A Model Selection Framework for Learning Rate-Free Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

The success of many reinforcement learning algorithms is dependent on the right choice of hyperparameters, with the learning rate being particularly influential. A suboptimal learning rate can hinder the algorithm's ability to converge. Mildly suboptimal choices may allow the algorithm to find an optimal policy only after requiring an extensive number of samples. In this work, we show the feasibility of using model selection meta-learning algorithms to select the best learning rates in reinforcement learning problems. We introduce the Model Selection Framework for Learning Rate-Free Reinforcement Learning and evaluate various model selection algorithms within our framework. Our results show that data-driven model selection strategies such as the D^{3}RB algorithm achieve better performance in the problem of learning rate selection for reinforcement learning algorithms, beating bandit strategies such as EXP3, and also standard hyperparameter selection methods such as the uniform sweep.

1 Introduction

The learning rate used in the optimization phase of many machine learning methods determines how much a model adjusts its internal parameters at each step. A well-chosen learning rate balances training speed and stability, ideally leading the model to converge to the global optimum. Due to its fundamental role in optimization, the learning rate is crucial for efficient training and model performance across nearly all machine learning domains. Theoretical analysis of optimization problems [4] suggests that an effective learning rate is closely tied to the distance between the optimizer's current step and the optimal point. Since this information is typically unknown, many modern optimization methods adjust the learning rate dynamically depending on the stage of learning [6]. In practice, many methods just set a constant learning rate prior to training accompanied by rate schedulers that are used to modify the learning rate throughout training. [8]

In this work, we focus on sequential decision-making problems such as reinforcement learning [16, 25] where a learner interacts with the world in a sequential manner as is tasked with finding a policy that achieves large rewards. The reward collected by a learner at any point during training provides information on the quality of the current policy. In these frameworks, the learning rate determines the extent to which model parameters are adjusted based on this reward feedback. Empirical rewards, gathered in real-time from interactions between the agent and the environment, contain information about the agent's proximity to the optimal policy and its stage of learning. Building on this intuition, we design a framework to utilize the empirical reward, available at no extra overhead during training, that can be used to adjust the learning rate. In this work, we introduce a framework that combines model selection methods with a general scheme of reinforcement learning algorithms to adaptively tune the learning rate. Model selection algorithms are uniquely suitable for tuning learning rates in decision-making frameworks:

1. Model selection methods are adaptive by design and learn not to choose deficient learning rates as frequently. We show that by regret balancing, model selection will not select an ill-performing learning rate for more than $\sqrt{N}$ episodes in a single run, where N is the total number of episodes.
2. Using model selection for tuning learning rates in reinforcement learning is more sample efficient compared to performing a uniform learning rate sweep. This is because model selection algorithms advance the state of each hyperparameter curve adaptively, thus not requiring the same amount of samples and compute for all choices of learning rate.

Model selection has proved to have several interesting theoretical guarantees, yet most of these theoretical results have not been deployed in application. In this paper, we deploy model selection for the task of learning rate tuning in reinforcement learning. The paper is organized as follows. In section 2, We cover the preliminaries of model selection and reinforcement learning. We introduce the specific model selection strategies that we use in our experiments, and then present a general scheme of policy optimization and Q-learning in reinforcement learning algorithms. Sections 3 and 4 demonstrate the formalization and algorithmic interface of the model selection framework for learning rate-free reinforcement learning, respectively. Finally, we provide the experiments for tuning the learning rate in PPO and DQN with model selection strategies. We analyze the results in section 5 and cover experiment details and full plots in the Appendix B.

2 Preliminaries and Background

2.1 Model Selection

In many machine learning domains, including reinforcement learning the true configuration of the problem is not known in advance. The goal of model selection is to consider several configurations and add a strategy on top that learns to pick up the best configuration adaptively. We call each configuration a base and refer to the model selection strategy as the meta-learner. The meta-learner has access to a set of m bases, in this case, different copies of the same reinforcement learning algorithm instantiated with different learning rates. In each round, $n = 1, 2, \ldots, N$, of the interaction between the meta-learner with the environment, the meta-learner selects a base $i_n \in [m]$ to play and follows its policy. Learner i_n's internal state is then updated with the data collected from its interaction with the environment.

Here, we inherit a similar meta-learning structure to [1]. We experiment with a diverse range of model selection algorithms that were previously introduced in the literature [1, 2, 5, 22, 23], including standard multi-armed bandit algorithms, regret balancing methods, etc. We investigate the Upper Confidence Bound (UCB)[2], the Exponential-weight algorithm for Exploration and Exploitation (EXP3) [5], and Corral [1, 23]. Corral is a meta-learning algorithm for selecting among multiple bandit algorithms. It is known that Stochastic Corral and EXP3 enjoy theoretical model selection guarantees [23] while unmodified UCB does not.

Regret balancing maintains an estimate of the empirical regret for each base and tries to equate the regret bounds across all the bases. In this approach, the base agent is selected for two reasons. It is either a well-performing base by achieving low regret, or it has not been played enough and the meta-learner hasn't collected adequate information on the performance of this base. Here, we investigate Doubling Data Driven Regret Balancing (D^{3}RB) [7], and the regret bound balancing algorithm [22] which we will refer to as the Classic Balancing algorithm.

Note that our model selection framework views these algorithms as a black box and does not require detailed knowledge of the underlying algorithm. Hence, the framework can be paired with various types of meta-learners and base algorithms.

2.2 Reinforcement Learning

Reinforcement learning is formalized as Markov Decision Process (MDP) $\langle S, A, R, P, \gamma \rangle$; where S denotes the set of states, A is the set of actions, $R : S \times A \to \mathbb{R}$ is the reward function, $P : S \times A \to [0, 1]$ is the dynamic transition probabilities, and lastly $\gamma \in [0, 1]$ is the discount factor. Here we consider episodic reinforcement learning with maximum horizon T where the goal of the agent is to learn the (near) optimal policy $\pi : S \rightarrow A$. The state-value function $V : S \rightarrow \mathbb{R}$ and action-value function $Q : S \times A \rightarrow \mathbb{R}$ with respect to policy π are defined as

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) | s_0 = s, s_t, a_t\right] \tag{1}$$

$$Q^{\pi}(s, a) = R(s, a) + \gamma\, \mathbb{E}_{s' \sim P(s,a)}\left[V^{\pi}(s')\right] \tag{2}$$

The policy π is commonly parameterized by the set of parameters θ, and is denoted as π_θ. Two of the predominant approaches for learning the (near) optimal policy in reinforcement learning are policy optimization and Q-learning. Policy optimization starts with an initial policy and in each episode updates the parameters by taking gradient steps toward maximizing the episodic return. Denote learning rate as $\alpha \in \mathbb{R}$, a common update rule in policy optimization methods is

$$\theta \leftarrow \theta + \alpha\, \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t)(Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t))\right] \tag{3}$$

Q-learning uses the temporal differences method to update the parameters of Q^{π_θ}. A common update rule is

$$\theta \leftarrow \theta + \alpha\, \mathbb{E}_{s,a,s',r \sim D}\left[\nabla_\theta (r + \gamma \max_{a' \in A} Q^{\pi_{\bar{\theta}}}(s', a') - Q^{\pi_\theta}(s, a))^2\right] \tag{4}$$

where D is the experience replay buffer and $\bar{\theta}$ is a frozen parameter set named target parameter. Proximal Policy Optimization (PPO) [24] and Deep Q-Networks (DQN) [20] are among the most popular deep reinforcement learning algorithms that follow the first and second approaches, respectively.

Algorithm 1: Model Selection Interface for Hyperparameter Tuning

```
Input: $m$, $\beta$, $\Psi$, $M$
1
2  Function sample():
       // Select base index according to $\Psi$
3      $i \sim \Psi$
4      $\pi^i, \alpha^i \leftarrow \beta^i$
5      return $i, \pi^i, \alpha^i$
6
7
8  Function update($index$, $R[1:T]$):
       // Calculate and normalize the episodic return
9      $R_{norm} \leftarrow normalize(R[1:T])$
       // Update base statistics according to the meta learning algorithm $M$
10     $\Psi \leftarrow M(\Psi, index, R_{norm})$
11
```

3 Problem Statement

We formalize the model selection framework for learning rate-free reinforcement learning as the tuple $\langle m, \beta, M, \Psi \rangle$ where m is the number of base agents, $\beta = \{\beta^1, \ldots, \beta^m\}$ denotes the set of base agents where $\beta^i = \langle \alpha^i, \pi^i \rangle$ $(1 \leq i \leq m)$ consists of learning rate α^i, and policy π^i. Lastly, M is the model selection strategy and Ψ is an attribute of M that expresses some statistics over the base agents. For instance, Ψ can either be a distribution $\Psi : \beta \rightarrow P(\beta)$ over base agents or represent the estimated empirical regret of the base agents.

At the beginning of each episode, the meta learner M selects base agent β^j according to Ψ. We abbreviate this as $j \sim \Psi$. The base agent interacts with the environment in a typical reinforcement learning manner for one episode. At state $s_t \in S$, the base agent takes action $a_t \sim \pi^j$, receives reward $r_t \in (0, 1)$, and move to the next state $s_{t+1} \in S$ following the environment transition dynamics. At the end of each episode, the base agent passes the realized rewards $(r_1, \ldots, r_T)$ to the meta learner, so that it updates Ψ based on the model selection strategy M.

The goal of the base agents is to interact with the environment and learn an optimal policy for the reinforcement learning problem. The goal of the meta learner is to learn a strategy to iteratively select base agents, so that agents with better learning rates are played more frequently. It's unique to model selection that neither the base agents with good learning rates nor the optimal reinforcement strategy are known in advance, and the framework learns both of them during a single run.

Algorithm 2: Learning Rate-Free Reinforcement Learning with Model Selection

```
Input: MDP $\langle S, A, R, P, \gamma \rangle$, Model Selection Interface $\mathfrak{M}$
1
// reinforcement learning loop over episode
2  for $n = 1, 2, ..., N$ do
3
       // Select the base agent
4      $i, \pi_\theta^i, \alpha^i = \mathfrak{M}.sample()$
5
       // Collect trajectories with selected base agent
6      for $t = 1, 2, ..., T$ do
7          $a \sim \pi_\theta^i$
8          $r, s' \xleftarrow{P,R} s, a$
9          $R[t] \leftarrow r$
10
       // Update parameters with selected learning rate
11     if Policy Optimization then
12         $\theta \leftarrow \theta + \alpha^i \, \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta^i(s_t, a_t)(Q^{\pi_\theta^i}(s_t, a_t) - V^{\pi_\theta^i}(s_t))\right]$
13     if Q-Learning then
14         $\theta \leftarrow \theta + \alpha^i \, \mathbb{E}_{s,a,s',r \sim D}\left[\nabla_\theta (r + \gamma \, \max_{a' \in A} Q^{\pi_\theta^i}(s', a') - Q^{\pi_\theta^i}(s, a))^2\right]$
15
       // Update the meta learner
16     $\mathfrak{M}.update(i, R[1:T])$
```

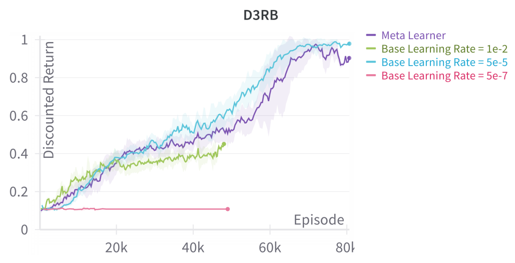
4 Method

We begin with a predefined set of learning rates $\alpha^i \in [\alpha^1, ..., \alpha^m]$, and we initiate m reinforcement learning agents $[\beta^1, \beta^2 .., \beta^m]$ of the same type. All hyperparameters and configurations of the base agents are similar except for the learning rate, where the learning rate of β^i is set to α^i for all $1 \leq i \leq m$.

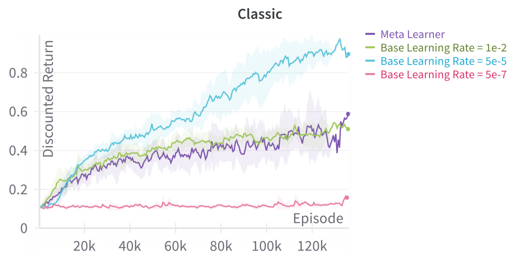
The model selection framework for learning rate-free reinforcement learning integrates the model selection interface for hyperparameter tuning with the reinforcement learning loop. The model selection interface, represented in Algorithm 1, consists of two procedures $sample$, and $update$ that the meta learner uses to select the base agent at the beginning of each episode, and update Ψ at the end of it. The integrated reinforcement learning loop, which we will refer to as Learning Rate-Free Reinforcement Learning, is shown in Algorithm 2. The algorithm contains the original agent-environment interaction in addition to the model selection components.

5 Experiments and Results

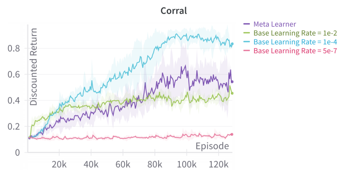
We begin our experiments with learning rate-free PPO. We initiate ten PPO base agents learning rates $\alpha = [1e^{-2}, 5e^{-3}, 1e^{-3}, 5e^{-4}, 1e^{-4}, 5e^{-5}, 1e^{-5}, 5e^{-6}, 1e^{-6}, 5e^{-7}]$. We run the experiment for five model selection strategies introduced in Section 2.



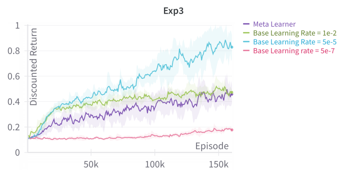
(a) D^{3}RB



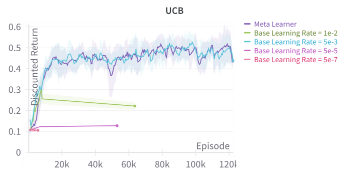
(b) Classic Balancing



(c) Corral



(d) EXP3



(e) UCB

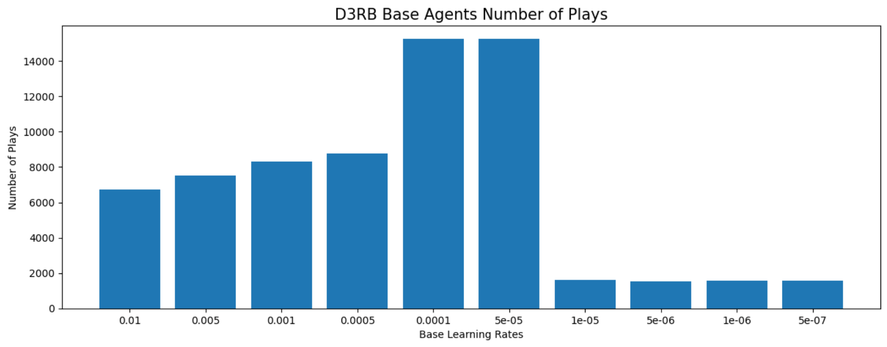
Figure 1: Learning Rate-Free PPO on Humanoid environment. Discounted return per episode across 5 model selection strategies; each curve showing the mean and standard deviation over three independent runs. The purple curve belongs to the learning rate-free PPO which demonstrates the advancement of the meta learner. Other curves show the advancement of a subset of the base agents.

Figure 1 represents our main findings. Each plot includes the episodic return for the meta learner (purple curve) and three of the base agents. Full plots showing the performance of all base agents are available in Appendix B. By comparing the meta learners, we can see that D^{3}RB strategy achieved the lowest regret and had the most advancement in the task. Figure 1(a) demonstrates that the meta learner with D^{3}RB strategy is performing nearly as good as to the best-performing base agent (blue curve). The plot also reflects the fact that D^{3}RB is learning not to select the ill-performing learning rates as often.

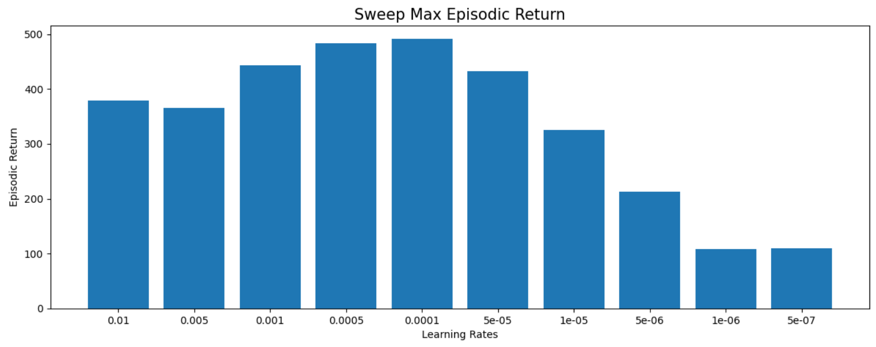
The meta learner with Classic Balancing strategy, shown in Figure 1(b) is working comparable to an average-performing base agent. The original algorithm proposes the elimination of the miss-specified base agents in order to succeed, whereas in this experiment we observed that the strategy didn't eliminate any base agents and played them with the same probability. This might be due to the sub-optimal putative bound input of the model selection strategy. More details about the Classic Balancing method are available in Appendix C.

Figures 1(c, d, e) show that bandit algorithms are not achieving the same good results as D^{3}RB. One drawback of applying bandit algorithms as meta-learners for reinforcement learning base agents can be seen in these experiments. Unlike standard multi-armed bandits where the mean rewards are stationary, rewards in reinforcement learning can depend on the state of learning. A specific choice of learning rate might achieve low rewards in the early stages of learning and be able to perform well later on. In contrast, another choice of learning rate might significantly advance in the beginning and slow down after a while. Standard bandit algorithms are not able to distinguish these state-dependant changes and therefore not be able to adapt to the best choice of learning rate during training. This lies at the heart of why the design of effective algorithms for model selection is a challenging problem and why typical multi-armed bandit algorithms do not possess provable guarantees for this problem setting.

These experiments further point out the capabilities of data-driven methods for the task of hyperparameter tuning. As D^{3}RB achieved the best performance in this task, we compared it to the sweep strategy over the same set of learning rates. For sweep, we initiate ten independent PPO agents with the same set of learning rates that we input to the model selection counterparts. We run each agent for approximately $\sim \frac{1}{10}$ fraction of total episodes in model selection experiments. Figure 2 demonstrates the results of this comparison. Figure 2(a) shows the number of episodes that the meta learner with D^{3}RB strategy has selected each base agent throughout the training. Figure 2(b) shows the maximum value of episodic return achieved by independent PPO agents. We can see that D^{3}RB strategy for



(a) D^{3}RB



(b) Sweep

Figure 2: D^{3}RB selection statistics are reflecting the results of sweep over the same learning rates

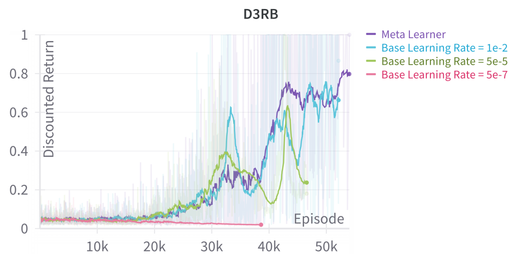
learning rate-free RL has learned to select the agents with higher reward (and lower regret) more frequently. Additionally, we can see that D^{3}RB is not choosing deficient learning rates as often. Check Appendix A for a theoretical explanation of this.

The same experiments are done for Learning Rate-Free DQN. The results are available at Figure 3. We can see that for simpler reinforcement learning tasks like the classic control environments, the bandit strategies like UCB and Corral were able to perform well. Full plots are available in Appendix B.

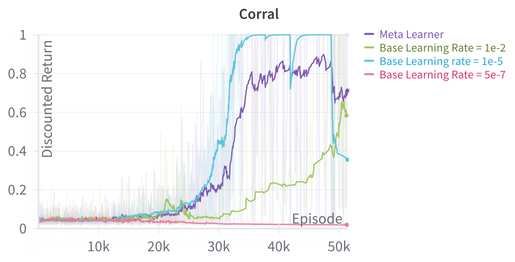
6 Related Work

Random Search [3] aimed to improve the exhaustive heuristics for hyperparameter tuning by considering a random subset of all possible hyperparameters. [26] formulated hyperparameter tuning as an optimization problem and used Bayesian inference to adaptively update the hyperparameters. Several other papers have studied parameter-free learning from the perspective of optimization [8, 15, 19]. These works successfully proposed learning rate schedulers for common optimizers like Adam [9] and SGD. Among prior the closest to our work is [18] which formulates hyperparmeter optimization as an infinite-arm bandit problem, but doesn't apply their method to reinforcement learning.

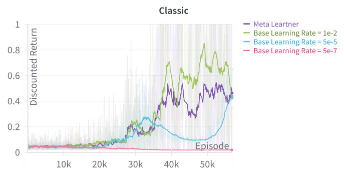
The model selection problem has been studied in both online and offline fashion in reinforcement learning [10, 11, 13, 21]. Despite the great capacities of model selection in machine learning applications, most of the prior works focused on the theoretical aspects of model selection [17], and only a few considered model selection in more practical problems such as feature selection [14, 21].



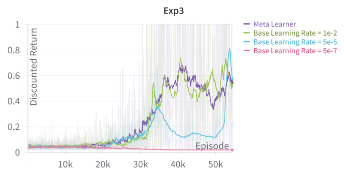
(a) D^{3}RB



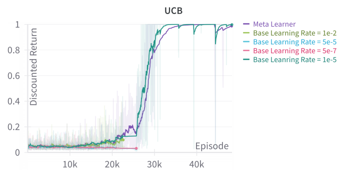
(b) Corral



(c) Classic Balancing



(d) EXP3



(e) UCB

Figure 3: Learning Rate-Free DQN on CartPole environment. Discounted return per episode across 5 model selection strategies. The purple curve belongs to the learning rate-free DQN which demonstrates the advancement of the meta learner. Other curves show the advancement of a subset of the base agents.

7 Conclusion and Future Work

We proposed a model selection framework for learning rate-free reinforcement learning and demonstrated its effectiveness using five model selection strategies. Our experiments showed that the data-driven regret balancing method, D^{3}RB generally serves as a good model selection strategy for learning rate-free reinforcement learning, consistently performing well across our tests. In contrast, bandit strategies appeared to be insufficient as meta-learners for PPO base agents.

There are several possible extensions to this work. One direction is studying the effect of sharing data across the base agents on the sample efficiency of the framework. Another direction is to come up with a framework that performs learning rate selection on a single instance of a reinforcement learning base agent. This can significantly improve the memory efficiency of the framework for deploying it in models of larger scale.

References

[1] Alekh Agarwal, Haipeng Luo, Behnam Neyshabur, and Robert E Schapire. Corralling a band of bandit algorithms. In *Conference on Learning Theory*, pages 12–38. PMLR, 2017.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[6] Ashok Cutkosky, Aaron Defazio, and Harsh Mehta. Mechanic: A learning rate tuner. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] Chris Dann, Claudio Gentile, and Aldo Pacchiano. Data-driven online model selection with regret guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 1531–1539. PMLR, 2024.

[8] Aaron Defazio, Ashok Cutkosky, Harsh Mehta, and Konstantin Mishchenko. When, why and how much? adaptive learning rate scheduling by refinement. *arXiv preprint arXiv:2310.07831*, 2023.

[9] P Kingma Diederik. Adam: A method for stochastic optimization. *(No Title)*, 2014.

[10] Amir-massoud Farahmand and Csaba Szepesvári. Model selection in reinforcement learning. *Machine learning*, 85(3):299–332, 2011.

[11] Assaf Hallak, Dotan Di-Castro, and Shie Mannor. Model selection in markovian processes. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–382, 2013.

[12] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

[13] Nan Jiang, Alex Kulesza, and Satinder Singh. Abstraction selection in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 179–188. PMLR, 2015.

[14] Parnian Kassraie, Nicolas Emmenegger, Andreas Krause, and Aldo Pacchiano. Anytime model selection in linear bandits. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Ahmed Khaled and Chi Jin. Tuning-free stochastic optimization. *arXiv preprint arXiv:2402.07793*, 2024.

[16] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[17] Jonathan Lee, Aldo Pacchiano, Vidya Muthukumar, Weihao Kong, and Emma Brunskill. Online model selection for reinforcement learning with function approximation. In *International Conference on Artificial Intelligence and Statistics*, pages 3340–3348. PMLR, 2021.

[18] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[19] Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. *arXiv preprint arXiv:2306.06101*, 2023.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[21] Aditya Modi, Nan Jiang, Ambuj Tewari, and Satinder Singh. Sample complexity of reinforcement learning using linearly combined model ensembles. In *International Conference on Artificial Intelligence and Statistics*, pages 2010–2020. PMLR, 2020.

[22] Aldo Pacchiano, Christoph Dann, Claudio Gentile, and Peter Bartlett. Regret bound balancing and elimination for model selection in bandits and rl. *arXiv preprint arXiv:2012.13045*, 2020.

[23] Aldo Pacchiano, My Phan, Yasin Abbasi Yadkori, Anup Rao, Julian Zimmert, Tor Lattimore, and Csaba Szepesvari. Model selection in contextual stochastic bandit problems. *Advances in Neural Information Processing Systems*, 33:10328–10337, 2020.

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

[25] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[26] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

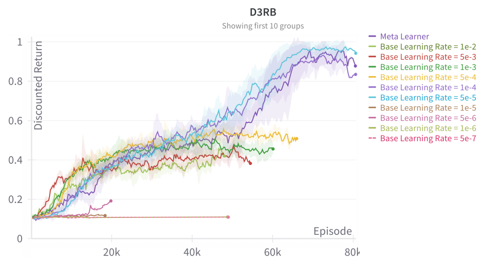
Appendix

A. Theoretical Remarks

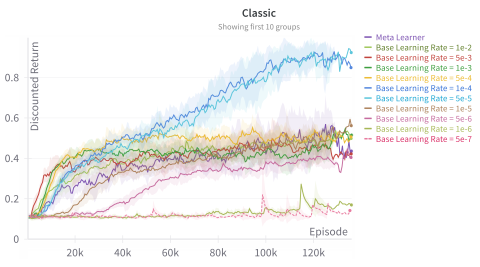
Regret balancing strategies strive to equate the regret of the different algorithms. Typically it is assumed the optimal algorithm's regret scales as $d_\star\sqrt{t}$, where $d_\star$ is the regret coefficient. In contrast, the regret of a linearly sub-optimal algorithm scales as Δt for some constant Δ. Without loss of generality let's call these two algorithms, Algorithm A and Algorithm B. A regret balancing strategy ensures that at episode N the number of episodes Algorithm A and Algorithm B were played, N_A, and N_B satisfy $d_\star\sqrt{N_A} \approx \Delta N_B$ thus implying that $N_B \approx \frac{d_\star\sqrt{N_A}}{\Delta} = \mathcal{O}\left(\frac{d_\star\sqrt{N}}{\Delta}\right)$.

B. Experiments/Plots

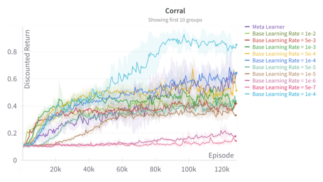
We use cleanRL library [12] for the implementation of RL algorithms. Implementaions of the framework and model selection strategies are available here: `https://github.com/Kinda-Anonymous/Learning-Rate-Free-Reinforcement-Learning`



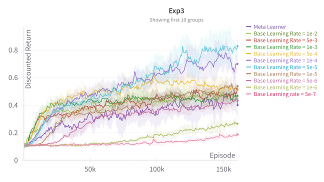
(a) D^{3}RB



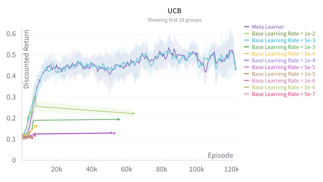
(b) Classic Balancing



(c) Corral



(d) EXP3



(e) UCB

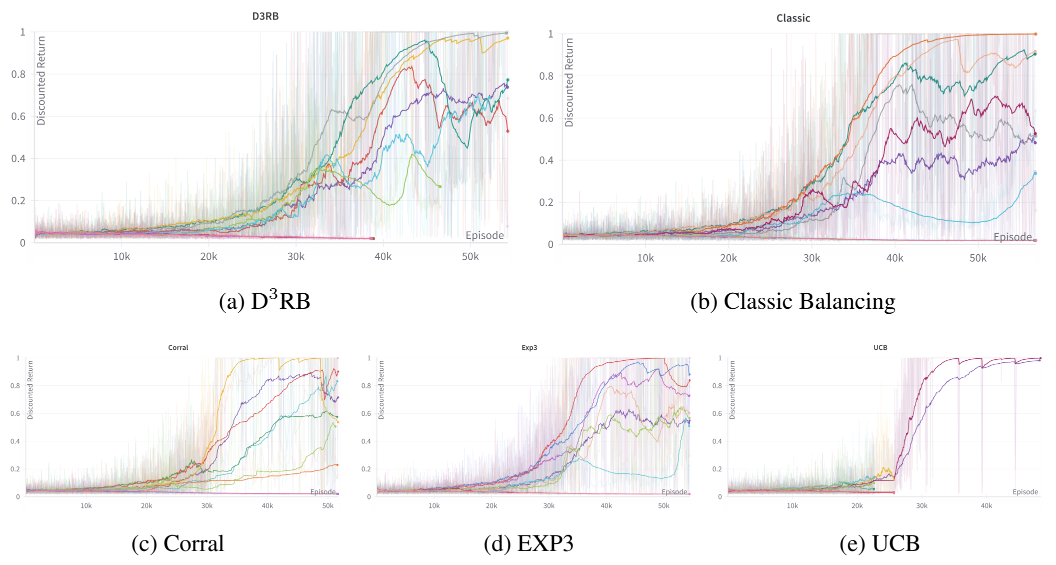
Figure 4: Learning Rate-Free PPO on Humanoid environment. Discounted return per episode across 5 model selection strategies.

C. Model Selection Algorithms Pseudocodes

Denote the number of times that the base agent i was played up to this time as n^i. Denote the regret coefficient of base learner i as d^i, and the total reward accumulated by base learner i up to this time by u^i.

D^{3}RB

Doubling Data Driven Regret Balancing (D^{3}RB) [7] tries to maintain and equal empirical regret for all the base agents. Denote the balancing potential of base agent i as $\Psi^i = d^i\sqrt{n^i}$. The D^{3}RB algorithm for learning rate-free RL works as follows,



(a) D^{3}RB (b) Classic Balancing

(c) Corral (d) EXP3 (e) UCB

Figure 5: Learning Rate-Free DQN on CartPole environment. Discounted return per episode across 5 model selection strategies for all 10 base agents. Purple curve belongs to the meta learner.

```
Algorithm 3: D^3RB
Input: $m$, $\beta$, $\Psi$, $\delta$

Function sample():
    // Sample base index
    $i = \arg\min_j \Psi_j$
    $\pi_i, \alpha_i \leftarrow \beta_i$
    return $i, \pi_i, \alpha_i$

Function update($i$, $R[1:T]$):
    $R_{norm} \leftarrow normalize(R[1:T])$
    // Update Statistics
    $u^i = u^i + R_{norm}$
    $n^i = n^i + 1$
    // Perform miss-specification test
    $\frac{u^i}{n^i} + \frac{d^i\sqrt{n^i}}{n^i} + c\sqrt{ln\frac{\frac{mlnn^i}{\delta}}{n^i}} \leq max_j \frac{u^j}{n^j} - c\sqrt{ln\frac{\frac{Mlnn^j}{\delta}}{n^j}}$
    // If test triggered double regret coefficient for base i
    $d^i \leftarrow 2d^i$
    // Update balancing potential
    $\Psi^i = d^i\sqrt{n^i}$
```

Classic Balancing

The Classic Regret Balancing Algorithm [22] starts with the full set of base agents $\beta = [\beta_1, ..., \beta_m]$, at each round the algorithm performs miss-specification on each of the base agents and eliminates the miss-specified one. Denote Ψ^j as empirical regret upper bound of base agent j.

Algorithm 4: Classic Balancing

```
Input: $m$, $\beta$, $\Psi$, $\delta$

Function sample():
    // Sample Base index
    $i = \arg\min_j \Psi_j$
    $\pi_i, \alpha_i \leftarrow \beta_i$
    return $i, \pi_i, \alpha_i$

Function update($i$, $R[1:T]$):
    $R_{norm} \leftarrow normalize(R[1:T])$
    // Update statistics
    $u^i = u^i + R_{norm}$
    $n^i = n^i + 1$
    // Perform miss-specification test for all the remaining base agents
    for $\beta_k \in \beta$ do
        $\frac{u^k}{n^k} + \frac{d^k\sqrt{n^k}}{n^i} + c\sqrt{ln\frac{\frac{mlnn^k}{\delta}}{n^k}} \leq max_j\frac{u^j}{n^j} - c\sqrt{ln\frac{\frac{Mlnn^j}{\delta}}{n^j}}$
        if miss-specified then
            $\beta \leftarrow \beta/\{\beta_k\}$
```

EXP3

Exponential-weight algorithm for exploration and exploitation (EXP3) learns a probability distribution $\Psi^i = \frac{exp(S^i)}{\sum_{j=1}^{m} exp(S^j)}$ over base learners, where S^i is a total estimated reward of base agent i up to this round.

Algorithm 5: EXP3

```
Input: $m$, $\beta$, $\Psi$, $\delta$

Function sample():

    // Sample Base index
    $i = \arg\max_j \Psi_j$
    $\pi_i, \alpha_i \leftarrow \beta_i$

    return $i, \pi_i, \alpha_i$

Function update($i$, $R[1:T]$):

    $R_{norm} \leftarrow normalize(R[1:T])$
    // Update statistics
    for $j \in 1, ..., m$ do
        $S^j = S^j + 1 - \frac{\mathbb{I}\{j=i\}(1-R_{norm})}{\Psi^i}$

    // Update Distribution
    $\Psi^i = \frac{exp(S^i)}{\sum_{j=1}^{m} exp(S^j)}$
```

Corral

Corral [1] learns a distribution Ψ over base agents and update it according to LOG-BARRIER-OMD algorithm. We skip the algorithmic details and refer to the updating rule mentioned in the original paper as Corral-Update.

Algorithm 6: Corral

```
Input: $m$, $\beta$, $\Psi$

Function sample():
    // Sample base index
    $i \sim \Psi$
    $\pi_i, \alpha_i \leftarrow \beta_i$

    return $i, \pi_i, \alpha_i$

Function update($i$, $R[1:T]$):
    $R_{norm} \leftarrow normalize(R[1:T])$
    // Update according to Corral
    $\Psi^j \leftarrow$ Corral-Update($R_{norm}$)
```

UCB

The Upper Confidence Bound algorithm (UCB) maintains an optimistic estimate of the mean for each arm [16]. Denote Ψ^i as the upper confidence bound of arm i. The UCB algorithm for learning rate-free RL works as follows,

Algorithm 7: UCB

```
Input: $m$, $\beta$, $\Psi$, $\delta$

Function sample():
    // Sample base index
    $i = \arg\max_j \Psi_j$
    $\pi_i, \alpha_i \leftarrow \beta_i$

    return $i, \pi_i, \alpha_i$

Function update($i$, $R[1:T]$):
    $R_{norm} \leftarrow normalize(R[1:T])$
    // Update statistics
    $u^i = u^i + R_{norm}$
    $n^i = n^i + 1$
    $\mu^i = \frac{u^i}{n^i}$
    // Update Upper Confidence Bounds
    $\Psi^i = UCB^i(\delta) = \mu^i + \sqrt{\frac{2log(1/\delta)}{n^i}}$
```