Reformulating Strict Monotonic Probabilities with a Generative Cost Model

Anonymous authors
Paper under double-blind review

Abstract

In numerous machine learning contexts, the relationship between input variables and predicted outputs is not only statistically significant but also strictly monotonic. Conventional approaches to ensuring monotonicity focus primarily on construction or regularization methods. This paper establishes that the problem of strict monotonic probability can be interpreted as a comparison between an observable revenue variable and a latent cost variable. This insight allows us to reformulate the original monotonicity challenge into modeling the latent cost variable and estimating its distribution. To address this issue, we introduce a generative model for the latent cost variable, called the Generative Cost Model (**GCM**), and derive a corresponding loss function. We further enhance the estimation of latent variables using variational inference, which reformulate our loss function accordingly. Lastly, we validate our approach through experiments on an artificial gamble simulation and several public datasets, demonstrating that our method significantly outperforms traditional techniques. The code of GCM is available in https://github.com/iclr-2025-4464/GCM.

1 Introduction

Many machine learning problems exhibit a monotonic relationship between inputs and outputs. Some of these relationships are statistical in nature, such as the correlation between a person's height and weight or the relationship between a company's stock price and its annual income. However, these monotonicities are often empirical and not strictly defined. In contrast, certain problems necessitate strict monotonicity, such as the relationship between equipment availability and its age, or the connection between auction winning rates and bidding prices. For these strict monotonic problems, we require a model capable of predicting strict monotonic probability based on specific input variables. We refer to these input variables as **revenue variables**, where higher revenue correlates with an increased probability of a more positive response.

The most common deep learning methods for addressing the monotonicity problem can be broadly categorized into two types (Runje & Shankaranarayana (2023)): monotonic by **construction** and by **regularization**. The construction approach maintains strict monotonicity through customized structures in deep neural networks, such as monotonic activation functions, positive weight matrices, and min-max structures (Sill (1997)). In contrast, the regularization approach promotes monotonicity by designing specific loss functions (Liu et al. (2020); Gupta et al. (2019); Sivaraman et al. (2020); Xu et al. (2024)).

Unlike traditional approaches, we propose a novel method to tackle the monotonicity problem using a **generative** framework. To estimate $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{r})$, where $\boldsymbol{y}$ is a multivariate response that maintains monotonicity with respect to the revenue variable $\boldsymbol{r}$ but is not necessarily monotonic with respect to $\boldsymbol{x}$, we employ a two-step process. (i) We simplify the multivariate problem into a Bernoulli case $p(y|\boldsymbol{x}, \boldsymbol{r})$ via variable substitution trick, so that y is reduced to binary values (0 or 1). (ii) We reformulate the monotonicity problem by defining a latent **cost variable** $\boldsymbol{c}$, such that $y = \mathbb{I}(\boldsymbol{c} \prec \boldsymbol{r}) \in \{0, 1\}$. This ensures that the monotonicity between y and $\boldsymbol{r}$ is preserved, as we have $Pr(y = 1|\boldsymbol{x}, \boldsymbol{r}) = Pr(\boldsymbol{c} \prec \boldsymbol{r}|\boldsymbol{x}, \boldsymbol{r})$. Here, $\prec$ denotes the partial order in the vector space and $\mathbb{I}$ represents the indicator function. Through this transformation, we can bypass the need to design a strictly monotonic function and instead focus on developing a generative model for the latent cost

variable $\boldsymbol{c}$. Consequently, we can use any structure to model $\boldsymbol{c}$ with the monotonicity constraints being ignored, as the monotonicity is inherently satisfied by the definition of $\boldsymbol{c}$.

To generate the latent cost variable, we propose a two-stage generative process: (i) Sampling from joint prior: In the first stage, we sample three variables $\boldsymbol{x}, \boldsymbol{r}$ and $\boldsymbol{z}$ from a joint prior $p_\theta(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z})$. Here, $\boldsymbol{x}, \boldsymbol{r}$ are observable variables, while $\boldsymbol{z}$ is a latent variable. We assume conditional independence holds: $\boldsymbol{z} \perp\!\!\!\perp \boldsymbol{r} \mid \boldsymbol{x}$. This leads to the factorization of the joint distribution as $p_\theta(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z}) = p_\theta(\boldsymbol{z}|\boldsymbol{x})p_\theta(\boldsymbol{x}, \boldsymbol{r})$. (ii) Generating the cost variable: We generate the cost variable $\boldsymbol{c}$ conditioned on $\boldsymbol{z}$ using $p_\theta(\boldsymbol{c}|\boldsymbol{z})$. This results in the joint distribution: $p_\theta(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z}, \boldsymbol{c}) = p_\theta(\boldsymbol{c}|\boldsymbol{z})p_\theta(\boldsymbol{z}|\boldsymbol{x})p_\theta(\boldsymbol{x}, \boldsymbol{r})$. By defining $y = \mathbb{I}(\boldsymbol{c} \prec \boldsymbol{r})$, we can express the evidence as: $p_\theta(\boldsymbol{x}, \boldsymbol{r}, y) = \int\int p_\theta(\boldsymbol{c}|\boldsymbol{z})p_\theta(\boldsymbol{z}|\boldsymbol{x})p_\theta(\boldsymbol{x}, \boldsymbol{r})\mathbb{I}(\boldsymbol{c} \vee_y \boldsymbol{r})d\boldsymbol{z}d\boldsymbol{c}$, where $\vee_y$ denotes $\prec$ if $y = 1$, and $\nprec$ if $y = 0$ (note that $\nprec$ is not equivalent to $\succeq$ in vector space). To simplify the model, we typically restrict our estimation of evidence to the conditional density $p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) = \int\int p_\theta(\boldsymbol{c}|\boldsymbol{z})p_\theta(\boldsymbol{z}|\boldsymbol{x})\mathbb{I}(\boldsymbol{c} \vee_y \boldsymbol{r})d\boldsymbol{z}d\boldsymbol{c}$ during the inference stage, since $\boldsymbol{x}$ and $\boldsymbol{r}$ are usually provided and we do not need to generate the entire evidence from scratch. Given that the latent variable $\boldsymbol{z}$ is high-dimensional, accurately calculating the evidence requires integration over $\boldsymbol{z}$, which can be computationally intensive. To address this, we propose two approaches to estimate the evidence: (i) Monte Carlo sampling on $\boldsymbol{z} \sim p_\theta(\boldsymbol{z}|\boldsymbol{x})$ to estimate $p_\theta(y|\boldsymbol{x}, \boldsymbol{r})$. (ii) Use variational inference to obtain a lower bound on the evidence. This allows us to optimize the log-evidence by sampling $\boldsymbol{z}$ from the recognition model $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)$. Furthermore, we choose a categorical latent variable $\boldsymbol{z}$ for a more efficient estimation of $p_\theta(y|\boldsymbol{x}, \boldsymbol{r})$.

In the last part, we conduct two types of experiments. First, we design a card gamble simulation in which the winning rate is monotonically increasing with respect to the amount of chips the player bets. The advantage of this simulation lies in its ability to reveal the true distribution of the latent cost variable $\boldsymbol{c}$, while the models are trained without observing the value of $\boldsymbol{c}$. We compare the performance of binary classification tasks between conventional methods and our generative cost model. Furthermore, we validate the predicted distribution of the latent cost variable $p_\theta(\boldsymbol{c}|\boldsymbol{x})$ by comparing it with the actual distribution. The results demonstrate that our method not only achieves superior predictive accuracy, but also effectively approximates the true distribution of the cost variable. To further assess the performance of the multivariate revenue variable $\boldsymbol{r}$, we conduct experiments on four public datasets: the Adult dataset (Becker & Kohavi (1996)), the COMPAS dataset (Larson et al. (2016)), the Diabetes dataset (Teboul) and the Blog Feedback dataset (Buza (2014)). In all four experiments, our model outperforms existing approaches. We perform several ablation studies to examine the impact of the hyperparameters and loss functions in our model, with detailed findings provided in the appendix.

The main contributions of our paper are summarized as follows:

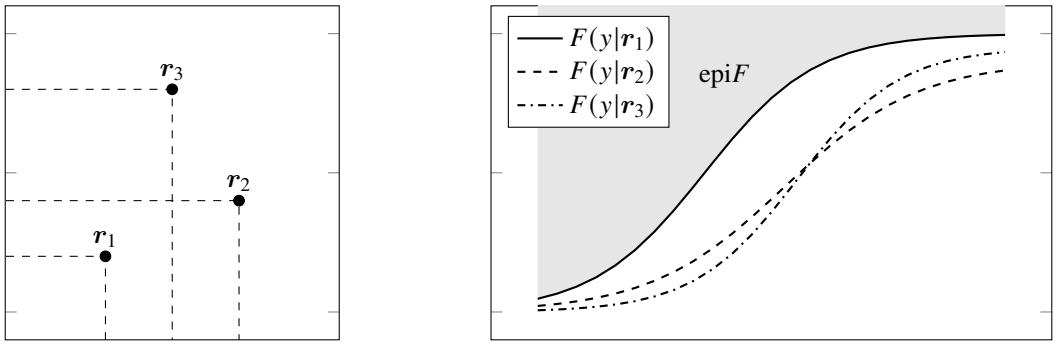
- We introduce a universal technique that reformulates the problem of monotonic probability into a modeling problem for latent cost variables, avoiding restrictions in conventional monotonic neural networks.
- We address the modeling of the cost variable using a generative approach called the Generative Cost Model (GCM), and we present loss functions derived from log-likelihood and variational inference.
- We evaluate our method for classification tasks using a simulated card gamble scenario and four public datasets, demonstrating that our model consistently outperforms traditional monotonic models.

2 Background

Partial Order between Vectors. For vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in $\mathbb{R}^n$, we define the partial order between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ as: $\boldsymbol{v}_1 \preceq \boldsymbol{v}_2$ *if and only if* $\boldsymbol{v}_1^{(k)} \leq \boldsymbol{v}_2^{(k)}$, *for any* $1 \leq k \leq n$. This relationship is illustrated in Figure 1a. Note that $\boldsymbol{v}_1 \preceq \boldsymbol{v}_2$ is equivalent to $\boldsymbol{v}_2 \succeq \boldsymbol{v}_1$.

The strict order is defined by: $\boldsymbol{v}_1 \prec \boldsymbol{v}_2$ *if and only if* $\boldsymbol{v}_1 \preceq \boldsymbol{v}_2$ *and* $\boldsymbol{v}_1 \neq \boldsymbol{v}_2$. We have $\boldsymbol{v}_1 \prec \boldsymbol{v}_2$ is equivalent to $\boldsymbol{v}_2 \succ \boldsymbol{v}_1$, but not equivalent to $\boldsymbol{v}_1 \not\succeq \boldsymbol{v}_2$.

Partial Order between Random Variables. In this paper, we adopt the definition of first-order stochastic dominance (Hadar & Russell (1969)): *for random variables* $\boldsymbol{r}_1$ *and* $\boldsymbol{r}_2$ *defined on* $\mathbb{R}^n$, *we say that* $\boldsymbol{r}_2$ *first-order stochastically dominates* $\boldsymbol{r}_1$ *(denoted* $\boldsymbol{r}_1 \prec_1 \boldsymbol{r}_2$*) if and only if* $Pr(\boldsymbol{r}_1 \succ$



(a) Example of $\boldsymbol{r}$'s, where $\boldsymbol{r}_1 \prec \boldsymbol{r}_2$ and $\boldsymbol{r}_1 \prec \boldsymbol{r}_3$.

(b) $\boldsymbol{r}_1 \prec \boldsymbol{r}_2 \Rightarrow \text{epi}F(y|\boldsymbol{r}_1) \subset \text{epi}F(y|\boldsymbol{r}_2)$ and $\boldsymbol{r}_1 \prec \boldsymbol{r}_3 \Rightarrow \text{epi}F(y|\boldsymbol{r}_1) \subset \text{epi}F(y|\boldsymbol{r}_3)$ due to the monotonicity of y and $\boldsymbol{r}$.

Figure 1: The CDFs of $F(y|\boldsymbol{r})$ with different $\boldsymbol{r}$'s, where y is monotonic with respect to $\boldsymbol{r}$.

$\boldsymbol{t}) < Pr(\boldsymbol{r}_2 \succ \boldsymbol{t})$ *for any* $\boldsymbol{t} \in \mathbb{R}^n$. Specifically, for one dimensional random variables, $r_1 \prec_1 r_2$ is equivalent to $F_1(t) > F_2(t)$ (or $\text{epi}F_1(t) \subset \text{epi}F_2(t)$) for any $t \in \mathbb{R}$. where F_i represents the cumulative distribution function (CDF) of the random variable r_i and $\text{epi}F_i$ refers to the epigraph of the CDF.

Monotonic Conditional Probability. *A conditional probability* $p(\boldsymbol{y}|\boldsymbol{r})$ *is defined as monotonic, if and only if* $\boldsymbol{y}|\boldsymbol{r}_1 \prec_1 \boldsymbol{y}|\boldsymbol{r}_2$ *for any* $\boldsymbol{r}_1 \prec \boldsymbol{r}_2$. Or, in other words, $Pr(\boldsymbol{y} \succ \boldsymbol{t}|\boldsymbol{r}_1) < Pr(\boldsymbol{y} \succ \boldsymbol{t}|\boldsymbol{r}_2)$ for any vector $\boldsymbol{t}$ and any pair $\boldsymbol{r}_1 \prec \boldsymbol{r}_2$. In this paper, we refer to the relationship between $\boldsymbol{y}$ and $\boldsymbol{r}$ as: $\boldsymbol{y}$ being (conditionally) monotonic (increasing) with respect to $\boldsymbol{r}$. All instances of monotonicity discussed here are assumed to be monotonically increasing; for decreasing relationships, we can simply replace the original variables with their opposites.

For example, if $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the mean $\boldsymbol{\mu}$ is also a random variable, then we have that $\boldsymbol{y}$ is monotonic with respect to $\boldsymbol{\mu}$. Similarly, if $y \sim \mathcal{B}ernoulli(\beta)$, then y is monotonic with respect to β. In these cases, $\boldsymbol{\mu}$ and β are referred to as monotonic parameters of $\boldsymbol{y}$.

The relationship between $\boldsymbol{r}$ and $p(y|\boldsymbol{r})$ is illustrated in Figure 1, where y is one-dimensional and monotonic with respect to $\boldsymbol{r}$. In Figure 1a, we plot three random variables $\boldsymbol{r}_1, \boldsymbol{r}_2$ and $\boldsymbol{r}_3$, with $\boldsymbol{r}_1 \prec \boldsymbol{r}_2$ and $\boldsymbol{r}_1 \prec \boldsymbol{r}_3$, while $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ are not comparable. Let $F(y|\boldsymbol{r}_i)$ denote the CDF of y conditioned on $\boldsymbol{r} = \boldsymbol{r}_i$. The corresponding conditional CDFs are plotted in Figure 1b, where $F(y|\boldsymbol{r}_1)$ is positioned at the top with the smallest epigraph, while $F(y|\boldsymbol{r}_2)$ intersects $F(y|\boldsymbol{r}_3)$ indicating the incomparability between $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$.

3 Related Work

Monotonic Modeling. In many machine learning tasks, we have the prior knowledge that the output should be monotonic with respect to certain input variables. A straightforward idea is to identify a monotonic function and optimize its parameters to approximate the desired monotonic output. The Min-Max architecture (Sill (1997)) is a pioneering work in monotonic neural networks, utilizing a piecewise linear model to approximate monotonic target functions. Its monotonicity is ensured through (i) positive weighting matrices, (ii) monotonic activation functions, and (iii) a Min-Max structure.

Along the direction of monotonic by construction, Nolte et al. (2022) introduced the Lipschitz monotonic network, which enhances robustness through weight constraint. Igel (2023) proposed the smoothed min-max monotonic network, which replaces the traditional min-max structure with a smoothed log-sum-exp function, preventing the network from becoming silent. Additionally, Runje & Shankaranarayana (2023) developed the constrained monotonic neural network, which improves the approximation of non-convex functions by modifying activation functions.

Another popular direction for improving monotonicity involves the use of regularization techniques. This includes monotonicity hints proposed by Sill & Abu-Mostafa (1996) , which utilize hint samples

to guide model learning. The certified monotonic neural networks proposed by Liu et al. (2020) certify monotonicity by verifying the lower bound of the partial derivative of monotonic features. Furthermore, Gupta et al. (2019) proposed a penalization method for negative gradients, while counter example guided methods were introduced by Sivaraman et al. (2020).

In addition, the lattice networks (Garcia & Gupta (2009)) can solve the monotonic problem by either construction or regularization approach; extensive works have been conducted in this area by Milani Fard et al. (2016), You et al. (2017), Gupta et al. (2019) and Yanagisawa et al. (2022), etc.

Monotonicity also plays an important role in many areas of machine learning. Ben-David (1995); Lee et al. (2003); van de Kamp et al. (2009); Chen & Guestrin (2016) bring monotonicity into tree models; Rashid et al. (2020) propose the QMIX method using monotonic value functions in multi-agent reinforcement learning; Lam et al. (2023) propose a multi-class loss function using monotonicity of gradients of convex functions; Haldar et al. (2020) and Xu et al. (2024) bring monotonicity into online business, etc.

Variational Inference and Generative models. Variational Inference (VI) (Peterson (1987); Parisi & Shankar (1988); Saul & Jordan (1995)) is a powerful technique for working with generative models, and recent years have seen significant advancements based on this approach (Kingma (2013); Rezende et al. (2014); Ozair & Bengio (2014); Sohl-Dickstein et al. (2015); Ho et al. (2020); Song et al. (2020)). VI transforms the complex task of Bayesian inference into a computationally manageable optimization problem by approximating the latent variables within a specified family of distributions. This is achieved by optimizing the evidence lower bound (ELB) rather than the original evidence.

Recent studies have highlighted the rapid growth of conditional generative models. In the realm of text-to-image generation, notable works include Ramesh et al. (2021), Ramesh et al. (2022), Saharia et al. (2022), and Rombach et al. (2022). For text-to-video generation, key contributions come from Esser et al. (2023) and Brooks et al. (2024). Unlike variational autoencoders (VAEs) (Kingma (2013)), which initiate generation from a latent variable, these conditional generative models begin with a pair comprising a condition (such as text, image, or video) and a latent variable. This is typically expressed through the decomposition: $p(x, z) = p(x)p(z|x)$, where x is the condition and z is the latent variable. Consequently, these models primarily focus on conditional probability $p(z|x)$. In this paper, we adopt this paradigm to construct our cost generation model.

Moreover, the normalizing flow is an important subject of generative models, it not only transforms a simple distribution to a complicated distribution, but also requires these transformations to be invertible, which is sufficient when the transformations are continuous and monotonic. There have been studies that involve monotonicity in normalizing flows: Ziegler & Rush (2019); Ho et al. (2019); Wehenkel & Louppe (2019); Müller et al. (2019); Jaini et al. (2019); Dinh et al. (2019); Ahn et al. (2022).

4 The Cost Variable Method

4.1 Problem Formulation

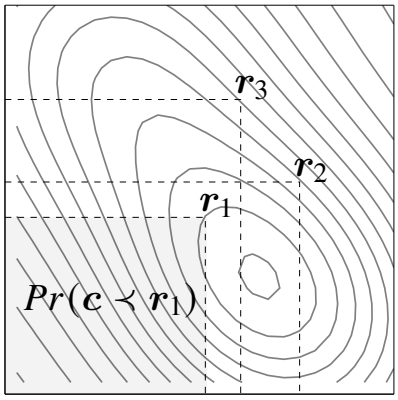
Consider a binary classification problem of $(\boldsymbol{x}, \boldsymbol{r}, y)$, where $\boldsymbol{x} \in \mathbb{R}^n$ represents the ordinary variables, $\boldsymbol{r} \in \mathbb{R}^m$ is the revenue variable, and $y \in \{0, 1\}$ is the binary output variable that exhibits monotonicity with respect to $\boldsymbol{r}$. We assume that y follows a Bernoulli distribution, with its mean parameter generated by a deep neural network $G : \mathbb{R}^n \times \mathbb{R}^m \to (0, 1)$:

$$y|\{\boldsymbol{x}, \boldsymbol{r}\} \sim \mathcal{B}ernoulli(y; G(\boldsymbol{x}, \boldsymbol{r})). \tag{1}$$

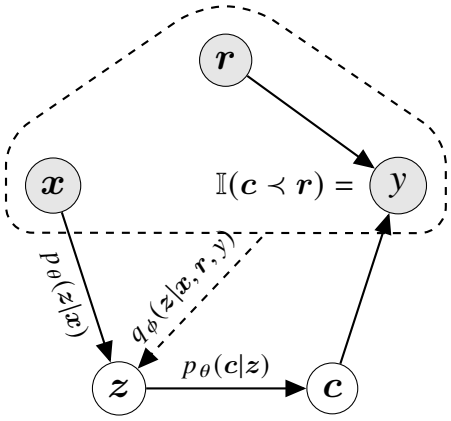
As defined in Section 2, the function G has to be monotonic with respect to $\boldsymbol{r}$. We refer to $\boldsymbol{r}$ as the **revenue variable** associated with y. The rationale is that, when y is viewed as a decision variable, a profit-maximizing decision will favor higher values of $\boldsymbol{r}$, thus ensuring the monotonicity of y with respect to $\boldsymbol{r}$.

For a general monotonic problem of $(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{y})$ with continuous multivariate output $\boldsymbol{y} \in \mathbb{R}^k$, the model takes the following form:

$$\boldsymbol{y}|\{\boldsymbol{x}, \boldsymbol{r}\} \sim \mathcal{F}(\boldsymbol{y}; \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{r})), \tag{2}$$



(a) In the density contour plot of the cost variable c, the shaded area represents the event where $c \prec r$. This indicates that the probability of a randomly selected c falling within this shaded region is given by $Pr(c \prec r) = Pr(y = 1|r)$. Therefore, for any $r_1 \prec r_2$, we can get $Pr(c \prec r_1) < Pr(c \prec r_2)$.



(b) The graph illustrates the probability graphical model for a monotonic probability $p(y|x, r)$. In this model, the grey nodes represent observable variables x, y and r, while the white nodes denote latent variables. Solid arrows indicate the generative model p_θ, whereas the dashed arrow represents the recognition model q_ϕ.

Figure 2: Definition (Figure 2a) and modeling (Figure 2b) of the latent cost variable.

where $\mathcal{F}$ denotes the chosen probability family for y. The function G produces a monotonic parameter for $\mathcal{F}$ and is monotonic with respect to r. Consequently, y maintains monotonicity with respect to r. For example, if $\mathcal{F}$ is a Gaussian distribution $\mathcal{N}(y; \mu(x, r), \mathrm{diag}(\sigma(x)^2))$ and $G = \mu(x, r)$ predicts its mean parameter, then G must be a monotonic function of r to ensure that y is monotonic with respect to r.

To reduce the general monotonic probability problem to the binary scenario, we introduce an assistant random variable t that occupies the same event space as y. We define the new response variable as $y^* = \mathbb{I}(y \succ -t) \in \{0, 1\}$, and the new revenue variable as $r^* = [r, t]$. For any $r_1^* \prec r_2^*$, we have:

$$Pr(y^* = 1|r_1^*) = Pr(y \succ -t_1|r_1) \leq Pr(y \succ -t_2|r_1) \leq Pr(y \succ -t_2|r_2) = Pr(y^* = 1|r_2^*). \quad (3)$$

The equality condition of this inequality is $r_1 = r_2$ and $t_1 = t_2$, which contradicts $r_1^* \prec r_2^*$. Thus, y^* is strictly monotonic with respect to r^*. Consequently, the triplet (y, x, r) is reduced to the binary problem of (y^*, x, r^*):

$$y^*|\{x, r^*\} \sim \mathcal{B}ernoulli(y^*; G(x, r^*)),\ \nabla_{r^*} G(x, r^*) \succ \mathbf{0}. \quad (4)$$

Lemma 1. *If y^* is monotonic with respect to r^*, then y is monotonic with respect to r.*

The proof of Lemma 1 is provided in Appendix A. This lemma establishes the equivalence between the problems of (y^*, x, r^*) and (y, x, r). From Equation 4, we derive $Pr(y \succ -t|x, r^*) = Pr(y^* = 1|x, r^*) = G(x, [r, t])$, leading to the density function of y:

$$p(y|x, r) = -\frac{\partial^k G(x, [r, -t])}{\partial t^{(1)} \cdots \partial t^{(k)}}\bigg|_{t=y}. \quad (5)$$

Which completes the transformation from a general monotonic probability problem to a binary monotonic problem. We give an example of calculating the max likelihood estimate of y as well as deriving the loss function in Appendix E.

4.2 Monotonicity via the Cost Variable

We now focus on the binary problem. The traditional approach, as defined in Equation 1, involves identifying a strictly (or weak) monotonic function $G(x, r)$ with respect to r. In this paper, instead of searching for a suitable function G, we introduce a random variable c to model y defined by:

$$y = \mathbb{I}(c \prec r). \quad (6)$$

Given that $\{\boldsymbol{c}|\boldsymbol{c} \prec \boldsymbol{r}_1\} \subset \{\boldsymbol{c}|\boldsymbol{c} \prec \boldsymbol{r}_2\}$, for any $\boldsymbol{r}_1 \prec \boldsymbol{r}_2$, it follows that $Pr(y = 1|\boldsymbol{r} = \boldsymbol{r}_1) < Pr(y = 1|\boldsymbol{r} = \boldsymbol{r}_2)$, which implies that y is strictly monotonic with respect to $\boldsymbol{r}$. Then we can define:

$$G(\boldsymbol{x}, \boldsymbol{r}) = \mathbb{E}[y|\boldsymbol{x}, \boldsymbol{r}] = Pr(\boldsymbol{c} \prec \boldsymbol{r}|\boldsymbol{x}, \boldsymbol{r}) = \int_{\boldsymbol{c} \prec \boldsymbol{r}} p(\boldsymbol{c}|\boldsymbol{x}) d\boldsymbol{c}, \tag{7}$$

demonstrating that $y \sim \mathcal{B}ernoulli(G(\boldsymbol{x}, \boldsymbol{r}))$. Thus, $G(\boldsymbol{x}, \boldsymbol{r})$ serves as the monotonic function proposed in Equation 1 . Notably, we do not need to derive the exact form of G, as long as we can estimate the conditional density $p(\boldsymbol{c}|\boldsymbol{x})$.

Unlike conventional methods that require G to be a strictly monotonic function, there are no constraints on $p(\boldsymbol{c}|\boldsymbol{x})$. We can take any form of $p(\boldsymbol{c}|\boldsymbol{x})$, and the monotonicity of $p(y|\boldsymbol{r})$ holds strictly due to the definition of y in Equation 6. We call $\boldsymbol{c}$ the **cost variable**. As illustrated in Figure 2a, the probability of y is equivalent to the probability that the revenue $\boldsymbol{r}$ domains the cost $\boldsymbol{c}$, that is, $Pr(y = 1) = Pr(\boldsymbol{c} \prec \boldsymbol{r})$. Thus, the original task of finding a monotonic function G reduces to determining the distribution of $\boldsymbol{c}$. However, since $\boldsymbol{c}$ is a latent variable, we must infer $\boldsymbol{c}$ based on the observable variables $\boldsymbol{x}$, $\boldsymbol{r}$ and y, which is a challenge that still needs to be addressed.

4.3 Generative Cost Model

As we focusing on modeling the cost variable $\boldsymbol{c}$, the distribution of $\boldsymbol{c}$ can be complicated, making it challenging to select an appropriate distribution family. To bypass the need for choosing a suitable distribution family, we adopt a generative approach that can automatically approximate complicated distributions. In this paper, we construct a simple generative model for $\boldsymbol{c}$ through the following process:

$$\begin{gathered} \lambda_z = \mathrm{DNN}_z(\boldsymbol{x}; \theta_1),\ p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{P}_z(\boldsymbol{z}; \lambda_z), \\ \lambda_c = \mathrm{DNN}_c(\boldsymbol{z}; \theta_2),\ p_{\theta_2}(\boldsymbol{c}|\boldsymbol{z}) = \mathcal{P}_c(\boldsymbol{c}; \lambda_c), \\ y = \mathbb{I}(\boldsymbol{c} \preceq \boldsymbol{r}). \end{gathered} \tag{8}$$

The generative model consists of two independent stages: $p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})$ and $p_{\theta_2}(\boldsymbol{c}|\boldsymbol{z})$, where $\theta = [\theta_1, \theta_2]$ is the generative parameter that need to be learned. In the first stage, we generate the latent variable $\boldsymbol{z}$ via $p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})$. Subsequently, the latent cost variable $\boldsymbol{c}$ is generated by $p_{\theta_2}(\boldsymbol{c}|\boldsymbol{z})$, which is set to be elementwise independent, that gives us the decomposition

$$p_{\theta_2}(y|\boldsymbol{z}, \boldsymbol{r}) = p_{\theta_2}(\boldsymbol{c} \vee_y \boldsymbol{r}|\boldsymbol{z}, \boldsymbol{r}) = 1 - y - (-1)^y \prod_i \int_{-\infty}^{\boldsymbol{r}^{(i)}} p_{\theta_2}(\boldsymbol{c}^{(i)}|\boldsymbol{z}) d\boldsymbol{c}^{(i)}. \tag{9}$$

As illustrated in Figure 2b, we assume the conditional independencies: $\boldsymbol{z} \perp\!\!\!\perp \boldsymbol{r} \mid \boldsymbol{x}$ and $\boldsymbol{x} \perp\!\!\!\perp y \mid \{\boldsymbol{z}, \boldsymbol{r}\}$ hold (we discuss another assumption in Appendix F where we abandon $\boldsymbol{z} \perp\!\!\!\perp \boldsymbol{r} \mid \boldsymbol{x}$). Thus the probability of y conditioned on $\boldsymbol{x}$ can be formulated as:

$$p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) = \int p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}) p_{\theta_2}(y|\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{r}) d\boldsymbol{z} = \int p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x}) p_{\theta_2}(y|\boldsymbol{z}, \boldsymbol{r}) d\boldsymbol{z} = \mathbb{E}_{\boldsymbol{z} \sim p_{\theta_1}} p_{\theta_2}(y|\boldsymbol{z}, \boldsymbol{r}). \tag{10}$$

To find the optimal parameter $\theta = [\theta_1, \theta_2]$, we maximize the log-likelihood (LL) of the observation y, which is $LL = \log p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) = \log \mathbb{E}_{\boldsymbol{z} \sim p_{\theta_1}(\boldsymbol{z}|x)} p_{\theta_2}(y|\boldsymbol{z}, \boldsymbol{r})$. We approximate RHS via a Monte Carlo sampling: $\mathbb{E}_{\boldsymbol{z} \sim p_{\theta_1}(\boldsymbol{z}|x)} p_{\theta_2}(y|\boldsymbol{z}, \boldsymbol{r}) \approx \frac{1}{k} \sum_{j=1}^{k} p_{\theta_2}(y|\boldsymbol{z}_j, \boldsymbol{r})$, where $\boldsymbol{z}_j \sim p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})$. Since we need to optimize both parameters θ_1 and θ_2 via gradient descent methods, it is essential that the sampling result z_j is differentiable with respect to θ_1. This is feasible for a Gaussian $p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})$ using the reparameterization trick (Kingma (2013)). However, for other assumptions of $p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})$, applying the reparameterization trick can be challenging. To address this issue, we introduce the importance sampling technique: first, we sample $\boldsymbol{z} \sim \pi$, where π is a distribution irrelevant to $\boldsymbol{x}$, usually we take π as the prior of $\boldsymbol{z}$, i.e. $\pi(\boldsymbol{z}) = p_\theta(\boldsymbol{z})$. Then by importance sampling technique, we have:

$$p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) = \mathbb{E}_{\boldsymbol{z} \sim p_\theta(\boldsymbol{z})} \frac{p_{\theta_1}(\boldsymbol{z}|x)}{p_\theta(\boldsymbol{z})} p_{\theta_2}(y|\boldsymbol{z}, \boldsymbol{r}) \approx \frac{1}{k} \sum_{j=1}^{k} \frac{p_{\theta_1}(\boldsymbol{z}_j|x)}{p_\theta(\boldsymbol{z}_j)} p_{\theta_2}(y|\boldsymbol{z}_j, \boldsymbol{r}). \tag{11}$$

The RHS includes the term $p_{\theta_1}(\boldsymbol{z}_j|x)$, which is differentiable with respect to θ_1. Therefore, we can now optimize $\theta = [\theta_1, \theta_2]$ through gradient descent methods for various form of $p_{\theta_1}(\boldsymbol{z}_j|x)$.

However, when $p_{\theta_1}(\boldsymbol{z}_j|x)$ significantly different from $p_\theta(\boldsymbol{z})$, this estimator suffers a high variance issue. To tackle this issue, we make the following efforts: i) enlarging the size of k; ii) reducing the range of $\boldsymbol{z}$ to a finite set, for example, letting $\boldsymbol{z}$ to be a categorical variable; iii) following the β-VAE (Higgins et al. (2017)), we add an regularization term $\beta D_{KL}(p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}))$ to the loss function, keeping $p_{\theta_1}(\boldsymbol{z}_j|x)$ close to $p_\theta(\boldsymbol{z})$. We put the ablation study on β in Appendix C.

Specially, when $p_{\theta_1}(\boldsymbol{z}|x)$ is a k-categorical variable, i.e. $\boldsymbol{z} \in \{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_k\}$, and we sample all possible values of $\boldsymbol{z}$ all together, we have the precise estimators:

$$\begin{aligned} \log p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) &= \log \sum_{j=1}^{k} p_{\theta_1}(\boldsymbol{z} = \boldsymbol{e}_j|\boldsymbol{x}) p_{\theta_2}(y|\boldsymbol{z} = \boldsymbol{e}_j, \boldsymbol{r}), \\ D_{KL}(p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})) &= \sum_{j=1}^{k} p_{\theta_1}(\boldsymbol{z} = \boldsymbol{e}_j|\boldsymbol{x}) \log[k \cdot p_{\theta_1}(\boldsymbol{z} = \boldsymbol{e}_j|\boldsymbol{x})]. \end{aligned} \tag{12}$$

This is a simplified form that avoids uncertainty in traditional Monte Carlo sampling. The complexity of this estimator is linearly growing with respect to the categorical number k, we do an ablation study on k in Appendix C. The details of modeling with categorical latent variable $\boldsymbol{z}$ are provided in Appendix B. The final GCM loss function is:

$$\mathcal{L}_{GCM}(\theta; \boldsymbol{x}, \boldsymbol{r}, y, \beta) = -\log p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) + \beta D_{KL}\left(p_{\theta_1}(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})\right). \tag{13}$$

4.4 Generative Cost Model with Variational Inference

A significant challenge arises from the difficulty in learning the distribution of $\boldsymbol{z}$ conditioned on $\boldsymbol{x}$, particularly since the observable variables $\boldsymbol{r}$ and y are not utilized in predicting $\boldsymbol{z}$. To improve the modeling of $\boldsymbol{z}$, we introduce the recognition model $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)$ to approximate the intractable posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)$. By applying variational inference on q_ϕ, we have the evidence lower bound formulated as:

$$ELB = \mathbb{E}_{\boldsymbol{z}\sim q_\phi} \log p_\theta(y|\boldsymbol{z}, \boldsymbol{r}) - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\|p_\theta(\boldsymbol{z}|\boldsymbol{x})\right) \leq \log p_\theta(y|\boldsymbol{x}, \boldsymbol{r}). \tag{14}$$

The derivation of ELB is available in Appendix D. Here, since the variables $\boldsymbol{x}$ and $\boldsymbol{r}$ are always given in the process of generating $\boldsymbol{c}$, the evidence $p_\theta(\boldsymbol{x}, \boldsymbol{r}, y) = p_\theta(y|\boldsymbol{x}, \boldsymbol{r})p(\boldsymbol{x}, \boldsymbol{r})$ can be reduced to $p_\theta(y|\boldsymbol{x}, \boldsymbol{r})$ as the RHS of Equation 14. We keep our assumption of $\boldsymbol{z}$ as a categorical variable, and we can simplify ELB by:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{z}\sim q_\phi} \log p_\theta(y|\boldsymbol{z}, \boldsymbol{r}) &= \sum_{j=1}^{k} q_\phi(\boldsymbol{z} = \boldsymbol{e}_j|\boldsymbol{x}, \boldsymbol{r}, y) \log p_{\theta_2}(y|\boldsymbol{z} = \boldsymbol{e}_j, \boldsymbol{r}), \\ D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\|p_\theta(\boldsymbol{z}|\boldsymbol{x})\right) &= \sum_{j=1}^{k} q_\phi(\boldsymbol{z} = \boldsymbol{e}_j|\boldsymbol{x}, \boldsymbol{r}, y) \log \frac{q_\phi(\boldsymbol{z} = \boldsymbol{e}_j|\boldsymbol{x}, \boldsymbol{r}, y)}{p_{\theta_1}(\boldsymbol{z} = \boldsymbol{e}_j|\boldsymbol{x})}. \end{aligned} \tag{15}$$

Direct optimization of ELB is problematic. Unlike classic VI, which has a KL term formed as $D_{KL}(q(z|x)\|p(z))$ with a fixed prior $p(z)$ that makes the optimization of $q(z|x)$ stable, both $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)$ and $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ in our KL term need to be learned, which could bring uncertainty in the optimization of q_ϕ. Fortunately, we can learn $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ by optimizing $\mathcal{L}_{GCM}$, so we combine ELB and $\mathcal{L}_{GCM}$ as the objective of the variational version of GCM (GCM-VI):

$$\mathcal{L}_{GCM-VI}(\theta, \phi; \boldsymbol{x}, \boldsymbol{r}, y, \beta) = \mathcal{L}_{GCM}(\theta; \boldsymbol{x}, \boldsymbol{r}, y, \beta) - \alpha ELB. \tag{16}$$

In this loss function, we stop the gradient of $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ in ELB and train $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ via $\mathcal{L}_{GCM-VI}$. Ablation studies for values of α and for whether to combine $\mathcal{L}_{GCM}$ and ELB are available in the Appendix C.

5 Experiment

5.1 A Gamble Simulation

We design a card gamble and the rules are listed as follows:

- There are n cards, each labeled with a number from 1 to n. The cards are shuffled and then the backsides are also labeled with numbers from 1 to n.
- In each round, the dealer shuffles the cards and then the player picks l cards from the top of the deck. The player sees the front sides of the selected cards and places a bet of r chips, where $r < n$.
- The dealer rolls a dice to select one card from the l selected cards. If the backside number of this card is less than r, the player wins and receives n chips as a prize, producing a net profit of $n - r$, otherwise the player loses, resulting in a profit of $-r$.
- In the whole game, a player can only see the front side, but not the backside of all cards.

In our gambling model, the rules state that the more chips a player bets, the higher the likelihood of winning, but correspondingly the prize of winning shrinks. We denote the winning event as $y = 1$ and the losing as $y = 0$. The selected cards of the player are represented as $\boldsymbol{x} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_l]$, where each $\boldsymbol{x}_i$ corresponds to an embedding of the i th card. Consequently, the probability $p(y|\boldsymbol{x}, r)$ is strictly monotonic with respect to the bet r. We train our generative cost model on a simulated dataset and evaluate the performance of our model $p_\theta(y|\boldsymbol{x}, r)$ using the same strategy. To assess the prize-winning capability of the models, we determine the optimal bet of model p_θ is: $r^* = \text{argmax}_r\{p_\theta(y|\boldsymbol{x}, r)n - r.\}$ The real profit generated by the choice r^* is $\mathbb{I}(r^* > c)n - r^*$. To maximize the total profit, a model has to learn the probability $p_\theta(y|\boldsymbol{x}, r)$ accurately for every combinations of $\boldsymbol{x}$ and r.

The cost variable c corresponds to a random choice of the unobservable values on the backsides of the picked cards $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_l$, and we note these backside values as $b_1, \cdots, b_l$. As a result, the model should infer the probabilities of the backside value of each $\boldsymbol{x}_i$. This inference is particularly challenging, as the models can only deduce these probabilities from training samples consisting of $(\boldsymbol{x}, r, y)$. In particular, the optimal solution for the generative cost model is to learn a precise mapping from $\boldsymbol{x}$ to $p(c|\boldsymbol{x})$, which is given by:

$$p(c|\boldsymbol{x}) = \frac{\mathbb{I}(c \in \{b_1, \cdots, b_l\})}{l} \tag{17}$$

We compare several classic methods to our generative model, all of which share the same baseline architecture: a two-layer perceptron network. This structure includes layer normalization (Ba (2016)), residual connection (He et al. (2016)) and leaky-ReLU activation (Xu (2015)). During the training phase, we employ the classic stochastic gradient descent method (Robbins & Monro (1951); Kiefer & Wolfowitz (1952) and Rumelhart et al. (1986)) to optimize network parameters. The model is trained on simulated data derived from the card game we designed, with hyperparameters set to $n = 10,000$ and $l = 4$. We assume that r is generated independently of $\boldsymbol{x}$. We train our model with mini-batches of size 100 in 50,000 rounds, resulting in a total of 5,000,000 training examples, while the methods are tested on 100,000 examples.

The methods we compare include: (i) the baseline MLP network (MLP); (ii) Min-Max network (MM) (Sill (1997)); (iii) smoothed Min-Max network (SMM) (Igel (2023)); (iv) constrained monotonic network (CMNN); (v) lattice network (Lattice) (Milani Fard et al. (2016)); (vi) monotonicity hint model (Hint) (Sill & Abu-Mostafa (1996)); (vii) pointwise loss method (PWL) (Gupta et al. (2019)). Note that the MLP method does not require monotonicity, it does not face the difficulties in strict monotonic structure designing as other methods. Here we regard it as a benchmark of a free-style model but not a baseline of the monotonic modeling family. Moreover, the Hint and PWL methods are weak monotonic methods which encourage but not assure monotonicity.

We evaluate these methods using the following metrics: (i) the area under the precision-recall curve (AUC) between $p_\theta(y|\boldsymbol{x}, r)$ and y; (ii) the Kullback-Leibler (KL) divergence between $p_\theta(y|\boldsymbol{x}, r)$ and the true $p(y|\boldsymbol{x}, r)$; (iii) Kendall's τ coefficient, calculated between multiple pairs of $p_\theta(y|\boldsymbol{x}, r)$ and r with fixed $\boldsymbol{x}$, for validating models' monotonicity; (iv) the prize money earned by each model.

In our experiments, we evaluate the two proposed methods: the Generative Cost Model (GCM) and its variational inference counterpart (GCM-VI). For the GCM, we utilize a categorical latent variable $\boldsymbol{z}$ and estimate the likelihood using the detection trick outlined in Equation 11. In the case of GCM-VI, we extend the GCM framework by incorporating a recognition network $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)$ that runs in parallel with the generative model $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Both $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)$ and $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ are categorical distributions, which simplifies the calculation of their KL divergence. This leads to the formulation

Table 1: Experimental results (with a 95% confidence interval) for the simulated card game.

Method	**AUC↑**	**KL Div.↓**	**Kendall's τ↑**	**Prize Profit↑**
MLP	0.8803 ±0.0006	0.0630 ±0.0012	0.8989 ±0.0042	1053.7 ±24.9
MM	0.8844 ±0.0012	0.0578 ±0.0033	**1** ±0	1251.5 ±68.1
SMM	0.8824 ±0.0031	0.0629 ±0.0072	**1** ±0	1104.6 ±130.6
CMNN	0.8823 ±0.0013	0.0624 ±0.0029	**1** ±0	1025.1 ±35.0
Lattice	0.8772 ±0.0002	0.0743 ±0.0003	**1** ±0	884.0 ±3.2
Hint	0.8850 ±0.0013	0.0585 ±0.0028	0.9499 ±0.0027	1164.1 ±71.0
PWL	0.8879 ±0.0013	0.0526 ±0.0036	**1** ±0	1355.9 ±91.4
GCM	0.8917 ±0.0005	0.0395 ±0.0019	**1** ±0	1699.2 ±48.1
GCM VI	**0.8920** ±0.0007	**0.0391** ±0.0014	**1** ±0	**1709.9** ±46.8

of the evidence lower bound, as presented in Equation 14. Detailed descriptions of the GCM and GCM-VI models can be found in the Appendix B. The experimental results comparing our models with other methods are summarized in Table 1.

As shown in Table 1, our experiments demonstrate that the Generative Cost Model (GCM) achieves superior performance compared to all other monotonic methods. Notably, the performance on Kendall's τ coefficient meets our expectations, as these models ensure strict monotonicity; the only exceptions are the MLP model and the Hint model, which fail to predict monotonic results since their architecture do not assure strict monotonicity.

5.2 Validate on Generation of Cost Variable

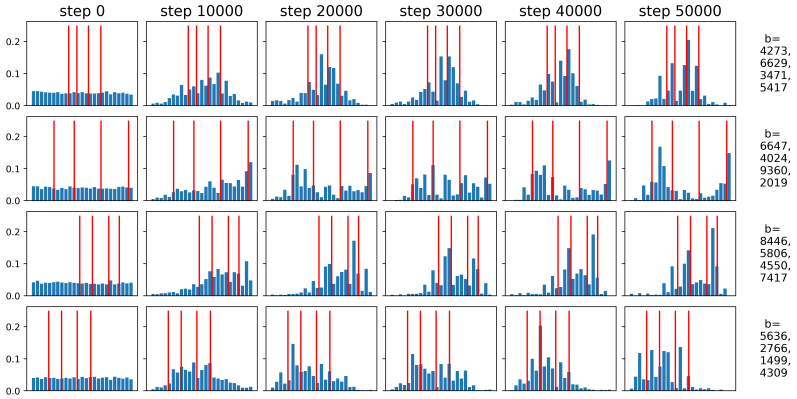


Figure 3: The predicted distribution of $p_\theta(c|\boldsymbol{x})$ (histogram in blue) is compared to the actual distribution of c (represented by the red lines). In each row, we fix the variable $\boldsymbol{x}$ and its corresponding cost variable $\boldsymbol{c}$. As the training steps progress from left to right, the predicted distribution increasingly converges to the real distribution.

Since our model focuses on modeling the distribution of the latent cost variable c, we can leverage the actual distribution of c formulated in Equation 17. During the training process, we record the prediction of $p_\theta(c|\boldsymbol{x})$ using an importance sampling method similar to Equation 11. As shown in Figure 3, the predicted density of c is increasingly aligned with the actual distribution as training progresses. This observation confirms that our generative cost model effectively learns the latent cost variable.

5.3 Experiments for Multidimensional Revenue on Public Datasets

To further evaluate the GCM model for multidimensional revenue variable, we use four public datasets: the Adult dataset (Becker & Kohavi (1996)), the COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) dataset (Larson et al. (2016)), the Diabetes dataset

(Teboul) and the Blog Feedback dataset (Buza (2014)). The property of each dataset is shown in Table 2 .

Table 2: Details of the datasets.

dataset	samples	ordinary features	monotonic features	target
Adult	48,842	33	4	classification
COMPAS	7,214	9	4	classification
Diabetes	253,680	105	4	classification
Blog Feedback	52,397	272	8	regression

The model we test are the same as we presented in Section 5.1, while the evaluation metrics are switched to loss, RMSE, AUC and ACC. And, as we stated in Section 5.1, we regard the MLP model as a benchmark of a freestyle model but not a baseline of the monotonic modeling family. For all four datasets, the training and testing sets are split in a 4:1 ratio. We follow the data preprocessing procedures outlined by Liu et al. (2020) for the COMPAS dataset. For the Blog Feedback dataset, we perform a logarithm transformation for numerical features and target value. In all experiments, we employ categorical distributions as the family of latent $\boldsymbol{z}$ in the GCM and GCM-VI, the hyperparameter settings of GCM and GCM-VI are listed in the Appendix G. The testing results are demonstrated in Table 3 and the full results are available in Appendix G. All experiments are repeated five times with different random seeds, the final results are reported with a 95% confidence interval.

Table 3: Experimental results on the multiple datasets.

Method	Adult AUC↑	COMPAS AUC↑	Diabetes AUC↑	Blog Feedback RMSE↓
MLP $*$	0.7818 ±0.0037	0.7385 ±0.0026	0.8241 ±0.0015	0.1048 ±0.0008
MM	0.7788 ±0.0047	0.7387 ±0.0040	0.8210 ±0.0065	0.1124 ±0.0023
SMM	0.7798 ±0.0059	0.7385 ±0.0022	0.8227 ±0.0011	0.1149 ±0.0025
CMNN	0.7761 ±0.0073	0.7374 ±0.0015	0.8204 ±0.0011	0.1123 ±0.0022
Lattice	0.7841 ±0.0016	0.7388 ±0.0032	0.8142 ±0.0002	0.2465 ±0.0025
Hint ♯	0.7804 ±0.0043	0.7398 ±0.0045	0.8244 ±0.0018	0.1046 ±0.0003
PWL ♯	0.7814 ±0.0032	0.7409 ±0.0037	0.8223 ±0.0017	0.1067 ±0.0008
GCM	0.7840 ±0.0014	**0.7449** ±0.0017	0.8251 ±0.0011	0.0994 ±0.0015
GCM VI	**0.7863** ±0.0029	0.7448 ±0.0025	**0.8256** ±0.0007	**0.0990** ±0.0010

$*$: No monotonicity requirements.
♯: Weak monotonicity via regularization.

Our GCM and GCM-VI models achieve the top two performances in all metrics in all datasets after $10,000$ training steps. Notably, GCM-VI achieves the best performance on all datasets except the COMPAS dataset, proving the effectiveness of introducing variational bound into our generative objective. The detailed results are available in the Appendix G.3. And a time complexity analysis is available in the Appendix H.

6 Conclusion

This paper presents an innovative generative method for monotonic modeling by reformulating the monotonicity problem through the incorporation of a latent cost variable. We have developed a robust generation process for this cost variable that accurately approximates the latent costs. Our experimental results demonstrate that the proposed Generative Cost Model (GCM) effectively addresses the monotonicity challenge, significantly outperforming traditional approaches across various datasets.

References

Byeongkeun Ahn, Chiyoon Kim, Youngjoon Hong, and Hyunwoo J Kim. Invertible monotone operators for normalizing flows. *Advances in Neural Information Processing Systems*, 35:16836–16848, 2022.

Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Arie Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19:29–43, 1995.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

Krisztian Buza. BlogFeedback. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C58S3F.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Laurent Dinh, Jascha Sohl-Dickstein, Hugo Larochelle, and Razvan Pascanu. A rad approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7346–7356, 2023.

Eric Garcia and Maya Gupta. Lattice regression. *Advances in Neural Information Processing Systems*, 22, 2009.

Akhil Gupta, Naman Shukla, Lavanya Marla, Arinbjörn Kolbeinsson, and Kartik Yellepeddi. How to incorporate monotonicity in deep networks while preserving flexibility? *arXiv preprint arXiv:1909.10662*, 2019.

Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American economic review*, 59(1):25–34, 1969.

Malay Haldar, Prashant Ramanathan, Tyler Sax, Mustafa Abdool, Lanbo Zhang, Aamir Mansawala, Shulin Yang, Bradley Turnbull, and Junshuo Liao. Improving deep learning for airbnb search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2822–2830, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International conference on machine learning*, pp. 2722–2730. PMLR, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Christian Igel. Smooth min-max monotonic networks. *arXiv preprint arXiv:2306.01147*, 2023.

Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pp. 3009–3018. PMLR, 2019.

Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pp. 462–466, 1952.

Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kevin H Lam, Christian Walder, Spiridon Penev, and Richard Nock. Legendretron: uprising proper multiclass loss learning. In *International Conference on Machine Learning*, pp. 18454–18470. PMLR, 2023.

Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm, 2016. URL https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

John WT Lee, Daniel S Yeung, and Xizhao Wang. Monotonic decision tree for ordinal classification. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, volume 3, pp. 2623–2628. IEEE, 2003.

Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. *Advances in Neural Information Processing Systems*, 33:15427–15438, 2020.

Mahdi Milani Fard, Kevin Canini, Andrew Cotter, Jan Pfeifer, and Maya Gupta. Fast and flexible monotonic functions with ensembles of lattices. *Advances in neural information processing systems*, 29, 2016.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (ToG)*, 38(5):1–19, 2019.

Niklas Nolte, Ouail Kitouni, and Mike Williams. Expressive monotonic neural networks. In *The Eleventh International Conference on Learning Representations*, 2022.

Sherjil Ozair and Yoshua Bengio. Deep directed generative autoencoders. *arXiv preprint arXiv:1410.0630*, 2014.

Giorgio Parisi and Ramamurti Shankar. Statistical field theory. 1988.

Carsten Peterson. A mean field theory learning algorithm for neural network. *Complex systems*, 1: 995–1019, 1987.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Davor Runje and Sharath M Shankaranarayana. Constrained monotonic neural networks. In *International Conference on Machine Learning*, pp. 29338–29353. PMLR, 2023.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Lawrence Saul and Michael Jordan. Exploiting tractable substructures in intractable networks. *Advances in neural information processing systems*, 8, 1995.

Joseph Sill. Monotonic networks. *Advances in neural information processing systems*, 10, 1997.

Joseph Sill and Yaser Abu-Mostafa. Monotonicity hints. *Advances in neural information processing systems*, 9, 1996.

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, and Guy Van den Broeck. Counterexample-guided learning of monotonic neural networks. *Advances in Neural Information Processing Systems*, 33:11936–11948, 2020.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Alex Teboul. Diabetes health indicators dataset. URL https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset.

Rémon van de Kamp, Ad Feelders, and Nicola Barile. Isotonic classification trees. In *Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31-September 2, 2009. Proceedings 8*, pp. 405–416. Springer, 2009.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in neural information processing systems*, 32, 2019.

Bing Xu. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Xiaoxiao Xu, Hao Wu, Wenhui Yu, Lantao Hu, Peng Jiang, and Kun Gai. Enhancing interpretability and effectiveness in recommendation with numerical features via learning to contrast the counterfactual samples. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 453–460, 2024.

Hiroki Yanagisawa, Kohei Miyaguchi, and Takayuki Katsuki. Hierarchical lattice layer for partially monotone neural networks. *Advances in Neural Information Processing Systems*, 35:11092–11103, 2022.

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. *Advances in neural information processing systems*, 30, 2017.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pp. 7673–7682. PMLR, 2019.

A Proof of Lemma 1

Proof. For any $\boldsymbol{r}_1 \prec \boldsymbol{r}_2$, we have $\boldsymbol{r}_1^* = [\boldsymbol{r}_1, \boldsymbol{t}] \prec [\boldsymbol{r}_2, \boldsymbol{t}] = \boldsymbol{r}_2^*$.

Since y^* is monotonic with respect to $\boldsymbol{r}^*$, we have $Pr(y^* = 1|\boldsymbol{x}, \boldsymbol{r}_1^*) < Pr(y^* = 1|\boldsymbol{x}, \boldsymbol{r}_2^*)$ for any $\boldsymbol{t}$.

Therefore, by the definition of y^*, we have $Pr(\boldsymbol{y} \succ -\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{r}_1, \boldsymbol{t}) < Pr(\boldsymbol{y} \succ -\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{r}_2, \boldsymbol{t})$ for any $\boldsymbol{t}$.

Thus, $\boldsymbol{y}|\{\boldsymbol{x}, \boldsymbol{r}_1\} \prec_1 \boldsymbol{y}|\{\boldsymbol{x}, \boldsymbol{r}_2\}$, i.e. $\boldsymbol{y}$ is monotonic with respect to $\boldsymbol{r}$. □

B Details of GCM and GCM-VI in the Experiments

The generative model with categorical latent variable $\boldsymbol{z}$ is designed by:

$$\begin{aligned} &w^{(1)}, \cdots, w^{(k)}, \boldsymbol{d} = \mathrm{DNN}_z(\boldsymbol{x}; \theta_1), \\ &\boldsymbol{z} \sim Categorical(w^{(1)}, \cdots, w^{(k)}), \\ &\boldsymbol{h} = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{d}, \\ &\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c = \mathrm{DNN}_c(\boldsymbol{h}; \theta_2), \\ &\boldsymbol{c} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2), \\ &y = \mathbb{I}(\boldsymbol{c} \preceq \boldsymbol{r}). \end{aligned} \tag{18}$$

The second line produces a one-hot vector $\boldsymbol{z}_1$, thus, $\boldsymbol{z}$ is not differentiated with respect to θ_1. But through Equation 11, we can skip the sampling of $\boldsymbol{z}$ and modify the generative model into:

$$\begin{aligned} &\tilde{\boldsymbol{z}}^{(j)} = \boldsymbol{e}^{(j)}, j = 1, \cdots, k \\ &\tilde{\boldsymbol{h}}^{(j)} = \boldsymbol{A}\tilde{\boldsymbol{z}}^{(j)} + \boldsymbol{d}, \\ &\tilde{\boldsymbol{\mu}}_c^{(j)}, \tilde{\boldsymbol{\sigma}}_c^{(j)} = \mathrm{DNN}_c(\tilde{\boldsymbol{h}}^{(j)}; \theta_2), \\ &\tilde{\boldsymbol{c}}^{(j)} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_c^{(j)}, \tilde{\boldsymbol{\sigma}}_c^{(j)2}), \\ &\tilde{y} = \mathbb{I}(\tilde{\boldsymbol{c}}^{(j)} \preceq \boldsymbol{r}), \end{aligned} \tag{19}$$

where $\boldsymbol{e}^{(j)}$ is a one-hot vector having j-th element equals 1, and $p(\boldsymbol{z} = \boldsymbol{e}^{(j)}|\boldsymbol{x}) = w^{(j)}$. Then we can estimate the probability of y by importance sampling:

$$p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) = \sum_{j=1}^{k} p(\boldsymbol{z} = \boldsymbol{e}^{(j)}|\boldsymbol{x}) p(y|\boldsymbol{z} = \boldsymbol{e}^{(j)}, \boldsymbol{r}) = \sum_{j=1}^{k} w^{(j)} \left(1 - y - (-1)^y \Phi((\boldsymbol{r} - \tilde{\boldsymbol{\mu}}_c^{(j)})/\tilde{\boldsymbol{\sigma}}_c^{(j)})\right), \tag{20}$$

since $w^{(j)}$ is differentiable with respect to θ_1 and $(\tilde{\boldsymbol{\mu}}_c^{(j)}, \tilde{\boldsymbol{\sigma}}_c^{(j)})$ is differentiable with respect to θ_2, we can optimize $\theta = [\theta_1, \theta_2]$ through a gradient-based optimization.

GCM-VI shares the whole structure of GCM, but has an additional recognition structure:

$$\begin{aligned} &\hat{w}^{(1)}, \cdots, \hat{w}^{(k)} = \mathrm{DNN}_q(\boldsymbol{x}, \boldsymbol{r}, y; \theta_3), \\ &\hat{\boldsymbol{z}} \sim Categorical(w^{(1)}, \cdots, w^{(k)}), \\ &\hat{\boldsymbol{h}} = \boldsymbol{A}\hat{\boldsymbol{z}} + \boldsymbol{d}, \end{aligned} \tag{21}$$

where $\boldsymbol{d}$ shares the same vector in Equation 19. The KL divergence between generation and recognition models is given by:

$$D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y) \| p_\theta(\boldsymbol{z}|\boldsymbol{x})\right) = \sum_{j=1}^{k} \hat{w}^{(j)} \left(\log \hat{w}^{(j)} - \log w^{(j)}\right). \tag{22}$$

In our experiment, we stop the gradient of $\log w^{(j)}$ when computing the KL divergence. So the evidence lower bound becomes:

$$ELB = \sum_{j=1}^{k} \hat{w}^{(j)} \log \left(1 - y - (-1)^y \Phi((\boldsymbol{r} - \tilde{\boldsymbol{\mu}}_c^{(j)})/\tilde{\boldsymbol{\sigma}}_c^{(j)})\right) - \sum_{j=1}^{k} \hat{w}^{(j)} \left(\log \hat{w}^{(j)} - \mathrm{SG}(\log w^{(j)})\right). \tag{23}$$

And now we can train GCM-VI based on minimizing $-LL - \alpha ELB$.

C Ablation Studies

We perform ablation studies for the GCM and GCM-VI models based on the Adult dataset, evaluating three main hyperparameters: α, β and the latent dimension.

C.1 The Weight for ELB

The final loss function of GCM-VI is a combination of the log-likelihood (LL) and the evidence lower bound (ELB), and the weighting of these terms may influence the training process. To investigate this, we conducted an ablation study on the weight hyperparameter α of the ELB, varying its value from 0.0 to 2.0, the results presented in Table 4.

Table 4: Experimental results on the Adult dataset with multiple α settings.

α	**Log Loss**	**RMSE**	**AUC**	**ACC**
0.0	0.2392 ±0.0015	0.2605 ±0.0004	0.7826 ±0.0031	0.8817 ±0.0018
0.2	0.2378 ±0.0006	0.2602 ±0.0003	0.7861 ±0.0009	0.8821 ±0.0012
0.4	0.2375 ±0.0013	0.2600 ±0.0003	0.7870 ±0.0030	0.8825 ±0.0010
0.6	0.2377 ±0.0011	0.2602 ±0.0001	0.7863 ±0.0035	0.8822 ±0.0005
0.8	0.2372 ±0.0008	0.2601 ±0.0001	0.7874 ±0.0020	0.8822 ±0.0008
1.0	0.2369 ±0.0005	0.2600 ±0.0002	0.7880 ±0.0008	0.8826 ±0.0008
1.2	0.2370 ±0.0009	0.2601 ±0.0005	0.7878 ±0.0033	0.8825 ±0.0014
1.4	0.2368 ±0.0007	0.2600 ±0.0003	0.7883 ±0.0022	0.8826 ±0.0006
1.6	0.2368 ±0.0011	0.2599 ±0.0004	0.7878 ±0.0036	0.8825 ±0.0013
1.8	0.2366 ±0.0011	0.2599 ±0.0004	0.7891 ±0.0043	0.8833 ±0.0020
2.0	0.2365 ±0.0006	0.2599 ±0.0003	0.7890 ±0.0019	0.8830 ±0.0012

The table shows that as α increases from 0.0 to 2.0, performance improves consistently, demonstrating the positive impact of incorporating variational inference into our training objective.

C.2 The Weight for KL Divergence

We evaluate the hyperparameter β in Equation 16 over a range of 0.0 to 2.0, and the details are in Table 5.

Table 5: Experimental results on the Adult dataset with multiple β settings.

β	**Log Loss**	**RMSE**	**AUC**	**ACC**
0.0	0.2385 ±0.0017	0.2603 ±0.0005	0.7840 ±0.0043	0.8820 ±0.0014
0.2	0.2377 ±0.0009	0.2601 ±0.0003	0.7860 ±0.0030	0.8820 ±0.0014
0.4	0.2379 ±0.0012	0.2602 ±0.0003	0.7855 ±0.0028	0.8824 ±0.0008
0.6	0.2383 ±0.0009	0.2602 ±0.0001	0.7848 ±0.0018	0.8822 ±0.0007
0.8	0.2382 ±0.0011	0.2603 ±0.0003	0.7846 ±0.0031	0.8822 ±0.0007
1.0	0.2383 ±0.0015	0.2602 ±0.0003	0.7843 ±0.0044	0.8822 ±0.0014
1.2	0.2376 ±0.0006	0.2601 ±0.0002	0.7866 ±0.0019	0.8824 ±0.0014
1.4	0.2382 ±0.0008	0.2603 ±0.0002	0.7849 ±0.0028	0.8821 ±0.0009
1.6	0.2378 ±0.0007	0.2602 ±0.0002	0.7860 ±0.0019	0.8823 ±0.0010
1.8	0.2385 ±0.0012	0.2604 ±0.0004	0.7839 ±0.0033	0.8821 ±0.0014
2.0	0.2378 ±0.0014	0.2602 ±0.0003	0.7860 ±0.0036	0.8819 ±0.0010

It shows that taking $\beta > 0$ improves average performance and reduces uncertainty.

C.3 The Dimension of Categorical Latent Variable

In this paper, we use a k-categorical latent variable $\boldsymbol{z}$. The choice of k can influence the test results, so we examine k varying from 8 to 96. The details of our findings are presented in Table 6.

Table 6: Experimental results on the Adult dataset with multiple latent dimension k settings.

k	**Log Loss**	**RMSE**	**AUC**	**ACC**
8	0.2377 ±0.0006	0.2601 ±0.0002	0.7861 ±0.0013	0.8824 ±0.0015
16	0.2372 ±0.0005	0.2600 ±0.0002	0.7873 ±0.0014	0.8824 ±0.0012
24	0.2378 ±0.0015	0.2602 ±0.0003	0.7856 ±0.0049	0.8823 ±0.0005
32	0.2373 ±0.0015	0.2601 ±0.0005	0.7872 ±0.0041	0.8826 ±0.0009
40	0.2380 ±0.0017	0.2603 ±0.0004	0.7851 ±0.0046	0.8824 ±0.0015
48	0.2374 ±0.0014	0.2601 ±0.0004	0.7870 ±0.0046	0.8823 ±0.0013
56	0.2373 ±0.0012	0.2600 ±0.0002	0.7875 ±0.0028	0.8824 ±0.0015
64	0.2373 ±0.0007	0.2600 ±0.0001	0.7872 ±0.0030	0.8824 ±0.0008
72	0.2376 ±0.0008	0.2602 ±0.0004	0.7862 ±0.0022	0.8824 ±0.0008
80	0.2372 ±0.0007	0.2600 ±0.0002	0.7874 ±0.0020	0.8827 ±0.0008
88	0.2374 ±0.0004	0.2601 ±0.0001	0.7870 ±0.0017	0.8826 ±0.0015
96	0.2374 ±0.0008	0.2601 ±0.0004	0.7870 ±0.0029	0.8823 ±0.0013

Our results indicate that the dimension parameter remains robust when $k > 48$.

C.4 Ablation on Whether Combining GCM Loss and ELB

We test the model effect trained only by ELB, the results are:

Table 7: Experimental results on the Adult dataset with different loss function.

Method	**RMSE**	**AUC**	**ACC**
GCM	0.2604 ±0.0003	0.7840 ±0.0014	0.8818 ±0.0011
GCM VI	0.2601 ±0.0004	0.7863 ±0.0029	0.8820 ±0.0010
ELB	0.2612 ±0.0002	0.7725 ±0.0020	0.8816 ±0.0012

We can see that training on $\mathcal{L}_{GCM-VI} = \mathcal{L}_{GCM} + \alpha ELB$ is significantly better than training based only on ELB.

D Derivation of the Evidence Lower Bound

As shown in Figure 2b, by $\boldsymbol{z} \perp\!\!\!\perp \boldsymbol{r} \mid \boldsymbol{x}$, we have $p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}) = p_\theta(\boldsymbol{z}|\boldsymbol{x})$, and by $\boldsymbol{x} \perp\!\!\!\perp y \mid \{\boldsymbol{z}, \boldsymbol{r}\}$, we have $p_\theta(y|\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{r}) = p_\theta(y|\boldsymbol{z}, \boldsymbol{r})$, as a result, we can get the following variational bound:

$$
\begin{aligned}
&\log p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi} \log p_\theta(y|\boldsymbol{x}, \boldsymbol{r}) \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi} \left[\log p_\theta(y, \boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}) - \log p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right] \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi} \left[\log p_\theta(y|\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{r}) + \log p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}) - \log p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right] \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi} \left[\log p_\theta(y|\boldsymbol{z}, \boldsymbol{r}) + \log p_\theta(\boldsymbol{z}|\boldsymbol{x}) - \log p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right] \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi} \left[\log p_\theta(y|\boldsymbol{z}, \boldsymbol{r}) + \log p_\theta(\boldsymbol{z}|\boldsymbol{x}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right] \\
&+ D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y) \| p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right) \\
\geq&\mathbb{E}_{\boldsymbol{z}\sim q_\phi} \left[\log p_\theta(y|\boldsymbol{z}, \boldsymbol{r}) + \log p_\theta(\boldsymbol{z}|\boldsymbol{x}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right] \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi} \log p_\theta(y|\boldsymbol{z}, \boldsymbol{r}) - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y) \| p_\theta(\boldsymbol{z}|\boldsymbol{x})\right) \\
=&ELB.
\end{aligned}
\tag{24}
$$

E Details of GCM for Continuous Regression

When $\boldsymbol{y}$ is a continuous variable, we can transform the regression problem into a binary classification problem according to Equation 4. Here we demonstrate how to obtain the maximum likelihood estimate for $y \in \mathbb{R}$.

First, we build the generative model for t and $\boldsymbol{c}$, such that

$$Pr(y + t > 0) = Pr(\boldsymbol{r} \succ \boldsymbol{c}). \tag{25}$$

We suppose y is a Gaussian variable, i.e. $y \sim \mathcal{N}(\mu, \sigma^2)$, where σ is a learnable variable and μ needs to be solved according to Equation 25. Then we have

$$\Phi\left(\frac{\mu + t}{\sigma}\right) = Pr(\boldsymbol{r} \succ \boldsymbol{c}) \triangleq p_1, \tag{26}$$

and we can solve μ as

$$\mu = \sigma\Phi^{-1}(p_1) - t, \tag{27}$$

which is also the maximum likelihood estimation of y. The MLE loss can be formulated as:

$$\mathcal{L} = \frac{(y - \mu)^2}{2\sigma^2} + \log\sigma. \tag{28}$$

So we can now train our model and estimate y.

F Co-generation of Cost and Revenue

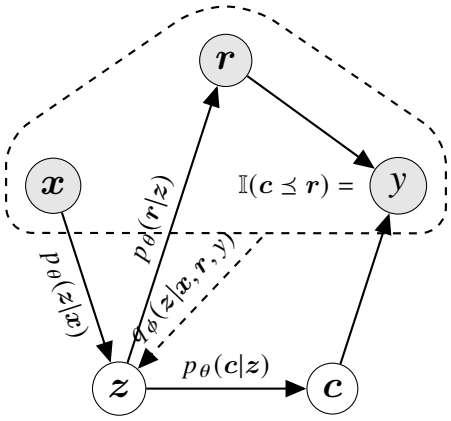


Figure 4: The generative graph for $p(y, \boldsymbol{r}|\boldsymbol{x}, \boldsymbol{z})$.

In certain cases, the assumption of conditional independence $\boldsymbol{z} \perp\!\!\!\perp \boldsymbol{r} \mid \boldsymbol{x}$ may be too restrictive. Instead, we can adjust the cost generative model $p(\boldsymbol{c}|\boldsymbol{x})$ to a cost-revenue generative model $p(\boldsymbol{c}, \boldsymbol{r}|\boldsymbol{x})$, as illustrated in Figure 4. In this context, we establish another weaker conditional independence relationship: $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{r} \mid \boldsymbol{z}$. Similar to Equation 24, the ELB is given by:

$$\begin{aligned}
&\log p_\theta(y, \boldsymbol{r}|\boldsymbol{x}) \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta(y, \boldsymbol{r}, \boldsymbol{z}|\boldsymbol{x}) - \log p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right] \\
=&\mathbb{E}_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta(y, \boldsymbol{r}|\boldsymbol{z}, \boldsymbol{x}) + \log p_\theta(\boldsymbol{z}|\boldsymbol{x}) - \log p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y)\right] \\
\geq&\mathbb{E}_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta(y|\boldsymbol{z}, \boldsymbol{r}) + \log p_\theta(\boldsymbol{r}|\boldsymbol{z})\right] - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{r}, y) \| p_\theta(\boldsymbol{z}|\boldsymbol{x})\right).
\end{aligned} \tag{29}$$

Here, the generation of $\boldsymbol{r}$ follows the same procedure as generating $\boldsymbol{c}$:

$$\lambda_r = \mathrm{DNN}_r(\boldsymbol{z}; \theta_3), \quad p_{\theta_3}(\boldsymbol{r}|\boldsymbol{z}) = \mathcal{P}(\boldsymbol{r}; \lambda_r). \tag{30}$$

We perform experiments of the co-generation of cost and revenue on multiple dataset, and the results are shown in Table 8. It shows that on the Adult and COMPAS datasets, the cogeneration version

of GCM-VI outperforms the original GCM-VI method. This shows an optimistic potential of the cogeneration method for GCM.

Table 8: Experimental results on the multiple datasets.

Method	**Adult** AUC↑	**COMPAS** AUC↑	**Diabetes** AUC↑	**Blog Feedback** RMSE↓
GCM VI	0.7863 ±0.0029	0.7448 ±0.0025	**0.8256** ±0.0007	**0.0990** ±0.0010
GCM VI cogeneration	**0.7883** ±0.0027	**0.7461** ±0.0018	0.8253 ±0.0003	0.1000 ±0.0006

G Experimental Details

G.1 Experimental Setups of GCM

For the GCM and GCM-VI experiments in the card gamble simulation and four public datasets, the hyperparameters of the α, β and latent dimension k are listed in Table 9.

Table 9: Details of the GCM parameters.

dataset	α	β	k
Card	0.5	0	32
Adult	0.5	0.3	64
COMPAS	0.5	0.3	64
Diabetes	0.5	0.3	64
Blog Feedback	0.3	1	64

G.2 Training Process Curve

We show the training process of the experiments in Figure 5 and Figure 6.

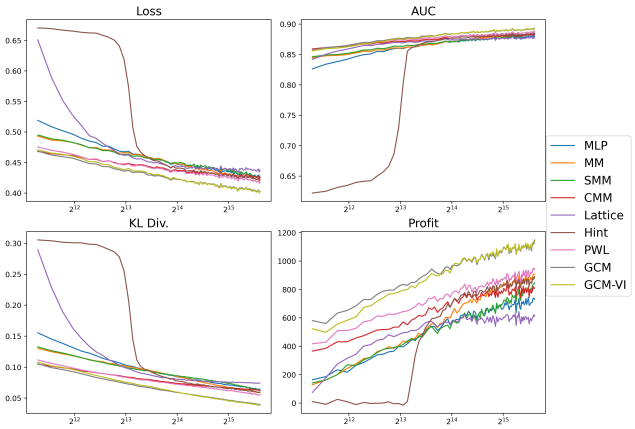
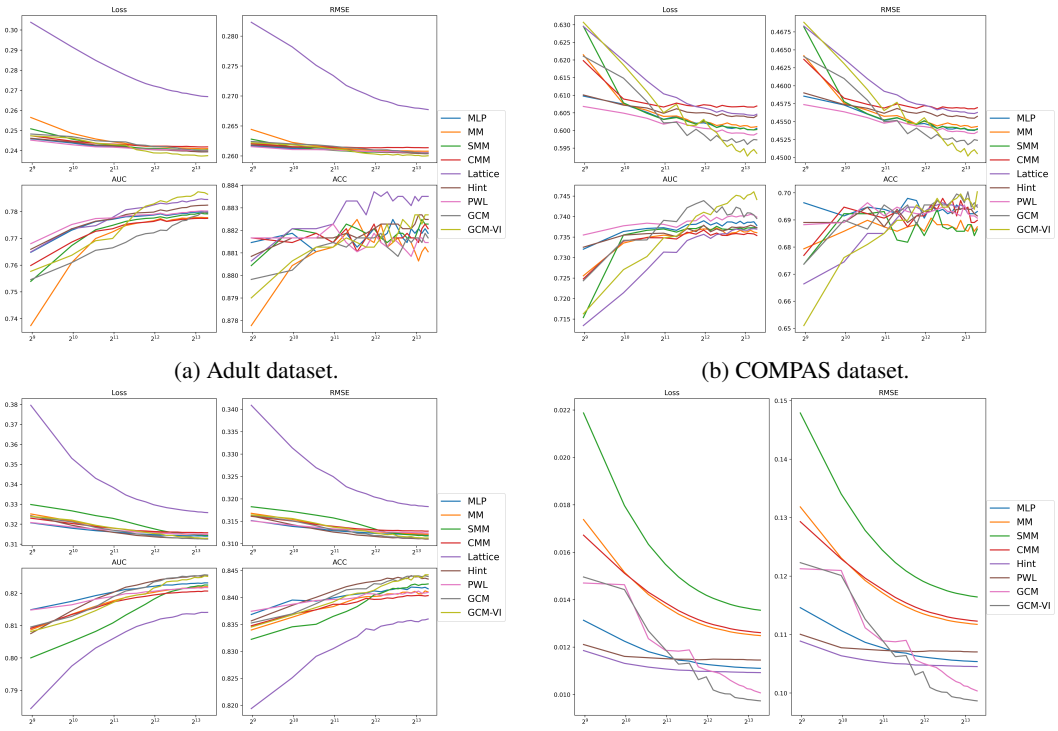


Figure 5: Loss, AUC, KL divergence and prize evaluated on the test sample versus train step (in log scale) in the card gamble simulation.



(a) Adult dataset.

(b) COMPAS dataset.

(c) Diabetes dataset.

(d) Blog Feedback dataset.

Figure 6: Loss, RMSE, AUC, and ACC evaluated on the test sample versus train step (in log scale) in multiple datasets.

G.3 Detailed Results

The details of our experiments on the four public datasets are shown in the following tables.

Table 10: Detailed result of experiments on the Adult dataset.

Method	Log Loss	RMSE	AUC	ACC
MLP	0.2396 ±0.0011	0.2604 ±0.0005	0.7818 ±0.0037	0.8829 ±0.0003
MM	0.2410 ±0.0020	0.2607 ±0.0010	0.7788 ±0.0047	0.8821 ±0.0007
SMM	0.2403 ±0.0019	0.2603 ±0.0007	0.7798 ±0.0059	0.8830 ±0.0006
CMNN	0.2421 ±0.0022	0.2613 ±0.0005	0.7761 ±0.0073	0.8824 ±0.0013
Lattice	0.2674 ±0.0007	0.2679 ±0.0002	0.7841 ±0.0016	0.8836 ±0.0008
Hint	0.2402 ±0.0016	0.2607 ±0.0007	0.7804 ±0.0043	0.8825 ±0.0013
PWL	0.2396 ±0.0012	0.2604 ±0.0006	0.7814 ±0.0032	0.8826 ±0.0006
GCM	0.2387 ±0.0008	0.2604 ±0.0003	0.7840 ±0.0014	0.8818 ±0.0011
GCM VI	0.2375 ±0.0011	0.2601 ±0.0004	0.7863 ±0.0029	0.8820 ±0.0010

Table 11: Detailed result of experiments on the COMPAS dataset.

Method	Log Loss	RMSE	AUC	ACC
MLP	0.6017 ±0.0034	0.4546 ±0.0016	0.7385 ±0.0026	0.6973 ±0.0030
MM	0.5991 ±0.0037	0.4533 ±0.0018	0.7387 ±0.0040	0.6952 ±0.0088
SMM	0.5993 ±0.0031	0.4535 ±0.0013	0.7385 ±0.0022	0.6947 ±0.0031
CMNN	0.6052 ±0.0030	0.4562 ±0.0012	0.7374 ±0.0015	0.6977 ±0.0021
Lattice	0.6027 ±0.0024	0.4554 ±0.0012	0.7388 ±0.0032	0.6972 ±0.0018
Hint	0.6007 ±0.0070	0.4541 ±0.0030	0.7398 ±0.0045	0.6973 ±0.0035
PWL	0.5989 ±0.0053	0.4533 ±0.0025	0.7409 ±0.0037	0.6973 ±0.0039
GCM	0.5942 ±0.0027	0.4510 ±0.0011	0.7449 ±0.0017	0.6989 ±0.0052
GCM VI	0.5944 ±0.0027	0.4511 ±0.0014	0.7448 ±0.0025	0.6998 ±0.0030

Table 12: Detailed result of experiments on the Diabetes dataset.

Method	Log Loss	RMSE	AUC	ACC
MLP	0.3136 ±0.0009	0.3117 ±0.0005	0.8241 ±0.0015	0.8426 ±0.0013
MM	0.3154 ±0.0041	0.3124 ±0.0024	0.8210 ±0.0065	0.8413 ±0.0053
SMM	0.3144 ±0.0008	0.3118 ±0.0004	0.8227 ±0.0011	0.8424 ±0.0012
CMNN	0.3159 ±0.0013	0.3128 ±0.0006	0.8204 ±0.0011	0.8406 ±0.0008
Lattice	0.3258 ±0.0002	0.3183 ±0.0002	0.8142 ±0.0002	0.8360 ±0.0003
Hint	0.3135 ±0.0013	0.3116 ±0.0008	0.8244 ±0.0018	0.8432 ±0.0015
PWL	0.3146 ±0.0012	0.3122 ±0.0008	0.8223 ±0.0017	0.8418 ±0.0014
GCM	0.3130 ±0.0007	0.3113 ±0.0003	0.8251 ±0.0011	0.8439 ±0.0012
GCM VI	0.3128 ±0.0004	0.3112 ±0.0002	0.8256 ±0.0007	0.8443 ±0.0007

Table 13: Detailed result of experiments on the Blog Feedback dataset.

Method	MSE Loss	RMSE
MLP	0.0110 ±0.0002	0.1048 ±0.0008
MM	0.0126 ±0.0005	0.1124 ±0.0023
SMM	0.0132 ±0.0006	0.1149 ±0.0025
CMNN	0.0126 ±0.0005	0.1123 ±0.0022
Lattice	0.0608 ±0.0012	0.2465 ±0.0025
Hint	0.0110 ±0.0001	0.1046 ±0.0003
PWL	0.0114 ±0.0002	0.1067 ±0.0008
GCM	0.0099 ±0.0003	0.0994 ±0.0015
GCM VI	0.0098 ±0.0002	0.0990 ±0.0010

H Comparison of Time Complexity

One of the key advantages of our GCM model is its efficiency during the inference stage. For each given $\boldsymbol{x}$, the model can easily calculate $p_\theta(y|\boldsymbol{x}, \boldsymbol{r}_i)$ for multiple $\boldsymbol{r}_i$ values. This efficiency arises because the GCM model predicts the latent variables $\boldsymbol{z}$ and $\boldsymbol{c}$ based solely on $\boldsymbol{x}$, allowing it to subsequently predict y using $\boldsymbol{c}$ and $\boldsymbol{r}_i$. As a result, we avoid the computation of inputting each pair of $(\boldsymbol{x}, \boldsymbol{r}_i)$ into a deep neural network as methods. We evaluated the inference efficiency for various numbers of $\boldsymbol{r}$ while keeping $\boldsymbol{x}$ stable, and the results are presented in Table 14 and Figure 7. As demonstrated, the GCM becomes the fastest method when the number of $\boldsymbol{r}$ exceeds 64, validating its inference efficiency in multi-revenue prediction scenarios. When the number of $\boldsymbol{r}$ reaches the extreme value of 1024, GCM can save up to 72% time cost compared to the fastest baseline model.

Table 14: Inference time cost (ms per batch) of different models with different numbers of r on the COMPAS dataset.

Method	**Inference r numbers per given x**										
	1	2	4	8	16	32	64	128	256	512	1024
MM	1.51	2.35	3.33	4.83	9.27	17.36	31.24	58.53	112.65	306.57	308.33
CMNN	3.39	5.17	9.02	15.87	28.95	51.96	102.01	198.07	394.63	869.76	877.47
Lattice	2.69	4.19	5.67	6.98	15.02	23.97	44.32	82.95	161.87	384.68	832.70
PWL	1.02	1.67	2.47	3.73	7.86	13.89	26.01	47.86	92.95	280.70	285.48
GCM	11.66	11.55	11.98	12.89	13.88	16.85	20.14	28.89	43.88	76.23	79.63

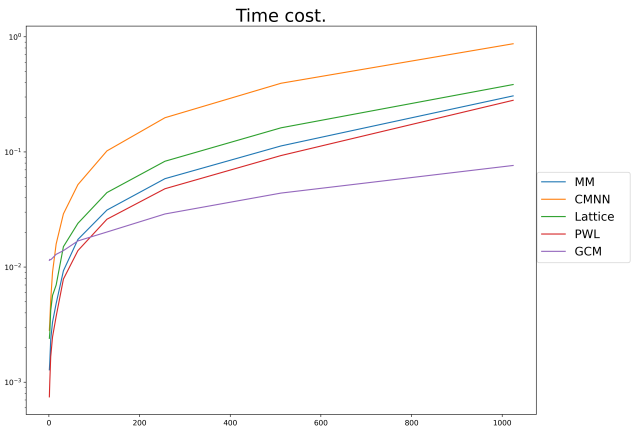


Figure 7: Time cost with different r numbers in inference.

I Application in Quantile Regression

For quantile regression problems, we have the observable variables x and y, and we hope to estimate the r'th quantile of y conditioned on x, that is, $Q_{y|x}(r)$, where $r \in (0, 1)$. It is obvious that $Q_{y|x}(r)$ is strict monotonic with respect to r. However, this problem is different from the original monotonic modeling, since the variable r here is unobservable. To solve this issue, we modify the monotonic modeling problem into the following form:

$$\begin{aligned} &\text{Sample } \ r \sim \mathcal{U}(0,1) \\ &\text{Sample } \ \hat{y} \sim p_\theta(y|x,r) \\ &\text{Minimize } \ r(y-\hat{y})_+ + (1-r)(\hat{y}-y)_+. \end{aligned} \tag{31}$$

Here, we can choose any form of p_θ in this modeling. We compare MLP, Min-Max network, and GCM for the quantile regression problem. The GCM follows the same procedure as formulated in Appendix E. We perform the experiment through a simulation with the setting:

$$\begin{aligned} &y = 0.3\sin(2(x+0.8)) + 0.4\sin(3(x-1.3)) + 0.3\sin(5x) + 0.4(0.8x^2+0.6)\epsilon \\ &x \in (-1.5, 1.5), \ \ \epsilon \sim \mathcal{U}(0,1) \end{aligned} \tag{32}$$

As shown in Figure 8, GCM predicts the most accurate quantile values of y for r ranging from 0.1 to 0.9. In addition, the GCM maintains strict monotonicity between $\hat{y}$ and r, but the MLP model cannot guarantee strict monotonicity when the training step is small.

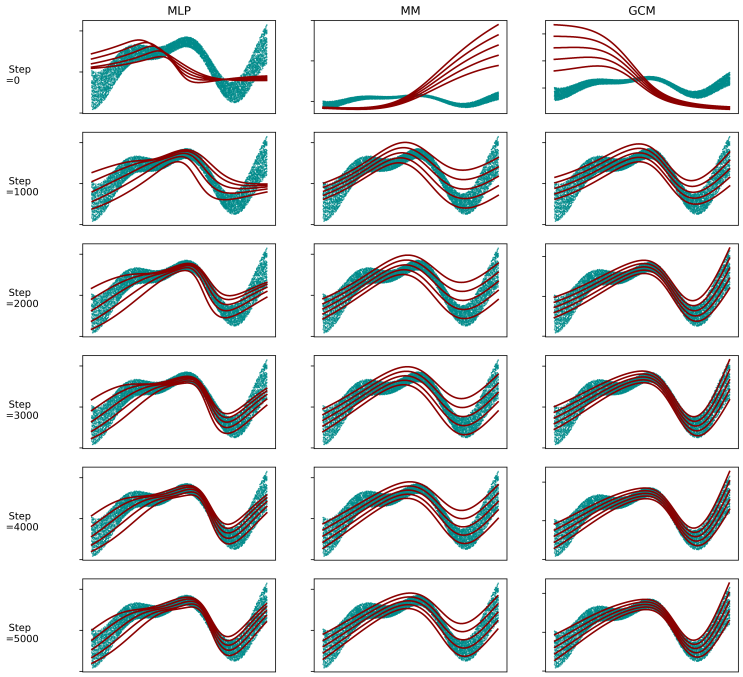


Figure 8: Plot of $\hat{y}|(x, r)$ for $r \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

As shown in Table 15, we test the MAE metric for these methods, proving that GCM has the best performance under all settings of the quantile variable r.

Table 15: MAE of the quantile regression experiment.

	MAE				
Method	r=0.1	r=0.3	r=0.5	r=0.7	r=0.9
MLP	0.1059	0.0897	0.0774	0.0767	0.0862
MM	0.1224	0.1157	0.1185	0.1350	0.1664
GCM	**0.0732**	**0.0685**	**0.0704**	**0.0752**	**0.0815**