Explicit Gradient Learning for Black-Box Optimization

Elad Sarafian* [1] **Mor Sinay*** [1] **Yoram Louzoun** [1] **Noa Agmon** [1] **Sarit Kraus** [1]

Abstract

Black-Box Optimization (BBO) methods can find optimal policies for systems that interact with complex environments with no analytical representation. As such, they are of interest in many Artificial Intelligence (AI) domains. Yet classical BBO methods fall short in high-dimensional non-convex problems. They are thus often overlooked in real-world AI tasks. Here we present a BBO method, termed Explicit Gradient Learning (EGL), that is designed to optimize high-dimensional ill-behaved functions. We derive EGL by finding weak spots in methods that fit the objective function with a parametric Neural Network (NN) model and obtain the gradient signal by calculating the parametric gradient. Instead of fitting the function, EGL trains a NN to estimate the objective gradient directly. We prove the convergence of EGL to a stationary point and its robustness in the optimization of integrable functions. We evaluate EGL and achieve state-of-the-art results in two challenging problems: (1) the COCO test suite against an assortment of standard BBO methods; and (2) in a high-dimensional non-convex image generation task.

1. Introduction

Optimization problems are prevalent in many artificial intelligence applications, from search-and-rescue optimal deployment (Zhen et al., 2014) to triage policy in emergency rooms (Rosemarin et al., 2019) to hyperparameter tuning in machine learning (Bardenet et al., 2013). In these tasks, the objective is to find a policy that minimizes a cost or maximizes a reward. Evaluating the cost of a single policy is a complicated and often costly process that usually has no analytical representation, e.g., due to interaction with real-world physics or numerical simulation. *Black-Box Optimization* (BBO) algorithms (Audet & Hare, 2017; Golovin et al., 2017) are designed to solve such problems, when the analytical formulation is missing, by repeatedly querying the Black-Box function and searching for an optimal solution while minimizing the number of queries (budget).

Related Work BBO problems have been studied in multiple fields with diverse approaches. Many works investigated *derivative–free* methods (Rios & Sahinidis, 2013), from the classic Nelder–Mead algorithm (Nelder & Mead, 1965) and Powell's method (Powell, 1964) to more recent evolutionary algorithms such as CMA-ES (Hansen, 2006). Another line of research is *derivative-based* algorithms, which first approximate the gradient and then apply line-search methods such as the Conjugate Gradient (CG) Method (Shewchuk et al., 1994) and Quasi-Newton Methods, e.g. BFGS (Nocedal & Wright, 2006). Other model-based methods such as SLSQP (Bonnans et al., 2006) and COBYLA (Powell, 2007) iteratively solve quadratic or linear approximations of the objective function. Some variants apply *trust-region* methods and iteratively find an optimum within a trusted subset of the domain (Conn et al., 2009; Chen et al., 2018). Another line of research is more focused on stochastic discrete problems, e.g. Bayesian methods (Snoek et al., 2015), and multi-armed bandit problems (Flaxman et al., 2004).

More recent works have studied the applications of NNs in BBO algorithms. (Mania et al., 2018; Vemula et al., 2019) showed that Markov Decision Processes could be solved by applying random search optimization over the weights of a parameterized policy. (Sener & Koltun, 2020) suggested learning an intermediate low dimensional manifold with a NN to reduce the complexity of the random search. (Maheswaranathan et al., 2018) suggested learning the objective and using the parametric gradient with respect to the inputs (Lillicrap et al., 2015) to guide a random search. (Saremi, 2019) studied NN architectures for learning gradients.

Our contribution In this paper, we suggest a new derivative-based algorithm, *Explicit Gradient Learning* (EGL), that learns a surrogate function for the gradient by averaging the numerical directional derivatives over small volumes bounded by ε radius. We control the accuracy of our model by controlling the ε radius parameter. We then use trust-regions and dynamic scaling of the objective function

*Equal contribution. [1]Department of Computer Science, Bar-Ilan University, Israel. Correspondence to: Elad Sarafian, Mor Sinay <elad.sarafian@gmail.com, mor.sinay@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

to fine-tune the convergence process to the optimal solution. This results in a theoretically guaranteed convergence of EGL to a stationary point. We compared the performance of EGL to eight different BBO algorithms in rigorous simulations in the COCO test suite (Hansen et al., 2019) and show that EGL outperforms all others in terms of final accuracy. EGL was further evaluated in a high-dimensional non-convex domain, involving searching in the latent space of a Generative Adversarial Network (GAN) (Pan et al., 2019) to generate an image with specific characteristics. EGL again outperformed existing algorithms in terms of accuracy and appearance, where other BBO methods failed to converge to a solution in a reasonable amount of time.

The paper is organized as follows: BBO background and motivation for the EGL algorithm are presented in Sections 2 and 3. The detailed description of the EGL algorithm and its theoretical analysis are shown in Section 4. The empirical evaluation of EGL's performance and comparison to state-of-the-art BBO algorithms are described in Section 5, and we conclude in Section 6. The proofs for all of the theoretical statements are found in the Appendix.

2. Background

A function $f : \Omega \to \mathbb{R}$, $\Omega \subseteq \mathbb{R}^n$ is considered to be a Black-Box if one can evaluate $y = f(x)$ at $x \in \Omega$, but has no prior knowledge of its analytical form. A BBO algorithm seeks to find $x^* = \arg\min_{x\in\Omega} f(x)$, typically with as few evaluations as possible (Audet & Hare, 2017). Since, in general, it is not possible to converge to the optimal value with a finite number of evaluations, we define a budget C and seek to find as good a solution as possible $x^\star$ with less than C evaluations (Hansen et al., 2010).

Many BBO methods operate with a two-phase iterative algorithm: (1) search or collect data with some heuristic; and (2) update a model to obtain a new candidate solution and improve the heuristic. Traditionally, BBO methods are divided into derivative-free and derivative-based methods. The former group relies on statistical models (Balandat et al., 2019), physical models (Van Laarhoven & Aarts, 1987), or Evolutionary Strategies (Back, 1996) to define the search pattern and are not restricted to continuous domains, so they can also be applied to discrete variables. Our algorithm, EGL, falls under the latter category, which relies on a gradient estimation to determine the search direction (Bertsekas & Scientific, 2015). Formally, derivative-based methods are restricted to differentiable functions, but here we show that EGL can be applied successfully whenever the objective function is merely locally integrable.

Recently, with the Reinforcement Learning renaissance (Silver et al., 2017; Schulman et al., 2015), new algorithms have been suggested for the problem of continuous action control (e.g., robotics control). Many of these can be viewed in the context of BBO as derivative-based methods applied with NN parametric models. One of the most prominent algorithms is DDPG (Lillicrap et al., 2015). DDPG iteratively collects data with some (often naive) exploration strategy and then fits a local NN parametric model f_θ around a candidate solution x_k. To update the candidate, it approximates the gradient ∇f at x_k with the parametric gradient ∇f_θ and then applies a gradient descent step (Ruder, 2016).[1] Algorithm 1 outlines the DDPG steps in the BBO formulation. We denote it as Indirect Gradient Learning (IGL) since it does not directly learn the gradient ∇f. In the next section, we will develop arguments as to why one should learn the gradient explicitly instead of using the parametric gradient.

Algorithm 1 Indirect Gradient Learning

Input: x_0, α, C
$k = 0$
while *budget* $C > 0$ **do**
  Build Local Model:
    Collect data $\mathcal{D}_k = \{(x_k + \varepsilon n_i, y_i)\}_{i=1}^m,\ n_i \sim \mathcal{N}(0, \mathbf{I})$
    Fit a model f_{θ_k} with
    $\theta_k = \arg\min_\theta \sum_{i=1}^m |f_\theta(x_k + \varepsilon n_i) - y_i|^2$
  Gradient Descent:
    $x_{k+1} \leftarrow x_k - \alpha \nabla f_{\theta_k}(x_k)$
    $k \leftarrow k + 1$
return x_k

3. Motivation

In Algorithm 1 only the gradient information $\nabla f(x_k)$ is required to update the next x_k candidate. However, the gradient function is never learned directly, and it is only inferred from the parametric model without any clear guarantee of its veracity. Hence we seek a method that learns the gradient function ∇f explicitly. Clearly, directly learning the gradient is infeasible since the Black-Box only outputs the $f(x)$ values. Instead, our approach would be to learn a surrogate function for ∇f, termed the *mean-gradient*, by sampling pairs of observations $\{(x_i, y_i), (x_j, y_j)\}_{i,j}$ and averaging the numerical directional derivatives over small volumes. This section formally defines the mean-gradient and then formulates two arguments that motivate its use, instead of ∇f_θ. We then illustrate these arguments using 1D and 2D examples taken from the COCO test suite.

3.1. The Mean-Gradient

For any differentiable function f with a continuous gradient, the first order Taylor expression is

$$f(x + \tau) = f(x) + \nabla f(x) \cdot \tau + O(\|\tau\|^2). \quad (1)$$

[1] f_θ is differentiable. Thus, it can be differentiated with respect to its input x, as done in adversarial training (Yuan et al., 2019).

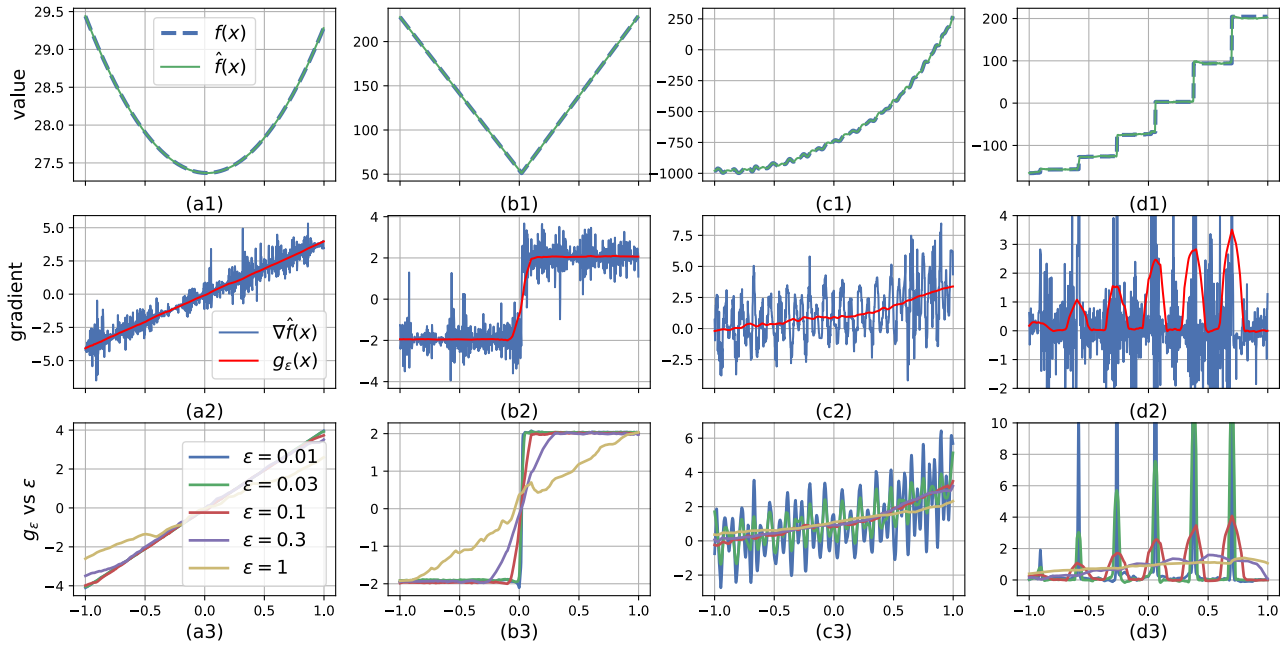

Figure 1. Comparing indirect gradient learning and explicit gradient learning for 4 typical functions: (a) parabolic; (b) piece-wise linear; (c) multiple local minima; (d) step function.

Thus, locally around x, the directional derivative satisfies $\nabla f(x) \cdot \tau \approx f(x+\tau) - f(x)$. We define the mean-gradient as the function that minimizes the Mean-Square-Error (MSE) of such approximations in a vicinity of x.

Definition 1. *The mean-gradient at x with $\varepsilon > 0$ averaging radius is*

$$g_\varepsilon(x) = \arg\min_{g\in\mathbb{R}^n} \int\limits_{V_\varepsilon(x)} |g\cdot\tau - f(x+\tau) + f(x)|^2 d\tau \quad (2)$$

where $V_\varepsilon(x) \subset \mathbb{R}^n$ is a convex subset s.t. $\|x' - x\| \le \varepsilon$ for all $x' \in V_\varepsilon(x)$ and the integral domain is over τ s.t. $x+\tau \in V_\varepsilon(x)$.

Proposition 1 (controllable accuracy)**.** *For any differentiable function f with a continuous gradient, there is $\kappa_g > 0$, so that for any $\varepsilon > 0$ the mean-gradient satisfies $\|g_\varepsilon(x) - \nabla f(x)\| \le \kappa_g \varepsilon$ for all $x \in \Omega$.*

In other words, the mean-gradient has a controllable accuracy parameter ε s.t. reducing ε improves the gradient approximation. As explained in Sec. 4.2, this property is crucial in obtaining the convergence of EGL to stationary points. Unlike the mean-gradient, the parametric gradient has no such parameter and the gradient accuracy is not directly controlled. Even for zero MSE error s.t. $f_\theta(x_i) \equiv y_i$ for all of the samples in the replay buffer (Mnih et al., 2015), there is no guarantee of the parametric gradient accuracy. On the contrary, overfitting the objective may severely hurt the gradients approximation.

Moreover, the parametric gradient can be discontinuous even when the parametric model has a Lipschitz continuous (Hansen & Jaumard, 1995) gradient. For example, a commonly used NN with ReLU activation is only piece-wise differentiable. This leads to very erratic gradients even for smooth objective functions. If the objective function is not smooth (e.g., for noise-like functions or in singular points), the gradient noise is exacerbated. On the other hand, the next proposition suggests that, due to the integral over $V_\varepsilon(x)$ that smooths the gradient, the mean gradient is smooth whenever the objective function is continuous.

Proposition 2 (continuity)**.** *If f is continuous in V and $V_\varepsilon(x) \subset V$ then g_ε is a continuous function at x.*

The mean-gradient is not necessarily continuous in discontinuity points of f. One can obtain an even smoother surrogate for ∇f by slightly modifying the mean-gradient definition

$$g^p_\varepsilon(x) = \arg\min_{g\in\mathbb{R}^n} \iint\limits_{V_\varepsilon(x)B_p(x)} |g\cdot(\tau - s) - f(\tau) + f(s)|^2 ds d\tau,$$

where the integral domains are $s \in B_p(x)$ and $\tau \in V_\varepsilon(x)$. Here, $B_p(x) \subset V_\varepsilon(x)$ is an n-ball perturbation set with $p < \varepsilon$ radius that dithers the reference point (which is fixed at x in Definition 1). We term this modified version the *perturbed mean-gradient*. Remarkably, while g^p_ε is still a controllably accurate model for Lipschitz continuous gradients, as the integrand is x independent, g^p_ε is continuous

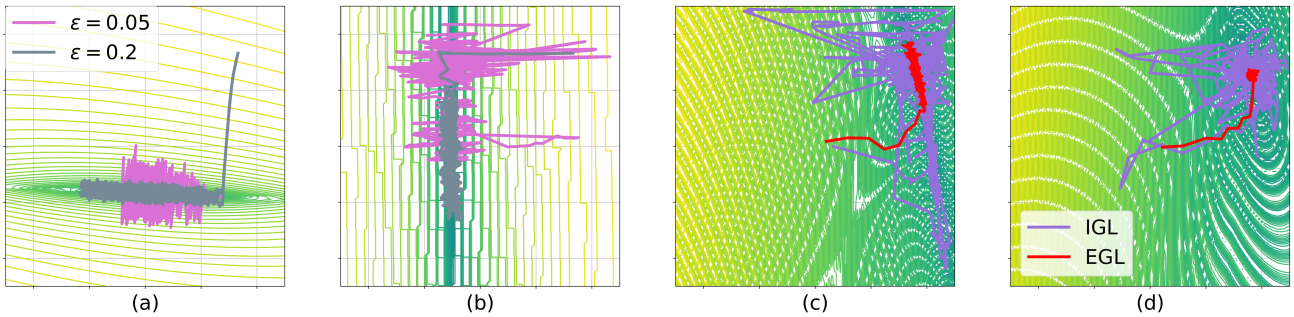

Figure 2. Visualizing explicit gradient learning with different ε for various 2D problems from COCO test suite: (a) sharp-ridge problem 194; (b) step-ellipsoid problem 97. Comparing EGL and IGL: (c) Schaffer F7 C1000 problem 255; (d) Schaffer F7 C10 problem 240.

whenever f is merely integrable. In practice, as g_ε is learned with a Lipschitz continuous model, we find that both forms are continuous for integrable functions. Nevertheless, Sec. 5 shows that g^p_ε adds a small gain to the EGL performance. Next, we demonstrate how the EGL smoothness (as opposed to ∇f_θ) leads to more stable and efficient trajectories in both continuous and discontinuous objective functions.

3.2. Illustrative Examples

To demonstrate these properties of the mean-gradient, we consider 1D [2] and 2D problems from the COCO test suite. We start by examining 4 typical 1D functions: (a) parabolic; (b) piece-wise linear; (c) multiple local minima; and (d) step function. We fit f with a NN and compare its parametric gradient to the mean-gradient, learned with another NN. The NN model is identical for both functions and is based on our spline embedding architecture (see Sec. 4.4). The results are presented in Fig. 1. The 1st row shows that the fit $\hat{f}$ is very strong s.t. the error is almost indistinguishable to the naked eye. Nevertheless, the calculated parametric gradient (2nd row) is noisy, even for smooth functions, as the NN architecture is piece-wise linear. Occasionally, there are even spikes that change the gradient sign. Traversing the surface curves with such a function is very unstable and inefficient. On the other hand, due to the smoothing parameter $\varepsilon = 0.1$ and since the NN is Lipschitz continuous, the mean-gradient is always smooth, even for singularity points and discontinuous gradients.

In the 3rd row we evaluate g_ε for different size ε parameters. We see that by setting ε sufficiently high, the gradient becomes smooth enough, so there is no problem descending over steps and multiple local minima functions. In practice, we may use this property and start the descent trajectory with a high ε. This way, the optimization process does not commit too early to a local minimum and searches for regions with lower valleys. After refining ε the process will settle into a local minimum, which in practice would be much lower than minima found around the initial point.

The next experiment (Fig. 2) is executed on 2D problems from the COCO test suite. In Fig. 2(a-b) we present the gradient-descent steps with the mean-gradient and a constant ε. In 2(a) the objective has a singular minimum, similar to the $|x|$ function's minimum (Fig.1(a)). As ε gets smaller, the gradients near the minimum get larger and there is a need to reduce the learning-rate (α) in order to converge. 2(b) presents a step function. Again, we observe that for a smaller ε the gradient has high spikes in the discontinuity points, leading to a noisier trajectory. For that purpose, the EGL algorithm decays both α and ε during the learning process (see Sec. 4 for details). This lends much smoother trajectories, as can be seen in Fig. 2(c-d). Here we executed both EGL and IGL with the same α, ε decay pattern. We observe that the IGL trajectories are noisy and inefficient since the parametric gradient error is not bounded. On the other hand, EGL smoothly travels through a ravine (Fig. 2(c)) and converges to a global minimum (Fig. 2(d)).

4. Design & Analysis

In this section, we lay out the practical EGL algorithm and analyze its asymptotic properties.

4.1. Monte-Carlo Approximation

To learn the mean-gradient, one may evaluate the integral in Eq. (2) with Monte-Carlo samples and learn a model that minimizes this term. Formally, for a model $g_\theta : \Omega \to \mathbb{R}^n$ and a dataset $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^m$, define the loss function

$$\mathcal{L}_{k,\varepsilon}(\theta) = \sum_{i=1}^{m} \sum_{x_j \in V_\varepsilon(x_i)} |(x_j - x_i) \cdot g_\theta(x_i) - y_j + y_i|^2 \quad (3)$$

and learn $\theta^*_k = \arg\min_\theta \mathcal{L}_{k,\varepsilon}(\theta)$, e.g. with gradient descent. This formulation can be used to estimate the mean-gradient for any x. Yet, practically, in each optimization iteration, we only care about estimating it in close proximity to the

[2] Since COCO does not have built-in 1D problems; we generated 1D problems based on 2D problems with $f_{1D}(x) : f_{2D}(x, x)$

current candidate solution x_k. Therefore, we assume that the dataset $\mathcal{D}_k$ holds samples only from $V_\varepsilon(x_k)$.

The accuracy of the learned model $g_{\theta_k^*}$ heavily depends on the number and locations of the evaluation points and the specific parameterization for g_θ. Still, for f with a Lipschitz continuous gradient, i.e. $f \in \mathcal{C}^{+1}$, we can set bounds for the model accuracy in $V_\varepsilon(x_k)$ with respect to ε. For that purpose, we require a set of at least $m \geq n+1$ evaluation points in $\mathcal{D}_k$ which satisfy the following poised set definition.

Definition 2 (poised set for regression)**.** *Let $\mathcal{D}_k = \{(x_i, y_i)\}_1^m$, $m \geq n+1$ s.t. $x_i \in V_\varepsilon(x_k)$ for all i. Define the matrix $\tilde{X}_i \in \mathbb{M}^{m \times n}$ s.t. the j-th row is $x_i - x_j$. Now define $\tilde{X} = \left(\tilde{X}_1^T \cdots \tilde{X}_m^T \right)^T$. The set $\mathcal{D}_k$ is a poised set for regression in x_k if the matrix $\tilde{X}$ has rank n.*

Intuitively, a set is poised if its difference vectors $x_i - x_j$ span $\mathbb{R}^n$. For the poised set, and a constant parameterization, the solution of Eq. (3) is unique and it is equal to the Least-Squares (LS) minimizer. If f has a Lipschitz continuous gradient, then, with an admissible parametric model, the error between $g_\theta(x)$ and $\nabla f(x)$ can be proportional to ε. We formalize this in the following theorem and corollary.

Theorem 1. *Let $\mathcal{D}_k$ be a poised set in $V_\varepsilon(x_k)$. The regression problem*

$$g^{MSE} = \arg\min_g \sum_{i,j \in \mathcal{D}_k} |(x_j - x_i) \cdot g - y_j + y_i|^2 \quad (4)$$

has the unique solution $g^{MSE} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T \delta$, where $\delta \in \mathbb{R}^{m^2}$ s.t. $\delta_{i \cdot (m-1)+j} = y_j - y_i$. Further, if $f \in \mathcal{C}^{1+}$ and $g_\theta \in \mathcal{C}^0$ is a parameterization with (equal or) lower regression loss than g^{MSE}, the following holds for all $x \in V_\varepsilon(x_k)$:

$$\|\nabla f(x) - g_\theta(x)\| \leq \kappa_g \varepsilon \quad (5)$$

Corollary 1. *For the $\mathcal{D}_k$ poised set, any Lipschitz continuous parameterization of the form $g_\theta(x) = F(Wx) + b$ is a controllably accurate model (Audet & Hare, 2017) in $V_\varepsilon(x_k)$ for the optimal set of parameters θ_k^*.*

This is obvious as we can simply set $W = 0$ and $b = g^{MSE}$. In this work, we are interested in using NNs which are much stronger parameterizations than constant models. If the NN satisfies the Lipschitz continuity property (e.g., with spectral normalization) and has at least a biased output layer, then its optimal set of parameters θ_k^* has lower regression loss than g^{MSE} and it is therefore a controllably accurate model.[3]

Finally, to incorporate learning of the perturbed mean gradient, we slightly modify Eq. (3). Note that we cannot directly dither the reference point x_i as we would need to collect more samples. Instead, we may dither the g_θ argument by evaluating it in $\bar{x}_{ir} = x_i + n_r$ where n_r is uniformly sampled in an n-ball with radius p. Thus, the perturbed loss function for small p s.t. $p \ll \varepsilon$ is

$$\mathcal{L}_{k,\varepsilon,p}(\theta) = \sum_{i,j \in \mathcal{D}_k, n_r} |(x_j - x_i) \cdot g_\theta(\bar{x}_{ir}) - y_j + y_i|^2 \quad (6)$$

4.2. Asymptotic Analysis

In this part, we assume that the function f has a Lipschitz continuous gradient s.t. $\|\nabla f(x) - \nabla f(x')\| \leq \kappa_f \|x - x'\|$. In this case, the classical gradient descent theorem states that the update $x_{k+1} = x_k - \alpha \nabla f(x_k)$ with learning parameter $\alpha \leq \frac{1}{\kappa_f}$ converges to a stationary point x^* s.t. $\|f(x_k)\| \to 0$ for $k \to \infty$ (Nesterov, 2013; Lee et al., 2016).

EGL descends over a surrogate function of the gradient that contains some amount of error. Far from x^*, where $\|\nabla f\|$ is large, the error in g_ε is small enough s.t. every new candidate improves the solution, i.e. $f(x_{k+1}) \leq f(x_k)$. If the stationary point x^* is locally convex, then, as x_k gets closer to x^*, the gradient $\|\nabla f\|$ decreases, and eventually the error, i.e. $g_\varepsilon(x_k) - \nabla f(x_k)$, becomes so significant that improvement is no longer guaranteed. Nevertheless, our analysis shows that a proper choice of ε_k yields monotonic decreasing steps that converge to a stationary point.

Theorem 2. *Let $f : \Omega \to \mathbb{R}$ have a Lipschitz continuous gradient and a Lipschitz constant κ_f. Suppose a controllable mean-gradient model g_ε with error constant κ_g, the gradient descent iteration $x_{k+1} = x_k - \alpha g_\varepsilon(x_k)$ with α s.t. $\frac{5\varepsilon}{\|\nabla f(x_k)\|} \leq \alpha \leq \min(\frac{1}{\kappa_g}, \frac{1}{\kappa_f})$ guarantees a monotonically decreasing step s.t. $f(x_{k+1}) \leq f(x_k) - 2.25\frac{\varepsilon^2}{\alpha}$.*

Corollary 2. *If Theorem 2 is satisfied for all k, the gradient descent iteration converges to a stationary point x^* s.t. $\frac{1}{K}\sum_{k=1}^K \|f(x_k)\|^2 \leq \frac{12|f(x_0) - f(x^*)|}{\alpha_K K}$.*

Utilizing Theorem 2 we can design an algorithm that converges to a stationary point. For that purpose we must make sure that the learning rate abides the requirement $\alpha \leq \min(\frac{1}{\kappa_g}, \frac{1}{\kappa_f})$. Since this factor cannot be easily estimated, a practical solution is to decay α during the optimization process. However, to converge, the ratio $\frac{\varepsilon}{\alpha}$ must also decay to zero. This means that ε must decay to zero faster than α. Algorithm 2 provides both of these conditions, so it is guaranteed to converge to a local minimum for $\bar{\varepsilon} \to 0$.

This analysis assumes a Lipschitz continuous gradient, but since EGL is also designed for non-smooth functions, our practical algorithm alleviates the requirement for strictly monotonically decreasing steps. Instead, it calculates a running mean over the last candidates to determine when to decrease α and ε. The complete practical EGL algorithm is found in the Appendix Sec. E. It also includes input and output mapping and scaling, as described in the next section.

[3] Provided that the optimization process recovers θ_k^*, which is not necessarily true in practice.

Algorithm 2 Convergent EGL

Input: $x_0, \alpha, \varepsilon, \gamma_\alpha < 1, \gamma_\varepsilon < 1, \bar{\varepsilon}$
$k = 0$
while $\varepsilon \leq \bar{\varepsilon}$ **do**
  Build Model:
    Collect data $\{(x_i, y_i)\}_1^m, x_i \in V_\varepsilon(x_k)$
    Learn a local model $g_\varepsilon(x_k)$
  Gradient Descent:
    $x_{k+1} \leftarrow x_k - \alpha g_\varepsilon(x_k)$
    if $f(x_{k+1}) > f(x_k) - 2.25\frac{\varepsilon^2}{\alpha}$ **then**
      $\alpha \leftarrow \gamma_\alpha \alpha$
      $\varepsilon \leftarrow \gamma_\alpha \gamma_\varepsilon \varepsilon$
  $k \leftarrow k + 1$
return x_k

4.3. Dynamic Mappings

Unlike supervised learning with a constant dataset, BBO is a dynamic problem. The input and output statistics change over time as the optimization progresses. There are several sources for this drift. First, traversing via g_ε changes the input's first moment by updating the center of the samples x_i and the output's first moment by collecting smaller costs y_i. At the same time, squeezing ε and α over time decreases the second moment statistics by reducing the variety in $\{(x_i, y_i)\}$. NNs are sensitive to such distribution changes (Brownlee, 2018) and require tweaking hyperparameters such as learning rate and initial weight distribution to maintain high performance. Default numbers (e.g. learning rate of 10^{-3}) usually work best when the input data is normalized. To regulate the statistics over the entire optimization process, we apply a method of double dynamic mappings for both input and output values.

From the input perspective, our dynamic mapping resembles *Trust-Region* methods (Conn et al., 2000; Nocedal & Wright, 2006). Instead of searching for x^* in the entire Ω domain by reducing ε and α over time, we fix ε and α and search for $x_j^\star$ in a sub-region Ω_j. After finding the best candidate solution in this sub-region we shrink Ω_j by a factor of $\gamma_\alpha > 0$ and ε by a factor of $\gamma_\varepsilon > 0$. To keep the input statistics regulated, we maintain a bijective mapping $h_j : \Omega_j \to \mathbb{R}^n$: that scales the x values to an unconstrained domain $\tilde{x}$ with approximately constant first and second moments.

In this work, Ω is assumed to be a rectangular box that denotes the upper and lower bounds for each entry in x. Thus we consider element-wise bijective mappings of the form $h_j(x) = (h_j^1(x^1), ..., h_j^n(x^n))$ and each new sub-region, $\Omega_{j+1} \subset \Omega_j$, is a smaller rectangular box centered at $x_j^\star$. In the unconstrained domain $\tilde{x}$, we can use the initial learning rate, yet, due to the squeezing factor, the effective learning rate is decayed by the γ_α factor. Equivalently, the effective accuracy parameter ε is reduced by a factor of $\gamma_\alpha \times \gamma_\varepsilon$. In other words, we use the same NN parameters to learn a zoomed-in problem of the original objective function.

In order to regulate the output statistics, we define a scalar, monotonically increasing, invertible mapping $r_k(y)$ which maps the y_i samples in the dataset $\mathcal{D}_k$ to approximately constant statistics. Since traversing with g_ε can significantly change the statistics even in the same sub-region, r_k must be dynamic and cannot be held fixed for the entire j sub-problem. Combining both input and output mappings, we obtain a modified set of samples $\{(\tilde{x}_i, \tilde{y}_i)\}_i = \{(h_j(x_i), r_k(y_i))\}_i$, so effectively we learn a modified mean-gradient, denoted as $\tilde{g}_{\varepsilon_{jk}}$, of a compressed and scaled function $\tilde{y} = \tilde{f}_{jk}(\tilde{x}) = r_k \circ f \circ h_j^{-1}(\tilde{x})$.

If both input and output mappings are linear, then the true mean-gradient is proportional to the modified mean-gradient

$$g_\varepsilon(x) = \left(\frac{\partial r_k}{\partial y}\right)^{-1} \nabla h_j(x) \odot \tilde{g}_{\varepsilon_{jk}}(\tilde{x}) \tag{7}$$

Even when the mappings are approximately linear inside $V_{\varepsilon_{jk}}(\tilde{x}_k)$, $g_\varepsilon(x_k)$ can be estimated according to Eq. (7). Hence, to maintain a controllably accurate model, we seek mappings that preserve the linearity as much as possible. On the other hand, strictly linear mappings may be insufficient since they are susceptible to outliers. After experimenting with several functions, we found a sweet spot: a composition of a linear mapping followed by a squash function that compresses only the outliers (see details in Appendix Sec. D). With such mappings, we make sure both that the model is controllably accurate and that the learning hyperparameters are adequate for the entire optimization process.

4.4. Spline Embedding

Many black-box applications, including the experiments in Sec. 5, have no clear inductive bias which can be exploited, as e.g., CNNs are suitable for natural images (Cohen & Shashua, 2016). In such cases, it is common to use feed-forward NNs. However, we found out that the quality of fitting objectives and gradients with such NN is unsatisfactory, especially for low dimensional problems. One method to augment the input layer is through learnable embeddings (Zhang et al., 2016). However, usually, embeddings are applied for categorical input and they do not preserve order. Since our input domain is continuous, we wish to design Lipschitz continuous learnable embeddings $s_\theta(x)$ s.t. for two inputs x_1 and x_2 the calculated embedding satisfies $\|s(x_1) - s(x_2)\| \leq \kappa_s \|x_1 - x_2\|$ for some $\kappa_s > 0$.

We chose to do so by learning a set of one-dimensional splines (Reinsch, 1967). A spline is a piece-wise polynomial defined on a set of disjoint intervals with smoothness conditions in the intersection points, denoted as knots. Learnable splines were also suggested in (Fey et al., 2018) for the problem of learning over irregular grids. We used piece-wise

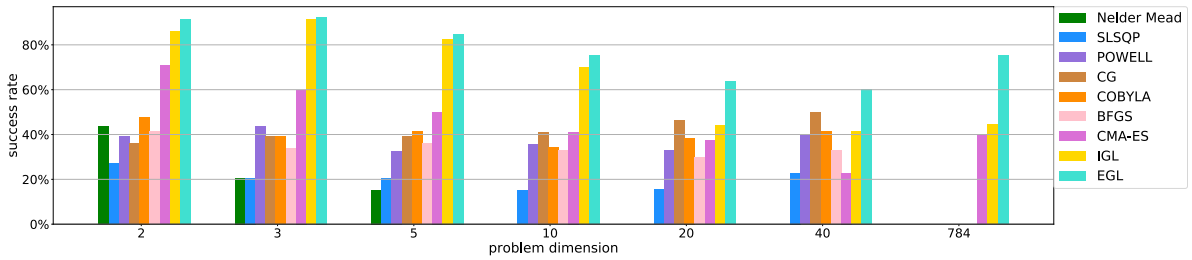

Figure 3. Comparing the success rate of EGL and other BBO algorithms for a budget $C = 150 \cdot 10^3$.

linear splines of the form

$$s_\theta(x) = \frac{\theta_i}{h_i}(t_{i+1} - x) + \frac{\theta_{i+1}}{h_i}(x - t_i) \tag{8}$$

where θ has $k+1$ elements (k is the number of knots), t_i is the position of the i-th knot and $h_i = t_{i+1} - t_i$. Instead of fitting θ to the data as usually done in classical spline applications, we learned these parameters as part of the optimization process, similar to categorical embeddings. Each entry in the input vector was expanded by a paramterization of several one-dimensional splines. We found this architecture particularly suitable for modeling complex functions defined over unstructured input domain. Please refer to the Appendix Sec. C for a full description and an empirical evaluation of fitting objective functions with spline-net.

5. Empirical Evaluation

In this section we describe the rigorous empirical analysis of EGL against various BBO methods in the COCO test suite and in a search task over the latent space of a GAN model. The code is available at `http://github.com/MorSinay/BBO`.

5.1. The COCO test suite

We tested EGL on the COCO test suite, a platform for systematic comparison of real-parameter global optimizers. COCO provides Black-Box functions in several dimensions (2,3,5,10,20,40), where each dimension comprises 360 distinct problems. To test higher dimensions, we created an extra set of 784D problems by composing a pre-trained encoder (Kingma & Welling, 2013) of FashionMnist images, each with 784 pixels (Xiao et al., 2017), with a 10D COCO problem as follows: $f_{784D}(x) : f_{10D}(Encoder(x))$.

We compared EGL with seven baselines, implemented on Scipy and Cma Python packages: Nedler Mead, SLSQP, POWELL, CG, COBYLA, BFGS, CMA-ES and with our IGL implementation based on the DDPG approach. For the 784D problems we evaluated only CG, CMA-ES and IGL which yielded results in a reasonable amount of time. We examined two network models. To compare against other baselines (Figues 3, 4 and 5(a)), we used our spline embedding model (details in Appendix Sec. C). Since spline-net is more time consuming, for the ablation tests in Fig. 5(b-d), we used a lighter Fully Connected (FC) net. For additional information, including a hyperparameters list, refer to Appendix Sec. F.

Fig. 3 presents the success rate of each algorithm with respect to the dimension number and for a budget of $C = 150 \cdot 10^3$. A single test-run is considered successful if: (1) $y_{best} - y^* \leq 1$; and (2) $\frac{y_{best} - y^*}{y_0 - y^*} \leq 10^{-2}$. Here, y_0 is the initial score $f(x_0)$, $y_{best} = \min y_k$ is the best-observed value for that run, and y^* is the minimal value obtained from all the baselines' test-runs. This definition guarantees that a run is successful only if its best-observed value is near y^*, both in terms of absolute distance and in terms of relative improvement with respect to the initial score. The results show that EGL outperforms all other baselines for any dimension. Most importantly, as the dimension number increases, the performance gap between EGL and all other baselines grows larger.

In Fig. 4 we present two performance profiles (Dolan & Moré, 2002) for the 40D and 10D problem sets: (1) Time-To-Solution (TTS); and (2) Best Observed (BO). TTS measures the percentage of solved problems with respect to the time step. We find that both EGL and IGL have a cold start behavior. This is a result of our design choice to start the training with a warm-up phase where we evaluate 384 points around x_0 and train the networks before executing the first gradient descent step. However, as the training process progresses, EGL outperforms all other methods and peaks at $C = 150 \cdot 10^3$ with the success rate of Fig. 3. The BO profile measures the percentage of problems whose best-observed value after $150 \cdot 10^3$ time steps is better than $\Delta y \in [y_0, y^*]$. We find that for all of the possible thresholds Δy, EGL solves more problems than any other method.

Fig. 5 visualizes the learning process of each method. To do that, we first calculate a scaled distance between the best value at time t and the optimal value $\Delta y^t_{best} = \frac{\min_{k \leq t} y_k - y^*}{y_0 - y^*}$. We then average this number for each t, over all runs in the same dimension problem set. This distance

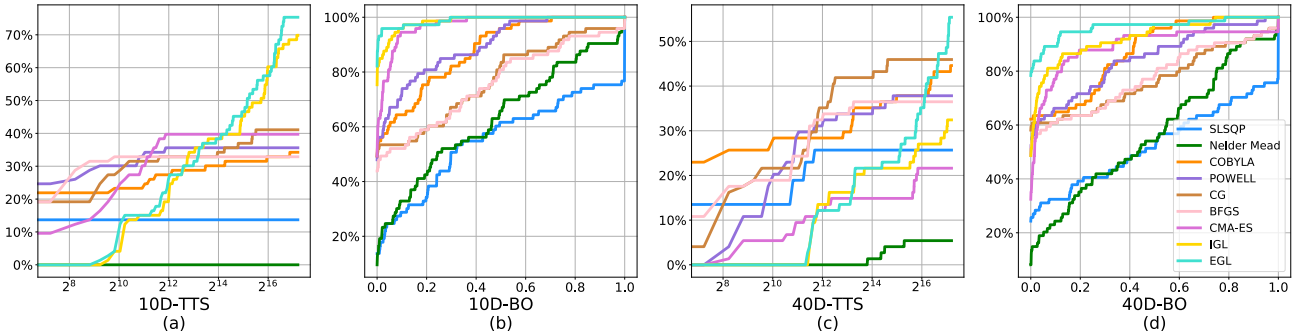

Figure 4. Performance Profile for 10D and 40D: Time-To-Solution (TTS) and Best-Observed (BO) after $150 \cdot 10^3$ steps. In BO the x-axis linearly maps $[y_0, y^*]$ to $[0, 1]$.

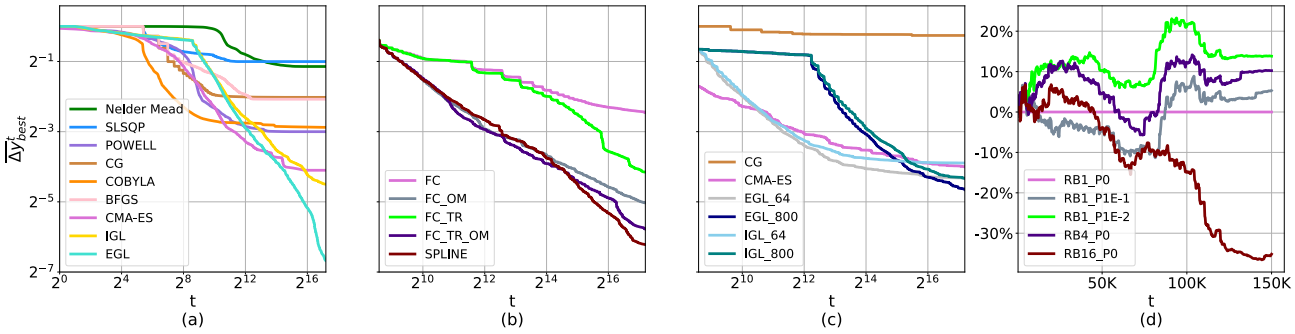

Figure 5. The scaled distance Δy^t_{best} as a function of $t \in [1, .., C]$ for: (a) EGL and baselines on 40D, (b) trust-region and output mapping ablation test, (c) EGL with different m samples and baselines on 784D, (d) the perturbed mean-gradient and long replay buffer on 40D.

is scaled to $[0, 1]$ and the results are presented on a log-log scale. Fig. 5(a) presents $\overline{\Delta y}^t_{best}$ on the 40D set. The elbow pattern in step 384 marks the switch from x_0 to x_1. Unlike the baselines that settle on a local minimum, EGL monotonically decreases during the entire optimization process. It overtakes the best baseline (CMA-ES) after $\sim 10^4$ steps.

Fig. 5(b) demonstrates the advantage of output-mapping (OM) and trust-region (TR). Here, we compared an FC net, trained with OM and TR (`FC_TR_OM`) against the variations `FC`, `FC_TR` and `FC_OM`. The results show a clear advantage to using both OM and TR, yet, while OM is crucial for the entire optimization process, the TR advantage materializes only near the minimal value. In addition, Fig. 5(b) manifests the gain of the spline-net (`SPLINE`) on top of `FC_TR_OM`.

The next experiment is executed on the high dimensional 784D problem set. In Fig. 5(c) we compare the performance of different numbers of exploration points $m = \{64, 800\}$ against CMA-ES, CG and the IGL baselines. While Theorem 1 guarantees a controllably accurate model when sampling $m \geq n+1$ points, remarkably, EGL generalized to an outstanding performance and outperformed all other baselines even for $m = 64 \ll n$. Nevertheless, as expected, more exploration points converged to a better final value. In practice, the choice of m should correspond to the allocated budget size. More exploration around each candidate comes with the cost of fewer gradient descent steps; thus, for low budgets, one should typically choose a small m.

Next, we tested the gain of two possible modifications to EGL: (1) perturbations with g^p_ε; and (2) training g_θ with a larger replay buffer (RB) with exploration points from the last L candidates. These tests were all executed with the same random seed. Fig. 5(d) presents the performance gain $(1 - \overline{\Delta y}^t_{best}/\overline{\Delta y}^t_{RB1\_P0})$ as a function of t for several different runs. Small perturbations $p = 0.01\varepsilon$ improved the performance (14%), possibly since they also regulate the NN training, yet, too large perturbations of $p = 0.1\varepsilon$ yielded inconsistent results. We also observed that a too long RB of $L = 16$ largely hurt the performance, probably since it adds high values to the RB which leads to a more compressed output mapping. However, moderate RB of $L = 4$ had a positive impact (10%), probably since more exploration points near the current candidate reduce the controllable accuracy factor κ_g and thus the accuracy of g_ε improves.

5.2. Searching the latent space of generative models

To examine EGL in a high-dimensional, complex, non-convex and noisy domain, we experimented with the task of searching the latent space of an image generative model

Figure 6. Searching latent space of generative models with EGL and IGL. Note that *the target image is not revealed to the optimizer*, only the face attributes and landmark points. The left-hand number is the average minimal value for each algorithm over 64 different problems.

(Volz et al., 2018). Generative models learn to map between a latent predefined distribution z to a complex real-world distribution x (e.g. images or audio). Given a trained Black-Box generator, while it is easy to sample from the distribution of x by sampling from z, it is not straightforward to generate an image with some desired characteristics. For that purpose, one may apply a BBO to search the latent space for a hidden representation z^* that generates an image with the desired traits x^*. Here, we used a face generative model and optimized z^* to generate an image with a required set of face attributes, landmark points and quality.

We trained a generator & discriminator for the CelebA dataset (Liu et al., 2015) based on the BigGAN (Brock et al., 2018) architecture and a classifier for the CelebA attributes. For the face landmark points, we used a pre-trained model (Kazemi & Sullivan, 2014). The BBO was trained to minimize the following objective:

$$f_{al}(z) = \lambda_a \mathcal{L}_a(G(z)) + \lambda_l \mathcal{L}_l(G(z)) + \lambda_g \tanh(D(G(z)))$$

Where: (1) $\mathcal{L}_a$ is the Cross-Entropy loss between the generated face attributes as measured by the classifier and the desired set of attributes a; (2) $\mathcal{L}_l$ is the MSE between the generated landmark points and the desired set of landmarks l ; and (3) $D(G(z))$ is the discriminator output, positive for low-quality images and negative for high-equality images.

Since each evaluation of z is costly, we limited the budget C to only 10^4 evaluations and the number of exploration points in each step was only $m = 32 \ll 512$. Such m violates the requirement in Theorem 1. However, we found that *in practice*, for finite budget and high dimensions, it is better to take more gradient steps with a less accurate gradient. To increase the sample efficiency we found that instead of sampling points in an n-ball around the candidate x_k, it is better to sample points in a cone with an apex at x_k and a vector equal to the mean-gradient estimation at x_k, i.e. $g(x_k)$. We term this exploration strategy as gradient-guided-exploration. For further details, please refer to the Appendix, Sec. H.

Due to the high-dimensional problem, classic methods such as CG and even CMA-ES fail to generate satisfying faces (see Appendix Sec. G for sample images). In Fig. 6 we compare the images generated by EGL and IGL. Generally, the quality of the results depends on the image target style. Some face traits which are more frequent in the CelebA dataset lead to a better face quality. Some faces, specifically with attributes such as a beard or a hat, are much harder to find, probably due to the suboptimality of the generator, which usually reduces the variety of images found in the dataset (Bau et al., 2019). Nevertheless, the results show that EGL produces better images, both visually and according to the final cost value. We observed that for hard targets, IGL frequently fails to find any plausible solutions while EGL yields non-perfect yet much more satisfactory candidates.

6. Conclusions

We presented EGL, a derivative-based BBO algorithm that achieves state-of-the-art results on a wide range of optimization problems. The essence of its success is a learnable function that estimates the mean gradient with a controllable smoothness factor. Starting with a high smoothness factor, let EGL find global areas in the function with low valleys. Gradually decreasing it lets EGL converge to a local minimum. The concept of EGL can be generalized to other related fields, such as sequential decision-making problems (i.e. Reinforcement Learning), by directly learning the gradient of the Q-function. We also demonstrated the use of EGL in an applicative high-dimensional Black-Box problem, searching the latent space of generative models.

Acknowledgements

This work was supported in part by the Ministry of Science & Technology, Israel.

References

Audet, C. and Hare, W. *Derivative-free and blackbox optimization*. Springer, 2017.

Back, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: Programmable Bayesian Optimization in PyTorch. *arxiv e-prints*, 2019. URL http://arxiv.org/abs/1910.06403.

Bardenet, R., Brendel, M., Kégl, B., and Sebag, M. Collaborative hyperparameter tuning. In *International conference on machine learning*, pp. 199–207, 2013.

Bau, D., Zhu, J.-Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., and Torralba, A. Seeing what a gan cannot generate. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4502–4511, 2019.

Bertsekas, D. P. and Scientific, A. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

Bonnans, J.-F., Gilbert, J. C., Lemaréchal, C., and Sagastizábal, C. A. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Brownlee, J. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.

Chen, R., Menickelly, M., and Scheinberg, K. Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169(2):447–487, 2018.

Cohen, N. and Shashua, A. Inductive bias of deep convolutional networks through pooling geometry. *arXiv preprint arXiv:1605.06743*, 2016.

Conn, A. R., Gould, N. I., and Toint, P. L. *Trust region methods*. SIAM, 2000.

Conn, A. R., Scheinberg, K., and Vicente, L. N. *Introduction to derivative-free optimization*, volume 8. Siam, 2009.

Dolan, E. D. and Moré, J. J. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

Fey, M., Eric Lenssen, J., Weichert, F., and Müller, H. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 869–877, 2018.

Flaxman, A. D., Kalai, A. T., and McMahan, H. B. Online convex optimization in the bandit setting: gradient descent without a gradient. *arXiv preprint cs/0408007*, 2004.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1487–1495, 2017.

Hansen, N. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pp. 75–102. Springer, 2006.

Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pp. 1689–1696, 2010.

Hansen, N., Brockhoff, D., Mersmann, O., Tusar, T., Tusar, D., ElHara, O. A., Sampaio, P. R., Atamna, A., Varelas, K., Batu, U., Nguyen, D. M., Matzner, F., and Auger, A. COmparing Continuous Optimizers: numbbo/COCO on Github, March 2019. URL https://doi.org/10.5281/zenodo.2594848.

Hansen, P. and Jaumard, B. Lipschitz optimization. In *Handbook of global optimization*, pp. 407–493. Springer, 1995.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Howard, J. and Gugger, S. fastai: A layered api for deep learning. *arXiv preprint arXiv:2002.04688*, 2020.

Kazemi, V. and Sullivan, J. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1867–1874, 2014.

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

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

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Loomis, L. H. and Sternberg, S. *Advanced calculus*. World Scientific, 1968.

Maheswaranathan, N., Metz, L., Tucker, G., Choi, D., and Sohl-Dickstein, J. Guided evolutionary strategies: Augmenting random search with surrogate gradients. *arXiv preprint arXiv:1806.10230*, 2018.

Mania, H., Guy, A., and Recht, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Nelder, J. A. and Mead, R. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Nocedal, J. and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., and Zheng, Y. Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 7:36322–36333, 2019.

Powell, M. J. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

Powell, M. J. A view of algorithms for optimization without derivatives. *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications*, 43(5):170–174, 2007.

Reinsch, C. H. Smoothing by spline functions. *Numerische mathematik*, 10(3):177–183, 1967.

Rios, L. M. and Sahinidis, N. V. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3): 1247–1293, 2013.

Rosemarin, H., Rosenfeld, A., and Kraus, S. Emergency department online patient-caregiver scheduling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 695–701, 2019.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Saremi, S. On approximating ∇f with neural networks. *arXiv preprint arXiv:1910.12744*, 2019.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Sener, O. and Koltun, V. Learning to guide random search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1gHokBKwS.

Shewchuk, J. R. et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pp. 2171–2180, 2015.

Van Laarhoven, P. J. and Aarts, E. H. Simulated annealing. In *Simulated annealing: Theory and applications*, pp. 7–15. Springer, 1987.

Vemula, A., Sun, W., and Bagnell, J. A. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. *arXiv preprint arXiv:1901.11503*, 2019.

Volz, V., Schrum, J., Liu, J., Lucas, S. M., Smith, A., and Risi, S. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 221–228, 2018.

Wang, X., Girshick, R., Gupta, A., and He, K. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Yuan, X., He, P., Zhu, Q., and Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9): 2805–2824, 2019.

Zhang, W., Du, T., and Wang, J. Deep learning over multi-field categorical data. In *European conference on information retrieval*, pp. 45–57. Springer, 2016.

Zhen, L., Wang, K., Hu, H., and Chang, D. A simulation optimization framework for ambulance deployment and relocation problems. *Computers & Industrial Engineering*, 72:12–23, 2014.

A. Theoretical Analysis

A.1. The Mean-Gradient

Definition 1. *The mean-gradient in a region around x of radius $\varepsilon > 0$ is*

$$g_\varepsilon(x) = \arg\min_{g\in\mathbb{R}^n} \int\limits_{V_\varepsilon(x)} |g\cdot\tau - f(x+\tau) + f(x)|^2 d\tau \tag{9}$$

where $V_\varepsilon(x) \subset \mathbb{R}^n$ is a convex subset s.t. $\|x' - x\| \le \varepsilon$ for all $x' \in V_\varepsilon(x)$ and the integral domain is over τ s.t. $x+\tau \in V_\varepsilon(x)$.

Proposition 1 (controllable accuracy)**.** *For any twice differentiable function $f \in \mathcal{C}^1$, there is $\kappa_g(x) > 0$, so that for any $\varepsilon > 0$ the mean-gradient satisfies $\|g_\varepsilon(x) - \nabla f(x)\| \le \kappa_g(x)\varepsilon$ for all $x \in \Omega$.*

Proof. Recall the Taylor theorem for a twice differentiable function $f(x+\tau) = f(x) + \nabla f(x)\cdot\tau + R_x(\tau)$, where $R_x(\tau)$ is the remainder. Since the gradient is continuous,by the fundamental theorem for line integrals

$$f(x+\tau) = f(x) + \int_0^1 \nabla f(x+t\tau)\cdot\tau dt = f(x) + \nabla f(x)\cdot\tau + \int_0^1 (\nabla f(x+t\tau) - \nabla f(x))\cdot\tau dt \tag{10}$$

Since $f \in \mathcal{C}^{1+}$, we also have $|\nabla f(x) - \nabla f(x+\tau)| \le \kappa_f\|\tau\|$. We can use this property to bound the remainder in the Taylor expression.

$$\begin{aligned} R_x(\tau) &= \int_0^1 (\nabla f(x+t\tau) - \nabla f(x))\cdot\tau dt \\ &\le \kappa_f \int_0^1 \|x+t\tau - x\|\cdot\|\tau\| dt = \kappa_f\|\tau\|^2 \int_0^1 t dt = \frac{1}{2}\kappa_f\|\tau\|^2 \end{aligned} \tag{11}$$

Now, by the definition of g_ε, an upper bound for $\mathcal{L}(g_\varepsilon(x))$ is

$$\begin{aligned} \mathcal{L}(g_\varepsilon(x)) \le \mathcal{L}(\nabla f(x)) &= \int\limits_{V_\varepsilon(x)} |\nabla f(x)\cdot\tau - f(\tau) + f(s)|^2 d\tau = \int\limits_{V_\varepsilon(x)} |R_x(\tau)|^2 d\tau \\ &\le \frac{1}{4}\kappa_f^2 \int\limits_{V_\varepsilon(x)} |\|\tau\|^2|^2 d\tau \le \kappa_f\varepsilon^4 |V_\varepsilon(x)| = \frac{1}{4}\kappa_f^2\varepsilon^{n+4}|V_1(x)| \end{aligned}$$

To develop the lower bound we will assume that $\dim(\mathrm{span}(V_\varepsilon(x))) = n$ and we will use the following definition

$$M_\varepsilon(x) = \min_{\hat{\mathbf{n}}} \int_{V_\varepsilon(x)\setminus V_{\frac{\varepsilon}{2}}(x)} \left|\frac{\tau}{\|\tau\|}\cdot\hat{\mathbf{n}}\right|^2 d\tau \tag{12}$$

where $\hat{\mathbf{n}} \in \mathbb{R}^n$ s.t. $\|\hat{\mathbf{n}}\| = 1$ and we assumed that $V_{\frac{\varepsilon}{2}} \subset V_\varepsilon$. As the dimension of $V_\varepsilon(x)$ is n, it is obvious that $M_\varepsilon(x) > 0$.

The lower bound is

$$
\begin{aligned}
\mathcal{L}(g_\varepsilon(x)) &= \int_{V_\varepsilon(x)} |g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau + \nabla f(x)\cdot\tau - f(x+\tau) + f(x)|^2\, d\tau \\
&\geq \int_{V_\varepsilon(x)} \left(|g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau| - |\nabla f(x)\cdot\tau - f(x+\tau) + f(x)|\right)^2 d\tau \\
&= \int_{V_\varepsilon(x)} |g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau|^2\, d\tau + \int_{V_\varepsilon(x)} |\nabla f(x)\cdot\tau - f(x+\tau) + f(x)|^2\, d\tau \\
&- 2\int_{V_\varepsilon(x)} |g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau| \cdot |\nabla f(x)\cdot\tau - f(x+\tau) + f(x)|\,\tau \\
&\geq \int_{V_\varepsilon(x)} |g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau|^2\, d\tau - 2\int_{V_\varepsilon(x)} |g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau| \cdot |\nabla f(x)\cdot\tau - f(x+\tau) + f(x)|\,\tau \\
&\geq \int_{V_\varepsilon(x)} |g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau|^2\, d\tau - \kappa_f \int_{V_\varepsilon(x)} |g_\varepsilon(x)\cdot\tau - \nabla f(x)\cdot\tau| \cdot \|\tau\|^2\tau \\
&\geq \|g_\varepsilon(x) - \nabla f(x)\|^2 \int_{V_\varepsilon(x)} |\hat{\mathbf{n}}(x)\cdot\tau|^2\, d\tau - \kappa_f \|g_\varepsilon(x) - \nabla f(x)\| \int_{V_\varepsilon(x)} \cdot\|\tau\|^3\tau \\
&\geq \|g_\varepsilon(x) - \nabla f(x)\|^2 \int_{V_\varepsilon(x)\setminus V_{\frac{\varepsilon}{2}}(x)} |\hat{\mathbf{n}}(x)\cdot\tau|^2\, d\tau - \kappa_f \|g_\varepsilon(x) - \nabla f(x)\|\, \varepsilon^{n+3}|V_1(x)| \\
&\geq \|g_\varepsilon(x) - \nabla f(x)\|^2 \left(\frac{\varepsilon}{2}\right)^2 \int_{V_\varepsilon(x)\setminus V_{\frac{\varepsilon}{2}}(x)} \left|\hat{\mathbf{n}}(x)\cdot\frac{\tau}{\|\tau\|}\right|^2 d\tau - \kappa_f \|g_\varepsilon(x) - \nabla f(x)\|\, \varepsilon^{n+3}|V_1(x)| \\
&\geq \|g_\varepsilon(x) - \nabla f(x)\|^2 \left(\frac{\varepsilon}{2}\right)^2 \varepsilon^n \int_{V_1(x)\setminus V_{\frac{1}{2}}(x)} \left|\hat{\mathbf{n}}(x)\cdot\frac{\tau}{\|\tau\|}\right|^2 d\tau - \kappa_f \|g_\varepsilon(x) - \nabla f(x)\|\, \varepsilon^{n+3}|V_1(x)| \\
&\geq \frac{1}{4}\|g_\varepsilon(x) - \nabla f(x)\|^2 \varepsilon^{n+2} M_1(x) - \kappa_f \|g_\varepsilon(x) - \nabla f(x)\|\, \varepsilon^{n+3}|V_1(x)|
\end{aligned}
$$

Combining the upper and lower bound we obtain

$$
\begin{aligned}
& \frac{1}{4}\|g_\varepsilon(x) - \nabla f(x)\|^2 \varepsilon^{n+2} M_1(x) - \kappa_f \|g_\varepsilon(x) - \nabla f(x)\|\, \varepsilon^{n+3}|V_1(x)| \leq \frac{1}{4}\kappa_f^2 \varepsilon^{n+4}|V_1(x)| \\
\Rightarrow\quad & M_1(x)\|g_\varepsilon(x) - \nabla f(x)\|^2 - 4\kappa_f\varepsilon|V_1(x)|\,\|g_\varepsilon(x) - \nabla f(x)\| - \kappa_f^2\varepsilon^2|V_1(x)| \leq 0 \\
\Rightarrow\quad & \|g_\varepsilon(x) - \nabla f(x)\| \leq \varepsilon\kappa_f \frac{2|V_1(x)| + \sqrt{4|V_1(x)|^2 + |V_1(x)|M_1(x)}}{M_1(x)}
\end{aligned}
$$

□

Proposition 2 (continuity)**.** *If $f(x)$ is continuous in V s.t. $V_\varepsilon(x) \subset V$, then the mean-gradient is a continuous function at x.*

Proof. Let us define the two-variable function $\mathcal{L}(x, g)$:

$$
\mathcal{L}(x, g) = \int_{V_\varepsilon(x)} |g\cdot\tau - f(x+\tau) + f(x)|^2 d\tau \tag{13}
$$

The mean-gradient is the global minimum of this function for each x. Notice that $\mathcal{L}(x, g)$ is a polynomial function in g

since if f is an integrable function, then we can write

$$\begin{aligned}\mathcal{L}(x,g) &= \int_{V_\varepsilon(x)} |g\cdot\tau - f(x+\tau) + f(x)|^2 d\tau \\ &= \int_{V_\varepsilon(x)} |g\cdot\tau|^2 d\tau - 2\int_{V_\varepsilon(x)} g\cdot\tau(f(x+\tau)-f(x))d\tau + \int_{V_\varepsilon(x)} |f(x+\tau)-f(x)|^2 d\tau \\ &= g\cdot\mathbf{A}(x)g + g\cdot b(x) + c(x)\end{aligned} \tag{14}$$

where $\mathbf{A}(x) = \int_{V_\varepsilon(x)} \tau\tau^T d\tau$, $b(x) = -2\int_{V_\varepsilon(x)} \tau(f(x+\tau)-f(x))d\tau$ and $c(x) = \int_{V_\varepsilon(x)} |f(x+\tau)-f(x)|^2 d\tau$. Without loss of generality for this proof, we can ignore the constant $c(x)$ as it does not change the minimum point. In addition, note that $A(x)$ is constant for all x since the domain $V_\varepsilon(x)$ is invariant for x and the integrand does not depend on f.

Since $\mathcal{L}(x,g)$ is bounded from below, $\mathcal{L}(x,g) \geq 0$, it must have a minimum s.t. $\mathbf{A} \geq 0$. Assume that $\mathbf{A} > 0$ (e.g. when V_ε is an n-ball of radius ε), then for each x there is a unique minimum for $\mathcal{L}(x,g)$ and this minimum is the mean-gradient $g_\varepsilon(x)$.

To show the continuity of $g_\varepsilon(x)$, define $F(x,g) = \nabla_g \mathcal{L}(x,g)$, F maps $\mathbb{R}^{2n} \to \mathbb{R}^n$. Assume that for x_0, $g_\varepsilon(x_0)$ is the mean-gradient. Therefore, $F(x_0, g_\varepsilon(x_0)) = 0$. Since $A > 0$, this means that the derivative $\nabla_g F(x_0, g_\varepsilon(x_0))$ is invertible. We will apply a version of the implicit function theorem (Loomis & Sternberg, 1968) (Theorem 9.3 pp. 230-231) to show that there exists a unique and continuous mapping $h(x)$ s.t. $F(x, h(x)) = 0$.

To apply the implicit function theorem, we need to show that $F(x,g)$ is continuous and the derivative $\nabla_g F$ is continuous and invertible. The latter is obvious since $\nabla_g F = \mathbf{A} > 0$ is a constant positive definite matrix. $F(x,g)$ is also continuous with respect to g, therefore it is left to verify that $F(x,g)$ is continuous with respect to x.

Lemma 3. *If $f(x)$ is continuous in V s.t. $V_\varepsilon(x) \subset V$, then $\mathcal{L}(x,g)$ is continuous in x.*

Proof.

$$\begin{aligned}|\mathcal{L}(x,g) - \mathcal{L}(x',g)| &= |g\cdot\mathbf{A}(x)g + g\cdot b(x) - g\cdot\mathbf{A}(x')g - g\cdot b(x)| \\ &= |g\cdot b(x) - g\cdot b(x)| \leq \|g\| \left\| \int_{V_\varepsilon(x)} \tau(f(x+\tau)-f(x))d\tau - \int_{V_\varepsilon(x')} \tau(f(x'+\tau)-f(x'))d\tau \right\|\end{aligned} \tag{15}$$

To write both integrals with the same variable, we change variables to $\tau = \tilde{\tau} - x$ in the first integrand and $\tau = \tilde{\tau} - x'$ in the second integrand.

$$\begin{aligned}|\mathcal{L}(x,g) - \mathcal{L}(x',g)| &\leq \|g\| \left\| \int_{V_\varepsilon(x)} (\tilde{\tau}-x)(f(\tilde{\tau})-f(x))d\tilde{\tau} - \int_{V_\varepsilon(x')} (\tilde{\tau}-x')(f(\tilde{\tau})-f(x'))d\tilde{\tau} \right\| \\ &\leq \|g\|C_1 + \|g\|C_2 + \|g\|C_3 + \|g\|C_4 + \|g\|C_5\end{aligned} \tag{16}$$

Where

$$C_1 = |f(x') - f(x)| \int_{V_\varepsilon(x)\cap V_\varepsilon(x')} \|\tilde{\tau}\| d\tilde{\tau} \tag{17}$$

$$C_2 = \|x - x'\| \int_{V_\varepsilon(x)\cap V_\varepsilon(x')} |f(\tilde{\tau})| d\tilde{\tau} \tag{18}$$

$$C_3 = \|xf(x) - x'f(x')\| \int_{V_\varepsilon(x)\cap V_\varepsilon(x')} d\tilde{\tau} \tag{19}$$

$$C_4 = \int_{V_\varepsilon(x)\setminus V_\varepsilon(x')} \|\tilde{\tau} - x\| \cdot |f(\tilde{\tau}) - f(x)| d\tilde{\tau} \tag{20}$$

$$C_5 = \int_{V_\varepsilon(x')\setminus V_\varepsilon(x)} \|\tilde{\tau} - x'\| \cdot |f(\tilde{\tau}) - f(x')| d\tilde{\tau} \tag{21}$$

$$\tag{22}$$

Taking $x' \to x$, C_1, C_2, C_3 all go to zero as the integral is finite but $x' \to x$ and $f(x') \to f(x)$. For C_4 and C_5, note that the integrand is bounded but the domain size goes to zero as $x' \to x$. To see that we will show that $|V_\varepsilon(x) \setminus V_\varepsilon(x')| \leq |A_\varepsilon(x)| \cdot \|x - x'\|$, where $|A_\varepsilon(x)|$ is the surface area of V_ε.

Lemma 4. $|V_\varepsilon(x) \setminus V_\varepsilon(x')| \leq |A_\varepsilon(x)| \cdot \|x - x'\|$

Proof. First, note that if $u \in V_\varepsilon(x)$, then $u + x' - x \in V_\varepsilon(x')$. Take $P \subset V_\varepsilon$ s.t. $p \in P$ if and only if $\text{distance}(A_\varepsilon(x), p) \geq \|x - x'\|$ and $p \in V_\varepsilon(x)$. For any $p \in P$, $p - x + x' \in V_\varepsilon(x)$, thus, following our first argument $p \in V_\varepsilon(x')$.

We obtain that $P \cap V_\varepsilon(x) \setminus V_\varepsilon(x') = \Phi$, thus $|V_\varepsilon(x) \setminus V_\varepsilon(x')| \leq |V_\varepsilon(x) \setminus P|$. However, all points $q \in V_\varepsilon(x) \setminus P$ satisfy $\text{distance}(A_\varepsilon(x), q) \leq \|x - x'\|$, therefore $|V_\varepsilon(x) \setminus P| \leq |A_\varepsilon(x)| \cdot \|x - x'\|$. □

Following Lemma 4 we obtain that the integral in C_4 and C_5 goes to zero and therefore the distance $|\mathcal{L}(x, g) - \mathcal{L}(x', g)| \to 0$ as $x \to x'$. □

$\mathcal{L}(x, g)$ continuous in x and g with a continuous derivative in g implies that $\nabla_g \mathcal{L}(x, g)$ is continuous in x. We can now apply Theorem 9.3 pp. 230-231 in (Loomis & Sternberg, 1968) and conclude that there is a unique continuous mapping $h(x)$ s.t. $F(x, h(x)) = 0$. Since $A > 0$, this means that such a mapping defines a local minimum for $\mathcal{L}(x, g)$ in g. Further, since $\mathcal{L}(x, g)$ is a second degree polynomial in g, this is a unique global mapping. Therefore, it must be equal to $g_\varepsilon(x)$ and hence g_ε is continuous in x.

□

A.2. Parametric approximation of the mean-gradient

In this section we analyze the Monte-Carlo learning of the mean-gradient with a parametric model. Generally, we define a parametric model g_θ and learn θ^* by minimizing the term

$$\mathcal{L}(g_\theta, \varepsilon) = \sum_{i=1}^{N} \sum_{x_j \in V_\varepsilon(x_i)} |(x_j - x_i) \cdot g_\theta(x_i) - y_j + y_i|^2 \tag{23}$$

We start by analyzing constant parameterization of the mean-gradient around a candidate x_k. We consider two cases: (1) interpolation, where there are exactly $n + 1$ evaluation points; and (2) regression where there are $m > n + 1$ evaluation points. This line of arguments follows the same approach taken in (Audet & Hare, 2017), Chapter 9.

A.2.1. Constant parameterization with $n + 1$ interpolation points

Definition 3. *A set of $n + 1$ points $\{x_i\}_0^n$, s.t. every subset of n points spans $\mathbb{R}^n$, is a poised set for constant interpolation.*

Proposition 3. *For a constant paramterization $g(x) = g$, a poised set has a unique solution with zero regression error.*

$$\min_g \sum_{i,j \in \mathcal{D}} |(x_j - x_i) \cdot g - y_j + y_i|^2 = 0 \tag{24}$$

Proof. Define the matrix $\tilde{X}_i \in \mathbb{M}^{n \times n}$ s.t. the j-th row is $x_i - x_j$ and $\delta_i \in \mathbb{R}^n$ s.t. $\delta_{i,j} = y_j - y_i$. We may transform Eq. (24) into $n + 1$ sets of linear equations:

$$\forall\, i \;\; \tilde{X}_i g = \delta_i \tag{25}$$

While there are $n + 1$ different linear systems of equations, they all have the same solution $g_{\min}$. To see that, define the system of equation $\tilde{X}\tilde{g} = r$ where

$$\tilde{X} = \begin{pmatrix} x_0 & 1 \\ x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, \quad g = \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \\ s \end{pmatrix}, \quad r = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{26}$$

and s is an additional slack variable. For all i we can apply an elementary row operation of subtracting the i-th row s.t. the updated system is

$$\left(\begin{array}{cc|c} x_0 - x_i & 0 & y_0 - y_i \\ \vdots & \vdots & \vdots \\ x_{i-1} - x_i & 0 & y_{i-1} - y_i \\ 0 & 0 & 0 \\ x_{i+1} - x_i & 0 & y_{i+1} - y_i \\ \vdots & \vdots & \vdots \\ x_n - x_i & 0 & y_n - y_i \end{array}\right) \tag{27}$$

Reducing the zeroed i-th row we get the system of equation $\tilde{X}_i g = \delta_i$ which has a unique solution since the set $\{x_j\} \setminus x_i$ spans $\mathbb{R}^n$. □

Corollary 3. *For any parameterization of the form $g_\theta = f(Wx) + b$ and a poised set $\{x_j\}_0^n$ we have an optimal solution where $W^* = 0$ and $b^* = g_{\min}$. Specifically it also holds for a Neural Network with a biased output layer.*

Lemma 5. *For a poised set $\{x_j\}_0^n$ s.t. $\|x_i - x_j\| \leq \varepsilon$ and a mean-gradient estimator $g_\theta \in \mathcal{C}^0$ with zero interpolation error, the following holds*

$$\|\nabla f(x) - g_\theta(x)\| \leq \kappa_g \varepsilon \tag{28}$$

Proof. $f \in \mathcal{C}^{1+}$, hence for any x_i in the poised set and x s.t. $\|x - x_i\| \leq \varepsilon$ we have

$$\begin{aligned} &\|\nabla f(x) - g_\theta(x)\| \leq \\ &\quad \|\nabla f(x) - \nabla f(x_i)\| + \|\nabla f(x_i) - g_\theta(x_i)\| + \|g_\theta(x_i) - g_\theta(x)\| \leq (\kappa_f + \kappa_{g_\theta})\varepsilon + \|\nabla f(x_i) - g_\theta(x_i)\| \end{aligned} \tag{29}$$

Where κ_{g_θ} is the Lipschitz constant of g_θ it is left to bound the last term. First, note that for all x_j in the poised set we have that

$$(x_j - x_i) \cdot g_\theta(x_i) = f(x_j) - f(x_i) \leq (x_j - x_i) \cdot \nabla f(x_i) + \frac{1}{2}\kappa_f \varepsilon^2 \tag{30}$$

where the last equation comes from the second error term in the Taylor series expansion in x_i (see Proposition A.1). Returning to our definition of $\tilde{X}_i$ (see proposition 3) we can write

$$\|\tilde{X}_i(\nabla f(x_i) - g_\theta(x_i))\| = \sqrt{\sum_i |(x_j - x_i) \cdot (g_\theta(x_i) - \nabla f(x_i))|^2} \leq \frac{1}{2}\sqrt{n}\kappa_f \varepsilon^2 \tag{31}$$

Using that property we have

$$\|\nabla f(x_i) - g_\theta(x_i)\| = \|\tilde{X}_i^{-1}\tilde{X}_i(\nabla f(x_i) - g_\theta(x_i))\| \leq \|\tilde{X}_i^{-1}\|\|\tilde{X}_i(\nabla f(x_i) - g_\theta(x_i))\| \leq \frac{1}{2}\sqrt{n}\kappa_f \|\tilde{X}_i^{-1}\|\varepsilon^2. \tag{32}$$

$\|\tilde{X}_i^{-1}\| = \frac{1}{\min \sigma(\tilde{X}_i)}$, where σ is the singular values. Notice that the rows of $\tilde{X}_i$ are $x_j - x_i \propto \varepsilon$, thus we can scale them by ε. In this case, since the poised set spans $\mathbb{R}^n$, the minimal singular value of $\frac{1}{\varepsilon}\tilde{X}_i$ is finite and does not depend on ε. Therefore, we obtain

$$\|\nabla f(x_i) - g_\theta(x_i)\| \leq \frac{1}{2}\sqrt{n}\|(\frac{1}{\varepsilon}\tilde{X}_i)^{-1}\|\kappa_f \varepsilon = O(n\varepsilon) \tag{33}$$

Therefore,

$$\|\nabla f(x) - g_\theta(x)\| \leq \left(\kappa_f + \frac{1}{2}\sqrt{n}\kappa_{g_\theta} + \|(\frac{1}{\varepsilon}\tilde{X}_i)^{-1}\|\kappa_f\right)\varepsilon \tag{34}$$

□

Notice that we only required g_θ to be a zero-order Lipschitz continuous and we do not set any restrictions on its gradient. For Neural Networks, having the $\mathcal{C}^0$ property is relatively easy, e.g. with spectral normalization (Miyato et al., 2018). However, many NNs are not $\mathcal{C}^1$, e.g. NN with ReLU activations.

A.2.2. CONSTANT PARAMETERIZATION WITH $m > n+1$ REGRESSION POINTS

We can extend the results of Sec. A.2.1 to the regression problem where we have access to $m > n+1$ points $\{x_i\}_0^{m-1}$. We wish to show that the bounds for a constant mean-gradient solution for the regression problem in Eq. (23) are also controllably accurate, i.e. $\|\nabla f(x) - g\| \leq \kappa_g \varepsilon$. As in the interpolation case, we start with the definition of the poised set for regression.

Definition 2 (poised set for regression)**.** *Let $\mathcal{D}_k = \{(x_i, y_i)\}_1^m$, $m \geq n+1$ s.t. $x_i \in V_\varepsilon(x_k)$ for all i. Define the matrix $\tilde{X}_i \in \mathbb{M}^{m\times n}$ s.t. the j-th row is $x_i - x_j$. Now define $\tilde{X} = \begin{pmatrix} \tilde{X}_1^T & \cdots & \tilde{X}_m^T \end{pmatrix}^T$. The set $\mathcal{D}_k$ is a poised set for regression in x_k if the matrix $\tilde{X}$ has rank n.*

Intuitively, a set is poised if its difference vectors $x_i - x_j$ span $\mathbb{R}^n$. For the poised set, and a constant parameterization, the solution of Eq. (35) is unique and it equals to the Least- Squares (LS) minimizer. If f has a Lipschitz continuous gradient, then the error between $g_\varepsilon(x)$ and $\nabla f(x)$ is proportional to ε. We formalize this argument in the next proposition.

Theorem 1. *Let $\mathcal{D}_k$ be a poised set in $V_\varepsilon(x_k)$. The regression problem*

$$g^{MSE} = \arg\min_g \sum_{i,j\in\mathcal{D}_k} |(x_j - x_i)\cdot g - y_j + y_i|^2 \tag{35}$$

has the unique solution $g^{MSE} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\delta$, where $\delta \in \mathbb{R}^{m^2}$ s.t. $\delta_{i\cdot(m-1)+j} = y_j - y_i$. Further, if $f \in \mathcal{C}^{1+}$ and $g_\theta \in \mathcal{C}^0$ is a parameterization with lower regression loss g^{MSE}, the following holds

$$\|\nabla f(x) - g_\theta(x)\| \leq \kappa_g \varepsilon \tag{36}$$

Proof. The regression problem can be written as

$$g = \arg\min_g \|\tilde{X}g - \delta\|^2 \tag{37}$$

This is the formulation for the mean-square error problem with matrix $\tilde{X}$ and target δ. The minimizer of this function is the standard mean-square error minimizer which is unique as $\tilde{X}$ has rank n.

$$g = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\delta \tag{38}$$

Lemma 6. *Let $f \in \mathcal{C}^{1+}$ on $B_\varepsilon(x_j)$ with a Lipschitz constant κ_f. For any triplet x_i, x_j, x_k s.t. $\|x_i - x_j\| \leq \varepsilon$ and $\|x_k - x_j\| \leq \varepsilon$*

$$|f(x_k) - f(x_j) - (x_k - x_j)\cdot\nabla f(x_i)| \leq \frac{3}{2}\kappa_f\varepsilon^2 \tag{39}$$

Proof.

$$\begin{aligned}
|f(x_k) - f(x_j) - (x_k - x_j)\cdot\nabla f(x_i)| &= \left|\int_0^1 (x_k - x_j)\cdot\nabla f(x_j + \tau(x_k - x_j))d\tau - (x_k - x_j)\cdot\nabla f(x_i)\right| \\
&= \left|\int_0^1 (x_k - x_j)\cdot(\nabla f(x_j + \tau(x_k - x_j)) - \nabla f(x_i))\, d\tau\right| \\
&\leq \left|\int_0^1 \|x_k - x_j\|\cdot\|\nabla f(x_j + \tau(x_k - x_j)) - \nabla f(x_i)\| d\tau\right| \\
&\leq \kappa_f\varepsilon\left|\int_0^1 \|x_j + \tau(x_k - x_j) - x_i\| d\tau\right| \\
&\leq \kappa_f\varepsilon\left|\int_0^1 \|x_j - x_i\| + \|\tau(x_k - x_j)\| d\tau\right| \\
&\leq \kappa_f\varepsilon\left|\varepsilon + \varepsilon\int_0^1 \tau d\tau\right| \\
&= \frac{3}{2}\kappa_f\varepsilon^2
\end{aligned} \tag{40}$$

□

Applying the previous Lemma, for all $x \in V_\varepsilon(x_k)$

$$\|\tilde{X}\nabla f(x) - \delta\|^2 = \sum_{k=0}^{m-1}\sum_{j=0}^{m-1} |(x_k - x_j)^T\nabla f(x) - (f(x_k) - f(x_j))|^2 \leq m^2\left(\frac{3}{2}\kappa_f\varepsilon^2\right)^2 \tag{41}$$

hence $\|\tilde{X}\nabla f(x) - \delta\| \leq \frac{3m}{2}\kappa_f\varepsilon^2$.

Notice also that if $\mathcal{D}_k$ is a poised set s.t. the matrix $\tilde{X}$ has rank n s.t. $\tilde{X}^\dagger = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T$ exists and we have that

$$\|\nabla f(x) - \tilde{X}^\dagger\delta\| = \left\|\tilde{X}^\dagger\left(\tilde{X}\nabla f(x) - \delta\right)\right\| \leq \|\tilde{X}^\dagger\| \cdot \|\tilde{X}\nabla f(x_i) - \delta\| \leq \|\tilde{X}^\dagger\|\frac{3m}{2}\kappa_f\varepsilon^2 \tag{42}$$

As in the interpolation case, we can multiply $\tilde{X}^\dagger$ by ε to obtain a matrix which is invariant to the size of ε. Denote the scaled pseudo-inverse as $\tilde{X}^\ddagger = \varepsilon(\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T$. Therefore,

$$\|\nabla f(x) - g^{MSE}\| \leq \|\tilde{X}^\ddagger\|\frac{3m}{2}\kappa_f\varepsilon \tag{43}$$

If g_θ has a lower regression error than g^{MSE}, then there exists at least one point x_i s.t. $x_i \in \mathcal{D}_k$ and $\|\nabla f(x_i) - g_\theta(x_i)\| \leq \|\nabla f(x_i) - g^{MSE}\|$. In this case we have

$$\begin{aligned}
\|\nabla f(x) - g_\theta(x)\| &\leq \|\nabla f(x) - \nabla f(x_i)\| + \|\nabla f(x_i) - g_\theta(x_i)\| + \|g_\theta(x_i) - g_\theta(x)\| \\
&\leq (\kappa_f + \kappa_{g_\theta})\varepsilon + \|\nabla f(x_i) - g_\theta(x_i)\| \\
&\leq (\kappa_f + \kappa_{g_\theta})\varepsilon + \|\nabla f(x_i) - g^{MSE}\| \\
&\leq (\kappa_f + \kappa_{g_\theta})\varepsilon + \|\tilde{X}^\ddagger\|\frac{3m}{2}\kappa_f\varepsilon \\
&= \left(\kappa_f + \kappa_{g_\theta} + \|\tilde{X}^\ddagger\|\frac{3m}{2}\kappa_f\right)\varepsilon
\end{aligned}$$

□

Corollary 1. *For the $\mathcal{D}_k$ poised set, any Lipschitz continuous parameterization of the form $g_\theta(x) = F(Wx)+b$, specifically NNs, is a controllably accurate model in $V_\varepsilon(x_k)$ for the optimal set of parameters θ^*.*

A.3. Convergence Analysis

For clarity, we replace the subscript θ in g_θ and write g_ε to emphasize that our model for the mean-gradient is controllably accurate.

Theorem 2. *Let $f : \Omega \to \mathbb{R}$ be a function with Lipschitz continuous gradient, i.e. $f \in \mathcal{C}^{+1}$ and a Lipschitz constant κ_f. Suppose a controllable mean-gradient model g_ε with error constant κ_g, the gradient descent iteration $x_{k+1} = x_k - \alpha g_\varepsilon(x_k)$ with α s.t. $\frac{5\varepsilon}{\|\nabla f(x_k)\|} \leq \alpha \leq \min(\frac{1}{\kappa_g}, \frac{1}{\kappa_f})$ guarantees a monotonically decreasing step s.t. $f(x_{k+1}) \leq f(x_k) - 2.25\frac{\varepsilon^2}{\alpha}$.*

Proof. For $f \in \mathcal{C}^{+1}$ the following inequality holds for all x_k (see proof in Proposition A.1)

$$f(x) \leq f(x_k) + (x - x_k)\cdot\nabla f(x_k) + \frac{1}{2}\kappa_f\|x - x_k\|^2 \tag{44}$$

Plugging in the iteration update $x_{k+1} = x_k - \alpha g_\varepsilon(x_k)$ we get

$$f(x_{k+1}) \leq f(x_k) - \alpha g_\varepsilon(x_k)\cdot\nabla f(x_k) + \alpha^2\frac{1}{2}\kappa_f\|g_\varepsilon(x_k)\|^2 \tag{45}$$

For a controllable mean-gradient we can write $\|g_\varepsilon(x) - \nabla f(x)\| \leq \varepsilon\kappa_g$, therefore we can write $g_\varepsilon(x) = \nabla f(x) + \varepsilon\kappa_g\xi(x)$ s.t. $\|\xi(x)\| \leq 1$ so the inequality is

$$f(x_{k+1}) \leq f(x_k) - \alpha\|\nabla f(x_k)\|^2 - \alpha\varepsilon\kappa_g\xi(x_k)\cdot\nabla f(x_k) + \alpha^2\frac{1}{2}\kappa_f\|\nabla f(x) + \varepsilon\kappa_g\xi(x)\|^2 \tag{46}$$

Using the equality $\|a+b\|^2 = \|a\|^2 + 2a\cdot b + \|b\|^2$ and the Cauchy-Schwartz inequality inequality $a\cdot b \leq \|a\|\|b\|$ we can write

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \alpha\|\nabla f(x_k)\|^2 + \alpha\varepsilon\kappa_g\|\xi(x_k)\|\cdot\|\nabla f(x_k)\| + \alpha^2\frac{1}{2}\kappa_f\left(\|\nabla f(x)\|^2 + 2\varepsilon\kappa_g\|\xi(x)\|\cdot\|\nabla f(x_k)\| + \varepsilon^2\kappa_g^2\|\xi(x)\|^2\right) \\ &\leq f(x_k) - \alpha\|\nabla f(x_k)\|^2 + \alpha\varepsilon\kappa_g\|\nabla f(x_k)\| + \frac{\alpha^2}{2}\kappa_f\|\nabla f(x)\|^2 + \alpha^2\kappa_f\varepsilon\kappa_g\|\nabla f(x_k)\| + \frac{\alpha^2\varepsilon^2}{2}\kappa_f\kappa_g^2 \end{aligned} \tag{47}$$

Using the requirement $\alpha \leq \min(\frac{1}{\kappa_g}, \frac{1}{\kappa_f})$ it follows that $\alpha\kappa_g \leq 1$ and $\alpha\kappa_f \leq 1$ so

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \alpha\|\nabla f(x_k)\|^2 + \varepsilon\|\nabla f(x_k)\| + \frac{\alpha}{2}\|\nabla f(x)\|^2 + \varepsilon\|\nabla f(x_k)\| + \frac{\varepsilon^2}{2}\kappa_g \\ &= f(x_k) - \frac{\alpha}{2}\|\nabla f(x_k)\|^2 + 2\varepsilon\|\nabla f(x_k)\| + \frac{\varepsilon^2}{2}\kappa_g \end{aligned} \tag{48}$$

Now, for α s.t. $\alpha \geq \frac{5\varepsilon}{\|\nabla f(x_k)\|}$ then $\varepsilon \leq \|\nabla f(x_k)\|\frac{\alpha}{5}$. Plugging it to our inequality we obtain

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \frac{\alpha}{2}\|\nabla f(x_k)\|^2 + \frac{2\alpha}{5}\|\nabla f(x_k)\|^2 + \frac{\alpha^2}{100}\kappa_g\|\nabla f(x_k)\|^2 \\ &\leq f(x_k) - \frac{\alpha}{2}\|\nabla f(x_k)\|^2 + \frac{2\alpha}{5}\|\nabla f(x_k)\|^2 + \frac{\alpha}{100}\|\nabla f(x_k)\|^2 \\ &= f(x_k) - 0.09\alpha\|\nabla f(x_k)\|^2 \\ &\leq f(x_k) - 2.25\frac{\varepsilon^2}{\alpha} \end{aligned} \tag{49}$$

Therefore, we obtain a monotonically decreasing step with finite size improvement, hence after a finite number of steps we obtain $x^\star$ for which $\|\nabla f(x^\star)\| \leq \frac{5\varepsilon}{\alpha}$. □

Corollary 2. *When Theorem 2 is satisfied for all k, the gradient descent iteration converges to a stationary point x^* s.t. $\frac{1}{K}\sum_{k=1}^{K}\|f(x_k)\|^2 \leq \frac{12|f(x_0)-f(x^*)|}{\alpha_K K}$.*

Proof. Recall that for all k we have

$$f(x_{k+1}) \leq f(x_k) - 0.09\alpha_k\|\nabla f(x_k)\|^2 \Rightarrow \|\nabla f(x_k)\|^2 \leq \frac{12}{\alpha_k}(f(x_k) - f(x_{k+1})) \tag{50}$$

Summing over these terms we obtain

$$\sum_{k=0}^{K-1}\|\nabla f(x_k)\|^2 \leq \sum_{k=0}^{K-1}\frac{12}{\alpha_k}(f(x_k) - f(x_{k+1})) \leq \frac{12}{\alpha_K}\sum_{k=0}^{K-1}(f(x_k) - f(x_{k+1})) = \frac{12}{\alpha_K}(f(x_0) - f(x_K)) \tag{51}$$

Since the domain is bounded and the gradient is Lipschitz continuous, from the fundamental theorem of line integral it follows that the function is bounded. Hence, a monotonically decreasing sequence bounded from below, must converge to the sequence infimum denoted as x^*. Therefore,

$$\sum_{k=0}^{K-1}\|\nabla f(x_k)\|^2 \leq \frac{12}{\alpha_K}|f(x_0) - f(x_K)| \leq \frac{12}{\alpha_K}|f(x_0) - f(x^*)| \tag{52}$$

□

B. The Perturbed Mean-Gradient

Definition 4. *The perturbed mean-gradient in x with averaging radius $\varepsilon > 0$ and perturbation radius $p < \varepsilon$ is*

$$g_\varepsilon^p(x) = \arg\min_{g\in\mathbb{R}^n} \iint\limits_{V_\varepsilon(x)B_p(x)} |g\cdot(\tau - s) - f(\tau) + f(s)|^2 ds d\tau \tag{53}$$

where $B_p(x) \subset V_\varepsilon(x) \subset \mathbb{R}^n$ are convex subsets s.t. $\|x' - x\| \le \varepsilon$ for all $x' \in V_\varepsilon(x)$ and the integral domain is over $\tau \in V_\varepsilon(x)$ and $s \in B_p(x)$.

We denote $V_\varepsilon(x)$ as the averaging domain and $B_p(x)$ as the perturbation domain and usually set $|V_\varepsilon| \gg |B_p|$. The purpose of V_ε is to average the gradient in a region of radius ε and the perturbation is required to obtain smooth gradients around discontinuity points.

Proposition 4 (controllable accuracy)**.** *For any function $f \in \mathcal{C}^1$, there is $\kappa_g > 0$, so that for any $\varepsilon > 0$ the perturbed mean-gradient satisfies $\|g_\varepsilon^p - \nabla f(x)\| \le \kappa_g \varepsilon$ for all $x \in \Omega$.*

Proof. Recall the Taylor theorem for a twice differentiable function $f(\tau) = f(s) + \nabla f(s)\cdot(\tau - s) + R_s(\tau)$, where $R_s(\tau)$ is the reminder. Since the gradient is continuous, we can write

$$f(\tau) = f(s) + \nabla f(s)\cdot(\tau - s) + \int_0^1 (\nabla f(s + t(\tau - s)) - \nabla f(s))\cdot(\tau - s)dt \tag{54}$$

Since $f \in \mathcal{C}^1$, we also have $|\nabla f(x) - \nabla f(s)| \le \kappa_f \|x - s\|$. We can use this property to bound the reminder in the Taylor expression.

$$\begin{aligned} R_s(\tau) &= \int_0^1 (\nabla f(s + t(\tau - s)) - \nabla f(s))\cdot(\tau - s)dt \\ &\le \kappa_f \int_0^1 \|s + t(\tau - s) - s\|\cdot\|\tau - s\|dt \le \frac{\kappa_f}{2}\|\tau - s\|^2 \end{aligned} \tag{55}$$

By the definition of g_ε^p, an upper bound for $\mathcal{L}(g_\varepsilon^p(x))$ is

$$\begin{aligned} \mathcal{L}(g_\varepsilon(x)) \le \mathcal{L}(\nabla f(x)) &= \iint\limits_{V_\varepsilon(x)B_p(x)} |\nabla f(x)\cdot(\tau - s) - f(\tau) + f(s)|^2 ds d\tau \\ &= \iint\limits_{V_\varepsilon(x)B_p(x)} |(\nabla f(x) - \nabla f(s) + \nabla f(s))\cdot(\tau - s) - f(\tau) + f(s)|^2 ds d\tau \\ &\le \iint\limits_{V_\varepsilon(x)B_p(x)} |\nabla f(s)\cdot(\tau - s) - f(\tau) + f(s)|^2 ds d\tau \\ &\quad + 2\iint\limits_{V_\varepsilon(x)B_p(x)} \|\nabla f(x) - \nabla f(s)\|\cdot\|\tau - s\|\cdot|\nabla f(s)\cdot(\tau - s) - f(\tau) + f(s)| ds d\tau \\ &\quad + \iint\limits_{V_\varepsilon(x)B_p(x)} \|\nabla f(x) - \nabla f(s)\|^2\cdot\|\tau - s\|^2 ds d\tau \\ &\le \iint\limits_{V_\varepsilon(x)B_p(x)} \frac{\kappa_f^2}{4}\|\tau - s\|^4 + \kappa_f^2\|x - s\|\cdot\|\tau - s\|^3 + \kappa_f^2\|x - s\|\cdot\|\tau - s\|^2 ds d\tau \\ &\le 16\kappa_f^2\varepsilon^4|V_\varepsilon(x)||B_p(x)| = 16\kappa_f^2\varepsilon^{n+4}p^n|V_1(x)||B_1(x)| \end{aligned}$$

Noticed that we used the inequalities: (1) $\|\nabla f(x) - \nabla f(s)\| \le \kappa_f\|x - s\| \le \kappa_f\varepsilon$, (2) $\|x - s\| \le \varepsilon$; and (3) $\|\tau - s\| \le 2\varepsilon$.

For the lower bound we assume that $p = \varepsilon\bar{p}$ and $\bar{p} < \frac{1}{4}$. Note that for any other upper bound on $\bar{p}$ we can derive an alternative bound.

The lower bound is

$$
\begin{aligned}
\mathcal{L}(g_\varepsilon(x)) &= \iint\limits_{V_\varepsilon(x)B_p(x)} |g_\varepsilon(x)\cdot(\tau-s) - f(\tau) + f(s)|^2 ds d\tau \\
&= \iint\limits_{V_\varepsilon(x)B_p(x)} |(g_\varepsilon(x) - \nabla f(x) + \nabla f(x))\cdot(\tau-s) - f(\tau) + f(s)|^2 ds d\tau \\
&\geq \iint\limits_{V_\varepsilon(x)B_p(x)} |(g_\varepsilon(x) - \nabla f(x))\cdot(\tau-s)|^2 - 2|(g_\varepsilon(x) - \nabla f(x))\cdot(\tau-s)|\cdot|\nabla f(x) - f(\tau) + f(s)| ds d\tau \\
&\geq \iint\limits_{V_\varepsilon(x)\setminus V_{\frac{3\varepsilon}{4}}(x)B_p(x)} |(g_\varepsilon(x) - \nabla f(x))\cdot(\tau-s)|^2 s d\tau \\
&- 4\varepsilon\|(g_\varepsilon(x) - \nabla f(x))\| \iint\limits_{V_\varepsilon(x)B_p(x)} \left(|\nabla f(s) - f(\tau) + f(s)| + |\nabla f(x) - \nabla f(s)|\right) ds d\tau \\
&\geq \|g_\varepsilon(x) - \nabla f(x)\|^2 \left(\frac{\varepsilon}{2}\right)^2 \iint\limits_{V_\varepsilon(x)\setminus V_{\frac{3\varepsilon}{4}}(x)B_p(x)} \left|\hat{\mathbf{n}}(x)\cdot\frac{\tau-s}{\|\tau-s\|}\right|^2 s d\tau \\
&- 4\varepsilon\|(g_\varepsilon(x) - \nabla f(x))\| \iint\limits_{V_\varepsilon(x)B_p(x)} \frac{1}{2}\kappa_f\|\tau-s\|^2 + \kappa_f\|x-s\|^2 ds d\tau \\
&\geq \|g_\varepsilon(x) - \nabla f(x)\|^2 \varepsilon^n p^n \left(\frac{\varepsilon}{2}\right)^2 \iint\limits_{V_1(x)\setminus V_{\frac{3}{4}}(x)B_1(x)} \left|\hat{\mathbf{n}}(x)\cdot\frac{\tau-s}{\|\tau-s\|}\right|^2 s d\tau - 2.5\varepsilon\|(g_\varepsilon(x) - \nabla f(x))\|\varepsilon^n p^n \kappa_f \frac{27}{32}\varepsilon^2 |V_1(x)||B_1(x)| \\
&= \frac{1}{4}\varepsilon^{n+2} p^n M_1 \|(g_\varepsilon(x) - \nabla f(x))\|^2 - 2.5\varepsilon^{n+3} p^n \kappa_f \|(g_\varepsilon(x) - \nabla f(x))\||V_1(x)||B_1(x)|
\end{aligned}
$$

Combining the lower and upper bound we obtain

$$
\begin{aligned}
& M_1\|(g_\varepsilon(x) - \nabla f(x))\|^2 - 10\varepsilon\kappa_f\|(g_\varepsilon(x) - \nabla f(x))\||V_1(x)||B_1(x)| - 64\kappa_f^2\varepsilon^2|V_1(x)||B_1(x)| \leq 0 \\
\Rightarrow\ & \|(g_\varepsilon(x) - \nabla f(x))\| \leq \kappa_g\varepsilon
\end{aligned}
$$

where

$$
\kappa_g = \kappa_f \frac{10|V_1(x)||B_1(x)| + \sqrt{100|V_1(x)|^2|B_1(x)|^2 + 256|V_1(x)||B_1(x)|M_1(x)}}{M_1(x)}
$$

□

Proposition 5 (continuity)**.** *If $f(x)$ is Riemann integrable in $V_\varepsilon(x) \subset V$ then the perturbed mean-gradient is a continuous function at x.*

Proof. We follow the same line of arguments as in Proposition 2, yet here we need to show that $\mathcal{L}(x, g)$ is continuous for any interable function f.

$$
\begin{aligned}
|\mathcal{L}(x,g)-\mathcal{L}(x',g)| &= |g\cdot\mathbf{A}(x)g+g\cdot b(x)-g\cdot\mathbf{A}(x')g-g\cdot b(x)| = |g\cdot b(x)-g\cdot b(x)| \\
&\leq \|g\|\left\| \iint\limits_{V_\varepsilon(x)B_p(x)} |g\cdot(\tau-s)-f(\tau)+f(s)|^2 ds d\tau - \iint\limits_{V_\varepsilon(x')B_p(x')} |g\cdot(\tau-s)-f(\tau)+f(s)|^2 ds d\tau \right\| \\
&\leq \|g\| \iint\limits_{V_\varepsilon(x)B_p(x)\setminus V_\varepsilon(x')B_p(x')} |g\cdot(\tau-s)-f(\tau)+f(s)|^2 ds d\tau \\
&+ \|g\| \iint\limits_{V_\varepsilon(x')B_p(x')\setminus V_\varepsilon(x)B_p(x)} |g\cdot(\tau-s)-f(\tau)+f(s)|^2 ds d\tau \\
&\leq M\|g\|\cdot|V_\varepsilon(x)B_p(x)\setminus V_\varepsilon(x')B_p(x')| + M\|g\|\cdot|V_\varepsilon(x')B_p(x')\setminus V_\varepsilon(x)B_p(x)|
\end{aligned}
$$

Applying the same arguments in Lemma 4, we have $|V_\varepsilon(x)B_p(x)\setminus V_\varepsilon(x')B_p(x')| \leq |A_{V_\varepsilon}(x)|\cdot|A_{B_p}(x)|\|x-x'\|^2$, where $A_{V_\varepsilon}(x)$ is the surface of $V_\varepsilon(x)$ and $A_{B_p}(x)$ is the surface of $B_p(x)$. Therefore, $\mathcal{L}(x,g)$ is continuous.

The rest of the proof, again, is identical to Proposition 2.

□

B.1. Monte-Carlo approximation of the perturb mean-gradient

For a parameterization g_θ, we may learn the perturb mean-gradient by sampling a reference point x_r and then uniformly sampling two evaluation points $x_i \sim U(B_p(x_r))$ $x_j \sim U(V_\varepsilon(x_r))$. With the tuples (x_r, x_i, x_j) we minimize the following loss

$$
\mathcal{L}_{\varepsilon,p}(\theta) = \sum_{x_r}\sum_{x_i}\sum_{x_j} |g_\theta(x_r)\cdot(x_j-x_i)-f(x_j)+f(x_i)|^2
$$

Since $x_i \sim U(B_p(x_r))$, we can write $x_i = x_r + n_i$ where n_i is uniformly sampled in an n-ball with p radius. To reduce the number of evaluation points, we may choose to fix x_i and sample $x_r = x_i + n_r$. If we assume that $\varepsilon \gg p$ then for a sample $x_j \sim U(V_\varepsilon(x_i))$ with very high probability we have that $\|x_r - x_j\| \leq \varepsilon$. So we can approximate $\mathcal{L}_{\varepsilon,p}$ with

$$
\mathcal{L}_{\varepsilon,p}(\theta) = \sum_{n_r}\sum_{x_i}\sum_{x_j} |g_\theta(x_i+n_r)\cdot(x_j-x_i)-f(x_j)+f(x_i)|^2
$$

C. Spline Embedding

When fitting f with a NN, we found out that feeding the input vector x directly into a Fully Connected NN provides unsatisfactory results when the dimension of the data is too small or when the target function is too complex. Specifically, gradient descent (with Adam optimizer (Kingma & Ba, 2014)) falls short in finding the global optimum. We did not investigate theoretically into this phenomena, but we designed an alternative architecture that significantly improves the learning process. This method adds a preceding embedding layer (Zhang et al., 2016) before the NN. These embeddings represent a set of learnable Spline functions (Reinsch, 1967).

Categorical Feature embedding (Howard & Gugger, 2020) is a strong, common practice, method to learn representations of multi-categorical information. It is equivalent to replacing the features with their corresponding one-hot vector representation and concatenating the one-hot vectors into a single vector which is then fed to the input of a NN. An important advantage of categorical embedding is the ability to expand the input dimension into an arbitrary large vector size. In practice, this expansion can help in representing complex non-linear problems.

For ordinal data, however, embedding may be viewed as an unnecessary step as one can feed the data directly into a NN input layer. Moreover, categorical feature embeddings do not preserve ordinality within each categorical variable as each class is assigned a different independent set of learnable embeddings. Nevertheless, motivated by the ability to expand the input dimension into an arbitrary large number, we designed an ordinal variable embedding that is Lipschitz continuous s.t. for two relatively close inputs x_1 and x_2 the embedding layer outputs $s(x_1)$ and $s(x_2)$ s.t. $\|s(x_1) - s(x_2)\| \leq \kappa_s \|x_1 - x_2\|$. To that end, for a given input vector $x \in \mathbb{R}^n$, we define the representation as $s_\theta : \mathbb{R}^n \to \mathbb{R}^{n_s}$, $y = s_\theta(x)$, where each entry $s^j_\theta(x^l)$ is a one-dimensional learnable Spline transformation. A Spline (Reinsch, 1967) is a piecewise polynomial with some degree of smoothness in the connection points. Spline is usually used to approximate smooth functions but here we use it to represent a learnable function.

To define a learnable spline, we need to determine the intersection points and the spline degree. Specifically, for a domain $x^i \in [a, b]$ we equally divided the domain into k intersection points, where each point is also termed as knot (in this work $[a, b] = [-1, 1]$ and $k = 21$, s.t. each segment is 0.1 long). Our next step is to define the spline degree and smoothness. We experimented with three options: (1) continuous piecewise linear splines (2) 3rd degree polynomials with continuous second derivative, termed C^2 Cubic spline and; (3) continuous C^0 Cubic splines. We found out that for the purpose of EGL, continuous piecewise linear splines yield the best performance and requires less computational effort. The explicit definition of a piecewise linear spline is

$$s(x, \theta) = \frac{\theta_i}{h_i}(x - t_{i-1}) + \frac{\theta_i}{h_i}(t_i - x) \tag{56}$$

where θ is a k elements (k is the number of knots), t_i is the location of the i-th knot and $h_i = t_i - t_{i-1}$.

We can learn more than a single spline for each element in the x vector. In this work we learned e different splines for each entry in x s.t. the output shape of the embedding block is $n \times e$. It is also possible to learn two or more dimensional splines but the number of free parameters grows to the power of the splines dimensions. Therefore, it is non practical to calculate these high degree splines. To calculate interactions between different entries of x we tested two different methods: (1) aggregation functions and; (2) attention aggregation after a non-local blocks (Wang et al., 2018).

In the first option, given a spline representation $s(x) \in \mathbb{R}^{n \times e}$ an average pooling aggregation is executed along the 1st dimension s.t. we end up with a $\bar{s}(x) \in \mathbb{R}^e$ representation vector. In the second option, the aggregation takes place after a non-local blocks which calculates interactions between each pairs of entries in the 1st dimension of $s(x)$ (i.e. the input dimension). To preserve the information of the input data, we concatenated x to the output of the aggregation layer. After the concatenation, stacks of Residual blocks (He et al., 2016) (Res-Blocks) layers have been applied to calculate the output vector (size of 1 in IGL and size of n in EGL). The complete Spline Embedding architecture that includes both average pooling aggregation and non-local blocks is presented in Fig. 7.

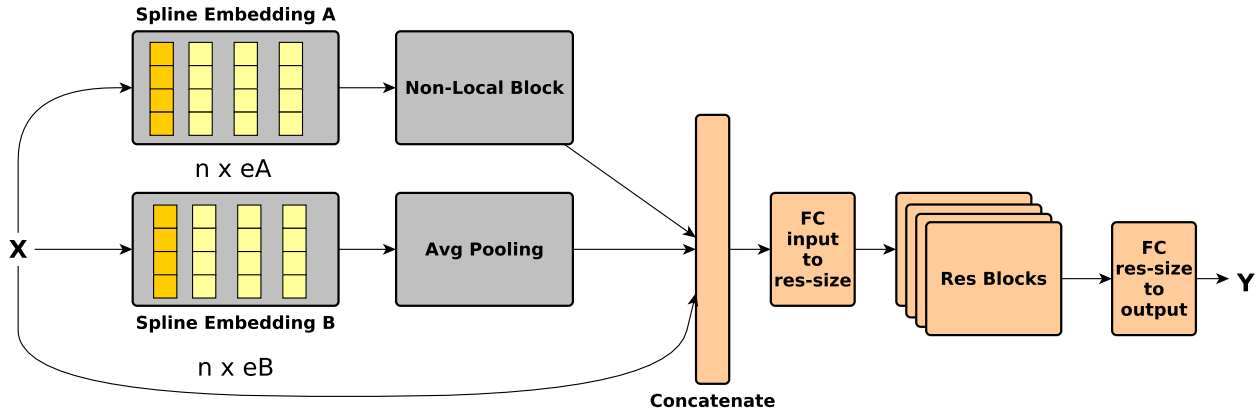

Figure 7. The Spline Architecture

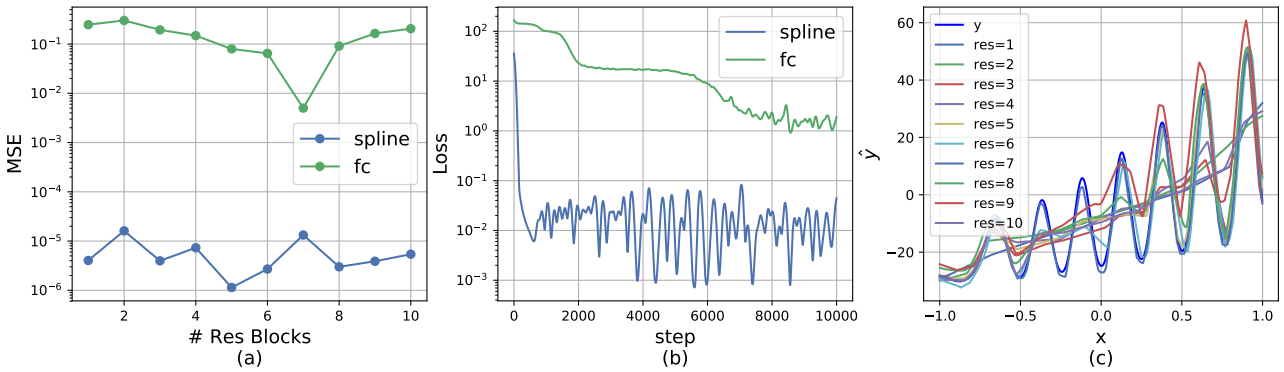

Figure 8. Comparing Spline fitting vs standard FC fitting.

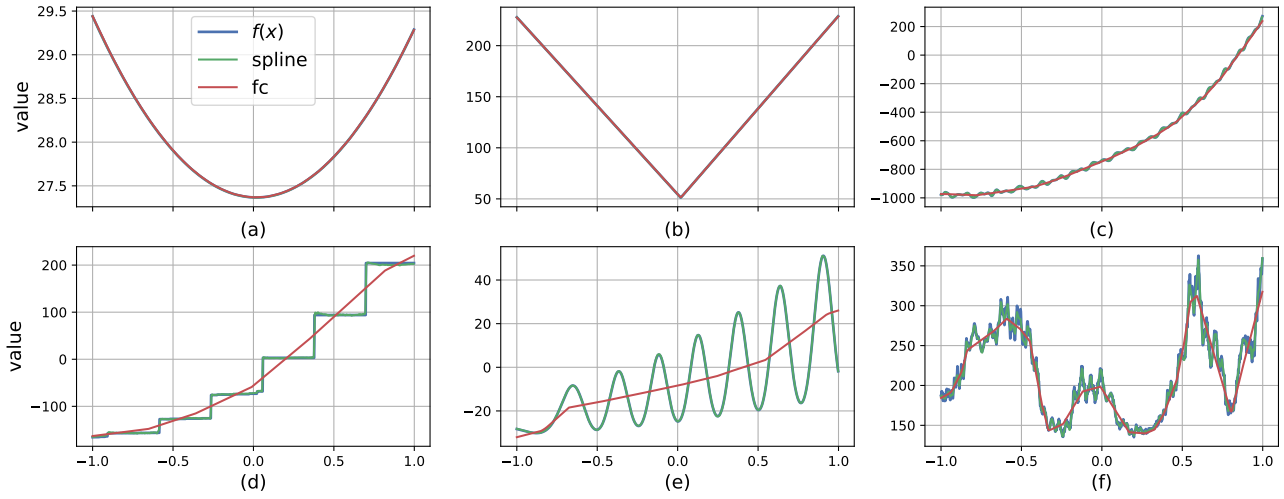

Figure 9. Comparing Spline fitting vs standard FC fitting.

In the next set of experiments, we evaluate the benefit of Spline embedding in 1D COCO problems. We compared the Spline architecture in Fig. 7 to the same architecture without the spline embedding branches (x is directly fed to the FC layer input). We used only $e = 8$ splines and a Res-Block layer size of 64. In Fig. 8(a), we evaluate the learning of a single problem (246, harmonic decaying function) with different number of Res-Blocks. Here, we used a mini-batch size of 1024 and a total of 1024 mini-batches iteration to learn the function (i.e. a total of 10^6 samples). We see that Spline embedding obtains much better MSE even for a single Res-Block and maintains its advantage for all the Res-Block sizes which we evaluated. Note that each Res-Block comprises two Fully Connected layers, thus with the additional input and output layers we have $2n + 2$ FC layers for n Res-Blocks.

In Fig. 8(b) we evaluated the learning process with 2 Res-Blocks for 10240 mini-batches (10^7 samples). We see that Spline embedding converges after roughly 500 minibatches while the FC layer learns very slowly. Interestingly, each significant drop in the loss function of the FC net corresponds to a fit of a different ripple in the harmonic decaying function. It seems like the FC architecture converges to local minima that prevent the network from fitting the entire harmonic function. This can be seen in Fig. 8(c) where we print the results of the learned FC models for different Res-Block sizes after 1024 mini-batches. The results show that all FC networks fail to fit the harmonic function completely.

To demonstrate expressiveness of Spline embedding, we fit the 4 functions in the 1-D illustrative examples in Sec. 3 and two additional functions: the harmonic decaying function and a noise like function. The results are presented in Fig. 9. Remarkably, while we use only $e = 8$ splines which sums up to only 680 additional weights (8×21 spline parameters and additional 8×64 input weights), we obtain significantly better results than the corresponding FC architecture.[4]

[4] In 1D problems there is no aggregation step.

D. Mappings

By applying the chain rule and the inverse function theorem, we can express the gradient of the original problem ∇f with the gradient of the scaled problem $\nabla \tilde{f}$:

$$\frac{\partial f(x)}{\partial x^l} = \left(\frac{\partial r_k}{\partial y}\right)^{-1} \frac{\partial h_j(x)}{\partial x^l} \frac{\partial \tilde{f}_{jk}(\tilde{x})}{\partial \tilde{x}^l} \tag{57}$$

Here, ∂x^l is the partial derivative with respect to the l-th entry of x (we assume that h maps each element independently s.t. the Jacobian of h is diagonal). For strictly linear mappings, it is easy to show that this property also holds for the mean-gradients.

Proposition 6. *Let $h_j : \mathbb{R}^n \to \mathbb{R}^n$ and $r_k : \mathbb{R} \to \mathbb{R}$ be two linear mapping functions s.t., $r_k(y) = \frac{y-\mu_k}{\sigma_k}$ and $h_j^l(x) = a_j^l x + b_j^l$, then the mean-gradient g_ε of f can be recovered from the mean-gradient $\tilde{g}_{\tilde{\varepsilon}}$ of $\tilde{f}$ with*

$$g_\varepsilon^l(x) = \frac{a_j^l}{\sigma_k} \tilde{g}_{\tilde{\varepsilon}}^l(\tilde{x}) \tag{58}$$

where V_ε is the projection $h_j^{-1}(V_{\tilde{\varepsilon}})$ which is bounded by an n-ball at x with radius $\varepsilon = \max_l \frac{1}{a_j^l}\tilde{\varepsilon}$, i.e. for all $x' \in V_\varepsilon(x), \|x' - x\| \leq \max_l \frac{1}{a_j^l}\tilde{\varepsilon}$.

Proof. Let us write the definition of $g_{\tilde{\varepsilon}}$ with a variable $\tilde{\tau}$ s.t. $\tilde{\tau} \in V_{\tilde{\varepsilon}}(\tilde{x})$ (contrary to the original definition where τ denoted the difference s.t. $x + \tau \in V_\varepsilon(x)$)

$$g_{\tilde{\varepsilon}}(\tilde{x}) = \arg\min_g \int_{\tilde{\tau} \in V_{\tilde{\varepsilon}(\tilde{x})}} |g \cdot (\tilde{\tau} - \tilde{x}) - \tilde{f}(\tilde{\tau}) + \tilde{f}(\tilde{x})|^2 d\tilde{\tau} \tag{59}$$

Recall the mapping $\tilde{x} = h(x)$, since it is invertable mapping, there exist τ s.t. $\tilde{\tau} = h(\tau)$. Substituting $\tilde{\tau}$ with τ, the integral becomes

$$g_{\tilde{\varepsilon}}(\tilde{x}) = \arg\min_g \int_{\tau \in h^{-1}(V_{\tilde{\varepsilon}(\tilde{x})})} |g \cdot (h(\tau) - h(x)) - \tilde{f}(h^{-1}(\tau)) + \tilde{f}(h^{-1}(x))|^2 |\det(Dh(\tau))| d\tau \tag{60}$$

where $\det(Dh(\tau))$ denotes the determinant of the Jacobian matrix of the mapping h. This determinant is constant for linear mapping so we can ignore it as we search for the arg-min value. We can also multiply the integral by the inverse slope $\frac{1}{\sigma_r}$ and get

$$\begin{aligned} g_{\tilde{\varepsilon}}(\tilde{x}) &= \arg\min_g \int_{\tau \in h^{-1}(V_{\tilde{\varepsilon}(\tilde{x})})} |\frac{1}{\sigma_r} g \cdot (h(\tau - x)) - \frac{1}{\sigma_r}\tilde{f}(h^{-1}(\tau)) + \frac{1}{\sigma_r}\tilde{f}(h^{-1}(x))|^2 d\tau \\ &= \arg\min_g \int_{\tau \in h^{-1}(V_{\tilde{\varepsilon}(\tilde{x})})} |\frac{1}{\sigma_r} a_j \odot g \cdot (\tau - x) - r^{-1}(\tilde{f}(h^{-1}(\tau))) + r^{-1}(\tilde{f}(h^{-1}(x)))|^2 d\tau \\ &= \arg\min_g \int_{\tau \in h^{-1}(V_{\tilde{\varepsilon}(\tilde{x})})} |\frac{1}{\sigma_r} a_j \odot g \cdot (\tau - x) - f(\tau) + f(x)|^2 d\tau \end{aligned} \tag{61}$$

Where the last equality holds since $r^{-1} \circ \tilde{f} \circ h = r^{-1} \circ r \circ f \circ h^{-1} \circ h = f$. Since the mapping $g \to \frac{1}{\sigma_r} a_j \odot g$ is bijective, the arg-min can be rephrased as

$$\frac{1}{\sigma_r} a_j \odot g_{\tilde{\varepsilon}}(\tilde{x}) = \arg\min_g \int_{\tau \in h^{-1}(V_{\tilde{\varepsilon}(\tilde{x})})} |g \cdot (\tau - x) - f(\tau) + f(x)|^2 d\tau \tag{62}$$

which is exactly the definition for g_ε so we get that $g_\varepsilon = \frac{1}{\sigma_r} a_j \odot g_{\tilde{\varepsilon}}(\tilde{x})$, as requested. Finally, we need to show that for all $\tau \in V_\varepsilon(x)$, $\|\tau - x\| \leq \max_l \frac{1}{a^l}\tilde{\varepsilon}$.

$$\|\tau - x\| = \|h^{-1}(\tilde{\tau}) - h^{-1}(\tilde{x})\| = \|h^{-1}(\tilde{\tau} - \tilde{x})\| \leq \|h^{-1}\| \|\tilde{\tau} - \tilde{x}\| \leq \max_l \frac{1}{a^l}\tilde{\varepsilon} \tag{63}$$

□

As discussed in Sec. 4.3, the design goals for mappings are twofold: (1) Fix the statistics of the input and output data and; (2) maintain the linearity as much as possible. Following these two goals we explored mappings of the form $y = q(l_{\mathbf{a}}(x))$, where q is an expansion non-linear mapping $\Omega \to \mathbb{R}^n$ for the input mapping and a squash mapping $\mathbb{R} \to \mathbb{R}$ for the output mapping. $l_{\mathbf{a}}$ is a linear mapping that is defined by the $\mathbf{a}$ parameters. For example in the scalar case we can uniquely define the linear function by mapping x_1 to y_1 and x_2 to y_2, in this case we denote $\mathbf{a} = [(x_1, y_1), (x_2, y_2)]$.

D.1. Input Mapping

Given a candidate solution x_{j-1}, we first construct a bounding-box Ω_j by squeezing the previous region by a factor of γ_α and placing it s.t. x_{j-1} is in the bounding-box center. For a region Ω_j such that the upper and lower bounds are found in $[b_l, b_u]$, we, first, construct a linear mapping of the form $\mathbf{a} = [(b_l, -1), (b_u, 1)]$. Then, our expansion function is the inverse hyperbolic tangent $\operatorname{arctanh}(x) = \frac{1}{2} \log\left(\frac{1+x}{1-x}\right)$. This function expands $[-1, 1] \to \mathbb{R}$ but maintains linearity at the origin. Given that the solution is approximately found in the center of the bounding-box we obtain high linearity except when the solution is found on the edges.

D.2. Output Mapping

For the output mapping we first fix the statistics with a linear mapping $\mathbf{a} = [(Q_{0.1}, -1), (Q_{0.9}, 1)]$ where $Q_{0.1}$ is the 0.1 quantile in the data and $Q_{0.9}$ is the 0.9 quantile. This mapping is also termed as robust-scaling as unlike z-score $\frac{x-\mu}{\sigma}$, it is resilient to outliers. On the downside it does not necessarily fix the first and second order statistics, but these are at least practically, bounded. The next step, i.e. squash mapping, makes sure that even outliers does not get too high values. For that purpose, we use the squash mapping

$$q(x) = \begin{cases} -\log(-x) - 1, & x < -1 \\ x, & -1 \leq x < 1 \\ \log(x) + 1, & x \geq 1 \end{cases} \tag{64}$$

E. The Practical EGL Algorithm

Algorithm 3 Explicit Gradient Learning

Input: $x_0, \Omega, \tilde{\alpha}, \tilde{\varepsilon}, \gamma_\alpha < 1, \gamma_\varepsilon < 1, n_{\max}$
$k = 0$
$j = 0$
$\Omega_j \leftarrow \Omega$
Map $h_0 : \Omega \to \mathbb{R}^n$
while *budget* $C > 0$ **do**
  Explore:
    Generate samples $\mathcal{D}_k = \{\tilde{x}_i\}_1^m, \tilde{x}_i \in V_{\tilde{\varepsilon}}(\tilde{x}_k)$
    Evaluate samples $y_i = f(h_0^{-1}(\tilde{x}_i)), \qquad i = 1, ..., m$
    Add samples to the replay buffer $\overline{\mathcal{D}} = \overline{\mathcal{D}} \cup \mathcal{D}_k$
  Output Map:
    $r_k = squash \circ l_{[Q_{0.1}, Q_{0.9}]}$
    $\tilde{y}_i = r_k(y_i)\,, \qquad i = 1, ..., m$
  Mean-Gradient learning:
    $\theta_k = \arg\min_\theta \sum_{q=0}^{l-1} \sum_{i,j \in \mathcal{D}_{k-q}} |(\tilde{x}_j - \tilde{x}_i) \cdot g_\theta(\tilde{x}_i) - \tilde{x}_j + \tilde{x}_i|^2$
  Gradient Descent:
    $x_{k+1} \leftarrow x_k - \tilde{\alpha} g_{\theta_k}(x_k)$
    if $f(h_j^{-1}(\tilde{x}_{k+1})) > f(h_j^{-1}(\tilde{x}_k))$ *for* $n_{\max}$ *times in a row* **then**
      Generate new trust-region s.t. $|\Omega_{j+1}| = \gamma_\alpha |\Omega_j|$ and its center at x_{best}
      Map $h_j : \Omega \to \mathbb{R}^n$
      $j \leftarrow j + 1$
      $\tilde{\varepsilon} \leftarrow \gamma_\varepsilon \tilde{\varepsilon}$
    if $f(h_j^{-1}(\tilde{x}_{k+1})) < f(h_j^{-1}(\tilde{x}_k))$ **then**
      $x_{best} = h_j^{-1}(\tilde{x}_k)$
  $k \leftarrow k + 1$
return x_{best}

F. Supplementary details: The COCO experiment

The COCO test suite provides many Black-Box optimization problems on several dimensions (2,3,5,10,20,40). For each dimension, there are 360 distinct problems. The problems are divided into 24 different classes, each contains 15 problems. To visualize all problem classes, we iterate over the 2D problem set and for each class we present (Fig. 10-15) a contour plot, 3D plot and the equivalent 1D problem ($f_{1D}(x) = f_{2D}(x, x)$) combined with the log view of the normalized problem ($\frac{f_{1D}(x) - f_{1D}^{min}}{f_{1D}^{max} - f_{1D}^{min}}$).

To visualize the average convergence rate of each method, we first calculate a scaled distance between the best value at time t and the optimal value $\Delta y_{best}^t = \frac{\min_{k \le t} y_k - y^*}{y_0 - y^*}$ where y^* is the minimal value obtained from all the baselines' test-runs. We then average this number, for each t, over all runs in the same dimension problem set. This distance is now scaled from zero to one and the results are presented on a log-log scale. The first-column in Fig. 10-15 presents $\overline{\Delta y}_{best}^t$ on each problem type of the 2D problem set and Fig. 16 show $\overline{\Delta y}_{best}^t$ in the 40D problem set. In table 1, we present the hyperparameters used for EGL and IGL in all our experiments (besides the ablation tests).

In future work, we will need to design a better mechanism for the ε scheduling. In problem 19 (Griewank-Rosenbrock F8F2), the ε scheduling was too slow, and only when we used a smaller initial ε, EGL started to converge to the global minimum (see Fig. 17(a) where we used initial $\varepsilon = 0.001 \times \sqrt{n}$, $\gamma_\alpha = 0.7$ and $L = 1$). On the other hand, in problems, 21 (Gallagher 101 peaks) and 22 (Gallagher 21 peaks) using small ε ends up in falling to local minima, and the choice of a larger ε could smooth the gradient which pushes x_k over the local minima (see 17(b-c) respectively where we used initial $\varepsilon = 0.5 \times \sqrt{n}$ and $L = 4$).

In Fig. 18-23 we present a histogram of the raw and scaled cost value (after the output-mapping) of a 200 samples snapshot from the replay buffer at different periods during the learning process ($t = 1K, t = 10K, t = 100K$). Typically, we expect that problems with Normal or Uniform distributions should be easier to learn with a NN (e.g. problems 15 (Rastrigin), 18 (Schaffer F7, cond1000), 23 (ats ras)), while problems with skewed distribution or multimodal distribution are much harder (e.g. problems 2 (Ellipsoid separable), 10 (Ellipsoid) and 11 (Discus)). However, simply, mapping from a hard distribution into a Normal distribution is not necessarily a good choice since we lose the mapping linearity s.t. the scaled mean-gradient may not correspond to the true mean-gradient. Thus, the output-mapping must balance between linearity and normalization. In future work, we would like to find better, more robust output-mappings that overcome this problem. Understanding the way that the values are distributed at run-time could also help us define a better mechanism for deciding on ε and the RB size L. If the function outputs are close to each other, large RB could be beneficial, but if the values have high variance, large RB could add unnecessary noise.

Table 1. The COCO experiment Hyperparameters

Parameter	Value	Description
n	[2,3,5,10,20,40,784]	coco space dimension
m	64	Exploration points
m_{warmup_factor}	5	$m \times m_{warmup\_factor}$ to adjust the network parameters around the TR initial point
`batch`	1024	Minibatch of EGL/IGL training
L	32	Number of exploration steps that constitute the replay buffer for EGL/IGL The replay memory size is: $RB = L \times m$
C	15×10^4	Budget
α	10^{-2}	Optimization steps' size
`g_lr`	10^{-3}	g_θ learning rate
γ_α	0.9	Trust region squeezing factor
γ_ε	0.97	ε squeezing factor
ε	$0.1 \times \sqrt{n}$	Initial exploration size
$n_{\max}$	10	The number of times in a row that $f(h_j^{-1}(\tilde{x}_{k+1})) > f(h_j^{-1}(\tilde{x}_k))$
$n_{\min}$	40	Minimum gradient descent iterations
p	0	Perturbation radius
g_θ	Spline	Network architecture (SPLINE/FC)
`OM`	log	Output Mapping
`OM_lr`	0.1	Moving average learning rate for the Output Mapping
`N_minibatches`	60	# of mini-batches for the mean-gradient learning in each k step
$V_\varepsilon(x)$	ball-explore	$V_\varepsilon(x) = x + \varepsilon \times U[-1, 1]$ (see Sec. H for details)

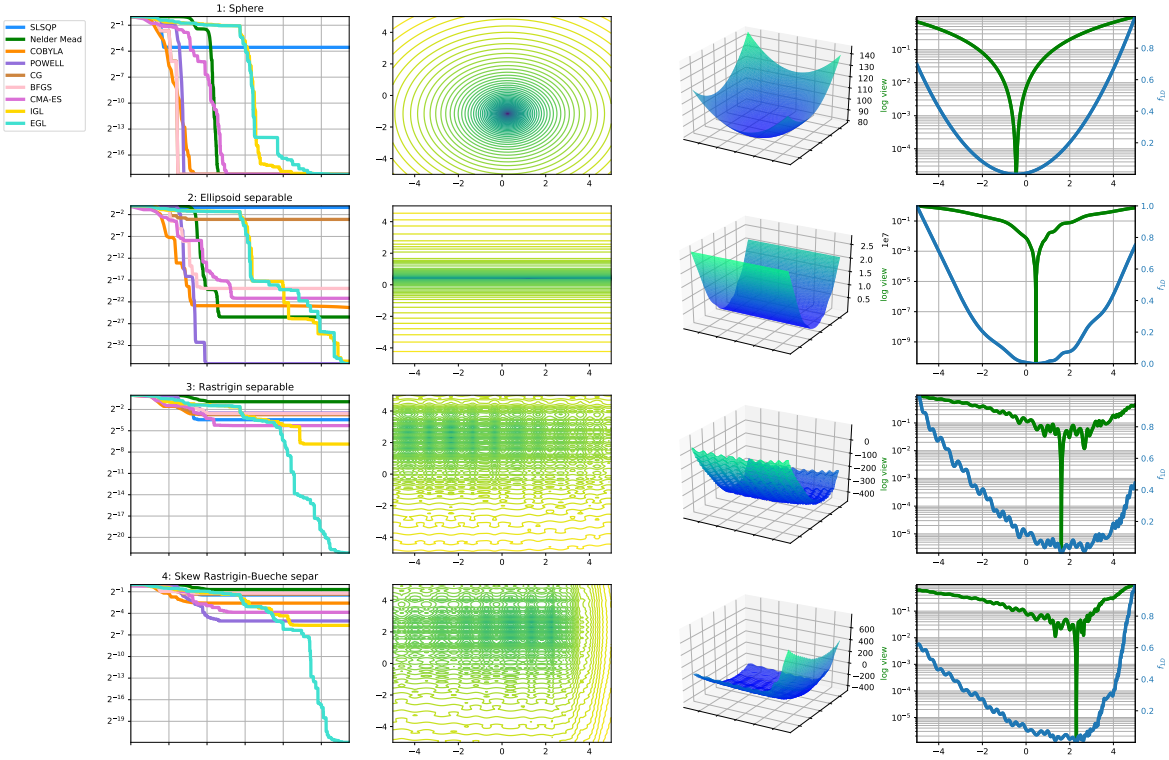

Figure 10. Visualization problems type 1-4 of 2D problems. First column: The scaled distance $\overline{\Delta y}^t_{best}$. Second column: Counter plot. Third column: 3D plot. Forth column: equivalent 1D problem with log view.

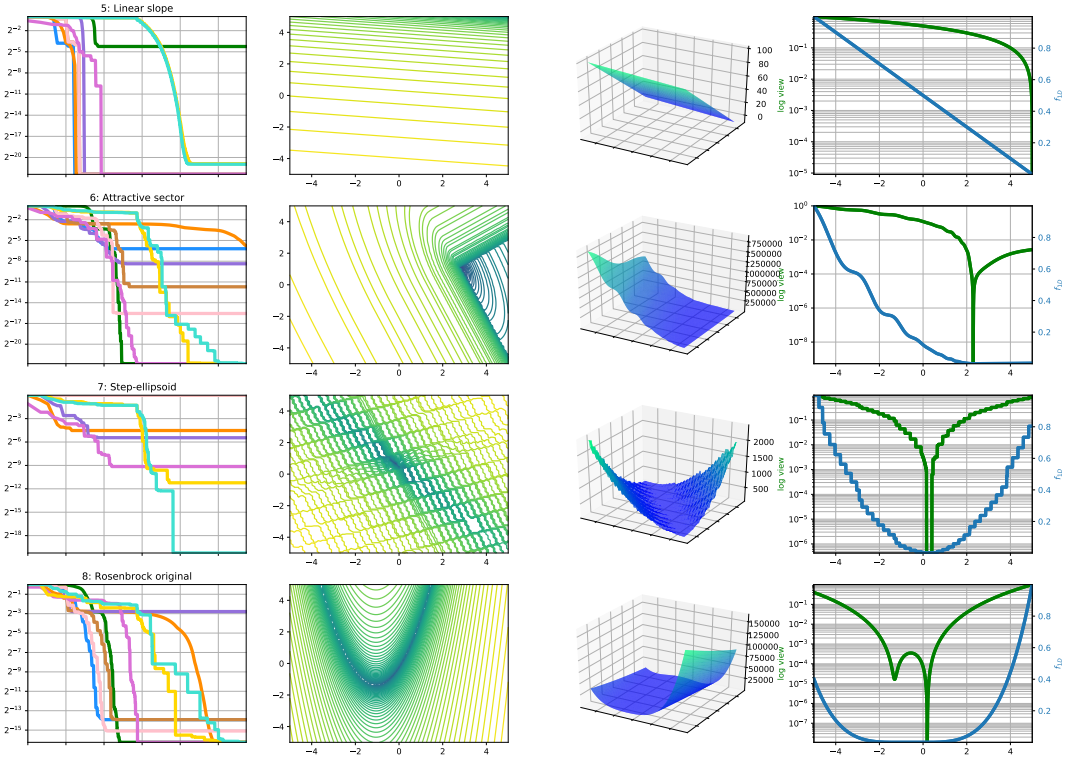

Figure 11. Visualization problems type 5-8 of 2D problems. First column: The scaled distance $\overline{\Delta y}^t_{best}$. Second column: Counter plot. Third column: 3D plot. Forth column: equivalent 1D problem with log view.

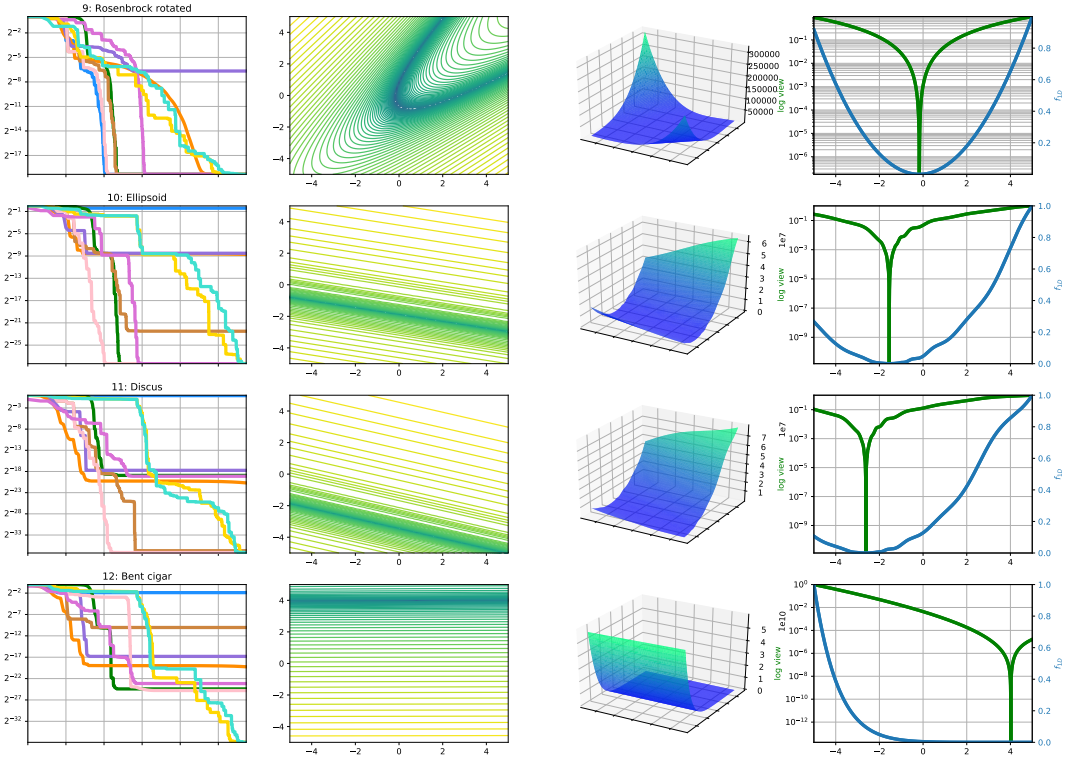

Figure 12. Visualization problems type 9-12 of 2D problems. First column: The scaled distance $\overline{\Delta y}^t_{best}$. Second column: Counter plot. Third column: 3D plot. Forth column: equivalent 1D problem with log view.

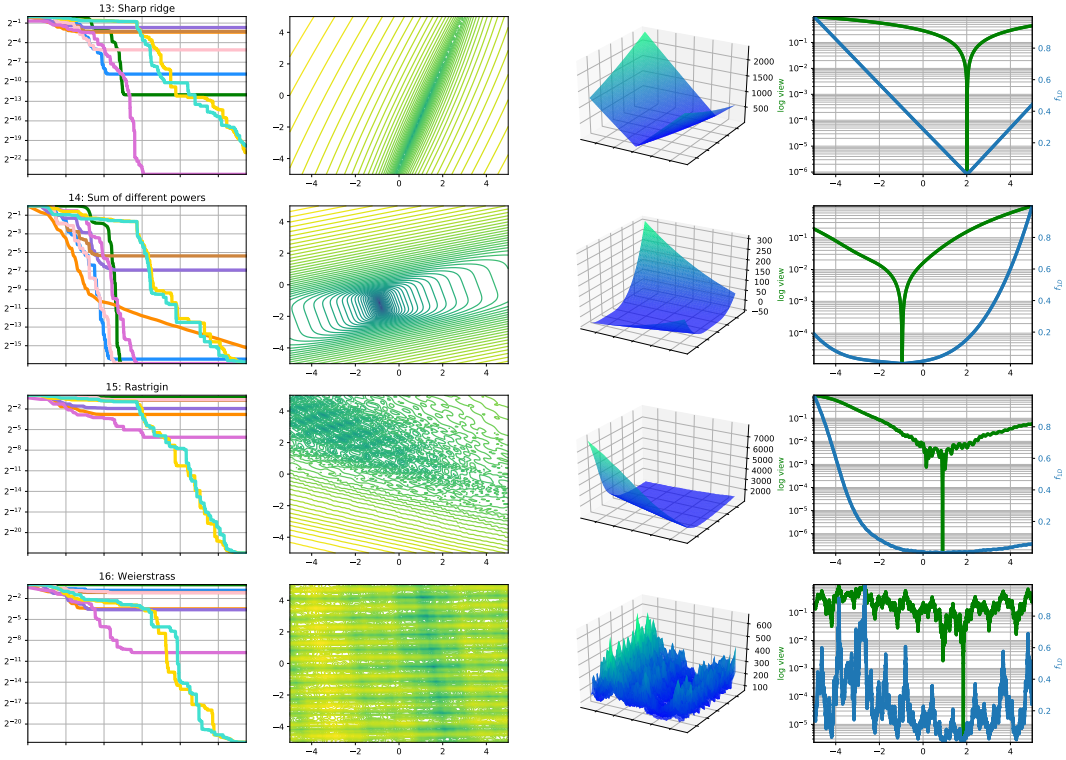

Figure 13. Visualization problems type 13-16 of 2D problems. First column: The scaled distance $\overline{\Delta y}^t_{best}$. Second column: Counter plot. Third column: 3D plot. Forth column: equivalent 1D problem with log view.

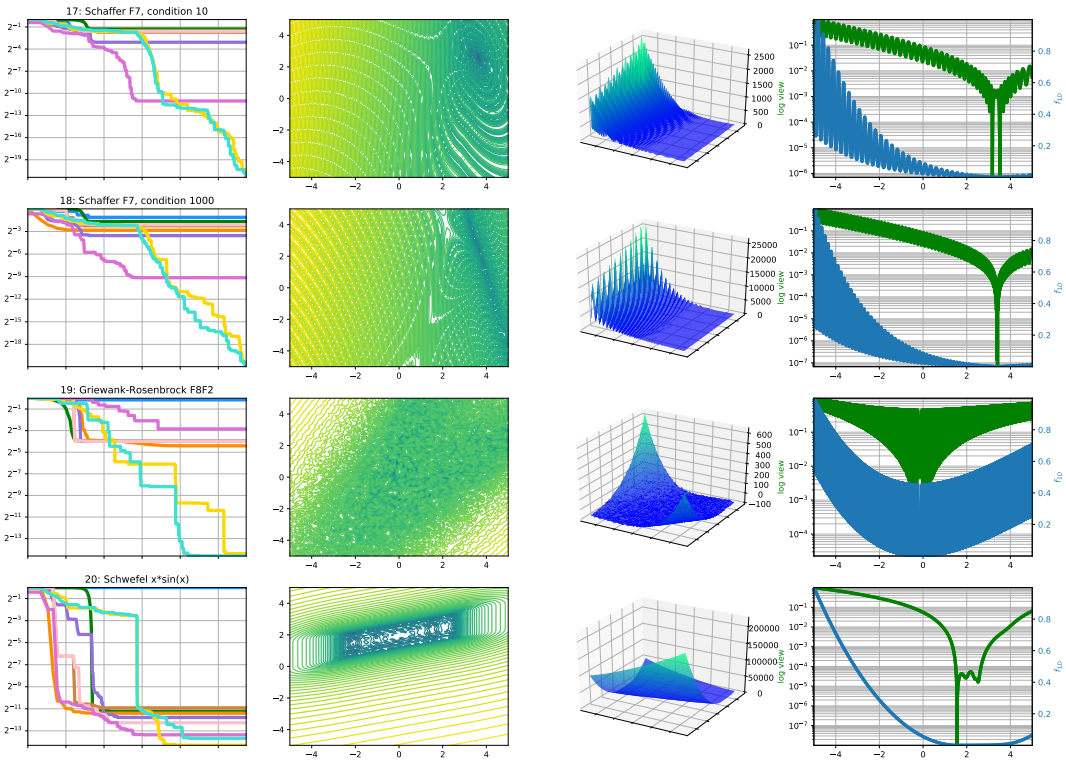

Figure 14. Visualization problems type 17-20 of 2D problems. First column: The scaled distance $\overline{\Delta y}^t_{best}$. Second column: Counter plot. Third column: 3D plot. Forth column: equivalent 1D problem with log view.

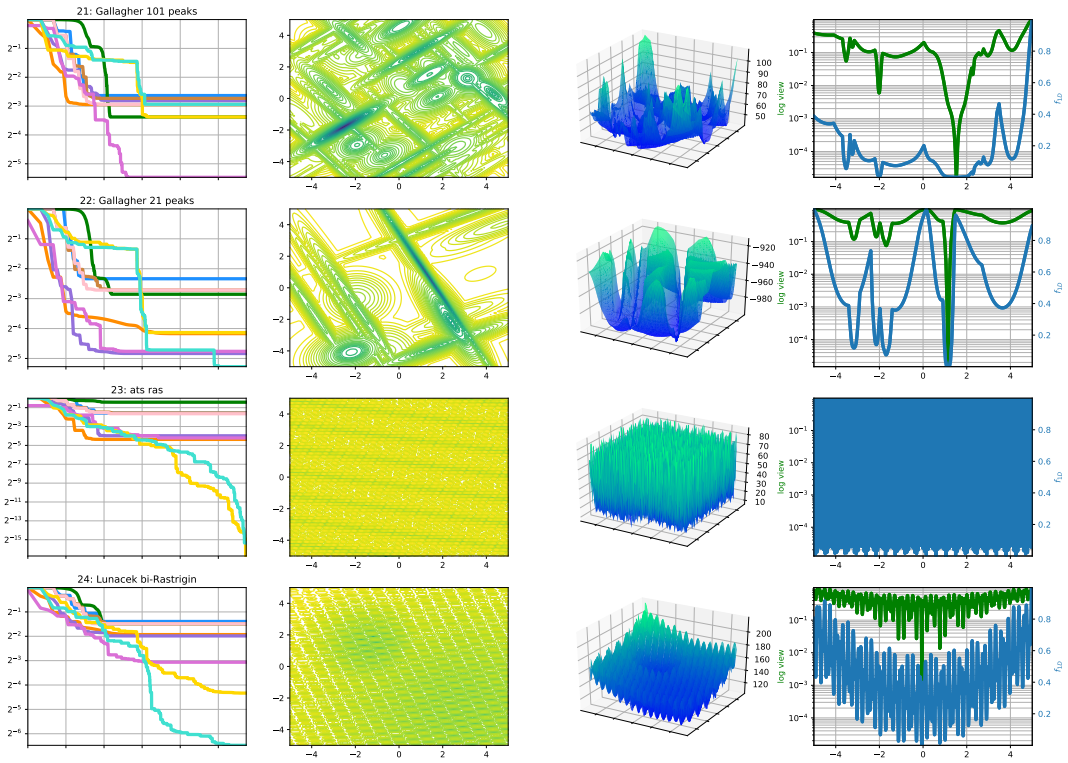

Figure 15. Visualization problems type 21-24 of 2D problems. First column: The scaled distance $\overline{\Delta y}^t_{best}$. Second column: Counter plot. Third column: 3D plot. Forth column: equivalent 1D problem with log view.

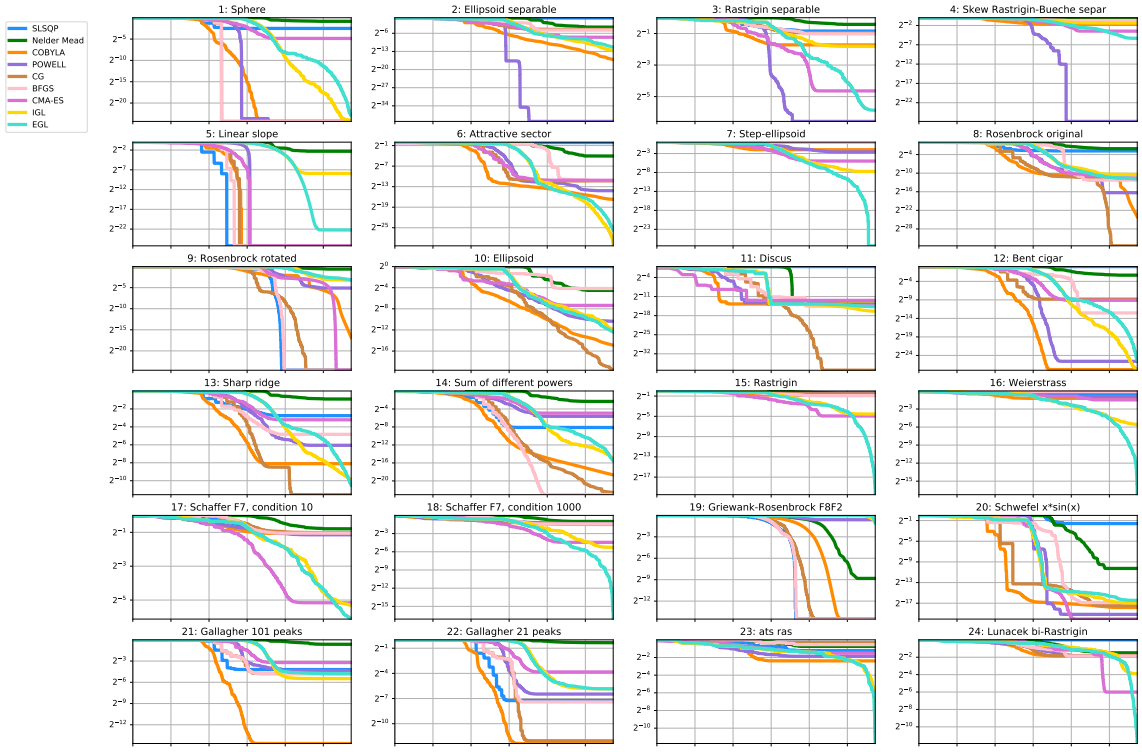

Figure 16. The scaled distance $\overline{\Delta y}^t_{best}$ per problem type on 40D

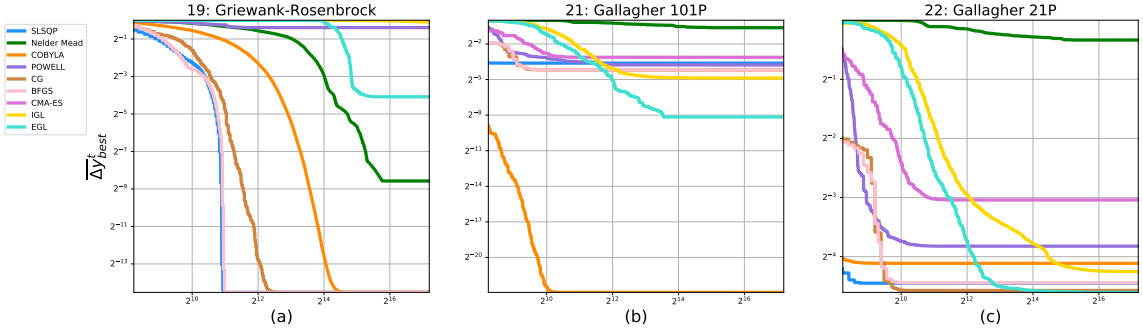

Figure 17. The scaled distance $\overline{\Delta y}^t_{best}$ with different ε on 40D. (a) problem type 19, (b) problem type 21, (c) problem type 22

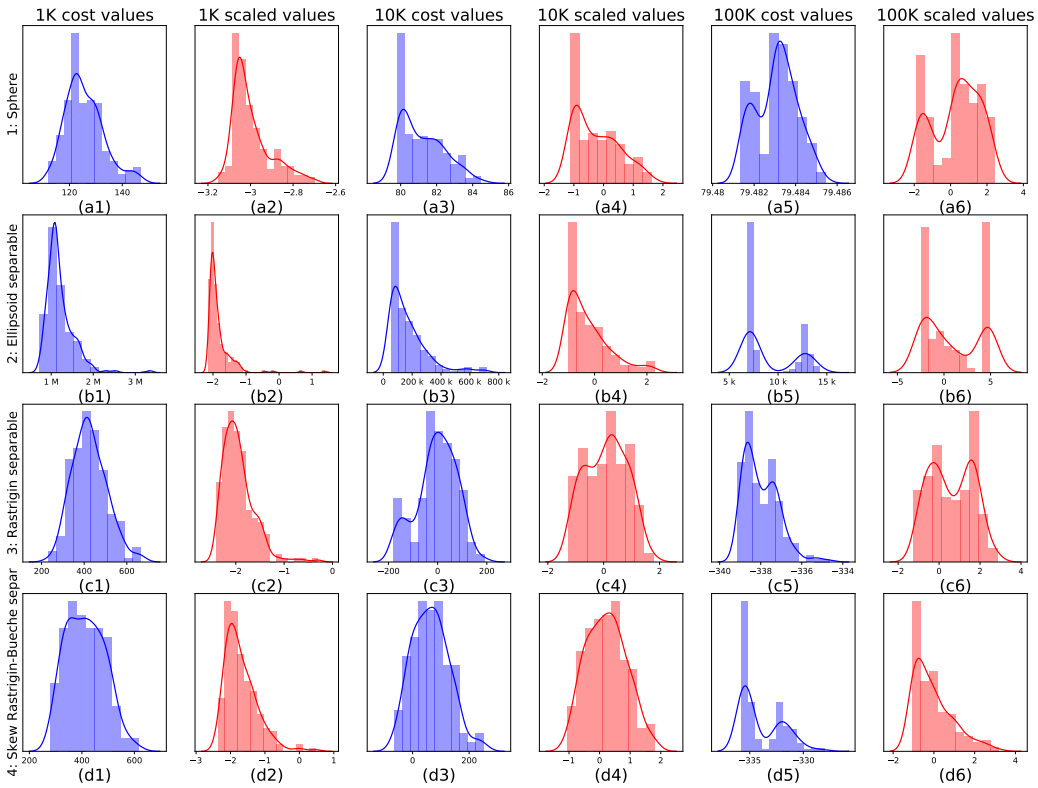

Figure 18. Histogram plot of a snapshot of 200 samples from the RB around different times ($t = 1K$, $t = 10K$, $t = 100K$) with and without OM for problems 1-4

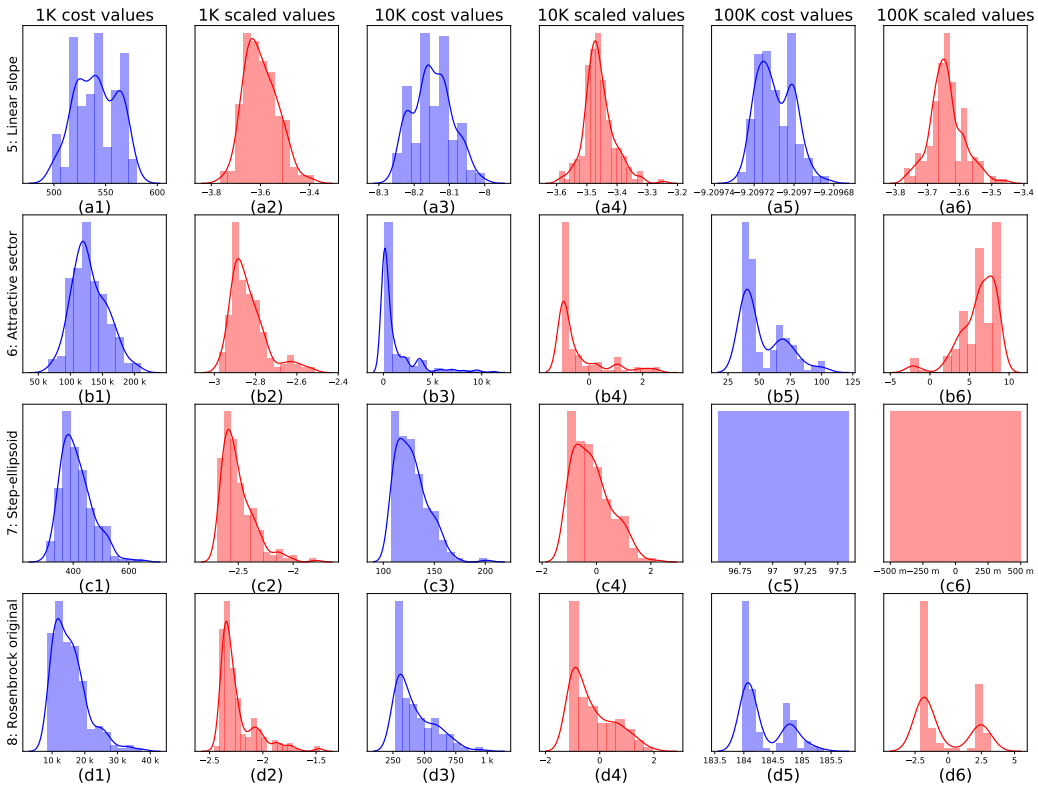

Figure 19. Histogram plot of a snapshot of 200 samples from the RB around different times ($t = 1K$, $t = 10K$, $t = 100K$) with and without OM for problems 5-8

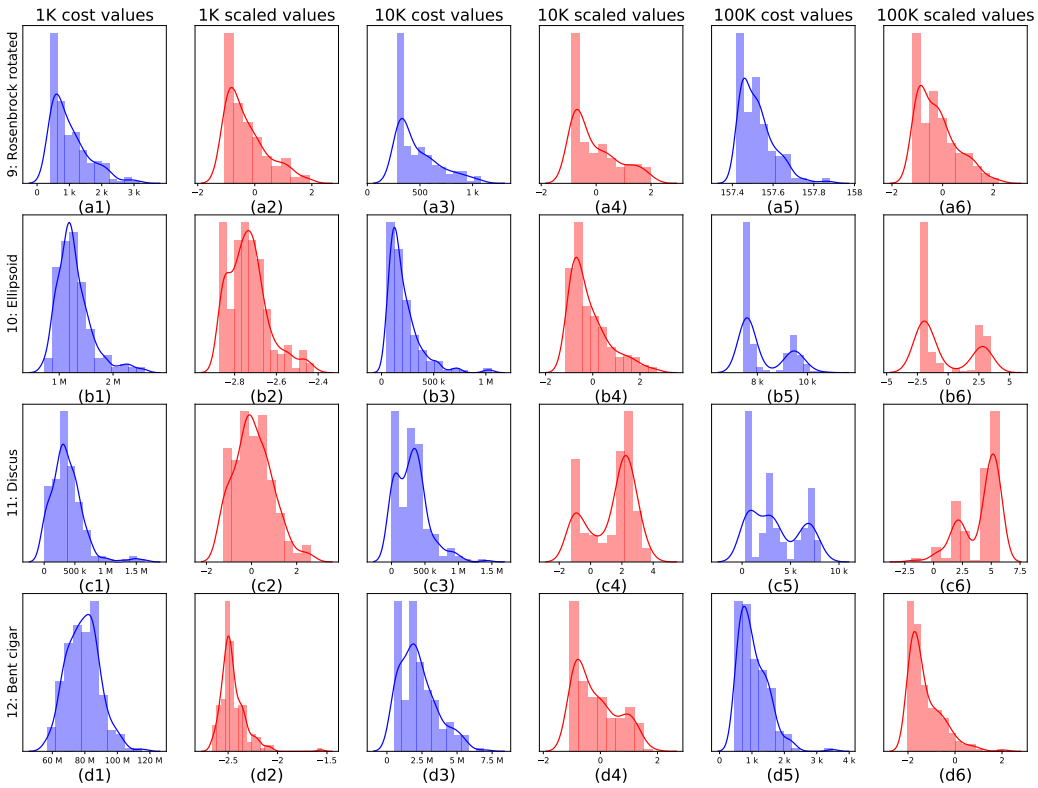

Figure 20. Histogram plot of a snapshot of 200 samples from the RB around different times ($t = 1K$, $t = 10K$, $t = 100K$) with and without OM for problems 9-12

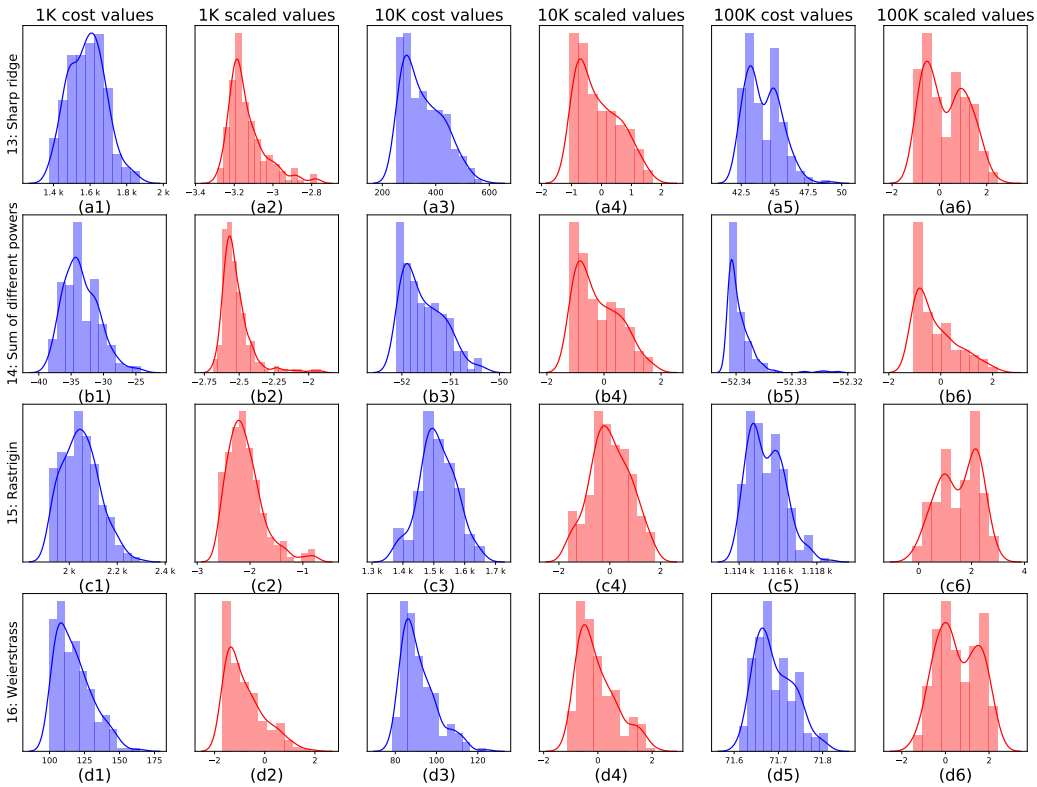

Figure 21. Histogram plot of a snapshot of 200 samples from the RB around different times ($t = 1K$, $t = 10K$, $t = 100K$) with and without OM for problems 13-16

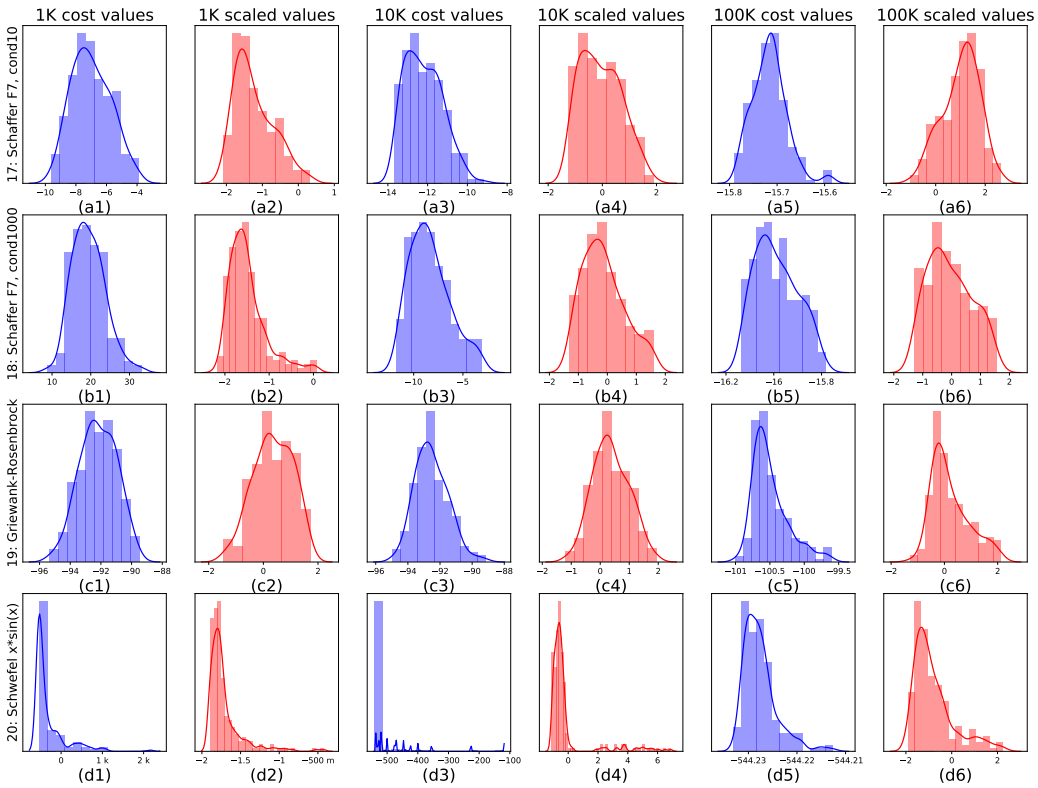

Figure 22. Histogram plot of a snapshot of 200 samples from the RB around different times ($t = 1K$, $t = 10K$, $t = 100K$) with and without OM for problems 17-20

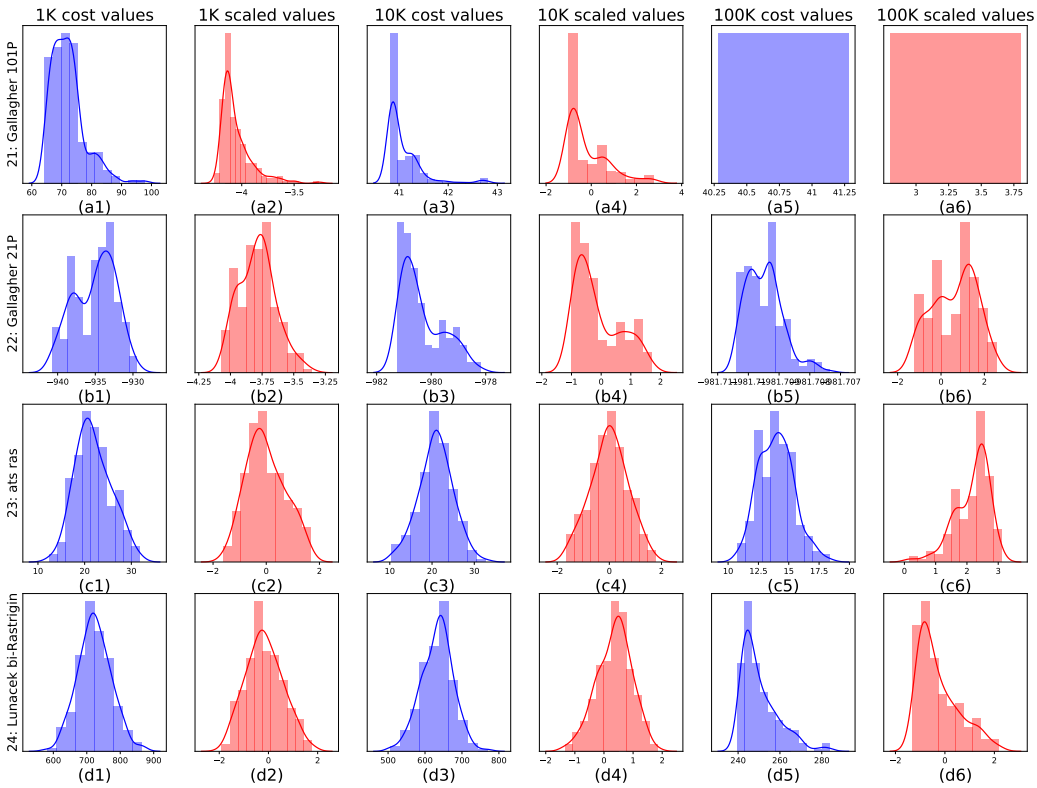

Figure 23. Histogram plot of a snapshot of 200 samples from the RB around different times ($t = 1K$, $t = 10K$, $t = 100K$) with and without OM for problems 21-24

G. Supplementary details: latent space search

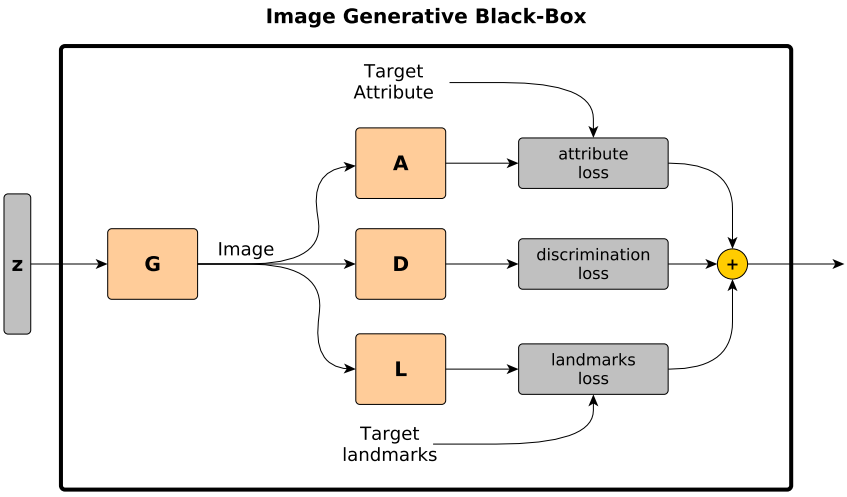

Figure 24. The image-generative BBO task

In this experiment, the task is to utilize a pre-trained Black-Box face image generator and generate a realistic face image with target face attributes and face landmark points. Formally, we have 4 Black-Box networks that constitute our BBO problem:

1. Generator $G : z \to x$, where $z \sim \mathcal{N}(0, \mathbf{I}_n)$ and x is an RGB image with $H = 218$, $W = 178$.
2. Discriminator $D : x \to \mathbb{R}$ s.t. positive $D(x)$ indicate poor fake images while negative $D(x)$ indicates real or a good fake image.
3. Attribute Classifier $A : x \to \mathbb{R}^{40}$ where each element in $A(x)$ is the probability of a single attribute (out of 40 different attributes).
4. Landmark points Estimator $L : x \to \mathbb{R}^{68}$ predicts the location of 68 different landmark points.

In addition, every BBO problem is characterized by two external parameters

1. Target attributes a a vector of 40 Booleans.
2. Target landmark points $l \in \mathbb{R}^{68}$ a vector of 68 landmark points locations.

The overall cost function is defined as

$$f_{al}(z) = \lambda_a \mathcal{L}_a(G(z)) + \lambda_l \mathcal{L}_l(G(z)) + \lambda_g \tanh(D(G(z)))$$

Where:

1. $\mathcal{L}_a$ is the Cross-Entropy loss between the generated face attributes as measured by the classifier and the desired set of attributes a.
2. $\mathcal{L}_l$ is the MSE between the generated landmark points and the desired set of landmarks l.
3. $D(G(z))$ is the discriminator output, positive for low-quality images and negative for high-quality images.

The objective is to find z^* that minimizes f_{al}. A graphical description of the BBO problem is given in Fig. 24. We used a constant starting point z_0, sampled from $\mathcal{N}(0, \mathbf{I}_n)$ for all our runs. To generate different problems, we sampled a target

Table 2. Latent-Space Search Hyperparameters

Parameter	Value	Description
n	512	Latent space dimension
λ_a	1	Attributes score weight
λ_d	2	Discriminator score weight
λ_l	100	Landmarks score weight
m	32	Exploration points
`batch`	1024	Minibatch of EGL/IGL training
L	64	Number of exploration steps that constitute the replay buffer for EGL/IGL The replay memory size is: $RB = L \times m$
C	10^4	Budget
ϕ	$\frac{2}{3}\pi$	Cone exploration angle
α	0.02	Optimization steps' size
`g_lr`	10^{-3}	g_θ learning rate
γ_α	0.9	Trust region squeezing factor
γ_ε	0.95	ε squeezing factor

image x_T from the CelebA dataset. We used its attributes as the target attribute and used its landmark points estimation $l = L(x_T)$ as the target landmarks. Note that the target image x_T was never revealed to the EGL optimizer, only its attributes and landmark points.

We applied the same EGL algorithm as in the COCO experiment with the Spline Embedding network architecture and with two notable changes: (1) a set of slightly modified hyperparameters (See Table 2); and (2) a modified exploration domain V_ε. The main reason for these adjustments was the high computational cost (comparing to the COCO experiment) of each different z vector. This led us to squeeze the budget C to only 10^4 evaluations and the number of exploration points to only $m = 32$. For such a low number of exploration points around each candidate (in a $n = 512$ dimension space), we designed an exploration domain V_ε, termed *cone-explore* which we found out to be more efficient than the uniform exploration inside an n-ball with ε radius that was executed in the COCO experiment. A description of the cone-explore method is given in Sec. H.

Additional results of EGL and IGL are given in Fig. 27 and Fig. 28. We also evaluated two classical algorithms: GC and CMA-ES, both provided unsatisfying results (see Fig. 26). We observed that CG converged to local minima around the initial point z_0 while CMA-ES converged to points that have close landmark point but poor discrimination score. We did not investigate into this phenomena but we postulate that it is the result of different landscapes statistics of the two factors in the cost function, i.e. $\mathcal{L}_l(G(z))$ and $\tanh(D(G(z)))$ which fool the CMA-ES algorithm.

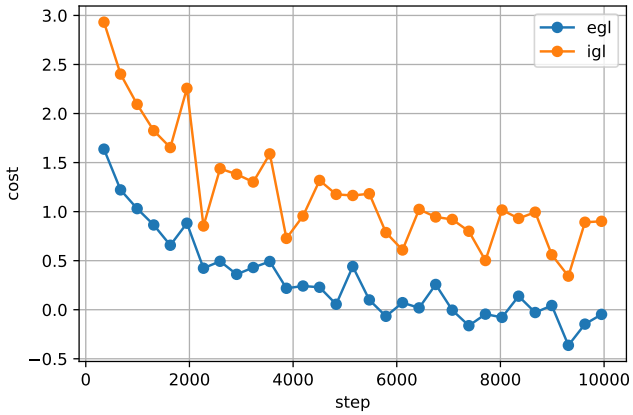

Figure 25. Average cost during training.

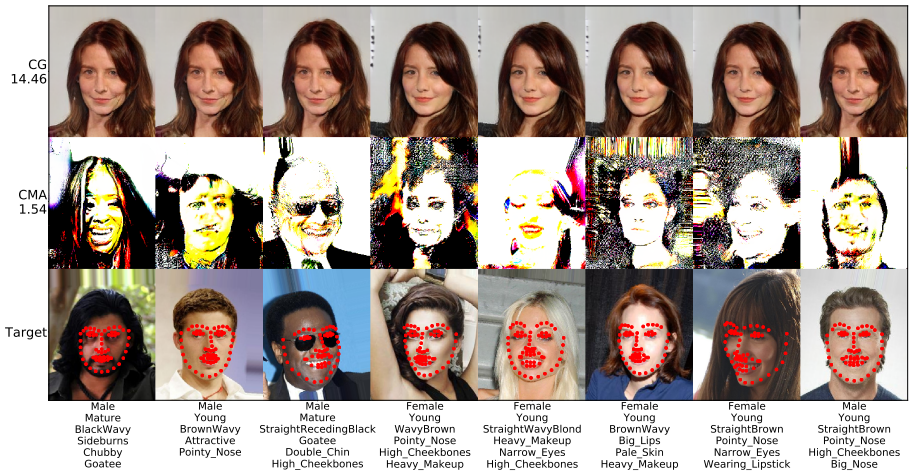

Figure 26. Baselines results: CG and CMA

Figure 27. Searching latent space of generative models with EGL and IGL

Figure 28. Searching latent space of generative models with EGL and IGL

H. Gradient Guided Exploration

In high-dimensional problems and low budgets, sampling $n+1$ exploration points for each new candidate x_k may consume the entire budget too fast without being able to take enough optimization steps. In practice, EGL works even with $m \ll n+1$ exploration points, however, we observed that one can improve the efficiency when $m \ll n+1$ by sampling the exploration points non uniformly around the candidate x_k. Specifically, using the previous estimation of the gradient to determine the search direction. Hence, we term this approach as *gradient guided exploration*.

In the COCO experiment (Sec. 5) we sampled the exploration points uniformly around each candidate. In other words, our V_ε domain was an n-ball with ε radius, we term this method as *ball-explore*. Ball-explore does not make any assumptions on the gradient direction at x_k and does not use any a-priori information. However, for continuous gradients, we do have a-priori information on the gradient direction as we have our previous estimator $g_{\theta_{k-1}}$. Since x_k is relatively close to x_{k-1}, the learned model $g_{\theta_{k-1}}$ can be used as a first-order approximation for the gradient in x_k, i.e. $g_{\theta_{k-1}}(x_k)$.

If $g_{\theta_{k-1}}(x_k)$ is a good approximation for $g_\varepsilon(x_k)$, then sampling points in a perpendicular direction to the mean-gradient, i.e. x s.t. $(x-x_k)\cdot g_{\theta_{k-1}}(x_k) = 0$, adds little information since $f(x)-f(x_k) \approx 0$ so the loss $|(x-x_k)\cdot g_{\theta_{k-1}}(x_k)-f(x)+f(x_k)|^2$ is very small. Therefore, we experimented with sampling points inside the intersection of a n-ball $B_\varepsilon(x_k)$ and a cone with apex at x_k, direction $-g_{\theta_{k-1}}(x_k)$ and some hyperparameter cone-angle of ϕ. The distance vector $x-x_k$ of a point inside such a cone has high cosine similarity with the mean-gradient and we postulate that this should improve the efficiency of the learning process. We term this alternative exploration method as *cone-explore* and denote the cone domain as $C_\varepsilon^\phi(x, g_{\theta_{k-1}}(x_k))$.

For high dimensions, cone-explore significantly reduces the exploration volume. A simple upper bound for the cone-to-ball volume ratio show that it decays exponentially in n

$$\frac{|C_\varepsilon^\phi|}{|B_\varepsilon|} \leq \frac{\sqrt{\pi}\Gamma(\frac{n+1}{2})}{n\Gamma(\frac{n}{2}+1)}(\sin\phi)^{n-1} \tag{65}$$

Unfortunately, cone-explore is not suitable for non-continuous gradients or too large optimization steps. To take into account gradient discontinuities, we suggest to sample half of the points inside the cone and half inside an n-ball.

In Fig. 29 we present an ablation test in the 784D COCO problem set of cone-explore and $\frac{1}{2}$-cone-$\frac{1}{2}$-ball explore with respect to the standard ball-explore. In this experiment, we used only $m = 32$ exploration points around each candidate. The results show that sampling half of the exploration points inside a cone improved the results by 18%. We found out that the strategy also improves the results in the latent space search experiment, yet we did not conduct a full ablation test.

On the downside, we found out that if $m \approx n$ then cone-explore hurts the performance. We hypothesize that near local minima, where the exact direction of the gradient is important, the mean-gradient learned with cone-explore has lower accuracy and therefore, ball-explore with sufficient sampling points converges to better solutions.

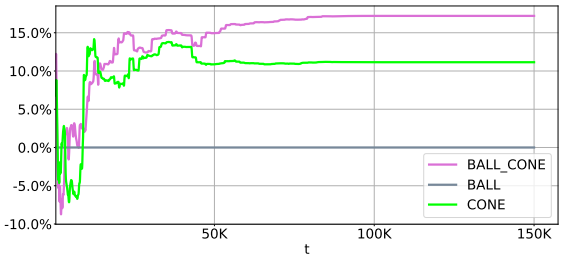

Figure 29. Ablation test on the 784D COCO problem set: Cone-explore vs Ball-explore