LEARNING TO GUIDE RANDOM SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

We are interested in the optimization of a high-dimensional function when only function evaluations are possible. Although this derivative-free setting arises in many applications, existing methods suffer from high sample complexity since their sample complexity depend on problem dimensionality, in contrast to the dimensionality-independent rates of first-order methods. The recent success of deep learning methods suggests that many data modalities lie on low-dimensional manifolds that can be represented by deep nonlinear models. Based on this observation, we consider derivative-free optimization of functions defined on low-dimensional manifolds. We develop an online learning approach that learns this manifold while performing the optimization. In other words, we jointly learn the manifold and optimize the function. Our analysis suggests that the proposed method significantly reduces sample complexity. We empirically evaluate the presented method on continuous optimization benchmarks and high-dimensional continuous control problems. Our method achieves significantly lower sample complexity than Augmented Random Search and other derivative-free optimization algorithms.

1 INTRODUCTION

A typical approach to machine learning problems is to define an objective function and optimize it over a dataset. First-order optimization methods are widely used for this purpose since they can scale to high-dimensional problems and their convergence rates are independent of the problem dimensionality in most cases. However, gradients are not available in many essential tasks such as control, black-box optimization, or interactive learning with humans in the loop. Derivative-free optimization (DFO) can be used to tackle such problems. However, the sample complexity of DFO methods scales poorly with the problem dimensionality. Thus a major open problem is to design DFO methods that solve high-dimensional problems with low sample complexity.

The success of deep learning methods suggests that high-dimensional real-world data can be represented in low-dimensional spaces via learned non-linear features. In other words, while the problems of interest are high-dimensional, the data typically lies on low-dimensional manifolds. If we could perform the optimization directly in the manifold instead of the full space, intuition suggests that we could reduce the sample complexity of DFO methods since their convergence rates are generally a function of the problem dimensionality (see analysis by Nesterov & Spokoiny (2017); Dvurechensky et al. (2018)). In this paper, we focus on high-dimensional data distributions that are drawn from low-dimensional manifolds. Since the manifold is typically not known prior to the optimization, we pose the question: *Can we develop an adaptive derivative-free optimization algorithm that learns the manifold in an online fashion while performing the optimization?*

There exist DFO methods that aim to identify a low-dimensional search space (Maheswaranathan et al., 2018; Choromanski et al., 2019). However, they are limited to linear subspaces. In contrast, we propose to use highly expressive, nonlinear models (specifically, neural networks) to represent the manifold. Our approach not only increases expressiveness but also enables utilization of domain knowledge on the geometry of the problem. For example, if the function of interest is known to be translation invariant, convolutional networks could be deployed to represent the underlying manifold structure. On the other hand, the high expressive power and flexibility brings challenges. Our approach requires solving for the parameters of the nonlinear manifold at each iteration of the optimization. To address this, we develop an efficient online method that learns the underlying manifold while the function is being optimized.

We specifically consider random search methods and extend them to the nonlinear manifold learning setting. Random search methods choose a set of random directions and perform perturbations to the current iterate in these directions. Differences of the function values computed at perturbed points are used to compute an estimator for the gradient of the function. We first extend this to a random search over a known manifold and show that sampling directions in the tangent space of the manifold provides a similar estimate. We further propose an online learning method that estimates this manifold while jointly performing the optimization. We theoretically analyze the sample complexity and show that our method reduces sample complexity. We conduct extensive experiments on learning continuous control, continuous optimization benchmarks, and gradient-free optimization of an airfoil. The results indicate that our method significantly outperforms prior work.

2 PRELIMINARIES

We are interested in high-dimensional stochastic optimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \mathbb{E}_\xi[F(\mathbf{x},\xi)], \tag{1}$$

where $\mathbf{x}$ is the optimization variable and $f:\mathbb{R}^d\to\mathbb{R}$ is the function of interest, which is defined as expectation over a noise ξ. We assume that the stochastic function is bounded ($|F(\mathbf{x},\xi)|\le\Omega$), $L-$Lipschitz and $\mu-$smooth [1] with respect to $\mathbf{x}$ for all ξ, and has uniformly bounded variance ($\mathbb{E}_\xi[(F(\mathbf{x},\xi)-f(\mathbf{x}))^2]\le V_F$). In DFO, we have no access to the gradients. Instead, we only have zeroth-order access by evaluating the function F (i.e. sampling $F(\mathbf{x},\xi)$ for the input $\mathbf{x}$).

We are specifically interested in random search methods in which an estimate of the gradient is computed using function evaluations at points randomly sampled around the current iterate. Before we formalize this, we introduce some definitions. Denote the d-dimensional unit sphere and unit ball by $\mathbb{S}^{d-1}$ and $\mathbb{B}^d$, respectively. Furthermore, we define a *smoothed* function following Flaxman et al. (2005). For a function $f:\mathbb{R}^d\to\mathbb{R}$, its δ-smoothed version is $\hat{f}(\mathbf{x})=\mathbb{E}_{\mathbf{v}\sim\mathbb{B}^d}[f(\mathbf{x}+\delta\mathbf{v})]$. The main workhorse of random search is the following result by Flaxman et al. (2005). Let $\mathbf{s}$ be a random vector sampled from the uniform distribution over $\mathbb{S}^{d-1}$. Then $f(\mathbf{x}+\delta\mathbf{s})\mathbf{s}$ is an unbiased estimate of the gradient of the smoothed function:

$$\mathbb{E}_{\xi,\mathbf{s}\in\mathbb{S}^{d-1}}[F(\mathbf{x}+\delta\mathbf{s},\xi)\mathbf{s}] = \frac{\delta}{d}\nabla_{\mathbf{x}}\hat{f}(\mathbf{x}). \tag{2}$$

We use antithetic samples since this is known to decrease variance (Kroese et al., 2013) and define the final gradient estimator as $y(\mathbf{x},\mathbf{s}) = (F(\mathbf{x}+\delta\mathbf{s},\xi)-F(\mathbf{x}-\delta\mathbf{s},\xi))\,\mathbf{s}$. Moreover, extending (2) to the antithetic case, $\mathbb{E}_{\xi,\mathbf{s}\in\mathbb{S}^{d-1}}[y(\mathbf{x},\mathbf{s})] = \frac{2\delta}{d}\nabla_{\mathbf{x}}\hat{f}(\mathbf{x})$.

A simple way to optimize the function of interest is to use the gradient estimate in stochastic gradient descent (SGD), as summarized in Algorithm 1. This method has been analyzed in various forms and its convergence is characterized well for nonconvex smooth functions. We restate the convergence rate and defer the constants and proof to Appendix C.2.

Proposition 1 (Flaxman et al., 2005; Vemula et al., 2019)**.** *Let $f(\mathbf{x})$ be differentiable, L-Lipschitz, and μ-smooth. Consider running random search (Algorithm 1) for T steps. Let $k=1$ for simplicity. Then*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|_2^2 \le \mathcal{O}\left(T^{-\frac{1}{2}}d + T^{-\frac{1}{3}}d^{\frac{2}{3}}\right).$$

3 ONLINE LEARNING TO GUIDE RANDOM SEARCH

Proposition 1 implies that the sample complexity of random search scales linearly with the dimensionality. This dependency is problematic when the function of interest is high-dimensional. We argue that for many practical problems, the function of interest lies on a low-dimensional nonlinear manifold. This structural assumption will allow us to significantly reduce the sample complexity of random search, without knowing the manifold a priori.

[1] $L-$Lipschitz and μ-smooth: $|f(\mathbf{x})-f(\mathbf{y})|\le L\|\mathbf{x}-\mathbf{y}\|_2$ and $\|\nabla_{\mathbf{x}}f(\mathbf{x})-\nabla_{\mathbf{x}}f(\mathbf{y})\|_2\le\mu\|\mathbf{x}-\mathbf{y}\|_2 \quad \forall_{\mathbf{x},\mathbf{y}}$

Algorithm 1 Random Search

for $t = 1$ **to** T **do**
  $\mathbf{g}^t, _ = \textsc{GradEst}(\mathbf{x}^t, \delta)$
  $\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha \mathbf{g}^t$
end for

procedure $\textsc{GradEst}(\mathbf{x}, \delta)$
  Sample: $\mathbf{s}_1, \ldots, \mathbf{s}_k \sim \mathbb{S}^{d-1}$
  Query: $y_i = f(\mathbf{x} + \delta \mathbf{s}_i) - f(\mathbf{x} - \delta \mathbf{s}_i)$
  Estimator: $\mathbf{g} = \frac{d}{2\delta} \sum_{i=1}^{k} y_i \mathbf{s}_i$
  return $\mathbf{g}, \{\mathbf{s}_i\}_{i \in [k]}$
end procedure

Algorithm 2 Manifold Random Search

for $t = 1$ **to** T **do**
  $\mathbf{g}^t, _ = \textsc{ManifoldGradEst}(\mathbf{x}^t, \mathbf{J}(\mathbf{x}^t; \theta^\star))$
  $\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha \mathbf{g}^t$
end for

procedure $\textsc{ManifoldGradEst}(\mathbf{x}, \theta, \delta)$
  Normalize: $\mathbf{J}_q$ = GramSchmidt($\mathbf{J}$)
  Sample: $\mathbf{s}_1, \ldots, \mathbf{s}_k \sim \mathbb{S}^{n-1}$
  Query: $y_i = f(\mathbf{x} + \delta \mathbf{J}_q \mathbf{s}_i) - f(\mathbf{x} - \delta \mathbf{J}_q \mathbf{s}_i)$
  Estimator: $\mathbf{g} = \frac{n}{2\delta} \sum_{i=1}^{k} y_i \mathbf{J}_q \mathbf{s}_i$
  return $\mathbf{g}, \{\mathbf{J}_q \mathbf{s}_i\}_{i \in [k]}$
end procedure

Assume that the function of interest is defined on an n-dimensional manifold ($n \ll d$) and this manifold can be defined via a nonlinear parametric family (e.g. a neural network). Formally, we are interested in derivative-free optimization of functions with the following properties:

- **Smoothness:** $F(\cdot, \xi) : \mathbb{R}^d \to \mathbb{R}$ is μ-smooth and L-Lipschitz for all ξ.
- **Manifold:** $F(\cdot, \xi)$ is defined on an n-dimensional manifold $\mathcal{M}$ for all ξ.
- **Representability:** The manifold $\mathcal{M}$ and the function of interest can be represented using parametrized function classes $r(\cdot; \theta)$ and $g(\cdot; \psi)$. Formally, given ξ, there exist $\theta^\star, \psi^\star$ such that $F(\mathbf{x}, \xi) = g(r(\mathbf{x}; \theta^\star); \psi^\star) \quad \forall \mathbf{x} \in \mathbb{R}^d$.

We will first consider an idealized setting where the manifold is already known (i.e. we know $\theta^\star$). Then we will extend the developed method to the practical setting where the manifold is not known in advance and must be estimated with no prior knowledge as the optimization progresses.

3.1 Warm-up: Random Search over a Known Manifold

If the manifold is known a priori, we can perform random search directly over the manifold instead of the full space. Consider the chain rule applied to $g(r(\mathbf{x}; \theta); \psi)$ as $\nabla_{\mathbf{x}} f(\mathbf{x}) = \mathbf{J}(\mathbf{x}; \theta^\star) \nabla_{\mathbf{r}} g(\mathbf{r}; \psi)$, where $\mathbf{J}(\mathbf{x}; \theta^\star) = {\partial r(\mathbf{x}; \theta^\star)}/{\partial \mathbf{x}}$ and $\mathbf{r} = r(\mathbf{x}, \theta^\star)$. The gradient of the function of interest lies in the column space of the Jacobian of the parametric family. In light of this result, we can perform random search in the column space of the Jacobian, which is lower-dimensional than the full space.

For numerical stability, we will first orthonormalize the Jacobian using the Gram-Schmidt procedure, and perform the search in the column space of this orthonormal matrix since it spans the same space. We denote the orthonormalized version of $\mathbf{J}(\mathbf{x}; \theta^\star)$ by $\mathbf{J}_q(\mathbf{x}; \theta^\star)$.

In order to perform random search, we sample an n-dimensional vector uniformly $\tilde{\mathbf{s}} \sim \mathbb{S}^{n-1}$, and lift it to the input space via $\mathbf{J}_q(\mathbf{x}; \theta^\star)\tilde{\mathbf{s}}$. As a consequence of the manifold Stoke's theorem, using the lifted vector as a random direction gives an unbiased estimate of gradient of the manifold smoothed function as

$$\mathbb{E}_{\xi, \mathbf{s} \in \mathbb{S}^{n-1}} [y(\mathbf{x}, \mathbf{J}_q(\mathbf{x}; \theta^\star)\tilde{\mathbf{s}})] = \frac{2\delta}{n} \nabla_{\mathbf{x}} \tilde{f}_{\theta^\star}(\mathbf{x}), \tag{3}$$

where the manifold smoothed function is defined as $\tilde{f}_{\theta^\star}(\mathbf{x}) = \mathbb{E}_{\tilde{\mathbf{v}} \sim \mathbb{B}^n} [f(\mathbf{x} + \delta \mathbf{J}_q(\mathbf{x}; \theta^\star)\tilde{\mathbf{v}})]$. We show this result as Lemma 1 in Appendix C.1. We use the resulting gradient estimate in SGD. The following proposition summarizes the sample complexity of this method. The constants and the proof are given in Appendix C.2.

Proposition 2. *Let $f(\mathbf{x})$ be differentiable, L-Lipschitz, and μ-smooth. Consider running manifold random search (Algorithm 2) for T steps. Let $k = 1$ for simplicity. Then*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 \leq \mathcal{O}\left(T^{-\frac{1}{2}} n + T^{-\frac{1}{3}} n^{\frac{2}{3}}\right).$$

3.2 Joint Optimization and Manifold Learning

When $n \ll d$, the reduction in the sample complexity of random search (summarized in Proposition 2) is significant. However, the setting of Algorithm 2 and Proposition 2 is impractical since the

manifold is generally not known a priori. We thus propose to minimize the function and learn the manifold jointly. In other words, we start with an initial guess of the parameters and solve for them at each iteration using all function evaluations that have been performed so far.

Our major objective is to improve the sample efficiency of random search. Hence, minimizing the sample complexity with respect to manifold parameters is an intuitive way to approach the problem. We analyze the sample complexity of SGD using biased gradients in Section C.3.1 and show the following informal result. Consider running manifold random search with a sequence of manifold parameters $\theta^1, \psi^1, \ldots, \theta^T, \psi^T$ for T steps. Then the additional suboptimality caused by biased gradients, defined as $\textsc{SubOptimality} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\nabla_{\mathbf{x}} f(\mathbf{x})]$, is bounded as follows:

$$\textsc{SubOptimality}(\theta^t, \psi^t) \leq \textsc{SubOptimality}(\theta^\star, \psi^\star) + \frac{\Omega}{T}\sum_{t=1}^{T} \|\nabla_{\mathbf{x}} \tilde{f}_{\theta^\star}(\mathbf{x}^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t)\|_2, \quad (4)$$

where $\textsc{SubOptimality}(\theta^\star, \psi^\star)$ is the suboptimality of the oracle case (Algorithm 2). Our aim is to minimize the additional suboptimality with respect to θ^t, ψ^t. However, we do not have access to $\nabla_{\mathbf{x}} f(\mathbf{x})$ since we are in a derivative-free setting. Hence we cannot directly minimize (4).

At each iteration, we observe $y(\mathbf{x}^t, \mathbf{s}^t)$. Moreover, $y(\mathbf{x}^t, \mathbf{s}^t) = 2\delta \mathbf{s}^{t\intercal} \nabla_{\mathbf{x}} \tilde{F}(\mathbf{x}^t, \xi) + \mathcal{O}(\delta^2)$ following the smoothness . Since we observe the projection of the gradient to the chosen directions, we minimize the projection of (4) to these directions. Formally, we define our one-step loss as

$$\mathcal{L}(\mathbf{x}^t, \mathbf{s}^t, \theta^t, \psi^t) = \left(\frac{y(\mathbf{x}^t, \mathbf{s}^t)}{2\delta} - \mathbf{s}^{t\intercal} \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t) \right)^2. \quad (5)$$

We use the Follow the Regularized Leader (FTRL) algorithm (Hazan, 2016; Shalev-Shwartz, 2012) to minimize the aforementioned loss function and learn the manifold parameters as

$$\theta^{t+1}, \psi^{t+1} = \arg\min_{\theta, \psi} \sum_{i=1}^{t} \mathcal{L}(\mathbf{x}^i, \mathbf{s}^i, \theta, \psi) + \lambda \mathcal{R}(\theta, \psi), \quad (6)$$

where the regularizer ($\mathcal{R}(\theta, \psi) = \|\nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta); \psi)\|_2$) is the temporal smoothness term which enforces that gradient estimates do not change significantly.

Algorithm 3 summarizes our algorithm. We add exploration by sampling a mix of directions from the manifold and the full space. In each iteration, we sample directions and produce two gradient estimates $\mathbf{g}_m, \mathbf{g}_e$ using the samples from the tangent space and the full space, respectively. We mix them to obtain the final estimate $\mathbf{g} = (1-\beta)\mathbf{g}_m + \beta \mathbf{g}_e$. We discuss the implementation details of the FTRL step in Section 4. In our theoretical analysis, we assume that (6) can be solved optimally. Although this is a strong assumption, experimental results suggest that neural networks can easily fit any training data (Zhang et al., 2017). Our experiments also support this observation.

Theorem 1 states our main result concerning the sample complexity of our method. As expected, the sample complexity of our method includes both the input dimensionality d and the manifold dimensionality n. On the other hand, the sample complexity only depends on $n\sqrt{d}$ rather than d. Thus our method significantly decreases sample complexity when $n \ll d$.

Theorem 1. *Let $f(\mathbf{x})$ be bounded, L-Lipschitz and μ-smooth. Consider running learned manifold random search (Algorithm 3) for T steps. Let $k_e = 1, k_m = 1$ for simplicity. Then*

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 \leq \mathcal{O}\left(d^{\frac{1}{2}} T^{-1} + (d^{\frac{1}{2}} + n + n d^{\frac{1}{2}}) T^{-\frac{1}{2}} + (n^{\frac{2}{3}} + d^{\frac{1}{2}} n^{\frac{2}{3}}) T^{-\frac{1}{3}} \right).$$

Proof sketch. We provide a short proof sketch here and defer the detailed proof and constants to Appendix C.3. We start by analyzing SGD with bias. The additional suboptimality of using θ^t, ψ^t instead of $\theta^\star, \psi^\star$ can be bounded by (4).

The empirical loss we minimize is the projection of (4) to randomly chosen directions. Next, we show that the expectation of the empirical loss is (4) when the directions are chosen uniformly at random from the unit sphere:

$$\mathbb{E}_{\mathbf{s}^t \in \mathbb{S}^{d-1}} \left[\mathcal{L}(\mathbf{x}^t, \mathbf{s}^t, \theta^t, \psi^t) \right] = \frac{1}{dT} \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} \tilde{f}_{\theta^\star}(\mathbf{x}^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t)\|_2. \quad (7)$$

A crucial argument in our analysis is the concentration of the empirical loss around its expectation. In order to study this concentration, we use Freedman's inequality (Freedman, 1975), similar to the seminal work studying the generalization of online learning (Kakade & Tewari, 2009). This analysis bounds the difference $\left|\mathbb{E}_{\mathbf{s}^t \in \mathbb{S}^{d-1}}\left[\sum_{t=1}^{T} \mathcal{L}^t\right] - \sum_{t=1}^{T} \mathcal{L}^t\right|$, where $\mathcal{L}^t = \mathcal{L}(\mathbf{x}^t, \mathbf{s}^t, \theta^t, \psi^t)$.

Next, we use the FTL-BTL Lemma (Kalai & Vempala, 2005) to analyze the empirical loss $\sum_{t=1}^{T} \mathcal{L}^t$. We bound the empirical loss in terms of the distances between the iterates $\sum_{t=1}^{T} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2$. Such a bound would not be useful in an adversarial setting since adversary chooses $\mathbf{x}^t$, but we choose appropriate step sizes resulting in sufficiently small steps for convergence.

Our analysis of learning requires the directions to be sampled from a unit sphere in (7). On the other hand, our optimization method requires directions to be chosen from the tangent space of the manifold. We mix exploration (directions sampled from $\mathbb{S}^{d-1}$) and exploitation (directions sampled from the tangent space of the manifold) to solve this mismatch. We show that the mixing weight $\beta = 1/d$ results in both fast optimization and no-regret learning. Finally, we combine analysis of empirical loss, concentration, and analysis of SGD to obtain the statement of the theorem. □

4 Implementation Details and Limitations

Parametric family. We use multilayer perceptrons (4 to 6 layers depending on the problem) with ReLU nonlinearities to define g and r. We initialize our models with standard normal distributions. Thus, our method thus starts with pure random search at initialization and then moves to full manifold random search as online learning progresses (see Appendix B for more details).

Solving FTRL. Results on training deep networks suggest that local SGD-based methods perform well. Hence, we use SGD with momentum as a solver for FTRL in (6). We do not solve each learning problem from scratch but initialize with the previous solution. Since this process may be stymied by local optima, we fully solve (6) from scratch for every 100th iteration of the method.

Computational complexity. Our method increases the amount of computation since we need to learn a model while performing the optimization. However, in DFO, the major computational bottleneck is typically the function evaluation. When efficiently implemented on a GPU, total time spent on learning the manifold is negligible in comparison to function evaluations.

Parallelization. Random search is highly parallelizable since directions can be processed independently. Communication costs include i) sending the current iterate to workers, ii) sending directions to each corresponding worker, and iii) each worker sending the function values back. When the directions are chosen independently, they can be indicated to each worker via a single integer by first creating a shared noise table in preprocessing. For a d-dimensional problem with k random directions, these costs are d, k, and k, respectively. Thus the total communication cost becomes $d + 2k$. In our method, each worker also needs a copy of the Jacobian, resulting in a communication cost of $d + 2k + kd$. Hence our method increases communication cost from $d + 2k$ to $d + 2k + kd$.

Algorithm 3 Learned Manifold Random Search

for $t = 1$ **to** T **do**
  $\mathbf{g}_e^t, \mathbf{S}_e^t = \textsc{GradEst}(\mathbf{x}^t, \delta)$
  $\mathbf{g}_m^t, \mathbf{S}_m^t = \textsc{ManifoldGradEst}(\mathbf{x}^t, \mathbf{J}(\mathbf{x}^t; \theta^t))$
  $\mathbf{g}^t = \frac{\beta k_e}{k_e + k_m}\mathbf{g}_e^t + \frac{(1-\beta)k_m}{k_e + k_m}\mathbf{g}_m^t$
  $\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha \mathbf{g}^t$
  $\theta^{t+1}, \psi^{t+1} = \arg\min_{\theta,\psi} \sum_{i=1}^{t} \mathcal{L}(\mathbf{x}^i, \mathbf{S}_{e,m}^i, \theta, \psi) + \lambda \mathcal{R}(\theta, \psi)$
end for

See Algorithms 1 & 2 for definitions of GradEst and ManifoldGradEst.

5 Experiments

In order to empirically evaluate our method, we perform experiments on three sets of problems. i) We use the MuJoCo simulator (Todorov et al., 2012) to evaluate our method on high-dimensional control problems. ii) We use widely used continuous optimization benchmarks and select 46 single-objective unconstrained functions from the Pagmo problem suite (Biscani et al., 2019). iii) We use the XFoil simulator (Drela, 1989) to evaluate our method on gradient-free optimization of an airfoil.

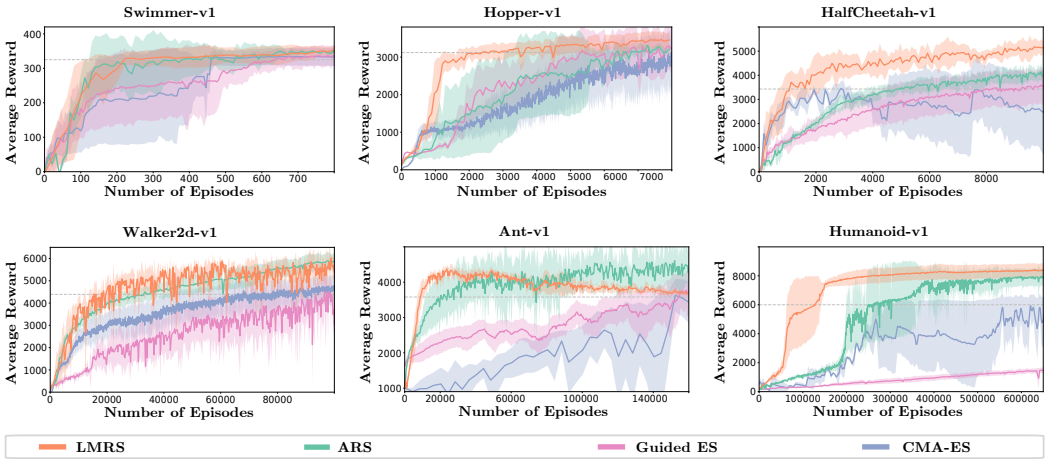

Figure 1: Average reward vs. number of episodes for MuJoCo locomotion tasks. We perform 5 random experiments with different seeds. Shaded areas represent 1 standard deviation. The grey horizontal line indicates the prescribed threshold at which the task is considered 'solved'.

We consider the following baselines. i) Augmented Random Search (ARS): Random search with all the augmentations from Mania et al. (2018). ii) Guided ES (Maheswaranathan et al., 2018): A recently proposed method to guide random search by adapting the covariance matrix. iii) CMA-ES (Hansen, 2016): Adaptive derivative-free optimization based on evolutionary search. iv) REMBO (Wang et al., 2016): A Bayesian optimization method which uses random embeddings in order to scale to high-dimensional problems. Although CMA-ES and REMBO are not based on random search, we include them for the sake of completeness. We give additional implementation details for baselines in Appendix B.

5.1 Learning Continuous Control

Following the setup of Mania et al. (2018), we use random search to learn control of highly articulated systems. The MuJoCo locomotion suite (Todorov et al., 2012) includes six problems with various difficulties, and we evaluate our method and the baselines on them. We use linear policies and include all the tricks (whitening the observation space and scaling the step size using the variance of the rewards) from Mania et al. (2018). We report average reward over five random experiments versus the number of episodes (i.e. number of function evaluations) in Figure 1. We also report the average number of episodes required to reach the prescribed reward threshold at which the task is considered 'solved' in Table 1. We include proximal policy optimization (PPO) (Schulman et al., 2017) from stable-baselines (Hill et al., 2018) for reference. Note that our results are slightly different from the numbers reported by Mania et al. (2018) as we use 5 random seeds instead of 3.

The results suggest that our method improves upon ARS in all environments. Our method also outperforms all other baselines. The improvement is significant for high-dimensional problems like Humanoid. Our method is at least twice as efficient as ARS in all environments except Swimmer, which is the only low-dimensional problem in the suite. Interestingly, Guided-ES fails to solve the Humanoid task, which we think is due to biased gradient estimation. Moreover, CMA-ES performs

								Self Baselines	
Task	Threshold	LMRS	ARS	CMA-ES	GuidedES	PPO	REMBO	No learning	Offline l.
Swimmer	325	**222**	381	460	640	790	520	320	325
Hopper	3120	**2408**	6108	14640	5288	10397	(0/5)	5561 (4/5)	11616
HalfCheetah	3430	**1128**	4284	2888	8004	6820	(0/5)	6528	3416
Walker	4390	**15525**	31240	96525	62544	102440	(0/5)	295493 (3/5)	118640
Ant	3580	**9240**	20440	154440	152400	52553	(0/5)	24520(4/5)	63720(2/5)
Humanoid	6000	**108133**	220733	1923030	(0/5)	200430	(0/5)	267260(2/5)	451260(2/5)

Table 1: Number of episodes required to reach the prescribed threshold on each MuJoCo locomotion task for our method, baselines, and ablations. Lower is better. We average over five random seeds. We denote the number of successful trials as (success/trial) and average over successful trials only. If the number of successful trials is not noted, the method solved the task for all random seeds.

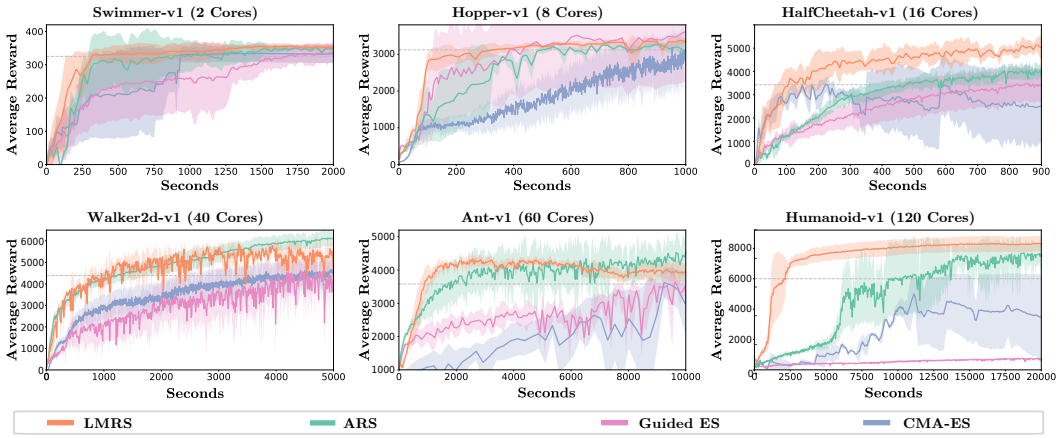

Figure 2: Average reward vs. wall-clock time for MuJoCo locomotion tasks. We perform 5 random experiments with different seeds. Shaded areas represent 1 standard deviation. The grey horizontal line indicates the prescribed threshold at which the task is considered 'solved'. All numbers are recorded using Intel Xeon E7-8890 v3 processor (288 cores) and Nvidia GeForce RTX 2080 Ti GPU. We also list the number of cores we use for each experiment.

similarly to ARS. These results suggest that a challenging task like Humanoid is out of reach for heuristics like local adaptation of the covariance matrix due to high stochasticity and non-convexity. Moreover, REMBO only solves the Swimmer task and fails to solve others. We believe this is due to the fact that these continuous control problems have no global structure and are highly non-smooth. The number of possible sets of contacts with the environment is combinatorial in the number of joints, and each contact configuration results in a distinct reward surface. This contradicts the global structure assumption in Bayesian optimization.

Wall-clock time analysis. Our method performs an additional computation as we learn the underlying manifold. In order to quantify the effect of the additional computation, we perform wall-clock time analysis by plotting average reward vs wall-clock time in Figure 2. Our method outperforms all baselines with similar margins. Moreover, trends and shapes of the curves do not change significantly. This is rather expected as the computation of all methods are rather negligible when compared with the simulation time of MuJoCo.

Only major differences we notice is on Hopper task. For Hopper task, margin between our method and other baselines are smaller and the relative ordering of Guided-ES and ARS changes. After a closer inspection, we notice that this is due to the fact that simulation stops when the agent falls. In other words, the simulation time depends on the current solution. Hence, methods which queries the simulator for these unstable solutions, looses less wall-clock time.

Quantifying manifold learning performance. In order to evaluate the learning performance, we project the gradient of the function to the tangent space of the learned manifold and plot the norm of the residual. Since we do not have access to the gradients, we estimate them at 30 time instants, evenly distributed through the learning process. We perform an accurate gradient estimation using a very large number of directions (2000). We compute the norm of the residual of the projection as $\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t) - \mathbf{P}_{\mathbf{J}_q(\mathbf{x}^t,\theta^t)}(\nabla_{\mathbf{x}} f(\mathbf{x}^t))\|_2$, where $\mathbf{P}_{\mathbf{A}}(\cdot)$ is a projection operator to the column space of $\mathbf{A}$. We plot the results in Figure 3. As the figure suggests, our method successfully and quickly learns the manifold in all cases.

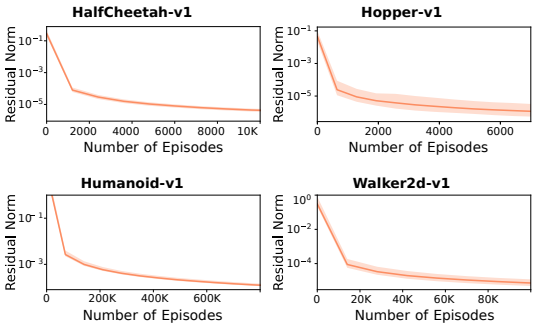

Figure 3: Manifold learning accuracy. We plot $\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t) - \mathbf{P}_{\mathbf{J}_q(\mathbf{x}^t,\theta^t)}(\nabla_{\mathbf{x}} f(\mathbf{x}^t))\|_2$. In order to estimate the gradient, we use GRADEST with a high number of directions.

Ablation studies. Our method uses three major ideas. i) We learn a manifold that the function lies on. ii) We learn this manifold in an online fashion. iii) We perform random search on the learned manifold. To study the impact of each of these ideas, we perform the following experiments. i) **No**

learning: We randomly initialize the manifold $r(\cdot;\theta)$ by sampling the entries of θ from the standard normal distribution. Then, we perform a random search on this random manifold. ii) **No online learning**: We collect an offline training dataset by sampling $\mathbf{x}_i$ values uniformly at random from a range which includes the optimal solutions. We evaluate function values at sampled points and learn the manifold. We use the learned model in a random search without any update. iii) **No search:** We use the gradients of the estimated function ($\nabla_{\mathbf{x}} g(r(\mathbf{x};\theta^t)\psi^t)$) as surrogate gradients and minimize the function of interest using first-order methods.

We list the results in Table 1 and do not include the no-search baseline since it fails to solve any of the tasks. Failure of the no-search baselines suggests that the estimated functions are powerful enough to guide the search, but they are not accurate enough to optimize. The no-learning baseline outperforms ARS in the simplest problem (Swimmer), but either fails completely or increases sample complexity in all other problems, suggesting that random features are not effective, especially in high-dimensional problems. Although the offline learning baseline solves more tasks than the no-learning one, it has worse sample complexity since initial offline sampling is expensive. This study indicates that all three of the ideas we use are crucial.

5.2 Continuous Optimization Benchmarks

We use continuous optimization problems from the Pagmo problem suite (Biscani et al., 2019). This benchmark includes minimization of 46 functions such as *Rastrigin, Rosenbrock, Schwefel,* etc. (See Appendix B for the complete list.) We use ten random starting points and report the average number of function evaluations required to reach a stationary point. Figure 4 reports the results as performance profiles (Dolan & Moré, 2002). Performance profiles represent how frequently a method is within distance τ of optimality. Specifically, when we denote the number of function evaluations that a method m requires to solve problem p as $T_m(p)$ and the number of function evaluations which the best method uses as $T^\star(p) = \min_m T_m(p)$, the performance profile is the fraction of problems for which the method is within τ distance of the best method: $1/N_p \sum_p \mathbb{1}[T_m(p) - T^\star(p) \leq \tau]$, where $\mathbb{1}[\cdot]$ is the indicator function and N_p is the number of problems.

Figure 4 suggests that our method outperforms all baselines. REMBO performs very close to our method and outperforms all other local search methods. The success of our method is not surprising since the functions are typically defined as non-convex functions of some statistics, making them lie on low-dimensional manifolds. Bayesian optimization outperforms ARS and other local adaptation heuristics such as CMA-ES and Guided-ES. We believe this is due to the global geometric structure of the considered functions. On the other hand, both CMA-ES and Guided-ES outperform ARS. CMA-ES and Guided-ES perform similarly: there is no statistically significant difference between them.

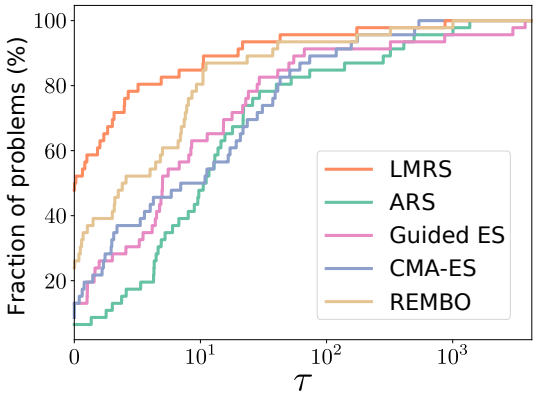

Figure 4: Performance profiles of our method and baselines on an optimization benchmark.

5.3 Optimization of an Airfoil

We apply our method to gradient-free optimization of a 2D airfoil. We use a computational fluid dynamics (CFD) simulator, XFoil (Drela, 1989), which can simulate an airfoil using its contour plot. We parametrize the airfoils using smooth polynomials of up to 36 degrees. We model the upper and lower parts of the airfoil with different polynomials. The dimensionality of the problem is thus 72. XFoil can simulate various viscosity properties, speeds, and angles of attack. We discuss the details in Appendix B. We plot the resulting airfoil after 1500 simulator calls in Table 2. We also report the lift and drag of the resulting shape. The reward function we optimize is Lift $-$ Drag. Table 2 suggests that all methods find airfoils that can fly (Lift $>$ Drag). Our method yields the highest Lift $-$ Drag. Bayesian optimization outperforms other local adaptation techniques.

		Airfoil after 1500 Simulations				
	Init	LMRS	ARS	GuidedES	CMA-ES	REMBO
LIFT	-0.229	$\mathbf{2.0126}$	0.567	1.231	1.418	1.8645
DRAG	0.040	0.0389	$\mathbf{0.007}$	0.0598	0.0249	0.02123
LIFT - DRAG	-0.269	$\mathbf{1.9737}$	0.560	1.1712	1.3931	1.8432
Foil						

Table 2: Generated airfoils with their lift and drag values after 1500 calls to XFoil (Drela, 1989).

6 RELATED WORK

Derivative-free optimization. We summarize the work on DFO that is relevant to our paper. For a complete review, readers are referred to Custódio et al. (2017) and Conn et al. (2009). We are specifically interested in random search methods, which have been developed as early as Matyas (1965) and Rechenberg (1973). Convergence properties of these methods have recently been analyzed by Agarwal et al. (2010), Bach & Perchet (2016), Nesterov & Spokoiny (2017), and Dvurechensky et al. (2018). A lower bound on the sample complexity for the convex case has been given by Duchi et al. (2015) and Jamieson et al. (2012). Bandit convex optimization is also highly relevant and we utilize the work of Flaxman et al. (2005) and Shamir (2013).

Random search for learning continuous control. Learning continuous control is an active research topic that has received significant interest in the reinforcement learning community. Recently, Salimans et al. (2017) and Mania et al. (2018) have shown random search methods to be competitive with state-of-the-art policy gradient methods in this setting. Vemula et al. (2019) analyzed this phenomenon theoretically and characterized the sample complexity of random search and policy gradient methods for continuous control.

Adaptive random search. There are various methods in the literature that adapt the search space by using anisotropic covariance as in the case of CMA-ES (Hansen et al., 2003; Hansen, 2016), guided evolutionary search (Maheswaranathan et al., 2018), and active subspace methods (Choromanski et al., 2019). There are also methods that enforce structure such as orthogonality in the search directions (Choromanski et al., 2018). There are also methods that use information geometry tools as in Wierstra et al. (2014) and Glasmachers et al. (2010). When a neuro-evolutionary search is used, Lehman et al. (2018) uses gradient magnitudes of the models to guide the search. As a slightly different approach, Staines & Barber (2012) use a variational lower bound to guide the search. In contrast to these methods, we explicitly make a nonlinear manifold assumption and directly learn this manifold via online learning. Our method is the only one which can learn an arbitrary nonlinear search space given a parametric class defining the geometry.

Adaptive Bayesian optimization. Bayesian optimization (BO) is another approach to zeroth-order optimization. Although we are only interested in random search based methods, some of the ideas we use have been utilized in BO. Calandra et al. (2016) used the manifold assumption for Gaussian processes. In contrast to our method, they use autoencoders for learning the manifold, which requires offline data. Similarly, Djolonga et al. (2013) consider the case where the function of interest lies on some linear manifold and collect offline data to identify this manifold. In contrast, we only use online information and our models are nonlinear. One interesting approach is using random low-dimensional features instead of adaptation as proposed by Wang et al. (2016) and later utilized by Kirschner et al. (2019). Finally, Rolland et al. (2018) design adaptive BO methods for additive models. Major distinctions between our work and the adaptive BO literature include our use of nonlinear manifolds, performing no offline exploration, and posing the problem as online learning. Moreover, we also compare our method with BO.

7 CONCLUSION

We described an approach to high-dimensional derivative free optimization. Our approach is based on the online learning of the underlying structure of the problem while performing the optimization. Our experiments suggest that the resulting method is effective in a wide range of problems and significantly increases the sample efficiency.

REFERENCES

Alekh Agarwal, Ofer Dekel, and Lin Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *Conference on Learning Theory (COLT)*, 2010.

Francis R. Bach and Vianney Perchet. Highly-smooth zero-th order online optimization. In *Conference on Learning Theory (COLT)*, 2016.

Francesco Biscani, Dario Izzo, Wenzel Jakob, Marcus Martens, Alessio Mereta, Cord Kaldemeyer, Sergey Lyskov, Sylvain Corlay, Benjamin Pritchard, Kishan Manani, et al. Esa/Pagmo2: Pagmo 2.10, 2019.

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold Gaussian processes for regression. In *International Joint Conference on Neural Networks (IJCNN)*, 2016.

Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard E. Turner, and Adrian Weller. Structured evolution with compact architectures for scalable policy optimization. In *International Conference on Machine Learning (ICML)*, 2018.

Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, and Vikas Sindhwani. Adaptive sample-efficient blackbox optimization via es-active subspaces. *arXiv:1903.04268*, 2019.

Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to Derivative-Free Optimization*. SIAM, 2009.

Ana Luísa Custódio, Katya Scheinberg, and Luís Nunes Vicente. Methodologies and software for derivative-free optimization. *Advances and Trends in Optimization with Engineering Applications*, 2017.

Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional Gaussian process bandits. In *Neural Information Processing Systems*, 2013.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2), 2002.

Mark Drela. Xfoil: An analysis and design system for low Reynolds number airfoils. In *Low Reynolds Number Aerodynamics*, 1989.

John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Trans. Information Theory*, 2015.

Pavel Dvurechensky, Alexander Gasnikov, and Eduard Gorbunov. An accelerated method for derivative-free smooth stochastic convex optimization. *arXiv:1802.09022*, 2018.

Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Symposium on Discrete Algorithms (SODA)*, 2005.

David A Freedman. On tail probabilities for martingales. *Annals of Probability*, 3(1), 1975.

Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4), 2013.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems*, pp. 131–162. Springer, 2010.

Tobias Glasmachers, Tom Schaul, Yi Sun, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2010.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv:1604.00772*, 2016.

Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1), 2003.

Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4), 2016.

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, et al. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

Kevin G. Jamieson, Robert D. Nowak, and Benjamin Recht. Query complexity of derivative-free optimization. In *Neural Information Processing Systems*, 2012.

Sham M Kakade and Ambuj Tewari. On the generalization ability of online strongly convex programming algorithms. In *Neural Information Processing Systems*, 2009.

Adam Tauman Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3), 2005.

Johannes Kirschner, Mojmír Mutný, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. *arXiv:1902.03229*, 2019.

Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of Monte Carlo Methods*. John Wiley & Sons, 2013.

Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O. Stanley. Safe mutations for deep and recurrent neural networks through output gradients. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2018.

Niru Maheswaranathan, Luke Metz, George Tucker, and Jascha Sohl-Dickstein. Guided evolutionary strategies: Escaping the curse of dimensionality in random search. *arXiv:1806.10230*, 2018.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Neural Information Processing Systems*, 2018.

Sébastien Marmin, Clément Chevalier, and David Ginsbourger. Differentiating the multipoint expected improvement for optimal batch design. In *Machine Learning, Optimization, and Big Data - First International Workshop, MOD*, 2015.

J Matyas. Random optimization. *Automation and Remote Control*, 26(2), 1965.

Yurii Nesterov and Vladimir G. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2), 2017.

Ingo Rechenberg. Evolutionsstrategie - optimierung technischer systeme nach prinzipien der biologischen information. *Stuttgart-Bad Cannstatt: Friedrich Frommann Verlag*, 1973.

Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional Bayesian optimization via additive models with overlapping groups. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Nikitas Rontsis, Michael A. Osborne, and Paul J. Goulart. Distributionally ambiguous optimization techniques for batch bayesian optimization, 2017.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv:1703.03864*, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2), 2012.

Ohad Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. In *Conference on Learning Theory (COLT)*, 2013.

Joe Staines and David Barber. Variational optimization. *arXiv:1212.4507*, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Anirudh Vemula, Wen Sun, and J. Andrew Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55, 2016.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research (JMLR)*, 15, 2014.

Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (KISS-GP). In *ICML*, 2015.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

A ADDITIONAL EXPERIMENTS

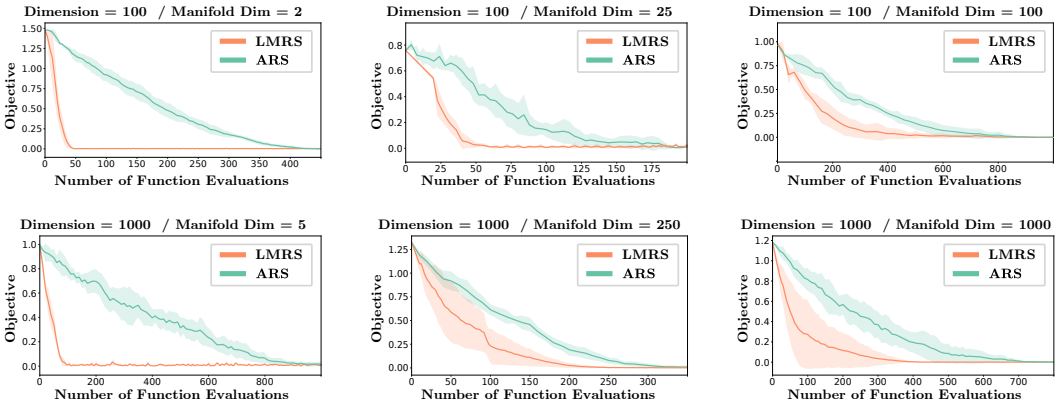

Figure 5: Average objective over 10 random seeds vs. number of function evaluations for various problem dimension (d) and manifold dimension (n). Shaded areas represent 1 standard deviation.

Experimental results in Seciton 5 suggests that the margin between our method and other baselines increases with larger dimension. In this section, we perform a controlled experiment to understand the effect of problem dimensionality (d) and the manifold dimensionality (n). We generate a collection of synthetic optimization problems. All synthesized functions follow the manifold hypothesis as $f(\mathbf{x}) = g(r(\mathbf{x}, \theta^\star))$ where $r(\mathbf{x}, \theta^\star)$ is a multilayer perceptron with the architecture Linear$(d, 2n) \to$ ReLU $\to$ Linear$(2n, n)$ and $g(\cdot)$ is a randomly sampled convex quadratic function. In order to sample a convex quadratic function, we sample the parameters of the quadratic function from a Gaussian distribution and project the resulting function to space of convex quadratic functions.

We choose $d \in \{100, 1000\}$ and plot the objective value with respect to the number of function evaluations for various manifold dimensionalities n in Figure 5. The results suggest that for a fixed dimensional problem, lower the manifold dimension, more sample efficient is our method. As our theory suggests, the improvement is very significant when $n \ll d$, as evident in the cases $(n = 5, d = 1000)$ and $(n = 2, d = 100)$.

One interesting observation is the fact that our method is effective even when the manifold assumption is violated ($n = d$). We hypothesize this is due to the anisotropic geometry. Although all directions are important as $n = d$, some directions will result in faster search since function changes more along them. In this case, our method can find these direction and result in faster search.

B ADDITIONAL IMPLEMENTATION DETAILS

In this section, we discuss the main details of the implementation. We will also share the code for our method, all baselines and all experiments open source upon acceptance. One key element of our method is the parametric family we use to learn the manifold. We consider a multi-layered perceptron with one to three hidden layers as Linear$(d, 2n) \to$ ReLU $\to$ Linear$(2n, n) \to$ ReLU $\to$ Linear(n, n) for $d < 1000$ and Linear$(d, 1/2d) \to$ ReLU $\to$ Linear$(1/2d, 2n) \to$ ReLU $\to$ Linear$(2n, n) \to$ ReLU $\to$ Linear(n, n) for $d > 1000$. We set the dimensionality of the manifold as the number of directions k which is a hyper parameter. We use grid search over δ and $n = k$ values and choose the best performing one in all experiments. Moreover, we also reinitialize the manifold parameters whenever the estimated gradient's magnitude is less than $1\text{e}{-6}$. We perform online gradient descent to learn the model parameters using SGD with momentum as 0.9. We also perform grid search for learning rate over $\{1\text{e}{-4}, 1\text{e}{-3}, 1\text{e}{-2}\}$. We further discuss experiment-specific details below:

MuJoCo experiments. We use linear policies and initialize them as zeros, which corresponds to no action. We use v2$-$t algorithm from Mania et al. (2018), which includes whitening of the observation space and using top-k directions instead of all. We use grid search over the parameter space described in Mania et al. (2018) for $n = k$, α and δ.

Low-dimensional unconstrained optimization suite. We use the following functions: *sphere*, *noisysphere, cigar, tablet, cigtab, cigtab2, elli, rosen, rosen_chained, diffpow, rosenelli, ridge, ridgecircle, happycat, branin, goldsteinprice, rastrigin, schaffer, schwefel2_22, lincon, rosen_nesterov, styblinski_tang, bukin* with dimensions $d = 10$ and $d = 100$ resulting in total 46 problems. We initialize all solutions with zero mean unit variance normal variables and use grid search over $\delta \in \{1\text{e}{-4}, 1\text{e}{-3}, 1\text{e}{-2}, 1\text{e}{-1}\}$, $k \in \{2, 5, 10, 50\}$, and $\alpha \in \{1\text{e}{-4}, 1\text{e}{-3}, 1\text{e}{-2}, 1\text{e}{-1}\}$.

Airfoil optimization. We initialize the parameters of the manifold with zero-mean unit-variance normal variables. We use grid search over $\delta \in \{1\text{e}{-4}, 1\text{e}{-3}, 1\text{e}{-2}, 1\text{e}{-1}\}$, $k \in \{2, 5, 10, 50\}$, and $\alpha \in \{1\text{e}{-4}, 1\text{e}{-3}, 1\text{e}{-2}, 1\text{e}{-1}\}$. For choosing hyper-parameters, we simulate all models with Reynolds number 12e6, speed 0.4 mach, and angle of attack 5 degrees. After the hyper-parameters are set, we used Reynolds number 14e6, speed 0.6 mach and angle of attack 2 degrees for evaluation.

Implementation Details for the Baselines. For ARS, we use grid search over the parameters recommended by the authors in (Mania et al., 2018). For CMA-ES, we use pycma (Hansen et al., 2019) with recommended hyper-parameters. For Bayesian optimization, we use the REMBO (Wang et al., 2016) optimizer with the following details. For continuous control problems, we use Structured Kernel Interpolation (SKI)(Wilson & Nickisch, 2015) and perform grid search over hyper-parameters over the range recommended by GPyTorch Gardner et al. (2018). For other problems, we use full GP inference since the number of samples is rather low and perform grid search over kernel parameters following GPyTorch Gardner et al. (2018). As an acquisition function, we perform grid search over Optimistic Expected Improvement (Rontsis et al., 2017), Multi-point Expected Improvement (Marmin et al., 2015), and (Ginsbourger et al., 2010).

C PROOFS

C.1 GRADIENT ESTIMATOR

In this section, we show that when random directions are sampled in the column space of the orthonormal matrix $\mathbf{U}$, perturbations give biased gradient estimates of the manifold smoothed function. Moreover, when $\mathbf{U} = \mathbf{U}^\star$, resulting gradients are unbiased. We formalize this with the following lemma.

Lemma 1. *Let $\mathbf{U}^\star$ be an orthonormal basis for the tangent space of the n-dimensional manifold $\mathcal{M}$ at point $\mathbf{x} \in \mathcal{M}$, $\mathbf{U}$ be another orthonormal matrix, and f be a function defined on this manifold. Fix $\delta > 0$. Then*

$$\mathbb{E}_{\mathbf{s}\sim\mathbb{S}^{n-1}}[f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})\mathbf{U}\mathbf{s}] = \frac{\delta}{n}\nabla_{\mathbf{x}}\tilde{f}_{\mathbf{U}}(\mathbf{x}) + \textsc{bias}(\mathbf{U})$$

where $\textsc{bias}(\mathbf{U}) = \mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}[f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})[\mathbf{U}-\mathbf{U}^\star]\mathbf{s}]$. *Moreover, bias is* 0 *and the resulting estimator is unbiased when* $\mathbf{U} = \mathbf{U}^\star$.

Proof. Without loss of generality, we can assume $\det \mathbf{U} = 1$ and $\det \mathbf{U}^\star = 1$. Using this remark, we can state the proof of the lemma as the straightforward application of the manifold Stoke's theorem

$$\begin{aligned}
\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}[f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})\mathbf{U}\mathbf{s}] &= \mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}[f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})\mathbf{U}^\star\mathbf{s}] + \mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}[f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})[\mathbf{U}-\mathbf{U}^\star]\mathbf{s}] \\
&\overset{(a)}{=} \frac{1}{\text{vol}(\delta\mathbb{S}^{n-1})}\int_{\delta\mathbb{S}^{n-1}} f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})\mathbf{U}^\star\mathbf{s}d\mathbf{s} + \underbrace{\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}[f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})[\mathbf{U}-\mathbf{U}^\star]\mathbf{s}]}_{\textsc{bias}(\mathbf{U})} \\
&= \frac{1}{\text{vol}(\delta\mathbb{S}^{n-1})}\int_{\delta\mathbb{S}^{n-1}} f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})\mathbf{U}^\star \det\mathbf{U}^\star\frac{\mathbf{s}}{\|\mathbf{s}\|}d\mathbf{s} + \textsc{bias}(\mathbf{U}) \\
&\overset{(b)}{=} \frac{1}{\text{vol}(\delta\mathbb{S}^{n-1})}\int_{\delta\mathbb{S}^{n-1}} f(\mathbf{x}+\delta\mathbf{U}\mathbf{U}^{\star\intercal}\mathbf{U}^\star\mathbf{s})\mathbf{U}^\star \det\mathbf{U}^\star\frac{\mathbf{s}}{\|\mathbf{s}\|}d\mathbf{s} + \textsc{bias}(\mathbf{U}) \qquad (8) \\
&\overset{(c)}{=} \frac{1}{\text{vol}(\delta\mathbb{S}^{n-1})}\nabla_{\mathbf{x}}\int_{\delta\mathbb{B}^{n}} f(\mathbf{x}+\delta\mathbf{U}\mathbf{U}^{\star\intercal}\mathbf{U}^\star\mathbf{v})d\mathbf{v} + \textsc{bias}(\mathbf{U}) \\
&\overset{(d)}{=} \frac{\text{vol}(\delta\mathbb{B}^{n})}{\text{vol}(\delta\mathbb{S}^{n-1})}\nabla_{\mathbf{x}}\frac{\int_{\delta\mathbb{B}^{n}} f(\mathbf{x}+\delta\mathbf{U}\mathbf{v})d\mathbf{v}}{\text{vol}(\delta\mathbb{B}^{n})} + \textsc{bias}(\mathbf{U}) \\
&\overset{(e)}{=} \frac{\delta}{n}\nabla_{\mathbf{x}}\tilde{f}_{\mathbf{U}}(\mathbf{x}) + \textsc{bias}(\mathbf{U}).
\end{aligned}$$

where vol denotes volume, and we use the definition of the expectation in (a, e), orthonormality of $\mathbf{U}^\star$ in (b,d), manifold Stoke's theorem in (c) and the fact that the ratio of volume to the surface area of a $n-$dimensional ball of radius δ is $\frac{\delta}{n}$ in (e). Moreover, bias vanishes when $\mathbf{U} = \mathbf{U}^\star$. □

C.2 Sample Complexity for Random Search and Manifold Random Search

In this section, we bound the sample complexity of the random search (Algorithm 1) and the manifold random search (Algorithm 2). Our analysis starts with studying the relationship between the function (f) and its smoothed ($\hat{f}$) as well as manifold smoothed ($\tilde{f}_{\mathbf{U}}$) versions in section C.2.2. We show that $L-$Lipschitzness and $\mu-$smoothness of the function extend to the smoothed functions. Moreover, we also bound the difference between the gradients of the original function and the gradients of the smoothed versions. Next, we study the second moment of the gradient estimator in section C.2.3. Finally, we state the sample complexity of SGD on non-convex functions in section C.2.1. Combining these results, we state the final sample complexity of random search and manifold random search in section C.2.4&C.2.5.

C.2.1 Convergence of SGD for non-convex fuctions

The convergence of the SGD has been widely studied and here we state its convergence result for non-convex functions from Ghadimi & Lan (2013) as a Lemma and give its proof for the sake of completeness.

Lemma 2 (Convergence of SGD (Ghadimi & Lan, 2013; Vemula et al., 2019))**.** *Consider running SGD on $f(\mathbf{x})$ that is μ-smooth and L-Lipschitz for T steps starting with initial solution $\mathbf{x}^0$. Denote $\Omega^0 = f(\mathbf{x}^0) - f(\mathbf{x}^\star)$ where $\mathbf{x}^\star$ is the globally optimal point and assume that the unbiased gradient estimate has second moment bounded with V. Then,*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 \leq \sqrt{\frac{8\Omega^0\mu V}{T}} \tag{9}$$

Proof. We denote the step size as α and the unbiased gradient estimate as $\mathbf{g}^t$. We analyze the step at t as;

$$\begin{aligned} f(\mathbf{x}^{t+1}) &= f(\mathbf{x}^t - \alpha\mathbf{g}^t) \\ &\leq f(\mathbf{x}^t) - \alpha\nabla_{\mathbf{x}} f(\mathbf{x}^t)^\intercal \mathbf{g}^t + \frac{\mu\alpha^2}{2}\|\mathbf{g}^t\|_2^2 \end{aligned} \tag{10}$$

where we used the $\mu-$smoothness of the function. Taking expectation of the inequality,

$$\mathbb{E}_{\mathbf{g}^t}[f(\mathbf{x}^{t+1})] \leq f(\mathbf{x}^t) - \alpha\|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 + \frac{\mu\alpha^2}{2}\mathbb{E}[\|\mathbf{g}^t\|_2^2] \tag{11}$$

Using the bounded second moment of the gradient, and summing from step 1 to T,

$$\sum_{t=1}^{T}\mathbb{E}_{\mathbf{g}^t}[f(\mathbf{x}^{t+1})] \leq \sum_{t=1}^{T} f(\mathbf{x}^t) - \alpha\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 + \frac{\mu\alpha^2 TV}{2} \tag{12}$$

Re-arranging the terms, we obtain,

$$\begin{aligned} \sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 &\leq \frac{1}{\alpha}\mathbb{E}_{\mathbf{g}^0,\ldots,\mathbf{g}^t}[f(\mathbf{x}^0) - f(\mathbf{x}^{t+1})] + \frac{\mu\alpha TV}{2} \\ &\leq \frac{\Delta^0}{\alpha} + \mu T\alpha V \end{aligned} \tag{13}$$

Set $\alpha = \sqrt{\frac{2\Delta^0}{\mu TV}}$, and divide the inequality to T in order to obtain the required inequality as

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 \leq \sqrt{\frac{8\Delta^0\mu V}{T}}. \tag{14}$$

□

C.2.2 PRELIMINARY RESULTS ON SMOOTHED FUNCTIONS

First, we will show that the $\mu-$smoothness and $L-$ Lipschitness properties of f applies to $\hat{f}$ and $\tilde{f}$.

$$\begin{aligned}|\tilde{f}_{\mathbf{U}}(\mathbf{x}_1) - \tilde{f}_{\mathbf{U}}(\mathbf{x}_2)| &= |E_{\mathbf{v}\in\mathbb{B}^n}[f(\mathbf{x}_1 + \delta\mathbf{U}\mathbf{v})] - E_{\mathbf{v}\in\mathbb{B}^n}[f(\mathbf{x}_2 + \delta\mathbf{U}\mathbf{v})]| \\ &= |E_{\mathbf{v}\in\mathbb{B}^n}[f(\mathbf{x}_1 + \delta\mathbf{U}\mathbf{v}) - f(\mathbf{x}_2 + \delta\mathbf{U}\mathbf{v})]| \\ &\leq |E_{\mathbf{v}\in\mathbb{B}_n}[L\|\mathbf{x}_1 - \mathbf{x}_2\|_2]| \\ &\overset{(a)}{=} L\|\mathbf{x}_1 - \mathbf{x}_2\|_2\end{aligned} \tag{15}$$

where we use $L-$Lipschitz continuity of f in (a), and,

$$\begin{aligned}\|\nabla\tilde{f}_{\mathbf{U}}(\mathbf{x}_1) - \nabla\tilde{f}_{\mathbf{U}}(\mathbf{x}_2)\|_2 &= \|\nabla E_{\mathbf{v}\in\mathbb{B}^n}[f(\mathbf{x}_1 + \delta\mathbf{U}\mathbf{v})] - \nabla E_{\mathbf{v}\in\mathbb{B}^n}[f(\mathbf{x}_2 + \delta\mathbf{U}\mathbf{v})]\|_2 \\ &= \|E_{\mathbf{v}\in\mathbb{B}^n}[\nabla f(\mathbf{x}_1 + \delta\mathbf{U}\mathbf{v})] - E_{\mathbf{v}\in\mathbb{B}^n}[\nabla f(\mathbf{x}_2 + \delta\mathbf{U}\mathbf{v})]\|_2 \\ &= \|E_{\mathbf{v}\in\mathbb{B}^n}[\nabla f(\mathbf{x}_1 + \delta\mathbf{U}\mathbf{v}) - \nabla f(\mathbf{x}_2 + \delta\mathbf{U}\mathbf{v})]\|_2 \\ &\overset{(b)}{\leq} E_{\mathbf{v}\in\mathbb{B}^n}[\|\nabla f(\mathbf{x}_1 + \delta\mathbf{U}\mathbf{v}) - \nabla f(\mathbf{x}_2 + \delta\mathbf{U}\mathbf{v})\|_2] \\ &\overset{(c)}{\leq} \mu\|\mathbf{x}_1 - \mathbf{x}_2\|_2\end{aligned} \tag{16}$$

where we use Jensen's inequality and convexity of the norm in (b) and $\mu-$smoothness of f in (c). Hence, $\mu-$smoothness and $L-$ Lipschitness applies to $\tilde{f}_{\mathbf{U}}$ for any $\mathbf{U}$. Take $n = d$ and $\mathbf{U} = \mathbf{I}$, then the $\mu-$smoothness and $L-$ Lipschitness applies to $\hat{f}$.

Next, we will study the impact of using the gradients of the smoothed function instead of the original function.

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x})\|_2^2 &= \frac{1}{T}\sum_{t=1}^{T}\|\nabla_x f(\mathbf{x}) - \nabla_x\tilde{f}_{\mathbf{U}}(\mathbf{x}) + \nabla_{\mathbf{x}}\tilde{f}_{\mathbf{U}}(\mathbf{x})\|_2^2 \\ &\leq \frac{2}{T}\sum_{t=1}^{T}\|\nabla_x f(\mathbf{x}) - \nabla_{\mathbf{x}}\tilde{f}_{\mathbf{U}}(\mathbf{x})\|_2^2 + \frac{2}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\tilde{f}_{\mathbf{U}}(\mathbf{x})\|_2^2.\end{aligned} \tag{17}$$

where we use $\|a + b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$. We further bound the left term as

$$\begin{aligned}\|\nabla_x f(\mathbf{x}) - \nabla_{\mathbf{x}}\tilde{f}_{\mathbf{U}}(\mathbf{x})\|_2^2 &= \|\nabla_x f(\mathbf{x}) - \nabla_{\mathbf{x}}\mathbb{E}_{\mathbf{v}\sim\mathbb{B}_n}[f(\mathbf{x} + \delta\mathbf{U}\mathbf{v})]\|_2^2 \\ &\overset{(a)}{=} \|\mathbb{E}_{\mathbf{v}\sim\mathbb{B}_n}[\nabla_{\mathbf{x}} f(\mathbf{x}) - \nabla_{\mathbf{x}} f(\mathbf{x} + \delta\mathbf{U}\mathbf{v})]\|_2^2 \\ &\overset{(b)}{\leq} \|\mathbb{E}_{\mathbf{v}\sim\mathbb{B}_n}[\delta\mu\|\mathbf{U}\mathbf{v}\|_2]\|_2^2 \\ &\overset{(c)}{\leq} \delta^2\mu^2\end{aligned} \tag{18}$$

using dominated convergence theorem in (a), the $\mu-$smoothness of f in (b) and orthonormality of $\mathbf{U}$ and the fact that norm of any point in a unit ball is bounded by 1. By taking $n = d$ and $\mathbf{U} = \mathbf{I}$, this result also implies the same for $\hat{f}$. Hence,

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x})\|_2^2 &\leq \frac{2}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\hat{f}(\mathbf{x})\|_2^2 + 2\delta^2\mu^2 \\ \frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x})\|_2^2 &\leq \frac{2}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\tilde{f}_{\mathbf{U}^t}(\mathbf{x})\|_2^2 + 2\delta^2\mu^2 \quad \forall_{\mathbf{U}^1,\ldots,\mathbf{U}^T}\end{aligned} \tag{19}$$

C.2.3 SECOND MOMENT OF THE GRADIENT ESTIMATOR

We will start with studying the second moment of our gradient estimate for the manifold case. We bound the expected square norm of the gradient estimate as

$$\begin{aligned}
\mathbb{E}_{\mathbf{s}\in\mathbb{S}^n,\xi}\left[\left\|\mathbf{g}_m^t\right\|_2^2\right] &= \mathbb{E}_{\mathbf{s}\in\mathbb{S}^n,\xi}\left[\left\|\frac{n}{2\delta}\left[F(\mathbf{x}^t+\delta\mathbf{U}\mathbf{s},\xi_1)-F(\mathbf{x}^t-\delta\mathbf{U}\mathbf{s},\xi_2)\right]\mathbf{U}\mathbf{s}\right\|_2^2\right] \\
&\overset{(a)}{=} \frac{n^2}{4\delta^2}\mathbb{E}_{\mathbf{s}\in\mathbb{S}^n,\xi}\left[\left(F(\mathbf{x}^t+\delta\mathbf{U}\mathbf{s},\xi_1)-F(\mathbf{x}-\delta\mathbf{U}\mathbf{s},\xi_2)\right)^2\right] \\
&\overset{(b)}{\leq} \frac{n^2}{2\delta^2}\mathbb{E}_{\mathbf{s}\in\mathbb{S}^n}\left[\left(f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})-f(\mathbf{x}-\delta\mathbf{U}\mathbf{s})\right)^2\right] \\
&\quad+\frac{n^2}{\delta^2}\mathbb{E}_{\mathbf{s}\in\mathbb{S}^n,\xi}\left[\left(F(\mathbf{x}+\delta\mathbf{U}\mathbf{s},\xi)-f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})\right)^2\right] \\
&\quad+\frac{n^2}{\delta^2}\mathbb{E}_{\mathbf{s}\in\mathbb{S}^n,\xi}\left[\left(F(\mathbf{x}-\delta\mathbf{U}\mathbf{s},\xi)-f(\mathbf{x}-\delta\mathbf{U}\mathbf{s})\right)^2\right] \\
&\overset{(c)}{\leq} 2n^2L^2+\frac{2n^2V_F}{\delta^2}
\end{aligned} \tag{20}$$

where we use orthonormality of $\mathbf{U}$ and unit norm property of $\mathbf{s}$ in (a), add and substract $f(\mathbf{x}+\delta\mathbf{U}\mathbf{s})-f(\mathbf{x}-\delta\mathbf{U}\mathbf{s})$ and use $(a+b)^2 \leq 2a^2+2b^2$ in (b), use the bounded variance of F and the Lipschitz smoothness of f in (c).

Second moment of the random search estimator can also computed similarly. And, the resulting bound would be

$$\mathbb{E}_{\mathbf{s}\in\mathbb{S}^n,\xi}\left[\left\|\mathbf{g}_e^t\right\|_2^2\right] \leq 2d^2L^2+\frac{2d^2V_F}{\delta^2} \tag{21}$$

C.2.4 PROOF OF PROPOSITION 1

Proof. Analysis of SGD from Lemma 2 shows that

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|_2^2 \leq \sqrt{\frac{8\Omega^0\mu V}{T}}. \tag{22}$$

Using the bound on the second moment of the estimator we derive in (21),

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|_2^2 \leq \sqrt{\frac{8\Omega^0\mu}{T}}\sqrt{2d^2L^2+\frac{2d^2V_F}{\delta^2}} \tag{23}$$

Using the relationship between f and $\hat{f}$ we derive in (19),

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\hat{f}(\mathbf{x})\|_2^2 \leq 2\delta^2\mu^2+2\sqrt{\frac{8\Omega^0\mu}{T}}\sqrt{2d^2L^2+\frac{2d^2V_F}{\delta^2}}. \tag{24}$$

We use the property $\sqrt{a+b}\leq\sqrt{a}+\sqrt{b}$, and solve for α. After we substituted the resulting δ,

$$\frac{1}{T}\sum_{t}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|_2^2 \leq \frac{c_0+c_1d}{T^{\frac{1}{2}}}+\frac{c_2d^{\frac{2}{3}}}{T^{\frac{1}{3}}} \tag{25}$$

where $c_0=4L\sqrt{2\Omega^0\mu}$, $c_1=\sqrt{2}c_0$, and $c_2=(4V_F\Omega^0)^{\frac{1}{3}}(2\mu+4\mu^{\frac{5}{6}})$. □

C.2.5 PROOF OF PROPOSITION 2

Proof. Sample complexity of the manifold random search follows closely the proof of Proposition 1. We summarize here for the sake of completeness. Using the analysis of SGD from Lemma 2,

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|_2^2 \leq \sqrt{\frac{8\Omega^0\mu V}{T}}. \tag{26}$$

Using the bound on the second moment of the estimator we derive in (21), and the relationship between f and $\tilde{f}_{\mathbf{U}^\star}$ we derive in (19),

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\hat{f}(\mathbf{x})\|_2^2 \leq 2\delta^2\mu^2 + 2\sqrt{\frac{8\Omega^0\mu}{T}}\sqrt{2n^2L^2 + \frac{2n^2V_F}{\delta^2}}. \tag{27}$$

We first use the property $\sqrt{a+b} \leq \sqrt{a}+\sqrt{b}$, then solve for α and δ. Finally, we substitute resulting δ to get the final result as;

$$\frac{1}{T}\sum_{t}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|_2^2 \leq \frac{c_0 + c_1 n}{T^{\frac{1}{2}}} + \frac{c_2 n^{\frac{2}{3}}}{T^{\frac{1}{3}}} \tag{28}$$

where $c_0 = 4L\sqrt{2\Omega^0\mu}$, $c_1 = \sqrt{2}c_0$, and $c_2 = (4V_F\Omega^0)^{\frac{1}{3}}(2\mu + 4\mu^{\frac{5}{6}})$.

□

C.3 Proof of the Theorem 1

Proof. We will prove our main theorem using a three major arguments. First, we analyze the SGD of a non-convex function with biased gradients in section C.3.1. Second, we show that the expected value of our loss function is equal to the bias term in section C.3.2. In order to bound the difference between the empirical loss function we minimize and its expectation, we use the Freedman's inequality Freedman (1975). Third, we bound the empirical loss in section C.3.3 in terms of the distance travelled by the iterates of the optimization $\|x^{t+1} - x^t\|_2$. Finally, we optimize the resulting bound in terms of the finite difference step (δ^t), SGD step size (α^t), and mixing coefficients (β) to obtain the final statement of the theorem in section C.3.4.

C.3.1 Analysis of SGD with bias

In order to analyze the SGD with bias, we will denote the gradient at the iteration t, as $\mathbf{g}^t$. Moreover, we will assume that its bias is $\mathbf{b}^t$ as $\mathbb{E}[\mathbf{g}^t] = \mathbf{b}^t + \nabla_{\mathbf{x}}\tilde{F}(\mathbf{x},\xi)$. Using the $\mu-$smoothness of the function $\tilde{F}$, we can state that

$$\tilde{F}(\mathbf{x}^{t+1},\xi) = \tilde{F}(\xi^t - \alpha\mathbf{g}^t,\xi) \leq \tilde{F}(\mathbf{x}^t,\xi) - \alpha\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)\mathbf{g}^t + \frac{\mu\alpha^2}{2}\|\mathbf{g}^t\|_2^2. \tag{29}$$

Let's assume the effect of the bias is bounded as $|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)^\intercal\mathbf{b}^t| \leq B^t$, then

$$\tilde{F}(\mathbf{x}^{t+1},\xi) \leq \tilde{F}(\mathbf{x}^t,\xi) - \alpha\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)[\mathbf{g}^t - \mathbf{b}^t] + \frac{\mu\alpha^2}{2}\|\mathbf{g}^t\|_2^2 + \alpha B^t. \tag{30}$$

Taking the expectation with respect to $\mathbf{s}$ and ξ, we get

$$\alpha\|\nabla_{\mathbf{x}}\tilde{f}(\mathbf{x}^t)\|_2^2 \leq \mathbb{E}_{\mathbf{s},\xi}[\tilde{f}(\mathbf{x}^{t+1})] - \tilde{f}(\mathbf{x}^t) + \frac{\mu\alpha^2\mathbf{V_g}}{2} + \alpha B^t \tag{31}$$

where $\mathbf{V_g} = \mathbb{E}[\|\mathbf{g}^t\|_2^2]$. Summing up from $t=1$ to T and dividing by α, we obtain

$$\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\tilde{f}(\mathbf{x}^t)\|_2^2 \leq \frac{\Omega}{\alpha} + \frac{\mu\alpha T\mathbf{V_g}}{2} + \sum_{t=1}^{T}B^t. \tag{32}$$

We now compute the bound on the bias term (B^t) using Lemma 1 and $\mathbf{g}^t = \beta\mathbf{g}_e^t + (1-\beta)\mathbf{g}_m^t$,

$$\begin{aligned}\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)^\intercal\mathbf{b}^t &= (1-\beta)\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}\left[F(\mathbf{x}+\delta\mathbf{U}\mathbf{s},\xi)\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)^\intercal[\mathbf{U}-\mathbf{U}^\star]\mathbf{s}\right] \\ &\leq (1-\beta)\Omega\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}\left[|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)^\intercal[\mathbf{U}-\mathbf{U}^\star]\mathbf{s}|\right]\end{aligned} \tag{33}$$

where we used the fact that function is bounded as $F(\mathbf{x},\xi) \leq \Omega$. Since $\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x},\xi)$ lies in the column space of $\mathbf{U}^\star$, for some $\mathbf{p}$,

$$\mathbf{p}^\intercal\mathbf{U}^{\star\intercal}[\mathbf{U}-\mathbf{U}^\star] = \mathbf{p}^\intercal[\mathbf{U}^{\star\intercal}\mathbf{U}-\mathbf{I}] = \mathbf{p}^\intercal[\mathbf{U}^{\star\intercal}\mathbf{U}-\mathbf{U}^\intercal\mathbf{U}] = [\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x},\xi)^\intercal - \mathbf{p}^\intercal\mathbf{U}^\intercal]\mathbf{U} \tag{34}$$

Choice of bases ($\mathbf{U}^\star$) is not unique in Lemma 1. It can be any orthonormal basis spanning the tangent space of the manifold. In order to minimize $\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{n-1}}\left[\left|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)^\intercal[\mathbf{U}-\mathbf{U}^\star]\mathbf{s}\right|\right]$, choose the basis which will set $\mathbf{U}\mathbf{p}$ to the projection of $\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)$ into the column space of $\mathbf{U}$. Then,

$$\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)^\intercal[\mathbf{U}-\mathbf{U}^\star]\|_2 \leq \min_{\mathbf{q}\in\mathbb{R}^n}\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)-\mathbf{U}\mathbf{q}\|_2 \leq \|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)-\nabla_{\mathbf{x}}g(r(\mathbf{x}^t;\theta^t);\psi^t)\|_2 \tag{35}$$

where last inequality is due to $\nabla_{\mathbf{x}}g(r(\mathbf{x}^t;\theta^t);\psi^t)$ being in the column space of $\mathbf{U}$. Combining with (32), (19), and $0\leq\beta\leq 1$;

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|_2^2 \leq \frac{\Omega}{\alpha T}+\frac{\mu\alpha\mathbf{V_g}}{2}+\frac{\Omega}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)-\nabla_{\mathbf{x}}g(r(\mathbf{x}^t;\theta^t);\psi^t)\|_2+\delta^2\mu^2. \tag{36}$$

We proceed to bound $\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi)-\nabla_{\mathbf{x}}g(r(\mathbf{x}^t;\theta^t);\psi^t)\|_2$ in the next section.

C.3.2 Role of the Bandit Feedback

The true loss we are interested in is the effect of bias on the SGD. The bias is the sum of the differences between the gradients of the true function ($\tilde{F}(\mathbf{x},\xi)$) and the estimated one ($g(r(\mathbf{x};\theta);\psi)$) as derived in (36). On the other hand, the empirical information we have is the projection of this loss to a random direction ($\mathbf{s}$) with an additional noise term. In this section, we will analyze the difference between the bias and the empirical loss without the noise. We will include the discussion on the noise in section C.3.3.

First, we will show that expectation of the empirical loss over a direction uniformly chosen from a unit sphere is the bias term. In order to show this, we need an elementary result which is $\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}[(\mathbf{s}^t\mathbf{v})^2]=\frac{\|\mathbf{v}\|^2}{d}$. We show this result in Section C.4.1. Using this result,

$$\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}\left[\left(\mathbf{s}^\intercal\left(\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x},\xi)-\nabla_{\mathbf{x}}g((r(\mathbf{x};\theta);\psi)\right)\right)^2\right]=\frac{1}{d}\left\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x},\xi)-\nabla_{\mathbf{x}}g((r(\mathbf{x};\theta);\psi)\right\|_2^2 \tag{37}$$

We introduce the notation $\Delta(\mathbf{s},\mathbf{x},\xi,\theta,\psi)=\left(\mathbf{s}^\intercal\left(\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x},\xi)-\nabla_{\mathbf{x}}g((r(\mathbf{x};\theta);\psi)\right)\right)^2$ for clarity and, proceed to bound the difference $\left|\Delta(\mathbf{s},\mathbf{x},\xi,\theta,\psi)-\mathbb{E}_{\mathbf{s}}[\Delta(\mathbf{s},\mathbf{x},\xi,\theta,\psi)]\right|$. Consider the sequence of differences as,

$$\mathbf{Z}^t=\Delta(\mathbf{s}^t,\mathbf{x}^t,\xi^t,\theta^t,\psi^t)-\frac{1}{d}\left\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)-\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right\|_2^2, \tag{38}$$

it is clear that $\mathbb{E}[\mathbf{Z}^t]=0$ for all t. Moreover, the differences are bounded due to the Lipschitz continuity. Hence, $\mathbf{Z}^t$ is a martingale difference sequence. We use the Freedman's inequality (Freedman, 1975) in order to bound $\sum\mathbf{Z}^t$, similar to the seminal work studying the generalization of online learning by Kakade & Tewari (2009). Freedman's inequality (Freedman, 1975) states that if $\mathbf{Z}^t$ is a martingale difference sequence,

$$P\left(\sum_{t=1}^{T}\mathbf{Z}^t\geq\max\left\{2\sqrt{\sum_{i=1}^{T}Var(\mathbf{Z}^t)},3b\sqrt{\ln(1/\gamma)}\right\}\sqrt{\ln(1/\gamma)}\right)\leq 4\gamma\ln(T) \tag{39}$$

where b is the bound on $\mathbf{Z}^t$ as $|\mathbf{Z}^t|\leq b$ for all t. Before we substitute the $\mathbf{Z}^t$ in the Freedman's inequality, we need to compute the variance of the $\mathbf{Z}^t$. We bound the variance using the definition of the variance as

$$\begin{aligned}
&\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}\left[\left(\left(\mathbf{s}^{t\intercal}\left(\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)-\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right)\right)^2-\frac{1}{d}\left\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)-\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right\|^2\right)^2\right]\\
&\leq\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}\left[\left(\frac{1}{d}\left(\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)-\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right)^\intercal\left(d\mathbf{s}^t-\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)+\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right)\right)^2\right]\\
&\overset{(a)}{\leq}\frac{\left\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)-\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right\|_2^2}{d^2}\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}\left[\left(\nu^\intercal\left(d\mathbf{s}^t-\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)+\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right)\right)^2\right]\\
&\overset{(b)}{\leq}\frac{1}{d^2}\left\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)-\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right\|_2^2\left(d^2\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}[(\nu^\intercal\mathbf{s}^t)^2]+2L^2\right)\\
&=\left(\frac{2L^2+d}{d^2}\right)\left\|\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t,\xi^t)-\nabla_{\mathbf{x}}g((r(\mathbf{x}^t;\theta^t);\psi^t)\right\|_2^2
\end{aligned} \tag{40}$$

where we denote the unit vector in the direction of $\nabla_{\mathbf{x}}\tilde{F}(\mathbf{x}^t, \xi^t) - \nabla_{\mathbf{x}} g((r(\mathbf{x}^t; \theta^t); \psi^t)$ as ν in (a) and use the Lipschitz property as well as (63) in (b). Finally we substitute this result in Freedman's inequality and use the expectation $\Delta(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^t)$ from (37). We also introduce the shorthand notation $\Delta^t = \Delta(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^)$. With probability at least $1 - 4\gamma \ln(T)$,

$$\sum_{t=1}^{T} \mathbb{E}[\Delta^t] \leq \sum_{t=1}^{T} \Delta^t + \max\left\{ 2\sqrt{\frac{2L^2+d}{d}} \sqrt{\sum_{t=q}^{T} \mathbb{E}[\Delta^t]}, 6L^2 \left(\frac{1+d}{d}\right) \sqrt{\ln(1/\gamma)} \right\} \sqrt{\ln(1/\gamma)} \tag{41}$$

We can further bound $\sum_{t=1}^{T} \mathbb{E}[\Delta^t]$ by using the fact that (41) is in the form of $s^2 \leq r + \max\{2as, 6bc\}c$ which can be solved for s using quadratic formula. We solve this quadratic in Section C.4.2, and show that it implies $s \leq r + 2ac\sqrt{r} + \max\{4a^2, 6b\}c^2$. Using the solution of the quadratic formula, we get the following final result describing the effect of bandit feedback. With probability $1 - 4\gamma \ln(T)$,

$$\sum_{t=1}^{T} \mathbb{E}[\Delta^t] \leq \sum_{t=1}^{T} \Delta^t + 2\sqrt{\frac{2L^2+d}{d}} \sqrt{\sum_{t=1}^{T} \Delta^t} \sqrt{\ln(1/\gamma)} + \max\left\{ \frac{8L^2+4d}{d}, 6L^2 \left(\frac{1+d}{d}\right) \right\} \ln(1/\gamma) \tag{42}$$

In summary, we bound the difference between the effect of the bias $\sum_{t=1}^{T} \mathbb{E}[\Delta^t]$ and the empirical loss we minimize $\sum_{t=1}^{T} \Delta^t$ without the noise term. In the next section, we procede to bound the empirical loss $\sum_{t=1}^{T} \mathcal{L}^t$ and the effect of the noise term $\sum_{t=1}^{T} |\Delta^t - \mathcal{L}^t|$.

C.3.3 Analysis of the Empirical Loss

In this section, we will analyze the empirical loss. Our analysis is similar to the regret analysis of Follow the Regularized Leader (FTRL). However, we do not get an adverserial bound. Our resulting bound is the function of distances of iterates denoted as $\|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2$. Such a bound would not be useful in adverserial setting since adversary chooses the iterates. However, we also design the optimization method. Hence, we bound $\|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2$ by setting step sizes accordingly.

We start our analysis with bounding the total empirical loss in terms of the length of the trajectory the learner takes. As a consequence of the FTL-BTL Lemma (Kalai & Vempala, 2005),

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^t) \leq \sum_{t=1}^{T} \left[\mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^t) - \mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^{t+1}, \psi^{t+1}) \right] + \mathcal{R}(\theta^T, \psi^T). \tag{43}$$

We use the Lipschitz smoothness property to convert this into distance travelled by the learner as

$$\begin{aligned}
&\mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^t) - \mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^{t+1}, \psi^{t+1}) \\
&= \left(\frac{y(\mathbf{x}^t, \mathbf{s}^t, \xi^t)}{2\delta} - \mathbf{s}^{t\intercal} \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t) \right)^2 - \left(\frac{y(\mathbf{x}^t, \mathbf{s}^t, \xi^t)}{2\delta} - \mathbf{s}^{t\intercal} \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^{t+1}); \psi^{t+1}) \right)^2 \\
&\leq 4L \sum_{t=1}^{T} \mathbf{s}^{t\intercal} (\nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^{t+1}); \psi^{t+1})) \\
&\leq 4L \underbrace{\sum_{t=1}^{T} \|\nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^{t+1}); \psi^{t+1})\|_2}_{\textsc{LearnerPathLength}}
\end{aligned} \tag{44}$$

With properly chosen λ, our regularizer enforces the smallest possible update, in terms of learner path length, which is consistent with the current sampled directions. This is due to the representability assumption which guarantees that manifold can be fit perfectly using the parametric family. Hence, there is a solution with $\mathcal{L} = 0$. Considering the regularizer is the learner path length, with proper choice of λ, the FTRL will choose the shortest learner path length.

Among all choices of $\mathbf{s}^t$, $\mathbf{s}^t = {}^{\nabla_{\mathbf{x}} F(\mathbf{x}^t, \xi^t)}/_{\|\nabla_{\mathbf{x}} F(\mathbf{x}^t, \xi^t)\|}$ would result in the longest distance. Hence, we can bound the learner path distance of our empirical problem with the distances of this oracle

problem. We denote the θ and ψ found by this oracle problem as $\hat{\theta}, \hat{\psi}$. Formally, this upper bound leads to

$$\begin{aligned}
\textsc{LearnerPathLength} &\leq \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \hat{\theta}^t); \hat{\psi}^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \hat{\theta}^{t+1}); \hat{\psi}^{t+1})\|_2 \\
&\leq \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \hat{\theta}^t); \hat{\psi}^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^{t-1}; \hat{\theta}^{t+1}); \hat{\psi}^{t+1})\|_2 \\
&\quad + \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} g(r(\mathbf{x}^{t-1}; \hat{\theta}^{t+1}); \hat{\psi}^{t+1}) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \hat{\theta}^{t+1}); \hat{\psi}^{t+1})\|_2 \\
&\overset{(a)}{\leq} \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \hat{\theta}^t); \hat{\psi}^t) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^{t-1}; \hat{\theta}^t); \hat{\psi}^t)\|_2 \\
&\quad + \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} g(r(\mathbf{x}^{t-1}; \hat{\theta}^{t+1}); \hat{\psi}^{t+1}) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \hat{\theta}^{t+1}); \hat{\psi}^{t+1})\|_2 \\
&\overset{(b)}{\leq} 2\mu \sum_{t=1}^{T} \|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2,
\end{aligned} \tag{45}$$

as the consequence of the fact that oracle problem solves all gradients perfectly (i.e. $\nabla_{\mathbf{x}} \tilde{F}(\mathbf{x}^t, \xi^t) = \nabla_{\mathbf{x}} g(r(\mathbf{x}; \theta^i); \psi^i)$ for all $i > t$) in (a), and the functions are $\mu-$smooth in (b). Using the fact that gradient norms are bounded,

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^t) \leq 8\mu L \sum_{t=1}^{T} \|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2 + 2L. \tag{46}$$

In order to extend this result to the Δ^t, we use the smoothness of the function as

$$\begin{aligned}
\Delta^t &= \left(\mathbf{s}^{t\intercal}[\nabla_{\mathbf{x}} \hat{F}(\mathbf{x}^t, \xi) - \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t)]\right)^2 \\
&\leq \left(\mathbf{s}^{t\intercal} \nabla_{\mathbf{x}} \hat{F}(\mathbf{x}^t, \xi) - \frac{y(\mathbf{x}^t, \mathbf{s}^t, \xi^t)}{2\delta}\right)^2 + \left(\frac{y(\mathbf{x}^t, \mathbf{s}^t, \xi^t)}{2\delta} - \mathbf{s}^{t\intercal} \nabla_{\mathbf{x}} g(r(\mathbf{x}^t; \theta^t); \psi^t)\right)^2 \\
&\leq \left(\mathbb{E}_{\mathbf{v} \in \mathbb{B}^d}\left[\mathbf{s}^{t\intercal} \nabla_{\mathbf{x}} F(\mathbf{x}^t + \delta \mathbf{v}, \xi) - \frac{y(\mathbf{x}^t, \mathbf{s}^t, \xi^t)}{2\delta}\right]\right)^2 + \mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^t) \\
&\leq \mu^2 \delta^2 + \mathcal{L}(\mathbf{s}^t, \mathbf{x}^t, \xi^t, \theta^t, \psi^t)
\end{aligned} \tag{47}$$

By combining (47) and (46), we state

$$\sum_{t=1} \Delta^t \leq 8\mu L \sum_{t=1}^{T} \|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2 + \mu^2 \delta^2 T + 2L. \tag{48}$$

C.3.4 Proof of the Theorem

We combine the aforementioned three arguments to state the final sample complexity of our method. Our analysis of SGD with bias from (36) combined with the definition of the Δ^t gives the following bound on the sample complexity.

$$\frac{1}{T} \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 \leq \frac{\Omega}{\alpha T} + \frac{\mu \alpha \mathbf{V_g}}{2} + \frac{\Omega}{T} \sum_{t=1}^{T} \sqrt{d \mathbb{E}[\Delta^t]} + \delta^2 \mu^2. \tag{49}$$

Using the concavity of the square root function with Jensen's inequality, we can convert this bound to

$$\frac{1}{T} \sum_{t=1}^{T} \|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|_2^2 \leq \frac{\Omega}{\alpha T} + \frac{\mu \alpha \mathbf{V_g}}{2} + \Omega \sqrt{d} \sqrt{\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\Delta^t]} + \delta^2 \mu^2. \tag{50}$$

Next, we will bound $\sqrt{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta^t]}$ using (42). For simplicity, we will analyze two cases ($\sum_{t=1}^{T}\Delta^t \leq 1$) and ($\sum_{t=1}^{T}\Delta^t > 1$) seperately.

Case 1, $\sum_{t=1}^{T}\Delta^t \leq 1$: We substitute this bound directly in (42). With probability $1-4\gamma\ln(T)$,

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta^t] \leq \frac{1}{T} + \frac{2}{T}\sqrt{\frac{2L^2+d}{d}}\sqrt{\ln(1/\gamma)} + \max\left\{\frac{8L^2+4d}{dT}, 6L^2\left(\frac{1+d}{dT}\right)\right\}\ln(1/\gamma) \quad (51)$$

Relaxing the upper bound with the fact that dimension is greater than 1,

$$\sqrt{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta^t]} \leq \sqrt{\frac{c_1}{T}} \quad (52)$$

where $c_1 = 1 + \sqrt{(8L^2+4)\ln(1/\gamma)} + \max\{8L^2+4, 12L^2\}\ln(1/\gamma)$.

Case 2, $\frac{1}{T}\sum_{t=1}^{T}\Delta^t > 1$: Using the fact that $\sqrt{x} < x$ for $x > 1$ and $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ as well as $d \geq 1$, we can state that with probability $1-4\gamma\ln(T)$,

$$\sqrt{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta^t]} \leq \sqrt{\frac{c_2}{T}} + c_3\sqrt{\frac{1}{T}\sum_{t=1}^{T}\Delta^t} \quad (53)$$

where $c_2 = \max\{8L^2+4, 12L^2\}\ln(1/\gamma)$ and $c_3 = 1 + \sqrt{2\sqrt{(2L^2+1)\ln(1/\gamma)}}$. Combining this with the bound (48) and $\sqrt{x} \leq x$ for $x > 1$,

$$\sqrt{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta^t]} \leq \sqrt{\frac{c_2}{T}} + \frac{2c_3L}{T} + c_3(\mu^2\delta^2 + 8\mu\alpha\mathbf{V_g}) \quad (54)$$

We combine two cases and substitute it in (50). The final sample complexity is,

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|^2 \leq \frac{\Omega}{\alpha T} + \mu\alpha\mathbf{V_g}\left(\frac{1}{2} + 8c_3\Omega\sqrt{d}\right) + \mu^2\delta^2\left(1+\Omega c_3\sqrt{d}\right) + \frac{2c_3\Omega L\sqrt{d}}{T} + \sqrt{\frac{c_{1,2}d}{T}}. \quad (55)$$

where $c_{1,2} = \sqrt{\Omega}\max\{c_1, c_2\}$. We minimize with respect to α^t and substitute it.

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|^2 \leq \sqrt{\frac{2\mu\mathbf{V_g}\Omega}{T}}\left(1+\sqrt{4c_3\Omega d}\right) + \mu^2\delta^2\left(1+\Omega c_3\sqrt{d}\right) + \frac{2c_3\Omega L\sqrt{d}}{T} + \sqrt{\frac{c_{1,2}d}{T}}. \quad (56)$$

Before we solve for δ, we bound $\mathbf{V_g}$ by choosing $\beta = 1/d$ as,

$$\mathbb{E}[\mathbf{V_g}] \leq \mathbb{E}\left[\left(\frac{1}{d}\mathbf{g}_e + \left(1-\frac{1}{d}\right)\mathbf{g}_m\right)\right] \leq 4L^2n^2 + \frac{4n^2V_F}{\delta^2}. \quad (57)$$

Next, we solve δ to obtain the statement of the theorem. With probability $1-4\gamma\ln(T)$,

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}}f(\mathbf{x}^t)\|^2 \leq \frac{k_1d^{\frac{1}{2}}}{T} + \frac{k_2d^{\frac{1}{2}} + k_3n + k_4nd^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{k_5n^{\frac{2}{3}} + k_6d^{\frac{1}{2}}n^{\frac{2}{3}}}{T^{\frac{1}{3}}} \quad (58)$$

where $k_1 = 2c_3\Omega L$, $k_2 = \sqrt{c_{1,2}}$, $k_3 = 2L\sqrt{2\mu\Omega}$, $k_4 = 4L\Omega\sqrt{\mu c_3}$, $k_5 = 3(2\Omega V_F)^{1/3}$, and $k_6 = k_5(3\Omega c_3)$. □

C.4 USEFUL ELEMENTARY RESULTS

C.4.1 EXPECTATION OF $(\mathbf{s}^\intercal \mathbf{v})^2$ WHEN $\mathbf{s}$ IS CHOSEN UNIFORMLY FROM $\mathbb{S}^{d-1}$

Consider $\int F(\mathbf{O}\mathbf{e})\mu(\mathbf{O})$ where $\int [\cdot]\mu(\mathbf{O})$ is an integral over orthogonal matrices with Haar measure. If $\mathbf{e}$ is a unit vector, we can show that

$$\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}[F(\mathbf{s})] = \int F(\mathbf{O}\mathbf{e})\mu(\mathbf{O}). \tag{59}$$

Before we use this result, we define $\|\mathbf{v}\|^2$ as an integral over orthogonal matrices $\mathbf{O}$. Using orthogonality and cyclic property of the trace,

$$\mathbf{v}^\intercal\mathbf{v} = \text{Tr}(\mathbf{v}\mathbf{v}^\intercal) = \int \text{Tr}(\mathbf{v}\mathbf{v}^\intercal)\mu(\mathbf{O}) = \int \text{Tr}\left(\mathbf{O}\mathbf{O}^\intercal\mathbf{v}\mathbf{v}^\intercal\right)\mu(\mathbf{O}) = \int \text{Tr}\left(\mathbf{O}^\intercal\mathbf{v}\mathbf{v}^\intercal\mathbf{O}\right)\mu(\mathbf{O}). \tag{60}$$

Since the indentity matrix is sum of outer products of one hot vectors $\mathbf{e}_i$ as $\mathbf{I} = \sum_i \mathbf{e}_i\mathbf{e}_i^\intercal$,

$$\int \text{Tr}\left(\mathbf{O}^\intercal\mathbf{v}\mathbf{v}^\intercal\mathbf{O}\right)\mu(\mathbf{O}) = \int \text{Tr}\left(\sum_i \mathbf{e}_i\mathbf{e}_i^\intercal\mathbf{O}^\intercal\mathbf{v}\mathbf{v}^\intercal\mathbf{O}\right)\mu(\mathbf{O}) = \sum_i \int \text{Tr}\left(\mathbf{e}_i^\intercal\mathbf{O}^\intercal\mathbf{v}\mathbf{v}^\intercal\mathbf{O}\mathbf{e}_i\right)\mu(\mathbf{O}). \tag{61}$$

Using (59), we can further show

$$\sum_i \int \text{Tr}\left(\mathbf{e}_i^\intercal\mathbf{O}^\intercal\mathbf{v}\mathbf{v}^\intercal\mathbf{O}\mathbf{e}_i\right)\mu(\mathbf{O}) = \sum_i \mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}[\text{Tr}(\mathbf{s}^\intercal\mathbf{v}\mathbf{v}^\intercal\mathbf{s})] = d\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}[(\mathbf{s}^\intercal\mathbf{v})] \tag{62}$$

Hence, combining all,

$$\mathbb{E}_{\mathbf{s}\in\mathbb{S}^{d-1}}[(\mathbf{s}^\intercal\mathbf{v})] = \frac{\|\mathbf{v}\|^2}{d}. \tag{63}$$

C.4.2 BOUNDING THE QUADRATIC FORM

We need to bound s when the following quadratic inequality is correct;

$$s \leq r + \max\{2a\sqrt{s}, 6bc\}c. \tag{64}$$

We first consider the max as two separate options. Given the original inequality, one of the following is correct;

$$s \leq r + 2ac\sqrt{s}, \qquad or \qquad s \leq r + 6bc^2. \tag{65}$$

For the first case, $(\sqrt{s})^2 - 2ac\sqrt{s} - r \leq 0$. Using the quadratic formula, $\sqrt{s}$ is smaller than the largest root as $\sqrt{s} \leq ac + \sqrt{a^2c^2 + r}$. Hence,

$$s = (\sqrt{s})^2 \leq (ac + \sqrt{a^2c^2 + r})^2 = a^2c^2 + a^2c^2 + r + 2ac\sqrt{a^2c^2 + r} \leq 4a^2c^2 + 2ac\sqrt{r} + r \tag{66}$$

Combining the resulting bound with the other option gives the final bound as,

$$s \leq r + 2ac\sqrt{r} + \max\{4a^2, 6b\}c^2 \tag{67}$$