Integrating Inference and Experimental Design for Contextual Behavioral Model Learning

Gongtao Zhou, Haoran Yu*

School of Computer Science & Technology, Beijing Institute of Technology
zhougt3@qq.com, yhrhawk@gmail.com

Abstract

The strategic behavior of users is significantly influenced by their hidden information such as private valuations, risk preferences, and price sensitivities. *Contextual behavioral model learning* refers to learning the dependence of users' hidden information on their observable context information. While many existing studies use offline data to learn contextual behavioral models, we study how to design sequential experiments to collect the most informative user behavioral data for learning. We propose a basic *inference-then-design* method. In each experimental period, it infers a probabilistic contextual behavioral model using historical experimental data, and then designs the new experiment to maximize the gain of information about the probabilistic model. We further improve the basic method in two aspects. First, we improve the inference step by specifying a more informative prior for learning the probabilistic contextual behavioral model. Second, we integrate the inference and design steps instead of conducting them separately. Our rigorous theoretic analysis reveals that the optimization objective of the inference step can be modified to account for the downstream experimental design step. Numerical experiments show that our methods lead to more effective experiments, i.e., the collected experimental data can help in learning a more accurate behavioral model.

1 Introduction

One key challenge in predicting user strategic behavior is that it depends on some hidden information that is not observed by the platforms. For example, purchasing behavior relies on users' hidden utilities across different products (Donnelly et al. 2021), and bidding behavior depends on users' private valuations of the items being sold (Duan et al. 2022). Platforms can only observe users' *context information* (e.g., demographic information and searching history), and users' hidden information is usually assumed to be dependent on their context information (Donnelly et al. 2021; Duan et al. 2022). Accurately learning the dependence between users' context information and hidden information is crucial for platforms to predict user strategic behavior and optimize their operations.

To learn users' contextual behavioral models, platforms can actively design experiments to interact with users and collect the behavioral data of users with different contexts. For example, it is increasingly common for trading platforms to use robo-advisors to interact with investors and elicit hidden information from their choices across portfolios (Alsabah et al. 2021; Wang and Yu 2021; Dai et al. 2021). As shown in Figure 1, a robo-advisor queries an investor regarding its preference over several portfolios with different return means and variances. The investor's choice helps the trading platform infer the relation between an investor's observable context information and its hidden information (e.g., risk preference and decision-making rationality).

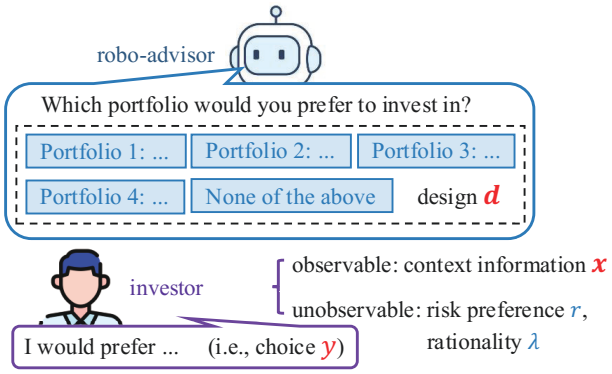

Figure 1: A Portfolio Choice Experiment.

Our work studies how to design sequential experiments to learn an accurate contextual behavioral model. We consider a general unknown mapping from context information to hidden information, and model it using a neural network. Essentially, our problem is to design experiments that collect the most informative data for learning the network weights.

We focus on the application of portfolio choice model learning in Figure 1, while our solution applies to general contextual behavioral model learning problems. Our basic idea is to conduct the *inference step* and *design step* in each period. First, we use all historical data to infer the distribution of the neural network weights (which characterize the mapping from the context to hidden information). Second, we choose the next experimental design to maximize the gain of information about the neural network weights.

Our work identifies the limitations of the above basic

*Corresponding author.

inference-then-design method, and improves it on two aspects. First, we improve the *inference* step by specifying a more informative prior distribution of neural network weights. Second, we integrate the *inference* and *design* steps instead of conducting them separately. We derive a new joint objective function that guides the implementations of both steps. Compared with baselines, our methods can collect more informative data for learning the mapping from the context to hidden information.

We summarize our contributions as follows:

- On the application side, we propose a general experimental design framework. It can be applied to learn a general complex mapping from users' contexts to their hidden information (e.g., risk preferences, rationalities, price sensitivities (Perivier and Goyal 2022), and valuations (Shah, Johari, and Blanchet 2019)) and further predict their selection, purchasing, and bidding behavior.
- On the methodology side, we study the integration of inference and design steps. Conventionally, the objective of variational inference is minimizing the KL divergence between the variational and true posterior distributions. Our theoretical analysis reveals that the inference objective should include an extra KL divergence term to account for the downstream experimental design step.

2 Related Work

2.1 Contextual Behavioral Model Learning

In offline contextual behavioral model learning, the problem is to learn the relation between contextual features and user hidden information, given an offline dataset of the observed contextual features and user behavior (Ling, Fang, and Kolter 2018; Athey et al. 2018; Perrault et al. 2020; Donnelly et al. 2021). Growing attention has been paid to the online case, and most existing studies investigated how to actively set prices to learn the mapping from contexts to user private valuations on products (Golrezaei, Javanmard, and Mirrokni 2019; Cohen, Lobel, and Paes Leme 2020; Xu and Wang 2021; Luo, Sun, and Liu 2022). However, these studies assumed linear context-valuation mappings. Our work explores a general setting where (i) the mapping between context and hidden information can be non-linear and complex and (ii) the hidden information is multidimensional (e.g., it includes both risk preference and rationality).

2.2 Active Learning

Active learning aims to train a machine learning model by selecting the most useful unlabeled data for labeling (Ren et al. 2021; Zhan et al. 2022). Related studies mainly focused on designing the selection criteria. One category of criteria measures uncertainty, e.g., *MaxEntropy*, *LeastConfidence*, *BALD*, and *MeanSTD* (Wang and Shang 2014; Gal, Islam, and Ghahramani 2017). For instance, *MaxEntropy* suggests selecting data points with the most uncertain predictive labels for labeling. Another category measures density, e.g., *CoreSet* and *Cluster-Margin* (Ash et al. 2020; Citovsky et al. 2021). The main idea is to choose data points that best represent the distribution of the entire unlabeled dataset.

Unlike active learning, we cannot actively choose users to query an oracle about their hidden information in our problem. Instead, the users arriving at each period and their contexts are exogenously given. Moreover, after choosing the experimental design, we can only observe the behavior of users rather than directly access their hidden information.

2.3 Bayesian Optimal Experimental Design

Bayesian optimal experimental design seeks to design experiments to maximize the information gain about an unknown quantity of interest (Foster et al. 2019; Zheng et al. 2020). Few studies in this field have specifically designed experiments to reduce uncertainty in neural network weights. Since exactly inferring the distribution of neural network weights is intractable, we approximate it using variational inference. In particular, we propose a novel method that integrates variational inference with the downstream experimental design. Furthermore, many studies assumed that the underlying distribution for generating the unknown quantity of interest is given and directly used it as the prior distribution (Kleinegesse and Gutmann 2020; Foster et al. 2021). Our work considers a general case where this underlying generative distribution is inaccessible.

3 Problem Formulation

In this section, we first introduce investors' portfolio choice model. Then, we describe the platform's experiment-based learning of investor risk aversions and decision-making rationalities. Last, we introduce the platform's experimental design problem.

3.1 Investor Portfolio Choice Model

We focus on learning the portfolio choice behavior of investors, which is crucial for trading platforms to effectively recommend portfolios to investors. Let $\boldsymbol{x}$ denote an investor's context information (e.g., age, education, occupation, and monthly income). We use $r(\boldsymbol{x}) \in \mathbb{R}$ to denote the investor's risk aversion coefficient, which is a function of its context $\boldsymbol{x}$. If $r(\boldsymbol{x}) < 0$, the investor is risk-seeking.

Suppose that the investor is presented with K distinct portfolios, where each portfolio k has a return mean $m_k \geq 0$ and a return variance $v_k \geq 0$.[1] According to Markowitz's mean-variance model (Markowitz 1952), an investor seeks to maximize $m_k - r(\boldsymbol{x})v_k$ (e.g., a risk-averse investor with $r(\boldsymbol{x}) > 0$ seeks to increase the return mean and reduce the return variance). We further use the logit model to characterize the randomness in the investor's choice (Werden, Froeb, and Tardiff 1996; Frijns, Koellen, and Lehnert 2008). The probability that the investor's choice y is portfolio k is modeled by

$$p(y=k|\boldsymbol{x},\boldsymbol{m},\boldsymbol{v}) = \frac{\exp(\lambda(\boldsymbol{x})(m_k - r(\boldsymbol{x})v_k))}{\sum_{\tilde{k}=1}^{K}\exp(\lambda(\boldsymbol{x})(m_{\tilde{k}} - r(\boldsymbol{x})v_{\tilde{k}}))}. \quad (1)$$

[1] Each portfolio k specifies the allocation of investments in multiple assets. Given the investment allocation and the expectation and covariance matrix of the returns of the assets, we can compute m_k and v_k.

Here, $\boldsymbol{m}$ and $\boldsymbol{v}$ represent the return means and variances of all portfolios. $\lambda(\boldsymbol{x}) > 0$ characterizes the investor's decision-making rationality, and the value also depends on the context $\boldsymbol{x}$. It quantifies the degree of randomness in the investor's choice, e.g., $\lambda(\boldsymbol{x}) \to \infty$ means that the investor is fully rational to choose the best portfolio.

To ensure that the investor always has the option to not invest, we set the K-th portfolio to be a virtual portfolio with $m_k = v_k = 0$ in (1).

In practice, the dependence of an investor's risk aversion coefficient and decision-making rationality on its context is complex. The trading platforms initially do not know $r(\boldsymbol{x}), \lambda(\boldsymbol{x})$, and need to learn these functions, through observing the choice y made by each investor with context $\boldsymbol{x}$ under the available portfolios characterized by $\boldsymbol{m}$ and $\boldsymbol{v}$.

We consider the commonly used portfolio choice model in (1) to show the effectiveness of our experimental design solution. We remark that our solution can be generalized to the contextual behavioral models in other applications. As long as the parameters in the behavioral models (e.g., the risk aversion coefficient and decision-making rationality in (1)) are functions of the users' contexts and we can compute the likelihood (e.g., the expression of $p(y = k|\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{v})$ in (1)), our experimental design solution can be applied.

3.2 Platform's Experiment-Based Learning

To learn $r(\boldsymbol{x}), \lambda(\boldsymbol{x})$, a trading platform can first collect data regarding different investors' choices given different sets of portfolios, and then train a neural network on the data to approximate $r(\boldsymbol{x}), \lambda(\boldsymbol{x})$. In order to collect the most informative data for training the neural network, the trading platform can actively select the set of portfolios and query the investors regarding their preferences (Alsabah et al. 2021; Wang and Yu 2021; Dai et al. 2021).

We consider a discrete-time system. In each period $t = 1, 2, ..., T$, a batch of investors arrives and the trading platform carries out the following procedures:

- *Observing Investor Context $\boldsymbol{X}_t$:* The platform observes the contexts of these investors, denoted by the matrix $\boldsymbol{X}_t$;
- *Designing Portfolio Set $\boldsymbol{d}_t$:* The platform designs which portfolios to show to the investors. Let $\boldsymbol{d}_t = (\boldsymbol{m}_t, \boldsymbol{v}_t)$ denote the platform's design at period t, which specifies the return means and variances of the shown portfolios. We further use $\mathcal{D}_t$ to denote the set of available designs, which can change over t. Note that an investor can always choose not to invest, i.e., every design $\boldsymbol{d} \in \mathcal{D}_t$ includes a virtual portfolio with a zero mean and a zero variance;
- *Observing Investor Choice $\boldsymbol{y}_t$:* After choosing $\boldsymbol{d}_t$ and showing the portfolios to the investors with context $\boldsymbol{X}_t$, the platform asks about their choices, denoted by $\boldsymbol{y}_t$. Each element of vector $\boldsymbol{y}_t$ records the portfolio choice of the corresponding investor.

Next, we describe how to learn $r(\boldsymbol{x}), \lambda(\boldsymbol{x})$ using the collected data. Let $\mathcal{H}_t \triangleq \{(\boldsymbol{X}_1, \boldsymbol{d}_1, \boldsymbol{y}_1), ..., (\boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{y}_t)\}$ denote the set of data points that the platform collects during the first t periods. After all the T periods, the platform gets the dataset $\mathcal{H}_T$ and can train a multi-output neural network parameterized by $\boldsymbol{\theta}$ on $\mathcal{H}_T$ to learn $r(\boldsymbol{x}), \lambda(\boldsymbol{x})$. Let $\hat{r}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $\hat{\lambda}_{\boldsymbol{\theta}}(\boldsymbol{x})$ denote the predictions made by the neural network. The platform can train the model by maximizing the log-likelihood of observing all investor choices:

$$\max_{\boldsymbol{\theta}} \sum_t \log p\left(\boldsymbol{y}_t | \boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{\theta}\right). \tag{2}$$

Here, the likelihood of observing each investor's choice can be computed by replacing the true $r(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$ in (1) by $\hat{r}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $\hat{\lambda}_{\boldsymbol{\theta}}(\boldsymbol{x})$, respectively.

3.3 Platform's Experimental Design Problem

The platform's sequential design of $\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_T$ affects the quality of $\mathcal{H}_T$ and the training of the neural network. We introduce the experimental design problem as follows.

Problem: *In each period $t = 1, 2, \ldots, T$, given the historical dataset $\mathcal{H}_{t-1}$ and the current investors' context matrix $\boldsymbol{X}_t$, we aim to optimize the design $\boldsymbol{d}_t \in \mathcal{D}_t$ so that the resulting dataset $\mathcal{H}_T$ is most informative for training an accurate $\boldsymbol{\theta}$-parameterized neural network.*

Note that we consider a general time-variant system where the distributions of $\boldsymbol{X}_t$ (i.e., the arrivals of investors with different contexts) and $\mathcal{D}_t$ can change over time and the dynamics is unknown to the platform.

4 Basic Inference-then-Design Method

This section introduces our basic inference-then-design method. First, we formulate the problem of selecting $\boldsymbol{d}_t$ as an expected information gain maximization problem. Then, we infer the distribution of $\boldsymbol{\theta}$, which is the prerequisite for computing the expected information gain.

4.1 Expected Information Gain

Our target of training an accurate $\boldsymbol{\theta}$-parameterized neural network can be viewed as collecting informative data to reduce the uncertainty in $\boldsymbol{\theta}$. Therefore, we select the design to maximize the Expected Information Gain (EIG) in $\boldsymbol{\theta}$ (Lindley 1956; Chaloner and Verdinelli 1995).

Single Experiment In a single experiment, after observing the investor context $\boldsymbol{X}$, the platform chooses $\boldsymbol{d}$ and receives the investor choice vector $\boldsymbol{y}$. The information gain in $\boldsymbol{\theta}$ from this experiment can be characterized by

$$\mathrm{IG}(\boldsymbol{d}, \boldsymbol{y}) \triangleq H[p(\boldsymbol{\theta})] - H[p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{d})]. \tag{3}$$

Here, $p(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{d})$ is its posterior distribution computed based on the experimental outcome, and $H[\cdot]$ denotes the entropy.

In (3), the investor choice vector $\boldsymbol{y}$ depends on the context $\boldsymbol{X}$ and the design $\boldsymbol{d}$. Given $\boldsymbol{X}$ and $\boldsymbol{d}$, its distribution can be estimated by $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{d}) = \mathbb{E}_{p(\boldsymbol{\theta})}[p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{d}, \boldsymbol{\theta})]$. Considering the uncertainty in $\boldsymbol{y}$, we choose $\boldsymbol{d}$ to maximize the EIG:

$$\mathrm{EIG}(\boldsymbol{d}) \triangleq \mathbb{E}\left[\mathrm{IG}(\boldsymbol{d}, \boldsymbol{y})\right]. \tag{4}$$

Since the exact computation of EIG is intractable, we apply the nested Monte Carlo method (Rainforth et al. 2018) to approximate it by

$$\widehat{\mathrm{EIG}}(\boldsymbol{d}) \triangleq \frac{1}{I} \sum_{i=1}^{I} \log \frac{p(\boldsymbol{y}^i|\boldsymbol{X}, \boldsymbol{d}, \boldsymbol{\theta}^{i,0})}{\frac{1}{J}\sum_{j=1}^{J} p(\boldsymbol{y}^i|\boldsymbol{X}, \boldsymbol{d}, \boldsymbol{\theta}^{i,j})}, \tag{5}$$

where $\boldsymbol{\theta}^{i,0}, \ldots, \boldsymbol{\theta}^{i,J}$ are sampled according to $p(\boldsymbol{\theta})$ and $\boldsymbol{y}^i$ is sampled according to $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{d}, \boldsymbol{\theta}^{i,0})$ for each i. Recall that $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{d}, \boldsymbol{\theta})$ is the conditional probability of observing $\boldsymbol{y}$, and can be estimated by replacing $r(\boldsymbol{x}), \lambda(\boldsymbol{x})$ in (1) by $\hat{r}_{\boldsymbol{\theta}}(\boldsymbol{x}), \hat{\lambda}_{\boldsymbol{\theta}}(\boldsymbol{x})$. Parameters I and J denote the number of samples of the outer and inner loops, respectively. The detailed derivation is provided in our supplemental material.

Sequential Experiments In period t of the sequential experiments, we also choose $\boldsymbol{d}_t$ to maximize $\widehat{\mathrm{EIG}}(\boldsymbol{d}_t)$. Compared with the single-experiment case, our knowledge of $\boldsymbol{\theta}$ before the experiment changes from $p(\boldsymbol{\theta})$ to $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$, where $\mathcal{H}_{t-1}$ denotes the data collected before t. Therefore, when we estimate $\widehat{\mathrm{EIG}}(\boldsymbol{d}_t)$ based on (5), we sample $\boldsymbol{\theta}^{i,0}, \ldots, \boldsymbol{\theta}^{i,J}$ according to $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$ rather than $p(\boldsymbol{\theta})$.

4.2 Inference of Neural Network Weights

We investigate the computation of $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$ for estimating $\widehat{\mathrm{EIG}}(\boldsymbol{d}_t)$. The $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$ characterizes the posterior distribution of the neural network weights given the dataset $\mathcal{H}_{t-1}$. It is challenging to exactly compute it using Bayes' theorem, because calculating $\int p(\mathcal{H}_{t-1}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ is intractable.

According to the studies on Bayesian neural networks (Graves 2011), we apply the variational inference method to approximate $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$ by a more tractable $\boldsymbol{\phi}$-parameterized distribution $q(\boldsymbol{\theta}|\boldsymbol{\phi})$. The parameter vector $\boldsymbol{\phi}$ is chosen to minimize the Kullback-Leibler (KL) divergence between the two distributions (Blundell et al. 2015):

$$\begin{aligned}\boldsymbol{\phi}^* &= \arg\min_{\boldsymbol{\phi}} \mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\phi})||p(\boldsymbol{\theta}|\mathcal{H}_{t-1})] \\ &= \arg\min_{\boldsymbol{\phi}} \mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\phi})||p(\boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}[\log p(\mathcal{H}_{t-1}|\boldsymbol{\theta})].\end{aligned} \tag{6}$$

Our work applies the mean-field variational inference (Blei, Kucukelbir, and McAuliffe 2017) and assumes that each element in $\boldsymbol{\theta}$ follows a Gaussian distribution, which offers high flexibility in approximating the true posterior distribution. Let $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote the means and variances of these Gaussian distributions, respectively. In this case, $\boldsymbol{\phi}$ consists of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.

4.3 Implementation

We present our Inference-then-Design (**ID**) method in Algorithm 1. In each period t, we mainly conduct the following two steps:

- *Inference:* Line 3 aims to infer the posterior distribution $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$. As discussed in Section 4.2, we approximate it by a variational posterior distribution $q(\boldsymbol{\theta}|\boldsymbol{\phi})$. We learn $\boldsymbol{\phi}$ by training the corresponding Bayesian neural network on dataset $\mathcal{H}_{t-1}$, where the loss function is derived in (6). In particular, when $t = 1$, there is no historical data (i.e., $\mathcal{H}_0 = \emptyset$), and the loss reduces to $\mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\phi})||p(\boldsymbol{\theta})]$;
- *Design:* Line 5 aims to choose the design to maximize the estimated EIG. As discussed in Section 4.1, we estimate the EIG using the nested Monte Carlo method. To compute $\widehat{\mathrm{EIG}}(\boldsymbol{d}_t)$ based on (5), we need to sample $\boldsymbol{\theta}^{i,0}, \ldots, \boldsymbol{\theta}^{i,J}$ for each i according to $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$. Since computing $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$ is intractable, we sample them from the distribution $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ obtained from line 3 instead.

Algorithm 1: Inference-then-Design (ID)

1: Set $p(\boldsymbol{\theta})$ randomly, and $\mathcal{H}_0 = \emptyset$.
2: **for** period $t = 1$ to T **do**
3: Minimize $\mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\phi})||p(\boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}[\log p(\mathcal{H}_{t-1}|\boldsymbol{\theta})]$ over $\boldsymbol{\phi}$ to get $q(\boldsymbol{\theta}|\boldsymbol{\phi})$.
4: Observe investor context matrix $\boldsymbol{X}_t$.
5: Choose $\boldsymbol{d}_t \in \mathcal{D}_t$ to maximize $\widehat{\mathrm{EIG}}(\boldsymbol{d}_t)$, which is computed based on $q(\boldsymbol{\theta}|\boldsymbol{\phi})$, $\boldsymbol{X}_t$, I, and J.
6: Observe investor choice vector $\boldsymbol{y}_t$ under $\boldsymbol{d}_t$.
7: $\mathcal{H}_t \leftarrow \mathcal{H}_{t-1} \cup \{(\boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{y}_t)\}$.
8: **end for**
9: **return** dataset $\mathcal{H}_T$.

5 Improved Inference-then-Design Methods

In this section, we discuss the limitations of our **ID** method, and propose two improved methods.

5.1 Inference-then-Design with Learnable Priors

One limitation of **ID** is its requirement for an informative prior distribution $p(\boldsymbol{\theta})$. Before the experiments, it is challenging to obtain the knowledge of $\boldsymbol{\theta}$, which is high-dimensional and approximates the mapping from the investor context information to the risk aversion coefficient and decision-making rationality.

When implementing **ID**, one solution is setting $p(\boldsymbol{\theta})$ to a random distribution. However, this randomly chosen $p(\boldsymbol{\theta})$ will disrupt the learning of $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ in the inference step of **ID** during the early periods. This is because the platform only accumulates a small dataset $\mathcal{H}_{t-1}$ during the early periods. When learning $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ in line 3 of Algorithm 1, the scale of the expected likelihood $\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}[\log p(\mathcal{H}_{t-1}|\boldsymbol{\theta})]$ in the loss function is small, and the loss function will be dominated by $\mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\phi})||p(\boldsymbol{\theta})]$. As a result, the randomly chosen $p(\boldsymbol{\theta})$ disrupts the learning of $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ and affects the choice of $\boldsymbol{d}_t$.

We propose Inference-then-Design with Learnable Priors (**ID-LP**), which specifies the prior distribution according to data (Krishnan, Subedar, and Tickoo 2020). Specifically, we set $p(\boldsymbol{\theta})$ according to the collected dataset $\mathcal{H}_{t-1}$ rather than an uninformative random distribution. At the beginning of each period t, we add a new step to set $p(\boldsymbol{\theta})$. We directly train the $\boldsymbol{\theta}$-parameterized neural network on $\mathcal{H}_{t-1}$, and use $\tilde{\boldsymbol{\theta}}$ to denote the neural network weights after the training. Note that $\tilde{\boldsymbol{\theta}}$ is essentially the maximum likelihood estimation of $\boldsymbol{\theta}$ given $\mathcal{H}_{t-1}$. We set $p(\boldsymbol{\theta})$ to a Gaussian distribution:

$$p(\boldsymbol{\theta}) \sim \mathcal{N}\left(\tilde{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\mathrm{prior}}\right), \tag{7}$$

where $\boldsymbol{\Sigma}_{\mathrm{prior}}$ is a predefined covariance matrix. For example, it can be the identity matrix multiplied by a constant.

We illustrate **ID-LP** in Figure 2. After learning $\tilde{\boldsymbol{\theta}}$ and setting $p(\boldsymbol{\theta})$, **ID-LP** performs the same inference and design steps as **ID**.

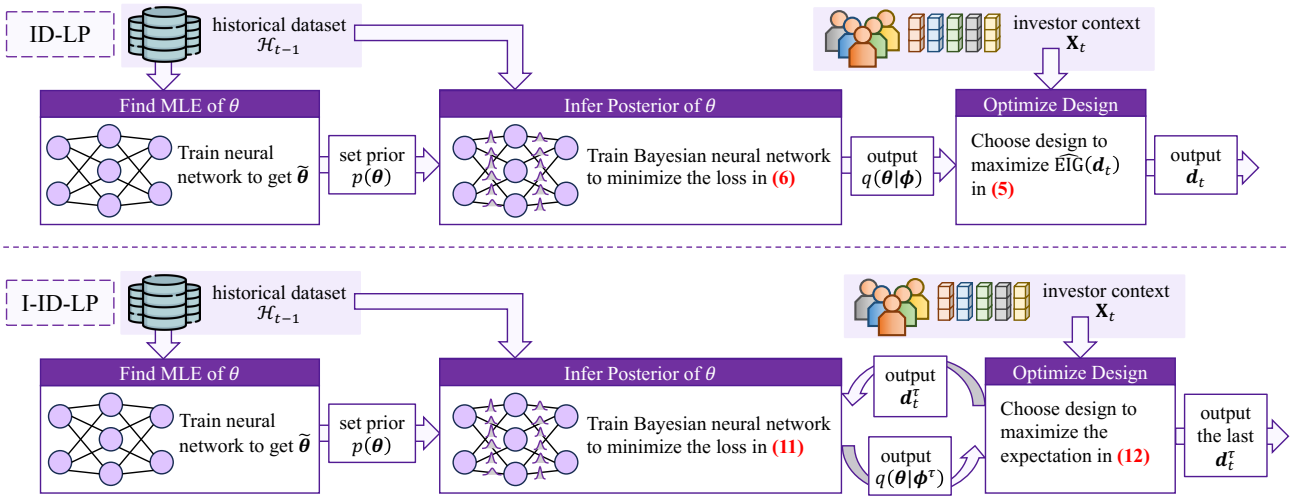

Figure 2: Illustrations of **ID-LP** and **I-ID-LP**.

5.2 Integrated Inference-and-Design with Learnable Priors

In **ID** and **ID-LP**, we conduct the inference and design separately. This leads to a sequential optimization of $\boldsymbol{\phi}$ and $\boldsymbol{d}_t$ in each period t and may degrade the performance (Lacoste-Julien, Huszár, and Ghahramani 2011; Mandi et al. 2020; Elmachtoub and Grigas 2022), because the optimization of $\boldsymbol{\phi}$ does not account for the eventual objective (maximizing the information gain from the experiment). Therefore, we propose Integrated Inference-and-Design with Learnable Priors (**I-ID-LP**) to integrate these two steps.

Joint Objective Function We derive a joint objective function with the decision variables $\boldsymbol{\phi}$ and $\boldsymbol{d}_t$. In period t, the EIG can be rearranged as a double integral over $\boldsymbol{\theta}$ and $\boldsymbol{d}_t$ (Foster et al. 2019):

$$\text{EIG}(\boldsymbol{d}_t) = \int p(\boldsymbol{\theta}|\mathcal{H}_{t-1}) g(\boldsymbol{\theta}, \boldsymbol{d}_t) d\boldsymbol{\theta}, \tag{8}$$

where function $g(\boldsymbol{\theta}, \boldsymbol{d}_t)$ is defined as

$$g(\boldsymbol{\theta}, \boldsymbol{d}_t) \triangleq \sum_{\boldsymbol{y}_t} p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{\theta}) \log \frac{p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{\theta})}{p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t)}. \tag{9}$$

Essentially, $g(\boldsymbol{\theta}, \boldsymbol{d}_t)$ is $\text{KL}\left[p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{\theta})||p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t)\right]$, which measures the difference between the two distributions. According to the property of KL divergence, $g(\boldsymbol{\theta}, \boldsymbol{d}_t)$ is always non-negative.

Changing the objective from $\text{EIG}(\boldsymbol{d}_t)$ to $\log \text{EIG}(\boldsymbol{d}_t)$ does not change the optimal $\boldsymbol{d}_t$. Next, we derive a lower bound of $\log \text{EIG}(\boldsymbol{d}_t)$ as a function of both $\boldsymbol{\phi}$ and $\boldsymbol{d}_t$:

$$\begin{aligned}
\log \text{EIG}(\boldsymbol{d}_t) &= \log \int p(\boldsymbol{\theta}|\mathcal{H}_{t-1}) g(\boldsymbol{\theta}, \boldsymbol{d}_t) d\boldsymbol{\theta} \\
&= \log \int q(\boldsymbol{\theta}|\boldsymbol{\phi}) \frac{p(\boldsymbol{\theta}|\mathcal{H}_{t-1}) g(\boldsymbol{\theta}, \boldsymbol{d}_t)}{q(\boldsymbol{\theta}|\boldsymbol{\phi})} d\boldsymbol{\theta} \\
&\geq \int q(\boldsymbol{\theta}|\boldsymbol{\phi}) \log \frac{p(\boldsymbol{\theta}|\mathcal{H}_{t-1}) g(\boldsymbol{\theta}, \boldsymbol{d}_t)}{q(\boldsymbol{\theta}|\boldsymbol{\phi})} d\boldsymbol{\theta}.
\end{aligned} \tag{10}$$

The last inequality is Jensen's inequality. Our **I-ID-LP** chooses the derived lower bound as the joint objective function, and iteratively optimizes it over $\boldsymbol{\phi}$ and $\boldsymbol{d}_t$.

Iterative Optimization Let $\boldsymbol{\phi}^\tau$ and $\boldsymbol{d}_t^\tau$ denote the decision variables obtained in the τ-th iteration ($\tau \in \{1, 2, \ldots\}$), and $\boldsymbol{\phi}^0$ and $\boldsymbol{d}_t^0$ be their initial values. In each iteration τ, we first maximize the lower bound shown in (10) over $\boldsymbol{\phi}$ while keeping $\boldsymbol{d}_t = \boldsymbol{d}_t^{\tau-1}$:

$$\begin{aligned}
\boldsymbol{\phi}^\tau &= \arg\max_{\boldsymbol{\phi}} \int q(\boldsymbol{\theta}|\boldsymbol{\phi}) \log \frac{p(\boldsymbol{\theta}|\mathcal{H}_{t-1}) g(\boldsymbol{\theta}, \boldsymbol{d}_t^{\tau-1})}{q(\boldsymbol{\theta}|\boldsymbol{\phi})} d\boldsymbol{\theta} \\
&= \arg\min_{\boldsymbol{\phi}} \text{KL}[q(\boldsymbol{\theta}|\boldsymbol{\phi})||p(\boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}[\log p(\mathcal{H}_{t-1}|\boldsymbol{\theta})] \\
&\qquad - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}[\log g(\boldsymbol{\theta}, \boldsymbol{d}_t^{\tau-1})].
\end{aligned} \tag{11}$$

The detailed derivation can be found in our supplemental material. Next, we explain the intuition behind (11). Based on the derivation in (6) and the fact that $g(\boldsymbol{\theta}, \boldsymbol{d}_t)$ is a KL divergence, (11) is equivalent to

$$\begin{aligned}
\boldsymbol{\phi}^\tau = \arg\min_{\boldsymbol{\phi}} & \text{KL}[q(\boldsymbol{\theta}|\boldsymbol{\phi})||p(\boldsymbol{\theta}|\mathcal{H}_{t-1})] \\
& - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}\left[\log \text{KL}[p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t^{\tau-1}, \boldsymbol{\theta})||p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t^{\tau-1})]\right].
\end{aligned}$$

Both terms are related to KL divergences:

- The first term is the same as the objective function in the inference step of **ID** and **ID-LP**, implying that the variational distribution $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ should be close to the actual posterior distribution $p(\boldsymbol{\theta}|\mathcal{H}_{t-1})$;
- The second term reveals the novelty of our **I-ID-LP**, implying that the inferred $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ should be consistent with the currently chosen design $\boldsymbol{d}_t^{\tau-1}$. Under the distribution $q(\boldsymbol{\theta}|\boldsymbol{\phi})$, $\boldsymbol{d}_t^{\tau-1}$ needs to be effective in gaining more information about $\boldsymbol{\theta}$: considering different $\boldsymbol{\theta}$, the gap between $p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t^{\tau-1}, \boldsymbol{\theta})$ and $p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t^{\tau-1})$ should be as large as possible.

We can learn $\boldsymbol{\phi}^\tau$ by training the corresponding Bayesian neural network on $\mathcal{H}_{t-1}$. The loss function corresponds to

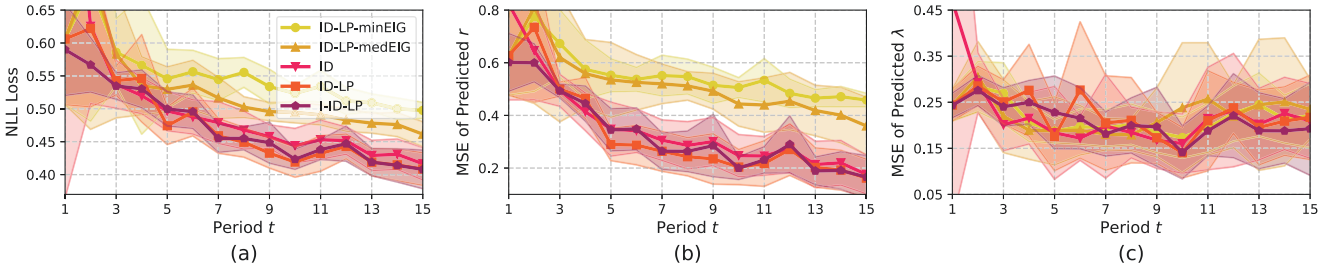

Figure 3: Comparison of ID Methods Under Setting A.

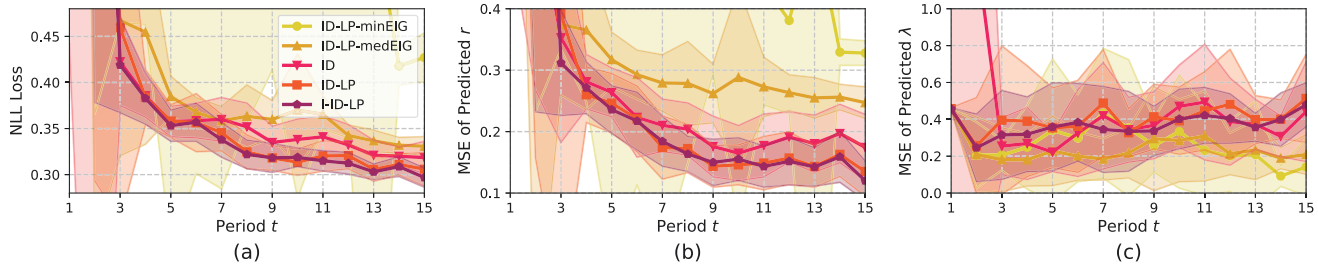

Figure 4: Comparison of ID Methods Under Setting B.

the right side of (11), and the expectations are approximated using the Monte Carlo method. Then, we maximize the lower bound shown in (10) over $\boldsymbol{d}_t$ while keeping $\boldsymbol{\phi} = \boldsymbol{\phi}^\tau$:

$$\boldsymbol{d}_t^\tau = \arg\max_{\boldsymbol{d}_t} \int q(\boldsymbol{\theta}|\boldsymbol{\phi}^\tau) \log g(\boldsymbol{\theta}, \boldsymbol{d}_t) d\boldsymbol{\theta}. \quad (12)$$

As illustrated in Figure 2, our **I-ID-LP** iteratively updates $\boldsymbol{\phi}^\tau$ and $\boldsymbol{d}_t^\tau$ based on (11) and (12). If we can optimally solve (11) and (12) in each iteration τ, the lower bound in (10) is non-decreasing in the number of iterations. In the implementation, we can terminate the iteration once the increase of this lower bound is below a predefined threshold. The design $\boldsymbol{d}_t^\tau$ obtained in the last iteration corresponds to the portfolios that the platform displays to the investors in period t. To accelerate the iteration, we can first run **ID-LP**, and then use the resulting design to initialize $\boldsymbol{d}_t^0$.

6 Experiments

6.1 Experimental Settings

We compare our **ID, ID-LP, I-ID-LP** with the following experimental design methods:

- **ID-LP-minEIG, ID-LP-medEIG:** These two methods are similar to **ID-LP**, but choose the designs with the lowest and median EIGs in each period, respectively;
- **PreEntropy:** Different from our methods, it focuses on the uncertainty in behavior prediction. It trains a $\boldsymbol{\theta}$-parameterized neural network on historical data, and chooses $\boldsymbol{d}_t$ to maximize the entropy $H[p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{\theta})]$;
- **MinMaxPro:** It trains a neural network on historical data, and then solves $\min_{\boldsymbol{d}_t} \max_{\boldsymbol{y}_t} \log p(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{d}_t, \boldsymbol{\theta})$;
- **LaplaceInfer:** It uses Laplace's method rather than variational inference to approximate the posterior distribution of $\boldsymbol{\theta}$, and chooses the design maximizing the EIG;
- **Random:** It chooses a design from $\mathcal{D}_t$ randomly;
- **MaxMean:** For each design, it computes the overall return mean of all portfolios (i.e., $\sum_k m_k$). It chooses the design with the highest overall return mean;
- **MaxVar, MaxMean+Var, MaxMean–Var:** They are similar to **MaxMean**, but choose the designs with the highest $\sum_k v_k$, $\sum_k (m_k + v_k)$, or $\sum_k (m_k - v_k)$.

We evaluate each method based on the quality of the collected behavioral dataset $\mathcal{H}_T$, measured by its effectiveness in training an accurate contextual behavioral model.

Next, we introduce the settings of $\boldsymbol{X}_t$ and $\mathcal{D}_t$. In each period t, a batch of 50 investors arrives. We generate each context vector $\boldsymbol{x} \in \mathbb{R}^{16}$ by randomly sampling its elements from the uniform distribution $\mathcal{U}[-1, 1]$. A set of designs (i.e., $\mathcal{D}_t$) is revealed in period t. We set $|\mathcal{D}_t| = 10$ and $K = 2$. For each portfolio, we randomly generate its return mean and variance as $m \sim \mathcal{U}[1, 10]$ and $v \sim \mathcal{U}[2, 10]$.

After an experimental design method shows a design from $\mathcal{D}_t$ to the investors with context $\boldsymbol{X}_t$, the investors report the choice vector $\boldsymbol{y}_t$ according to (1). We consider 5 different settings of the true $r(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$, and show setting A here.

Setting A: *We define* $c_1(\boldsymbol{x}) = \boldsymbol{w}_1^T \boldsymbol{x} + b_1 + \epsilon_1, c_2(\boldsymbol{x}) = \boldsymbol{w}_2^T \boldsymbol{x} + b_2 + \epsilon_2$. *Here,* $\boldsymbol{w}_1, \boldsymbol{w}_2, b_1, b_2$ *are randomly chosen weights, and* $\epsilon_1, \epsilon_2 \sim \mathcal{N}[0, 0.01]$ *are noises. We compute*

$$\bar{r}(\boldsymbol{x}) = \text{Sigmoid}(\max\{c_1(\boldsymbol{x}), c_2(\boldsymbol{x})\}),$$
$$\bar{\lambda}(\boldsymbol{x}) = \text{Sigmoid}(\min\{c_1(\boldsymbol{x}), c_2(\boldsymbol{x})\}).$$

We get $r(\boldsymbol{x})$ *and* $\lambda(\boldsymbol{x})$ *by normalizing* $\bar{r}(\boldsymbol{x})$ *and* $\bar{\lambda}(\boldsymbol{x})$ *across all data, respectively.*

Due to space limit, we describe the other settings in our supplemental material. For each setting, we implement different experimental design methods to collect data $\mathcal{H}_T$ for learning $r(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$. To measure the quality of $\mathcal{H}_T$, we

Method	Setting A	Setting B	Setting C	Setting D	Setting E
ID	0.416 ± 0.025	0.319 ± 0.017	0.397 ± 0.010	0.381 ± 0.013	0.391 ± 0.009
ID-LP	0.409 ± 0.025	0.306 ± 0.018	0.390 ± 0.007	0.376 ± 0.016	0.383 ± 0.010
I-ID-LP	$\mathbf{0.408 \pm 0.029}$	$\mathbf{0.297 \pm 0.010}$	$\mathbf{0.387 \pm 0.007}$	$\mathbf{0.369 \pm 0.003}$	0.383 ± 0.006
ID-LP-minEIG	0.498 ± 0.012	0.427 ± 0.026	0.565 ± 0.113	0.472 ± 0.121	0.527 ± 0.067
ID-LP-medEIG	0.461 ± 0.019	0.331 ± 0.010	0.435 ± 0.047	0.392 ± 0.038	0.416 ± 0.024
PreEntropy	0.557 ± 0.111	0.366 ± 0.028	0.634 ± 0.216	0.814 ± 0.232	0.546 ± 0.057
MinMaxPro	0.611 ± 0.122	0.366 ± 0.030	0.705 ± 0.196	0.825 ± 0.247	0.547 ± 0.066
LaplaceInfer	0.462 ± 0.026	0.327 ± 0.012	0.411 ± 0.013	0.393 ± 0.021	0.426 ± 0.039
Random	0.459 ± 0.022	0.329 ± 0.012	0.425 ± 0.035	0.393 ± 0.026	0.427 ± 0.037
MaxMean	0.468 ± 0.003	0.404 ± 0.045	0.510 ± 0.012	0.501 ± 0.004	0.523 ± 0.030
MaxVar	0.560 ± 0.008	0.340 ± 0.008	0.624 ± 0.010	0.600 ± 0.010	$\mathbf{0.381 \pm 0.002}$
MaxMean+Var	0.468 ± 0.003	0.336 ± 0.004	0.501 ± 0.003	0.491 ± 0.002	0.476 ± 0.002
MaxMean–Var	0.512 ± 0.013	0.803 ± 0.424	0.562 ± 0.063	0.559 ± 0.101	0.690 ± 0.075

Table 1: Negative Log-Likelihood Loss on Testing Dataset (20 Random Seeds).

train a separate neural network using 80% of the data from $\mathcal{H}_T$ for training and 20% for validation. This neural network outputs the predictions of $r(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$, and we train it by maximizing the log-likelihood as in (2). At last, we test the prediction performance of the trained neural network by a large testing dataset $\mathcal{H}_{\text{test}}$, using the following metrics:

- **NLL Loss:** It is the negative log-likelihood loss of predicting investor choice y in $\mathcal{H}_{\text{test}}$;
- **MSEs of Predicting $r(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$:** They are the mean squared errors between the outputs of the neural network and the actual $r(\boldsymbol{x}), \lambda(\boldsymbol{x})$ of investors in $\mathcal{H}_{\text{test}}$.

The NLL is the most important metric, as it reflects the accuracy of predicting investor behavior. Note that we train the network using $\mathcal{H}_T$, which includes labels of investor choice y but not labels of the true hidden information $r(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$. Therefore, accurately predicting $r(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$ simultaneously is challenging.

Under each setting, we repeat our experiments with 20 random seeds. For each random seed, all experimental design methods are exposed to the same sequences of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T$ and $\mathcal{D}_1, \ldots, \mathcal{D}_T$, ensuring a fair comparison. Our codes and generated data are available at: https://github.com/zhougongtao/IIDLP.

6.2 Experimental Results

Comparison Among ID Methods Figures 3 and 4 compare the quality of $\mathcal{H}_t$ ($t = 1, \ldots, 15$) collected by different ID methods under settings A and B, respectively. We measure the quality of each $\mathcal{H}_t$ using $\mathcal{H}_{\text{test}}$ and the aforementioned metrics. The shaded areas represent the standard deviations over 20 random seeds.

Figures 3 and 4 show that **I-ID-LP** collects the most informative behavioral dataset $\mathcal{H}_t$ for training a neural network to predict investor behavior. **ID-LP-minEIG** performs the worst, demonstrating the sensitivity of the quality of $\mathcal{H}_t$ to the EIG of the design. Due to space limit, we include the figures for settings C$\sim$E in the supplemental material.

Next, we compare the computational costs of these methods based on the time required to complete 15 periods under Setting A. Each method is run with 20 random seeds simultaneously on a computer equipped with an Intel Core i7-12700KF (3.6 GHz) and 32 GB of RAM. **ID-LP** is the fastest method, taking an average of 425.96 seconds per seed. The running times for **ID** and **I-ID-LP** are 2521.84 and 3720.74 seconds, respectively. Unlike **ID**, **ID-LP** involves an extra step to train a neural network for setting the prior distribution $p(\boldsymbol{\theta})$. Compared with a randomly chosen prior distribution, this $p(\boldsymbol{\theta})$ is meaningful and closer to the actual posterior distribution. As a result, it accelerates the training of the Bayesian neural network, and compensates for the computational cost of learning $p(\boldsymbol{\theta})$.

Comparison with Other Methods We show the means and standard deviations of the NLL loss in Table 1, after running different methods for 15 periods under settings A$\sim$E. Our three methods significantly outperform the baselines.

Predicting investor behavior depends on the predictions of r and λ. Due to space limit, we leave the comparison between different methods for predicting r and λ to the supplemental material.

7 Conclusion

In this paper, we proposed experimental design methods for learning general (possibly non-linear) contextual behavioral models. Using portfolio selection as an example, we showed that our methods can simultaneously learn the dependence of risk preferences and rationalities on investor contexts. To collect more informative behavioral data, we integrated variational inference with experimental design.

As discussed in Section 3.3, our work considers a system with unknown dynamics of investor arrival. One direction for extension is to address the case where this information is known and improve the sequential experimental design by considering the future arrival of investors.

Another extension direction is to evaluate experimental design methods based on the performance gains in downstream tasks (e.g., the profit generated by learning an accurate portfolio selection model) and further optimize experimental designs by accounting for these performance gains.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62202050) and the Beijing Institute of Technology Research Fund Program for Young Scholars.

The contributions of the authors are as follows. Gongtao Zhou detailed the implementation of different ID methods and baselines, designed the experimental setup, wrote and debugged all the code, conducted and analyzed experiments, and wrote the manuscript. Haoran Yu proposed the research idea, developed different ID methods, performed theoretical analysis, debugged and optimized the code, conducted and analyzed experiments, and wrote and finalized the manuscript.

References

Alsabah, H.; Capponi, A.; Ruiz Lacedelli, O.; and Stern, M. 2021. Robo-advising: Learning investors risk preferences via portfolio choices. *Journal of Financial Econometrics*, 19(2): 369–392.

Ash, J. T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; and Agarwal, A. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *Proceedings of the 8th International Conference on Learning Representations*.

Athey, S.; Blei, D.; Donnelly, R.; Ruiz, F.; and Schmidt, T. 2018. Estimating heterogeneous consumer preferences for restaurants and travel time using mobile location data. In *AEA Papers and Proceedings*, volume 108, 64–67.

Blei, D. M.; Kucukelbir, A.; and McAuliffe, J. D. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518): 859–877.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning*, 1613–1622.

Chaloner, K.; and Verdinelli, I. 1995. Bayesian experimental design: A review. *Statistical science*, 273–304.

Citovsky, G.; DeSalvo, G.; Gentile, C.; Karydas, L.; Rajagopalan, A.; Rostamizadeh, A.; and Kumar, S. 2021. Batch active learning at scale. *Advances in Neural Information Processing Systems*, 34: 11933–11944.

Cohen, M. C.; Lobel, I.; and Paes Leme, R. 2020. Feature-based dynamic pricing. *Management Science*, 66(11): 4921–4943.

Dai, M.; Jin, H.; Kou, S.; and Xu, Y. 2021. Robo-advising: A dynamic mean-variance approach. *Digital Finance*, 3(2): 81–97.

Donnelly, R.; Ruiz, F. J.; Blei, D.; and Athey, S. 2021. Counterfactual inference for consumer choice across many product categories. *Quantitative Marketing and Economics*, 1–39.

Duan, Z.; Tang, J.; Yin, Y.; Feng, Z.; Yan, X.; Zaheer, M.; and Deng, X. 2022. A context-integrated transformer-based neural network for auction design. In *International Conference on Machine Learning*, 5609–5626.

Elmachtoub, A. N.; and Grigas, P. 2022. Smart predict, then optimize. *Management Science*, 68(1): 9–26.

Foster, A.; Ivanova, D. R.; Malik, I.; and Rainforth, T. 2021. Deep adaptive design: Amortizing sequential bayesian experimental design. In *Proceedings of the 38th International Conference on Machine Learning*, 3384–3395.

Foster, A.; Jankowiak, M.; Bingham, E.; Horsfall, P.; Teh, Y. W.; Rainforth, T.; and Goodman, N. 2019. Variational Bayesian optimal experimental design. *Advances in Neural Information Processing Systems*, 32.

Frijns, B.; Koellen, E.; and Lehnert, T. 2008. On the determinants of portfolio choice. *Journal of Economic Behavior & Organization*, 66(2): 373–386.

Gal, Y.; Islam, R.; and Ghahramani, Z. 2017. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, 1183–1192.

Golrezaei, N.; Javanmard, A.; and Mirrokni, V. 2019. Dynamic incentive-aware learning: Robust pricing in contextual auctions. *Advances in Neural Information Processing Systems*, 32.

Graves, A. 2011. Practical variational inference for neural networks. *Advances in Neural Information Processing Systems*, 24.

Kleinegesse, S.; and Gutmann, M. U. 2020. Bayesian experimental design for implicit models by mutual information neural estimation. In *Proceedings of the 37th International Conference on Machine Learning*, 5316–5326.

Krishnan, R.; Subedar, M.; and Tickoo, O. 2020. Specifying weight priors in bayesian deep neural networks with empirical bayes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 4477–4484.

Lacoste-Julien, S.; Huszár, F.; and Ghahramani, Z. 2011. Approximate inference for the loss-calibrated Bayesian. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 416–424.

Lindley, D. V. 1956. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4): 986–1005.

Ling, C. K.; Fang, F.; and Kolter, J. Z. 2018. What game are we playing? End-to-end learning in normal and extensive form games. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 396–402.

Luo, Y.; Sun, W. W.; and Liu, Y. 2022. Contextual dynamic pricing with unknown noise: Explore-then-UCB strategy and improved regrets. *Advances in Neural Information Processing Systems*, 35: 37445–37457.

Mandi, J.; Stuckey, P. J.; Guns, T.; et al. 2020. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1603–1610.

Markowitz, H. 1952. Portfolio selection. *The Journal of Finance*, 7(1): 77–91.

Perivier, N.; and Goyal, V. 2022. Dynamic pricing and assortment under a contextual MNL demand. *Advances in Neural Information Processing Systems*, 35: 3461–3474.

Perrault, A.; Wilder, B.; Ewing, E.; Mate, A.; Dilkina, B.; and Tambe, M. 2020. End-to-end game-focused learning of adversary behavior in security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1378–1386.

Rainforth, T.; Cornish, R.; Yang, H.; Warrington, A.; and Wood, F. 2018. On nesting monte carlo estimators. In *Proceedings of the 35th International Conference on Machine Learning*, 4267–4276.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Gupta, B. B.; Chen, X.; and Wang, X. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9): 1–40.

Shah, V.; Johari, R.; and Blanchet, J. 2019. Semi-parametric dynamic contextual pricing. *Advances in Neural Information Processing Systems*, 32.

Wang, D.; and Shang, Y. 2014. A new active labeling method for deep learning. In *Proceedings of the International Joint Conference on Neural Networks*, 112–119.

Wang, H.; and Yu, S. 2021. Robo-advising: Enhancing investment with inverse optimization and deep reinforcement learning. In *Proceedings of IEEE International Conference on Machine Learning and Applications*, 365–372.

Werden, G. J.; Froeb, L. M.; and Tardiff, T. J. 1996. The use of the logit model in applied industrial organization. *International Journal of the Economics of Business*, 3(1): 83–105.

Xu, J.; and Wang, Y.-X. 2021. Logarithmic regret in feature-based dynamic pricing. *Advances in Neural Information Processing Systems*, 34: 13898–13910.

Zhan, X.; Wang, Q.; Huang, K. H.; Xiong, H.; Dou, D.; and Chan, A. B. 2022. A comparative survey of deep active learning. *arXiv preprint arXiv:2203.13450*.

Zheng, S.; Hayden, D.; Pacheco, J.; and Fisher III, J. W. 2020. Sequential bayesian experimental design with variable cost structure. *Advances in Neural Information Processing Systems*, 33: 4127–4137.