Derivative-Free Meta-Blackbox Optimization on Manifold

Bilgehan Sel

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Ming Jin, Chair
Ruoxi Jia
Naren Ramakrishnan

May 2, 2024
Blacksburg, Virginia

Keywords: Meta Optimization, Zeroth Order Search, Meta Reinforcement Learning

Derivative-Free Meta-Blackbox Optimization on Manifold

Bilgehan Sel

ABSTRACT

Solving a sequence of high-dimensional, nonconvex, but potentially similar optimization problems poses a significant computational challenge in various engineering applications. This thesis presents the *first* meta-learning framework that leverages the shared structure among sequential tasks to improve the computational efficiency and sample complexity of derivative-free optimization. Based on the observation that most practical high-dimensional functions lie on a latent low-dimensional manifold, which can be further shared among problem instances, the proposed method jointly learns the meta-initialization of a search point and a meta-manifold. This novel approach enables the efficient adaptation of the optimization process to new tasks by exploiting the learned meta-knowledge. Theoretically, the benefit of meta-learning in this challenging setting is established by proving that the proposed method achieves improved convergence rates and reduced sample complexity compared to traditional derivative-free optimization techniques. Empirically, the effectiveness of the proposed algorithm is demonstrated in two high-dimensional reinforcement learning tasks, showcasing its ability to accelerate learning and improve performance across multiple domains. Furthermore, the robustness and generalization capabilities of the meta-learning framework are explored through extensive ablation studies and sensitivity analyses. The thesis highlights the potential of meta-learning in tackling complex optimization problems and opens up new avenues for future research in this area.

Derivative-Free Meta-Blackbox Optimization on Manifold

Bilgehan Sel

GENERAL AUDIENCE ABSTRACT

Optimization problems are ubiquitous in various fields, from engineering to finance, where the goal is to find the best solution among a vast number of possibilities. However, solving these problems can be computationally challenging, especially when the search space is high-dimensional and the problem is nonconvex, meaning that there may be multiple locally optimal solutions. This thesis introduces a novel approach to tackle these challenges by leveraging the power of meta-learning, a technique that allows algorithms to learn from previous experiences and adapt to new tasks more efficiently.

The proposed framework is based on the observation that many real-world optimization problems share similar underlying structures, even though they may appear different on the surface. By exploiting this shared structure, the meta-learning algorithm can learn a low-dimensional representation of the problem space, which serves as a guide for efficiently searching for optimal solutions in new, unseen problems. This approach is particularly useful when dealing with a sequence of related optimization tasks, as it allows the algorithm to transfer knowledge from one task to another, thereby reducing the computational burden and improving the overall performance.

The effectiveness of the proposed meta-learning framework is demonstrated through rigorous theoretical analysis and empirical evaluations on challenging reinforcement learning tasks. These tasks involve high-dimensional search spaces and require the algorithm to adapt to

changing environments. The results show that the meta-learning approach can significantly accelerate the learning process and improve the quality of the solutions compared to traditional optimization methods.

To loved ones

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Ming Jin, for his invaluable guidance, unwavering support, and constant encouragement throughout my master's journey. His expertise, patience, and dedication have been instrumental in shaping my research and helping me overcome the challenges I faced along the way. I am truly grateful for the opportunity to have worked under his supervision.

I would like to extend my gratitude to Prof. Ruoxi Jin and Prof. Naren Ramakrishnan, who gave crucial guidance during the writing of this thesis. I also want to thank Dr. Yuhao Ding for his incredible guidance in theory.

To my loved ones, I am forever indebted for their unconditional love, understanding, and support. Their constant belief in me has been a source of strength and motivation, especially during the most challenging times. I am truly blessed to have such a wonderful support system, and I could not have achieved this milestone without them by my side.

Contents

List of Figures

List of Abbreviations

ADAM Adaptive Moment Estimation

CAML Curvature-aware Meta-learning

DFO Derivative-Free Optimization

FTL-BTL Follow The Leader - Be The Leader

FTPL Follow-The-Perturbed Leader

FTPML Follow-The-Perturbed Meta Leader

FTRL Follow-The-Regularized Leader

FTRML Follow-The-Regularized Meta Leader

HOCA Human-level Hyperparameter Optimization

LMRS Learned Manifold Random Search

MAML Model-Agnostic Meta-Learning

Meta-LMRS Meta-Learned Manifold Random Search

NHO Non-Differentiable Hyperparameter Optimization

PCA Principal Component Analysis

RL Reinforcement Learning

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

t-SNE t-distributed Stochastic Neighbor Embedding

TARS Task-Averaged Regret on Stationarity

Chapter 1

Introduction

1.1 Background and Motivation

Solving a sequence of optimization problems with similar structures often arises in engineering applications. For instance, in time-varying optimization, the problem usually has a fixed structure, but its instantiations depend on time-varying data obtained at predefined sampling times [1]. Real-time computation of the solution is often required for problems in power systems [2, 3, 4, 5], communication systems [6, 7], online learning [8, 9], and signal processing [10]. Similarly, in multi-task learning, multiple loss functions corresponding to similar tasks are jointly optimized for improved generalization performance [11]. These problems share a common characteristic: they involve solving a series of related optimization tasks, each with its own unique data or parameters, but sharing an underlying structure or objective.

Despite advances in computing, resolving each instance of single-task optimization separately can be impractical. For example, a reinforcement learning (RL) algorithm can potentially take millions of steps before converging to a good policy from scratch, which is restrictive in real-world settings [12]. In many applications, such as robotics or autonomous vehicles, the environment may change over time, requiring the RL agent to adapt quickly to new conditions. If the agent had to learn from scratch each time the environment changed, it would be prohibitively slow and inefficient.

A promising direction, as exemplified by meta-learning, or learning-to-learn, is to leverage prior and similar experiences to accelerate optimization [13]. The goal of meta-learning is to learn a learning algorithm that can quickly adapt to new tasks by leveraging knowledge from previous tasks. This is analogous to how humans can learn new skills more quickly by drawing on their past experiences and knowledge. Notwithstanding the empirical success [14], most efforts to analyze initialization-based meta-learning focus on the setting with decomposable single-task loss functions that are often convex for theoretical tractability [15, 16, 17]; nonconvex single-task settings are studied usually for multi-task representation learning [18, 19, 20, 21]. Therefore, the first challenge is to analyze the theoretical benefits of meta-learning for nonconvex optimizations with shared structures.

Furthermore, while gradient-based meta-learning has gained traction, there are many applications such as hyperparameter tuning [22], reinforcement learning [23], simulation-based optimization [24], and generating adversarial examples [25], which resist access to the gradient, let alone the Hessian of the objective function. In general, derivative-free optimization (DFO) has been employed for blackbox optimizations [26]. DFO methods are particularly useful when the objective function is not differentiable, or when computing gradients is prohibitively expensive. Examples of DFO methods include evolutionary algorithms, particle swarm optimization, and Bayesian optimization. However, contrary to first-order techniques, where the convergence rates are independent of the problem dimensionality, DFO methods, such as Bayesian optimization [27] and random search [28], often scale poorly with the dimensionality [29]. This is because the number of function evaluations required to explore the search space grows exponentially with the number of dimensions, a phenomenon known as the "curse of dimensionality". Adaptivity to the latent low-dimensional structure of the search space has been pursued by subspace methods [30, 31]. In particular, [32] proposed the learned manifold random search (LMRS), which learns a latent manifold while performing

the optimization. The second challenge, therefore, is to enable meta-learning of DFO that exploits the latent structure of a potentially high-dimensional problem.[1]

1.2 Contributions

In view of the aforementioned challenges, the first framework that leverages the shared structure among potentially high-dimensional, nonconvex, but similar problem instances to improve computational efficiency and sample complexity of repeated application of DFO is proposed in this thesis. Specifically, using LMRS as an exemplary base algorithm, Meta-LMRS is developed that adaptively and jointly learns a meta-initialization of a search point and a meta-manifold. Preliminary results on two high-dimensional RL problems demonstrate that Meta-LMRS facilitates each task to be solved with a small handful of iterations. For analysis, two notions of similarity among optimization tasks are introduced, namely V^*_{init} and V^*_{manifold}, which measure closeness based on the initial point and optimization landscape as captured by some shared manifold, respectively. A key theoretical benefit of the proposed Meta-LMRS is established. Notably, the task-averaged regret on stationarity (Def. 3.3) can be bounded as $\mathcal{O}\left(M^{-\frac{1}{2}} + \max\left(V_{\text{init}}, c + V^{\text{manifold}}\right)\right)$, where M is the number of tasks and c is some constant. This bound improves upon single-task learning when M is large enough or $V\text{init}^*$ and V^*_{manifold} are small enough (the tasks are sufficiently similar). To contextualize the contribution, the proposed technique can also be viewed as a step forward for semi-amortized models over a latent space without requiring gradients [34, Sec. 3.2.2].

[1] Note that here the aim is to learn a low-dimensional manifold to improve sample complexity or computational efficiency, which is conceptually different from the classical field of manifold optimization [33], where the manifold is formed by predetermined constraints.

Chapter 2

Literature Review and Preliminaries

2.1 Related Research

2.1.1 Meta-Learning

Meta-learning, also known as "learning-to-learn", has emerged as a promising approach to enable artificial intelligence systems to adapt quickly to new tasks and environments. By leveraging knowledge from previous experiences, meta-learning algorithms aim to improve the efficiency and generalization capabilities of machine learning models. This literature review focuses on recent advances in meta-learning, covering various aspects such as learning initial conditions, hyperparameters, step directions, stepsizes, and manifolds.

Learning Initial Conditions

One of the key aspects of meta-learning is learning good initial conditions for the model parameters. [14] proposed the Model-Agnostic Meta-Learning (MAML) algorithm, which learns an initialization that can be quickly adapted to new tasks with just a few gradient steps. The algorithm optimizes the initial parameters such that they can be easily fine-tuned for each task, enabling fast adaptation. An extension of MAML, called Meta-SGD, was introduced by [35], which learns both the initialization and the learning rates for each

parameter, further enhancing the adaptability of the model.

Learning Hyperparameters

Hyperparameter optimization is crucial for achieving optimal performance in machine learning models. The Human-level Hyperparameter Optimization framework, proposed by [36], learns to optimize hyperparameters by leveraging a population-based training approach. The framework maintains a population of models with different hyperparameter configurations and evolves them over time to discover optimal settings. [37] introduced Non-Differentiable Hyperparameter Optimization, a meta-learning approach that learns to optimize hyperparameters using non-differentiable optimization techniques, such as Bayesian optimization and evolutionary strategies.

Learning Step Directions and Stepsizes

In addition to learning initial conditions and hyperparameters, meta-learning algorithms can also learn the step directions and stepsizes for optimization. Meta-SGD, proposed by [35], learns a preconditioning matrix that determines the step direction and magnitude for each parameter update. This allows the algorithm to adapt the optimization process to the specific characteristics of each task. MetaTrace, introduced by [38], is a meta-learning algorithm that learns stepsizes for gradient descent using a trace of the optimization trajectory. Bilevel optimization approaches to learn the stepsizes for gradient-based optimization algorithms were proposed by [39] and [40]. Learning stepsizes in the context of non-differentiable hyperparameter optimization was also explored by [37].

Learning Manifolds

Recent work has explored the idea of learning manifolds for meta-learning. The Curvature-aware Meta-learning (CAML) framework, proposed by [41], learns a Riemannian manifold that captures the geometric structure of the task space. By optimizing the model parameters on this learned manifold, CAML enables more efficient adaptation to new tasks. The manifold is learned using a Riemannian optimization approach, which takes into account the curvature information of the task space.

Recurrent Meta-Learning

Another approach to meta-learning is to use recurrent neural networks (RNNs) to embed previous task experiences. RL^2, a meta-learning framework for reinforcement learning, was proposed by [42]. It uses an RNN to learn a policy that can adapt to new tasks by incorporating information from previous tasks. The RNN acts as a memory that stores and retrieves relevant knowledge from past experiences, enabling more efficient learning of new tasks.

Theoretical Analysis

While most meta-learning methods have been empirically successful, theoretical understanding of their behavior is still an active area of research. The convergence properties of online meta-learning algorithms in the convex setting were analyzed by [15] and [16], providing regret bounds and highlighting the benefits of meta-learning over single-task learning. [17] studied the sample complexity of gradient-based meta-learning algorithms and derived provable guarantees for their performance.

In the nonconvex setting, theoretical analysis has primarily focused on multitask representation learning. [20] analyzed the few-shot learning performance of meta-learning algorithms

that learn a shared representation across tasks. They derived sample complexity bounds and highlighted the importance of task diversity for effective meta-learning. The role of task diversity in transfer learning was studied by [21], providing theoretical insights into the benefits of meta-learning in the presence of diverse tasks.

2.1.2 Derivate-Free Optimization

Derivative-Free Optimization (DFO) has an extensive body of literature, with comprehensive reviews provided by [43], [44], and [26]. This literature review focuses on DFO approaches related to the current work, including local gradient estimation, smoothing techniques, and random search methods.

Local Gradient Estimation

Local gradient estimation is a key component of many DFO algorithms. [45] proposed a linear interpolation-based approach for estimating gradients in DFO. The complexity of gradient estimation in high-dimensional spaces was studied by [31], who introduced techniques for efficient gradient approximation. [46] proposed a guided approach for estimating gradients in the context of reinforcement learning, while [47] explored the use of evolution strategies for gradient estimation.

Smoothing Techniques

Smoothing techniques are often employed in DFO to handle non-smooth objective functions. [48] introduced an online convex optimization algorithm that uses a smoothing technique to estimate gradients. [49] proposed a local smoothing approach for DFO, which constructs a smooth approximation of the objective function in the neighborhood of the current iterate.

Random Search Methods

Random search methods are a popular class of DFO algorithms that estimate the directional derivative by performing perturbations to the current iterate [50]. These methods have been widely studied in the literature, with convergence analysis established for various settings. [51] analyzed the convergence properties of random search methods for convex and nonconvex functions, providing convergence rates and complexity bounds. [29] studied the convergence of stochastic zeroth-order methods for nonconvex optimization, while [52] investigated the optimal convergence rates for nonsmooth convex optimization using random search methods.

Sample Complexity Bounds

Several works have focused on understanding the sample complexity of DFO algorithms. [53] derived optimal rates of convergence and sample complexity bounds for stochastic convex optimization problems in the zeroth-order setting. [54] established lower bounds on the query complexity of convex optimization algorithms that rely solely on function evaluations.

These works collectively contribute to the rich literature on DFO, providing insights into various aspects such as gradient estimation, smoothing techniques, random search methods, convergence analysis, and sample complexity bounds. The advancements in DFO have enabled the development of efficient optimization algorithms for a wide range of applications where explicit gradient information is unavailable or computationally expensive to obtain.

2.1.3 High-Dimensional Analysis with Low-Dimensional Structure

High-dimensional optimization problems often suffer from the curse of dimensionality, which makes gradient estimation and optimization challenging. To alleviate this issue and improve

the efficiency of optimization algorithms, several approaches have been developed that exploit the inherent low-dimensional structure present in many high-dimensional problems [55].

Direct Projection Methods

One class of approaches for handling high-dimensional problems is based on directly projecting the directional primitives into a lower-dimensional space. These methods often employ variants of principal component analysis (PCA) to identify the most informative directions in the search space. [56] proposed a generalization of PCA for dimensionality reduction in the context of robot learning. [57] used a dimensionality reduction technique based on PCA to efficiently learn motor skills in high-dimensional spaces. [58] extended this idea to contextual policy search, where the low-dimensional space is learned in a context-dependent manner.

Guided Evolutionary Search

Another approach to tackle high-dimensional problems is through guided evolutionary search. [59] proposed a guided evolutionary strategy that uses a low-dimensional subspace to guide the search process in high-dimensional spaces. [60] introduced a novel evolutionary algorithm that combines dimensionality reduction techniques with covariance matrix adaptation to efficiently explore high-dimensional search spaces.

Active Subspace Methods

Active subspace methods are a class of techniques that aim to identify a low-dimensional linear subspace that captures the most important directions in the high-dimensional space. [31] studied the complexity of active subspace methods for high-dimensional optimization

and proposed efficient algorithms for subspace estimation. [46] used active subspace methods to guide the search process in high-dimensional reinforcement learning problems.

Learning Low-Dimensional Manifolds

Recent work has focused on learning low-dimensional manifolds that capture the intrinsic structure of high-dimensional problems. [32] proposed a method for learning a low-dimensional manifold using neural networks, which can be used to guide the optimization process in high-dimensional spaces. This idea is extended to the meta-learning setting in the present work, where the low-dimensional manifold is learned across multiple related tasks to improve the efficiency of optimization algorithms.

The approaches discussed above have shown promising results in mitigating the curse of dimensionality and enabling efficient optimization in high-dimensional problems. By exploiting the low-dimensional structure present in many real-world problems, these methods can significantly reduce the computational complexity and improve the convergence rates of optimization algorithms. The continued development and integration of these techniques are crucial for tackling the challenges posed by high-dimensional optimization problems in various domains.

2.2 Preliminaries

2.2.1 Problem Setup

An online sequence of tasks $\{\mathcal{T}_i\}_{i=1}^{M}$ that arrive sequentially is considered. Each task $\mathcal{T}_i$ is defined as solving a high-dimensional stochastic optimization problem, which is represented by the equation:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f_i(\mathbf{x}) = \mathbb{E}_\xi[F_i(\mathbf{x},\xi)],$$

where $\mathbf{x}$ is designated as the optimization variable and $f_i : \mathbb{R}^d \to \mathbb{R}$ is defined as the function of task i, which is expressed as an expectation over some noise variable ξ. In the context of DFO (Derivative-Free Optimization), instead of evaluating the gradients directly, only zeroth-order access to the objective function is available through the sampling operator $\square$. This means that $\square f_i(\mathbf{x}) \sim F_i(\mathbf{x},\xi)$ is considered a random variable for the input $\mathbf{x}$ and some noise variable ξ.

The goal of the presented approach is to learn meta-parameters $\theta \in \Theta$ that are capable of producing a good task-specific solution $\mathbf{x}_i$ after a few steps of random search. Specifically, $\mathcal{A}lg^t(\theta, \square f_i)$ is defined as performing t steps of DFO initialized with θ. As an example, if one step of random search is taken and θ corresponds to the initial point, the equation $\mathbf{x}_i \equiv \mathcal{A}lg^1(\theta, \square f_i) = \theta - \alpha\hat{g}_i$ is obtained, where α is designated as the stepsize and $\hat{g}_i$ is defined as the estimated gradient.

To estimate the gradient using function evaluations, the classical result by [48, 51] is recalled. The d-dimensional unit sphere and unit ball are denoted by $\mathbb{S}^{d-1}$ and $\mathbb{B}^d$, respectively, and ω is defined as a random vector sampled from the uniform distribution over $\mathbb{S}^{d-1}$. For a function $f : \mathbb{R}^d \to \mathbb{R}$, its δ-smoothed version is expressed as $\hat{f}(\mathbf{x}) = \mathbb{E}_{\mathbf{v}\sim\mathbb{B}^d}[f(\mathbf{x}+\delta\mathbf{v})]$. Then, the equation:

$$\hat{g}(\mathbf{x},\omega) = (F(\mathbf{x}+\delta\omega,\xi) - F(\mathbf{x}-\delta\omega,\xi))\,\omega \tag{2.1}$$

is considered an unbiased estimate of the gradient of the smoothed function $\mathbb{E}_{\xi,\omega\in\mathbb{S}^{d-1}}[\hat{g}(\mathbf{x},\omega)] = \frac{2\delta}{d}\nabla_{\mathbf{x}}\hat{f}(\mathbf{x})$.

The ultimate goal of gradient-based meta-learning is to find an inductive bias that enables the solving of a new instance of optimization f with a few steps of adaptation (t is small) by a DFO algorithm denoted as $\mathcal{A}lg^t(\theta, \square f)$.

It is important to note that the presented approach relies on the concept of meta-learning, where the aim is to learn meta-parameters that can be quickly adapted to new tasks with a limited number of optimization steps. By leveraging the zeroth-order information obtained through the sampling operator, the proposed method seeks to estimate the gradient and find an inductive bias that facilitates efficient optimization of new instances.

2.2.2 Assumptions

We make several assumptions about the stochastic function $F(\mathbf{x}, \xi)$ to establish a theoretical foundation for our analysis. First, we assume that the function is bounded, meaning that there exists a constant Ω such that $|F(\mathbf{x}, \xi)| \leq \Omega$ for all values of $\mathbf{x}$ and ξ. This assumption ensures that the function does not take on arbitrarily large values, which is necessary for the convergence analysis of optimization algorithms.

Next, we assume that the function is L-Lipschitz with respect to $\mathbf{x}$ for all values of ξ. This means that for any two points $\mathbf{x}_1$ and $\mathbf{x}_2$, the following inequality holds: $|F(\mathbf{x}_1, \xi) - F(\mathbf{x}_2, \xi)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$, where L is a positive constant known as the Lipschitz constant. The Lipschitz continuity assumption is commonly used in optimization theory to ensure that the function does not change too rapidly between nearby points.

In addition to being Lipschitz continuous, we also assume that the function is μ-smooth with respect to $\mathbf{x}$ for all values of ξ. This means that the gradient of the function, denoted by $\nabla_{\mathbf{x}} F(\mathbf{x}, \xi)$, is also Lipschitz continuous with a constant μ. Mathematically, this can be expressed as $\|\nabla_{\mathbf{x}} F(\mathbf{x}1, \xi) - \nabla_{\mathbf{x}} F(\mathbf{x}_2, \xi)\| \leq \mu\|\mathbf{x}_1 - \mathbf{x}_2\|$ for any two points $\mathbf{x}_1$ and $\mathbf{x}_2$.

The smoothness assumption is important for the convergence analysis of gradient-based optimization algorithms.

Furthermore, we assume that the stochastic function has uniformly bounded variance. This means that for all values of $\mathbf{x}$, the variance of the function with respect to the random variable ξ is bounded by a constant V_F. In mathematical notation, this can be expressed as $\mathbb{E}_\xi[(F(\mathbf{x},\xi) - f(\mathbf{x}))^2] \leq V_F$, where $f(\mathbf{x}) = \mathbb{E}_\xi[F(\mathbf{x},\xi)]$ is the expected value of the function with respect to ξ. The bounded variance assumption is crucial for the convergence analysis of stochastic optimization algorithms, as it ensures that the noise in the function evaluations does not overwhelm the optimization process.

In addition to the above assumptions, we also assume that given a specific value of ξ, the function $F(\cdot,\xi)$ lies on an n-dimensional manifold, where n is typically much smaller than the ambient dimension d (i.e., $n \ll d$). This low-dimensional manifold assumption is motivated by the observation that many real-world problems exhibit low-dimensional structure, even though they may be embedded in a high-dimensional space.

To parameterize this low-dimensional manifold, we assume that it can be defined via a nonlinear parametric family, such as a neural network. Specifically, we assume that there exist optimal parameters $\psi_r^\star$ and $\psi_h^\star$ such that $F(\mathbf{x},\xi) = h(r(\mathbf{x};\psi_r^\star);\psi_h^\star)$ for all $\mathbf{x} \in \mathbb{R}^d$, where $r(\cdot;\psi_r)$ is a mapping from the ambient space $\mathbb{R}^d$ to the low-dimensional manifold, and $h(\cdot;\psi_h)$ is a mapping from the low-dimensional manifold to the scalar output space. These assumptions are adopted by [32] in their work on learning low-dimensional manifolds for stochastic optimization.

For simplicity, we denote the concatenated parameter vector as $\psi = (\psi_r, \psi_h)$, which belongs to the parameter space Ψ. We assume that the parameter space has a bounded ℓ_∞ diameter, denoted by $D_\Psi = \sup_{\psi,\psi' \in \Psi} \|\psi - \psi'\|_\infty$. This assumption is necessary for the convergence

analysis of optimization algorithms in the parameter space.

Unless otherwise specified, we assume that the above assumptions hold for each task $i \in [M]$ in the context of multi-task learning. These assumptions provide a solid theoretical foundation for the analysis of stochastic optimization algorithms in the presence of low-dimensional structure and multiple tasks.

2.2.3 Learning to guide random search in a single task

The function of interest in many practical problems has been observed to lie on a low-dimensional nonlinear manifold. Based on this observation, Learned Manifold Random Search (LMRS) was proposed by [32] to minimize the function and learn the manifold jointly. A succinct review of LMRS is provided here for completeness (refer to Algorithm 1 for the pseudo-code).

Algorithm 1 Learned Manifold Random Search [32]

1: **for** $t = 1, \ldots, T$ **do**
2: $g_e^t = \textsc{GradEst}\,(x^t, \delta)$ [32, Alg. 1].
3: $g_m^t = \textsc{ManifoldGradEst}\,(x^t, J_q(x^t; \psi_r^t))$ [32, Alg. 2].
4: $g^t = (1-\beta) g_m^t + \beta g_e^t$
5: $x^{t+1} = x^t - \alpha g^t$
6: $\psi^{t+1} = \arg\min_\psi \sum_{k=1}^t \mathcal{L}(x^k, \omega^k, \psi) + \lambda \mathcal{R}(\psi)$
7: **end for**

Random search over a manifold. Let $f_\psi = h(r(\cdot; \psi_r); \psi_h)$. By applying the chain rule, the gradient $\nabla_{\mathbf{x}} f_\psi(\mathbf{x})$ can be expressed as $\mathbf{J}(\mathbf{x}; \psi_r) \nabla_{\mathbf{r}} h(\mathbf{r}; \psi_h)$, where $\mathbf{J}(\mathbf{x}; \psi_r) = \frac{\partial r(\mathbf{x}; \psi_r)}{\partial \mathbf{x}}$ and $\mathbf{r} = r(\mathbf{x}, \psi_r)$. In LMRS, the Jacobian $\mathbf{J}(\mathbf{x}; \psi_r)$ is first orthonormalized using the Gram-Schmidt procedure for numerical stability. Then, random search is performed in the column space of the resulting orthonormal matrix $\mathbf{J}_q(\mathbf{x}; \psi_r)$, which has a lower dimensionality compared to the full space. For each step, an n-dimensional vector $\tilde{\omega}$ is sampled uniformly from $\mathcal{S}^{n-1}$ and lifted to the input space via $\mathbf{J}_q(\mathbf{x}; \psi_r)\tilde{\omega}$. By applying the manifold Stokes' theorem and

using the lifted vector as a random direction, an unbiased estimate of the gradient of the smoothed function can be obtained (c.f., (2.1)):

$$\mathbb{E}_{\xi,\tilde{\omega}\in\mathcal{S}^{n-1}}[\hat{g}(\mathbf{x},\mathbf{J}_q(\mathbf{x};\psi_r)\tilde{\omega})] = \frac{2\delta}{n}\nabla_{\mathbf{x}}\tilde{f}_\psi(\mathbf{x}),$$

where $\tilde{f}_\psi(\mathbf{x}) = \mathbb{E}_{\tilde{\mathbf{v}}\sim\mathcal{B}^n}[f(\mathbf{x}+\delta\mathbf{J}_q(\mathbf{x};\psi_r)\tilde{\mathbf{v}})]$ represents the smoothed function. In each iteration of LMRS, exploration is added by sampling directions from both the manifold $\mathbf{g}_m$ and the full space $\mathbf{g}_e$. These directions are then mixed to obtain the final estimate $\mathbf{g} = (1-\beta)\mathbf{g}_m + \beta\mathbf{g}_e$.

Manifold learning. At each iteration, the projection of the gradient onto the chosen directions is observed, given by $y(\mathbf{x}^t,\omega^t) = f(\mathbf{x}^t + \delta\mathbf{J}_q(\mathbf{x}^t;\psi_r^t)\omega^t) - f(\mathbf{x}^t - \delta\mathbf{J}_q(\mathbf{x}^t;\psi_r^t)\omega^t)$. Consequently, the one-step loss can be defined as follows:

$$\mathcal{L}(\mathbf{x}^t,\omega^t,\psi^t) = \left(\frac{y(\mathbf{x}^t,\omega^t)}{2\delta} - \omega^{t\intercal}\nabla_{\mathbf{x}}h(r(\mathbf{x}^t;\psi_r^t);\psi_h^t)\right)^2.$$

The manifold parameters can be learned by minimizing the aforementioned loss along the trajectory, following an approach similar to the Follow the Regularized Leader (FTRL) algorithm [61]:

$$\psi^{t+1} = \arg\min_{\psi}\sum_{k=1}^{t}\mathcal{L}(\mathbf{x}^k,\omega^k,\psi) + \lambda\mathcal{R}(\psi), \tag{2.2}$$

where $\mathcal{R}(\psi) = \|\nabla_{\mathbf{x}}h(r(\mathbf{x}^t;\psi_r^t);\psi_h^t) - \nabla_{\mathbf{x}}h(r(\mathbf{x}^t;\psi_r);\psi_h)\|_2^2$ serves as a temporal smoothness regularizer that penalizes sudden changes in the gradient estimates. In the theoretical analysis, it is assumed that (2.2) can be solved optimally. Although this assumption may appear strong, it is supported by the experimental results presented in [32], as neural networks can have high capacity. It is worth noting that further relaxation is possible by adopting the

result from [62].

Chapter 3

Meta-Learning Learned Manifold Random Search

In this chapter, we present a novel approach to meta-learning for derivative-free optimization (DFO) that leverages the shared structure among potentially high-dimensional, nonconvex, but similar problem instances to improve computational efficiency and sample complexity. Building upon the Learned Manifold Random Search (LMRS) algorithm introduced in the previous chapter, we propose Meta-LMRS in Algorithm 2, a meta-learning framework that adaptively and jointly learns a meta-initialization of a search point and a meta-manifold.

Algorithm 2 Meta - Learned Manifold Random Search (Meta-LMRS)

1: **for** $i = 1, \ldots, M$ **do**
2: **for** $t = 1, \ldots, T$ **do**
3: $x_i^{t+1}, \psi_i^{t+1} = \text{LMRS}\,(x_i^t, \psi_i^t)$
4: **end for**
5: Evaluate $U_i(x_i^1)$ and $U_i'(\psi_i^1)$ using (3.5) and (3.9), respectively
6: Update the next task parameter and manifold initialization

$$\kappa_{i+1} = \text{FTRL}\big(\{U_m^{\text{sim}}\}_{m<i}\big), x_{i+1}^1 = \text{FTR}\textsc{m}\text{L}(\{\kappa_m U_m\}_{m<i}), \psi_{i+1}^1 = \text{FTP}\textsc{m}\text{L}(\{\kappa_m U_m'\}_{m<i})$$

7: **end for**

We begin by analyzing the single-task performance guarantee of LMRS, revealing the dependence of the convergence rate on both the initial search point and the manifold parameter. These insights lay the foundation for the development of Meta-LMRS.

Informally, the relaxed version of the problem we would like to solve can be seen as follows:

$$\min_{x_0 \in \mathbf{R}^n, C \subset \mathbf{R}^n} \sum_{i=1}^{M} \min_{x \in x_0 + C} f_i(x), \tag{3.1}$$

$$\text{s.t.} \quad C \text{ lies in } d\text{-dimensional unit ball.}$$

We want to find a good initialization, together with a lower dimensional manifold that enables us to achieve parameters that are good for individual tasks. This implies that, the initial points x_0 rely on manifold dimension d. This is a crucial observation, since it can be thought that for a given d, an initialization found by other means such as meta-learning with derivatives or by considering lower or higher dimensions than d, can still be perform well. However, initializations need to be find together with their manifolds and subsequently the allowed dimension. We will later show Meta-LMRS algorithm works well while simply utilizing LMRS on individual tasks with initializations found by other means do not in Chapter 4.

Next, we introduce the concept of task-averaged regret on stationarity (TARS) as a performance metric for meta-learning in the context of DFO. We discuss the challenges associated with minimizing TARS and highlight the importance of task similarity in achieving improved performance.

To address these challenges, we propose a two-pronged approach. First, we introduce the Follow-The-Regularized Meta Leader (FTRML) algorithm for meta-learning the initial search point. We define a novel task-similarity measure based on the initial point and optimization trajectories, and provide a theoretical bound on TARS for FTRML.

Second, we develop the Follow-The-Perturbed Meta Leader (FTPML) algorithm for meta-learning the search manifold. We introduce a task-similarity measure based on the manifold

parameters and single-task optimization procedure, and derive a theoretical bound on TARS for FTPML.

Finally, we propose a joint meta-learning approach that combines FTRML and FTPML to adaptively learn both the initial search point and the manifold parameters. We provide a theoretical analysis of the proposed joint approach, demonstrating its potential to achieve improved performance compared to single-task learning.

The theoretical developments in this chapter lay the groundwork for the experimental evaluation of Meta-LMRS on high-dimensional reinforcement learning tasks, which will be presented in Chapter 4.

3.1 Single Task Performance Guarantee

The dependence of the convergence rate on the initial parameter $\mathbf{x}^1$ is revealed by the following result.

Lemma 3.1. *Consider running learned manifold random search in Algorithm 2 for T steps. Let $k_e = 1$ and $k_m = 1$ for simplicity and set $\alpha = c_0 T^{-\frac{1}{2}}$, $\beta = d^{-1}$, and $\delta = (2n)^{\frac{1}{3}}(V_F\Omega)^{\frac{1}{6}}\mu^{-\frac{1}{2}}T^{-\frac{1}{6}}$. Then, with probability $1 - 4\gamma\ln(T)$,*

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|^2 \leq \frac{c_1}{\sqrt{T}}\sqrt{\sum_{t=2}^{T}\|\mathbf{x}^t - \mathbf{x}^1\|_2} + \frac{c_2}{T^{\frac{1}{2}}} + \frac{c_3}{T^{\frac{1}{3}}}, \tag{3.2}$$

where $c_0 = \left(L^2 n^2 \mu \Omega^{-1} + 2^{-\frac{2}{3}} n^{\frac{4}{3}} V_F^{\frac{2}{3}} \mu^2 \Omega^{-\frac{4}{3}} T^{\frac{1}{3}}\right)^{-\frac{1}{2}}$, $c_1 = \Omega\sqrt{dc_1'}$, $c_2 = 4Ln\sqrt{\Omega\mu} + \Omega\sqrt{d}(\sqrt{c_5'} + \sqrt{c_1' c_4'})$, $c_3 = 5\mu(V_F\Omega)^{\frac{1}{3}} n^{\frac{2}{3}}$, with $\{c_j'\}_{j\in[5]}$ defined in (A.10).

Proof. See Appendix A. □

It is important to note that, strictly speaking, the above result does not constitute a standard result on convergence, as the bound on the right-hand side of (3.2) is dependent on the initial parameter $\mathbf{x}^1$ and the random trajectory data $\{\mathbf{x}^t\}_{t=1}^T$. This dependence is the key difference between this result and the one presented in [32, Theorem 1]. The development of this result is a crucial step towards enabling principled meta-initialization. It is possible to replace the term $\sqrt{\sum_{t=1}^T \|\mathbf{x}^t - \mathbf{x}^1\|_2}$ in (3.2) with $\sqrt{\|\mathbf{x}^T - \mathbf{x}^1\|_2}$ (to reflect the relation between the initial point and the final point) with a slight modification of the constants. In this case, the rate of $\mathcal{O}(T^{-\frac{1}{3}})$ can be recovered, as shown in [32, Theorem 1]. However, the dependence on the trajectory is introduced in this result for the purpose of trajectory-based meta-learning; further discussion on task-similarity can be found in Sec. 3.3.

The dependence of the convergence rate on the manifold parameter is revealed by the following result.

Lemma 3.2. *Consider running learned manifold random search in ALgorithm 2 for T steps. Let $k_e = 1$ and $k_m = 1$ for simplicity and set $\alpha = b_0 T^{-\frac{1}{2}}$, $\beta = d^{-1}$, and $\delta = 2^{\frac{1}{2}} n^{\frac{1}{3}} (V_F \Omega)^{\frac{1}{6}} \mu^{-\frac{1}{2}} T^{-\frac{1}{6}}$. Then, with probability $1 - 4\gamma \ln(T)$, it is shown that*

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla_{\mathbf{x}} f(\mathbf{x}^t)\|^2 \leq \frac{b_1}{\sqrt{T}} \sqrt{\sum_{t=2}^{T} \|\nabla_{\mathbf{x}} h(r(\mathbf{x}^t; \psi_r^t); \psi_h^t) - \nabla_{\mathbf{x}} h(r(\mathbf{x}^t; \psi_r^1); \psi_h^1)\|_2} + \frac{b_2}{T^{\frac{1}{2}}} + \frac{b_3}{T^{\frac{1}{3}}}. \tag{3.3}$$

where $b_0 = \left(L^2 n^2 \mu \Omega^{-1} + n^{\frac{4}{3}} V_F^{\frac{2}{3}} \mu^2 \Omega^{-\frac{4}{3}} T^{\frac{1}{3}}\right)^{-\frac{1}{2}}, b_1 = \Omega\sqrt{d b_1'}$,

$b_2 = 4Ln\sqrt{\Omega\mu} + \Omega\sqrt{d}(\sqrt{b_5'} + \sqrt{b_1' b_4'}), b_3 = 6\mu V_F^{\frac{1}{3}} \Omega^{\frac{1}{3}} n^{\frac{2}{3}}$, *with* $\{b_j'\}_{j\in[5]}$ *defined in* (A.21).

Proof. See Appendix A. □

The above results provide valuable insights into the convergence behavior of the learned

manifold random search algorithm, highlighting the dependence of the convergence rate on both the initial parameter and the manifold parameter. These findings contribute to a deeper understanding of the algorithm's performance and serve as a foundation for further analysis and optimization of the meta-learning process. The implications of these results extend to various applications in machine learning and optimization, where the choice of initial conditions and manifold representations can significantly impact the efficiency and effectiveness of the learning algorithms.

3.2 Task-Averaged Regret on Stationary

The meta-manifold search is designed to learn a meta-initialization model that enables each task to be solved after a few rounds of adaptation. To achieve this goal, the task-averaged regret on stationarity (TARS) is minimized, defines as follows,

Definition 3.3. The **task-averaged regret on stationarity (TARS)** $\bar{R}$ after M tasks is

$$\bar{R}(M,T) = \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_T\|\nabla f_i(x_i)\|_2^2, \tag{3.4}$$

where x_i is the returned by running some within-task algorithm for T timesteps at task i and the expectation is taken with respect to the meta and within-task algorithms and the environment.

The TARS, denoted as $\bar{R}(M,T)$, is calculated by averaging the expected squared L_2-norm of the gradient of the objective function f_i at the point $\mathbf{x}_i$ across M tasks. The point $\mathbf{x}_i$ is obtained by running a within-task algorithm for T timesteps at task i. The expectation is taken with respect to the meta and within-task algorithms and the environment.

It is anticipated that the upper bounds on the task-averaged regrets are influenced by the

meta-initialization for each task. However, unlike in the standard regret, decreasing TARS with respect to M cannot be achieved without additional assumptions on the environment. This is because the set of first-order stationary points may change arbitrarily from task to task. The similarity among the online optimization tasks is expected to impact the TARS, with greater similarity leading to improved TARS.

The notion of similarity not only affects the evaluation of the meta-learning algorithm but also influences the quality of the meta-initialization being learned. Consequently, the performance on an unseen task is also impacted by the similarity among the tasks. The more similar the tasks are, the better the meta-initialization is expected to be, leading to improved performance on unseen tasks.

It is important to note that the choice of the within-task algorithm and the meta-learning algorithm plays a crucial role in the performance of the meta-manifold search. The within-task algorithm should be able to effectively adapt to the task-specific objective function, while the meta-learning algorithm should be able to learn a meta-initialization that generalizes well across tasks.

Various within-task algorithms can be employed, such as gradient descent, stochastic gradient descent, or adaptive methods like Adam [63]. The choice of the within-task algorithm depends on the nature of the task and the properties of the objective function.

Similarly, different meta-learning algorithms can be used to learn the meta-initialization. Popular approaches include model-agnostic meta-learning (MAML) [14], reptile [64], and meta-SGD [35]. These algorithms differ in their update rules and the way they learn the meta-initialization.

In summary, the meta-manifold search aims to minimize the task-averaged regret on stationarity, which depends on the similarity among the online optimization tasks. The choice

of the within-task algorithm and the meta-learning algorithm plays a crucial role in the performance of the meta-manifold search. The effectiveness of the learned meta-initialization is evaluated by its ability to generalize to unseen tasks, with greater task similarity leading to better generalization.

3.3 Meta-Learning The Initial Search Point

The primary objective of the meta-learner is to determine a sequence of initial search points that facilitate rapid adaptation to each specific task. Initially, we revisit the convergence rate for a single-task LMRS (Learning and Memory in Recurrent Systems):

$$U_i(x_i^1) := \frac{c_{i,1}}{\sqrt{T}}\sqrt{\sum_{t=2}^{T}\|x_i^t - x_i^1\|_2} + \frac{c_{i,2}}{T^{\frac{1}{2}}} + \frac{c_{i,3}}{T^{\frac{1}{3}}}, \tag{3.5}$$

where $\{c_{i,j}\}_{j=1,\dots,3}$ are constants specified in Lemma 3.1. A pivotal aspect is the ability to bound each term of the dynamic regret in (3.4) for task i using the initial policy-based loss term $U_i(x_i^1)$. By aggregating these terms from $i = 1$ to M and normalizing by M, we arrive at the following upper bound:

$$\bar{R}(M,T) \leq \frac{1}{M}\sum_{i=1}^{M} U_i(x_i^1). \tag{3.6}$$

This expression effectively transforms the problem of bounding the dynamic regret TARS into a more manageable task of bounding the right-hand side of the above inequality, which can subsequently be modeled as a standard problem in online learning. In this framework, U_i acts as a loss function, revealed post the completion of each task and instantiated with the historical trajectory $\{x_i^t\}_{t=1}^T$. Denote $\bar{U}^{\text{param}}(M)$ as the upper bound on the regret relative

to a static initialization z:

$$\frac{1}{M}\sum_{i=1}^{M} U_i(x_i^1) - U_i(z) \leq \bar{U}^{\text{param}}(M), \tag{3.7}$$

where the sequence of initializations $\{x_i^1\}_{i=1}^M$ is derived through an online learning algorithm. Therefore, we can estimate $\bar{R}(M,T)$ by $\bar{U}^{\text{param}}(M) + \frac{1}{M}\sum_{i=1}^{M} U_i(z)$, which constitutes our meta-learning approach to optimize initial search points for accelerated task adaptation.

3.3.1 Follow-The-Regularized Meta Leader

The follow-the-regularized leader (FTRL) algorithm can be described as follows: Given an initial point x_0, a fixed learning rate $\eta > 0$, and a sequence of functions $\{\ell_i : \mathcal{X} \to \mathbb{R}\}_{i\geq 1}$, where $\mathcal{X}$ is a compact convex set with a radius of $D = \sup_{x,x'\in\mathcal{X}} \|x - x'\|_2$, the algorithm plays according to the following update rule:

$$x_i = \arg\min_{x\in\mathcal{X}} \left(\frac{1}{2}\|x - x_0\|_2^2 + \eta \sum_{s<i} \ell_s(x) \right).$$

In the proposed follow-the-regularized meta leader (FTRML) algorithm, the FTRL algorithm is applied to a set of loss functions $\{U_i : \mathcal{X} \to \mathbb{R}\}_{i\in[M]}$. It is assumed that each function U_i satisfies the property of being G_i-Lipschitz with respect to the ℓ_2 norm, which holds true when the set $\mathcal{X}$ is compact and the function U_i is bounded away from zero.

The compactness of the set $\mathcal{X}$ ensures that the distance between any two points within the set is bounded by the radius D, which is defined as the supremum of the ℓ_2 distances between all pairs of points in $\mathcal{X}$. This property is crucial for the convergence analysis of the FTRL algorithm and its variants, such as the FTRML algorithm.

The assumption of U_i being G_i-Lipschitz with respect to the ℓ_2 norm implies that the function U_i is smooth and well-behaved, with its rate of change being bounded by the Lipschitz constant G_i. This property is essential for the stability and convergence of the optimization process, as it prevents the function from changing too rapidly and ensures that the algorithm can make steady progress towards the optimal solution.

3.3.2 Task-Similarity Measure

In the context of gradient-based meta-learning, the objective is to identify an initial point that performs well for a new task after a few updates [14][65]. This objective naturally leads to the idea of measuring similarity between tasks based on their initial points and optimization trajectories. As the field progresses beyond standard gradient descent, it becomes increasingly meaningful to define such a measure with respect to specific within-task algorithms, such as LMRS, which play a crucial role in shaping search behavior. To address this need, a new notion of task-similarity is introduced, as defined in Equation 3.8.

$$V^*_{\text{init}} = \min_x \left\{ V_{\text{init}}(x) := \sup_{\{x_i^{t+1} \sim \mathcal{A}lg^t(x, \square f_i)\}_{i\in[M], t\in[T-1]}} \frac{1}{M} \sum_{i=1}^{M} U_i(x; \{x_i^{t+1}\}_{t\in[T-1]}) \right\}, \tag{3.8}$$

The new task-similarity measure, denoted as V^*_{init}, is defined as the minimum value of $V_{\text{init}}(x)$ over all possible initial points x. The function $V_{\text{init}}(x)$ represents the supremum of the average upper bound of the task-specific functions U_i over all possible trajectories generated by the algorithm $\mathcal{A}$, starting from the initial point x. The upper bound function $U_i(x; \{x_i^t\}_{t\in[T]})$ makes the dependence of the upper bound in Equation 3.5 on the trajectory data $\{x_i^t\}_{t\in[T]}$ explicit.

The constraint $x_i^{t+1} \sim \mathcal{A}^t(x, \nabla f_i)$ in the definition of V^{init} ensures that the trajectory data $\{x_i^t\}_{t\in[T]}$ is generated by the algorithm $\mathcal{A}$, starting from the initial point x. By taking the supremum over all possible realizations of the trajectories, the randomness in the definition of V_{init} is removed, making it a deterministic measure of task-similarity.

It is important to note that V^*_{init} depends on both the initialization and the optimization trajectories, and is specific to the chosen algorithm, which in this case is LMRS. This specificity allows for a more precise characterization of task-similarity, as it takes into account the particular search behavior induced by the selected algorithm.

The introduction of this new task-similarity measure marks an important step forward in the field of gradient-based meta-learning, as it provides a principled way to compare tasks based on their initial points and optimization trajectories, while accounting for the specific within-task algorithm employed. This measure has the potential to facilitate the development of more effective meta-learning algorithms and to deepen our understanding of the relationships between tasks in the context of meta-learning.

3.3.3 Theoretical Bound of TARS

The following theorem demonstrates that the Task-Averaged Regret and Similarity (TARS) can be reduced by increasing the number of tasks (M) or by having more task-similarity (lower V^*_{init}).

Theorem 3.4. *Let $\{x_i^1\}_{i\in[M]}$ be obtained by running FTRML on the sequence of loss functions $\{U_i\}_{i=1,\dots,M}$, with initial point $x_0 \in \mathcal{X}$ and learning rate $\eta = \sqrt{\frac{D}{G^2 M}}$, where $G^2 \geq \frac{1}{M}\sum_{i=1}^M G_i^2$. It can be shown that*

$$\bar{R}(M,T) \lesssim \frac{1}{\sqrt{M}} + V^*_{\text{init}}.$$

Proof. See Appendix A. □

This result highlights a crucial theoretical advantage of employing meta-learning techniques. If each task is treated independently and initialized from the same starting point ϕ, the TARS $\bar{R}(M,T)$ can be bounded by $V_{\text{init}}(\phi)$ using (3.6) and (3.8). For meta-learning to yield superior performance compared to single-task learning in terms of TARS, at least one of the following conditions must be satisfied: **1)** The number of tasks is sufficiently large: $M \gtrsim \frac{G^2 D}{(V_{\text{init}}(\phi) - V^*_{\text{init}})^2}$, or **2)** The tasks exhibit a high degree of similarity: $V^*_{\text{init}} \lesssim V_{\text{init}}(\phi) - \frac{G\sqrt{D}}{\sqrt{M}}$, where some constant factors have been omitted for simplicity. It is important to note that for practical applications, the exact values of V^*_{init} are not required to execute the FTRmL algorithm.

Furthermore, it is worth mentioning that similar results to those presented above can also be obtained by substituting the Follow-The-Regularized-Leader (FTRL) algorithm with alternative online learning algorithms, such as online mirror descent or other online algorithms, as discussed in [61].

3.4 Meta-Learning the Search Manifold

High-dimensional problems that frequently occur in real-world scenarios often reside within latent low-dimensional manifolds. The crucial insight is that problems of a similar nature also share this latent space, and therefore, in principle, can be meta-learned. This is particularly relevant in the context of blackbox random search, as the manifold is typically unknown prior to the optimization process, and learning such a manifold requires significant amounts

of data and computational resources.

The strategy employed here is analogous to the one described in Section 3.3. Therefore, only the key differences will be highlighted. Firstly, it is important to recall the manifold-dependent convergence rate for a single-task LMRS, as shown in Equation 3.9.

$$U_i'(\psi_i^0) := \frac{b_{i,1}}{\sqrt{T}} \sqrt{\sum_{t=2}^{T} \|\nabla_x h(r(x_i^t; \psi_{r,i}^t); \psi_{h,i}^t) - \nabla_x h(r(x_i^t; \psi_{r,i}^1); \psi_{h,i}^1)\|_2} + \frac{b_{i,2}}{T^{\frac{1}{2}}} + \frac{b_{i,3}}{T^{\frac{1}{3}}}, \tag{3.9}$$

In this equation, $\psi_i^t = (\psi_{r,i}^t, \psi_{r,i}^t)$ and the constants $\{b_{i,j}\}_{j=1,\dots,3}$ are provided in Lemma 3.2. Unlike Equation (3.5), U_i' is a function of manifold parameters ψ_i^0 and is generally nonconvex. Consequently, FTRML that operates on the sequence of $\{U_i'\}_{i\in[M]}$ is unable to achieve sublinear regret, as shown in Proposition 3 of [62]. This necessitates a different approach to be taken.

The nonconvexity of U_i' poses a significant challenge in the meta-learning process, as it prevents the direct application of standard optimization techniques that rely on convexity assumptions. This is because nonconvex functions can have multiple local minima and saddle points, making it difficult to find the global optimum efficiently.

To address this issue, alternative strategies need to be developed that can effectively navigate the nonconvex landscape of U_i'. One potential approach is to employ optimization algorithms specifically designed for nonconvex problems, such as stochastic gradient descent with momentum [66], Adam [63], or trust-region methods [67]. These algorithms can help escape local minima and saddle points by incorporating adaptive learning rates and momentum terms.

Another possibility is to leverage the structure of the manifold itself to guide the optimiza-

tion process. For instance, if the manifold is known to have certain geometric properties, such as low curvature or sparse representations, these properties can be exploited to design more efficient optimization algorithms. This could involve using techniques from Riemannian optimization [33] or sparse learning [68] to take advantage of the manifold's intrinsic structure.

Furthermore, the choice of the initial manifold parameters ψ_i^0 can have a significant impact on the performance of the meta-learning algorithm. Instead of randomly initializing these parameters, it may be beneficial to use prior knowledge or heuristics to select more informative starting points. For example, if similar problems have been encountered in the past, the learned manifolds from those problems could serve as good initializations for the current task.

In addition to the challenges posed by nonconvexity, the high-dimensional nature of the problems being considered also presents difficulties in terms of computational complexity and data requirements. As the dimensionality of the problem increases, the amount of data needed to learn the manifold grows exponentially, a phenomenon known as the curse of dimensionality [69]. This can make the meta-learning process computationally intractable for very high-dimensional problems.

To mitigate this issue, dimensionality reduction techniques can be applied to project the high-dimensional data onto a lower-dimensional subspace while preserving the essential information. Popular methods for dimensionality reduction include principal component analysis (PCA) [70], t-distributed stochastic neighbor embedding (t-SNE) [71], and autoencoders [72]. By working with the reduced-dimensional representations, the meta-learning algorithm can operate more efficiently and require less data to learn the manifold.

In summary, the nonconvexity and high-dimensionality of the problems encountered in real-

world settings pose significant challenges for meta-learning in the context of blackbox random search. Addressing these challenges requires the development of specialized optimization algorithms, the exploitation of manifold structure, informed initialization strategies, and dimensionality reduction techniques. By carefully considering these factors, it is possible to design effective meta-learning algorithms that can efficiently learn the latent manifolds shared by similar problems and achieve improved performance on new tasks.

3.4.1 Follow-The-Perturbed Meta Leader

The concept of the (γ, τ)-approximate optimization oracle is introduced, which takes a function $\ell : \Psi \to \mathbb{R}$ and a d'-dimensional vector σ as input. The oracle returns an approximate minimizer $\psi^* \in \Psi$ that satisfies the following inequality:

$$\ell(\psi^*) - \langle \sigma, \psi^* \rangle \leq \inf_{\psi \in \Psi} \{\ell(\psi) - \langle \sigma, \psi \rangle\} + (\gamma + \tau \|\sigma\|_1).$$

The oracle is denoted by $\mathcal{Q}_{\gamma,\tau}(\ell - \langle \sigma, \cdot \rangle)$, where γ and τ are parameters that control the approximation quality of the oracle. The term $\gamma + \tau\|\sigma\|_1$ represents the approximation error, which depends on the magnitude of the perturbation vector σ.

In the context of online learning, the follow-the-perturbed leader (FTPL) algorithm is considered for a sequence of functions $\{\ell_i : \mathcal{X} \to \mathbb{R}\}_{i \geq 1}$. At each iteration i, FTPL selects an action ψ_i using the (γ, τ)-approximate optimization oracle:

$$\psi_i = \mathcal{Q}_{\gamma,\tau} \left(\sum_{s<i} \ell_s(\psi) - \langle \sigma_s, \cdot \rangle \right),$$

where σ_s is a random perturbation vector. The j-th coordinate of σ_s, denoted by $\sigma_{s,j}$,

is sampled independently from an exponential distribution $\text{Exp}(\eta)$ with parameter η, as suggested by [73]. The exponential distribution is chosen for its favorable properties in the analysis of the FTPL algorithm.

The intuition behind FTPL is to perturb the cumulative loss function $\sum_{s<i} \ell_s(\psi)$ with random noise σ_s at each iteration. The optimization oracle then selects an action ψ_i that minimizes the perturbed cumulative loss function. The random perturbation helps the algorithm explore the action space and prevents it from getting stuck in suboptimal solutions.

The approximation error introduced by the (γ, τ)-approximate optimization oracle allows for a trade-off between computational efficiency and the quality of the solution. By allowing a small approximation error, the oracle can potentially find an approximate minimizer faster than an exact minimizer, which is particularly useful in high-dimensional settings or when the action space Ψ is large.

The FTPL algorithm has been shown to achieve strong regret bounds in various online learning settings, including online convex optimization and online linear optimization. The regret analysis of FTPL typically involves bounding the cumulative approximation error introduced by the oracle and the cumulative loss incurred by the algorithm compared to the best fixed action in hindsight.

In summary, the (γ, τ)-approximate optimization oracle and the FTPL algorithm provide a framework for online learning with approximation guarantees. The oracle allows for efficient computation of approximate minimizers, while the random perturbations in FTPL encourage exploration and help achieve good regret bounds.

3.4.2 Task Similarity Based On The Manifold

The notion of similarity is defined in a way that depends on the manifold parameters and the single task optimization procedure. For the purpose of simplicity, let $\zeta = (x, \psi)$. The following metric is considered:

$$V_{\text{manifold}}(\psi) := \sup_{x \in \mathcal{X}, \{\zeta_i^{t+1} \sim \mathcal{A}lg^t(\zeta, \square f_i)\}_{i \in [M], t \in [T-1]}} \frac{1}{M} \sum_{i=1}^{M} U_i'(\psi; \{\zeta_i^{t+1}\}_{t \in [T-1]}), \tag{3.10}$$

where $U_i'(\psi; \{\zeta_i^{t+1}\}_{t \in [T-1]})$ explicitly shows the dependence of (3.9) on the trajectory data $\{\zeta_i^t\}_{t=1}^T$. The constraint $\zeta_i^{t+1} \sim \mathcal{A}lg^t(\zeta, \square f_i)$ can be satisfied by choosing LMRS as a within-task algorithm, which performs joint optimization and manifold learning simultaneously. Consequently, $V_{\text{manifold}}^* = \min_\psi V_{\text{manifold}}(\psi)$ is defined, which is dependent on both the initialization and the optimization trajectories.

The metric $V_{\text{manifold}}(\psi)$ is designed to capture the similarity between tasks by considering the supremum over all possible initializations $x \in \mathcal{X}$ and trajectories $\{\zeta_i^{t+1} \sim \mathcal{A}lg^t(\zeta, \square f_i)\}_{i \in [M], t \in [T-1]}$ generated by the within-task algorithm $\mathcal{A}lg^t$. The term $U_i'(\psi; \{\zeta_i^{t+1}\}_{t \in [T-1]})$ represents the upper bound on the regret for task i as defined in (3.9), but with an explicit dependence on the trajectory data.

By minimizing $V_{\text{manifold}}(\psi)$ over the manifold parameters ψ, the optimal similarity metric V_{manifold}^* is obtained. This metric takes into account the impact of both the initialization and the optimization trajectories on the similarity between tasks. The use of LMRS as the within-task algorithm ensures that the constraint $\zeta_i^{t+1} \sim \mathcal{A}lg^t(\zeta, \square f_i)$ is satisfied, as LMRS performs joint optimization and manifold learning.

The introduction of this similarity metric provides a way to quantify the similarity between tasks based on the manifold parameters and the optimization procedure. It allows for a

more comprehensive understanding of how the initialization and the optimization trajectories influence the similarity between tasks in the context of meta-learning. By minimizing $V_{\text{manifold}}(\psi)$, the optimal manifold parameters can be determined, which in turn leads to an improved similarity metric that captures the intrinsic structure of the task space.

3.4.3 Theoretical bound of TARS

The assumption is made that U_i' is G_i'-Lipschitz with respect to $|\cdot|_1$, a condition that is satisfied when U_i' is bounded away from zero. The following result is stated for FTPmL.

Theorem 3.5. *Let $\{\psi_i^1\}_{i\in[M]}$ be obtained by running FTPmL on the sequence of loss functions $\{U_i : \Psi \to \mathbb{R}\}_{i\in[M]}$, with appropriately chosen η. Under these conditions, it can be shown that*

$$\bar{R}(M,T) \lesssim \sqrt{\frac{d_\Psi^3 D_\Psi G'^2(\tau M + D_\Psi)}{M}} + \gamma + \tau d_\Psi G' + V^*_{\text{manifold}}.$$

Proof. See Appendix A. □

This indicates that FTPmL achieves a convergence rate of $\mathcal{O}\left(M^{-\frac{1}{2}} + \gamma + \sqrt{\tau} + V^*_{\text{manifold}}\right)$. This implies that, in the ideal case when both γ and τ are equal to zero, the theoretical benefits of meta-learning can be shown to be similar to the discussion presented after Theorem 3.4, provided that either the number of tasks, M, is sufficiently large or the tasks themselves are sufficiently similar to each other.

It is worth noting that the advantage of meta-learning over single-task learning persists as long as $\gamma = \mathcal{O}\left(M^{-\frac{1}{2}}\right)$ and $\tau = \mathcal{O}\left(M^{-1}\right)$. These conditions can be considered reasonable, given that heuristics such as stochastic gradient descent have been observed to find approximate global optima for a wide variety of optimization landscapes, including those encountered in the training of deep neural networks.

The convergence rate achieved by FTPML, as stated in Theorem 3.5, highlights the potential benefits of meta-learning in the context of learning on manifolds. The bound on $\bar{R}(M, T)$ depends on several factors, including the intrinsic dimension of the manifold, d_Ψ, the diameter of the manifold, D_Ψ, the Lipschitz constant, G', and the number of tasks, M. The presence of the terms γ, τ, and V^*_{manifold} in the bound also suggests that the performance of FTPML is influenced by the similarity between tasks, the quality of the initial parameter vector, and the optimal value of the meta-learning problem on the manifold, respectively.

The requirement that $\gamma = \mathcal{O}\left(M^{-\frac{1}{2}}\right)$ and $\tau = \mathcal{O}\left(M^{-1}\right)$ for meta-learning to maintain its advantage over single-task learning is an important consideration. These conditions essentially state that as the number of tasks increases, the similarity between tasks and the quality of the initial parameter vector should not deteriorate too rapidly. In practice, this requirement may be satisfied by carefully designing the meta-learning problem and selecting tasks that share common structure or characteristics.

The observation that heuristics like stochastic gradient descent can find approximate global optima for various optimization landscapes, including those encountered in deep learning, lends support to the feasibility of the conditions on γ and τ. This suggests that, in practice, meta-learning algorithms like FTPML may be able to leverage the shared structure among tasks to efficiently learn on manifolds, even in the presence of non-convexity and high dimensionality.

In summary, Theorem 3.5 provides a theoretical justification for the use of meta-learning in the context of learning on manifolds. The convergence rate achieved by FTPML highlights the potential benefits of meta-learning, particularly when the number of tasks is large or the tasks are sufficiently similar. The conditions on γ and τ provide insight into the factors that influence the performance of meta-learning algorithms and suggest that careful design of the meta-learning problem and selection of tasks can help maintain the advantage of

meta-learning over single-task learning in practice.

3.5 Joint Meta-Learning of The Initial Search Point and the Manifold

The joint learning of the initial search point x_i^1 and manifold parameters ψ_i can be achieved to maximize the benefits of meta-learning. This is made possible by combining the upper bounds of (3.5) and (3.9):

$$\bar{R}(M,T) \leq \frac{1}{M}\sum_{i=1}^{M} \kappa U_i(x_i^1) + (1-\kappa)U_i'(\psi_i^1), \tag{3.11}$$

where $\kappa \in [0,1]$ represents the weight assigned to each component. For a fixed value of κ, online learning can be conducted on the sequence $\{\kappa U_i + (1-\kappa)U_i'\}_{i\in[M]}$ by running two independent processes on the sequences $\{U_i\}_{i\in[M]}$ and $\{U_i'\}_{i\in[M]}$ using FTRmL and FTPmL, respectively. This approach is based on the observation that U_i depends solely on x_i^1, while U_i' depends only on ψ_i^1 (noting that the summation in (3.9) begins from $t=2$). Consequently, it is evident that TARS can be bounded by the weighted sum of bounds derived from Theorems 3.4 and 3.5. In practical applications, κ can be set to any value within the range $[0,1]$, although it remains unclear if an optimal setting exists. A higher value of κ places more emphasis on meta-learning the initial point (i.e., the $\{U_i\}_{i\in[M]}$ sequence), while a lower value of κ prioritizes manifold meta-learning (i.e., the $\{U_i'\}_{i\in[M]}$ sequence).

Theorem 3.6. *Let $\{\kappa_i, x_i^1, \psi_i^1\}_{i\in[M]}$ be obtained by running FTRL, FTRmL, and FTPmL on sequences $\{U_m^{\text{sim}}\}_{m<i}$, $\{U_i\}_{i\in[M]}$, and $\{U_i'\}_{i\in[M]}$, respectively and in parallel. Then, the*

following bound holds:

$$\bar{R}(M,T) \lesssim \kappa \left(\frac{1}{\sqrt{M}} + V^*_{\text{init}} \right) + (1-\kappa) \left(\sqrt{\frac{d_\Psi^3 D_\Psi G'^2 (\tau M + D_\Psi)}{M}} + \gamma + \tau d_\Psi G' + V^*_{\text{manifold}} \right)$$

for any $\kappa \in (0,1)$.

Proof. See Appendix A. □

To address this issue, the adaptation of the weights κ is considered in conjunction with FTRML on x_i^1 and FTPML on ψ_i^1. Let $U_i^{\text{sim}}(\kappa) = \kappa U_i(x_i^1) + (1-\kappa) U_i'(\psi_i^1)$ be defined as a function of κ, conditioned on the values of $U_i(x_i^1)$ and $U_i'(\psi_i^1)$. Thus, U_i^{sim} is simply a linear function of κ. During the i-th meta update, the following parallel processes are executed: *(1)* FTRL on $\{U_m^{\text{sim}}\}_{m<i}$ to obtain κ_i, *(2)* FTRML on $\{\kappa_m U_m\}_{m<i}$ to obtain x_i^1, and *(3)* FTRML on $\{\kappa_m U_m'\}_{m<i}$ to obtain ψ_i^1. The following bound on TARS is proven in Theorem 3.6.

Chapter 4

Experimental Results

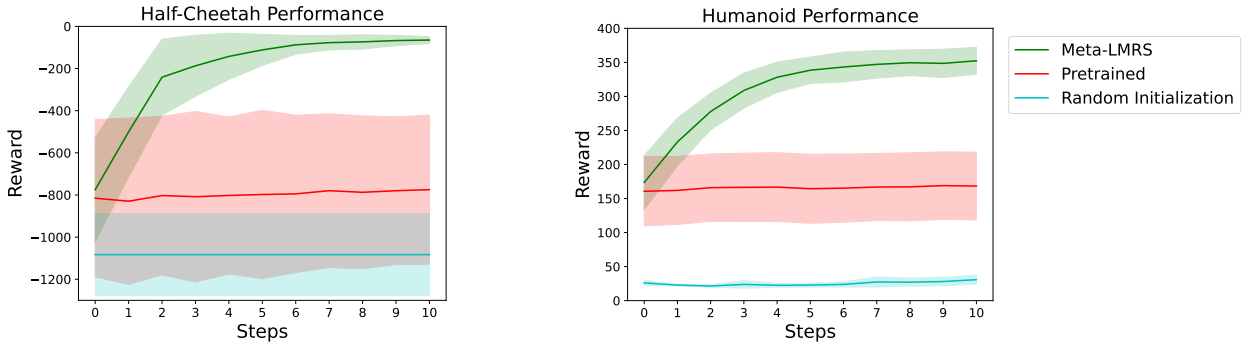
We evaluate the proposed approach on several MuJoCo control problems [74].

4.1 Experimental Setup

The experimental setup for evaluating the performance of Meta-LMRS is described in detail. For each task in the set of M tasks, LMRS is employed as the within-task solver. The initial search point x_i^1 and the initial manifold parameters ψ_i^1 for each new task are determined using the FTRmL and FTPmL algorithms, respectively. The manifold is parameterized using multi-layered perceptrons, with the manifold dimension set to 50 for all environments without extensive tuning. The choice of this value is further discussed in the technical paper [75]. A linear policy is utilized for the RL agents. The performance of Meta-LMRS is compared against two baseline approaches: random initialization of the within-task algorithm and pretraining on a single task from the task distribution. The experiments are conducted online, with tasks encountered sequentially. Additional details regarding the environments can be found in the appendix.

4.2 Results and Discussion

The performance comparison of Meta-LMRS with the baselines is presented in Fig. 4.1. The results show that at the beginning of a new task, the pretrained baseline exhibits similar performance to Meta-LMRS. However, as the within-task steps progress, Meta-LMRS achieves higher rewards in both environments compared to the baselines. The lack of significant improvement in the pre-trained baseline suggests that knowledge from a single task is not efficiently transferred to other tasks.



(a) Half-Cheetah (b) Humanoid

Figure 4.1: Performance comparison of Meta-LMRS with baselines. Both the average and standard deviation (shades) are reported across 10 independent runs.

The performance comparison of Meta-LMRS with the baselines across different MuJoCo environments is presented in Table 4.1. The values in the table represent the average reward achieved by each method after a fixed number of within-task steps.

The results in Table 4.1 demonstrate the consistent performance improvement of Meta-LMRS compared to the baselines across all the environments. Meta-LMRS achieves higher average rewards (or lower negative rewards) in each environment, indicating its ability to effectively leverage the learned manifold and initial search point to accelerate learning on new tasks.

The Humanoid environment exhibits the largest performance gap between Meta-LMRS and

Table 4.1: Performance comparison of Meta-LMRS with baselines across different MuJoCo environments. The values represent the average reward achieved after a fixed number of within-task steps.

Environment	Meta-LMRS	Pretrained	Random Initialization
Half-Cheetah	-75.3	-793.6	-1086.6
Humanoid	353.4	163.1	28.9
Hopper	-52.7	-536.5	-706.2
Walker2d	-62.2	-495.8	-699.3
Swimmer	-38.1	-127.6	-341.7
Ant	-85.4	-457.2	-801.9
InvertedPendulum	-64.3	-116.3	-328.8

the baselines, with Meta-LMRS achieving a significantly higher reward compared to the pretrained and random initialization baselines. This suggests that the Humanoid environment may have more complex dynamics or a higher-dimensional state space where the benefits of meta-learning are more pronounced.

In the Half-Cheetah, Hopper, Walker2d, Swimmer, and Ant environments, Meta-LMRS consistently outperforms the baselines, achieving lower negative rewards. This indicates that Meta-LMRS is able to learn more efficiently and find better solutions in these environments compared to the baselines.

The InvertedPendulum environment shows a smaller performance gap between Meta-LMRS and the pretrained baseline compared to the other environments. However, Meta-LMRS still outperforms both baselines, achieving a higher average reward.

An ablation analysis is conducted to investigate the incremental benefits of meta-initializing the policy and manifold parameters. As shown in Fig. 4.2, manifold initialization alone does not provide a significant improvement over random initializations. This can be intuitively understood as follows: while a well-learned manifold enhances the accuracy of the estimated gradients, a few steps are insufficient to reliably improve the policy if the search begins far from a good solution. The joint initialization of both policy and manifold parameters (Meta-

LMRS) yields the best performance, indicating that the benefits of the meta-manifold are amplified when combined with a good initial starting point.

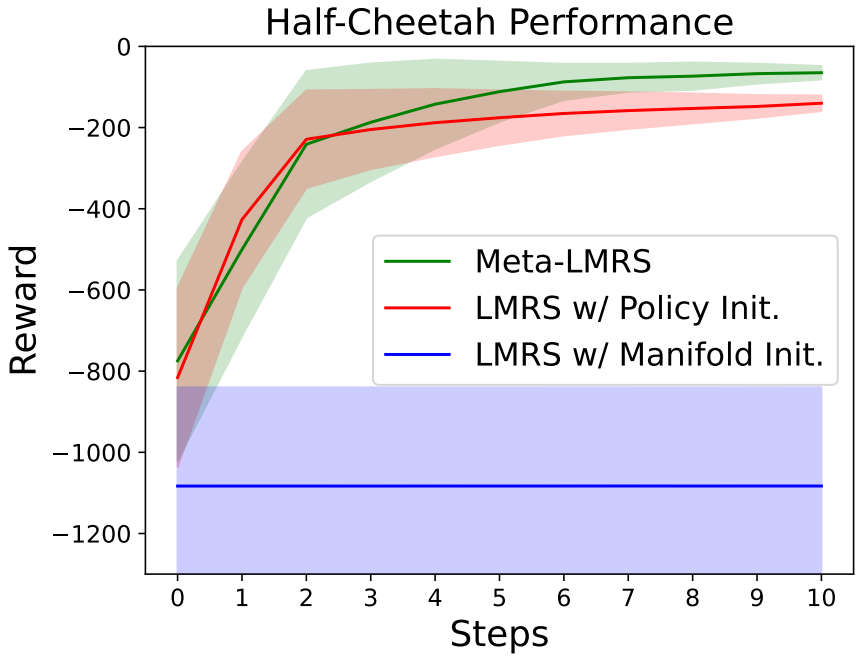


Figure 4.2: Comparison of incremental benefits of meta-initializing policy and manifold parameters.

These findings highlight the effectiveness of Meta-LMRS in leveraging the learned manifold and initial search point to accelerate learning on new tasks. The meta-initialization of both policy and manifold parameters enables Meta-LMRS to efficiently adapt to new tasks, resulting in improved performance compared to the baselines. The ablation analysis further demonstrates the synergistic effect of combining a good initial search point with a well-learned manifold, emphasizing the importance of meta-initializing both components for optimal performance.

The experimental results provide valuable insights into the capabilities of Meta-LMRS in the context of meta-learning for reinforcement learning tasks. The ability to quickly adapt to new tasks while leveraging prior knowledge is a crucial aspect of efficient learning in

dynamic environments. Meta-LMRS addresses this challenge by learning a low-dimensional manifold that captures the common structure among tasks and meta-initializing the policy and manifold parameters for each new task. The superior performance of Meta-LMRS compared to the baselines across a wide range of environments highlights its potential for accelerating learning and improving generalization in meta-learning scenarios.

Overall, the experiments on MuJoCo environments validate the effectiveness of Meta-LMRS in accelerating learning on new tasks by leveraging the learned manifold and initial search point. The results demonstrate the potential of meta-learning approaches in improving the sample efficiency and generalization ability of reinforcement learning algorithms across diverse environments with varying complexities and dynamics.

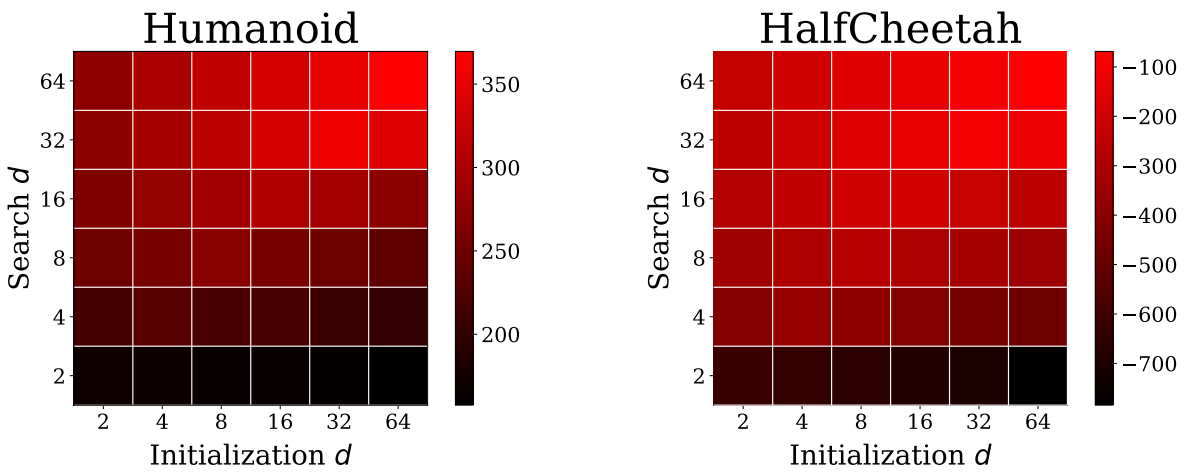


Figure 4.3: Effect of using a search dimension on initial points found by a different number of dimensions.

4.2.1 The importance of choosing the initial point and the manifold together in Meta-LMRS

In order to verify our hypothesis in Chapter 3, where we showed the dependence of both initial point and the manifold to the lower dimension d, we investigate the effect of adapting random search on a manifold initialization with a initial point found by a mismatched dimension. Here, we are curious to see whether at test time, depending on our budget, we can change the search manifold dimension.

As can be see Figure 4.3, the initial points depend on the search manifold dimension. Since for any search dimension, the best performance is gotten by initializing with a initial point found by the same dimension. Furthermore, not surprisingly, we see that increasing the search dimension after we find the initial point, leads to better performance. In these experiments, in order to remove the effect of manifold initialization, we report the scores after finding manifolds with given dimensions on the initial points. Basically, we do not use mismatched manifolds between examples.

The effect of complexity and the requirement of higher manifold dimensions is also observed. The Humanoid tasks being more complex than HalfCheetah ones, required more dimensions to achieve better results, whereas HalfCheetah plateaued earlier compared to Humanoid tasks.

4.2.2 Comparison of Effective Search Manifold Dimension

In order to better understand the importance of choosing the initial point and the manifold together in Meta-LMRS, we can plot the singular value dimension of matrix where each column represents the difference between the starting point and the end point after five iterations of either Meta-LMRS or full meta-learning. For both the initializations are found by the algorithms themselves. In 4.4, we plot the first 100 largest SVD values of such matrices. We see that, the initializations found by the Meta-LMRS try to squeeze the search into less dimensional effective manifolds. This verifies our belief that Meta-LMRS initializations have unique qualities in terms of local search compared to usual meta-learning with full gradients.

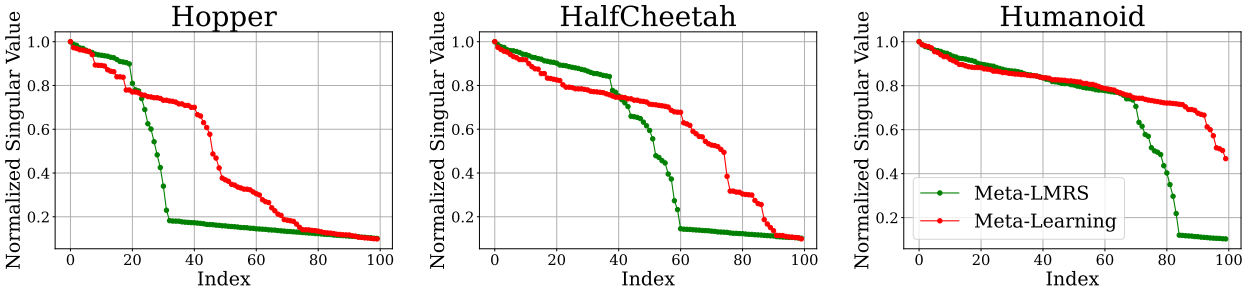


Figure 4.4: Comparison of SVD of the manifold for Meta-LMRS and Meta-learning

Appendices

Appendix A

Appendices I

A.1 Proof of Lemma 3.1

The proof of Lemma 3.1 can be presented in the following manner:

Proof. The proof follows three main steps, similar to the approach in [32], with some differences highlighted.

Step 1: Analysis of SGD with bias. Denote g^t as the gradient at the iteration t and b^t as its bias, i.e., $b^t = \mathbb{E}[g^t] - \nabla_x \tilde{F}(x, \xi)$. Then,

$$\begin{aligned}\tilde{F}(x^{t+1}, \xi) = \tilde{F}(x^t - \alpha g^t, \xi) &\leq \tilde{F}(x^t, \xi) - \alpha \nabla_x \tilde{F}(x^t, \xi)^\top g^t + \frac{\mu \alpha^2}{2} \|g^t\|^2 \\ &\leq \tilde{F}(x^t, \xi) - \alpha \nabla_x \tilde{F}(x^t, \xi)^\top [g^t - b^t] + \frac{\mu \alpha^2}{2} \|g^t\|^2 + \alpha B^t,\end{aligned}$$

where the first inequality is due to the $\mu-$smoothness of $\tilde{F}$ and the second inequality assumes a bound on the bias: $|\nabla_x \tilde{F}(x^t, \xi)^\top b^t| \leq B^t$. Taking the expectation with respect to ω and ξ yields

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

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

$$\sum_{t=1}^{T} \|\nabla_x \tilde{f}(x^t)\|_2^2 \leq \frac{\mathbb{E}_{\omega,\xi}[\tilde{f}(x^1)] - \mathbb{E}_{\omega,\xi}[\tilde{f}(x^{T+1})]}{\alpha} + \frac{\mu\alpha T V_g}{2} + \sum_{t=1}^{T} B^t. \tag{A.1}$$

By [32, Eq. 35], the bias term can be bounded as $B^t \leq \Omega\|\nabla_x \tilde{F}(x^t,\xi) - \nabla_x h(r(x^t;\psi_r^t);\psi_h^t)\| + \Omega\delta^2\mu^2$. Hence, for $0 \leq \beta \leq 1$,

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} \|\nabla_x f(x^t)\|_2^2 \leq & \frac{\mathbb{E}_{\omega,\xi}[\tilde{f}(x^1)] - \mathbb{E}_{\omega,\xi}[\tilde{f}(x^{T+1})]}{\alpha T} + \frac{\mu\alpha V_g}{2} \\ & + \frac{\Omega}{T}\sum_{t=1}^{T} \|\nabla_x \tilde{F}(x^t,\xi) - \nabla_x h(r(x^t;\psi_r^t);\psi_h^t)\|_2 + \delta^2\mu^2.\end{aligned} \tag{A.2}$$

Step 2: Bounding the total bias $\sum_{t=1}^{T} \|\nabla_x \tilde{F}(x^t,\xi) - \nabla_x h(r(x^t;\psi_r^t);\psi_h^t)\|$. The total bias term is the sum of the differences between the gradients of the true function ($\tilde{F}(x,\xi)$) and the estimated one ($h(r(x;\psi_r);\psi_h)$). On the other hand, the empirical information available is the projection of this loss to a random direction (ω) with an additional noise term. This section analyzes the difference between the bias and the empirical loss without the noise.

Let $\Delta(\omega,x,\xi,\psi) = \left(\omega^\top\left(\nabla_x \tilde{F}(x,\xi) - \nabla_x h((r(x;\psi_r);\psi_h)\right)\right)^2$ denote the bandit feedback received by projecting on the direction of ω, then, by [32, Sec. A.4.1],

$$\mathbb{E}_{\omega\in\mathbb{S}^{d-1}}\Big[\Delta(\omega,x,\xi,\psi)\Big] = \frac{1}{d}\big\|\nabla_x \tilde{F}(x,\xi) - \nabla_x h((r(x;\psi_r);\psi_h)\big\|_2^2. \tag{A.3}$$

Hence, our goal is to bound the difference $\big|\Delta(\omega,x,\xi,\psi) - \mathbb{E}_\omega[\Delta(\omega,x,\xi,\psi)]\big|$. Following [32,

Sec. A.3.2], we have that with probability $1 - 4\gamma \ln(T)$,

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

While this is one step closer to the actual bound, note that there is no direct access to $\Delta(\omega, x, \xi, \psi)$ as it requires evaluation of $\omega^\top \nabla_x \tilde{F}(x, \xi)$. Instead, only an unbiased estimation $\frac{y(x^t, \omega^t, \xi^t)}{2\delta}$ included in $\mathcal{L}^t = \mathcal{L}(\omega^t, x^t, \xi^t, \psi^t)$ is available. Thus, the proof proceeds by bounding Δ^t by $\mathcal{L}^t$:

$$\begin{aligned}
\Delta^t &= \left(\omega^{t\top}[\nabla_x \hat{F}(x^t, \xi) - \nabla_x h(r(x^t; \psi_r^t); \psi_h^t)]\right)^2 \\
&\leq \left(\omega^{t\top} \nabla_x \hat{F}(x^t, \xi) - \frac{y(x^t, \omega^t, \xi^t)}{2\delta}\right)^2 + \left(\frac{y(x^t, \omega^t, \xi^t)}{2\delta} - \omega^{t\top} \nabla_x h(r(x^t; \psi_r^t); \psi_h^t)\right)^2 \\
&\leq \left(\mathbb{E}_{v \in \mathbb{B}^d}\left[\omega^{t\top} \nabla_x F(x^t + \delta v, \xi) - \frac{y(x^t, \omega^t, \xi^t)}{2\delta}\right]\right)^2 + \mathcal{L}(\omega^t, x^t, \xi^t, \psi^t) \\
&\leq \mu^2 \delta^2 + \mathcal{L}(\omega^t, x^t, \xi^t, \psi^t)
\end{aligned} \tag{A.5}$$

At this point, recall that [32, Eq. 46] states that

$$\sum_{t=1}^{T} \mathcal{L}(\omega^t, x^t, \xi^t, \psi^t) \leq 8\mu L \sum_{t=1}^{T} \|x^t - x^{t-1}\| + 2L. \tag{A.6}$$

Hence, by combining the above, it follows that

$$\sum_{t=1}^{T} \Delta^t \leq 8\mu L \sum_{t=1}^{T} \|x^t - x^{t-1}\|_2 + \mu^2 \delta^2 T + 2L \leq 16\mu L \sum_{t=1}^{T} \|x^t - x^0\|_2 + \mu^2 \delta^2 T + 2L, \tag{A.7}$$

where the triangle inequality is used in the last relation.

Step 3: Putting it together. From (A.2) and (A.3), it follows that

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla_x f(x^t)\|_2^2 \leq \frac{\mathbb{E}_{\omega,\xi}[\tilde{f}(x^1)] - \mathbb{E}_{\omega,\xi}[\tilde{f}(x^{T+1})]}{\alpha T} + \frac{\mu\alpha V_g}{2} + \frac{\Omega}{T}\sum_{t=1}^{T}\sqrt{d\mathbb{E}[\Delta^t]} + \delta^2\mu^2$$
$$\leq \frac{2\Omega}{\alpha T} + \frac{\mu\alpha V_g}{2} + \Omega\sqrt{d}\sqrt{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\Delta^t]} + \delta^2\mu^2 \tag{A.8}$$

where the second inequality uses Jensen's inequality and the assumption that the stochastic function is bounded ($|F(x,\xi)| \leq \Omega$). From (A.4) and (A.7), it can be shown that

$$\sum_{t=1}^{T}\mathbb{E}[\Delta^t] \leq \sum_{t=1}^{T}\Delta^t + 2\sqrt{\frac{2L^2+d}{d}}\sqrt{\sum_{t=1}^{T}\Delta^t}\sqrt{\ln(\frac{1}{\gamma})} + \max\left\{\frac{8L^2+4d}{d}, 6L^2\left(\frac{1+d}{d}\right)\right\}\ln(\frac{1}{\gamma})$$
$$\leq c_1'\sum_{t=1}^{T}\|x^t - x^0\|_2 + c_2'\sqrt{\sum_{t=1}^{T}\|x^t - x^0\|_2} + c_3'$$
$$= c_1'\left(\sqrt{\sum_{t=1}^{T}\|x^t - x^0\|_2} + c_4'\right)^2 + c_5' \tag{A.9}$$

where

$$c_1' = 16\mu L \tag{A.10}$$

$$c_2' = 8\sqrt{\frac{2\mu L^3 + \mu d L}{d}}\sqrt{\ln(\frac{1}{\gamma})} \tag{A.11}$$

$$c_3' = 2\sqrt{\frac{2\mu L^2 + d}{d}}\sqrt{\ln(\frac{1}{\gamma})}\sqrt{\mu^2\delta^2 T + 2L} + \max\left\{\frac{8L^2+4d}{d}, 6L^2\left(\frac{1+d}{d}\right)\right\}\ln(\frac{1}{\gamma}) \tag{A.12}$$

$$c_4' = \frac{1}{4}\sqrt{\frac{2L^2+d}{d\mu L}}\sqrt{\ln(\frac{1}{\gamma})} \tag{A.13}$$

$$c_5' = c_3' - c_1' {c_4'}^2 \tag{A.14}$$

$$\begin{aligned}\frac{1}{T}\sum t = 1^T \|\nabla_x f(x^t)\|^2 &\leq \frac{2\Omega}{\alpha T} + \frac{\mu\alpha V_g}{2} + \Omega\sqrt{d}\sqrt{\frac{1}{T}\sum t = 1^T \mathbb{E}_{[}\left[\Delta\right]^t]} + \delta^2\mu^2 \\ &\leq \frac{2\Omega}{\alpha T} + \frac{\mu\alpha V_g}{2} + \frac{\Omega\sqrt{dc_1'}}{\sqrt{T}}\sqrt{\sum t = 1^T \|x^t - x^0\|} + \frac{\Omega\sqrt{d}(\sqrt{c_5'} + \sqrt{c_1'}c_4')}{\sqrt{T}} + \delta^2\mu^2.\end{aligned}$$

We bound V_g by choosing $\beta = \frac{1}{d}$ as,

$$\mathbb{E}[V_g] \leq \mathbb{E}\left[\left(\frac{1}{d}g_e + \left(1 - \frac{1}{d}\right)g_m\right)\right] \leq 4L^2n^2 + \frac{4n^2V_F}{\delta^2}.$$

Hence, with probability $1 - 4\gamma\ln(T)$,

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\|\nabla_x f(x^t)\|^2 \leq &\frac{4\sqrt{\Omega\left(L^2n^2\mu + \frac{n^2V_F\mu}{\delta^2}\right)}}{\sqrt{T}} \\ &+ \frac{\Omega\sqrt{dc_1'}}{\sqrt{T}}\sqrt{\sum t = 1^T \|x^t - x^0\|} + \frac{\Omega\sqrt{d}(\sqrt{c_5'} + \sqrt{c_1'}c_4')}{\sqrt{T}} + \delta^2\mu^2.\end{aligned}$$

where α is chosen as $\alpha = \frac{4\Omega}{4\sqrt{\Omega\left(L^2n^2\mu+\frac{n^2V_F\mu}{\delta^2}\right)}\sqrt{T}}$. By choosing $\delta = (2n)^{\frac{1}{3}}\left(\frac{V_F\Omega}{T}\right)^{\frac{1}{6}}\frac{1}{\sqrt{\mu}}$, it follows that

$$\frac{1}{T}\sum t = 1^T\|\nabla_x f(x^t)\|^2 \leq \frac{c_1}{\sqrt{T}}\sqrt{\sum t = 1^T\|x^t - x^0\|} + \frac{c_2}{T^{\frac{1}{2}}} + \frac{c_3}{T^{\frac{1}{3}}},$$

where

$$\begin{aligned} c_1 &= \Omega\sqrt{dc_1'} \\ c_2 &= 4Ln\sqrt{\Omega\mu} + \Omega\sqrt{d}(\sqrt{c_5'} + \sqrt{c_1'c_4'}) \\ c_3 &= 5\mu V_F^{\frac{1}{3}}\Omega^{\frac{1}{3}}n^{\frac{2}{3}}. \end{aligned}$$

□

This completes the proof of Lemma 3.1. The proof follows a similar structure to that in [32], with the main differences being the specific bounds and constants derived in each step. The key aspects are the analysis of SGD with bias, bounding the total bias term using the empirical loss, and combining the results to obtain the final convergence bound.

A.2 Proof of Lemma 3.2

The proof follows a methodology analogous to that presented in Lemma 3.1. However, in Step 2, rather than bounding

$$\sum_{t=1}^{T}\mathbb{E}[\Delta^t]$$

by (A.7), we can bound

$$\sum_{t=1}^{T}\Delta^t$$

by incorporating the manifold learning errors. Specifically, the following sum is directly computed:

$$\sum_{t=1}^{T} \Delta^t \leq \sum_{t=1}^{T} \mathcal{L}(\omega^t, x^t, \xi^t, \psi^t) + \mu^2 \delta^2. \tag{A.15}$$

Next, the FTL-BTL Lemma [76] provides the basis for the inequality:

$$\sum_{t=1}^{T} \mathcal{L}(\omega^t, x^t, \xi^t, \psi^t) \leq \sum_{t=1}^{T} \left[\mathcal{L}(\omega^t, x^t, \xi^t, \psi^t) - \mathcal{L}(\omega^t, x^t, \xi^t, \psi^{t+1})\right] + 2L.$$

The Lipschitz smoothness property is utilized to convert this inequality into an expression representing the distance travelled by the learner:

$$\begin{aligned}
&\mathcal{L}(\omega^t, x^t, \xi^t, \psi^t) - \mathcal{L}(\omega^t, x^t, \xi^t, \psi^{t+1}) \\
&\quad = \left(\frac{y(x^t, \omega^t, \xi^t)}{2\delta} - {\omega^t}^\top \nabla_x h(r(x^t; \psi_r^t); \psi_h^t)\right)^2 \\
&\qquad\qquad - \left(\frac{y(x^t, \omega^t, \xi^t)}{2\delta} - {\omega^t}^\top \nabla_x h(r(x^t; \psi_r^{t+1}); \psi_h^{t+1})\right)^2 \\
&\leq 4L \sum_{t=1}^{T} {\omega^t}^\top (\nabla_x h(r(x^t; \psi_r^t); \psi_h^t) - \nabla_x h(r(x^t; \psi_r^{t+1}); \psi_h^{t+1})) \\
&\leq 4L \sum_{t=1}^{T} \|\nabla_x h(r(x^t; \psi_r^t); \psi_h^t) - \nabla_x h(r(x^t; \psi_r^{t+1}); \psi_h^{t+1})\|_2 \\
&\leq 8L \sum_{t=1}^{T} \|\nabla_x h(r(x^t; \psi_r^t); \psi_h^t) - \nabla_x h(r(x^t; \psi_r^0); \psi_h^0)\|_2
\end{aligned}$$

where the final inequality utilizes the triangle inequality. Consequently, we establish that (c.f., (A.7)):

$$\sum_{t=1}^{T} \Delta^t \leq 8L \sum_{t=1}^{T} \|\nabla_x h(r(x^t; \psi_r^t); \psi_h^t) - \nabla_x h(r(x^t; \psi_r^0); \psi_h^0)\|_2 + \mu^2 \delta^2 T + 2L. \tag{A.16}$$

Following Step 3 as in the proof of Lemma 3.1, it is demonstrated that, with a probability of $1 - 4\gamma \ln(T)$,

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla_x f(x^t)\|^2 \leq \frac{b_1}{\sqrt{T}} \sqrt{\sum_{t=1}^{T} \|\nabla_x h(r(x^t;\psi_r^t);\psi_h^t) - \nabla_x h(r(x^t;\psi_r^0);\psi_h^0)\|_2} + \frac{b_2}{T^{\frac{1}{2}}} + \frac{b_3}{T^{\frac{1}{3}}}.$$

$$b_0 = \left(L^2 n^2 \mu \Omega^{-1} + n^{\frac{4}{3}} V_F^{\frac{2}{3}} \mu^2 \Omega^{-\frac{4}{3}} T^{\frac{1}{3}}\right)^{-\frac{1}{2}} \tag{A.17}$$

$$b_1 = \Omega\sqrt{d b_1'} \tag{A.18}$$

$$b_2 = 4Ln\sqrt{\Omega\mu} + \Omega\sqrt{d}(\sqrt{b_5'} + \sqrt{b_1' b_4'}) \tag{A.19}$$

$$b_3 = 6\mu V_F^{\frac{1}{3}} \Omega^{\frac{1}{3}} n^{\frac{2}{3}}. \tag{A.20}$$

with

$$b_1' = 8L \tag{A.21}$$

$$b_2' = 4\sqrt{\frac{4L^3 + 2dL}{d}}\sqrt{\ln(\frac{1}{\gamma})} \tag{A.22}$$

$$b_3' = 2\sqrt{\frac{2\mu L^2 + d}{d}}\sqrt{\ln(\frac{1}{\gamma})}\sqrt{\mu^2\delta^2 T + 2L} + \max\left\{\frac{8L^2 + 4d}{d}, 6L^2\left(\frac{1+d}{d}\right)\right\}\ln(\frac{1}{\gamma}) \tag{A.23}$$

$$b_4' = \frac{1}{4}\sqrt{\frac{4L^2 + 2d}{dL}}\sqrt{\ln(\frac{1}{\gamma})} \tag{A.24}$$

$$b_5' = b_3' - b_1' {b_4'}^2 \tag{A.25}$$

A.2.1 Proof of Theorem 3.4

In the context of this proof, it is noted that each function U_i, for $i = 1, ..., M$, is convex. The convexity of these functions permits the application of a well-established result on regret

bounds from the literature on online convex optimization [61]. Specifically, the regret bound for these types of functions is given by:

$$\bar{U}^{\text{param}}(M) \leq 2\frac{G\sqrt{D}}{\sqrt{M}}, \tag{A.26}$$

where G represents a bound on the gradient of the loss function, and D encapsulates a measure of the diameter of the decision space.

The result from Equation (A.26) is then integrated with other related bounds from the literature, which include:

- The task adaptation regret bound ((3.6)),
- The follow-the-regularized-leader (FTRL) regret bound ((3.7)),
- The initial task similarity bound ((3.8)).

The synthesis of these equations underpins the main claim of the theorem. The integration of these bounds effectively demonstrates that the parameterized utility function $\bar{U}^{\text{param}}(M)$ is bounded above by a term that decreases with the increase in the number of tasks M, thereby affirming the effectiveness of the learning algorithm under consideration. This result substantiates the theoretical foundation for the scalability and efficiency of the algorithm as M increases.

A.2.2 Proof of Theorem 3.5

The theorem addresses the aggregation of manifold regularization over multiple tasks and leverages advanced results from the online learning framework. Based on the methodology

provided by [62, Thm. 2] and the specific choice of the learning rate parameter,

$$\eta = \sqrt{\frac{\tau M + D_\Psi}{M d_\Psi D_\Psi G'^2}},$$

where τ represents the trade-off parameter between the regularization and the loss, M is the number of tasks, d_Ψ indicates the dimensionality of the manifold, D_Ψ is the diameter of the manifold space, and G' is the bound on the gradient of the loss function.

Applying these parameters, the average regret bound across all tasks can be computed as:

$$\frac{1}{M}\sum_{i=1}^{M} U_i'(\psi_i^1) - U_i'(\psi) \lesssim \sqrt{\frac{d_\Psi^3 D_\Psi G'^2(\tau M + D_\Psi)}{M}} + \gamma + \tau d_\Psi G', \tag{A.27}$$

where ψ_i^1 denotes the initial parameter settings for the i-th task, and ψ represents a baseline parameter setting against which the regret is measured.

The bound given in Equation (A.27) integrates several factors influencing the learning dynamics on the manifold, namely the complexity of the manifold (through d_Ψ and D_Ψ), the scaling with the number of tasks (M), and the inherent variations in task characteristics (captured by G' and τ).

The concluding result of the theorem is drawn by linking this bound with the definition of the optimal manifold regularization value, V^*_{manifold}, which encapsulates the minimum possible average deviation from the optimal utility function over all tasks. Thus, the result underscores the efficacy of the manifold learning approach in minimizing the regret over a diverse set of tasks, supporting the theorem's claim on the benefits of this approach in a multi-task learning setting.

A.2.3 Proof of Theorem 3.6

Let $R_M^{\mathrm{init}}(x) \geq \frac{1}{M}\sum_{i=1}^M U_i(x_i^1) - U_i(x)$ denote the regret upper bound compared to a static decision $x \in \mathcal{X}$ when employing FTRmL on the set $\{U_i\}_{i\in[M]}$. Similarly, define $R_M^{\mathrm{mani}}(\psi)$ and $R_M^{\mathrm{sim}}(\kappa)$ as the regret upper bounds against comparators ψ and κ, respectively, when utilizing FTPmL and FTRL on $\{U_i'\}_{i\in[M]}$ and $\{U_i^{\mathrm{sim}}\}_{i\in[M]}$. Under these definitions, the following inequalities are established:

$$\begin{aligned}
\bar{R}(M,T) &\leq \frac{1}{M}\sum_{i=1}^M \kappa_i U_i(x_i^1) + (1-\kappa_i)U_i'(\psi_i^1) \\
&\leq \min_{\kappa\in[0,1]} R_M^{\mathrm{sim}}(\kappa) + \sum_{i=1}^M \kappa U_i(x_i^1) + (1-\kappa)U_i'(\psi_i^1) \\
&\leq \min_{\kappa\in[0,1]} R_M^{\mathrm{sim}}(\kappa) + \kappa \underbrace{\left(\min_{x\in\mathcal{X}} R_M^{\mathrm{init}}(x) + \sum_{i=1}^M U_i(x)\right)}_{(i)} + (1-\kappa)\underbrace{\left(\min_{\psi\in\Psi} R_M^{\mathrm{mani}}(\psi) + \sum_{i=1}^M U_i'(\psi)\right)}_{(ii)},
\end{aligned}$$

where the first inequality is justified by (3.5) and (3.9), and the subsequent inequalities are derived from the definitions of $R_M^{\mathrm{sim}}(\kappa)$, $R_M^{\mathrm{init}}(x)$, and $R_M^{\mathrm{mani}}(\psi)$. The terms in the inequalities are further bounded by Theorems 3.4 and 3.5:

$$(i) \lesssim \frac{1}{\sqrt{M}} + V_{\mathrm{init}}^*, \quad (ii) \lesssim \sqrt{\frac{d_\Psi^3 D_\Psi G'^2(\tau M + D_\Psi)}{M}} + \gamma + \tau d_\Psi G' + V_{\mathrm{manifold}}^*.$$

Given that $R_M^{\mathrm{sim}}(\kappa) \lesssim \frac{1}{\sqrt{M}}$ for FTRL, these bounds collectively confirm the theorem's assertion.

A.3 Experiment Details

Humanoid. The Humanoid simulation involves a 3D bipedal robot designed to mimic human locomotion. This robot comprises a torso (abdomen) with two arms and two legs, where each limb segment is articulated at the joints (knees and elbows). For our experiments, the manifold dimension (n) is set at 50, significantly lower than the ambient dimension (d = 6392). Task variations are introduced by specifying different desired forward directions for the robot, randomly selected from a uniform distribution $\mathcal{U}[-\frac{\pi}{2}, \frac{\pi}{2}]$. Each experimental episode is capped at a maximum length of 250 time steps.

Half-Cheetah. The Half-Cheetah model is a simplified 2-dimensional robotic simulation featuring 9 links and 8 joints, mimicking a cheetah's running motion. Here, the manifold dimension is also set at (n=50), against an ambient dimension of (d = 119). Variability in the tasks is generated by assigning different goal velocities, randomly sampled from a uniform distribution $\mathcal{U}[0, 2]$. The duration for each episode in these experiments is set to 1000 time steps.

Hopper. The Hopper environment consists of a 2D one-legged robot that aims to learn hopping behavior. The robot has a total of 4 joints, including the hip, knee, and ankle joints, as well as a sliding joint between the foot and the ground. The manifold dimension (n) is set to 50, while the ambient dimension is (d = 33). Task variations are introduced by modifying the target hopping height, which is randomly sampled from a uniform distribution $\mathcal{U}[0.5, 1.5]$. Each episode has a maximum length of 1000 time steps.

Walker2d. The Walker2d environment features a 2D bipedal robot with a torso and two legs. The robot has 6 joints in total, including hip, knee, and ankle joints for each leg. The manifold dimension is set to (n=50), and the ambient dimension is (d = 21). Task variations are generated by altering the desired walking speed, which is randomly sampled

from a uniform distribution $\mathcal{U}[0.5, 2.5]$. The maximum episode length is set to 1000 time steps.

Swimmer. The Swimmer environment involves a 2D robot with 3 links connected by two joints. The goal is to learn swimming behavior by applying torques at the joints. The manifold dimension is set to (n=50), and the ambient dimension is (d = 10). Task variations are introduced by changing the target swimming speed, which is randomly sampled from a uniform distribution $\mathcal{U}[0.1, 0.5]$. Each episode has a maximum length of 1000 time steps.

Ant. The Ant environment consists of a 3D quadruped robot with 4 legs, each having two joints (hip and ankle). The robot also has a torso that connects the legs. The manifold dimension is set to (n=50), while the ambient dimension is (d = 125). Task variations are generated by modifying the desired walking direction, which is randomly sampled from a uniform distribution $\mathcal{U}[-\frac{\pi}{2}, \frac{\pi}{2}]$. The maximum episode length is set to 1000 time steps.

InvertedPendulum. The InvertedPendulum environment features a simple pendulum that starts in an upright position and needs to be balanced by applying torque at the joint connecting the pendulum to the ground. The manifold dimension is set to (n=50), and the ambient dimension is (d = 5). Task variations are introduced by changing the initial angle of the pendulum, which is randomly sampled from a uniform distribution $\mathcal{U}[-\frac{\pi}{8}, \frac{\pi}{8}]$. Each episode has a maximum length of 1000 time steps.

Rationale for setting the manifold dimension. In setting the manifold dimension (n=50) across our experiments, we align with the number of zeroth-order blackbox queries used for estimating search gradients. This choice is grounded in the practical heuristic that, absent advanced perturbation techniques (e.g., coordinate-wise perturbation), the number of necessary queries typically equals the dimensionality of the exploration space. Consequently, as we perform 50 policy gradient estimations per query in the simulated environ-

ments (termed as performing 50 rollouts),

Bibliography

[1] V. M. Zavala and M. Anitescu, "Real-time nonlinear optimization as a generalized equation," *SIAM Journal on Control and Optimization*, vol. 48, no. 8, pp. 5444–5467, 2010.

[2] E. Dall'Anese, S. S. Guggilam, A. Simonetto, Y. C. Chen, and S. V. Dhople, "Optimal regulation of virtual power plants," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1868–1881, 2017.

[3] A. Hauswirth, I. Subotić, S. Bolognani, G. Hug, and F. Dörfler, "Time-varying projected dynamical systems with applications to feedback optimization of power systems," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 3258–3263, IEEE, 2018.

[4] Y. Tang and S. Low, "Distributed algorithm for time-varying optimal power flow," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 3264–3270, IEEE, 2017.

[5] Y. Ding, J. Lavaei, and M. Arcak, "Time-variation in online nonconvex optimization enables escaping from spurious local minima," *IEEE Transactions on Automatic Control*, 2021.

[6] J. Chen and V. K. Lau, "Convergence analysis of saddle point problems in time varying wireless systems—control theoretical approach," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 443–452, 2011.

[7] S. H. Low and D. E. Lapsley, "Optimization flow control. i. basic algorithm and convergence," *IEEE/ACM Transactions on networking*, vol. 7, no. 6, pp. 861–874, 1999.

[8] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 7195–7201, IEEE, 2016.

[9] T. Yang, L. Zhang, R. Jin, and J. Yi, "Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient," in *International Conference on Machine Learning*, pp. 449–457, PMLR, 2016.

[10] A. Balavoine, C. J. Rozell, and J. Romberg, "Discrete and continuous-time soft-thresholding for dynamic signal recovery," *IEEE Transactions on Signal Processing*, vol. 63, no. 12, pp. 3165–3176, 2015.

[11] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[12] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[13] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[14] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*, pp. 1126–1135, PMLR, 2017.

[15] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *International Conference on Machine Learning*, pp. 1920–1930, PMLR, 2019.

[16] G. Denevi, C. Ciliberto, R. Grazzi, and M. Pontil, "Learning-to-learn stochastic gradient descent with biased regularization," in *International Conference on Machine Learning*, pp. 1566–1575, PMLR, 2019.

[17] M.-F. Balcan, M. Khodak, and A. Talwalkar, "Provable guarantees for gradient-based meta-learning," in *International Conference on Machine Learning*, pp. 424–433, PMLR, 2019.

[18] M.-F. Balcan, A. Blum, and S. Vempala, "Efficient representations for lifelong learning and autoencoding," in *Conference on Learning Theory*, pp. 191–210, PMLR, 2015.

[19] A. Maurer, M. Pontil, and B. Romera-Paredes, "The benefit of multitask representation learning," *Journal of Machine Learning Research*, vol. 17, no. 81, pp. 1–32, 2016.

[20] S. S. Du, W. Hu, S. M. Kakade, J. D. Lee, and Q. Lei, "Few-shot learning via learning the representation, provably," *arXiv preprint arXiv:2002.09434*, 2020.

[21] N. Tripuraneni, M. Jordan, and C. Jin, "On the theory of transfer learning: The importance of task diversity," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7852–7862, 2020.

[22] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, pp. 2902–2911, PMLR, 2017.

[23] L. Kirsch, S. Flennerhag, H. v. Hasselt, A. Friesen, J. Oh, and Y. Chen, "Introducing symmetries to black box meta reinforcement learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 7202–7210, Jun. 2022.

[24] A. Gosavi *et al.*, *Simulation-based optimization.* Springer, 2015.

[25] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26, 2017.

[26] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, 2019.

[27] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[28] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[29] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[30] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Feitas, "Bayesian optimization in a billion dimensions via random embeddings," *Journal of Artificial Intelligence Research*, vol. 55, pp. 361–387, 2016.

[31] K. M. Choromanski, A. Pacchiano, J. Parker-Holder, Y. Tang, and V. Sindhwani, "From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] O. Sener and V. Koltun, "Learning to guide random search," in *International Conference on Learning Representations*, 2020.

[33] P.-A. Absil, R. Mahony, and R. Sepulchre, "Optimization algorithms on matrix manifolds," in *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, 2009.

[34] B. Amos, "Tutorial on amortized optimization for learning to optimize over continuous domains," *arXiv preprint arXiv:2202.00665*, 2022.

[35] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[36] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[37] P. Micaelli and A. Storkey, "Non-greedy gradient-based hyperparameter optimization over long horizons," 2020.

[38] K. Young, B. Wang, and M. E. Taylor, "Metatrace: Online step-size tuning by meta-gradient descent for reinforcement learning control," *arXiv preprint arXiv:1805.04514*, 2018.

[39] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *International Conference on Machine Learning*, pp. 1568–1577, PMLR, 2018.

[40] A. Antoniou, H. Edwards, and A. Storkey, "How to train your maml," *arXiv preprint arXiv:1810.09502*, 2018.

[41] Z. Gao, Y. Wu, M. T. Harandi, and Y. Jia, "Curvature-adaptive meta-learning for fast

adaptation to manifold data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[42] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl ^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[43] A. L. Custódio, K. Scheinberg, and L. Nunes Vicente, "Methodologies and software for derivative-free optimization," *Advances and trends in optimization with engineering applications*, pp. 495–506, 2017.

[44] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. SIAM, 2009.

[45] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg, "Linear interpolation gives better gradients than gaussian smoothing in derivative-free optimization," *arXiv preprint arXiv:1905.13043*, 2019.

[46] N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein, "Guided evolutionary strategies: Augmenting random search with surrogate gradients," in *International Conference on Machine Learning*, pp. 4264–4273, PMLR, 2019.

[47] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

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

[49] B. Addis, M. Locatelli, and F. Schoen, "Local optima smoothing for global optimization," *Optimization Methods and Software*, vol. 20, no. 4-5, pp. 417–437, 2005.

[50] J. C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, 2005.

[51] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[52] O. Shamir, "An optimal algorithm for bandit and zero-order convex optimization with two-point feedback," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1703–1713, 2017.

[53] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: The power of two function evaluations," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2788–2806, 2015.

[54] K. G. Jamieson, R. Nowak, and B. Recht, "Query complexity of derivative-free optimization," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[55] P. G. Constantine, *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM, 2015.

[56] H. B. Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters, "Generalization of human grasping for multi-fingered robot hands," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2043–2050, IEEE, 2012.

[57] A. Colomé and C. Torras, "Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 602–615, 2018.

[58] S. Tosatto, G. Chalvatzaki, and J. Peters, "Contextual latent-movements off-policy optimization for robotic manipulation skills," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10815–10821, IEEE, 2021.

[59] N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein, "Guided evolutionary strategies: augmenting random search with surrogate gradients," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 4264–4273, PMLR, 09–15 Jun 2019.

[60] J. Zhang, H. Tran, D. Lu, and G. Zhang, "A novel evolution strategy with directional gaussian smoothing for blackbox optimization," *arXiv preprint arXiv:2002.03001*, 2020.

[61] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[62] A. S. Suggala and P. Netrapalli, "Online non-convex learning: Following the perturbed leader is optimal," in *Algorithmic Learning Theory*, pp. 845–861, PMLR, 2020.

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

[64] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[65] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10657–10665, 2019.

[66] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, PMLR, 2013.

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

[68] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, *et al.*, "Optimization with sparsity-inducing penalties," *Foundations and Trends® in Machine Learning*, vol. 4, no. 1, pp. 1–106, 2012.

[69] R. Bellman, "Dynamic programming," *science*, vol. 153, no. 3731, pp. 34–37, 1966.

[70] I. T. Jolliffe, "Principal component analysis," *Technometrics*, vol. 45, no. 3, p. 276, 2003.

[71] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[72] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[73] N. Agarwal, A. Gonen, and E. Hazan, "Learning in non-convex games with an optimization oracle," in *Conference on Learning Theory*, pp. 18–29, PMLR, 2019.

[74] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, IEEE, 2012.

[75] B. Sel, A. Al-Tawaha, Y. Ding, R. Jia, R. Jia, B. Ji, J. Lavaei, and M. Jin, "Learning-to-learn to guide random search: Derivative-free meta blackbox optimization on manifold," 2022.

[76] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems," *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.