Gradient-Based Nonlinear Rehearsal Learning with Multivariate Alterations

Tian Qin, Tian-Zuo Wang, Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artifcial Intelligence, Nanjing University, China
{qint, wangtz, zhouzh}@lamda.nju.edu.cn

Abstract

Machine learning (ML) has made significant advancements across various domains, with a shifting focus from purely predictive tasks to decision-making. The recent proposal by Zhou (2022b) introduced a line of research known as rehearsal learning, which provides a novel perspective on modeling decision-making tasks. However, previous studies mainly focused on the linear Gaussian setting to constrain the modeling complexity. Furthermore, it has been demonstrated that finding exact optimal multivariate decisions within the sampling-based rehearsal framework is computationally infeasible in polynomial time, necessitating the development of approximate methods. In this work, we present Grad-Rh, the first gradient-based rehearsal learning method that can efficiently find multivariate decisions under nonlinear and non-Gaussian settings. We address the uncertainty in decision-making tasks using flexible and expressive conditional normalizing flow models and derive four surrogate loss functions to enable efficient gradient-based optimization. Experimental results show that Grad-Rh performs comparably to exact baselines on linear data and significantly outperforms them on nonlinear data in both decision quality and running time.

1 Introduction

Machine learning (ML) has been successfully applied in various fields, including computer vision (LeCun, Bengio, and Hinton 2015), natural language processing (Vaswani et al. 2017; Brown et al. 2020), autonomous driving (Chen et al. 2015; Chitta et al. 2023), recommender systems (Cheng et al. 2016), etc. As the predictive abilities of machine learning methods have been substantially enhanced by vast datasets, accelerated hardware, and advanced algorithms, more attention has been paid to enabling ML models to make decisions rather than merely provide predictions (Zhou 2022b).

Typically, ML models capture the *correlation* between feature variables and the label variable, which suffices for predictions in stationary distributions. However, decision-making requires more concrete variable relationships that elucidate how data are generated through concrete physical processes, so that the model could identify the appropriate variables to act upon to achieve decision goals. Researchers often turn to causal relations (Lattimore, Lattimore, and Reid 2016; Lee and Bareinboim 2018), which describe stable cause-effect relationships among variables (Pearl 2009; Imbens and Rubin 2015). However, as noted by Zhou (2022b), a comprehensive understanding of causal surroundings is not always necessary for effective decision-making, and identifying exact causal relations from data is challenging (Spirtes, Glymour, and Scheines 2000) and often computationally impractical (Chickering 1995). Additionally, real-world decision environments can be open and dynamic (Zhou 2022a), which can undermine the stability of causal relations. Thus, the concept of the *influence relation* was proposed (Zhou 2022b, 2023), which lies between *correlation* and *causation*. Influence relations can model mutually influenced variables, and take into account the dynamic and time-varying decision environments, which makes it a more natural modeling choice for decision-making tasks.

Building on the influence relation, Qin, Wang, and Zhou (2023b) introduced an effective decision-making framework known as *rehearsal learning*, aimed at avoiding undesired future events. For instance, if a machine learning model predicts a decline in future sales, a sales manager might seek to prevent it, and rehearsal learning can leverage past data to recommend effective actions to achieve that. They proposed *structural rehearsal models* (SRM) to encode influence relations among variables and describe the data-generating process in such problems. A sampling-based method is then used to execute rehearsals of potential decision outcomes given the learned SRM, from which an optimal decision could be found to achieve the decision goal. They derived an efficient solution for the case with a single decision variable and linear SRMs, but left the more practically important multi-variate nonlinear case untouched. The main challenge lies in efficiently finding effective decisions given the structural model: It has been proven that even for the simplest linear case, the sampling-based approach does not permit polynomial-time solutions if the number of decision variables is larger than one (Qin, Wang, and Zhou 2023b), which highlights the need for approximate solutions. Recently, Du et al. (2024) further considered decision costs for the linear Gaussian case. The nonlinear and non-Gaussian settings remain unexplored.

In this work, we propose the first gradient-based rehearsal learning method that achieves efficient and effective nonlinear modeling and decision-finding with multi-variate decision variables. Specifically, we extend rehearsal learning to

nonlinear and non-Gaussian settings using the conditional normalizing flow model (Papamakarios et al. 2021). The flow model can effectively capture the uncertainty in the data-generating process and drop the Gaussian assumption by transforming a given noise distribution to another that best fits the observed distribution with invertible transformations. For the complicated decision-finding step involving multiple decision variables in rehearsal learning (Qin, Wang, and Zhou 2023b), we develop an efficient approximate gradient-based solution that optimizes a surrogate loss function related to the Chebyshev center (Boyd and Vandenberghe 2004) of the desired outcome region. We propose two classes of surrogate losses: one focusing on the Chebyshev center and the other on the largest inscribed ball in the desired region. Experiments on both linear and nonlinear datasets demonstrate that our method is effective and efficient, achieving performance comparable to exact baselines on linear data and significantly better results on nonlinear data with improved scalability.

2 Background

In this section, we provide some necessary background information about the structural rehearsal model that encodes the influence relation and the normalizing flow model.

2.1 Structural Rehearsal Model

A structural rehearsal model (SRM) consists of a set of potentially time-varying rehearsal graphs and corresponding structural equations (Qin, Wang, and Zhou 2023b). The qualitative relations among variables are described by a *rehearsal graph*, where a vertex corresponds to a variable and a directed edge depicts that the downstream variables are generated by their parents. Of notice is that rehearsal graphs use bi-directional edges to connect variables that are mutually influenced. The formal definitions of rehearsal graphs are given below (Qin, Wang, and Zhou 2023b).

Definition 1 (Mixed graph)**.** Let $G = (\mathbf{V}, \mathbf{E})$ be a graph, where $\mathbf{V}$ denotes the vertices and $\mathbf{E}$ the edges. G is a *mixed graph* if for any distinct vertices $u, v \in \mathbf{V}$, there is at most one edge connecting them, and the edge is either *directional* ($u \to v$ or $u \leftarrow v$) or *bi-directional* ($u \leftrightarrow v$).

Definition 2 (Bi-directional clique)**.** A *bi-directional clique* $C = (\mathbf{V}^c, \mathbf{E}^c)$ of a mixed graph $G = (\mathbf{V}, \mathbf{E})$ is a complete subgraph induced by $\mathbf{V}^c \subseteq \mathbf{V}$ such that any edge $e \in \mathbf{E}^c$ is bi-directional. C is *maximal* if adding any other vertex does not induce a bi-directional clique.

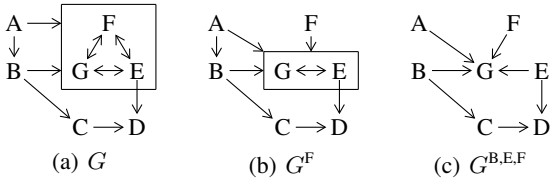
Definition 3 (Rehearsal graph)**.** Let $G = (\mathbf{V}, \mathbf{E})$ be a mixed graph. Let $\{C_i\}_{i=1}^l$ denote all maximal bi-directional cliques of G, where $C_i = (\mathbf{V}_i^c, \mathbf{E}_i^c)$. G is a *rehearsal graph* if:

1. $\mathbf{V}_i^c \cap \mathbf{V}_j^c = \emptyset$ for any $i \neq j$.
2. $\forall i \in [l]$, $u \in \mathbf{V} \setminus \mathbf{V}_i^c$, if there is any edge pointing from u to $\mathbf{V}_i^c$, then $\forall v \in \mathbf{V}_i^c$, $u \to v$.
3. The directional edges permit a topological ordering for $\{C_i\}_{i=1}^l$.

Associated with the graphical representation is a set of structural equations, which characterizes the generating process of variables quantitatively. Given a rehearsal graph G, the structural equations accompanying the rehearsal graph are defined over bi-directional cliques $\{(\mathbf{V}_i^c, \mathbf{E}_i^c)\}_{i=1}^l$. Let $\mathbf{PA}_i^G \triangleq \{u \mid \exists v \in \mathbf{V}_i^c, u \to v \text{ in } G\}$ denote the parents of $\mathbf{V}_i^c$. Assuming additive noise models, the structural equation describing the generation process of $\mathbf{V}_i^c$ is

$$\mathbf{V}_i^c := f_i(\mathbf{PA}_i^G, \boldsymbol{\varepsilon}_i; \boldsymbol{\beta}_i), \tag{1}$$

where $f_i : \mathbb{R}^{|\mathbf{PA}_i^G| + |\boldsymbol{\varepsilon}_i|} \to \mathbb{R}^{|\mathbf{V}_i^c|}$ is parameterized by $\boldsymbol{\beta}_i$ and $\boldsymbol{\varepsilon}_i$ denotes the unobserved noise.



(a) G (b) G^{F} (c) $G^{\mathrm{B,E,F}}$

Figure 1: The figure is adapted from Qin, Wang, and Zhou (2023b). (a) displays an example rehearsal graph. (b) and (c) display the alteration graph after rehearsal operations $\mathrm{Rh}(F)$ and $\mathrm{Rh}(B, E, F)$, respectively, which characterize how the data will be generated after certain alterations are performed on relevant variables.

The rehearsal model describes how the data are generated in a decision task. One can perform rehearsal operations $\mathrm{Rh}(\cdot)$ on certain variables in the model to simulate a decision, which is equivalent to altering the decision variables to certain values and removing the associated incoming edges, then generating the data following the new graph, the alteration graph, as shown in Fig. 1. Since decision-making tasks typically involve great uncertainties, properly accounting for and handling the uncertainty raised by unobserved noise is important in rehearsal learning. Specifically, the linear-Gaussian case was explored in the work of Qin, Wang, and Zhou (2023b). The full SRM allows dynamic modeling by defining both the graphs and equations over time, which accounts for possible evolutions of the environment.

2.2 Normalizing Flows

Normalizing flows are a class of generative models that allow for efficient density estimation and sampling by transforming a simple distribution into a more complex one through a series of invertible and differentiable mappings (Papamakarios 2019; Kobyzev, Prince, and Brubaker 2021; Papamakarios et al. 2021). The core idea is to start with a base distribution $p_V(v)$ (e.g., Gaussian) and apply a sequence of transformations g parameterized by θ to obtain a complex distribution $p_U(u)$. Mathematically, this is represented as:

$$u = g(v; \theta).$$

The change of variables formula allows us to compute

$$p_U(u) = p_V(v) \left| \det \frac{\partial g^{-1}(u)}{\partial u} \right|,$$

where $v = g^{-1}(u)$ and $\left| \det \frac{\partial g^{-1}(u)}{\partial u} \right|$ is the Jacobian determinant of the inverse transformation. Usually, simple invertible neural networks are sequentially connected to form the

transformation (Dinh, Krueger, and Bengio 2015; Dinh, Sohl-Dickstein, and Bengio 2017; Kingma and Dhariwal 2018).

Conditional flows extend the concept of normalizing flows to conditional distributions (Ardizzone et al. 2019; Winkler et al. 2023). In a conditional flow model, the transformation is conditioned on some external variable w, allowing for modeling $p_U(u \mid w)$. The transformation now depends on both v and w:

$$u = g(v, w; \theta).$$

The density is then given by

$$p_U(u \mid w) = p_V(v) \left| \det \frac{\partial g^{-1}(u \mid w)}{\partial u} \right|,$$

where $v = g^{-1}(u, w)$. This enables learning flexible conditional distributions and allows for properly characterizing the uncertainties in an SRM and consequently leads to more flexible rehearsal learning.

3 Problem Setup

In this section, we formulate the avoiding undesired future (AUF) problem, which is the specific decision goal that rehearsal learning tries to achieve. Previous work (Qin, Wang, and Zhou 2023b; Du et al. 2024) considered multiple decision rounds. In this work, we focus on a single-round setting and design efficient methods, which can be directly extended to multi-round settings.

Specifically, the variables observable in a decision-making task are divided into three disjoint sets $\mathbf{X} = \{X_i\}_i, \mathbf{Z} = \{Z_j\}_j$, and $\mathbf{Y} = \{Y_k\}_k$ following the time order. $\mathbf{X}$ contains contextual variables that are observed before a decision is made and $\mathbf{Y}$ denotes the outcomes that will happen after the decision. We consider decisions in the form of altering the values of certain decision variables in the intermediate set $\mathbf{Z}$, in which the variables appear after the context is observed and before the outcome is realized. For instance, after observing the sales data of last month ($\mathbf{X}$), a sales manager decides to change the discount level ($\mathbf{Z}$) to avoid the potential sales decrease in this month ($\mathbf{Y}$). We assume the variables in this problem are generated from an SRM associated with a rehearsal graph G and the structural equations in Eq. (1). The whole decision-making process is illustrated in Fig. 2.

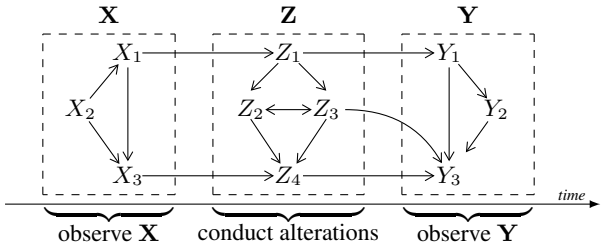


Figure 2: The rehearsal graph G describes the data-generating process qualitatively. The timeline shows the order in which the three sets of variables are generated.

Let $A = \{a_i\}_i$ and $\mathbf{Z}_A = \{\mathbf{Z}_{a_1}, \ldots, \mathbf{Z}_{a_{|A|}}\}$ denote the alterable variables in the intermediate stage and $\Delta(Z_i)$ denote the alterable range of Z_i. A decision can then be represented as $\mathbf{z}_A = (z_{a_1}, \ldots, z_{a_{|A|}})^T$ where $z_{a_i} \in \Delta(Z_{a_i})$. We focus on avoiding undesired events, i.e., the decisions should prevent $\mathbf{Y}$ from falling outside a desired set $\mathcal{S}$. The desired set is assumed to be a convex polytope, i.e.,

$$\mathcal{S} = \{\mathbf{y} \in \mathbb{R}^{|\mathbf{Y}|} \mid \mathbf{M}\mathbf{y} \preceq \mathbf{d}\}, \tag{2}$$

where $\mathbf{d} \in \mathbb{R}^q$ and $\mathbf{M} \in \mathbb{R}^{q \times |\mathbf{Y}|}$ are known parameters that describe the decision objective. For simplicity, we asssume a known graph G, which could be obtained via domain knowledge (Wang, Qin, and Zhou 2023b) or learning (Qin, Wang, and Zhou 2023a; Wang, Du, and Zhou 2024).

The AUF task is then given some historically observed data $D = \{(\mathbf{x}^i, \mathbf{z}^i, \mathbf{y}^i)\}_{i=1}^m$ and G, we want to find a decision ξ that maximizes the probability of the outcome $\mathbf{Y}$ falling into the desired set $\mathcal{S}$ after observing the context $\mathbf{X}$. Formally, we want to solve

$$\begin{aligned} \max_{\mathbf{z}_A} \quad & \mathbb{P}\left(\mathbf{Y} \in \mathcal{S} \mid \mathbf{X} = \mathbf{x}, Rh(\mathbf{Z}_A = \mathbf{z}_A)\right) \\ \text{s.t.} \quad & z_a \in \Delta(Z_a), \quad \forall a \in A. \end{aligned} \tag{3}$$

We assume $\Delta(Z_{a_i}) = [l_{a_i}, h_{a_i}]$ is a closed interval with non-empty interior. Note that the objective function in (3) cannot be evaluated since we do not have the true structural equations. Therefore, we have to learn the SRM first and then approximate the objectives. Moreover, even if we are provided with the complete SRM and a formula for computing the objective (Wang, Qin, and Zhou 2023a; Qin et al. 2024), optimizing for it could still be computationally infeasible because we allow the unknown structural equations to take nonlinear forms and do not restrict the noise distribution, which is in stark contrast to the linear setting considered in related work (Qin, Wang, and Zhou 2023b; Du et al. 2024).

4 Proposed Method

In this section, we present a gradient-based approximate method for tackling the problem in (3). The overall method can be divided into two parts. The first part is about learning an SRM from the given rehearsal graph G and observational data D by properly accounting for the uncertainty raised by unobserved noise. The second part approximates the original problem in (3) with a sampling-based unconstrained continuous optimization problem, which permits the use of efficient gradient-based optimization techniques.

4.1 Learning Structural Equations

Learning structural equations can be decomposed into learning the equations associated with each maximal bi-directional clique $\mathbf{V}_i^c$ in G, which is equivalent to learning the multi-output function f_i and the distribution of ε_i (see Eq. 1). Let $\mathbf{C}$, $\mathbf{PA}$, and $\boldsymbol{\varepsilon}$ denote a maximal bi-directional clique, its parents, and the unobserved noise, respectively. The observed data subset in D corresponding to $\mathbf{C}$ and $\mathbf{PA}$ are denoted by $\{(\mathbf{c}_i, \mathbf{pa}_i)\}_{i=1}^m$. In the following, we only need to focus on learning the structural equations on $\mathbf{C}$.

We use conditional normalizing flows to fit

$$p\left(\mathbf{c} \mid \mathbf{pa}\right),$$

which is the consequence of the structural equations and unobserved noise. Roughly, a conditional flow can be seen as a

deep neural network that takes a condition $\mathbf{pa}_i$ and a random Gaussian noise $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{|\mathbf{C}|})$ as inputs and then converts the distribution from a Gaussian to the target distribution. The network layers form an invertible and differentiable mapping that transforms the Gaussian noise to the target. The model is typically learned by minimizing the KL divergence between the target distribution and the transformed distribution. It is noteworthy that although the noise $\mathbf{u}$ is sampled from a Gaussian distribution, the flow model has strong expressiveness and can transform it into more complex ones (Hyvärinen and Pajunen 1999; Bogachev, Kolesnikov, and Medvedev 2005). Therefore, the model proposed here applies to problems with nonlinear or non-Gaussian SRMs.

Let g denote the transformation encoded by a conditional normalizing flow and $\boldsymbol{\theta}$ its parameters. The output of the flow model can be represented by

$$\hat{\mathbf{c}} = g(\mathbf{pa}, \mathbf{u}; \boldsymbol{\theta}).$$

The transformed distribution is then

$$\hat{p}(\mathbf{c} \mid \mathbf{pa}; \boldsymbol{\theta}) = p_{\mathbf{u}}(g^{-1}(\mathbf{c}, \mathbf{pa})) \left| \det \frac{\partial g^{-1}(\mathbf{c}, \mathbf{pa})}{\partial \mathbf{c}} \right|.$$

The KL divergence between the true distribution and the transformed distribution is

$$\begin{aligned} &\mathrm{KL}\left(p\left(\mathbf{c} \mid \mathbf{pa}\right) \| \, \hat{p}(\mathbf{c} \mid \mathbf{pa}; \boldsymbol{\theta})\right) \\ =& - \mathbb{E}_{p(\mathbf{c}|\mathbf{pa})} \left[\log \hat{p}(\mathbf{c} \mid \mathbf{pa}; \boldsymbol{\theta})\right] + \text{const.} \\ =& - \mathbb{E}_{p(\mathbf{c}|\mathbf{pa})} \left[\log p_{\mathbf{u}}(g^{-1}(\mathbf{c}, \mathbf{pa})) + \log J_{g^{-1}}(\mathbf{c})\right] + \text{const.} \end{aligned}$$

where $J_{g^{-1}} = \frac{\partial g^{-1}(\mathbf{c}, \mathbf{pa})}{\partial \mathbf{c}}$. The model is fitted by minimizing an empirical version of the KL divergence, i.e., minimizing

$$\ell(\boldsymbol{\theta}) = -\frac{1}{m} \sum_{i=1}^{m} \log p_{\mathbf{u}}(g^{-1}(\mathbf{c}^i, \mathbf{pa}^i)) + \log J_{g^{-1}}(\mathbf{c}^i).$$

For the design of g, we chain multiple transformations $g_1, \ldots, g_K$ and set $g = g_K \circ \cdots \circ g_1$. With a slight abuse of notations, let $\mathbf{z}^0 = \mathbf{u}$ and $\mathbf{z}^k \in \mathbb{R}^{|\mathbf{C}|}$ denote the output of g_k. Then $g_k(\mathbf{z}^{k-1}) = \mathbf{z}^k$. Let $d = \lfloor |\mathbf{C}|/2 \rfloor$ and $\mathbf{w}^k = \mathrm{concat}\left(\mathbf{z}^k_{1:d}, \mathbf{pa}\right)$ for $k = 0, \ldots, K-1$. We use a transformation based on the one introduced in Dinh, Sohl-Dickstein, and Bengio (2017) such that

$$\begin{aligned} \mathbf{z}^k_{1:d} &= \mathbf{z}^{k-1}_{1:d}, \\ \mathbf{z}^k_{d+1:|\mathbf{C}|} &= \mathbf{z}^{k-1}_{d+1:|\mathbf{C}|} \odot \exp\left(s(\mathbf{w}^k)\right) + t(\mathbf{w}^k), \end{aligned}$$

where $s(\cdot)$ and $t(\cdot)$ are two multi-layer perceptrons that map from $\mathbb{R}^d$ to $\mathbb{R}^{|\mathbf{C}|-d}$, and $\odot$ denotes the Hadamard product. In this way, g is a differentiable and invertible, and the associated Jacobian is a triangular matrix, whose determinant can be easily determined by multiplying the diagonal elements. We also perform permutation on dimensions for the output of each layer to increase the expressiveness, which is omitted in the formula for simplicity.

By fitting the conditional normalizing flows for each maximal bi-directional clique, we have an estimated SRM, which characterizes the data-generating process and can be used to generate samples that reflect the consequences of decisions. The overall procedure for learning the SRM is given in Algorithm 1, where the optimization on line 4 is done by applying standard neural network training techniques.

Algorithm 1 Learning the SRM

Input: Rehearsal graph G, dataset D
1: **for** maximal bi-directional clique $\mathbf{V}^c_i \in G$ **do**
2: $\quad D_{\mathbf{C}} = \{\boldsymbol{\xi}[\mathbf{V}^c_i] \mid \boldsymbol{\xi} \in D\}$ ▷ Data on selected variables
3: $\quad D_{\mathbf{PA}} = \{\boldsymbol{\xi}[\mathbf{PA}_i] \mid \boldsymbol{\xi} \in D\}$
4: $\quad \boldsymbol{\theta}_i \leftarrow \arg\min_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$ ▷ Calculated with $D_{\mathbf{C}}$ and $D_{\mathbf{PA}}$
5: **end for**
Output: $\{g(\cdot, \cdot; \boldsymbol{\theta}_i)\}_i$

4.2 Approximating Optimal Decisions

In this part, we try to solve the optimization problem in (3). With the learned SRM, we effectively have a generative model for how the outcomes will evolve after a decision. Recall the AUF problem in Section 3, the possible outcome after the decision can be represented by

$$\hat{\mathbf{y}} = h\left(\mathbf{z}_A; \mathbf{x}, \mathbf{U}, \boldsymbol{\Theta}\right),$$

where $\mathbf{x}$ is the observed context, $\mathbf{U} = \{\mathbf{u}_i\}_i$ is the noise, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_i$ is the parameters of the SRM, and h denotes the function that generates the outcome, which is a composition of related g_is following the alteration graph G^A. Since the distribution of $\mathbf{U}$ is known, ideally one can obtain the estimated distribution after the decision, i.e.,

$$\hat{p}\left(\mathbf{y} \mid \mathbf{x}, Rh(\mathbf{Z}_A = \mathbf{z}_A)\right).$$

However, the above distribution is implicitly encoded in the learned flow models and does not have a simple analytical form to compute with. Hence, we use a sampling-based Monte Carlo approach: We randomly sample $\mathbf{U}_1, \ldots, \mathbf{U}_n$ and use them to estimate the probability of $\hat{\mathbf{y}} \in \mathcal{S}$. We try to solve the empirical version of (3):

$$\begin{aligned} \max_{\mathbf{z}_A} \quad & \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\left(h\left(\mathbf{z}_A; \mathbf{x}, \mathbf{U}_i, \boldsymbol{\Theta}\right) \in \mathcal{S}\right) \\ \text{s.t.} \quad & l_a \le z_a \le h_a, \quad \forall a \in A, \end{aligned} \tag{4}$$

where $\mathbb{I}(\cdot)$ is the indicator function. When $|A|$ is greater than one and h is a linear function, it has been proved that the problem in (4) cannot be solved within polynomial time if P $\neq$ NP (Qin, Wang, and Zhou 2023b). Since h can take arbitrary forms that are encoded by a deep neural network, solving (4) would be more challenging: Obtaining an exact optimal solution for (4) would be computationally unrealistic. Instead, we present an approximate solution using gradient-based optimization and surrogate loss functions.

We first convert (4) into an unconstrained problem. Since $\tanh(\cdot)$ takes values between -1 and 1, by setting

$$e_a(\lambda_a) = \frac{h_a - l_a}{2} \tanh(\lambda_a) + \frac{h_a + l_a}{2}, \quad \forall a \in A,$$

we can ensure that the value is between l_a and h_a for any $\lambda_a \in \mathbb{R}$. By replacing the optimization variables $\mathbf{z}_A$ with

$\boldsymbol{\lambda} = (\lambda_{a_1}, \ldots, \lambda_{a_{|A|}})$, we can drop the constraints in (4) and form an equivalent programming:

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^{|A|}} \quad \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\left(h\left(e(\boldsymbol{\lambda}); \mathbf{x}, \mathbf{U}_i, \boldsymbol{\Theta}\right) \in \mathcal{S}\right), \tag{5}$$

where $e(\boldsymbol{\lambda}) = (e_{a_1}(\lambda_{a_1}), \ldots, e_{a_{|A|}}(\lambda_{a_{|A|}}))$. Still, the problem does not admit efficient solutions due to the discrete indicator function in the objective. In the following, we approximate the objective with differentiable surrogate losses that are connected to the Chebyshev center of $\mathcal{S}$.

The Chebyshev center of a polytope is the center of its largest inscribed ball. Let $\mathbf{o}$ denote the Chebyshev center of $\mathcal{S}$ and r denote the radius of the corresponding inscribed ball. Recall the definition of $\mathcal{S}$ in Eq. 2, $\mathbf{o}$ and r can be found by solving the following linear programming (Boyd and Vandenberghe 2004):

$$\begin{aligned} \max_{\mathbf{o}, r} \quad & r \\ \text{s.t.} \quad & r\|\mathbf{M}_i\| + \mathbf{M}_i^\top \mathbf{o} \leq d_i, \quad i = 1, \ldots, q, \end{aligned} \tag{6}$$

where the $\mathbf{M}_i^\top$ denotes the i-th row vector of $\mathbf{M}$ and d_i is the i-th element of $\mathbf{d}$. The above programming can be efficiently solved by many optimization toolboxes. The objective function in (5) aims to maximize the number of points that fall in $\mathcal{S}$. Noticing that the Chebyshev center and the corresponding inscribed ball are always inside $\mathcal{S}$, we substitute the objective function with two classes of surrogate loss functions, namely the *center-based* and the *ball-based* ones.

The center-based surrogate losses make use of the Chebyshev center, and push the points to approach the center. We provide two instances:

- Center MSE (Mean Squared Error):

$$\ell_{\text{MSE}}(\boldsymbol{\lambda}) = \frac{1}{n} \sum_{i=1}^{n} \left\|h\left(e(\boldsymbol{\lambda}); \mathbf{x}, \mathbf{U}_i, \boldsymbol{\Theta}\right) - \mathbf{o}\right\|^2$$

- Center MAE (Mean Absolute Error):

$$\ell_{\text{MAE}}(\boldsymbol{\lambda}) = \frac{1}{n} \sum_{i=1}^{n} \left\|h\left(e(\boldsymbol{\lambda}); \mathbf{x}, \mathbf{U}_i, \boldsymbol{\Theta}\right) - \mathbf{o}\right\|_1$$

The ball-based surrogate losses take the inscribed ball into account, which could be more robust since they tolerate points that are far from the center as long as the points are inside the ball. We also provide two instances:

- r-insensitive loss:

$$\ell_r(\boldsymbol{\lambda}) = \frac{1}{n} \sum_{i=1}^{n} \max\left(0, \left\|h\left(e(\boldsymbol{\lambda}); \mathbf{x}, \mathbf{U}_i, \boldsymbol{\Theta}\right) - \mathbf{o}\right\|_2 - r\right)$$

- Huber loss:

$$\ell_{\text{Huber}}(\boldsymbol{\lambda}) = \frac{1}{n} \sum_{i=1}^{n} H_r\left(\left\|h\left(e(\boldsymbol{\lambda}); \mathbf{x}, \mathbf{U}_i, \boldsymbol{\Theta}\right) - \mathbf{o}\right\|_2\right),$$

$$\text{where } H_r(a) = \begin{cases} a^2 & \text{for } |a| \leq r, \\ r \cdot (2|a| - r) & \text{otherwise.} \end{cases}$$

For points outside the ball, both of the above two losses punish them with the absolute loss. For the points that have already been in the ball, the r-insensitive loss ignores them while the Huber loss punishes them with the squared loss. If all the points coincide with the center $\mathbf{o}$, which will maximize the objective in (5), then all four surrogate loss functions will give a zero loss, which indicates that there is certain consistency between maximizing the original objective and minimizing the surrogate losses.

With a differentiable surrogate loss function $\ell_{\text{surrogate}}(\cdot)$, we approximate (5) with

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}^{|A|}} \quad \ell_{\text{surrogate}}(\boldsymbol{\lambda}). \tag{7}$$

Noticing that the programming in (7) is unconstrained and the objective is differentiable, we can efficiently solve the problem, at least find a saddle point or local minima, with gradient-based optimization techniques. We present the full method Grad-Rh in Algorithm 2: As the structural equations associated with the variables involved in the decision will not take effect in determining the outcome, we only need to learn the SRM associated with the alteration graph G^A. Then we sample n noise instances and solve the approximation in (7), finally, we transform the results back to the decision space and output the recommended decision. Although this method does not ensure optimality, it has great running efficiency by leveraging gradient-based optimization techniques.

The Grad-Rh method is quite flexible and does not have to be bundled with the conditional flow models in Algorithm 1. One can fit the SRMs with any other differentiable generative models and plug them into Algorithm 1 seamlessly. As a special case, the linear Gaussian SRM considered in Qin, Wang, and Zhou (2023b), which is obviously differentiable, can be solved by Grah-Rh as well: The $\{g(\cdot, \cdot; \boldsymbol{\theta}_i)\}_i$, which is now a linear generative model, can be learned by simply fitting linear regression models for each clique and estimating the residual variances. Then the recommended decisions can be found by running Algorithm 2 starting from line 3.

Algorithm 2 Grad-Rh

Input: Rehearsal graph G, alterable set A, dataset D, number of rehearsal samples n, observed context $\mathbf{x}$, parameter for the desired set $\mathbf{M}$ and $\mathbf{d}$

1: $G^A \leftarrow$ the alteration graph of G after a decision on A
2: $\{g(\cdot, \cdot; \boldsymbol{\theta}_i)\}_i \leftarrow$ the outputs of Algorithm 1 given G^A and D
3: $h(\cdot; \mathbf{x}, \cdot, \boldsymbol{\Theta}) \leftarrow$ the function that generates the outcome composited with $\{g(\cdot, \cdot; \boldsymbol{\theta}_i)\}_i$ given $\mathbf{x}$
4: $\mathbf{o}, r \leftarrow$ solution of the programming in (6)
5: Sample $\mathbf{U}_1, \ldots, \mathbf{U}_n$
6: $\boldsymbol{\lambda}^\star \leftarrow$ solving (7) with certain surrogate loss
7: $\mathbf{z}_A^\star \leftarrow e(\boldsymbol{\lambda}^*)$

Output: The recommended decision $\mathbf{z}_A^\star$

5 Experiments

We conducted experiments on both linear and nonlinear datasets to evaluate the performance of the Grad-Rh method against the baseline method QWZ23 (Qin, Wang, and Zhou 2023b). The baseline method gives exact solutions for the

linear Gaussian case by using the time-consuming mixed integer linear programming approach when the decision involves multiple variables, while our Grad-Rh method provides approximate solutions via gradient-based optimization, which could be very efficient. Therefore, we want to answer three questions through the experiments: (1) How large is the gap between the approximate solution generated by Grad-Rh and the exact solution? (2) How does the running time scale for both methods with an increased number of samples? (3) How is the sensitivity of the performance of Grad-Rh concerning the hyper-parameters, such as the learning rate in the optimization process and the surrogate loss function? All experiments were run on a Nvidia Tesla A100 GPU and two Intel Xeon Platinum 8358 CPUs. In the following, we give descriptions of the datasets and implementation details.

5.1 Datasets

Linear setting. We adopted the two datasets, Bermuda and Ride-hailing, used in Qin, Wang, and Zhou (2023b) and two additional synthetic datasets for the case with linear SRMs. The Bermuda dataset (Courtney et al. 2017; Andersson and Bates 2018) records some environmental variables, aiming to maintain a high net coral ecosystem calcification level. The graphical structure that generates the data is known (Courtney et al. 2017), and the true structural equations are obtained by fitting linear regression models. The ride-hailing data is abstracted from a real scenario, where variables such as the weather and traffic are considered and the goal of decision-making is to achieve a high user rating for a ride-hailing app. The synthetic datasets are generated via the following procedure: First sampling a random rehearsal graph by randomly sampling the number of variables in $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{Y}$ and connecting each pair of variables with a given probability; then we select the coefficients and bias of each linear structural equation uniformly at random between -1 and 1. We manually inspect the observational outcome distribution and select a region with a probability mass smaller than 0.1 as the desired set. The total number of variables in the two synthetic datasets is 20; the number of alterable variables is 4 and 3, respectively. The range of alterable values is set to be $[-3, 3]$.

Nonlinear setting. As we cannot find real datasets with known nonlinear SRMs, we generate four synthetic datasets. For each dataset, we randomly sample a rehearsal graph as described in the linear setting, then we instantiate the structural equations with a randomly initialized three-layer MLP (Multi-Layer Perceptron). We use the sigmoid function as the activation function. The MLP parameters are uniformly chosen from $[-2, -0.5]$ and $[0.5, 2]$. The total number of variables in the four datasets is 12, 13, 16 and 24, and the number of alterable variables is 5, 5, 6 and 4, respectively. The desired sets are regions with a probability mass below 0.1 and are manually determined by examining the observational distributions. Since the scale of the variables can differ a lot after the nonlinear mappings, for each alterable variable Z_{a_i}, the alterable range is set to $\Delta(Z_{a_i}) = [\mu - \alpha \cdot \sigma, \mu + \alpha \cdot \sigma]$, where μ and σ are respectively the empirical mean and standard deviation estimates of Z_{a_i}, and α controls the alterable range and is typically set to 1.0 or 2.0. Please refer to the supplementary materials for detailed setups.

5.2 Implementations

Grad-Rh. Our proposed gradient-based rehearsal learning method uses a unified network structure and training procedure to learn each structural equation in every nonlinear experiment setup as in Algorithm 1. Specifically, the network contains 16 blocks, each of which combines affine coupling, permutation, and global affine transformation (Ardizzone et al. 2018), to form an invertible flow. When fitting a structural equation, 70% observational data are used as the training set, and the left data are used as a validation set for early stopping. We train each model for a maximum of 1,000 epochs with the Adam optimizer (Kingma and Ba 2015). The learning rate is 0.001, and the batch size is 128. For linear experiments, training a deep neural network is not necessary, and we fit a simple network with one linear layer to reduce computational overhead. For Algorithm 2, we experiment with all four surrogate losses described in Section 4, and solving (7) with a maximum of 1,000 rounds of Adam optimizer with 0.001 learning rate.

Experiment setups. For each experiment setup, the size of initial observed dataset D is 1,000, and the number of decision runs is 100. For each decision round and each method, the number of rehearsal samples n is 1,000. After a decision is found by a method, we evaluate the decision quality (the probability that the decision successfully avoids the happening of undesired future) by sampling 10,000 trials from the true altered SRM and counting the success frequency. Since mixed-integer linear programming could be extremely time-consuming in some cases, we limit the running time of each method to 120 seconds. If a method cannot find a solution within the constraints, we take a default all-zero vector as the output. For each setup, we report the average success probability and standard deviation.

5.3 Results

Tables 1 and 2 show the success probability of the baseline and Grad-Rh with four surrogate losses in linear and nonlinear settings, respectively. Grad-Rh has a significantly shorter overall running time and maintains good decision quality. For linear settings, Grad-Rh achieves almost as good decision quality as the exact baseline approach while less than half running time in three out of four settings. For nonlinear settings, the baseline cannot perform well, and can struggle to find a solution, whereas Grad-Rh significantly improves the success probability and running time. The baseline method runs faster on the MLP4 dataset, but that comes at the cost of a significantly lower success probability, while our method achieves a good balance between decision quality and running time. In addition, the center MAE loss produces the best decisions among the surrogate losses, which could be due to its robustness against extreme values.

We further compare the relation between the decision quality and the surrogate loss, as shown in Fig. 3 (a). It shows that each surrogate loss has a monotonic relation with the success probability, which indicates the consistency between the decision objective and the loss function and justifies the approximation procedure in Grad-Rh. The scalability of Grad-Rh and baseline is illustrated in Fig. 3 (b), where Grad-Rh

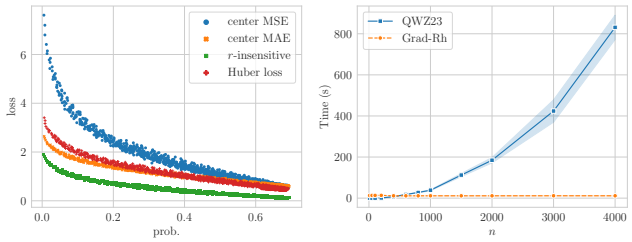
Method	Obs.	QWZ23		Center MSE		Center MAE		r-insensitive		Huber loss	
Dataset	prob.	prob.	time (s)	prob.	time (s)	prob.	time (s)	prob.	time (s)	prob.	time (s)
Bermuda	.31±.07	**.68±.01**	0.8±0.2	.67±.02	**0.2±0.1**	.67±.01	**0.2±0.1**	.66±.02	**0.2±0.1**	.66±.02	**0.2±0.1**
Traffic	.11±.11	**.63±.20**	0.2±0.2	**.63±.19**	**0.1±0.1**	.62±.19	**0.1±0.1**	**.63±.19**	**0.1±0.1**	.62±.19	**0.1±0.1**
Linear1	.09±.21	**.96±.01**	1.1±0.8	.94±.05	**0.2±0.2**	.95±.02	**0.2±0.2**	.94±.04	**0.2±0.2**	.95±.02	**0.2±0.2**
Linear2	.06±.18	**.67±.37**	**0.3±0.1**	.64±.36	0.6±0.3	.66±.36	0.6±0.2	.64±.36	0.6±0.3	.64±.36	0.6±0.3

Table 1: Results for linear SRM settings (avg. ± std.). The Obs. column shows the probability that undesired things will not happen if no actions are taken. Results with the highest average success probability or shortest running time are shown in bold.

Method	Obs.	QWZ23		Center MSE		Center MAE		r-insensitive		Huber loss	
Dataset	prob.	prob.	time (s)	prob.	time (s)	prob.	time (s)	prob.	time (s)	prob.	time (s)
MLP1	.10±.08	.00±.00	120.2±0.3	.15±.11	**7.7±0.4**	**.26±.09**	**7.7±0.4**	.17±.10	**7.7±0.4**	.18±.09	**7.7±0.3**
MLP2	.11±.03	.36±.15	42.5±8.8	.37±.18	**12.0±0.5**	**.40±.16**	**12.0±0.5**	.37±.17	12.1±0.5	.38±.17	12.1±0.5
MLP3	.14±.02	.27±.00	120.1±0.0	**.42±.16**	**12.0±0.3**	**.42±.20**	**12.0±0.3**	.41±.18	**12.0±0.3**	.41±.19	**12.0±0.3**
MLP4	.15±.15	.26±.18	**0.9±0.6**	**.44±.17**	19.8±0.7	**.44±.18**	19.9±1.1	**.44±.18**	20.1±1.2	**.44±.18**	20.0±1.0

Table 2: Results for nonlinear SRM settings (avg. ± std.). The Obs. column shows the probability that undesired things will not happen if no actions are taken. Results with the highest average success probability or shortest running time are shown in bold.

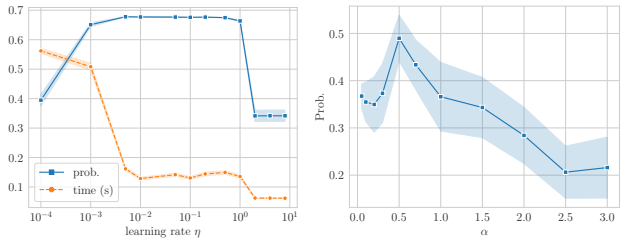
enjoys the parallel computing power of GPUs and does not significantly increase the computing time when the number of rehearsal samples increases, whereas the running time of baseline method drastically increases, showing the good scalability of our method.



(a) (b)

Figure 3: (a) The relation between the four surrogate losses of a decision evaluated on the samples generated by the fitted SRM and the probability of successfully avoiding $\mathbf{Y} \notin \mathcal{S}$ on the Bermuda dataset. (b) The relation between the running time and the number of samples on the MLP2 dataset.

We also examine the effect of the learning rate in Grad-Rh, which is shown in Fig. 4 (a). Grad-Rh works well for a wide range of learning rates and a larger learning rate corresponds to faster convergence. Fig. 4 (b) shows how the decision quality changes with the alteration range. One may conjecture that a larger alteration range corresponds to a larger decision space and therefore a method could make better decisions. But that does not hold: The decision quality decreases as the alteration range becomes winder, which we attribute to the distribution difference between observation and decision. The learned SRMs are fitted on observational data, so one can expect that the model could perform well on a range that has been observed, but the model could fail to generalize when the altered variable is set to values in an



(a) (b)

Figure 4: (a) The relation between the success probability/running time and the learning rate η on the Bermuda dataset. (b) The probability of successfully avoiding the happening of $\mathbf{Y} \notin \mathcal{S}$ changes with varying alteration range $\Delta(Z_i) = [\mu - \alpha \cdot \sigma, \mu + \alpha \cdot \sigma]$ on the MLP2 dataset.

unobserved range. So there is a tradeoff between the search space (the quality of optimal decisions) and the learnability of the model, which we think is worth more exploration in future studies.

6 Conclusion

In this work, we investigate the rehearsal learning problem, a promising paradigm for decision-making tasks. We propose the first method capable of addressing multi-variate decision-making problems for nonlinear and non-Gaussian structural rehearsal models. The proposed method, Grad-Rh, leverages the strong expressive power of invertible flow models and specifically designed surrogate loss functions to optimize decision objectives. Grad-Rh demonstrates a superior ability to identify high-quality decisions with significantly reduced computational time compared to baseline methods. We hope that this work will establish a new baseline for future research in the promising field of rehearsal learning.

Acknowledgements

This research was supported by Jiangsu Science Foundation Leading-edge Technology Program (BK20232003) and National Postdoctoral Program for Innovative Talent (BX20240162).

References

Andersson, A.; and Bates, N. 2018. In situ measurements used for coral and reef-scale calcification structural equation modeling including environmental and chemical measurements, and coral calcification rates in Bermuda from 2010 to 2012 (BEACON project). Http://lod.bco-dmo.org/id/dataset/720788.

Ardizzone, L.; Bungert, T.; Draxler, F.; Köthe, U.; Kruse, J.; Schmier, R.; and Sorrenson, P. 2018. Framework for Easily Invertible Architectures (FrEIA).

Ardizzone, L.; Lüth, C.; Kruse, J.; Rother, C.; and Köthe, U. 2019. Guided Image Generation with Conditional Invertible Neural Networks. arXiv:1907.02392.

Bogachev, V. I.; Kolesnikov, A. V.; and Medvedev, K. V. 2005. Triangular Transformations of Measures. *Sbornik: Mathematics*, 196(3): 309.

Boyd, S.; and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*.

Chen, C.; Seff, A.; Kornhauser, A. L.; and Xiao, J. 2015. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In *Proceedings of the IEEE International Conference on Computer Vision*, 2722–2730.

Cheng, H.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; Anil, R.; Haque, Z.; Hong, L.; Jain, V.; Liu, X.; and Shah, H. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 7–10.

Chickering, D. M. 1995. Learning Bayesian Networks is NP-Complete. *Learning from Data. Lecture Notes in Statistics*, 112: 121–130.

Chitta, K.; Prakash, A.; Jaeger, B.; Yu, Z.; Renz, K.; and Geiger, A. 2023. TransFuser: Imitation With Transformer-Based Sensor Fusion for Autonomous Driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11): 12878–12895.

Courtney, T. A.; Lebrato, M.; Bates, N. R.; Collins, A.; de Putron, S. J.; Garley, R.; Johnson, R.; Molinero, J.-C.; Noyes, T. J.; Sabine, C. L.; and Andersson, A. J. 2017. Environmental Controls on Modern Scleractinian Coral and Reef-scale Calcification. *Science Advances*, 3(11): e1701356.

Dinh, L.; Krueger, D.; and Bengio, Y. 2015. NICE: Nonlinear Independent Components Estimation. In *Proceedings of the 3rd International Conference on Learning Representations*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density Estimation Using Real NVP. In *Proceedings of the 5th International Conference on Learning Representations*.

Du, W.-B.; Qin, T.; Wang, T.-Z.; and Zhou, Z.-H. 2024. Avoiding Undesired Future with Minimal Cost in Non-Stationary Environments. In *Advances in Neural Information Processing Systems*.

Hyvärinen, A.; and Pajunen, P. 1999. Nonlinear Independent Component Analysis: Existence and Uniqueness Results. *Neural Networks*, 12(3): 429–439.

Imbens, G. W.; and Rubin, D. B. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*, 10236–10245.

Kobyzev, I.; Prince, S. J. D.; and Brubaker, M. A. 2021. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11): 3964–3979.

Lattimore, F.; Lattimore, T.; and Reid, M. D. 2016. Causal Bandits: Learning Good Interventions via Causal Inference. In *Advances in Neural Information Processing Systems*, 1181–1189.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature*, 521: 436–444.

Lee, S.; and Bareinboim, E. 2018. Structural Causal Bandits: Where to Intervene? In *Advances in Neural Information Processing Systems*, 2573–2583.

Papamakarios, G. 2019. Neural Density Estimation and Likelihood-free Inference. arXiv:1910.13233.

Papamakarios, G.; Nalisnick, E. T.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22: 57:1–57:64.

Pearl, J. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition.

Qin, T.; Li, L.-F.; Wang, T.-Z.; and Zhou, Z.-H. 2024. Tracking Treatment Effect Heterogeneity in Evolving Environments. *Machine Learning*, 113(6): 3653–3673.

Qin, T.; Wang, T.-Z.; and Zhou, Z.-H. 2023a. Learning Causal Structure on Mixed Data with Tree-structured Functional Models. In *Proceedings of the 23rd SIAM International Conference on Data Mining*, 613–621.

Qin, T.; Wang, T.-Z.; and Zhou, Z.-H. 2023b. Rehearsal Learning for Avoiding Undesired Future. In *Advances in Neural Information Processing Systems*, 80517–80542.

Spirtes, P.; Glymour, C.; and Scheines, R. 2000. *Causation, Prediction, and Search*. MIT press, 2nd edition.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Wang, T.-Z.; Du, W.-B.; and Zhou, Z.-H. 2024. An Efficient Maximal Ancestral Graph Listing Algorithm. In *Proceedings of the 41st International Conference on Machine Learning*.

Wang, T.-Z.; Qin, T.; and Zhou, Z.-H. 2023a. Estimating Possible Causal Effects with Latent Variables via Adjustment. In *Proceedings of the 40th International Conference on Machine Learning*, 36308–36335.

Wang, T.-Z.; Qin, T.; and Zhou, Z.-H. 2023b. Sound and Complete Causal Identification with Latent Variables Given Local Background Knowledge. *Artificial Intelligence*, 322: 103964.

Winkler, C.; Worrall, D.; Hoogeboom, E.; and Welling, M. 2023. Learning Likelihoods with Conditional Normalizing Flows. arXiv:1912.00042.

Zhou, Z.-H. 2022a. Open-environment Machine Learning. *National Science Review*, 9(8).

Zhou, Z.-H. 2022b. Rehearsal: Learning From Prediction to Decision. *Frontiers of Computer Science*, 16(4): 164352.

Zhou, Z.-H. 2023. Rehearsal: Learning from Prediction to Decision. Keynote at the CCF Conference on AI, Urumqi, China.