1))': 0.001,
↪  'r2_log(Y)_log(exp(x_1))': 0.128, 'r2_log(Y)_log(sqrt(x_1))':
↪  0.133, 'r2_log(Y)_log(sin(x_2))': 0.004,
↪  'r2_log(Y)_log(cos(x_2))': 0.001, 'r2_log(Y)_log(exp(x_2))':
↪  0.128, 'r2_log(Y)_log(sqrt(x_2))': 0.133,
↪  'r2_log(Y)_log(sin(x_3))': 0.003, 'r2_log(Y)_log(cos(x_3))':
↪  0.001, 'r2_log(Y)_log(exp(x_3))': 0.129,
↪  'r2_log(Y)_log(sqrt(x_3))': 0.134}


### Your Task:
```

```
Based on the statistics above, analyze and evolve the mathematical
↪  function skeleton.

IMPORTANT:
FIRST, provide **BRIEF** reasoning inside a <thought>...</thought>
↪  block about how you interpret the data and how it guides your
↪  function structure decisions.

THEN, AFTER THE </thought> BLOCK, output your evolved mathematical
↪  function skeleton in Python. The function should begin with a
↪  'def' line and follow standard Python formatting.

Note: DO NOT use more than 10 params
"""



import numpy as np

#Initialize parameters
MAX_NPARAMS = 10
params = [1.0]*MAX_NPARAMS


def equation_v0(x_0: np.ndarray, x_1: np.ndarray, x_2: np.ndarray,
↪  x_3: np.ndarray, params: np.ndarray) -> np.ndarray:
    """ Mathematical function

    Args:
        x_0: A numpy array.
        x_1: A numpy array.
        x_2: A numpy array.
        x_3: A numpy array.

        params: Array of numeric constants or parameters to be
        ↪  optimized
```

```
    Return:
        A numpy array as the result of applying the mathematical
        ↪  function to the inputs.
    """
    output = params[0] * x_0 + params[1] * x_1 + params[2] * x_2 +
    ↪  params[3] * x_3 + params[4]
    return output


def equation_v1(x_0: np.ndarray, x_1: np.ndarray, x_2: np.ndarray,
↪  x_3: np.ndarray, params: np.ndarray) -> np.ndarray:
    """Improved version of `equation_v0`."""
```

Prompts of ProAug

Equation Task Instruction

You are a helpful assistant tasked with discovering mathematical function structures for scientific data science tasks.

Physical Context

""" Find the mathematical function skeleton that represents the initial intensity, given data on the resulting intensity, the angle of diffraction, and the number of slits. """

Prior and To Be Completed Equations

```
def equation_v0(Int: np.ndarray, theta: np.ndarray, n: np.ndarray,
↪  params: np.ndarray) -> np.ndarray:
    """
    Args:
        Int: Resulting intensity.
        theta: Angle of diffraction.
        n: Number of slits.
```

```
        params: Array of numeric constants or parameters to be
        ↪  optimized

    Return:
        Initial intensity.
    """
    output = params[0] * Int + params[1] * theta + params[2] * n +
    ↪  params[3]
    return output

def equation_v1(Int: np.ndarray, theta: np.ndarray, n: np.ndarray,
↪  params: np.ndarray) -> np.ndarray:
    """Improved version of `equation_v0`."""
```

Data Analysis Instrution

You are to write Python code to analyze the dataset in a sandbox environment. The code output will be returned to aid reasoning. You can do investigate common statistics, propose engineered features and compositions, and apply statistical tools (correlation, regression, log–log fits, periodicity tests) to uncover potential functional forms such as additive, multiplicative, exponential, and power-law.

Data Analysis Code Template

```
def extract_dataset_info(Int_0: np.ndarray, Int: np.ndarray, theta:
↪  np.ndarray, n: np.ndarray):
    """
    Args:
        Int_0: Initial intensity.
        Int: Resulting intensity.
        theta: Angle of diffraction.
        n: Number of slits.

    Return:
        Dict summarizing the extracted dataset information.
    """
```

```
    # Basic statistics: mean, variance
    info['stats'] = {}
    # Features: propose candidate engineered features and
    ↪  compositions
    features = {}
    # Linear correlation
    info['correlations_with_Int_0'] = {}
    # R2 values from linear/log regression fits between Int_0 and
    ↪  features
    info['regression_r2'] = {}
    return info
```

Data Analysis Results

```
info = {'stats': {}, 'correlations_with_Int_0': {}, 'regression_r2':
↪  {}}
```

References

[1] Luca Biggio et al. *Neural Symbolic Regression that Scales*. 2021. arXiv: 2106.06427 [cs.LG]. URL: https://arxiv.org/abs/2106.06427.

[2] Miles Cranmer. "Interpretable machine learning for science with PySR and SymbolicRegression. jl". In: *arXiv preprint arXiv:2305.01582* (2023).

[3] Benjamin L Davis and Zehao Jin. "Discovery of a Planar Black Hole Mass Scaling Relation for Spiral Galaxies". In: *The Astrophysical Journal Letters* 956.1 (2023), p. L22.

[4] Arthur Grundner et al. "Data-driven equation discovery of a cloud cover parameterization". In: *Journal of Advances in Modeling Earth Systems* 16.3 (2024), e2023MS003763.

[5] Mikel Landajuela et al. "A unified framework for deep symbolic regression". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 33985–33998.

[6] Matteo Merler, Nicola Dainese, and Katsiaryna Haitsiukevich. "In-Context Symbolic Regression: Leveraging Language Models for Function Discovery". In: *arXiv preprint arXiv:2404.19094* (2024).

[7] Elliot Meyerson et al. *Language Model Crossover: Variation through Few-Shot Prompting*. 2024. arXiv: 2302.12170 [cs.NE]. URL: https://arxiv.org/abs/2302.12170.

[8] Brenden K Petersen et al. “Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients”. In: *arXiv preprint arXiv:1912.04871* (2019).

[9] Bernardino Romera-Paredes et al. “Mathematical discoveries from program search with large language models”. In: *Nature* 625.7995 (2024), pp. 468–475.

[10] Ameesh Shah et al. “Learning differentiable programs with admissible neural heuristics”. In: *Advances in neural information processing systems* 33 (2020), pp. 4940–4952.

[11] Trevor Stephens. *gplearn: Genetic Programming in Python, with a scikit-learn inspired API*. Accessed: 2024-05-22. 2024. URL: https://github.com/trevorstephens/gplearn.

[12] Wassim Tenachi, Rodrigo Ibata, and Foivos I. Diakogiannis. “Deep Symbolic Regression for Physics Guided by Units Constraints: Toward the Automated Discovery of Physical Laws”. In: *ApJ* 959.2, 99 (Dec. 2023), p. 99. DOI: 10.3847/1538-4357/ad014c. arXiv: 2303.03192 [astro-ph.IM].

[13] Silviu-Marian Udrescu and Max Tegmark. “AI Feynman: A physics-inspired method for symbolic regression”. In: *Science advances* 6.16 (2020), eaay2631.

[14] Sergiy Verstyuk and Michael R Douglas. “Machine learning the gravity equation for international trade”. In: *Available at SSRN 4053795* (2022).

[15] Hengzhe Zhang et al. “RAG-SR: Retrieval-Augmented Generation for Neural Symbolic Regression”. In: *International Conference on Learning Representations*. 2025.