FROM FUNCTION TO DISTRIBUTION MODELING:
A PAC-GENERATIVE APPROACH TO OFFLINE OPTIMIZATION

Qiang Zhang[*]
Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843

Ruida Zhou[*]
Department of Electrical and Computer Engineering
University of California, Los Angeles
Los Angeles, CA 90095

Yang Shen
Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843

Tie Liu
Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843

ABSTRACT

This paper considers the problem of offline optimization, where the objective function is unknown except for a collection of "offline" data examples. While recent years have seen a flurry of work on applying various machine learning techniques to the offline optimization problem, the majority of these work focused on learning a surrogate of the unknown objective function and then applying existing optimization algorithms. While the idea of modeling the unknown objective function is intuitive and appealing, from the learning point of view it also makes it very difficult to tune the objective of the learner according to the objective of optimization. Instead of learning and then optimizing the unknown objective function, in this paper we take on a less intuitive but more direct view that optimization can be thought of as a process of sampling from a generative model. To learn an effective generative model from the offline data examples, we consider the standard technique of "re-weighting", and our main technical contribution is a probably approximately correct (PAC) lower bound on the natural optimization objective, which allows us to jointly learn a weight function and a score-based generative model. The robustly competitive performance of the proposed approach is demonstrated via empirical studies using the standard offline optimization benchmarks.

Keywords Offline optimization · Distribution modeling · Generative model · End-to-end learning

1 Introduction

Offline optimization refers to the problem of optimizing an *unknown* real-valued objective function f based only on a collection of "offline" data examples $((\boldsymbol{x}_i, f(\boldsymbol{x}_i)) : i \in [m] := \{1, 2, \ldots, m\})$, where each $\boldsymbol{x}_i$ is an independent sample drawn from an *unknown* data-generating distribution p_{data}. Aside from these examples, no additional information on the objective function f is available prior to or during the optimization process, and hence the name "offline optimization". This rather restrictive setting is particularly relevant to the optimization scenarios where: i) the objective function is very complex and no structural information is available; and ii) querying the objective function is very expensive. Potential applications include the design of proteins [Kolli et al., 2022], chemical molecules [Gómez-Bombarelli et al., 2018], DNA sequences [Killoran et al., 2017], aircrafts [Hoburg and Abbeel, 2014], robots [Liao et al., 2019], and hardware accelerators [Kumar et al., 2022].

Obviously, offline optimization is a much more challenging setting than standard optimization [Beck, 2017] (where *full* structural information on the objective function is available) or black-box optimization [Audet and Hare, 2017] (where even though no *structural* information on the objective function is available, the objective function can be queried upon

---

[*]Q. Zhang and R. Zhou should both be considered first authors.

during the optimization process). Therefore, instead of aiming at the global optima, for offline optimization we are usually satisfied with finding a few candidates, among which there are *significantly* better solutions than the existing offline observations.

Traditionally, offline optimization has been mainly approached through the *Bayesian* view, i.e., by endowing the unknown objective function f a *prior* distribution. This has led to a large body of work under the name *Bayesian optimization*; see Fu and Levine [2021] and the references therein for the recent progress in this direction. Motivated by the rapid progress in machine learning, recent years have also seen a flurry of work on offline optimization from a *frequentist's* view [Brookes et al., 2019, Gupta and Zou, 2019, Kumar and Levine, 2020, Trabucco et al., 2021], i.e., by modeling the objective function f as a *deterministic but unknown* function. However, most of these work have been focusing on learning a surrogate of the unknown objective function and then applying existing optimization algorithms. Prime examples include Trabucco et al. [2021], which focused on adapting the *gradient* method to the offline setting, and Brookes et al. [2019], Gupta and Zou [2019], which focused on adapting the *cross-entropy* method [Rubinstein, 1999] to the offline setting. While the idea of modeling the unknown objective function is intuitive and appealing, from the learning point of view it also makes it very difficult to tune the objective of the learner according to the objective of optimization [Trabucco et al., 2021, Brookes et al., 2019, Gupta and Zou, 2019]. As a result, it is very difficult to gauge whether these previous approaches actually come with any theoretical guarantees.

In this paper we take on a less intuitive but more *direct* view of optimization and consider it as a process of *sampling* from a *generative model*. There are two natural advantages to this view. First, through sampling *exploration* is now intrinsic in the optimization process. Second, this view allows us to shift our focus from modeling the objective function to modeling a *target distribution*. Unlike learning a surrogate on the objective function, as we shall show, the objective of learning a target distribution can be *naturally* aligned with the objective of optimization, thus bringing *theoretical guarantees* on the optimization performance.

Let $p_{\text{target}}^{\theta}$ be a *generative model* from which sampling can produce, with high probability, samples whose objective values are significantly better than the offline observations. Note that unlike the traditional generative models, which are trained to generate samples that are "similar" to the training examples, the goal of our generative model $p_{\text{target}}^{\theta}$ is to generate samples with *superior* objective values than the offline observations. Relative to the data-generating distribution p_{data}, these targeted samples with superior objective values are the "outliers". Therefore, from the learning perspective, our main challenge here is to learn a generative model that generates *outliers* rather than the norm.

To facilitate the learning of a desired generative model, in this paper we shall consider the standard technique of "re-weighting". Roughly speaking, we shall consider a *weight* function that assigns higher weights to the domain points with higher objective values and then train a generative model using the *weighted* offline examples. This helps to tune the generative model towards generating samples with high objective values.

Formally, let

$$q_{\text{target}}(\boldsymbol{x}) = \tilde{w}(f(\boldsymbol{x}))p_{\text{data}}(\boldsymbol{x}) \tag{1}$$

be a *hypothetical* target distribution, where $\tilde{w}$ is a normalized, non-negative *weight* function such that $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\tilde{w}(f(\mathbf{x}))] = 1$, and p_{data} is the (unknown) data-generating distribution from which the offline observations $\boldsymbol{x}_{[m]} := (\boldsymbol{x}_i : i \in [m])$ were drawn. In our approach, the hypothetical target distribution q_{target} plays *dual* roles: On one hand, it serves as the *hypothetical* learning target of the generative model $p_{\text{target}}^{\theta}$; on the other hand, it is also connected to the unknown data-generating distribution p_{data} via the normalized weight function $\tilde{w}$ and hence allows a generative model $p_{\text{target}}^{\theta}$ to be learned from the offline data examples. Operationally, we would like to train a generative model $p_{\text{target}}^{\theta}$ such that $p_{\text{target}}^{\theta} \approx q_{\text{target}}$. But what would be a suitable choice for the normalized weight function $\tilde{w}$?

Intuitively, there are two considerations for selecting a normalized weight function $\tilde{w}$. On one hand, from the *utility* point of view, we would like to choose $\tilde{w}$ such that the hypothetical target distribution q_{target} focuses most of its densities on the domain points with *superior* objective values. This can be achieved, for example, by choosing $\tilde{w}$ to be heavily *skewed* towards superior objective values. On the other hand, from the *learning* viewpoint, the generative model $p_{\text{target}}^{\theta}$ is learned from the offline observations, which were generated from the unknown data-generating distribution p_{data}. If $\tilde{w}$ is chosen to be heavily skewed, the hypothetical target distribution q_{target} then becomes very *different* from the data-generating distribution p_{data}. In this case, learning the generative model $p_{\text{target}}^{\theta}$ from the offline data examples may be subject to very high *sample* complexity.

Given these two seemingly *conflicting* considerations, it is natural to make the normalized weight function $\tilde{w}$ (and hence the hypothetical target distribution q_{target}) to be *learnable* as well. Assume without loss of generality that our goal is to *maximize* the (unknown) objective function f. From the optimization point of view, a *natural* optimization objective for identifying a desired generative model $p_{\text{target}}^{\theta}$ is by maximizing the expected objective value:

$$J_{\text{opt}}(\theta) = \mathbb{E}_{\mathbf{x} \sim p_{\text{target}}^{\theta}}[f(\mathbf{x})]. \tag{2}$$

2

The above objective, however, *cannot* be evaluated for any given model parameter θ, because the objective function f is *unknown*. Instead of trying to learn a surrogate on f (and then use it to guide the training of the generative model), in this paper we shall follow the more traditional learning-theoretic approach of constructing a *probably approximately correct (PAC)* lower bound on the natural optimization objective $J_{\text{opt}(\theta)}$. Unlike $J_{\text{opt}(\theta)}$, the PAC lower bound depends on both θ and the normalized weight function $\tilde{w}$. As we shall see, not only it captures *both* the aforementioned utility and learnability considerations for selecting $\tilde{w}$, it will also *naturally* suggest a objective, from θ and $\tilde{w}$ can be *jointly* learned.

The rest of the paper is organized as follows. In Section 2, we introduce a few technical results for establishing the PAC lower bound on J_{opt}. The PAC lower bound is formally presented in Section 3. In Section 4, we discuss how to leverage the PAC lower bound for jointly learning a weight function and a *score-based* generative model. In Section 5, we demonstrate the legitimacy and the robustly competitive performance of the proposed learner, first through a toy example and then through the standard offline optimization benchmark datasets. Finally in Section 6, we conclude the paper with an in-depth discussion on the contribution of this paper in the context of several related work.

2   Preliminaries

In this section, we introduce a few technical results that are essential for constructing the desired PAC lower bound.

2.1   Wasserstein distance

Let μ and ν be two probability distributions on $\mathbb{R}^d$. A *coupling* γ between μ and ν is a joint distribution on $\mathbb{R}^d \times \mathbb{R}^d$ whose marginals are μ and ν. The p-Wasserstein distance between μ and ν (with respect to the Euclidean norm) is given by:

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \mathbb{E}_{(\mathbf{x}, \tilde{\mathbf{x}}) \sim \gamma} \left[\|\mathbf{x} - \tilde{\mathbf{x}}\|^p \right] \right)^{1/p}, \tag{3}$$

where $\Gamma(\mu, \nu)$ is the set of all couplings between μ and ν, and $\| \cdot \|$ denotes the standard Euclidean norm.

The 1-Wasserstein distance, also known as the *earth mover's distance*, has an important equivalent representation that follows from the *duality* theorem of Kantorovich-Rubenstein [Ambrosio et al., 2021]:

$$W_1(\mu, \nu) = \frac{1}{K} \sup_{\|\tilde{f}\|_{\text{Lip}} \leq K} \left\{ \mathbb{E}_{\mathbf{x} \sim \mu}[\tilde{f}(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim \nu}[\tilde{f}(\mathbf{x})] \right\}, \tag{4}$$

where $\| \cdot \|_{\text{Lip}}$ denotes the *Lipschitz* norm. In our construction, this *dual* representation of the 1-Wasserstein distance serves as the bridge between the *objective-specific* generative loss and the *generic* generative loss. By the standard Jensen's inequality, we also have

$$W_1(\mu, \nu) \leq W_2(\mu, \nu) \tag{5}$$

for any two distributions μ and ν. As we shall see, this simple relationship between the 1-Wasserstein and 2-Wasserstein distances can help to further connect the *objective-specific* generative loss to the *denoising score matching loss* for training a *score-based* generative model.

2.2   Denoising diffusion probabilistic model

In this paper, we shall mainly focus on training a *score-based* model p_{target}^θ such that $p_{\text{target}}^\theta \approx q_{\text{target}}$. This is motivated by the following connection between the *score function* of the hypothetical target distribution q_{target} and the *gradient* of the unknown objective function f:

$$s_{\text{target}}(\boldsymbol{x}) = \nabla \log q_{\text{target}}(\boldsymbol{x}) = \nabla \log \left[\tilde{w}(f(\boldsymbol{x})) p_{\text{data}}(\boldsymbol{x}) \right]$$

$$= s_{\text{data}}(\boldsymbol{x}) + \frac{\tilde{w}'(f(\boldsymbol{x}))}{\tilde{w}(f(\boldsymbol{x}))} \nabla f(\boldsymbol{x}), \tag{6}$$

where s_{target} and s_{data} are the score functions of p_{target}^θ and p_{data}, respectively. If $\tilde{w}$ is *monotone increasing*, the derivative $\tilde{w}'(f(\boldsymbol{x})) > 0$ for all $\boldsymbol{x} \in \mathcal{X}$. In this case, sampling along the direction of s_{target} can *naturally* produce samples with higher objective values.

We are particularly interested in the *denoising diffusion probabilistic model (DDPM)* [Song et al., 2020a, Ho et al., 2020] due to its stability and performance on high-dimensional datasets. Here we recall a few essential results on the

DDPM. Consider a *forward* process of continuously injecting white Gaussian noise into a signal $\mathbf{x}_t$:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)}d\mathbf{w}_t, \ \ t \in [0,1], \tag{7}$$

where $\beta : [0,1] \to \mathbb{R}_{++}$ is a positive *noise scheduler*, $\mathbf{w}_t$ is a standard *Wiener* process, and time in this process is assumed to flow in the *forward* direction from $t = 0$ to $t = 1$. Denote by q_t the marginal distribution of $\mathbf{x}_t$ from the forward process (7). DDPM is mainly motivated by the fact that the marginal distributions q_t, $t \in [0,1]$, can be *recovered* through the following *reverse* process [Anderson, 1982]:

$$d\mathbf{x}_t = -\beta(t)\left(\frac{1}{2}\mathbf{x}_t + \boldsymbol{s}_t^\theta(\mathbf{x}_t)\right)dt + \sqrt{\beta(t)}d\bar{\mathbf{w}}_t, \tag{8}$$

where $\boldsymbol{s}_t^\theta$ is a model of the score function of q_t and $\bar{\mathbf{w}}_t$ is (again) a standard *Wiener* process but with time flowing *backward* from $t = 1$ to $t = 0$. More specifically, let p_t^θ be the marginal distribution of $\mathbf{x}_t$, $t \in [0,1]$ from the reverse process (8). If we let $p_1^\theta = q_1$ and $\boldsymbol{s}_t^\theta$ be the *exact* score function of q_t for all $t \in [0,1]$, we have $p_t^\theta = q_t$ for *all* $t \in [0,1)$ [Bogachev et al., 2022]. In particular, the initial distribution of the forward process q_0 can be recovered at the end of the reverse process via the score functions of q_t, $t \in [0,1]$. Thus, to learn the initial distribution of the forward process q_0, it suffices to learn a model $\boldsymbol{s}_t^\theta$ that approximates the score functions of q_t for all $t \in [0,1]$.

There are several methods [Hyvärinen, 2005, Vincent, 2011, Song et al., 2020b] that allow a model $\boldsymbol{s}_t^\theta$ to be learned from a training dataset drawn from q_0. Here we focus on the *denoising score matching method* due to its scalability to large datasets. For the denoising score matching method, a model $\boldsymbol{s}_t^\theta$ is learned by minimizing the following *denoising score matching loss*:

$$\mathcal{L}_{\text{DSM}}(\theta; q_0) = \mathbb{E}_{\mathbf{x} \sim q_0}\left[\ell_{\text{DSM}}^\theta(\mathbf{x})\right], \tag{9}$$

where

$$\ell_{\text{DSM}}^\theta(\mathbf{x}) = \int_0^1 \lambda(t)\mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0,\boldsymbol{I})}\left[\left\|\boldsymbol{s}_t^\theta(\mathbf{x}_t) + \frac{\mathbf{z}}{\sigma(t)}\right\|^2\right]dt \tag{10}$$

is the *point-wise denoising score matching loss*, $\lambda : [0,1] \to \mathbb{R}_{++}$ can be any positive function,

$$\mathbf{x}_t = \sqrt{1 - \sigma(t)^2}\mathbf{x} + \sigma(t)\mathbf{z},$$

and

$$\sigma(t) = \sqrt{1 - \exp\left[-\int_0^t \beta(s)ds\right]}.$$

While the previous denoising score matching loss can be easily estimated from a dataset drawn from q_0 (and hence is very *conductive* to learning), *a priori* it is unclear how it would connect to any *generative* loss between q_0 and p_0^θ. Interestingly, it was shown in Theorem 2 and Corollary 3 of Kwon et al. [2022] that under some (relatively) mild conditions[2] on β, q_0, and $\boldsymbol{s}_\theta$, by choosing $\lambda(t) = \beta(t)$ we have

$$W_2(q_0, p_0^\theta) \le c_0\sqrt{\mathcal{L}_{\text{DSM}}(\theta; q_0)} + c_1 W_2(q_1, p_1)$$
$$= c_0\sqrt{\mathbb{E}_{\mathbf{x} \sim q_0}\left[\ell_{\text{DSM}}^\theta(\mathbf{x})\right]} + c_1 W_2(q_1, p_1), \tag{11}$$

where c_0 and c_1 are constants that only depend on the choice of the noise scheduler β and some prior knowledge on p_0 and $\boldsymbol{s}_t^\theta$ but is independent of the model parameter θ. In Kwon et al. [2022], this result was coined as "score-based generative modeling *secretly* minimizes the Wasserstein distance".

For our purposes, let $q_0 = q_{\text{target}}$ and p_1^θ be the standard Gaussian distribution $\mathcal{N}$. If we denote q_1 and p_0^θ by $\bar{q}_{\text{target}}$ and p_{target}^θ respectively, we have

$$W_2(q_{\text{target}}, p_{\text{target}}^\theta) \le c_0\sqrt{\mathbb{E}_{\mathbf{x} \sim q_{\text{target}}}\left[\ell_{\text{DSM}}^\theta(\mathbf{x})\right]} + c_1 W_2(\bar{q}_{\text{target}}, \mathcal{N}), \tag{12}$$

where c_0 and c_1 are constants that only depend on the choice of the noise scheduler β and some prior knowledge on p_{data}, $\tilde{w}$, and $\boldsymbol{s}_t^\theta$; otherwise, they are independent of the model parameter θ and the choice of the normalized weight function $\tilde{w}$. We mention here that the output distribution of the forward process $\bar{q}_{\text{target}}$ is potentially dependent on the choice of $\tilde{w}$, even though this dependency is not explicit from the notation. In practice, $\bar{q}_{\text{target}}$ can be made very close to the standard Gaussian distribution $\mathcal{N}$ with an appropriate choice of the noise scheduler β. Therefore, the Wasserstein distance $W_2(\bar{q}_{\text{target}}, \mathcal{N})$ is very small and is usually disregarded from the learning process.

---

[2]The readers are referred to Section 3.1 of Kwon et al. [2022] for the assumptions under which the inequality (11) holds.

2.3 Generalization bound for weighted learning

Let $\ell_\theta : \mathcal{X} \to \mathbb{R}$ be a *bounded* loss function parameterized by $\theta \in \Theta$ such that $0 \leq \ell_\theta(\boldsymbol{x}) \leq \Delta$ for all $\boldsymbol{x} \in \mathcal{X}$ and all $\theta \in \Theta$. Consider the problem of estimating the expected *weighted* loss $\mathcal{L}_p(\theta, \tilde{w}) = \mathbb{E}_{\mathbf{x} \sim p}[\tilde{w}(\mathbf{x})\ell_\theta(\mathbf{x})]$, where $\tilde{w} : \mathcal{X} \to \mathbb{R}_+$ is a normalized, *bounded* weight function such that $\mathbb{E}_{\mathbf{x} \sim p}[\tilde{w}(\mathbf{x})] = 1$ and $0 \leq \tilde{w}(\boldsymbol{x}) \leq B$ for all $\boldsymbol{x} \in \mathcal{X}$. We have the following PAC upper bound, with respect to the parameter family Θ, on the expected weighted loss $\mathcal{L}_p(\theta, \tilde{w})$ for any given $\tilde{w}$.

Lemma 1. *For any given $\tilde{w}$, with probability $\geq 1 - \delta$ we have for* any $\theta \in \Theta$

$$\mathcal{L}_p(\theta, \tilde{w}) \leq \hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) + 2\Delta \sqrt{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} + 2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) + 3\sqrt{\frac{2B\Delta \log(2/\delta)}{m}}, \tag{13}$$

where

$$\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) = \frac{1}{m} \sum_{i=1}^{m} \tilde{w}(\boldsymbol{x}_i)\ell_\theta(\boldsymbol{x}_i)$$

is the empirical *weighted loss over the training dataset $\boldsymbol{x}_{[m]}$,*

$$\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m} \sum_{i=1}^{m} (\tilde{w}(\boldsymbol{x}_i) - 1)^2$$

is the empirical *variance of $\tilde{w}$ over $\boldsymbol{x}_{[m]}$, and*

$$\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m} \sum_{i=1}^{m} \sigma_i \ell_\theta(\boldsymbol{x}_i) \right]$$

is the empirical *Rademacher complexity with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$.*

A proof of the above lemma is deferred to Appendix A.1 to enhance the flow of the paper. The main insight from the above lemma is that the *generalization error* (with respect to the parameter θ) between the expected weighted loss $\mathcal{L}_p(\theta, \tilde{w})$ and the empirical weighted loss $\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})$ can be controlled by controlling the *complexity* of the model class Θ and the *variance* of the normalized weight function $\tilde{w}$.

3 Main Results

We first introduce a *distribution-dependent* surrogate on the natural optimization objective $J_{\text{opt}}(\theta)$ for a *score-based* generative model p_{target}^θ.

Proposition 1. *Assume that the unknown objective function f is K-Lipschitz and the generative model p_{target}^θ is a DDPM. Under the assumptions from Section 3.1 of Kwon et al. [2022] on the noise scheduler β, the data-generating distribution p_{data}, the normalized weight function $\tilde{w}$, and the score-function model $\boldsymbol{s}_t^\theta$, we have*

$$J_{opt}(\theta) \geq \underbrace{\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})\right]}_{\text{expected utility}} - \underbrace{c_0 K \sqrt{\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))\ell_{DSM}^\theta(\mathbf{x})\right]}}_{\text{expected generative loss}} - c_1 K W_2(\bar{q}_{target}, \mathcal{N}), \tag{14}$$

where ℓ_{DSM}^θ is the point-wise denoising score matching loss of $\boldsymbol{s}_\theta$ as defined in (10), $\bar{q}_{target}$ is the output distribution of the forward process (7), Φ is the standard Gaussian distribution, and c_0 and c_1 are constants that are independent of the model parameter θ and $\tilde{w}$.

Proof. We start by writing $J_{\text{opt}}(\theta)$ as:

$$J_{\text{opt}}(\theta) = \mathbb{E}_{\mathbf{x} \sim q_{\text{target}}}[f(\mathbf{x})] - \left\{ \mathbb{E}_{\mathbf{x} \sim q_{\text{target}}}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim p_{\text{target}}^\theta}[f(\mathbf{x})] \right\}. \tag{15}$$

By the definition of q_{target} from (1), we have

$$\mathbb{E}_{\mathbf{x} \sim q_{\text{target}}}[f(\mathbf{x})] = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})]. \tag{16}$$

Furthermore,

$$\mathbb{E}_{\mathbf{x}\sim q_{\text{target}}}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_{\text{target}}^{\theta}}[f(\mathbf{x})] \leq \sup_{\|\tilde{f}\|_{\text{Lip}}\leq K}\left\{\mathbb{E}_{\mathbf{x}\sim q_{\text{target}}}[\tilde{f}(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_{\text{target}}^{\theta}}[\tilde{f}(\mathbf{x})]\right\}$$
$$= K \cdot W_1(q_{\text{target}}, p_{\text{target}}^{\theta}), \tag{17}$$

where the first inequality follows directly from the assumption that f is K-Lipschitz, and the second equality follows from the dual representation (17) of the 1-Wasserstein distance.

Under the assumption that $p_{\text{target}}^{\theta}$ is a DDPM, we can further bound the 1-Wasserstein distance $W_1(q_{\text{target}}, p_{\text{target}}^{\theta})$ as:

$$W_1(q_{\text{target}}, p_{\text{target}}^{\theta}) \leq W_2(q_{\text{target}}, p_{\text{target}}^{\theta}) \leq c_0\sqrt{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))\ell_{\text{DSM}}^{\theta}(\mathbf{x})\right]} + c_1 W_2(\bar{q}_{\text{target}}, \mathcal{N}), \tag{18}$$

where the first inequality follows from (5), and the second inequality follows from (12) and the definition of q_{target} in (1).

Substituting (16), (17), and (18) into (15) completes the proof of (14). □

The distribution-dependent surrogate on the right-hand side of (14) can be converted into a *PAC* lower bound using the standard *complexity* theory for machine learning. The result is summarized in the following theorem.

Theorem 1. *Assume that: i) the unknown objective function f is K-Lipschitz and satisfies $|f(\boldsymbol{x})| \leq F$ for all $\boldsymbol{x} \in \mathcal{X}$; ii) the generative model $p_{\text{target}}^{\theta}$ is a DDPM; iii) the point-wise denoising score matching loss ℓ_{DSM}^{θ} satisfies $0 \leq \ell_{DSM}^{\theta}(\boldsymbol{x}) \leq \Delta$ for all $\boldsymbol{x} \in \mathcal{X}$ and all $\theta \in \Theta$; and iv) the conditions from Section 3.1 of Kwon et al. [2022] on the noise scheduler β, the data-generating distribution p_{data}, the normalized weight function $\tilde{w}$, and the score-function model $\boldsymbol{s}_t^{\theta}$ are satisfied. Let $\tilde{\mathcal{W}}$ be the collection of all normalized weight functions $\tilde{w}$ that are L-Lipschitz and satisfy $0 \leq \tilde{w}(y) \leq B$ for any $y \in [-F, F]$. With probability $\geq 1 - \delta$, we have for any $\tilde{w} \in \tilde{\mathcal{W}}$ and any $\theta \in \Theta$,*

$$J_{opt}(\theta) \geq \underbrace{\hat{J}_{\boldsymbol{x}_{[m]}}(\tilde{w})}_{\text{empirical utility}} - \underbrace{c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})}}_{\text{empirical generative loss}} - \underbrace{c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})}}_{\text{empirical variance}} -$$
$$c_1 K W_2(\bar{q}_{target}, \mathcal{N}) - c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - O\left(1/\sqrt[8]{m}\right), \tag{19}$$

where

$$\hat{J}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m}\tilde{w}(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i)$$

is the empirical *utility* of $\tilde{w}$,

$$\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) = \frac{1}{m}\sum_{i=1}^{m}\tilde{w}(f(\boldsymbol{x}_i))\ell_{DSM}^{\theta}(\boldsymbol{x}_i)$$

is the empirical *weighted denoising score matching loss of $\boldsymbol{s}_t^{\theta}$,*

$$\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m}(\tilde{w}(f(\boldsymbol{x}_i)) - 1)^2$$

is the empirical *variance of $\tilde{w}$,*

$$\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta}\frac{1}{m}\sum_{i=1}^{m}\sigma_i\ell_{DSM}^{\theta}(\boldsymbol{x}_i)\right]$$

is the empirical *Rademacher complexity with respect to the parameter family Θ, and the last term $O\left(1/\sqrt[8]{m}\right)$ is independent of the model parameter θ and $\tilde{w}$.*

For completeness, here we outline the main steps of the proof. To prove (19), let us first fix a $\tilde{w} \in \tilde{\mathcal{W}}$. Given $\tilde{w}$, we can: i) apply the standard Hoeffding's inequality [Hoeffding, 1994] to obtain a data-dependent lower bound on $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})]$; and ii) apply Lemma 1 to obtain a PAC upper bound on $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\tilde{w}(f(\mathbf{x}))\ell_{\text{DSM}}^{\theta}(\mathbf{x})]$ with respect to the parameter family Θ. In light of Proposition 1, combining these two bounds gives us a *conditional* PAC lower bound on $J_{\text{opt}}(\theta)$ with respect to the parameter family Θ for any *fixed* $\tilde{w}$. Finally, we get rid of the conditioning on $\tilde{w}$ via the standard *covering* argument. The details of the proof can be found in Appendix A.2.

According to the PAC lower bound (19), in order to maximize $J_{\text{opt}}(\theta)$, we need to simultaneously maximize the *utility* of $\tilde{w}$ and minimize the *weighted generative loss* of $\boldsymbol{s}_{\theta}$ and the *variance* of $\tilde{w}$. Therefore, the PAC lower bound (19) captures both the utility and learnability considerations for selecting a normalized weight function $\tilde{w}$.

4 Algorithm

To jointly learn a normalized function $\tilde{w}$ and a score-function model $\boldsymbol{s}_t^\theta$, first note that the last two terms of the PAC lower bound (19) are independent of θ and $\tilde{w}$ and hence can be ignored from the learning objective. The forth term is due to the "initial" sampling error of the reverse process. As discussed previously in Section 2.2, while this term is potentially dependent on the normalized weight function $\tilde{w}$, in practice it can be made very small by choosing an appropriate noise scheduler β and hence will be ignored from our learning objective. To make the first three terms *learnable*, we consider the following two modifications to the bound.

First, the coefficients $c_0 K$ and $c_0 K \sqrt{2\Delta}$ in the second and the third term require some prior knowledge on the unknown data-generating distribution p_{data} and the unknown objective function f. In practice, we replace them by two *hyper-parameters* λ and α, respectively. We mention here that the hyper-parameter α plays a particular important role in the learning objective, as it controls the utility-learnability tradeoff for selecting a normalized weight function $\tilde{w}$.

Second, the weight function $\tilde{w}$ needs to be *normalized* with respect to the unknown data-generating distribution p_{data} and the unknown objective function f. In practice, we let $\tilde{w}(\cdot) = \frac{w_\phi(\cdot)}{Z_\phi}$, where w_ϕ is an *un-normalized* weight function parameterized by a second parameter ϕ, and $Z_\phi = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[w_\phi(f(\mathbf{x}))]$ is the normalizing constant. While the exact calculation of Z_ϕ again requires the knowledge of p_{data} and f, in practice it can be easily estimated from the offline data examples as $\hat{Z}_\phi = \frac{1}{m} \sum_{i=1}^m w_\phi(f(\boldsymbol{x}_i))$.

Incorporating the above changes to the PAC lower bound (19) leads to the following objective for jointly learning a (un-normalized) weight function w_ϕ and a score-function model $\boldsymbol{s}_t^\theta$:

$$J_{\alpha,\lambda}(\theta,\phi) = \frac{1}{m} \sum_{i=1}^m \frac{w_\phi(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i)}{\hat{Z}_\phi} - \lambda \sqrt{\frac{1}{m} \sum_{i=1}^m \frac{w_\phi(f(\boldsymbol{x}_i))\ell_{\text{DSM}}^\theta(\boldsymbol{x}_i)}{\hat{Z}_\phi}} - \alpha \sqrt[4]{\frac{1}{m} \sum_{i=1}^m \left(\frac{w_\phi(f(\boldsymbol{x}_i))}{\hat{Z}_\phi} - 1\right)^2}. \quad (20)$$

To maximize $J_{\alpha,\lambda}$, we consider the standard *alternating* procedure between maximizing over the parameter θ and maximizing over the parameter ϕ. One advantage of this strategy is that for a fixed ϕ, maximizing $J_{\alpha,\lambda}$ over θ is equivalent to minimizing the weighted denoising score matching loss $\frac{1}{m} \sum_{i=1}^m w_\phi(f(\boldsymbol{x}_i))\ell_{\text{DSM}}^\theta(\boldsymbol{x}_i)$ over θ, which is *additive* with respect to the training examples. Therefore, for a fixed ϕ, updating θ can be based on a very efficient *stochastic* estimate of the gradient. On the other hand, for a fixed θ, updating ϕ may have to rely on the much more cumbersome task of calculating the *exact* gradient. Fortunately, w_ϕ is only a *scalar* function, which makes the gradient calculation with respect to ϕ somewhat manageable. The detailed training and optimization procedures can be found in Appendix B.2.

5 Experimental Results

5.1 A toy example

We first experimentally validate the proposed learning algorithm using a toy example in $\mathbb{R}^2$. In this example, the unknown objective function f is a mixture of two Gaussian density functions:

$$f(\boldsymbol{x}) = 2\sqrt{3}\pi \left[0.45 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + 0.55 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right],$$

where $\boldsymbol{\mu}_1 = [1.5, 1.5]^t$, $\boldsymbol{\mu}_2 = [-1.5, -1.5]^t$, $\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$, and the unknown data-generating distribution p_{data} is a mixture of two Gaussian distributions:

$$p_{\text{data}}(\boldsymbol{x}) = 0.3 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_3, \boldsymbol{I}) + 0.7 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_4, \boldsymbol{I}),$$

where $\boldsymbol{\mu}_3 = [-4, -4]^t$, $\boldsymbol{\mu}_4 = [4, 4]^t$, and $\boldsymbol{I}$ is the 2×2 identity matrix. The weight function w_ϕ and the score-function model $\boldsymbol{s}_\theta$ are jointly learned by maximizing the proposed objective (20).

The filled contour plot of the objective function f is shown in Figure 1a, with warmer colors representing higher objective values. Figure 1b shows 300 samples drawn from the data-generating distribution p_{data}, with the color of each sample rendered according to its ground-truth objective value. These 300 samples and their corresponding objective values are the offline examples from which the weight function w_ϕ and the score-function model $\boldsymbol{s}_\theta$ are trained. Figure 2 shows the optimized samples and the learned weight function w_ϕ for several different values of the hyper-parameter α while fixing the hyper-parameter $\lambda = 0.1$.
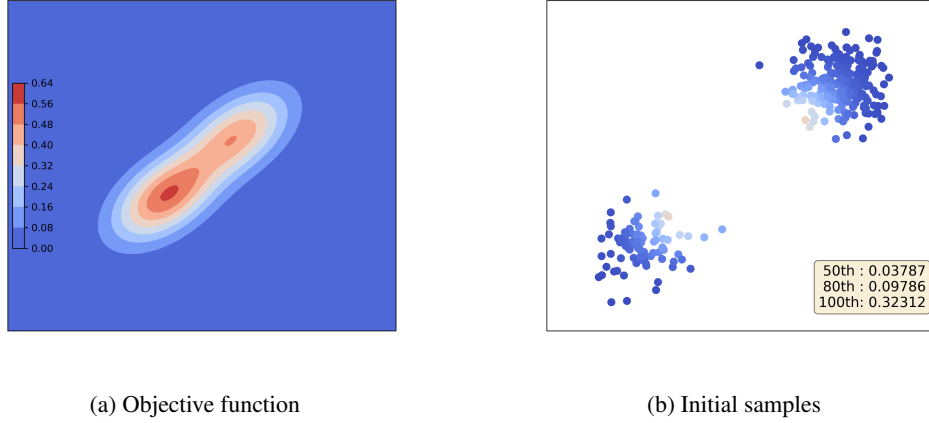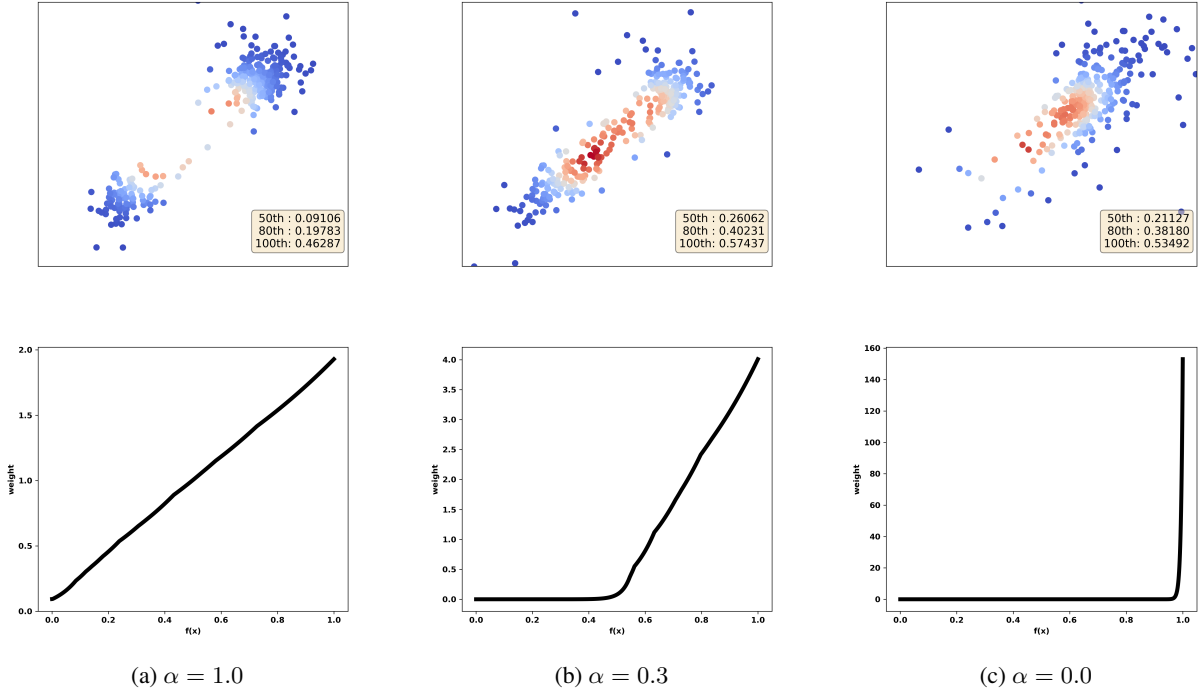
(a) Objective function   (b) Initial samples

Figure 1: A toy example: The objective function and the initial samples.

(a) $\alpha = 1.0$ (b) $\alpha = 0.3$ (c) $\alpha = 0.0$

Figure 2: A toy example: The optimized samples and the learned weight function for different values of α. Top: Optimized samples; Bottom: Learned weight function.

The legitimacy of the proposed approach is demonstrated by the following observations. i) Even though the weight function is *not* constrained to be monotonic *a priori*, as shown in Figure 2c, the learned weight functions are *monotone increasing* and hence put higher weights to samples with higher objective values. ii) When $\alpha = 1$, the learned weight function is relative "flat" across its input domain. As a result, the learned generative model is very close to the data-generating distribution, and the optimized samples are very "similar" to the initial samples. As we decrease the value of α from 1 to 0.3, the learned weight function becomes much more skewed towards the higher input values. As a result, some of the optimized samples have been nudged along the direction of the gradient of the objective function and hence have much higher objective values than the initial samples. When we further decrease the value of α to 0, the learned weight function becomes extremely skewed. In this case, the hypothetical target distribution is *not* learnable. As a result, instead of the gradient direction, the optimized samples have been nudged along *all* directions. Therefore, the hyper-parameter α can effectively control the *utility–learnability tradeoff* for selecting a weight function w_ϕ. iii)

Compared with the data-generating distribution, with an appropriate choice of the hyper-parameters α and λ, the learned generative model is substantially more *capable* of generating samples with higher objective values, as demonstrated by the differences of the 50th, 80th, and 100th percentiles between the samples drawn from these two distributions. More results on this toy example can be found in Appendix B.3.1.

5.2 Benchmark datasets

Next, we assess the performance of the proposed learning algorithm using the five standard tasks (Superconductor, TF Bind 8, Ant Morphology, GFP, and UTR) from the Design-Bench benchmark [Trabucco et al., 2022]. In addition, we have included the "Fluorescence" task from Fannjiang et al. [2022] for a comprehensive evaluation.

Evaluation. We generated a total of $N = 128$ designs for each task and subsequently computed the mean and standard deviation of the 100th percentile of the normalized ground truth over eight independent trials. The normalization process is defined by $y = (\tilde{y} - y_{\min})/(y_{\max} - y_{\min})$, where $y_{\min}$ and $y_{\max}$ are chosen from the *whole* dataset from which the offline examples were sampled.

Table 1: Experimental results on the benchmark datasets [↑].

	Supercond.	TFBind8	AntMorph.	GFP	UTR	Fluores.	Ave. Improv.
$\mathcal{D}_{\text{best}}$	0.399	0.439	0.565	0.789	0.593	0.485	
Grad	**0.518**±**0.024**	**0.977**±**0.025**	0.293±0.023	0.864±0.001	**0.695**±**0.013**	0.618±0.204	0.264
COMs	0.439±0.033	0.945±0.033	**0.944**±**0.016**	0.864±0.000	**0.699**±**0.011**	0.588±0.074	0.402
CbAS	**0.503**±**0.069**	0.927±0.051	0.876±0.031	**0.865**±**0.000**	**0.694**±**0.010**	0.574±0.020	0.395
Ours	**0.500**±**0.051**	0.953±0.038	0.844±0.023	**0.865**±**0.000**	**0.698**±**0.011**	**0.721**±**0.063**	**0.446**

Results. The results are listed in Table 1, where $\mathcal{D}_{\text{best}}$ denotes the normalized maximum objective value among the initial samples; "Grad" refers to the vanilla gradient ascent method, in which a surrogate of the unknown objective function is learned through supervised regression; "COMs" refers to the conservative objective models introduced in [Trabucco et al., 2021]; "CbAS" refers to the conditioning-by-adaptive-sampling method introduced in [Brookes et al., 2019]; and "Ours" refers to the method proposed in this paper with the hyper-parameters $\alpha = 0.2$ and $\lambda = 0.1$. Except for the Fluorescence task, the results of existing methods are all excerpted from Trabucco et al. [2021]. Results that fall within one standard deviation of the best performance are highlighted in bold. Across all tasks, our method demonstrates not only notable improvement over the best initial samples, but also *consistently competitive* performances against the other three prominent offline optimization algorithms. Quantitatively, our method achieves the *highest* average improvement over all six tasks, where the improvement over a specific task is defined as $(y - \mathcal{D}_{\text{best}})/\mathcal{D}_{\text{best}}$. We believe that this superior consistency is rooted in our *modeling* perspective and *principled* design of the learning algorithm. Implementation details and additional results can be found in Appendix B.2 and B.3.2.

6 Concluding Remarks

In this final section, we put the proposed *PAC-generative* approach to offline optimization in the context of several related work. This will help to further elucidate the main contribution of this paper.

Modeling target distribution *vs*. modeling objective function. As mentioned previously in Section 1, the "standard" approach to offline optimization is to first learn a surrogate of the unknown objective function and then apply existing optimization algorithms. The main challenge for modeling the objective function is the so-called *distributional shift*. That is, when the optimization algorithm explores regions *away* from the offline observations, the leaned surrogate tends to become less accurate. It is thus crucial to understand how far the optimization algorithm can explore away from the offline observations and how to maintain the accuracy of the learned surrogate throughout the exploration process. Notable effort in the literature include Qi et al. [2022] and Trabucco et al. [2021], which considered regularized surrogate models in favor of invariance and conservatism; Fannjiang and Listgarten [2020] and Chen et al. [2022], which considered surrogate models learned via importance sampling and contrastive learning; and Fannjiang et al. [2022], which used conformal prediction to quantify the uncertainty of the learned surrogate.

Despite these effort, however, it remains unclear how to align the objective of learning a surrogate of the unknown objective function with the objective of optimization. This is evidenced by the very recent work Beckham and Pal [2023], which discussed how one may interpret the conservative approach proposed inTrabucco et al. [2021], and Beckham et al. [2023], which suggested that an alternative evaluation metric is potentially better than simply choosing the best candidates using the learned surrogate. In contrast, the PAC-generative approach proposed in this paper is

based on modeling a target distribution (as opposed to the objective function). As we have shown, under this generative view, it is possible to tune the objective of the learner according to a natural optimization objective.

Weighted learning *vs*. conditional/guided generation. Recent years have seen remarkable success in conditional/guided image generation [Dhariwal and Nichol, 2021, Ho and Salimans, 2022]. Conditional/guided generation can be easily adapted to the offline optimization setting. More specifically, to learn a generative model for the purpose of offline optimization, one can simultaneously learn a *standard* score-based generative model and a surrogate of the objective function and then use the *gradient* of the learned surrogate to guide the generation of the optimized samples [Mazé and Ahmed, 2022]. Alternatively, one may also model the target distribution q_{target} as the *conditional* distribution p_{data} given $f(\mathbf{x}) \geq y_0$ for some *threshold* y_0 and train a generative model that approximates this conditional distribution [Brookes et al., 2019, Gupta and Zou, 2019]. However, learning the conditional distribution p_{data} given $f(\mathbf{x}) \geq y_0$ may also require a surrogate of the objective function. In contrast, in our approach we model the target distribution q_{target} using a weight function. As we have previously discussed in Section 2.2, in our *weighted-learning* model, the score of q_{target} is *intrinsically* aligned with the gradient of the objective function. In our approach, we directly train a generative model from the offline data examples to learn the score of q_{target}, and there is *no* need to learn a surrogate of the objective function separately.

Offline optimization *vs*. offline reinforcement learning (RL). While the focus of this paper is offline optimization, recent years have also seen a substantial amount of interest in *offline RL* [Kumar et al., 2020, Wang et al., 2022, Janner et al., 2022, Yuan et al., 2023]. Even though these two problems face some similar challenges, in our evaluation offline RL is the considerably more challenging setting. It is thus of interest to see whether the proposed PAC-generative approach can lead to any success in offline RL as well.

References

Sathvik Kolli, Amy X Lu, Xinyang Geng, Aviral Kumar, and Sergey Levine. Data-driven optimization for protein design: Workflows, algorithms and metrics. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José M Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Nathan Killoran, Leo J Lee, Andrew Delong, David Duvenaud, and Brendan J Frey. Generating and designing DNA with deep generative models. *arXiv preprint arXiv:1712.06148*, 2017.

Warren Hoburg and Pieter Abbeel. Geometric programming for aircraft design optimization. *AIAA Journal*, 52(11): 2414–2426, 2014.

Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation*, pages 2488–2494, 2019.

Aviral Kumar, Amir Yazdanbakhsh, Milad Hashemi, Kevin Swersky, and Sergey Levine. Data-driven offline optimization for architecting hardware accelerators. In *International Conference on Learning Representations*, 2022.

Amir Beck. *First-Order Methods in Optimization*. SIAM, 2017.

Charles Audet and Warren Hare. *Introduction: Tools and Challenges in Derivative-Free and Blackbox Optimization*, pages 3–14. Springer International Publishing, 2017.

Justin Fu and Sergey Levine. Offline model-based optimization via normalized maximum likelihood estimation. In *International Conference on Learning Representations*, 2021.

David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International Conference on Machine Learning*, pages 773–782. PMLR, 2019.

Anvita Gupta and James Zou. Feedback GAN for DNA optimizes protein functions. *Nature Machine Intelligence*, 1(2): 105–111, 2019.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *Advances in Neural Information Processing Systems*, 33:5126–5137, 2020.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1:127–190, 1999.

Luigi Ambrosio, Elia Brué, and Daniele Semola. *Lecture 3: The Kantorovich–Rubinstein Duality*, pages 23–34. Springer International Publishing, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020a.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Brian D Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3): 313–326, 1982.

Vladimir I Bogachev, Nicolai V Krylov, Michael Röckner, and Stanislav V Shaposhnikov. *Fokker–Planck–Kolmogorov Equations*, volume 207. American Mathematical Society, 2022.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7): 1661–1674, 2011.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020b.

Dohyun Kwon, Ying Fan, and Kangwook Lee. Score-based generative modeling secretly minimizes the wasserstein distance. *Advances in Neural Information Processing Systems*, 35:20205–20217, 2022.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pages 21658–21676. PMLR, 2022.

Clara Fannjiang, Stephen Bates, Anastasios N Angelopoulos, Jennifer Listgarten, and Michael I Jordan. Conformal prediction under feedback covariate shift for biomolecular design. *Proceedings of the National Academy of Sciences*, 119(43):e2204569119, 2022.

Han Qi, Yi Su, Aviral Kumar, and Sergey Levine. Data-driven offline decision-making via invariant representation learning. *Advances in Neural Information Processing Systems*, 35:13226–13237, 2022.

Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. *Advances in Neural Information Processing Systems*, 33:12945–12956, 2020.

Can Chen, Yingxue Zhang, Jie Fu, Xue S Liu, and Mark Coates. Bidirectional learning for offline infinite-width model-based optimization. *Advances in Neural Information Processing Systems*, 35:29454–29467, 2022.

Christopher Beckham and Christopher Pal. Conservative objective models are a special kind of contrastive divergence-based energy model. *arXiv preprint arXiv:2304.03866*, 2023.

Christopher Beckham, Alexandre Piche, David Vazquez, and Christopher Pal. Exploring validation metrics for offline model-based optimisation. *arXiv preprint arXiv:2304.03866*, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

François Mazé and Faez Ahmed. Topodiff: A performance and constraint-guided diffusion model for topology optimization. *arXiv preprint arXiv:2208.09591*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Hui Yuan, Kaixuan Huang, Chengzhuo Ni, Minshuo Chen, and Mengdi Wang. Reward-directed conditional diffusion: Provable distribution estimation and reward improvement. *arXiv preprint arXiv:2307.07055*, 2023.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A   Proof of Technical Results

A.1   Proof of Lemma 1

By assumption, we have $0 \leq \tilde{w}(\boldsymbol{x}) \leq B$ for any $\boldsymbol{x} \in \mathcal{X}$ and $0 \leq \ell_\theta(\boldsymbol{x}) \leq \Delta$ for any $\boldsymbol{x} \in \mathcal{X}$ and any $\theta \in \Theta$. It follows immediately that the *weighed* loss function $\tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x})$ satisfies $0 \leq \tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x}) \leq B\Delta$ for any $\boldsymbol{x} \in \mathcal{X}$ and any $\theta \in \Theta$. Applying the standard Rademacher bound to the *weighted* loss function class $(\tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x}) : \theta \in \Theta)$, with probability $\geq 1 - \delta$ we have for any $\theta \in \Theta$

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\tilde{w}(\mathbf{x})\ell_\theta(\mathbf{x})] \leq \hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) + 2\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) + 3\sqrt{\frac{2B\Delta\log(2/\delta)}{m}}, \tag{21}$$

where $\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) = \frac{1}{m}\sum_{i=1}^m \tilde{w}(\boldsymbol{x}_i)\ell_\theta(\boldsymbol{x}_i)$ is the empirical weighted loss over $\boldsymbol{x}_{[m]}$, and $\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i \tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x}_i)\right]$ is the empirical *weighted* Rademacher complexity with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$. The empirical weighted Rademacher complexity $\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta)$ can be further bounded from above as:

$$\begin{aligned}
\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) &= \mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i \left(w(\boldsymbol{x}_i) - 1 + 1\right)\ell_\theta(\boldsymbol{x}_i)\right] \\
&\leq \mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i(\tilde{w}(\boldsymbol{x}_i) - 1)\ell_\theta(\boldsymbol{x}_i)\right] + \mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i \ell_\theta(\boldsymbol{x}_i)\right].
\end{aligned} \tag{22}$$

Further note that

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^m \sigma_i(\tilde{w}(\boldsymbol{x}_i) - 1)\ell_\theta(\boldsymbol{x}_i) &\leq \sqrt{\frac{\sum_{i=1}^m (\tilde{w}(\boldsymbol{x}_i) - 1)^2}{m}}\sqrt{\frac{\sum_{i=1}^m (\sigma_i \ell_\theta(\boldsymbol{x}_i))^2}{m}} \\
&= \sqrt{\frac{\sum_{i=1}^m (\tilde{w}(\boldsymbol{x}_i) - 1)^2}{m}}\sqrt{\frac{\sum_{i=1}^m \ell_\theta^2(\boldsymbol{x}_i)}{m}} \\
&\leq \Delta\sqrt{\frac{\sum_{i=1}^m (\tilde{w}(\boldsymbol{x}_i) - 1)^2}{m}}
\end{aligned}$$

for any $\theta \in \Theta$ and any realization of $\boldsymbol{\sigma}_{[m]}$, where the first inequality follows from the standard Cauchy-Schwarz inequality, the second equality follows from the fact that the square of a Rademacher variable takes a constant value of 1, and the last inequality follows from the assumption that $0 \leq \ell_\theta(\boldsymbol{x}) \leq \Delta$ for any $\boldsymbol{x} \in \mathcal{X}$ and any $\theta \in \Theta$. It follows immediately that

$$\mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i(\tilde{w}(\boldsymbol{x}_i) - 1)\ell_\theta(\boldsymbol{x}_i)\right] \leq \Delta\sqrt{\frac{\sum_{i=1}^m (\tilde{w}(\boldsymbol{x}_i) - 1)^2}{m}}. \tag{23}$$

Substituting (23) into (22) gives

$$\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) \leq \Delta\sqrt{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} + \hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta), \tag{24}$$

where $\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^m \left(\tilde{w}(\boldsymbol{x}_i) - 1\right)^2$ is the empirical variance of $\tilde{w}$ over $\boldsymbol{x}_{[m]}$, and $\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i \ell_\theta(\boldsymbol{x}_i)\right]$ is the empirical (un-weighted) Rademacher complexity with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$. Substituting (24) into (21) completes the proof of (13) and hence Lemma 1.

A.2   Proof of Theorem 1

For the proof, we shall write q_{target} and $\bar{q}_{\text{target}}$ as $q^{\tilde{w}}_{\text{target}}$ and $\bar{q}^{\tilde{w}}_{\text{target}}$ respectively, to emphasize their dependencies on the normalized weight function $\tilde{w}$. Let us first recall from Proposition 1 that the natural optimization objective $J_{\text{opt}}(\theta)$ can be bounded from below as:

$$J_{\text{opt}}(\theta) \geq \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\tilde{w}(f(\mathbf{x}))f(\mathbf{x})] - c_0 K\sqrt{\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\tilde{w}(f(\mathbf{x}))\ell^\theta_{\text{DSM}}(\mathbf{x})]} - c_1 K W_2(\bar{q}^{\tilde{w}}_{\text{target}}, \mathcal{N}). \tag{25}$$

To turn the right-hand side into a *PAC* lower bound on $J_{\text{opt}}(\theta)$, let us first *fix* a normalized weight function $\tilde{w} \in \tilde{\mathcal{W}}$.

Given $\tilde{w}$, let us first apply the standard Hoeffding's inequality to obtain a *concentration* lower bound on the expected utility $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\tilde{w}(f(\mathbf{x}))f(\mathbf{x})]$. More specifically, by assumption we have $|f(\boldsymbol{x})| \leq F$ for any $\boldsymbol{x} \in \mathcal{X}$ and $0 \leq \tilde{w}(y) \leq$

B for any $y \in [-F, F]$. It follows that the *weighed* objective function $\tilde{w}(f(\boldsymbol{x}))f(\boldsymbol{x})$ satisfies $|\tilde{w}(f(\boldsymbol{x}))f(\boldsymbol{x})| \leq BF$ for any $\boldsymbol{x} \in \mathcal{X}$. By Hoeffding's inequality, with probability $\geq 1 - \delta'/2$ we have

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\tilde{w}(f(\mathbf{x}))f(\mathbf{x})] \geq \hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - \sqrt{\frac{BF \log(2/\delta')}{m}}, \tag{26}$$

where $\hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m} \tilde{w}(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i)$ is the empirical utility of $\tilde{w}$. Next by Lemma (1), with probability $\geq 1 - \delta'/2$ we have for *any $\theta \in \Theta$*

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\tilde{w}(f(\mathbf{x}))\ell_{\text{DSM}}^\theta(\mathbf{x}) \right] \leq \hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{w}) + 2\Delta\sqrt{\hat{V}_{\boldsymbol{x}[m]}(\tilde{w})} + 2\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta) + 3\sqrt{\frac{2B\Delta \log(4/\delta')}{m}}$$

and hence

$$
\begin{aligned}
\sqrt{\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\tilde{w}(f(\mathbf{x}))\ell_{DSM}^\theta(\mathbf{x}) \right]} &\leq \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{w}) + 2\Delta\sqrt{\hat{V}_{\boldsymbol{x}[m]}(\tilde{w})} + 2\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta) + 3\sqrt{\frac{2B\Delta \log(4/\delta')}{m}}} \\
&\leq \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{w})} + \sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}[m]}(\tilde{w})} + \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta)} + \sqrt[4]{\frac{18B\Delta \log(4/\delta')}{m}}, \tag{27}
\end{aligned}
$$

where $\hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{w}) = \frac{1}{m}\sum_{i=1}^{m} \tilde{w}(\boldsymbol{x}_i)\ell_{\text{DSM}}^\theta(\boldsymbol{x}_i)$ is the empirical weighted denoising score matching loss of s_θ over $\boldsymbol{x}_{[m]}$, $\hat{V}_{\boldsymbol{x}[m]}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m} (\tilde{w}(\boldsymbol{x}_i) - 1)^2$ is the empirical variance of $\tilde{w}$ over $\boldsymbol{x}_{[m]}$, and $\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}} \left[\sup_{\theta \in \Theta} \frac{1}{m}\sum_{i=1}^{m} \sigma_i\ell_{\text{DSM}}^\theta(\boldsymbol{x}_i) \right]$ is the empirical Rademacher complexity with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$. Substituting (26) and (27) into (25), with probability $\geq 1 - \delta'$ we have for *any $\theta \in \Theta$*

$$
\begin{aligned}
J_{\text{opt}}(\theta) \geq &\ \hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{w})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}[m]}(\tilde{w})} - c_1 K W_2(\bar{q}_{\text{target}}^{\tilde{w}}, \mathcal{N}) - \\
&\ c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta)} - c_0 K\sqrt[4]{\frac{18B\Delta \log(4/\delta')}{m}} - \sqrt{\frac{BF \log(2/\delta')}{m}}. \tag{28}
\end{aligned}
$$

To remove the *conditioning* on $\tilde{w}$, let $\tilde{\mathcal{W}}_\epsilon$ be an ϵ-*cover* of $\tilde{\mathcal{W}}$ under the L_∞ norm. By (28), for any *given $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$*, with probability $\geq 1 - \delta'$ we have for *any $\theta \in \Theta$*

$$
\begin{aligned}
J_{\text{opt}}(\theta) \geq &\ \hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) - c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{v})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}[m]}(\tilde{v})} - c_1 K W_2(\bar{q}_{\text{target}}^{\tilde{v}}, \mathcal{N}) - \\
&\ c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta)} - c_0 K\sqrt[4]{\frac{18B\Delta \log(4/\delta')}{m}} - \sqrt{\frac{BF \log(2/\delta')}{m}}.
\end{aligned}
$$

By the *union* bound, with probability $\geq 1 - |\tilde{\mathcal{W}}_\epsilon|\delta'$ we have for *any $\theta \in \Theta$ and any $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$*

$$
\begin{aligned}
J_{\text{opt}}(\theta) \geq &\ \hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) - c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{v})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}[m]}(\tilde{v})} - c_1 K W_2(\bar{q}_{\text{target}}^{\tilde{v}}, \mathcal{N}) - \\
&\ c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta)} - c_0 K\sqrt[4]{\frac{18B\Delta \log(4/\delta')}{m}} - \sqrt{\frac{BF \log(2/\delta')}{m}}.
\end{aligned}
$$

Choosing $\delta' = \delta/|\tilde{\mathcal{W}}_\epsilon|$, with probability $\geq 1 - \delta$ we have for *any $\theta \in \Theta$ and any $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$*

$$
\begin{aligned}
J_{\text{opt}}(\theta) \geq &\ \hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) - c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}[m]}(\theta, \tilde{v})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}[m]}(\tilde{v})} - c_1 K W_2(\bar{q}_{\text{target}}^{\tilde{v}}, \mathcal{N}) - \\
&\ c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}[m]}(\Theta)} - c_0 K\sqrt[4]{\frac{18B\Delta \log(4|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}} - \sqrt{\frac{BF \log(2|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}}. \tag{29}
\end{aligned}
$$

By the definition of ϵ-cover, for any $\tilde{w} \in \tilde{\mathcal{W}}$, there exists an $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$ such that $|\tilde{w}(f(\boldsymbol{x})) - \tilde{v}(f(\boldsymbol{x}))| \leq \epsilon$ for any $\boldsymbol{x} \in \mathcal{X}$. Note that this immediately implies that:

$$
\begin{aligned}
\hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - \hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) &= \frac{1}{m}\sum_{i=1}^{m} \left(\tilde{w}(f(\boldsymbol{x}_i)) - \tilde{v}(f(\boldsymbol{x}_i)) \right) f(\boldsymbol{x}_i) \\
&\leq \frac{1}{m}\sum_{i=1}^{m} |\tilde{w}(f(\boldsymbol{x}_i)) - \tilde{v}(f(\boldsymbol{x}_i))| \, |f(\boldsymbol{x}_i)| \leq F\epsilon, \tag{30}
\end{aligned}
$$

where the last inequality follows from the assumption that $|f(\boldsymbol{x})| \leq F$ for any $\boldsymbol{x} \in \mathcal{X}$;

$$\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{v})} - \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\tilde{v}(f(\boldsymbol{x}_i))\ell_{\mathrm{DSM}}^{\theta}(\boldsymbol{x}_i)} - \sqrt{\frac{1}{m}\sum_{i=1}^{m}\tilde{w}(f(\boldsymbol{x}_i))\ell_{\mathrm{DSM}}^{\theta}(\boldsymbol{x}_i)}$$

$$\leq \sqrt{\frac{1}{m}\sum_{i=1}^{m}|\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i))|\,\ell_{\mathrm{DSM}}^{\theta}(\boldsymbol{x}_i)} \leq \sqrt{\Delta\epsilon}, \quad (31)$$

where the last inequality follows from the assumption that $0 \leq \ell_{\mathrm{DSM}}^{\theta}(\boldsymbol{x}) \leq \Delta$ for any $\boldsymbol{x} \in \mathcal{X}$;

$$\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{v})} - \sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} = \sqrt[4]{\frac{1}{m}\sum_{i=1}^{m}(\tilde{v}(f(\boldsymbol{x}_i)) - 1)^2} - \sqrt[4]{\frac{1}{m}\sum_{i=1}^{m}(\tilde{w}(f(\boldsymbol{x}_i)) - 1)^2}$$

$$= \sqrt[4]{\frac{1}{m}\sum_{i=1}^{m}(\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i)) + \tilde{w}(f(\boldsymbol{x}_i)) - 1)^2} - \sqrt[4]{\frac{1}{m}\sum_{i=1}^{m}(\tilde{w}(f(\boldsymbol{x}_i)) - 1)^2}$$

$$\leq \sqrt[4]{\frac{1}{m}\sum_{i=1}^{m}(\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i)))^2} + \sqrt[4]{\frac{2}{m}\sum_{i=1}^{m}|\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i))|\,|\tilde{w}(f(\boldsymbol{x}_i)) - 1|}$$

$$\leq \sqrt{\epsilon} + \sqrt[4]{2(B+1)\epsilon}, \quad (32)$$

where the last inequality follows from the assumption that $0 \leq \tilde{w}(f(\boldsymbol{x})) \leq B$ for any $\boldsymbol{x} \in \mathcal{X}$; and

$$W_2(\bar{q}_{\mathrm{target}}^{\tilde{v}}, \mathcal{N}) - W_2(\bar{q}_{\mathrm{target}}^{\tilde{w}}, \mathcal{N}) \leq W_2(\bar{q}_{\mathrm{target}}^{\tilde{v}}, \bar{q}_{\mathrm{target}}^{\tilde{w}})$$

$$\leq c_2 W_2(q_{\mathrm{target}}^{\tilde{v}}, q_{\mathrm{target}}^{\tilde{w}})$$

$$\leq c_2 d_2(\mathcal{X}) d_{\mathrm{TV}}(q_{\mathrm{target}}^{\tilde{v}}, q_{\mathrm{target}}^{\tilde{w}})$$

$$= \frac{1}{2} c_2 d_2(\mathcal{X}) \int_{\mathcal{X}} |\tilde{v}(f(\boldsymbol{x})) - \tilde{w}(f(\boldsymbol{x}))| p_{\mathrm{data}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

$$\leq \frac{1}{2} c_2 d_2(\mathcal{X}) \epsilon, \quad (33)$$

where c_2 is the *Wasserstein contraction constant* of the forward process (7), $d_2(\mathcal{X}) := \max_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}'\|_2$ is the *diameter* of $\mathcal{X}$ with respect to the ℓ_2 norm, and $d_{\mathrm{TV}}(q_{\mathrm{target}}^{\tilde{v}}, q_{\mathrm{target}}^{\tilde{w}})$ denotes the *total variation distance* between $q_{\mathrm{target}}^{\tilde{v}}$ and $q_{\mathrm{target}}^{\tilde{w}}$. Here, the first inequality follows from the fact that the 2-Wasserstein distance is a metric and hence follows the triangle inequality, the second inequality follows from the *Wasserstein contraction* property of the forward process, and the third inequality follows from the *total-variation* bound on the 2-Wasserstein distance. Substituting (30)–(33) into (29), with probability $\geq 1 - \delta$ we have for *any* $\theta \in \Theta$ and *any* $\tilde{w} \in \tilde{\mathcal{W}}$

$$J_{\mathrm{opt}}(\theta) \geq \hat{J}_{\boldsymbol{x}_{[m]}}(\tilde{w}) - c_0 K \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})} - c_0 K \sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} - c_1 K W_2(\bar{q}_{\mathrm{target}}^{\tilde{w}}, \mathcal{N}) -$$

$$c_0 K \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - c_0 K \sqrt[4]{\frac{18 B \Delta \log(4|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}} - \sqrt{\frac{BF \log(2|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}} -$$

$$\left(F + \frac{1}{2} c_2 d_2(\mathcal{X})\right)\epsilon - \left(\sqrt{\Delta} + 1\right)\sqrt{\epsilon} - \sqrt[4]{2(B+1)\epsilon}. \quad (34)$$

By assumption, any normalized weight function from $\tilde{\mathcal{W}}$ is L-Lipschitz and bounded by B. Therefore, the covering number $|\tilde{\mathcal{W}}_\epsilon|$ is of the order $O(\exp(1/\epsilon))$. Let $\epsilon = m^{-\gamma}$ for some $\gamma \in (0, 1)$, and we have from (34)

$$J_{\mathrm{opt}}(\theta) \geq \hat{J}_{\boldsymbol{x}_{[m]}}(\tilde{w}) - c_0 K \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})} - c_0 K \sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} - c_1 K W_2(\bar{q}_{\mathrm{target}}^{\tilde{w}}, \mathcal{N}) -$$

$$c_0 K \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - O(m^{-(1-\gamma)/4}) - O(m^{-\gamma/4}). \quad (35)$$

Choosing $\gamma = 1/2$ in (35) completes the proof of (19) and hence Theorem 1.

Table 2: The benchmark datasets

	Supercond.	TFBind8	AntMorph.	GFP	UTR	Fluores.
Type	continuous	discrete	continuous	discrete	discrete	discrete
Dimension	86	8	60	237	50	13
Category	N/A	4	N/A	20	4	2
# Train/Total	17014/21263	32898/65792	10004/25009	5000/56086	140k/280k	4096/8192
Min/Max	0.0/185.0	0.0/1.0	-386.9/590.2	1.283/4.123	0.0/12.0	0.155/1.692
$\mathcal{D}_{\text{best}}$	74.0/0.4	0.439/0.439	165.326/0.565	3.525/0.789	7.123/0.594	0.900/0.485

B Detailed Experimental Results

B.1 Benchmark datasets

We conducted experiments on six standard offline optimization tasks:

- **Superconductor**, which aims to design a superconductor with 86 components to maximize the critical temperature;

- **TF (Transcription Factor) Bind 8**, which aims to find a DNA sequence of 8 base pairs to maximize its binding affinity to a particular transcription factor;

- **Ant Morphology**, which aims to design the morphology of a quadrupedal ant with 60 components to enable rapid crawling;

- **GFP (Green Fluorescent Protein)**, which aims to find a protein sequence of 238 amino acids to maximize the fluorescence;

- **UTR (Untranslated Region)**, which aims to find a human 5' UTR DNA sequence of 50 base pairs to maximize the expression level of its corresponding gene;

- **Fluorescence**, which aims to identify a protein with high brightness. At each position, the selection of an amino acid is limited to those found in the sequences of the two parent fluorescent proteins. These parent proteins differ at precisely 13 positions in their sequences while being identical at all other positions.

For all previous tasks except for the Fluorescence, we utilized the Design-Bench package [Trabucco et al., 2022] to generate the training data, pre-process the data (including the conversion of categorical features to numerical values), and evaluate new designs. For the Fluorescence task, we collected raw data from Fannjiang et al. [2022]. The objective value in this case is represented by the combined brightness. From a total of $2^{13} = 8192$ samples, we selected the worst 4096 examples as our training dataset. While the features in the Fluorescence dataset are binary, we simply treated them as continuous inputs to our algorithm.

The key attributes of the aforementioned benchmark datasets can be found in Table 2, which include:

- **Type**: The type of features represented in the dataset, which can be either continuous or discrete;

- **Dimension**: The feature dimension of the dataset;

- **Category**: The number of categories for each feature (only applicable to the discrete datasets);

- **# Train/Total**: The number of samples in the training and entire datasets. The entire dataset includes both the training dataset and additional data examples, which are used to help evaluate the new designs;

- **Min/Max**: The minimum and maximum objective values within the entire dataset;

- **$\mathcal{D}_{\text{best}}$**: The un-normalized and normalized maximum objective values within the training dataset.

B.2 Implementation details

Normalization. As we adopted DDPM as our generative model, we normalized each feature to the interval $[-1, 1]$. For the objective values, we mapped the original values in the training dataset to the range of $[0, 1]$. This step ensures consistency in the learning of the (un-normalized) weight function w_ϕ. For the GFP task, we employed a variational auto-encoder [Kingma and Welling, 2013] to embed the high-dimensional features into a lower-dimensional space before normalizing them into the interval $[-1, 1]$.

Networks. In our implementation, we used neural networks to model both the (un-normalized) weight function w_ϕ and the score function s_t^θ. The weight function is a simple *scalar* function. In our implementation, we simply used a 4-layer multi-layer perceptron (MLP) with ReLU activation functions. In addition, we applied an *exponential* function to the output of the MLP to enforce the *non-negativity* of the weight function. The architecture for the score function model consists of a time-embedding layer and five blocks of "Dense-BatchNorm-ELU". Before each block, we injected time-embedding information by concatenating it with the input to the block.

Training. The noise scheduler for the DDPM was chosen as $\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min})t$ for $t \in [0, 1]$, where $\beta_{\min} = 0.1$ and $\beta_{\max} = 20$. The detailed training procedure is described in Algorithm 1. The training scheme involves first identifying a suitable *initialization* of ϕ and θ and then followed by an *alternating* maximization over ϕ and θ. More specifically, to obtain a suitable initialization of ϕ and θ, we first note that the model θ *only* shows up in the second term of our learning objective (20). Maximizing the other two terms over ϕ gives us an initial estimate ϕ_0 (see Line 1 of Algorithm 1). In our implementation, this maximization was performed via full-batch gradient descent (GD), for which we used the Adam optimizer [Kingma and Ba, 2014] with a constant leaning rate 10^{-3}. Once an initial estimate ϕ_0 has been obtained, we can obtain an initial estimate θ_0 by minimizing the second term over θ while setting $\phi = \phi_0$ (see Line 2 of Algorithm 1). To minimize the weighted denoising score matching loss, we considered a time range of $t \in [10^{-3}, 1]$ and used the Adam optimizer with a variable learning rate via stochastic gradient descent (SGD). The learning rate was gradually decreased from 10^{-3} to 10^{-4} during training. The alternating maximization of the learning objective (20) over the parameters ϕ and θ is described in Line 3–6 of Algorithm 1. Again the Adam optimizer was used, and the learning rates were set as $\eta_1 = \eta_2 = 10^{-4}$.

Algorithm 1 TRAINING

Input: Offline examples $((\boldsymbol{x}_i, f(\boldsymbol{x}_i)) : i \in [m])$; hyper-parameters α, λ; learning-rate parameters η_1, η_2.
General step:

1: $\phi_0 \leftarrow \arg\max_{\phi \in \Phi} \left\{ \frac{1}{m} \sum_{i=1}^{m} \frac{w_\phi(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i)}{\hat{Z}_\phi} - \alpha \sqrt[4]{\frac{1}{m} \sum_{i=1}^{m} \left(\frac{w_\phi(f(\boldsymbol{x}_i))}{\hat{Z}_\phi} - 1 \right)^2} \right\}$ $\triangleright$ via GD

2: $\theta_0 \leftarrow \arg\min_{\theta \in \Theta} \left\{ \frac{1}{m} \sum_{i=1}^{m} \frac{w_{\phi_0}(f(\boldsymbol{x}_i)) \cdot \ell_{\text{DSM}}^\theta(\boldsymbol{x}_i)}{\hat{Z}_{\phi_0}} \right\}$ $\triangleright$ via SGD

3: **for** $k = 0, 1, \ldots, K - 1$, **do**

4:      $\phi_{k+1} \leftarrow \phi_k + \eta_1 \cdot \nabla_\phi J_{\alpha, \lambda}(\theta_k, \phi_k)$ $\triangleright$ via GD

5:      $\theta_{k+1} \leftarrow \theta_k - \eta_2 \cdot \nabla_\theta \left\{ \frac{1}{m} \sum_{i=1}^{m} \frac{w_{\phi_{k+1}}(f(\boldsymbol{x}_i)) \cdot \ell_{\text{DSM}}^{\theta_k}(\boldsymbol{x}_i)}{\hat{Z}_{\phi_{k+1}}} \right\}$ $\triangleright$ via SGD

6: **end for**

Output: Model parameters $(\phi^*, \theta^*) = (\phi_K, \theta_K)$.

Sampling/Optimization. The sampling/optimization procedure is described in Algorithm 2. This procedure is identical to the probability-flow sampler in Song et al. [2020a].

Algorithm 2 SAMPLING/OPTIMIZATION

Input: Score function model $s_t^{\theta^*}(\boldsymbol{x})$, number of samples N, number of steps T, noise scheduler parameters $(\beta_{\min}, \beta_{\max})$, and $\tilde{\beta}(t) = \frac{1}{T} \left[\beta_{\min} + \frac{t}{T}(\beta_{\max} - \beta_{\min}) \right]$.
General step:

1: Draw N samples $\boldsymbol{x}_T^{(1)}, \boldsymbol{x}_T^{(2)}, \ldots, \boldsymbol{x}_T^{(N)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \boldsymbol{I})$

2: **for** $n = 1, 2, \ldots, N$, **do**

3:      **for** $t = T, T - 1, \ldots, 1$, **do**

4:          $\boldsymbol{x}_{t-1}^{(n)} \leftarrow \left(2 - \sqrt{1 - \tilde{\beta}(t)} \right) \cdot \boldsymbol{x}_t^{(n)} + \frac{1}{2}\tilde{\beta}(t) \cdot s_{t/T}^{\theta^*}(\boldsymbol{x}_t^{(n)})$

5:      **end for**

6: **end for**

Output: Optimized samples $\boldsymbol{x}_0^{(1)}, \boldsymbol{x}_0^{(2)}, \ldots, \boldsymbol{x}_0^{(N)}$.

B.3 Additional experimental results

B.3.1 Toy example

Additional choices of the hyper-parameter α. Previously in Section 5.1, we described a toy example in $\mathbb{R}^2$ and used it to validate our proposed approach. In particular, in Figure 2 we reported the optimized samples and the learned weight function w_{ϕ^*} for several choices of the hyper-parameter α. Here in Figures 3 and 4 we report the optimized samples and the learned weight function w_{ϕ^*} for some additional choice of the hyper-parameter α. Note that when $\alpha = \infty$, the learned weight function w_{ϕ^*} is completely flat across its domain, and thus the hypothetical target distribution q_{target} is identical to the data-generating distribution p_{data}. It should become very clear from these reported results that the hyper-parameter α can effectively control the utility-learnability tradeoff for selecting a weight function.
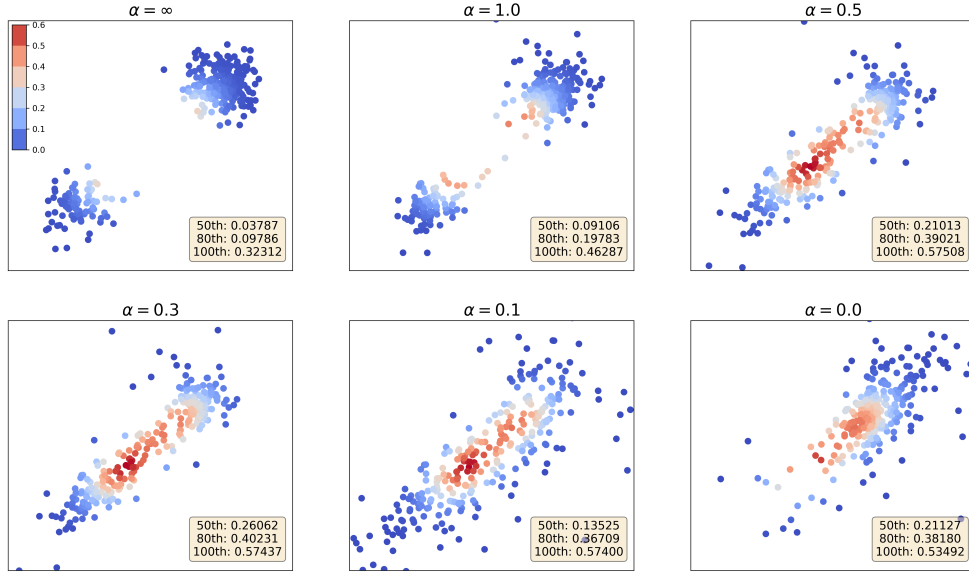

Figure 3: Optimized samples (with trainable weight function) for different choices of the hyper-parameter α.

Predefined weight function. Instead of considering a *trainable* weight function w_ϕ, we may also consider using a *predefined* weight function to train the generative model p_{target}^θ. Motivated by the learned weight functions w_{ϕ^*} from Figure 4, here we consider the simple *exponential* function $w_\psi(y) = \exp(\psi y)$, where ψ is a hyper-parameter. Note that when $\psi = 0$, the weight function w_ψ is completely flat across its domain, and as we increase the value of ψ, w_ψ becomes increasingly skewed towards the higher values in its domain. The optimized samples and the corresponding predefined weight functions are reported in Figures 5 and 6. Note here that we have purposely chosen the values of the hyper-parameter ψ such that the predefined weight functions w_ψ in Figure 6 *mimic* the learned weight function w_{ϕ^*} in Figure 4. As a result, the optimized samples from Figures 5 have similar statistical profiles as those from Figures 3. Next, we shall use the benchmark datasets to illustrate that a trainable weight function can significantly outperform a predefined weight function in terms of generating samples with a consistent and superior statistical profile.

B.3.2 Benchmark datasets

Here we report additional results on the benchmark datasets using both the trainable weight function w_ϕ and the predefined exponential weight function w_ψ. In our experiments, we fixed the value of the hyper-parameter $\lambda = 0.1$ and considered several different values for the hyper-parameter α (trainable weight function) and ψ (predefined weight function). The mean and standard deviation of the *best* generated samples for each benchmark dataset are reported in Table 3. The average improvements for different choices of the hyper-parameter α (trainable weight function) and ψ (predefined weight function) are reported in Table 4. (All values reported in Tables 3 and 4 are based on *un-normalized* objective values.) It is clear that the use of a trainable weight function with $\alpha = 0.2$ significantly outperform any
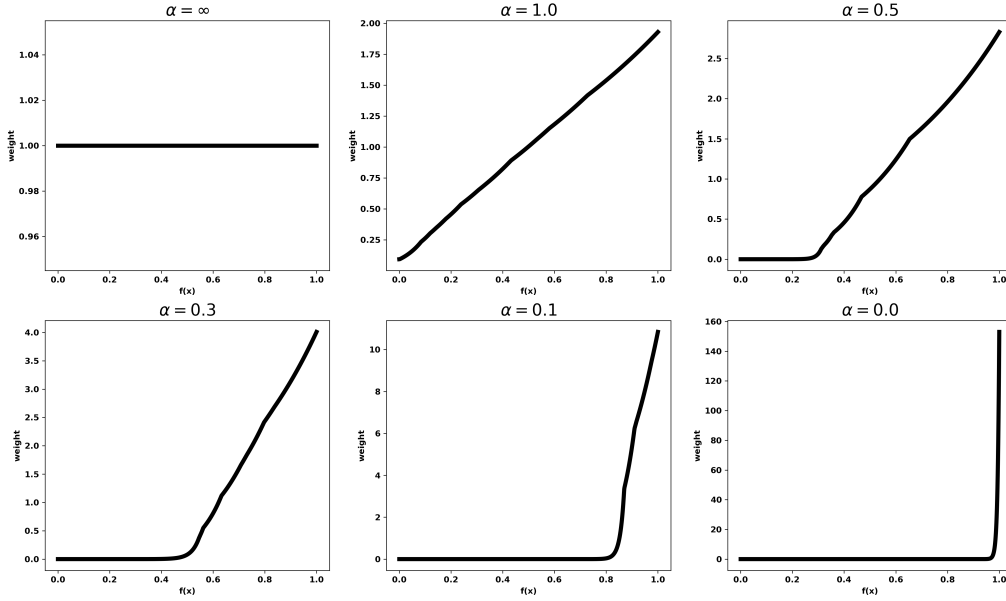
17

Figure 4: Learned Weight function w_{ϕ^*} for different choices of the hyper-parameter α.
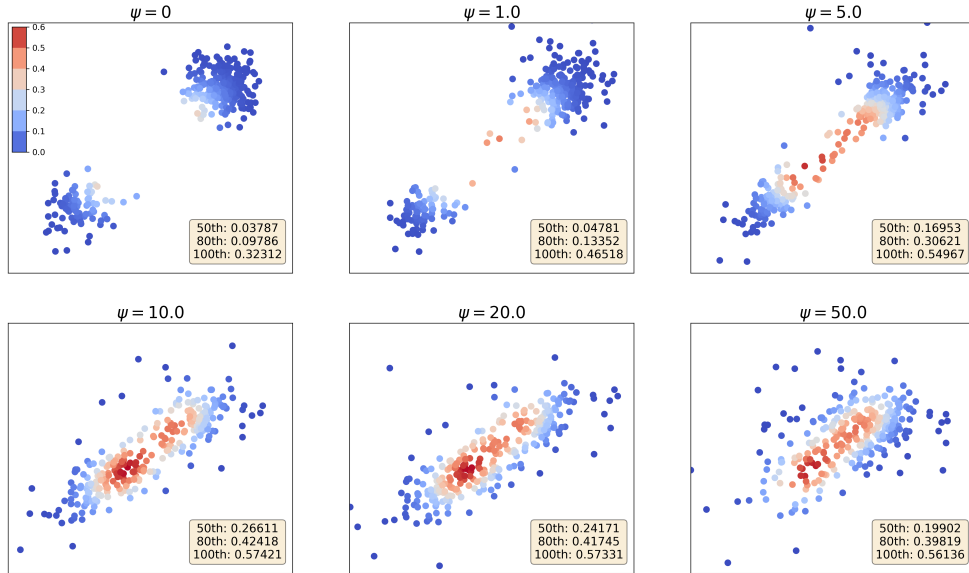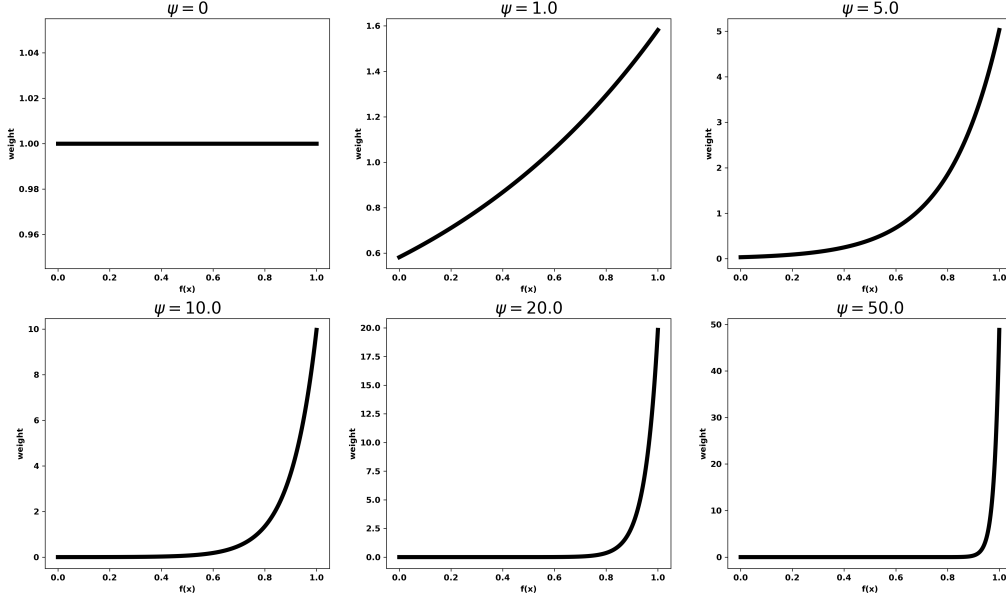

Figure 5: Optimized samples with predefined weight function for different choices of the hyper-parameter ψ.

Figure 6: Predefined weight function w_ψ for different choices of the hyper-parameter ψ.

predefined weight function considered here in terms of the average improvement. The learned weight functions w_{ϕ^*} that correspond to $\alpha = 0.2$ for each of the benchmark datasets are reported in Figure 7. Clearly, for the benchmark datasets, the learned weight functions are substantially different from the exponential predefined weight functions. To unleash the true power of the generative approach to offline optimization, it is thus critical to make the weight function trainable as well.

Table 3: Mean and standard deviation of the best generated samples for different choices of the hyper-parameter α (trainable weight function) and ψ (predefined weight function)

|	Supercond.	TFBind8	AntMorph.	GFP	UTR	Fluores.
$\mathcal{D}_{\text{best}}$	74	0.439	165.326	3.525	7.061	0.900
$\alpha = 0.15$	75.251±11.186	0.939±0.053	397.651±25.994	3.739±0.001	8.328±0.088	1.312±0.092
$\alpha = 0.2$	92.570±9.398	0.953±0.038	437.968±22.148	3.739±0.001	8.380±0.128	1.263±0.097
$\alpha = 0.25$	98.848±8.940	0.929±0.032	415.724±31.983	3.738±0.001	8.390±0.111	1.335±0.065
$\alpha = 0.3$	90.352±5.624	0.922±0.056	402.858±61.336	3.739±0.001	8.369±0.127	1.323±0.103
$\psi = 0.5$	84.509±4.248	0.930±0.051	362.179±51.306	3.739±0.001	7.969±0.162	1.465±0.069
$\psi = 1.0$	91.931±6.699	0.893±0.031	325.462±73.974	3.739±0.001	8.030±0.114	1.443±0.137
$\psi = 5.0$	91.206±7.440	0.887±0.064	381.254±48.028	3.739±0.001	8.390±0.179	1.436±0.068
$\psi = 20.0$	75.589±7.690	0.942±0.045	392.044±63.137	3.739±0.001	8.336±0.078	1.325±0.078

Table 4: Average improvement for different choices of the hyper-parameter α (trainable weight function) and ψ (predefined weight function)

(a) Trainable weight function

α	Ave. Improvement
0.15	0.543
0.2	0.620
0.25	0.616
0.3	0.579

(b) Predefined weight function

ψ	Ave. Improvement
0.5	0.545
1.0	0.508
5.0	0.567
20.0	0.542
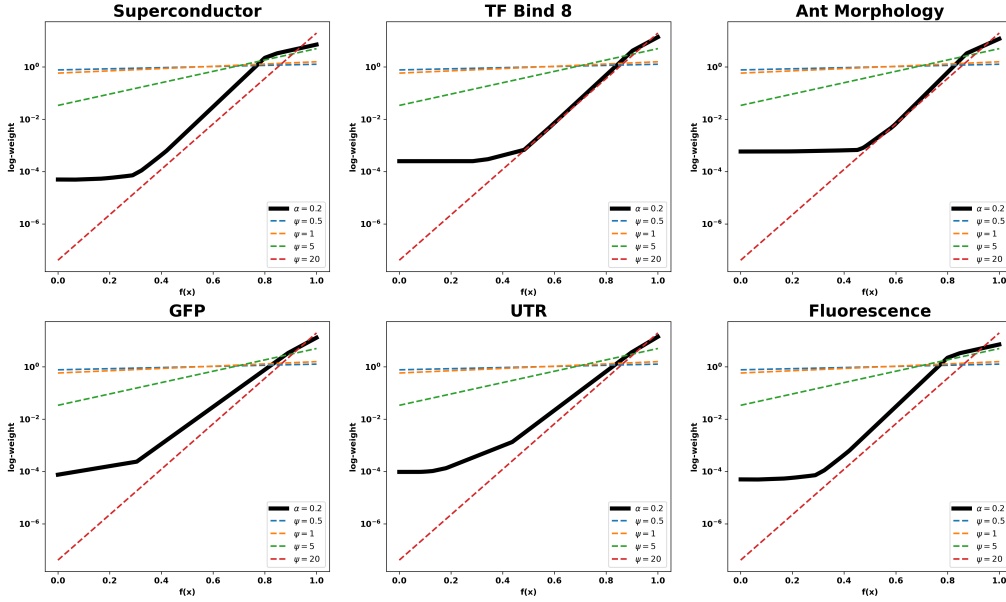
Figure 7: Learned weight functions w_{ϕ^*} for the benchmark datasets.