Title

Enhancing Simulation with Machine Learning and Artificial Intelligence

ISBN

9798288864544

Author

Zhang, Haoting

Publication Date

2025-05-15

Peer reviewed|Thesis/dissertation

Enhancing Stochastic Simulation with Machine Learning and Artificial Intelligence

By

Haoting Zhang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Industrial Engineering and Operations Research

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Zeyu Zheng, Co-Chair
Professor Rhonda Righter, Co-Chair
Professor Zuo-Jun (Max) Shen
Assistant Professor Park Sinchaisri

Spring 2025

Enhancing Stochastic Simulation with Machine Learning and Artificial Intelligence

Abstract

Enhancing Stochastic Simulation with Machine Learning and Artificial Intelligence

by

Haoting Zhang

Doctor of Philosophy in Engineering - Industrial Engineering and Operations Research

University of California, Berkeley

Associate Professor Zeyu Zheng, Co-Chair

Professor Rhonda Righter, Co-Chair

Stochastic simulation serves as a vital tool for approximating and analyzing complex systems under uncertainty across domains such as healthcare, finance, and supply chain management. However, when systems involve high-dimensional information and require large numbers of simulated samples, classical simulation methods may encounter challenges related to model specification and computational complexity. These limitations can impede timely decision-making and effective system optimization in today's data-intensive environments.

Recent advances in machine learning (ML) and artificial intelligence (AI) offer promising solutions to these challenges. By capturing highly nonlinear relationships and patterns within stochastic simulation models, ML and AI techniques have the potential to enhance both approximation accuracy and computational efficiency. At the same time, while ML and AI significantly enhance the capabilities of stochastic simulation, simulation methods also contribute to ML and AI by providing cost-effective and sample-efficient techniques—thereby helping democratize access to advanced AI tools.

In line with this mutually beneficial relationship between stochastic simulation and ML/AI, this dissertation introduces: (1) a new surrogate model that integrates neural networks and Gaussian processes to approximate complex stochastic systems; and (2) a simulation optimization procedure that efficiently facilitates language model prompt selection to enhance language model performance.

Contents

List of Figures

List of Tables

Acknowledgments

I am profoundly grateful for the support and guidance I have received throughout my Ph.D. journey. First and foremost, I extend my deepest appreciation to my advisors, Professor Zeyu Zheng and Professor Rhonda Righter, whose expertise and insightful guidance have been invaluable to both my research and personal development. I began working with Professor Zheng during my master's program in our department, and his mentorship was a major reason I chose to return and pursue a Ph.D. in the same department. I had the privilege of serving as a teaching assistant for two of Professor Righter's courses, during which I not only gained valuable teaching experience but also learned a great deal about effective instruction and mentorship from her. Both Professor Zheng and Professor Righter have been incredibly patient and generous with their time, offering support in all aspects of my academic journey. Beyond academics, they have also shown genuine care for my well-being, providing mental and emotional support during difficult times.

I am also truly grateful to Professor Zuo-Jun (Max) Shen for serving on my dissertation committee. His invaluable suggestions, constructive feedback, and insightful discussions have played a crucial role in shaping both my research and career plans. I would also like to thank Professor Park Sinchaisri for serving on my dissertation committee. I sincerely appreciate his time, support, and thoughtful guidance throughout this journey. In addition, I would like to express my heartfelt thanks to Dr. Qing Zhu from Berkeley Lab, whose invaluable advice and encouragement have greatly influenced both my research and personal growth.

I owe a great deal of gratitude to my seniors, peers, and friends at Berkeley: Sheng Liu, Renyuan Xu, Junyu Cao, Han Feng, Meng Qi, Mengxin Wang, Anran Hu, Haoyang Cao, Julie Mulvaney Kemp, Marie Pelagie Elimbi Moudio, Shunan Jiang, Meng Li, Mo Liu, Hyungki Im, Yufeng Zheng, Yiyang Wu, Jiahao Fu, Tor Nitayanont, Haixiang Zhang, Zhe Fu, Mengzi Guo, Donghao Ying, Yunkai Zhang, Shuo Sun, Yuhang Wu, Hao Wang, Zhihao Liu, Lingfei Zhong, Pengxiang Zhou, Sukanya Kudva, Joo Seung Lee, and Hannah Lucia Davalos. Their encouragement and companionship have been a constant source of strength throughout my Ph.D. journey. In particular, I want to thank Jinghai He. I first met him and began collaborating with him when he was an undergraduate student. His later decision to join Professor Zheng's group allowed our friendship and collaboration to continue and deepen over the years. I am especially grateful to my former roommates, Jingxu Xu and Jiaming Wang. We have known each other since my master's program, and the memories we have shared over the years are full of joy and support. They were always there for me, especially during the difficult times of the pandemic. Living with them brought laughter and comfort during some of the most stressful periods. I know I was not always the perfect roommate, and I'm truly thankful for their patience and understanding. I would also like to thank another roommate, Yuhao Ding, who has been like a big brother to me. His humor and warmth never failed to lift my spirits. Special thanks go to Ziyang Liu and Runhan Xie for their companionship and support during the final year of my Ph.D. journey. The job application process was far more grueling than I had imagined, and I truly cannot picture going through it without them by my side. My heartfelt thanks also go to Hansheng Jiang, the very first

person in Berkeley to welcome me. She generously shared advice on both academics and life in Berkeley, helping me settle in with ease. I would like to thank Tianyi Lin. His perseverance in academia has been deeply inspiring, and I greatly appreciate the guidance he has offered me regarding research and career planning. Finally, I want to express my special gratitude to Yunduan Lin for her unwavering encouragement and thoughtfulness. We supported each other through many academic challenges, and her advice and insights—particularly during the job market—were invaluable. Her friendship has made a meaningful difference in both my professional and personal growth.

I would like to sincerely thank my managers and mentors during my internships at Amazon in Bellevue, including Shekhar Jain, Manan Chopra, Albert Kuo, Mitchell Joblin, and Rajabi Siamak. Their guidance, support, and thoughtful feedback provided me with a valuable and enriching learning experience.

I would also like to express my heartfelt gratitude to my high school classmates who are now in the United States, including Hanzhang Jin, Liwei Jiang, Jiatong Xie, Liangchen Liu, Wei Kuang, Rongguan Yin, Zice Tang, Mutian Shen, and Yichun Hu. Reuniting with them and reminiscing about our high school days has brought me great joy and comfort throughout my time here. I am especially thankful to Yuchun Jin, whose generosity and friendship have been a constant source of support. During weekends, he would often drive from the southern Bay Area—despite the long distance and busy schedule—to take me out for dinner and help with errands like grocery shopping. His thoughtfulness and willingness to lend a hand, often without being asked, made a meaningful difference during times when I felt overwhelmed or isolated. I am deeply grateful for his presence and unwavering support.

I would also like to extend my heartfelt thanks to Donglin Zhan, currently a Ph.D. candidate in the Department of Electrical Engineering at Columbia University. Donglin and I have been close friends for nearly a decade, since the very first day of our undergraduate studies in mathematics at Sichuan University. Together, we made the decision to pursue Ph.D. degrees in the United States after graduation. The path was far from easy—our limited research experience and the challenges brought by the pandemic made it particularly difficult—but we supported each other throughout. Our collaboration began even before we started our Ph.D. programs, and I still vividly remember the moment our co-authored paper was accepted—my first journal publication—while we were both in Bellevue for our summer internships. The experiences we have shared go far beyond what words can capture here. I am deeply grateful for his companionship, encouragement, and support over the years, and I wish him continued success and happiness in all his endeavors.

I am also grateful to those whose companionship and support helped me through some of the most challenging moments of this journey.

Last but certainly not least, I wish to express my deepest gratitude to my parents. Their unconditional love, unwavering support, and constant encouragement have been the foundation of everything I have achieved. As the only child in the family, I sometimes feel a deep sense of guilt for not being able to stay close to them. Yet, they have never hesitated to make sacrifices and have always believed in my potential. I am eternally grateful for their strength, patience, and enduring faith in me.

Chapter 1

Introduction

Stochastic simulation serves as a vital tool for approximating and analyzing complex systems under uncertainty across domains such as healthcare, finance, and supply chain management [191, 189]. It enables decision-makers to evaluate system performance, assess risk, and optimize strategies in situations where analytical solutions are infeasible [190]. However, as systems become increasingly data-rich and high-dimensional, classical simulation methods face growing challenges. Accurate model specification becomes increasingly difficult as it often requires extensive prior knowledge, and the computational burden of generating large numbers of simulated samples can be substantial. These challenges are further exacerbated by storage constraints and the need for rapid turnaround in real-time or near-real-time applications. As a result, classical stochastic simulation methods may struggle to keep pace with the demands of contemporary decision-making environments, limiting their effectiveness in supporting timely and data-driven applications.

Recent advances in machine learning (ML) and artificial intelligence (AI) offer promising solutions to the challenges faced by classical stochastic simulation methods. In particular, ML models—especially deep neural networks—are capable of capturing complex, highly nonlinear relationships and patterns that classical models often struggle to represent [194, 201], thereby improving the fidelity of simulation-based approximations. These techniques can also reduce computational burdens by serving as surrogate models or emulators, enabling faster evaluation of complex systems without the need for repeated, expensive simulations. Moreover, ML algorithms can adapt to high-dimensional input spaces and uncover latent structures in data, enhancing the scalability and flexibility of simulation analyses.

On the other hand, the "black-box" nature of some ML/AI models limits the interpretability of system outputs, which is crucial in applications that require transparency. Therefore, instead of fully relying on ML/AI models, we integrate them with classical stochastic models to (1) enhance model flexibility, approximation accuracy, and computational efficiency, while (2) retaining the interpretable structure of stochastic models and enabling explicit quantification of approximation uncertainty [199, 193, 192, 198].

At the same time, the relationship between simulation and ML/AI is not one-directional. Simulation methods—particularly those designed for uncertainty quantification and sample

efficiency—also play a critical role in supporting and advancing ML and AI applications, which is especially beneficial for small businesses and individuals with limited resources. Specifically, we leverage simulation optimization methods to facilitate the training and application of language models [195, 196], providing cost-effective approaches that contribute to the democratization of AI by making advanced tools more accessible and applicable across a broader range of domains and organizations.

In line with the mutual relationship between stochastic simulation and ML/AL, we provide a summary of the next two chapters of this dissertation: (1) employing ML techniques to improve model flexibility and approximation accuracy of stochastic simulation models (**Chapter 2**); and (2) applying sample-efficient and cost-effective simulation methods to AI applications (**Chapter 3**).

- **Chapter 2:** Contextual decision-making problems, in which sequential decisions are made taking into account the current contextual information, have witnessed extensive applications in various fields such as online content recommendation, personalized healthcare, and autonomous vehicles, where a core practical challenge is to select a suitable surrogate model for capturing unknown complicated reward functions. It is often the case that both high approximation accuracy and explicit uncertainty quantification are desired. In this work, we propose a neural network-accompanied Gaussian process (NN-AGP) model, which leverages neural networks to approximate the unknown and potentially complicated reward function regarding the contextual variable, and maintains a Gaussian process surrogate model with respect to the decision variable. Our model is shown to outperform existing approaches by offering better approximation accuracy thanks to the use of neural networks and possessing explicit uncertainty quantification from the Gaussian process. We also analyze the maximum information gain of the NN-AGP model and prove regret bounds for the corresponding algorithms. Moreover, we conduct experiments on both synthetic and practical problems, illustrating the effectiveness of our approach.

  This chapter is based on

  > [193] Haoting Zhang, Jinghai He, Rhonda Righter, Zuo-Jun (Max) Shen, Zeyu Zheng. "Contextual Gaussian Process Bandits with Neural Networks." *Advances in Neural Information Processing Systems* 36 (2023).

- **Chapter 3:** With the advancement in generative language models, the selection of prompts has gained significant attention in recent years. A prompt is an instruction or description provided by the user, serving as a guide for the generative language model in content generation. Despite existing methods for prompt selection that are based on human labor, we consider facilitating this selection through simulation optimization, aiming to maximize a pre-defined score for the selected prompt. Specifically, we propose a two-stage framework. In the first stage, we determine a feasible set of prompts in sufficient numbers, where each prompt is represented by a moderate-dimensional vector.

In the subsequent stage for evaluation and selection, we construct a surrogate model of the score regarding the moderate-dimensional vectors that represent the prompts. We propose sequentially selecting the prompt for evaluation based on this constructed surrogate model. We prove the consistency of the sequential evaluation procedure in our framework. We also conduct numerical experiments to demonstrate the efficacy of our proposed framework, providing practical instructions for implementation.

This chapter is based on

> [196] Haoting Zhang, Jinghai He, Rhonda Righter, Zeyu Zheng. "Language Model Prompt Selection via Simulation Optimization." Under Major Revision at *Management Science*.

Chapter 2

Contextual Gaussian Process Bandits with Neural Networks

2.1 Introduction

Various applications, including online content recommendation [1], healthcare [62, 26], and autonomous vehicles [15], demand the sequential selection of a decision variable, conditional on the observed contextual variable representing the state of the environment in each round. These applications can generally be framed as contextual bandit problems [11, 114, 118, 3], especially when the reward function associated with each pair of decision and contextual variables is unknown. When both the decision and contextual variables are drawn from continuous sets, a significant challenge is selecting an appropriate surrogate model to approximate the reward function, considering both approximation accuracy and uncertainty quantification. A common approach to alleviate this issue is to employ a Gaussian process (GP) to model the reward function, yielding the GP bandit method [156, 158]. Indeed, GP has proven to be an effective surrogate model to address the exploration-exploitation trade-off in estimating the unknown function while optimizing over it; see [174, 146, 180, 76, 96]. On the other hand, most of the existing GP bandit literature does not take the exogenous contextual variable into consideration, despite its critical role in capturing effects beyond the decision variable that influence the reward – effects that are integral to many of the applications previously mentioned [94, 116]. When the contextual variable is included in GP bandit problems, previous work largely adopts a GP to jointly model the reward function with both contextual and decision variables, employing a composite kernel that is either the sum or product of two separate kernels; see [111, 18].

While GP-based bandit methods have proven effective in various applications [6, 16, 182, 183, 169, 14], they may fall short in scenarios where the reward function exhibits intricate dependence on complex contextual variables, for example, time-varying rewards [24, 53] and graph-structured contextual variables [135]. Specifically, it is a challenge to pre-define an appropriate composite kernel function for the joint GP, which is critical for the performance

of the corresponding bandit algorithms, as documented by [35, 159]. Neural networks (NN), on the other hand, have been utilized elsewhere as surrogate models for the reward function in bandit problems [27, 105, 106], thanks to their flexibility and strong approximation power. However, they bring their own set of challenges. The "black-box" nature of the neural network hinders explicit uncertainty quantification and complicates theoretical analysis of the associated algorithms. In particular, the acquisition functions guiding point selection in these algorithms necessitate an approximation of uncertainty [202, 105]. Although this approximation tends to be accurate when the NN's width is large, this can also lead to overparameterization.

Contribution. This chapter proposes a *neural network-accompanied Gaussian process* (NN-AGP) model for solving contextual bandit problems, especially when the reward functions have intricate dependence on complex contextual variables. The proposed model is an inner product of a neural network and a multi-output GP, where the neural network captures the dependence of the reward function on the contextual variables, and the GP is employed to model the mapping from the decision variable to the reward. Our model generates a joint GP with both contextual and decision variables, which outperforms the existing GP-based bandit methods by specifying the data-driven kernel function through the lens of neural networks, thereby leading to an accurate approximation for the reward function. Moreover, compared with entirely relying on NN's, our model stands out due to the explicit GP expression with respect to the decision variable. This feature enables bandit algorithms with NN-AGP to be implemented efficiently with existing GP-based acquisition functions and provides a theoretical guarantee of the regret bounds. Our main contributions can be summarized as follows:

1. We propose an NN-AGP model and its upper confidence bound (UCB) algorithm for contextual bandit problems, referred to as NN-AGP-UCB. Our algorithm offers a data-driven procedure to specify the kernel functions of the joint GP, thereby achieving superior accuracy and model flexibility. We also prove the upper bound for both the maximum information gain of the NN-AGP model and the regret of the NN-AGP-UCB algorithm.

2. We conduct the experiments to evaluate our approach for complex reward functions, including those with time-varying dependence on sequenced and graph-structured contextual variables. Experimental results demonstrate the superiority of our approach over existing approaches that entirely rely on either GP or NN.

Related Work

Since the seminal work by Srinivas et al. [156], the Gaussian process (GP) bandit problem has been extensively studied, where the bandit feedback is modeled as a GP regarding the decision variable (arms to be pulled). Some recent work includes [59, 55, 24, 23, 22, 31, 121]. In addition, GP bandits are also related to Bayesian optimization (BO) problems [67,

33, 184, 66, 56, 49, 167, 100], where both problems consider optimizing black-box functions and therefore require surrogate models (GP in particular). When the number of decision variables is finite, the black-box optimization problem is also known as Ranking & Selection (R&S) [79, 139, 133, 7, 170], where GP models are widely employed as well; see [32, 112, 151, 119]. Our NN-AGP model can also be employed in (contextual) BO or R&S, but the discussion is beyond the scope of this chapter.

Here we specifically take the exogenous contextual variables into consideration. Previous work employs multiplicative and additive kernels to incorporate continuous context spaces into the scalar GP; see [111]. Other work considers safe contextual Bayesian optimization, employing a similar strategy to construct composite kernels; see [73, 18]. Another line of research explores distributionally robust BO [110, 161], where the contextual variable distribution is selected from an ambiguity set. The methodology of representing the objective function using a joint GP has also been widely used in contextual policy search; see [136, 35, 71].

The connection between GP and NN has been explored in [115, 127], documenting that NN's with infinite width approach a GP model when the weight parameters are assigned with Gaussian priors. In addition, deep GP's have been proposed to enhance the model flexibility of neural networks where variational inference is employed; see [47, 181]. GP models in which the parameters are represented by neural networks are studied in [199, 197].

2.2 Main Procedure

We consider a problem of sequentially selecting a system's input variable (decision variable) for T (not necessarily known a priori) rounds. In each round, we receive a contextual variable $\boldsymbol{\theta}_t \in \Theta \subset \mathbb{R}^{d'}$ from a set Θ, and select a decision variable $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^d$ from a set $\mathcal{X}$ of decisions. We then receive an observation

$$y_t = f(\mathbf{x}_t; \boldsymbol{\theta}_t) + \epsilon_t,$$

where $f(\mathbf{x}; \boldsymbol{\theta})$ is the reward (objective) function and $\epsilon_t \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_\epsilon^2)$ denotes the noise that is independent of both the contextual variables and decision variables. We consider the scenarios when both Θ and $\mathcal{X}$ are continuous and $\boldsymbol{\theta}_t$'s are fully exogenous. That is, the selection of the decision variable in each round does not influence the future contextual variables. Although we focus on unconstrained problems in this work, our proposed model can be employed to approximate the constraints in optimization problems as well; see [18]. We also note that, for the description and discussion of our approach, the contextual variable is represented by a vector. However, we show through experiments in Section 2.4 that our approach is applicable to contextual variables with other structures.

Since $f(\mathbf{x}, \boldsymbol{\theta})$ is unknown, we will not generally be able to choose the optimal action, and will thus incur regret $r_t = \sup_{\mathbf{x}' \in \mathcal{X}} f(\mathbf{x}', \boldsymbol{\theta}_t) - f(\mathbf{x}_t, \boldsymbol{\theta}_t)$, indicating the difference between the optimal reward and the reward we actually receive in each round. After T rounds, the

cumulative regret is $\mathcal{R}_T = \sum_{t=1}^{T} r_t$. Our goal is to develop an algorithm that achieves sub-linear contextual regret, i.e., $\mathcal{R}_T/T \to 0$ for $T \to \infty$, which requires a statistical model of the reward function with respect to both context variables and decision variables.

We specifically consider a reward function in the form

$$f(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{p}(\mathbf{x}), \tag{2.1}$$

where $\boldsymbol{g}(\boldsymbol{\theta})$ and $\mathbf{p}(\mathbf{x})$ are both m-dimensional vector-valued (unknown) functions, and $m \in \mathbb{N}$ is a user-selected quantity to indicate the complexity of the function. There are two main reasons for considering this reward function. First, this formulation is a generalization of linear contextual bandits, where the reward function is the inner product of the contextual variables and the unknown parameters. Here, we assume that the inner product is taken with two vector-valued functions with respect to decision variables and the contextual variable, which is similar in spirit to [42, 186, 74, 204]. Second, this reward function is consistent with the tensor-product approximation of a general function; see [50, 52, 87]. Therefore, further analysis on model mis-specification of the reward function can be supported by existing results of tensor-product approximation.

Here we specifically assume that $\boldsymbol{g}(\boldsymbol{\theta})$ is a vector-valued deterministic mapping from $\mathbb{R}^{d'}$ to $\mathbb{R}^m$, represented by a neural network with a given structure and some weight parameter $\mathbf{W}$. In addition, $\mathbf{p}(\mathbf{x})$ is a multi-output Gaussian process (MGP) defined on $\mathcal{X} \subset \mathbb{R}^d$. The MGP model is a generalization of the scalar-valued GP, where the output $\mathbf{p}(\mathbf{x})$ at each $\mathbf{x}$ is an m-dimensional vector. The MGP model captures not only the dependence between two outputs but also the dependence between different entries of each output. Thus, the covariance of an MGP $\mathbf{p}(\mathbf{x})$ is represented by a matrix-valued covariance function, denoted by $\mathcal{K}(\mathbf{x}, \mathbf{x}')$, and the vector of parameters involved in the MGP is denoted by Φ. We postpone the detailed description of the NN-AGP model to Section 2.3 and conclude our brief introduction of NN-AGP with an associated proposition, which follows easily from the fact that the normal distribution is preserved under linear transformations.

Proposition 2.1. *The reward function $f(\mathbf{x}; \boldsymbol{\theta})$ is a scalar-valued mean-zero Gaussian process with respect to $\mathbf{x}$ and $\boldsymbol{\theta}$. The kernel function of this Gaussian process is*

$$\tilde{K}\left((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')\right) = \boldsymbol{g}(\boldsymbol{\theta})^\top \mathcal{K}(\mathbf{x}, \mathbf{x}')\, \boldsymbol{g}(\boldsymbol{\theta}'),$$

where $\mathcal{K}(\mathbf{x}, \mathbf{x}')$ is the covariance of the MGP.

Next, we provide a bandit algorithm with the NN-AGP model, during which the surrogate model is sequentially learned from data. We name the algorithm *neural network-accompanied Gaussian process upper confidence bound* (NN-AGP-UCB). Suppose we are now in round t and observe the contextual variable $\boldsymbol{\theta}_t$. In addition, we also have the historic data $\mathcal{D}_{t-1} = \{(\boldsymbol{\theta}_1, \mathbf{x}_1, y_1), (\boldsymbol{\theta}_2, \mathbf{x}_2, y_2), \ldots, (\boldsymbol{\theta}_{t-1}, \mathbf{x}_{t-1}, y_{t-1})\}$ in hand. Denote by $\mathbf{y}_{t-1} = (y_1, y_2, \ldots, y_{t-1})$ the vector of observations. The selection of the next decision variable $\mathbf{x}_t$ depends on the posterior distribution of the reward function, which is $f(\mathbf{x}; \boldsymbol{\theta}_t) \mid \mathcal{D}_{t-1} \sim$

$\mathcal{N}\left(\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right),\sigma^2_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)\right)$. Here

$$\begin{aligned}\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)=&\tilde{\mathcal{K}}^{\top}_{(\mathbf{x};\boldsymbol{\theta}_t)}\left[\tilde{\mathcal{K}}_{\mathcal{D}_{t-1}}+\sigma^2_{\epsilon}I_{t-1}\right]^{-1}\mathbf{y}_{t-1},\\ \sigma^2_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)=&\boldsymbol{g}(\boldsymbol{\theta}_t)^{\top}\mathcal{K}\left(\mathbf{x},\mathbf{x}\right)\boldsymbol{g}(\boldsymbol{\theta}_t)-\tilde{\mathcal{K}}^{\top}_{(\mathbf{x};\boldsymbol{\theta}_t)}\left[\tilde{\mathcal{K}}_{\mathcal{D}_{t-1}}+\sigma^2_{\epsilon}I_{t-1}\right]^{-1}\tilde{\mathcal{K}}_{(\mathbf{x};\boldsymbol{\theta}_t)},\end{aligned} \tag{2.2}$$

where $\tilde{\mathcal{K}}_{(\mathbf{x};\boldsymbol{\theta}_t)}$ denotes the covariance vector between $f\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ and $\{f\left(\mathbf{x}_\tau;\boldsymbol{\theta}_\tau\right)\}_{\tau=1}^{t-1}$. In addition, for the $(t-1)\times(t-1)$-dimensional covariance matrix for historical data $\tilde{\mathcal{K}}_{\mathcal{D}_{t-1}}$, the (i,j)-th entry is $\boldsymbol{g}(\boldsymbol{\theta}_i)^{\top}\mathcal{K}\left(\mathbf{x}_i,\mathbf{x}_j\right)\boldsymbol{g}(\boldsymbol{\theta}_j)$ as in **Proposition 2.1**. The required parameters in (2.2), including the weight parameters $\mathbf{W}$ of the neural network $\boldsymbol{g}(\boldsymbol{\theta})$, the parameters Φ involved in the MGP $\mathbf{p}(\mathbf{x})$ and the variance σ^2_{ϵ} of the noise ϵ_t, are all learned and updated with the observations through (2.5), which we will discuss in Section 2.3.

In terms of the acquisition function (the function that decides the decision variable in the following iteration), we employ the contextual Gaussian process-upper confidence bound (CGP-UCB) introduced in [111]. That is, we decide $\mathbf{x}_t$ as

$$\mathbf{x}_t=\arg\max_{\mathbf{x}\in\mathcal{X}}\left\{\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)+\beta_t^{1/2}\sigma_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)\right\}, \tag{2.3}$$

where $\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ and $\sigma_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ are the posterior mean and standard deviation of $f\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ as calculated in (2.2). The optimization problem (2.3) can be solved efficiently by global search heuristics, as suggested in [30]. In addition, β_t is a user-selected hyper-parameter in each round, addressing the exploration-exploitation trade-off; see discussions in Section 2.3. The procedure for NN-AGP-UCB is summarized in **Algorithm 2.1**. We note that other commonly selected acquisition functions for GP bandit problems or Bayesian optimization can be employed with NN-AGP as well, including knowledge gradient [147, 179, 58] and Thompson sampling [44, 145]. We postpone the discussion of these acquisition functions to the supplements.

Algorithm 2.1 NN-AGP-UCB

Input: Initial values of $(\mathbf{W},\Phi,\sigma^2_{\epsilon})$;
for $t=1,2,\ldots,T$ **do**
  Observe the contextual variable $\boldsymbol{\theta}_t$;
  Choose $\mathbf{x}_t=\arg\max_{\mathbf{x}\in\mathcal{X}}\left\{\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)+\beta_t^{1/2}\sigma_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)\right\}$;
  Sample y_t at $(\boldsymbol{\theta}_t,\mathbf{x}_t)$;
  Update $\left(\hat{\mathbf{W}}_t,\hat{\Phi}_t,\hat{\sigma^2_{\epsilon;t}}\right)$ as in (2.5) ;
end for

2.3 Statistical Properties

Specification of NN-AGP

We describe the NN-AGP model here, which is employed as the surrogate for the reward function. Recall that the reward function with the pair of contextual and decision variables $(\boldsymbol{\theta}, \mathbf{x})$ is $f(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{p}(\mathbf{x})$, where $\boldsymbol{g}(\boldsymbol{\theta})$ is a vector-valued neural network (with weight parameters $\mathbf{W}$) from $\mathbb{R}^{d'}$ to $\mathbb{R}^m$ and $\mathbf{p}(\mathbf{x})$ is an m-dimensional output Gaussian process defined on $\mathcal{X} \subset \mathbb{R}^d$. To be more specific, the m outputs $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_m)^\top$ are assumed to follow a multi-output Gaussian process (MGP) as

$$\mathbf{p}(\mathbf{x}) \sim \mathcal{MGP}\left(\mathbf{0}, \mathcal{K}\left(\mathbf{x}, \mathbf{x}'\right)\right).$$

Here, $\mathcal{K}\left(\mathbf{x}, \mathbf{x}'\right)$ denotes the covariance matrix of $\mathbf{p}(\mathbf{x})$ and $\mathbf{p}\left(\mathbf{x}'\right)$, defined as $\mathcal{K}\left(\mathbf{x}, \mathbf{x}'\right) \doteq \begin{pmatrix} K_{11}\left(\mathbf{x}, \mathbf{x}'\right) & \cdots & K_{1m}\left(\mathbf{x}, \mathbf{x}'\right) \\ \vdots & \ddots & \vdots \\ K_{m1}\left(\mathbf{x}, \mathbf{x}'\right) & \cdots & K_{mm}\left(\mathbf{x}, \mathbf{x}'\right) \end{pmatrix}$ which is positive and semi-definite. The (l, l')-th entry $K_{ll'}\left(\mathbf{x}, \mathbf{x}'\right)$ represents the covariance (similarity) between outputs $\mathbf{p}_l(\mathbf{x})$ and $\mathbf{p}_{l'}\left(\mathbf{x}'\right)$. The NN-AGP model results in a scalar-valued Gaussian process with both the contextual and decision variables and therefore facilitates explicit acquisition functions and theoretical analysis. This GP representation arises from the linear structure between the MGP and the mapping $\boldsymbol{g}(\boldsymbol{\theta})$.

To specify the covariance matrix, we adopt the collaborative multi-output Gaussian process model [132] as a representative, which generalizes the commonly-used linear model of coregionalization (LMC) and the intrinsic coregionalization model (ICM); see [123]. That is, the MGP is determined by the linear transformation of multiple independent scalar-valued Gaussian processes as

$$\mathbf{p}_l(\mathbf{x}) = \sum_{q=1}^{Q} a_{l,q} u_q(\mathbf{x}) + v_l(\mathbf{x}). \tag{2.4}$$

Here, $\mathbf{p}_l(\mathbf{x})$ is the l-th element of $\mathbf{p}(\mathbf{x})$, Q is the number of involved GP's, $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ and $\{v_l\left(\mathbf{x}\right)\}_{l=1}^{m}$ are independent scalar-valued GP's, and the $a_{l,q}$'s are coefficient parameters. In this way, the correlation between different entries in the MGP $\mathbf{p}(\mathbf{x})$ is captured by $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ through the $a_{l,q}$'s. Morever, $v_l\left(\mathbf{x}\right)$ represents specific independent features of $\mathbf{p}_l(\mathbf{x})$ itself, for $l = 1, 2, \ldots, m$. Suppose the kernel functions of the $u_q(\mathbf{x})$'s and $v_l(\mathbf{x})$'s are $k_q\left(\mathbf{x}, \mathbf{x}'\right)$'s and $\tilde{k}_l\left(\mathbf{x}, \mathbf{x}'\right)$'s and all these kernel functions are less than or equal to one, as is regularly assumed [156, 158, 111]. Then the matrix-valued kernel function of $\mathbf{p}(\mathbf{x})$ is $\mathcal{K}\left(\mathbf{x}, \mathbf{x}'\right) = \sum_{q=1}^{Q} \mathbf{A}_q k_q\left(\mathbf{x}, \mathbf{x}'\right) + \operatorname{Diag}\left\{\tilde{k}_1\left(\mathbf{x}, \mathbf{x}'\right), \ldots, \tilde{k}_m\left(\mathbf{x}, \mathbf{x}'\right)\right\}$. Here, $\mathbf{A}_q$ denotes the semi-definite matrix in which the (l, l')-th entry is $a_{l,q} a_{l',q}$. In some applications, the parameters involved in the kernel functions $k_q\left(\mathbf{x}, \mathbf{x}'\right)$ and $k_l\left(\mathbf{x}, \mathbf{x}'\right)$, and the coefficients $a_{l,q}$'s are not known in advance. We denote these unknown parameters and coefficients in the MGP as Φ. In addition to the model (2.4), other types of MGP's can be employed in our methodology as well [25], while the selection of model (2.4) enables the theoretical analysis of our approach.

In terms of the mapping $\boldsymbol{g}(\boldsymbol{\theta})$, since we have no prior knowledge, we select the neural network as the surrogate model, due to 1) its strong approximation power for intricate dependence on complex variables; 2) its flexibility of adaptation to different application scenarios (e.g. time-series or graph-structured contextual variables) and 3) the availability of fruitful methods and tools for the training procedure. Given the data $\mathcal{D}_t = \{(\boldsymbol{\theta}_1, \mathbf{x}_1, y_1), (\boldsymbol{\theta}_2, \mathbf{x}_2, y_2), \ldots, (\boldsymbol{\theta}_t, \mathbf{x}_t, y_t)\}$, the learning of unknown parameters in the MGP and the weight parameters in the neural network (as well as the noise variance) is through maximum likelihood estimation (MLE). That is

$$\left(\hat{\mathbf{W}}_t, \hat{\Phi}_t, \hat{\sigma^2_{\epsilon;t}}\right) = \arg\max_{(\mathbf{W}, \Phi, \sigma^2_\epsilon)} L_t\left(\mathbf{W}, \Phi, \sigma^2_\epsilon\right), \tag{2.5}$$

where the likelihood function is $L_t = -\ln\left[\left|\tilde{K}_{\mathcal{D}_t} + \sigma^2_\epsilon I_t\right|\right] - \mathbf{y}_t^\top \left[\tilde{K}_{\mathcal{D}_t} + \sigma^2_\epsilon I_t\right]^{-1} \mathbf{y}_t$. Here, $\mathbf{y}_t = (y_1, y_2, \ldots, y_t)$ is the vector of observations and $\tilde{K}_{\mathcal{D}_t}$ is the covariance matrix of the data $\mathcal{D}_t$. The parameters $\mathbf{W}$ and Φ are contained in this covariance matrix. That is, instead of pre-defining a kernel function of the GP, the kernel function of NN-AGP is specified through learning the neural network from the data, yielding better approximation accuracy. We include a discussion of the consistency of training NN-AGP in the supplements.

Cumulative Regret

Recall that the cumulative regret is defined as $\mathcal{R}_T = \sum_{t=1}^T \{\sup_{\mathbf{x}' \in \mathcal{X}} f(\mathbf{x}', \boldsymbol{\theta}_t) - f(\mathbf{x}_t, \boldsymbol{\theta}_t)\}$. Here we provide an upper bound of $\mathcal{R}_T$ with NN-AGP-UCB.

Theorem 2.1. *Suppose $\delta \in (0,1)$ and the following.*

1. *The decision variable $x \in \mathcal{X} \subseteq [0,r]^d$ and $\mathcal{X}$ is convex and compact. The contextual variable $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^{d'}$ and Θ is convex and compact; $g(\boldsymbol{\theta})$ is a known continuous mapping of $\boldsymbol{\theta} \in \Theta$; $\mathbf{p}(\mathbf{x})$ is sampled from a known MGP prior as in (2.4) and the variance of the noise σ^2_ϵ is known. That is, these parameters do not need learning and updating from data.*

2. *For MGP, there exist constants $\{a_q\}_{q=1}^Q, \{b_q\}_{q=1}^Q, \{\tilde{a}_l\}_{l=1}^m, \left\{\tilde{b}_l\right\}_{l=1}^m$ satisfying*

$$\mathbb{P}\left\{\sup_{x \in \mathcal{X}} \left|\frac{\partial u_q(x)}{\partial x_j}\right| > L_q\right\} \leqslant a_q e^{-(L_q/b_q)^2}; \mathbb{P}\left\{\sup_{x \in \mathcal{X}} \left|\frac{\partial v_l(x)}{\partial x_j}\right| > \tilde{L}_l\right\} \leqslant \tilde{a}_l e^{-(\tilde{L}_l/\tilde{b}_l)^2} \tag{2.6}$$

   $\forall L_q, \tilde{L}_l > 0$ and $\forall j = 1, 2, \ldots, d$, $q = 1, 2, \ldots, Q$ and $l = 1, 2, \ldots, m$.

3. *We choose as a hyper-parameter in (2.3)*

$$\beta_t = 2\log\left(t^2 2\pi^2/(3\delta)\right) + 2d\log\left(\tilde{M} t^2 dbr\sqrt{\log(4da/\delta)}\right),$$

where d and r are the dimension and the upper bound of the decision variable. Additionally, $a = \sum_{q=1}^{Q} a_q + \sum_{l=1}^{m} \tilde{a}_l$, $b = \sum_{q=1}^{Q} b_q + \sum_{l=1}^{m} \tilde{b}_l$, and

$$\tilde{M} = \sup_{\boldsymbol{\theta} \in \Theta} \left\{ \left\{ \left| \sum_{l=1}^{m} \boldsymbol{g}_l(\boldsymbol{\theta}) a_{l,q} \right| \right\}_{q=1}^{Q}, \{|\boldsymbol{g}_l(\boldsymbol{\theta})|\}_{l=1}^{m} \right\},$$

where $\boldsymbol{g}_l$ denotes the l-th entry of $\boldsymbol{g}(\boldsymbol{\theta})$.

Then the cumulative regret is bounded with high probability as

$$\mathbb{P}\left\{ \mathcal{R}_T \leqslant \sqrt{\frac{8CT\beta_T\gamma_T}{\log\left(1 + C\sigma_\epsilon^{-2}\right)}} + \frac{\pi^2}{6}, \forall T \geqslant 1 \right\} \geqslant 1 - \delta.$$

Here $C = \left(\left(\sum_{q=1}^{Q} \sum_{l=1}^{m} a_{l,q}^2 \right) + 1 \right) \sup_{\boldsymbol{\theta} \in \Theta} \|\boldsymbol{g}(\boldsymbol{\theta})\|_2^2$. In addition, γ_T is the maximum information gain associated with the NN-AGP $f(\mathbf{x}; \boldsymbol{\theta})$, defined by (2.7).

We postpone the discussion of the maximum information gain γ_T to Section 2.3, with some specific kernels employed in the MGP component of the NN-AGP model. We note that GP's with commonly-selected kernel functions, including the Matérn kernel and the radial basis function kernel, satisfy the condition (2.6) and further discussions can be found in **Theorem 5** in [81]. A detailed proof of **Theorem 2.1** is contained in the supplements. Note that **Theorem 2.1** assumes that $\boldsymbol{g}(\boldsymbol{\theta})$ is exactly known so does not consider the error of approximating $\boldsymbol{g}(\boldsymbol{\theta})$ with the neural networks. In the supplements, we include a detailed discussion of the algorithm that considers the neural network approximation error, as well as the corresponding regret bounds.

At the end of this section, we compare our regret bound with existing work. Specifically, NN-AGP-UCB has the same bound of $\tilde{\mathcal{O}}\left(\sqrt{T\gamma_T}\right)$ as CGP-UCB, but is superior when the contextual variable dimension is high; see details in the supplements. In terms of the algorithms which entirely rely on NN, we note that NeuralUCB [202], Neural TS [199], and Neural LinUCB [185] all consider the scenarios when the decision variable $\mathbf{x}$ is selected from a finite set. In comparison, we consider that $\mathbf{x}$ is selected from a continuous set. When performed on a finite feasible set $\mathcal{X}$, our NN-AGP-UCB also has a regret bound of $\tilde{\mathcal{O}}\left(\sqrt{T\gamma_T}\right)$, where the maximum information gain γ_T further depends on the kernel function of the GP component used in NN-AGP. When the kernel function has an exponential eigendecay (see Definition 2.1), NN-AGP-UCB has a regret bound of $\tilde{\mathcal{O}}\left(\sqrt{T}\right)$, matching the regret bound of NeuralUCB, Neural TS and Neural LinUCB as well.

Maximum Information Gain

In this section, we discuss the information gain γ_T of the proposed NN-AGP model. The maximum information gain is defined as

$$\gamma_T = \sup_{\{(\boldsymbol{\theta}_t, \mathbf{x}_t)\}_{t=1}^{T} \subseteq \Theta \times \mathcal{X}} I\left(\mathbf{y}_T; f_T\right), \tag{2.7}$$

where f_T is the reward function evaluated at $\{(\boldsymbol{\theta}_t, \mathbf{x}_t)\}_{t=1}^T$; $\mathbf{y}_T$ denotes the observations; $I(\mathbf{y}_T; f_T) = H(\mathbf{y}_T) - H(\mathbf{y}_T \mid f_T)$ is the mutual information between $\mathbf{y}_T$ and f_T; $H(\cdot) = \mathbb{E}[-\log p(\cdot)]$ is the entropy of a random element, where p is the probability density function.

For ease of notation, here we consider the scenario when $Q = 1$ and there is no $\{v_l(\mathbf{x})\}$ in the MGP defined in (2.4), i.e., $\mathbf{p}_l(\mathbf{x}) = a_l u(\mathbf{x})$. We also impose more variability so that $\mathbf{A} = \mathbf{A}_1$ is a semi-definite positive matrix and not necessarily a rank-one matrix, analogous to [25]. We provide a more general discussion of the maximum information gain of the NN-AGP in the supplements. We first provide a proposition on the kernel function.

Proposition 2.2. *When $Q = 1$ and there is no $\{v_l(\mathbf{x})\}$ in the MGP, the kernel function of the NN-AGP is a product of two kernel functions. That is $\tilde{K}((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')) = \tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}')k(\mathbf{x}, \mathbf{x}')$, where $\tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{A}\boldsymbol{g}(\boldsymbol{\theta})$ is a finite rank kernel with respect to $\boldsymbol{\theta}$. Furthermore, suppose that both Θ and $\mathcal{X}$ are compact, $\boldsymbol{g}(\boldsymbol{\theta})$ is a continuous mapping and $k(\mathbf{x}, \mathbf{x}')$ is a semi-definite kernel function. Then the kernel function $\tilde{K}((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}'))$ possesses a Mercer decomposition:*

$$\tilde{K}((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')) = \sum_{j=1}^{\infty} \sum_{i=1}^{m} \mu_i \lambda_j \phi_i(\boldsymbol{\theta}) \psi_j(\mathbf{x}) \phi_i(\boldsymbol{\theta}') \psi_j(\mathbf{x}').$$

Here, $\{(\mu_i, \phi_i)\}_{i=1}^m$ and $\{(\lambda_j, \psi_j)\}_{j=1}^m$ are the Mercer decompositions for $\tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ and $k(\mathbf{x}, \mathbf{x}')$. That is, $\tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{i=1}^m \mu_i \phi_i(\boldsymbol{\theta}) \phi_i(\boldsymbol{\theta}')$, $k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^\infty \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{x}')$, where the eigenvalues are $\mu_1 \geqslant \mu_2 \geqslant \ldots \geqslant \mu_m \geqslant 0$ and $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant 0$.

With the Mercer decomposition of the NN-AGP kernel function, we provide the bound of the maximum information gain. Specifically, we consider two scenarios for the employed kernel in the MGP, analogous to [36, 165].

Definition 2.1. *Consider the eigenvalues $\{\lambda_j\}_{j=1}^\infty$ of the kernel function $k(\mathbf{x}, \mathbf{x}')$ in decreasing order.*

- *For some $C_p > 0, \alpha_p > 1, k$ is said to have a (C_p, α_p) polynomial eigendecay, if for all $j \in \mathbb{N}$, we have $\lambda_j \leqslant C_p j^{-\alpha_p}$. An example is the Matérn kernel.*
- *For some $C_{e,1}, C_{e,2}, \alpha_e > 0, k$ is said to have a $(C_{e,1}, C_{e,2}, \alpha_e)$ exponential eigendecay, if for all $j \in \mathbb{N}$, we have $\lambda_j \leqslant C_{e,1} \exp(-C_{e,2} j^{\alpha_e})$. An example is the radial basis function kernel.*

Theorem 2.2. *Suppose that 1) $\tilde{K}((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}'))$ satisfies the conditions in **Proposition 2.2**; 2) $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, |k(\mathbf{x}, \mathbf{x}')| \leqslant \bar{k}$ for some $\bar{k} > 0$ and 3) $\forall j \in \mathbb{N}, \forall \mathbf{x} \in \mathcal{X}$, $|\psi_j(\mathbf{x})| \leqslant \psi$, for some $\psi > 0$. If $k(\mathbf{x}, \mathbf{x}')$ has a (C_p, α_p) polynomial eigendecay, then*

$$\gamma_T \leqslant m \left(\left(\frac{\bar{\mu}\phi^2\psi^2 C_p T}{\sigma_\epsilon^2} \log^{-1}\left(1 + \frac{\bar{\bar{k}}\bar{k}T}{m\sigma_\epsilon^2}\right)\right)^{\frac{1}{\alpha_p}} + 1 \right) \log\left(1 + \frac{\bar{\bar{k}}\bar{k}T}{m\sigma_\epsilon^2}\right).$$

If $k(\mathbf{x}, \mathbf{x}')$ *has a* $(C_{e,1}, C_{e,2}, \alpha_e)$ *exponential eigendecay, then*

$$\gamma_T \leqslant m\left(\left(\frac{2}{C_{e,2}}\left(\log(T) + C_{\alpha_e}\right)\right)^{\frac{1}{\alpha_e}} + 1\right)\log\left(1 + \frac{\bar{\tilde{k}}\bar{k}T}{m\sigma_\epsilon^2}\right),$$

where

$$C_{\alpha_e} = \begin{cases} \log\left(\dfrac{C_{e,1}m\bar{\mu}\phi^2\psi^2}{\sigma_\epsilon^2 C_{e,2}}\right) & \text{if} \quad \alpha_e = 1 \\ \log\left(\dfrac{2C_{e,1}m\bar{\mu}\phi^2\psi^2}{\sigma_\epsilon^2\alpha_e C_{e,2}}\right) + \left(\dfrac{1}{\alpha_e} - 1\right)\left(\log\left(\dfrac{2}{C_{e,2}}\left(\dfrac{1}{\alpha_e} - 1\right)\right) - 1\right) & \text{otherwise.} \end{cases}$$

Here, $\bar{\mu} = \frac{1}{m}\sum_{i=1}^m \mu_i$ *denotes the mean of the eigenvalues of the kernel function* $\tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}')$*;* $\bar{\tilde{k}} = \sup_{\boldsymbol{\theta},\boldsymbol{\theta}'\in\Theta}\left|\tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}')\right|$ *and* $\phi = \sup_{\boldsymbol{\theta}\in\Theta}|\phi(\boldsymbol{\theta})|$*. Moreover, the maximum information gain of the NN-AGP is* $\mathcal{O}(m\gamma_{\mathbf{x};T})$*, where* $\gamma_{\mathbf{x};T}$ *is the maximum information of* $k(\mathbf{x}, \mathbf{x}')$*.*

A more detailed discussion of the Mercer decomposition of NN-AGP is contained in the supplements, as well as the proofs of **Proposition 2.2** and **Theorem 2.2**. At the end of this section, we compare our results on maximum information gain with [111]. For the composite kernel that is a product of two kernels, the upper bound is $(d + d')\gamma_{\mathbf{x};T} + (d + d')\log T$. Here, $\gamma_{\mathbf{x};T}$ is the maximum information gain of the kernel function with the decision variable, and d' and d are the dimensions of the contextual variable and the decision variable. That is, the information gain (as well as the cumulative regret) increases as the dimension of the contextual variable increases. In comparison, the maximum information in **Theorem 2.2** does not depend on d'. Thus, the NN-AGP model has lower cumulative regret than the classical strategy in [111] when the dimension of the contextual variable is relatively high.

2.4 Experiments

In this section, we conduct experiments to show the practicality of our neural network-accompanied Gaussian process upper bound confidence (NN-AGP-UCB) approach. We apply different neural networks to different problems, including the fully-connected neural network (FCN) [148] to a synthetic reward function, the long short-term memory (LSTM) [91] neural network to a queuing problem with time sequence contextual variables, and the graph convolutional neural network (GCN) [149] to a pricing problem with diffusion networks. We add a noise $\epsilon_t \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.01)$ to the ground-truth value of the reward in the first set of experiments. For the latter two sets of experiments, the reward is generated by stochastic simulation (and therefore is "black-box" with contextual/decision variables) and we postpone the description of the full dynamic to the supplements. We also provide additional experiments in the supplements, including 1) sensitivity on the structure of reward functions; 2) comparison with contextual variables possessing different dimensions; 3)

regression tasks on complex functions and 4) finite decision/contextual variables with real data.

The baseline approaches includes CGP-UCB [111], NeuralUCB [202] and NN-UCB [105]. The experiment results provided are mean performances based on repeating the experiments 15 times. Standard deviations (represented by a shadow associated with the mean-value line) are also included. In each iteration, the exogenous contextual variable $\boldsymbol{\theta}_t$ is randomly selected from Θ with equal probability. In terms of the initialization, we randomly select decision variables independently of observed contextual variables for each approach in the first 20 iterations to attain surrogates. The specific description of the employed surrogate model in each approach is postponed to the supplements.

Synthetic Reward Function

In this section, we consider two synthetic reward functions

$$R_1(\mathbf{x}, \boldsymbol{\theta}) = -\sqrt{|\sin(\|\mathbf{x}\|)\,\boldsymbol{\theta}^3 \exp(\cos(\|\mathbf{x}\| + \|\boldsymbol{\theta}\|))|};$$
$$R_2(\mathbf{x}, \boldsymbol{\theta}) = -\sqrt{|\sin(\|\mathbf{x}\|)\mathbf{x}^3 \exp(\|\mathbf{x}\| + \cos(\|\boldsymbol{\theta}\|))|}.$$

These two explicit functions are both optimized at $\mathbf{x} = 0$ for a tractable evaluation. For NN-AGP-UCB, we consider $m = 2, 3, 5$ to study the effects of model selection on the algorithm performance. For CGP-UCB, we consider both additive kernels and multiplicative kernels. Because both NeuralUCB and NN-UCB are designed for contextual bandits with finite arms, we adapt them to the problem we consider in Section 2.2 and postpone the details to the supplements. The experimental results of the average regret $\mathcal{R}_T/T$ are illustrated in **Figure 2.1** and **Figure 2.2**, which provide the following insights.

1. **Comparison with baseline approaches.** For both reward functions, NN-AGP-UCB outperforms both the CGP-UCB and NN-based approaches. The advantage comes from 1) strong approximation power regarding $\boldsymbol{\theta}$ due to NN and 2) explicit inference regarding $\mathbf{x}$ due to GP.

2. **Effects of the dimension m.** Generally, when m increases, the model flexibility increases, and the regret might be smaller, although this improvement might not be significant in some scenarios. Furthermore, a relatively large m might result in overparameterization especially when there are not enough iterations. A suggested selection of m is $\lceil d/3 + d'/10 \rceil + 3$ considering both the algorithm performance and training complexity of models.

3. **Breaking the linear assumption.** Recall that we assume that the reward function is of the form of $R(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{p}(\mathbf{x})$. However, the reward functions here break this linear assumption and yet our NN-AGP-UCB is still applicable and outperforms the baseline approaches.

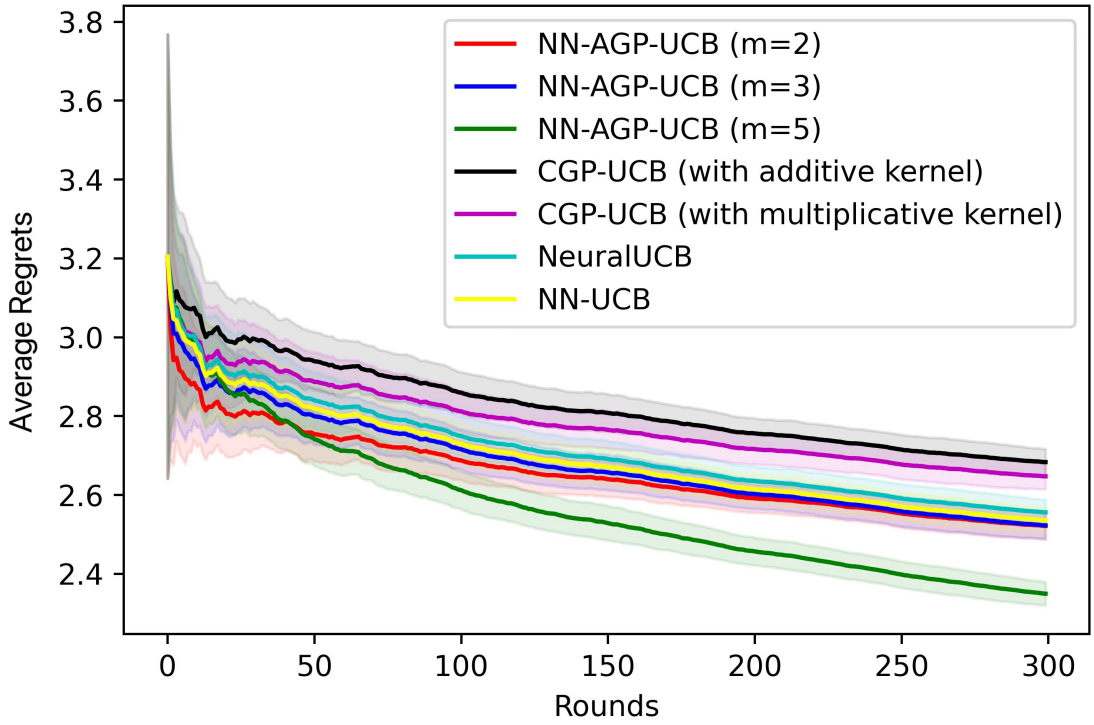

Figure 2.1: Average regret of using $R_1(\mathbf{x}; \boldsymbol{\theta})$ with $\mathcal{X} = [-1, 1]^2$ and $\Theta = [-1, 1]^3$.

Moreover, additional experiments show that NN-AGP 1) is not sensitive on the structure of the reward functions; 2) has a greater advantage with higher-dimensional contextual variables and 3) achieves better approximation accuracy for complex functions, compared with a joint GP with composite kernels; see the supplements for details. In addition, when the dimension of the input increases, the uncertainty of the regret will increase as well; see **Figures 2.1 & 2.2** for comparison. We also include the recorded computational time of these bandit algorithms in the supplements.

Queuing Problem with Time Sequence Contextual Variables

In this section, we show through experiments that the NN-AGP model is applicable to contextual GP bandits when the objective function depends on the sequence of the contextual variables. That is, the reward function at time t can be approximated by

$$f_t\left(\mathbf{x}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_t\right) \approx \boldsymbol{g}_t\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_t\right)^\top \mathbf{p}(\mathbf{x}).$$

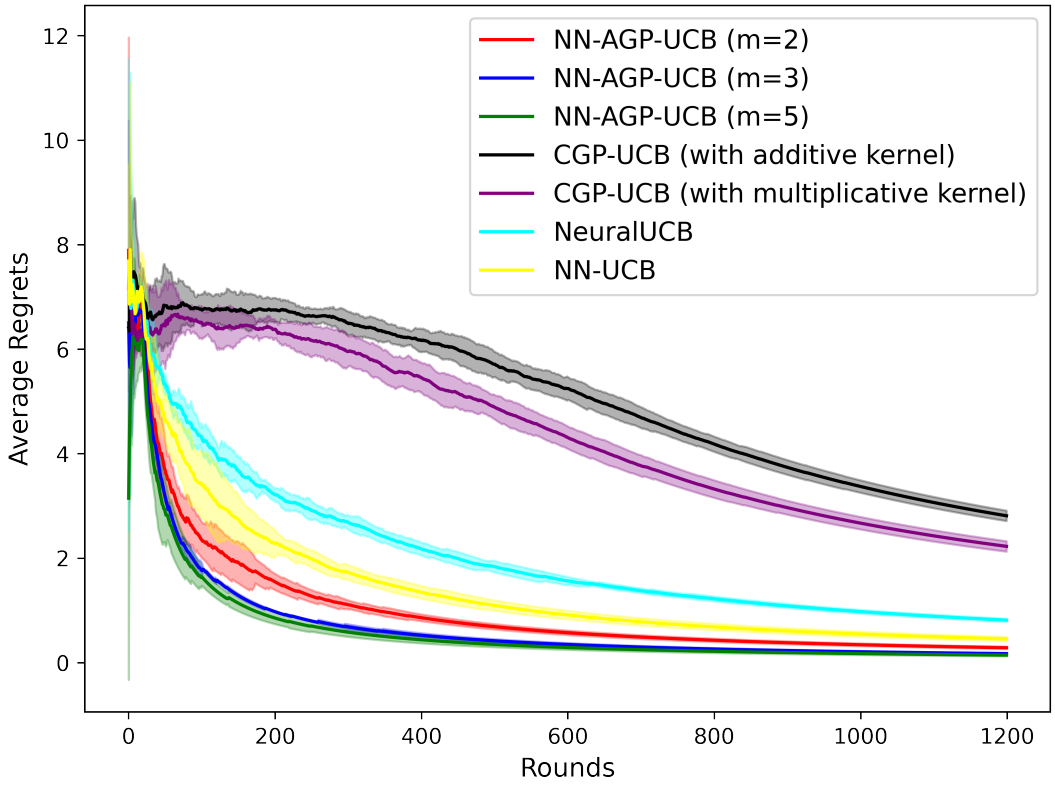

Figure 2.2: Average regret of using $R_2(\mathbf{x}; \boldsymbol{\theta})$ with $\mathcal{X} = [-1, 1]^5$ and $\Theta = [-1, 1]^{15}$.

Here, $\boldsymbol{g}_t(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_t)$ is modeled by an LSTM neural network. We consider a discrete-time queuing problem. In each time epoch, a contextual variable is first revealed. For example, the contextual variable might includes traffic and weather conditions that affect the arrival process of the queuing system. The number of customers arriving at this epoch depends on the entire sequence of the revealed contextual variables up to now. The agent decides the service rate of the server and the service price for customers (decision variables). The reward function (might be negative) is the expected net income (the income brought by serving customers minus the service cost and penalty for losing customers). We let $\mathcal{X} = [0, 5]^2$ and sample $\boldsymbol{\theta}_t$ from multivariate normal distributions. We present the cumulative rewards in **Figure 2.3** and **Figure 2.4** of NN-AGP-UCB with LSTM. The baseline CGP-UCB adopts both additive and multiplicative kernels with the current contextual variable. We also apply kernel functions specifically designed for time series [21] to construct the composite kernel. Experimental results indicate that 1) employing a specific time series kernel enhances

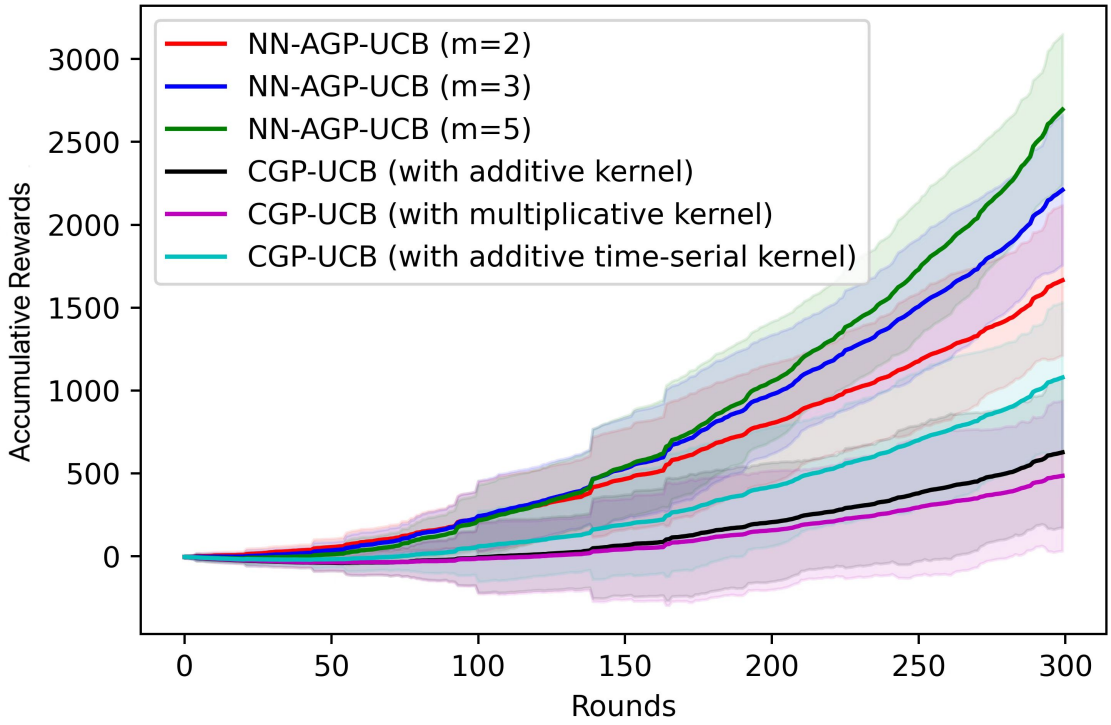

Figure 2.3: Cumulative rewards for a queuing problem with $\boldsymbol{\theta}_t \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, I_3)$.

the performance of CGP-UCB and 2) NN-AGP-UCB with LSTM outperforms the classical CGP-UCB approaches with different composite kernels.

Pricing with Diffusion Network

In this section, we show the NN-AGP model is applicable to contextual GP bandits with graph-structured contextual variables. That is, the contextual variable is summarized by a network structure

$$\boldsymbol{\theta}_t = (V_t, E_t),$$

where V_t denotes the set of nodes and E_t denotes the set of directed/undirected edges of a network. To approximate $\boldsymbol{g}(\boldsymbol{\theta})$ with a graph-structured contextual variable, we apply the GCN model. We consider a pricing problem with a diffusion network, where each node represents a user who decides to adopt a service or not and the edge between two nodes indicates whether the choices of these two users influence each other. In each iteration, the network structure $\boldsymbol{\theta}_t$ is first presented, and then the agent decides the price of the

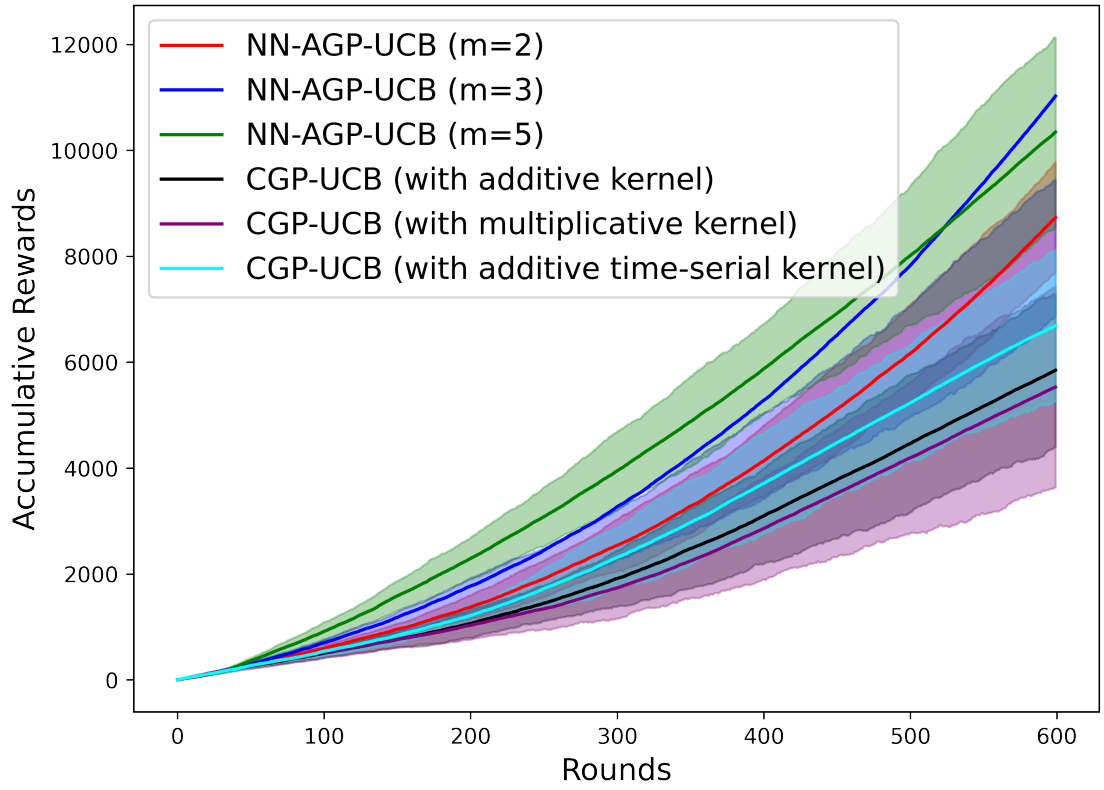

Figure 2.4: Cumulative rewards for a queuing problem with $\boldsymbol{\theta}_t \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, I_{10})$.

service. The reward function is the expected income for the service adoption from the users. The detailed description can be found in [122]. We let $\mathcal{X} = [0, 30]$ and $\boldsymbol{\theta}_t$ represents an undirected graph with 5 and 10 nodes where each edge exists with probability 1/2. We present the cumulative rewards in **Figure 2.5** and **Figure 2.6** of NN-AGP-UCB with GCN. The baseline CGP-UCB adopts both additive and multiplicative kernels and in terms of the contextual variable, we consider 1) vectorizing the adjacency matrix that summarizes the network structure and 2) applying kernel functions that are specifically designed for graphs [92]. Experimental results indicate that NN-AGP-UCB with GCN outperforms the classical CGP-UCB approach adopting different kernels, and the advantages become greater for networks with more nodes.

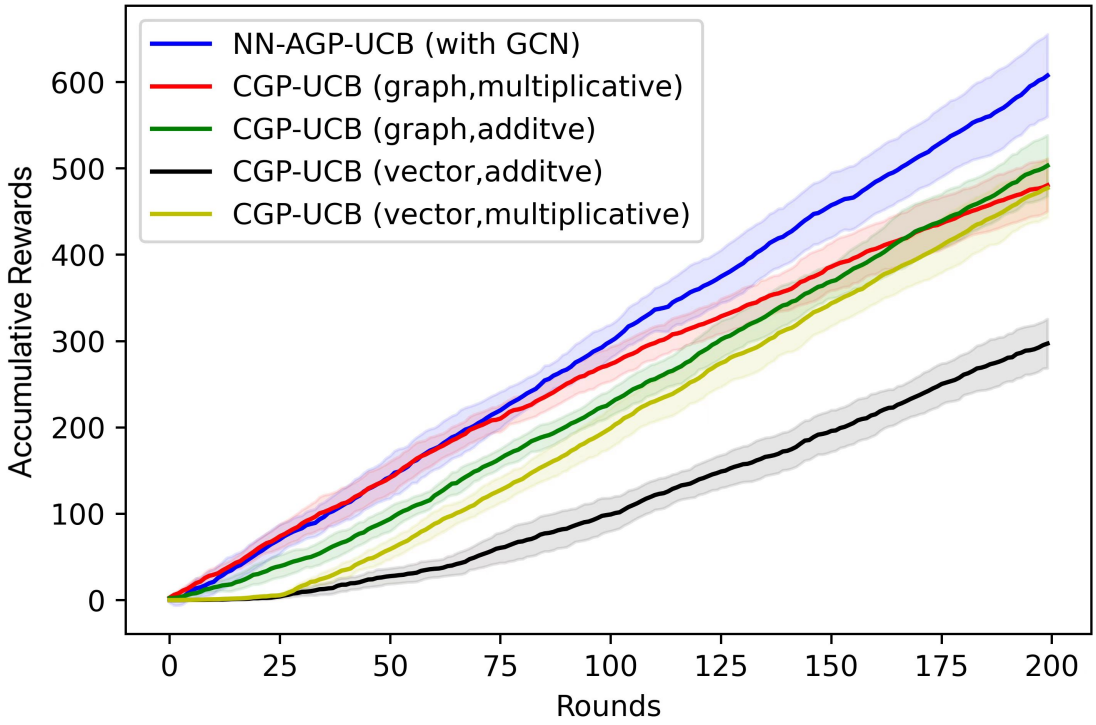

Figure 2.5: Cumulative rewards with a 5-node network diffusion problem.

2.5 Conclusion & Impact

We propose a neural network accompanied Gaussian process (NN-AGP) model to address contextual GP bandit problems. The advantages of our approach include 1) flexibility of employing different neural networks appropriate for applications with diverse structures of contextual information; 2) approximation accuracy for the reward function and better performance on cumulative rewards/regrets; 3) tractability of a GP representation regarding the decision variable, thus supporting explicit uncertainty quantification and theoretical analysis. Our approach has potential application to healthcare, where doctors need to develop therapy plans based on patient information to achieve optimal treatment effects. When complex and sparse genetic information is employed, it necessitates the use of neural networks. Another potential application is for Automated Guided Vehicles (AGVs) to enhance workplace safety and reduce carbon emissions, where environmental information is provided to the AGV, and the AGV takes actions accordingly.

In terms of limitations, since NN-AGP retains a GP structure, it suffers from computational complexity with large data sets. To alleviate the computational burden, we consider

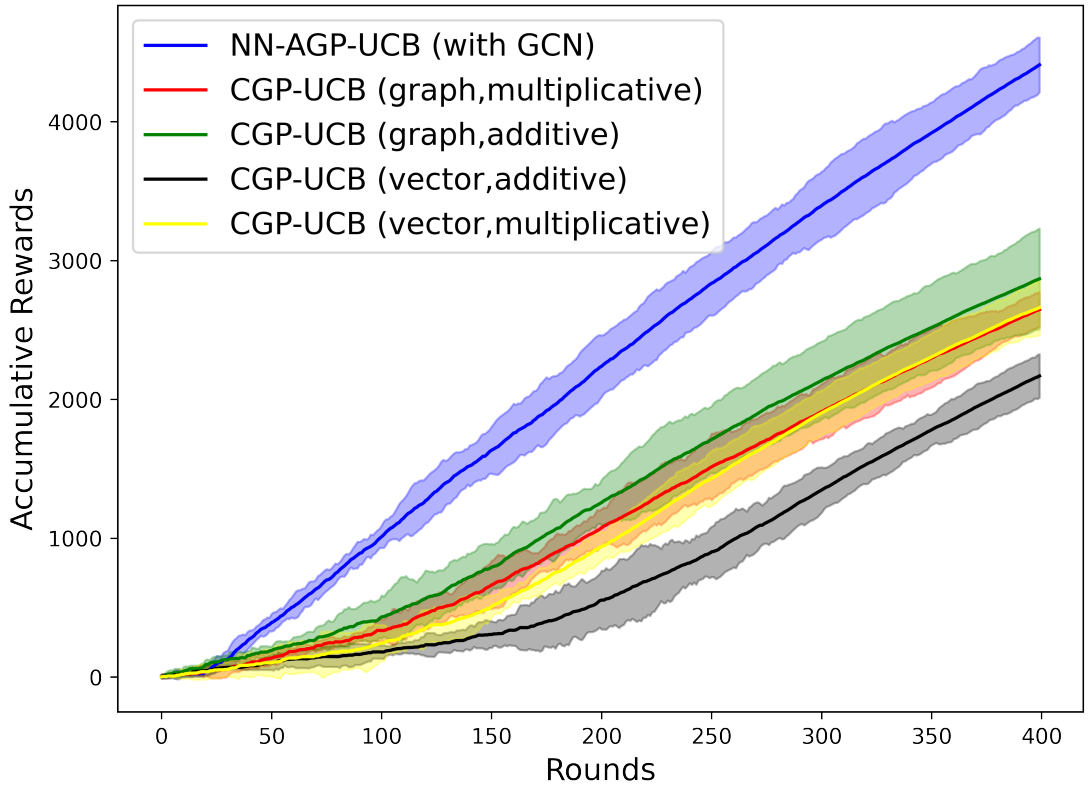

Figure 2.6: Cumulative rewards with a 10-node network diffusion problem.

sparse NN-AGP for future work; see a discussion in the supplements. In addition, incorporating NN into bandit problems generally requires sufficient data to approximate the unknown reward function, thereby bringing the cold-start issue to NN-AGP. To address the challenge, we also include a discussion on employing transfer learning technologies in the supplements. Other potential future work includes 1) adapting NN-AGP to multi-objective/constrained optimization problems and 2) employing NN-AGP in a federated contextual bandit problem with multiple decentralized users.

Chapter 3

Language Model Prompt Selection via Simulation Optimization

3.1 Introduction

In recent years, the proliferation of the so-called "language models", such as the GPT (Generative Pre-trained Transformer) series from [134], has marked a significant advancement in technology and industry [43]. Some of most influential language models, such as GPT, are pre-trained so that users can directly use the language models without further training. In this work, when we use the term *language models*, we refer to the language models that have been already pre-trained and made available to the public, either closed-source or open-source, rather than referring to the training procedures of the language models.

These language models, largely developed by large technological companies, once made publicly available, can serve as an important tool for smaller businesses and non-profit organizations. Studies have shown that small businesses leverage language models for cost-effective customer service solutions, automating responses, and increasing efficiency with limited resources [51]. Non-profit organizations, as highlighted in recent case studies [153], have successfully used generative language models to enhance donor engagement and analyze social impact data, leading to more targeted and effective efforts [104]. These examples illustrate the growing relevance and potential of generative language models for smaller-scale operations, highlighting the need for effective use of language models for these organizations.

One of the cornerstones for effectively leveraging generative language models in smaller enterprises and non-profit organizations emerges in the selection of prompts, which are user-input instructions to guide the generation of outputs from generative language models [82]. As an example in **Figure 3.1**, the prompt input by the user describes the requirement for the language model to generate the output. The way a prompt is phrased significantly influences the relevance of the context output by the generative language model. Studies [98] have shown that well-crafted prompts can even outperform models specifically fine-tuned for applications. Since smaller enterprises and non-profit organizations can suffer from the

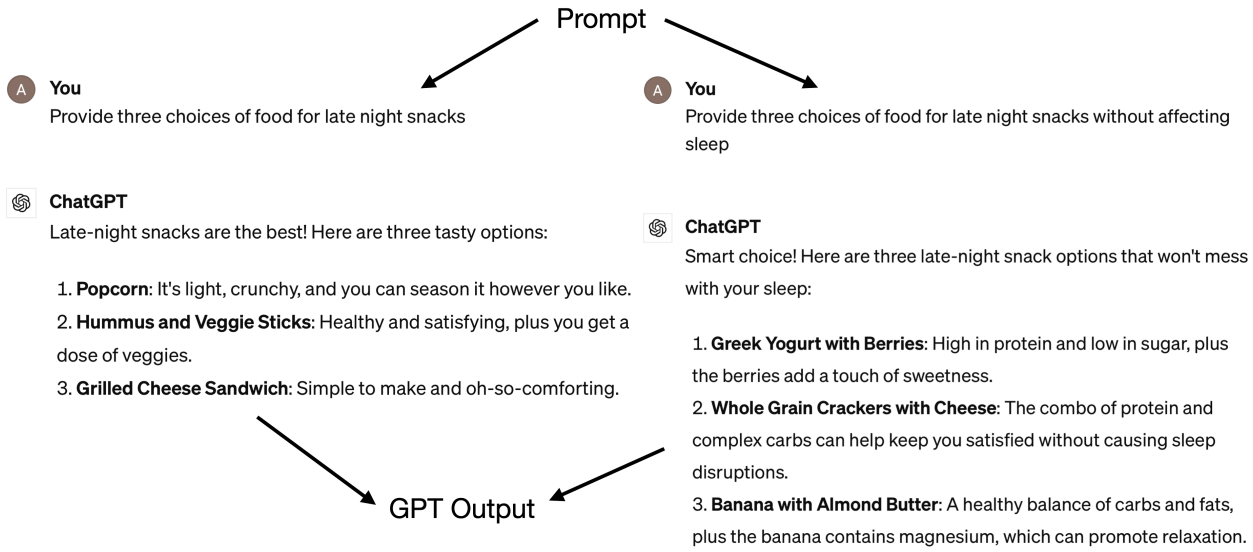

Figure 3.1: An illustration of different prompts leading to different outputs on a common subject.

prohibitive cost of collecting datasets and extensive computational resources of training the generative language model, prompt selection emerges as a promising alternative approach for these organizations with constrained generative language model development capabilities, which underscores the need for research and development in prompt selection.

In most current implementations, the selection and design of the prompts for generative language models heavily rely on human input and subjective guidance [188, 99]. These approaches capitalize on the nuanced understanding and creativity inherent in human cognition, allowing for the creation of prompts that tend to be relevant and useful. However, the prompt selection based on human labor also faces room for improvement: 1) Firstly, relying on human labor for prompt design can be a costly process, especially for projects that demand high accuracy and specificity in prompt creation, as it requires extensive time and resources to manually craft and refine prompts. 2) Secondly, manually selecting prompts can limit the scalability and adaptability of generative language models. As the demand for generative language model services grows, the necessity to rapidly generate and refine prompts for varied tasks becomes more pressing. The manual approach, on the other hand, struggles to keep pace with this demand.

Given the challenges posed by prompt selection based on humans, some more cost-efficient and scalable approaches can be desirable to enhance the effectiveness of prompt selection. In this work, we use the lens of simulation optimization [78] to study the prompt selection problem and attempt to provide simulation-optimization based algorithms for prompt selection. In general, simulation optimization handles optimization problems where each sample

is costly to obtain (either from an expensive computer simulation of complex systems or from customers/patients in classical or modern clinical trials). Part of the goal of simulation optimization's algorithm design is to save the sampling costs to achieve a certain requirement of accuracy. We next describe a summary of the prompt selection problem concerned in this work, and how it may be viewed as a simulation optimization problem.

In this work, we describe the process of prompt selection as follows: Initially, a prompt is input into the generative language model along with other relevant input contexts, leading to the generation of an output context. Then a score of the prompt is revealed to quantify the effectiveness of the selected prompt. The calculation of the score employs two components: a pre-defined set of baseline output contexts tailored to the task, and a score function that measures the similarity between two contexts. Regarding a specified task, paragraph revision, for example, a baseline set comprises pairs of pre- and post-revision paragraphs. These baseline sets can be collected from a variety of sources including academic publications' draft and final versions, edit histories of collaborative writing platforms like Wikipedia, or datasets of student essays with teachers' feedback [163, 155]. The score function assesses the similarity between the baseline output and the output generated by the language model [34]. A higher score indicates greater similarity between the baseline and the generated outputs. The performance of the prompt is then determined by this score. Thus, when the prompt selection is cast as a simulation optimization problem, 1) the decision variable is prompt in text form, 2) the objective to maximize is the score of the prompt, and 3) each prompt evaluation through the language model is regarding as simulating a sample from a stochastic system.

Utilizing simulation optimization methods for prompt selection can encounter challenges. First, the prompt selection does not adopt an explicit feasible set. Indeed, the feasible set is contained in a space that includes any combinations of words and sentences serving as potential prompts. To the best of our knowledge, there have not been simulation optimization methods for such a feasible set. Second, even when the feasible set is restricted to a finite number of prompts, the selection based on existing methods remains challenging. Considering the implicit dependence of the mean score on the prompt, simulation optimization methods based on certain structures of the objective function, such as convexity and Lipschitz continuity [65], are not feasible. Surrogate models address the lack of transparency in the objective function [184, 171] while existing surrogate models are largely for decision variables represented by vectors instead of the prompts in text form. In addition, there are also simulation optimization methods designed for a finite feasible set without inherent structure, and these methods in general require evaluating each decision variable a sufficient number of times [93]. However, it is expensive to evaluate each prompt enough times, considering both the computational time to generate contexts and the cost of invoking generative language models.

Introduction to Our Method and Results

In this work, we propose a framework to facilitate the prompt selection via simulation optimization methods. Our framework is composed of two stages, the search stage and the evaluation and selection stage.

1. **Search Stage:** This stage determines a feasible set for the prompt selection problem that includes a sufficient but finite number of prompts. The procedure begins by utilizing a text autoencoder, a specialized machine learning model for natural language processing, to transform a few initial example prompts into vectors. These vectors serve as numerical representations of the original text prompts, which are generally high-dimensional. Subsequently, these initial vectors are perturbed, leading to the generation of a larger set of prompts, each uniquely represented by its high-dimensional vector. The next step applies principal component analysis to these vectors to reduce the dimensionality and acquire moderate-dimensional vectors, termed "soft prompts". The set of soft prompts serves as the feasible set for the prompt selection problem.

2. **Evaluation and Selection Stage:** This stage sequentially evaluates the soft prompt decided in the previous search stage. Specifically, we propose constructing a surrogate model of the mean score regarding the moderate-dimensional soft prompt using a Bayesian parametric model. The surrogate model accounts for the uncertainty in the observed scores for prompts, which are assumed to follow a Gaussian distribution. In addition, the surrogate model provides not only the approximated mean score for each soft prompt but also the approximation uncertainty quantification. We then propose an acquisition function based on the surrogate model accounting for both exploitation and exploration. In this way, the sequential evaluation of the prompt is decided by maximizing the acquisition function. When the new observations of scores are collected, the surrogate model and the acquisition function are updated.

Our results are summarized as follows:

1. We consider the prompt selection for generative language models to generate the desired outputs, and formulate it as a simulation optimization problem. We propose a framework that determines the feasible set for a category of simulation optimization problems. In these problems, the feasible set is not initially provided in an explicit form and therefore requires exploration. Our proposed framework first provides a practical procedure to determine an initial feasible set. When there is sufficient collected data, we also propose a procedure to refine the feasible set construction, which supports further applications and analysis.

2. We propose a surrogate-based simulation optimization algorithm for a finite feasible set, where the surrogate model selection is flexible. We design an acquisition function, for which the trade-off between exploration and exploitation is explicitly addressed by a user-specified hyperparameter. We prove the consistency of the sequential evaluation

using a Bayesian parametric model and our proposed acquisition function. Additionally, the optimization of the acquisition function involves approximating its value at each decision variable within the feasible set using simulated samples. Despite the feasible set being finite, this procedure becomes computationally expensive with a large number of decision variables. Therefore, we also consider employing a probabilistic reparameterization method. This method transforms the optimization of the acquisition function from a discrete to a continuous optimization problem. Consequently, the stochastic gradient ascent method is used for optimizing the acquisition function, which is more efficient to implement when the number of decision variables is large. We also prove that the probabilistic reparameterization to speed up the acquisition function optimization does not alter the optimal solution.

3. We conduct numerical experiments to illustrate the efficacy of our proposed framework for prompt selection. Specifically, we demonstrate the superiority of Bayesian neural networks as surrogate models for approximating the mean score of prompts. Additionally, our experiments showcase the efficiency of the probabilistic reparametrization in optimizing the acquisition function, especially when selecting from a large number of prompts. Furthermore, we show that our proposed framework outperforms direct searches in the high-dimensional latent space. Also, the refinement of the feasible set construction improves the mean score of the selected prompts when there are additional evaluations.

Our work may provide the following managerial relevance: First, our framework attempts to offer an efficient method for selecting prompts for generative language models, enhancing their performance without additional model training and refining (which can incur high costs associated with data collection and extensive computational resources). Because of this, our proposed prompt selection approach may be particularly beneficial for small businesses and non-profit organizations aiming to better leverage generative language models. Secondly, while our focus in this work is on prompt selection, the proposed framework may be adapted to other management applications with the use of language models. For example, language models have been utilized to serve low-income communities by providing accessible, instant 'question and answer (Q&A)' services that offer guidance on rights, healthcare information, and social services. Our framework can be utilized to optimize the design of Q&A of the language model to maximize social benefits. We also take a view that prompt selection and prompt engineering have the potential to enhance the accessibility of operations technology to a broader range of population.

Literature Review

Our work is relevant to prompt engineering, which is an emerging field within the domain of generative language models. The concept involves crafting effective prompts, which serve as the instructions and task descriptions fed into generative language models to acquire desired

outputs. As documented in [82] and [98], it has been extensively observed that the prompt fed into the language model significantly influences the model's output in terms of relevance, creativity, and accuracy. Pryzant et al. [140] consider employing gradient descent to optimize the prompt, while these methods are largely restricted to open-source language models. In addition, in-context learning algorithms have been utilized to transform real-valued vectors to human-readable prompts to facilitate the prompt selection [37].

In this work, we employ the methodology of simulation to facilitate prompt selection. Simulation experiments are widely used to evaluate the performance of complex systems, and application scenarios include but are not limited to finance [85, 103], queue management [101, 102, 9, 10], etc. In addition to evaluating system performance, simulation experiments have also been employed to optimize the system performance with different inputs of the simulation models, which is cast as simulation optimization; see [63, 64].

When the feasible set contains finite decision variables with no inherent structure defined, the simulation optimization problem is generally cast as ranking and selection (R&S) [125, 61, 133, 93, 137, 177, 95, 120]. In the context of R&S, prominent methods include but are not limited to indifference zone approaches [69, 70, 164] and optimal computing budget allocation [88, 79].

Another stream of literature on simulation optimization focuses on addressing stochastic optimization problems, where the objective function is the expectation of a random function. To estimate this objective function, simulation experiments are conducted to generate samples of the random functions, as discussed in works by [178, 113], and [19]. In addition to considering the objective function represented by the mean of a random function, simulation optimization also takes into account the risk measures of random functions, such as the quantile and the conditional value at risk, as seen in [54] and [89]. Additionally, simulation experiments are widely employed to estimate the gradient of the objective function, with detailed discussions found in [4, 203, 138], and [168].

Our work benefits from surrogate models in simulation [40, 60, 169], which are statistical models employed to approximate the simulation output and alleviate the expensive execution of simulation experiments. Owing to this advantage, surrogate models have wide applications in simulation fields, including simulation input uncertainty analysis [16, 182] and simulation optimization [112, 184, 151, 96, 171].

3.2 Problem Description

In this section, we describe the setting of the prompt selection. We consider a problem context where a user-pre-specified task is given, e.g., aiming to select, from many potential prompts, a good prompt that facilitates paragraph writing refinements. The set of prompts is $\tilde{\mathcal{P}} = \{\mathfrak{prom}_1, \mathfrak{prom}_2, \ldots\}$, where $\mathfrak{prom}_n$ denotes a human-readable prompt in text form (e.g., "Revise the following paragraph:"). The prompt and the input context (e.g., the paragraph requiring refinements) are fed into the generative language model and the model generates

the output as

$$\hat{\boldsymbol{y}} = \boldsymbol{\mathcal{A}}(\boldsymbol{x}, \mathfrak{prom}). \tag{3.1}$$

Here, $\boldsymbol{x}$ denotes the input context, $\boldsymbol{\mathcal{A}}$ represents the generative language model, and $\hat{\boldsymbol{y}}$ is the output context generated by the generative language model. We note that, given a fixed pair of input contexts and the prompt $(\boldsymbol{x}, \mathfrak{prom})$, the output contexts $\hat{\boldsymbol{y}}$ is a random object that can exhibit variability in different trials.

To quantify the performance of a prompt, our framework incorporates two components. Firstly, there is a baseline set, consisting of baseline contexts tailored to the task, providing a standard for comparison. Secondly, a score function is employed to quantitatively assess the quality of the output generated by the generative language model [34]. We define the baseline set as $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_M, \boldsymbol{y}_M)\}$. Each pair $(\boldsymbol{x}_m, \boldsymbol{y}_m)$ represents a baseline input-output context. For example, regarding the task of paragraph revision, $\boldsymbol{x}_i$ denotes the initial paragraph before revision and $\boldsymbol{y}_i$ denotes the revised paragraph. Such datasets can be collected from academic publications' draft and final versions, edit histories of collaborative writing platforms like Wikipedia, or datasets of student essays with teachers' feedback [163, 155]. To evaluate a prompt, say $\mathfrak{prom}'$, we input $(\boldsymbol{x}_m, \mathfrak{prom}')$ into the language model $\boldsymbol{\mathcal{A}}$, generating the output $\hat{\boldsymbol{y}}_m$. The score of the prompt $\mathfrak{prom}'$ is calculated by $h(\hat{\boldsymbol{y}}_m, \boldsymbol{y}_m)$, where $h(\cdot, \cdot) \in \mathbb{R}$ is the employed score function comparing the generated output $\hat{\boldsymbol{y}}_m$ with the baseline $\boldsymbol{y}_m$. A detailed description of this score function is in the supplements. A higher score implies greater similarity between $\hat{\boldsymbol{y}}_m$ and $\boldsymbol{y}_m$, indicating a 'better' performance of the prompt. In this way, given a baseline set $\mathcal{B}$ and a score function $h(\cdot, \cdot)$, the performance of a prompt is evaluated by

$$\widehat{v}(\mathfrak{prom}) = v(\mathfrak{prom}) + \epsilon(\mathfrak{prom}).$$

Here, $\widehat{v}(\mathfrak{prom})$ is the observed score, $v(\mathfrak{prom}) \doteq \mathbb{E}_{(\boldsymbol{x}_m, \boldsymbol{y}_m)\sim\mathcal{D}}[h(\boldsymbol{\mathcal{A}}(\boldsymbol{x}_m, \mathfrak{prom}), \boldsymbol{y}_m)]$ is the mean score of the prompt, and $\epsilon(\mathfrak{prom})$ is the uncertainty when observing the score of $\mathfrak{prom}$. We note that the uncertainty $\epsilon(\mathfrak{prom})$ comes from two aspects: 1) the random selection of a pair of $(\boldsymbol{x}_m, \boldsymbol{y}_m) \in \mathcal{B}$ with equal probabilities, and 2) the phenomenon that language model $\boldsymbol{\mathcal{A}}$ generates different output contexts even when the input context and the prompt are fixed. In this work, we assume the uncertainty ϵ's are Gaussian random variables that are independent across different evaluations.

With the mean score of a prompt defined, the selection is formulated as an optimization problem:

$$\mathfrak{prom}^* \in \arg\max_{\mathfrak{prom}\in\tilde{\mathcal{P}}} v(\mathfrak{prom}). \tag{3.2}$$

That is, we aim to select the prompt that achieves the highest mean score with a fixed baseline set $\mathcal{B}$ and a score function $v(\cdot)$. Considering the implicit dependence of the mean score on the selected prompt, we cast the selection (3.2) as a *simulation optimization* problem, regarding each evaluation of a prompt as simulating a sample from a stochastic system. On the other hand, utilizing classical simulation optimization methods for selecting the prompt encounters challenges:

1. **Implicit feasible set:** Regarding the prompt selection problem (3.2), there is not a specified set of prompts to be selected. The feasible set $\tilde{\mathcal{P}}$ of the optimization problem (3.2) is defined in a space that contains any potential combination of words and sentences. Existing simulation optimization methods in general consider feasible sets that are that are either finite or defined within a vector space, which is not feasible for prompt selection.

2. **Non-structural objective function & expensive evaluations:** The prompt selection problem remains challenging even when the feasible set is restricted to be finite. Firstly, given the complex mechanisms of the generative language model, the score as a function at prompts does not exhibit structural properties, such as convexity or Lipschitz continuity. This poses challenges to simulation optimization methods that rely on such structure [68, 65]. Secondly, while surrogate-based simulation optimization methods are feasible to optimize objective functions lacking explicit structural properties [141, 184, 171], these surrogate models in general use vector-represented inputs rather than text-based prompts. Lastly, in cases where the feasible set includes decision variables without inherent structure, the problem can be cast as a ranking and selection (R&S) problem [93]. On the other hand, most R&S methods require sufficient evaluation of each decision variable. In the context of prompt selection, such extensive evaluation is expensive, considering the computational time and costs associated with engaging the generative language model to generate contexts.

To address these challenges, we propose a framework for prompt selection. The framework is composed of two sequential stages: **1. the search stage** and **2. the sequential evaluation and selection stage**. In the initial search stage, we construct a set of candidate prompts to be evaluated and represent these human-readable prompts with real-valued and moderate-dimensional vectors. We first transform a few prompts in text form to high-dimensional vectors in a latent space, using the text autoencoder, a specialized machine learning model that numericalizes texts. We then perturb these vectors to attain a larger set of prompts and apply principal component analysis to reduce the dimensionality of the vectors representing prompts. Then, in the evaluation and selection stage, with a finite set of prompts represented by these moderate-dimensional vectors, we propose a sequential evaluation procedure for prompts based on a Bayesian parametric model. Specifically, we construct a surrogate model using the observed scores regarding vector-represented prompts to approximate the mean score of prompts and propose an acquisition function based on the constructed surrogate model. In each round, we select the prompt to be evaluated by maximizing the acquisition function, which accounts for both exploitation (evaluating prompts that have evidence for high scores) and exploration (evaluating prompts with high uncertainty). When the total budget of prompt evaluation is reached, the prompt that yields the highest mean observed score is selected. We also propose a refinement procedure following the sequential evaluation and selection stage. This involves constructing a surrogate model and a projection mapping from the high-dimensional latent space in the initial search

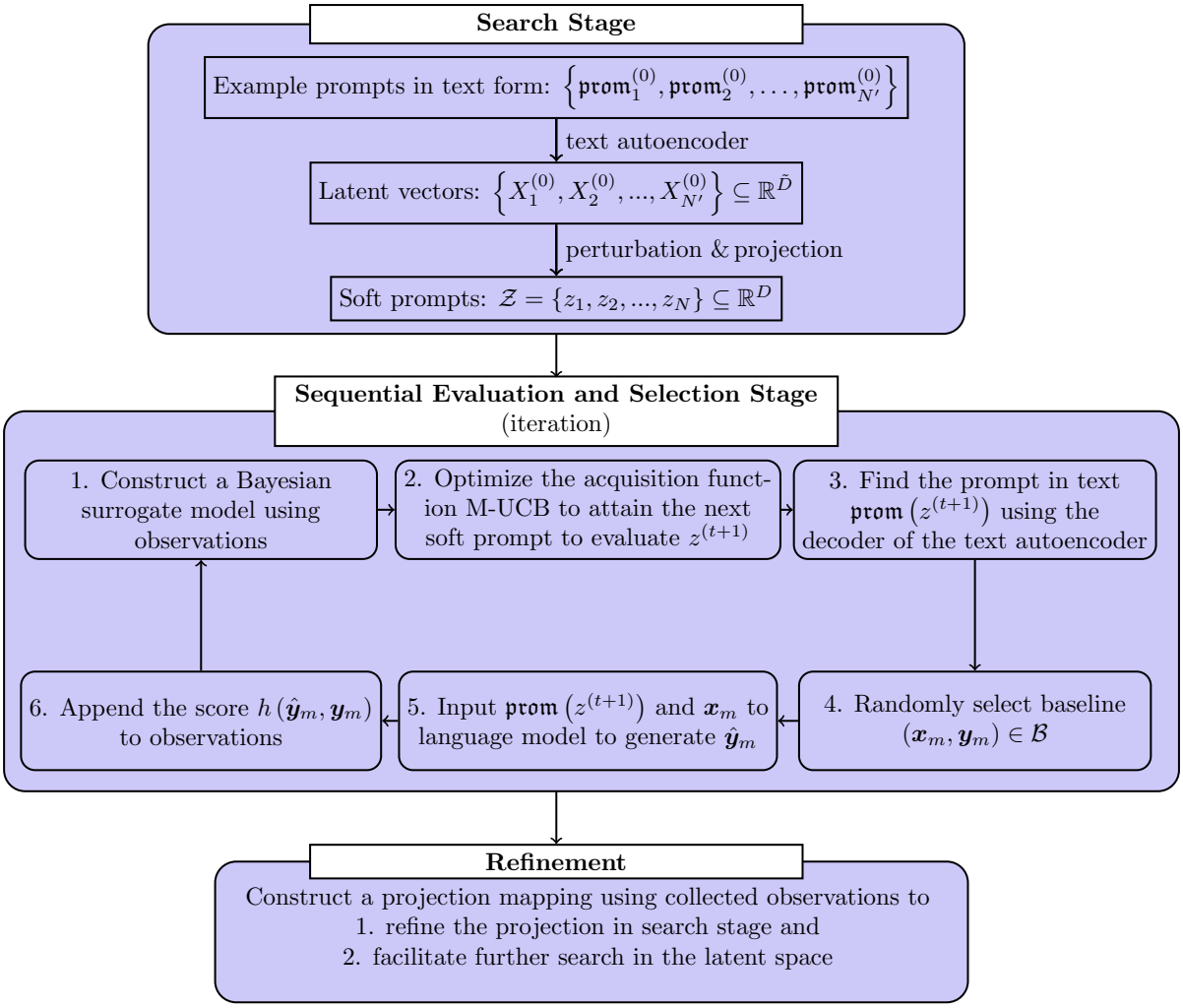

Figure 3.2: Our framework of prompt selection.

stage to a moderate-dimensional subspace. This surrogate model, along with the associated projection mapping, is then used to search for prompts that achieve higher scores than those represented by soft prompts. We present a summary illustration of our framework in **Figure 3.2**. Section 3.3 presents the search stage in detail; Section 3.4 describes the sequential evaluation and selection stage; and the refinement procedure is in Section 3.5.

3.3 Search Stage

We describe the search stage of our proposed prompt selection framework. We propose a procedure to construct a feasible set for the prompt selection problem (3.2), which transforms a language space containing any potential combinations of words and sentences in text form into a set of moderate-dimensional vectors. This numerical representation provides explicit quantification of the relationships and patterns among prompts, which may be implicit in their raw text form. We name these moderate-dimensional vectors *soft prompts*. In Section 3.3, we describe the procedure to represent a prompt in text form with a high-dimensional vector, which we name *latent vector*. In Section 3.3, we describe the procedure for constructing a set of soft prompts based on the latent vectors, serving as the feasible set for the prompt selection problem.

Prompt Vector Representation

Our procedure begins with a few example prompts that have been recognized as plausible for the specified task. For example, in terms of the paragraph revision task to be accomplished by ChatGPT, these example prompts include: 1. "Please revise the paragraph:", 2. "Can you revise the paragraph to be more formal and in the third person? Here's the original paragraph:", 3. "Could you please revise the paragraph to fix any grammatical errors and enhance its style? The paragraph is as follows:", etc. Instead of selecting from a limited number of these example prompts, it is anticipated that other prompts could lead to higher scores for the revision task. This anticipation is supported by existing prompt engineering procedures accomplished by human efforts; see [188, 99]. They found that the performance of prompts will be affected by adding descriptions and adverbials, changing word order, etc. In our framework, the example prompts serve as initial reference points. The scope of the feasible set $\tilde{\mathcal{P}}$ in the prompt selection problem (3.2), extends beyond these initial examples. On the other hand, actions on the example prompts, such as adding descriptions and adverbials, require human labor and are subjective and expensive in some scenarios. Instead of exploring new prompts directly based on these example prompts, we consider transforming these example prompts to vectors and conducting exploration sampling in the vector space.

Denote the set of the initial example prompts by $\tilde{\mathcal{P}}^{(0)} = \left\{\mathfrak{prom}_1^{(0)}, \mathfrak{prom}_2^{(0)}, \ldots, \mathfrak{prom}_{N'}^{(0)}\right\}$, where $\mathfrak{prom}_n^{(0)}$ is an example prompt for the specified task and N' denotes the number of the example prompts. To transform an example prompt in text form into a vector, we employ the text autoencoder model [117],

$$\mathrm{AE}(\mathrm{text}) = \mathrm{Dec}\left(\mathrm{Enc}\left(\mathrm{text}\right)\right) = \mathrm{text}'. \tag{3.3}$$

Here $\mathrm{AE}(\cdot)$ represents the autoencoder model, which is generally composed of two parts: an encoder and a decoder. The encoder model $\mathrm{Enc}(\cdot)$ is a nonlinear mapping that maps a text into a real-valued vector. Then, the decoder model $\mathrm{Dec}(\cdot)$ takes the vector as the

input and outputs another text. Here we focus on the reconstruction of the input text, i.e., we aim to train the autoencoder model such that the output text in (3.3) text$'$ is the same as the input text. A detailed description of the text autoencoder is in the supplements.

Note that the human-readable prompts are in text form, which is consistent with the input form of the text autoencoder (3.3). That is, after appropriate training, the text autoencoder described in (3.3) can transform each example prompt $\mathfrak{prom}_n^{(0)} \in \tilde{\mathcal{P}}^{(0)}$ into a vector $\mathrm{Enc}\left(\mathfrak{prom}_n^{(0)}\right)$ and then transform this vector back into the original prompt $\mathfrak{prom}_n^{(0)}$. In this way, we attain 1) a mapping $\mathrm{Dec}(\cdot)$ that transforms a real-valued vector to a text and 2) the vectors $\mathrm{Enc}\left(\mathfrak{prom}_n^{(0)}\right)$'s that represent the initial example prompts. We let $X_n^{(0)} \doteq \mathrm{Enc}\left(\mathfrak{prom}_n^{(0)}\right)$ and name it the latent vector. In this manner, the latent vector set is $\mathcal{X}^{(0)} = \left\{X_1^{(0)}, X_2^{(0)}, \ldots, X_{N'}^{(0)}\right\}$, where $X_n^{(0)} = \mathrm{Enc}\left(\mathfrak{prom}_n^{(0)}\right) \in \mathbb{R}^{\tilde{D}}$, and $\mathrm{Dec}\left(X_n^{(0)}\right) = \mathfrak{prom}_n^{(0)}$. We denote $\tilde{\mathcal{X}}$ as the *latent space*, which is a subspace of the $\tilde{D}$-dimensional vector space. In most applications, the latent space is set as $\tilde{\mathcal{X}} = [-1, 1]^{\tilde{D}}$ by default.

Soft Prompt Set Construction

We now present the procedure of extending the latent vector set and then projecting these vectors to a subspace with a moderate dimension. Recall that we have the set of latent vectors $\mathcal{X}^{(0)} = \left\{X_1^{(0)}, X_2^{(0)}, \ldots, X_{N'}^{(0)}\right\}$, where each $X_n^{(0)} \in \mathbb{R}^{\tilde{D}}$ represents a human-readable prompt $\mathrm{Dec}\left(X_n^{(0)}\right)$ for the pre-specified task. Instead of directly exploring the language space containing combinations of words and sentences, we search for other prompts by sampling new vectors in the latent space. That is, $\forall X^* \in \mathbb{R}^{\tilde{D}}$, the decoder $\mathrm{Dec}(X^*)$ will generate a text output, which might serve as a potential prompt. To decide whether a latent vector X^* is associated with a prompt that will be suitable for the pre-specified task, we compare $\mathrm{Dec}(X^*)$ with a baseline prompt $\mathfrak{prom}'$. The comparison is supported by the score function $v(\cdot, \cdot)$, which quantifies the similarity between two texts, and we provide details in the supplements. In this manner, we retain the latent vector when $v(\mathrm{Dec}(X^*), \mathfrak{prom}') \in (r_1, r_2)$, where r_1, r_2 are two user-selected thresholds. The selected thresholds are used to guarantee that the new prompt $\mathrm{Dec}(X^*)$ 1) exhibits sufficient difference from the baseline prompt so that it leads to a different mean score, and 2) maintains certain similarity so that it also works for the task. We sample new latent vectors based on the initial latent vectors. We first set $\mathcal{X} = \{X_1, X_2, \ldots, X_{N'}\}$, where $X_n = X_n^{(0)}$. Then the latent vector set is sequentially extended as follows:

1. Calculate a perturbation matrix $\mathbf{\Sigma}_p \in \mathbb{R}^{\tilde{D}\times\tilde{D}}$ based on the current set

$$\mathcal{X} = \{X_1, X_2, \ldots, X_L\}.$$

   In our method, we select the sample covariance matrix for perturbation. That is, $\mathbf{\Sigma}_p = \frac{1}{L}\sum_{n=1}^{L}\left(X_n - \bar{X}\right)\left(X_n - \bar{X}\right)^\top$, where $\bar{X} = \frac{1}{L}\sum_{n=1}^{L} X_n$.

2. Randomly select a latent vector $X_l \in \mathcal{X}$ with probability $p_l = \frac{\exp(-\tilde{n}_l)}{C_L}$, where $\tilde{n}_l$ denotes the number of times X_l has been selected and $C_L = \sum_{n=1}^{L} \exp(-\tilde{n}_n)$ is a normalizing constant. Then generate from X_l a new latent vector $X' = X_l + V, V \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_p)$.

3. Accept the new latent vector X' when 1) $X' \in \tilde{\mathcal{X}} = [0,1]^{\tilde{D}}$ and 2) $v(\mathrm{Dec}(X'), \mathfrak{prom}') \in (r_1, r_2)$, where $\mathfrak{prom}' = \mathrm{Dec}(X_l)$. If the latent vector X' is not accepted, go back to Step 1. If the latent vector X' is accepted, then append it to the latent vector set $\mathcal{X}$ and go back to Step 1.

The procedure terminates when the number of elements in the latent vector set reaches a pre-specified threshold, N, and we attain an extended set of vectors, $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$. As mentioned in Section 3.3, the dimensionality of the latent vectors, $\tilde{D}$, is generally high, which poses challenges to the subsequent evaluation and selection stage, where we construct a surrogate model of the mean value regarding prompt's numerical representations. High-dimensional vectors increase the sample size required for the surrogate model construction, thus leading to excessive costs. Therefore, we apply a dimension reduction algorithm to the latent vector set $\mathcal{X}$, transforming each $X_n \in \mathcal{X}$ into a vector $z_n \in \mathbb{R}^D$ with a moderate dimension D, named as a soft prompt. Specifically, we employ the method of principal component analysis (PCA); see [29]. We describe the detailed procedure of PCA in the supplements.

Upon completion of dimensionality reduction, we attain a soft prompt set

$$\mathcal{Z} = \{z_1, z_2, \ldots, z_N\} \subseteq \mathbb{R}^D.$$

Each soft prompt z_n is associated with the initial latent vector X_n, which further links to a human-readable prompt $\mathrm{Dec}(X_n)$ by the lens of the decoder model in Section 3.3. In this manner, we propose a procedure that constructs a feasible set $\mathcal{Z}$ for the prompt selection problem from the language space that contains any combination of words and sentences. The constructed feasible set $\mathcal{Z}$ consists of a finite but sufficient number of vectors, providing explicit quantification of relationships and patterns among prompts and facilitating the prompt selection procedure.

3.4 Evaluation and Selection Stage

In this section, we describe the procedure of evaluating and selecting the prompt for a generative language model. The prompts to be evaluated and selected from are represented by moderate-dimensional vectors, termed soft prompts, $z_n \in \mathcal{Z} = \{z_1, z_2, \ldots, z_N\} \subseteq \mathbb{R}^D$, where N is the number of soft prompts and D is the dimension of the vector-valued soft prompt. Each soft prompt z_n is derived from a latent vector $X_n \in \mathcal{X} \subseteq \mathbb{R}^{\tilde{D}}$ by a dimension reduction procedure as in Section 3.3. Thus, we have a one-to-one mapping from z_n to X_n, and the latent vector can be further mapped to a human-readable prompt in text form by

applying the encoder $\mathrm{Enc}(X_n)$. In this way, each soft represents a human-readable prompt denoted by $\mathfrak{prom}(z_n)$.

To observe a score of the soft prompt, our framework involves two components: 1) a baseline set $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_M, \boldsymbol{y}_M)\}$, where each $(\boldsymbol{x}_m, \boldsymbol{y}_m)$ denotes a pair of baseline input-output contexts for comparison; and 2) a score function $h(\cdot, \cdot)$ to quantify the similarities between two contexts. In this way, given a soft prompt z_n, we attain the human-readable prompt $\mathfrak{prom}(z_n)$ in text form. We then randomly select a pair of baseline input-output contexts $(\boldsymbol{x}_m, \boldsymbol{y}_m)$ from the baseline set $\mathcal{B}$ with equal probabilities. After feeding the human-readable prompt $\mathfrak{prom}(z_n)$ and the input context $\boldsymbol{x}_m$ to the language model, we have the generated output $\hat{\boldsymbol{y}}_m(z_n) = \boldsymbol{f}(\boldsymbol{x}_m, \mathfrak{prom}(z_n))$ as in (3.1), where $\boldsymbol{f}$ represents the generative language model. By comparing the generated output $\hat{\boldsymbol{y}}_m(z_n)$ and baseline output context $\boldsymbol{y}_m$ by the score function, we then observe a score

$$\widehat{v}_{n,m} = h(\hat{\boldsymbol{y}}_m(z_n), \boldsymbol{y}_m) = v(\mathfrak{prom}(z_n)) + \epsilon_{n,m}. \tag{3.4}$$

Here, $v(\mathfrak{prom}(z_n)) = \mathbb{E}[\widehat{v}_{n,m}]$is the mean score of the prompt associated with the soft prompt z_n, which serves as the objective function to be maximized. Since the human-readable prompt $\mathfrak{prom}(z_n)$ is fully dependent on the soft prompt z_n, we let $v(z_n) \doteq v(\mathfrak{prom}(z_n))$ for notational simplicity. In addition, $\epsilon_{n,m} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_n^2)$ represents the uncertainty contained in the observed score, which is independent across different selections of z_n and different trials with a fixed z_n.

To find the prompt with the highest mean score $v(z_n)$, we propose a sequential selection procedure that employs a Bayesian parametric model as the surrogate model for the mean score with respect to soft prompts. Specifically, the procedure first selects some soft prompts to observe the scores during the warm-up step (Section 3.4). The observed scores are used to construct a surrogate model (Section 3.4) and an associated acquisition function (Section 3.4). Then, during the sequential evaluation step, the next soft prompt to be evaluated is determined by maximizing the acquisition function. After observing new scores, the surrogate model and the acquisition function are updated and used to guide the selection of the next prompt to be evaluated. The procedure terminates when the budget of evaluating prompts T runs out.

Warm-up Step

In this section, we describe the warm-up step that observes scores for some soft prompts z_n's in preparation for the sequential evaluation step. Recall that we have the set of prompts $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$. In the warm-up stage, we select $N_W (< N)$ soft prompts $z_{W;n} \in \mathcal{Z}$ and for each we observe R scores as in (3.4). The selection of $z_{W;n}$'s is flexible and user-specified. Here we provide a practical approach. As described in Section 3.3, each soft prompt $z_n \in \mathcal{Z}$ is attained by transforming a latent vector $X_n \in \mathcal{X}$ with a dimension reduction algorithm. Furthermore, this latent vector set $\mathcal{X}$ is expanded by an initial latent vector set $\mathcal{X}^{(0)}$, which contains the latent vectors $X_n^{(0)}$'s that are attained from the initial

example prompts $\left\{\mathfrak{prom}_1^{(0)}, \mathfrak{prom}_2^{(0)}, \ldots, \mathfrak{prom}_{N'}^{(0)}\right\}$. In other words, the soft prompts z_n's (as well as the human-readable prompts $\mathfrak{prom}(z_n)$'s) are attained by perturbing the initial example prompts since the latent vectors are attained by perturbation on the initial latent vectors as the sampling procedure in Section 3.3. Therefore, the soft prompts are generated directly or indirectly from the soft prompts that represent the initial example prompts. Thus, the soft prompts representing the initial example prompts are among the most representative ones, and we therefore select these soft prompts in the warm-up stage to evaluate. That is, we set $\mathcal{Z}_W = \{z_{W;1}, z_{W;2}, \ldots, z_{W;N_W}\} \subset \mathcal{Z}$, where $N_W = N'$ and each $z_{W;n}$ is attained by $X_n^{(0)} \in \mathcal{X}^{(0)}$.

After deciding the set $\mathcal{Z}_W$, we collect $R = 5$ observations for each $z_{W;n} \in \mathcal{Z}_W$. We then approximate the variance of the observed scores $\sigma_n^2 = \mathrm{Var}\left[\widehat{v}_{n,m}\right]$ by

$$\widehat{\sigma}_n^2 = \frac{1}{R-1}\sum_{r=1}^{R}\left(\widehat{v}_{n,r} - \bar{v}_n\right)^2, \forall n \in \{n : z_n \in \mathcal{Z}_W\},$$

where $\bar{v}_n = \frac{1}{R}\sum_{r=1}^{R}\widehat{v}_{n,r}$. With the set $\mathcal{D}_W \doteq \left\{\left(z_{W;1}, \widehat{\sigma}_{W;1}^2\right), \ldots, \left(z_{W;N_W}, \widehat{\sigma}_{W;N_W}^2\right)\right\}$, we construct a predictive model $g^*(z_n) \in \mathbb{R}, z_n \in \mathcal{Z}$ by $g^* \in \arg\min_{g\in\mathcal{G}}\sum_{z_n\in\mathcal{Z}_W}\left(g(z_n) - \widehat{\sigma}_n^2\right)^2$, where $\mathcal{G}$ denotes a class of predictive models (e.g., a linear regression model with unknown parameters) and the selection of the predictive model is flexible and user-specified. In our experiments, we use the kriging method as suggested by [5]. After the predictive model g^* is attained, we then approximate σ_n^2 for $\forall z_n \in \mathcal{Z}$ by $g^*(z_n)$. In our work, we treat the variance σ_n^2 as a nuisance parameter, and the difference between the approximated variances $g^*(z_n)$'s and the ground-truth variances σ_n^2's will be ignored. These approximated variances will be incorporated into the surrogate model in the sequential evaluation step, which resembles classical surrogate models construction [5, 40, 39].

Sequential Evaluation Step

In this section, we describe the sequential evaluation step, which consists of two actions: **1. Approximation of Mean Score:** We approximate the mean score of each soft prompt, from historical data of observed scores by constituting a Bayesian parametric model. This model also provides the explicit uncertainty quantification of the mean score approximation. **2. Optimization of Acquisition Function:** Once the surrogate model is constructed, the next step involves optimizing an acquisition function that accounts for both the approximated mean scores of each soft prompt and the approximation uncertainty. The next soft prompt to be evaluated is decided by maximizing the acquisition function.

Bayesian Parametric Surrogate Model

We assume that the mean performance v with respect to the soft prompt z is represented by a parametric model $v(z) = \boldsymbol{f}(z; \boldsymbol{W})$, where the form of the model $\boldsymbol{f}(\cdot;\cdot)$ is known and

$\boldsymbol{W} \in \mathbb{R}^p$ is an unknown parameter. In this work, the selection of the parametric model is flexible, and we propose a procedure to sequentially select the soft prompt z_n to evaluate using Bayesian inference. Specifically, we regard the unknown parameter as a random vector with prior distribution $\pi(\boldsymbol{W})$. Suppose we have evaluated the prompts for t rounds (including the observations in the warm-up step), and have collected the observed scores $\mathcal{S}_t \doteq \left\{\left(z^{(1)}, \widehat{v}^{(1)}\right), \ldots, \left(z^{(t)}, \widehat{v}^{(t)}\right)\right\}$. Here $z^{(\tau)} \in \mathcal{Z} = \{z_1, z_2, \ldots, z_n\}$ denotes the selected soft prompt in the τ-th round, and $\widehat{v}^{(\tau)}$ is the observed score associated with $z^{(\tau)}$ as in (3.4). We note that an equivalent representation of the dataset is $\mathcal{S}_t = \{(z_n, \widehat{v}_{n,m})\}$ for $m \in \{1, 2, \ldots, r_n(t)\}$ and $n \in \{1, 2, \ldots, N\}$. Here $\widehat{v}_{n,m}$ denotes the m-th observed score for soft prompt z_n as in (3.4), and $r_n(t)$ denotes the number of evaluations at z_n up to time t. That is, $\sum_{n=1}^{N} r_n(t) = t$. Conditional on the dataset $\mathcal{S}_t$, the posterior distribution of $\boldsymbol{W}$ is updated by

$$p\left(\boldsymbol{W} \mid \mathcal{S}_t\right) = \frac{p\left(\mathcal{S}_t \mid \boldsymbol{W}\right) \pi\left(\boldsymbol{W}\right)}{p\left(\mathcal{S}_t\right)} \propto p\left(\mathcal{S}_t \mid \boldsymbol{W}\right) \pi\left(\boldsymbol{W}\right), \tag{3.5}$$

where

$$p\left(\mathcal{S}_t \mid \boldsymbol{W}\right) = \prod_{n=1}^{N} \prod_{m=1}^{r_n(t)} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{\left(\widehat{v}_{n,m} - \boldsymbol{f}\left(z_n; \boldsymbol{W}\right)\right)^2}{2\sigma_n^2}\right)$$

is the likelihood of the observed scores. In this way, inference for the mean score function is based on $\boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}\right), \widehat{\boldsymbol{W}} \sim p\left(\boldsymbol{W} \mid \mathcal{S}_t\right)$. For parametric models in which the exact posterior is intractable, Markov Chain Monte Carlo (MCMC) methods [8] or variational inference (VI) [20] can be employed to approximate and generate samples from the posterior distribution. We postpone the sampling methods of the posterior distribution to the supplements.

We now provide two examples of the parametric model that can be employed in our approach.

Example 3.1 (Gaussian Process)**.** *The Gaussian process (GP) has been widely employed in extensive applications including simulation input uncertainty analysis [182, 16] and simulation optimization [141, 171]. Specifically, the GP-based method assumes that* $\boldsymbol{f}(z) \sim \mathcal{GP}\left(0, \boldsymbol{K}\left(z, z'\right)\right)$*, where* $\boldsymbol{K}\left(z, z'\right)$ *is a pre-specified kernel function. The kernel function quantifies the similarity of the surrogate model* f *between different inputs* z*'s. A common selection is the radial basis function (RBF) kernel* $\boldsymbol{K}_{\mathrm{RBF}}\left(z, z'\right) = \exp\left\{-\frac{\|z - z'\|^2}{\sigma^2}\right\}$*, where* $\|\cdot\|$ *denotes the Euclidean norm of a vector and* σ^2 *is a user-specified hyperparameter. Given the observed points, inference for the function value* f *at an arbitrary point* $\tilde{z}$ *is based on the conditional distribution of a multivariate normal distribution. That is,* $\boldsymbol{f}\left(\tilde{z}\right) \mid \tilde{S}_t \sim \mathcal{N}\left(\mu_{GP;t}\left(\tilde{z}\right), \sigma^2_{GP;t}\left(\tilde{z}\right)\right)$*. Here,* $\tilde{S}_t = \left\{\left(z^{(1)}, y^{(1)}\right), \ldots, \left(z^{(t)}, y^{(t)}\right)\right\}$ *denotes the pairs of inputs and observations up to time* t*, where* $y^{(\tau)} = \boldsymbol{f}\left(z^{(\tau)}\right) + \epsilon^{(\tau)}$ *denotes the noisy observation, and* $\epsilon^{(\tau)} \overset{i.i.d.}{\sim} \mathcal{N}\left(0, \sigma^2\right)$ *denotes the noise. The conditional mean and variance*

adopt explicit expressions

$$\begin{aligned}\mu_{GP;t}(\tilde{z}) &= \boldsymbol{K}_t(\tilde{z})^\top \left(\tilde{K}_t + \sigma^2 \boldsymbol{I}_t\right)^{-1} \tilde{\boldsymbol{y}}_t, \\ \sigma^2_{GP;t}(\tilde{z}) &= \boldsymbol{K}(\tilde{z},\tilde{z}) - \boldsymbol{K}_t(\tilde{z})^\top \left(\tilde{K}_t + \sigma^2 \boldsymbol{I}_t\right)^{-1} \boldsymbol{K}_t(\tilde{z}),\end{aligned} \tag{3.6}$$

where $\tilde{\boldsymbol{y}}_t = \left(y^{(1)},\ldots,y^{(t)}\right)^\top \in \mathbb{R}^t$ *is the vector of observations;* $\tilde{K}_t \in \mathbb{R}^{t\times t}$ *is the kernel matrix with* (τ,τ')*-th entry* $\boldsymbol{K}\left(z^{(\tau)}, z^{(\tau')}\right)$*; and*

$$\boldsymbol{K}_t(\tilde{z}) = \left(\boldsymbol{K}\left(\tilde{z}, z^{(1)}\right), \boldsymbol{K}\left(\tilde{z}, z^{(2)}\right), \ldots, \boldsymbol{K}\left(\tilde{z}, z^{(t)}\right)\right)^\top \in \mathbb{R}^t$$

is the vector of kernel function values between $\tilde{z}$ *and* $\left\{z^{(\tau)}\right\}_{\tau=1}^t$.

The GP model is generally regarded as a Bayesian nonparametric model. However, when the selected kernel function has a finite rank, that is $\boldsymbol{K}(z,z') = \phi(z)^\top \phi(z')$ *for* $\phi(z) \in \mathbb{R}^p$*, the GP model has an alternative parametric representation* $\boldsymbol{f}(z;\boldsymbol{W}) = \phi(z)^\top \boldsymbol{W}$*. Here,* $\boldsymbol{W} \in \mathbb{R}^p$ *denotes the unknown parameter with the prior distribution* $\pi(\boldsymbol{W}) \overset{D}{=} \mathcal{N}(\boldsymbol{0}_p, \boldsymbol{I}_p)$*, and the posterior distribution of the unknown parameters is*

$$p\left(\boldsymbol{W} \mid \tilde{S}_t\right) \overset{D}{=} \mathcal{N}\left(\left(\Phi_t^\top \Phi_t + \sigma^2 \boldsymbol{I}_t\right)^{-1} \Phi_t^\top \tilde{\boldsymbol{y}}_t, \sigma^2\left(\Phi_t^\top \Phi_t + \sigma^2 \boldsymbol{I}_t\right)^{-1}\right),$$

where $\Phi_t = \left[\phi\left(z^{(1)}\right), \ldots, \phi\left(z^{(t)}\right)\right]^\top \in \mathbb{R}^{t\times m}$*. In this way, inference for*

$$\boldsymbol{f}\left(\tilde{z}; \widehat{\boldsymbol{W}}\right) = \phi(\tilde{z})^\top \widehat{\boldsymbol{W}}$$

with the posterior distribution $\widehat{\boldsymbol{W}} \sim p\left(\boldsymbol{W} \mid \tilde{S}_t\right)$ *matches the results in (3.6). That is, GP is equivalent to a Bayesian parametric model, parametrized by* $\boldsymbol{W}$ *when the kernel function has a finite rank.*

As documented in [152], and [57], the performance of GP-based algorithms depends heavily on the appropriate selection of the kernel function $\boldsymbol{K}(z,z')$. In some applications when the unknown function to be approximated has highly non-structural dependence on the inputs, it is challenging to pre-specify the employed kernel function. Furthermore, because of calculating the inverse matrix, GP-based algorithms suffer from computational complexities of the cubic order of the observed data. Thus, existing work also considers incorporating Bayesian inference with deep learning models to enhance the model's flexibility and inference accuracy. Next we consider the neural network [138, 168] as an example, where the weight parameters of the neural network are regarded as random and updated by data as in (3.5).

Example 3.2 (Bayesian Neural Network)**.** *Neural networks are computing systems that are used to approximate unknown mappings. A neural network is composed of connected layers of nodes, which are denoted as neurons. The layers of a neural network include the input layer,*

the output layer, and the hidden layers that connect the input layer and the output layer. For each pair of adjacent layers, the input of the latter layer is the output of the former layer. The connection between the neurons of adjacent layers is represented by the linear functions, while the activation function in each layer imposes non-linearity to the neural network. To be more specific, let $\boldsymbol{f}(z;\boldsymbol{W})$ *be the neural network with* L *layers of hidden layers. The innermost layer (the input layer) is represented as the initial layer, and suppose the* l*-th layer contains* m_l *neurons. Thus,* $m_0 = d^{in}$ *and* $m_{L+1} = d^{out}$*, consistent with the dimensions of the inputs and outputs of the mapping to be approximated. In this way, the neural network* $\boldsymbol{f}(z;\boldsymbol{W})$ *is represented as*

$$\boldsymbol{f}(z;\boldsymbol{W}) = W_L \cdot \varphi\left(W_{L-1} \cdots \varphi\left(W_0 z + b_0\right) \cdots + b_{L-1}\right) + b_L.$$

Here, W_l *is the* $m_{l+1} \times m_l$ *weight matrix, and* b_l *is the* m_{l+1}*-dimensional intercept. All these weight matrices and intercepts compose the weight parameters* $\boldsymbol{W}$ *of a neural network. The activation function* $\varphi(\cdot)$ *is defined on each entry (neuron) respectively and imposes non-linearity to the neural network. Common selections of activation functions include the rectified linear unit (ReLU) function and the tanh function. When the weight parameters are regarded as random variables and adopt a prior distribution, the neural network* $\boldsymbol{f}(z;\boldsymbol{W})$ *is known as a Bayesian neural network.*

Other models including Bayesian linear regression models [129], Bayesian hierarchical models [143] and variational autoencoders (VAE) [108] can also be employed to approximate the mean score function $v(\cdot)$. The selection of the surrogate model is flexible and depends on the complexity of the problem and the sample size of the observations. For example, when there are not sufficient observations, the parametric model with few parameters and a relatively explicit form (e.g., Bayesian linear regression model) is preferred. On the other hand, when a large number of observations have been acquired and the mean score is highly non-structural with soft prompts, deep learning models are preferable. With the specified Bayesian parametric model in hand, we next describe the *acquisition function*, which is used to guide the selection of the soft prompt to be evaluated in the next round.

Acquisition Function & Optimization

In this section, we present the acquisition function in the sequential evaluation step, which is used to guide the selection of the next soft prompt $z^{(t+1)}$ to be evaluated. Based on the surrogate model in each round $\boldsymbol{f}\left(z;\widehat{\boldsymbol{W}}\right), \widehat{\boldsymbol{W}} \sim p\left(\boldsymbol{W} \mid \mathcal{S}_t\right)$, we propose an acquisition function named *Modified Upper Confidence Bound (M-UCB)*, defined on $z_n \in \mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$,

$$\alpha_t\left(z_n\right) = \mu_t\left(z_n\right) + \beta_t\left(\sigma_t\left(z_n\right) + \gamma\left(r_n\left(t\right)\right)\right). \tag{3.7}$$

Here $\mu_t\left(z_n\right) = \mathbb{E}\left[\boldsymbol{f}\left(z_n;\widehat{\boldsymbol{W}}\right) \mid \mathcal{S}_t\right]$ and $\sigma_t\left(z_n\right) = \left\{\mathrm{Var}\left[\boldsymbol{f}\left(z_n;\widehat{\boldsymbol{W}}\right) \mid \mathcal{S}_t\right]\right\}^{1/2}$ are the posterior mean and standard deviation of the surrogate model at z_n, and $r_n(t)$ denotes the number of

evaluations at z_n up to time t. In addition, $\{\beta_\tau \in \mathbb{R}\}_{\tau=1,2,\ldots,t}$ is a user-selected non-decreasing sequence, and $\gamma(\cdot) \in \mathbb{R}$ is a pre-specified decreasing function defined on $\mathbb{N} = \{0, 1, 2, \ldots\}$ satisfying $\lim_{n\to\infty} \gamma(n) = 0$. The selection of $\{\beta_\tau \in \mathbb{R}\}_{\tau=1,2,\ldots,t}$ and $\gamma(\cdot)$ is discussed in the supplements.

The soft prompt to be evaluated in the next round is selected by maximizing the acquisition function. That is, $z^{(t+1)} \in \arg\max_{z_n \in \mathcal{Z}} \alpha_t(z_n)$, where $z^{(t+1)}$ is the soft prompt to be evaluated in the next round. After observing the score $\widehat{v}^{(t+1)}$, the surrogate model is updated via updating the posterior distribution (3.5) with $\mathcal{S}_{t+1} = \left\{\left(z^{(t+1)}, \widehat{v}^{(t+1)}\right)\right\} \cup \mathcal{S}_t$. The sequential selection procedure iterates until the number of rounds t meets the total budget T. The procedure of the entire evaluation and selection stage is summarized in **Algorithm 3.1**.

Algorithm 3.1 General Procedure of the Evaluation and Selection Stage with M-UCB

Require: The feasible set of prompts $\mathcal{Z} = \{z_1, \ldots, z_N\}$, a parametric model $\boldsymbol{f}(\cdot; \boldsymbol{W})$, a prior distribution of unknown parameters $\pi(\boldsymbol{W})$, the total budget of evaluating prompts T, and the number of evaluations at each soft prompt R in the warm-up stage.

Ensure: The selected soft prompt $\widehat{z}^*$.

Warm-Up Step:

Decide the set of prompts $\mathcal{Z}_W$ to be evaluated in the warm-up step as in Section 3.4.
for $\forall z_{W;n} \in \mathcal{Z}_W$ **do**
    Evaluate $z_{W;n}$ for R times.
    Record the sample variances $\hat{\sigma}^2_{W;n}$.
end for
Approximate σ_n^2 for $n \in \{1, 2, \ldots, N\}$.
Set $T_W = |\mathcal{Z}_W| R$.

Sequential Evaluation Step:

Let $t = T_W$ and $\mathcal{S}_t = \left\{\left(z^{(1)}, \widehat{v}^{(1)}\right), \ldots, \left(z^{(t)}, \widehat{v}^{(t)}\right)\right\}$.
while $t < T$ **do**
    Update the posterior distribution $p(\boldsymbol{W} \mid \mathcal{S}_t)$ as in (3.5).
    Selecting the next soft prompt $z^{(t+1)}$ by maximizing the M-UCB function

$$z^{(t+1)} \in \arg\max_{z_n \in \mathcal{Z}} \alpha_t(z_n).$$

    Observe $\widehat{v}^{(t+1)}$ with $z^{(t+1)}$ as in (3.4).
    Update the historical dataset $\mathcal{S}_{t+1} = \left\{\left(z^{(t+1)}, \widehat{v}^{(t+1)}\right)\right\} \cup \mathcal{S}_t$ and let $t \leftarrow t + 1$.
end while
return $n^* = \arg\max_{n\in 1,2,\ldots,N} \frac{1}{r_n(T)} \sum_{m=1}^{r_n(T)} \widehat{v}_{n,m}$ and $\widehat{z}^* = z_{n^*}$.

The acquisition function M-UCB balances the exploitation-exploration trade-off. Specifically, the posterior mean represents the model's approximation of the mean scores of soft prompts, guiding the exploitation of known high-performing areas. The exploration is led by both the posterior standard deviation $\sigma_t(z_n)$ and the number of evaluations $r_n(t)$. We

provide the selection of β_t and $\gamma(\cdot)$ in our numerical experiments. We now present the consistency of **Algorithm 3.1**. We make the following assumptions.

Assumption 3.1.

1. *For each soft prompt, the mean score is identifiable with different $\boldsymbol{W}$'s and the squared mean score is integrable with respect to the prior. That is, $\forall z_n \in \mathcal{Z}, \boldsymbol{f}(z_n; \boldsymbol{W}) \neq \boldsymbol{f}(z_n; \boldsymbol{W}')$ when $\boldsymbol{W} \neq \boldsymbol{W}'$, and $\mathbb{E}_{\boldsymbol{W}\sim\pi(\boldsymbol{W})}[\boldsymbol{f}^2(z_n; \boldsymbol{W})] < \infty$.*
2. *For the user-selected hyperparameter β_t, $\lim_{t\to\infty} \beta_t = \infty$.*

Theorem 3.1. *Let $z^* \in \arg\max_{z_n\in\mathcal{Z}} v(z_n)$ be the prompt with the highest mean score and $\widehat{z}^*$ be the selected soft prompt by **Algorithm 3.1**. Under Assumption 3.1,*

$$\lim_{T\to\infty} v(\widehat{z}^*) \overset{w.p.1}{=} v(z^*).$$

Theorem 3.1 gives us the consistency of the sequential evaluation step of our framework. The first condition in Assumption 3.1 appears to be a common assumption used in Bayesian inference to regularize the model; see [166] for example. The second condition is commonly employed in the literature related to the Gaussian process and neural network bandit algorithms [157, 202]. As $t \to \infty$, the posterior variance at each $\boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}\right)$ shrinks to zero. In this way, the rescaled M-UCB function $\alpha'_t(z_n) = \frac{\mu_t(z_n)}{\beta_t} + \sigma_t(z_n) + \gamma(r_n(t))$ approaches zero if and only if $r_n(t) \to \infty$. The detailed proof is postponed to the supplements.

When the posterior distribution of the unknown parameters $p(\boldsymbol{W} \mid \mathcal{S}_t)$ does not adopt a closed-form expression, it is challenging to obtain the closed-form expression of $\mu_t(z_n)$ and $\sigma_t(z_n)$ as well. In these scenarios, we simulate samples $\widehat{\boldsymbol{W}}_k \sim p(\boldsymbol{W} \mid \mathcal{S}_t)$ with a selected sampling algorithm. Then, for each soft prompt z_n, we have a set of samples $\mathcal{A}_n \doteq \left\{\boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}_1\right), \boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}_2\right), \ldots, \boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}_K\right)\right\}$ to infer the mean score $v(z_n)$. Thus, when optimizing the acquisition function M-UCB (3.7), we instead consider

$$\arg\max_{z_n} \left\{\widehat{\alpha}_t(z_n) \doteq \widehat{\mu}_t(z_n) + \beta_t\left(\widehat{\sigma}_t(z_n) + \gamma(r_n(t))\right)\right\},$$

where

$$\widehat{\mu}_t(z_n) = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}_k\right) \quad \text{and} \quad \widehat{\sigma}_t(z_n) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(\boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}_k\right) - \widehat{\mu}_t(z_n)\right)^2} \tag{3.8}$$

are the sample mean and standard deviation associated with the simulated samples $\mathcal{A}_n$. Recall that $z^{(t+1)} \in \arg\max_{z_n\in\mathcal{Z}} \alpha_t(z_n)$.

Proposition 3.1. *Suppose $\widehat{\boldsymbol{W}}_1, \ldots, \widehat{\boldsymbol{W}}_K \overset{i.i.d.}{\sim} p(\boldsymbol{W} \mid \mathcal{S}_t)$ and $\max_{z_n\in\mathcal{Z}} \mathbb{E}[f^2(z_n; \boldsymbol{W}) \mid \mathcal{S}_t] < \infty$. Let $\bar{z}_t^* \in \arg\max_{z_n\in\mathcal{Z}} \widehat{\alpha}_t(z_n)$. Then*

$$\lim_{K\to\infty} \widehat{\alpha}_t(\bar{z}_t^*) \overset{w.p.1.}{=} \max_{z\in\mathcal{Z}} \alpha_t(z).$$

We refer to [8] for detailed discussions on practical implementations and theoretical support of algorithms for simulating samples $\widehat{\boldsymbol{W}}_k$, and provide two specific methods in the supplements.

On the other hand, the number of soft prompts could be large in some scenarios. Thus, it is time-consuming to acquire each $\mathcal{A}_n$ in each round. To alleviate the computational burdens, we employ a probabilistic reparameterization method [48] to optimize the acquisition function. That is, we assign a probability mass function to the feasible set of soft prompts $z_n \sim p(z;\theta)$, where $p(z;\theta)$ is a probability mass function defined on $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$, and parametrized by a continuous parameter $\theta \in \Theta$. Specifically, we set $\Theta = [0,1]^N$ and let $z = z_i$ with probability $\frac{\theta^{(i)}}{\sum_{j=1}^N \theta^{(j)}}$, where $\theta^{(i)}$ denotes the i-th entry of θ. We define

$$\theta_t^* = \arg\max_{\theta\in\Theta} \left\{ \widetilde{\alpha}_t(\theta) \doteq \mathbb{E}_{p(z;\theta)} \left[\alpha_t(z) \right] \right\},$$

which is a continuous optimization problem regarding $\theta \in \Theta$, and can be facilitated efficiently with gradient-based algorithms. After determining θ_t^*, we generate $\widehat{z}^{(t+1)} \sim p(z;\theta_t^*)$ and then observe the score associated with $\widehat{z}^{(t+1)}$ as in (3.4).

We name the acquisition function $\widetilde{\alpha}_t(\theta)$ *Probabilistic Reparameterized Modified Upper Confidence Bound (PR-M-UCB)*, and provide the theoretical support of using PR-M-UCB.

Theorem 3.2. *Let* $\mathcal{H}_t^* = \{z \in \arg\max_{z\in\mathcal{Z}} \alpha_t(z)\}$ *and* $\mathcal{J}_t^* = \{\theta \in \arg\max_{\theta\in\Theta} \widetilde{\alpha}_t(\theta)\}, \forall t > 0$. *Let*

$$\hat{\mathcal{H}}_t^* = \{z : \theta \in \mathcal{J}_t^*, z \in \text{support}\,(p(z;\theta))\}.$$

Then, $\widehat{\mathcal{H}}_t^* = \mathcal{H}_t^*, \forall t > 0$.

Theorem 3.2 states that the maximizers of PR-M-UCB lead to a probability distribution that generates the maximizers of M-UCB. Specifically, when the maximizer of M-UCB is unique, the maximizers of PR-M-UCB lead to a point mass distribution supported only at the unique maximizer of M-UCB.

The advantage of PR-M-UCB over M-UCB is that the maximization of PR-M-UCB is a continuous optimization problem, which can be facilitated by gradient ascent efficiently and effectively [142]. Indeed, the differentiability of PR-M-UCB is supported by the differentiability of $p(z;\theta)$ with θ, and is formalized into the following proposition.

Proposition 3.2. $\widetilde{\alpha}_t(\theta)$ *is differentiable with* $\theta \in \Theta$, *and the gradient is*

$$\nabla_\theta \widetilde{\alpha}(\theta) = \mathbb{E}_{\nabla_\theta p(z;\theta)} \left[\alpha_t(z) \right].$$

Furthermore, we provide an unbiased estimator of the gradient with the following proposition.

Proposition 3.3. *An unbiased estimator of the gradient* $\nabla_\theta \widetilde{\alpha}_t(\theta)$ *is*

$$\widehat{\nabla_\theta \widetilde{\alpha}_t}(\theta) \doteq \frac{1}{I} \sum_{i=1}^{I} \alpha_t(\widehat{z}_i) \nabla_\theta \log\left(p(\widehat{z}_i;\theta)\right), \tag{3.9}$$

where $\widehat{z}_i \overset{i.i.d.}{\sim} p(z;\theta)$ *and* $p\left(\widehat{z}_i;\theta\right) > 0$.

The acquisition function $\alpha_t\left(\widehat{z}_i\right)$ in (3.9) is approximated by (3.8). In this way, the stochastic gradient ascent method can be performed to optimize PR-M-UCB with multiple starts. The detailed procedure of the sequential evaluation step with PR-M-UCB is in **Algorithm 3.2**.

Algorithm 3.2 Detailed Procedure of Sequential Evaluation Step with PR-M-UCB

Require: The feasible set of prompts $\mathcal{Z} = \{z_1, \ldots, z_N\}$, a parametric model $\boldsymbol{f}(\cdot\,; \boldsymbol{W})$, a prior distribution of unknown parameters $\pi(\boldsymbol{W})$, the total budget of evaluating prompts T, the collected observed scores after the warm-up stage $\mathcal{S}_{T_W}$, the estimated evaluation uncertainty $\sigma_n^2, n \in \{1, 2, \ldots, N\}$, the number of iterations $\mathfrak{T}$ for gradient ascent, the sequence of learning rates $\ell_{\mathfrak{t}}, \mathfrak{t} \in \{1, 2, \ldots, \mathfrak{T}\}$, and the number of the starting points $\mathfrak{M}$ for gradient ascent.

Ensure: The selected soft prompt $\widehat{z}^*$.

Let $t = T_W$.
while $t < T$ **do**
  Update the posterior distribution $p\left(\boldsymbol{W} \mid \mathcal{S}_t\right)$ as in (3.5).
  Sample $\widehat{W}_k \sim p\left(\boldsymbol{W} \mid \mathcal{S}_t\right)$ for $k \in \{1, 2, \ldots, K\}$.
  Let $\mathfrak{t} = 0$ and uniformly select $\theta_{\mathfrak{m}}^{(0)} \in \Theta$ in random for $\mathfrak{m} = \{1, 2, \ldots, \mathfrak{M}\}$.
  while $\mathfrak{t} < \mathfrak{T}$ **do**
    for $\mathfrak{m} = \{1, 2, \ldots, \mathfrak{M}\}$ **do**
      for $i = 1, 2, \ldots, I$ **do**
        Sample $\widehat{z}_i \sim p\left(z; \theta_{\mathfrak{m}}^{(\mathfrak{t})}\right)$.
        Estimate $a_{\mathfrak{t},\mathfrak{m},i} \doteq \widehat{\alpha}_t\left(\widehat{z}_i\right)$ with $\left\{\boldsymbol{f}\left(\widehat{z}_i; \widehat{\boldsymbol{W}}_1\right), \boldsymbol{f}\left(\widehat{z}_i; \widehat{\boldsymbol{W}}_2\right), \ldots, \boldsymbol{f}\left(\widehat{z}_i; \widehat{\boldsymbol{W}}_K\right)\right\}$ as in (3.8).
      end for
      Calculate the estimated gradient $\eta = \frac{1}{I}\sum_{i=1}^{I} a_{\mathfrak{t},\mathfrak{m},i} \nabla_\theta \log\left(p\left(\widehat{z}_i; \theta\right)\right)$.
      Let $\theta_{\mathfrak{m}}^{(\mathfrak{t}+1)} = \theta_{\mathfrak{m}}^{(\mathfrak{t})} + \ell_{\mathfrak{t}}\eta$.
    end for
    Let $\mathfrak{t} \to \mathfrak{t} + 1$.
  end while
  Select $\mathfrak{m}^* = \arg\max_{\mathfrak{m} \in \{1,2,\ldots,\mathfrak{M}\}} \sum_{i=1}^{I} a_{\mathfrak{T},\mathfrak{m},i}$.
  Let $\theta_t^* = \theta_{\mathfrak{m}^*}^{\mathfrak{T}}$.
  Sample $\widehat{z}^{(t+1)} \sim p\left(z; \theta_t^*\right)$.
  Observe $\widehat{v}^{(t+1)}$ with $\widehat{z}^{(t+1)}$ as in (3.4).
  Update the historical dataset $\mathcal{S}_{t+1} = \left\{\left(\widehat{z}^{(t+1)}, \widehat{v}^{(t+1)}\right)\right\} \cup \mathcal{S}_t$.
  Let $t \to t + 1$.
end while
return $n^* = \arg\max_{n \in 1,2,\ldots,N} \frac{1}{r_n(T)} \sum_{m=1}^{r_n(T)} \widehat{v}_{n,m}$ and $\widehat{z}^* = z_{n^*}$.

3.5 Refinement

In this section, we provide a procedure to refine the proposed two-stage framework with the observed scores after completing the evaluation and selection stage. Specifically, we first refine the projection mapping from the latent space $\tilde{\mathcal{X}} \subseteq \mathbb{R}^{\tilde{D}}$. Recall that each soft prompt $z_n \in \mathcal{Z} \subseteq \mathbb{R}^D$ is attained by projecting a latent vector X_n in the latent vector set $\mathcal{X}$ to $z_n = AX_n$, where A is the projection matrix attained through principal component analysis, as described in Section 3.3. That is, without prior knowledge, we assume that the projection is linear. We then refine the projection mapping based on the scores observed during the evaluation and selection stage, introducing a non-linear mapping $\tilde{A}^*(\cdot) : \mathbb{R}^{\tilde{D}} \to \mathbb{R}^{D^*}$. This enhances the flexibility of the projection from the latent space. The dimension of the space after this refined projection, denoted as D^*, is not necessarily equivalent to the dimension of the soft prompt space D before the refinement.

We introduce a *projection stochastic kriging* (PSK) model to facilitate the construction of the nonlinear projection mapping $\tilde{A}^*(X)$. We assume that the observed score regarding the latent vector is

$$\widehat{v}_m(X) = v(X) + \epsilon_m(X), X \in \tilde{\mathcal{X}}, \tag{3.10}$$

where

$$v(X) \sim \mathcal{GP}\left(0, \tilde{A}^*(X)^\top \tilde{A}^*(X')\right). \tag{3.11}$$

Here, $\widehat{v}_m(X)$ denotes a score that will be observed when evaluating the human-readable prompt $\mathfrak{prom}(X) \doteq \mathrm{Dec}(X)$. That is, given a vector in the latent space $X \in \tilde{\mathcal{X}}$, the decoder model described in Section 3.3 will output a human-readable text, serving as a prompt. In addition, $v(X)$ is the mean score of $\mathfrak{prom}(X)$; and $\epsilon_m(X) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_\epsilon^2(X))$ is the uncertainty of the scores to be observed when evaluating $\mathfrak{prom}(X)$. Furthermore, the mean score regarding the latent vector is a realization from a zero-mean Gaussian process, and the kernel function $\boldsymbol{K}(X, X') = \tilde{A}^*(X)^\top \tilde{A}^*(X')$ is defined on the latent space $\tilde{\mathcal{X}}$.

We now describe the preparation for learning the projection mapping from the observed scores. Recall that the set of observed scores upon completing the evaluation and selection stage is $\mathcal{S}_T = \{(z_n, \widehat{v}_{n,m})\}$ for $m \in \{1, 2, \ldots, r_n(T)\}$ and $n \in \{1, 2, \ldots, N\}$. Here $z_n \in \mathcal{Z}$ denotes the soft prompt, $\widehat{v}_{n,m}$ denotes the m-th observed score when evaluating the prompt associated with z_n, and $r_n(T)$ is the number of evaluations at z_n when the total budget T is reached. Without loss of generality, we assume that $r_n(T) \geqslant 1$ for $n \in \{1, 2, \ldots, N\}$. Each soft prompt z_n has an associated latent vector $X_n \in \mathcal{X} \subseteq \tilde{\mathcal{X}}$. In the remaining part, we consider the set of the observed scores as $\mathcal{S}_T = \{(X_n, \widehat{v}_{n,m})\}$. That is, the soft prompts are replaced with the corresponding latent vectors. Regarding the observation uncertainty, we attain the sample variance for each X_n and then construct a predictive model to estimate $\sigma_\epsilon^2(X), X \in \tilde{\mathcal{X}}$, following the method described in Section 3.4.

Let us now consider the learning procedure of the projection mapping $\tilde{A}^*$ through the lens of the PSK model. Note that the kernel function of PSK is the inner product between the projections $\tilde{A}^*(X)$ and $\tilde{A}^*(X')$. Suppose that the refined projection mapping $\tilde{A}^*(\cdot)$ is selected from a known class of mappings $\tilde{\mathcal{A}}$, such as a neural network model with fixed

layers of nodes and unknown weight parameters, and the learning of the refined projection mapping is through

$$\tilde{A}^* = \arg\max_{\tilde{A}\in\tilde{\mathcal{A}}} \ell\left(\mathcal{S}_T; \tilde{A}\right),$$

where

$$\ell\left(\mathcal{S}_T; \tilde{A}\right) = -\frac{1}{2}\ln\left[\left|\boldsymbol{K}_N\left(\tilde{A}\right) + \boldsymbol{\Sigma}_{\varepsilon;N}\right|\right] - \frac{1}{2}\bar{\boldsymbol{V}}_N^\top\left[\boldsymbol{K}_N\left(\tilde{A}\right) + \boldsymbol{\Sigma}_{\varepsilon;N}\right]^{-1}\bar{\boldsymbol{V}}_N \tag{3.12}$$

is the normalized log-likelihood function of an N-dimensional Gaussian distribution. Here $\boldsymbol{K}_N\left(\tilde{A}\right) \in \mathbb{R}^{N\times N}$ is the kernel matrix, of which the (i,j)-th entry is $\tilde{A}\left(X_i\right)^\top \tilde{A}\left(X_j\right)$; $\boldsymbol{\Sigma}_{\epsilon;N} = \operatorname{diag}\left\{\sigma_\epsilon^2\left(X_1\right)/r_1(T), \ldots, \sigma_\epsilon^2\left(X_N\right)/r_N(T)\right\}$ denotes the observation uncertainty matrix; and $\boldsymbol{V}_N = \left(\bar{v}_1, \ldots, \bar{v}_N\right)^\top$ is the mean vector of the observed scores, where $\bar{v}_n = \frac{1}{r_n(T)}\sum_{m=1}^{r_n(T)} \widehat{v}_{n,m}$. In this way, the refined projection matrix is learned by maximizing the log-likelihood function through the kernel matrix $\boldsymbol{K}_N\left(\tilde{A}\right)$. We note that, instead of a pre-specified kernel function, our PSK model learns a data-driven kernel function using observed scores, thereby enjoying a higher approximation accuracy for the mean score v. Regarding the dimension of the refined soft prompt space D^*, we choose it using the cross-validation method [160]. That is, we first determine a set of potential D^*'s, denoted by $\mathcal{D}^* = \left\{D_1^*, \ldots, D_q^*\right\}$. For each $D_l^* \in \mathcal{D}^*$, we randomly select a proportion of the observations $(X_n, \widehat{v}_{n,m}) \in \mathcal{S}_T$, say 70%, to construct a PSK model with dimension D_l^*. We then use the remaining 30% of the observations to evaluate the prediction accuracy for the constructed PSK model [17]. The selected D^* is the one that achieves the highest prediction accuracy among those in $\mathcal{D}^*$.

In addition to the refinement of the projection, the PSK model can also be used to select latent vectors $X \in \tilde{\mathcal{X}}$ when we have additional budgets to evaluate the prompt after the evaluation and selection stage. Without loss of generality, we assume that $I \geqslant 0$ additional evaluations have been observed associated with different latent vectors, say $\left\{\left(\tilde{X}_1, \tilde{v}_1\right), \ldots, \left(\tilde{X}_I, \tilde{v}_I\right)\right\}_{i=1}^I$. Here $\tilde{X}_i \notin \mathcal{X}$ is an additional latent vector different from those in the latent vector set decided during the search stage in Section 3.3, and $\tilde{v}_i = v\left(\tilde{X}_i\right) + \epsilon\left(\tilde{X}_i\right)$ is the observed score associated with $\tilde{X}_i$. We denote the set of additional latent vectors selected to evaluate during the refinement procedure as $\mathcal{R} = \left\{\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_I\right\}$. Based on (3.10) and (3.11), for any latent vector in the latent space $\forall X \in \tilde{\mathcal{X}}$, we have a PSK predictor for the mean score $v\left(X\right)$ as

$$\widehat{\mu}\left(X\right) = \left(\boldsymbol{K}_N\left(X\right)^\top, \tilde{\boldsymbol{K}}_I\left(X\right)^\top\right)\left(\begin{pmatrix}\boldsymbol{K}_N & \tilde{\boldsymbol{K}}_{N,I}\\ \tilde{\boldsymbol{K}}_{N,I}^\top & \tilde{\boldsymbol{K}}_I\end{pmatrix} + \boldsymbol{\Sigma}_{\varepsilon;N+I}\right)^{-1}\begin{pmatrix}\bar{\boldsymbol{V}}_N\\ \tilde{\boldsymbol{V}}_I\end{pmatrix}, \tag{3.13}$$

with the associated prediction uncertainty quantified by

$$\widehat{\sigma^2}\left(X\right) = \tilde{A}^*\left(X\right)^\top \tilde{A}^*\left(X\right) - \left(\boldsymbol{K}_N\left(X\right)^\top, \tilde{\boldsymbol{K}}_I\left(X\right)^\top\right)\left(\begin{pmatrix}\boldsymbol{K}_N & \tilde{\boldsymbol{K}}_{N,I}\\ \tilde{\boldsymbol{K}}_{N,I}^\top & \tilde{\boldsymbol{K}}_I\end{pmatrix} + \boldsymbol{\Sigma}_{\varepsilon;N+I}\right)^{-1}\begin{pmatrix}\boldsymbol{K}_N\left(X\right)\\ \tilde{\boldsymbol{K}}_I\left(X\right)\end{pmatrix}. \tag{3.14}$$

Here

$$\begin{aligned}
&\boldsymbol{K}_N(X) \in \mathbb{R}^N \text{ with } [\boldsymbol{K}_N(X)]_i = \tilde{A}^*(X)^\top \tilde{A}^*(X_i) \text{ for } X_i \in \mathcal{X};\\
&\tilde{\boldsymbol{K}}_I(X) \in \mathbb{R}^I \text{ with } [\tilde{\boldsymbol{K}}_I(X)]_i = \tilde{A}^*(X)^\top \tilde{A}^*(\tilde{X}_i) \text{ for } \tilde{X}_i \in \mathcal{R};\\
&\boldsymbol{K}_N \in \mathbb{R}^{N\times N} \text{ with } [\boldsymbol{K}_N]_{i,j} = \tilde{A}^*(X_i)^\top \tilde{A}^*(X_j) \text{ for } X_i, X_j \in \mathcal{X};\\
&\tilde{\boldsymbol{K}}_I \in \mathbb{R}^{I\times I} \text{ with } [\tilde{\boldsymbol{K}}_I]_{i,j} = \tilde{A}^*(\tilde{X}_i)^\top \tilde{A}^*(\tilde{X}_j) \text{ for } \tilde{X}_i, \tilde{X}_j \in \mathcal{R};\\
&\tilde{\boldsymbol{K}}_{N,I} \in \mathbb{R}^{N\times I} \text{ with } [\tilde{\boldsymbol{K}}_{N,I}]_{i,j} = \tilde{A}^*(X_i)^\top \tilde{A}^*(\tilde{X}_j) \text{ for } X_i \in \mathcal{X} \text{ and } \tilde{X}_j \in \mathcal{R}.
\end{aligned}$$

Also, $\boldsymbol{\Sigma}_{\varepsilon;N+I} = \text{Diag}\left\{\sigma_\epsilon^2(X_1)/r_1(T), \ldots, \sigma_\epsilon^2(X_N)/r_N(T), \sigma_\epsilon^2\left(\tilde{X}_1\right), \ldots, \sigma_\epsilon^2\left(\tilde{X}_I\right)\right\}$ denotes the observation uncertainty matrix; $\boldsymbol{V}_N = (\bar{v}_1, \ldots, \bar{v}_N)^\top$ is the mean vector of the observed scores as defined in (3.12); and $\tilde{\boldsymbol{V}}_I = (\tilde{v}_1, \ldots, \tilde{v}_I)^\top$ denotes the observation vector of the additional evaluations.

In this way, the refinement with the PSK model enables an explicit mean score predictor and the prediction uncertainty $\forall X \in \tilde{\mathcal{X}}$ extending beyond the finite set $\mathcal{X}$ determined in the search stage. Consequently, classical simulation experimental designs and simulation optimization methods, particularly those based on the Gaussian process model, can be directly applied to the latent space $\tilde{\mathcal{X}} \subseteq \mathbb{R}^D$. For instance, with an explicit Gaussian process representation for the latent vector $X \in \tilde{\mathcal{X}}$, Bayesian optimization algorithms or Gaussian process search algorithms [171] can be employed. These methods aim to identify prompts that may achieve higher mean scores than the current optimal prompt selected at the end of the evaluation and selection stage. We note that this search process can extend beyond the previously selected latent vector set $\mathcal{X} = \{X_1, \ldots, X_N\}$. Additionally, the proposed PSK model can quantify the uncertainty of observed scores when the input latent vector $\widehat{X}$ to the PSK model is itself a random vector. For detailed procedures on employing the stochastic kriging model for input uncertainty analysis, we refer to [182, 16]. In general, suppose we select a set of $\mathcal{R} = \left\{\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_I\right\}$ as the additional latent vectors to evaluate with $I \geqslant 2$, and $\tilde{X}_i \neq \tilde{X}_j$ for $i \neq j$. We consider the *cumulative uncertainty*

$$U_I = \sum_{i=1}^{I} \widehat{\sigma^2}\left(\tilde{X}_i\right), \tag{3.15}$$

where $\widehat{\sigma^2}\left(\tilde{X}_i\right)$ denotes the uncertainty (3.14) associated with the PSK predictor (3.13). To provide the upper bound of the cumulative uncertainty, we make the following assumptions.

Assumption 3.2.

1. *The norm of the refined projection mapping is finite, that is,* $\sup_{X\in\tilde{\mathcal{X}}} \left\|\tilde{A}^*(X)\right\| < \infty$.
2. *The uncertainty in observed scores satisfies* $\sigma_\epsilon^2(X) \in \left(\underline{\sigma_\epsilon^2}, \overline{\sigma_\epsilon^2}\right), \forall X \in \tilde{\mathcal{X}}$, *where* $0 < \underline{\sigma_\epsilon^2}, \overline{\sigma_\epsilon^2} < \infty$.

Theorem 3.3. *Under Assumption 3.2, the cumulative uncertainty of the PSK predictor*

$$U_I \leqslant CD^* \log I.$$

Here U_I is the cumulative uncertainty defined in (3.15); D^* is the dimension of the refined projected space; I denotes the number of additional evaluations in the refinement stage; and C is a constant that does not depend on D^* or I.

Theorem 3.3 establishes that the cumulative uncertainty increases logarithmically in I. Specifically, the mean uncertainty, represented as $\bar{U}_I = U_I/I$, shrinks to zero as I approaches infinity. Moreover, U_I increases in the dimension of the refined projected space D^* rather than the dimension of the original latent space D. In essence, the projection mapping $\tilde{A}^*(\cdot)$ reduces the uncertainty in approximating the mean score v.

Given that the PSK model approximates the mean score from the latent space $\mathcal{X}$, an alternative strategy for the prompt selection problem is employing the PSK model as a surrogate to facilitate sequential evaluation in selecting prompts within the latent space, rather than using the soft prompt set. In Section 3.6, we conduct numerical experiments to compare our proposed two-stage framework with the approach that evaluates and selects the prompt using the PSK model. The experimental results demonstrate the superiority of our two-stage framework over the approach that directly relies on the PSK model.

3.6 Experiments

In this section, we present the numerical experiments of our proposed framework for selecting prompts. Regarding the generative language models for evaluating the prompt selection procedure, we select 1) GPT-3.5-turbo and 2) text-davinci-003; details about the background of these two generative language models can be found at `https://platform.openai.com/docs/models`. We implement our prompt selection procedure to select the prompt for language models to accomplish following tasks: 1) Word sorting: Given a list of words as input, output the sorted list in alphabetical order; 2) Rhymes: Given a word A and a list of words as input, output the word from the list that rhymes with A; 3) First letter: Given a word as input, output its first letter; 4) Largest animals: Given a list of animals as input, identify the largest animal; 5) Nums to verbal: Given a numerical number as input, output its English word form; 6) Count objectives: Given a list of objectives as input, output the number of objectives in the list.

Recall that, in the first search stage of our proposed prompt selection procedure, the prompt in text form is first transformed into a latent vector $X_n \in \mathbb{R}^{\tilde{D}}$ and then a soft prompt $z_n \in \mathbb{R}^D$. In our experiments, the dimension of the latent vector is $\tilde{D} = 3584$, and the dimension of the soft prompt is $D = 50$. Upon the completion of the search stage, we have a soft prompt set $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$ which contains finite but sufficient vectors, serving as the feasible set of the prompt selection problem. In the subsequent evaluation and selection stage, we sequentially select prompts in the soft prompt set to evaluate. The sample size of

the soft prompt set and the total budget for evaluation will be specified in each experiment. Our experiments were conducted with Pytorch and Python 3.9 on a computer equipped with two AMD Ryzen Threadripper 3970X 32-Core Processors, 256 GB memory, and two Nvidia GeForce RTX 3090 GPUs with 24GB of RAM each. The experimental results presented here are mean performances based on 15 repetitions, with standard deviations represented by a shadow on the mean-value line.

We evaluate each step of the proposed prompt selection procedure in our experiments. Specifically, in Section 3.4, we propose to use the Bayesian parametric model as the surrogate model to approximate the dependence of the mean score on the soft prompt. We then select three Bayesian parametric models and compare their approximation accuracy in Section 3.6. In Section 3.4, we propose two acquisition functions 1) modified upper confidence bound (M-UCB) and 2) probabilistic reparametrization modified upper confidence bound (PR-M-UCB) to sequentially select the prompt to evaluate. We compare the optimization procedure associated with the acquisition functions in Section 3.6. In Section 3.5, we also propose a projection stochastic kriging (PSK) model to refine the proposed two-stage framework with the observed scores after completing the evaluation and selection stage. Then in Section 3.6, we compare our two-stage framework with the direct search using the PSK model.

Surrogate Model Comparison

In this section, we conduct numerical experiments to compare the approximation accuracy of surrogate models on approximating the mean score with respect to soft prompts. Specifically, we choose 1) a Bayesian linear regression model (BLR), 2) a Gaussian process (GP) with a finite-rank kernel, 3) a Bayesian neural network (BNN), and 4) a variational autoencoder (VAE) as the surrogate model. We postpone the detailed descriptions of these surrogate models to the supplements. These selected surrogate models are all represented by a general parametric form $\boldsymbol{f}(z;\boldsymbol{W})$, where z is the soft prompt and $\boldsymbol{W} \subseteq \mathbb{R}^D$ is the unknown parameter of the surrogate model assigned with a prior distribution $\boldsymbol{W} \sim \pi(\boldsymbol{W})$. In this set of experiments, the prior distribution for each surrogate model is set as the independent standard normal distribution.

For each set of experiments, we first fix a soft prompt set $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$. We then randomly select $N_{\text{tr}} = 0.7N$ soft prompts from $\mathcal{Z}$ with equal probabilities, and denote the set of selected soft prompts as $\mathcal{Z}_{\text{tr}}$. For each $z_n \in \mathcal{Z}_{\text{tr}}$, we observe the scores as in (3.4) for 5 times. We denote the training set as $\mathcal{S}_{\text{tr}} = \{(z_n, \widehat{v}_{n,m})\}$ for $m \in \{1, 2, \ldots, 5\}$ for $z_n \in \mathcal{Z}_{\text{tr}}$, where $\widehat{v}_{n,m}$ denotes the m-th observation of the score evaluating with the soft prompt z_n. We let $\mathcal{Z}_{\text{test}} = \mathcal{Z} \setminus \mathcal{Z}_{\text{tr}}$. For each $z_n \in \mathcal{Z}_{\text{test}}$, we evaluate it and observe the scores for $\tilde{r} = 50$ times. We let $\mathcal{S}_{\text{test}} = \{(z_n, \bar{v}_n)\}$ for $z_n \in \mathcal{Z}_{\text{test}}$ and $\bar{v}_n = \frac{1}{\tilde{r}} \sum_{m=1}^{\tilde{r}} \widehat{v}_{n,m}$, the sample mean of the observed score. In each set of experiments, we use the same training set $\mathcal{S}_{\text{tr}}$ to calculate the posterior distribution of the unknown parameter $p(\boldsymbol{W} \mid \mathcal{S}_{\text{tr}})$ as in (3.5). We then generate $\left\{\widehat{\boldsymbol{W}}_1, \widehat{\boldsymbol{W}}_2, \ldots, \widehat{\boldsymbol{W}}_K\right\}$ from the posterior distribution $p(\boldsymbol{W} \mid \mathcal{S}_{\text{tr}})$, where $K = 100$. Regarding the comparison metric of the surrogate models' performances, we consider 1) root mean

squared error (RMSE) and 2) covering ratio (CR). That is

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{z_n \in \mathcal{Z}_{\text{test}}} \left(\widehat{\mu}\left(z_n\right) - \bar{v}_n\right)^2}, \qquad \text{CR} = \frac{1}{N_{\text{test}}} \sum_{z_n \in \mathcal{Z}_{\text{test}}} \mathbb{I}\left\{\bar{v}_n \in [\mathcal{L}_n, \mathcal{U}_n]\right\},$$

where $\widehat{\mu}\left(z_n\right) = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}_k\right)$ and $[\mathcal{L}_n, \mathcal{U}_n]$ is the 90%-sample quantile of

$$\left\{\boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}_k\right)\right\}_{k=1}^{K}.$$

We also consider the impact of the sample size on the surrogate models' performances, with $N \in \{50, 100, 200, 500\}$.

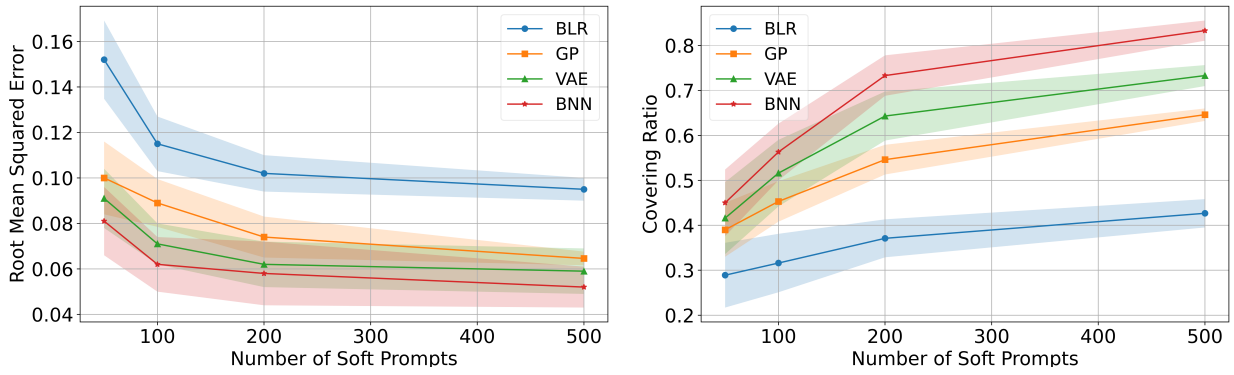

Figure 3.3: Experimental results for approximating the mean score with the soft prompts using different Bayesian parametric models. The task is word sorting and the generative language model is text-davinci-003.

The experiment results in **Figure 3.3** and **Figure 3.4** provide the following insights on the comparison among surrogate models. First, when there are not sufficient observations to construct the surrogate model ($N = 50$), the difference in the performances of surrogate models on MSE and CR is not significant. In general, these surrogate models all result in relatively high approximation error (indicated by high MSE) and biased approximation confidence (indicated by low CR). This illustrates that constructing a surrogate model for appropriate use requires sufficient observations. Second, as the number of observed scores increases, all the surrogate models achieve better approximation performances, indicated by lower MSE and higher CR. On the other hand, the improvement in the BLR model's performance is not as significant as in other models. That is, BLR cannot capture the dependence of the mean score on the soft prompt because of the lack of expressive power. Lastly, among all experiments, BNN achieves the more preferable performance with the

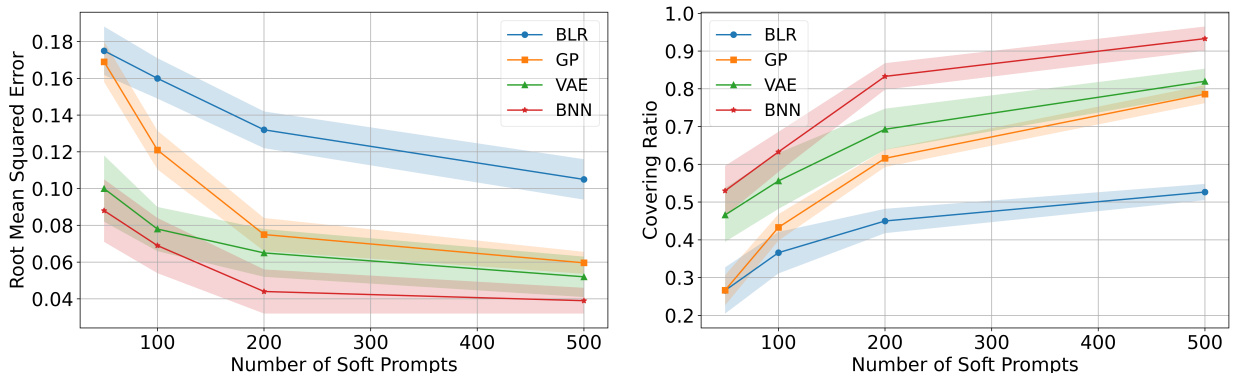

Figure 3.4: Experimental results for approximating the mean score with the soft prompts using different Bayesian parametric models. The task is word sorting and the generative language model is gpt-3.5-turbo.

lowest MSE and highest CR. This is due to BNN's greater flexibility and expressiveness compared to BLR and GP, and its strong approximation power to capture the highly non-structural dependence of the mean score on the soft prompt. On the other hand, VAE, as another deep learning model, does not perform as well as BNN in our experiments. This is because VAE has a much more complex model structure and generally requires more data (observations of scores in our experiments) to train. In our experiments, even when $N = 500$, the observations are not sufficient to train a VAE model. For these reasons, we conclude that BNN achieves the more preferable performance among the selected surrogate models for approximating the mean score of the soft prompt, and we select BNN as the surrogate model for approximating the mean score function in the following experiments.

Acquisition Function Optimization

In this section, we compare the two acquisition functions used in the sequential evaluation step in our framework: M-UCB and PR-M-UCB in terms of their performances on 1) mean score of the selected prompt and 2) implementation time. Recall that M-UCB is $\alpha_t(z_n) = \mu_t(z_n) + \beta_t(\sigma_t(z_n) + \gamma(r_n(t)))$, where $\mu_t(z_n)$ and $\sigma_t(z_n)$ are the posterior mean and standard deviation of the surrogate model at z_n, and $r_n(t)$ denotes the number of evaluations at z_n up to time t. In this set of experiments, we set $\beta_t = \sqrt{2\log(t)}$ and $\gamma(m) = 2m^{-1/2}$. Since maximizing M-UCB requires evaluating each $z_n \in \mathcal{Z}$, when the total number of soft prompts N is large, it is time-consuming to select the next soft prompt to evaluate. We also consider maximizing PR-M-UCB $\widetilde{\alpha}_t(\theta) \doteq \mathbb{E}_{p(z_n;\theta)}[\alpha_t(z_n)]$, where $p(z_n)$ is a categorical distribution as described in Section 3.4. Since it is a continuous optimization problem θ, we employ the method of gradient ascent with multiple starting points. Specifically, the algorithm using

the gradient descent method is represented by PR-M-UCB $(\mathfrak{M}, \mathfrak{T})$, where $\mathfrak{M}$ is the number of starting points and $\mathfrak{T}$ is the number of gradient ascent iterations.

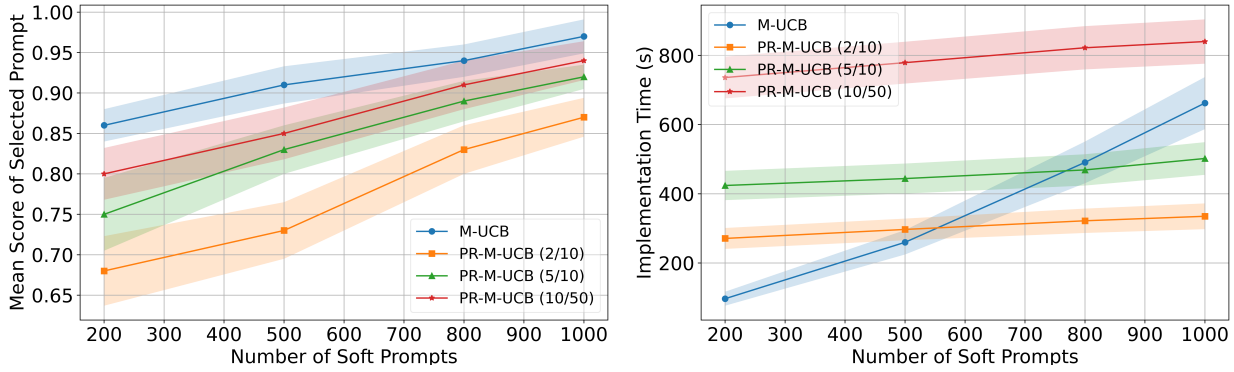

Figure 3.5: Experimental results for comparison between two acquisition functions: M-UCB and PR-M-UCB (number of starting points, number of gradient ascent iterations). The task is finding the largest animals given the names and the generative language model is gpt-3.5-turbo.

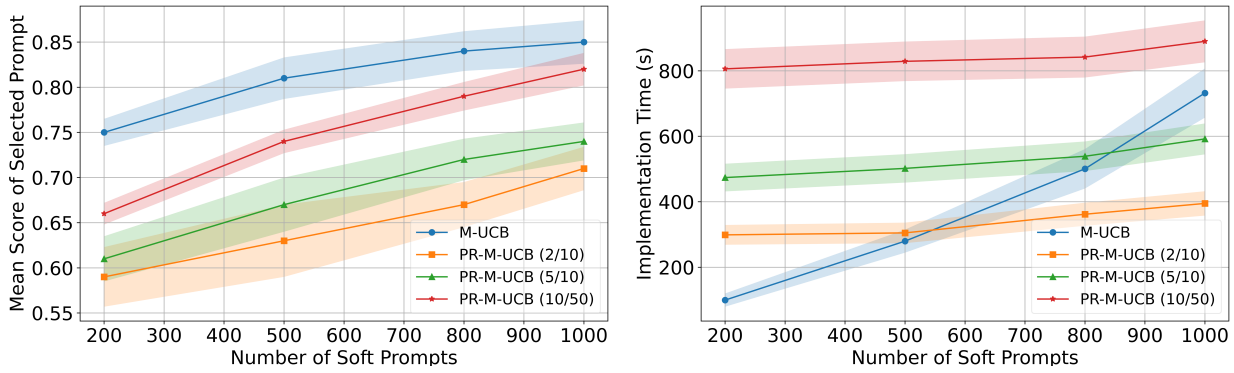

Figure 3.6: Experimental results for comparison between two acquisition functions: M-UCB and PR-M-UCB (number of starting points, number of gradient ascent iterations). The task is word sorting and the generative language model is text-davinci-003.

We consider different numbers of the soft prompts $N \in \{200, 500, 800, 1000\}$. The total budget is fixed as $T = 500$. We select BNN as the surrogate model and use variational inference (VI) to generate unknown parameters $\left\{\widehat{\boldsymbol{W}}_k\right\}_{k=1}^{100}$. In each set of experiments, we

randomly select $N_W = 0.05N$ soft prompts with equal probabilities and observe the score of each soft prompt 5 times in the warm-up step (Section 3.4). We then construct a BNN model using these scores. The sequential evaluation step, the procedures associated with M-UCB and PR-M-UCB employ the same BNN model constructed in the warm-up step. Regarding the comparing metric, we record the mean score of the selected prompt when the total budget is used up. The mean score of the selected prompt is approximated by additionally evaluating it for 50 times. We also record the implementation time of the methods during the sequential evaluation step, which includes the time for updating the surrogate model, generating unknown parameters from the posterior distribution, and optimizing the acquisition function.

The experiment results presented in **Figure 3.5** and **Figure 3.6** provide following insights. First, M-UCB achieves higher scores than PR-M-UCB across all sets of experiments, since M-UCB evaluates all soft prompts in each iteration. Second, the performance of PR-M-UCB improves as the number of iterations and starting points in the gradient ascent algorithm increase. Third, the implementation time of M-UCB increases almost linearly with the number of soft prompts. In contrast, the implementation time of PR-M-UCB is determined by the settings of the gradient ascent algorithm, specifically the number of iterations and starting points. As the number of soft prompts increases, the implementation time of PR-M-UCB does not increase significantly. Considering both the mean score of the selected prompt and implementation time, we recommend using M-UCB for a relatively small number of soft prompts, and switching to PR-M-UCB when the number of soft prompts is large.

Two-stage Framework v.s. Direct Search in Latent Space

Recall that a prompt in text form is transformed to a high-dimensional vector X in the latent space $\tilde{\mathcal{X}}$, and we propose in Section 3.5 a projection stochastic kriging (PSK) model that provides an approximation from the latent vector to the mean score. In this section, we conduct numerical experiments to compare our proposed two-stage framework with the approach that directly evaluates and selects latent vectors for the prompt selection problem.

We here provide details of the direct search procedure in the latent space using PSK. As one of Gaussian process models, the PSK model is compatible with classical simulation optimization methods. In this set of experiments, we employ the acquisition function expected improvement (EI) (see [77]), and regard every observation of the score as deterministic without noise. Specifically, in each iteration t, the latent vector to evaluate is selected by maximizing

$$\mathrm{EI}_t(X) = \left(\widehat{\mu}\left(X\right) - v_t^\star\right)\Phi\left(\frac{\widehat{\mu}\left(X\right) - v_t^\star}{\sqrt{\widehat{\sigma^2}\left(X\right)}}\right) + \sqrt{\widehat{\sigma^2}\left(X\right)}\varphi\left(\frac{\widehat{\mu}\left(X\right) - v_t^\star}{\sqrt{\widehat{\sigma^2}\left(X\right)}}\right). \tag{3.16}$$

Here Φ and ϕ are the cumulative distribution function and the probability density function of the standard normal distribution; $v_t^\star$ is the highest observed score up to time t; $\widehat{\mu}(X)$ and

$\widehat{\sigma^2}(X)$ denote the PSK predictor and the associated prediction uncertainty as described in (3.13) and (3.14). We denote the direct search in the latent space using the PSK model and the EI function as PSK-EI.

In comparison, we implement our proposed two-stage framework and employ the acquisition function M-UCB. We consider different total budgets as $T \in \{50, 200, 500, 1000\}$. In terms of the search stage, we construct a set of $N = 200$ soft prompts for all sets of experiments. In addition, we also consider the refinement procedure of our framework. That is, after the total budget is used up, we construct a PSK model using the observations and search within the latent space using the EI function (3.16) with 20 additional evaluations. We denote this method as M-UCB-r in the experimental results. Regarding PSK-EI and M-UCB, we record the mean score of the selected prompt when the total budget is used up. For M-UCB-r, we record the mean score when the additional evaluations are used up as well. The mean score of the selected prompt is approximated by additionally evaluating it more 50 times.

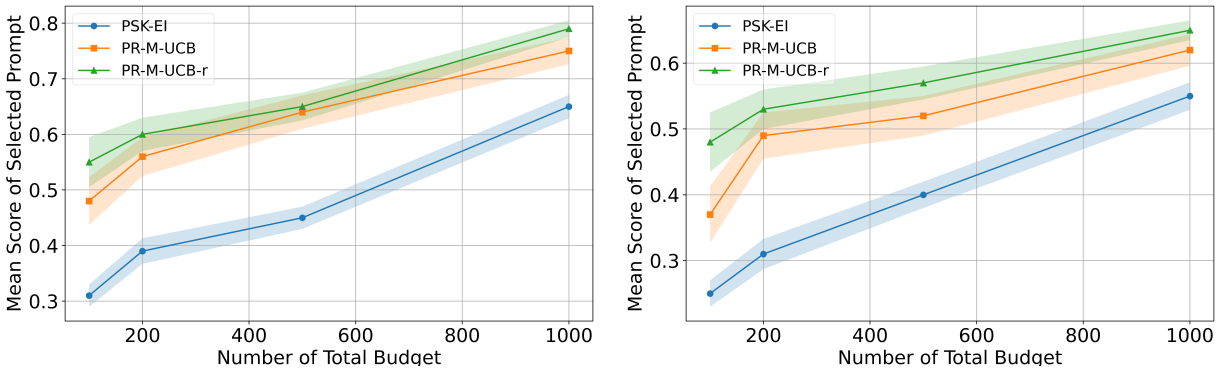

Figure 3.7: Experimental results for comparison between the proposed two-stage framework and the direct search in the latent space with PSK. The pair of task and generative language model includes 1) counting the objectives with gpt-3.5-turbo (left) and 2) finding the rhymes with text-davinci-003 (right).

The experimental results contained in **Figure 3.7** provide following insights. First, in all sets of experiments, our proposed two-stage framework achieves higher scores than direct search in the high-dimensional latent space using PSK. This is because the PSK model requires sufficient observations to construct an accurate surrogate model. During the sequential evaluation and selection without sufficient observations, the constructed PSK model often fails to provide a satisfactory approximation of the mean score for the latent vector, thereby not always selecting the prompts that are worth evaluating. Second, as the total budget increases, the improvement in direct search within the latent space is more significant than that of our two-stage framework. This is due to the increasingly accurate approximation

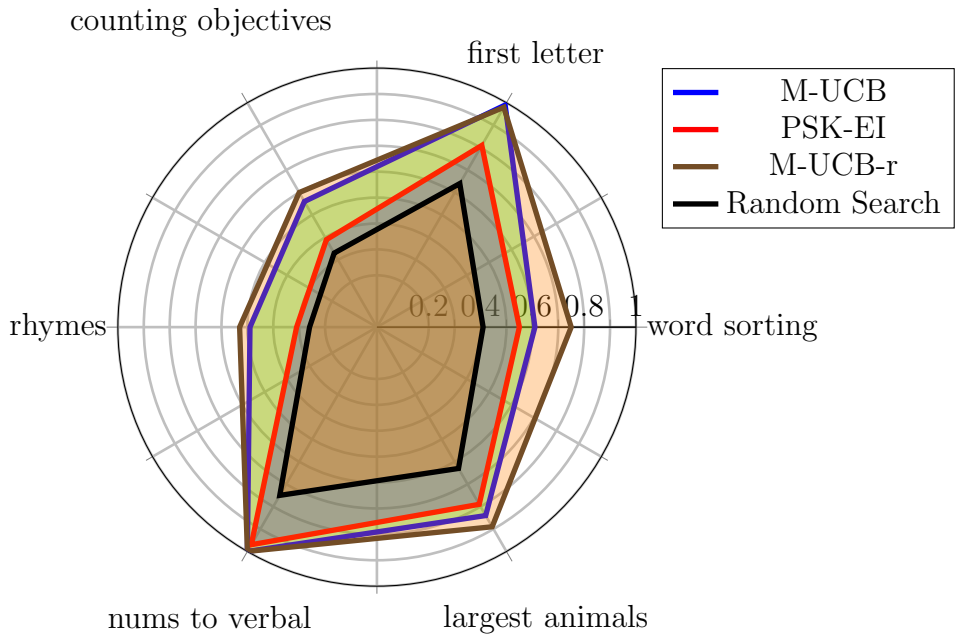

Figure 3.8: The mean score of the selected prompts in six tasks. The generative language model used to implement the tasks is gpt-3.5-turbo. The total budget is $T = 500$.

provided by PSK, which depends on the number of observations. In comparison, since our two-stage framework is restricted to a finite set of constructed soft prompts, its improvement is relatively limited. Lastly, our two-stage framework benefits from the refinement step with additional evaluations. That is, the score of the selected soft prompt is enhanced after refinement using PSK. Therefore, when we have sufficient budgets to evaluate the prompt, leave some additional budget for a refinement procedure after the two-stage framework leads to a higher-score prompt. The allocation of the total budget to 1) the budget for the two-stage framework and 2) the budget for additional evaluations after the two-stage framework will be discussed in future work.

At the end of this section, we provide a summary result of comparison between our two-stage framework and direct search in the latent space in **Figure 3.8**, which includes all six tasks implemented by the generative language model gpt-3.5-turbo. We also include the mean score of the random search as a baseline procedure. In each iteration, the random search procedure randomly selects a soft prompt from the constructed soft prompt set $z_n \in \mathcal{Z}$ to observe the score with equal probabilities. Both our proposed two-stage framework for prompt selection and the direct search procedure outperform the random search procedure with higher mean scores over six tasks. The experimental results also indicate the superiority of the proposed two-stage framework over the direct search in the latent space, and the selected prompts that achieve the highest mean score are included in the supplements.

3.7 Conclusion

In this work, we facilitate the selection of language prompts for generative language models using simulation optimization methods. We conclude by pointing out limitations and some future work of this work. In the initial search stage, the latent space of the prompts is decided by the text autoencoder. Although the text autoencoder has achieved empirical success in practice, it lacks theoretical guarantees due to its complex structure. This limitation presents a challenge in proving theoretical results for the entire framework we propose. Additionally, in our experiments, we observe that the proposed refinement procedure enhances the performance of our two-stage framework when additional budgets are available for evaluating prompts. This raises the question of how to allocate the total budget between the evaluation and selection stage, and the refinement procedure. We defer this discussion to future work.

Chapter 4

Conclusion & Future Work

4.1 Conclusion

In this dissertation, we explored the intersection between stochastic simulation and machine learning (ML):

- In **Chapter 2**, we considered integrating the neural network (NN) and the Gaussian process (GP) to facilitate sequential decision-making. In each iteration, a contextual variable is observed, followed by the selection of a decision variable aimed at maximizing an unknown objective function that depends on both the contextual and decision variables. To approximate this unknown objective function, we proposed the neural network-accompanied Gaussian process (NN-AGP) model. The NN-AGP model uses NN to approximate the dependence of the unknown objective function on the contextual variables observed in each iteration, while GP is employed to model the mapping from the decision variables to the objective function. This integration allows the model to leverage both: 1) the high approximation power of NN for the contextual variables and 2) the uncertainty quantification of GP for the decision variables. Therefore, NN-AGP offers improved flexibility and accuracy compared to standard GP-based algorithms, particularly in scenarios where contextual variables exhibit intricate structures, such as time series or graph-structured data. Moreover, the uncertainty quantification from GP supports the construction of an acquisition function, which selects the decision variable in each iteration by balancing exploration (selecting decision variables with high approximation uncertainty) and exploitation (selecting points with high values of the approximated objective function).

- In **Chapter 3**, we formulated prompt selection as a simulation optimization problem, aiming to maximize a pre-defined score for the selected prompt. We introduced a two-stage framework. In the first stage, we generate a feasible set of prompts, each represented by a moderate-dimensional vector. In the second stage, we sequentially select and evaluate prompts by constructing a Bayesian parametric model to approx-

imate the score based on these vectors. We proved the consistency of our procedure and provide numerical experiments demonstrating the effectiveness of our framework, and offered practical guidance for implementation.

These two chapters illustrate the mutually beneficial relationship between stochastic simulation and ML: (1) ML enhances the flexibility, approximation accuracy, and computational efficiency of stochastic simulation model, while (2) stochastic simulation offers sample-efficient and cost-effective implementations for ML applications.

4.2 Future Work

I envision two directions for my future research. The first is to enhance the computational efficiency of large-scale supply chain digital twins using machine learning technologies, and the second is to continue to make language model applications accessible to small organizations and individuals with limited resources.

Large-scale Supply Chain Digital Twin with Machine Learning

Digital twins are virtual models that replicate real-world systems using data, allowing for simulation and optimization of the system's performance. My interest in digital twins stems from my internship at Amazon, where a large-scale simulation model of its supply chain serves as a digital twin to predict metrics like inventory and shipments. Beyond business applications, digital twins hold potential for sustainability by addressing the significant carbon emissions from supply chain operations. However, running simulations for large-scale supply chains is computationally expensive, posing challenges for timely analysis of what-if scenarios, such as changes in demand or lead time. Machine learning models like neural networks offer a way to efficiently approximate supply chain outcomes based on simulated data, though they struggle with capturing the physical relationships within supply chains and providing explainable results.

To address these limitations, my future work will focus on integrating ML with domain-specific knowledge of supply chain networks. By modeling the digital twin component by component, such as customers, fulfillment centers, and vendors, some processes (e.g., vendor shipments) can be represented by stochastic models, while more complex processes (e.g., fulfillment center orders) can be approximated using ML. This hybrid approach enhances simulation efficiency while preserving the explanatory features of the supply chain network. The integration of stochastic models and ML aligns with my previous work on combining these methods for improved system performance approximation [193, 198].

Language Models with Limited Resources

Language models have wide applications in operations research but are often financially and technically inaccessible to small organizations and individuals due to the high costs associated

with 1) data collection, 2) model training, and 3) content generation. My future research explores integrating stochastic methods to develop more efficient, scalable, and cost-effective strategies.

Synthetic Data Generation. Synthetic data generation using well-established AI models offers a promising solution to reduce the high costs of developing language models by bypassing the need for extensive manual data collection, allowing developers to generate diverse, complex datasets that enhance model generalization. However, challenges remain in generating high-quality, diversified synthetic data and integrating it effectively with real data during training. Techniques like importance sampling can focus synthetic data on rare, informative scenarios, while multi-level Monte Carlo simulation can generate data at varying granularity to balance quality and computational efficiency, which is in a similar spirit to [195].

Learning Small Language Models. Small language models (SLMs) are becoming increasingly popular for domain-specific applications due to their reduced computational requirements compared to large language models (LLMs), making them more efficient and accessible. Rather than learning from scratch, SLMs are often developed using techniques like model distillation, where smaller models inherit knowledge from larger, pre-trained models. This process is analogous to metamodeling in simulation, where simpler models approximate more complex ones. Future research could leverage metamodeling approaches to develop SLMs and frame the optimization of SLM structure and scale as a black-box optimization problem, building our my previous work [193, 189, 196].

Efficient Content Generation. The application of pre-trained language models for content generation requires substantial computational resources. I plan to focus on accelerating the content generation process through two approaches. The first approach is speculative sampling, which aims to speed up generation by utilizing a smaller, faster draft model to generate a preliminary sequence of tokens. These tokens are then verified by a target model. If the draft tokens are acceptable, they are used; otherwise, corrections are made by the target model. This method accelerates generation by allowing multiple tokens to be processed at once. However, open questions remain regarding the selection of the optimal draft model and the adaptive determination of the draft sequence length to balance content generation speed and accuracy.

The second approach is Retrieval-Augmented Generation (RAG), where a language model supplements its internal knowledge with external data retrieval. When a query requires additional information, relevant documents are fetched from external sources, improving the quality of the generated content. This process can be framed as a queuing system, where queries are either processed directly by the model or delayed for document retrieval. By analyzing system performance through queuing models, such as single or multi-server queues, we can enhance generation efficiency while maintaining content quality.

Bibliography

[1] Naoki Abe, Alan W Biermann, and Philip M Long. "Reinforcement learning with immediate rewards and linear hypotheses". In: *Algorithmica* 37.4 (2003), pp. 263–293.

[2] Ernesto P Adorio and U Diliman. "Mvf-multivariate test functions library in c for unconstrained global optimization". In: *Quezon City, Metro Manila, Philippines* (2005), pp. 100–104.

[3] Shipra Agrawal and Navin Goyal. "Thompson sampling for contextual bandits with linear payoffs". In: *International conference on machine learning*. PMLR. 2013, pp. 127–135.

[4] TP Imthias Ahamed, Vivek S Borkar, and S Juneja. "Adaptive importance sampling technique for Markov chains using stochastic approximation". In: *Operations Research* 54.3 (2006), pp. 489–504.

[5] Bruce Ankenman, Barry L Nelson, and Jeremy Staum. "Stochastic Kriging for Simulation Metamodeling". In: *Operations Research* 58.2 (2010), pp. 371–382.

[6] Bruce Ankenman, Barry L Nelson, and Jeremy Staum. "Stochastic kriging for simulation metamodeling". In: *2008 Winter simulation conference*. IEEE. 2008, pp. 362–370.

[7] Eric A Applegate et al. "Multi-objective ranking and selection: Optimal sampling laws and tractable approximations via SCORE". In: *Journal of Simulation* 14.1 (2020), pp. 21–40.

[8] Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*. Vol. 57. Springer, 2007.

[9] Baris Ata, J Michael Harrison, and Nian Si. "Drift control of high-dimensional rbm: A computational method based on neural networks". In: *arXiv:2309.11651* (2023).

[10] Baris Ata, J Michael Harrison, and Nian Si. "Singular Control of (Reflected) Brownian Motion: A Computational Method Suitable for Queueing Applications". In: *arXiv:2312.11823* (2023).

[11] Peter Auer. "Using confidence bounds for exploitation-exploration trade-offs". In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 397–422.

[12] François Bachoc. "Asymptotic analysis of maximum likelihood estimation of covariance parameters for Gaussian processes: an introduction with proofs". In: *Advances in Contemporary Statistics and Econometrics*. Springer, 2021, pp. 283–303.

[13] François Bachoc. "Asymptotic analysis of the role of spatial sampling for covariance parameter estimation of Gaussian processes". In: *Journal of Multivariate Analysis* 125 (2014), pp. 1–35.

[14] Yuanlu Bai et al. "Efficient Calibration of Multi-Agent Simulation Models from Output Series with Bayesian Optimization". In: *Proceedings of the Third ACM International Conference on AI in Finance*. 2022, pp. 437–445.

[15] Rojeena Bajrachrya and Haejoon Jung. "Contextual bandits approach for selecting the best channel in industry 4.0 network". In: *2021 International Conference on Information Networking (ICOIN)*. IEEE. 2021, pp. 13–16.

[16] Russell R Barton, Barry L Nelson, and Wei Xie. "Quantifying input uncertainty via simulation confidence intervals". In: *INFORMS journal on computing* 26.1 (2014), pp. 74–87.

[17] Leonardo S Bastos and Anthony O'Hagan. "Diagnostics for Gaussian process emulators". In: *Technometrics* 51.4 (2009), pp. 425–438.

[18] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics". In: *Machine Learning* (2021), pp. 1–35.

[19] Jose Blanchet, Karthyek Murthy, and Fan Zhang. "Optimal transport-based distributionally robust optimization: Structural properties and iterative schemes". In: *Mathematics of Operations Research* 47.2 (2022), pp. 1500–1529.

[20] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[21] Christian Bock et al. "A wasserstein subsequence kernel for time series". In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2019, pp. 964–969.

[22] Ilija Bogunovic and Andreas Krause. "Misspecified Gaussian process bandit optimization". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 3004–3015.

[23] Ilija Bogunovic, Andreas Krause, and Jonathan Scarlett. "Corruption-tolerant Gaussian process bandit optimization". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 1071–1081.

[24] Ilija Bogunovic, Jonathan Scarlett, and Volkan Cevher. "Time-varying Gaussian process bandit optimization". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 314–323.

[25] Edwin V Bonilla, Kian Chai, and Christopher Williams. "Multi-task Gaussian process prediction". In: *Advances in neural information processing systems* 20 (2007).

[26] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. "Survey on applications of multi-armed and contextual bandits". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2020, pp. 1–8.

[27] J Bowden et al. *Deep Kernel Bayesian Optimization*. Tech. rep. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2021.

[28] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[29] Rasmus Bro and Age K Smilde. "Principal component analysis". In: *Analytical methods* 6.9 (2014), pp. 2812–2831.

[30] Eric Brochu, Vlad M Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv:1012.2599* (2010).

[31] Xu Cai and Jonathan Scarlett. "On lower bounds for standard and robust Gaussian process bandit optimization". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1216–1226.

[32] Sait Cakmak, Enlu Zhou, and Siyang Gao. "Contextual ranking and selection with Gaussian processes". In: *2021 Winter Simulation Conference (WSC)*. IEEE. 2021, pp. 1–12.

[33] Sait Cakmak et al. "Bayesian optimization of risk measures". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20130–20141.

[34] Dhivya Chandrasekaran and Vijay Mago. "Evolution of semantic similarity—a survey". In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–37.

[35] Ian Char et al. "Offline contextual Bayesian optimization". In: *Advances in Neural Information Processing Systems* 32 (2019).

[36] Niladri Chatterji, Aldo Pacchiano, and Peter Bartlett. "Online learning with kernel losses". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 971–980.

[37] Lichang Chen et al. "InstructZero: Efficient Instruction Optimization for Black-Box Large Language Models". In: *arXiv:2306.03082* (2023).

[38] Minshuo Chen et al. "Distribution Approximation and Statistical Estimation Guarantees of Generative Adversarial Networks". In: *arXiv:2002.03938* (2020).

[39] Xi Chen, Bruce E Ankenman, and Barry L Nelson. "Enhancing Stochastic Kriging Metamodels with Gradient Estimators". In: *Operations Research* 61.2 (2013), pp. 512–528.

[40] Xi Chen, Bruce E Ankenman, and Barry L Nelson. "The Effects of Common Random Numbers on Stochastic Kriging Metamodels". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 22.2 (2012), pp. 1–20.

[41] Xinyun Chen, Yunan Liu, and Guiyu Hong. "An online learning approach to dynamic pricing and capacity sizing in service systems". In: *arXiv:2009.02911* (2020).

[42] Victor Chernozhukov et al. "Semi-parametric efficient policy learning with continuous actions". In: *Advances in Neural Information Processing Systems* 32 (2019).

[43] Aakanksha Chowdhery et al. "Palm: Scaling language modeling with pathways". In: *Journal of Machine Learning Research* 24.240 (2023), pp. 1–113.

[44] Sayak Ray Chowdhury and Aditya Gopalan. "On kernelized multi-armed bandits". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 844–853.

[45] Kenneth Ward Church. "Word2Vec". In: *Natural Language Engineering* 23.1 (2017), pp. 155–162.

[46] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[47] Andreas Damianou and Neil D Lawrence. "Deep Gaussian processes". In: *Artificial intelligence and statistics*. PMLR. 2013, pp. 207–215.

[48] Samuel Daulton et al. "Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 12760–12774.

[49] Samuel Daulton et al. "Robust Multi-Objective Bayesian Optimization Under Input Noise". In: *arXiv:2202.07549* (2022).

[50] L De, B De-Moor, and J Vandewalle. "On the Best Rank-1 and Rank-(R1 R2... RN) Approximation of Higher-order Tensors". In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1324–1342.

[51] Ivan Martins De Andrade and Cleonir Tumelero. "Increasing customer service efficiency through artificial intelligence chatbot". In: *Revista de Gestão* 29.3 (2022), pp. 238–251.

[52] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition". In: *SIAM journal on Matrix Analysis and Applications* 26.2 (2004), pp. 295–327.

[53] Yuntian Deng et al. "Weighted Gaussian Process Bandits for Non-stationary Environments". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 6909–6932.

[54] Anand Deo and Karthyek Murthy. "Achieving Efficiency in Black-Box Simulation of Distribution Tails with Self-Structuring Importance Samplers". In: *Operations Research* (2023).

[55] Thomas Desautels, Andreas Krause, and Joel W Burdick. "Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization". In: *Journal of Machine Learning Research* 15 (2014), pp. 3873–3923.

[56] Liang Ding, Rui Tuo, and Xiaowei Zhang. "High-Dimensional Simulation Optimization via Brownian Fields and Sparse Grids". In: *arXiv:2107.08595* (2021).

[57] Liang Ding and Xiaowei Zhang. "Sample and computationally efficient stochastic kriging in high dimensions". In: *Operations Research* (2022).

[58] Liang Ding et al. "Knowledge gradient for selection with covariates: Consistency and computation". In: *Naval Research Logistics (NRL)* 69.3 (2022), pp. 496–507.

[59] Josip Djolonga, Andreas Krause, and Volkan Cevher. "High-dimensional Gaussian process bandits". In: *Advances in neural information processing systems* 26 (2013).

[60] Jing Dong, M Ben Feng, and Barry L Nelson. "Unbiased metamodeling via likelihood ratios". In: *2018 Winter Simulation Conference (WSC)*. IEEE. 2018, pp. 1778–1789.

[61] Jing Dong and Yi Zhu. "Three asymptotic regimes for ranking and selection with general sample distributions". In: *2016 Winter Simulation Conference (WSC)*. IEEE. 2016, pp. 277–288.

[62] Audrey Durand et al. "Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis". In: *Machine learning for healthcare conference*. PMLR. 2018, pp. 67–82.

[63] David J Eckman, Shane G Henderson, and Sara Shashaani. "Diagnostic tools for evaluating and comparing simulation-optimization algorithms". In: *INFORMS Journal on Computing* 35.2 (2023), pp. 350–367.

[64] David J Eckman, Shane G Henderson, and Sara Shashaani. "SimOpt: A testbed for simulation-optimization experiments". In: *INFORMS Journal on Computing* 35.2 (2023), pp. 495–508.

[65] David J Eckman, Matthew Plumlee, and Barry L Nelson. "Plausible screening using functional properties for simulations with large solution spaces". In: *Operations Research* 70.6 (2022), pp. 3473–3489.

[66] David Eriksson and Matthias Poloczek. "Scalable constrained Bayesian optimization". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 730–738.

[67] David Eriksson et al. "Scalable global optimization via local Bayesian optimization". In: *Advances in neural information processing systems* 32 (2019).

[68] Qi Fan and Jiaqiao Hu. "Surrogate-based promising area search for Lipschitz continuous simulation optimization". In: *INFORMS Journal on Computing* 30.4 (2018), pp. 677–693.

[69] Weiwei Fan, L Jeff Hong, and Barry L Nelson. "Indifference-zone-free selection of the best". In: *Operations Research* 64.6 (2016), pp. 1499–1514.

[70] Weiwei Fan, L Jeff Hong, and Xiaowei Zhang. "Distributionally robust selection of the best". In: *Management Science* 66.1 (2020), pp. 190–208.

[71] Tristan Fauvel and Matthew Chalk. "Contextual Bayesian optimization with binary outputs". In: *arXiv:2111.03447* (2021).

[72] JC Ferreira and VA2501172 Menegatto. "Eigenvalues of integral operators defined by smooth positive definite kernels". In: *Integral Equations and Operator Theory* 64.1 (2009), pp. 61–81.

[73] Marcello Fiducioso et al. "Safe contextual Bayesian optimization for sustainable room temperature PID control tuning". In: *arXiv:1906.12086* (2019).

[74] Dylan J Foster et al. "Adapting to misspecification in contextual bandits". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11478–11489.

[75] Peter Frazier, Warren Powell, and Savas Dayanik. "The knowledge-gradient policy for correlated normal beliefs". In: *INFORMS journal on Computing* 21.4 (2009), pp. 599–613.

[76] Peter I Frazier. "A tutorial on Bayesian optimization". In: *arXiv:1807.02811* (2018).

[77] Peter I Frazier. "Bayesian optimization". In: *Recent advances in optimization and modeling of contemporary problems*. Informs, 2018, pp. 255–278.

[78] Michael C Fu et al. *Handbook of Simulation Optimization*. Vol. 216. New York: Springer, 2015.

[79] Michael C Fu et al. "Simulation allocation for determining the best design in the presence of correlated sampling". In: *INFORMS Journal on Computing* 19.1 (2007), pp. 101–111.

[80] David JH Garling. *Inequalities: a journey into linear analysis*. Cambridge University Press, 2007.

[81] Subhashis Ghosal and Anindya Roy. "Posterior consistency of Gaussian process prior for nonparametric binary regression". In: *The Annals of Statistics* 34.5 (2006), pp. 2413–2429.

[82] Louie Giray. "Prompt engineering with ChatGPT: a guide for academic writers". In: *Annals of biomedical engineering* 51.12 (2023), pp. 2629–2633.

[83] Mark Girolami and Ben Calderhead. "Riemann manifold langevin and hamiltonian monte carlo methods". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 73.2 (2011), pp. 123–214.

[84] Peter W Glynn and Mariana Olvera-Cravioto. "Likelihood ratio gradient estimation for steady-state parameters". In: *Stochastic Systems* 9.2 (2019), pp. 83–100.

[85] Michael B Gordy and Sandeep Juneja. "Nested simulation in portfolio risk measurement". In: *Management Science* 56.10 (2010), pp. 1833–1848.

[86] Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

[87] Wolfgang Hackbusch and Boris N Khoromskij. "Tensor-product approximation to operators and functions in high dimensions". In: *Journal of Complexity* 23.4-6 (2007), pp. 697–714.

[88] Donghai He, Stephen E Chick, and Chun-Hung Chen. "Opportunity cost and OCBA selection procedures in ordinal optimization for a fixed number of alternative systems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.5 (2007), pp. 951–961.

[89] Shengyi He et al. "Adaptive Importance Sampling for Efficient Stochastic Root Finding and Quantile Estimation". In: *Operations Research* (2023).

[90] James Hensman et al. "MCMC for variationally sparse Gaussian processes". In: *Advances in Neural Information Processing Systems* 28 (2015).

[91] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[92] Søren Højsgaard and Steffen L Lauritzen. "Graphical Gaussian models with edge and vertex symmetries". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.5 (2008), pp. 1005–1027.

[93] L Jeff Hong, Weiwei Fan, and Jun Luo. "Review on ranking and selection: A new perspective". In: *Frontiers of Engineering Management* 8.3 (2021), pp. 321–343.

[94] L Jeff Hong and Guangxin Jiang. "Offline simulation online application: A new framework of simulation-based decision making". In: *Asia-Pacific Journal of Operational Research* 36.06 (2019), p. 1940015.

[95] L Jeff Hong, Guangxin Jiang, and Ying Zhong. "Solving large-scale fixed-budget ranking and selection problems". In: *INFORMS Journal on Computing* 34.6 (2022), pp. 2930–2949.

[96] L Jeff Hong and Xiaowei Zhang. "Surrogate-based simulation optimization". In: *Tutorials in Operations Research: Emerging Optimization Methods and Modeling Techniques with Applications*. INFORMS, 2021, pp. 287–311.

[97] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[98] Eric Horvitz. *The Power of Prompting*. 2023. URL: `https://www.microsoft.com/en-us/research/blog/the-power-of-prompting/`.

[99] Chenyu Hou et al. "Prompt-based and Fine-tuned GPT Models for Context-Dependent and-Independent Deductive Coding in Social Annotation". In: *Proceedings of the 14th Learning Analytics and Knowledge Conference*. 2024, pp. 518–528.

[100] Shouri Hu et al. "Adjusted Expected Improvement for Cumulative Regret Minimization in Noisy Bayesian Optimization". In: *arXiv:2205.04901* (2022).

[101] Rouba Ibrahim and Ward Whitt. "Delay predictors for customer service systems with time-varying parameters". In: *Proceedings of the 2010 Winter Simulation Conference*. IEEE. 2010, pp. 2375–2386.

[102] Rouba Ibrahim et al. "On the modeling and forecasting of call center arrivals". In: *Proceedings of the 2012 Winter Simulation Conference (WSC)*. IEEE. 2012, pp. 1–12.

[103] Guangxin Jiang, L Jeff Hong, and Barry L Nelson. "Online risk monitoring using offline simulation". In: *INFORMS Journal on Computing* 32.2 (2020), pp. 356–375.

[104] Beth Kanter and Allison H Fine. *The Smart Nonprofit: Staying Human-centered in an Automated World*. John Wiley & Sons, Incorporated, 2022.

[105] Parnian Kassraie and Andreas Krause. "Neural contextual bandits without regret". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 240–278.

[106] Parnian Kassraie, Andreas Krause, and Ilija Bogunovic. "Graph Neural Network Bandits". In: *arXiv:2207.06456* (2022).

[107] Jaehyeon Kim, Jungil Kong, and Juhee Son. "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5530–5540.

[108] Diederik P Kingma, Max Welling, et al. "An introduction to variational autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[109] Paul DW Kirk and Michael PH Stumpf. "Gaussian process regression bootstrapping: exploring the effects of uncertainty in time course data". In: *Bioinformatics* 25.10 (2009), pp. 1300–1306.

[110] Johannes Kirschner et al. "Distributionally robust Bayesian optimization". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2174–2184.

[111] Andreas Krause and Cheng Ong. "Contextual Gaussian process bandit optimization". In: *Advances in neural information processing systems* 24 (2011).

[112] Peter L. Salemi et al. "Gaussian Markov Random Fields for Discrete Optimization via Simulation: Framework and Algorithms". In: *Operations Research* 67.1 (2019), pp. 250–266.

[113] Henry Lam and Fengpei Li. "General Feasibility Bounds for Sample Average Approximation via Vapnik–Chervonenkis Dimension". In: *SIAM Journal on Optimization* 32.2 (2022), pp. 1471–1497.

[114] John Langford and Tong Zhang. "The epoch-greedy algorithm for contextual multi-armed bandits". In: *Advances in neural information processing systems* 20.1 (2007), pp. 96–1.

[115] Jaehoon Lee et al. "Deep neural networks as Gaussian processes". In: *arXiv:1711.00165* (2017).

[116] Cheng Li, Siyang Gao, and Jianzhong Du. "Convergence Analysis of Stochastic Kriging-Assisted Simulation with Random Covariates". In: *INFORMS Journal on Computing* 35.2 (2023), pp. 386–402.

[117] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents". In: *arXiv:1506.01057* (2015).

[118] Lihong Li et al. "A contextual-bandit approach to personalized news article recommendation". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 661–670.

[119] Yanwen Li and Siyang Gao. "On the Finite-Time Performance of the Knowledge Gradient Algorithm". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 12741–12764.

[120] Zaile Li, Weiwei Fan, and L Jeff Hong. "The (Surprising) Sample Optimality of Greedy Procedures for Large-Scale Ranking and Selection". In: *arXiv:2303.02951* (2023).

[121] Zihan Li and Jonathan Scarlett. "Gaussian process bandit optimization with few batches". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 92–107.

[122] Yunduan Lin et al. "Nonprogressive Diffusion on Social Networks: Approximation and Applications". In: *Available at SSRN* (2022).

[123] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. "Remarks on multi-output Gaussian process regression". In: *Knowledge-Based Systems* 144 (2018), pp. 102–121.

[124] François V Louveaux and Rüdiger Schultz. "Stochastic integer programming". In: *Handbooks in operations research and management science* 10 (2003), pp. 213–266.

[125] Jun Luo et al. "Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments". In: *Operations Research* 63.5 (2015), pp. 1177–1194.

[126] Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: theory of majorization and its applications*. Vol. 143. Springer, 1979.

[127] Alexander G de G Matthews et al. "Gaussian process behaviour in wide deep neural networks". In: *arXiv:1804.11271* (2018).

[128] Jeffrey W Miller. "A detailed treatment of Doob's theorem". In: *arXiv:1801.03122* (2018).

[129] Thomas Minka. *Bayesian linear regression*. Tech. rep. Citeseer, 2000.

[130] Ivan Montero, Nikolaos Pappas, and Noah A Smith. "Sentence bottleneck autoencoders from transformer language models". In: *arXiv:2109.00055* (2021).

[131] Whitney K Newey. "Uniform convergence in probability and stochastic equicontinuity". In: *Econometrica: Journal of the Econometric Society* (1991), pp. 1161–1167.

[132] Trung V Nguyen, Edwin V Bonilla, et al. "Collaborative Multi-output Gaussian Processes." In: *UAI*. Citeseer. 2014, pp. 643–652.

[133] Eric C Ni et al. "Efficient ranking and selection in parallel computing environments". In: *Operations Research* 65.3 (2017), pp. 821–836.

[134] OpenAI. *Chatgpt*. `https://openai.com/blog/chatgpt`. 2023.

[135] Felix Opolka et al. "Adaptive Gaussian Processes on Graphs via Spectral Graph Wavelets". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 4818–4834.

[136] Michael Pearce and Juergen Branke. "Continuous multi-task Bayesian optimisation with correlation". In: *European Journal of Operational Research* 270.3 (2018), pp. 1074–1085.

[137] Linda Pei, Barry L Nelson, and Susan R Hunter. "Parallel adaptive survivor selection". In: *Operations Research* (2022).

[138] Yijie Peng et al. "A new likelihood ratio method for training artificial neural networks". In: *INFORMS Journal on Computing* 34.1 (2022), pp. 638–655.

[139] Yijie Peng et al. "Efficient simulation resource sharing and allocation for selecting the best". In: *IEEE Transactions on Automatic Control* 58.4 (2012), pp. 1017–1023.

[140] Reid Pryzant et al. "Automatic prompt optimization with" gradient descent" and beam search". In: *arXiv:2305.03495* (2023).

[141] Ning Quan et al. "Simulation optimization via kriging: a sequential search using expected improvement with computing budget constraints". In: *Iie Transactions* 45.7 (2013), pp. 763–780.

[142] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[143] Jeffrey N Rouder and Jun Lu. "An introduction to Bayesian hierarchical models with an application in the theory of signal detection". In: *Psychonomic bulletin & review* 12.4 (2005), pp. 573–604.

[144] Daniel Russo and Benjamin Van Roy. "Learning to optimize via posterior sampling". In: *Mathematics of Operations Research* 39.4 (2014), pp. 1221–1243.

[145] Daniel J Russo et al. "A tutorial on thompson sampling". In: *Foundations and Trends® in Machine Learning* 11.1 (2018), pp. 1–96.

[146] Ilya O Ryzhov. "On the convergence rates of expected improvement methods". In: *Operations Research* 64.6 (2016), pp. 1515–1528.

[147] Ilya O Ryzhov, Warren B Powell, and Peter I Frazier. "The knowledge gradient algorithm for a general class of online learning problems". In: *Operations Research* 60.1 (2012), pp. 180–195.

[148] Tara N Sainath et al. "Convolutional, long short-term memory, fully connected deep neural networks". In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 4580–4584.

[149] Franco Scarselli et al. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[150] Warren Scott, Peter Frazier, and Warren Powell. "The correlated knowledge gradient for simulation optimization of continuous parameters using Gaussian process regression". In: *SIAM Journal on Optimization* 21.3 (2011), pp. 996–1026.

[151] Mark Semelhago et al. "Rapid discrete optimization via simulation with Gaussian Markov random fields". In: *INFORMS Journal on Computing* 33.3 (2021), pp. 915–930.

[152] Haihui Shen, L Jeff Hong, and Xiaowei Zhang. "Enhancing stochastic kriging for queueing simulation with stylized models". In: *IISE Transactions* 50.11 (2018), pp. 943–958.

[153] Millicent Skiles. *AI for Nonprofits: How to Use Artificial Intelligence for Good*. `https://donorbox.org/nonprofit-blog/ai-for-nonprofits`. Accessed: 2023-12-27. Dec. 2023.

[154] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical Bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[155] Alexander Spangher et al. "Newsedits: A news article revision dataset and a novel document-level reasoning challenge". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2022, pp. 127–157.

[156] Niranjan Srinivas et al. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". In: *Proceedings of the 27th International Conference on Machine Learning*. Omnipress. 2010, pp. 1015–1022.

[157] Niranjan Srinivas et al. "Gaussian process optimization in the bandit setting: No regret and experimental design". In: *arXiv:0912.3995* (2009).

[158] Niranjan Srinivas et al. "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting". In: *IEEE transactions on information theory* 58.5 (2012), pp. 3250–3265.

[159] William T Stephenson et al. "Measuring the robustness of Gaussian processes to kernel choice". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 3308–3331.

[160] Mervyn Stone. "Cross-validatory choice and assessment of statistical predictions". In: *Journal of the royal statistical society: Series B (Methodological)* 36.2 (1974), pp. 111–133.

[161] Sebastian Shenghong Tay et al. "Efficient distributionally robust Bayesian optimization with worst-case sensitivity". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 21180–21204.

[162] Michalis Titsias. "Variational learning of inducing variables in sparse Gaussian processes". In: *Artificial intelligence and statistics*. PMLR. 2009, pp. 567–574.

[163] Kristina Toutanova et al. "A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs". In: *EMNLP*. 2016.

[164] Shing Chih Tsai et al. "Adaptive fully sequential selection procedures with linear and nonlinear control variates". In: *IISE Transactions* 55.6 (2023), pp. 561–573.

[165] Sattar Vakili, Kia Khezeli, and Victor Picheny. "On information gain and regret bounds in Gaussian process bandits". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 82–90.

[166] Aad W Van der Vaart. *Asymptotic statistics*. Vol. 3. Cambridge university press, 2000.

[167] Songhao Wang, Szu Hui Ng, and William Benjamin Haskell. "A multilevel simulation optimization approach for quantile functions". In: *INFORMS Journal on Computing* 34.1 (2022), pp. 569–585.

[168] Tan Wang and L Jeff Hong. "Large-Scale Inventory Optimization: A Recurrent Neural Networks–Inspired Simulation Approach". In: *INFORMS Journal on Computing* 35.1 (2023), pp. 196–215.

[169] Wenjing Wang and Xi Chen. "An adaptive two-stage dual metamodeling approach for stochastic simulation experiments". In: *IISE Transactions* 50.9 (2018), pp. 820–836.

[170] Wenyu Wang, Hong Wan, and Xi Chen. "Bonferroni-Free and Indifference-Zone-Flexible Sequential Elimination Procedures for Ranking and Selection". In: *Operations Research* (2023).

[171] Xiuxian Wang et al. "Gaussian Process-Based Random Search for Continuous Optimization via Simulation". In: *Operations Research* (2023).

[172] Jian Wei et al. "Collaborative filtering and deep learning based recommendation system for cold start items". In: *Expert Systems with Applications* 69 (2017), pp. 29–39.

[173] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[174] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[175] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8 (1992), pp. 229–256.

[176] Cameron R Wolfe and Anastasios Kyrillidis. "Cold Start Streaming Learning for Deep Networks". In: *arXiv:2211.04624* (2022).

[177] Di Wu, Yuhao Wang, and Enlu Zhou. "Data-driven Ranking and Selection under Input Uncertainty". In: *Operations Research* (2022).

[178] Di Wu, Helin Zhu, and Enlu Zhou. "A Bayesian risk approach to data-driven stochastic optimization: Formulations and asymptotics". In: *SIAM Journal on Optimization* 28.2 (2018), pp. 1588–1612.

[179] Jian Wu and Peter Frazier. "The parallel knowledge gradient method for batch Bayesian optimization". In: *Advances in neural information processing systems* 29 (2016).

[180] Jian Wu et al. "Bayesian optimization with gradients". In: *Advances in neural information processing systems* 30 (2017).

[181] George Wynne and Veit Wild. "Variational Gaussian Processes: A Functional Analysis View". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 4955–4971.

[182] Wei Xie, Barry L Nelson, and Russell R Barton. "A Bayesian Framework for Quantifying Uncertainty in Stochastic Simulation". In: *Operations Research* 62.6 (2014), pp. 1439–1452.

[183] Wei Xie, Bo Wang, and Pu Zhang. "Metamodel-Assisted Sensitivity Analysis for Controlling the Impact of Input Uncertainty". In: *2019 Winter Simulation Conference (WSC)*. 2019, pp. 3681–3692. DOI: 10.1109/WSC40007.2019.9004730.

[184] Wei Xie, Yuan Yi, and Hua Zheng. "Global-local Metamodel-assisted Stochastic Programming via Simulation". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 31.1 (2020), pp. 1–34.

[185] Pan Xu et al. "Neural contextual bandits with deep representation and shallow exploration". In: *arXiv:2012.01780* (2020).

[186] Yunbei Xu and Assaf Zeevi. "Upper counterfactual confidence bounds: a new optimism principle for contextual bandits". In: *arXiv:2007.07876* (2020).

[187] Dmitry Yarotsky. "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94 (2017), pp. 103–114.

[188] JD Zamfirescu-Pereira et al. "Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–21.

[189] Donglin Zhan et al. "Collaborative Bayesian Optimization via Wasserstein Barycenters". In: *submitted* ().

[190] Haoting Zhang and Jingxu Xu. "Robustify Simulation Uncertainty Quantification against Input Data Outlier". In: *submitted* ().

[191] Haoting Zhang and Zeyu Zheng. "Simulating Nonstationary Spatio-Temporal Poisson Processes using the Inversion Method". In: *2020 Winter Simulation Conference (WSC)*. IEEE. 2020, pp. 492–503.

[192] Haoting Zhang et al. "Clustering Then Estimation of Spatio-Temporal Self-Exciting Processes". In: *INFORMS Journal on Computing* (2024).

[193] Haoting Zhang et al. "Contextual Gaussian Process Bandits with Neural Networks". In: *Advances in Neural Information Processing Systems*. 2023.

[194] Haoting Zhang et al. "Daily Physical Activity Monitoring–Adaptive Learning from Multi-source Motion Sensor Data". In: *Conference on Health, Inference, and Learning (CHIL)*. 2024.

[195] Haoting Zhang et al. "Enhancing Language Model with Both Human and Artificial Intelligence Feedback Data". In: *2024 Winter Simulation Conference (WSC)*. 2024.

[196] Haoting Zhang et al. "Language Model Prompt Selection via Simulation Optimization". In: *working paper* (). Under Major Revision at *Management Science*.

[197] Haoting Zhang et al. "Machine learning-assisted stochastic kriging for offline simulation online application". In: *Working Paper* ().

[198] Haoting Zhang et al. "Machine Learning-Assisted Stochastic Kriging Metamodels for Offline Simulation Online Application". In: *working paper* (). Under Major Revision at *INFORMS Journal on Computing*.

[199] Haoting Zhang et al. "Neural network-assisted simulation optimization with covariates". In: *2021 Winter Simulation Conference (WSC)*. IEEE. 2021, pp. 1–12.

[200] Shuyi Zhang et al. "Cautionary tales on air-quality improvement in Beijing". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2205 (2017), p. 20170457.

[201] Yunkai Zhang et al. "Does Attention in Transformers Help Wildfire Prediction?" In: *working paper* (). short version accepted by *INFORMS Workshop on Data Mining and Decision Analytics*.

[202] Dongruo Zhou, Lihong Li, and Quanquan Gu. "Neural contextual bandits with ucb-based exploration". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11492–11502.

[203] Yi Zhu and Jing Dong. "On constructing confidence region for model parameters in stochastic gradient descent via batch means". In: *2021 Winter Simulation Conference (WSC)*. IEEE. 2021, pp. 1–12.

[204] Yinglun Zhu et al. "Contextual bandits with large action spaces: Made practical". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 27428–27453.

[205] Fuzhen Zhuang et al. “A comprehensive survey on transfer learning”. In: *Proceedings of the IEEE* 109.1 (2020), pp. 43–76.

Appendix A

Supplements for Chapter 2

A.1 Acquisition Functions

In the main text, we employ the upper confidence bound function as the acquisition function in the contextual Bayesian optimization approach. Here, we provide two alternative choices: Thompson sampling (TS) and knowledge gradient (KG). We describe the two procedures of the contextual GP bandit problems with NN-AGP, where the acquisition function is replaced by TS or KG. Both of them utilize the posterior distribution of the NN-AGP model

$$f\left(\mathbf{x};\boldsymbol{\theta}_t\right) \mid \mathcal{D}_{t-1} \sim \mathcal{N}\left(\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right), \sigma_{t-1}^2\left(\mathbf{x};\boldsymbol{\theta}_t\right)\right) \tag{A.1}$$

with

$$\begin{aligned}
\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right) =& \tilde{\mathcal{K}}_{(\mathbf{x};\boldsymbol{\theta}_t)}^{\top}\left[\tilde{\mathcal{K}}_{\mathcal{D}_{t-1}} + \sigma_\epsilon^2 I_{t-1}\right]^{-1}\mathbf{y}_{t-1}, \\
\sigma_{t-1}^2\left(\mathbf{x};\boldsymbol{\theta}_t\right) =& \boldsymbol{g}(\boldsymbol{\theta}_t)^{\top}\mathcal{K}\left(\mathbf{x},\mathbf{x}\right)\boldsymbol{g}(\boldsymbol{\theta}_t) - \tilde{\mathcal{K}}_{(\mathbf{x};\boldsymbol{\theta}_t)}^{\top}\left[\tilde{\mathcal{K}}_{\mathcal{D}_{t-1}} + \sigma_\epsilon^2 I_{t-1}\right]^{-1}\tilde{\mathcal{K}}_{(\mathbf{x};\boldsymbol{\theta}_t)}.
\end{aligned}$$

NN-AGP-TS

Thompson sampling (TS) is a heuristic for choosing actions in the multi-armed bandit problem. It chooses the action that maximizes the expected reward with respect to a random belief that is drawn for a posterior distribution. Besides the multi-armed bandit problems, TS has also achieved both theoretical and practical success in BO and Gaussian process regression. For more detailed discussions on TS, we refer to [144, 145].

Specifically, we propose a neural network-accompanied Gaussian process Thompson sampling (NN-AGP-TS) approach to address contextual GP bandits. The approach works as follows. In each iteration, NN-AGP-TS first fits an NN-AGP model with the historic data. Then, given the current contextual variable, a realization of the Gaussian process with respect to $\mathbf{x} \in \mathcal{X}$ is sampled from the posterior distribution conditional on the historic data[1].

[1] An efficient implementation of sampling Gaussian processes given the mean and covariance functions can be found in https://www.r-bloggers.com/2019/07/sampling-paths-from-a-gaussian-process/.

That is,

$$\hat{f}\left(\mathbf{x};\boldsymbol{\theta}_t\right) \sim \mathcal{N}\left(\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right), \sigma_{t-1}^2\left(\mathbf{x};\boldsymbol{\theta}_t\right)\right), \mathbf{x} \in \mathcal{X}.$$

The realization $\hat{f}\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ is a deterministic function and adopts a closed-form expression with respect to $\mathbf{x}$. Thus, efficient optimization approaches (e.g. global heuristic search) can be applied to find the next point to sample $\mathbf{x}_t$ by solving the optimization problem

$$\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \hat{f}\left(\mathbf{x};\boldsymbol{\theta}_t\right).$$

The complete procedure of NN-AGP-TS is summarized as in **Algorithm A.1**.

Algorithm A.1 NN-AGP-TS

Input: A prior of $(\mathbf{W}, \Phi, \sigma_\epsilon^2)$;
for $t = 1, 2, \ldots, T$ **do**
  Observe the contextual variable $\boldsymbol{\theta}_t$;
  Sample a realization $\hat{f}\left(\mathbf{x};\boldsymbol{\theta}_t\right) \sim \mathcal{N}\left(\mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right), \sigma_{t-1}^2\left(\mathbf{x};\boldsymbol{\theta}_t\right)\right), \mathbf{x} \in \mathcal{X}$;
  Select $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \hat{f}\left(\mathbf{x};\boldsymbol{\theta}_t\right)$;
  Sample y_t at $(\boldsymbol{\theta}_t, \mathbf{x}_t)$;
  Update $\left(\hat{\mathbf{W}}_t, \hat{\Phi}_t, \hat{\sigma_{\epsilon;t}^2}\right)$;
end for

Different from the upper confidence bound (UCB)-based algorithms, the TS method considers a Bayesian cumulative regret

$$\tilde{\mathcal{R}}_T = \sum_{t=1}^{T} \mathbb{E}\left[\sup_{\mathbf{x}'\in\mathcal{X}} f\left(\mathbf{x}', \boldsymbol{\theta}_t\right) - f\left(\mathbf{x}_t, \boldsymbol{\theta}_t\right)\right],$$

where the expectation is taken over the prior distribution of $f(\mathbf{x};\boldsymbol{\theta})$. We provide the upper bound of the cumulative regret for the NN-AGP-TS as a sanity check. For simplicity, we consider the scenario when $|\mathcal{X}|$ is finite. For a more general setting when $\mathcal{X}$ is continuous, the strategy of discretization that is adopted in the proof of **Theorem 2.1** can be applied as well.

Theorem A.1. *Suppose that $g(\boldsymbol{\theta})$ is a known continuous mapping of $\boldsymbol{\theta} \in \Theta$; $\mathbf{p}(\mathbf{x})$ is sampled from a known MGP prior and the variance of the noise σ_ϵ^2 is known. In addition, $|\mathcal{X}|$ is finite and Θ is a convex and compact set. The Bayesian cumulative regret of NN-AGP-TS is bounded by*

$$\tilde{\mathcal{R}}_T \leqslant C + 2\sqrt{\frac{CT\gamma_T}{\log\left(1 + C\sigma_\epsilon^{-2}\right)} \log\left(\frac{(T^2+1)|\mathcal{X}|}{\sqrt{2\pi}}\right)},$$

where $C = \left(\left(\sum_{q=1}^{Q}\sum_{l=1}^{m} a_{l,q}^2\right) + 1\right) \sup_{\boldsymbol{\theta}\in\Theta} \|g(\boldsymbol{\theta})\|_2^2$ is a constant.

The proof is largely based on the methodology proposed in [144] (therefore we will not present here considering the length of the supplements) and the upper bound of the posterior variance $\sigma_t^2(\mathbf{x};\boldsymbol{\theta}_t)$. We postpone the discussion on the upper bound of posterior variance to Section A.3, which is also used in the proof of **Theorem 2.1**.

NN-AGP-KG

In this section, we present the procedure of the neural network-accompanied knowledge gradient (NN-AGP-KG) approach, where the acquisition function employed in each iteration is contextual knowledge gradient (C-KG). The C-KG function at time t is defined as

$$\text{C-KG}_t\left(\mathbf{x};\boldsymbol{\theta}_t\right) = \mathbb{E}_{y_t}\left[\mu_t^*\left(\boldsymbol{\theta}_t\right) - \mu_{t-1}^*\left(\boldsymbol{\theta}_t\right) \mid \mathbf{x}_t = \mathbf{x}\right], \tag{A.2}$$

where $\mu_{t-1}^*\left(\boldsymbol{\theta}_t\right) = \max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ and $\mu_t^*\left(\boldsymbol{\theta}_t\right) = \max_{\mathbf{x}\in\mathcal{X}} \mu_t\left(\mathbf{x};\boldsymbol{\theta}_t\right)$. The difference between $\mu_{t-1}^*\left(\boldsymbol{\theta}_t\right)$ and $\mu_t^*\left(\boldsymbol{\theta}_t\right)$ is that, given the data-set $\mathcal{D}_{t-1}$, $\mu_{t-1}^*\left(\boldsymbol{\theta}_t\right)$ is deterministic, while $\mu_t^*\left(\boldsymbol{\theta}_t\right)$ depends on y_t and therefore is random. Thus, the C-KG function requires taking the expectation with the unrevealed observation y_t. As is the regular KG acquisition function, simulated samples of y_t as in (A.1) are required to approximate both the value of $\text{C-KG}_t\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ and the gradient $\nabla_{\mathbf{x}}\,\text{C-KG}_t\left(\mathbf{x};\boldsymbol{\theta}_t\right)$.

We present the procedure of selecting the decision variable $\mathbf{x}_t$ in each iteration with the C-KG and the NN-AGP model in **Algorithm A.2**. For more detailed discussions on the knowledge gradient, including the statistical properties, we refer to [75, 150].

Algorithm A.2 Selection of $\mathbf{x}_t$ based on C-KG

Input: Number of points for the multi-start global optimization approach $\mathfrak{R}$; Number of iterations in the gradient descent for each starting point $\mathfrak{T}$; A rule to decide the step size $\alpha_{\mathfrak{t}}$;
for $\mathfrak{r} = 1, 2, \ldots, \mathfrak{R}$ **do**
  Select $\mathbf{x}_0^{(\mathfrak{r})}$ uniformly at random from $\mathcal{X}$;
  for $\mathfrak{t} = 1, 2, \ldots, \mathfrak{T}$ **do**
    Attain the stochastic gradient estimate of $\nabla_{\mathbf{x}}\,\text{C-KG}_t\left(\mathbf{x}_{\mathfrak{t}}^{(\mathfrak{r})};\boldsymbol{\theta}_t\right)$ as in **Algorithm A.4**, denoted as $\mathfrak{G}$;
    Decide the step size $\alpha_{\mathfrak{t}}$;
    $\mathbf{x}_{\mathfrak{t}}^{(\mathfrak{r})} = \mathbf{x}_{\mathfrak{t}-1}^{(\mathfrak{r})} + \alpha_{\mathfrak{t}}\mathfrak{G}$;
  end for
  Estimate $\text{C-KG}_t\left(\mathbf{x}_{\mathfrak{T}}^{(\mathfrak{r})};\boldsymbol{\theta}_t\right)$ as in **Algorithm A.3**;
end for
Return $\mathbf{x}_t = \arg\max_{\mathbf{x}_{\mathfrak{T}}^{(\mathfrak{r})}} \text{C-KG}_t\left(\mathbf{x}_{\mathfrak{T}}^{(\mathfrak{r})};\boldsymbol{\theta}_t\right)$.

At the end of this section, we note that some classical acquisition functions that are widely adopted in Bayesian optimization might not be directly employed when the objective function

Algorithm A.3 Estimation of C-KG

Input: The conditional mean $\mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$ and variance $\sigma^2_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$ as attained in (2.2);
Attain $\mu^*_{t-1}(\boldsymbol{\theta}_t) = \max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$;
for $\mathfrak{j} = 1, 2, \ldots, \mathfrak{J}$ **do**
  Sample $y_t \sim \mathcal{N}\left(\mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t), \sigma^2_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)\right)$;
  Update $\mu_t(\mathbf{x}';\boldsymbol{\theta}_t), \mathbf{x}' \in \mathcal{X}$ with $\mathcal{D}_{t-1} \cup \{(\boldsymbol{\theta}_t, \mathbf{x}, y_t)\}$;
  Attain $\mu^*_t(\boldsymbol{\theta}_t) = \max_{\mathbf{x}'\in\mathcal{X}} \mu_t(\mathbf{x}';\boldsymbol{\theta}_t)$;
  $\Delta^{(\mathfrak{j})} = \mu^*_t(\boldsymbol{\theta}_t) - \mu^*_{t-1}(\boldsymbol{\theta}_t)$;
end for
Return C-KG$_t(\mathbf{x};\boldsymbol{\theta}_t) = \frac{1}{\mathfrak{J}} \sum_{\mathfrak{j}=1}^{\mathfrak{J}} \Delta^{(\mathfrak{j})}$.

Algorithm A.4 Estimation of $\nabla_{\mathbf{x}}$ C-KG

Input: The conditional mean $\mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$ and variance $\sigma^2_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$ as attained in (2.2);
Attain $\mu^*_{t-1}(\boldsymbol{\theta}_t) = \max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$;
for $\mathfrak{i} = 1, 2, \ldots, \mathfrak{J}$ **do**
  Sample $y_t \sim \mathcal{N}\left(\mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t), \sigma^2_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)\right)$;
  Update $\mu_t(\mathbf{x}';\boldsymbol{\theta}_t), \mathbf{x}' \in \mathcal{X}$ with $\mathcal{D}_{t-1} \cup \{(\boldsymbol{\theta}_t, \mathbf{x}, y_t)\}$;
  Attain $\mathbf{x}^* = \arg\max_{\mathbf{x}'\in\mathcal{X}} \mu_t(\mathbf{x}';\boldsymbol{\theta}_t)$;
  $\mathfrak{G}^{(\mathfrak{i})} = \nabla_{\mathbf{x}} \mu_t(\mathbf{x};\boldsymbol{\theta}_t)\,|_{\mathbf{x}=\mathbf{x}^*}$;
end for
Return $\nabla_{\mathbf{x}}$ C-KG$_t(\mathbf{x};\boldsymbol{\theta}_t) = \frac{1}{\mathfrak{J}} \sum_{\mathfrak{i}=1}^{\mathfrak{J}} \mathfrak{G}^{(\mathfrak{i})}$.

involves contextual variables. An example is the expected improvement (EI) acquisition function, which is formulated as

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}\left[\max\left\{f(\mathbf{x}) - f^*, 0\right\}\right],$$

where f^* denotes the maximum values among the observations up to now, and the expectation is taken with the posterior distribution of the GP model f conditional on the observed data. In other words, f^* serves as a so-called incumbent that guides how to select the next point. However, when the objective function involves contextual variables, it is very likely that there are no observed data under the current value of the contextual variable. Thus, we can not define an incumbent for the EI function with the current contextual variable. A similar difficulty appears in the probability of improvement (PI) acquisition function as well. The comparison between different acquisition functions with our NN-AGP model will be contained in future work.

A.2 NN-AGP with Neural Network Error

In the main text, we provide an NN-AGP-UCB algorithm in Section 2.3 and provide the upper bound of the regrets in **Theorem 2.1**, where the error of approximating the map-

ping $\boldsymbol{g}(\boldsymbol{\theta})$ using the neural networks is not taken into consideration. In this section, we provide a detailed discussion on incorporating the neural network approximation error into consideration. Specifically, we select the MGP model $\mathbf{p}(\mathbf{x})$ in (2.4) as well. That is,

$$\begin{aligned} f(\mathbf{x};\boldsymbol{\theta}) =& \boldsymbol{g}(\boldsymbol{\theta})^{\top}\mathbf{p}(\mathbf{x}) \\ =& \sum_{q=1}^{Q}\left(\sum_{l=1}^{m}\boldsymbol{g}_l\left(\boldsymbol{\theta}\right)a_{l,q}u_q\left(\mathbf{x}\right)\right)+\sum_{l=1}^{m}\boldsymbol{g}_l\left(\boldsymbol{\theta}\right)v_l\left(\mathbf{x}\right) \end{aligned}$$

where u_q's and v_l's are independent scalar Gaussian processes with known kernel functions, and $a_{l,q}$'s are also known in advance. We note that the prior knowledge of the Gaussian process model is a regular assumption in Gaussian process bandits; see [111, 158]. Meanwhile, in terms of $\boldsymbol{g}(\boldsymbol{\theta}) = (\boldsymbol{g}_1(\boldsymbol{\theta}), \boldsymbol{g}_2(\boldsymbol{\theta}), \ldots, \boldsymbol{g}_m(\boldsymbol{\theta}))^{\top}$, we use

$$\hat{\boldsymbol{g}}_t(\boldsymbol{\theta}) = (\hat{\boldsymbol{g}}_{1;t}(\boldsymbol{\theta}), \hat{\boldsymbol{g}}_{2;t}(\boldsymbol{\theta}), \ldots, \hat{\boldsymbol{g}}_{m;t}(\boldsymbol{\theta}))^{\top}$$

as the approximation, where $\hat{\boldsymbol{g}}_t$ denotes the learned neural network in round t with the historical data set $\mathcal{D}_{t-1}$. Here, we select the rectified linear unit (ReLU) function as the activation function in the neural networks and assume that each entry of the deterministic mapping $\boldsymbol{g}(\boldsymbol{\theta})$ is an $\boldsymbol{\alpha}$-Hölder function. That is, given a Hölder index $\boldsymbol{\alpha} > 0$, for any multi-index $s \in \mathbb{N}^{d'}$ with $|s| = \sum_{i=1}^{d'} s_i \leq \lfloor \alpha \rfloor$, the derivative $\partial^s \boldsymbol{g}_l = \frac{\partial^{|s|}\boldsymbol{g}_l}{\partial\theta_{(1)}^{s_1}\ldots\partial\theta_{(d')}^{s_{d'}}}$ exists, where $\theta_{(i)}$ denotes the i-th entry of $\boldsymbol{\theta}$. Meanwhile, for any s satisfying $|s| = \lfloor \boldsymbol{\alpha} \rfloor$, we have

$$\sup_{\boldsymbol{\theta}\neq\boldsymbol{\theta}'}\frac{|\partial^s\boldsymbol{g}_l(\boldsymbol{\theta}) - \partial^s\boldsymbol{g}_l\left(\boldsymbol{\theta}'\right)|}{\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|_2^{\alpha-\lfloor\boldsymbol{\alpha}\rfloor}} < \infty$$

for any $\boldsymbol{\theta}, \boldsymbol{\theta}'$ in the interior of Θ.

By the universal approximation theorem of neural networks [38], when the weight parameters are properly chosen, the difference between $\hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta})$ and $\boldsymbol{g}_l(\boldsymbol{\theta})$ can be arbitrarily small (denoted by e_t) when providing enough layers of neurons. Therefore, in our bandit algorithm, we iteratively reduce the error bound e_t and enlarge the used neural network structure by adding more layers of nodes, since more observations are observed. We note that, since the data (observations y_t) come in stream, it is challenging to pre-specify a fixed, precise and relatively small error bound when the data is not sufficient at the beginning. Meanwhile, the training of the neural network may also suffer from overparameterization with a large amounts of layers of nodes when there is no sufficient data. Thus, our strategy of iteratively reducing the error bound and enlarging the neural network is reasonable. In practical implementations, when the neural network is enlarged in some iteration, we can first fix the trained components of the neural network and train the added components with the data. We note that similar strategies have been widely used in transfer learning technology; see [173, 205].

Next, we present the corresponding bandit algorithms of NN-AGP with the neural network error. Specifically, the selection of the next point is based on the NN-AGP model with

the approximated NN

$$\hat{f}_t(\mathbf{x}, \boldsymbol{\theta}) = \hat{\boldsymbol{g}}_t(\boldsymbol{\theta})^\top \mathbf{p}(\mathbf{x}).$$

Moreover, we denote

$$\begin{aligned}
\hat{\mu}_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t) =& \tilde{\tilde{\mathcal{K}}}_{(\mathbf{x};\boldsymbol{\theta}_t);t}^\top \left[\tilde{\tilde{\mathcal{K}}}_{(\mathcal{D}_{t-1});t} + \sigma_\epsilon^2 I_{t-1}\right]^{-1} \tilde{\mathbf{y}}_{t-1}, \\
\tilde{\sigma}_{t-1}^2(\mathbf{x}; \boldsymbol{\theta}_t) =& \hat{\boldsymbol{g}}_t(\boldsymbol{\theta}_t)^\top \mathcal{K}(\mathbf{x}, \mathbf{x}) \hat{\boldsymbol{g}}_t(\boldsymbol{\theta}_t) - \tilde{\tilde{\mathcal{K}}}_{(\mathbf{x};\boldsymbol{\theta}_t);t}^\top \left[\tilde{\tilde{\mathcal{K}}}_{(\mathcal{D}_{t-1});t} + \sigma_\epsilon^2 I_{t-1}\right]^{-1} \tilde{\tilde{\mathcal{K}}}_{(\mathbf{x};\boldsymbol{\theta}_t);t}.
\end{aligned}$$

Compared with the posterior mean and variance in (2.2), here we use $\hat{\boldsymbol{g}}_t(\boldsymbol{\theta})$ in stead of $\boldsymbol{g}(\boldsymbol{\theta})$. Besides, we also note that, $\hat{\mu}_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t)$ depends on the observations $\mathbf{y}_t$ while $\tilde{\sigma}_{t-1}^2(\mathbf{x}; \boldsymbol{\theta}_t)$ does not. In this way, we provide the acquisition function as

$$\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} \left\{ \hat{\mu}_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t) + \left(\tilde{\beta}_t^{1/2} + \frac{\tilde{e}_t \sqrt{t}}{\sigma_\epsilon} \right) \tilde{\sigma}_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t) \right\}. \tag{A.3}$$

Here, $\tilde{\beta}_t$ is an increasing sequence to address the exploitation-exploration trade-off, similar as β_t in NN-AGP-UCB, and $\tilde{e}_t$ is a decreasing sequence that depends on the neural network error bound e_t. Both $\tilde{\beta}_t$ and $\tilde{e}_t$ will be illustrated. We note that, by taking the neural network error into consideration, the exploration term will be enlarged. In other words, the bandit algorithm is performed more conservatively. In this way, we name the bandit algorithm that uses the acquisition function (A.3) as *NN-AGP-UCB+*. In Section A.4, we present the numerical experiments of NN-AGP-UCB+. We note that NN-AGP-UCB+ does not always outperform NN-AGP-UCB in practical since the enlarged exploration terms is overly conservative. At the end of this section, we provide the theoretical support of NN-AGP-UCB+.

Theorem A.2. *Suppose $\delta \in (0, 1)$ and the following.*

1. *The decision variable $x \in \mathcal{X} \subseteq [0, r]^d$ and $\mathcal{X}$ is convex and compact. The contextual variable $\boldsymbol{\theta} \in \Theta \subseteq [0, 1]^{d'}$ and Θ is convex and compact; each entry of $\boldsymbol{g}(\boldsymbol{\theta})$ is an $\boldsymbol{\alpha}$-Hölder function ($\boldsymbol{\alpha} > 1$); $\mathbf{p}(\mathbf{x})$ is sampled from a known MGP prior as in (2.4) and the variance of the noise σ_ϵ^2 is known.*

2. *In terms of the neural network $\boldsymbol{g}_t(\boldsymbol{\theta})$, we use the ReLU activation function. Meanwhile, in the t-th iteration, weight parameters are properly chosen such that each entry satisfies*

$$\sup_{\boldsymbol{\theta} \in \Theta} |\hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta}) - \boldsymbol{g}_l(\boldsymbol{\theta})| \leqslant e_t,$$

   where neural network error sequence is selected as $e_t = \mathcal{O}\left(\frac{1}{t^{1+\Delta}}\right), \Delta > 0$.

3. *For MGP, there exist constants $\{a_q\}_{q=1}^Q, \{b_q\}_{q=1}^Q, \{\tilde{a}_l\}_{l=1}^m, \left\{\tilde{b}_l\right\}_{l=1}^m$ satisfying*

$$\mathbb{P}\left\{\sup_{x \in \mathcal{X}} \left| \frac{\partial u_q(x)}{\partial x_j} \right| > L_q \right\} \leqslant a_q e^{-(L_q/b_q)^2}; \mathbb{P}\left\{\sup_{x \in \mathcal{X}} \left| \frac{\partial v_l(x)}{\partial x_j} \right| > \tilde{L}_l \right\} \leqslant \tilde{a}_l e^{-(\tilde{L}_l/\tilde{b}_l)^2}$$

   $\forall L_q, \tilde{L}_l > 0$ *and* $\forall j = 1, 2, \ldots, d$, $q = 1, 2, \ldots, Q$ *and* $l = 1, 2, \ldots, m$.

4. We choose as a hyper-parameter in (A.3)

$$\tilde{\beta}_t = 2\log\left(t^2 2\pi^2/(3\delta)\right) + 2d\log\left(\tilde{\tilde{M}} t^2 dbr\sqrt{\log(4da/\delta)}\right),$$

where d and r are the dimension and the upper bound of the decision variable, $a = \sum_{q=1}^{Q} a_q + \sum_{l=1}^{m} \tilde{a}_l$ and $b = \sum_{q=1}^{Q} b_q + \sum_{l=1}^{m} \tilde{b}_l$. Meanwhile,

$$\tilde{\tilde{M}} = \sup_{\boldsymbol{\theta}\in\Theta;t} \left\{ \left\{ \left| \sum_{l=1}^{m} \hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta}) a_{l,q} \right| \right\}_{q=1}^{Q}, \{|\hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta})|\}_{l=1}^{m} \right\}.$$

Then the cumulative regret is bounded with high probability as

$$\mathbb{P}\left\{ \mathcal{R}_T = \mathcal{O}\left(\sqrt{T\gamma_T\tilde{\beta}_T}\right) + \mathcal{O}\left(T\left(\gamma_T\right)^{\frac{1}{4}}\left(\tilde{\beta}_T\right)^{\frac{1}{2}}\right), \forall T \geqslant 1 \right\} \geqslant 1-\delta.$$

Here $C = \left(\left(\sum_{q=1}^{Q}\sum_{l=1}^{m} a_{l,q}^2\right) + 1\right) \sup_{\boldsymbol{\theta}\in\Theta} \|\boldsymbol{g}(\boldsymbol{\theta})\|_2^2$. In addition, γ_T is the maximum information gain associated with the NN-AGP $f(\mathbf{x};\boldsymbol{\theta})$, defined by (2.7). We also note that $\tilde{e}_t = \mathcal{C}_1 e_t$ and the neural network in the t-th iteration $\hat{\boldsymbol{g}}_t$ has (i) no more than $\mathcal{C}_2(1-\log(e_t))$ layers and (ii) at most $\mathcal{C}_3 e_t^{-\frac{d'}{\alpha}}(1-\log(e_t))$ neurons and weight parameters, where $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ are all constants.

Compared with the results in **Theorem 2.1**, the regret bound of NN-AGP-UCB+ involves additional term $\mathcal{O}\left(T\left(\gamma_T\right)^{\frac{1}{4}}\left(\tilde{\beta}_T\right)^{\frac{1}{2}}\right)$, which results from the neural network approximation error. The proof of **Theorem A.2** is contained in Section A.3.

A.3 Proofs

We present the detailed discussions on **Theorem 2.1**, **Theorem 2.2** and **Theorem A.2** in this section, as well as the consistency of training the NN-AGP model from the data. We note that all these theoretical results are based on the assumption that $\mathbf{p}(\mathbf{x})$ is a linear combination of independent GP realizations (u_q's and v_l's). It is a common assumption that u_q's and v_l's are realizations from GP's, while an alternative review is to assume that each u_q or v_l is a deterministic function that lives in reproducing kernel Hilbert space (RKHS), which is consistent with relevant literature [156]. The difference between modeling the reward function as a GP sample or an element in an RKHS reflects the difference between Bayesianists and frequentists, as discussed in [156]. In our work, we adopt a Bayesian view since it helps us better understand the construction of the acquisition function. If the reward function $f(\mathbf{x};\boldsymbol{\theta})$ is assumed as a linear combination of deterministic functions u_q's and v_l's that are from RKHS, martingale-based technologies introduced in [44] can be employed to derive the regret bound as well. This technology leads to a slightly tighter bound with less restrictive assumptions on $\mathcal{X}$, while it is beyond the scope of this work.

Proof of Theorem 2.1

In this section, we present the detailed proof of **Theorem 2.2**. The employed methodologies borrow the ideas from [156] and [111].

Lemma A.1 ([156])**.** *The information gain for the points selected can be expressed in terms of the predictive variances. Specifically,*

$$I\left(\mathbf{y}_T; f_T\right) = \frac{1}{2}\sum_{t=1}^{T} \log\left(1 + \sigma_\epsilon^{-2}\sigma_{t-1}^2\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right)\right).$$

Lemma A.2. *When β_t's are selected nondecreasing,*

$$\sum_{t=1}^{T} 4\beta_t \sigma_{t-1}^2\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right) \leqslant \frac{8C\beta_T\gamma_T}{\log\left(1 + C\sigma_\epsilon^{-2}\right)}, \forall T \geqslant 1.$$

Here $C = \left(\left(\sum_{q=1}^{Q}\sum_{l=1}^{m} a_{l,q}^2\right) + 1\right)\sup_{\boldsymbol{\theta}\in\Theta}\|\boldsymbol{g}(\boldsymbol{\theta})\|_2^2$ is a constant.

Proof. Note that

$$\tilde{K}\left(\left(\mathbf{x}, \boldsymbol{\theta}\right), \left(\mathbf{x}', \boldsymbol{\theta}'\right)\right) = \sum_{q=1}^{Q} \boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{A}_q \boldsymbol{g}(\boldsymbol{\theta}') k_q\left(\mathbf{x}, \mathbf{x}'\right) + \sum_{l=1}^{m} \boldsymbol{g}_l(\boldsymbol{\theta})\boldsymbol{g}_l(\boldsymbol{\theta}')\tilde{k}_l(\mathbf{x}, \mathbf{x}').$$

With the regular conditions that all the kernel functions of u_q's and v_l's are less than one,

$$\begin{aligned}
\sigma_{t-1}^2\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right) &\leqslant \tilde{K}\left(\left(\mathbf{x}_t, \boldsymbol{\theta}_t\right), \left(\mathbf{x}_t, \boldsymbol{\theta}_t\right)\right) \\
&\leqslant \sum_{q=1}^{Q} \boldsymbol{g}\left(\boldsymbol{\theta}_t\right)^\top \mathbf{A}_q \boldsymbol{g}\left(\boldsymbol{\theta}_t\right) + \sum_{l=1}^{m}\left(\boldsymbol{g}_l\left(\boldsymbol{\theta}_t\right)\right)^2 \\
&\leqslant \left(\left(\sum_{q=1}^{Q}\sum_{l=1}^{m} a_{l,q}^2\right) + 1\right)\sup_{\boldsymbol{\theta}\in\Theta}\|g(\boldsymbol{\theta})\|_2^2 \\
&= C.
\end{aligned}$$

Indeed, since $\mathbf{A}_q = \begin{pmatrix} a_{1,q} \\ \vdots \\ a_{m,q} \end{pmatrix}\begin{pmatrix} a_{1,q} & \dots & a_{m,q} \end{pmatrix}$, the only non-zero eigenvalue of $\mathbf{A}_q$ is

$$\begin{pmatrix} a_{1,q} & \dots & a_{m,q} \end{pmatrix}\begin{pmatrix} a_{1,q} \\ \vdots \\ a_{m,q} \end{pmatrix} = \sum_{l=1}^{m} a_{l,q}^2.$$

This is the reason why the last inequality holds.

Next, since β_t's are nondecreasing as t increases,

$$\begin{aligned}4\beta_t\sigma_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) &\leqslant 4\beta_T\sigma_\epsilon^2\left(\sigma_\epsilon^{-2}\sigma_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)\right)\\ &\leqslant \beta_T\frac{8C}{\log\left(1+C\sigma_\epsilon^{-2}\right)}\frac{1}{2}\log\left(1+\sigma_\epsilon^{-2}\sigma_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)\right).\end{aligned}$$

The second inequality holds since $s/\log(1+s)$ is an increasing function with s and

$$\sigma_\epsilon^{-2}\sigma_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)\leqslant C\sigma_\epsilon^{-2}.$$

Thus, summing up the inequalities with t to T, we have

$$\sum_{t=1}^{T}4\beta_t\sigma_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)\leqslant\frac{8C\beta_T\gamma_T}{\log\left(1+C\sigma_\epsilon^{-2}\right)}.$$

□

Lemma A.3 ([156]). *$\forall\delta\in(0,1)$, choose $\beta_t=2\log(\pi_t/\delta)$, where $\sum_{t\geqslant1}\pi_t^{-1}=1$.*

$$\mathbb{P}\left\{\forall t,\left|f(\mathbf{x}_t;\boldsymbol{\theta}_t)-\mu_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)\right|\leqslant\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)\right\}\geqslant1-\delta.$$

Lemma A.4 ([156]). *Suppose that candidate decision variables $\mathbf{x}_t$ are finite in each round, that is, $|\mathcal{X}_t|$ is finite. $\forall\delta\in(0,1)$, choose $\beta_t=2\log(|\mathcal{X}_t|\,\pi_t/\delta)$, where $\sum_{t\geqslant1}\pi_t^{-1}=1$.*

$$\mathbb{P}\left\{\forall t,\forall\mathbf{x}\in\mathcal{X}_t,\left|f(\mathbf{x};\boldsymbol{\theta}_t)-\mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)\right|\leqslant\beta_t^{1/2}\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)\right\}\geqslant1-\delta.$$

The difference between **Lemma A.3** and **Lemma A.4** is that **Lemma A.3** considers a specific $\mathbf{x}_t$ while **Lemma A.4** considers a uniform bound on $\mathcal{X}_t$.

To connect the continuous decision variable space $\mathcal{X}$ to results with finite selections of $\mathbf{x}_t$, we then discretize $\mathcal{X}$ based on the smoothness condition imposed in **Theorem 2.1**. Recall that

$$f(\mathbf{x};\boldsymbol{\theta})=\sum_{q=1}^{Q}\left(\sum_{l=1}^{m}\boldsymbol{g}_l\left(\boldsymbol{\theta}\right)a_{l,q}u_q\left(\mathbf{x}\right)\right)+\sum_{l=1}^{m}\boldsymbol{g}_l\left(\boldsymbol{\theta}\right)v_l\left(\mathbf{x}\right),$$

where u_q's and v_l's are independent scalar Gaussian processes. Meanwhile, for smoothness, we assume that for the components of the MGP, there exist constants

$$\left\{a_q\right\}_{q=1}^{Q},\left\{b_q\right\}_{q=1}^{Q},\left\{\tilde{a}_l\right\}_{l=1}^{m},\left\{\tilde{b}_l\right\}_{l=1}^{m}$$

satisfying

$$\begin{aligned}\mathbb{P}\left\{\sup_{x\in\mathcal{X}}\left|\frac{\partial u_q(x)}{\partial x_j}\right|>L_q\right\}&\leqslant a_qe^{-(L_q/b_q)^2},\\ \mathbb{P}\left\{\sup_{x\in\mathcal{X}}\left|\frac{\partial v_l(x)}{\partial x_j}\right|>\tilde{L}_l\right\}&\leqslant\tilde{a}_le^{-(\tilde{L}_l/\tilde{b}_l)^2},\end{aligned}$$

$\forall L_q, \tilde{L}_l > 0$ and $\forall j = 1, 2, \ldots, d$ for $q = 1, 2, \ldots, Q$ and $l = 1, 2, \ldots, m$.

For any fixed k, we could select L_q and $\tilde{L}_l$ such that,

$$k = \frac{L_1^2}{b_1^2} = \ldots = \frac{L_Q^2}{b_Q^2} = \frac{\tilde{L}_1^2}{\tilde{b}_1^2} = \ldots = \frac{\tilde{L}_m^2}{\tilde{b}_m^2}.$$

Let $L = \tilde{M}(L_1 + \ldots + \tilde{L}_m)$, we have

$$\begin{aligned}
\mathbb{P}\left\{\sup_{\mathbf{x}\in\mathcal{X}}\left|\frac{\partial f}{\partial \mathbf{x}_j}\right| > L\right\} &\leqslant \mathbb{P}\left\{\sup_{\mathbf{x}\in\mathcal{X}}\left\{\sum_{q=1}^{Q}\left|\frac{\partial u_q(\mathbf{x})}{\partial \mathbf{x}_j}\right| + \sum_{l=1}^{m}\left|\frac{\partial v_l(\mathbf{x})}{\partial \mathbf{x}_j}\right|\right\} > \frac{L}{\tilde{M}}\right\} \\
&\leqslant \sum_{q=1}^{Q}\mathbb{P}\left\{\sup_{x\in\mathcal{X}}\left|\frac{\partial u_q(x)}{\partial x_j}\right| > L_q\right\} + \sum_{l=1}^{m}\mathbb{P}\left\{\sup_{x\in\mathcal{X}}\left|\frac{\partial v_l(x)}{\partial x_j}\right| > \tilde{L}_l\right\} \\
&\leqslant \sum_{q=1}^{Q} a_q e^{-(L_q/b_q)^2} + \sum_{l=1}^{m}\tilde{a}_l e^{-\left(\tilde{L}_l/\tilde{b}_l\right)^2} \\
&\leqslant ae^{-k},
\end{aligned}$$

where $a = a_1 + \ldots + a_Q + \tilde{a}_1 + \ldots + \tilde{a}_m$. That is,

$$\mathbb{P}\left\{\forall\boldsymbol{\theta}, \forall\mathbf{x}, \mathbf{x}', \left|f(\mathbf{x};\boldsymbol{\theta}) - f\left(\mathbf{x}';\boldsymbol{\theta}\right)\right| \leqslant L\left\|\mathbf{x} - \mathbf{x}'\right\|_1\right\} \geqslant 1 - dae^{-k}, \tag{A.4}$$

where $L = \tilde{M}b\sqrt{k}, b = b_1 + \ldots + b_Q + \tilde{b}_1 + \ldots + \tilde{b}_m$.

In this way, we discretize $\mathcal{X}$ as $\mathcal{X}_t$ of size $(\tau_t)^d$ in each round so that for all $\mathbf{x} \in \mathcal{X}$, we can find

$$\left\|\mathbf{x} - [\mathbf{x}]_t\right\|_1 \leq rd/\tau_t,$$

where $[\mathbf{x}]_t$ denotes a point in $\mathcal{X}_t$ that is the closest to $\mathbf{x}$. A sufficient discretization has each coordinate in $\mathcal{X}$ with τ_t uniformly spaced points.

Lemma A.5. *$\forall\delta \in (0,1)$, choose $\beta_t = 2\log\left(2\pi_t/\delta\right) + 2d\log\left(dt^2r\tilde{M}b\sqrt{\log\left(\frac{2ad}{\delta}\right)}\right)$, where $\sum_{t\geqslant 1}\pi_t^{-1} = 1$.*

$$\mathbb{P}\left\{\forall t, \left|f(\mathbf{x}_t^*;\boldsymbol{\theta}_t) - \mu_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t)\right| \leqslant \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t) + \frac{1}{t^2}\right\} \geqslant 1 - \delta.$$

Proof. For any $\delta \in (0,1)$, we let $k = \log\left(\frac{2ad}{\delta}\right)$ in (A.4). We have

$$\mathbb{P}\left\{\left|f(\mathbf{x};\boldsymbol{\theta}) - f\left([\mathbf{x}]_t;\boldsymbol{\theta}\right)\right| \leqslant \frac{1}{t^2}\right\} \geqslant 1 - \frac{\delta}{2},$$

with

$$\tau_t = dt^2r\tilde{M}b\sqrt{\log\left(\frac{2ad}{\delta}\right)}.$$

Choose $\beta_t = 2\log\left(2\left(\tau_t\right)^d \pi_t/\delta\right) = 2\log\left(2\pi_t/\delta\right) + 2d\log\left(dt^2 r\tilde{M}b\sqrt{\log\left(\frac{2ad}{\delta}\right)}\right)$, we have

$$\mathbb{P}\left\{\left|f(\mathbf{x}_t^*;\boldsymbol{\theta}) - f\left(\left[\mathbf{x}_t^*\right]_t;\boldsymbol{\theta}\right)\right| \leqslant \frac{1}{t^2}\right\} \geqslant 1 - \frac{\delta}{2}.$$

Based on **Lemma A.4**, we have

$$\mathbb{P}\left\{\forall t, \left|f([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t) - \mu_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t)\right| \leqslant \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t)\right\} \geqslant 1 - \frac{\delta}{2}.$$

for the reason that β_t here is larger than the required β_t in **Lemma A.4**. Thus,

$$\mathbb{P}\left\{\forall t, \left|f(\mathbf{x}_t^*;\boldsymbol{\theta}_t) - \mu_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t)\right| \leqslant \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t) + \frac{1}{t^2}\right\} \geqslant 1 - \delta.$$

□

Lemma A.6. *Choose* $\beta_t = 2\log\left(4\pi_t/\delta\right) + 2d\log\left(dt^2 r\tilde{M}b\sqrt{\log\left(\frac{4ad}{\delta}\right)}\right)$, *where* $\sum_{t\geqslant 1}\pi_t^{-1} = 1$.

$$\mathbb{P}\left\{\forall t, r_t \leqslant 2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) + \frac{1}{t^2}\right\} \geqslant 1 - \delta.$$

Proof. Select $\delta/2$ in **Lemma A.3** and **Lemma A.5**. With probability at least $1-\delta$, we have

$$\forall t, \left|f\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) - \mu_{t-1}\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)\right| \leqslant \beta_t^{1/2}\sigma_{t-1}\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)$$

and

$$\forall t, \left|f(\mathbf{x}_t^*;\boldsymbol{\theta}_t) - \mu_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t)\right| \leqslant \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t) + \frac{1}{t^2},$$

since β_t here is larger than that required in **Lemma A.3**. Thus,

$$\begin{aligned}
r_t &= f\left(\mathbf{x}_t^*;\boldsymbol{\theta}_t\right) - f\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) \\
&\leqslant \mu_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t) + \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t) + \frac{1}{t^2} - f\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) \\
&\leqslant \beta_t^{1/2}\sigma_{t-1}\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) + \frac{1}{t^2} + \mu_{t-1}\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) - f\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) \\
&\leqslant 2\beta_t^{1/2}\sigma_{t-1}\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) + \frac{1}{t^2},
\end{aligned}$$

□

Based on these results, we provide the proof of **Theorem 2.1** in the main text.

Proof. Recall that $\mathcal{R}_T = \sum_{t=1}^T r_t$ and

$$\sum_{t=1}^T 4\beta_t\sigma_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) \leqslant \frac{8C\beta_T\gamma_T}{\log\left(1 + C\sigma_\epsilon^{-2}\right)}.$$

Based on **Lemma A.6**, with probability at least $1-\delta$, we have

$$\begin{aligned}\mathcal{R}_T \leqslant & \sqrt{T\sum_{t=1}^{T} 4\beta_t\sigma_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)+\frac{\pi^2}{6}} \\ \leqslant & \sqrt{\frac{8C\beta_T\gamma_T T}{\log\left(1+C\sigma_\epsilon^{-2}\right)}+\frac{\pi^2}{6}}.\end{aligned}$$

The first inequality holds because of the Cauchy–Schwarz inequality. Choose $\pi_t = \frac{2\pi^2 t^2}{3\delta}$, we attain the results in **Theorem 2.1**.

□

At the end of this section, we compare our result with that of [111]. First, we impose the smoothness conditions on the separate components of the MGP (for further discussion on this condition refer to **Theorem 5** in [81]. [111] impose a similar condition directly on the joint GP with composite kernels. Thus, we offer a more explicit condition to verify. Second, we discretize the decision variable space, instead of the joint space of the decision variable and the contextual variable. Therefore, the upper bound on the cumulative regret $\mathcal{R}_T$ increases as the dimension of the decision variable space increases and is not related to the size of the contextual variable space. In comparison, the cumulative regret bound derived in [111] increases when the dimension of either the decision variable or the contextual variable increases. Therefore, NN-AGP-UCB adopts the advantage of a smaller upper bound on the cumulative regret when the dimension of the contextual variable is relatively high, which is also supported by experiment results in Section A.4.

Proof of Theorem 2.2

The discussion of **Theorem 2.2** is based on the Mercer decomposition and is inspired by [165]. To begin with, we first present the Mercer Theorem.

Theorem A.3 (Mercer Theorem [72])**.** *Suppose $K(\mathbf{x},\mathbf{x}')$ is a continuous symmetric non-negative definite kernel defined on $\mathcal{X}\times\mathcal{X}$. The kernel function $K(\mathbf{x},\mathbf{x}')$ is called positive semi-definite (PSD) if any Gram-matrix generated by the kernel function is PSD. That is, for any sequence $\{\mathbf{x}_1<\mathbf{x}_2<\ldots<\mathbf{x}_n\}$, the matrix*

$$\begin{pmatrix} K\left(\mathbf{x}_1,\mathbf{x}_1\right) & \cdots & K\left(\mathbf{x}_1,\mathbf{x}_n\right) \\ \vdots & \ddots & \vdots \\ K\left(\mathbf{x}_n,\mathbf{x}_1\right) & \cdots & K\left(\mathbf{x}_n,\mathbf{x}_n\right) \end{pmatrix} \geqslant \mathbf{0}.$$

Furthermore, there exists an orthonormal basis $\{e_i\}_i$ consisting of eigenfunctions such that the corresponding sequence of eigenvalues $\{\lambda_i\}$ is non-negative. The eigenfunctions corre-

sponding to non-zero eigenvalues are continuous on $\mathcal{X}$ and K has the representation

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i e_i(\mathbf{x}) e_i(\mathbf{x}'),$$

where the convergence is absolute and uniform. Specifically, the eigenfunctions and eigenvalues satisfy that

$$\int_{\mathcal{X}} e_i(\mathbf{x}) e_j(\mathbf{x}) \, \mathrm{d}\mathbf{x} = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$
$$\int_{\mathcal{X}} e_i(\mathbf{x}) K(\mathbf{x}, \mathbf{x}') \, \mathrm{d}\mathbf{x} = \lambda_i e_i(\mathbf{x}').$$

Next, we present the Mercer decomposition for our NN-AGP model. Recall that the MGP employed in the NN-AGP model is determined by the linear transformation of multiple independent scalar Gaussian processes

$$\mathbf{p}_l(\mathbf{x}) = \sum_{q=1}^{Q} a_{l,q} u_q(\mathbf{x}) + v_l(\mathbf{x}).$$

Here, $\mathbf{p}_l(\mathbf{x})$ is the l-th element of $\mathbf{p}(\mathbf{x})$, where $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ and $\{v_l(\mathbf{x})\}_{l=1}^{m}$ are independent scalar-output Gaussian processes. In addition, $a_{l,q}$'s are coefficient parameters. In this way, the correlation between different entries in the MGP $\mathbf{p}(\mathbf{x})$ is captured by $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ through $a_{l,q}$. Meanwhile, $v_l(\mathbf{x})$ represents specific independent features of $\mathbf{p}_l(\mathbf{x})$ itself, for $l = 1, 2, \ldots, m$. Suppose the kernel function of $u_q(\mathbf{x})$'s and $v_l(\mathbf{x})$'s are $k_q(\mathbf{x}, \mathbf{x}')$'s and $\tilde{k}_l(\mathbf{x}, \mathbf{x}')$'s, the matrix-valued kernel function of $\mathbf{p}(\mathbf{x})$ is

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \sum_{q=1}^{Q} \mathbf{A}_q k_q(\mathbf{x}, \mathbf{x}') + \mathrm{Diag}\left\{\tilde{k}_1(\mathbf{x}, \mathbf{x}'), \ldots, \tilde{k}_m(\mathbf{x}, \mathbf{x}')\right\}.$$

Here, $\mathbf{A}_q$ denotes the semi-definite matrix, of which the (l, l')-th entry is $a_{l,q} a_{l',q}$. In this way, the kernel function for the NN-AGP is

$$\tilde{K}((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')) = \sum_{q=1}^{Q} g(\boldsymbol{\theta})^{\top} \mathbf{A}_q g(\boldsymbol{\theta}') k_q(\mathbf{x}, \mathbf{x}') + \sum_{l=1}^{m} g_l(\boldsymbol{\theta}) g_l(\boldsymbol{\theta}') \tilde{k}_l(\mathbf{x}, \mathbf{x}').$$

Thus, the kernel function of the NN-AGP model is decomposed into a summation, where each term is a product of the two kernel functions.

Proposition A.1 ([111])**.** *Suppose a kernel function $K(\mathbf{x}, \mathbf{x}')$ can be represented by a summation of kernel functions. That is, $K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} K_i(\mathbf{x}, \mathbf{x}')$. Then*

$$\gamma_T(K) \leqslant \sum_{i=1}^{n} \gamma_T(K_i).$$

Here $\gamma_T\left(K_{(\cdot)}\right)$ denotes the maximum information gain associated with kernel function $K_{(\cdot)}$.

That is, the maximum information gain of $\tilde{K}\left((\mathbf{x},\boldsymbol{\theta}),(\mathbf{x}',\boldsymbol{\theta}')\right)$ is bounded by the summation of maximum information gains of each term. Thus, for ease of notion, we focus on the scenario when $Q = 1$ and there are no v_l's. We also relax the restriction on $\mathbf{A}_1$ so that $\mathbf{A}_1 = \mathbf{A}$ is a positive semi-definite matrix and not necessarily a rank-one matrix.

Proposition A.2 (Mercer decomposition of $\tilde{k}$)**.** *Given a positive semi-definite matrix* $\mathbf{A}$ *and a continuous function* $\boldsymbol{g}(\boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta$*,*

$$\tilde{k}\left(\boldsymbol{\theta},\boldsymbol{\theta}'\right) = \boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{A} \boldsymbol{g}\left(\boldsymbol{\theta}'\right)$$

defines a positive semi-definite (PSD) kernel function. Furthermore, when Θ *is compact, the kernel function* $\tilde{k}\left(\boldsymbol{\theta},\boldsymbol{\theta}'\right)$ *has a finite-rank Mercer decomposition*

$$\tilde{k}\left(\boldsymbol{\theta},\boldsymbol{\theta}'\right) = \sum_{i=1}^{m} \mu_i \phi_i(\boldsymbol{\theta}) \phi_i\left(\boldsymbol{\theta}'\right).$$

Proof. Since $\mathbf{A}$ is a PSD matrix, it has a Cholesky decomposition as $\mathbf{A} = LL^\top$, where L is a lower triangular matrix. Denote $\tilde{\boldsymbol{g}}(\boldsymbol{\theta}) = L^\top \boldsymbol{g}(\boldsymbol{\theta})$, we have $\tilde{k}\left(\boldsymbol{\theta},\boldsymbol{\theta}'\right) = \tilde{\boldsymbol{g}}(\boldsymbol{\theta})^\top \tilde{\boldsymbol{g}}\left(\boldsymbol{\theta}'\right)$. In this way, the covariance matrix of any sequence $\{\boldsymbol{\theta}_\tau\}_{\tau=1}^t$ is

$$\begin{pmatrix} \tilde{k}\left(\boldsymbol{\theta}_1,\boldsymbol{\theta}_1\right) & \cdots & \tilde{k}\left(\boldsymbol{\theta}_1,\boldsymbol{\theta}_t\right) \\ \vdots & \ddots & \vdots \\ \tilde{k}\left(\boldsymbol{\theta}_t,\boldsymbol{\theta}_1\right) & \cdots & \tilde{k}\left(\boldsymbol{\theta}_t,\boldsymbol{\theta}_t\right) \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{g}}\left(\boldsymbol{\theta}_1\right)^\top \\ \cdots \\ \tilde{\boldsymbol{g}}\left(\boldsymbol{\theta}_t\right)^\top \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{g}}(\boldsymbol{\theta}_1) & \ldots & \tilde{\boldsymbol{g}}(\boldsymbol{\theta}_t) \end{pmatrix} \succcurlyeq \mathbf{0}.$$

Thus, $\tilde{k}\left(\boldsymbol{\theta},\boldsymbol{\theta}'\right)$ defines a PSD kernel function. With a slight abuse of notation, we let $\tilde{k}\left(\boldsymbol{\theta},\boldsymbol{\theta}'\right) = \sum_{l=1}^m \boldsymbol{g}_l(\boldsymbol{\theta}) \boldsymbol{g}_l\left(\boldsymbol{\theta}'\right)$ and will re-define $\tilde{\boldsymbol{g}}$ and $\tilde{\boldsymbol{g}}_l$ in the following part. Since Θ is bounded, we define

$$\langle \boldsymbol{g}_l, \boldsymbol{g}_{l'} \rangle = \int_\Theta \boldsymbol{g}_l(\boldsymbol{\theta}) \boldsymbol{g}_{l'}(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}$$

and

$$\|\boldsymbol{g}_l\| = \sqrt{\langle \boldsymbol{g}_l, \boldsymbol{g}_l \rangle}.$$

Next, we let $\tilde{\boldsymbol{g}}_1(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta})$. For $l = 2, \ldots, m$, we sequentially let

$$\tilde{g}_l(\boldsymbol{\theta}) = \boldsymbol{g}_l(\boldsymbol{\theta}) - \sum_{i=1}^{l-1} \frac{\langle \boldsymbol{g}_l, \tilde{\boldsymbol{g}}_i \rangle}{\|\tilde{\boldsymbol{g}}_i\|^2} \tilde{\boldsymbol{g}}_i(\boldsymbol{\theta}).$$

Without loss of generality, we assume that $\|\tilde{\boldsymbol{g}}_l(\boldsymbol{\theta})\| = 1$. In fact, $\{\boldsymbol{g}_l\}_{l=1}^m$ composes a basis of a reproducing kernel Hilbert space, and the above procedure is exactly the Gram-Schmidt process of the basis. It can be easily verified that

$$\langle \boldsymbol{g}_l, \boldsymbol{g}_{l'} \rangle = \int_\Theta \boldsymbol{g}_l(\boldsymbol{\theta}) \boldsymbol{g}_{l'}(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta} = \delta_{ll'},$$

where $\delta_{ll'} = 1$ when $l = l'$ and $\delta_{ll'} = 0$ otherwise. Since the aforementioned procedure is entirely based on linear operations, we denote

$$\begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_m \end{pmatrix} = M \begin{pmatrix} \tilde{\boldsymbol{g}}_1 \\ \tilde{\boldsymbol{g}}_2 \\ \vdots \\ \tilde{\boldsymbol{g}}_m \end{pmatrix}.$$

In this way, we have that

$$\begin{aligned} \tilde{k}\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right) =& \tilde{\boldsymbol{g}}(\boldsymbol{\theta})^\top M^\top M \tilde{\boldsymbol{g}}(\boldsymbol{\theta}) \\ =& \tilde{\boldsymbol{g}}(\boldsymbol{\theta})^\top Q^\top \Lambda Q \tilde{\boldsymbol{g}}(\boldsymbol{\theta}). \end{aligned}$$

Here $\tilde{\boldsymbol{g}}(\boldsymbol{\theta}) = \left(\tilde{\boldsymbol{g}}_1(\boldsymbol{\theta}), \tilde{\boldsymbol{g}}_2(\boldsymbol{\theta}), \ldots, \tilde{\boldsymbol{g}}_m(\boldsymbol{\theta})\right)^\top$. $Q^\top \Lambda Q$ is the eigendecomposition of a PSD matrix $M^\top M$. That is, Λ is a diagonal matrix and Q is an orthogonal matrix. Let μ_i the i-th entry of Λ and $\phi_i(\boldsymbol{\theta}$ the i-th entry of $Q\tilde{\boldsymbol{g}}(\boldsymbol{\theta})$. It can be easily verified that $\{\mu_i, \phi_i\}_{i=1}^m$ satisfies the conditions in the Mercer theorem. That is, we get the Mercer decomposition of $\tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}')$. □

Next, we show that the kernel function of NN-AGP adopts a Mercer decomposition as well.

Proposition A.3 (Mercer decomposition of NN-AGP)**.** *Suppose that the kernel function with respect to decision variable $k(\mathbf{x}, \mathbf{x}')$ is PSD, and therefore adopts a Mercer decomposition*

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sum_{j=1}^{\infty} \lambda_j \psi_j\left(\mathbf{x}\right) \psi_j\left(\mathbf{x}'\right),$$

the kernel function of the NN-AGP then adopts a Mercer decomposition

$$\tilde{K}\left((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')\right) = \sum_{j=1}^{\infty} \sum_{i=1}^{m} \mu_i \lambda_j \phi_i\left(\boldsymbol{\theta}\right) \psi_j\left(\mathbf{x}\right) \phi_i\left(\boldsymbol{\theta}'\right) \psi_j\left(\mathbf{x}'\right),$$

if both Θ and $\mathcal{X}$ are compact, $g(\boldsymbol{\theta})$ is a continuous mapping.

Proof. Recall that $\tilde{K}\left((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')\right) = \tilde{k}\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right) k\left(\mathbf{x}, \mathbf{x}'\right)$. Based on the previous proposition and the Mercer decomposition of $k(\mathbf{x}, \mathbf{x}')$ (the convergence is uniform), we have

$$\tilde{K}\left((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')\right) = \sum_{j=1}^{\infty} \sum_{i=1}^{m} \mu_i \lambda_j \phi_i\left(\boldsymbol{\theta}\right) \psi_j\left(\mathbf{x}\right) \phi_i\left(\boldsymbol{\theta}'\right) \psi_j\left(\mathbf{x}'\right).$$

Meanwhile, it can be easily verified that $\{(\mu_i \lambda_j, \phi_i\left(\boldsymbol{\theta}\right) \psi_j\left(\mathbf{x}\right))\}$ for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, \infty$, are eigenvalues and eigen-functions of $\tilde{K}\left((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}', \boldsymbol{\theta}')\right)$, which also completes the proof of **Proposition 2.2** in the main text. □

Lemma A.7. *The NN-AGP model adopts a representation*

$$f(\mathbf{x};\boldsymbol{\theta}) = \sum_{j=1}^{\infty}\sum_{i=1}^{m} \xi_{ji}\lambda_j^{\frac{1}{2}}\mu_i^{\frac{1}{2}}\phi_i(\boldsymbol{\theta})\,\psi_j(\mathbf{x}),$$

where ξ_{ji}'s are i.i.d. standard normal random variables.

Based on the Mercer decomposition of $\tilde{K}((\mathbf{x},\boldsymbol{\theta}),(\mathbf{x}',\boldsymbol{\theta}'))$, we derive the maximum information gain of the NN-AGP based on the technologies in [165]. Specifically, we select the first D largest eigenvalues of the kernel function $k(\mathbf{x},\mathbf{x}')$. Recall that in terms of the kernel function $\tilde{k}(\boldsymbol{\theta},\boldsymbol{\theta}')$, there are at most m non-zero eigenvalues. In this way, we project the NN-AGP onto the subspace and attain

$$\hat{f}(\mathbf{x};\boldsymbol{\theta}) = \sum_{j=1}^{D}\sum_{i=1}^{m} \xi_{ji}\lambda_j^{\frac{1}{2}}\mu_i^{\frac{1}{2}}\phi_i(\boldsymbol{\theta})\,\psi_j(\mathbf{x})$$

and the residual part

$$\hat{f}_r(\mathbf{x};\boldsymbol{\theta}) = \sum_{j=D+1}^{\infty}\sum_{i=1}^{m} \xi_{ji}\lambda_j^{\frac{1}{2}}\mu_i^{\frac{1}{2}}\phi_i(\boldsymbol{\theta})\,\psi_j(\mathbf{x}).$$

We then derive the bound of the maximum information gain by first considering the two separate terms.

Lemma A.8. *Suppose that 1) $\tilde{K}((\mathrm{x},\boldsymbol{\theta}),(\mathrm{x}',\boldsymbol{\theta}'))$ satisfies the conditions in **Proposition 2.2**; 2) $\forall \mathbf{x},\mathbf{x}' \in \mathcal{X}, |k(\mathbf{x},\mathbf{x}')| \leqslant \bar{k}$ for some $\bar{k} > 0$ and 3) $\forall j \in \mathbb{N}, \forall \mathbf{x} \in \mathcal{X}$, $|\psi_j(\mathbf{x})| \leqslant \psi$, for some $\psi > 0$.*

$$\gamma_T \leqslant \frac{1}{2} mD \log\left(1 + \frac{\bar{\tilde{k}}\bar{k}T}{\sigma_\epsilon^2 mD}\right) + \frac{\delta_{mD}T}{2\sigma_\epsilon^2}. \tag{A.5}$$

Here, $\delta_{mD} = \sum_{j=D+1}^{\infty}\sum_{i=1}^{m}\lambda_j\mu_i\psi^2\phi^2$; σ_ϵ^2 is the variance of the noise ϵ; $\bar{\mu} = \frac{1}{m}\sum_{i=1}^{m}\mu_i$ denotes the mean of the eigenvalues of the kernel function $\tilde{k}(\boldsymbol{\theta},\boldsymbol{\theta}')$; and $\bar{\tilde{k}} = \sup_{\boldsymbol{\theta},\boldsymbol{\theta}'\in\Theta}\left|\tilde{k}(\boldsymbol{\theta},\boldsymbol{\theta}')\right|$ and $\phi = \sup_{\boldsymbol{\theta}\in\Theta}|\phi(\boldsymbol{\theta})|$.

This is a direct result from **Theorem 3** in [165]. In terms of the right-hand side of (A.5), the first term bounds the maximum information gain associated with the projected GP while the second term bounds the remaining part of the GP. The NN-AGP can be projected onto such a sub-space for the reason that the kernel function with respect to $\boldsymbol{\theta}$ is a finite-rank kernel function. Otherwise, when both the kernel functions have infinite Mercer decompositions, the truncation of a product of two infinite series and the quantification of the residuals requires more cautious discussions.

Consider now a scalar GP with the kernel function $k(\mathbf{x}, \mathbf{x}')$, denoted as $p(\mathbf{x})$. The observations $y'_t = p(\mathbf{x}_t) + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2)$. That is, we do not consider the contextual variable here. In this way, the maximum information gain of $p(\mathbf{x})$ is bounded as

$$\gamma_{\mathbf{x};T} \leqslant \frac{1}{2} D \log\left(1 + \frac{\bar{k}T}{\sigma_\epsilon^2 D}\right) + \frac{\delta_D T}{2\sigma_\epsilon^2},$$

where $\delta_D = \sum_{j=D+1}^{\infty} \lambda_j \psi^2$. Compared with the result in (A.5), we have that

$$\gamma_T = \mathcal{O}\left(m\gamma_{\mathbf{x};T}\right).$$

Next, we specifically consider two types of the kernel function $k(\mathbf{x}, \mathbf{x}')$ as in **Definition 2.1**. Consider the eigenvalues $\{\lambda_j\}_{j=1}^{\infty}$ of the kernel function $k(\mathbf{x}, \mathbf{x}')$ in decreasing order.

1. For some $C_p > 0, \alpha_p > 1, k$ is said to have a (C_p, α_p) polynomial eigendecay, if for all $j \in \mathbb{N}$, we have $\lambda_j \leqslant C_p j^{-\alpha_p}$. Examples include the Matérn kernel.
2. For some $C_{e,1}, C_{e,2}, \alpha_e > 0, k$ is said to have a $(C_{e,1}, C_{e,2}, \alpha_e)$ exponential eigendecay, if for all $j \in \mathbb{N}$, we have $\lambda_j \leqslant C_{e,1} \exp\left(-C_{e,2} j^{\alpha_e}\right)$. Examples include the radial basis function kernel.

Based on the discussions above, we finally present the proof of **Theorem 2.2**.

Proof. Under the polynomial eigendecay (C_p, α_p), we have

$$\begin{aligned}
\delta_{mD} &\leqslant m\bar{\mu}\phi^2\psi^2 \sum_{j=D+1}^{\infty} C_p j^{-\beta_p} \\
&\leqslant m\bar{\mu}\phi^2\psi^2 \int_{z=D}^{\infty} C_p z^{-\beta_p} \,\mathrm{d}z \\
&= m\bar{\mu} C_p D^{1-\beta_p} \psi^2 \phi^2.
\end{aligned}$$

We select

$$D = \left\lceil \left(\frac{\bar{\mu}\psi^2\phi^2 C_p T}{\log\left(1 + \frac{\bar{\bar{k}}\bar{k}T}{m\sigma_\epsilon^2}\right)\sigma_\epsilon^2} \right)^{1/\alpha_p} \right\rceil,$$

so that

$$\frac{\delta_{mD} T}{\sigma_\epsilon^2} \leqslant mD \log\left(1 + \frac{\bar{\bar{k}}\bar{k}T}{\sigma_\epsilon^2 m}\right).$$

In this way, we have that

$$\gamma_T \leqslant m \left(\left(\frac{\bar{\mu}\phi^2\psi^2 C_p T}{\sigma_\epsilon^2} \log^{-1}\left(1 + \frac{\bar{\bar{k}}\bar{k}T}{m\sigma_\epsilon^2}\right) \right)^{\frac{1}{\alpha_p}} + 1 \right) \log\left(1 + \frac{\bar{\bar{k}}\bar{k}T}{m\sigma_\epsilon^2}\right).$$

Under the exponential eigendecay $(C_{e,1}, C_{e,2}, \alpha_e)$, we have

$$\begin{aligned}\delta_{mD} &\leqslant m\bar{\mu}\phi^2\psi^2 \sum_{m=D+1}^{\infty} C_{e,1}\exp\left(-C_{e,2}m^{\alpha_e}\right)\\ &\leqslant m\bar{\mu}\phi^2\psi^2 \int_{z=D}^{\infty} C_{e,1}\exp\left(-C_{e,2}z^{\alpha_e}\right)\,\mathrm{d}z.\end{aligned}$$

When $\alpha_e = 1$,

$$\begin{aligned}\int_{z=D}^{\infty}\exp\left(-C_{e,2}z^{\alpha_e}\right)\,\mathrm{d}z &= \int_{z=D}^{\infty}\exp\left(-C_{e,2}z\right)\,\mathrm{d}z\\ &= \frac{1}{C_{e,2}}\exp\left(-C_{e,2}D\right).\end{aligned}$$

In this way, we select

$$D = \left\lceil \frac{1}{C_{e,2}}\log\left(\frac{C_{e,1}m\bar{\mu}\phi^2\psi^2 T}{\sigma_\epsilon^2 C_{e,2}}\right)\right\rceil,$$

so that $\delta_{mD}T/\sigma_\epsilon^2 \leqslant 1$.

When $\alpha_e \neq 1$,

$$\begin{aligned}\int_{z=D}^{\infty}\exp\left(-C_{e,2}z^{\alpha_e}\right)\,\mathrm{d}z &= \frac{1}{\alpha_e}\int_{z=D^{\alpha_e}}^{\infty} z^{\frac{1}{\alpha_e}-1}\exp\left(-C_{e,2}z\right)\,\mathrm{d}z\\ &= \frac{1}{\alpha_e}\int_{z=D^{\alpha_e}}^{\infty} z^{\frac{1}{\alpha_e}-1}\exp\left(-C_{e,2}\frac{z}{2}\right)\exp\left(-C_{e,2}\frac{z}{2}\right)\,\mathrm{d}z\\ &\leqslant \frac{1}{\alpha_e}\int_{z=D^{\alpha_e}}^{\infty}\left(\frac{2}{C_{e,2}}\left(\frac{1}{\alpha_e}-1\right)\right)^{\frac{1}{\alpha_e}-1}\exp\left(-\left(\frac{1}{\alpha_e}-1\right)\right)\exp\left(-C_{e,2}\frac{z}{2}\right)\,\mathrm{d}z\\ &= \frac{2}{C_{e,2}\alpha_e}\left(\frac{2}{C_{e,2}}\left(\frac{1}{\alpha_e}-1\right)\right)^{\frac{1}{\alpha_e}-1}\exp\left(-\left(\frac{1}{\alpha_e}-1\right)\right)\exp\left(-C_{e,2}\frac{D^{\alpha_e}}{2}\right).\end{aligned}$$

In this way, we select

$$D = \left\lceil \left(\frac{2}{C_{e,2}}\left(\log(T)+\log\left(\frac{2C_{e,1}m\bar{\mu}\phi^2\psi^2}{\sigma_\epsilon^2\alpha_e C_{e,2}}\right)+\left(\frac{1}{\alpha_e}-1\right)\left(\log\left(\frac{2}{C_{e,2}}\left(\frac{1}{\alpha_e}-1\right)\right)-1\right)\right)\right)^{\frac{1}{\alpha_e}}\right\rceil,$$

so that $\delta_{mD}T/\sigma_\epsilon^2 \leqslant 1$. Thus, whether $\alpha_e = 1$ or not, the second term in the upper bound of γ_T as in (A.5) is bounded by $1/2$, a constant. Therefore,

$$\gamma_T \leqslant mD\log\left(1+\frac{\bar{\bar{k}}\bar{k}T}{m\sigma_\epsilon^2}\right),$$

when T is sufficiently large. To summarize the results for the exponential eigendecay, we have

$$\gamma_T \leqslant m\left(\left(\frac{2}{C_{e,2}}\left(\log(T)+C_{\alpha_e}\right)\right)^{\frac{1}{\alpha_e}}+1\right)\log\left(1+\frac{\bar{\bar{k}}\bar{k}T}{m\sigma_\epsilon}^2\right),$$

where

$$C_{\alpha_e} = \log\left(\frac{C_{e,1} m\bar{\mu}\phi^2\psi^2}{\sigma_\epsilon^2 C_{e,2}}\right)$$

if $\alpha_e = 1$, and

$$C_{\alpha_e} = \log\left(\frac{2C_{e,1} m\bar{\mu}\phi^2\psi^2}{\sigma_\epsilon^2 \alpha_e C_{e,2}}\right) + \left(\frac{1}{\alpha_e} - 1\right)\left(\log\left(\frac{2}{C_{e,2}}\left(\frac{1}{\alpha_e} - 1\right)\right) - 1\right)$$

otherwise.

□

Proof of Theorem A.2

In this section, we present the detailed proof of **Theorem A.2**. For the following discussion, we assume that the conditions described in **Theorem A.2** are satisfied.

First, we begin with a lemma that indicates the neural network approximation error $\|\hat{\boldsymbol{g}}_t(\boldsymbol{\theta}) - \boldsymbol{g}(\boldsymbol{\theta})\|$ can be arbitrarily small, providing enough layers of nodes. Here $\hat{\boldsymbol{g}}_t(\boldsymbol{\theta})$ is the employed neural networks in the t-th round and $\boldsymbol{g}(\boldsymbol{\theta})$ is the ground-truth function, which is assume to be an $\boldsymbol{\alpha}$-Hölder function as in **Theorem A.2**.

Lemma A.9. *With properly chosen weight parameters, there exists a ReLU neural network, such that*

$$\sup_{\boldsymbol{\theta}\in\Theta} |\hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta}) - \boldsymbol{g}_l(\boldsymbol{\theta})| \leqslant e_t, \tag{A.6}$$

with a given $e_t \in (0,1)$. Here the neural network in the t-th iteration $\hat{\boldsymbol{g}}_t$ has (i) no more than $\mathcal{C}_2(1 - \log(e_t))$ layers and (ii) at most $\mathcal{C}_3 e_t^{-\frac{d'}{\boldsymbol{\alpha}}}(1 - \log(e_t))$ neurons and weight parameters, where $\mathcal{C}_2, \mathcal{C}_3$ are both constants.

Lemma A.9 follows directly from **Lemma 2** in [38]. For further discussions on neural network generalization error, we refer to [187].

Based on **Lemma A.9**, we then provide that, with a high probability, the error between the ground-truth NN-AGP $f(\mathbf{x};\boldsymbol{\theta})$ and the NN-AGP with the approximated neural network $\hat{f}_t(\mathbf{x};\boldsymbol{\theta})$ can be bounded

Lemma A.10. *With probability at least $1-\delta$, we have that*

$$\forall \mathbf{x} \in \mathcal{X}, \boldsymbol{\theta} \in \Theta, \left|\hat{f}_t(\mathbf{x};\boldsymbol{\theta}) - f(\mathbf{x};\boldsymbol{\theta})\right| \leqslant \mathcal{C}_1 e_t,$$

where $\mathcal{C}_1$ is a constant.

Proof. Note that,

$$\begin{aligned}\left|\hat{f}_t\left(\mathbf{x};\boldsymbol{\theta}\right)-f\left(\mathbf{x};\boldsymbol{\theta}\right)\right| &= \left|\sum_{q=1}^{Q}\left(\sum_{l=1}^{m}\left(\hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta})-\boldsymbol{g}_l(\boldsymbol{\theta})\right)a_{l,q}u_q\left(\mathbf{x}\right)\right)+\sum_{l=1}^{m}\left(\hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta})-\boldsymbol{g}_l(\boldsymbol{\theta})\right)v_l\left(\mathbf{x}\right)\right| \\ &\leqslant \left(\sum_{q=1}^{Q}\left(\sum_{l=1}^{m}\left|a_{l,q}\right|\left|u_q\left(\mathbf{x}\right)\right|\right)+\sum_{l=1}^{m}\left|v_l\left(\mathbf{x}\right)\right|\right)e_t.\end{aligned}$$

Recall that, based on the condition (2.6), all u_q's and v_l's are Lipschitz with probability at least $1-\delta$. This also implies that u_q's and v_l's are bounded as well, since $\mathcal{X}$ is compact. Thus, we denote

$$\tilde{C}_1 \doteq \sup_{\mathbf{x}\in\mathbf{X}}\left(\sum_{q=1}^{Q}\left(\sum_{l=1}^{m}\left|a_{l,q}\right|\left|u_q\left(\mathbf{x}\right)\right|\right)+\sum_{l=1}^{m}\left|v_l\left(\mathbf{x}\right)\right|\right)$$

and $\tilde{e}_t = \mathcal{C}_1 e_t$, and accomplish the proof. □

Next, we focus on the misspecified NN-AGP $\hat{f}_t(\mathbf{x};\boldsymbol{\theta}) = \hat{g}_t(\boldsymbol{\theta})^\top \mathbf{p}(\mathbf{x})$. Recall that, the ground-truth observations satisfy that

$$y_t = f(\mathbf{x}_t;\boldsymbol{\theta}_t)+\epsilon_t.$$

We then construct "virtual" observations as

$$\tilde{y}_t = \hat{f}_t(\mathbf{x}_t;\boldsymbol{\theta}_t)+\epsilon_t.$$

Consequently, we have that

$$|y_t-\tilde{y}_t| = \left|f(\mathbf{x};\boldsymbol{\theta})-\hat{f}_t(\mathbf{x};\boldsymbol{\theta})\right| \leqslant \tilde{e}_t$$

with high probability, based on **Lemma A.10**. Besides, suppose we are now in the t-th round. Then the inference of the reward function uses the "virtual" observations. Specifically, the posterior mean and covariance that is associated with the NN-AGP model in the t-th round $\hat{f}_t\left(\mathbf{x};\boldsymbol{\theta}_t\right)$ are denoted as

$$\begin{aligned}\tilde{\mu}_{t-1}\left(\mathbf{x};\boldsymbol{\theta}_t\right) =& \tilde{\tilde{\mathcal{K}}}_{(\mathbf{x};\boldsymbol{\theta}_t);t}^\top\left[\tilde{\tilde{\mathcal{K}}}_{(\mathcal{D}_{t-1});t}+\sigma_\epsilon^2 I_{t-1}\right]^{-1}\tilde{\mathbf{y}}_{t-1}, \\ \tilde{\sigma}_{t-1}^2\left(\mathbf{x};\boldsymbol{\theta}_t\right) =& \hat{g}_t(\boldsymbol{\theta}_t)^\top\mathcal{K}\left(\mathbf{x},\mathbf{x}\right)\hat{g}_t(\boldsymbol{\theta}_t)-\tilde{\tilde{\mathcal{K}}}_{(\mathbf{x};\boldsymbol{\theta}_t);t}^\top\left[\tilde{\tilde{\mathcal{K}}}_{(\mathcal{D}_{t-1});t}+\sigma_\epsilon^2 I_{t-1}\right]^{-1}\tilde{\tilde{\mathcal{K}}}_{(\mathbf{x};\boldsymbol{\theta}_t);t}.\end{aligned}$$

Compared with the quantities used in the acquisition function (A.3) in NN-AGP-UCB+, the difference lies in between the posterior means $\hat{\mu}_{t-1}$ and $\tilde{\mu}_{t-1}$, where the real/"virtual" observations are incorporated. Actually, we have the following lemma to quantify the difference.

Lemma A.11. *It is satisfied that,* $\forall t$

$$|\tilde{\mu}_t(\mathbf{x};\boldsymbol{\theta}_t) - \hat{\mu}_t(\mathbf{x};\boldsymbol{\theta}_t)| \leqslant \frac{\tilde{e}_t\sqrt{t}}{\sigma_\epsilon}\tilde{\sigma}_t(\mathbf{x};\boldsymbol{\theta}_t),$$

if $|y_t - \tilde{y}_t| \leqslant \tilde{e}_t$.

The proof of **Lemma A.11** is similar to **Lemma 2** in [22]. In addition, since $\tilde{\mu}_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$ and $\tilde{\sigma}^2_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$ denote the posterior mean and variance of $\hat{f}_t(\mathbf{x};\boldsymbol{\theta}_t)$, we could still employ the union bound and discretization technologies in **Lemma A.5** and **Lemma A.6**, and reach the following lemma.

Lemma A.12. *Choose* $\tilde{\beta}_t = 2\log(4\pi_t/\delta) + 2d\log\left(dt^2r\tilde{\tilde{M}}b\sqrt{\log\left(\frac{4ad}{\delta}\right)}\right)$, *we have*

$$\mathbb{P}\left\{\begin{array}{l}\forall t, \left|\hat{f}_t(\mathbf{x}_t^*;\boldsymbol{\theta}_t) - \tilde{\mu}_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t)\right| \leqslant \tilde{\beta}_t^{1/2}\tilde{\sigma}_{t-1}([\mathbf{x}_t^*]_t;\boldsymbol{\theta}_t) + \frac{1}{t^2} \\ \quad \left|\hat{f}_t(\mathbf{x}_t^*;\boldsymbol{\theta}_t) - \tilde{\mu}_{t-1}(\mathbf{x}_t^*;\boldsymbol{\theta}_t)\right| \leqslant \tilde{\beta}_t^{1/2}\tilde{\sigma}_{t-1}(\mathbf{x}_t^*;\boldsymbol{\theta}_t)\end{array}\right\} \geqslant 1-\delta,$$

Here, $\tilde{\tilde{M}} = \sup_{\boldsymbol{\theta}\in\Theta;t}\left\{\{|\sum_{l=1}^m \hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta})a_{l,q}|\}_{q=1}^Q, \{|\hat{\boldsymbol{g}}_{l;t}(\boldsymbol{\theta})|\}_{l=1}^m\right\}$ *and* $[\mathbf{x}]_t$ *denotes a point in* $\mathcal{X}_t$ *that is the closest to* $\mathbf{x}$.

The difference lies on that, in the previous proof, there is only one NN-AGP model $f(\mathbf{x};\boldsymbol{\theta})$ in all the iterations. However, when the approximation of the neural network is taken into consideration, the kernel function (therefore, the NN-AGP) changes in every iteration. Although the NN-AGP model changes every time, the union bound can be employed as well. In addition, the GP's u_q's and v_l's remain the same to be discretized. In this way, we reach **Lemma A.12**.

Next, we discuss the difference between $\tilde{\sigma}_t^2(\mathbf{x}_t;\boldsymbol{\theta}_t)$ and $\sigma_t^2(\mathbf{x}_t;\boldsymbol{\theta}_t)$. Specifically,

$$\begin{aligned}\tilde{\sigma}_t^2(\mathbf{x}_t;\boldsymbol{\theta}_t) - \sigma_t^2(\mathbf{x}_t;\boldsymbol{\theta}_t) =& \tilde{\tilde{K}}_t((\mathbf{x}_t;\boldsymbol{\theta}_t),(\mathbf{x}_t;\boldsymbol{\theta}_t)) - \tilde{\tilde{\mathcal{K}}}^\top_{(\mathbf{x}_t;\boldsymbol{\theta}_t);t}\left[\tilde{\tilde{\mathcal{K}}}_{(\mathcal{D}_t);t} + \sigma_\epsilon^2 I_t\right]^{-1}\tilde{\tilde{\mathcal{K}}}_{(\mathbf{x}_t;\boldsymbol{\theta}_t);t} \\ & - \tilde{K}((\mathbf{x}_t;\boldsymbol{\theta}_t),(\mathbf{x}_t;\boldsymbol{\theta}_t)) + \tilde{\mathcal{K}}^\top_{(\mathbf{x}_t;\boldsymbol{\theta}_t)}\left[\tilde{\mathcal{K}}_{\mathcal{D}_t} + \sigma_\epsilon^2 I_t\right]^{-1}\tilde{\mathcal{K}}_{(\mathbf{x}_t;\boldsymbol{\theta}_t)} \\ \leqslant& \tilde{\tilde{K}}_t((\mathbf{x}_t;\boldsymbol{\theta}_t),(\mathbf{x}_t;\boldsymbol{\theta}_t)) - \tilde{K}((\mathbf{x}_t;\boldsymbol{\theta}_t),(\mathbf{x}_t;\boldsymbol{\theta}_t)) \\ & + \tilde{\mathcal{K}}^\top_{(\mathbf{x}_t;\boldsymbol{\theta}_t)}\left[\tilde{\mathcal{K}}_{\mathcal{D}_t} + \sigma_\epsilon^2 I_t\right]^{-1}\tilde{\mathcal{K}}_{(\mathbf{x}_t;\boldsymbol{\theta}_t)}.\end{aligned}$$

In terms of the last term $\tilde{\mathcal{K}}^\top_{(\mathbf{x}_t;\boldsymbol{\theta}_t)}\left[\tilde{\mathcal{K}}_{\mathcal{D}_t} + \sigma_\epsilon^2 I_t\right]^{-1}\tilde{\mathcal{K}}_{(\mathbf{x}_t;\boldsymbol{\theta}_t)}$, we have that

$$\tilde{\mathcal{K}}^\top_{(\mathbf{x}_t;\boldsymbol{\theta}_t)}\left[\tilde{\mathcal{K}}_{\mathcal{D}_t} + \sigma_\epsilon^2 I_t\right]^{-1}\tilde{\mathcal{K}}_{(\mathbf{x}_t;\boldsymbol{\theta}_t)} \leqslant \frac{C\sqrt{t}}{\sigma_\epsilon}\sigma_t(\mathbf{x}_t;\boldsymbol{\theta}_t).$$

Here,

$$C = \left(\left(\sum_{q=1}^{Q} \sum_{l=1}^{m} a_{l,q}^2 \right) + 1 \right) \sup_{\boldsymbol{\theta} \in \Theta} \|g(\boldsymbol{\theta})\|_2^2$$

denotes the upper bound of $\tilde{K}\left(\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right), \left(\mathbf{x}_t; \boldsymbol{\theta}_t\right)\right)$. The proof of this inequality can be referred to **Theorem 2** in [44]. On the other, in terms of $\tilde{\hat{K}}_t\left(\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right), \left(\mathbf{x}_t; \boldsymbol{\theta}_t\right)\right) - \tilde{K}\left(\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right), \left(\mathbf{x}_t; \boldsymbol{\theta}_t\right)\right)$, the difference comes from the bias using neural networks $\hat{\boldsymbol{g}}_t$ to approximate the mapping $\boldsymbol{g}$. That is,

$$\begin{aligned}
&\tilde{\hat{K}}_t\left(\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right), \left(\mathbf{x}_t; \boldsymbol{\theta}_t\right)\right) - \tilde{K}\left(\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right), \left(\mathbf{x}_t; \boldsymbol{\theta}_t\right)\right) \\
\leqslant &\sum_{q=1}^{Q} \left(\hat{\boldsymbol{g}}_t\left(\boldsymbol{\theta}_t\right) - \boldsymbol{g}\left(\boldsymbol{\theta}_t\right)\right)^{\top} \mathbf{A}_q \left(\hat{\boldsymbol{g}}_t\left(\boldsymbol{\theta}_t\right) - \boldsymbol{g}\left(\boldsymbol{\theta}_t\right)\right) + \left\|\hat{\boldsymbol{g}}_t\left(\boldsymbol{\theta}_t\right) - \boldsymbol{g}\left(\boldsymbol{\theta}_t\right)\right\|^2 \\
\leqslant &\left(\left(\sum_{q=1}^{Q} \sum_{l=1}^{m} a_{l,q}^2 \right) + 1 \right) \left\|\hat{\boldsymbol{g}}_t\left(\boldsymbol{\theta}_t\right) - \boldsymbol{g}\left(\boldsymbol{\theta}_t\right)\right\|^2 \\
\leqslant &\left(\left(\sum_{q=1}^{Q} \sum_{l=1}^{m} a_{l,q}^2 \right) + 1 \right) m e_t^2.
\end{aligned}$$

In this way, by letting $\left(\left(\sum_{q=1}^{Q} \sum_{l=1}^{m} a_{l,q}^2 \right) + 1 \right) m e_t^2 \doteq e_t'$, we have

$$\tilde{\sigma}_t^2\left(\mathbf{x}_t; \theta_t\right) \leqslant \frac{C\sqrt{t}}{\sigma_\epsilon} \sigma_t\left(\mathbf{x}_t; \theta_t\right) + \sigma_t^2\left(\mathbf{x}_t; \theta_t\right) + e_t'.$$

Lastly, we present the proof of **Theorem A.2**.

Proof. First, we notice that

$$\mathcal{R}_T \leqslant \tilde{\mathcal{R}}_T + 2 \sum_{t=1}^{T} \tilde{e}_t,$$

since $\left| f\left(\mathbf{x}; \boldsymbol{\theta}\right) - \hat{f}_t\left(\mathbf{x}; \boldsymbol{\theta}\right) \right| \leqslant \tilde{e}_t$. Here $\tilde{\mathcal{R}}_T = \sum_{t=1}^{T} \tilde{r}_t$ denotes the "virtual" cumulative regrets based on the NN-AGP with neural network $\hat{\boldsymbol{g}}_t\left(\boldsymbol{\theta}\right)$. That is, $\mathcal{R}_T$ and $\tilde{\mathcal{R}}_T$ are in the same order of T, since $\sum_{t=1}^{\infty} \tilde{e}_t < \infty$.

Next, we focus on the "virtual" regret $\tilde{r}_t$. Specifically, based on **Lemma A.12**, with probability at least $1 - \delta$

$$\begin{aligned}
\tilde{r}_t &= \hat{f}_t\left(\mathbf{x}_t^*; \boldsymbol{\theta}_t\right) - \hat{f}_t\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right) \\
&\leqslant \tilde{\mu}_{t-1}\left(\left[\mathbf{x}_t^*\right]_t; \boldsymbol{\theta}_t\right) + \left(\tilde{\beta}_t^{1/2} + \frac{\tilde{e}_{\mathcal{N}} \sqrt{t}}{\sigma_\epsilon} \right) \tilde{\sigma}_{t-1}\left(\left[\mathbf{x}_t^*\right]_t; \boldsymbol{\theta}_t\right) + \frac{1}{t^2} - \hat{f}_t\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right) \\
&\leqslant \tilde{\mu}_{t-1}\left(\mathbf{x}_t^*; \boldsymbol{\theta}_t\right) + \left(\tilde{\beta}_t^{1/2} + \frac{\tilde{e}_{\mathcal{N}} \sqrt{t}}{\sigma_\epsilon} \right) \tilde{\sigma}_{t-1}\left(\mathbf{x}_t^*; \boldsymbol{\theta}_t\right) + \frac{1}{t^2} - \hat{f}_t\left(\mathbf{x}_t; \boldsymbol{\theta}_t\right) \\
&\leqslant 2 \left(\tilde{\beta}_t^{1/2} + \frac{\tilde{e}_{\mathcal{N}} \sqrt{t}}{\sigma_\epsilon} \right) \tilde{\sigma}_{t-1}\left(\mathbf{x}_t^*; \boldsymbol{\theta}_t\right) + \frac{1}{t^2}.
\end{aligned}$$

Note that

$$\begin{aligned}\sum_{t=1}^{T}\tilde{\sigma}_{t-1}\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) &\leqslant \sqrt{T\sum_{t=1}^{T}\tilde{\sigma}_{t-1}^2\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right)} \\ &\leqslant \sqrt{T\left(\frac{2C\gamma_T}{\log\left(1+C\sigma_\epsilon^{-2}\right)}+\frac{C^{\frac{3}{2}}T}{\sigma_\epsilon}\sqrt{\frac{2\gamma_T}{\log\left(1+C\sigma_\epsilon^{-2}\right)}}+\sum_{t=1}^{T}e_t'\right)} \\ &= \mathcal{O}\left(\sqrt{T\gamma_T}\right)+\mathcal{O}\left(T\left(\gamma_T\right)^{\frac{1}{4}}\right),\end{aligned}$$

where $\sum_{t=1}^{\infty}e_t'<\infty$. In addition, since $e_t=\mathcal{O}(\frac{1}{t^{1+\Delta}})$ and $\Delta>0$, $\tilde{e}_t\sqrt{t}$ is bounded by some constant, say $\mathcal{B}$. Thus,

$$\begin{aligned}\tilde{\mathcal{R}}_T &= \sum_{t=1}^{T}\tilde{r}_t \\ &\leqslant 2\left(\tilde{\beta}_T^{1/2}+\mathcal{B}\right)\sum_{t=1}^{T}\tilde{\sigma}_{t-1}\left(\mathbf{x}_t;\boldsymbol{\theta}_t\right) \\ &= \mathcal{O}\left(\sqrt{T\gamma_T\tilde{\beta}_T}\right)+\mathcal{O}\left(T\left(\gamma_T\right)^{\frac{1}{4}}\left(\tilde{\beta}_T\right)^{\frac{1}{2}}\right).\end{aligned}$$

□

Proof of Consistency

In this section, we discuss the consistency of training the NN-AGP model from the data. The consistency requires specific conditions on the sampling strategies of $\mathbf{x}_t$ and $\boldsymbol{\theta}_t$, which might be difficult to verify during the contextual GP bandits. However, we still present it as a sanity check here for two reasons. First, we note that the NN-AGP model can also be employed in supervised learning tasks when the function to be approximated involves both user-selected inputs $\mathbf{x}$ and observed contextual variables $\boldsymbol{\theta}$. Examples of these tasks can be found in [94]. In these task, the NN-AGP model still adopts the advantage of explicit kernel expression with respect to user-selected inputs $\mathbf{x}$ and approximation accuracy with respect to observed contextual variables $\boldsymbol{\theta}$. Meanwhile, the required conditions for consistency can be satisfied in these tasks. Second, existing theoretical results on GP bandits (as well as **Theorem 2.1** in our work) largely assume that the surrogate model is well-specified and does not require updating from the observations. That is, the discussion on the consistency of training NN-AGP from the data does not conflict with existing theoretical results on GP bandits.

To begin with, we first set up the notation used in the proof. Recall that the training objective of NN-AGP is to maximize the likelihood function, which is in the form of

$$L_t\left(\mathbf{W},\Phi,\sigma_\epsilon^2\right)=-\ln\left[(2\pi)^{t/2}\right]-\frac{1}{2}\ln\left[\left|\tilde{K}_{\mathcal{D}_t}+\sigma_\epsilon^2 I_t\right|\right]-\frac{1}{2}\mathbf{y}_t^\top\left[\tilde{K}_{\mathcal{D}_t}+\sigma_\epsilon^2 I_t\right]^{-1}\mathbf{y}_t.$$

It is equivalent to considering the optimization problem

$$\left(\hat{\mathbf{W}}_t, \hat{\Phi}_t, \hat{\sigma^2_{\epsilon;t}}\right) = \arg\min_{(\mathbf{W},\Phi,\sigma^2_\epsilon)} \ell_t\left(\mathbf{W}, \Phi, \sigma^2_\epsilon\right),$$

where $\ell_t\left(\mathbf{W}, \Phi, \sigma^2_\epsilon\right) = \frac{1}{t}\left(\ln\left[\left|\tilde{K}_{\mathcal{D}_t} + \sigma^2_\epsilon I_t\right|\right] + \mathbf{y}_t^\top\left[\tilde{K}_{\mathcal{D}_t} + \sigma^2_\epsilon I_t\right]^{-1}\mathbf{y}_t\right)$. For ease of notation, we let $\mathbf{K}_t = \tilde{K}_{\mathcal{D}_t} + \sigma^2_\epsilon I_t$ and $\mathbf{K}^*_t$ denotes the covariance matrix with the ground-truth parameter plug-in. In addition, for the parameters involved in the MGP Φ, we separate them as Φ_K to denote the parameters in the kernel function and the weight parameters $a = \{a_l\}_{l=1}^m$. We generally denote $\tilde{\Phi} = (\Phi_K, \mathbf{W}, a, \sigma^2_\epsilon)$. In the proof, the ground truth parameters or the quantities that are with ground truth parameters will be indicated by a superscript "$*$". We will also sometimes hide the subscript "t" that indicates the iteration for ease of notation.

Assumption A.1. *We assume that*

1. *The set of parameters to be optimized $\tilde{\Phi} \in S_{\tilde{\Phi}}$ is a compact set. Especially, $\sigma^2_\epsilon \in [\sigma^2_a, \sigma^2_b]$ and $\sigma^2_a > 0$. The ground-truth parameters are contained in the set of parameters to be optimized. That is, $\tilde{\Phi}^* \in S_{\tilde{\Phi}}$.*

2. *The kernel function of $u(\mathbf{x})$ is a stationary kernel function. That is, $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$. In addition, it is also satisfied that*

$$\max_{s=0,1,2,3}\max_{j_1,\ldots,j_s}\sup_{\tilde{\Phi}\in S_{\tilde{\Phi}}}\left|\frac{\partial^s}{\partial\tilde{\Phi}_{j_1},\ldots,\partial\tilde{\Phi}_{j_s}}k\left(\|\mathbf{x} - \mathbf{x}'\|\right)\right| \leqslant \frac{C_{sup}}{1 + \|\mathbf{x} - \mathbf{x}'\|^{d+C_{inf}}}$$

   for some positive fixed constants C_{inf} and C_{sup}.

3. *The sampling strategy satisfies that , there exists a fixed constant Δ*

$$\inf_{\substack{\tau,\tau'\in\mathbb{N}\\ \tau\neq\tau'}}\|\mathbf{x}_\tau - \mathbf{x}_{\tau'}\| \geqslant \Delta$$

   for the decision variable; $g_l(\boldsymbol{\theta}_t) = \mathcal{O}(1/t)$ and $\frac{\partial}{\partial\mathbf{W}_j}g_l(\boldsymbol{\theta}) = \mathcal{O}(1/t)$ for the contextual variable with any $\mathbf{W}$ involved with the neural network $g(\boldsymbol{\theta})$.

4. *The ground-truth parameters are well-separated from other potential values of parameters. That is, for $\forall\epsilon > 0$*

$$\liminf_{t\to\infty}\inf_{\substack{\tilde{\Phi}\in S_{\tilde{\Phi}}\\ \|\tilde{\Phi}-\tilde{\Phi}^*\|\geqslant\epsilon}}\frac{1}{t}\sum_{\tau,\tau'=1}^{t}\left(\tilde{K}\left(\left(\mathbf{x}_\tau, \boldsymbol{\theta}_\tau\right), \left(\mathbf{x}_{\tau'}, \boldsymbol{\theta}_{\tau'}\right)\right) - \tilde{K}^*\left(\left(\mathbf{x}_\tau, \boldsymbol{\theta}_\tau\right), \left(\mathbf{x}_{\tau'}, \boldsymbol{\theta}_{\tau'}\right)\right) + \delta_{\tau\tau'}\left(\sigma^2_\epsilon - \sigma^{2*}_\epsilon\right)\right)^2 > 0.$$

Theorem A.4 (Consistency of Learning NN-AGP)**.** *Under **Assumption A.1**, the training of the NN-AGP through (2.5) is consistent. That is,*

$$\lim_{t\to\infty}\left(\hat{\mathbf{W}}_t,\hat{\Phi}_t,\hat{\sigma^2_{\epsilon;t}}\right)\xrightarrow{P}\left(\mathbf{W}^*,\Phi^*,\sigma_\epsilon^{2^*}\right).$$

Here, $\left(\mathbf{W}^*,\Phi^*,\sigma_\epsilon^{2^*}\right)$ denotes the ground-truth values of the parameters in the NN-AGP model and the noise, and "$\xrightarrow{P}$" denotes the convergence in probability.

Proof. Note that, the ground-truth parameters minimize the mean value of the loss function $\ell_t\left(\mathbf{W},\Phi,\sigma_\epsilon^2\right)$. That is,

$$\left(\mathbf{W}^*,\Phi^*,\sigma_\epsilon^{2^*}\right)=\arg\min_{(\mathbf{W},\Phi,\sigma_\epsilon^2)}\mathbb{E}\left\{\ell_t\left(\mathbf{W},\Phi,\sigma_\epsilon^2\right)\right\}.$$

Therefore, in order to prove the consistency, we require the uniform convergence of the likelihood function, that is

$$\lim_{t\to\infty}\sup_{(\mathbf{W},\Phi,\sigma_\epsilon^2)}\left|\ell_t\left(\mathbf{W},\Phi,\sigma_\epsilon^2\right)-\mathbb{E}\left\{\ell_t\left(\mathbf{W},\Phi,\sigma_\epsilon^2\right)\right\}\right|\xrightarrow{P}0. \tag{A.7}$$

To begin with, we first establish the point-wise convergence of the loss function.

$$\begin{aligned}\mathrm{Var}\left\{\ell_t\left(\mathbf{W},\Phi,\sigma_\epsilon^2\right)\right\}=&\frac{1}{t^2}\mathrm{Var}\left\{\mathbf{y}_t^\top\mathbf{K}_t^{-1}\mathbf{y}_t\right\}\\ =&\frac{2}{t^2}\mathrm{Tr}\left\{K_t^{-1}K_t^*K_t^{-1}K_t^*\right\}.\end{aligned}$$

For $\forall t$, the maximum eigenvalue of the matrix K_t^* is bounded as

$$\begin{aligned}\lambda_{sup}\left\{K_t^*\right\}&\leqslant\lambda_{sup}\left\{\tilde{K}_{D_t}^*\right\}+\sigma_\epsilon^{2^*}\\ &\leqslant\max_{\tau=1,\dots,t}\sum_{\tau'=1}^{t}\left|k^*\left(\mathbf{x}_\tau,\mathbf{x}_{\tau'}\right)\right|\left|\tilde{k}\left(\boldsymbol{\theta}_\tau,\boldsymbol{\theta}_{\tau'}\right)\right|+\sigma_\epsilon^{2^*}\\ &\leqslant\max_{\tau=1,\dots,t}\sum_{\tau'=1}^{t}\frac{C_{sup}}{1+\left\|\mathbf{x}_\tau-\mathbf{x}_{\tau'}\right\|^{d+C_{inf}}}\tilde{\mathcal{K}}+\sigma_\epsilon^{2^*},\end{aligned} \tag{A.8}$$

where $\tilde{\mathcal{K}}$ is the upper bound of $\left|\tilde{k}\left(\boldsymbol{\theta}_\tau,\boldsymbol{\theta}_{\tau'}\right)\right|$. The second inequality comes from the Gershgorin circle theorem [97]. Based on condition 2 and condition 3 in **Assumption A.1**, there exists a constant A_1 such that $\lambda_{sup}\left\{K_t^*\right\}\leqslant A_1$; see aslo [12]. On the other hand, $\lambda_{sup}\left\{K_t^{-1}\right\}=\left(\lambda_{inf}\left\{K_t\right\}\right)^{-1}\leqslant\left(\sigma_a^2\right)^{-1}=A_2$, for $\forall t,\tilde{\Phi}$. Thus, we have

$$\mathrm{Var}\left\{\ell_t\left(\mathbf{W},\Phi,\sigma_\epsilon^2\right)\right\}\leqslant\frac{2A_1^2A_2^2}{t}.$$

Thus, we have the point-wise convergence of $\ell_t(\mathbf{W}, \Phi, \sigma_\epsilon^2)$ to its mean value. Next, we show that the convergence is uniform. To prove the uniform convergence, we consider the gradient of $\ell_t(\tilde{\Phi})$. We let $\tilde{K}_{D_t} = K_{\Phi,t} \odot K_{\mathbf{W},t}$, where "$\odot$" denotes the Hadamard product of two matrices. $K_{\Phi,t}$ and $K_{\mathbf{W},t}$ are the covariance matrix associated with $k(\mathbf{x}, \mathbf{x}')$ and $\tilde{k}(\boldsymbol{\theta}, \boldsymbol{\theta}')$. In this way,

$$\frac{\partial \ell_t\left(\tilde{\Phi}\right)}{\partial \tilde{\Phi}_j} = \frac{1}{t}\operatorname{tr}\left\{K_t^{-1}\frac{\partial K_t}{\partial \tilde{\Phi}_j}\right\} - \frac{1}{t}\mathbf{y}_t^\top K_t^{-1}\frac{\partial K_t}{\partial \tilde{\Phi}_j}K_t^{-1}\mathbf{y}_t.$$

In terms of the gradient, we specifically have

$$\begin{aligned}
\frac{\partial K_t}{\partial \Phi_{K;j}} &= \frac{\partial K_{\Phi,t}}{\partial \Phi_{K;j}} \odot K_{\mathbf{W},t} + \sigma_\epsilon^2 I_t \\
\frac{\partial K_t}{\partial \mathbf{W}_j} &= K_{\Phi,t} \odot 2 \begin{pmatrix} \left(\frac{\partial}{\mathbf{w}_j}\boldsymbol{g}(\boldsymbol{\theta}_1)\right)^\top a \\ \vdots \\ \left(\frac{\partial}{\mathbf{w}_j}\boldsymbol{g}(\boldsymbol{\theta}_t)\right)^\top a \end{pmatrix} \left(\boldsymbol{g}(\boldsymbol{\theta}_1)^\top a, \ldots, \boldsymbol{g}(\boldsymbol{\theta}_t)^\top a\right) + \sigma_\epsilon^2 I_t \\
\frac{\partial K_t}{\partial a_j} &= K_{\Phi,t} \odot 2 \begin{pmatrix} \boldsymbol{g}_j(\boldsymbol{\theta}_1) \\ \vdots \\ \boldsymbol{g}_j(\boldsymbol{\theta}_t) \end{pmatrix} \left(\boldsymbol{g}(\boldsymbol{\theta}_1)^\top a, \ldots, \boldsymbol{g}(\boldsymbol{\theta}_t)^\top a\right) + \sigma_\epsilon^2 I_t \\
\frac{\partial K_t}{\partial \sigma_\epsilon^2} &= I_t.
\end{aligned}$$

We want to show that there exists a constant A_3 that bounds the singular value of the gradient of K_t such that

$$\rho_{sup}\left(\frac{\partial K_t}{\partial \tilde{\Phi}_j}\right) \leqslant A_3, \, \forall t.$$

Note that, given any two matrices K_1 and K_2, the singular values satisfy that

$$\rho_{sup}(K_1 \odot K_2) \leqslant \rho_{sup}(K_1)\,\rho_{sup}(K_2)$$

and

$$\rho_{sup}(K_1 + K_2) \leqslant \rho_{sup}(K_1) + \rho_{sup}(K_2),$$

see [126].

Specifically, $\rho_{sup}\left\{\frac{\partial K_t}{\partial \Phi_{K;j}}\right\}$ is bounded by a constant, based on condition 2, as is proved in [12] with a similar argument of (A.8). $\rho_{sup}\{K_{\Phi;t}\}$ is bounded by a constant with a similar argument as in (A.8) as well. In addition, $\sum_{\tau=1}^{\infty}\left[\boldsymbol{g}(\boldsymbol{\theta}_\tau)^\top a\right]^2 < \infty$,

$$\sum_{\tau=1}^{\infty}\left[\left(\frac{\partial}{\mathbf{w}_j}\boldsymbol{g}(\boldsymbol{\theta}_\tau)\right)^\top a\right]\left[\boldsymbol{g}(\boldsymbol{\theta}_\tau)^\top a\right] < \infty,$$

and $\sum_{\tau=1}^{\infty} \boldsymbol{g}_l(\boldsymbol{\theta}_\tau) \left[\boldsymbol{g}(\boldsymbol{\theta}_\tau)^\top a\right] < \infty$ based on the condition that $\boldsymbol{g}(\boldsymbol{\theta}_t) = \mathcal{O}(1/t)$ and $\frac{\partial}{\partial \mathbf{W}_j} \boldsymbol{g}_l(\boldsymbol{\theta}) = \mathcal{O}(1/t)$ in **Assumption A.1**, which further bounds the maximum singular value of matrices $K_{\mathbf{W},t}$, $\begin{pmatrix} \left(\frac{\partial}{\mathbf{w}_j} \boldsymbol{g}(\boldsymbol{\theta}_1)\right)^\top a \\ \vdots \\ \left(\frac{\partial}{\mathbf{w}_j} \boldsymbol{g}(\boldsymbol{\theta}_t)\right)^\top a \end{pmatrix} \left(\boldsymbol{g}(\boldsymbol{\theta}_1)^\top a, \ldots, \boldsymbol{g}(\boldsymbol{\theta}_t)^\top a\right)$ and

$$\begin{pmatrix} \boldsymbol{g}_j(\boldsymbol{\theta}_1) \\ \vdots \\ \boldsymbol{g}_j(\boldsymbol{\theta}_t) \end{pmatrix} \left(\boldsymbol{g}(\boldsymbol{\theta}_1)^\top a, \ldots, \boldsymbol{g}(\boldsymbol{\theta}_t)^\top a\right).$$

In this way, we have that $\rho_{sup}\left(\frac{\partial K_t}{\partial \tilde{\Phi}_j}\right) \leqslant A_3$, $\forall t$. Thus,

$$\max_j \sup_{\tilde{\Phi} \in S_{\tilde{\Phi}}} \left| \frac{\partial \ell_t\left(\tilde{\Phi}\right)}{\partial \tilde{\Phi}_j} \right| \leqslant A_2 A_3 + A_2^2 A_3 \frac{\mathbf{y}_t^\top \mathbf{y}_t}{t} = \mathcal{O}_p(1) \tag{A.9}$$

since $\frac{\mathbf{y}_t^\top \mathbf{y}_t}{t}$ is a non-negative random variable with bounded expectation. With a similar argument, we also have

$$\max_j \sup_{\tilde{\Phi} \in S_{\tilde{\Phi}}} \left| \frac{\partial \mathbb{E}\left\{\ell_t\left(\tilde{\Phi}\right)\right\}}{\partial \tilde{\Phi}_j} \right| = \mathcal{O}(1). \tag{A.10}$$

Because of (A.9) and (A.10), along with the point-wise convergence, we attain the uniform convergence of the loss function $\ell_t(\tilde{\Phi})$; see [131] for detailed discussions.

To guarantee consistency of the learning procedure, we also require that the ground-truth parameters can be specified when minimizing the loss function. It can be verified that, there exists a constant A_4

$$\begin{aligned} &\mathbb{E}\left\{\ell_t\left(\tilde{\Phi}\right)\right\} - \mathbb{E}\left\{\ell_t\left(\tilde{\Phi}^*\right)\right\} \\ \geqslant &A_4 \frac{1}{t} \sum_{\tau,\tau'=1}^{t} \left(\tilde{K}\left((\mathbf{x}_\tau, \boldsymbol{\theta}_\tau), (\mathbf{x}_{\tau'}, \boldsymbol{\theta}_{\tau'})\right) - \tilde{K}^*\left((\mathbf{x}_\tau, \boldsymbol{\theta}_\tau), (\mathbf{x}_{\tau'}, \boldsymbol{\theta}_{\tau'})\right) + \delta_{\tau\tau'}\left(\sigma_\epsilon^2 - \sigma_\epsilon^{2*}\right)\right)^2 \end{aligned}$$

where the detailed proof can be found in [13]. In this way, based on condition 4 in **Assumption A.1**, for $\forall \epsilon > 0$,

$$\liminf_{t\to\infty} \inf_{\substack{\tilde{\Phi} \in S_{\tilde{\Phi}} \\ \left\|\tilde{\Phi} - \tilde{\Phi}^*\right\| \geqslant \epsilon}} \mathbb{E}\left\{\ell_t\left(\tilde{\Phi}\right)\right\} - \mathbb{E}\left\{\ell_t\left(\tilde{\Phi}^*\right)\right\} \geqslant A_5,$$

for some constant A_5. Thus, along with the uniform consistency of the loss function (A.7), we attain the consistency of the training procedure (2.5), which follows a regular argument on consistency of M-estimation; see [166].

□

A.4 Experimental Details & Additional Experiments

Experimental details

In this section, we describe the experiment settings in the main context in detail. In terms of the training of NN-AGP through (2.5) and maximizing the likelihood function of a joint GP, we apply the alternating direction method of multipliers (ADMM) with a learning rate 10^{-4}; see [28]. All the experiments are based on running PyTorch and Python 3.8 on Nvidia GeForce RTX 3090 (GPU) with 24GB of RAM. An implementation is provided at `https://github.com/Oceanjinghai/NN-AGP-UCB`.

Synthetic Reward

In terms of the joint GP model, we consider both additive kernels and multiplicative kernels, of which each separate kernel is the radial basis function (RBF) kernel. In terms of the NN-AGP model, we select $m = 2, 3, 5$. Besides, we select the ICM model with the RBF kernel as the MGP component ($Q = 1$) and an FCN with 2 hidden layers with 64 and 32 nodes. The parameters of the MGP ($a_{l,q}$'s) are updated through learning the NN-AGP model in (2.5).

In addition, both NeuralUCB [202] and NN-UCB [105] are designed for contextual bandits with K arms. To adapt them into the problem we consider in Section 2.2, we take $\mathbf{z} = (\boldsymbol{\theta}, \mathbf{x})$ as a joint input to the neural networks representing the arm. We discretize the joint space $\Theta \times \mathcal{X}$ with 10 points in each dimension with equal distances. The best arm is selected with some of the dimension fixed by $\boldsymbol{\theta}_t$. In terms of NN's used in both algorithms, we select an FCN with 3 hidden layers of 64, 32, and 32 nodes.

Queuing Problem with Time Sequence Contextual Variables

We consider a discrete-time queuing problem, where decision-making in each time period is required. In each epoch, a contextual variable $\boldsymbol{\theta}_t$ is first revealed to the agent. In some application scenarios, the contextual variable might includes traffic conditions and weather conditions that affect the arrival process of the queuing system. The number of customers who will come to the queue, denoted as N_t is drawn from a Poisson distribution Poisson(u_t). Here $u_t = \exp\left(\sum_{\tau=1}^{t} a^\top \boldsymbol{\theta}_\tau\right)$, and $a \sim \mathcal{N}(\mathbf{1}/\mathbf{4}, \frac{1}{4} I_{d'})$ is the weight generated and fixed in advance. In this way, the number of customers who will come to the queue in this round depends on the entire sequence of contextual variables.

On the other hand, the decision variable is composed of two parts $p_{1;t}$ and $p_{2;t}$, denoting the service price and the service rate respectively. The service price indicates the reward that completes serving a customer. On the other hand, a customer comes to queue, sees the service price, and then determines to join the queue with the probability of $p(p_{1;t})$, where $p(\,\cdot\,)$ is a decreasing function with respect to $p_{1;t}$. In addition, in each iteration, the number of service completion $N'(t)$ is a Poisson random variable with the mean of the service rate $p_{2;t}$, that is $N'(t) \sim$ Poisson$(p_{2;t})$. The higher the service rate, the higher the service cost will be. After each iteration, the customers who decide to join the queue and do not receive

the service will leave as well, resulting in a penalty. In this way, the observed reward in each iteration is

$$y_t = p_{1;t} \max\{N(t), N'(t)\} - c_1 p_{2;t} - c_2 \max\{N(t) - N'(t), 0\},$$

where on the right-hand side the first term is the reward of completing service, the second term is the service cost and the last term denotes the penalty of not satisfying customers. Such decision problems in a queuing system are also considered in [41]. For the NN-AGP model, we select the long short-term memory (LSTM) [91] neural network to approximate the mapping $\boldsymbol{g}_t(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t)$. The training of LSTM (as well as MGP) is accomplished by maximizing the likelihood function. Since we do not have the ground-truth value of the expected maximum reward, we instead record the cumulative rewards to compare the performance of different methods.

Specifically, in terms of the experiment results contained in the main text, we set $c_1 = 0.5$ and $c_2 = 0.3$. In terms of the NN-AGP model, we select $m = 2, 3, 5$. Besides, we select the ICM model with the RBF kernel as the MGP component and an LSTM with 1) sequence length = 10; 2) hidden size = 64; 3) projection size = m; 4) batch size = 1. We utilize the implementation from https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html. Besides, we select the ICM model with the RBF kernel as the MGP component ($Q = 1$). For CGP-UCB, we utilize the RBF kernel for the scenario when we only utilize the current contextual variable. We also apply the Wasserstein subsequence kernel [21] that is specifically designed for time series, of which the implementation can be found at

https://github.com/BorgwardtLab/WTK.

Pricing with a Diffusion Network

We consider a pricing problem with a diffusion network, which is explored in [122]. Specifically, we represent the network at time t as a graph $\boldsymbol{\theta}_t = (V_t, E_t)$, where $V_t := \{1, 2 \ldots, |V_t|\}$ is the set of all users (nodes) and $E_t := \{1, 2 \ldots, |E_t|\}$ is the set of all directed edges. A directed edge $(i, j) \in E_t$, where $i, j \in V$, implies that user i is influenced by user j, and we call j an in-neighbor of i. We use $\mathcal{N}_{i;t}$ to denote the set of all in-neighbors for agent i at time t and $n_{i;t} := |\mathcal{N}_{i;t}|$ to denote her in-degree (i.e., the number of in-neighbors).

In each iteration, the user $i \in V$ will decide whether to adopt the service based on her realized utility in period $t : Y_i(t) := \mathbb{I}\{u_i(t) \geq 0\} \in \{0, 1\}$, where $u_i(t)$ is the utility of user i to adopt the service in period t, and is defined as

$$u_i(t) = v_i - \alpha \mathbf{x}_t + \beta \cdot \frac{\sum_{j \in \mathcal{N}_{i;t}} Y_j(t-1)}{n_{i;t}} + \epsilon_i(t).$$

Here $\mathbf{x}_t$ is the service price (decision variable); v_i denotes the user (node) preference while α and β are intrinsic network parameters; and ϵ_t is the i.i.d. Gaussian noise. In each iteration, the graph structure $\boldsymbol{\theta}_t = (V_t, E_t)$ is presented to the agent to determine a price $\mathbf{x}_t$. In this

experiment, we consider maximizing the total profit brought by users' service adoption in the network, and therefore the reward function is

$$f(\mathbf{x}_t;\theta) = \mathbf{x}_t \times \mathbb{E}\left[\sum_i Y_i(t;\theta)\right],$$

where x_t denotes the price of the service. An increase in prices is likely to have a negative impact on the adoption rate of service.

For the NN-AGP model, we select the graph convolutional neural network (GCN) [149] to approximate the mapping $\boldsymbol{g}(\boldsymbol{\theta})$. The training of GCN (as well as MGP) is accomplished by maximizing the likelihood function. Since we do not have the ground-truth value of the expected maximum reward, we instead record the cumulative rewards to compare the performance of different methods.

In terms of the experiment results in the main text, we let $\mathcal{X} = [0, 30]$ and $\boldsymbol{\theta}_t$ represents an undirected graph with 5 and 10 nodes where each edge exists with probability 1/2. Besides, we set $\alpha = \beta = 1$ and $v_i = 3$ for each node. In terms of the NN-AGP model, we select $m = 3$. Besides, we select the ICM model with the RBF kernel as the MGP component and a GCN with convolution size = 3. We utilize the implementation from https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html. Besides, we select the ICM model with the RBF kernel as the MGP component ($Q = 1$). For CGP-UCB, we utilize the RBF kernel for the vectorized contextual variable. We also apply the Gaussian RBF kernel between vertex histogram [92] that is specifically designed for graphs. The experimental results indicate that our NN-AGP-UCB has a greater advantage than CGP-UCB when the contextual variable exhibits more complexity (with more nodes).

Additional Experiments

Computational Time

In this section, we record the computational time of the algorithms. We record 1) the training time that constructs the surrogate model based on the historical data and 2) the execution time that selects the decision variable after the contextual variable is revealed. We record the time (seconds) for exactly one round in the 50-th, 100-th, and 300-th rounds. We take the first set of experiments in Section 4.1 as an example and present the results in **Table A.1**.

We notice that CGP-UCB is the most efficient in training time since it employs a pre-specified GP model which does not update during iterations. The training procedure of CGP-UCB only requires matrix operations, which can be implemented efficiently. On the other hand, all the algorithms that involve NN require learning NN from data and longer training time than CGP-UCB. In terms of the execution time, NN-AGP-UCB requires similar time as CGP-UCB, since the selection of the decision variable of NN-AGP-UCB is based on GP as well. We also note that both NN-UCB and NeuralUCB are initially designed for

	50-th round	100-th round	300-th round
NN-AGP-UCB (m=2)	0.07/0.25	0.10/0.81	0.27/1.31
NN-AGP-UCB (m=5)	0.11/0.29	0.14/0.89	0.35/1.42
CGP-UCB (additive kernel)	0.01/0.26	0.02/0.77	0.04/1.24
NN-UCB	0.14/2.28	0.35/4.13	0.40/6.51
NeuralUCB	0.11/1.64	0.23/3.62	0.37/5.45

Table A.1: The mean of training time/ execution time of bandit algorithms associated with the first set of experiments in Section 4.1.

finite selections of decision variables. Thus, the computational cost of the execution time is largely due to searching for the optimal decision variable from the discretized feasible set. In addition, we consider sparse NN-AGP to alleviate the computational burden for future work; see also a discussion in Section A.5.

Sensitivity on Reward Function Structure

As suggested by [111], commonly selected composite kernel functions of the joint Gaussian process in CGP-UCB are additive kernels and multiplicative kernels. In this section, we show through experiments that the performance of CGP-UCB is sensitive on whether the form of the composite kernel is consistent with the structure of the reward function, while our NN-AGP-UCB achieves acceptable performance through the experiments.

Specifically, we consider two synthetic reward functions in the form of

$$R_3(\mathbf{x}, \boldsymbol{\theta}) = \sin\left(\|\mathbf{x}\|_2\right) \left|\cos\left(\|\boldsymbol{\theta}\|_2\right)\right|;$$
$$R_4(\mathbf{x}, \boldsymbol{\theta}) = \sin\left(\|\mathbf{x}\|_2\right) + \cos\left(\|\boldsymbol{\theta}\|_2\right).$$

That is, $R_3(R_4)$ is a multiplicative(additive) function with contextual/decision variables. We consider $d = 2$ and $d' = 3$. We let $\mathcal{X} = [-\sqrt{2}, \sqrt{2}]^2$ and $\Theta = [-1, 1]^3$. In terms of the joint GP model, we consider both additive kernels and multiplicative kernels, of which each separate kernel is the radial basis function (RBF) kernel. In terms of the NN-AGP model, we select $m = 3$. Besides, we select the ICM model with the RBF kernel as the MGP component and an FCN with 2 hidden layers of 64 and 32 nodes.

The experiment results (mean performance of 15 times experiments) are contained in **Figure A.1** and **Figure A.2**, which provide the insights as follows. First, for both reward functions, the NN-AGP-UCB approach does not outperform the classical CGP-UCB approach in initial iterations, since the neural networks require sufficient data to learn. Second,

as the size of the data increases, NN-CGP-UCB outperforms CGP-UCB in both experiments, owing to the strong approximation power of FCN. Last, the performance of the CGP-UCB is sensitive to the form of the composite kernels and the structure of reward functions. That is, for reward $R_3(\mathbf{x}, \boldsymbol{\theta})$ with a multiplicative structure, CGP-UCB with the multiplicative kernel outperforms CGP-UCB with the additive kernel, while for reward $R_4(\mathbf{x}, \boldsymbol{\theta})$ with an additive structure, CGP-UCB with the additive kernel outperforms CGP-UCB with the multiplicative kernel. On the other hand, the performance of the NN-AGP-UCB remains acceptable and outperforms baseline approaches no matter the structure of the reward function.

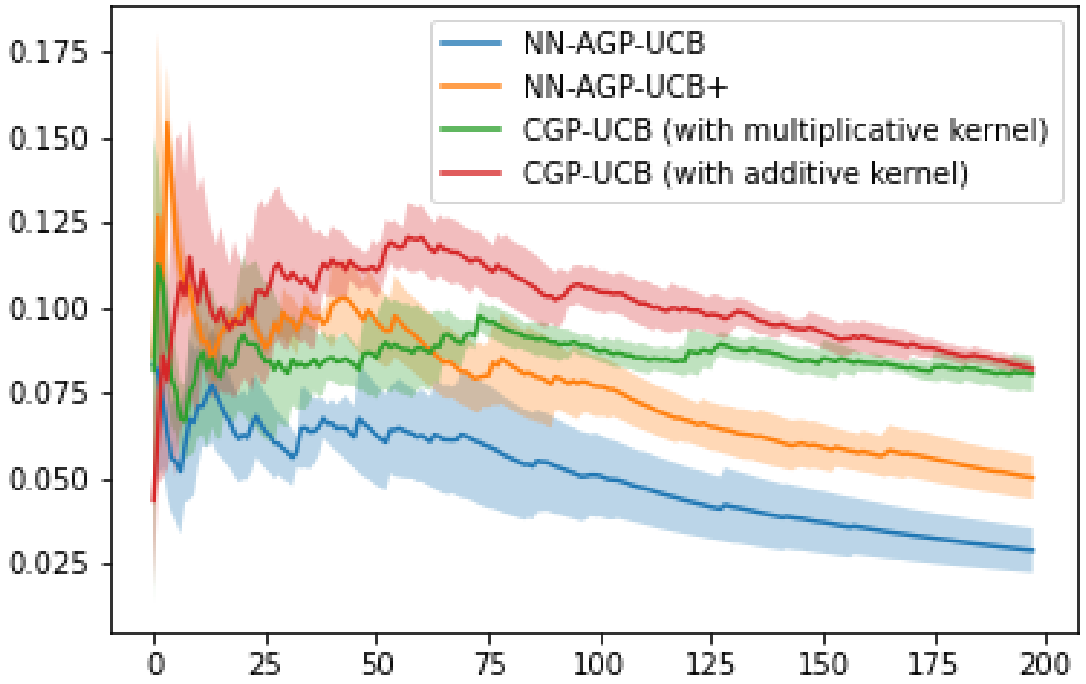

Figure A.1: Average regrets of NN-AGP-UCB and CGP-UCB with multiplicative reward function $R_3(\mathbf{x}; \boldsymbol{\theta})$.

Advantage with higher-dimensional contextual variables

In the previous section, we present the experimental results when $d = 1$ and $d' = 3$. Here, in terms of the additive reward function R_4, we increase the dimension of the contextual variable d' and present the results (mean performance of 15 times experiments) in **Figure**

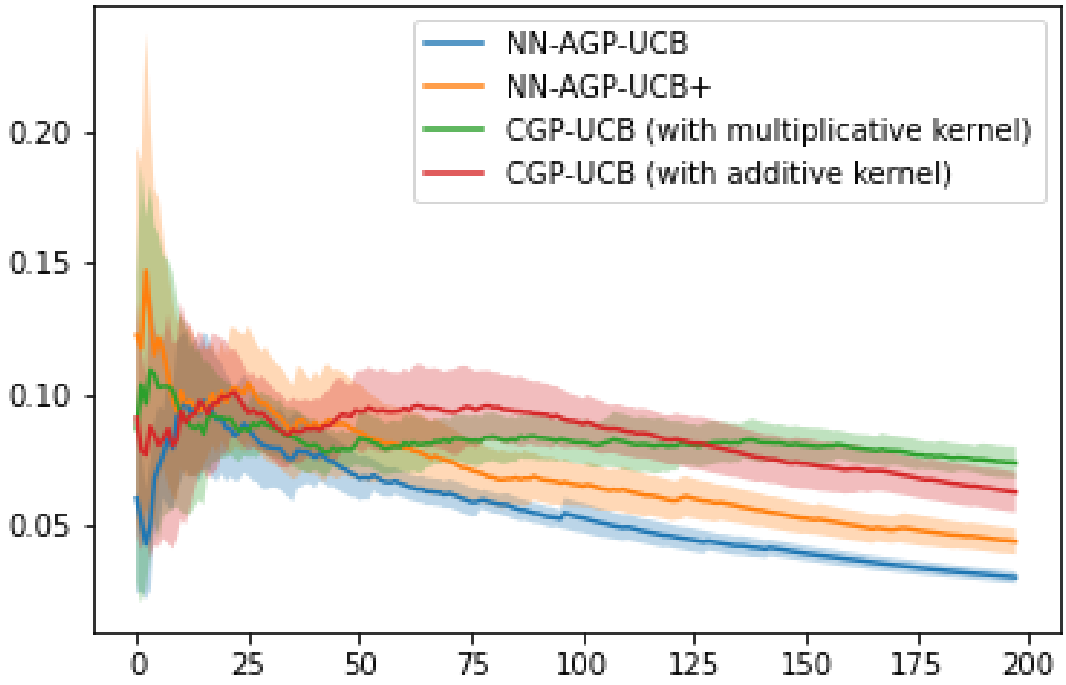

Figure A.2: Average regrets of NN-AGP-UCB and CGP-UCB with additive reward function $R_4(\mathbf{x}; \boldsymbol{\theta})$ when $d' = 3$.

A.2, **Figure A.3** and **Figure A.4**. Experimental results indicate that the superiority of our NN-AGP-UCB becomes more significant as the dimension of the contextual variable increases, considering the larger gaps (the scale of vertical axis in each figure is different) between the average regrets.

Moreover, we also contain the results of NN-AGP-UCB+ in **Figure A.1**, **Figure A.2**, **Figure A.3** and **Figure A.4**. The experimental results indicate that NN-AGP-UCB+ does not generally outperform NN-AGP-UCB, since NN-AGP-UCB+ is overly-conservative. On the other hand, NN-AGP-UCB+ still outperforms the baseline CGP-UCB with both multiplicative/additive kernels.

In addition, we also conduct additional experiments with higher-dimensional contextual variables, while the reward function exhibits a sparse structure. We consider that the observed contextual variable $\boldsymbol{\theta}$ are randomly selected with equal probability from $\Theta = [-1/2, 1/2]^{50}$. That is $d' = 50$. Meanwhile, we select the reward function

$$\tilde{R}_3(\mathbf{x}, \boldsymbol{\theta}) = 2 \sin\left(\|\mathbf{x}\|_2\right) \left|\cos\left(\|\boldsymbol{\theta}_{\text{eff}}\|_2\right)\right|$$

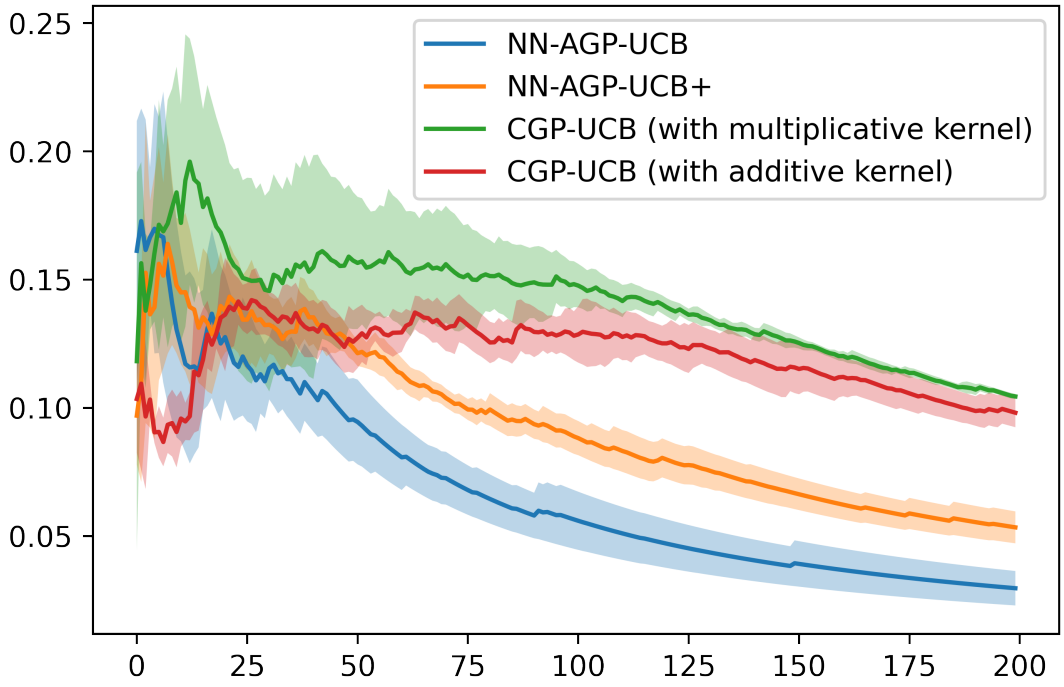

Figure A.3: Average regrets of NN-AGP-UCB and CGP-UCB with additive reward function $R_4(\mathbf{x}; \boldsymbol{\theta})$ when $d' = 6$.

as a sparse version of the reward function $R_3(\mathbf{x}, \boldsymbol{\theta})$. Here, $\mathbf{x} \in [-\sqrt{2}, \sqrt{2}]^2$ and $\boldsymbol{\theta}_{\text{eff}}$ denotes the first 20 dimensions of $\boldsymbol{\theta}$. That is, the remaining 30 dimensions of $\boldsymbol{\theta}$ will not affect the reward function while the user does not know. The experimental results are contained in **Figure A.5**. The results on average regret indicate the superiority of our approach in high-dimensional scenarios, even when the dimension of the NN-AGP model $m \ll d'$. That is to say, by utilizing the neural network, the NN-AGP model effectively extracts the information from the contextual variable and propagates it to the MGP component.

Regression Tasks with Complex Functions

As is discussed in the main context, the NN-AGP model inherits the strong approximation power from neural networks, which leads to the better performance on contextual GP bandit problems. To support this intuition, we conduct experiments to compare the prediction performance of NN-AGP and a joint GP. That is, we select in advance all the points to be samples and attain the observations. We then use these observations to train both NN-AGP

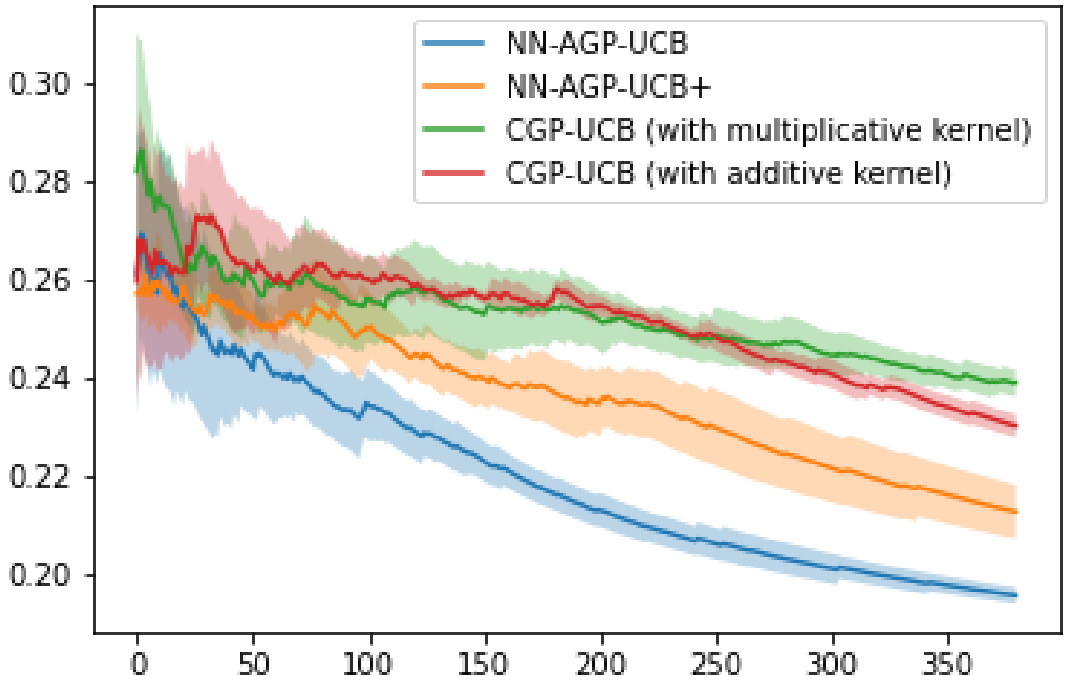

Figure A.4: Average regrets of NN-AGP-UCB and CGP-UCB with additive reward function $R_4(\mathbf{x}; \boldsymbol{\theta})$ when $d' = 9$.

and a joint GP. For both models, the prediction value of the unknown function is the posterior mean. Here, we select the Ackley function [2] as a representative to be approximated

$$f\left(\mathbf{x}; \boldsymbol{\theta} = (a, b, c)\right) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}\mathbf{x}_{(i)}^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(c\mathbf{x}_{(i)}\right)\right) + a + \exp(1).$$

A plot of the Ackley function is contained in **Figure A.6**. We let $\mathcal{X} = [-32.768, 32.768]^2$. In terms of the contextual variable, we set $a \in [15, 25], b \in [0.15, 0.25]$ and $c \in [1.5\pi, 2.5\pi]$. In terms of the joint GP model, we consider both additive kernels and multiplicative kernels, of which each separate kernel is the radial basis function (RBF) kernel. In terms of the NN-AGP model, we select $m = 3$. Besides, we select the ICM model with the RBF kernel as the MGP component ($Q = 1$) and an FCN with 2 hidden layers with 64 and 32 nodes.

We present the experimental results in **Figure A.7**. The experiments are performed 15 times and the experimental results indicate that our NN-AGP model achieves a better

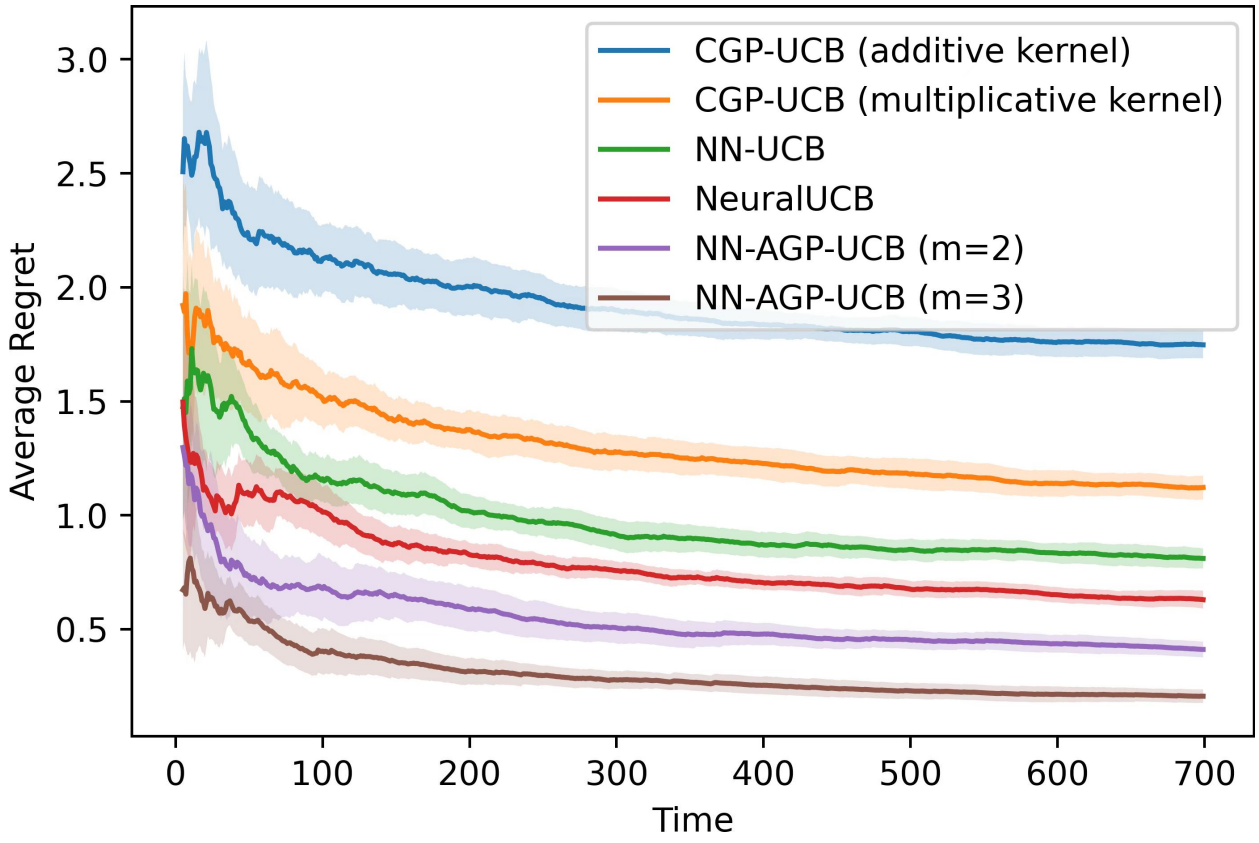

Figure A.5: Average regrets of NN-AGP and baseline approaches with the high-dimensional reward function $\tilde{R}_3(\mathbf{x}; \boldsymbol{\theta})$.

performance on the approximation accuracy, which is quantified by rooted mean square error (RMSE) as well as the corresponding standard deviation.

Since the NN-AGP model achieves a better performance in approximating the highly-nonstructural function, we would expect that NN-AGP-UCB would also achieve a better performance in contextual GP bandits when the reward function is highly-nonstructural as well.

Air-quality Monitoring Sites

In this set of experiments, we consider sequentially selecting the site that will record the worst air-quality, among multiple air-quality monitoring sites. That is, each $\mathbf{x}$ denotes an air-quality monitoring site and $|\mathcal{X}|$ denotes the number of sites. We use the data collected by Beijing Municipal Environmental Monitoring Center[2]; see also [200]. The data set includes

[2]https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data.

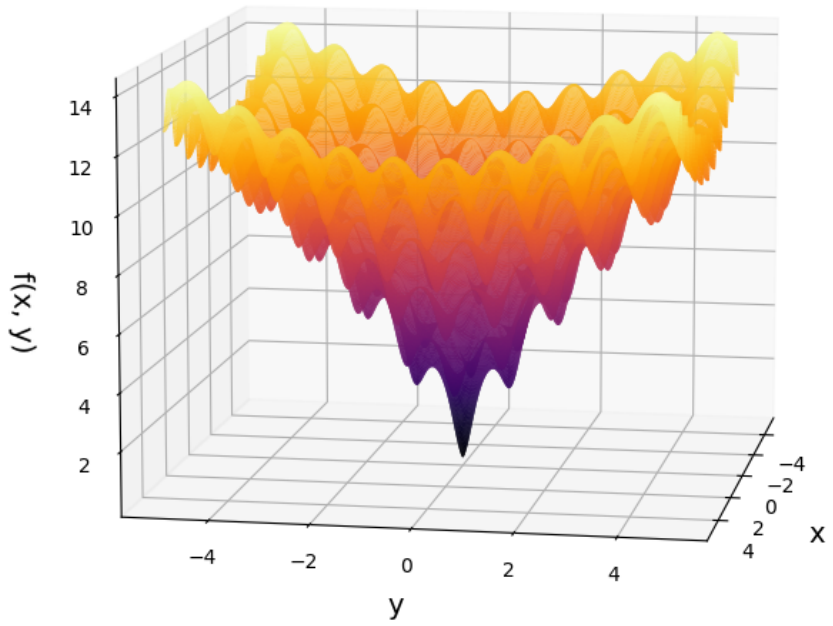

Figure A.6: An Ackley function $f(x, y) = -20 \exp\left(-0.2\sqrt{0.5\,(x^2 + y^2)}\right) - \exp(0.5(\cos(2\pi x) + \cos(2\pi y))) + e + 20$.

hourly air pollutants data from 12 nationally-controlled air-quality monitoring sites ($|\mathcal{X}| = 12$). The time period is from March 1st, 2013 to February 28th, 2017. The recorded quantities in each iteration include

- PM2.5: PM2.5 concentration (ug/m^3)
- PM10: PM10 concentration (ug/m^3)
- SO2: SO2 concentration (ug/m^3)
- NO2: NO2 concentration (ug/m^3)
- CO: CO concentration (ug/mm^3)
- O3: O3 concentration (ug/m^3)

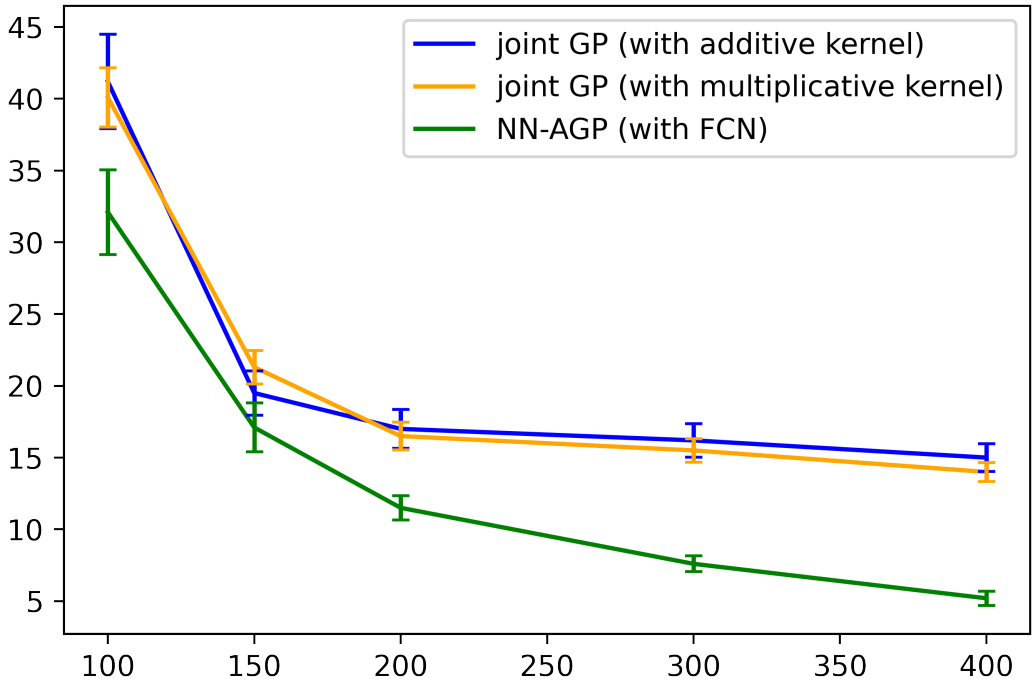

Figure A.7: RMSE's of NN-AGP and joint GP with Ackley function.

- TEMP: temperature (degree Celsius)
- PRES: pressure (hPa)
- DEWP: dew point temperature (degree Celsius)
- RAIN: precipitation (mm)
- wd: wind direction
- WSPM: wind speed (m/s).

In our experiment, we regard PM 2.5 as the unknown reward and the remaining quantities as the observed contextual variables in each round, that is $d' = 11$. That is, we would like to select the site that records the largest PM 2.5. In terms of the decision variable, we simulate an $\mathbf{x}^{(i)} \sim \text{Unif}(0, 1)$ for $i = 1, 2, \ldots, |\mathcal{X}|$, and use this generated random number to represent the site in all rounds. That is, $\mathcal{X} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(|\mathcal{X}|)}\right\}$. Since PM 2.5 in all the sites is

contained in the data set, the regret in each round is then the maximum PM 2.5 minus the PM 2.5 recorded in the selected site.

The setting of the approaches (NN-AGP-UCB, CGP-UCB, NN-UCB and NeuralUCB) is consistent with that in Section A.4. The experiment results are presented in **Figure A.8**. The experiments are performed 15 times. Although we use a same data set, the uncertainty comes from randomly selecting the decision variables for initialization. The experimental results indicate that our approach is applicable to real-world applications when the selection of the decision variable is finite, and outperforms existing approaches. We also note that, all the compared approaches exhibit fluctuation at the same time since the air-quality is influenced by human factors in certain period.

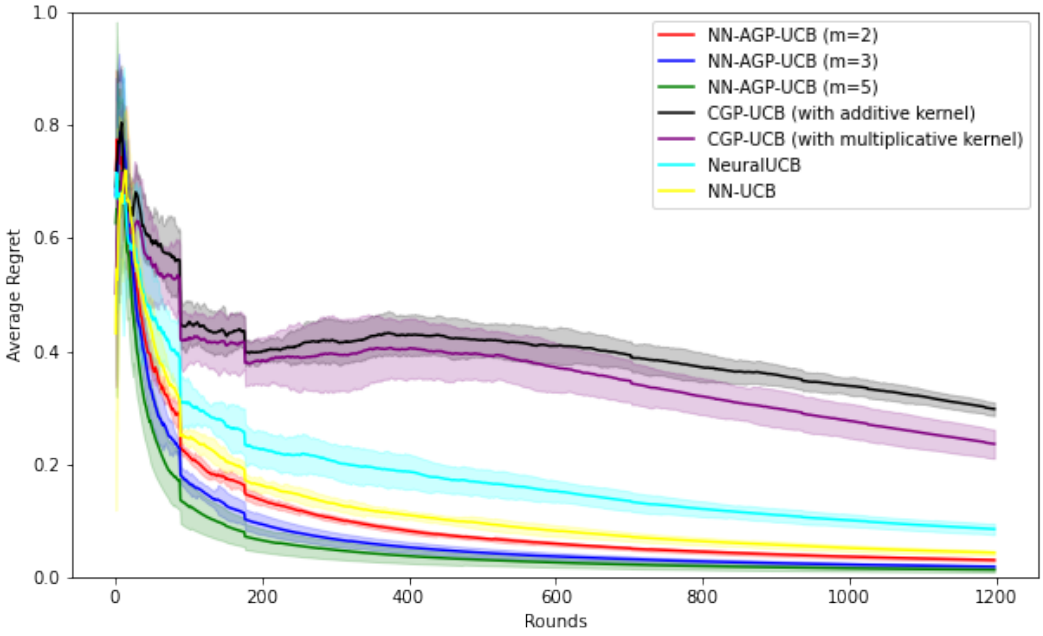

Figure A.8: Average regrets of NN-AGP-UCB and baseline approaches with the real-data collected from air-quality monitoring sites.

A.5 Limitations & Future work

In this section, we describe the limitations of NN-AGP, and propose potential future work to address these limitations.

Sparse NN-AGP

As is widely known, the Gaussian process model suffers from a computational complexity of $\mathcal{O}(t^3)$ (t denotes the sample size of data) and the NN-AGP model encounters the same challenge as well, because of the GP expression with respect to the decision variable. In this way, we briefly introduce the sparse NN-AGP model, which alleviates the computational burdens of the NN-AGP. A more detailed discussion on this direction will be contained in future work.

In terms of the MGP in the NN-AGP, we specifically consider the scenario when $Q = 1$ and there is no v_l's. Denote $\mathbf{p} = (\mathbf{p}(\mathbf{x}_1), \mathbf{p}(\mathbf{x}_2), \ldots, \mathbf{p}(\mathbf{x}_N))^\top$. In addition,

$$\mathbf{u} = (\mathbf{p}(\mathbf{z}_1), \mathbf{p}(\mathbf{z}_2), \ldots, \mathbf{p}(\mathbf{z}_M))^\top$$

are inducing points of the MGP on pseudo-inputs $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M$. In this way,

$$\begin{pmatrix} \mathbf{p} \\ \mathbf{u} \end{pmatrix} \sim \mathcal{MN}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{K_{pp}} & \mathbf{K_{up}^\top} \\ \mathbf{K_{up}} & \mathbf{K_{uu}} \end{pmatrix}, \mathbf{A}\right).$$

That is, the matrix $\begin{pmatrix} \mathbf{p} \\ \mathbf{u} \end{pmatrix}$ is sampled from a matrix-variate normal distribution; see [86]. Here $\mathbf{K}_{(\cdot,\cdot)}$ is the covariance matrix generated by the kernel function $k(\mathbf{x}, \mathbf{x}')$. Meanwhile, $\mathbf{A} = \mathbf{a}\mathbf{a}^\top$, where $\mathbf{a} = (a_1, a_2, \ldots, a_m)$.

Remark A.1. *A matrix-valued random element* $\mathbf{X} \sim \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V})$ *is equivalent with that*

$$\mathrm{vec}\{\mathbf{X}\} \sim \mathcal{N}(\mathrm{vec}\{\mathbf{M}\}, \mathbf{V} \otimes \mathbf{U}),$$

where $\mathrm{vec}\{\cdot\}$ *is the "vectorize" operator. Both* $\mathbf{U}$ *and* $\mathbf{V}$ *serve as the covariance matrix, where* $\mathbf{U}$ *captures the covariance among rows (samples) while* $\mathbf{V}$ *captures that among columns (dimensions). Meanwhile, for the p.d.f. of* $\mathbf{X}$*, we denote is as* $p(\mathbf{X}) = \mathcal{MN}(\mathbf{X} \mid \mathbf{M}, \mathbf{U}, \mathbf{V})$*. This notation is also used for the multivariate Gaussian case.*

Define $\psi_{\mathbf{u}}(\mathbf{x}) = \mathbf{K_{uu}^{-1}} k_{\mathbf{u}}(\mathbf{x})$, where $k_{\mathbf{u}}(\mathbf{x})$ denotes the covariance vector between $\mathbf{p}(\mathbf{x})$ and $\mathbf{u}$. Meanwhile, let

$$\mathbf{\Phi} = (\psi_{\mathbf{u}}(\mathbf{x}_1), \psi_{\mathbf{u}}(\mathbf{x}_2), \ldots, \psi_{\mathbf{u}}(\mathbf{x}_N)).$$

We then have

$$p(\mathbf{p} \mid \mathbf{u}) = \mathcal{MN}\left(\mathbf{p} \mid \mathbf{\Psi}^\top \mathbf{u}, \mathbf{K_{pp}} - \mathbf{\Psi}^\top \mathbf{K_{uu}} \mathbf{\Psi}, \mathbf{A}\right).$$

Supppose that a variational prior is imposed on the inducing points as

$$q_v(\mathbf{u}) = \mathcal{MN}\left(\mathbf{u} \mid \mathbf{B}, \mathbf{L}\mathbf{L}^\top, \mathbf{A}\right).$$

The selection of $\mathbf{B}$ and $\mathbf{L}$ is postponed. Based on the conditional distribution $p(\mathbf{p} \mid \mathbf{u})$, we have the joint variational distribution $q_v(\mathbf{p}, \mathbf{u}) = p(\mathbf{p} \mid \mathbf{u}) q_v(\mathbf{u})$. By marginalizing, we have

$$q_v(\mathbf{p}) = \mathcal{MN}\left(\mathbf{p} \mid \mathbf{\Psi}^\top \mathbf{B}, \mathbf{K_{pp}} - \mathbf{\Psi}^\top \left(\mathbf{K_{uu}} - \mathbf{L}\mathbf{L}^\top\right) \mathbf{\Psi}, \mathbf{A}\right) \tag{A.11}$$

With the variational distribution of $q_v(\mathbf{p})$, the inference of the MGP at any new point $\mathbf{x}^*$ with a given $\boldsymbol{\theta}$ is

$$\hat{f}(\mathbf{x}^*;\boldsymbol{\theta}) \sim \mathcal{N}\left(\boldsymbol{g}(\boldsymbol{\theta})^\top \psi_{\mathbf{u}}(\mathbf{x}^*)^\top \mathbf{B}, \boldsymbol{g}(\boldsymbol{\theta})^\top \left(k(\mathbf{x}^*,\mathbf{x}^*) - \psi_{\mathbf{u}}(\mathbf{x}^*)^\top \left(\mathbf{K_{uu}} - \mathbf{L}\mathbf{L}^\top\right)\psi_{\mathbf{u}}(\mathbf{x}^*)\right)\mathbf{A}\boldsymbol{g}(\boldsymbol{\theta})\right).$$

In this way, the inference at a new point $(\boldsymbol{\theta}, \mathbf{x}^*)$ requires the computational complexity of $\mathcal{O}(tM^2)$ instead of $\mathcal{O}(t^3)$. Next, we describe the procedure of deciding the parameters of the variational prior $q_v(\mathbf{u})$, that is $(\mathbf{B}, \mathbf{L})$, and the location of inducing points $\mathbf{Z}$. The selection is through the variational inference approach, which minizes the kullback-leibler(KL)-divergence between $q_v(\mathbf{p},\mathbf{u})$ and $p(\mathbf{p},\mathbf{u} \mid \mathbf{y})$. Specifically

$$\begin{aligned}
\mathrm{KL}\left[q_v(\mathbf{p},\mathbf{u}) \| p(\mathbf{p},\mathbf{u} \mid \mathbf{y})\right] &= \mathbb{E}_{q_v(\mathbf{p},\mathbf{u})}\left[\log \frac{q_v(\mathbf{p},\mathbf{u})}{p(\mathbf{p},\mathbf{u} \mid \mathbf{y})}\right] \\
&= \log p(\mathbf{y}) + \mathbb{E}_{q_v(\mathbf{p},\mathbf{u})}\left[\log \frac{q_v(\mathbf{p},\mathbf{u})}{p(\mathbf{p},\mathbf{u},\mathbf{y})}\right] \\
&= \log p(\mathbf{y}) - \mathrm{ELBO}(\boldsymbol{v}, \mathbf{Z}),
\end{aligned}$$

where the evidence lower bound (ELBO) is defined as

$$\mathrm{ELBO}(v, \mathbf{Z}) \triangleq \mathbb{E}_{q_v(\mathbf{p},\mathbf{u})}\left[\log \frac{p(\mathbf{p},\mathbf{u},\mathbf{y})}{q_v(\mathbf{p},\mathbf{u})}\right].$$

Since $\log p(\mathbf{y})$ is fixed and not affected by the variational distribution, minimizing the KL-divergence is then equivalent with maximizing ELBO, which is further decomposed as

$$\mathrm{ELBO}(v, \mathbf{Z}) = \mathbb{E}_{q_v(\mathbf{p})}[\log p(\mathbf{y} \mid \mathbf{p})] - \mathrm{KL}\left[q_v(\mathbf{u}) \| p(\mathbf{u})\right].$$

Note that

$$\log p(\mathbf{y} \mid \mathbf{p}) = \log p(\mathbf{y} \mid \mathbf{p}) = -\frac{N}{2}\log\left(2\pi\sigma_\epsilon^2\right) - \frac{1}{2\sigma_\epsilon^2}(\mathbf{y}-\mathbf{f})^\top(\mathbf{y}-\mathbf{f}),$$

where

$$\mathbf{f} = \left(\boldsymbol{g}(\boldsymbol{\theta}_1)^\top \mathbf{p}(\mathbf{x}_1), \boldsymbol{g}(\boldsymbol{\theta}_2)^\top \mathbf{p}(\mathbf{x}_2), \ldots, \boldsymbol{g}(\boldsymbol{\theta}_N)^\top \mathbf{p}(\mathbf{x}_N)\right)^\top$$

is not a linear function with respect to $\mathbf{p}$. Thus, we employ the Markov chain Monte Carlo method to evaluate $\mathbb{E}_{q_v(\mathbf{p})}[\log p(\mathbf{y} \mid \mathbf{p})]$. In addition, the KL-divergence adopts a closed-form expression as

$$\mathrm{KL}\left[q_v(\mathbf{u}) \| p(\mathbf{u})\right] = \frac{1}{2}\left(\mathrm{vec}\left\{\mathbf{B}\right\}^\top \mathrm{vec}\left\{\mathbf{K_{uu}^{-1}}\mathbf{B}\mathbf{A}^{-1}\right\} + m\,\mathrm{tr}\left\{\mathbf{K_{uu}^{-1}}\mathbf{L}\mathbf{L}^\top\right\} - m\ln\frac{\left|\mathbf{L}\mathbf{L}^\top\right|}{\left|\mathbf{K_{uu}}\right|} - Mm\right).$$

In this way, the stochastic gradient descent method can be employed to maximize ELBO. We note that the locations of the inducing points $\mathbf{u}$ can also be optimized as well. For more

detailed discussions on the sparse Gaussian process, we refer to [162, 90]. In addition, we compare sparse NN-AGP with sparse joint GP with contextual/decision variables. In terms of NN-AGP, the sparsity is built on a GP where the input dimension is d. In comparison, for the joint GP, the sparsity is built on a GP where the input dimension is $d + d'$. Intuitively, sparse NN-AGP requires fewer inducing points to achieve a prescribed accuracy, which alleviates the computational complexity. We admit that further discussions are required in future work.

Transfer Learning with NN-AGP

It is widely accepted that incorporating NN into bandit problems generally requires sufficient data to approximate the unknown reward function. Thus, the cold-start issue is brought to, in principle, all bandit algorithms that uses NN. Compared with the algorithms that fully rely on NN (e.g., NeuralUCB and NN-UCB), our NN-AGP-UCB actually suffers less from the cold-start issue, which is supported by numerical results in Section 4.1. The reason is that, in existing NN-based bandit algorithms, NN is responsible for approximating the entire reward function. In comparison, for the NN-AGP model, NN is focused to only be used for approximating the mapping from the contextual variable to the reward function and the approximation regarding the decision variable is supported by GP. It has been widely accepted that GP generally requires less data than NN in practical applications, and therefore NN-AGP helps to ease the cold-start issue.

Moreover, to further address the cold-start issue brought to NN-AGP, the transfer learning technology [173] can be incorporated into the bandit algorithm. We conduct numerical experiments and present the results in **Figure A.9**. Specifically, we consider an unknown reward function f_T and we also have access to functions $f_s, s = 1, 2, \ldots, 5$ that have a similar structure with f_T. We first sample each f_s for 50 or 100 rounds and learn an NN-AGP model with these samples. The NN component in NN-AGP helps to transfer the knowledge from f_s to f_T. That is, during the initial rounds of NN-AGP-UCB with f_T, we first fix the input layer of the pretrained NN and update the remaining layers with the new data, which is a widely-used transfer learning method named freezing.

Experimental results indicate that transfer learning from similar tasks helps to address the cold-start issue, and NN-AGP-UCB with/without transfer learning will converge to the similar regrets as the round increases. We also not that, to the best of our knowledge, there has not been sufficient work on transfer learning with NN-based bandit algorithms. These NN-based bandit algorithms largely rely on the neural tangent kernel (NTK) to address the exploration-exploitation trade-off when selecting the decision variable. However, it remains an open question on how to transfer the knowledge between different domains with NTK. In comparison, the exploration-exploitation trade-off in NN-AGP-UCB is supported by GP, and existing transfer learning technologies with NN can be easily adapted into our algorithm. Other methodologies for addressing cold-start in learning NN in an online setting can also be employed; see [172, 176].

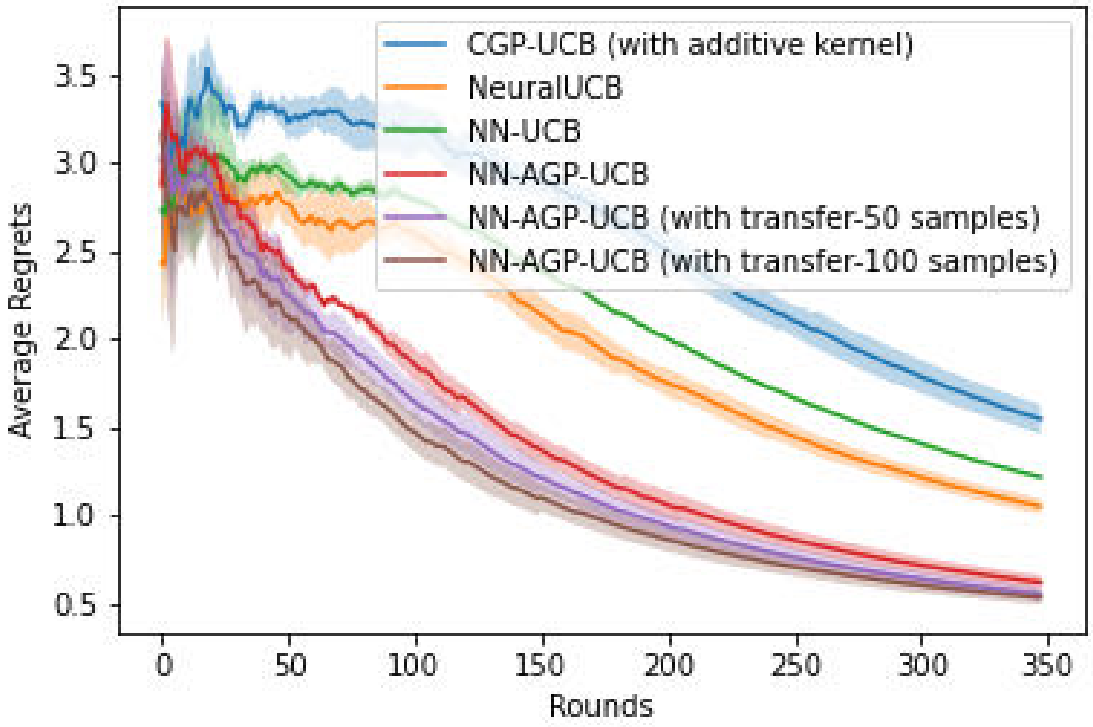

Figure A.9: Average regrets of $f_T(\mathbf{x};\boldsymbol{\theta}) = \exp\{\cos(\|\mathbf{x}\|_2) + \sin(\|\boldsymbol{\theta}\|_2)\}$ with $\mathcal{X} = [-1,1]^2$ and $\Theta = [-1,1]^3$. In each similar task, samples are generated by $f_s(\mathbf{x};\boldsymbol{\theta}) = \exp\{\cos(\|\mathbf{x}\|_2) + k_s \sin(\|\boldsymbol{\theta}\|_2)\}$, where k_s is randomly selected from $\{1, 2, \ldots, 10\}$ with equal probability for $s = 1, 2, \ldots, 5$.

Appendix B

Supplements for Chapter 3

B.1 Score Function

In this section, we describe the score functions that are used to compare the similarity between the generated texts and the baseline text. The score function $h(x, y)$ compares the distance between two texts x and y. By regarding the text y as the baseline, a higher value of h indicates a higher similarity between two texts and therefore a higher score for x. Here we provide two examples of methods to compare similarities between texts: 1) cosine similarity and 2) Word2Vec [45]. For a detailed overview of methods used to compare texts, we refer to [34].

Cosine Similarity

Cosine similarity is a measure used to gauge the cosine of the angle between two non-zero vectors in an inner product space. This metric is particularly useful in the field of text analysis for comparing the similarity between documents or text data. The detailed procedure involves these steps:

1. **Vector Representation of Text:** First, each text document is converted into a vector. Each dimension of this vector represents a unique term (word) from the text. If a term occurs in the text, its value in the vector is non-zero. Examples of the representation include:

    - **Term Frequency (TF):** The most commonly used representation for the non-zero value is TF, which stands for the number of times a term appears in the text;
    - **Term Frequency-Inverse Document Frequency (TF-IDF):** TF is adjusted by the inverse document frequency (IDF) to down-weight terms that appear frequently across the texts. The TF-IDF value increases proportionally to the num-

ber of times a word appears in the document but is offset by the frequency of the word in the corpus.

In this way, the two texts to be compared are transformed into vectors with the same dimensionality, say $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^n$.

2. **Cosine Similarity between Vectors:** After we attain the two vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, the similarity between the two vectors is represented by the cosine similarity between two vectors as
$$\cos(\theta) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}},$$
where A_i, B_i denote the i-th entry of $\boldsymbol{A}, \boldsymbol{B}$, $\boldsymbol{A} \cdot \boldsymbol{B}$ is the dot product of vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\|\boldsymbol{A}\|$ and $\|\boldsymbol{B}\|$ are the Euclidean norms of the vectors.

Word2Vec

Word2Vec is a technique in natural language processing (NLP) used to learn word embeddings, which are vector representations of words. Unlike simpler bag-of-words models like TF or TF-IDF, Word2Vec captures much richer semantic and syntactic relationships between words. There are two main architectures in Word2Vec: Continuous Bag of Words (CBOW) and Skip-Gram. CBOW aims to predict a target word from its context words (surrounding words), and it takes multiple context words as input and tries to predict the word that is most likely to appear in the center of these context words. Skip-Gram, on the other hand, aims to predict context words from a target word. It takes a single word as input and tries to predict its surrounding context words. Here is a general mathematical representation

1. **Input Layer:** Each word in the vocabulary is represented as a one-hot encoded vector. In a vocabulary of size **V**, each word is a **V**-dimensional vector with one element set to 1 and the rest set to 0.

2. **Hidden Layer:** The hidden layer is a fully connected layer with **N** neurons, where **N** is the dimensionality of the word embeddings to learn. This layer serves as a lookup table: the output is the **N**-dimensional word vector corresponding to the input word.

3. **Output Layer:**

   - For CBOW, the output layer is a softmax layer with **v** neurons (one for each word in the vocabulary). It outputs a probability distribution over the vocabulary, representing the likelihood of each word being the target word.
   - For Skip-Gram, the output layer predicts multiple context words. Each context word prediction is a separate softmax operation over the vocabulary.

4. **Training:**

- The model is trained using pairs of context words and target words derived from sentences in the training corpus.
- The training objective is to maximize the probability of the correct target word (in CBOW) or context words (in Skip-Gram) given the input words, using a loss function like cross-entropy.

5. **Word Embeddings:** After training, the weights of the hidden layer become the word embeddings. Each row in the weight matrix corresponds to the vector representation of a particular word in the vocabulary.

After this procedure, Word2Vec then provides a numerical representation of the texts in the form of vectors. The distance defined on the vector space can be used to calculate the similarity between texts, and the selection of the vectors' distance is flexible, and can depend on the application.

B.2 Text Autoencoder

In this section, we provide details of the text autoencoder [117, 107], which is used in our framework to find the latent vectors of the human-readable prompts as in Section 3.3. To begin with, we first introduce the notion of "token".

Definition B.1 (Token)**.** *In the context of text processing and natural language processing (NLP), a "token" generally refers to an individual piece of a larger whole, usually a word, but it can also include punctuation marks, numbers, or other elements depending on the granularity of the tokenization process. In general, tokenization breaks down text into smaller parts (tokens). For example, the sentence "Prompts are helpful" can be tokenized into the tokens"Prompts,", "are," and "helpful."*

By decomposing the input text into a series of tokens $x = \{x_1, x_2, \ldots, x_n\}$, where x denotes the input text and x_i denotes the i-th token, the text autoencoder then transforms each token to a numerical vector (called as embedding), say e_i. Transforming a token x_i to an embedding involves mapping the token to an integer identification (ID) and then using this ID to look up a corresponding dense vector in an embedding matrix. This matrix is part of the model and is adjusted during training to capture semantic relationships between words. Here is a more detailed procedure:

1. **Token Representation:** Each unique token in the vocabulary is assigned a unique integer ID. For example, in a simple case, we have a vocabulary where "cat" is 1, "dog" is 2, etc. The token is then represented by its ID.

2. **Embedding Layer:** An embedding layer is essentially a lookup table that maps integer ID's to high-dimensional vectors. This table is represented as a matrix E in general, where each row denotes a vector representation of a token in the vocabulary.

If the embedding size is d and the vocabulary size is V, then $E \in \mathbb{R}^{V \times d}$, where each row $E_j \in \mathbb{R}^d$ denotes the embedding of the token with ID j.

3. **Lookup Process:** To find the embedding e_i of the token x_i, the model looks up the row in E that is identical to the integer ID of x_i. That is, if x_i is represented by the integer ID k, then the associated embedding is $e_i = E_k$, the k-th row of E.

4. **Learning:** Before learning from the data, the embeddings contained in E are usually initialized randomly. During the learning procedure (which we will describe later), these embeddings are adjusted through backpropagation based on the specific task the model is learning (e.g., classification, translation). The goal is to learn embeddings where similar or related words have similar embeddings. For example, in an effective text autoencoder model, the tokens "rabbit" and "bunny" will have different but similar embeddings, and the similarity is indicated by the distance between their numerical embeddings.

After transforming the tokens into embeddings, the input text is then represented by a sequence of numerical vectors $e = (e_1, e_2, \ldots, e_n)$, where $e_n \in \mathbb{R}^d$ denotes the embedding of the n-th token in the input. In this way, the text input is then transformed into inputs that can be fed into a general autoencoder, which consists of two components: an encoder model and a decoder model. Both the encoder model and the decoder model refer to the functionality and are not restricted to specific types of models. The selection of encoders and decoders depends on the forms of the input/output of the autoencoder model. Common selections include the recurrent neural network (RNN), the convolutional neural network (CNN), and the transformer. In some scenarios (specifically in NLP), both the inputs and output are sequences of embeddings that represent the texts and therefore are in the form of matrices. Here we generally denote them as vectors. Specifically, the encoder model transforms the input to a vector (named the latent vector) with a dimension lower than the input, and then the decoder model transforms the latent vector to a higher-dimensional vector. Mathematically, this procedure is represented by

$$X = f_{enc}(e)$$

and

$$y = f_{dec}(X),$$

where e is the input to the autoencoder, X denotes the latent vector, and y represents of the output of the autoencoder.

In the context of the text autoencoder, the output y, in the form of a numerical vector, is further transformed to text form $\text{text}(y)$ through a reverse process of the embedding process described above. In this way, for each input text x, the output of the text autoencoder is represented by $\text{text}(y(x))$ through the embedding process, the autoencoder model, and the reverse process of the embedding process. Given the structure of a text autoencoder, the learning process involves learning the parameters both contained in the embedding matrix

and those in the encoder/decoder. Specifically, the learning procedure aims to choose the parameters that minimize

$$\sum_i \text{Loss}\left(\text{text}\left(y\left(x_i\right)\right), \tilde{y}_i\right),$$

where Loss is a selected loss function, $(x_i, \tilde{y}_i)$ represents a pair of training data that contains the input x_i and the associated label $\tilde{y}_i$, which are both in text form. In addition, for the task of text reconstruction, $\tilde{y}_i = x_i$. That is, the text autoencoder is trained to generate the input text itself. In addition, we note that training an effective text autoencoder requires a large amount of data and computational resources. However, in our framework, we only have a set of example prompts. Therefore, we employ a pre-trained text autoender [130] and fine-tune it with the set of example prompts. After training, the latent vector then provides a numerical representation of each human-readable prompt.

B.3 Principal Component Analysis

In the search stage, we construct soft prompts by reducing the dimensionality of the latent vectors. We use principle component analysis (PCA). The procedure works as follows:

1. Calculate the sample covariance matrix $C = \frac{1}{L}\sum_{n=1}^{N}\left(X_n - \bar{X}\right)\left(X_n - \bar{X}\right)^\top$, where $\bar{X} = \frac{1}{L}\sum_{n=1}^{L} X_n$. Since the latent vectors are contained in the latent space $\tilde{\mathcal{X}} = [-1, 1]^{\tilde{D}}$, here it is not necessary to standardize the latent vectors X_n's as in a regular procedure of PCA.

2. Decompose the eigenvalue of C by solving the equation $C\mathbf{v} = \lambda\mathbf{v}$ for eigenvalues λ and eigenvectors $\mathbf{v}$ and then sort the eigenvalues and their corresponding eigenvectors in descending order as $\lambda^{(1)} \geqslant \lambda^{(2)} \geqslant \ldots \geqslant \lambda^{(\tilde{D})}$. This step identifies the directions for the latent vectors that exhibit the most variability by finding the eigenvectors that are associated with the largest values. These directions with the most variability have the potential to be most informative.

3. Let $A^\top = \left(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(D)}\right)$ and calculate $z_n = AX_n$ for each $X_n \in \mathcal{X}$. Here D is the selected number of dimensions we would like to retain. In this way, we get the transformed embedding in reduced dimensions and retain the dimensions with the most variability.

This dimensionality reduction procedure also supports the previous procedure for extending the latent vector set. Specifically, the procedure of perturbing existing vectors with the sample covariance matrix has two advantages over traditional methods that use a pre-specified covariance matrix for perturbation. Firstly, it aligns perturbations with PCA, ensuring that the perturbation is fully data-driven and preserves the intrinsic structure and relationships. This contrasts with pre-specified covariance matrices, which lead to a dimension reduction process that reflects the perturbation matrix's structure rather than the

structure of the latent space of the prompts. Secondly, this adaptive method allows for the inclusion of new samples that align with the existing prompts' characteristics. In this way, even when high-dimensional vectors are transformed to a moderate dimension, these vectors remain informative, as the dimensionality reduction procedure selectively retains dimensions that reveal important information during the procedure for extending the latent vector set.

B.4 Sampling Procedures

In the sequential evaluation step of our proposed framework, a Bayesian parametric model is used to approximate the mean score of the soft prompt as in Section 2.3. Given the historical observations of the score $\mathcal{S}_t$, the posterior distribution of the unknown parameters is represented by $p(\boldsymbol{W} \mid \mathcal{S}_t)$. When the posterior distribution does not adopt an explicit form, sampling algorithms are then required to generate samples $\left\{\widehat{\boldsymbol{W}}_1, \widehat{\boldsymbol{W}}_2, \ldots, \widehat{\boldsymbol{W}}_K\right\}$ from the posterior distribution $p(\boldsymbol{W} \mid \mathcal{S}_t)$. Specifically, we select 1) the Hamilton Monte Carlo (HMC) algorithm and 2) variational inference (VI) as representatives.

Hamilton Monte Carlo

Here we describe the Hamilton Monte Carlo (HMC) algorithm [83], starting with the definition of its Hamiltonian.

Definition B.2 (Hamiltonian)**.** *The Hamiltonian $H(\boldsymbol{W}, \boldsymbol{m})$ of an HMC algorithm is defined as*

$$H(\boldsymbol{W}, \boldsymbol{m}) = U(\boldsymbol{W}) + K(\boldsymbol{m}).$$

Here $U(\boldsymbol{W}) = -\log p(\mathcal{S}_t \mid \boldsymbol{W}) - \log(\pi(\boldsymbol{W}))$ is called potential energy, and $K(\boldsymbol{m}) = \frac{1}{2}\boldsymbol{m}^\top \boldsymbol{M}^{-1}\boldsymbol{m}$ is called kinetic energy, where $\boldsymbol{m} \in \mathbb{R}^{d'}$ is the momentum vector and $\boldsymbol{M}$ is a pre-specified mass matrix that is a positive definite and diagonal matrix. In general, $\boldsymbol{M}$ is selected as an identity matrix if there is no prior knowledge.

The procedure of HMC is summarized as follows: 1. In each iteration, HMC samples a new momentum from a normal distribution, then enters a loop where it performs leapfrog steps to propose a new state in the parameter space. This involves making half a step in updating the momentum, a full step updating the parameters, and then another half step updating the momentum. 2. After leapfrogging, it computes a Metropolis acceptance probability to decide whether to accept or reject the new state. If accepted, the algorithm updates the parameters to the new state; if rejected, it retains the old state. This procedure leverages both the current position and momentum to explore the target distribution efficiently, leading to faster convergence compared to many other MCMC methods. The detailed procedure is presented in **Algorithm B.1**.

Algorithm B.1 Hamiltonian Monte Carlo algorithm for generating samples from $p(\boldsymbol{W} \mid \mathcal{S}_t)$

Require: The Hamiltonian $H(\boldsymbol{W}, \boldsymbol{m})$ regarding $\boldsymbol{W}$ and $\boldsymbol{m}$; initial parameters $\boldsymbol{W}_0$, a positive definite and diagonal mass matrix $\boldsymbol{M}$, the number of required samples for the posterior distribution K, the number of samples to be discarded in the burn-in stage K', a step size ϵ, and the number of iterations of a leapfrog step L.

Ensure: The samples from the posterior distribution $\left\{\widehat{\boldsymbol{W}}_1, \ldots, \widehat{\boldsymbol{W}}_K\right\}$.

Let $k = 0$.
while $k < K' + K$ **do**
  Sample momentum $\boldsymbol{m}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{M})$.
  Set $\boldsymbol{W}^{(k+1)} = \boldsymbol{W}_k$ and $\boldsymbol{m}^{(k+1)} = \boldsymbol{m}_k$.
  for $l = 1$ to L **do**
    Update $\boldsymbol{m}^{(k+1)} = \boldsymbol{m}^{(k+1)} - \frac{\epsilon}{2}\nabla_{\boldsymbol{W}} U\left(\boldsymbol{W}^{(k+1)}\right)$.
    Update $\boldsymbol{W}^{(k+1)} = \boldsymbol{W}^{(k+1)} + \epsilon \boldsymbol{m}^{(k+1)}$.
    Update $\boldsymbol{m}^{(k+1)} = \boldsymbol{m}^{(k+1)} - \frac{\epsilon}{2}\nabla_{\boldsymbol{W}} U\left(\boldsymbol{W}^{(k+1)}\right)$.
  end for
  Compute Metropolis acceptance probability $\alpha_c = \min\left(1, \frac{\exp(-H(\boldsymbol{W}^{(k+1)}, \boldsymbol{m}^{(k+1)}))}{\exp(-H(\boldsymbol{W}_k, \boldsymbol{m}_k))}\right)$.
  Sample $u \sim \text{Uniform}(0, 1)$
  if $u < \alpha_c$. **then**
    Accept the new state: $(\boldsymbol{W}_{k+1}, \boldsymbol{m}_{k+1}) = (\boldsymbol{W}^{(k+1)}, \boldsymbol{m}^{(k+1)})$ and let $k = k + 1$.
    if $k > K'$ **then**
      Let $\widehat{\boldsymbol{W}}_{k-K'} = \boldsymbol{W}_k$.
    end if
  end if
end while
return The samples from the posterior distribution $\left\{\widehat{\boldsymbol{W}}_1, \ldots, \widehat{\boldsymbol{W}}_K\right\}$.

Variational Inference

In this section, we provide the procedure of variational inference (VI) that is used to approximate an unknown probability; see also [20]. Here we specifically focus on approximating the posterior distribution $p(\boldsymbol{W} \mid \mathcal{S}_t)$, where $\boldsymbol{W}$ denotes the unknown parameter and $\mathcal{S}_t$ is the set of the observed data. VI aims to find an easy-to-simulate distribution family with an explicit form, say $q(\boldsymbol{W}; \boldsymbol{\lambda}), \boldsymbol{\lambda} \in \boldsymbol{\Lambda}$, and this distribution is used to approximate the posterior distribution $p(\boldsymbol{W} \mid \mathcal{S}_t)$. The approximation is facilitated by optimizing the parameter $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ to minimize the distance (e.g., Kullback–Leibler divergence) between $q(\boldsymbol{W}; \boldsymbol{\lambda})$ and $p(\boldsymbol{W} \mid \mathcal{S}_t)$. After determining the variational distribution $q(\boldsymbol{W}; \boldsymbol{\lambda}^*)$, samples of $\widehat{\boldsymbol{W}} \sim q(\boldsymbol{W}; \boldsymbol{\lambda}^*)$ are efficiently generated for inference.

Furthermore, the Kullback-Leibler divergence between $q(\boldsymbol{w} \mid \boldsymbol{\lambda})$ and $p(\boldsymbol{W} \mid \mathcal{S}_t)$ is challenging to directly optimize in some scenarios. Instead, an equivalent procedure is maximiz-

ing the Evidence Lower Bound (ELBO):

$$\mathrm{ELBO}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{W};\boldsymbol{\lambda})}\left[\log p\left(\mathcal{S}_t, \boldsymbol{W}\right) - \log q(\boldsymbol{W};\boldsymbol{\lambda})\right].$$

The maximization of ELBO is largely supported by the stochastic gradient ascent by generating samples from the variational distribution $q(\boldsymbol{W};\boldsymbol{\lambda})$. This can be accomplished efficiently since the variational distribution is selected to be easy to simulate. After deciding an optimal $\boldsymbol{\lambda}^*$ that maximizes ELBO, samples $\left\{\widehat{\boldsymbol{W}}_1, \widehat{\boldsymbol{W}}_2, \ldots, \widehat{\boldsymbol{W}}_K\right\}$ are then generated from $q(\boldsymbol{W};\boldsymbol{\lambda}^*)$. We summarize the procedure in **Algorithm B.2**.

Algorithm B.2 Variational inference for approximating posterior distribution $p\left(\boldsymbol{W}|\mathcal{S}_t\right)$

Require: A family of variational distributions $q(\boldsymbol{W}|\boldsymbol{\lambda}), \boldsymbol{\lambda} \in \boldsymbol{\Lambda}$, a selected stochastic gradient ascent algorithm with $\boldsymbol{N}$ iterations, and a learning rate $\boldsymbol{\eta}$.
Ensure: The approximate posterior distribution $q(\boldsymbol{W}|\boldsymbol{\lambda}^*)$.
  Initialize variational parameters $\boldsymbol{\lambda}$.
  for $i \in \{1, 2, \ldots, \boldsymbol{N}\}$ **do**
    Compute the gradient $\nabla_{\boldsymbol{\lambda}} \mathrm{ELBO}(\boldsymbol{\lambda})$.
    Update the variational parameters: $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \boldsymbol{\eta}\nabla_{\boldsymbol{\lambda}} \mathrm{ELBO}(\boldsymbol{\lambda})$.
  end for
  Set $\boldsymbol{\lambda}^* = \boldsymbol{\lambda}$.
  return The approximate posterior distribution $q(\boldsymbol{W}|\boldsymbol{\lambda}^*)$.

B.5 Proofs

In this section, we present the proof of theoretical results in the main text.

Proof of Theorem 1

To prove the consistency of **Algorithm 2.1**, we first assume that not all the soft prompts will be evaluated an infinite number of times when the total budget $T \to \infty$. Since the number of soft prompts is finite, as $T \to \infty$, there will always be some soft prompts that are evaluated an infinite number of times, and we denote by

$$\mathcal{Z}_\infty = \left\{z_n \mid \lim_{T\to\infty} r_n(T) = \infty\right\},$$

the set of such soft prompts. Recall that the posterior distribution of the unknown parameters is

$$p\left(\boldsymbol{W} \mid \mathcal{S}_t\right) = \frac{p\left(\mathcal{S}_t \mid \boldsymbol{W}\right)\pi\left(\boldsymbol{W}\right)}{p\left(\mathcal{S}_t\right)} \propto p\left(\mathcal{S}_t \mid \boldsymbol{W}\right)\pi\left(\boldsymbol{W}\right),$$

where

$$p\left(\mathcal{S}_t \mid \boldsymbol{W}\right) = \prod_{n=1}^{N} \prod_{m=1}^{r_n(t)} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{\left(\widehat{v}_{n,m} - \boldsymbol{f}\left(z_n; \boldsymbol{W}\right)\right)^2}{2\sigma_n^2}\right)$$

is the likelihood of the observed scores and $\mathcal{S}_t$ denotes the set of the observed scores up to time t.

We note that, for any selected $z_m \in \mathcal{Z}$, $\boldsymbol{f}\left(z_m; \widehat{\boldsymbol{W}}\right), \widehat{\boldsymbol{W}} \sim p\left(\boldsymbol{W} \mid \mathcal{S}_t\right)$ serves as the approximation of the mean score $v\left(z_m\right)$, and the posterior variance

$$\lim_{t\to\infty} \operatorname{Var}\left[\boldsymbol{f}\left(z_m; \widehat{\boldsymbol{W}}\right) \mid \mathcal{S}_t\right] \overset{w.p.1}{=} 0. \tag{B.1}$$

To see this, we first consider a simple case where we are forced to evaluate only one fixed prompt z_1. In this way, it will be evaluated an infinite number of times and we observe $\mathcal{S}_t^{(1)} = \{\widehat{v}_{1,1}, \widehat{v}_{1,2}, \ldots, \widehat{v}_{1,t}\}$. Since $\forall z_n \in \mathcal{Z}$, $\boldsymbol{f}\left(z_n; \boldsymbol{W}\right) \neq \boldsymbol{f}\left(z_n; \boldsymbol{W}'\right)$ when $\boldsymbol{W} \neq \boldsymbol{W}'$, and $\widehat{v}_{1,m} \overset{i.i.d.}{\sim} \mathcal{N}\left(\boldsymbol{f}\left(z_1; \boldsymbol{W}\right), \sigma_1^2\right)$, by Doob's consistency theorem [166, 128], for any integrable function g, we have

$$\lim_{t\to\infty} \mathbb{E}\left[g\left(\widehat{\boldsymbol{W}}\right) \mid \mathcal{S}_t^{(1)}\right] \overset{w.p.1.}{=} g\left(\boldsymbol{W}\right).$$

Thus, we have

$$\begin{aligned} &\lim_{t\to\infty} \operatorname{Var}\left[\boldsymbol{f}\left(z_1; \widehat{\boldsymbol{W}}\right) \mid \mathcal{S}_t^{(1)}\right] \\ &= \lim_{t\to\infty} \mathbb{E}\left[\boldsymbol{f}^2\left(z_1; \widehat{\boldsymbol{W}}\right) \mid \mathcal{S}_t^{(1)}\right] - \lim_{t\to\infty} \left[\mathbb{E}\left[\boldsymbol{f}\left(z_1; \widehat{\boldsymbol{W}}\right) \mid \mathcal{S}_t^{(1)}\right]\right]^2 \overset{w.p.1.}{=} 0. \end{aligned} \tag{B.2}$$

where

$$p\left(\boldsymbol{W} \mid \mathcal{S}_t^{(1)}\right) = \frac{p\left(\mathcal{S}_t^{(1)} \mid \boldsymbol{W}\right) \pi\left(\boldsymbol{W}\right)}{\int p\left(\mathcal{S}_t^{(1)} \mid \boldsymbol{W}\right) \pi\left(\boldsymbol{W}\right) \mathrm{d}\boldsymbol{W}}.$$

Recall that soft prompts are classified into two categories, $\mathcal{Z}_\infty$ and $\mathcal{Z} \setminus \mathcal{Z}_\infty$. Compared with the special scenario in (B.2) where the observations are i.i.d., the general scenario we want to show, as in (B.1), has two main differences: 1) there are soft prompts that only provide a finite number of observations and 2) there might be multiple soft prompts that are evaluated an infinite number of times. Thus, to prove the statement in (B.1), we must show that these two differences will not change the convergence. Since the number of soft prompts is finite, without loss of generality, we consider two simplified scenarios. In the first scenario, there are two soft prompts to be evaluated, say z_1 and z_2, and z_1 will be evaluated an infinite number of times and z_2 will only be evaluated $n_s < \infty$ times. In this way, we consider the likelihood function when t is sufficiently large is

$$p\left(\mathcal{S}_t^{(1)}, \mathcal{S}_t^{(2)} \mid \boldsymbol{W}\right) = \left(\prod_{m=1}^{t-n_s} p\left(\widehat{v}_{1,m} \mid \boldsymbol{W}\right)\right)\left(\prod_{m=1}^{n_s} p\left(\widehat{v}_{2,m} \mid \boldsymbol{W}\right)\right).$$

The posterior distribution is then

$$\begin{aligned} p\left(\boldsymbol{W} \mid \mathcal{S}_t^{(1)}, \mathcal{S}_t^{(2)}\right) =& \frac{\left(\prod_{m=1}^{t-n_s} p\left(\widehat{v}_{1,m} \mid \boldsymbol{W}\right)\right)\left(\prod_{m=1}^{n_s} p\left(\widehat{v}_{2,m} \mid \boldsymbol{W}\right)\right)\pi(\boldsymbol{W})}{\int \left(\prod_{m=1}^{t-n_s} p\left(\widehat{v}_{1,m} \mid \boldsymbol{W}\right)\right)\left(\prod_{m=1}^{n_s} p\left(\widehat{v}_{2,m} \mid \boldsymbol{W}\right)\right)\pi(\boldsymbol{W})\,\mathrm{d}\boldsymbol{W}} \\ =& \frac{\left(\prod_{m=1}^{t-n_s} p\left(\widehat{v}_{1,m} \mid \boldsymbol{W}\right)\right)\pi'(\boldsymbol{W})}{\int \left(\prod_{m=1}^{t-n_s} p\left(\widehat{v}_{1,m} \mid \boldsymbol{W}\right)\right)\pi'(\boldsymbol{W})\,\mathrm{d}\boldsymbol{W}} \end{aligned}$$

Here $\pi'(\boldsymbol{W}) = \left(\prod_{m=1}^{n_s} p\left(\widehat{v}_{2,m} \mid \boldsymbol{W}\right)\right)\pi(\boldsymbol{W})$ is fixed as $t \to \infty$ since z_2 will only be evaluated $n_s < \infty$ times. In other words, the prior distribution $\pi(\boldsymbol{W})$ is rescaled by $\mathcal{S}_t^{(2)}$. Thus, by applying Doob's consistency theorem and letting $t \to \infty$, we attain an analogous result to (B.2). Thus, the observed scores for the soft prompts in $\mathcal{Z} \setminus \mathcal{Z}_\infty$ will not influence the convergence described in (B.1).

Next, we see that for different soft prompts in $z_n \in \mathcal{Z}$, although the increasing rates $r_n(t)$ might differ as $t \to \infty$, the convergence remains. Without loss of generality, we consider two soft prompts that are both evaluated an infinite number of times, say z_1 and z_3. We then denote two filtrations as

$$\begin{aligned} \mathfrak{F}_m &= \sigma\left\{\left(\widehat{v}_{1,1}, \widehat{v}_{3,1}\right), \left(\widehat{v}_{1,2}, \widehat{v}_{3,2}\right), \ldots, \left(\widehat{v}_{1,m}, \widehat{v}_{3,m}\right)\right\}, \\ \mathfrak{G}_{m,k} &= \sigma\left\{\widehat{v}_{1,1}, \ldots, \widehat{v}_{1,m}, \widehat{v}_{3,1}, \ldots \widehat{v}_{3,k}\right\}. \end{aligned}$$

Thus, we have $\mathfrak{G}_{m,k} \subseteq \mathfrak{F}_{m \wedge k}$, where $m \wedge k = \max\{m, k\}$. In addition, we note that

$$\lim_{m \to \infty} \operatorname{Var}\left[\boldsymbol{f}\left(z_n; \widehat{\boldsymbol{W}}\right) \mid \mathfrak{F}_m\right] \overset{w.p.1}{=} 0, \quad \forall z_n \in \mathcal{Z}$$

by regarding $\left(\widehat{v}_{1,m}, \widehat{v}_{3,m}\right)$ as i.i.d. samples from a joint distribution. Then we have

$$\begin{aligned} \left|\mathbb{E}\left[g\left(\widehat{\boldsymbol{W}}\right) \mid \mathfrak{G}_{m,k}\right] - g(\boldsymbol{W})\right| =& \left|\mathbb{E}\left[\mathbb{E}\left[g\left(\widehat{\boldsymbol{W}}\right) \mid \mathfrak{F}_{m \wedge k}\right] - g(\boldsymbol{W}) \mid \mathfrak{G}_{m,k}\right]\right| \\ \leqslant& \mathbb{E}\left[\left|\mathbb{E}\left[g\left(\widehat{\boldsymbol{W}}\right) \mid \mathfrak{F}_{m \wedge k}\right] - g(\boldsymbol{W})\right| \mid \mathfrak{G}_{m,k}\right] \end{aligned} \tag{B.3}$$

for an integrable g. Note that $\forall z_n \in \mathcal{Z}$, both $\boldsymbol{f}(z_n; \boldsymbol{W})$ and $\boldsymbol{f}^2(z_n; \boldsymbol{W})$ are integrable. By replacing g in (B.3) with $\boldsymbol{f}(z_n; \boldsymbol{W})$ and $\boldsymbol{f}^2(z_n; \boldsymbol{W})$ respectively, we then have

$$\lim_{m,k \to \infty} \operatorname{Var}\left[\boldsymbol{f}(z_n; \boldsymbol{W}) \mid \mathfrak{G}_{m,k}\right] \overset{w.p.1.}{=} 0$$

from the dominated convergence theorem, giving us (B.1). That is, for every soft prompt $z_n \in \mathcal{Z}$, the posterior variance of $\boldsymbol{f}\left(z_n; \widehat{W}\right)$ shrinks to zero as $t \to 0$.

On the other hand, maximizing the acquisition function M-UCB is equivalent to maximizing a rescaled M-UCB

$$\alpha_t'(z_n) = \frac{\mu_t(z_n)}{\beta_t} + \sigma_t(z_n) + \gamma\left(r_n(t)\right).$$

As $\lim_{t\to} \beta_t = \infty$, and $\lim_{r_n(t)\to\infty} \gamma\left(r_n\left(t\right)\right) = 0$, we have

$$\lim_{t\to\infty} \alpha'_t\left(z_n\right) \begin{cases} = 0, & \forall z_n \in \mathcal{Z}_\infty \\ > 0, & \forall z_n \in \mathcal{Z} \setminus \mathcal{Z}_\infty \end{cases}. \tag{B.4}$$

Since the soft prompt selected in each iteration is the soft prompt that maximizes the (rescaled) acquisition function, which contradicts (B.4), the assumption that $\mathcal{Z}_\infty \subsetneq \mathcal{Z}$ is violated. Thus, every soft prompt will be evaluated an infinite number of times with probability one as the total budget T approaches infinity.

Because the evaluation uncertainty is Gaussian noise and is independently and identically distributed, the strong law of large numbers holds [166], and for any $z_m \notin \arg\max_{z_n} v\left(z_n\right)$, there will be a discrepancy, say δ. Thus, when T is sufficiently large, we have

$$\bar{v}\left(z_m\right) < v\left(z_m\right) + \frac{\delta}{2} < v\left(z^*\right) - \frac{\delta}{2} < \bar{v}\left(z^*\right)$$

with probability one. In other words, when $T \to \infty$, the soft prompts that are not optimal will not be selected.

Proof of Theorem 2

In this section, we prove the consistency of the probabilistic reparametrization of the acquisition function and propositions in Section 3.4.

Regarding Proposition 1, because

$$\mathbb{E}\left[\boldsymbol{f}^2\left(z_n; \boldsymbol{W}\right) \mid \mathcal{S}_t\right] < \infty \quad \forall z_n \in \mathcal{Z},$$

we have

$$\lim_{K\to\infty} \widehat{\alpha}_t\left(z_n\right) \overset{w.p.1}{=} \alpha_t\left(z_n\right)$$

from the strong law of large numbers [166]. Furthermore, since there are only a finite number of z_n's in $\mathcal{Z}$, this convergence is uniform. That is,

$$\lim_{K\to\infty} \left\{ \delta_K \doteq \max_{z_n \in \mathcal{Z}} \left\{ \widehat{\alpha}_t\left(z_n\right) - \alpha_t\left(z_n\right) \right\} \right\} \overset{w.p.1}{=} 0.$$

Thus, we prove Proposition 1 because

$$\left| \widehat{\alpha}_t\left(\bar{z}^*\right) - \alpha_t\left(z^{(t+1)}\right) \right| \leqslant \delta_K.$$

Regarding Theorem 2, we consider a mapping

$$\varphi : [0,1]^N \to \mathcal{P}_{\mathcal{Z}},$$

with

$$p_{\varphi(\theta)}\left(\{z_i\}\right) = \frac{\theta^{(i)}}{\sum_{j=1}^N \theta^{(i)}},$$

which maps $\theta \in \Theta = [0,1]^N$ to a discrete probability distribution $\mathcal{P}_{\mathcal{Z}}$ defined on $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$. Here $\theta^{(i)}$ represents the i-th entry of θ. We note that this function is continuous in θ by regarding $(\mathcal{P}_{\mathcal{Z}}, \|\cdot\|)$ as a metric space with any norm $\|\cdot\|$ defined on an N-dimensional vector space.

Furthermore, for any $\theta^* \in \mathcal{J}_t^* = \{\theta \mid \theta \in \arg\max_{\theta\in\Theta} \widehat{\alpha}_t(\theta)\}$ and $z^* \in \text{support}\,(p(z_n;\theta^*))$, we have $z^* \in \mathcal{H}_t^*$. To prove this, we assume that there exists $z' \in \text{support}\,(p(z_n;\theta^*))$ and $z' \notin \mathcal{H}_t^*$. Thus,

$$p_{\varphi(\theta^*)}(\{z'\}) > 0.$$

We consider a probability measure

$$p'(\{z\}) = \begin{cases} 0 & \text{if } z = z' \\ p_{\varphi(\theta^*)}(\{z^*\}) + p_{\varphi(\theta^*)}(\{z'\}) & \text{if } z = z^* \\ p_{\varphi(\theta^*)}(\{z\}) & \text{otherwise.} \end{cases}$$

In this way, we have

$$\mathbb{E}_{p'(\{z\})}[\alpha_t(z_n)] = \alpha_t(z^*) + p_{\varphi(\theta^*)}(\{z'\})(\alpha_t(z^*) - \alpha_t(z')) > \alpha_t(z^*).$$

On the other hand, since $\varphi(\theta)$ is continuous, there will be a $\theta' \in \Theta$ such that $\varphi(\theta') = p'$, which contradicts the assumption that $\theta^* \in \mathcal{J}_t^*$. Therefore, we have $z' \in \mathcal{H}_t^*$ and

$$\hat{\mathcal{H}}_t^* \subseteq \mathcal{H}_t^*. \tag{B.5}$$

Next, we prove that if $z^* \in \mathcal{H}_t^*$, then

$$\alpha_t(z^*) = \max_{\theta\in\Theta} \widetilde{\alpha}_t(\theta). \tag{B.6}$$

To see this, for any $z^* \in \mathcal{H}_t^*$, we set θ^* be the parameters such that $p(z^*;\theta^*) = 1$. In this way, to prove (B.6), we assume that there exists θ' such that $\widetilde{\alpha}_t(\theta') > \widetilde{\alpha}_t(\theta^*)$. On the other hand, since $z^* \in \arg\max_{z_n} \alpha_t(z_n)$, no convex combinations (expected values) of α_t can be larger. Thus, there is contradiction and (B.6) holds. Therefore,

$$\mathcal{H}_t^* \subseteq \hat{\mathcal{H}}_t^*.$$

By taking (B.5) into consideration, we have that

$$\hat{\mathcal{H}}_t^* = \mathcal{H}_t^*.$$

Furthermore, we note that φ is differentiable in θ for any given z_n. Thus, PR-M-UCB is a finite combination of differentiable $p(z_n;\theta)\,\alpha_t(z_n)$'s, and therefore is also differentiable. Moreover, because

$$\nabla_\theta \log p(z_n \mid \theta) = \frac{\nabla_\theta p(z \mid \theta)}{p(z_n \mid \theta)},$$

we have

$$\begin{aligned}\nabla_\theta \mathbb{E}_{p(z_n;\theta)}\left[\alpha_t\left(z_n\right)\right] &= \sum_{n=1}^{N} \alpha_t\left(z_n\right) \nabla_\theta p\left(z_n;\theta\right) \\ &= \sum_{n=1}^{N} \alpha_t\left(z_n\right) \nabla_\theta \log p\left(z_n \mid \theta\right) p\left(z_n;\theta\right) \\ &= \mathbb{E}_{p(z_n;\theta)}\left[\alpha_t\left(z_n\right) \nabla_\theta \log p\left(z_n \mid \theta\right)\right].\end{aligned}$$

Thus, we prove Proposition 3. The unbiased estimator

$$\widehat{\nabla_\theta \widetilde{\alpha}_t}(\theta) \doteq \frac{1}{I} \sum_{i=1}^{I} \alpha_t\left(\widehat{z}_i\right) \nabla_\theta \log\left(p\left(\widehat{z}_i;\theta\right)\right)$$

is also known as REINFORCE [175] and the likelihood ratio estimator [84].

In addition, we note that the maximization of PR-M-UCB is reformulated as a linear programming problem when we impose the constraint $\sum_{i=1}^{N} \theta^{(i)} = 1$. On the other hand, the coefficients $\left(\alpha_t\left(z_1\right), \alpha_t\left(z_2\right), \ldots, \alpha_t\left(z_N\right)\right)$ in the objective function are unknown and require estimation, which is time-consuming if N is large. Therefore, we employ stochastic gradient ascent to optimize PR-M-UCB. We refer to [124] for detailed discussions on stochastic linear programming.

Proof of Theorem 3

In this section, we provide the upper bound of the cumulative uncertainty of the PSK predictor $U_I = \sum_{i=1}^{I} \widehat{\sigma^2}\left(\tilde{X}_i\right)$, where $\widehat{\sigma^2}\left(\cdot\right)$ is the prediction uncertainty of the PSK predictor. First, we have

$$\widehat{\sigma^2}\left(X\right) \leqslant \sup_{X \in \tilde{\mathcal{X}}} \left\| \tilde{A}^*\left(X\right) \right\|^2 \leqslant A_M$$

for some constant $A_M > 0$. Then, since $\frac{s}{\log(1+s)}$ is a non-decreasing function with $s > 0$, we have

$$\widehat{\sigma^2}\left(\tilde{X}_i\right) \leqslant \frac{A_M}{\log\left(1 + A_M \underline{\sigma_\epsilon^{-2}}\right)} \log\left(1 + \underline{\sigma_\epsilon^{-2}} \widehat{\sigma^2}\left(\tilde{X}_i\right)\right) \tag{B.7}$$

for each additional evaluation $\tilde{X}_i \in \mathcal{R}$. Here $\underline{\sigma_\epsilon^{-2}} \widehat{\sigma^2} = \inf_{X \in \tilde{\mathcal{X}}} \sigma_\epsilon^2\left(X\right)$.

We then employ the information gain [46] of the Gaussian process, which is

$$\mathcal{I}\left(\widehat{V}_j; V_j\right) = H\left(\widehat{V}_j\right) - H\left(\widehat{V}_j \mid V_j\right),$$

where $\widehat{V}_j = \left(\widehat{v}\left(\tilde{X}_1\right), \ldots, \widehat{v}\left(\tilde{X}_j\right)\right)^\top$ is the vector of the observations, and

$$V_j = \left(v\left(\tilde{X}_1\right), \ldots, v\left(\tilde{X}_j\right)\right)^\top$$

denotes the mean score vector. Also, $H\left(\widehat{V}_j\right)$ is the entropy of $\widehat{V}_j$ and $H\left(\widehat{V}_j \mid V_j\right)$ denotes the conditional entropy of $\widehat{V}_j$ on V_j. Note that, for a Gaussian random vector $\xi \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $H(\xi) = \frac{1}{2}\log|2\pi e\boldsymbol{\Sigma}|$, where e is Euler's number. Thus

$$\begin{aligned} H\left(\widehat{V}_j\right) &= H\left(\widehat{V}_{j-1}\right) + H\left(\widehat{v}\left(\tilde{X}_j\right) \mid \widehat{V}_{j-1}\right) \\ &= H\left(\widehat{V}_{j-1}\right) + \frac{1}{2}\log\left(2\pi e\left(\sigma_\epsilon^2\left(\tilde{X}_j\right) + \widehat{\sigma}^2\left(\tilde{X}_j\right)\right)\right) \\ &\geqslant H\left(\hat{V}_{j-1}\right) + \frac{1}{2}\log\left(2\pi e\left(\underline{\sigma_\epsilon^2} + \widehat{\sigma^2}\left(\tilde{X}_j\right)\right)\right). \end{aligned}$$

Since $\mathcal{I}\left(\widehat{V}_I; V_I\right) = H\left(\widehat{V}_I\right) - H\left(\widehat{V}_I \mid V_I\right)$, we then have

$$\mathcal{I}\left(\widehat{V}_I; V_I\right) \geqslant \frac{1}{2}\sum_{i=1}^{I}\log\left(1 + \underline{\sigma_\epsilon^{-2}}\widehat{\sigma^2}\left(\tilde{X}_i\right)\right).$$

Therefore, considering (B.7), the cumulative uncertainty is bounded by

$$U_I = \sum_{i=1}^{I}\widehat{\sigma^2}\left(\tilde{X}_i\right) \leqslant \frac{2A_M}{\log\left(1 + A_M\underline{\sigma_\epsilon^{-2}}\right)}\mathcal{I}\left(\widehat{V}_I; V_I\right). \tag{B.8}$$

Next, we provide the upper bound of $\mathcal{I}\left(\widehat{V}_I; V_I\right)$. Specifically,

$$\begin{aligned} \mathcal{I}\left(\widehat{V}_I; V_I\right) =& H\left(\widehat{V}_I\right) - H\left(\widehat{V}_I \mid V_I\right) \\ =& \frac{1}{2}\log\left|2\pi e\left(\boldsymbol{K}_I + \mathrm{Diag}\left\{\sigma_\epsilon^2\left(\tilde{X}_1\right), \sigma_\epsilon^2\left(\tilde{X}_2\right), \ldots, \sigma_\epsilon^2\left(\tilde{X}_I\right)\right\}\right)\right| \\ & - \frac{1}{2}\log\left|2\pi e \cdot \mathrm{Diag}\left\{\sigma_\epsilon^2\left(\tilde{X}_1\right), \sigma_\epsilon^2\left(\tilde{X}_2\right), \ldots, \sigma_\epsilon^2\left(\tilde{X}_I\right)\right\}\right| \\ \leqslant& \frac{1}{2}\log\left|\underline{\sigma_\epsilon^{-2}}\boldsymbol{K}_I + \overline{\sigma_\epsilon^2}/\underline{\sigma_\epsilon^2}\boldsymbol{I}_I\right|, \end{aligned} \tag{B.9}$$

where $\boldsymbol{I}_I$ denotes the I-dimensional identity matrix, and $\boldsymbol{K}_I$ denotes the kernel matrix

$$\begin{pmatrix} \tilde{A}^*\left(\tilde{X}_1\right)^\top \\ \vdots \\ \tilde{A}^*\left(\tilde{X}_I\right)^\top \end{pmatrix}\left(\tilde{A}^*\left(\tilde{X}_1\right), \cdots, \tilde{A}^*\left(\tilde{X}_I\right)\right) \in \mathbb{R}^{I\times I}.$$

Denote $\boldsymbol{A}_I = \left(\tilde{A}^*\left(\tilde{X}_1\right), \cdots, \tilde{A}^*\left(\tilde{X}_I\right)\right) \in \mathbb{R}^{D^*\times I}$. Thus $\boldsymbol{K}_I = \boldsymbol{A}_I^\top\boldsymbol{A}_I$. Note that,

$$\begin{aligned} \left|\overline{\sigma_\epsilon^2}/\underline{\sigma_\epsilon^2}\boldsymbol{I}_I + \underline{\sigma_\epsilon^{-2}}\boldsymbol{A}_I^\top\boldsymbol{A}_I\right| &= \left|\overline{\sigma_\epsilon^2}/\underline{\sigma_\epsilon^2}\boldsymbol{I}_{D^*} + \underline{\sigma_\epsilon^{-2}}\boldsymbol{A}_I\boldsymbol{A}_I^\top\right| \\ &\leqslant \prod_{d=1}^{D^*}\left(\overline{\sigma_\epsilon^2}/\underline{\sigma_\epsilon^2} + \underline{\sigma_\epsilon^{-2}}\boldsymbol{a}_{dd}\right), \end{aligned}$$

where $\boldsymbol{a}_{dd}$ denotes the (d,d)-th entry of $\boldsymbol{A}_I\boldsymbol{A}_I^\top \in \mathbb{R}^{D^*\times D^*}$. Here the equality comes from that $\boldsymbol{A}_I^\top\boldsymbol{A}_I$ and $\boldsymbol{A}_I\boldsymbol{A}_I^\top$ have the same non-zero eigenvalues, and the inequality comes from the Hadamard inequality; see [80]. Since the norm of $A^*(X)$ is bounded, the largest eigenvalue of $\boldsymbol{A}_I\boldsymbol{A}_I^\top$ is at most at order of I. Thus,

$$\log\left|\overline{\sigma_\epsilon^2}/\underline{\sigma_\epsilon^2}\boldsymbol{I}_I + \underline{\sigma_\epsilon^{-2}}\boldsymbol{A}_I^\top\boldsymbol{A}_I\right| = \mathcal{O}\left(D^*\log I\right),$$

which provides the upper bound of the cumulative uncertainty U_I considering both (B.8) and (B.9).

B.6 Surrogate Models

In our work, we propose using Bayesian parametric models as the surrogate model for approximating the mean score of soft prompts. The reason for selecting the Bayesian parametric model is two-fold. 1. Why do we use a surrogate model? Although simulation optimization methods for the finite feasible set have been extensively explored in existing literature, these algorithms in general require either 1) a specific known structure of the objective function (convexity, Lipschitz continuity, etc.) or 2) evaluating each decision variable a sufficient number of times. In the context of the prompt selection, it is not guaranteed that the mean score of the prompts has such a structure. Also, it is also expensive to evaluate each prompt for enough times, considering both the computational time to generate output contexts and the cost of invoking proprietary generative language models. On the other hand, each prompt to be evaluated is represented by a vector-valued soft prompt z_n. The mean score of the soft prompt has an implicit dependence on the vector z_n, and we use a surrogate model to approximate the dependence. Thus, for a prompt that has never been evaluated, we also have inference based on the surrogate model, which is constructed with observations associated with other prompts. 2. Why do we employ Bayesian inference? In order to sequentially evaluate the prompt in an efficient manner, the trade-off between exploitation and exploration must be addressed appropriately. That is, we are supposed to evaluate the prompt that has the evidence to achieve a high score (exploitation) while evaluating the prompt that has more uncertainty to achieve a high score (exploration). Thus, in addition to the approximated value of the mean score $v(\cdot)$, the uncertainty of the approximation is also required. Bayesian inference provides an effective strategy to quantify the approximation uncertainty, which is used to address the exploitation-exploration trade-off. In this section, we provide details of the surrogate models used in our experiments to approximate the mean score function with respect to the soft prompt. Specifically, we consider surrogate models including 1) the Bayesian linear regression model, 2) the Gaussian process (GP) process model, 3) the Bayesian neural network (BNN) model, and 4) the variational autoencoder (VAE). In terms of GP and BNN, the descriptions are contained in Section 2.3 as examples. Here we focus on the Bayesian linear regression model and VAE.

Bayesian Linear Regression

Bayesian linear regression is a statistical method in which we use Bayes' Theorem to estimate the parameters of a linear regression model. In Bayesian linear regression, the model parameters are treated as random variables and a prior probability distribution is imposed on these parameters. This distribution is then updated by Bayes' Theorem with the observed data.

Specifically, the model is represented by

$$\widehat{v}(z) = \boldsymbol{W}^\top z + \epsilon,$$

where $\widehat{v}(z)$ is the predicted output, z is the input vector, $\boldsymbol{W}$ is the vector of unknown weights or parameters, and ϵ is the noise term, typically assumed to be Gaussian with zero mean and variance σ^2.

In Bayesian linear regression, rather than finding single point estimates of $\boldsymbol{W}$, we calculate the posterior distribution of $\boldsymbol{W}$ given the data. This is done using Bayes' theorem:

$$P(\boldsymbol{W} \mid \boldsymbol{Z}, \boldsymbol{V}) = \frac{P(\boldsymbol{V} \mid \boldsymbol{Z}, \boldsymbol{W})\pi\left(\boldsymbol{W}\right)}{P(\boldsymbol{V} \mid \boldsymbol{Z})}$$

where $\boldsymbol{Z}$ is the matrix of input data, $\boldsymbol{V}$ is the vector of output data, $\pi\left(\boldsymbol{W}\right)$ is the prior distribution of $\boldsymbol{W}$, $P(\boldsymbol{V} \mid \boldsymbol{Z}, \boldsymbol{W})$ is the likelihood of observing $\boldsymbol{V}$ given $\boldsymbol{Z}$ and $\boldsymbol{W}$, and $P(\boldsymbol{V} \mid \boldsymbol{Z})$ is the marginal likelihood or evidence. Here we assume that the variance of the noise σ^2 is known. In this way, a common selected prior of the unknown parameters is the Gaussian distribution, that is, $P(\boldsymbol{W}) = \mathcal{N}\left(\boldsymbol{W} \mid \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\right)$, where $\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0$ are user-specified. We refer to [129] for a more detailed description.

Variational Autoencoder

Recall that in Section B.2, we describe the model of text autoencoder, in which tokens are first transformed to numerical representations and then fed into an autoencoder model. VAE, as a type of autoencoder model, is also composed of two components: 1) an encoder model and 2) a decoder model, and encoder/decoder models are largely neural networks, and depend on the applications. The difference between a regular autoencoder and VAE is that the input (numerical representation) is transformed into a latent vector in the latent space by the encoder of a regular autoencoder model. In the VAE model, each input, say z, fed into the encoder is then mapped to a distribution in the latent space, say $p\left(y \mid z, \boldsymbol{W}_{enc}\right)$. Here y denotes the latent vectors that are random, z is the input to VAE and $\boldsymbol{W}_{enc}$ denotes the unknown parameters in the encoder part. The latent vector, once generated, is then mapped by the coder model to the output of VAE. In this way, given a fixed input of z, VAE will generate different samples of latent vectors $\{y_1, y_2, \ldots, y_K\}$ and then provide a set of outputs. This generative property allows VAEs to be powerful tools for both supervised and unsupervised learning problems, particularly in tasks like image generation, anomaly detection, and feature extraction.

In a VAE model, the unknown parameters are implicitly contained in the encoder/decoder models. That is, if the encoder and the decoder are both selected to be neural networks, the unknown parameters of VAE are the weight parameters of the neural networks. As a Bayesian parametric model, the unknown parameters can be updated by the Bayes' Theorem and the method of variational inference is widely employed to approximate the posterior distribution of the unknown parameters. On the other hand, when the encoder and decoder models are complex models (e.g., transformers), it is challenging to approximate the posterior distribution, even for the method of variational inference. Instead, an alternative method to train VAE is to minimize a combined loss function

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{regression}} + \beta \times \mathcal{L}_{KL},$$

where $\mathcal{L}_{\text{regression}}$ denotes the loss between the observed groud-truth data and the predicted output generated by VAE, and $\mathcal{L}_{KL}$ quantifies the distance (Kullback–Leibler divergence) between the learned latent space representation and a predefined distribution (usually a standard normal distribution), providing a form of regularization. For more detailed discussions of VAE, we refer to [108].

B.7 Hyperparameter Selection in Acquisition Function

Recall that, the proposed acquisition function modified upper confidence bound (M-UCB) is

$$\alpha_t(z_n) = \mu_t(z_n) + \beta_t(\sigma_t(z_n) + \gamma(r_n(t)))$$

where $\mu_t(z_n)$ and $\sigma_t(z_n)$ are the posterior mean and standard deviation of the surrogate model at z_n, and $r_n(t)$ denotes the number of evaluations at z_n up to time t. In addition, β_t is a sequence of hyperparameters and $\gamma(\cdot)$ is a user-specified non-negative decreasing function. In this section, we provide a practical procedure to decide the sequence of hyperparameters β_t and $\gamma(\cdot)$. The procedure involves constructing a stochastic kriging (SK) model to approximate the scores of the soft prompts [5], using the observations collected during the warm-up stage (Section 3.4). This SK model, once constructed, serves as a synthetic observation generative model. That is, when evaluating each selection of the hyperparameter in the acquisition function before the sequential evaluation and selection step, we use the synthetic observation generated by the SK model instead of the score observations collected from the language model. Specifically, $\forall z_n \in \mathcal{Z}$, the constructed SK model generates

$$\tilde{v}(z_n) \sim \mathcal{N}\left(\mu_{\text{SK}}(z_n), \sigma^2_{\text{SK}}(z_n) + g^*(z_n)\right),$$

where $\mu_{\text{SK}}(z_n)$ and $\sigma^2_{\text{SK}}(z_n)$ are the predicted value and mean squared error of the SK predictor, and $g^*(z_n)$ denotes the approximated observation variances as described in Section 3.4. Regarding the explicit form of $\mu_{\text{SK}}(z_n)$ and $\sigma^2_{\text{SK}}(z_n)$, we refer to [5]. The procedure shares similar spirits with parametric bootstrap [109].

To implement the procedure, we denote the set of the observations as $\mathcal{S}_W \doteq \{(z_i, \hat{v}_i)\}_{i=1}^{I_W}$, where I_W is the number of the observations collected during the warm-up step. Furthermore, we assume that the pair of sequence of hyperparameters β_t and the function $\gamma(\cdot)$ are selected from a pre-defined set $\mathfrak{C} = \left\{\left(\beta_t^{(1)}, \gamma^{(1)}(\cdot)\right), \left(\beta_t^{(2)}, \gamma^{(2)}(\cdot)\right), \ldots, \left(\beta_t^{(Q)}, \gamma^{(Q)}(\cdot)\right)\right\}$. With the notation set up, the procedure works as follows:

1. Construct a SK model using the dataset $\mathcal{S}_W$;
2. For each pair $\left(\beta_t^{(q)}, \gamma^{(q)}(\cdot)\right) \in \mathfrak{C}$
   a) Perform the sequential evaluation and selection step (Section 3.4) using the synthetic observations $\tilde{v}(z_n)$ generated by the constructed SK model instead of using the language model to collect score observations;
   b) Record the selected soft prompt $z_{n^*}^{(q)}$ when the iteration ends;
3. Select the pair $\left(\beta_t^{(q)}, \gamma^{(q)}(\cdot)\right)$ with the maximum value of $\mu_{\text{SK}}\left(z_{n^*}^{(q)}\right)$.

In addition to a finite set of candidate hyperparameters $\mathfrak{C}$, the proposed procedure can also be adapted to the scenarios where β_t and $\gamma(\cdot)$ are parametrized by continuous parameters. For more methods of hyperparameter selection, we refer to [154].

B.8 Selected Prompts

In **Table B.1**, we present the selected prompts using the algorithm M-UCB-r (described in Section 3.6) regarding all six tasks. The experimental setting is consistent with that in **Figure 3.8**.

Table B.1: Selected prompts for six tasks using M-UCB-r.

Task	**turbo-3.5**	**davinci-003**
word sorting	Sort the following word list in alphabetic order.	Sort the words in alphabet order.
first letter	Return the first letter of the selected word.	Select the initial letter in the word.
counting objectives	Count the total number of mentioned products.	Estimate the total quantity for all products.
rhymes	Choose the word with the closest pronunciation.	Select the word with most similar pronunciation.
nums to verbal	Turn number list to word list.	Turn number list to word list.
largest animals	Choose the larger animal from the following animals.	Select the largest animals in size.