Active Preference Alignment

Firstname1 Lastname1 [*1] **Firstname2 Lastname2** [*1,2,3] **Firstname3 Lastname3** [2] **Firstname4 Lastname4** [3]

Abstract

Diffusion models are highly effective at modeling complex data distributions, including images and text. However, in applications like personalized recommender systems, the objective often shifts to modeling specific regions of the distribution that maximize user preferences—initially unknown but gradually uncovered through interactive feedback. This can naturally be framed as a reinforcement learning problem, where the goal is to fine-tune a diffusion model to maximize a reward function based on preferences. However, the main challenge lies in learning a parameterized reward model, which typically requires large-scale preference data—something that is often not feasible in practice. In this work, we introduce a novel framework that bypasses the requirement for a pretrained reward model by directly optimizing the dynamics of the reverse diffusion process using real-time user feedback. Our framework enables feedback-efficient preference alignment, drawing inspiration from the Fynman-Kac-based Fokker-Planck framework. We demonstrate our framework's effectiveness through extensive experiments and ablation studies across diverse domains. Additionally, based on theoretical insights, we propose an enhanced fine-tuning strategy that requires less computational budget and accelerates the fine-tuning process, further boosting its suitability for real-world deployment.

1. Introduction

Diffusion models are powerful deep generative frameworks that synthesize data by reversing a diffusion process, enabling them to capture complex distributions such as natural image manifolds. Yet, in applications like personalized product recommendation, the goal shifts to steering generation toward items that match individual user preferences—preferences that gradually emerge from user interactions. Similar challenges occur in other domains. For example, diffusion models trained on large internet datasets are often used for image generation, but practical use cases demand outputs with specifically desired attributes, such as high aesthetic quality. Comparable situations arise in drug discovery, where generation must be guided toward molecules with strong bioactivity. These tasks can be formulated as reinforcement learning (RL) problems, where the diffusion model is fine-tuned to maximize a reward function encoding target properties or user preferences. However, RL-based methods typically require substantial preference data to learn accurate reward models, making them impractical for settings like personalized recommendation systems, where user feedback is limited and expensive to collect interactively.

The challenge is twofold: Firstly, achieving this objective requires efficient exploration. However, in high-dimensional spaces, such as those of natural images, this goes beyond simply discovering new regions. It also necessitates respecting the structural constraints of the problem. For instance, in areas like product recommendation, valid solutions—such as realistic-looking products—are typically confined to a lower-dimensional manifold within a much larger design space. Therefore, an effective, feedback-efficient fine-tuning method must explore this space while staying within the feasible area, as venturing outside would lead to wasteful invalid queries. Moreover, fine-tuning the diffusion model to aggressively optimize based on the preferences collected so far can reduce sample diversity. This is because human preferences are often multimodal, and the model, if overly focused on a narrow set of preferences, may fail to capture the full spectrum of diverse user preferences, leading to a less varied sample generation. Therefore, efficient exploration is crucial to maintaining the quality of generated samples and ensuring greater diversity in sample generation.

Secondly, a key challenge in many applications is the high cost of acquiring feedback for the ground-truth reward function. For instance, in a product recommendation system, de-

[*]Equal contribution [1]Department of XXX, University of YYY, Location, Country [2]Company Name, Location, Country [3]School of ZZZ, Institute of WWW, Location, Country. Correspondence to: Firstname1 Lastname1 <first1.last1@xxx.edu>, Firstname2 Lastname2 <first2.last2@www.uk>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

termining user preferences requires subjective human judgment, which is both costly and time-consuming. This challenge is further compounded by the need for the model to not only explore new options but also to exploit the information it has gathered to generate samples that align with the user's preferences. If the model continues to explore without producing samples that meet the user's expectations, it risks disengaging the user. In a nutshell, the model must strike a balance between exploration and exploitation—effectively generating preference-aligned samples while minimizing costly reward queries. While several recent works have proposed RL-based fine-tuning methods for diffusion models, none directly tackle the challenge of feedback efficiency in an online setting. Uehera et. al. introduced a framework that accounts for the online nature of feedback but still relies on a separate parameterized reward model for optimization. Our goal instead is to develop a feedback-efficient online fine-tuning approach that entirely eliminates the need for a separate pre-trained reward model, instead directly leveraging inference time scaling of a base diffusion model using real-time user feedback.

2. Preliminaries

TODO

3. Problem Formulation

TODO

4. Related Work

TODO

5. Methodology

TODO

ADD OUR METHOD STRUCTURE HERE.

6. Experiments and Analysis

TODO

7. Conclusions

TODO

Algorithm 1 Budget-Constrained Feynman-Kac Diffusion

Input: Budget B, particles to observe per step k, total particles N, reward function $r(x)$
Output: Trained reward network f_θ, observed high-reward particles
Initialize CNN f_θ for reward prediction
Initialize total_steps $\leftarrow B/k$, step $\leftarrow 0$
Cold Start (Step 0):
  Sample k particles $\{x_i^{(0)}\}_{i=1}^k$ and run baseline diffusion
  Observe true rewards: $\{r(x_i^{(0)})\}_{i=1}^k$
  Train CNN: $f_\theta \leftarrow$ optimize($\{(x_i^{(0)}, r(x_i^{(0)}))\}$)
  Update budget: $B \leftarrow B - k$, step $\leftarrow 1$
while $B > 0$ **do**
  // Generate particles with FKC
  Sample N particles $\{x_i\}_{i=1}^N$ from random, initialize $w_i \leftarrow 0$
  for timestep t from 1 to 0 **do**
    Compute reward gradient: $\nabla r_\theta(x_i) \leftarrow \nabla f_\theta(x_i)$
    Update position: $x_i \leftarrow x_i + \sigma_t^2(\nabla \log q_t + \frac{\beta_t}{2}\nabla r_\theta)dt + \sigma_t dW_t$
    Update weight: $w_i \leftarrow w_i + [\beta_t' r_\theta(x_i) + \langle \beta_t \nabla r_\theta, \text{terms}\rangle]|dt|$
  end for
  // Epsilon-greedy selection
  Compute $\epsilon \leftarrow 0.8 - 0.7\times$ (step/total_steps)
  Calculate $n_{\text{random}} \leftarrow \lfloor \epsilon \times k \rfloor$, $n_{\text{top}} \leftarrow k - n_{\text{random}}$
  Select top-k indices: $\mathcal{I}_{\text{top}} \leftarrow$ argsort($\{w_i\}$)[$-n_{\text{top}}$:]
  Select random indices: $\mathcal{I}_{\text{random}} \leftarrow$ random($N \setminus \mathcal{I}_{\text{top}}$, n_{random})
  Combine: $\mathcal{I}_{\text{observe}} \leftarrow \mathcal{I}_{\text{top}} \cup \mathcal{I}_{\text{random}}$
  // Observe and learn
  Observe true rewards: $\{r(x_i)\}_{i \in \mathcal{I}_{\text{observe}}}$
  Update CNN: $f_\theta \leftarrow$ optimize($\{(x_i, r(x_i)) : i \in \mathcal{I}_{\text{observe}}\}$)
  Update budget: $B \leftarrow B - k$, step $\leftarrow$ step $+1$
end while
return f_θ

Algorithm 2 FKC SDE Step

Input: Position x_t, weight w_t, time t, timestep dt, reward network f_θ
Output: Updated position x_{t+dt}, updated weight w_{t+dt}
Compute $\sigma_t \leftarrow \text{diffusion\_coefficient}(t)$
Compute $\nabla \log q_t \leftarrow \text{score\_function}(x_t, t)$
Compute $\nabla r_\theta \leftarrow \nabla f_\theta(x_t)$
// Position update (Equation 29)
$\text{drift} \leftarrow \sigma_t^2(\nabla \log q_t + \frac{\beta_t}{2}\nabla r_\theta) - f_t(x_t)$
$x_{t+dt} \leftarrow x_t + \text{drift} \cdot dt + \sigma_t\sqrt{|dt|} \cdot \mathcal{N}(0, I)$
// Weight update (Equation 30, with —dt— fix)
$r_\theta \leftarrow f_\theta(x_t)$
$\text{term}_1 \leftarrow \beta_t' \cdot r_\theta \cdot |dt|$
$\text{term}_2 \leftarrow -\langle \beta_t \nabla r_\theta, f_t(x_t)\rangle \cdot |dt|$
$\text{term}_3 \leftarrow \langle \beta_t \nabla r_\theta, \frac{\sigma_t^2}{2}\nabla \log q_t\rangle \cdot |dt|$
$dw \leftarrow \text{clip}(\text{term}_1 + \text{term}_2 + \text{term}_3, -0.1, 0.1)$
$w_{t+dt} \leftarrow \text{clip}(w_t + dw, 0.1, 10.0)$
return x_{t+dt}, w_{t+dt}