Biasing Exploration towards Positive Error for Efficient Reinforcement Learning

Adam Parker
Whiting School of Engineering
Johns Hopkins University
Baltimore, MD 21218, USA

John W. Sheppard
Gianforte School of Computing
Montana State University
Bozeman, MT 59717, USA

Abstract

Efficient exploration remains a critical challenge in Reinforcement Learning (RL), significantly affecting sample efficiency. This paper demonstrates that biasing exploration towards state-action pairs with positive temporal difference error speeds up convergence and, in some challenging environments, has the potential to result in an improved policy. We show that this Positive Error Bias (PEB) method achieves statistically significant performance improvements across various tasks and estimators. Empirical results demonstrate PEB's effectiveness in bandits, grid worlds, and classic control tasks with exact and approximate estimators. PEB is particularly effective when unbiased exploration struggles with policy discovery.

Introduction

Reinforcement Learning (RL) enables an agent to learn behaviors by interacting with their environment, with applications ranging from game playing (Badia et al. 2020; Van Hasselt, Guez, and Silver 2016; Mnih et al. 2013) and robotics (Kober, Bagnell, and Peters 2013) to healthcare (Yu et al. 2021). In RL, an environment provides an agent with a state, and the agent chooses an action based on this state. The environment rewards this action (which may be delayed), and the cycle continues. It is the agent's job to maximize its accumulated reward, not only in the present state but also in all future states (Sutton 2018). Exploration of these states is a paramount concern, as spending extra time on unproductive search undesirably slows learning. A standard method for exploration presently is ϵ-greedy which, with some probability ϵ, takes an entirely random action. While this makes total exploration more likely, it is far from efficient. Using this paradigm, exploring distant states with specific preconditions is challenging, if not impossible, under reasonable time constraints.

We hypothesize that we can increase exploration's effectiveness in RL by biasing exploration towards state-action pairs where recent outcomes have exceeded expectations. Actions resulting in better-than-expected outcomes might bear repeating. This idea is the intuition behind what we refer to as Positive Error Bias (PEB). PEB increases exploration effectiveness by estimating a state-action's recent possible overperformance and giving these actions a higher probability of being selected during exploration.

In many of our tests, PEB shows a statistically significant increase in early learning performance (higher total episode reward). This result suggests agents approach reasonable policies quicker, requiring less training to achieve satisfactory performance in their environments. In particularly hard-to-explore environments, we demonstrate that PEB finds a policy where ϵ-greedy exploration struggles or fails completely. We also show that it is able to learn hard-to-find optimal policies better than easier sub-optimal policies.

Background

Traditionally, reinforcement learning is posed based upon a Markov decision process such that $\mathcal{MDP} = \langle \mathbf{S}, \mathbf{A}, R, \mathbf{T}, \gamma \rangle$ where $S_t \in \mathbf{S}$ is the state at a given timestep t, $A_t \in \mathbf{A}$ is the action taken at that time, and R is the reward from the environment for taking that action. $\mathbf{T} = P(S'|S, A)$ is the transition distribution of transitioning from state S to state S') when action A is taken, and $\gamma \in [0, 1]$ is a discount factor. These transitions chain forward from an initial state to some terminating condition. We refer to these terminated chains as episodes under which an agent attempts to maximize their reward. Note there may be no terminal state, whereby the chain will continue forever.

Watkins introduced the Q-learning method in 1989 as a means for the agent to estimate an agent's expected long-run reward (denoted $Q(S_t, A_t)$) when taking action A_t in state S_t and performing according to policy π thereafter (Watkins 1989). Q-Learning is a model-free method based upon the concept of temporal difference learning (TD-Learning) that Sutton proposed (Sutton 1988). With TD-Learning, an error (TD-Error) is calculated based on the difference between the current and previous estimates. If the TD-Error is positive, this is a signal that the recent experience with a State-Action pair overperformed relative to the present estimate. Given that TD updates are iterative, we claim that it likely benefits the algorithm to revisit this pair to clarify its estimate further.

Such updates are accomplished as follows:

$$\begin{aligned} q &\leftarrow R(S_t, A_t) + \gamma \max_{a \in \mathbf{A}} Q(S_{t+1}, a) \\ err_{td} &\leftarrow q - Q(S_t, A_t) \end{aligned}$$

$$Q(S_t, A_t) \quad \leftarrow \quad Q(S_t, A_t) + \alpha(err_{td}).$$

This estimation can be kept in an exact way (e.g., a table containing a Q value for each state-action pair) or an approximate method (e.g., Neural Network, Online Random Forest, etc.). In exact methods, each state-action pair is updated individually. In contrast, approximate methods can generalize across similar states and actions, enabling scalability in large or continuous environments. In this paper, we use both as each has its own unique concerns and approaches. We show that PEB is effective for both situations.

Just performing estimates and following best values does not guarantee convergence to optimal performance. If an agent hones in on an action with a small reward and does not explore other actions, it may miss a larger reward in another action. This concept underscores the exploration-exploitation tradeoff. An agent must explore all possible actions to find the optimal policy. However, endlessly exploring does not allow the agent to utilize what it knows to achieve optimal rewards. As mentioned, ϵ-greedy is conceptually simple and, on an infinite timescale, will explore all states infinitely but may explore inefficiently in large or sparse environments. Furthermore, it does not use knowledge from previous experience to guide further exploration.

This paper provides evidence for the utility of PEB, focusing on its integration with ϵ-greedy in the context of model-free methods. However, this exploration is far from exhaustive, and the principle of biasing exploration using TD-error likely extends beyond these specific context. RL advancements are rarely independent, as evidence by the Rainbow framework (Hessel et al. 2017). PEB's design suggests potential compatibility with other techniques, to further improve performance.

Related Work

One of the simplest exploration methods in RL is ϵ-greedy. Exploration in this scheme is unbiased, choosing a random action with probability ϵ. This randomness requires epsilon to be annealed or adapted to achieve reasonable performance (Tokic 2010). Unbiased exploration can struggle in environments with complex dynamics requiring long sequences of specific actions. Randomly drawing these sequences is unlikely and can slow learning. PEB builds on this by providing a bias for exploration, increasing the likelihood of exploring promising actions. Overly narrow exploration bias may lead to suboptimal policies by neglecting less-explored actions. Sufficient exploration is necessary to ensure an optimal policy, and care should be taken when biasing exploration in any way. However, introducing a well-calibrated bias can often improve learning efficiency.

A direct improvement is often biased exploration in some way, and this usually takes the form of biasing towards actions with a high Q value. Such is the case with the Upper Confidence Bound, which adds a confidence bound to an estimate of Q and takes the highest upper bound. As states are visited more often, the Q value approaches its actual value, and the bounds shrink. This action allows the agent to refine lesser-used actions continually, exploring their actual value (Auer 2000; 2002). Softmax and Boltzmann exploration prioritize an action based on its Q value relative to its peers. They convert the Q values into a probability distribution and draw from it, utilizing the distribution to control exploration (Cesa-Bianchi and Lugosi 2006; Cesa-Bianchi et al. 2017). The key difference between PEB and these methods is that the bias for exploration comes from TD error, which gets updated immediately, unlike Q-based methods, which must wait for one Q value to overtake another to become prioritized.

PEB is not the first method to attempt to bias the learning process or even exploration. In Prioritized Experience Replay (PER), transitions are stored in a buffer and drawn based on the Q-value prediction error they cause. This bias has been shown to improve sample efficiency (Schaul 2015; Liu et al. 2022; Pan et al. 2022). Still, unlike PEB, it does not bias the exploration process explicitly, which means high error actions may take longer to find. Curiosity-driven exploration, on the other hand, does bias the exploration process but only on model prediction errors (Pathak et al. 2017; Burda et al. 2018). While this approach does not depend on external rewards like PEB does, it requires maintaining a model of the environment, the added computational overhead of which may not be necessary to learn a policy. Furthermore, QXplore (Simmons-Edler et al. 2021) and TDU (Flennerhag et al. 2021), attempt to bias exploration using reward predictions, similar ot PEB. However, both rely on additional computations to explicitly model uncertainty, such as estimating variance or maintaining multiple value predictions. Our results suggest that such complexity is unnecessary for many applications.

Algorithm

PEB introduces a new estimator, $Z(S, A)$ (colloquially Zest), which estimates the recent TD error of a given state-action pair. We initially set $Z(S, A)$ to zero for all state-action pairs for exact methods; however, they may be initialized as typical (e.g., small initial weights in NN) for approximate methods. We update $Z(S, A)$ each time we visit a state-action pair. One approach may perform a TD-style update on $Z(S, A)$, but delaying a response to a change in TD error may not be as beneficial to early learning. Delayed updates reduce the method's ability to respond to recently overperforming actions, which can slow discovery.

We continue with the notion that with some probability ϵ, an agent should explore. However, unlike traditional ϵ-greedy, the exploration action is not chosen randomly. Instead, we pass the present Z estimate into a softmax function, generating a probability distribution that biases selection towards actions with recent positive TD errors. This approach is still stochastic, ensuring total exploration while biasing the exploration towards actions with recent positive TD error. The softmax distribution is given by:

$$softmax_\tau(Z(S, A)) = \frac{e^{Z(S,A)/\tau}}{\sum_a e^{Z(S,a)/\tau}}$$

Furthermore, PEB's bias is naturally transient. As the agent improves its Q-value estimates, the TD error (and thus Z) for that action diminishes over time. Once this occurs,

Algorithm 1 Positive Error Bias (PEB)

Initialize Q-values $Q(S, A) \leftarrow 0$ for all S, A
Initialize Z-values $Z(S, A) \leftarrow 0$ for all S, A
Set hyperparameters ϵ, τ, α, and γ
for each episode **do**
    Initialize $S \leftarrow$ starting state
    while S is not terminal **do**
        Generate a random number $r \in [0, 1)$
        if $r < \epsilon$ **then** ▷ Exploration
            $P(A|S) \leftarrow softmax_\tau(Z(S, A))$
            Select action A according to $P(A|S)$
        else ▷ Exploitation
            Select $A \leftarrow \arg\max_a Q(S, a)$
        end if
        Execute action A, observe R and next state S'
        $\delta \leftarrow R + \gamma \max_{a'} Q(S', a') - Q(S, A)$
        $Q(S, A) \leftarrow Q(S, A) + \alpha\delta$
        $Z(S, A) \leftarrow \delta$
        $S \leftarrow S'$
    end while
end for

the bias shifts to actions with the following highest Z values, dynamically prioritizing promising areas of the state-action space. Softmax with a tuned temperature setting (τ) helps continually probe other actions for updated TD error estimates (Z values). This mechanism preserves total exploration, critical for avoiding premature convergence and identifying optimal policies.

One can adjust or anneal the τ to scale the bias from totally random to essentially deterministic. The algorithm may be sensitive to the selection of τ in a fixed budget learning environment. If an immediate sub-optimal reward has a more positive TD error than a distant reward and annealing of exploration degrades too quickly, the algorithm may not find the distant reward.

During exploitation, the agent selects the action with the highest Q-value as in standard ϵ-greedy. The interplay between ϵ and τ is critical; ϵ controls how often exploration occurs, and τ controls the strength of the bias when exploration occurs. Algorithm 1 outlines the main steps of PEB, including initialization, decision-making, and updates to the Z and Q estimators.

Experiments

We evaluate the effect of PEB through three experiments, a k-armed bandit, a simple gridworld, and three classic control problems. The k-armed bandit tests PEB's ability to make locally optimal choices during exploration, resulting in an optimal policy. We include the gridworld to stack these locally optimal choices into efficient multi-state optimization. The final experiment highlights the continued usefulness of PEB under the generalization of Z. We claim these problems represent typical RL problems and provide evidence of PEB's effectiveness in varied environments.

A k-armed bandit is a reference to slot machines. In this environment, each arm of the bandit gives a set distribution of rewards. An agent must find and exploit the arm with the highest average return to maximize its reward; thus the problem reflects the classic exploration-exploitation trade-off. Reward distributions with overlapping standard deviations may complicate this as the optimality of an arm becomes more challenging to distinguish. In the base k-armed bandit, the bandit does not have a state, and the player has no control over future rewards. This environment characterizes the ability of an algorithm to measure its environment and make locally optimal choices. We used an open source OpenAI Gymnasium (Towers et al. 2024) compatible environment for k-armed bandits, Buffalo Gym (Parker 2024), which provides several standard and non-standard k-armed bandit implementations with the aim of reproducibility in results. From this package, we use the "Bandit-v0" environment, which implements a non-contextual bandit. We use the environment settings in Table 1 and measure the average reward over the steps of the run for each of 1000 agents, providing the average and standard deviation of performance at a given step.

Setting	Value
Seed	42
Arms	10
Optimal Arms	1
Optimal Mean	10
Optimal Std	1
Min Suboptimal Mean	0
Max Suboptimal Mean	5
Suboptimal Std	1
Dynamic Rate	None
Number of agents	1000

Table 1: Environment settings for k-armed bandit of Buffalo-Gym.

The gridworld is a simple maze-like environment. The agent knows where it is in the gridworld and can move in one of the cardinal directions. For our experiment, we built a "corridor" gridworld of size of h, which denotes the number of tiles inside the outer perimeter of the map (Figure 1). The map has walls, and any attempt to traverse a wall puts the agent back in its previous state. Walls surround the map, and there is a wall down the center of the map, separating it lengthwise. In the center of the wall is a single-tile corridor to get to the other half of the map. The map has two terminal states, one immediately south of the fixed starting position in the top left and the other in the bottom right corner. The terminal state closer to the starting position has a reward of h and the further reward is $2 \times h$.

The agent also incurs a cost of -1 for each step taken. This reward structure incentivizes the agent to reach a goal quickly. The dual reward setup measures the agent's ability to find an optimal reward when tempted with a local optimum. The metric of success is the total reward for an episode, which ends at a terminal state. The episode can also end when it hits a maximum number of steps, which we set to $3 \times h$. Again, we calculate the mean and standard deviation of the performance using 1000 agents.

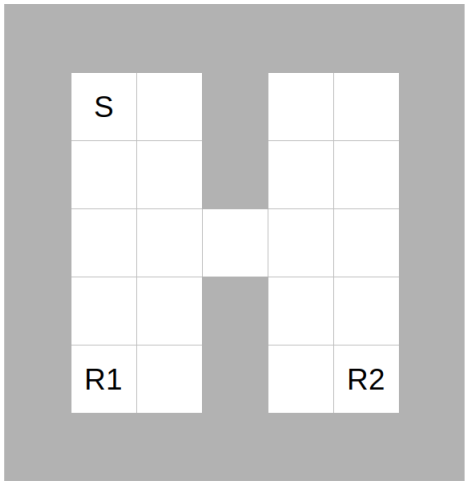

Figure 1: Gridworld where S = the start state, R1 = the sub-optimal terminal state and R2 = the optimal terminal state.

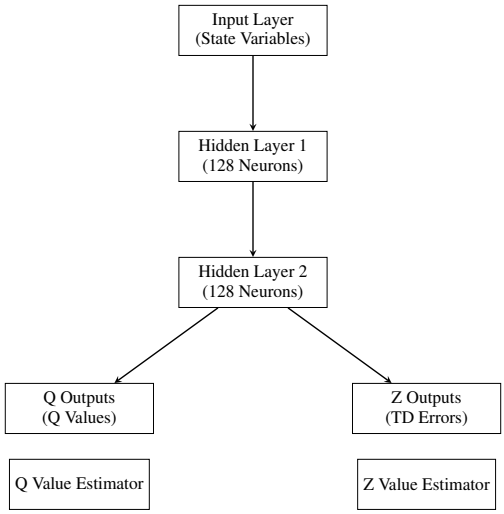

Figure 2: Neural network architecture used for approximate estimator.

We ran experiments on both the k-armed bandit and the gridworld with a tabular (exact) estimator. The estimator used a fixed learning rate of $\alpha = 0.1$ and a discount factor of $\gamma = 1$. ϵ started at 0.5, and was annealed to a final value of 0.01 over the first 75% of the episodes. We set $\tau = 10$ during the bandit experiments and 0.4 for the gridworld experiments. We chose the values through manual tuning via grid search. For the control, we use the tabular Q-Learning algorithm with $epsilon$-greedy search, which is effective in bandit environments (Kuleshov and Precup 2014). All values were kept the same between the two runs. We tested the statistical significance of the results using a two-tailed two-sample t-test with $p \leq 0.05$.

For the final experiment, we used three problems from the OpenAI Gymnasium Environment. Specifically, we tested on Acrobot, Cartpole, and Mountain Car. These problems each have real-valued states, making the state space infinite and requiring generalization to solve effectively. In Acrobot, the goal is to apply torque to the center joint of a two-segment chain and raise the free end of the chain above a certain height. Cartpole lets the agent control a cart to which an inverted pendulum is attached. The pendulum starts upright, and the agent aims to keep it upright by moving the cart to the left and right. Finally, Mountain Car requires the agent to accelerate the car strategically from its position at the bottom of a hill to reach the goal on top. All problems used discrete actions: -1, 0, 1 torque in Acrobot, left/right push in Cartpole, and left/none/right acceleration in Mountain Car.

For these latter control environments, we used a neural network as the estimator. We initialized the network from scratch for each agent. The network's architecture consisted of a neuron for each element of the state value vector, two fully connected hidden layers with 128 neurons each, and two sets of outputs. The output layers consisted of one neuron for each action, representing the corresponding Q value, and one neuron for each action, representing the corresponding Z value.

Figure 2 shows the architecture of the neural network. The shared trunk with separate prediction outputs is similar to the Dueling Network architecture, where the outputs predict Value and Advantage (Wang et al. 2016). A RELU activation was used at all layers except the outputs, which were linear. We use the AdamW optimizer with a learning rate of 0.001 and 0.005 for Q and Z, respectively. The weight updates from the Q node flows back to the inputs, and the weight updates for Z stops where it branches from the trunk of the network. In our testing, Z's lack of effect on the main trunk helped stabilize the learning of both Q and Z.

During weight updates, we drew random batches of 32 transitions from an experience replay buffer of size 10,000. Epsilon started at 0.9 and was annealed geometrically with an annealing rate of $\zeta = 0.99$ in all environments except Acrobot, where annealing occurred at a rate of $\zeta = 0.9$. Annealing occurred once per episode as $\epsilon \leftarrow \epsilon \times \zeta$. We set $\tau = 0.4$ after tuning.

The control in these experiments is regular ϵ-greedy with a Neural Network estimator for Q (herein called DQN). The agent's performance was measured as the total reward from an episode, averaged over 20 agents for both PEB and the control. The architecture of the PEB neural net estimator and DQN were identical, except the control did not have a Z branch.

Results

The k-armed bandit

Performance on the k-armed bandit is shown in Figure 3. As shown, performance separates quickly between PEB and Q-Learning. Significant differences in performance are revealed as early as the 7th step (out of 1000) ($p = 0.0002$). PEB results in a 10% higher average reward (7.2 vs 6.5) at the end of the trial ($p < 0.0001$). In this case, PEB's focus

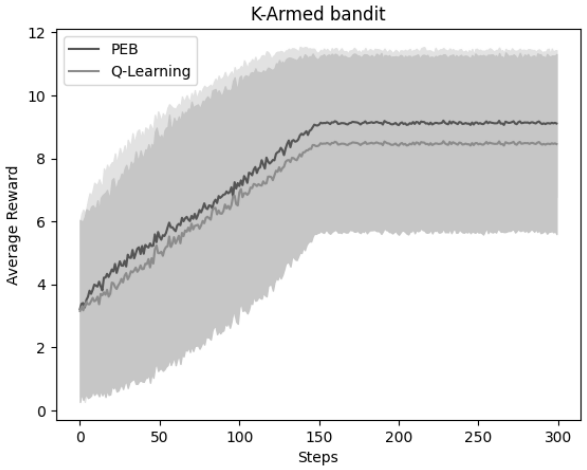

Figure 3: Learning curve for k-armed bandit with $k = 10$. Shaded regions indicate 1 standard deviation off the mean.

on arms with recent positive TD errors (and soft bias toward the highest positive TD errors) accumulates rewards early. In contrast, Q-Learning spreads experience evenly, leading to slower reward accumulation. However, despite this focused exploration, PEB's exploration does not appear to get stuck on suboptimal arms, as evident in the final performance.

The gridworld

We ran the gridworld at two settings, $h = 7$ and $h = 9$ (Figure 4). Both graphs clearly distinguish between the algorithms in terms of terminal performance. In both experiments, PEB achieves the average terminal performance of > 3.8 (with 4 being optimal). ϵ-greedy has an average negative total reward with PEB's lead deepening as we increase h. Raising h from 7 to 9 dramatically increased the exploration difficulty, as the agent needed to navigate longer sequences of actions to reach the optimal goal. In our experimentation, the dip in early performance becomes shallower as we lower τ. PEB explores the most promising regions first. As the Z signal diminishes, softmax leads the agent to explore the next most interesting paths, which give worse rewards at first, but total exploration helps the algorithm find the optimal path.

Control tasks

Finally, we show the results for the classic control tasks in Figure 5. Mountain Car represents a challenging environment for ϵ-greedy. Given a 400-episode training budget, DQN did little to no learning. In contrast, PEB began learning at 200 episodes and maintained steady learning throughout the remaining episodes. PEB biases the exploration towards building momentum and increasing overall reward, where ϵ-greedy tended to wander.

CartPole is a medium-difficulty task, which we evaluated with a 200-episode budget. DQN began to improve in under 100 episodes on average; however, PEB began to distinguish itself as early as 25 episodes ($p < 0.05$). Furthermore,

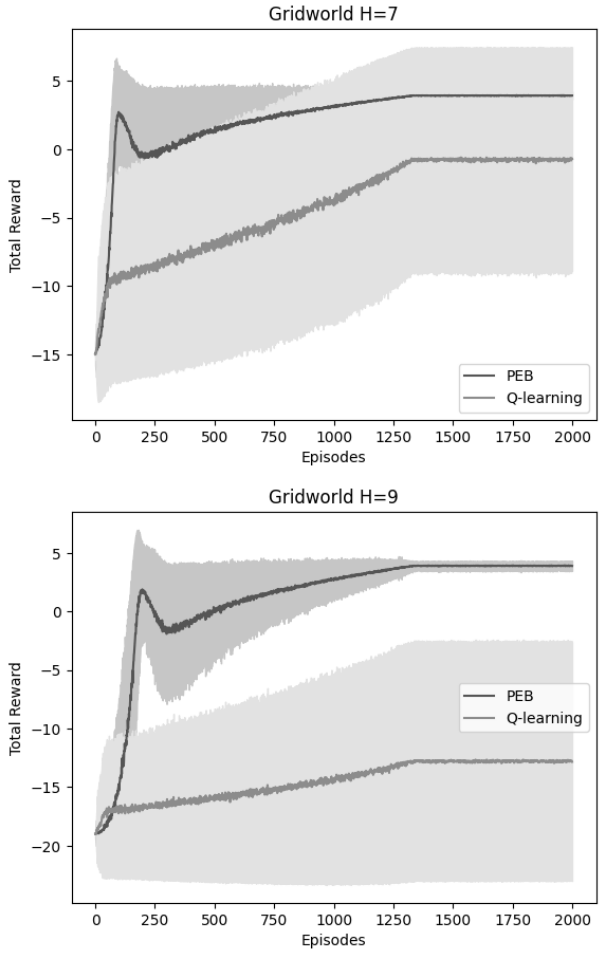

Figure 4: Learning curves for gridworld with $h = 7$ and $h = 9$. Shaded regions indicate 1 standard deviation off the mean. The optimal reward is scaled to 4 and the suboptimal reward is scaled to 2.

its lead in performance continued throughout the reminder of the 200-episode budget. Agents within a single standard deviation of the mean achieved a near-perfect score of 500 during this budget, which means approximately one third of agents had at least one perfect or near-perfect episode during this period. These results suggest that PEB excels in environments where approximate solutions allow Z values to generalize effectively across states. This generalization enables the agent to prioritize promising regions of the state-action space, even those it has not recently visited, ensuring exploration remains focused and efficient.

Finally, PEB essentially tied with DQN in the Acrobot task. Neither algorithm had a statistically significant lead over the other at any part of the task. It could be that the bias was unneeded as ϵ-greedy found the task to be relatively easy; any random action was acceptable during exploration.

The range of results presented demonstrates that PEB provides a bias that can significantly help in some environments. In almost all cases, early learning performance sub-

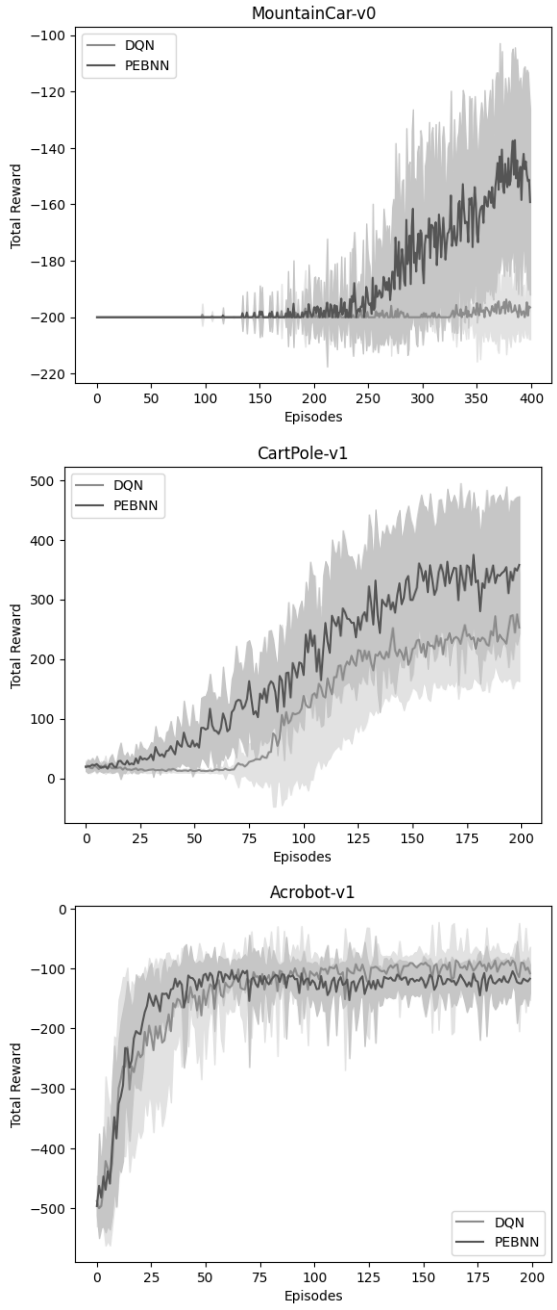

Figure 5: Learning curves for the three classic control tasks. Shaded regions indicate 1 standard deviation off the mean.

stantially improved. It seems particularly effective when exploration was crucial, but it appeared to struggle as complexity increased.

Discussion

This paper demonstrates the utility of Positive Error Bias (PEB) as a practical exploration strategy in Reinforcement Learning (RL). PEB's advantage appears to lie in its targeted exploration of actions with the highest positive TD errors. This focus appears to allow agents to prioritize actions with immediate promise, contributing to faster early learning. With proper tuning, PEB can lead to shorter, more consistent training cycles and reduced total compute requirements, as fewer episodes or iterations may be needed to achieve meaningful performance. This effect sometimes translates into superior terminal policies, as observed in the $h = 7$ and $h = 9$ gridworld and Mountain Car.

There are several hyperparameter interactions to remember when using PEB. An excessively low τ may get exploration trapped by rewards with high variance. Despite the early performance, a low τ may require a longer ϵ-anneal to provide a robust policy as it deprioritizes total exploration. Too high of a τ and a low γ will weaken the effect of distant rewards on Z, and exploration may default to near-random, limiting PEB's impact.

Including a Z estimator introduces additional computational overhead, doubling the storage requirements for exact solutions. This overhead is mitigated in approximate implementations by sharing lower network layers between Q and Z estimators; however, it still necessitates updating a marginally larger set of parameters. Therefore, one needs to weigh the tradeoff between PEB's benefits and these resource demands. Notably, the Z branch would be omitted during runtime if continual learning is unnecessary, as its primary function is to guide exploration.

In light of these increased storage and computational costs, it is tempting to compare this algorithm to softmax exploration, under which the softmax of the Q function selects actions. We explored integrating PEB with softmax exploration by combining the softmax of Q and Z values into a weighted probability distribution. This method showed promise, but the added complexity introduced confounding factors that require further investigation beyond the scope of this study. Furthermore, PEB is only a bias, and its bias may be compatible with the biases of other modifications to DQN (e.g., Dueling Networks, PER, Noisy Networks). However, the interaction of these biases requires more study.

This study is limited in scope, focusing only on classic control tasks. Future work should investigate PEB's effects in more diverse and challenging settings, such as multi-agent RL, dynamic reward environments, and vision-based tasks. Finally, PEB's mechanisms may go beyond policy optimization to planning-level control, potentially unlocking benefits at a meta-control level.

In summary, PEB shows potential as an exploration bias. Enhancing RL efficiency through improved exploration offers meaningful advantages to researchers and practitioners alike. Further studies could make evident greater potential, cementing PEB's role as a valuable tool in the RL toolkit.

References

Auer, P. 2000. Using Upper Confidence Bounds for Online Learning. In *Proceedings 41st Annual IEEE Symposium on Foundations of Computer Science*, 270–279.

Auer, P. 2002. *Finite-time Analysis of the Multiarmed Bandit Problem*. Kluwer Academic Publishers.

Badia, A. P.; Piot, B.; Kapturowski, S.; Sprechmann, P.;

Vitvitskyi, A.; Guo, Z. D.; and Blundell, C. 2020. Agent57: Outperforming the Atari Human Benchmark. In *International Conference on Machine Learning*, 507–517. PMLR.

Burda, Y.; Edwards, H.; Pathak, D.; Storkey, A.; Darrell, T.; and Efros, A. A. 2018. Large-scale Study of Curiosity-driven Learning. arXiv:1808.04355; https://arxiv.org/abs/1808.04355.

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, Learning, and Games*. Cambridge university press.

Cesa-Bianchi, N.; Gentile, C.; Lugosi, G.; and Neu, G. 2017. Boltzmann Exploration Done Right. In *Advances in neural information processing systems*, volume 30.

Flennerhag, S.; Wang, J. X.; Sprechmann, P.; Visin, F.; Galashov, A.; Kapturowski, S.; Borsa, D. L.; Heess, N.; Barreto, A.; and Pascanu, R. 2021. Temporal difference uncertainties as a signal for exploration.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2017. Rainbow: Combining improvements in deep reinforcement learning.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement Learning in Robotics: A Survey. *The International Journal of Robotics Research* 32(11):1238–1274.

Kuleshov, V., and Precup, D. 2014. Algorithms for Multi-armed Bandit Problems. arXiv:1402.6028; https://arxiv.org/abs/1402.6028.

Liu, X.; Zhu, T.; Jiang, C.; Ye, D.; and Zhao, F. 2022. Prioritized Experience Replay Based on Multi-armed Bandit. *Expert Systems with Applications* 189:116023.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602; https://arxiv.org/abs/1312.5602.

Pan, Y.; Mei, J.; Farahmand, A.-m.; White, M.; Yao, H.; Rohani, M.; and Luo, J. 2022. Understanding and Mitigating the Limitations of Prioritized Experience Replay. In *Uncertainty in Artificial Intelligence*, 1561–1571. PMLR.

Parker, A. 2024. Buffalo Gym. version 0.2.0, https://github.com/foreverska/buffalo-gym.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *International Conference on Machine Learning*, 2778–2787. PMLR.

Schaul, T. 2015. Prioritized Experience Replay. arXiv:1511.05952; https://arxiv.org/abs/1511.05952.

Simmons-Edler, R.; Eisner, B.; Yang, D.; Bisulco, A.; Mitchell, E.; Seung, S.; and Lee, D. 2021. Reward prediction error as an exploration objective in deep rl.

Sutton, R. S. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine learning* 3:9–44.

Sutton, R. S. 2018. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

Tokic, M. 2010. Adaptive ε-greedy Exploration in Reinforcement Learning Based on Value Differences. In *KI 2010: Advances in Artificial Intelligence*, 203–210. Springer.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; Cola, G. D.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; Perez-Vicente, R.; Pierré, A.; Schulhoff, S.; Tai, J. J.; Tan, H.; and Younis, O. G. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv:2407.17032; https://arxiv.org/abs/2407.17032.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *International Conference on Machine Learning*, 1995–2003. PMLR.

Watkins, C. J. C. H. 1989. *Learning From Delayed Rewards*. Ph.D. Dissertation, King's College, Cambridge United Kingdom.

Yu, C.; Liu, J.; Nemati, S.; and Yin, G. 2021. Reinforcement Learning in Healthcare: A Survey. *ACM Computing Surveys (CSUR)* 55(1):1–36.