BAYESIAN EXPERIMENTAL DESIGN BASED ON MEAN OBJECTIVE COST OF UNCERTAINTY

A Dissertation

by

GUANG ZHAO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2021

Major Subject: Electrical and Computer Engineering

ABSTRACT

We propose several efficient algorithms for Bayesian experimental design when studying complex systems under uncertainty with specific operational objectives. Throughout this dissertation, the uncertainty is quantified by the mean objective cost of uncertainty (MOCU).

First, we develop MOCU-based experimental design for physical systems described by Stochastic Differential Equations (SDEs) with uncertain model parameters. We assume the observed signals are from a system whose dynamics is governed by SDEs. The observations can be degraded by blurring and additive noise. We aim to derive a optimal robust filter minimizing the expected filtering error. We further derive an optimal experimental design framework to determine the importance of the SDE parameters. Such a framework can update the knowledge about the system and thereafter the signal processes. As a result, it guides the systems knowledge discovery to help derive better filters.

In the MOCU-based framework, we further study Bayesian active learning to sequentially sample queries to improve predictive models. For classification, the goal is to learn the optimal classifier with high prediction accuracy when classification labels is difficult or costly to obtain. The MOCU-based active learning procedure is shown to get stuck before converging to the optimal classifier, due to the piece-wise linearity of MOCU. We propose two methods to address this myopic issue of MOCU-based active learning by approximating the MOCU functions with strict concavities, named Weighted-MOCU (WMOCU) and Soft-MOCU (SMOCU). We provide theoretical proofs of the convergence of these methods and demonstrate their sampling efficiency with both synthetic and real-world experiments.

Finally, to explore more practical MOCU-based experimental design, we study MOCU-based active learning of both pool-based and query synthetic scenarios for Gaussian Process Classification (GPC). We develop computationally efficient algorithms for MOCU-based active learning with GPC. Our algorithms compute the joint predictive distribution of label pairs as a one-dimensional integral, which enables us to compute MOCU-based acquisition functions without incrementally retraining GPC for each possible query. By deriving the gradient chain rule to efficiently calculate the gradient of SMOCU reduction, we also develop the first MOCU-based query synthesis active learning algorithm.

DEDICATION

To my wife Fangyuan and my little girl Muchen.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my PhD advisor, Professor Xiaoning Qian, for his continuous guidance and help throughout my PhD study. During the past few years, I have learned a lot from not only his knowledge and expertise, but also his enthusiasm on research and hardworking attitude. I would also like to thank Dr. Edward Dougherty, Dr. Tie Liu, and Dr. Raymundo Arróyave for serving on my thesis committee. Their comments and suggestions really helped me improve my dissertation work. I would like to thank Randy Ardywibowo for invaluable and insightful discussions. Finally I need to thank my wife Fangyuan Chen, for her endless patience and loving encouragement. I cherish every day in my PhD studies with her accompany, understanding and love.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Xiaoning Qian [advisor] and Professors Edward R. Dougherty and Tie Liu of the Department of Electrical and Computer Engineering and Professor Raymundo Arróyave of the Department of Materials Science and Engineering.

All work for the dissertation was completed by the student, under the advisement of Professor Xiaoning Qian of the Department of Electrical and Computer Engineering

Funding Sources

The work was supported by the National Science Foundation, through NSF awards 1553281, 1835690; the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Multifaceted Integrated Capability Centers program under Award DE-SC0019303; and the U.S. Department of Defense, Defense Advanced Research Projects Agency (DARPA) Warfighter Analytics using Smartphones for Health (WASH) Project FA8750-18-2-0027.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Motivation

A wide variety of engineering problems can be formulated as building a mathematical model, including surrogate models by machine learning, to describe the system and then finding an optimal operator to minimize a cost function with respect to an operational objective, such as the ones adopted in filtering, classification, or control problems. Usually complex systems cannot be perfectly modeled or accurately identified, and the resulting model uncertainty will affect the estimated operational objective, thereafter requiring a robust operator over the uncertainty. To reduce the model uncertainty and thereby facilitate the efficient attainment of the final operational objective, observations or experiments related to the complex system are required. The experiments may directly probe the underlying system states or collect measurements from the system. From the experimental results, one may gather information of the system and therefore reduce the model uncertainty. However, these experiments can be both resource- and time-consuming in many areas such as materials discovery [19], cell signaling pathway identification [3] and customer preference understanding [8]. Therefore, efficient experimental design methods are needed.

Experimental design aims at reducing the model uncertainty by conducting as fewer experiments as possible. In Bayesian settings, the model uncertainty is expressed in the form of the distribution over possible models. Lindley first presented a decision-theoretic approach to Bayesian experimental design, in which the experimental design is solved as an optimization problem, aimed at maximizing the utility related to the operational objective provided by the new experiments [52]. The design can be carried out either in sequential [58] or in parallel [30]. In this dissertation we discuss the sequential experimental design. The experimental design iteration is divided in three stages: design stage, experiment stage, and inference stage . In the design stage, the best experiment is chosen by maximizing an acquisition function, which describes the benefit provided by a new experiment in reducing the model uncertainty. The benefit can be formulated in different ways based on the design criterion and is usually averaged over the model distribution. In the experiment stage, new data is collected from the chosen experiment. In the inference stage, the newly obtained experiment data is used to update the distribution over models, and the updated distribution is further used for the calculation of the acquisition function in the next iteration. After finishing the experimental design procedure, the optimal robust operator is taken under the reduced model uncertainty.

The performance of the experimental design procedure can be measured by metrics related to the model uncertainty. For example, Shannon entropy of the model distribution. However, Shannon entropy fails to take the operational objective into account and therefore cannot adequately reflect the performance of the experimental design procedure. Especially, some model uncertainties may be unrelated to the derivation of the optimal operator and therefore will not affect the operational objective. But the entropy will still count these unrelated uncertainties. In this dissertation, we adopt an objective-oriented uncertainty quantification framework, the mean objective cost of uncertainty (MOCU) [84], which directly measures the effect of the model uncertainty on the operational objective. MOCU is defined as the expected objective cost due to using a robust operator instead of using the optimal operator for a particular model.

MOCU can be used to measure the experimental design performance as smaller MOCU indicates a better performance with less effect of the remaining model uncertainty on the operational objective. It can also be treated as a design criterion to define a sequential strategy that selects the experiment to maximally reduce the MOCU in a one-step-look-ahead manner. The MOCU-based experimental design only focuses on reducing the uncertainty that directly affects the operational objective and can be more efficient comparing with other experimental design strategies.

1.2 Literature Review

Sequential Bayeisan experimental design has been discussed extensively in the literature and here we give a brief review, focusing on different operational objectives.

A significant portion of the experimental design literature is focused on the model parameter estimation. For example, there are experimental design papers on pharmaceutical applications [66], biomedical research [45], clinical trail analysis [4], and chemical engineering systems [79]. Among these papers, the most commonly used design strategy is based on mutual information, choosing the experiment that can provide the maximum mutual information of the model parameters [79]. Such a strategy is equivalent to maximizing the Kullback-Leibler (KL) distance between the prior and posterior model distributions. There are also other strategies choosing the experiment minimizing the posterior variance [21], or sequentially minimizing the Alphabetical optimality [80].

Active learning is another specific direction of the experimental design in machine learning to efficiently acquire training data. The operational objective is optimizing the model prediction for the whole instance space, either for regression or for classification problems, in a data efficient manner. Usually in active learning, the experiments are queries of the instances. The simplest active learning strategy is Maximum Entropy Sampling (MES), which queries the instance that we are most uncertain about the outputs [51]. The uncertainty is measured by the entropy of the predictive probability. MES strategies are simple to implement and perform well in many cases. However, since it fails to differentiate between model uncertainty and observation uncertainty, in the practical cases with significant observation uncertainty, it may repeatedly query instances in regions of high observation uncertainty. Bayesian Active Learning by Disagreement (BALD) is a mutual information based strategy [37]. Taking the effect of observation uncertainty on learning in consideration, the strategy selects the query providing the maximum mutual information of the model parameters. Query-By-Committee (QBC) includes a variety of models trained on the current observed data, and the algorithms choose those points for which the "committee" disagrees the most [74]. There are also MOCU-based active learning strategies, for example, Expected Error Reduction (EER). EER aims directly at reducing the prediction error after observing the new data [65].

Another experimental design area, Bayesian optimization, is widely applied in materials design [27] or drug discovery [62]. The operational objective is to find the global optimal points for black-box functions, and the experiments are still the queries from the input instance space. Bayesian optimization only focuses on reducing the uncertainties of the optimal points. The common acquisition functions of BO includes Probability of Improvement (PI), which is defined as the probability of improving upon the best value observed so far by querying the instance, and the Expected Improvement (EI), which takes the amount of expected improvement of the utility function by a query into consideration [55]. Upper Confidence Bound (UCB) combines the effect of exploitation and exploration linearly [77]. There is also mutual information based strategies called Predictive Entropy Search (PES), which maximizes the mutual information of the optimal points provided by the query [35]. We can also apply MOCU-based experimental design for Bayesian optimization, and the resulting strategy is equivalent to Knowledge Gradient (KG), which provides the one-step optimal policy based on the prior knowledge and the observations [26].

In optimal experimental design, an important consideration for theoretical analysis is the sampling efficiency of different strategies. Although this dissertation focuses on the convergence analysis and empirical evaluation of new strategies benchmarking with the state of the arts, there have been reported theoretical results on sample complexity analysis of some existing experimental design strategies. For example, Freund *et al.* [28] have shown that in the online active learning scenario, with a correct Bayesian prior on the set of hypotheses and the realizability assumption, the sample efficiency of QBC can be achieved with an exponential improvement over traditional passive learning. The theoretical result on the Agnostic Active (A^2) learning strategy has further demonstrated the exponential improvement on sample efficiency without the need of knowing the Bayesian prior [2]. Krause *et al.* [47] have quantified the performance difference between sequential and parallel strategies for active learning for regression problems. Srinivas *et al.* [77] have analyzed the convergence rate of the cumulative regret of UCB for BO, and have obtained a sublinear regret bound. There are also more recent relevant theoretical analyses in multi-arm bandits and more general reinforcement learning, which we consider out of the scope of the current dissertation.

1.3 Thesis Organization

MOCU-based experimental design methods have been applied on different scenarios [15, 18, 7]. However, there are still some unsolved challenges. 1. Usually MOCU-based methods are applied on the simple surrogate models with uncertainty on the model parameter. However, in practice, the uncertainty in the physical procedure cannot directly connect to the model uncertainty. 2. The one-step-look-ahead strategy used in MOCU-based method is myopic and can not guarantee the asymptotic performance. There is no theoretical discussion on the convergence of the MOCU-based methods. 3. The computation of MOCU-based acquisition functions is typically prohibitive as it requires incrementally retraining the model with every new experiment. Also in some cases, the acquisition function is not smooth and cannot be combined with gradient-based optimization techniques to efficiently explore the continuous experiment space.

Throughout this dissertation, we propose several Bayesian experimental design methods to address these challenges. We first derive robust linear filtering and experimental design for physical systems governed by stochastic differential equations (SDEs) under model uncertainty. Then we propose active learning methods for Bayesian classification problem with theoretical guarantee. Last but not the least, we develop efficient algorithms for MOCU-based active learning with Gaussian Process Classification (GPC) models. The organization of the rest of this proposal is as follows:

Chapter 2

In Chapter 2, we consider a linear filtering problem for physical systems modeled by SDEs. The model uncertainty is characterised by the uncertain parameters of the SDEs and the experiments measure the uncertain model parameters. Since the stochastic processes in SDE systems are time-variant, there is no closed form solution for the optimal filter. We discretize the processes and approximately solve the problem with matrix solutions. We further derive an intrinsically Bayesian robust (IBR) linear filter and an optimal experimental design acquisition function to determine the priority of the experiments. We apply the proposed method to an SDE-based pharmacokinetic two-compartment model to esimate drug concentration levels with IBR and MOCU calculated by Monte Carlo combined with Euler-Maruyama method. This chapter is based on [86].

Chapter 3

In Chapter 3 we discuss active learning problem, and the experiments query labels of candidates. We first analyses MOCU-based active learning and shows that MOCU-based active learning fails to converge to the optimal classifier due to the piece-wise linearity of MOCU as a function of the uncertain model distribution, although they can identify the optimal one-step queries. To address the stuck problem and keep the myopic sampling efficiency at the same time, we propose two active learning methods based on approximation functions of MOCU with partial or total strictly concavity. The Weighted MOCU (WMOCU) function multiplies a weight function with each summation term of MOCU with a weight; the weight is designed to generate strictly concave function for each nonzero term. Soft-MOCU (SMOCU) uses logsumexp function to approximate the maximum function, so the resulting function is a strictly concave function. The acquisition functions based on both approximation functions can guarantee the active learning procedures converge to the optimal classifier. We provides the convergence proof for both methods. The experiments with both synthetic and real-world datasets demonstrate the expected sample efficiency of WMOCU and SMOCU based active learning. This chapter is based on [88] and [87].

Chapter 4

In Chapter 4. we further develop efficient algorithms for MOCU-based active learning for Gaussian Process Classification (GPC) problems. GPC is a nonparametric model that works in continuous feature space. We study active learning in both pool-based scenario (discrete instance set) and query synthesis scenario (continuous instance space). To avoid retraining GPC for each query during the active learning procedure, we calculate the posterior predictive distribution through the joint predictive distribution of label pairs, the expression of which is derived as a one-dimensional integral. We further utilize the smooth property of SMOCU to enable efficient query synthesis active learning with gradient-based optimization technique, by deriving the chain rule of the gradient computation of the SMOCU reduction.

With the research focus on Bayesian learning and experimental design, we summarize our methodological contributions and discuss future research directions in **Chapter 5**.

2. MODEL-BASED ROBUST FILTERING AND EXPERIMENTAL DESIGN FOR STOCHASTIC DIFFERENTIAL EQUATION SYSTEMS*

2.1 Overview

We derive robust linear filtering and experimental design for systems governed by stochastic differential equations (SDEs) under model uncertainty. Given a model of signal and observation processes, an optimal linear filter is found by solving the Wiener-Hopf equation; with model uncertainty, it is desirable to derive a corresponding robust filter. This chapter assumes that the physical process is modeled via a SDE system with unknown parameters; the signals are degraded by blurring and additive noise. Due to time-dependent stochasticity in SDE systems, the system is nonstationary; and the resulting Wiener-Hopf equation is difficult to solve in closed form. Hence, we discretize the problem to obtain a matrix system to carry out the overall procedure. We further derive an Intrinsically Bayesian Robust (IBR) linear filter together with an optimal experimental design framework to determine the importance of SDE parameter(s). We apply the theory to an SDE-based pharmacokinetic two-compartment model to estimate drug concentration levels.

2.2 Introduction

It is common practice in signal processing to begin with a stochastic-process model (signal plus noise), a covariance (or power spectra) model, or a state-observation model, as with Kalman filtering. However, in a physical context, the signal model may be derived from a physical model, which can be a parameterized mathematical system. Hence, the properties of the signal, and of the resulting filter, depend on the physical model, and the signal parameters are expressed in terms of the parameters of the physical model. If there is uncertainty with regard to some parameters in the physical model, this uncertainty is propagated to the signal model, for instance, uncertainty in the covariance matrix. The key factor for the work presented in this chapter is that, if the uncertainty in

*Reprinted with permission from "Model-Based Robust Filtering and Experimental Design for Stochastic Differential Equation Systems." by Zhao, Guang, Xiaoning Qian, Byung-Jun Yoon, Francis J. Alexander, and Edward R. Dougherty, 2020. IEEE Transactions on Signal Processing, 68, 3849-3859, copyright © 2020 IEEE.

the physical model arises from lack of scientific knowledge and the uncertainty is characterized by a prior distribution governing the uncertain (random) parameters, thereby characterizing our scientific understanding of the uncertainty, then that prior distribution continues to govern the uncertain parameters in the signal model. In summary, both the signal model and its uncertainty are dictated by the physical model, and not hypothesized independently.

We focus on optimally filtering signals generated by stochastic differential equations (SDEs) when some parameters of the SDEs are uncertain. Given an SDE, the desired (random) signal satisfies the SDE and is derived from it. Its parameters are from the SDE, and to the extent that the latter ones are uncertain, the signal is uncertain.

We apply Intrinsically Bayesian Robust (IBR) filtering to the signal. An IBR filter is optimal relative to both the standard mean-square-error (MSE) for linear filtering (which leads to the Wiener-Hopf integral equation [39, 25]) and the uncertainty, that is, the prior distribution on the parameters of the SDE. With model uncertainty, the ordinary Wiener-Hopf equation is replaced by the effective Wiener-Hopf equation, which incorporates the expectation of the correlation functions across the uncertainty class.

When originally applied in [12], stationarity was assumed, thereby leading to the IBR Wiener filter expressed in terms of effective power spectra. Although a general continuous-time non-stationary effective Wiener-Hopf equation is also presented in [12], methods of solving non-stationary setups are not in discussed. Here, because the signal is nonstationary due to the SDE model in which we are operating, an IBR linear filter will have to be derived directly from the Wiener-Hopf equation, which must be done numerically. Different approaches have been proposed to approximate the nonstationary optimal linear filter. Here we discretize the effective Wiener-Hopf integral equation to obtain an approximate solution. Note that since there are no new observations, there is no state-observation pair and recursive filtering does not apply.

An IBR filter is robust in the sense that it performs best on average across the uncertainty class; however, it is not optimal relative to the true model. Since the true model is unknown, we quantify the cost of uncertainty relative to the MSE by averaging the loss of performance across the uncertainty class. The *mean objective cost of uncertainty* (MOCU) is the average increase in error across the uncertainty class arising from using an IBR filter rather than individual optimal filters for each model in the uncertainty class [84]. The MOCU provides a cost of the uncertainty relative to the objective. A key aspect of the work is to apply MOCU-based optimal experimental design [18] in the framework of signal models derived from physical models: determine the unknown parameter in the SDE physical system model whose experimental assessment optimally reduces the expected (residual) MOCU when the obtained parameter value is put into the model and a new IBR filter is derived. The procedure can be done iteratively to yield sequential experimental design. We would like to emphasize that here we have two types of data. One is the observations of the signals generated from the SDE model, which we aim to estimate using our IBR filter; the other is the experimental assessments of the parameters in the SDE model, which help reduce the model uncertainty and MOCU by experimental design.

The general idea is to tie physical modeling, optimal signal processing, and experimental design (here in the SDE framework). We will demonstrate aspects of the procedure via a synthetic example. Then we will apply it to an SDE-based pharmacokinetic two-compartment model that differentiates the body into a central compartment (plasma) and a peripheral compartment (tissues), and describes the relationship between the drug concentrations in the central and peripheral compartments, and the measurement of the drug concentration in the central compartment.

2.3 Background

In this section, we briefly review the background for IBR operators and optimal experimental design in the IBR context.

2.3.1 IBR Filtering

Optimal operator design involves a mathematical model for the physical system, a class of operators, and an optimization problem defined by a cost function:

$$\psi_{\text{opt}} = \arg\min_{\psi \in \mathcal{F}} C(\psi), \tag{2.1}$$

where $\mathcal{F}$ is the operator class and $C(\psi)$ is the cost of applying an operator ψ. With model uncertainty, the true model is assumed to belong to an *uncertainty class* of models parameterized by a vector $\theta \in \Theta$. We define an *intrinsically Bayesian robust (IBR) operator* by

$$\psi_{\mathrm{IBR}}^{\Theta} = \arg\min_{\psi \in \mathcal{F}} \mathbb{E}_{\Theta}[C_{\theta}(\psi)], \tag{2.2}$$

where each $\theta \in \Theta$ corresponds to a model and the *prior probability distribution* $\pi(\theta)$ quantifies our prior knowledge regarding the physical system. Note that the expectation is with respect to $\pi(\theta)$ on the uncertainty class Θ and $C_{\theta}(\psi)$ denotes the corresponding cost of applying ψ to the model θ [12, 84]. If there is no prior knowledge beyond the uncertainty class itself, then the prior can be taken to be uniform and $\pi(\theta)$ is noninformative. An IBR operator is robust in the sense that on average it performs well over the uncertainty class Θ.

When there is a data sample S, the prior can be updated to a posterior distribution $\pi^{*}(\theta) = \pi(\theta|S)$, and (2.2) then defines an *optimal Bayesian operator (OBO)* $\psi_{\mathrm{OBO}}^{\Theta}$ [63, 20]. An IBR operator is an OBO with $S = \varnothing$, namely, when there is no data but only prior knowledge constraining the model $\theta \in \Theta$.

In the signal filtering problem, the operators mentioned above are just filters. Signal filtering involves a joint random process $(X(t), Y(s))$, $t \in T, s \in S$, and optimal filtering involves estimating the signal $Y(s)$ at time s via a filter ψ given observations $\{X(t)\}_{t \in T}$. A filter $\psi \in \mathcal{F}$ is a mapping on the space $\mathcal{S}$ of possible observed signals and a cost function takes the form $C(Y(s), \widehat{Y}(s))$, with $\widehat{Y}(s) = \psi(X)(s)$. For fixed $s \in S$, an optimal filter is defined by (2.1) with $C(\psi) = C(Y(s), \psi(X)(s))$. With uncertainty, there is an *uncertainty class* $\{(X_{\theta}(t), Y_{\theta}(s)), t \in T, s \in S, \theta \in \Theta\}$. An IBR filter, or optimal Bayesian filter, is defined by (2.2) with $C_{\theta}(\psi) = C_{\theta}(Y_{\theta}(s), \psi(X_{\theta})(s))$ [12, 63].

Finding IBR filters involves developing a theory by which (2.2) can be solved – in a similar way as (2.1) is solved except that the *effective characteristics* pertaining to the full uncertainty class are used rather than the characteristics of a single signal model. An observation-signal pair

$(X(t), Y(s))$ is *solvable* under the function class $\mathcal{F}$ and cost C if there exists a solution to (2.1) under the processes. An observation-signal pair $(X_\Theta(t), Y_\Theta(s))$ is referred to as an *effective process* under the function class $\mathcal{F}$, uncertainty class Θ, and costs C if for all $\psi \in \mathcal{F}$,

$$\mathbb{E}_\Theta[C(Y_\theta(s), \psi(X_\theta)(s))] = C(Y_\Theta(s), \psi(X_\Theta)(s)). \tag{2.3}$$

If there exists a solvable effective process $(X_\Theta(t), Y_\Theta(s))$ with the optimal filter ψ_Θ, then $\psi^\Theta_{\text{IBR}} = \psi_\Theta$ [12].

Robust filter design goes back to the late 1970s, with robust Wiener filtering involving minimax optimality in regard to uncertain power spectra [49, 41, 61, 81]. Robust design was extended to nonlinear filters and placed into a Bayesian framework by assuming a prior probability distribution governing the uncertainty class, the aim being to find a filter with minimal expected error across the uncertainty class [34]. IBR filters are fully optimal under this framework.

Other robust filters include a minimax estimator (τ-robust) associated with τ-divergence space [94], a minimax estimator under covariance uncertainty with the given eigenvector matrix and bounded eigenvalues [24], a minimax estimator with an uncertain model matrix [23], and a distributed estimation formulation with model uncertainties [67].

Although we are not using recursive filters, for the sake of completeness we mention some robust Kalman filters. Adaptive Kalman filters simultaneously estimate the noise covariances along with the state estimation [54, 68]. Finite-impulse-response analogues have also been proposed [50, 76]. A regularized least-squares framework has been employed in which unknown parameters embody the deviation of the model parameters from their nominal values [69]. Another approach penalizes sensitivity of estimation relative to modeling error [89]. It has also been extended to the situation in which the observation can be randomly lost [90]. Last but not least, robust Kalman filtering has been addressed in the IBR framework [17, 14].

2.3.2 Experimental Design

While an IBR operator is optimal over the uncertainty class, it is likely to be suboptimal relative to the true model. This performance loss is the cost of uncertainty. For any $\theta \in \Theta$ and operator family $\mathcal{F}$, the *objective cost of uncertainty* relative to θ is $C_\theta(\psi_{\mathrm{IBR}}^{\Theta}) - C_\theta(\psi_\theta)$. The *mean objective cost of uncertainty* (MOCU) [84] is the expectation of this cost over all possible models:

$$\mathrm{M}_{\mathcal{F}}(\Theta) = \mathbb{E}_{\Theta}[C_\theta(\psi_{\mathrm{IBR}}^{\Theta}) - C_\theta(\psi_\theta)]. \tag{2.4}$$

While we have defined MOCU for an IBR operator relative to the prior, it can also be defined for an OBO relative to the posterior.

MOCU is used to choose experiments to optimally reduce the model uncertainty relevant to the operational objective. For example, given k experiments $T_1, ..., T_k$, where experiment T_i exactly determines the uncertain parameter θ_i in $\theta = (\theta_1, \theta_2, ..., \theta_k)$, the issue for experimental design is which experiment to conduct first. Let $\theta|\bar{\theta}_i = \theta|(\theta_i = \bar{\theta}_i)$ be the *conditional uncertainty vector* composed of all uncertain parameters other than θ_i with $\theta_i = \bar{\theta}_i$. $\Theta|\bar{\theta}_i = \{\theta|\bar{\theta}_i : \theta \in \Theta\}$ is the *reduced uncertainty class* given $\theta_i = \bar{\theta}_i$. The IBR operator for $\Theta|\bar{\theta}_i$ is denoted $\psi_{\mathrm{IBR}}^{\Theta|\bar{\theta}_i}$ and is called the *reduced IBR operator* relative to $\bar{\theta}_i$.

If the experiment T_i obtains the model parameter value $\bar{\theta}_i$, then the *remaining MOCU* given $\theta_i = \bar{\theta}_i$ is

$$\mathrm{M}_{\mathcal{F}}(\Theta|\bar{\theta}_i) = \mathbb{E}_{\Theta|\bar{\theta}_i}[C_{\theta|\bar{\theta}_i}(\psi_{\mathrm{IBR}}^{\Theta|\bar{\theta}_i}) - C_{\theta|\bar{\theta}_i}(\psi_{\theta|\bar{\theta}_i})], \tag{2.5}$$

where the expectation is relative to the conditional distribution $\pi(\theta|\bar{\theta}_i)$. The remaining MOCU is the MOCU for $\psi_{\mathrm{IBR}}^{\Theta|\bar{\theta}_i}$ relative to $\Theta|\bar{\theta}_i$.

Treating the remaining MOCU as a function of θ_i and taking the expectation with respect to $\pi(\theta_i)$ yields the expected remaining MOCU, given parameter θ_i,

$$\mathbb{E}_{\theta_i}[\mathrm{M}_{\mathcal{F}}(\Theta|\theta_i)] = \mathbb{E}_{\theta_i}[\mathbb{E}_{\Theta|\theta_i}[C_{\theta|\theta_i}(\psi_{\mathrm{IBR}}^{\Theta|\theta_i}) - C_{\theta|\theta_i}(\psi_{\theta|\theta_i})]], \tag{2.6}$$

which is called the *experimental design value* and denoted by $\mathrm{D}(\theta_i)$. An *optimal experiment* T_{i^*} is defined by

$$i^* = \underset{i=1,\ldots,k}{\arg\min}\ \mathrm{D}(\theta_i) \ = \underset{i=1,\ldots,k}{\arg\min}\ \mathrm{R}(\theta_i), \tag{2.7}$$

where

$$\mathrm{R}(\theta_i) = \mathbb{E}_{\theta_i}[\mathbb{E}_{\Theta|\theta_i}[C_{\theta|\theta_i}(\psi_{\mathrm{IBR}}^{\Theta|\theta_i})]] \tag{2.8}$$

is called the *residual IBR cost* for T_i, and θ_{i^*} is called the *primary parameter* [15]. The resulting T_{i^*} is the experiment that is expected to minimize the model uncertainty pertaining the cost. Experiments can be chosen in a greedy sequential manner by repeating the process for the remaining unknown parameters, or by using dynamical programming. This sequential experimental design procedure is illustrated in Fig. 2.1.

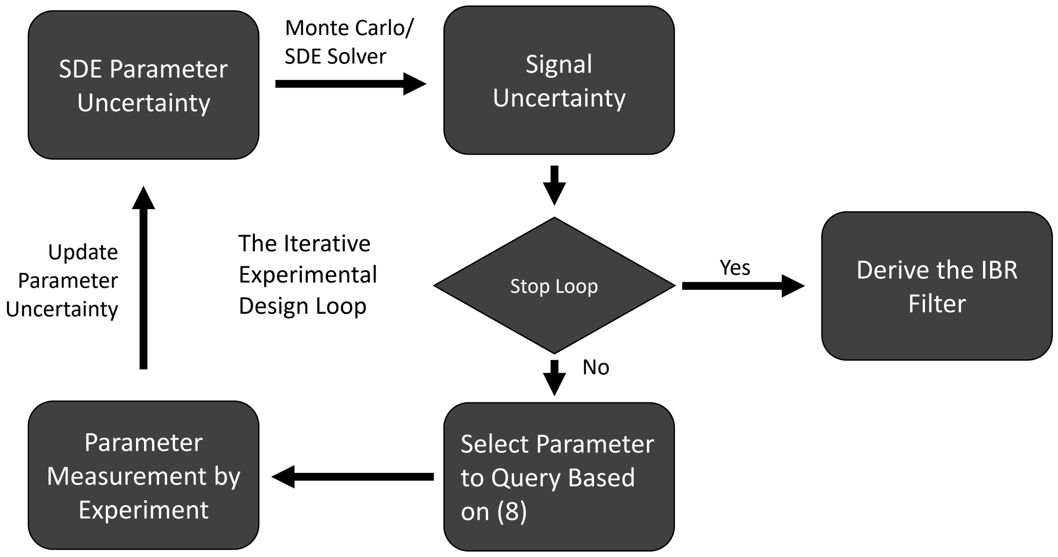

SDE Parameter Uncertainty
Monte Carlo/ SDE Solver
Signal Uncertainty
Update Parameter Uncertainty
The Iterative Experimental Design Loop
Stop Loop
Yes
Derive the IBR Filter
No
Parameter Measurement by Experiment
Select Parameter to Query Based on (8)

Figure 2.1: MOCU-based experimental design loop for robust filtering. Reprinted with permission from [86], copyright © 2020 IEEE.

Note that in the discussion above, we assume that the experiment T_i can determine θ_i exactly.

The strategy can be easily extended to more general cases with imprecise experiments [56]. When the value $\bar{\theta}_i$ obtained from the experiment T_i is imprecise with distribution $p(\bar{\theta}_i|\theta_i)$, the piror $\pi(\theta)$ can be accordingly updated to a posterior distribution $\pi(\theta|\bar{\theta}_i)$. Then the IBR filter $\psi_{\text{IBR}}^{\Theta|\bar{\theta}_i}$ defined by (2.2) is optimal with respect to the posterior $\pi(\theta|\bar{\theta}_i)$, and the residual IBR cost is still in the same form as (2.8).

2.4 IBR Linear Filter for Nonstationary Signals

Consider an uncertain signal model $(X_\theta, Y_\theta), \theta \in \Theta$, with the MSE cost function and the class of linear functions

$$\mathcal{F} = \left\{ \psi : \psi(X)(s) = \int_T g(s,t)X(t)dt \right\}. \tag{2.9}$$

The solvable class Φ consists of all process pairs (X, Y) such that $\psi(X)(s)$ has a finite second moment for any $g(s,t)$ and there exists $\widehat{g}(s,t)$ for which the Wiener-Hopf equation is satisfied:

$$R_{YX}(s,t) = \int_T \widehat{g}(s,u)R_X(u,t)du, \tag{2.10}$$

where $R_X(u,t)$ and $R_{YX}(s,t)$ are autocorrelation and cross-correlation functions, respectively.

With the uncertain signal model, we now define the *effective correlation functions* by $R_{\Theta,Y}(s,v) = \mathbb{E}_\Theta[R_{Y_\theta}(s,v)]$, $R_{\Theta,X}(t,u) = \mathbb{E}_\Theta[R_{X_\theta}(t,u)]$, and $R_{\Theta,YX}(s,t) = \mathbb{E}_\Theta[R_{Y_\theta X_\theta}(s,t)]$. As an autocorrelation function, $R_{X_\theta}(t,u)$ is conjugate symmetric and nonnegative definite for all $\theta \in \Theta$. $R_{\Theta,X}(t,u)$ has the same properties and is therefore also a valid autocorrelation function. It is straightforward to show that (2.3) is satisfied. If $(X_\Theta, Y_\Theta) \in \Phi$, meaning that the Wiener-Hopf equation relative to (X_Θ, Y_Θ) is satisfied, then (X_Θ, Y_Θ) is an effective process for the uncertainty class Θ and an IBR linear filter is given by the solution, $\widehat{g}(s,t)$, to the *effective Wiener-Hopf equation* [12]:

$$R_{\Theta,YX}(s,t) = \int_T \widehat{g}(s,u)R_{\Theta,X}(u,t)du. \tag{2.11}$$

All basic equations hold with characteristics replaced by effective characteristics $R_{\Theta,Y}$, $R_{\Theta,X}$, and $R_{\Theta,YX}$.

In the nonstationary case, the integral-form Wiener-Hopf equation can be difficult to solve in closed form, and numerical approximations are employed. The authors in [43] proposed a time-frequency formulation of the nonstationary linear filter, which can be approximately valid for underspread cases. It is also possible to approximately approach it by solving discrete-time Wiener-Hopf equations [78, 57]. Here we use the discrete-time approach to approximate the continuous time signal and observation with signal vector $\boldsymbol{Y} = \{Y(s_i)\}$ at N discrete time points $s_i, i \le N$ and observation vector $\boldsymbol{X} = \{X(t_j)\}$ at M discrete time points $t_j, j \le M$. The integral form of the Wiener-Hopf equation then turns into the following matrix form:

$$\boldsymbol{R}_{YX} = \widehat{\boldsymbol{G}}\boldsymbol{R}_X, \tag{2.12}$$

where $\boldsymbol{R}_X = \mathbb{E}[\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}]$, $\boldsymbol{R}_{YX} = \mathbb{E}[\boldsymbol{Y}\boldsymbol{X}^{\mathrm{T}}]$ are the autocorrelation and cross-correlation of the matrix form, respectively; and $\widehat{\boldsymbol{G}}$ is the matrix-form optimal filter. Similarly, the effective Wiener-Hopf equation in the matrix form can be written as:

$$\boldsymbol{R}_{\Theta,YX} = \widehat{\boldsymbol{G}}_{\Theta}\boldsymbol{R}_{\Theta,X}, \tag{2.13}$$

and the solution is

$$\widehat{\boldsymbol{G}}_{\Theta} = \boldsymbol{R}_{\Theta,YX}[\boldsymbol{R}_{\Theta,X}]^{+}, \tag{2.14}$$

where the superscript $+$ denotes the pseudoinverse. The error covariance matrix of $\hat{\boldsymbol{Y}}_{\mathrm{IBR}} = \widehat{\boldsymbol{G}}_{\Theta}\boldsymbol{X}_{\theta}$ is

$$\begin{aligned}
&\mathbb{E}_{\Theta}[\mathbb{E}[(\hat{\boldsymbol{Y}}_{\mathrm{IBR}} - \boldsymbol{Y}_{\theta})(\hat{\boldsymbol{Y}}_{\mathrm{IBR}} - \boldsymbol{Y}_{\theta})^{\mathrm{T}}]] \\
&= \mathbb{E}_{\Theta}[\mathbb{E}[(\hat{\boldsymbol{Y}}_{\mathrm{IBR}} - \boldsymbol{Y}_{\theta})\hat{\boldsymbol{Y}}_{\mathrm{IBR}}^{\mathrm{T}}]] - \mathbb{E}_{\Theta}[\mathbb{E}[(\hat{\boldsymbol{Y}}_{\mathrm{IBR}} - \boldsymbol{Y}_{\theta})\boldsymbol{Y}_{\theta}^{\mathrm{T}}]] \\
&= -\mathbb{E}_{\Theta}[\mathbb{E}[(\hat{\boldsymbol{Y}}_{\mathrm{IBR}} - \boldsymbol{Y}_{\theta})\boldsymbol{Y}_{\theta}^{\mathrm{T}}]] \\
&= \mathbb{E}_{\Theta}[\mathbb{E}[\boldsymbol{Y}_{\theta}\boldsymbol{Y}_{\theta}^{\mathrm{T}}]] - \mathbb{E}_{\Theta}[\mathbb{E}[\hat{\boldsymbol{Y}}_{\mathrm{IBR}}\boldsymbol{Y}_{\theta}^{\mathrm{T}}]] \\
&= \boldsymbol{R}_{\Theta,Y} - \boldsymbol{R}_{\Theta,YX}[\boldsymbol{R}_{\Theta,X}]^{+}\boldsymbol{R}_{\Theta,YX}^{\mathrm{T}},
\end{aligned} \tag{2.15}$$

where the second equality holds because $\hat{\boldsymbol{Y}}_{\mathrm{IBR}} - \boldsymbol{Y}_\theta$ is orthogonal to any linear combination of $\boldsymbol{Y}_\theta$. Especially, $\hat{\boldsymbol{Y}}_{\mathrm{IBR}}$ as an IBR filter achieves the Bayesian optimality [42]. The last equality follows from (2.14). The MSE of the IBR filter is just the trace of the error covariance matrix.

2.4.1 MOCU for the Discrete Wiener-Hopf Equation

With the derived IBR Wiener filter, we can quantify the model uncertainty in the MOCU framework relative to the IBR filter:

$$\begin{aligned}
M_{\mathcal{F}}(\Theta) &= \mathbb{E}_\Theta[C_\theta(\widehat{\boldsymbol{G}}_\Theta) - C_\theta(\widehat{\boldsymbol{G}}_\theta)] \\
&= \mathbb{E}_\Theta[C_\theta(\widehat{\boldsymbol{G}}_\Theta)] - \mathbb{E}_\Theta[C_\theta(\widehat{\boldsymbol{G}}_\theta)] \\
&= \mathrm{tr}(\boldsymbol{R}_{\Theta,Y} - \boldsymbol{R}_{\Theta,YX}[\boldsymbol{R}_{\Theta,X}]^+\boldsymbol{R}^{\mathrm{T}}_{\Theta,YX}) - \mathbb{E}_\Theta[\mathrm{tr}(\boldsymbol{R}_{\theta,Y} - \boldsymbol{R}_{\theta,YX}[\boldsymbol{R}_{\theta,X}]^+\boldsymbol{R}^{\mathrm{T}}_{\theta,YX})] \\
&= -\mathrm{tr}(\boldsymbol{R}_{\Theta,YX}[\boldsymbol{R}_{\Theta,X}]^+\boldsymbol{R}^{\mathrm{T}}_{\Theta,YX}) + \mathbb{E}_\Theta[\mathrm{tr}(\boldsymbol{R}_{\theta,YX}[\boldsymbol{R}_{\theta,X}]^+\boldsymbol{R}^{\mathrm{T}}_{\theta,YX})]. \qquad (2.16)
\end{aligned}$$

Experimental design for IBR Wiener filtering involves minimizing the IBR residual cost: $i^* =$

$$\begin{aligned}
&\arg\min_{i\in 1,\dots,k} \mathbb{E}_{\bar{\theta}_i}[\mathrm{tr}(\boldsymbol{R}_{\Theta|\bar{\theta}_i,Y} - \boldsymbol{R}_{\Theta|\bar{\theta}_i,YX}[\boldsymbol{R}_{\Theta|\bar{\theta}_i,X}]^+\boldsymbol{R}^{\mathrm{T}}_{\Theta|\bar{\theta}_i,YX})] \qquad (2.17) \\
= &\arg\max_{i\in 1,\dots,k} \mathbb{E}_{\bar{\theta}_i}[\mathrm{tr}(\boldsymbol{R}_{\Theta|\bar{\theta}_i,YX}[\boldsymbol{R}_{\Theta|\bar{\theta}_i,X}]^+\boldsymbol{R}^{\mathrm{T}}_{\Theta|\bar{\theta}_i,YX})], \qquad (2.18)
\end{aligned}$$

where $\boldsymbol{R}_{\Theta|\bar{\theta}_i,Y} = \mathbb{E}_{\Theta|\bar{\theta}_i}[\boldsymbol{R}_{Y_\theta}]$, $\boldsymbol{R}_{\Theta|\bar{\theta}_i,X} = \mathbb{E}_{\Theta|\bar{\theta}_i}[\boldsymbol{R}_{X_\theta}]$, $\boldsymbol{R}_{\Theta|\bar{\theta}_i,YX} = \mathbb{E}_{\Theta|\bar{\theta}_i}[\boldsymbol{R}_{Y_\theta X_\theta}]$, and the second equation holds for the reason that

$$\mathbb{E}_{\bar{\theta}_i}[\boldsymbol{R}_{\Theta|\bar{\theta}_i,Y}] = \mathbb{E}_{\bar{\theta}_i}[\mathbb{E}_{\Theta|\bar{\theta}_i}[\boldsymbol{R}_{Y_\theta}]] = \mathbb{E}_\Theta[\boldsymbol{R}_{Y_\theta}] = \boldsymbol{R}_{\Theta,Y} \qquad (2.19)$$

is unrelated to the index i. As shown in (2.17) and (2.18), the IBR residual cost is only a function of the auto- and cross-correlations, in the form of the MSE of the linear IBR filter. Therefore there is no need to re-derive the filter during the experimental design procedure.

2.5 IBR Linear Filter and Experimental Design with Stochastic Differential Equations

Stochastic differential equations (SDEs) are widely applied for stochastic process modeling in areas such as pharmacology [83], population biology [6, 33] and mathematical finance [53]. In addition to the differential equations governing the processes under study, SDEs include diffusion processes to model potential random effects disturbing the processes of interest. Usually the diffusion process is a Wiener process. Assume that the n-dimensional random process under study, $\boldsymbol{Y}(t) \in \mathcal{Y} \subseteq \mathbb{R}^n$, is defined within the time interval $t \in [0, T]$; and the corresponding SDE is driven by an m-dimensional Wiener process $\boldsymbol{W}(t)$. Then the typical form of an Itô SDE is [1]:

$$\mathrm{d}\boldsymbol{Y}(t) = \boldsymbol{f}(t, \boldsymbol{Y}(t))\mathrm{d}t + \boldsymbol{g}(t, \boldsymbol{Y}(t))\mathrm{d}\boldsymbol{W}(t), \tag{2.20}$$

where $\boldsymbol{f} : [0, T] \times \mathbb{R}^n \to \mathbb{R}^n$, $\boldsymbol{g} : [0, T] \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are the drift vector and diffusion matrix, respectively.

If $\boldsymbol{f}$ and $\boldsymbol{g}$ are in the linear form shown in (2.21), the solutions of the corresponding SDEs can be Gaussian processes. Assume the functions $\boldsymbol{f}$ and $\boldsymbol{g}$ are given by

$$\begin{aligned} \boldsymbol{f}(t, \boldsymbol{Y}(t)) &= \boldsymbol{A}(t)\boldsymbol{Y}(t) + \boldsymbol{a}(t), \\ \boldsymbol{g}(t, \boldsymbol{Y}(t)) &= \boldsymbol{B}(t), \end{aligned} \tag{2.21}$$

where $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$ are matrices of size $n \times n$ and $n \times m$, respectively, and $\boldsymbol{a}(t)$ is a vector of size n. The resulting SDE takes the form

$$\mathrm{d}\boldsymbol{Y}(t) = (\boldsymbol{A}(t)\boldsymbol{Y}(t) + \boldsymbol{a}(t))\mathrm{d}t + \boldsymbol{B}(t)\mathrm{d}\boldsymbol{W}(t), \boldsymbol{Y}(0) = c. \tag{2.22}$$

The initial-valued SDE has a unique solution if and only if the initial condition c is either a constant or a Gaussian distributed random variable. The mean and auto-correlation of the Gaussian process are given by

$$\boldsymbol{m}(t_i) = \boldsymbol{\Phi}(t_i)(\mathbb{E}[c] + \int_0^{t_i} \boldsymbol{\Phi}(s)^{-1} a(s) \mathrm{d}s) \tag{2.23}$$

and

$$\begin{aligned}\boldsymbol{\Psi}(t_i, t_j) = & \boldsymbol{\Phi}(t_i)\Big(\mathbb{E}\big[(c - \mathbb{E}[c])(c - \mathbb{E}[c])^{\mathrm{T}}\big] \\ & + \int_0^{t_i} \boldsymbol{\Phi}(u)^{-1} \boldsymbol{B}(u) \boldsymbol{B}(u)^{\mathrm{T}} (\boldsymbol{\Phi}(u)^{-1})^{\mathrm{T}} \mathrm{d}u\Big) \boldsymbol{\Phi}(t_j)^{\mathrm{T}},\end{aligned} \tag{2.24}$$

where $0 \le t_i \le t_j \le T$ and $\boldsymbol{\Phi}(t)$ is the fundamental matrix of the ordinary differential equation

$$\mathrm{d}\boldsymbol{Y}(t) = \boldsymbol{A}(t)\boldsymbol{Y}(t)\mathrm{d}t. \tag{2.25}$$

When there is no closed-form solution, approximate numerical solutions of SDEs can be obtained by the Euler-Maruyama method [46]: Partition the interval $[0, T]$ into N equal subintervals of the width $\Delta t = T/N$: $0 = t_0 < t_1 < \ldots < t_N = T$. Then the numerical solution to the SDE is computed recursively by the difference equation:

$$\boldsymbol{Y}_{n+1} = \boldsymbol{Y}_n + \boldsymbol{f}(t_n, \boldsymbol{Y}_n)\Delta t + \boldsymbol{g}(t_n, \boldsymbol{Y}_n)\Delta \boldsymbol{W}_n, \tag{2.26}$$

where $\Delta \boldsymbol{W}_n = \boldsymbol{W}_{t_{n+1}} - \boldsymbol{W}_{t_n}$ is a Gaussian random vector with independent components and the variance of each component is Δt. Monte Carlo discrete samples of $\boldsymbol{Y}(t)$ can be generated according to (2.26), based on which we can estimate the stochastic characteristics.

In this chapter, we consider IBR filtering and optimal experimental design for the stochastic signal $\boldsymbol{Y}(t)$ described by an SDE with a vector $\theta = (\theta_1, \theta_2, ..., \theta_k)$ of uncertain parameters, so that $\boldsymbol{Y}(t)$ satisfies the SDE

$$\mathrm{d}\boldsymbol{Y}(t) = \boldsymbol{f}(t, \boldsymbol{Y}(t); \theta)\mathrm{d}t + \boldsymbol{g}(t, \boldsymbol{Y}(t); \theta)\mathrm{d}\boldsymbol{W}(t). \tag{2.27}$$

The model uncertainty can be characterized by $\pi(\theta)$, the prior distribution of θ. Denote the observation of $\boldsymbol{Y}(t)$ as $\boldsymbol{X}(t)$. Assume the observation procedure follows a linear observation model:

$$\boldsymbol{X}(t) = \int_0^T \boldsymbol{Y}(s)h(s,t)\mathrm{d}s + n(t), \tag{2.28}$$

where $h(s,t)$ is the blurring function and $n(t)$ is white noise. We derive the IBR linear filter to estimate $\boldsymbol{Y}(t)$ from $\boldsymbol{X}(t)$. The function class $\mathcal{F}$ is defined by (2.9) and the MSE is used as the cost function. Among the experiments that can exactly determine one of the uncertain parameters, we aim to predict the one minimizing the design value defined in (2.6).

2.6 Computational Complexity Analysis

Here we analyze the complexity of optimal experimental design for SDE model-based filtering considered in this chapter. Assume the dimensions of the signal vector $\boldsymbol{Y}$ and observation vector $\boldsymbol{X}$ are N_y and N_x, respectively. Note that N_y and N_x are equal to the multiplication of the number of discrete time points for discrete approximation and the channel numbers of the corresponding signal and observation processes. In addition, we assume that we have k uncertain parameters in the SDE system and therefore there are k possible experiments to specify each parameter for our experimental design setup, which requires solving the optimization problem in (2.18) over k parameters.

The objective function in (2.18) involves the computation of the expectation over $\bar{\theta}_i$, which can be calculated by Monte Carlo (MC) integration. Assume we sample M_1 samples of $\theta_i^{(j)}, j \leq M_1$. Given each $\theta_i^{(j)}$, if we have closed-form effective correlation matrices in (2.18), we just need to calculate the matrix multiplication and the trace given effective correlation matrices inside the expectation. First, computing the pseudoinverse $[\boldsymbol{R}_{\Theta|\theta_i,X}]^+$ has cubic complexity $O((N_x)^3)$. The matrix multiplication to derive $A = \boldsymbol{R}_{\Theta|\bar{\theta}_i,YX}[\boldsymbol{R}_{\Theta|\bar{\theta}_i,X}]^+$ has complexity of $O(N_y(N_x)^2)$ and calculating the trace $tr(A\boldsymbol{R}^{\mathrm{T}}_{\Theta|\bar{\theta}_i,YX})$ has the complexity $O(N_xN_y)$. The complexity of the matrix calculations for each sample is $O(N_y(N_x)^2 + (N_x)^3)$, and therefore the complexity of optimal experimental design by solving (2.18) is $O(kM_1(N_y(N_x)^2 + (N_x)^3))$.

In practice, there is typically no closed-form solution to the underlying SDE system modeling the signal process, hence there is no closed-form expression for effective correlation matrices. In such cases, we would also need to estimate the effective correlation matrices in (2.18) by MC sampling in addition to the matrix calculations analyzed above. Assume we generate M_2 samples of $(\theta|\theta_i^{(j)}, \boldsymbol{Y}^{(j)}, \boldsymbol{X}^{(j)}), j \leq M_2$, where $\boldsymbol{Y}^{(j)}$ can be generated by (2.26) and $\boldsymbol{X}^{(j)}$ by (2.28). Due to the Markovian property of (2.26), the complexities of sampling $\boldsymbol{Y}^{(j)}$ and $\boldsymbol{X}^{(j)}$ are all linear. The effective cross-correlation is estimated by:

$$\boldsymbol{R}_{\Theta|\bar{\theta}_i,YX} = \frac{1}{M_2}\sum_{j=1}^{M_2} \boldsymbol{Y}^{(j)}(\boldsymbol{X}^{(j)})^{\mathrm{T}}, \tag{2.29}$$

with the complexity $O(M_2 N_y N_x)$. Similarly, the complexity of estimating the effective auto-correlation of $\boldsymbol{X}$ is $O(M_2 N_x^2)$. With these, the complexity of optimal experimental design is

$$O(kM_1[N_y(N_x)^2 + (N_x)^3 + M_2(N_x^2 + N_y N_x)]). \tag{2.30}$$

Note that the MC integration procedure for effective correlation matrix estimation has to sample the uncertainty class of all k parameters and M_2 can grow exponentially with k.

2.7 Synthetic Experiments

To demonstrate the performance of the proposed robust filtering and optimal experimental design methods, we first consider a synthetic example, which assumes that the original signal $\boldsymbol{Y}(t)$ is generated by an SDE of the form in (2.22). Assume $\boldsymbol{Y}(t)$ is a two-channel signal and the parameters of the corresponding SDE are given by

$$
\begin{aligned}
\mathbf{A}(t) &= \frac{\theta_1}{100}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \\
\mathbf{a}(t) &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \\
\mathbf{B}(t) &= 0.1\begin{pmatrix} 1 & \theta_2 \\ \theta_2 & 1 \end{pmatrix}, \\
\boldsymbol{Y}(0) &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad (2.31)
\end{aligned}
$$

where $\theta = (\theta_1, \theta_2)$ is the uncertain parameter vector, $\boldsymbol{Y}(t)$ is defined within the time interval $[0, T = 100]$.

$\boldsymbol{X}(t)$ is the observation of $\boldsymbol{Y}(t)$, which is corrupted by a blurring function $h(t)$ with additive noise $\boldsymbol{N}(t)$:

$$\boldsymbol{X}(t) = \int_0^T h(t-s)\boldsymbol{Y}(s)ds + \boldsymbol{N}(t), \qquad (2.32)$$

where

$$h(t) = \frac{1}{B}(\text{sgn}(t) - \text{sgn}(t-B)), \qquad (2.33)$$

with $B = 10$, is a scalar function, so the blurring effect is the same for both channels. The variances of additive noise for both channels are the same, $\sigma^2 = 0.01$.

As mentioned earlier, the SDE has a unique solution as a Gaussian process. Therefore, we can obtain a closed-form expression of correlations between $\boldsymbol{Y}(t)$ and $\boldsymbol{X}(t)$. Let's begin with the fundamental matrix of (2.25):

$$\Phi(t) = \begin{pmatrix} e^{\theta_1 t} & 0 \\ 0 & e^{\theta_1 t} \end{pmatrix}, \qquad (2.34)$$

with $\mathrm{d}\Phi(t)/\mathrm{d}t = \boldsymbol{A}(t)\Phi(t)$. The auto-correlation of $\boldsymbol{Y}(t)$ can be calculated by (2.24):

$$\begin{aligned}
&\boldsymbol{R}_Y(t_i, t_j) \\
&= \boldsymbol{\Phi}(t_i)\left(\int_{t_0}^{t_i} \boldsymbol{\Phi}(u)^{-1}\mathbf{B}(u)\mathbf{B}(u)^{\mathrm{T}}(\boldsymbol{\Phi}(u)^{-1})^{\mathrm{T}}\mathrm{d}u\right)\boldsymbol{\Phi}(t_j)^{\mathrm{T}} \\
&= \frac{1}{2\theta_1}\{e^{\theta_1(t_i+t_j)} - e^{\theta_1(t_j-t_i)}\}\begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix} \\
&= r_Y(t_i, t_j)\begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix},
\end{aligned} \tag{2.35}$$

where

$$r_Y(t_i, t_j) = \frac{1}{2\theta_1}\{e^{\theta_1(t_i+t_j)} - e^{\theta_1(t_j-t_i)}\}. \tag{2.36}$$

Equation (2.35) holds for $t_j \geq t_i \geq 0$, and we have $\boldsymbol{R}_Y(t_i, t_j) = \boldsymbol{R}_Y(t_j, t_i)$. Based on the observation model in (2.32), the cross-correlation is

$$\begin{aligned}
&\boldsymbol{R}_{YX}(t_i, t_j) \\
=&\int_0^T h(t_j - s)\boldsymbol{R}_Y(t_i, s)\mathrm{d}s \\
=&\int_0^T h(t_j - s)r_Y(t_i, s)\mathrm{d}s\begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix} \\
=&r_{YX}(t_i, t_j)\begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix},
\end{aligned} \tag{2.37}$$

with

$$r_{YX}(t_i, t_j) = \int_0^T h(t_j - s)r_Y(t_i, s)\mathrm{d}s. \tag{2.38}$$

The auto-correlation of $\boldsymbol{X}(t)$ is

$$\begin{aligned}
&\boldsymbol{R}_X(t_i, t_j) \\
=&\int_0^T \int_0^T h(t_i - s)\boldsymbol{R}_Y(s,u)h(t_j - u)\mathrm{d}s\mathrm{d}u + \sigma^2\delta(t_i - t_j)I_2 \\
=&\int_0^T \int_0^T h(t_i - s)r_Y(s,u)h(t_j - u)\mathrm{d}s\mathrm{d}u \cdot \\
&\begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix} + \sigma^2\delta(t_i - t_j)I_2 \\
=&r_X(t_i, t_j)\begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix} + \sigma^2\delta(t_i - t_j)I_2, \qquad (2.39)
\end{aligned}$$

with

$$r_X(t_i, t_j) = \int_0^T \int_0^T h(t_i - s)r_Y(s,u)h(t_j - u)\mathrm{d}s\mathrm{d}u. \qquad (2.40)$$

The integrals in (2.38) and (2.40) can be calculated directly and have piecewise closed-form expressions depending on the value relationships between t_i, t_j, B and T.

As noted previously, we consider the discrete filtering problem. The signals from the SDE system are sampled at discrete time points $t = 0, 1, 2, ..., 100$. We denote the flattened discrete time vectors of $\boldsymbol{X}(t)$ and $\boldsymbol{Y}(t)$ as:

$$\boldsymbol{X}^N = (X_0^1, \ldots, X_N^1, X_1^2, \ldots, X_N^2)^{\mathrm{T}} \qquad (2.41)$$

and

$$\boldsymbol{Y}^N = (Y_0^1, \ldots, Y_N^1, Y_1^2, \ldots, Y_N^2)^{\mathrm{T}}, \qquad (2.42)$$

where Y_j^i and X_j^i indicate the signal and observation values at the i-th channel and time j, respectively, for $i = 1, 2$ and $0 \leq j \leq N$.

Then the matrix forms of correlations are

$$\boldsymbol{R}_{\theta,YX}^N = \begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix} \otimes \boldsymbol{r}_{YX}^N, \tag{2.43}$$

$$\boldsymbol{R}_{\theta,X}^N = \begin{pmatrix} 1+\theta_2^2 & 2\theta_2 \\ 2\theta_2 & 1+\theta_2^2 \end{pmatrix} \otimes \boldsymbol{r}_{X}^N + \sigma^2 I_{2N}, \tag{2.44}$$

where $\otimes$ indicates the Kronecker product, and

$$\boldsymbol{r}_{YX}^N = \begin{pmatrix} r_{YX}(t_0,t_0) & \cdots & r_{YX}(t_0,t_N) \\ \vdots & \ddots & \vdots \\ r_{YX}(t_N,t_0) & \cdots & r_{YX}(t_N,t_N) \end{pmatrix},$$
$$\boldsymbol{r}_{X}^N = \begin{pmatrix} r_{X}(t_0,t_0) & \cdots & r_{X}(t_0,t_N) \\ \vdots & \ddots & \vdots \\ r_{X}(t_N,t_0) & \cdots & r_{X}(t_N,t_N) \end{pmatrix} \tag{2.45}$$

are corresponding matrix forms of $r_{YX}(t_i,t_j)$ and $r_X(t_i,t_j)$.

2.7.1 IBR Filter Performance

To examine the performance of the IBR filter $\widehat{\boldsymbol{G}}_\Theta = \boldsymbol{R}_{\Theta,YX}^N[\boldsymbol{R}_{\Theta,X}^N]^+$, fix $\theta_1 = 1$ and let θ_2 be uniformly distributed over the interval $(-1,1)$. We then have a closed-form expression for the expectation over θ:

$$\boldsymbol{R}_{\Theta,YX}^N = \begin{pmatrix} 4/3 & 0 \\ 0 & 4/3 \end{pmatrix} \otimes \boldsymbol{r}_{YX}^N, \tag{2.46}$$

$$\boldsymbol{R}_{\Theta,X}^N = \begin{pmatrix} 4/3 & 0 \\ 0 & 4/3 \end{pmatrix} \otimes \boldsymbol{r}_{X}^N + \sigma^2 I_{2N}. \tag{2.47}$$

Note that the effective correlation doesn't correspond to any specific value of θ_2. Inserting (2.46) and (2.47) to (2.14) yields the matrix-form IBR filter $\widehat{\boldsymbol{G}}_{\Theta}$.

To show the performance of the IBR filter, we compare it with the optimal filter for $\theta_2 = 0.8$ and the recently developed τ-robust filter which is robust with bounded τ-divergence and is the optimal filter based on the nominal statistics with respect to $\theta_2 = 0$ [94]. The result for applying the three filters on the observation of the signal generated by the SDE with $\theta_2 = 0.8$ is shown in Fig. 2.2. Note that the IBR filter has a performance (Mean Square Error (MSE) = 2.6015) fairly close to the filter that is optimal for $\theta_2 = 0.8$ (MSE = 2.5269) and performs better than the τ-robust filter (MSE = 2.6121).

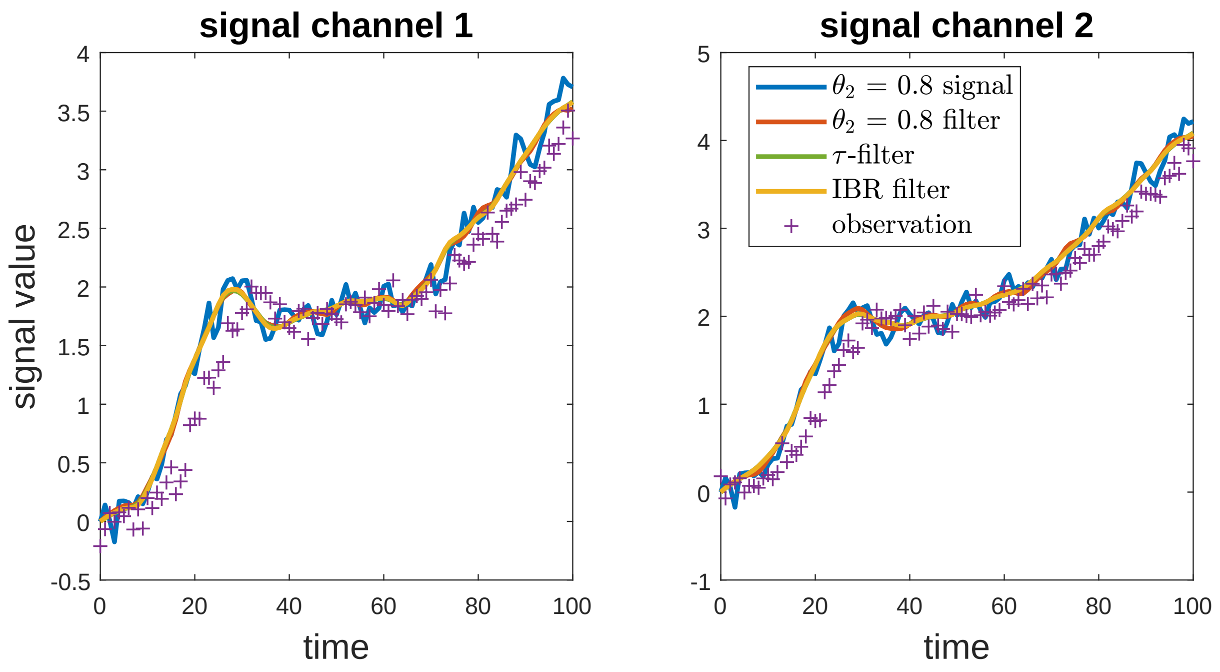


Figure 2.2: One sample from the SDE (2.22), (2.31) with $\theta_2 = 0.8$. Left and right sub-figures show the signals from the first and the second channels, respectively. The original signals are in blue. The filtered signals based on the optimal filter for $\theta_2 = 0.8$ are in red. The filtered signals using the τ-robust filter are in green. The IBR filtered signals are in yellow. The corrupted observations are in purple crosses. Reprinted with permission from [86], copyright © 2020 IEEE.

Next we applied the same filters on the observation of the signal generated by the SDE with

$\theta_2 = -0.7$, the result being shown in Fig. 2.3. Here the IBR filter still maintains relatively good performance (MSE = 2.2360), followed by the τ-robust filter with MSE = 2.2416, but the filter optimal for $\theta_2 = 0.8$ shows a significantly degraded performance (MSE = 4.4331) due to the model mismatch.

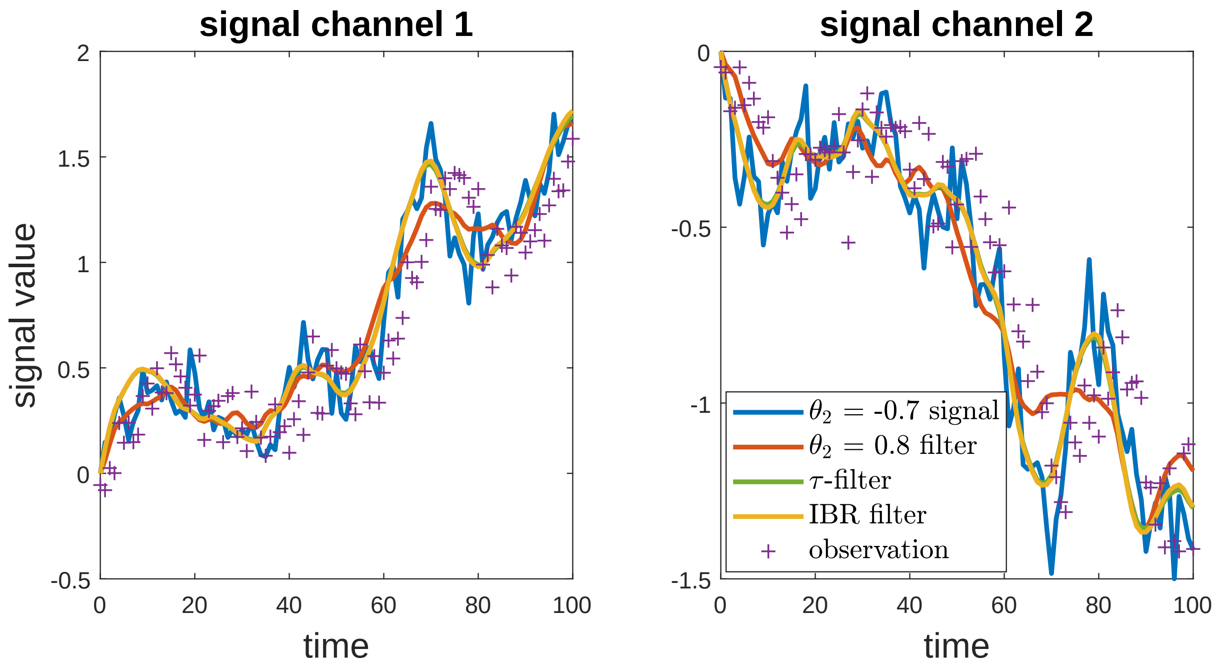


Figure 2.3: One sample simulated from the SDE (2.22), (2.31) with $\theta_2 = -0.7$. Left and right sub-figures show the signals from the first and the second channels, respectively. The original signals are in blue. The filtered signals based on the optimal Wiener filter for $\theta_2 = 0.8$ are in red. The filtered signals using the τ-robust filter are in green. The IBR filtered signals are in yellow. The corrupted observations are in purple crosses. Reprinted with permission from [86], copyright © 2020 IEEE.

2.7.2 Optimal Experimental Design

The optimal experimental design problem is to determine which one of the two parameters, θ_1 or θ_2, should be specified first to minimize the cost due to uncertainty. Taking MSE for signal filtering as the cost, the cost function for experimental design is the residual IBR cost of two parameters, expressed as:

$$\mathrm{R}(\theta_1) = \mathbb{E}_{\theta_1}[\mathbb{E}_{\Theta|\theta_1}[C_{\Theta|\theta_1}(\widehat{\boldsymbol{G}}_{\Theta|\theta_1})]], \tag{2.48}$$

$$\mathrm{R}(\theta_2) = \mathbb{E}_{\theta_2}[\mathbb{E}_{\Theta|\theta_2}[C_{\Theta|\theta_2}(\widehat{\boldsymbol{G}}_{\Theta|\theta_2})]]. \tag{2.49}$$

We assume θ_1 and θ_2 are independent. θ_2 is distributed over the interval $(-1, 1)$ as:

$$\theta_2 = 2\epsilon - 1, \quad \epsilon \sim Beta(\beta, \beta), \tag{2.50}$$

with β the distribution parameter. θ_1 is uniformly distributed as

$$\theta_1 \sim U(5 - L/2, \quad 5 + L/2), \tag{2.51}$$

with distribution parameter L.

In our simulations, we set three different values for $\beta = 0.5, 1.5, 5$ and $L = 0.5, 1.5, 2$, so that we have 9 combinations of distribution hyperparameters. The residual IBR cost is calculated by Monte Carlo sampling. For $\mathrm{R}(\theta_1)$, for each given pair of distribution parameters, 200 sample pairs of θ_1 are taken for Monte Carlo computation, and for each θ_1, the inner term has a closed-form expression as in (2.17):

$$\begin{aligned} & \mathbb{E}_{\Theta|\theta_1}[C_{\Theta|\theta_1}(\widehat{\boldsymbol{G}}_{\Theta|\theta_1})] \\ = \; & \mathbb{E}_{\theta_1}[\mathrm{tr}(\boldsymbol{R}_{\Theta|\theta_1,Y} - \boldsymbol{R}_{\Theta|\theta_1,YX}[\boldsymbol{R}_{\Theta|\theta_1,X}]^{+}\boldsymbol{R}'_{\Theta|\theta_1,YX})]. \end{aligned}$$

$\mathrm{R}(\theta_2)$ is calculated similarly by Monte Carlo sampling. We just need to substitute θ_1 with θ_2 in the above expression to calculate $\mathbb{E}_{\Theta|\theta_2}[C_{\Theta|\theta_2}(\widehat{\boldsymbol{G}}_{\Theta|\theta_2})]$. The residual IBR costs are shown in Fig. 2.4. The variances of the two parameters are $\mathrm{Var}(\theta_2) = \frac{1}{2\beta+1}$ and $\mathrm{Var}(\theta_1) = \frac{L^2}{12}$.

From the figure we can see how the variances of the uncertain parameters influence the IBR residuals. The variance of θ_2 has a higher influence on the IBR residuals than the variance of θ_1 does. As $\mathrm{Var}(\theta_2)$ increases, both IBR residuals increase as a larger variance introduces more uncertainty in the model. But when Var(θ_2) is large, R(θ_2) is smaller than R(θ_1), because estimating θ_2 can reduce the uncertainty (or the cost thereof) more than estimating θ_1, pertaining to the filtering performance in this case. For small Var(θ_2), we have the opposite conclusion.

To further illustrate the strength of the MOCU-based experimental design, here we perform experiments with a more complicated uncertainty class and compare its performance with both entropy-based [9] and random sequential experimental design.

Assume that we have four uncertain parameters, $\theta = (\theta_1, \theta_2, B, \sigma)$ in the model described by (2.31)-(2.33). These uncertain model parameters follow independent uniform distributions. For each parameter, we assume an experiment can be performed to obtain its value. In addition we assume all the parameter measurements have Gaussian errors. We perform a sequential experimental design to decide which model parameter to measure in each iteration so that we can most effectively improve the filtering performance within a relatively small number of iterations. For this experimental design problem, we compare the MOCU-based strategy described by (2.18) with both entropy-based strategy and random strategy. The entropy-based strategy chooses the experiment to measure the parameter with the largest Shannon entropy; and the random strategy simply chooses one out of the uncertain parameters in a random fashion. To compare the different strategies in different cases, we set three different groups of parameter distributions for sequential experimental design: (1) $\theta_1 \sim U(3,6)$, $\theta_2 \sim U(-2,2)$, $B \sim U(8,10.5)$, $\sigma \sim U(0.01,1.2)$; (2) $\theta_1 \sim U(3,6)$, $\theta_2 \sim U(-1.4,1.4)$, $B \sim U(8,10.5)$, $\sigma \sim U(0.01,1.2)$; (3) $\theta_1 \sim U(3.7,6)$, $\theta_2 \sim U(-1,1)$, $B \sim U(8,10.5)$, $\sigma \sim U(0.01,2)$. For all the cases, the parameters have Gaussian measurement error with a variance $\sigma_\epsilon^2 = 0.05$. In each cases, we randomly generate 100 groups of parameters, and perform sequential experimental design following the three strategies. After each iteration, we calculate the remaining MSE of the corresponding IBR filter to quantify the remaining uncertainty. Figure 2.5 provides the change of the average MSE with the number of experiment iterations for these three experimental design strategies. As expected, our MOCU-based strategy consistently identifies the most critical uncertain parameter, whose measurement leads to the maximum reduction of the MSE with our filtering objective in design. As a result, after two experiments, when two

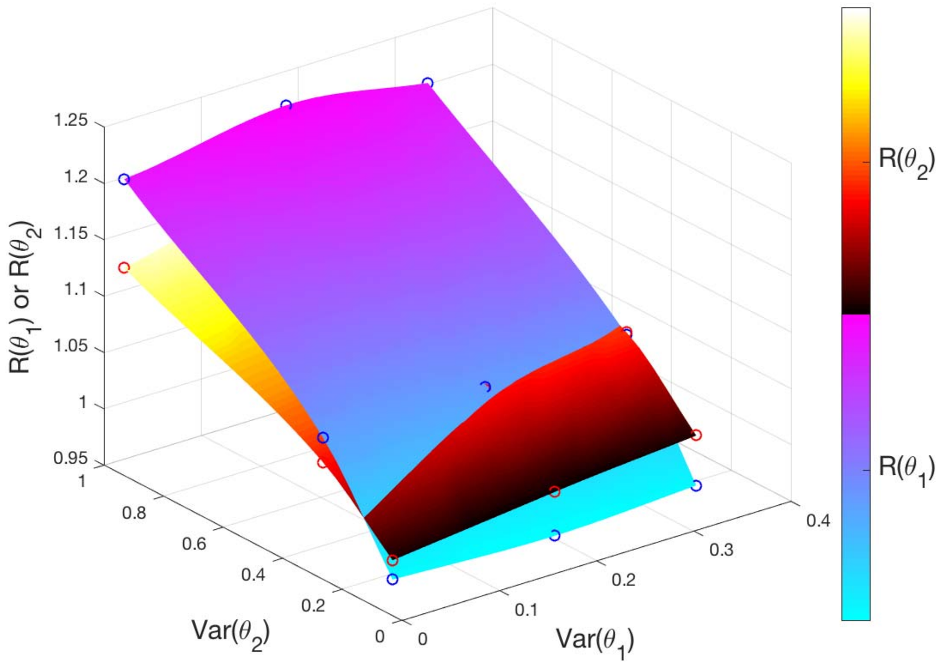


Figure 2.4: Comparison of Residual IBR cost of θ_1 and θ_2. The parameter with less Residual IBR cost is suggested to be specified by next experiment. Red circles and blue circles are precise calculations of $R(\theta_1)$ and $R(\theta_2)$, respectively, and the surfaces are obtained by cubic spline interpolation. Reprinted with permission from [86], copyright © 2020 IEEE.

parameters have been determined, the performance of our MOCU-based strategy has almost reached the level obtained when there is no uncertainty remaining (after four experiments), whereas for both entropy and random design there remains significant uncertainty after two experiments, meaning that they have not identified the best two parameters to estimate.

2.8 Pharmacokinetics Model

In this section, we illustrate the IBR filter and experimental design for a pharmacokinetic two-compartment model based on a SDE system [75]. Differentiating the body into a central compartment (plasma) and a peripheral compartment (tissues), the two-compartment model describes the relationship between the drug concentration in the central compartment $Y_1(t)$, the drug concentration in the peripheral compartment $Y_2(t)$, and the measurement x_t of the drug concentration in the central compartment. The transit of the drug throughout the body is described by the SDE

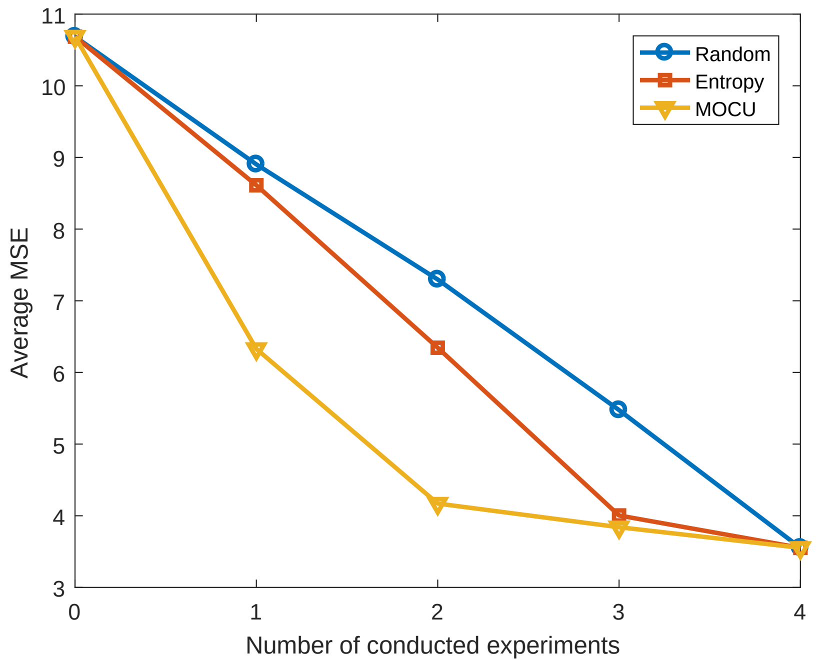


Figure 2.5: The average performance of sequential experimental design with different strategies. In each setup, the MSE is obtained after conducting each experiment in a sequence of experiments for the SDE signal and observation model with four unknown parameters. Reprinted with permission from [86], copyright © 2020 IEEE.

shown below:

$$\begin{aligned}
\mathrm{d}Y_1(t) &= (k_{21}Y_2(t) - k_{12}Y_1(t) - k_{10}Y_1(t))\,\mathrm{d}t + \sigma_1 \mathrm{d}\,W_1(t),\\
\mathrm{d}Y_2(t) &= (k_{12}Y_1(t) - k_{21}Y_2(t))\,\mathrm{d}t + \sigma_2\,\mathrm{d}W_2(t),\\
X(t) &= Y_1(t) + \epsilon, \qquad \epsilon \sim N(0, \sigma_\epsilon^2),
\end{aligned} \tag{2.52}$$

where $W_1(t)$, $W_2(t)$ are independent Wiener processes, and k_{10}, k_{12}, and k_{21} are individual rate constants (parameters) possessing the joint prior distribution

$$\theta = (k_{10},\, k_{12},\, k_{21})^{\mathrm{T}} \sim N(\mu, \Omega).$$

Following the case example in [83], we set the statistics of the prior as: $\mu = (0.2, 0.5, 0.25)^{\mathrm{T}}$, and Ω a diagonal matrix with $diag(\Omega) = (0.01^2, 0.1^2, 0.02^2)^{\mathrm{T}}$. Other parameters are set to $\sigma_\epsilon^2 = 0.04$, $\sigma_1 = 0.1$, $\sigma_2 = 0.1$. The initial condition of $Y_1(t)$ and $Y_2(t)$ are set to be 10 and 0, respectively, which corresponds to the case of Intravenous injection: the pharmacy is initially injected to the plasma and then diffuse to the tissue. After finding the IBR filter using the preceding theory, we consider its performance, and then turn to the problem of specifying in what order to determine the individual rate constants to optimally reduce the MSE of estimating $\boldsymbol{Y}(t) = (Y_1(t), Y_2(t))^{\mathrm{T}}$. We consider the discrete case with sampling points from 0 to 10 by an increment of 0.01.

The SDE in (2.52) also follows the form of (2.20), with the matrices below:

$$\mathbf{A}(t) = \begin{pmatrix} -k_{12} - k_{10} & k_{21} \\ k_{12} & -k_{21} \end{pmatrix}, \qquad \mathbf{a}(t) = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$
$$\mathbf{B}(t) = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}, \qquad \boldsymbol{Y}(0) = \begin{pmatrix} 10 \\ 0 \end{pmatrix}.$$

Therefore, similar to the procedure in the synthetic example, we can calculate the IBR filter $\widehat{\boldsymbol{G}}_\Theta$ through numerical integrals.

We compare the performance of the IBR filter with two other filters: the Wiener filters given specific values of the parameters $\theta = \mu + 3\sigma_\theta$ with $\sigma_\theta = (0.01, 0.1, 0.02)^{\mathrm{T}}$ the vector of standard deviations of parameters and the tau-robust filter. The comparison of the filtering performance on signals generated with $\theta = \mu$ and $\theta = \mu + 3\sigma_\theta$ are shown in Fig. 2.6 and Fig. 2.7, respectively. Since there is no direct observation in the peripheral compartment, the main source of the estimation error is in the peripheral compartment. Observation from the peripheral compartment shows that the IBR filter performs fairly well in both cases, while the Wiener filter with $\theta = \mu + 3\sigma_\theta$ performs well only in the case with matched parameters as expected. We notice that the τ-robust filter does not show its robustness on signals generated with $\theta = \mu + 3\sigma_\theta$, probably because in the setting of this pharmacokinetics model, the uncertain parameters follow unbounded Gaussian distributions, while the τ-robust filter is proposed under the bounded τ-divergence assumption.

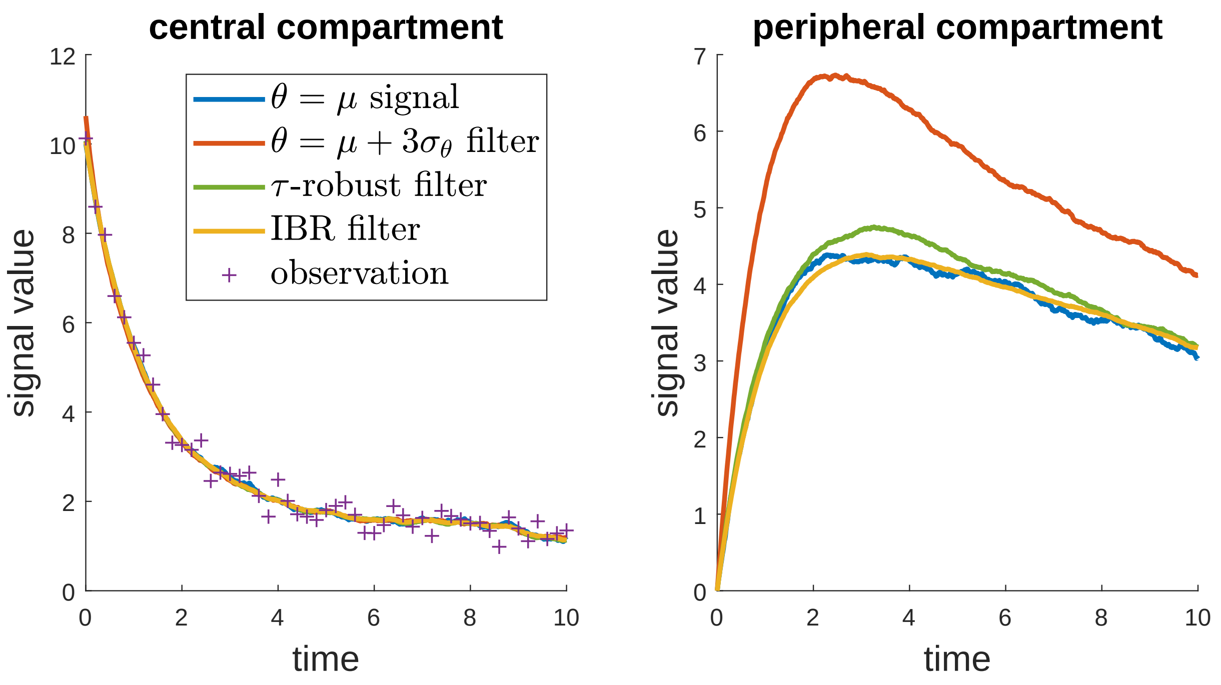


Figure 2.6: One example based on the SDE with $\theta = \mu$. Left and right sub-figures show the drug concentration levels of central and peripheral compartments, respectively. Blue curves correspond to the actual signals; purple crosses indicate the measurements of concentration in the central compartment; red curves depict the estimation with the optimal filter for $\theta = \mu + 3\sigma_\theta$; green curves are the estimated signals with the τ-robust filter; and yellow curves are filtered signals using the IBR filter. Only one out of every 20 measurements is visualized here to avoid curve cluttering. Reprinted with permission from [86], copyright © 2020 IEEE.

Then we perform sequential experimental design by calculating the design values of parameters k_{10}, k_{12} and k_{21}. Here we suppose our experimental budget can afford to perform two experiments. For the first experiment, the IBR residual costs are $R^1(k_{10}) = 51.3$, $R^1(k_{12}) = 45.2$ and $R^1(k_{21}) = 36.4$. So the first experiment should determine k_{21}. Following the first experiment, the true value of k_{21} is put into the model and the design values are calculated based on the updated model. We randomly sample 10 values of k_{21} as the result of the first experiment, and then calculate the IBR residual costs $R^2(k_{10}|k_{21})$ and $R^2(k_{12}|k_{21})$. All 10 random cases show that k_{12} should be determined in the second experiment, and the average design values are $\mathbb{E}_{k_{21}}[D^2(k_{10}|k_{21})] = 33.4$ and $\mathbb{E}_{k_{21}}[D^2(k_{12}|k_{21})] = 14.2$. Although the choice of the second experiment in our example is the same for all sampled values of the primary parameter, in general, the choice of the second

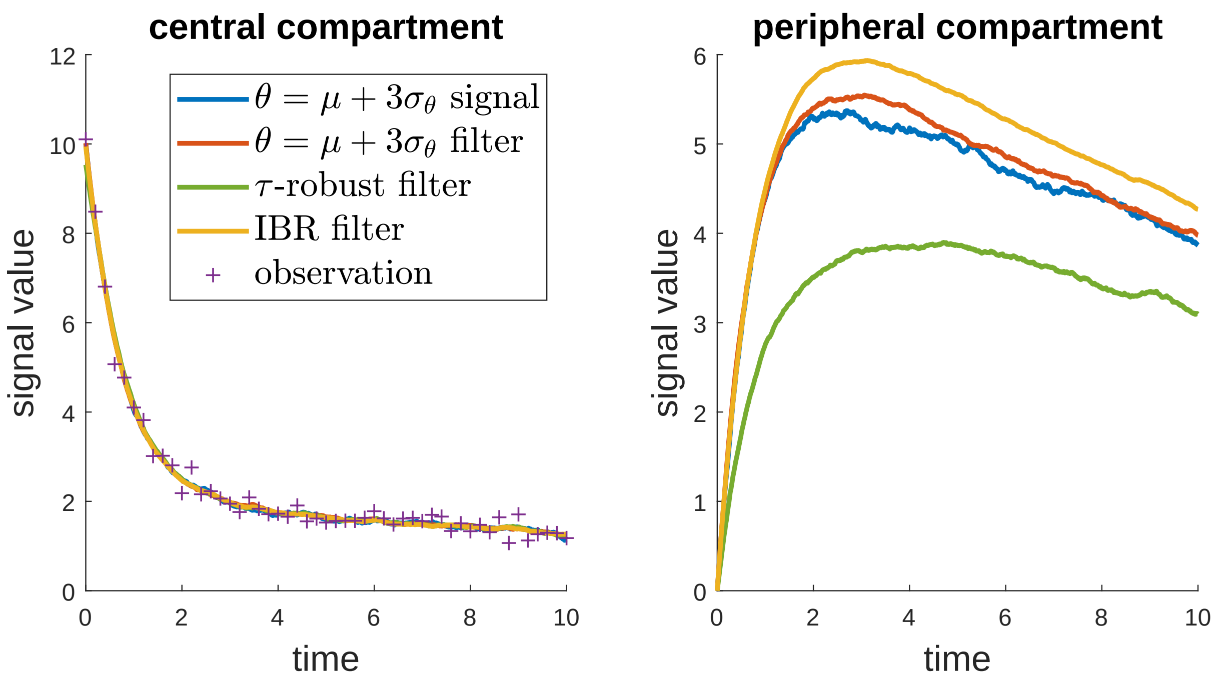


Figure 2.7: One example based on the SDE with $\theta = \mu + 3\sigma_\theta$. Left and right sub-figures show the drug concentration levels of central and peripheral compartments, respectively. Blue curves correspond to the actual signals; purple crosses indicate the measurements of concentration in the central compartment; red curves depict the estimation with the optimal filter for $\theta = \mu + 3\sigma_\theta$; green curves are the estimated signals with the τ-robust filter; and yellow curves are filtered signals using the IBR filter. Only one out of every 20 measurements is visualized here to avoid curve cluttering. Reprinted with permission from [86], copyright © 2020 IEEE.

experiment depends upon the value of the primary parameter, so that the choice of the second experiment can vary depending on the determined value of the primary parameter. In this example, the estimation error of $Y_2(t)$ dominates the full estimation cost, since $Y_2(t)$ is not observed directly, and the estimation of $Y_2(t)$ is based on its correlation with $Y_1(t)$. Therefore, k_{10} is less important than the other parameters, since it is conditionally independent of $Y_2(t)$ given $Y_1(t)$. Our calculation confirms this observation, preferring the estimation of k_{12} over that of k_{10}.

3. UNCERTAINTY-AWARE ACTIVE LEARNING FOR OPTIMAL BAYESIAN CLASSIFIER

3.1 Overview

To achieve label efficiency for training supervised learning models, pool-based active learning sequentially selects samples from a set of candidates as queries to label by optimizing an acquisition function. One category of existing methods adopts one-step-look-ahead strategies based on acquisition functions tailored with the learning objectives, for example based on the expected loss/error reduction (ELR/EER) or the mean objective cost of uncertainty (MOCU) proposed recently. These active learning methods are optimal with the maximum classification error reduction when one considers a single query. However, it is well-known that there is no performance guarantee in the long run for these myopic methods. In this chapter, we show that these methods are not guaranteed to converge to the optimal classifier of the true model because MOCU is not strictly concave. To address this problem, we propose two Bayesian active learning methods guarantee convergence to the optimal classifier of the true model. In the first method, we propose an acquisition function based on a Weighted form of MOCU. Similar to MOCU-based method, the proposed method focuses on the reduction of the uncertainty that pertains to the classification error, and the tailored weight can guarantee the convergence of the proposed method on binary classification problems. For training Bayesian classifiers with both synthetic and real-world data, our experiments demonstrate the superior performance of active learning by Weighted MOCU and Soft MOCU compared to other existing methods.

3.2 Introduction

In this chapter, we focus on learning optimal Bayesian classifiers with limited training data. To achieve sample and label efficiency, we study pool-based Bayesian active learning. It starts with a prior of an uncertain model and collects training data in a sequential manner by optimizing an acquisition function measuring the benefit to our learning objective from querying labels for corresponding candidates. By reducing model uncertainty through the active learning procedure, we aim to approach the optimal classifier of the unknown true model, which has the minimum prediction error.

Several notable Bayesian active learning methods have been proposed using different acquisition functions. Maximum Entropy Search (MES) or Uncertainty Sampling selects the candidate with the maximum predictive probability entropy [71, 59]. However, observing the most uncertain candidate may not provide the most useful model information if the observation itself is noisy. Another Shannon entropy based method, Bayesian Active Learning by Disagreement (BALD) [37, 44], selects the candidate to minimize the entropy of the uncertain model parameters. The Equivalence Class Edge Cutting algorithm (EC^2) targeting at active learning with finite possible models, chooses the candidate that maximally reduces the version space probability mass [32]. Based on the policy Gibbs error, a generalization of the Shannon entropy, Cuong et al. [11] proposed the maximum Gibbs error criterion (maxGEC) to query the candidate that has the maximum Gibbs error so that the remaining posterior entropy is minimized. While various different acquisition functions are used, most of the existing active learning methods focus on reducing the model uncertainty instead of directly reducing the classification error, despite this being the ultimate learning objective. To rectify this shortcoming, in this chapter, we focus on active learning that directly focuses on reducing the model uncertainty that impacts the classification accuracy of the resulting classifier. While the reduction of model uncertainty often results in the decrease in classification error, it is important to note that not all model uncertainty affects classification error. Rather than reducing uncertainty in general, an active learning scheme that aims at reducing the model uncertainty that critically impacts the objective (*i.e.*, classification accuracy) can significantly improve its label efficiency.

There is one category of methods based on Expected Loss/Error Reduction (ELR/EER) that aims to maximize the reduction in classification error directly in a one-step-look-ahead manner [65, 92, 40]. They directly target at reducing the classification error and can achieve the expected optimal performance that is achievable with one single query [65]. However, these methods do not have any theoretical convergence guarantee, and empirically, they suffer from myopic behavior with degraded efficiency in the long run. Yoon *et al.* [84] proposed a metric, Mean Objective Cost of

Uncertainty (MOCU), which enables model uncertainty quantification by estimating the expected classification performance loss compared with the optimal classifier due to the uncertainty. MOCU is equivalent to ELR when applied to active learning and provides a tool to analyze the convergence of active learning methods to the true optimal classifier.

In this chapter, we first analyze why ELR- or MOCU-based active learning methods may get stuck before collecting enough data to identify the true optimal classifier—despite their efficacy in identifying optimal one-step queries. We further propose a novel Weighted-MOCU active learning method that can focus only on the uncertainty related to the loss for efficient active learning and is guaranteed to converge to the optimal classifier of the true model on binary classification problem. We also propose a novel acquisition function based on a strictly concave approximation of MOCU, referred to as Soft MOCU, to address this problem. Thanks to the strict concavity of Soft MOCU, the resulting acquisition function can capture the continuous change in model uncertainty. As a result, one-step-look-ahead active learning guided by this acquisition function alleviates the limitations due to its myopic nature and is guaranteed to converge to the optimal classifier. We provide theoretical proof of the convergence of the Weighted-MOCU- and Soft-MOCU-based method. Last but not least, we demonstrate the expected sample efficiency of Weighted-MOCU- and Soft-MOCU-based active learning with both synthetic and real-world datasets.

3.3 Background

We first review the basic concepts in Bayesian active learning for classification, focusing on the acquisition function targeting directly at the learning objective.

3.3.1 Mean Objective Cost of Uncertainty

Mean Objective Cost of Uncertainty (MOCU) is a metric measuring the direct influence on the performance with respect to the learning objective due to model uncertainty [84, 85]. We here provide a review in the context of learning Bayesian classifiers.

Consider the classification problem in the input feature space $\boldsymbol{x} \in \mathcal{X}$ and output label space $y \in \mathcal{Y} = \{0, 1, \ldots, M-1\}$ with a probabilistic model characterized by θ as $p(y|\boldsymbol{x}, \theta)$. The aim is to find a classifier $\psi : \mathcal{X} \to \mathcal{Y}$ to estimate the label given a testing feature vector $\boldsymbol{x}^* \in \mathcal{X}$ as $\psi(\boldsymbol{x}^*)$. In this chapter we focus on the 0-1 loss to measure the performance of a classifier, which directly reflects the classification error. Denote the expected classification error of ψ on $\boldsymbol{x}$ as $C_\theta(\psi, \boldsymbol{x}) = \mathbb{E}_{p(y|\boldsymbol{x},\theta)}[\mathbb{1}(\psi(\boldsymbol{x}) \neq y)] = 1 - p(y = \psi(\boldsymbol{x})|\boldsymbol{x}, \theta)$. The optimal classifier of θ, denoted as ψ_θ, is defined as the classifier that minimizes the error $\psi_\theta := \arg\min_\psi C_\theta(\psi, \boldsymbol{x}) = \arg\max_y p(y|\boldsymbol{x}, \theta)$.

In the practical situations with model uncertainty where the true model parameter θ_r is unknown, we often assume that based on prior knowledge or observed data, we can derive a distribution $\pi(\theta)$ over the uncertain model parameter set $\theta \in \Theta$. As we do not know the true model, the learning objective is to train an *Optimal Bayesian Classifier* (OBC) $\psi_{\pi(\theta)}$ that minimizes the expected classification error over $\pi(\theta)$ [13]:

$$\psi_{\pi(\theta)} = \arg\min_{\psi} \mathbb{E}_{\pi(\theta)}[C_\theta(\psi, \boldsymbol{x})] = \arg\max_{y} p(y|\boldsymbol{x}), \tag{3.1}$$

where $p(y|\boldsymbol{x}) = \mathbb{E}_{\pi(\theta)}[p(y|\boldsymbol{x}, \theta)]$ is the *predictive distribution*. OBC is the optimal classifier based on the current knowledge. If we can observe enough data to update our model knowledge $\pi(\theta)$ with reduced model uncertainty, OBC will converge to the true optimal classifier based on the true model θ_r .

In Bayesian classification, MOCU can be defined as the expected difference between the expected error of OBC and the optimal classifier due to model uncertainty:

$$\mathcal{M}(\pi(\theta)) = \mathbb{E}_{\boldsymbol{x}}[\mathbb{E}_{\pi(\theta)}[C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}) - C_\theta(\psi_\theta, \boldsymbol{x})]], \tag{3.2}$$

where $\mathbb{E}_{\boldsymbol{x}}$ stands for averaging over the feature space $\mathcal{X}$. The first term is the OBC error. Since ψ_θ is the optimal classifier with a specific θ, for the terms inside the expectation, $C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}) - C_\theta(\psi_\theta, \boldsymbol{x}) \geq 0$. So the second term is a lower bound of the OBC error. Denote supp(π) as the support of $\pi(\theta)$. If $\mathcal{M}(\pi(\theta)) = 0$, then $\forall \boldsymbol{x} \in \mathcal{X}$, $\forall \theta \in$ supp(π), $\psi_{\pi(\theta)} = \psi_\theta$, i.e. $\arg\max_y p(y|\boldsymbol{x}) = \arg\max_y p(y|\boldsymbol{x}, \theta)$, indicating that OBC is the true optimal classifier. Note that MOCU does not capture all the model uncertainty as we only require $\arg\max_y p(y|\boldsymbol{x}, \theta) = \arg\max_y p(y|\boldsymbol{x})$ instead of $p(y|\boldsymbol{x}, \theta) = p(y|\boldsymbol{x})$ to make MOCU$= 0$. But with MOCU$= 0$, we have found the true optimal classifier and there is no need to further reduce the model uncertainty considering our learning objective.

3.3.2 Pool-based Bayesian Active Learning

Bayesian active learning sequentially searches for candidates in $\mathcal{X}$ as queries to acquire their labels by optimizing an acquisition function. Then by including the new observed data into the training dataset D, the learning algorithm updates the posterior distribution $\pi(\theta|D)$, with which the acquisition function will be computed to guide active learning in each iteration. In the following discussion, to simplify the notations, we use $\pi(\theta)$ and $p(y|\boldsymbol{x})$ for the posterior and predictive distribution conditioned on D by omitting D in the notations. When a new observation pair $(\boldsymbol{x}^*, y^*)$ is collected, the posterior and the predictive distribution are updated by $\pi(\theta|\boldsymbol{x}^*, y^*) = \frac{\pi(\theta)p(y^*|\boldsymbol{x}^*,\theta)}{p(y^*|\boldsymbol{x}^*)}$ and $p(y|\boldsymbol{x}; \boldsymbol{x}^*, y^*) = \mathbb{E}_{\pi(\theta|\boldsymbol{x}^*,y^*)}[p(y|\boldsymbol{x}, \theta)]$.

We can define the acquisition function based on MOCU in a one-step-look-ahead manner:

$$U^{\mathrm{M}}(\boldsymbol{x}; \pi(\theta)) = \mathcal{M}(\pi(\theta)) - \mathbb{E}_{y|\boldsymbol{x}}\mathcal{M}(\pi(\theta|\boldsymbol{x}, y)), \tag{3.3}$$

which is the expected reduction of MOCU if observing the new pair $(\boldsymbol{x}, y)$. As y is not known at the current iteration to acquire the label, it is averaged over all possible values of y.

We can show that $C_\theta(\psi_\theta, \boldsymbol{x}')$ in the MOCU definition (3.2) can be cancelled in two MOCUs in (3.3). Since $\pi(\theta) = \mathbb{E}_{p(y|\boldsymbol{x})}[\pi(\theta|\boldsymbol{x}, y)]$ ($\boldsymbol{x}$ is often assumed to be independent of θ so $\pi(\theta|\boldsymbol{x}) = \pi(\theta)$), we can rewrite the first term in (3.3) as:

$$\mathcal{M}(\pi(\theta)) = \mathbb{E}_{\boldsymbol{x}'}\{\mathbb{E}_{y|\boldsymbol{x}}[\mathbb{E}_{\pi(\theta|\boldsymbol{x},y)}[C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}')]]\} \tag{3.4}$$

while the second term in (3.3) can be expanded as:

$$\mathbb{E}_{y|\boldsymbol{x}}[\mathcal{M}(\pi(\theta|\boldsymbol{x}, y))] = \mathbb{E}_{\boldsymbol{x}'}\{\mathbb{E}_{y|\boldsymbol{x}}[\mathbb{E}_{\pi(\theta|\boldsymbol{x},y)}[C_\theta(\psi_{\pi(\theta|\boldsymbol{x},y)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}')]]\}. \tag{3.5}$$

So the term $C_\theta(\psi_\theta, \boldsymbol{x}')$ can be cancelled out. The acquisition function is just the OBC prediction error reduction after observing the new pair $(\boldsymbol{x}, y)$:

$$U^{\mathrm{M}}(\boldsymbol{x}; \pi(\theta)) = \mathbb{E}_{\boldsymbol{x}'}\{\mathbb{E}_{\pi(\theta)}[C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}')]\} - \mathbb{E}_{\boldsymbol{x}'}\{\mathbb{E}_{y|\boldsymbol{x}}[\mathbb{E}_{\pi(\theta|\boldsymbol{x},y)}[C_\theta(\psi_{\pi(\theta|\boldsymbol{x},y)}, \boldsymbol{x}')]]\}, \quad (3.6)$$

which is the same acquisition function as Error Loss Reduction (ELR) [65].

In this chapter, we focus on MOCU-based active learning with the OBC as the classifier. As shown in (3.6), MOCU-based active learning queries the candidate to achieve the maximum expected reduction in OBC classification error in each iteration. Hence, the MOCU-based method is the optimal strategy for active learning of the OBC with a single query.

3.4 Methods

In this section, we first show that MOCU-based active learning based on a one-step-look-ahead strategy may get stuck before MOCU converges to 0 with the corresponding OBC converging to the true optimal classifier. We then propose a new acquisition function that has the guarantee that the OBC converges to the true optimal classifier.

3.4.1 Analysis of MOCU-based Active Learning

We first analyze the MOCU reduction to show that MOCU-based active learning ignores the uncertainty irrelevant to the OBC prediction. By that, we indicate that not all the model uncertainties directly affect the OBC prediction. Denote the contribution to the MOCU at point $\boldsymbol{x}$ as $K(\boldsymbol{x}, \pi(\theta)) = \mathbb{E}_{\pi(\theta)}[C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}) - C_\theta(\psi_\theta, \boldsymbol{x})]$, so that $\mathcal{M}(\pi(\theta)) = \mathbb{E}_{p(\boldsymbol{x})}[K(\boldsymbol{x}, \pi(\theta))]$. If $K(\boldsymbol{x}, \pi(\theta)) = 0$, then we have $\forall\theta \in$ supp(π), $\psi_\theta(\boldsymbol{x}) = \psi_{\pi(\theta)}(\boldsymbol{x})$, i.e. $\arg\max_y p(y|\boldsymbol{x}, \theta) = \arg\max_y p(y|\boldsymbol{x})$. This means that for all the possible models, the optimal predictions are the same, and the OBC prediction on $\boldsymbol{x}$ will not be affected by the remaining uncertainty of $p(y|\boldsymbol{x}, \theta)$, if any. In fact, $K(\boldsymbol{x}, \pi(\theta)) = 0$ does not necessarily mean that there is no uncertainty associated with $p(y|\boldsymbol{x}, \theta)$, for which it requires that the value of $p(y|\boldsymbol{x}, \theta)$ is the same $\forall\theta \in$ supp(π), apparently a stronger statement than $K(\boldsymbol{x}, \pi(\theta))$ being 0. Therefore, not all the uncertainties of $p(y|\boldsymbol{x}, \theta)$ are captured in MOCU when $K(\boldsymbol{x}, \pi(\theta)) = 0$. We consider the uncertainty in $p(y|\boldsymbol{x}, \theta)$ to be "objective-irrelevant" to the OBC prediction if $K(\boldsymbol{x}, \pi(\theta)) = 0$. In the active learning procedure, when a new observation is obtained, it reduces the uncertainty of the parameter θ; and as a result, it reduces the uncertainty of $p(y|\boldsymbol{x}, \theta)$ for each $\boldsymbol{x} \in \mathcal{X}$. If an observation only reduces objective-irrelevant uncertainty, the value of MOCU will not change. That explains why in the first several active learning iterations, the MOCU-based active learning can be more efficient than the methods guided by total uncertainty reduction, such as BALD.

We now analyze why MOCU-based active learning may get stuck before the OBC converges to the true optimal classifier. In other words, when the acquisition function for all the candidates in the pool is 0, i.e. $\forall \boldsymbol{x} \in \mathcal{X}, U^{\mathrm{M}}(\boldsymbol{x}; \pi(\theta)) = 0$, the active learning will degenerate to random sampling and keep selecting the candidate based on the adopted tie-breaking strategy. When that happens, we say that active learning gets stuck without converging to the true optimal classifier if MOCU is still larger than 0.

We first show that the MOCU (3.2) is a concave function of $\pi(\theta)$, but it is not strictly concave everywhere with nonzero curvature to guide active learning. From the definition of ψ_θ and $\psi_{\pi(\theta)}$, we have $C_\theta(\psi_\theta, \boldsymbol{x}) = 1 - \max_y p(y|\boldsymbol{x}, \theta)$ and $\mathbb{E}_{\pi(\theta)}[C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x})] = \mathbb{E}_{\pi(\theta)}[1 - p(\psi_{\pi(\theta)}(\boldsymbol{x})|\boldsymbol{x}, \theta)] = 1 - \max_y \mathbb{E}_{\pi(\theta)}[p(y|\boldsymbol{x}, \theta)]$. Substituting them in (3.2),

$$\mathcal{M}(\pi(\theta)) = \mathbb{E}_{\boldsymbol{x}}\{\mathbb{E}_{\pi(\theta)}[\max_y p(y|\boldsymbol{x}, \theta)] - \max_y \mathbb{E}_{\pi(\theta)}[p(y|\boldsymbol{x}, \theta)]\}, \tag{3.7}$$

$$= \mathbb{E}_{\boldsymbol{x}}\{\mathbb{E}_{\pi(\theta)}[\max_y p(y|\boldsymbol{x}, \theta)] - p(\psi_{\pi(\theta)}(\boldsymbol{x})|\boldsymbol{x}, \theta)]\} \tag{3.8}$$

In (3.7), the first term $\mathbb{E}_{\pi(\theta)}[\max_y p(y|\boldsymbol{x}, \theta)]$ is a linear function of $\pi(\theta)$. While the second term $\max_y \mathbb{E}_{\pi(\theta)}[p(y|\boldsymbol{x}, \theta)]$ is the maximum over M linear functions and thus is a convex piecewise linear function. As a result, (3.7) equals to a linear function subtracting a convex function and therefore it is concave and also piecewise linear. It is thus not strictly concave everywhere. Within each piece of the linear functions in (3.8), the OBC classifier $\psi_{\pi(\theta)}(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}$ takes the same label for different $\pi(\theta)$. To gain the intuition, we study a binary classification problem with the uncertainty class of two posssible models $\Theta = \{\theta_1, \theta_2\}$ and a candidate pool of two training samples to query: $\mathcal{X} = \{x_1, x_2\}$. Further details of the model setup can be found in Appendix A.2. Since $\pi(\theta_1) = 1 - \pi(\theta_2)$, we can express the MOCU function as a univariate function of $\pi(\theta_1)$ as shown in Fig. 3.1. It is clear that the MOCU function is a concave piece-wise linear function.

In (3.3), we observe that $\pi(\theta) = \mathbb{E}_{p(y|\boldsymbol{x})}[\pi(\theta|\boldsymbol{x}, y)]$. Since MOCU is a concave function, based on Jensen's Inequality, we have $U^{\mathrm{M}}(\boldsymbol{x}; \pi(\theta)) \geq 0$. While MOCU is not a strictly concave function as explained, we can find two conditions that make the equality to hold: first, $\forall y \in \mathcal{Y}$, $\pi(\theta|\boldsymbol{x}, y) = \pi(\theta)$, which means observing $\boldsymbol{x}$ does not help change the knowledge about θ; second, $\pi(\theta|\boldsymbol{x}, y)$ changes but the change of $\pi(\theta|\boldsymbol{x}, y)$, $\forall y \in \mathcal{Y}$, is within the same linear piece of MOCU. In the second condition, observing $\boldsymbol{x}$ one time only provides little information of θ, and that information will not change the OBC classifier. If MOCU is larger than 0 but all the $\boldsymbol{x}$ cannot provide enough information to update the OBC classifier, the acquisition function can then be 0 for all the candidates, with which MOCU-based active learning will get stuck before converging to the true optimal classifier. Appendix B.2 provides such a synthetic example. From the discussion above, we can see that MOCU-based active learning may get stuck because MOCU is not strictly concave.

In the next section, we impose concavity to the approximations of MOCU and propose one-step-look-ahead acquisition functions based on them. The approximation makes the corresponding active learning to have similar short-term optimality for single iterations as in MOCU-based active learning. More importantly, the imposed strictly concavity leads to the theoretical guarantee that the OBC will converge to the true optimal classifier without getting stuck.

3.4.2 Weighted MOCU-based active learning

In this section, we propose a modified MOCU-based acquisition function that has the theoretical guarantee to converge to the optimal classifier. Specifically, we propose a modified MOCU function that multiplies a weight with each loss difference between the OBC $\psi_{\pi(\theta)}$ and the optimal classifier ψ_θ in the original MOCU definition:

$$\mathcal{M}^w(\pi(\theta)) = \mathbb{E}_{p(\boldsymbol{x}')}\{\mathbb{E}_{\pi(\theta)}\{w(\pi(\theta), \boldsymbol{x}', \theta)[C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}')]\}\}, \tag{3.9}$$

where $w(\pi(\theta), \boldsymbol{x}', \theta) > 0$ is the weighting function. The corresponding acquisition function is:

$$U^w(\boldsymbol{x}; \pi(\theta)) = \mathcal{M}^w(\pi(\theta)) - \mathbb{E}_{y|\boldsymbol{x}}[\mathcal{M}^w(\pi(\theta|\boldsymbol{x}, y))]. \tag{3.10}$$

In (3.9), as more data are collected and the model parameter distribution $\pi(\theta)$ changes, $w(\pi(\theta), \boldsymbol{x}', \theta)$ will change accordingly. The change of $w(\pi(\theta), \boldsymbol{x}', \theta)$ cannot affect the value of the Weighted MOCU if $C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}') = 0$, $\forall \theta \in \text{supp}(\pi(\theta))$, indicating the uncertainty at $\boldsymbol{x}'$ is objective-irrelevant. This makes sure that the acquisition function based on the Weighted MOCU will inherit the property of MOCU-based active learning to directly target at classification error reduction while ignoring irrelevant uncertainty. On the other hand, by introducing the predictive probability into the weighting functions, the probability change from one-step samples can be captured by the Weighted-MOCU based acquisition function such that it can have theoretical convergence guaranteed to the optimal classifier as shown below.

We would like to emphasize that there are also active learning algorithms, such as the ones based on the cyclic sampling and ϵ-greedy policies [36], that can almost surely converge to the true model, and as a result, the OBC converges to the true optimal classifier. However, these policies focus on the *total* uncertainty reduction to derive the *full* knowledge of the true model. However, this is unnecessary and therefore inefficient, since we only need the OBC to converge to the true optimal classifier if the classification performance is the primary concern. Unlike such policies, our Weighted-MOCU based policy directly reduces the *objective* uncertainty affecting classification, and as a result, it is much more efficient by focusing only on those queries that are helpful for improving the prediction. As a result, our proposed algorithm guarantees efficiency both in the short term as well as in the longer term.

In the following, we design a weighting function to make $\mathcal{M}^w(\pi(\theta)) = 0$ if and only if $\forall \boldsymbol{x} \in \mathcal{X}, U^w(\boldsymbol{x}; \pi(\theta)) = 0$ and show that active learning based on this Weighted MOCU converges to the optimal classifier. Specifically, we propose the following weighting function:

$$w(\pi(\theta), \boldsymbol{x}', \theta) = 1 - c \cdot K(\boldsymbol{x}', \pi(\theta)), \text{ with} \tag{3.11}$$

$$K(\boldsymbol{x}', \pi(\theta)) = \mathbb{E}_{\pi(\theta)}[C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}')] \tag{3.12}$$

$$= \min_{y'} \mathbb{E}_{\pi(\theta)}[1 - p(y'|\boldsymbol{x}', \theta)] - \mathbb{E}_{\pi(\theta)}[\min_{y'}(1 - p(y'|\boldsymbol{x}', \theta)] \tag{3.13}$$

$$= \mathbb{E}_{\pi(\theta)}[\max_{y'} p(y'|\boldsymbol{x}', \theta)] - \max_{y'} p(y'|\boldsymbol{x}'), \tag{3.14}$$

where $0 < c \leq 1$ is a parameter controlling the approximation of the Weighted MOCU to the original MOCU, with smaller c giving a better approximation. The choice of c depends on the specific classification problem and the total query budget. Methods using a smaller c approximate the ELR methods better, hence they will perform well in the first several iterations but may converge slowly in the long run. On the other hand, when c is closer to 1, the acquisition function weighs more heavily on long-term benefits. It is clear that $K(\boldsymbol{x}', \pi(\theta)) \geq 0$ by (3.13). For binary classification, $\max_{y'} p(y'|\boldsymbol{x}') \geq 0.5$. As $\mathbb{E}_{\pi(\theta)}[\max_{y'} p(y'|\boldsymbol{x}', \theta)] \leq 1$, from (3.14), we have $K(\boldsymbol{x}', \pi(\theta)) \leq 0.5$, demonstrating that the weighting function in (3.11) satisfies the requirement $w(\pi(\theta), \boldsymbol{x}', \theta) \geq 0.5 > 0$.

Note that this simple weighting function does not change with respect to the model parameter values. Substituting it into the Weighted MOCU expression, we have:

$$\mathcal{M}^w(\pi(\theta)) = \mathbb{E}_{p(\boldsymbol{x}')}\{(1 - cK(\boldsymbol{x}', \pi(\theta))) \cdot K(\boldsymbol{x}', \pi(\theta))\}, \tag{3.15}$$

which is a strictly concave function of K. We also illustrate the Weighted-MOCU function in Fig. 3.1 for the comparison with MOCU on the same example as in Section 3.4.1. As shown in the figure, a smaller c provides better approximation to the MOCU function, and all the Weighted MOCU functions are strictly concave functions of $\pi(\theta_1)$ instead of being piece-wise linear, which guarantees that the acquisition function $U^w(x_1; \tilde{\pi}(\theta))$ is positive. In general, Weighted MOCU is strictly concave along most of the directions and only changes linearly along the directions that $K(\boldsymbol{x}, \pi(\theta))$ is constant for $\boldsymbol{x} \in \mathcal{X}$, which correspond to the queries that only reduce irrelevant uncertainties. Such a property can guarantee the convergence to the true optimal classifier.

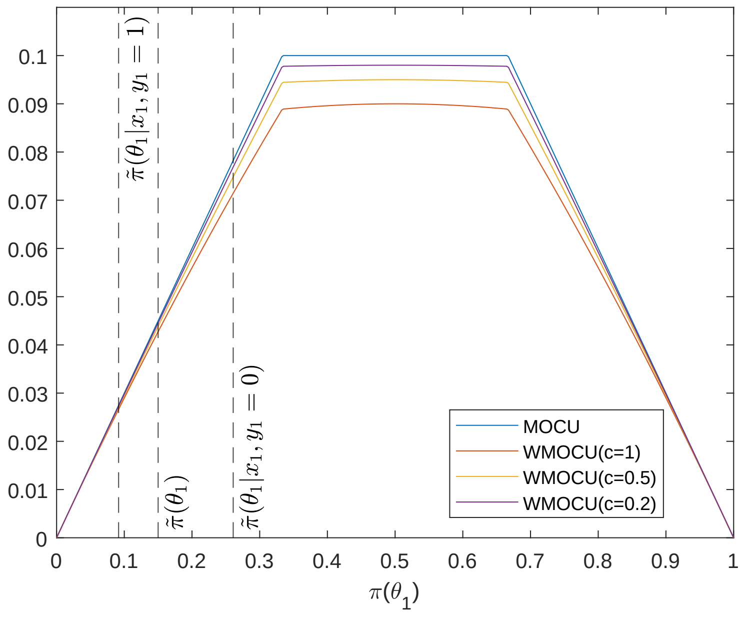


Figure 3.1: MOCU and Weighted-MOCU functions of a binary classification problem with $\Theta = \{\theta_1, \theta_2\}$.

Theoretical Convergence Guarantee

Now we show that if active learning for a binary classification problem is guided by the acquisition function defined by (3.10) and (3.11), MOCU will converge to 0 almost surely and hence the procedure will converge to learning the optimal classifier of the true model. We assume that both $\mathcal{X}$ and Θ are discrete with finite elements; the true model parameter $\theta_r \in \Theta$ and the prior distribution $\pi^0(\theta)$ over Θ satisfies $\pi^0(\theta_r) > 0$. We denote the posterior by $\pi^n(\theta)$ and predictive probability $p^n(y|\boldsymbol{x})$ in the n-th Weighted MOCU based active learning iteration, respectively.

Lemma 1. *Given $\pi(\theta)$, $\mathcal{M}(\pi(\theta)) = 0$ if and only if $\mathcal{M}^w(\pi(\theta)) = 0$.*

Lemma 1 indicates that if $\mathcal{M}^w(\pi(\theta)) = 0$, the OBC $\psi_{\pi(\theta)}$ converges to the optimal classifier ψ_{θ_r} as explained in Section 3.4.1.

Proof. Based on (3.2), since $C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}') \geq 0$, so $\mathcal{M}(\pi(\theta)) = 0$ iff $C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}') = 0 \, \forall \boldsymbol{x}' \in \mathcal{X}, \forall \theta \in \text{supp}(\pi)$. In addition, in (3.9), $w(\pi(\theta), \boldsymbol{x}', \theta) > 0$, so $C_\theta(\psi_{\pi(\theta)}, \boldsymbol{x}') - C_\theta(\psi_\theta, \boldsymbol{x}') = 0 \; \forall \boldsymbol{x}' \in \mathcal{X}, \; \forall \theta \in \text{supp}(\pi)$ iff $\mathcal{M}^w(\pi(\theta)) = 0$, which concludes the proof. $\square$

Lemma 2. *Define* $G(\boldsymbol{x}', \pi(\theta)) = (1 - cK(\boldsymbol{x}', \pi(\theta)))K(\boldsymbol{x}', \pi(\theta))$*,* $0 < c \leq 1$*.* $G(\boldsymbol{x}', \pi(\theta))$ *is a concave function of* $\pi(\theta)$*.*

It is important to choose a weighting scheme that renders a concave function G as it guarantees the acquisition function to be larger than or equal to 0, so that adding a new observation helps to reduce Weighted MOCU to effectively guide active learning.

Proof. In the following proof, we omit the argument $\boldsymbol{x}'$ in G and K for simplicity. Owing to the concavity of the min operator, $\min_{y'} \mathbb{E}_{\pi(\theta)}[1 - p(y'|\boldsymbol{x}', \theta)]$ is a concave function of $\pi(\theta)$. With $\mathbb{E}_{\pi(\theta)}[\min_{y'}(1 - p(y'|\boldsymbol{x}', \theta)]$ being a linear function of $\pi(\theta)$, based on (3.13), $K(\pi(\theta))$ equals to a concave function subtracting a linear function and thus is also a concave function.

As we have analyzed, $0 \leq K(\pi(\theta)) \leq 0.5$. We define $T(\kappa) = (1 - c\kappa)\kappa, \kappa \in [0, 0.5]$, a strictly increasing and strictly concave function. $G(\pi(\theta)) = T[K(\pi(\theta))]$ is a composite function of T and K. So we conclude the proof with the property of the concavity for the composite functions:

$$\begin{aligned} T[K(\lambda\pi_1(\theta) + (1-\lambda)\pi_2(\theta))] &\geq T[\lambda K(\pi_1(\theta)) + (1-\lambda)K(\pi_2(\theta))] \\ &\geq \lambda T[K(\pi_1(\theta))] + (1-\lambda)T[K(\pi_2(\theta))]. \end{aligned} \tag{3.16}$$

The first inequality is because T is increasing and K is concave; and the second inequality holds as T is a concave function. □

Lemma 3. $\forall \boldsymbol{x} \in \mathcal{X}, U^w(\boldsymbol{x}; \pi(\theta)) \geq 0.$

Proof. Since $\pi(\theta) = \sum_y p(y|\boldsymbol{x})\pi(\theta|\boldsymbol{x}, y)$, by Jensen's inequality, we have $G(\boldsymbol{x}', \pi(\theta)) \geq \mathbb{E}_{y|\boldsymbol{x}}[G(\boldsymbol{x}', \pi(\theta|\boldsymbol{x}, y))]$ as G is a concave function. So the Weighted MOCU acquisition function:

$$U^w(\boldsymbol{x}; \pi(\theta)) = \mathbb{E}_{\boldsymbol{x}'}[G(\boldsymbol{x}', \pi(\theta))] - \mathbb{E}_{\boldsymbol{x}'}[\mathbb{E}_{y|\boldsymbol{x}}[G(\boldsymbol{x}', \pi(\theta|\boldsymbol{x}, y))]] \geq 0. \tag{3.17}$$

□

Lemma 4. *At the* n*-th active learning iteration, if* $U^w(\boldsymbol{x}; \pi^n(\theta)) = 0$*,* $\forall \boldsymbol{x} \in \mathcal{X}$*,* $\mathcal{M}^w(\pi^n(\theta)) = 0$*.*

This lemma states that if the acquisition function values of all candidates with respect to $\pi(\theta)$ are 0, the Weighted MOCU is 0. By Lemma 1, so is MOCU. With these, we can conclude that the OBC with respect to $\pi(\theta)$ has converged to the optimal classifier. This is significant when comparing with original ELR and MOCU-based methods as we have shown that this is not the case for them, which may get stuck earlier and therefore lose the long-term efficiency.

Proof. We will prove the contrapositive of the lemma: assuming $\mathcal{M}^w(\pi^n(\theta)) > 0$, $\exists \boldsymbol{x} \in \mathcal{X}$ s.t. $U^w(\boldsymbol{x}; \pi^n(\theta)) > 0$.

Based on (3.15), $M^w(\pi^n(\theta)) > 0$ indicating $\exists \boldsymbol{x} \in \mathcal{X}\ s.t.\ K(\boldsymbol{x}, \pi^n(\theta)) > 0$. It is sufficient to show that if $K(\boldsymbol{x}, \pi^n(\theta)) > 0$, then $U^w(\boldsymbol{x}; \pi^n(\theta)) > 0$. To prove that, we only need to prove $G(\boldsymbol{x}, \pi^n(\theta)) > \mathbb{E}_{p^n(y|x)}[G(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y))]$; then by (3.17), $U^w(\boldsymbol{x}; \pi^n(\theta)) > 0$.

Since G is a concave function, we know $G(\boldsymbol{x}, \pi^n(\theta)) \geq \mathbb{E}_{p^n(y|x)}[G(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y))]$. With $\pi^n(\theta) = \sum_y p^n(y|\boldsymbol{x})\pi^n(\theta|\boldsymbol{x}, y)$, we can rewrite (3.16) as:

$$T[K(\boldsymbol{x}, \pi^n(\theta))] \geq T[\mathbb{E}_{p^n(y|x)}[K(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y))]] \geq \mathbb{E}_{p^n(y|x)}[T[K(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y))]].$$

The second equality holds only if $\forall y \in \{0, 1\}$, $K(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y)) = K(\boldsymbol{x}, \pi^n(\theta))$, which means that to prove $G(\boldsymbol{x}, \pi^n(\theta)) > \mathbb{E}_{y|\boldsymbol{x}}[G(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y))]$, we need to show $\exists y \in \{0, 1\}$, $K(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y)) \neq K(\boldsymbol{x}, \pi^n(\theta))$. In the following proof, we will show if $K(\boldsymbol{x}, \pi^n(\theta)) > 0$, then $\exists y \in \{0, 1\}$, s. t. $K(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y)) \neq K(\boldsymbol{x}, \pi^n(\theta))$.

Denote $\hat{y} = \arg\max_y p^n(y|\boldsymbol{x})$. By (3.14) we have:

$$K(\boldsymbol{x}, \pi^n(\theta)) = \sum_{\theta \in \text{supp}(\pi^n)} \pi^n(\theta)[\max_y p(y|\boldsymbol{x}, \theta) - p(\hat{y}|\boldsymbol{x}, \theta)]. \tag{3.18}$$

Since $K(\boldsymbol{x}, \pi^n(\theta)) > 0$, the parameter set $\Theta_o^n = \{\theta \in \text{supp}(\pi^n) : \arg\max_y p(y|\boldsymbol{x}, \theta) \neq \hat{y}\}$ is not empty. We only keep the nonzero terms in K:

$$K(\boldsymbol{x}, \pi^n(\theta)) = \sum_{\theta \in \Theta_o} \pi^n(\theta)[\max_y p(y|\boldsymbol{x}, \theta) - p(\hat{y}|\boldsymbol{x}, \theta)]. \tag{3.19}$$

For binary classification, $\hat{y} = \arg\max_y p^n(y|\boldsymbol{x})$, indicating that the predictive probability $p^n(\hat{y}|\boldsymbol{x}) \geq 0.5$. For $\theta \in \Theta_o$, $p(\hat{y}|\boldsymbol{x}, \theta) < 0.5$, we have: if $\theta \in \Theta_o,\ \pi^n(\theta|\boldsymbol{x}, \hat{y}) = \frac{\pi^n(\theta)p(\hat{y}|\boldsymbol{x},\theta)}{p^n(\hat{y}|\boldsymbol{x})} < \pi^n(\theta)$.

In the case that we observe $(\boldsymbol{x}, \hat{y})$ in $(n$+1$)$-th iteration, the updated posterior predictive probability $p^n(\hat{y}|\boldsymbol{x}, \{\boldsymbol{x}, \hat{y}\}) \geq p^n(\hat{y}|\boldsymbol{x}) \geq 0.5$ and therefore $\max_y p^n(y|\boldsymbol{x}, \{\boldsymbol{x}, \hat{y}\}) = \hat{y}$. Hence,

$$K(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, \hat{y})) = \sum_{\theta \in \Theta_o} \pi^n(\theta|\boldsymbol{x}, \hat{y})[\max_y p(y|\boldsymbol{x}, \theta) - p(\hat{y}|\boldsymbol{x}, \theta)] < K(\boldsymbol{x}, \pi^n(\theta)). \tag{3.20}$$

Since $K(\pi^n(\theta|\boldsymbol{x}, \hat{y}), \boldsymbol{x}) \neq K(\pi^n(\theta), \boldsymbol{x})$, we have $G(\boldsymbol{x}, \pi^n(\theta)) > \mathbb{E}_{p^n(y|x)}[G(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y))]$ and

$$\begin{aligned} U^w(\boldsymbol{x}; \pi^n(\theta)) &= \mathbb{E}_{p(\boldsymbol{x}')}[G(\boldsymbol{x}', \pi^n(\theta))] - \mathbb{E}_{p(\boldsymbol{x}')}[\mathbb{E}_{p^n(y|x)}[G(\boldsymbol{x}', \pi^n(\theta|\boldsymbol{x}, y))]] \\ &\geq p(\boldsymbol{x})[G(\boldsymbol{x}, \pi^n(\theta)) - \mathbb{E}_{p^n(y|x)}[G(\boldsymbol{x}, \pi^n(\theta|\boldsymbol{x}, y))]] > 0. \end{aligned} \tag{3.21}$$

This concludes our proof. □

Lemma 5. *If following some policy a candidate $\boldsymbol{x}$ is measured infinitely often almost surely, then $\lim_{n\to\infty} U^w(\boldsymbol{x}; \pi^n(\theta)) = 0$ almost surely.*

Intuitively, if a candidate has been measured many times, there is no benefit to measure it again.

Proof. Adding a new data point $(\boldsymbol{x}, y)$ to D, the posterior change is: $\pi^n(\theta|\boldsymbol{x}, y) = \frac{\pi^n(\theta)p(y|\boldsymbol{x},\theta)}{p^n(y|\boldsymbol{x})}$. Define $\Theta_x = \{\theta \in \Theta : p(y|\boldsymbol{x}, \theta) = p(y|\boldsymbol{x}, \theta_r)\}$. Denote $N_x(n)$ as the times of the candidate $\boldsymbol{x}$ being queried at the n-th iteration. Based on the posterior consistency theory we have $\sum_{\theta \in \Theta_x} \pi^n(\theta) \xrightarrow{a.s.} 1$ as $N_x(n) \to \infty$ [31]. Since $p^n(y|\boldsymbol{x}) = \sum_{\theta \in \Theta} \pi^n(\theta)p(y|\boldsymbol{x}, \theta)$, we have $\lim_{n\to\infty} p^n(y|\boldsymbol{x}) \xrightarrow{a.s.} p(y|\boldsymbol{x}, \theta_r)$. Hence $\lim_{n\to\infty} \pi^n(\theta|\boldsymbol{x}, y) - \pi^n(\theta) = 0$ almost surely, which indicates $\lim_{n\to\infty} U^w(\boldsymbol{x}; \pi^n(\theta)) = 0$ almost surely. □

With these lemmas, we can prove the convergence of Weighted-MOCU based active learning:

Theorem 1. *Assume that both $\mathcal{X}$ and Θ are discrete with finite elements, the true model parameter $\theta_r \in \Theta$ and the prior distribution $\pi^0(\theta)$ over Θ satisfies $\pi^0(\theta_r) > 0$; then for the active learning algorithm defined by the acquisition function (3.10), we have $\lim_{n\to\infty} \mathcal{M}(\pi^n(\theta)) = 0$ almost surely.*

Proof. As the number of active learning iterations $n \to \infty$, following the acquisition function (3.10), some of the candidates can be measured infinite times. Define $\mathcal{X}_A \subset \mathcal{X}$ as the set whose candidates have been measured infinite times. Denote the measuring sequence of the candidates following (3.10) as $\{x_n\}$, we have: $\exists N,\ s.t.\ \forall n > N, x_n \in \mathcal{X}_A$. Based on Lemma 5, $\lim_{n\to\infty} U^w(x_n; \pi^n(\theta)) = 0$.

On the other hand, since with the Weighted MOCU $U^w(x_n; \pi^n(\theta)) = \max_{\boldsymbol{x}\in\mathcal{X}} U^w(\boldsymbol{x}; \pi^n(\theta))$, then $\lim_{n\to\infty} U^w(x_n; \pi^n(\theta)) = 0$ indicates that $\forall \boldsymbol{x} \in \mathcal{X}$, $U^w(\boldsymbol{x}; \pi^n(\theta))$ uniformly converges to 0. Based on Lemma 4, $\lim_{n\to\infty} \mathcal{M}_n^w = 0$ and we can conclude the proof with Lemma 1. $\square$

3.4.3 Soft-MOCU-based Active Learning

As we discussed, the myopic behavior that MOCU-based active learning has is due to the linear function pieces causing the acquisition function for active learning to lose the guiding capability when the update of $\pi(\theta|\boldsymbol{x}, y)$ is not significant enough. To address this problem, we propose a new acquisition function based on modified MOCU, which has the theoretical convergence guarantee to the true optimal classifier.

In this chapter, We approximate the maximum operator in (3.2) by the `log-sum-exp` function:

$$\max_y p(y|\boldsymbol{x}) \approx \frac{1}{k} \log[\sum_y \exp(k \cdot p(y|\boldsymbol{x}))], \tag{3.22}$$

where k is a parameter controlling the approximation. Using a larger k in `log-sum-exp` gives a better approximation to the maximum operator. Note that other functions can also be used. With this approximation, we can define the following *Soft MOCU* (SMOCU) as:

$$\mathcal{M}^s(\pi(\theta)) = \mathbb{E}_{\boldsymbol{x}}\{\mathbb{E}_{\pi(\theta)}[\max_y p(y|\boldsymbol{x}, \theta)] - \frac{1}{k} \log[\sum_y \exp(k \cdot p(y|\boldsymbol{x}))]\}, \tag{3.23}$$

which is now a strictly concave function instead of being piecewise linear. Similarly, as with larger k, Soft MOCU gets closer to MOCU. We illustrate the modified MOCU with different k values in Fig. 3.2 for the example described in Section 3.4.1.

We now define an acquisition function by the reduction of Soft MOCU to guide active learning

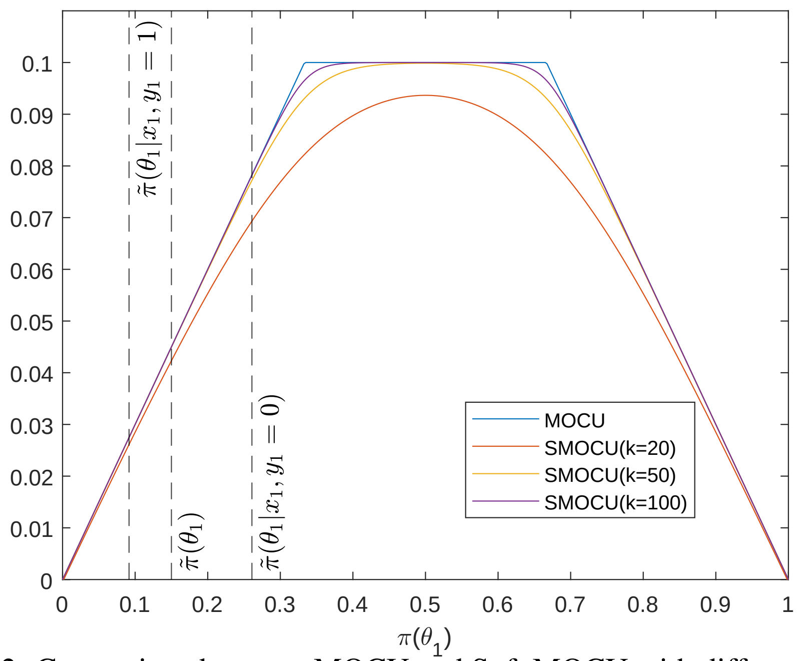


Figure 3.2: Comparison between MOCU and Soft MOCU with different k values.

in the one-step-look-ahead manner:

$$U^s(\boldsymbol{x}; \pi(\theta)) = \mathcal{M}^s(\pi(\theta)) - \mathbb{E}_{y|\boldsymbol{x}}[\mathcal{M}^s(\pi(\theta|\boldsymbol{x}, y))]. \tag{3.24}$$

As shown in the example, Soft MOCU can provide a good approximation to MOCU. More critically, it also has large curvature on the changing points of MOCU so that the above acquisition function has large values when the update of $\pi(\theta|\boldsymbol{x}, y)$ is significant causing the change of OBC. While when the update of $\pi(\theta|\boldsymbol{x}, y)$ is not significant (falling within intervals of linear pieces in the original MOCU), Soft MOCU still has small curvature so that the acquisition function has small positive values instead of being zero as in (3.3). With these properties of the Soft-MOCU-based acquisition function, when the model has high uncertainty with large MOCU values, the approximation by Soft MOCU will not affect the choice of candidates and the corresponding active learning performs similarly as the original MOCU-based method to achieve short-term optimality. On the other hand, when the model has low uncertainty and a single query will not be able to change $\pi(\theta|\boldsymbol{x}, y)$ significantly, for example when $\pi(\theta_1)$ is close to 0 or 1 in Fig. 3.2, the MOCU-based method will get stuck. However, our Soft-MOCU-based acquisition function can still guide active learning out of the myopic behavior. Please refer to Appendix B.1 for the pseudo-code of our Soft-MOCU-based active learning and the complexity analysis.

Theoretical Convergence Guarantee

We now first prove that Soft MOCU (3.23) is a strictly concave function. If active learning is guided by the acquisition function (3.24) based on Soft MOCU, MOCU will converge to 0. This means that we can learn the optimal classifier of the true model without getting stuck with the theoretical convergence guarantee.

We assume that both $\mathcal{X}$ and Θ are discrete with finite elements, and the true model parameter $\theta_r \in \Theta$ with $\pi^0(\theta_r) > 0$ for the prior $\pi^0(\theta)$.

Lemma 6. *$\mathcal{M}^s(\pi(\theta))$ is a strictly concave function of $\pi(\theta)$.*

Proof. It is known that `log-sum-exp` is a convex function (page 74, Sec. 3.1 in [5]). We now prove that $f(p(y|\boldsymbol{x})) = \frac{1}{k}\log[\sum_y \exp(k \cdot p(y|\boldsymbol{x}))]$ is a strictly convex function of $p(y|\boldsymbol{x})$ conditioning on $\sum_y p(y|\boldsymbol{x}) = 1$. In the following proof, we denote $p(y|\boldsymbol{x})$ for $y \in \mathcal{Y}$ as the vector $\boldsymbol{z}$ for simplicity. From [5],

$$\nabla^2 f(\boldsymbol{z}) = k(\text{diag}(g) - gg^T), \ g := \frac{\exp(\boldsymbol{z})}{\mathbf{1}^T \exp(\boldsymbol{z})}, \tag{3.25}$$

where $\exp(\boldsymbol{z}) = (e^{z_1}, \dots, e^{z_M})$. Note that in the expression of $\nabla^2 f(\boldsymbol{z})$, diag$(g)$ is a full-rank matrix and rank(gg^T) is 1. Therefore, $rank(\nabla^2 f) = n - 1$ and $f(\boldsymbol{z})$ is affine (being a linear function) along only one direction. Apparently that direction is along the all-ones vector $\mathbf{1}$, as can be verified by: $\mathbf{1}^T \nabla^2 f(\boldsymbol{z}) \mathbf{1} = 0$, and f is strictly convex along any other directions. In addition, since $\boldsymbol{z}$ denotes a probability mass function, it is constrained on the hyperplane $\mathbf{1}^T \boldsymbol{z} = 1$. On the hyperplane, no vector is parallel to $\mathbf{1}$, as $\mathbf{1}^T(\boldsymbol{z} + \alpha\mathbf{1}) \neq 1$ for $\alpha \neq 0$. Hence, within the hyperplane f is a strictly convex function.

Since $f(\cdot)$ is a strictly concave function and $p(y|\boldsymbol{x})$ is a linear function of $\pi(\theta)$, $\log[\sum_y \exp(k \cdot p(y|\boldsymbol{x}))]$ is therefore a strictly convex function of $\pi(\theta)$. $\mathcal{M}^s(\pi(\theta))$ is equal to a linear function subtracting a strictly convex function and hence is a strictly concave function of $\pi(\theta)$. $\square$

Lemma 7. $\forall \boldsymbol{x} \in \mathcal{X}, U^s(\boldsymbol{x}; \pi(\theta)) \geq 0$*; the equality only holds for the case* $\pi(\theta) = \pi(\theta|\boldsymbol{x}, y),\ \forall y \in \mathcal{Y}$.

Proof. Since Soft MOCU is a strictly concave function and $\mathbb{E}_{y|\boldsymbol{x}}[\pi(\theta|\boldsymbol{x}, y))] = \pi(\theta)$, by Jensen's inequality, we have

$$U^s(\boldsymbol{x}; \pi(\theta)) = \mathcal{M}^s(\pi(\theta)) - \mathbb{E}_{y|\boldsymbol{x}}[\mathcal{M}^s(\pi(\theta|\boldsymbol{x}, y))] \geq 0.$$

and the equality only holds if $\pi(\theta) = \pi(\theta|\boldsymbol{x}, y),\ \forall y \in \mathcal{Y}$. $\square$

Lemma 8. *If* $U^s(\boldsymbol{x}; \pi(\theta)) = 0$, $\forall \boldsymbol{x} \in \mathcal{X}$*, then* $\mathcal{M}(\pi(\theta)) = 0$.

Proof. This lemma states that if the acquisition function values of all the candidates are 0, then we can conclude that MOCU is 0. This means that the OBC of $\pi(\theta)$ has converged to the true optimal classifier ψ_{θ_r}. MOCU-based active learning does not have such a property. Because of that, it may get stuck before converging to the true optimal classifier. $\square$

Proof. We will show that the lemma holds by proving the contraposition: if $\mathcal{M}(\pi(\theta)) > 0$, $\exists \boldsymbol{x} \in \mathcal{X}\ s.t.\ U^s(\boldsymbol{x}; \pi(\theta)) > 0$.

Based on (3.2), $M(\pi(\theta)) > 0$ indicating $\exists \boldsymbol{x}^* \in \mathcal{X}\ \exists \theta^* \in$ supp(π) s.t. $\psi_{\pi(\theta)}(\boldsymbol{x}^*) \neq \psi_{\theta^*}(\boldsymbol{x}^*)$, i.e. $\max_y p(y|\boldsymbol{x}^*) \neq \max_y p(y|\boldsymbol{x}^*, \theta^*)$, where supp($\pi$) is the support of $\pi(\theta)$. So $\exists y^* \in \mathcal{Y}$ s.t. $p(y^*|\boldsymbol{x}^*, \theta^*) \neq p(y^*|\boldsymbol{x}^*)$. Now we assume that we observe $(\boldsymbol{x}^*, y^*)$, then the update of $\pi(\theta^*)$ can be written as:

$$\pi(\theta^*|\boldsymbol{x}^*, y^*) = \frac{\pi(\theta^*)p(y|\boldsymbol{x}^*, \theta^*)}{p(y^*|\boldsymbol{x}^*)}. \tag{3.26}$$

Since $p(y^*|\boldsymbol{x}^*, \theta^*) \neq p(y^*|\boldsymbol{x}^*)$, we have $\pi(\theta^*|\boldsymbol{x}^*, y^*) \neq \pi(\theta^*)$. With Lemma 7, we can have $U^s(\boldsymbol{x}^*; \pi(\theta)) > 0$. That concludes our proof. $\square$ $\square$

Lemma 9. *If a candidate* $\boldsymbol{x}$ *is measured infinitely often almost surely (a.s.),* $U^s(\boldsymbol{x}; \pi^n(\theta)) \xrightarrow{a.s.} 0$ *as* $n \to \infty$.

Intuitively, if a candidate has been measured many times, there is no benefit to measure it again.

Proof. For a candidate $\boldsymbol{x}$, define a set of θ as $\Theta_{\boldsymbol{x}} = \{\theta \in \Theta : p(y|\boldsymbol{x}, \theta) = p(y|\boldsymbol{x}, \theta_r)\}$. At the n-th iteration, assume that the candidate $\boldsymbol{x}$ has been observed $N_{\boldsymbol{x}}(n)$ times and $\lim_{n\to\infty} N_{\boldsymbol{x}}(n) = \infty$. Based on the posterior consistency theory we have $\sum_{\theta\in\Theta_x} \pi^n(\theta) \xrightarrow{a.s.} 1$ as $n \to \infty$ [31]. Since $p^n(y|\boldsymbol{x}) = \sum_{\theta\in\Theta} \pi^n(\theta)p(y|\boldsymbol{x}, \theta)$, we have $p^n(y|\boldsymbol{x}) \xrightarrow{a.s.} p(y|\boldsymbol{x}, \theta_r)$. By Bayes' rule, $\pi^n(\theta|\boldsymbol{x}, y) = \frac{\pi^n(\theta)p(y|\boldsymbol{x},\theta)}{p^n(y|\boldsymbol{x})}$, and hence we have $\pi^n(\theta|\boldsymbol{x}, y) - \pi^n(\theta) \xrightarrow{a.s.} 0$. With Lemma 7, we can conclude that $U^s(\boldsymbol{x}_n; \pi^n(\theta)) \xrightarrow{a.s.} 0$ as $n \to \infty$. □ □

Theorem 2. *Assume that both $\mathcal{X}$ and Θ are discrete with finite elements, the true model parameter $\theta_r \in \Theta$ and $\pi^0(\theta_r) > 0$; then for the active learning algorithm defined by the acquisition function (3.24), we have $\mathcal{M}(\pi^n(\theta)) \xrightarrow{a.s.} 0$ as $n \to \infty$.*

Proof. As the number of active learning iterations $n \to \infty$, some of the candidates will be measured infinitely often. Following the Soft-MOCU-based method by the acquisition function (3.24), denote the set of candidates being measured infinitely often as $\mathcal{X}_I = \{\boldsymbol{x} \in \mathcal{X} : \lim_{n\to\infty} N_{\boldsymbol{x}}(n) = \infty\}$. With the query sequence of the candidates as $\{\boldsymbol{x}_n\}$, we have $\exists N,\ s.t.\ \forall n > N, \boldsymbol{x}_n \in \mathcal{X}_I$, which means that after N iterations, we can only observe candidates from the set $\mathcal{X}_I$. Based on Lemma 9, this indicates $U^s(\boldsymbol{x}_n; \pi^n(\theta)) \xrightarrow{a.s.} 0$.

On the other hand, as Soft-MOCU-based active learning maximizes the acquisition function in each iteration, we have $U^s(\boldsymbol{x}_n; \pi^n(\theta)) = \max_{\boldsymbol{x}\in\mathcal{X}} U^s(\boldsymbol{x}; \pi^n(\theta))$. Then the maximum value $U^s(\boldsymbol{x}_n; \pi^n(\theta))$ converging to 0 means that $U^s(\boldsymbol{x}; \pi^n(\theta)),\ \boldsymbol{x} \in \mathcal{X}$ converges to 0 uniformly. Based on Lemma 8, we have $\mathcal{M}(\pi^n(\theta)) \xrightarrow{a.s.} 0$ and we can conclude the proof. □ □

We should emphasize that the inverse of Lemma 8 is not true. When MOCU is 0, the acquisition function of some candidate $\boldsymbol{x}$'s can still be positive. To understand this, as we have shown in Section 3.4.1, MOCU does not capture all the model uncertainty. On the other hand, based on Lemma 7, the acquisition function based on Soft MOCU can only be 0 when there is no model uncertainty.

3.5 Empirical Results of Weighted-MOCU-based Method

We benchmark our Weighted-MOCU method with other active learning algorithms, including random sampling, MES [71], BALD [37] and ELR [65], on both simulated and real-world classification datasets. In the following experiments, we set $c = 1$ for the Weighted MOCU function.

3.5.1 Simulated Experiments.

In addition to the one-dimensional simulated example introduced in Section 3.2, we test our model on a similar simulation setting as the *block in the middle* dataset in [37], where noisy observations with flip error are simulated in a block region on the decision boundary. We generate data based on a two-dimensional Bayesian logistic regression model: $p(y = 1|\boldsymbol{x}, \boldsymbol{w}, b) = \frac{1}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x}-b)}$ with $\boldsymbol{x} \in [-4, 4]^2$. The block region is within $[-0.5, 0.5]^2$ with the flip error rate equal to 0.3. For the model parameter prior, $w_1 \sim \mathcal{U}(0.3, 0.8)$ is uniformly distributed and $w_2 \sim \mathcal{U}(-0.25, 0.25)$ and $b \sim \mathcal{U}(-0.25, 0.25)$; w_1, w_2 and b are independent.

We randomly sample 100 particles from the parameter prior with one of the particles as the true model parameter. The five active learning algorithms are compared for 500 iterations by the OBC error with respect to the testing data generated from the true model. We repeat the simulations for 500 runs and plot the average performance with standard deviation bars in Fig. 3.3. The error regret is defined as the error difference between the OBC and the true optimal classifier. From the figure, MES simply chooses the candidates with the predictive probability closest to 0.5, it can sample many noisy observations from the block region. ELR performs well in the first several iterations but poorly after 200 samples. Our Weighted MOCU performs the best.

We have also benchmarked our Weighted-MOCU based method with other active learning methods for a synthetic multi-class classification problem. We assume that the probabilistic model $p(y|\boldsymbol{x}, \sigma_y^2) = f_y(\boldsymbol{x}, \sigma_y^2)/\sum_{y'} f(\boldsymbol{x}, \sigma_{y'}^2)$ with $\boldsymbol{x} \in [-2, 2]^2$, $y \in \{0, 1, 2\}$ and $f_y = \exp(-(x - m_y)^2/2\sigma_y^2)$. We set m_y to be $(0, 0)$, $(1, 0)$, $(0, 1)$ for $y = 0, 1, 2$ respectively; and $\sigma_y^2 \sim \mathcal{U}(1, 5)$ being the uncertain parameters. Same as the previous binary classification experiment, we test for 300 runs and plot the average performance with standard deviations in Fig. 3.4. We can observe that

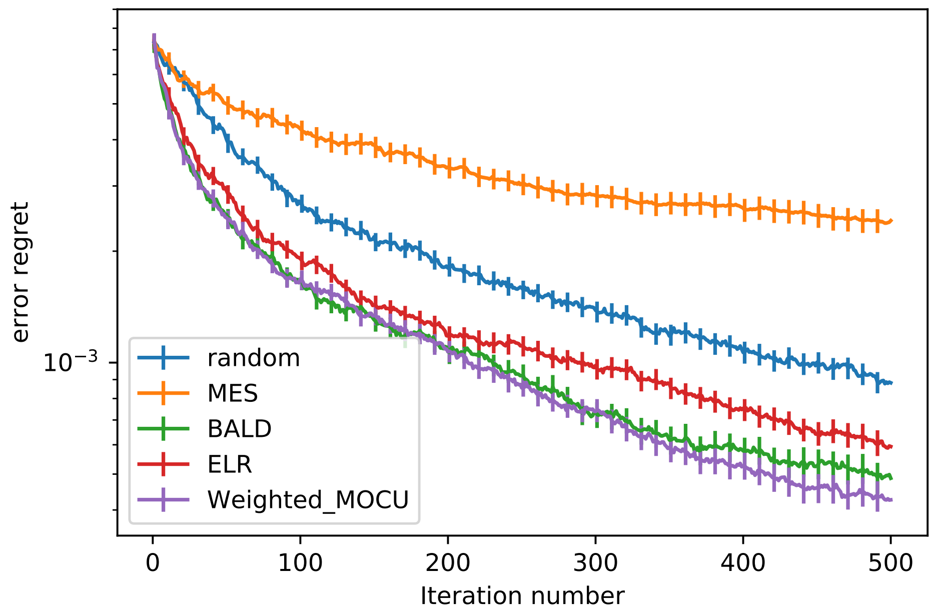


Figure 3.3: The expected OBC error regret comparison between different active learning algorithms on binary classification.

ELR performs poorly in the long run while our Weighted MOCU has better empirical performance on par with BALD.

More results and discussion are in Appendix A.5.

3.5.2 Real-world Benchmark Experiments.

We also present the results on the UCI User Knowledge dataset [38]. The dataset includes 403 samples assigned to 4 classes (High, Medium, Low, Very Low) with each sample having five features in $[0, 1]^5$. We have grouped the samples into two classes with 224 samples in High or Medium, 179 in Low or Very Low. We consider the first and fifth features for classification and equally divide the feature space into 4×4 bins. For the i-th bin, the probability of candidates belonging to High or Medium is denoted by $\theta_i, 1 \leq i \leq 16$ and θ_i's are independent and $\theta_i \sim \text{Beta}(\alpha_i, \beta_i)$, with hyperparameters α_i and β_i. We present the results with the uncertainty class by setting $\alpha_i = \beta_i = 10$ in eight randomly chosen bins and for the other bins, $\alpha_i = 5, \beta_i = 2$ if the true frequency of High or Medium in the i-th bin is lower than 0.5 and $\alpha_i = 2, \beta_i = 5$ otherwise. We have randomly drawn

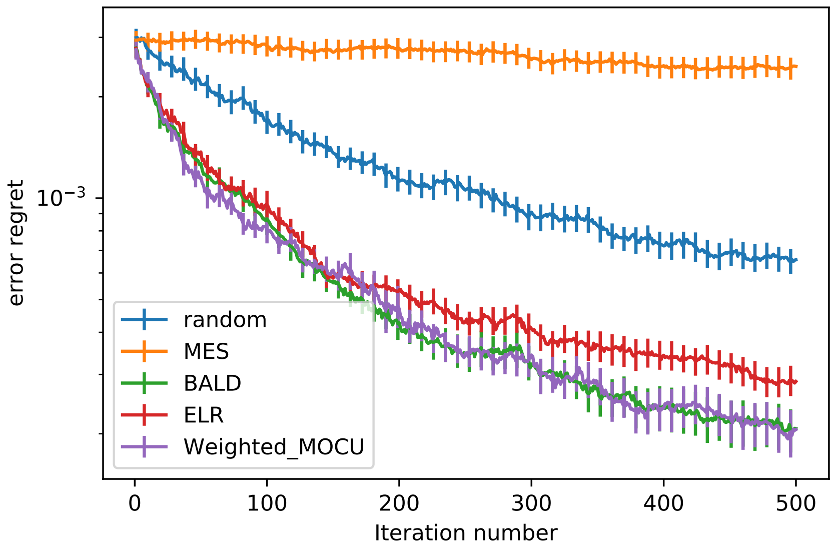


Figure 3.4: The expected OBC error regret comparison between different active learning algorithms on 3 class classification.

150 samples from each class as the candidate pool and perform the five different active learning algorithms. We repeat the whole procedure 150 times and the average error rates are shown in Fig. 3.5. While ELR clearly gets stuck in this setup, our Weighted MOCU method can converge to the optimal classifier with less samples than all the competing methods. BALD performs poorly as the bins with $\alpha = \beta = 10$ have less uncertainty but have more impact on OBC prediction and BALD fails to identify that. More comprehensive results and discussion, including results on the UCI Letter Recognition dataset [22], can be found in Appendix A.6.

3.6 Empirical Results of Soft-MOCU-based Method

We first investigate the influence of the parameter k on the performance of our Soft-MOCU-based active learning (SMOCU). We then benchmark SMOCU with other active learning methods, including random sampling, MES [71], BALD [37] and MOCU, on both simulated and real-world classification datasets.

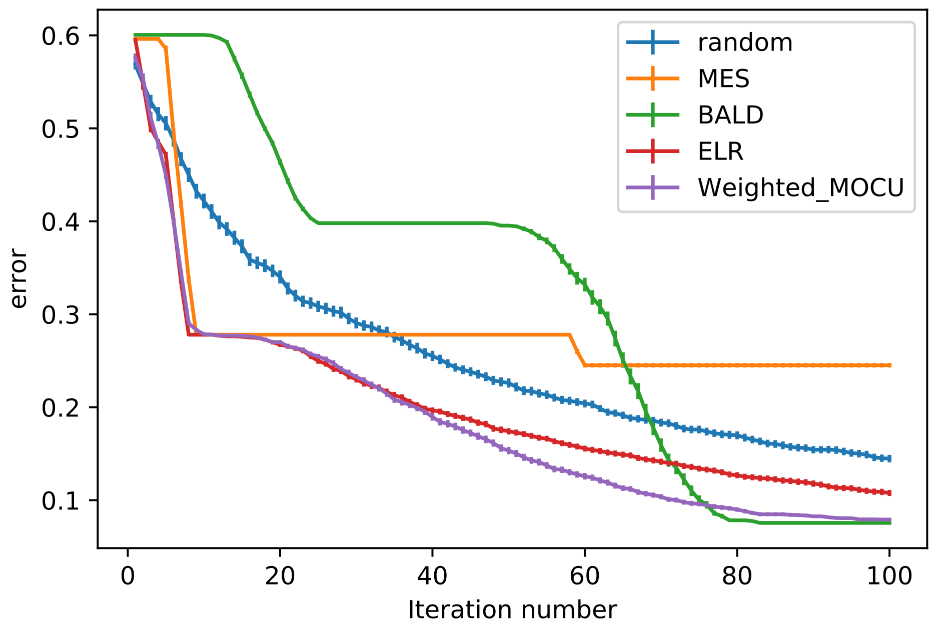


Figure 3.5: Classification error rate comparison on UCI User Knowledge dataset.

3.6.1 Performance of Soft-MOCU with Different k Values

Here we compare the performance of SMOCU with different k values together with MOCU and BALD on a binary classification problem with one feature $x \in [-4, 4]$. The underlying probabilistic model is:

$$\begin{aligned} p(y=1|x,\alpha,\beta) &= S(x) + \epsilon(x,\alpha,\beta) \\ S(x) &= 0.6\frac{\exp(x)}{1+\exp(x)} + 0.2 \\ \epsilon(x,\alpha,\beta) &= \alpha\exp(-x^2) + \beta[\exp(-(x-4)^2) + \exp(-(x+4)^2)], \end{aligned} \tag{3.27}$$

where $\boldsymbol{\theta} = (\alpha, \beta)^{\mathrm{T}}$ is the uncertain parameter vector with α and β independently uniformly distributed in the intervals $[-0.1, 0.1]$ and $[-0.2, 0.2]$ respectively. Fig. 3.6a illustrates the uncertain probabilistic model with red lines indicating the upper and lower bounds of the predictive probability. The probabilistic model has higher uncertainty near $x = \pm 4$ depending on β than the uncertainty near $x = 0$ depending on α. Observing data near $x = \pm 4$ can reduce model uncertainty significantly and is preferred by BALD, but it cannot help on the label prediction since the optimal classifier will always label $x = \pm 4$ as 1 or 0. On the other hand, as the optimal labels of the points in the middle are uncertain given the prior knowledge, MOCU-based active learning will query these points first to better reduce the classification error at the beginning.

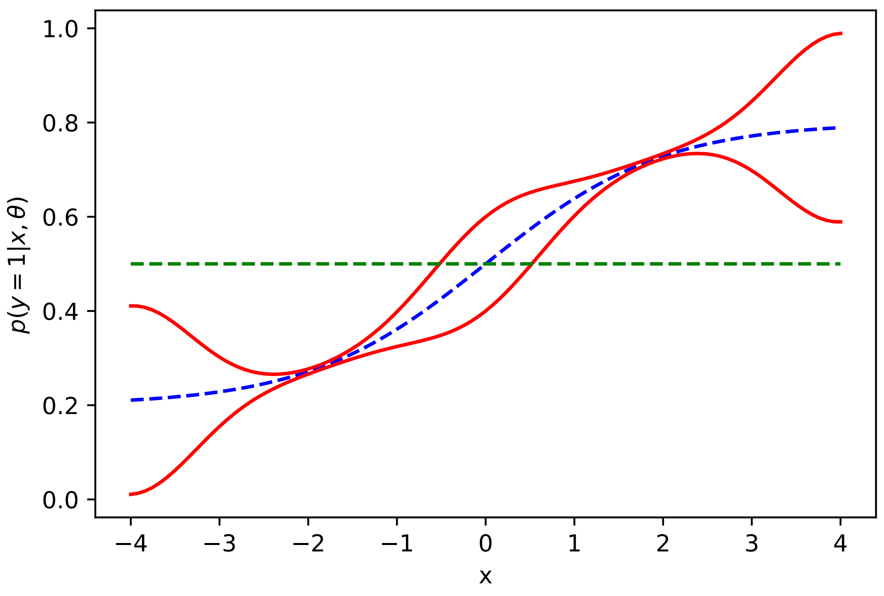
We randomly sample the true parameters from the prior and perform different active learning methods for 300 iterations. We compare different methods by the error regret, which is defined as the error difference between the OBC and the true optimal classifier. We repeat the simulations for 500 runs and plot the average performance with standard deviation bars in Fig. 3.6b. From the figure, not surprisingly, BALD performs inefficiently at the beginning since it queries the candidates on both sides. MOCU performs well at the beginning but becomes inefficient after about 100 iterations, indicating some of the 500 simulations get stuck as we analyzed in Section 3.4.1.

For Soft MOCU with different k values, as we shown in Fig. 3.2, Soft MOCU gets closer to MOCU with increasing k. As a result, Soft-MOCU-based active learning should perform more similar to the MOCU-based method as k increases. We can see from the figure that, when k is small (k = 1), the performance is close to BALD that aims to reduce the total model uncertainty. With increasing k (= 10 or 100), the performance of Soft-MOCU-based active learning at the beginning gets closer to MOCU and more importantly, in the long run it performs better than both BALD and MOCU, demonstrating Soft-MOCU-based active learning can converge to the optimal classifier with fewer iterations. As expected, when k is really large ($k = 10000$), Soft MOCU can get really close to MOCU with very small curvature with respect to $\pi(\theta)$ as illustrated in Fig. 3.2, which leads to similar performance degradation as shown in Fig. 3.6b.

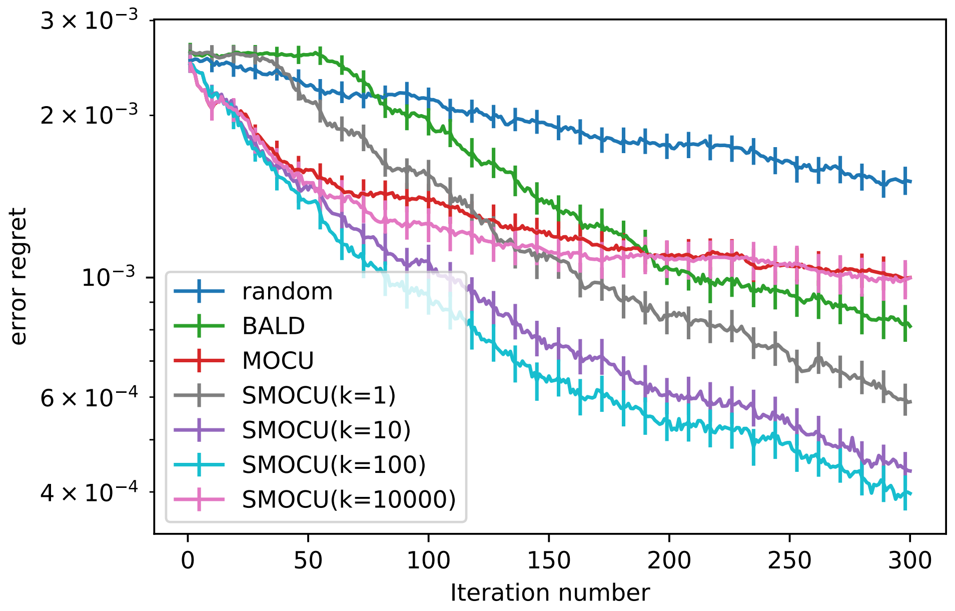
We next benchmark Soft-MOCU-based active learning for more simulated experiments and real-world experiments, for which we compare active learning methods based on random sampling, MES, BALD, MOCU, and our Soft MOCU with $k = 10$ and 100.

3.6.2 Simulated Experiments

We test these active learning methods on a simulated experiment similar as the *block in the center* dataset in Houlsby et al. [37]. The experiment includes a binary classification problem with candidates from 2-d feature space $[-4, 4]^2$. The simulated data are described by a Bayesian logistic



(a) Predictive probability of class 1



(b) Error regret comparison among methods

Figure 3.6: (a) Predictive probability of class 1 under uncertainty: the red lines indicate the upper and lower bounds of the predictive probability; the blue dashed line is the mean of the predictive probability; the green dashed line indicates that the probability is equal to 0.5. (b) Active learning performance.

regression model:

$$p(y=1|\boldsymbol{x},\boldsymbol{w},b) = \frac{1}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x}-b)}, \tag{3.28}$$

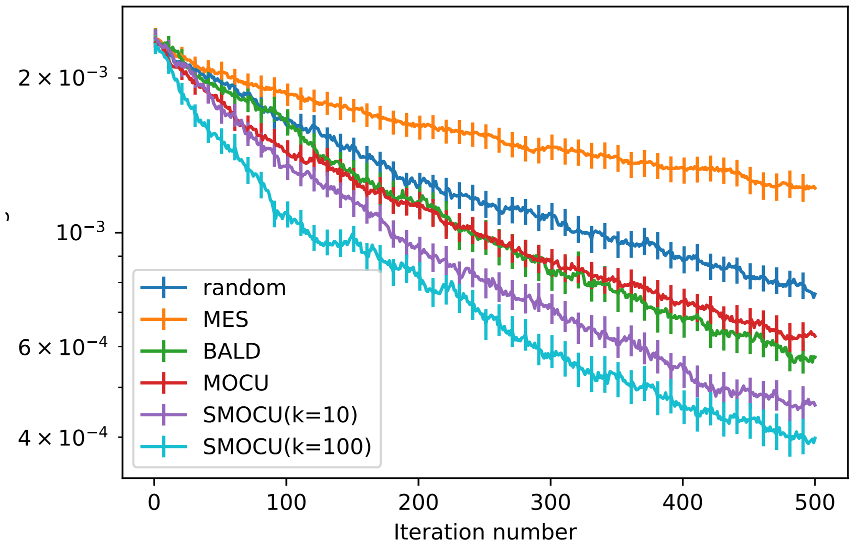


Figure 3.7: Comparison of different active learning methods based on the expected OBC error regret for binary classification.

with a uniform parameter prior $w_1 \sim \mathcal{U}(0.3, 0.8)$, $w_2 \sim \mathcal{U}(-0.1, 0.1)$ and $b \sim \mathcal{U}(-0.25, 0.25)$; w_1, w_2 and b are independent. With this prior setting, the uncertainty of $p(y|\boldsymbol{x}, \boldsymbol{w}, b)$ is low in the region near the x_2 axis where the decision boundary lies and the uncertainty is high in the region far away from the x_2 axis. Within the block region of $[-1, 1]^2$, the observed labels are flipped with the probability 0.3.

We randomly sample 100 particles from the parameter prior as the uncertain parameter set, and randomly choose one of them as the true parameter. We also uniformly sample 100 candidates from the feature space as the candidate pool. Then we perform these different methods for 500 iterations and calculate the error regret. We repeat the simulation for 500 times and plot the performance comparison with standard deviations in Fig. 3.7. From the figure, MES has quite poor performance as it simply queries the candidates with the predictive probability close to 0.5. It may sample many noisy observations from the noisy block region. BALD performs poorly at the beginning since it cannot identify which uncertainty is related to the learning objective. MOCU performs well in the first several iterations, but poorly in the long run. As expected, our Soft-MOCU-based methods perform better than other competing methods with $k = 100$ performing the best.

We also compare these different methods on a multi-class classification setup. We assume the

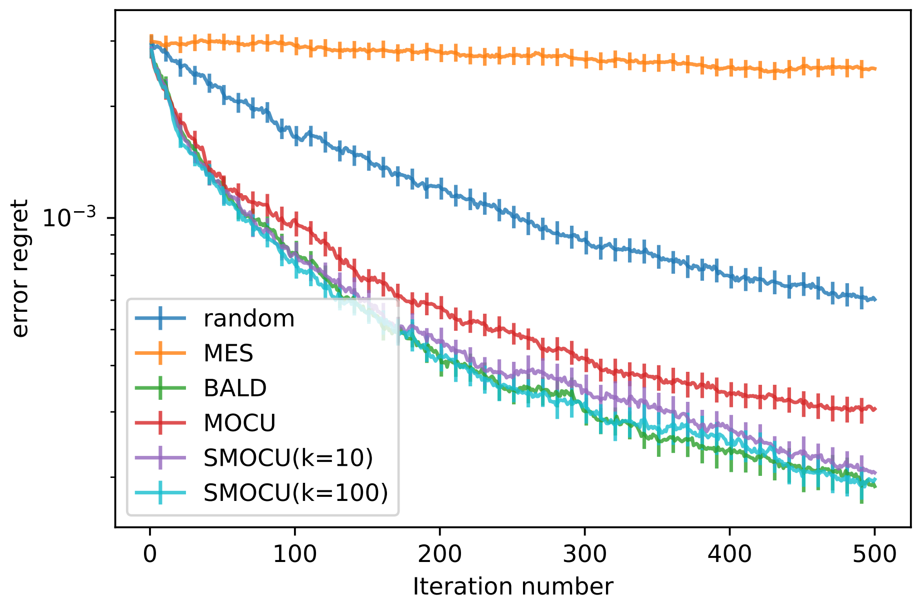


Figure 3.8: Comparison of different active learning methods based on the expected OBC error regret for three-class classification.

feature space $\mathcal{X} = [-2, 2]^2$ and label space $\mathcal{Y} = \{0, 1, 2\}$ with the probabilistic model $p(y|\boldsymbol{x}, \sigma_y^2) = f_y(\boldsymbol{x}, \sigma_y^2)/\sum_{y'} f(\boldsymbol{x}, \sigma_{y'}^2)$, where $f_y(\boldsymbol{x}, \sigma_y^2) = \exp(-(x - m_y)^2/2\sigma_y^2),\ y \in \mathcal{Y}$. We set m_y to be $(0, 0)$, $(1, 0)$, $(0, 1)$ for $y = 0, 1, 2$ respectively; and independent uncertain parameters $\sigma_y^2 \sim \mathcal{U}(1, 5),\ y \in \mathcal{Y}$. Similar as the previous binary classification experiment, we perform the five methods for 500 times and plot the average error regret with standard deviations in Fig. 3.8. From the figure, MES performs poorly as it samples the candidates with maximal predictive entropy, while querying these candidates provides little information to improve classification. We again observe that MOCU performs poorly in the long run while both Soft-MOCU-based methods have better empirical performance on a par with BALD.

3.6.3 Real-world Benchmark Experiments

We compare different active learning methods on the UCI User Knowledge dataset [38]. The dataset assigns the knowledge status of 403 students into four levels (High, Medium, Low, Very Low) based on five input features in $[0, 1]^5$, which reflect the degree of study or exam performance. Here we use the 1st and 5th features as inputs for classification and equally separate the 2-d feature

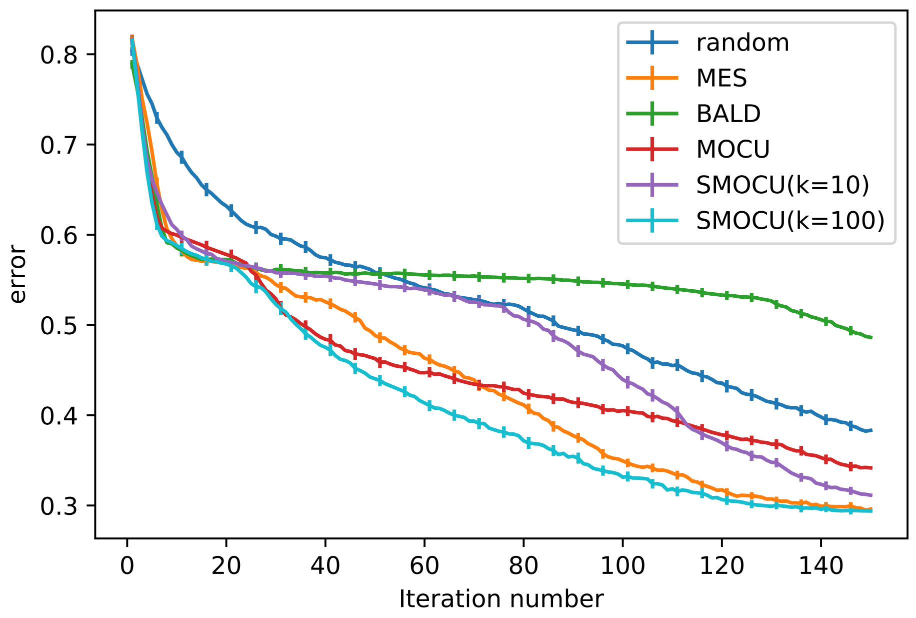


Figure 3.9: Classification error comparison on UCI User Knowledge dataset.

space into 4×4 bins. Within the ith bin we assume a categorical distribution for the knowledge levels $p(y|\boldsymbol{x} \in i\text{-th bin}) = \boldsymbol{p}^{(i)},\ 1 \leq i \leq 16$ with parameters $\boldsymbol{p}^{(i)} = (p_0^{(i)}, p_1^{(i)}, p_2^{(i)}, p_3^{(i)})$. Assume that each parameter independently follows a Dirichlet distribution $\boldsymbol{p}^{(i)} \sim \text{Dir}(\boldsymbol{\alpha}^{(i)})$ with $\boldsymbol{\alpha}^{(i)}$ as the hyperparameters. We randomly choose 8 bins and set uniform priors on them with $\boldsymbol{\alpha}^{(i)} = \mathbf{1}$. For the other 8 bins, we set the prior by setting $\alpha_j^{(i)} = 1$ if j is the true label, and $\alpha_j^{(i)} = 10$ for other labels.

To obtain a balanced classification problem, we randomly sample 50 samples from each class to test the five different methods. We repeat the active learning procedures for 150 times and compare the average classification error in Fig. 3.9. From the figure, we can clearly observe two stages in the active learning procedures: the first stage has about 20 iterations, in which all the methods learn the optimal classification rules in the 8 bins with the uniform prior; while in the following iterations as the second stage, different methods perform differently based on their acquisition functions. BALD keeps choosing candidates from the bins with the uniform prior in the second stage as those bins still have larger model uncertainty. However, they cannot help improve the classification. MOCU performs well at the beginning, and then converges slowly. Our Soft-MOCU-based method with $k = 100$ is again demonstrated to converge faster than other methods.

4. EFFICIENT ACTIVE LEARNING FOR GAUSSIAN PROCESS CLASSIFICATION BY ERROR REDUCTION

4.1 Overview

Active learning sequentially selects the best instance for labeling by optimizing an acquisition function to enhance data/label efficiency. The selection can be either from a discrete instance set (pool-based scenario) or a continuous instance space (query synthesis scenario). In this work, we study both active learning scenarios for Gaussian Process Classification (GPC). The existing active learning strategies that maximize the Estimated Error Reduction (EER) aim at reducing the classification error after training with the new acquired instance in a one-step-look-ahead manner. The computation of EER-based acquisition functions is typically prohibitive as it requires retraining the GPC with every new query. Moreover, as the EER is not smooth, it can not be combined with gradient-based optimization techniques to efficiently explore the continuous instance space for query synthesis. To overcome these critical limitations, we develop computationally efficient algorithms for EER-based active learning with GPC. We derive the joint predictive distribution of label pairs as a one-dimensional integral, as a result of which the computation of the acquisition function avoids retraining the GPC for each query, remarkably reducing the computational overhead. We also derive the gradient chain rule to efficiently calculate the gradient of the acquisition function, which leads to the first query synthesis active learning algorithm implementing EER-based strategies. Our experiments clearly demonstrate the computational efficiency of the proposed algorithms. We also benchmark our algorithms on both synthetic and real-world datasets, which show superior performance in terms of sampling efficiency compared to the existing state-of-the-art algorithms.

4.2 Introduction

Compared to traditional passive learning with randomly sampled training instances, active learning aims at "optimally" querying instances for labeling to achieve label efficiency when training machine learning models, especially when labeling is difficult or costly. There are two fundamental scenarios of active learning discussed in the literature: query synthesis and pool-based sampling [73]. In query synthesis, the leaner can request labels for any instance generated from a continuous feature space while pool-based sampling selects the instance from a finite set. Query synthesis is more challenging due to the infinite search space and inherent higher label uncertainty. Recent research on query synthesis with deep generative models has shown promising potential [91, 70]. However, in many science and engineering applications, acquiring the label for even one instance is prohibitively resource-demanding and therefore active learning with deep models may not be practical.

For example, one of critical materials science research questions is to identify phase transition diagrams, where the phase transition response surface can be complex [60]. Identifying phase transitions can be formulated as finding the optimal classification boundaries between different phases in the enormous materials design space. However, precisely knowing the phase of each design with the corresponding input features requires costly and time-consuming materials synthesis and profiling experiments or running complex simulation models. Furthermore, there may exist significant uncertainty in acquired phase labels due to technical limitations. Hence, active learning for optimal Bayesian classifiers is a natural solution to help identify the phase diagram with as few as possible synthesized or simulated materials under such uncertainty and complexity. Gaussian Process Classification (GPC) as a popular and powerful Bayesian classifier with the flexibility of adopting different kernels [82], is suitable for solving this problem with appropriate active learning strategies. Fig.4.1 shows an example of phase identification in a two-dimensional design space by active learning with GPC. In the figure, the black solid line indicates the transition response surface, the crosses and dots indicate the queries of different phases, and the colorbar indicates the predictive distribution of GPC. From the figure we can see, the predictive distribution identifies the phase transition boundary with a few samples guided by active learning using GPC surrogate models, which has significant cost and time savings compared to the traditional trial-and-error phase diagram identification paradigm in the current materials science practice.

Many active learning strategies have been proposed for GPC. For example, Bayesian Active

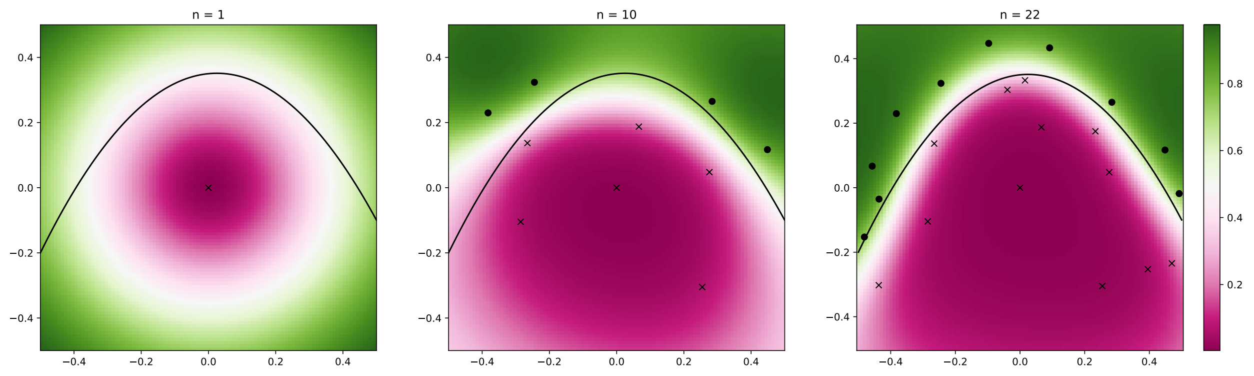


Figure 4.1: Phase diagram identification by active learning with GPC.

Learning by Disagreement (BALD) selects the instance with the maximum mutual information between the observation and the derived uncertain model [37]. There also have been strategies targeting at reducing classification error directly or indirectly. Estimated Error Reduction (EER) strategies optimize for the error reduction after training with the new queries [72, 65, 40]. We note here that EER-based active learning has been studied for both Gaussian Process Regression (GPR) and GPC [72, 40]. For GPR, the regression error can be represented by the posterior predictive variance with the analytical expression and the acquisition function is easy to calculate for efficient active learning. In contrast, for GPC, there is no analytical expression for the posterior predictive. The model updates need approximate inference, such as Expectation Propagation (EP), an algorithm iteratively approximating the GPC posterior [48]. Moreover, the classification error computation requires the new query labels; so the calculation of the corresponding EER-based acquisition function requires incrementally retraining the model for each possible query label to calculate the expected future error as the utility to guide active learning. The complexity of training GPC with EP approximation is $O(n^3)$, where n is the number of observed data. This has been the main hurdle to develop active learning for GPC models.

To reduce the calculation cost of EER, the paper [40] proposed to use a non-iterative but less accurate method approximating the new posterior when calculating the acquisition functions. The paper [29] proposed a novel approximated error reduction (AER) criterion, in which the error reduction of a candidate is estimated based on the impact over its nearby instances. The approximated estimation avoids re-inferring the labels of massive instances. These methods are relatively efficient in computation. But besides EP approximation, they need additional approximations in calculating acquisition functions so the acquisition functions are not precisely calculated, which may degrade the desired data efficiency.

In this chapter, within the EER-based active learning framework for GPC, we develop computationally efficient algorithms to compute EER to guide both query synthesis and pool-based active learning. In particular, we consider EER as the reduction of the Mean Objective Cost of Uncertainty (MOCU) [87] since the learning objective of GPC, in particular for identifying phase diagram, is to reduce the classification error. By deriving the joint distribution of queries as a one-dimensional integral, we avoid retraining the GPC for each query when calculating the EER/MOCU-based acquisition function. We further leverage a smooth approximation of MOCU, Soft MOCU (SMOCU) [88], to enable efficient gradient computation of the SMOCU reduction by deriving the corresponding chain rule for efficient query synthesis with GPC. We emphasize that this is the first algorithms for query synthetic active learning based on EER strategies to the best of our knowledge. We show in experiment that our algorithm accelerate the computation of the acquisition functions. Compared with other benchmark algorithms, we demonstrate the sample efficiency of our algorithms with both synthetic and real-world datasets.

4.3 Problem Setting and Background

4.3.1 Gaussian Process Classification (GPC)

Consider a binary classification problem with the instance space $\mathcal{X}$ and binary label set $\mathcal{Y} = \{0, 1\}$, we aim to train a classifier $\psi : \mathcal{X} \to \{0, 1, \ldots, M-1\}$ to predict labels for unobserved instances $\psi(\boldsymbol{x}_*), \boldsymbol{x}_* \in \mathcal{X}$. The Gaussian Process classification (GPC) framework connects the instance and the label by a latent function f, which is a random process depending on $\boldsymbol{x}$. Assume that f follows a Gaussian Process (GP) prior $f \sim \text{GP}(\mu(\cdot), k(\cdot, \cdot))$, where $\mu(\cdot)$ is a mean function and $k(\cdot, \cdot)$ is a covariance kernel function [64]. GP is a popular and powerful model for both regression and classification. A good property of GP is that given any finite number of instances

$\boldsymbol{x}_i$, the joint distribution of $f(\boldsymbol{x}_i)$ is still Gaussian. For classification problems, given f, the label y takes a Bernoulli distribution with probability $p(y=1|\boldsymbol{x}, f) = \Phi(f(\boldsymbol{x}))$, where Φ is the Gaussian cumulative distribution function.

Given a sequence of observations $D = \{X, Y\}$ with $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, the class labels are conditionally independent given the latent function. Therefore, the joint likelihood can be factorized as: $p(Y|X, f) = \prod_i p(y_i|\boldsymbol{x}_i, f)$. Since the likelihood probability is non-Gaussian, the posterior $\pi(f|X,Y) \propto \pi(f)p(Y|f,X)$ can not be computed analytically and approximation are often adopted. The Expectation Propagation (EP) algorithm approximates $\pi(f|X,Y)$ with a Gaussian approximation $q(f|X,Y) = \mathcal{N}(f|\tilde{\mu}(\cdot), \tilde{\Sigma}(\cdot,\cdot))$ by iteratively moment matching marginal posteriors [64].

With the approximated posterior, the marginal distribution of $f_* = f(\boldsymbol{x}_*)$ is still Gaussian. Denote the mean and variance as μ_* and σ_{**} respectively, then there is a closed-form expression for $p(y_*|\boldsymbol{x}_*, X, Y)$ as:

$$p(y_* = 1|X, Y, \boldsymbol{x}_*) = \int \Phi(f_*)\phi(f_*|\mu_*, \sigma_*^2)df_* = \Phi(\frac{\mu_*}{\sqrt{1+\sigma_{**}}}). \tag{4.1}$$

Given the predictive probability, we assume the prediction of the instance is the most probable label $\arg\max_{y_*} p(y_*|X, Y, \boldsymbol{x}_*)$, which is known as the Optimal Bayesian Classifier (OBC) [13].

4.4 Efficient Active Learning for GPC

In this section, we present our EER-based active learning algorithms for GPC based on the acquisition functions defined by the MOCU and SMOCU reduction. As defined in the previous chapter, the expressions of MOCU reduction $U^{\mathrm{M}}(\boldsymbol{x}_*)$ and SMOCU reduction $U^{\mathrm{S}}(\boldsymbol{x}_*)$ are:

$$U^{\mathrm{M}}(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{x}_s}\{\mathbb{E}_{y_*|\boldsymbol{x}_*}[\max_{y_s} p(y_s|\boldsymbol{x}_s, \boldsymbol{x}, y)] - \max_{y_s} p(y_s|\boldsymbol{x}_s)\}, \tag{4.2}$$

$$U^{\mathrm{S}}(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{x}_s}\{\mathbb{E}_{y_*|\boldsymbol{x}_*}[\frac{1}{k}\texttt{LogSumExp}(k\cdot p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*))] - \frac{1}{k}\texttt{LogSumExp}(k\cdot p(y_s|\boldsymbol{x}_s))\}. \tag{4.3}$$

We emphasize that the posterior predictive distributions $p(y_s|\boldsymbol{x}_s)$ and $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*))$ are all conditional on X, Y. For the sake of clarity, we omit X, Y from the notations. $U^{\mathrm{M}}(\boldsymbol{x}_*)$ is not a smooth function with the maximization operators, while $U^{\mathrm{S}}(\boldsymbol{x}_*)$ is a smooth function of $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$. We will leverage this smooth acquisition function to derive the first efficient gradient-based query synthesis active learning algorithm for GPC.

At each iteration, with the updated GPC given previous observations, the acquisition function (4.2) or (4.3) can be optimized to guide the selection of the next query for active learning. We first present a straightforward algorithm for both the discrete instance set (pool-based sampling) and continuous instance space (query synthesis) scenarios, where the acquisition function is optimized by random optimization. Random optimization first collects a random sample set $\mathcal{X}_* \subset \mathcal{X}$ of size M_1, calculates the acquisition function for each sample in the set and then takes the instance with the maximum acquisition function value as the query.

To calculate (4.2) or (4.3), the integral over $\mathcal{X}$ space is not analytical. Hence we need to calculate the integral by Monte Carlo sampling with M_2 samples of $\boldsymbol{x}_s \in \mathcal{X}$. Define $g^{\mathrm{M}}(\boldsymbol{x}_s; \boldsymbol{x}_*)$ and $g^{\mathrm{S}}(\boldsymbol{x}_s; \boldsymbol{x}_*)$ such that $U^{\mathrm{M}}(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{x}_s}\{g^{\mathrm{M}}(\boldsymbol{x}_s; \boldsymbol{x}_*)\}$ and $U^{\mathrm{S}}(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{x}_s}\{g^{\mathrm{S}}(\boldsymbol{x}_s; \boldsymbol{x}_*)\}$. Let $g(\boldsymbol{x}_s; \boldsymbol{x}_*)$ denote either $g^{\mathrm{M}}(\boldsymbol{x}_s; \boldsymbol{x}_*)$ or $g^{\mathrm{S}}(\boldsymbol{x}_s; \boldsymbol{x}_*)$. For each $\boldsymbol{x}_s$, the calculation of $g(\boldsymbol{x}_s; \boldsymbol{x}_*)$ requires deriving the probability distribution of $p(y_s|\boldsymbol{x}_s)$ and $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*), \forall y_* \in \{0, 1, \ldots, M-1\}$. Here, $p(y_s|\boldsymbol{x}_s)$ can be calculated directly from (4.1) while updating $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ needs incremental training of GPC with observations $\{X, Y, \boldsymbol{x}_*, y_*\}$ based on the EP approximation, and then calculating (4.1). The whole procedure of optimizing the acquisition function at the n-th iteration is illustrated in Algorithm 1, in which we need to retrain GPC for each possible pair $(\boldsymbol{x}_*, y_*)$ as shown in the 6-th line. Therefore, the EP approximation needs to be performed $2 \times M_1$ times.

There are three issues of the acquisition function calculation in Algorithm 1. First, we need a large number of samples to have a reliable estimation of the integral in (4.2) or (4.3). Second, the calculation of $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ requires incremental retraining of the GPC model for each pair of $(\boldsymbol{x}_*, y_*)$, with computational complexity $O(M_1 n^3)$. Third, even though $U^{\mathrm{S}}(\boldsymbol{x}_*)$ is a differentiable function of $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$, in the algorithm we actually use the EP approximation $q(f|\boldsymbol{x}_*, y_*)$

Algorithm 1 Random EER-based active learning for GPC: n-th iteration

```
1:  function RandomOptimization($p(\boldsymbol{x}), q(f|X,Y)$)
2:      Sample $M_1$ samples of $\boldsymbol{x}_* \sim p(\boldsymbol{x})$
3:      Sample $M_2$ samples of $\boldsymbol{x}_s \sim p(\boldsymbol{x})$
4:      for each $\boldsymbol{x}_*$ do
5:          Calculate $p(y_*|\boldsymbol{x}_*)$ by (4.1)
6:          for $y_*$ in $\{0,1\}$ do
7:              Use EP to approximate the posterior $q(f|\boldsymbol{x}_*, y_*)$
8:              for each $\boldsymbol{x}_s$ do
9:                  Calculate $p(y_s|\boldsymbol{x}_s)$ and $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ by (4.1)
10:                 Update $g(\boldsymbol{x}_s; \boldsymbol{x}_*)$ with $y_*$
11:             end for
12:         end for
13:         $U(\boldsymbol{x}_*) = \frac{1}{M_2}\sum_{\boldsymbol{x}_s} g(\boldsymbol{x}_s; \boldsymbol{x}_*)$
14:     end for
15:     return $\tilde{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}_*} U(\boldsymbol{x}_*)$
16: end function
```

to calculate $U^{\text{S}}(\boldsymbol{x}_*)$, and it is impossible to calculate the gradient $\nabla q(f|\boldsymbol{x}_*, y_*)$ during the EP procedure.

We develop our EER-based active learning algorithms to address these three presented challenges: 1) By importance sampling leveraging inhere GPC assumptions, we reduce the required number of samples for estimating acquisition functions; 2) We also propose to obtain $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ by deriving analytic solution to marginalize the joint distribution $p(y_s, y_*|\boldsymbol{x}_s, \boldsymbol{x}_*)$ from the approximated posterior $q(f|X, Y)$, which avoids the retraining with EP approximation; 3) More critically, we derive the first gradient-based query synthesis algorithm when using the SMOCU-based acquisition function by deriving the gradient $\nabla q(f|\boldsymbol{x}_*, y_*)$ for efficient active learning together with the aforementioned two strategies.

4.4.1 Importance Sampling

Regarding the issue in requiring the high sampling number of $\boldsymbol{x}_s$ for reliable estimation of the acquisition function, we notice that most of the kernels applied in GPC assume that the observed data only have influence on neighboring regions. Hence, we only need to account for the samples near $\boldsymbol{x}_*$ to estimate its influence on the classification error, thereafter the acquisition function for each new query. More specifically, we can use importance sampling to reduce the required sampling number. Reformulate the expectation $U(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{x}_s \sim p(\boldsymbol{x}_s)}[g(\boldsymbol{x}_s; \boldsymbol{x}_*)]$ over an assistant distribution $\tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)$ as:

$$U(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{x}_s \sim \tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)}[p(\boldsymbol{x}_s) g(\boldsymbol{x}_s; \boldsymbol{x}_*) / \tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)]. \tag{4.4}$$

The variance of the expectation is minimized when $\tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)$ is proportional to $p(\boldsymbol{x}_s) g(\boldsymbol{x}_s; \boldsymbol{x}_*)$. But this requires knowledge of the value of $U(\boldsymbol{x}_*)$, which is the acquisition function that we try to estimate. In practice, we can choose $\tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*) \propto k(\boldsymbol{x}_s, \boldsymbol{x}_*) p(\boldsymbol{x}_s)$ instead, as $k(\boldsymbol{x}_s, \boldsymbol{x}_*)$ reflects the non-zero region of $g(\boldsymbol{x}_s; \boldsymbol{x}_*)$: $k(\boldsymbol{x}_s, \boldsymbol{x}_*) \approx 0$ means $(\boldsymbol{x}_s, y_s)$ and $(\boldsymbol{x}_*, y_*)$ are independent, thus $p(y_s | \boldsymbol{x}_s, \boldsymbol{x}_*, y_*) \approx p(y_s | \boldsymbol{x}_s)$ and $g(\boldsymbol{x}_s; \boldsymbol{x}_*) \approx 0$. For example, if $p(\boldsymbol{x}_s)$ is uniformly distributed within a finite region, $k(\boldsymbol{x}_s, \boldsymbol{x}_*)$ is a square exponential kernel, then $\tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)$ can be chosen as a truncated Gaussian distribution. Note that importance sampling and random optimization is only suitable for continuous instance space, or discrete set with large cardinality, For small instance set, we can traverse all the elements for calculating the expectation and optimizing the acquisition function.

4.4.2 Joint Distribution Calculation

To avoid the retraining of GPC for each $(\boldsymbol{x}_*, y_*)$, we can calculate the posterior predictive by $p(y_s | \boldsymbol{x}_s, \boldsymbol{x}_*, y_*) = p(y_s, y_* | \boldsymbol{x}_s, \boldsymbol{x}_*) / p(y_* | \boldsymbol{x}_*)$, which requires computing the joint distribution of $p(y_s, y_* | \boldsymbol{x}_s, \boldsymbol{x}_*)$. We remind the reader all the probabilities are conditioned on $\{X, Y\}$ and we omit them for the seek of brevity. Denote $f_s = f(\boldsymbol{x}_s)$ and $f_* = f(\boldsymbol{x}_*)$. Now we show how to simplify the calculation of $p(y_s, y_* | \boldsymbol{x}_s, \boldsymbol{x}_*)$. In the Gaussian approximation of the posterior $q(f | X, Y)$, the joint distribution of f_s, f_* is still Gaussian. Since y_s and y_* are conditionally independent given f_s and f_*, the joint distribution can be expressed as:

$$p(y_s = 1, y_* = 1 | \boldsymbol{x}_s, \boldsymbol{x}_*) = \iint \Phi(f_s) \Phi(f_*) \phi(f_s, f_* | \mu_{s*}, \Sigma_{s*}) df_s df_*, \tag{4.5}$$

where μ_{s*} and Σ_{s*} are the marginal mean and covariance matrix of f_s and f_*. This integral can be simplified as a one-dimensional integral. Denote the elements in the variance as:

$$\mu_{s*} = \begin{pmatrix} \mu_s \\ \mu_* \end{pmatrix}, \qquad \Sigma_{s*} = \begin{pmatrix} \sigma_{ss} & \sigma_{s*} \\ \sigma_{s*} & \sigma_{**} \end{pmatrix}. \tag{4.6}$$

We can decompose the joint Gaussian distribution as the marginal distribution of f_s times the conditional distribution of f_* given f_s, i.e.

$$\phi(f_s, f_*|\mu_{s*}, \Sigma_{s*}) = \phi(f_s|\mu_s, \sigma_{ss})\phi(f_*|\tilde{\mu}_*(f_s), \tilde{\sigma}_{**}), \tag{4.7}$$

where $\tilde{\mu}_*(f_s) = \mu_* + (f_s - \mu_s)\sigma_{s*}/\sigma_{ss}$ and $\tilde{\sigma}_{**} = \sigma_{**} - \sigma_{s*}^2/\sigma_{ss}$. Therefore, (4.7) can be transformed as:

$$\begin{aligned} p(y_s = 1, y_* = 1|\boldsymbol{x}_s, \boldsymbol{x}_*) &= \iint \Phi(f_*)\Phi(f_s)\phi(f_s, f_*|\mu_{s*}, \Sigma_{s*})df_s df_*, \\ &= \iint \Phi(f_*)\phi(f_*|\tilde{\mu}_*(f_s), \tilde{\sigma}_{**})df_*\ \Phi(f_s)\phi(f_s|\mu_s, \sigma_{ss})df_s \\ &= \int \Phi(\frac{\tilde{\mu}_*(f_s)}{\sqrt{\tilde{\sigma}_{**} + 1}})\Phi(f_s)\phi(f_s|\mu_s, \sigma_{ss})df_s. \end{aligned} \tag{4.8}$$

The last line is based on the integral equation introduced in [64]. The above equation (4.8) calculates the joint distribution with the 1-d integral in constant time. With the joint distribution $p(y_s = 1, y_* = 1|\boldsymbol{x}_s, \boldsymbol{x}_*)$, we can easily obtain the joint distribution of y_s, y_* with other label pairs, and finally we can obtain the posterior predictive $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ without retraining the GPC with EP approximation. Based on the techniques in this and previous subsection, Algorithm 1 can be modified as illustrated in Algorithm 2. We name the active learning algorithm with MOCU reduction as Non-Retraining MOCU reduction with Random Optimization (NR-MOCU-RO), and name the algorithm with SMOCU reduction as Non-Retraining Soft MOCU with Random Optimization (NR-SMOCU-RO).

Algorithm 2 NR-(S)MOCU-RO: n-th iteration

```
1:  function RandomOptimization($p(\boldsymbol{x}), q(f|X, Y)$)
2:      Sample $M_1$ samples of $\boldsymbol{x}_* \sim p(\boldsymbol{x})$
3:      for each $\boldsymbol{x}_*$ do
4:          Calculate $p(y|\boldsymbol{x}_*)$ by (4.1)
5:          Sample $M_2$ samples of $\boldsymbol{x}_s \sim \tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)$
6:          for $y_*$ in $\{0, 1\}$ do
7:              for each $\boldsymbol{x}_s$ do
8:                  Calculate $p(y_s, y_*|\boldsymbol{x}_s, \boldsymbol{x}_*)$ by (4.8)
9:                  Calculate $p(y_s|\boldsymbol{x}_s)$ by (4.1)
10:                 Calculate posterior $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*) = p(y_s, y_*|\boldsymbol{x}_s, \boldsymbol{x}_*)/p(y_s|\boldsymbol{x}_s)$
11:                 Update $g(\boldsymbol{x}_s; \boldsymbol{x}_*)$ with $y_*$
12:             end for
13:         end for
14:         $U(\boldsymbol{x}_*) = \frac{1}{M_2} \sum_{\boldsymbol{x}_s} p(\boldsymbol{x}_s) g(\boldsymbol{x}_s; \boldsymbol{x}_*)/\tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)$
15:     end for
16:     return $\tilde{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}_*} U(\boldsymbol{x}_*)$
17: end function
```

4.4.3 Gradient Calculation

With the introduced marginalization strategy and importance sampling, we can significantly improve the computational efficiency for pool-based active learning with GPC. However, in the query synthesis problems, we would like to optimize the acquisition functions with gradient-based algorithms. Usually the acquisition functions are multi-modal in the feature space, so the common practice is to perform random optimization first, and then take the optimal point as the initial point to perform the gradient-based algorithms [35].

Here we consider the gradient calculation of the acquisition function based on the SMOCU reduction $\nabla U^{\mathrm{S}}(\boldsymbol{x}_*)$ for query synthesis active learning as $U^{\mathrm{S}}(\boldsymbol{x}_*)$ is a smooth function whose gradients exist everywhere. For a Gaussian Process, the gradients of its mean and covariance functions have closed-form expressions of the gradients of its adopted kernel function. Here we assume that given the EP approximation $q(f|X, Y)$ and any pair of points $(\boldsymbol{x}_s, \boldsymbol{x}_*)$, we already know the gradient of $\nabla\mu_*$, $\nabla\sigma_{**}$ and $\nabla\sigma_{s*}$ with respect to $\boldsymbol{x}_*$. With this assumption, we can use the chain rule to compute the gradients and finally express the gradients $U^{\mathrm{S}}(\boldsymbol{x}_*)$ in the form of $\nabla\mu_*$,

$\nabla\sigma_{**}$ and $\nabla\sigma_{s*}$.

In (4.3), the second term is unrelated to $\boldsymbol{x}_*$, so the gradient of the SMOCU reduction can be expressed as:

$$\begin{aligned}\nabla U^{\mathrm{S}}(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{x}_s}[\nabla g^{\mathrm{S}}(\boldsymbol{x}_s,\boldsymbol{x}_*)] &= \nabla\mathbb{E}_{\boldsymbol{x}_s}\{\sum_{y_*} p(y_*|\boldsymbol{x}_*)\frac{1}{k}\texttt{LogSumExp}[k\cdot p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*)]\}\\ &= \sum_{y_*}\nabla p(y_*|\boldsymbol{x}_*)\cdot\mathbb{E}_{\boldsymbol{x}_s}\{\frac{1}{k}\texttt{LogSumExp}[k\cdot p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*)]\}\\ &+\sum_{y_*} p(y_*|\boldsymbol{x}_*)\cdot\mathbb{E}_{\boldsymbol{x}_s}\{\nabla\frac{1}{k}\texttt{LogSumExp}[k\cdot p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*)]\}.\end{aligned} \tag{4.9}$$

In the first term, $\nabla p(y_*|\boldsymbol{x}_*)$ can be calculated with $\nabla\mu_*$ and $\nabla\sigma_{**}$ based on (4.1). For the second term, we can use the chain rule to compute $\nabla\texttt{LogSumExp}[k\cdot p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*)] = g_1\cdot g_2$, where:

$$g_1 = \frac{\partial\texttt{LogSumExp}[k\cdot p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*)]}{\partial p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*)}, \qquad g_2 = \nabla p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*). \tag{4.10}$$

Since $p(y_s|\boldsymbol{x}_s,\boldsymbol{x}_*,y_*) = p(y_s,y_*|\boldsymbol{x}_s,\boldsymbol{x}_*)/p(y_*|\boldsymbol{x}_*)$, g_2 can be calculated with $\nabla p(y_*|\boldsymbol{x}_*)$ and $\nabla p(y_s,y_*|\boldsymbol{x}_s,\boldsymbol{x}_*)$ based on the derivative of a fraction. $\nabla p(y_s,y_*|\boldsymbol{x}_s,\boldsymbol{x}_*)$ can be calculate based on (4.8):

$$\begin{aligned}\nabla p(y_s=1,y_*=1|\boldsymbol{x}_s,\boldsymbol{x}_*) =&\nabla\int\Phi(f_s)\phi(f_s|\mu_s,\sigma_{ss})\Phi(\frac{\tilde{\mu}_*(f_s)}{\sqrt{\tilde{\sigma}_{**}+1}})df_s\\ &=\int\Phi(f_s)\phi(f_s|\mu_s,\sigma_{ss})\phi(\frac{\tilde{\mu}_*(f_s)}{\sqrt{\tilde{\sigma}_{**}+1}})\cdot\nabla(\frac{\tilde{\mu}_*(f_s)}{\sqrt{\tilde{\sigma}_{**}+1}})df_s,\end{aligned} \tag{4.11}$$

which is again a 1-d integral. The gradient of $\nabla(\frac{\tilde{\mu}_*(f_s)}{\sqrt{\tilde{\sigma}_{**}+1}})$ can be again calculated by chain rule, and connected to the calculation of $\nabla\tilde{\mu}_*(f_s)$ and $\nabla\tilde{\sigma}_{**}$:

$$\nabla\tilde{\mu}_*(f_s) = \nabla\mu_* + \frac{f_s-\mu_s}{\sigma_{ss}}\nabla\sigma_{s*}, \qquad \nabla\tilde{\sigma}_{**} = \nabla\sigma_{**} - \frac{2\nabla\sigma_{s*}}{\sigma_{ss}}. \tag{4.12}$$

Therefore, $\nabla(\frac{\tilde{\mu}_*(f_s)}{\sqrt{\tilde{\sigma}_{**}+1}})$ is a linear function of f_s, and we can use numerical integral methods to calculate (4.11). The query synthesis algorithm with the integral computation is summarized in

Algorithm 3 named as NR-SMOCU with Stochastic Gradient Descent (NR-SMOCU-SGD).

In summary, to reduce the number of samples $\boldsymbol{x}_s$ for relable estimation of acquisition functions, our algorithm utilizes an importance sampling with an assistant distribution chosen according to the kernel function. By calculating the posterior predictive directly from the joint distribution, the algorithm avoids retraining GPC with EP approximation. The introduction of the joint distribution also enables the efficient calculation of the gradient of the smooth acquisition function, with which we develop an efficient active learning algorithm for query synthesis.

Algorithm 3 NR-SMOCU-SGD: n-th iteration

```
1: function GradientOpt($p(\boldsymbol{x}), q(f|X,Y)$)
2:     Obtain initial point $\boldsymbol{x}_*$ from RandomOpt($p(\boldsymbol{x}), q(f|X,Y)$)
3:     while not converge do
4:         Sample $M_2$ samples of $\boldsymbol{x}_s \sim \tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)$
5:         Calculate $p(y_*|\boldsymbol{x}_*)$ and $\nabla p(y_*|\boldsymbol{x}_*)$ by (4.1)
6:         for each $\boldsymbol{x}_s$ do
7:             for $y_*$ in $\{0,1\}$ do
8:                 Calculate $p(y_s, y_*|\boldsymbol{x}_s, \boldsymbol{x}_*)$ , $\nabla p(y_s, y_*|\boldsymbol{x}_s, \boldsymbol{x}_*)$ and $p(y_s|\boldsymbol{x}_s)$ by (4.1, 4.8, 4.11)
9:             end for
10:            Calculate $\nabla g^{\mathrm{S}}(\boldsymbol{x}_s, \boldsymbol{x}_*)$ by (4.9 - 4.11)
11:        end for
12:        $\nabla U^{\mathrm{S}}(\boldsymbol{x}_*) = \frac{1}{M_2}\sum_{\boldsymbol{x}_s} p(\boldsymbol{x}_s)\nabla g^{\mathrm{S}}(\boldsymbol{x}_s; \boldsymbol{x}_*)/\tilde{p}(\boldsymbol{x}_s; \boldsymbol{x}_*)$
13:        Update $\boldsymbol{x}_*$ with $\nabla U^{\mathrm{S}}(\boldsymbol{x}_*)$
14:    end while
15:    return $\boldsymbol{x}_*$
16: end function
```

4.5 Experiments

In this section we demonstrate the efficiency of our active learning algorithms combined with either random optimization (NR-MOCU-RO, NR-SMOCU-RO) or Adagrad (NR-SMOCU-SGD) in the following sets of experiments. In the first set of experiments, we analyze and benchmark the running time of our algorithm by comparing to the naive computation of the MOCU/SMOCU reduction. Then we benchmark our algorithms with other active learning algorithms, including random sampling, Maximum Entropy Search (MES) [71] and Bayesian Active Learning by Disagreement (BALD) [37] for both query synthesis on synthetic benchmark datasets, and pool-based active learning on real-world datasets. In our experiments, we use GP prior for f with the squared-exponential kernels $k(\boldsymbol{x}, \boldsymbol{x}') = \gamma^2 \text{exp}(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/l^2)$, where $\{\gamma, l\}$ are model hyperparameters. The label probability is modeled with the probit function as $p(y|\boldsymbol{x}, f) = \Phi(f(\boldsymbol{x}))$.

4.5.1 Estimation Accuracy and Running Time Comparison

We first evaluate the effect of using the joint distribution integral in calculating the acquisition functions. We compare the estimation of $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ through the joint distribution integral (4.8), with $p'(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ estimated by retraining GPC with EP approximation. For this set of experiments, we generate the initial data points using a latent function f sampled form the GP prior. The instance space $\mathcal{X} = [-4, 4]$, and the hyperparameters $\gamma^2 = 0.5$, $l^2 = 1$. We initially sample 100 data points to train GPC, then we compare the values $g = \texttt{LogSumExp}(k \cdot p(y_s|\boldsymbol{x}_s, \boldsymbol{x}, y))/k$ and $g' = \texttt{LogSumExp}(k \cdot p'(y_s|\boldsymbol{x}_s, \boldsymbol{x}, y))/k$, since these are related to the calculation of the SMOCU reduction. With 1000 pairs of $(\boldsymbol{x}_s, \boldsymbol{x}_*)$ randomly sampled from $\mathcal{X}$, we calculate the absolute error ratio as $|g - g'|/g'$. The average absolute error ratio is *1.8e-5* and the maximum error ratio is *2.4e-3*. We also compare the values $g = \max p(y_s|\boldsymbol{x}_s, \boldsymbol{x}, y)$ and $g' = \max p'(y_s|\boldsymbol{x}_s, \boldsymbol{x}, y)$, which is used to calculate the MOCU reduction. The average absolute error ratio is *1.3e-5* and the maximum ratio is *2.2e-3*. These results validate that using (4.8) can provide accurate estimates of the acquisition functions.

Next, we compare the running time of estimating acquisition functions by three algorithms: 1) a naive algorithm calculating $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ with GPC retraining (naive), 2) sampling $\boldsymbol{x}_s$ by Importance Sampling and calculating $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ with GPC retraining (IS), 3) sampling $\boldsymbol{x}_s$ by Importance Sampling and calculating $p(y_s|\boldsymbol{x}_s, \boldsymbol{x}_*, y_*)$ with the Joint Distribution Integral (IS+JDI). The algorithms are implemented in Python 3.7 on a personal computer with Intel i5-10400 2.9 GHz CPU and 16G RAM. We set the initial datasets of size $n = 10$, 50 and 100, respectively. For all three competing algorithms, we sample 1000 $\boldsymbol{x}_s$'s, and calculate $U^{\text{S}}(\boldsymbol{x}_*)$. We benchmark them with a "ground truth" algorithm: calculating $U^{\text{S}}(\boldsymbol{x}_*)$ by the naive algorithm with *1e6* $\boldsymbol{x}_s$ samples. We perform this comparison for 100 $\boldsymbol{x}_*$'s. The average running time and absolute error rate are shown in Table 4.1. From the table, the absolute error rate of the naive algorithm with 1000 samples is much larger than the other two as expected since the other two algorithms use importance sampling with smaller estimation variance. For the running time, retraining GPC in the naive and IS algorithms is the main time-consuming component while IS+JDI spends most of the time in calculating the joint distribution integral for marginalization. As n increase, the running time of both naive and IS algorithms increases fast since GPC training has a complexity of $O(n^3)$, while the running time of IS+JDI does not change much because the joint distribution integral is calculated in constant time. Note that in all three algorithms, inferring the predictive distribution for each $\boldsymbol{x}_s$ also takes a considerable part of computations, especially when n is small. The results show that importance sampling does not impose much extra computation load in active learning. When n is large, the joint distribution integral is faster than retraining GPC with EP approximation.

Table 4.1: Running time in seconds (s) and estimation accuracy comparison.

	n = 10		n = 50		n = 100	
Algorithm	Time (s)	Error rate	Time (s)	Error rate	Time (s)	Error rate
naive	1.390	0.166	2.496	0.184	2.704	0.162
IS	1.364	0.012	2.465	0.032	2.685	0.024
IS+JDI	1.950	0.012	2.188	0.028	2.209	0.025

4.5.2 Query Synthesis with Synthetic Datasets

We now test the algorithms in the task of finding the optimal classifier of the unknown probabilistic model $p(y|\boldsymbol{x}, f)$ with f generated from the GP prior. In this set of experiments, the domain of f is $\mathcal{X} = [-4, 4]$. Each f is generated by first sampling 1000 function values from the GP prior with $\gamma^2 = 0.5$, $l^2 = 1$. f is then given by the resulting GP posterior mean. We generate a total of 200 f's following the procedure.

We perform all the competing active learning algorithms on these probabilistic models. MES, BALD, NR-MOCU-RO and NR-SMOCU-RO are all optimized by random optimization with $M_2 = 1000$. In NR-SMOCU-SGD, we first perform random optimization with $M_2 = 800$ and set the best point as the initial point for Adagrad so that NR-SMOCU-SGD has similar running time compared with NR-SMOCU-RO at their corresponding setups for fair comparison. Algorithm performance is measured in terms of the error regret defined as the OBC error at n-th iteration minus the optimal classifier error of the simulated ground truth. Fig.4.2 shows the average error regret with standard deviation bars in the logarithmic scale obtained by each algorithm across the 150 different probabilistic models. The results show that in the first few iterations, all three EER-based algorithms (NT-MOCU-RO, NT-SMOCU-RO, NT-SMOCU-SGD) outperform the competing algorithms. With more observations

included, the decrease of the error regret slows down for NT-MOCU-RO. This is because the MOCU-based acquisition function cannot take into account the long-term effect of a query, also discussed in [87]. The plot also shows that the best algorithm in this setting is NR-SMOCU-SGD as it utilizes the gradient information during optimization.

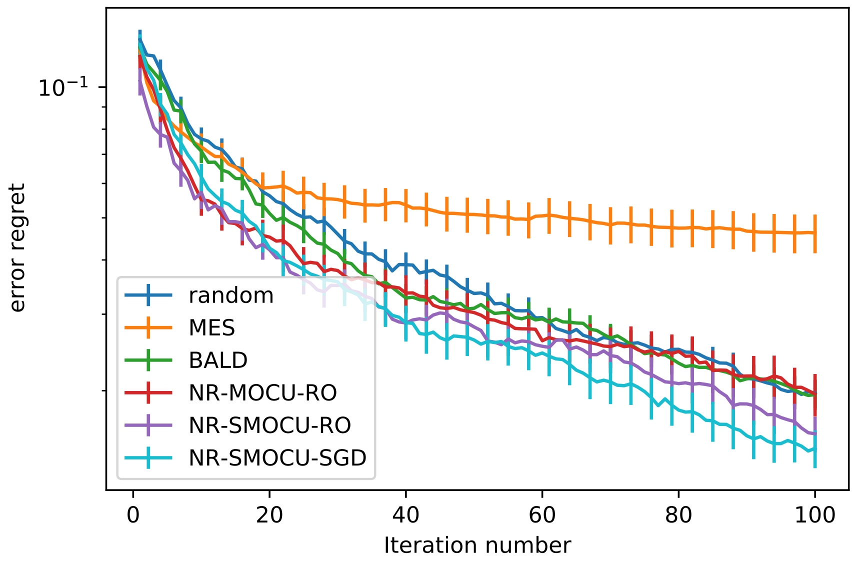


Figure 4.2: Algorithm performance comparison on 1-d GPC.

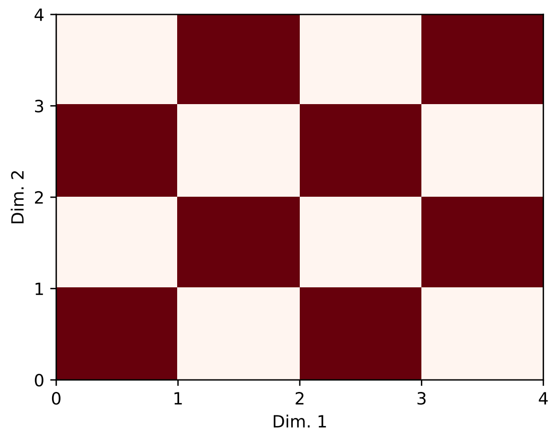
We further benchmark our proposed algorithms with a challenging synthetic dataset *checker-board* 4×4, similar as the one tested in [37], which emulates the setup of phase diagram identification in materials science. Fig. 4.3a illustrates the classification boundaries. To show the influence of the observation error on the performance of different algorithms, we further assume that there is a constant flip error rate on the observing labels, and we take different error rate equal to 0, 0.1 and 0.2. In the experiments, we initially draw 30 samples for labeling to estimate the hyperparameters $\{\gamma, l\}$, then we perform different algorithms to collect new data. We repeat the procedure for 150 runs and plot the average performance with standard deviations in Figs. 4.3b-d.

We can observe from these figures that MES performs bad, which is because MES tends to query the point close to the decision boundary, while this problem has multiple intertwined boundaries and MES cannot differentiate different boundaries. We also observe that as the error rate increases, the difference between NR-MOCU and NR-SMOCU also increases, that is because as the error rate increases, MOCU reduction tends to ignore the long-term effect of a query, leading to the degraded long-term performance. Among these algorithms, NR-SMOCU-SGD performs better than other proposed algorithms again by leveraging the gradient information in the optimization procedure.

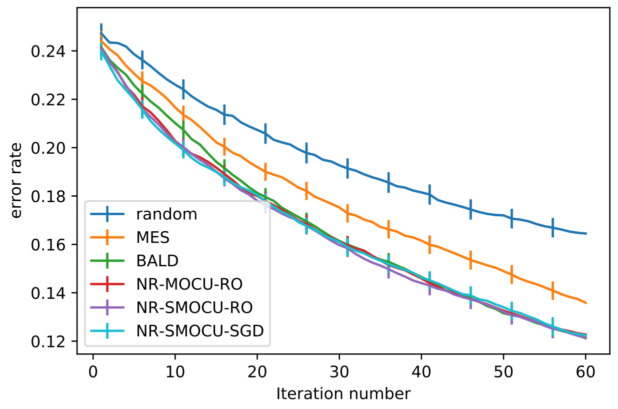
4.5.3 Pool-based Active Learning with Real-world Datasets

We also compare algorithms on the UCI datasets [22] for pool-based active learning. NR-SMOCU-SGD is not included as it is designed to search the continuous space. For each dataset, we split it into training and testing datasets. We take the training dataset as the sampling pool for active learning, initially we randomly choose two samples from each class for labelling, and use them to estimate the GPC hyperparameters. Then we apply the competing active learning algorithms to sequentially select the query from the training dataset and estimate the OBC error with the testing dataset after each iteration. Details of the tested UCI datasets are provided in Table 4.2.

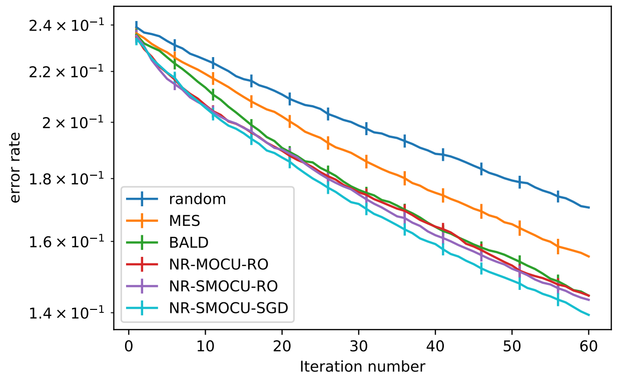
Similarly, we repeat the active learning procedures for 100 runs and plot the average OBC error with standard deviation values for each algorithm for performance comparison shown in Fig. 4.4. Overall, NR-MOCU-RO and NR-SMOCU-RO are better than MES and BALD, which validates that our algorithms achieve better sample or in particular label efficiency. Note that in the *Wine* and



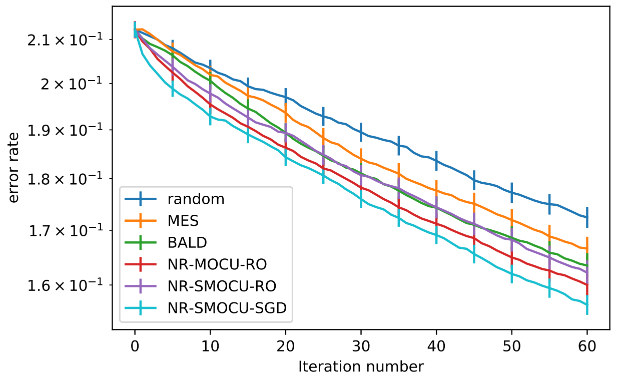
(a) *Checkerboard* 4×4 problem



(b) Flip error rate = 0



(c) Flip error rate = 0.1



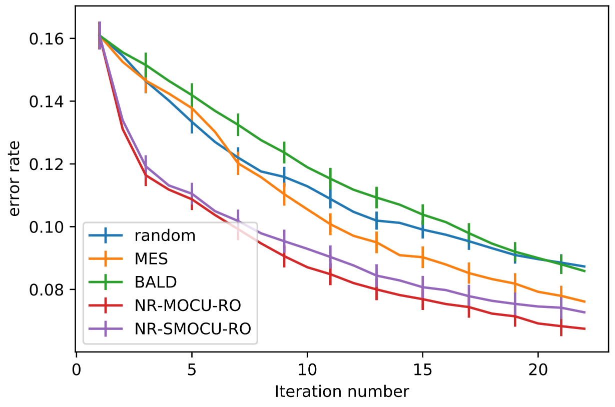
(d) Flip error rate = 0.2

Figure 4.3: The expected OBC error regret comparison on *checkerboard* 4×4 problem.

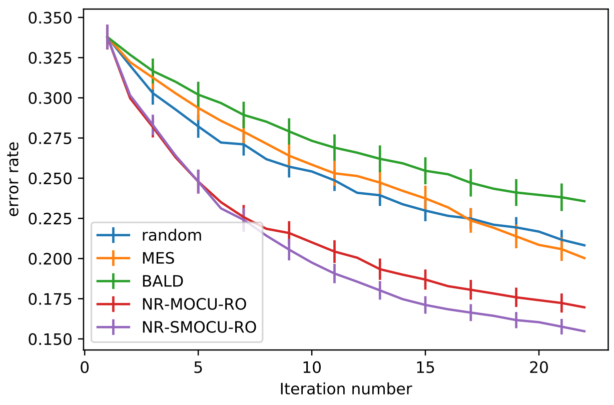
Table 4.2: Details of the tested UCI datasets.

Dataset	n_{train}	n_{test}	d	Dataset description
WDBC	284	285	30	Wisconsin diagnostic breast cancer
Ionosphere	175	176	34	Radar returns from the ionospher
Vehicle	208	208	18	Features extracted from silhouettes image
Wine [10]	65	65	13	Wine quality

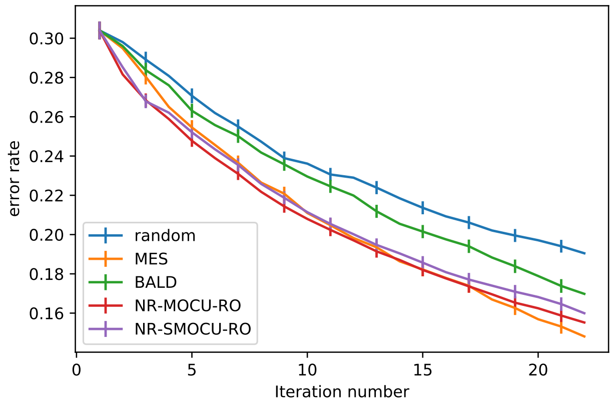
Vehicle datasets, MES performs closely to our algorithms. When checking the converging GPC models, their classification boundaries are relatively simple on these two datasets with relatively low-dimensional feature spaces. The *WDBC* and *Ionosphere* data are in higher dimensions, for which our proposed algorithms perform significantly better.



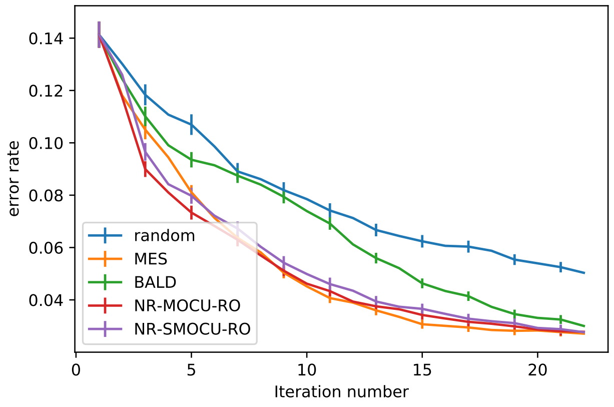
(a) WDBC



(b) Ionosphere



(c) Vehicle



(d) Wine

Figure 4.4: Classification error rate comparison on UCI datasets.

5. CONCLUSIONS & FUTURE RESEARCH

In this dissertation, we focus on Bayesian learning and experimental design in an objective-oriented uncertainty quantification framework based on the concept of mean objective cost of uncertainty (MOCU) [84]. Several MOCU-based algorithms have been developed for Bayesian experimental design or active learning for filtering and classification problems when studying corresponding uncertain complex systems.

In Chapter 2, we have developed MOCU-based experimental design for optimal filtering of data from the systems described by SDEs under uncertainty. We have propagated the prior distribution through the SDE so that the same distribution governs the uncertainty of signal processes. In the former classification study [93], no assumption was made on the distribution of the parameter uncertainty class and it was then assumed that a normal-inverse-Wishart distribution governed the mean and covariance matrix of the uncertain Gaussian features constructed by sampling the signal trajectories. This was convenient because it allowed direct application of the theory of optimal Bayesian classification for Gaussian features [13], thereby resulting in a closed-form solution for the optimal Bayesian classifier. The convenience of the previous assumption comes at a significant price: if there is physical knowledge regarding the distribution of the uncertain parameters, it has been ignored. Thus, we believe that uncertainty propagation, as used in the present dissertation is more sound from a physical perspective, even if it leaves us with no hope of a convenient closed-form solution. Although computational complexity did not impede us in Chapter 2, it can become a problem when there is high dimensionality, especially when the uncertainty class is large. Model reduction can be used to reduce the computations. For instance, a regulatory network model can be compressed by eliminating or combining nodes [16]. Model reduction remains an important research topic from a practical perspective, and to be effective, such reduction should be made in a way that maintains the structure most relevant to the objective – which makes it application dependent.

In Chapter 3, we have investigated why ELR- or MOCU-based methods may perform poorly in the long run—both theoretically and empirically, though they are optimal for active learning when considering single queries. Based on the analysis, we proposed novel active learning strategies for classification based on weighted MOCU and Soft MOCU. Our weighted MOCU directly targets at decreasing the classification error and ignores uncertainty irrelevant to the classification performance. More critically, it can capture continuous change in objective-relevant uncertainty. Hence, our new active learning can be efficient both at the beginning and in the long run with the guarantee of converging to the optimal classifier. Our new Soft-MOCU-based active learning is efficient for the initial iterations as it approximates the original MOCU-based active learning scheme. A critical feature of Soft MOCU is that its strict concavity enables the resulting acquisition function to capture small model uncertainty reduction and thus guarantees the OBC to converge to the true optimal classifier even when the myopic one-step-look-ahead queries may not provide significant changes to the model posterior $\pi(\theta|\boldsymbol{x}, y)$. Consequently, our proposed active learning methods can be efficient both at the beginning as well as in the long run. In addition to the theoretical guarantee, our empirical results also demonstrated the superior performance of our weighted-MOCU and Soft-MOCU-based methods. Finally, as analyzed and observed in our experiments, Soft MOCU with larger k performs better at the beginning as it closely approximates MOCU with local optimality whereas Soft MOCU with smaller k performs better in the long run. Adaptively updating the value of k during the active learning procedure is an interesting research direction.

In Chapter 4, we have proposed efficient MOCU-based active learning algorithms with GPC, which estimate the MOCU/SMOCU reduction by querying instances based on the joint distribution of label pairs. We have derived the joint distribution as a one-dimensional integral with constant computational cost to calculate the predictive posterior based on it. Together with importance sampling, the acquisition function can be estimated efficiently by 1-d integral of the joint distribution without incrementally retraining GPC with EP approximation, which has a computation complexity of $O(n^3)$. Without the need for EP approximation, we can further derive the chain rule to calculate the gradient of the SMOCU reduction, which provides us an efficient query synthesis active learning algorithm. Our experiments have demonstrated both the accuracy and the running speed of our algorithms. Comparing with benchmark algorithms on both synthetic and real-world datasets, our algorithm have shown the sampling efficiency in active learning with GPC models.

In summary, throughout this dissertation, we have studied different scenarios of Bayesian learning and Bayesian experimental design focusing on the operational objective with MOCU. In the regression problems with SDEs, we study the experimental design problem that sequentially selects the experiments minimizing the mean squared error of the signal estimates, and we manage to connect the system parameter uncertainties with MOCU. In the classification problems, the operational objective is minimizing the classification error. To achieve that we study the active learning based on MOCU and its extensions to choose the query that helps maximally reduce the classification error. We solve the myopic performance of MOCU-based active learning methods by imposing concavity to the MOCU approximations and develop efficient algorithms for these methods with optimal Bayesian classifiers, including GPC models.

For more general machine learning models, including deep neural networks, the joint predictive distribution does not have closed-form expressions as discussed in this dissertation. In order to further extend the developed algorithms to more general cases, for both low- and high-dimensional problems, we need to explore general methods in evaluating the joint predictive distributions, either by sampling methods or by approximate inference. The theoretical analysis and empirical benchmarking is necessary to gain better understanding of the performance in these setups, which we leave for future research.

REFERENCES

[1] L Arnold and A V Balakrishnan. Stochastic Differential Equations Theory and Applications. *Bull. Amer. Math. Soc*, 81:837–840, 1975. ISSN 1088-9485.

[2] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89, 2009.

[3] Samuel Bandara, Johannes P Schlöder, Roland Eils, Hans Georg Bock, and Tobias Meyer. Optimal experimental design for parameter estimation of a cell signaling model. *PLoS Comput Biol*, 5(11):e1000558, 2009.

[4] Donald A Berry. Bayesian clinical trials. *Nature reviews Drug discovery*, 5(1):27–36, 2006.

[5] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] Carlos A Braumann. Itô versus Stratonovich calculus in random population growth. *Mathematical biosciences*, 206(1):81–107, 2007.

[7] Ariana Broumand, Mohammad Shahrokh Esfahani, Byung-Jun Yoon, and Edward R Dougherty. Discrete optimal bayesian classification with error-conditioned sequential sampling. *Pattern Recognition*, 48(11):3766–3782, 2015.

[8] Kwok-Wai Cheung, James T Kwok, Martin H Law, and Kwok-Ching Tsui. Mining customer product ratings for personalized marketing. *Decision Support Systems*, 35(2):231–243, 2003.

[9] Darrell Coles and Michael Prange. Toward efficient computation of the expected relative entropy for nonlinear experimental design. *Inverse problems*, 28(5):055019, 2012.

[10] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009.

[11] Nguyen Viet Cuong, Wee Sun Lee, Nan Ye, Kian Ming A Chai, and Hai Leong Chieu. Active learning for probabilistic hypotheses using the maximum gibbs error criterion. In *Advances in Neural Information Processing Systems*, pages 1457–1465, 2013.

[12] Lori A Dalton and Edward R Dougherty. Intrinsically optimal Bayesian robust filtering. *IEEE Transactions on Signal Processing*, 62(3):657–670, 2013.

[13] Lori A Dalton and Edward R Dougherty. Optimal classifiers with minimum expected error within a Bayesian framework—Part I: Discrete and Gaussian models. *Pattern Recognition*, 46 (5):1301–1314, 2013.

[14] R Dehghannasiri, X Qian, and ER Dougherty. Bayesian robust Kalman smoother in the presence of unknown noise statistics. *EURASIP J. Adv. Signal Process.*, 55:1982–1996, 2018.

[15] Roozbeh Dehghannasiri, Byung-Jun Yoon, and Edward R Dougherty. Optimal experimental design for gene regulatory networks in the presence of uncertainty. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 12(4):938–950, 2015.

[16] Roozbeh Dehghannasiri, Byung-Jun Yoon, and Edward R Dougherty. Efficient experimental design for uncertainty reduction in gene regulatory networks. In *BMC bioinformatics*, volume 16, pages 1–18. Springer, 2015.

[17] Roozbeh Dehghannasiri, Mohammad Shahrokh Esfahani, and Edward R Dougherty. Intrinsically Bayesian robust Kalman filter: An innovation process approach. *IEEE Transactions on Signal Processing*, 65(10):2531–2546, 2017.

[18] Roozbeh Dehghannasiri, Xiaoning Qian, and Edward R Dougherty. Optimal experimental design in the context of canonical expansions. *IET Signal Processing*, 11(8):942–951, 2017. ISSN 1751-9683.

[19] Roozbeh Dehghannasiri, Dezhen Xue, Prasanna V Balachandran, Mohammadmahdi R Yousefi, Lori A Dalton, Turab Lookman, and Edward R Dougherty. Optimal experimental design for materials discovery. *Computational Materials Science*, 129:311–322, 2017.

[20] Edward R Dougherty. *Optimal Signal Processing Under Uncertainty*. SPIE Press, 2018.

[21] Christopher C Drovandi, James M McGree, and Anthony N Pettitt. Sequential monte carlo for bayesian sequentially designed experiments for discrete data. *Computational Statistics & Data Analysis*, 57(1):320–335, 2013.

[22] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://`

archive.ics.uci.edu/ml.

[23] Yonina C Eldar. Minimax estimation of deterministic parameters in linear models with a random model matrix. *IEEE Transactions on Signal Processing*, 54(2):601–612, 2006.

[24] Yonina C Eldar and Neri Merhav. Minimax mse-ratio estimation with signal covariance uncertainties. *IEEE Transactions on Signal Processing*, 53(4):1335–1347, 2005.

[25] Vladimir Nikolaevich Fomin. *Optimal filtering: volume I: filtering of stochastic processes*, volume 457. Springer Science & Business Media, 2012.

[26] Peter Frazier, Warren Powell, and Savas Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613, 2009.

[27] Peter I Frazier and Jialei Wang. Bayesian optimization for materials design. In *Information Science for Materials Discovery and Design*, pages 45–75. Springer, 2016.

[28] Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997.

[29] Weijie Fu, Meng Wang, Shijie Hao, and Xindong Wu. Scalable active learning by approximated error reduction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1396–1405, 2018.

[30] Federico Galvanin, Sandro Macchietto, and Fabrizio Bezzo. Model-based design of parallel experiments. *Industrial & engineering chemistry research*, 46(3):871–882, 2007.

[31] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[32] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, pages 766–774, 2010.

[33] Alison Gray, David Greenhalgh, Liangjian Hu, Xuerong Mao, and Jiafeng Pan. A stochastic differential equation SIS epidemic model. *SIAM Journal on Applied Mathematics*, 71(3): 876–902, 2011.

[34] Artyom M Grigoryan and Edward R Dougherty. Design and analysis of robust binary filters in

the context of a prior distribution for the states of nature. *Journal of Mathematical Imaging and Vision*, 11(3):239–254, 1999.

[35] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *arXiv preprint arXiv:1406.2541*, 2014.

[36] Trong Nghia Hoang, Bryan Kian Hsiang Low, Patrick Jaillet, and Mohan Kankanhalli. Non-myopic ϵ-bayes-optimal active learning of gaussian processes. 2014.

[37] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[38] H Tolga Kahraman, Seref Sagiroglu, and Ilhami Colak. The development of intuitive knowledge classifier and the modeling of domain dependent data. *Knowledge-Based Systems*, 37: 283–295, 2013.

[39] Thomas Kailath. Lectures on Wiener and Kalman filtering. In *Lectures on Wiener and Kalman Filtering*, pages 1–143. Springer, 1981.

[40] Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *IJCAI*, volume 7, pages 877–882, 2007.

[41] Saleem A Kassam and Tong Leong Lim. Robust Wiener filters. *Journal of the Franklin Institute*, 304(4-5):171–185, 1977. ISSN 0016-0032.

[42] Steven M Kay. *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[43] Heinrich Kirchauer, Franz Hlawatsch, and Werner Kozek. Time-frequency formulation and design of nonstationary Wiener filters. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1549–1552. IEEE, 1995.

[44] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pages 7024–7035, 2019.

[45] Steven Kleinegesse and Michael U Gutmann. Efficient bayesian experimental design for implicit models. In *The 22nd International Conference on Artificial Intelligence and Statistics*,

pages 476–485. PMLR, 2019.

[46] Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

[47] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th international conference on Machine learning*, pages 449–456, 2007.

[48] Malte Kuss, Carl Edward Rasmussen, and Ralf Herbrich. Assessing approximate inference for binary Gaussian process classification. *Journal of machine learning research*, 6(10), 2005.

[49] V P Kuznetsov. Stable detection when the signal and spectrum of normal noise are inaccurately known, Telecomm. Radio Eng. *(English Transl.)*, 3031:58–64, 1976.

[50] Oh Kyu Kwon, Wook Hyun Kwon, and Kyu Seung Lee. FIR filters and recursive forms for discrete-time state-space models. *Automatica*, 25(5):715–728, 1989.

[51] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[52] Dennis Victor Lindley. *Bayesian statistics: A review*. SIAM, 1972.

[53] Paul Malliavin and Anton Thalmaier. *Stochastic calculus of variations in mathematical finance*. Springer Science & Business Media, 2006.

[54] Raman Mehra. On the identification of variances and adaptive Kalman filtering. *IEEE Transactions on automatic control*, 15(2):175–184, 1970.

[55] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[56] Daniel N Mohsenizadeh, Roozbeh Dehghannasiri, and Edward R Dougherty. Optimal objective-based experimental design for uncertain dynamical gene networks with experimental error. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(1): 218–230, 2016.

[57] Masakatu Morii, Masao Kasahara, and Douglas L. Whiting. Efficient bit-serial multiplication and the discrete-time Wiener-Hopf equation over finite fields. *IEEE Transactions on*

Information Theory, 35(6):1177–1183, 1989.

[58] Peter Müller, Don A Berry, Andy P Grieve, Michael Smith, and Michael Krams. Simulation-based sequential bayesian design. *Journal of statistical planning and inference*, 137(10): 3140–3150, 2007.

[59] Stephen Mussmann and Percy Liang. On the relationship between data efficiency and error for uncertainty sampling. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3674–3682, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[60] GE Poirier, WP Fitts, and JM White. Two-dimensional phase diagram of decanethiol on au (111). *Langmuir*, 17(4):1176–1183, 2001.

[61] H Poor. On robust Wiener filtering. *IEEE Transactions on Automatic Control*, 25(3):531–536, 1980. ISSN 0018-9286.

[62] Edward O Pyzer-Knapp. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2–1, 2018.

[63] Xiaoning Qian and Edward R Dougherty. Bayesian regression with network prior: Optimal Bayesian filtering perspective. *IEEE Transactions on Signal Processing*, 64(23):6243–6253, 2016.

[64] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[65] N Roy and A McCallum. Toward optimal active learning through sampling estimation of error reduction. int. conf. on machine learning, 2001.

[66] Elizabeth G Ryan, Christopher C Drovandi, and Anthony N Pettitt. Fully bayesian experimental design for pharmacokinetic studies. *Entropy*, 17(3):1063–1089, 2015.

[67] Alireza Sani and Azadeh Vosoughi. On distributed linear estimation with observation model uncertainties. *IEEE Transactions on Signal Processing*, 66(12):3212–3227, 2018.

[68] Simo Sarkka and Aapo Nummenmaa. Recursive noise adaptive Kalman filtering by variational

Bayesian approximations. *IEEE Transactions on Automatic control*, 54(3):596–600, 2009.

[69] Ali H Sayed. A framework for state-space estimation with uncertain models. *IEEE Transactions on Automatic Control*, 46(7):998–1013, 2001.

[70] Raphael Schumann and Ines Rehbein. Active learning via membership query synthesis for semi-supervised sentence classification. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 472–481, 2019.

[71] Paola Sebastiani and Henry P Wynn. Maximum entropy sampling and optimal bayesian experimental design. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(1):145–157, 2000.

[72] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Mustererkennung 2000*, pages 27–34. Springer, 2000.

[73] Burr Settles. Active learning literature survey. Computer Science Technical Report 1648, University of Wisconsin-Madison. 2009.

[74] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[75] Leon Shargel, BC Andrew, and Susanna Wu-Pong. *Applied biopharmaceutics and pharmacokinetics*. Appleton & Lange Stamford, 1999.

[76] Yuriy S Shmaliy. Linear optimal FIR estimation of discrete time-invariant state-space models. *IEEE Transactions on Signal Processing*, 58(6):3086–3096, 2010.

[77] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[78] Yasuo Sugiyama. An algorithm for solving discrete-time Wiener-hopf equations based upon Euclid's algorithm. *IEEE Transactions on Information Theory*, 32(3):394–409, 1986.

[79] Gabriel Terejanu, Rochan R Upadhyay, and Kenji Miki. Bayesian experimental design for the

active nitridation of graphite by atomic nitrogen. *Experimental Thermal and Fluid Science*, 36:178–193, 2012.

[80] Wim J van der Linden and Hao Ren. Optimal bayesian adaptive design for test-item calibration. *Psychometrika*, 80(2):263–288, 2015.

[81] K Vastola and H Poor. Robust Wiener-Kolmogorov theory. *IEEE Transactions on Information Theory*, 30(2):316–327, 1984. ISSN 0018-9448.

[82] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[83] Fang-Rong Yan, Ping Zhang, Jun-Lin Liu, Yu-Xi Tao, Xiao Lin, Tao Lu, and Jin-Guan Lin. Parameter estimation of population pharmacokinetic models with stochastic differential equations: implementation of an estimation algorithm. *Journal of Probability and Statistics*, 2014, 2014.

[84] Byung-Jun Yoon, Xiaoning Qian, and Edward R Dougherty. Quantifying the objective cost of uncertainty in complex dynamical systems. *IEEE Transactions on Signal Processing*, 61(9): 2256–2266, 2013.

[85] Byung-Jun Yoon, Xiaoning Qian, and Edward R. Dougherty. Quantifying the multi-objective cost of uncertainty. *arXiv preprint arXiv:2010.04653*, 2020.

[86] Guang Zhao, Xiaoning Qian, Byung-Jun Yoon, Francis J Alexander, and Edward R Dougherty. Model-based robust filtering and experimental design for stochastic differential equation systems. *IEEE Transactions on Signal Processing*, 68:3849–3859, 2020.

[87] Guang Zhao, E Dougherty, Byung-Jun Yoon, F Alexander, and Xiaoning Qian. Uncertainty-aware active learning for optimal Bayesian classifier. In *Proc. 9th Int. Conf. Learn. Repre-sent.(ICLR)*, pages 1–20, 2021.

[88] Guang Zhao, Edward Dougherty, Byung-Jun Yoon, Francis J Alexander, and Xiaoning Qian. Bayesian active learning by soft mean objective cost of uncertainty. In *International Confer-ence on Artificial Intelligence and Statistics*, pages 3970–3978. PMLR, 2021.

[89] Tong Zhou. Sensitivity penalization based robust state estimation for uncertain linear systems.

IEEE Transactions on Automatic Control, 55(4):1018–1024, 2010.

[90] Tong Zhou. Robust recursive state estimation with random measurement droppings. *IEEE Transactions on Automatic Control*, 61(1):156–171, 2015.

[91] Jia-Jie Zhu and José Bento. Generative adversarial active learning. *arXiv preprint arXiv:1702.07956*, 2017.

[92] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3, 2003.

[93] Amin Zollanvari and Edward R Dougherty. Incorporating prior knowledge induced from stochastic differential equations in the classification of stochastic observations. *EURASIP Journal on Bioinformatics and Systems Biology*, 2016(1):2, 2016. ISSN 1687-4153.

[94] Mattia Zorzi. On the robustness of the Bayes and Wiener estimators under model uncertainty. *Automatica*, 83:133–140, 2017.

APPENDIX A

APPENDIX FOR WEIGHTED-MOCU BASED METHOD

In this appendix section, we provide the pseudo-code of active learning algorithms, as well as more detailed descriptions of our experiments, additional results, and discussions on the weighted MOCU for multi-class classification.

A.1 Weighted-MOCU-based Active Learning & Computational Complexity

The pseudo-code of our weighted-MOCU based active learning is provided in Algorithm 4, with the following computational complexity analysis.

We now estimate the computational complexity of Algorithm 4 for the discrete feature and parameter spaces. Assume that the size of the discrete feature space is $N_x = |\mathcal{X}|$, the size of the uncertainty set of classifiers is $N_\theta = |\Theta|$. For active learning, there are T iterations of queries as the total budget. We study the total complexity of the active learning algorithm. In the WMOCU function, line 6 is called for $O(N_x N_\theta M)$ times. In ACQUISITIONFUN, WMOCU is called for constant times. Finally, in the main procedure, in each iteration, ACQUISITIONFUN is called for each $\boldsymbol{x}$. Hence, the total complexity of Weighted MOCU-based active learning is $O(TN_x^2 N_\theta)$.

A.2 Details of the Binary Classification Example in Figure 3.1

In the binary classification problem, $\Theta = \{\theta_1, \theta_2\}$, $\mathcal{X} = \{x_1, x_2\}$. The probabilistic model setting for the two candidates is symmetric:

$$p(y_1|x_1, \theta_1) = (0.6, 0.4), p(y_1|x_1, \theta_2) = (0.3, 0.7)$$
$$p(y_2|x_2, \theta_1) = (0.7, 0.3), p(y_2|x_2, \theta_2) = (0.4, 0.6)$$

There are three intervals corresponding to the linear function pieces of MOCU in Fig. 3.1: $[0, 0.33]$, $(0.33, 0.67]$ and $(0.67, 1]$. In the three intervals, $\psi_{\pi(\theta)}(x_1)$, the OBC predictions of x_1 are 1, 1 and 0, respectively; $\psi_{\pi(\theta)}(x_2)$, the OBC predictions of x_2 are 1, 0, and 0, respectively.

In Fig. 3.1 we set the prior $\tilde{\pi}(\theta_1) = 0.15$, then based on the Bayes's rule we can obtain the posterior with the observations of (x_1, y_1). Based on the observation result of y_1, the posteriors are $\tilde{\pi}(\theta_1|x_1, y_1 = 0) = 0.2609$ and $\tilde{\pi}(\theta_1|x_1, y_1 = 1) = 0.0916$, both of which fall into the first linear piece of MOCU.

Algorithm 4 Weighted-MOCU based active learning

```
1: function MAINPROCEDURE( )
2:     Set a discrete candidate set $\mathcal{X}$, the probability array $p_x$, and iteration number T
3:     Set the discrete parameter set $\Theta$ and the corresponding probability array $\pi_\theta$
4:     Initialize the data set $D = \emptyset$
5:     $\pi_{\theta|D} = \pi_\theta$
6:     for $t = 1$ to $T$ do
7:         for $x$ in $\mathcal{X}$ do
8:             Store ACQUISITIONFUN($x, \pi_{\theta|D}$) to the array $U_\mathcal{X}$
9:         end for
10:        Optimize $U_\mathcal{X}$ and find the maximum point $x^*$
11:        Obtain the label $y^*$ corresponds to $x^*$ and update $D = D \cup \{x^*, y^*\}$
12:        for $\theta$ in $\Theta$ do
13:            Update $\pi_{\theta|D} \propto \pi_{\theta|D} \cdot p(y^*|x^*, \theta)$
14:        end for
15:        $\mathcal{X} = \mathcal{X}/\{x^*\}$
16:    end for
17: end function

18: function ACQUISITIONFUN($x, \pi_{\theta|D}$)
19:     $wmocu\_current$ =WMOCU($\pi_{\theta|D}$)
20:     $wmocu\_next = 0$
21:     for $y$ in $\{0, 1\}$ do
22:         for $\theta$ in $\Theta$ do
23:             Generate array $p(\theta, y|D, x) = \pi_{\theta|D} \cdot p(y|x, \theta)$
24:         end for
25:         $p(y|D, x)$ = $\sum_\theta p(\theta, y|D, x)$
26:         $\pi_{\theta|D,x,y} = p(\theta, y|D, x)/p(y|D, x)$
27:         $wmocu\_next = wmocu\_next + p(y|D, x) \cdot$ WMOCU($\pi_{\theta|D,x,y}$)
28:     end for
29:     return $wmocu\_current - wmocu\_next$
30: end function
```

```
function WMOCU($\pi_{\theta|D}$)
    $wmocu = 0$
    for $x'$ in $\mathcal{X}$ do
        $bayesian\_error = 0$
        for $\theta$ in $\Theta$ do
            $bayesian\_error = bayesian\_error + \pi_{\theta|D} \cdot (1 - \max_{y'} p(y'|x', \theta))$
        end for
        for $y'$ in $\{0, 1\}$ do
            $p(y'|D, x') = \sum_\theta \pi_{\theta|D} \cdot p(y'|x', \theta)$
        end for
        $obc\_error = 1 - \max_{y'} p(y'|D, x')$
        $K = obc\_error - bayesian\_error$
        $wmocu = wmocu + p(x') \cdot [(1 - K)K]$
    end for
    return $wmocu$
end function
```

A.3 Multi-class Classification

Although we have shown in chapter 3 that our weighted-MOCU can achieve good empirical performance of converging to OBC with the simulated multi-class classification experiment, active learning for multi-class classification problems can be complicated. The weighting function (3.10) adopted in chapter 3 may not have the same theoretical convergence guarantee to the optimal classifier if applied to multi-class classification problems. Here we just show a counter example, for which Lemma 4 does not hold if using the same weighting function.

Assume a three-class classification problem $y \in \{0, 1, 2\}$. The candidate pool only has one candidate $\mathcal{X} = \{x\}$ and the parameter set $\Theta = \{\theta_1, \theta_2, \theta_3\}$. In addition we set the probabilistic model $p(y|x, \theta)$ and the prior $\pi(\theta)$ as shown in Tables S1 and S2, and calculate the posterior and posterior predictive probabilities. In the tables, y_o denotes the one-step-look-ahead observation corresponding to x, and x is omitted for simplicity. Without loss generality, we just set the weighted MOCU parameter $c = 1$.

Here two properties in the setting are worth mentioning:

1. $\pi(\theta_1)$ is close to 1 and as a result $\forall y_o \in \{0, 1, 2\}$, we have $\max_y p(y) = \max_y p(y|y_o) =$

Table A.1: The probabilities of $p(y|x,\theta)$ and $p(y|x,y_o)$.

	$p(y\|\theta_1)$	$p(y\|\theta_2)$	$p(y\|\theta_3)$	$p(y)$	$p(y\|y_o=0)$	$p(y\|y_o=1)$	$p(y\|y_o=2)$
$y=0$	0.4	0.4	0.4	0.4	0.4	0.4	0.4
$y=1$	0.3	0.1	0.5	0.3	0.3	0.327	0.273
$y=2$	0.3	0.5	0.1	0.3	0.3	0.273	0.327

Table A.2: The prior and posterior of $\pi(\theta)$.

	$\pi(\theta)$	$\pi(\theta\|y_o=0)$	$\pi(\theta\|y_o=1)$	$\pi(\theta\|y_o=2)$
$\theta=\theta_1$	0.8	0.8	0.8	0.8
$\theta=\theta_2$	0.1	0.1	0.17	0.03
$\theta=\theta_3$	0.1	0.1	0.03	0.17

$\max_y p(y|\theta_1) = 0.4$;

2. $p(y|\theta_2)$ and $p(y|\theta_3)$ are symmetric and $\pi(\theta_2) = \pi(\theta_3)$, as a result $\forall y_o \in \{0,1,2\}$, $\pi(\theta_1) = \pi(\theta_1|y_o) = 0.8$ and therefore $\mathbb{E}_{\pi(\theta)}[\max_{y'} p(y|\theta)] = \mathbb{E}_{\pi(\theta|y_o)}[\max_{y'} p(y|\theta)] = 0.8\times0.4+0.2\times0.5 = 0.42$.

Recall that the K function and weighted MOCU are:

$$K(\pi(\theta)) = \mathbb{E}_{\pi(\theta)}[\max_{y'} p(y|\theta)] - \max_{y'} p(y), \tag{A.1}$$

$$\mathcal{M}^w(\pi(\theta)) = [1 - K(\pi(\theta))] \cdot K(\pi(\theta)). \tag{A.2}$$

Therefore, we have $\forall y_o \in \{0,1,2\}$, $K(\pi(\theta)) = K(\pi(\theta|y_o)) = 0.02$, which means $\mathcal{M}^w(\pi(\theta)) = \mathcal{M}^w(\pi(\theta|y_o)) > 0$. On the other hand,

$$U^w(\pi(\theta)) = \mathcal{M}^w(\pi(\theta)) - \mathbb{E}_{p(y_o)}[\mathcal{M}^w(\pi(\theta|y_o))] = 0, \tag{A.3}$$

which means that the algorithm may get stuck. Here we just give an extreme case where only one candidate is in the search pool, but it is straightforward to build a more practical example based on what we have shown here.

We can see from the example that, unlike in the cases of binary classification problems, the weighting function $1 - cK$ may remain unchanged for a single observation in multi-class problems. Because of this, the weighted-MOCU algorithm may get stuck. Since OBC prediction is the maximum of the predictive distribution $p(y|x)$, the weight function is introduced to capture the changes of $p(y|x)$, as that indicates the potential shift of OBC prediction in the long run. K is a function of $\max_y p(y|x)$, in binary case, $\max_y p(y|x)$ must change as $p(y|x)$ changes. However, in multi-class problems, the probability of the optimal label $\max_y p(y|x)$ may remain unchanged, when the probability of other labels change, just like in the example above where $\max_y p(y) = \max_y p(y|y_o = 1)$. In the next section, we propose a weighting function that can capture the change of any element in $p(y|x)$.

A.4 Another Weighted MOCU Scheme for Multi-class Classification

To extend the weighted MOCU scheme suit for the multi-class problem, we propose a weight function that can capture the change of $p(y|x)$. The weighting function is defined as the softmax of $p(y|x)$:

$$w(\pi(\theta), x', \theta) = \frac{\exp(\max_y p(y|x))}{\sum_i \exp(p(y_i|x))}, \tag{A.4}$$

where $p(y|x)$ is the posterior predictive distribution at the current active learning iteration. We compare this Weighted MOCU with other active learning algorithms empirically on the synthetic three-class classification problem and the performance comparison is shown in Fig. A.1. This new Weighted MOCU (Weighted MOCU2) performs slightly better than other algorithms on this multi-class classification problem.

A.5 Additional WMOCU Synthetic Experiments

We run the same synthetic experiment of Fig. 3.3 with a different prior setting: $w_1 \sim \mathcal{U}(0.3, 0.8)$, $w_2 \sim \mathcal{U}(-0.02, 0.02)$ and $b \sim \mathcal{U}(-0.25, 0.25)$, and the results is shown in Fig. A.2. The performance shows that only our Weighted MOCU method performs better than the random benchmark.

Here we benchmark different active learning strategies for OBC with another synthetic example.

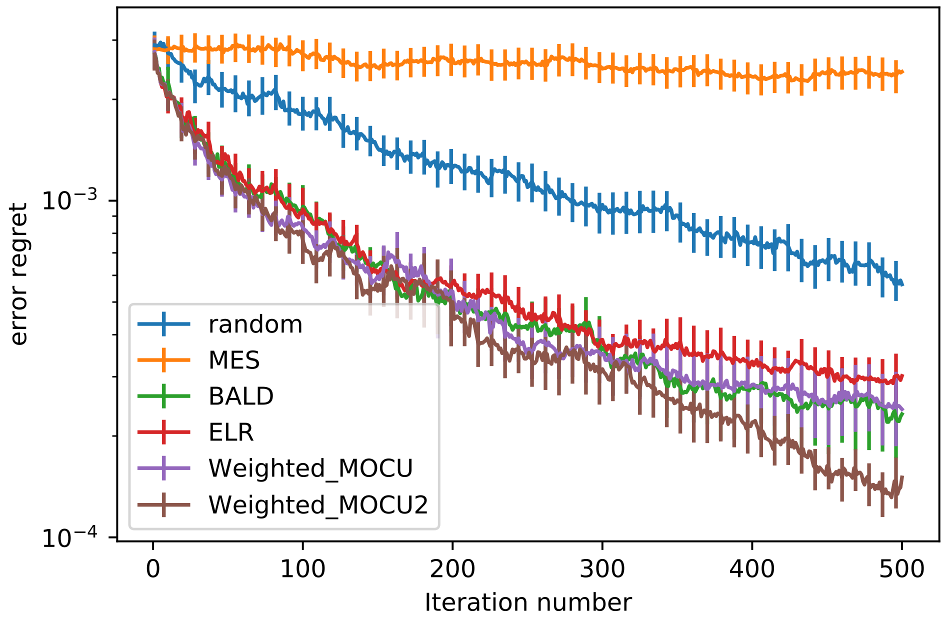


Figure A.1: The expected OBC error regret comparison between different active learning algorithms for the three-class classification problem.

Assume the classification problem with two dimensional input features $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ and binary class labels $y \in \{0, 1\}$. The computational model is derived by a decision boundary in a quadratic form: $x_2 = ax_1^2 + bx_1 + c$, i.e. $p(y = 1|\boldsymbol{x}, a, b, c) = \mathbb{1}(x_2 > ax_1^2 + bx_1 + c)$. The parameter vector $\theta = (a, b, c) \in \mathbb{R}^3$ is uncertain and the true model is characterized by a true parameter θ^*. Unlike Monte Carlo sampling in chapter 3, here we consider a discrete grid setting for both input space and parameter space with discretization for each variable as follows:

1. x_1 ranges in $[-0.5, 0.5]$ with increment 0.05
2. x_2 ranges in $[0, 2]$ with increment 0.1.
3. a ranges in [-4.3, -3.8] with increment 0.05,
4. b ranges in [-0.25, 0.25] with increment 0.05,
5. c ranges in [1, 2] with increment 0.05.

For now, we simply assume that the distributions over the feature space and parameter space are

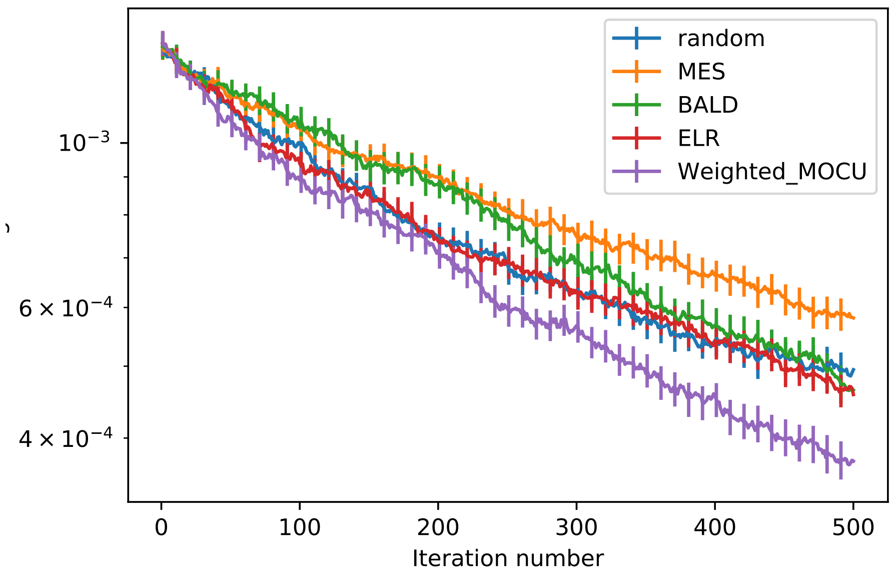


Figure A.2: The expected OBC error regret comparison between different active learning algorithms on binary classification.

all uniform to illustrate the effectiveness of MOCU-based active learning. With prior knowledge of the system of interest, knowledge-driven prior should be incorporated. Following the weighted-MOCU based active learning algorithm in Algorithm 1, we can sequentially query the true system and reduce the model uncertainty in a way that maximally reduces the classification error of the corresponding OBC.

Now we assume that when querying the system, the class label is given with a heterogeneous random flipping error with the error probability being a function of x_1: $p(y = 1|z = 0) = p(y = 0|z = 1) = 0.3 \times (1 - 4x_1^2) + 0.1$. Therefore, when $x_1 = 0$, the flipping error is 0.4; and when $x_1 = \pm 0.5$, the flipping error is 0.1. We have implemented the same methods as in chapter 3 with 50 iterations and 100 runs, The active learning results are illustrated in Fig. A.3. As we can see, in this figure, MES does not perform well as it cannot differentiate between model uncertainty and observation error. ELR performs similarly to BALD and our weighted-MOCU based method at the beginning, but then it gets stuck before finding the true boundary. BALD and our weighted MOCU perform similarly. This is because in this setting $p(y|x,\theta)$ is either 1 or 0, so there is no irrelevant uncertainty with which $p(y|x,\theta)$ is always larger or smaller than 0.5 but the value is uncertain.

In addition to the average performance comparison, we deliberately choose one of the runs in which the ELR method gets stuck to better illustrate the difference between the existing ELR methods and the proposed weighted-MOCU based method. In this run, the randomly chosen parameters are $(a=-3, b=0, c=1.9)$. Fig. A.4a shows the error regret (the OBC error minus the true optimal classifier error) comparison, in which ELR gets stuck and the weighted-MOCU based method reaches 0. Notice that the y-axis is in the logarithm scale, so the vertical line in the WMOCU plot implies that the value turns to 0. Error regret equals to 0 indicates that the OBC classifier equals to the true optimal classifier, but in practice we don't know the true optimal classifier, so we need the value of MOCU to quantify the expected error difference between OBC and the optimal classifier of each $\theta=(a,b,c)$. Fig. A.4b shows the changes of MOCU value during the two active learning procedures. Not surprisingly, the MOCU value during the iterations of the ELR method also gets stuck, while the MOCU value in the iterations of the weighted-MOCU method continues to decrease. Fig. A.4c shows the changes of the maximum value of acquisition function in each iteration. The acquisition function of ELR decrease to 0 after 22 iterations, and that explains why ELR gets stuck. On the other hand, the maximum acquisition function of WMOCU is always positive as the corresponding MOCU is positive, until it gets close to 10^{-16}, which is the rounding error in floating point arithmetic. In theory, as the observation is noisy, we can not be sure of the optimal prediction. Therefore, the MOCU and the acquisition function of weighted-MOCU should always be positive, which is demonstrated in the figures.

We have also performed an experiment to show the algorithm performance change under different noise levels. We set the flipping error rate as $p(y\neq z|\boldsymbol{x})=\epsilon\times(1-4x_1^2)+\epsilon, 0\leq\epsilon\leq 0.25$. Therefore, when $x_1=0$, the flipping error is 2ϵ; and when $x_1=\pm0.5$, the flipping error is ϵ. We perform the same methods with 100 iterations and 100 runs on the noise level $\epsilon=0.05$ and $\epsilon=0.25$. The resulting active learning performance curves are illustrated in Fig. A.5. We can see from the figure that the performance of MES degrades significantly with high noise while the performance of

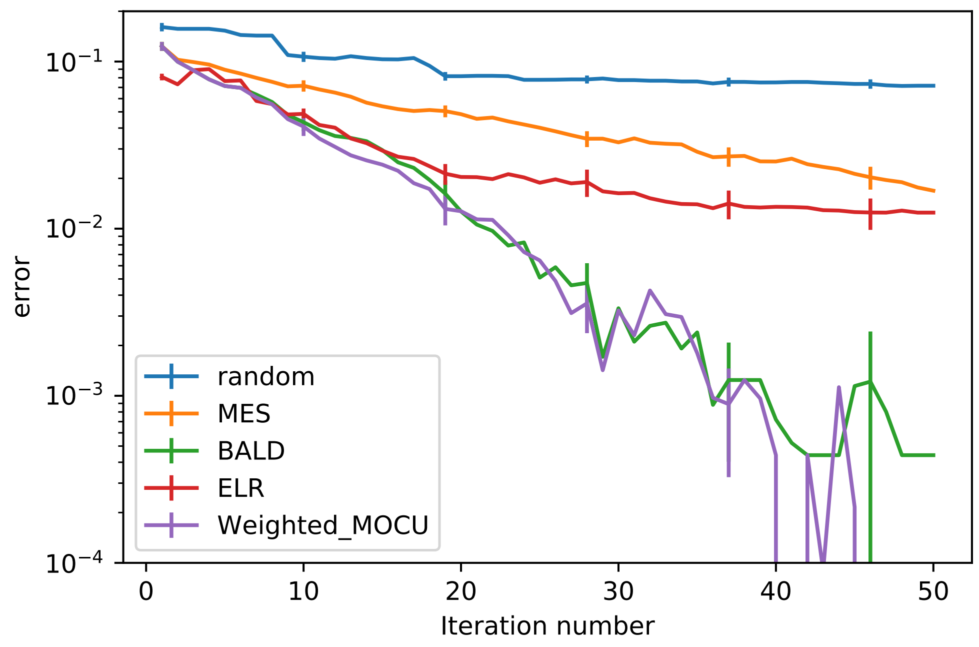
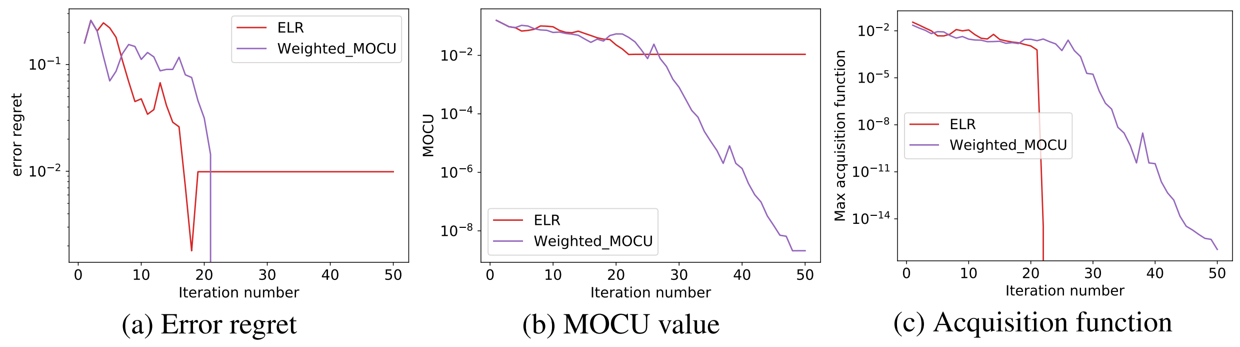


Figure A.3: The expected OBC error comparison between different active learning algorithms in the setting with heterogeneous observation error.



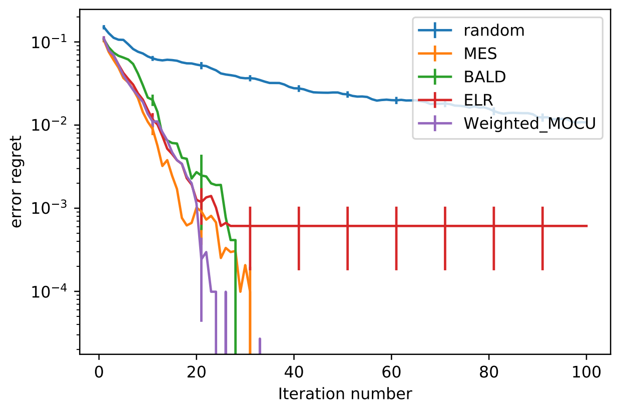
(a) Error regret (b) MOCU value (c) Acquisition function

Figure A.4: Comparison of ELR and weighted MOCU on a specific run.

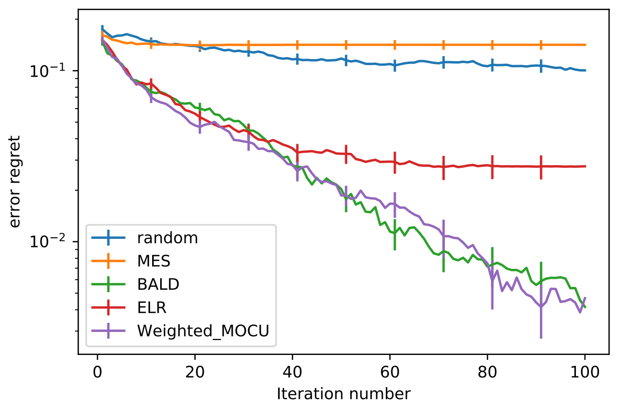
other methods does not appear to be very sensitive to the increasing noise level. .

A.6 Additional WMOCU Real-world Benchmark Experiments

We here present the complete results on the UCI User Knowledge dataset [38]. In addition to the uncertainty class setup in chapter 3, we have tested two other setups of hyperparameter values: 1) ‘uniform prior’ with $\alpha_i = \beta_i = 1$, and 2) ’good prior’ with $\alpha_i = \beta_i = 10$ in eight bins chosen



(a) Noise level $\epsilon = 0.05$

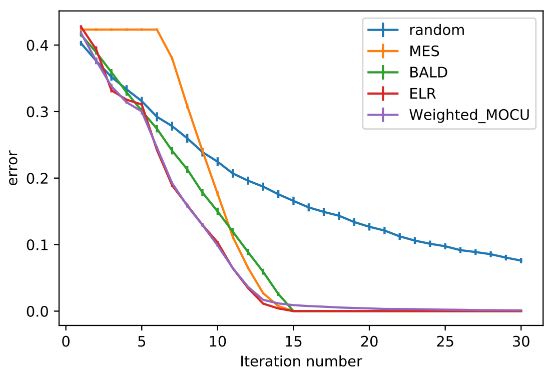


(b) Noise level $\epsilon = 0.25$

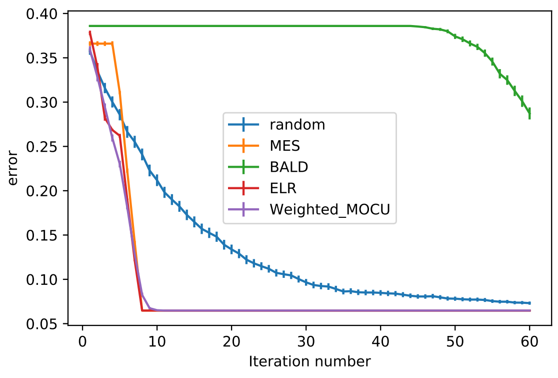
Figure A.5: Active learning algorithm performance comparison with different noise levels.

randomly, for other bins $\alpha_i = 5, \beta_i = 2$ if the true frequency of High or Medium in the i-th bin is higher than 0.5 and $\alpha_i = 2, \beta_i = 5$ if the frequency is lower than 0.5. We also randomly draw 150 samples from each class as the candidate pool and perform the five different active learning algorithms. We repeat the whole procedure 150 times and the average error rates are shown in Fig. A.6. In both Fig. A.6a and Fig. A.6b, ELR performs the best in these two setups while our Weighted MOCU performs similarly. BALD performs reasonably in Fig. A.6a but it again performs poorly in Fig. A.6b. This is because the bins with $\alpha = \beta = 10$ have less uncertainty but have more impact on OBC prediction and BALD fails to identify that in this setup again.

We also present the results on the UCI Letter Recognition dataset [22]. Letter Recognition is a multi-class classification dataset with each sample having 16 numerical features generated from typed images of the capital letters in the English alphabet. We select two pairs of hard-to-distinguish letters: E vs. F and D vs. P. The total number of training samples is 1543 and 1608 for E vs. F and D vs. P, respectively. Active learning algorithms are applied with Bayesian logistic regression models. We randomly take 100 data points first to construct the prior, and use the rest of the data as the pool to test the five active learning algorithms. For prior construction, we train a logistic regression model on the 100 data points and take the trained parameters as the mean of a normal distributed prior with the variance equal to 1. Then we sample 1000 particles from the prior as the

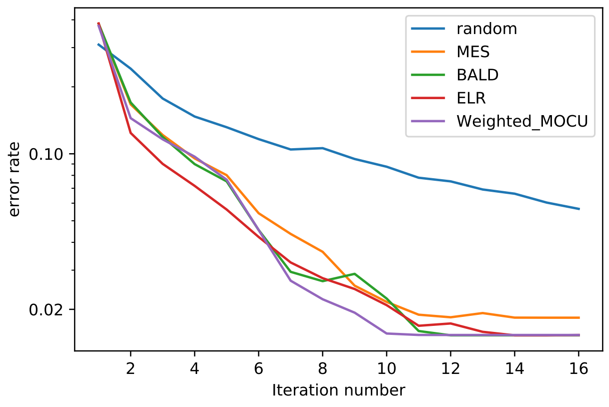


(a) Uniform prior

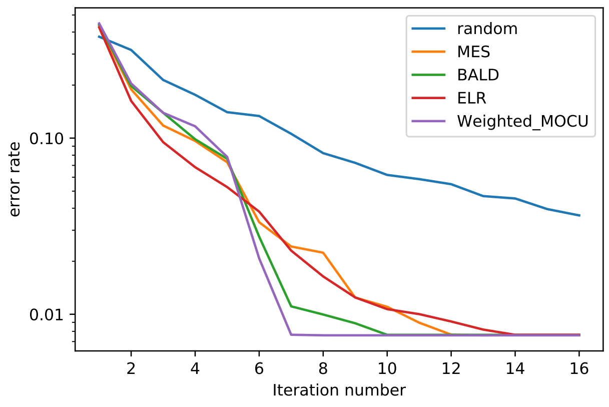


(b) Good prior

Figure A.6: Classification error rate comparison on UCI User Knowledge dataset.



(a) Performance of letter E vs. F classification



(b) Performance of letter P vs. D classification

Figure A.7: Classification error rate comparison on UCI Letter Recognition dataset.

uncertain parameter set. We repeat the whole procedure 100 times and the average error rates are shown in Fig. A.7. Unlike the synthetic datasets, the real-world datasets have no corresponding true models. We can only find the optimal models that approximate the data best. However, we can still see the trends of different algorithms. Compared with random sampling, all the algorithms quickly converge to the optimal models. ELR performs the best in the first several iterations, while converges slowly in the latter iterations. Our weighted MOCU based method is again demonstrated to converge faster than other competing methods.

It is clear from all our experiments for both simulated and real-world data that, in addition to its theoretical guarantee for active learning with OBC, our weighted MOCU method has achieved consistently better or similar empirical performance compared to the best performing ones among the existing pool-based active learning methods, approaching the corresponding OBCs faster with fewer labeled samples.

APPENDIX B

APPENDIX FOR SOFT-MOCU BASED ACTIVE LEARNING

In this appendix section, we provide the pseudo-code of our Soft-MOCU-based active learning method together with complexity analysis in Appendix B.1, an illustrative example to demonstrate the problem of MOCU-based active learning in Appendix B.2, and more detailed descriptions of our experiments, additional results, and discussions in Appendix B.3 & B.4.

B.1 Soft-MOCU-based Active Learning & Computational Complexity

The pseudo-code of our Soft-MOCU-based active learning method is given in Algorithm 5 with the detailed descriptions of AcquisitionFun and smocu functions. We further estimate the computational complexity of our Soft-MOCU-based active learning.

Given the discrete feature space with the cardinality $N_x = |\mathcal{X}|$ and the uncertainty set of classifiers with $N_\theta = |\Theta|$ different models. For active learning, there are T iterations of queries as the total budget. We study the total complexity of the active learning method. In the smocu function, line 6 is called for $O(N_x N_\theta)$ times. In AcquisitionFun, smocu is called for constant times. Finally, in the main procedure, in each iteration, AcquisitionFun is called for each $\boldsymbol{x}$. Hence, the total complexity of Soft-MOCU-based active learning is $O(TN_x^2 N_\theta)$.

B.2 A Synthetic Example Where Mocu-based Active Learning Gets Stuck

We provide an example for intuitive illustration of how MOCU-based active learning can get stuck without converging to the true optimal classifier. Consider a binary classification problem with the uncertain class of two models $\Theta = \{\theta_1, \theta_2\}$ and the candidate pool with two candidates $\mathcal{X} = \{x_1, x_2\}$. We set the probabilistic model and the prior $\pi(\theta)$ as shown in Tables B.1 and B.2. In this setting, we can calculate the predictive probabilities, which are also shown in Table B.1, indicating the OBC $\psi_{\pi(\theta)}(x_1) = 1$ and $\psi_{\pi(\theta)}(x_2) = 1$. In the MOCU calculation, there is only one nonzero term $C_{\theta_2}(\psi_{\pi(\theta)}, x_2) - C_{\theta_2}(\psi_{\theta_2}, x_2)$ within the expectation in (3.2). So the MOCU $\mathcal{M}(\pi(\theta)) = 0.01 > 0$, indicating the OBC has not converged to the true optimal classifier corresponding to either the underlying true model θ_1 or θ_2.

Assume that the observation labels of x_1 and x_2 are y_1^* and y_2^* respectively. Based on the different observations of y_1^* or y_2^*, we can calculate the posterior probability of θ as shown in Table B.2, and the posterior predictive probability as shown in Table B.1. Since for x_1 the probability of y_1 is the

Algorithm 5 Soft-MOCU-based active learning

```
 1: function MainProcedure( )
 2:     Set a discrete candidate set $\mathcal{X}$, the probability array $p_x$, and iteration number T
 3:     Set the discrete parameter set $\Theta$ and the corresponding probability array $\pi_\theta$
 4:     Set approximation parameter $k$
 5:     Initialize the data set $D = \emptyset$
 6:     $\pi_{\theta|D} = \pi_\theta$
 7:     for $t = 1$ to $T$ do
 8:         for $x$ in $\mathcal{X}$ do
 9:             Store AcquisitionFun($x, \pi_{\theta|D}$) to the array $U_{\mathcal{X}}$
10:         end for
11:         Optimize $U_{\mathcal{X}}$ and find the maximum point $x^*$
12:         Obtain the label $y^*$ corresponds to $x^*$ and update $D = D \cup \{x^*, y^*\}$
13:         for $\theta$ in $\Theta$ do
14:             Update $\pi_{\theta|D} \propto \pi_{\theta|D} \cdot p(y^*|x^*, \theta)$
15:         end for
16:         $\mathcal{X} = \mathcal{X}/\{x^*\}$
17:     end for
18: end function

19: function AcquisitionFun($x, \pi_{\theta|D}$)
20:     $smocu\_current$ =SMOCU($\pi_{\theta|D}$)
21:     $smocu\_next = 0$
22:     for $y$ in $\{0, 1, \ldots, M-1\}$ do
23:         for $\theta$ in $\Theta$ do
24:             Generate array $p(\theta, y|D, x) = \pi_{\theta|D} \cdot p(y|x, \theta)$
25:         end for
26:         $p(y|D, x)$ = $\sum_\theta p(\theta, y|D, x)$
27:         $\pi_{\theta|D,x,y} = p(\theta, y|D, x)/p(y|D, x)$
28:         $smocu\_next = smocu\_next + p(y|D, x) \cdot \text{SMOCU}(\pi_{\theta|D,x,y})$
29:     end for
30:     return $smocu\_current - smocu\_next$
31: end function
```

```
 1: function SMOCU($\pi_{\theta|D}$)
 2:     $smocu = 0$
 3:     for $x'$ in $\mathcal{X}$ do
 4:         $bayesian\_max = 0$
 5:         for $\theta$ in $\Theta$ do
 6:             $bayesian\_max = bayesian\_max + \pi_{\theta|D} \cdot \max_{y'} p(y'|x', \theta)$
 7:         end for
 8:         $p(y'|D, x') = \sum_{\theta} \pi_{\theta|D} \cdot p(y'|x', \theta)$
 9:         $G = bayesian\_max - \text{LogSumExp}[k \cdot p(y'|D, x')]/k$
10:         $smocu = smocu + p(x') \cdot G$
11:     end for
12:     return $smocu$
13: end function
```

Table B.1: The probabilities of $p(y|x, \theta)$ and predictive probabilities.

	$x = x_1,\ y = y_1$	$x = x_2,\ y = y_2$
$p(y\|x, \theta_1)$	(0.3, 0.7)	(0.4, 0.6)
$p(y\|x, \theta_2)$	(0.3, 0.7)	(0.6, 0.4)
$p(y\|x)$	(0.3, 0.7)	(0.44, 0.56)
$p(y\|x, y_1^* = 0)$	(0.3, 0.7)	(0.44, 0.56)
$p(y\|x, y_1^* = 1)$	(0.3, 0.7)	(0.44, 0.56)
$p(y\|x, y_2^* = 0)$	(0.3, 0.7)	(0.4546, 0.5454)
$p(y\|x, y_2^* = 1)$	(0.3, 0.7)	(0.4286, 0.5714)

Table B.2: The prior and posterior of $\pi(\theta)$.

	$\pi(\theta)$	$\pi(\theta\|x_1, y_1^* = 0)$	$\pi(\theta\|x_1, y_1^* = 0 \text{ or } 1)$	$\pi(\theta\|x_2, y_2^* = 0)$	$\pi(\theta\|x_2, y_2^* = 1)$
$\theta = \theta_1$	0.8	0.8	0.8	0.727	0.857
$\theta = \theta_2$	0.2	0.2	0.2	0.273	0.143

same for both models, the posterior distribution of $\pi(\theta|x_1, y_1^*)$ does not change no matter what is the true label y_1^*. Therefore, querying x_1 cannot provide any information to the model and it is easy to verify from (3.5) that:

$$U^M(x_1, \pi(\theta)) = \mathcal{M}(\pi(\theta)) - \mathbb{E}_{y|\boldsymbol{x}}\mathcal{M}(\pi(\theta|x_1, y_1^*)) = 0. \tag{B.1}$$

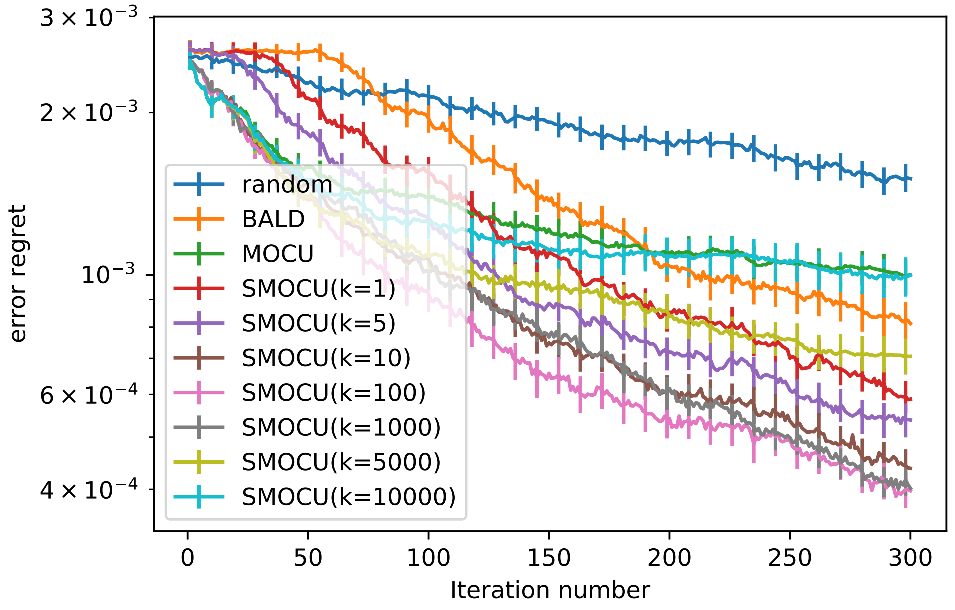


Figure B.1: Comparison of the expected OBC error regret by SMOCU with different k values together with random sampling as well as MOCU- and BALD-based active learning methods. The performance trends clearly show that SMOCU with appropriately chosen k values can significantly outperform the competing methods. Note that MOCU and SMOCU with large k values achieve the fastest convergence at the first 50 iterations as expected due to their local optimality. SMOCU-based methods with small k values ($k = 1$ or 5) have similar performance trends as BALD but perform better.

On the other hand, querying y_2^* changes the posterior distribution of θ but it does not change the OBC for y_2^* being either 0 or 1, i.e. $\psi_{\pi(\theta)} = \psi_{\pi(\theta|x_2,y_2^*=0)} = \psi_{\pi(\theta|x_2,y_2^*=1)}$. As we have discussed, that means $\pi(\theta)$ and $\pi(\theta|x_2, y_2^*)$ are within the same linear piece of MOCU, and in this case the acquisition function is 0. In fact, based on (3.6), we have:

$$\begin{aligned}U^{\mathrm{M}}(x_2;\pi(\theta)) &= \mathbb{E}_x\{\mathbb{E}_{\pi(\theta)}[C_\theta(\psi_{\pi(\theta)},x)]\} - \mathbb{E}_x\{\mathbb{E}_{p(y_2^*|x_2)}[\mathbb{E}_{\pi(\theta|x_2,y_2^*)}[C_\theta(\psi_{\pi(\theta|x_2,y_2^*)},x)]]\},\\ &= \mathbb{E}_x\{\mathbb{E}_{\pi(\theta)}[C_\theta(\psi_{\pi(\theta)},x)]\} - \mathbb{E}_x\{\mathbb{E}_{p(y_2^*|x_2)}[\mathbb{E}_{\pi(\theta|x_2,y_2^*)}[C_\theta(\psi_{\pi(\theta)},x)]]\} = 0, \end{aligned} \quad \text{(B.2)}$$

where the second line holds as $\pi(\theta) = \mathbb{E}_{p(y_2^*|x_2)}[\pi(\theta|x_2, y_2^*)]$. Therefore, although MOCU is larger than 0, the corresponding acquisition function is 0 for all the candidates. Hence, the MOCU-based active learning can get stuck in this case without identifying whether the true model is θ_1 or θ_2 to derive the true optimal classifier.

From the example above we can see that the MOCU-based method gets stuck when a single query of y_2^* does not change the label of the OBC even though it changes the posterior of $\pi(\theta)$. However, if we repetitively observe y_2^*, the posterior of $\pi(\theta_1)$ will converge to either 0 or 1 with reduced model uncertainty until identifying the true underlying model θ_1 or θ_2 and the MOCU will converge to 0. That shows the one-step-look-ahead strategy based on MOCU reduction cannot identify the long term effect of a single query. On the other hand, our Soft-MOCU-based acquisition function can capture the changes of posterior, even with small changes, due to the strict concavity of SMOCU. Therefore, SMOCU-based active learning does not have this problem and alleviates the myopic behavior as demonstrated in our experiments.

B.3 Additional SMOCU Synthetic Experiments

In Fig. B.1, we show more results with different values of k for our Soft-MOCU-based active learning method on the example shown in Fig. 3.6. From the figure, we can see as the value of k increases, the performance curve of SMOCU changes gradually, from the curve ($k = 1$) close to the performance curve of BALD, to the best performing curve ($k = 100$), then to the performance curve ($k = 10000$) close to that of MOCU. The performance trends clearly show that SMOCU with appropriately chosen k values can significantly outperform the competing methods. Note that MOCU and SMOCU with large k values achieve the fastest convergence at the first 50 iterations as expected due to their local optimality. SMOCU-based methods with small k values ($k = 1$ or 5) have similar performance trends as BALD but perform better.

We further run the same synthetic experiment of Fig. 3.7 with a different prior setting: $w_1 \sim \mathcal{U}(0.3, 0.8)$, $w_2 \sim \mathcal{U}(-0.25, 0.25)$ and $b \sim \mathcal{U}(-0.25, 0.25)$. With this prior setting, the region near the x_2 axis where the decision boundary lies has similar uncertainty of $p(y|\boldsymbol{x}, \boldsymbol{w}, b)$ compared with the region far away from the x_2 axis. The results are shown in Fig. B.2. Under this setting, BALD performs well as expected since the model uncertainty affects the classification performance similarly, but our Soft-MOCU-based method still performs better than BALD and other competing methods.

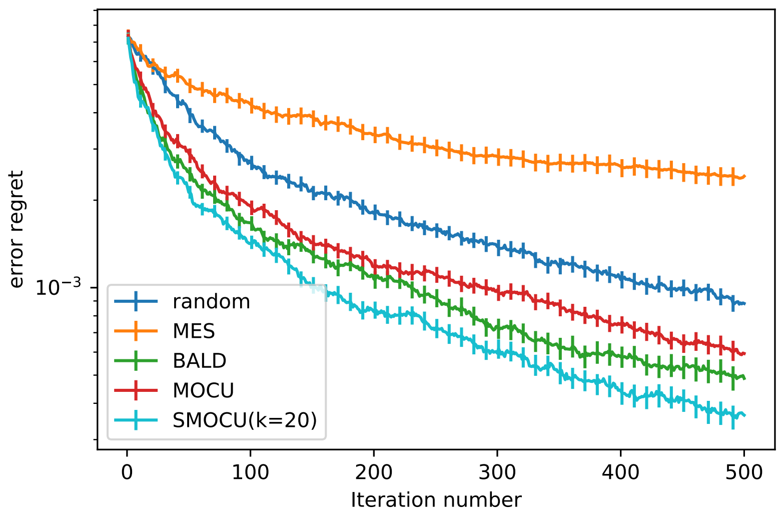
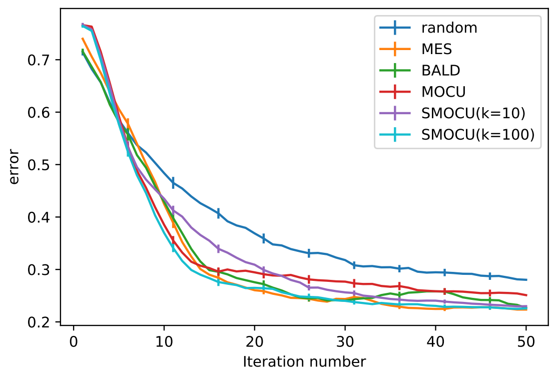


Figure B.2: The expected OBC error comparison between different active learning methods on binary classification.

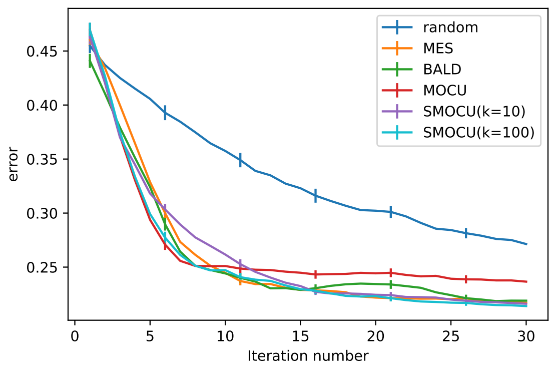
B.4 Additional SMOCU Real-world Benchmark Experiments

We here present the complete results on the UCI User Knowledge dataset [38]. In addition to the uncertainty class setup in section 3.6.3, we have tested two other setups of hyperparameter values. In the first setup (`Uniform prior`), uniform priors are adopted for all the 16 bins with $\boldsymbol{\alpha}^{(i)} = \mathbf{1}$, $1 \leq i \leq 16$. In the second setup (`Good prior`), we randomly choose 8 bins and set uniform priors on them with $\boldsymbol{\alpha}^{(i)} = \mathbf{1}$; for the other 8 bins, the priors are set as $\alpha_j^{(i)} = 10$ if j corresponds to the true label, and $\alpha_j^{(i)} = 1$ for the other labels. This setting is opposite to the setting in section 3.6.3.

We also randomly draw 50 samples from each class as the candidate pool and perform the same active learning methods. We repeat the whole procedure for 150 times and the averaged classification error is shown in Fig B.3. In both prior settings, MOCU-based method gets stuck (degraded to random sampling) before converging to the optimal classifier, though it converges the fastest at the beginning (first 10 iterations). On the other hand, in both cases our Soft-MOCU-based method with $k = 100$ has the fast convergence both at the beginning and in the long run.



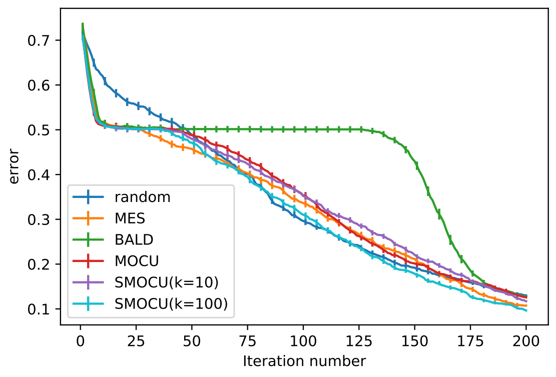
(a) `Uniform prior`



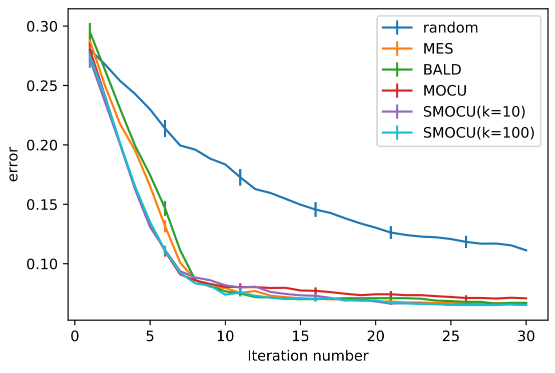
(b) `Good prior`

Figure B.3: Classification error rate comparison on the four-class classification problem with UCI User Knowledge dataset.

We also have made a binary classification problem from the UCI User Knowledge dataset for comparison between these different methods. We group these samples into two classes: 0: `High` or `Medium`; 1: `Low` or `Very Low`. With the same feature space and bin settings, we have tested two setups of hyperparameter values for the Dirichlet priors. The first setup (`Bad prior`) is the same as the one we have used to generate Fig. 3.9, and the second setup (`Good prior`) is the same as the one adopted for Fig. B.3b.



(a) `Bad prior`



(b) `Good prior`

Figure B.4: Classification error rate comparison on the binary classification problem with UCI User Knowledge dataset.

We have randomly drawn 150 samples from each class as the candidate pool and perform the same comparison with different active learning methods. Again we repeat the whole procedure 150 times and the averaged classification error is shown in Fig. B.4. The performance trends are similar as those observed for the four-class classification problem. Our Soft-MOCU-based method with $k = 100$ has achieved better or similar performance compared with the best performing one among the competing methods.