Learning Rate-Free Reinforcement Learning: A Case for Model Selection with Non-Stationary Objectives

Aida Afshar
Boston University
aafshar@bu.edu

Aldo Pacchiano
Boston University
Broad Institute of MIT and Harvard
pacchian@bu.edu

Abstract

The performance of reinforcement learning (RL) algorithms is sensitive to the choice of hyperparameters, with the learning rate being particularly influential. RL algorithms fail to reach convergence or demand an extensive number of samples when the learning rate is not optimally set. In this work, we show that model selection can help to improve the failure modes of RL that are due to suboptimal choices of learning rate. We present a model selection framework for Learning Rate-Free Reinforcement Learning that employs model selection methods to select the optimal learning rate on the fly. This approach of adaptive learning rate tuning neither depends on the underlying RL algorithm nor the optimizer and solely uses the reward feedback to select the learning rate; hence, the framework can input any RL algorithm and produce a learning rate-free version of it. We conduct experiments for policy optimization methods and evaluate various model selection strategies within our framework. Our results indicate that data-driven model selection algorithms are better alternatives to standard bandit algorithms when the optimal choice of hyperparameter is time-dependent and non-stationary.

1 Introduction

We focus on sequential decision-making problems such as reinforcement learning and bandits (Lattimore & Szepesvári, 2020; Sutton & Barto, 2018) where a learner interacts with the world in a sequential manner and is tasked with finding a policy that maximizes the reward. It is a common scenario in RL that the right choice of hyperparameters is not known in advance and the success of RL algorithms demands the effort of hyperparameter tuning (Eimer et al., 2023) to reach convergence. Among all the algorithm-specific hyperparameters, the learning rate is known to have a notable impact on the convergence of RL agents. The learning rate determines the extent to which the model parameters are adjusted at each optimization step, and prior literature on learning rate scheduling (Mishchenko & Defazio, 2023; Defazio et al., 2023) suggests that optimal learning rate is dependent on the *distance to the solution*. Central to RL, is the notion of reward that contains information about the proximity of the current policy to the optimal one. Building on this intuition that the reward feedback can be used as a proxy of the distance to the solution, we propose a framework that utilizes the empirical reward to adjust the learning rate on the fly during RL training. Model selection methods are inherently designed with the goal of detecting the right configuration of the problem and are uniquely suitable for the task of hyperparameter selection in RL;

1. Model selection algorithms advance the state of each hyperparameter curve adaptively, thus not requiring the same samples and compute for all hyperparameter choices. 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. Standard methods for hyperparameter tuning such as Bayesian Optimization (Wu et al., 2019) or Random Search (Bergstra & Bengio, 2012) require multiple training runs to indicate a good choice of hyperparameter, and are not adaptive to the non-stationarity of many RL scenarios. Model Selection methods are designed to select the best-performing configuration in a single run, and we show that data-driven model selection algorithms are able to adapt to the non-stationarity of the objective hyperparameter throughout the training.

In this paper, we show that model selection methods can be simply integrated into RL algorithms without heavily changing the original implementation. We evaluate six model selection strategies for this task including Regret Balancing methods (Dann et al., 2024; Pacchiano et al., 2020a), Corraling (Agarwal et al., 2017), and standard bandit strategies like UCB and EXP3 (Auer et al., 2002; Bubeck et al., 2012). Our results suggest that data-driven regret balancing methods achieve the best performance for the task of learning rate selection in RL. Additionally, we show that when the optimal choice of learning rate is time-dependent and varies with the state of learning, bandit strategies like UCB and EXP3 are not sufficient for the task of learning rate selection. [1]

2 Method

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, $\beta = \{\beta^1, \dots, \beta^m\}$ 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 $\Psi : \beta \to P(\beta)$ over base agents or represent the estimated empirical regret of the base agents.

Algorithm 1: Model Selection Interface for Hyperparameter Tuning

Input: m, β, Ψ, M

Function `sample()`:
- `// Select base index according to` Ψ
- $i \sim \Psi$
- $\pi^i, \alpha^i \leftarrow \beta^i$
- **return** i, π^i, α^i

Function `update(`$index$, $R[1:T]$`)`:
- `// Calculate and normalize the episodic return`
- $R_{norm} \leftarrow normalize(R[1:T])$
- `// Update base statistics according to the meta learning algorithm` M
- $\Psi \leftarrow M(\Psi, index, R_{norm})$

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 $(r_1, \dots, r_T)$ 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. 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

[1] Codes are available at `https://github.com/AidaAfshar/Learning-Rate-Free-Reinforcement-Learning`.

the end of it. The protocol to integrate the model selection interface with the reinforcement learning loop is represented in Algorithm 2 of Appendix A.

3 Experiments and Results

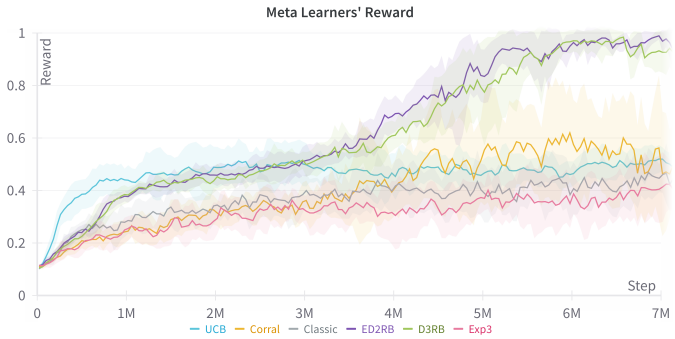
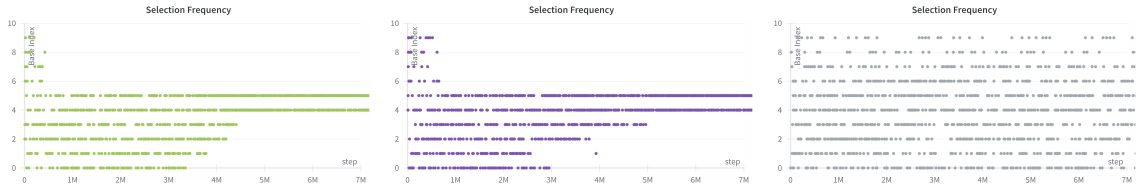
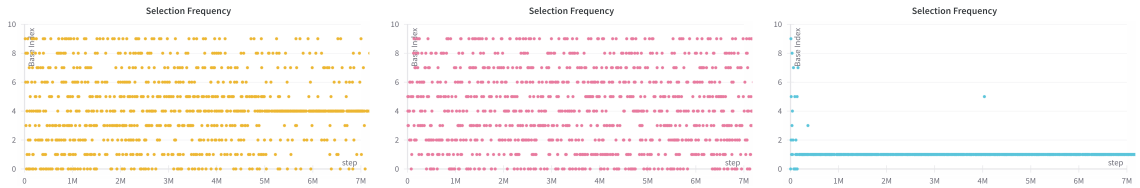


Figure 1: Learning Rate-Free PPO on Humanoid Environment. Each curve shows the mean and standard deviation of normalized reward per step over three seeds.

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 six model selection strategies introduced in Section 2. Figure 1 represents the reward plot of six meta learners on the Mujoco Humanoid environment (Tassa et al., 2012). By comparing the meta learners, we can see that D^{3}RB and ED^{2}RB strategies achieved the lowest regret and had the most advancement in the reward curve. The meta learners with Corral and Classic Balancing strategies, in addition to MAB meta learners, are showing sub-optimal performance in this task.



(a) D^{3}RB (b) ED^{2}RB (c) Classic Balancing



(d) Corral (e) EXP3 (f) UCB

Figure 2: Selection frequency of each base learner for Learning Rate-Free PPO on Humanoid environment. The y-axis indicates the base learner's index, and the x-axis indicates the timestep. Each (x,y) point shows that base learner y was selected and played by meta learner at time x.

To better understand the Model Selection strategies, we plotted the selection frequency of each base learner for all of the model selection strategies. Figure 2 represents our main findings. Each (x, y) point in the plot shows that at time x meta learner has selected base learner y. We can see that D^{3}RB and ED^{2}RB meta learners have learned to detect the best-performing base learners and played them more often, resulting in more advancement in the reward curve. Figure 2 (c, d, e) show that

meta learners with Classic Balancing, Corral, and EXP3 Strategies are selecting all the base learners quite often and therefore have an average performance in the task. Figure 2 (d) also explains the slightly better performance of Corral, as this meta learner has learned to detect one of the optimal base learners at the end.

One major drawback of applying standard MAB algorithms such as UCB in non-stationary domains like RL can be seen in these plots. A specific choice of learning rate might achieve high rewards in the early stages of learning and not perform well later on. As seen in Figure 2 (f) the base learner with index 1 was selected by UCB meta learner, since it was the best choice of learning rate in the early stages of learning. Figure 1 shows that this learning rate was performing well in the early stages of learning. Later on, this base learner was no longer an optimal learning rate but UCB couldn't adapt and continued to select that base learner. Algorithms like UCB are not able to distinguish these non-stationary changes and therefore are not suitable strategies for learning rate selection in reinforcement learning. The same challenge casts to common hyperparameter tuning techniques such as Bayesian Optimization, where the objective function is assumed to be stationary.

Additionally, we initiate ten independent PPO agents with the same set of learning rates that we input to the model selection counterparts and run each agent for $\sim \frac{1}{10}$ fraction of total episodes in model selection experiments. Figure 3 demonstrates the results of this comparison. Figure 3(a) shows the number of episodes that the meta learner with D^3RB strategy has selected each base agent throughout the training. Figure 3(b) shows the maximum episodic return achieved by PPO agents initiated with the same learning rates. We can see that D^3RB strategy for learning rate-free PPO has learned to select the agents with higher reward (and lower regret) more frequently, and dedicate less sample and compute to suboptimal bases. In fact, through regret balancing a linearly suboptimal base will not be selected for more than $\sqrt{N}$ rounds, where N is the total number of episodes. Check Appendix A for more theoretical details.

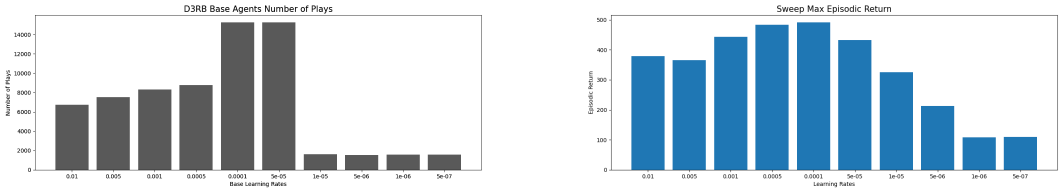


Figure 3: **(Left)** Number of times D^3RB has played each learning rate. **(Right)** Maximum reward of PPO agents initiated with the same set of learning rates. We can see that D^3RB is playing base learners with higher rewards more frequently than base learners with suboptimal performance.

4 Conclusion and Future Work

We proposed a model selection framework for learning rate-free reinforcement learning and demonstrated its effectiveness using six model selection strategies. Our experiments showed that the data-driven regret balancing method, D^3RB, and ED^2RB generally serve as good model selection strategies 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. The span of hyperparameter optimization with model selection techniques is not limited to the learning rate. Applying model selection methods for tuning a set of different hyperparameters is an interesting direction that requires sample-efficient algorithms that can be deployed in RL. Studying the effect of sharing data across the base agents is another interesting direction that can further improve the efficiency and generalizability of the framework.

References

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

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

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

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

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.

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

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.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *International Conference on Machine Learning*, pp. 9104–9149. PMLR, 2023.

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. URL http://jmlr.org/papers/v23/21-1342.html.

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

Ron Kohavi and George H John. Automatic parameter selection by minimizing estimated error. In *Machine Learning Proceedings 1995*, pp. 304–312. Elsevier, 1995.

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

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.

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

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.

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*, 2020a.

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, 2020b.

Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in neural information processing systems*, 33:17200–17211, 2020.

Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, pp. 8276–8285. PMLR, 2020.

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

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

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, 2012. doi: 10.1109/IROS.2012.6386025.

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

A1. Theoretical Remarks on Regret Balancing

Regret balancing strategies strive to maintain the regret of the different algorithms. Typically it is assumed the optimal algorithm's regret scales as $d_\star\sqrt{t}$. 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 1 and Algorithm 2. A regret balancing strategy ensures that at time N the number of times Algorithm 1 and Algorithm 2 were played, N_1, and N_2 satisfy $d_\star\sqrt{N_1} \approx \Delta N_2$ thus implying that $N_2 \approx \frac{d_\star\sqrt{N_1}}{\Delta} = \mathcal{O}\left(\frac{d_\star\sqrt{N}}{\Delta}\right)$.

A 2.1 Preliminaries of 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. Base i_n's internal state is then updated with the data collected from its interaction with the environment.

We review a few definitions from bandits to better explain these methods. The *regret* of a policy π is defined as

$$reg(\pi) = v^\star - v^\pi, \tag{1}$$

where $v^\star$ is the value of the optimal policy $\pi^\star \in \arg\max_\pi v^\pi$. In stochastic settings, it's most common to assume that the total regret of base i after being played for k rounds is bounded by $\sqrt{k}$ rate,

$$\sum_{l=1}^{k} reg(\pi^i_{(l)}) \leq d^i_{(k)}\sqrt{k}, \tag{2}$$

where $d^i_{(k)}$ is called the *regret coefficient* of base i. The subscripts (k) denote the number of times that a base has been played up to this round.

Regret balancing methods aim 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^3RB), and Estimating Data-Driven Regret Balancing (ED^2RB) (Dann et al., 2024). D^3RB maintains an estimate of the regret coefficient of base i and performs a miss-specification test to see whether this estimate is compatible with data that has been collected so far. When base i is selected for the k'th time, if the estimate is too small to correctly represent the regret of base i, it doubles the estimated regret coefficient $d^i_{(k)}$. On the other hand, ED^2RB directly estimates $d^i_{(k)}$ as the maximum difference of average reward between base i and other bases, scales by $\sqrt{k}$. Both methods, update the regret bound for the selected base and then choose the base with the lowest regret for the next round. These methods are designed to work with the *realized* regret as opposed to the *expected* regret which is used in other model selection methods like Regret Bound Balancing (Pacchiano et al., 2020a), and Corral (Agarwal et al., 2017; Pacchiano et al., 2020b).

The Regret Bound Balancing algorithm which we will refer to as "Classic Balancing" takes as input *fixed* regret coefficients that it uses to perform a miss-specification test. When a base learner fails this test the Classic Balancing algorithm eliminates it. This makes it crucial to strike an appropriate balance when deciding what putative regret coefficients are used to initialize the meta learner. When these input coefficients underestimate the real ones, Classic Balancing may eliminate the optimal base learner. If instead, the input coefficients overestimate the true regret coefficients, Classic Balancing may over-exploit sub-optimal base learners. Corral (Agarwal et al., 2017) is a model selection meta-learning algorithm based on a carefully designed adversarial bandit strategy that uses log barrier regularization. Unfortunately, the regret guarantees the Corral algorithm satisfies are only known to hold in expectation, and not in high probability. We also investigate the Upper Confidence Bound (UCB) (Auer et al., 2002), the Exponential-weight algorithm for Exploration and Exploitation (EXP3) (Bubeck et al., 2012) which has been recommended by the previous literature (Li et al., 2018) for hyperparameter tuning. The theoretical details and pseudo-code of all six algorithms are included in Appendix C.

A 2.2 Preliminaries of 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 $\pi : S \to A$. The state-value function $V : S \to \mathbb{R}$ and action-value function $Q : S \times A \to \mathbb{R}$ with respect to policy π are defined as

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) | s_0 = s, s_t, a_t\right] \tag{3}$$

$$Q^\pi(s, a) = R(s, a) + \gamma\, \mathbb{E}_{s' \sim P(s,a)}\left[V^\pi(s')\right] \tag{4}$$

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

$$\theta \leftarrow \theta + \alpha\, \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t)(Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t))\right] \tag{5}$$

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

$$\theta \leftarrow \theta + \alpha\, \mathbb{E}_{s,a,s',r\sim D}\left[\nabla_\theta (r + \gamma \max_{a' \in A} Q^{\pi_{\bar{\theta}}}(s', a') - Q^{\pi_\theta}(s, a))^2\right] \tag{6}$$

where D is the experience replay buffer and $\bar{\theta}$ is a frozen parameter set named target parameter. Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Deep Q-Networks (DQN) (Mnih et al., 2015) follow the first and second approaches, respectively.

A3. Learning Rate-Free Reinforcement Learning

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

Input: MDP $\langle S, A, R, P, \gamma \rangle$, Model Selection Interface $\mathfrak{M}$

```
// Reinforcement learning loop over episode
for n = 1, 2, ..., N do
    // Select the base agent
    i, π^i_θ, α^i = 𝔐.sample()
    // Collect trajectories with selected base agent
    for t = 1, 2, ..., T do
        a ~ π^i_θ
        r, s' ←(P,R)— s, a
        R[t] ← r
    // Update parameters with selected learning rate
    if Policy Optimization then
        θ ← θ + α^i E[Σ_{t=0}^{T} ∇_θ log π^i_θ(s_t, a_t)(Q^{π^i_θ}(s_t, a_t) − V^{π^i_θ}(s_t))]
    if Q-Learning then
        θ ← θ + α^i E_{s,a,s',r~D}[∇_θ(r + γ max_{a'∈A} Q^{π^i_θ̄}(s', a') − Q^{π^i_θ}(s, a))^2]
    // Update the meta learner
    𝔐.update(i, R[1 : T])
```

A4. Related Work

Hyperparameter tuning rises as a crucial step in machine learning problems and various works have proposed methods for hyperparameter tuning since the early days of machine learning (Kohavi & John, 1995; Bergstra et al., 2011). Random Search (Bergstra & Bengio, 2012) aimed to improve the naive grid search by considering a random subset of all possible hyperparameters, instead of the full grid. Bayesian optimization (BO)(Brochu et al., 2010; Wu et al., 2019; Ru et al., 2020) is a black-box optimization method that advanced the sample efficiency of naive approaches drastically. Despite being effective, BO requires multiple training runs to detect a good configuration of the problem, which can be deficient when combined with RL. This motivates the studying of dynamic scheduling that can be achieved in a single run. Several papers have studied learning rate scheduling, and learning rate-free learning (Defazio et al., 2023; Khaled & Jin, 2024; Mishchenko & Defazio, 2023), where they analyze the problem through the lens of convex optimization.

Deploying Bandit algorithms for hyperparameter tuning is not a new idea. Hyperband (Li et al., 2018) formulates hyperparameter optimization as an infinite-arm bandit problem that tends to adaptively dedicate resources to the better-performing configurations and stop the ones with poor performance. In a sequential domain like RL, if one configuration is performing poorly at the early stages of learning but is optimal at the later stages, hyperband will not be able to detect it. Among the available literature, the closest to our work is (Parker-Holder et al., 2020) which uses a population-based bandit method for online hyperparameter optimization, where they fit a Gaussian Process model to select the hyperparameter at each step. Replacing the latter with methods that enjoy model selection guarantees is an interesting future direction.

B. Codes

All codes and implementations of the learning rate-free RL algorithms are available at `github.com/AidaAfshar/Learning-Rate-Free-Reinforcement-Learning`. We use cleanRL library (Huang et al., 2022) for the implementation of RL algorithms. The model selection strategies were originally implemented at `https://github.com/pacchiano/modelselection`.

C. Model Selection Algorithms Pseudocodes

We provide the Model Selection Algorithms in hyperparparemeter tuning interface in this section. To avoid including all the theoretical details, there might be a slight abuse of notation in the pseudocodes. We encourage the reader to check the details with the original paper.

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) (Dann et al., 2024) 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,

Algorithm 3: D^{3}RB

Input: m, β, Ψ, δ

Function `sample()`**:**

  // Sample base index

  $i = \arg\min_j \Psi_j$

  $\pi_i, \alpha_i \leftarrow \beta_i$

  return i, π_i, α_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

  $\frac{u^i}{n^i} + \frac{d^i\sqrt{n^i}}{n^i} + c\sqrt{ln\frac{mlnn^i}{\delta}}{n^i} \leq max_j \frac{u^j}{n^j} - c\sqrt{ln\frac{Mlnn^j}{\delta}}{n^j}$

  // If test triggered double regret coefficient for base i

  $d^i \leftarrow 2d^i$

  // Update balancing potential

  $\Psi^i = d^i\sqrt{n^i}$

ED^{2}RB

Estimating Data Driven Regret Balancing (ED^{2}RB) (Dann et al., 2024) is similar to D3RB, though it tries to directly estimate the regret coefficients.

Algorithm 4: ED^{2}RB

Input: m, β, Ψ, δ

Function `sample()`**:**

  // Sample base index

  $i = \arg\min_j \Psi_j$

  $\pi_i, \alpha_i \leftarrow \beta_i$

  return i, π_i, α_i

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

  $R_{norm} \leftarrow normalize(R[1:T])$

  // Update Statistics

  $u^i = u^i + R_{norm}$

  $n^i = n^i + 1$

  // Estimate active regret coefficient

  $d^i = max d_{min}, \sqrt{n_t^{i_t}}(max_j \frac{u^j}{n^j} - c\sqrt{ln\frac{Mlnn^j}{\delta}}{n^j} - c\sqrt{ln\frac{mlnn^i}{\delta}}{n^i} - \frac{u^i}{n^i})$

  // Update balancing potential

  $\Psi^i = clip(d^i\sqrt{n^i}, \Psi^i, 2\Psi^i)$

Classic Balancing

The Classic Regret Balancing Algorithm (Pacchiano et al., 2020a) 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 5: Classic Balancing

Input: m, β, Ψ, δ

Function `sample()`**:**

  // Sample Base index

  $i = \arg\min_j \Psi_j$

  $\pi_i, \alpha_i \leftarrow \beta_i$

  return i, π_i, α_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**

    $\frac{u^k}{n^k} + \frac{d^k\sqrt{n^k}}{n^i} + c\sqrt{ln\frac{mlnn^k}{\delta}}{n^k} \leq max_j \frac{u^j}{n^j} - c\sqrt{ln\frac{Mlnn^j}{\delta}}{n^j}$

    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 6: EXP3

Input: m, β, Ψ, δ

Function `sample()`**:**

```
// Sample Base index
```

$i = \arg\max_j \Psi_j$

$\pi_i, \alpha_i \leftarrow \beta_i$

return i, π_i, α_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 (Agarwal et al., 2017) 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 7: Corral

Input: m, β, Ψ

Function `sample()`**:**

```
// Sample base index
```

$i \sim \Psi$

$\pi_i, \alpha_i \leftarrow \beta_i$

return i, π_i, α_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 (Lattimore & Szepesvári, 2020). Denote Ψ^i as the upper confidence bound of arm i. The UCB algorithm for learning rate-free RL works as follows,

Algorithm 8: UCB

Input: m, β, Ψ, δ

Function `sample()`**:**

```
// Sample base index
```

$i = \arg\max_j \Psi_j$

$\pi_i, \alpha_i \leftarrow \beta_i$

return i, π_i, α_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
```

$\Psi^i = UCB^i(\delta) = \mu^i + \sqrt{\frac{2log(1/\delta)}{n^i}}$