Exploring strategies in the Multi-Arm bandit testbed

A. Todorov, C. Kobriger, C. Szoke, L. Sawala

1 Introduction

In this report, we examine different exploration strategies in reinforcement learning by applying them to the 10-armed bandit testbed. We analyze five algorithms: ε-greedy, greedy with optimistic initialization, Upper Confidence Bound (UCB), Softmax, and Gradient Bandit (Action Preference with baseline). Each algorithm is described in terms of its theoretical basis and implementation details.

The report begins with the formal statement of the multi-armed bandit problem. Next, an overview of the examined algorithms is presented and the experiments and their configurations are defined. Finally, we present the experimental results along with an analysis of their performance, with key insights regarding the effectiveness of each method.

2 Methods

This section describes the formalization and implementation of the environment and algorithms used in the conducted experiments.

2.1 Multi-Armed Bandits

The 10-armed bandit problem assumes the setting as described in Sutton and Barto (2018).

The multi-armed bandit problem consists of choosing between k actions, each of which provides a reward, modeled by a probability distribution. The goal is to maximize the expected total reward over some time period, for example, over 1000 action selections.

Formally, the problem can be described as a set of (unknown) distributions $\{\mathcal{R}_a \mid a \in \mathcal{A}\}$, where $\mathcal{A}$ denotes the set of possible actions $\{1, 2, \dots\}$ that can be played in the environment, resulting in $k = |\mathcal{A}|$ possible actions. At each timestep $t \in [1, T]$, where T is the total number of steps, an agent selects an action A_t and receives a reward $R_t \sim R_{A_t}$.

The objective is to maximize the cumulative received reward $\sum_{t=1}^{T} R_t$ by learning a distribution over actions π, also called a policy, which prioritizes actions with higher expected rewards. The true value of an arbitrary action $a \in \mathcal{A}$ is given by $q_*(a) = \mathbb{E}[R_t \mid A_t = a]$. Since the reward distributions might be unknown, the agent maintains an estimate of the action values, denoted by $Q_t(a)$ and updates these estimates as learning progresses.

In this setting, we define $k = 10$ arms, each following a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu \sim \mathcal{N}(0, 1)$ and $\sigma^2 = 1$. To evaluate performance, we simulate 500 independent runs, each initializing a distinct 10-armed bandit setup. Each run involves 1000 action selections.

2.2 Algorithms

This section presents the algorithms used in this study and their implementation details. For all algorithms, the initial value estimates $Q_0(a) = 0, \forall a \in \mathcal{A}$, unless stated explicitly.

2.2.1 ε-greedy

In the ε-greedy algorithm, at each timestep, the agent selects an action A_t according to the probability distribution

$$\pi_t(a) = \begin{cases} (1-\varepsilon) + \frac{\varepsilon}{|\mathcal{A}|}, & \text{if } a = \arg\max_{a \in \mathcal{A}} Q_t(a), \\ \frac{\varepsilon}{|\mathcal{A}|}, & \text{otherwise,} \end{cases}$$

where $\varepsilon \in [0, 1]$ balances the exploration and exploitation of the agent. With probability $1 - \varepsilon$, it selects the action with the highest estimated Q-value (exploitation) and with probability ε, a random action is selected (exploration). The estimated Q values up to time step t are updated using the sample average method, namely,

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{n_t} \left[R_t - Q_t(a) \right], \qquad (1)$$

where n_t is the number of times the action a has been selected previously.

We interchangeably use the $\varepsilon = 0$ agent and the greedy agent terminology.

2.2.2 Greedy with Optimistic Initialization

In the greedy with optimistic initialization algorithm, all initial values of the estimated Q values are intentionally set to a higher (optimistic) value than the expected rewards, denoted by Q_0. This

encourages initial exploration by making all actions equally preferable in the beginning, and as learning progresses, the agent is more informed about which actions should be exploited.

At each timestep t, the agent selects an action with the highest estimated value

$$A_t = \arg\max_{a \in \mathcal{A}} Q_t(a).$$

Then, the value estimates are updated according to the weighted average method

$$Q_{t+1}(a) = Q_t(a) + \alpha(R_t - Q_t(a)),$$

where α is a step-size hyperparameter, controlling how fast incoming rewards affect the value estimates.

2.2.3 Upper Confidence Bound

The UCB algorithm adds an additional term in the value estimates to account for the uncertainty in selecting each action. Namely, at each timestep t, the agent selects the action that maximizes the sum of the estimated value and the exploration term,

$$A_t = \arg\max_{a \in \mathcal{A}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{n_t(a)}} \right],$$

where $n_t(a)$ is the number of times an action has been selected up to time t and $c > 0$ is a hyperparameter that controls the weight of the uncertainty term. If an action has not been selected for some time, its uncertainty grows, making it more viable to be selected. The exploration term is initialized to $+\infty$ if an action has never been selected.

The value estimates are updated according to the sample average method, as seen in Equation 1.

2.2.4 Softmax

The Softmax algorithm uses a probabilistic approach to select actions. Each action is selected according to a probability distribution, based on its estimated Q-value, calculated using the softmax function

$$\pi(a) = \frac{e^{Q_t(a)/\tau}}{\sum_b e^{Q_t(b)/\tau}},$$

where $\tau > 0$ is a temperature hyperparameter controlling exploration and exploitation. With high τ values, actions have uniform probabilities, encouraging exploration, and with low τ values, probabilities are concentrated on actions with high Q-values, favoring exploitation.

The Q values are updated according to the sample average method as in Equation 1.

2.2.5 Action Preferences with Baseline

In the action preference with baseline strategy, also called a gradient bandit, the agent keeps track of preferences $H_t(a)$ for each action, which are converted to a probability distribution using the plain softmax ($\tau = 1$)

$$\pi(a) = \frac{e^{H_t(a)}}{\sum_b e^{H_t(b)}}.$$

After selecting an action A_t and observing a reward R_t, the preferences are updated as

$$H_{t+1}(a) = H_t(a) + \alpha(R_t - b_t)(\mathbb{1}_{\{A_t\}}(a) - \pi(A_t)),$$

where α is a step size hyperparameter, $\mathbb{1}$ denotes the indicator function

$$\mathbb{1}_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise,} \end{cases}$$

and b_t corresponds to all the average reward up to time t, namely,

$$b_t = \frac{1}{t}\sum_{i=1}^{t} R_i.$$

The baseline reduces variance in the updates, ensuring that actions with rewards higher than the average are reinforced, while those with lower rewards are penalized. All initial preferences are set to 0.

2.3 Experiments

We conducted two different experiments, as described below. A random seed of 1 was chosen to ensure the reproducibility of the experiments and results.

2.3.1 Learning Curves

This experiment compares the performance of the ε-greedy algorithm under two settings ($\varepsilon = 0.0$ and $\varepsilon = 0.1$) by plotting the average reward over 1000 steps. The agents are set up as described in Section 2.2.1 and the 10-armed bandit problem is initialized as in Section 2.1.

Each simulation consists of 1000 steps, during which the agent selects an action at each step according to the ε-greedy policy with the two hyperparameter configurations. The simulation is repeated for 500 independent runs to average out variability.

The average reward at each timestep t is computed across all runs and the mean and standard deviation of the average reward are recorded, which serve as evaluation metrics. The mean rewards are smoothed out for visualization using the `uniform_filter1d` (smoothing window of 50) from the `scipy` library, which is equivalent to a one-dimensional convolution.

2.3.2 Hyperparameter Comparison

This experiment compares the performance of different exploration strategies across various hyperparameter settings by comparing their average rewards over the first 1000 steps in the 10-armed bandit problem, as described in Section 2.1. All of the following agents are initialized according to their respective descriptions in Section 2.2.

We examine the greedy with optimistic initialization (fixed $\alpha = 0.1$), ε-greedy, UCB, Softmax, and the gradient bandit with the hyperparameter configurations seen in Table 2.1.

Each algorithm runs for 1000 steps per trial. Each configuration is tested across 500 independent runs. The average reward of the first 1000 steps is computed for each algorithm. The mean and standard deviation of the average rewards are recorded across all runs for comparison and serve as evaluation metrics. Smoothing is applied in the same manner as in Section 2.3.1 with a smoothing window of 5.

Hyperparameter	Values
ε in ε-greedy	$1/128, 1/64, 1/32, 1/16, 1/8, 1/4$
c in UCB	$1/16, 1/4, 1/2, 1, 2, 4$
Q_0 in Optimistic Greedy	$1/4, 1/2, 1, 2, 4$
α in Gradient Bandit	$1/32, 1/16, 1/8, 1/4, 1/2, 1, 2, 4$
τ in Softmax	$1/16, 1/8, 1/4, 1/2$

Table 2.1: Hyperparameter configurations across algorithms tested in the hyperparameter comparison experiment of the study.

3 Results

This section presents the results of the two experiments conducted for 1000 steps and across 500 different runs.

3.1 Learning Curves

The learning curve for average rewards per step with ± 1 standard error (SE), averaged across 500 runs can be seen in Figure 3.1. The average reward and SE across all 1000 steps and 500 runs are presented in Table 3.1.

ε	Average Reward	Standard Error
0	1.036	0.0369
0.1	1.329	0.0407

Table 3.1: Average Reward and Standard Error at Time Step 1000 for ε-Greedy Algorithm

3.2 Hyperparameter Comparison

The average reward over the first 1000 steps, along with the standard error across the 500 runs can be seen in Figure 3.2 and Table 3.2.

Hyperparameter	Value	Mean	SE
ε in ε-greedy	1/128	1.172	0.022
	1/64	1.252	0.023
	1/32	1.325	0.023
	1/16	1.322	0.024
	1/8	1.290	0.025
	1/4	1.095	0.027
c in UCB	1/16	1.417	0.021
	1/4	1.441	0.021
	1/2	1.494	0.022
	1	1.487	0.022
	2	1.393	0.025
	4	1.133	0.028
Q_0 in Optimistic Greedy	1/4	1.317	0.022
	1/2	1.444	0.022
	1	1.411	0.023
	2	1.410	0.025
	4	1.319	0.029
α in Gradient Bandit	1/32	1.115	0.031
	1/16	1.269	0.028
	1/8	1.363	0.025
	1/4	1.437	0.023
	1/2	1.402	0.022
	1	1.270	0.022
	2	1.116	0.021
	4	0.953	0.021
τ in Softmax	1/16	1.123	0.021
	1/8	1.297	0.022
	1/4	1.380	0.022
	1/2	1.178	0.023

Table 3.2: Hyperparameter configurations with their corresponding mean and standard error.

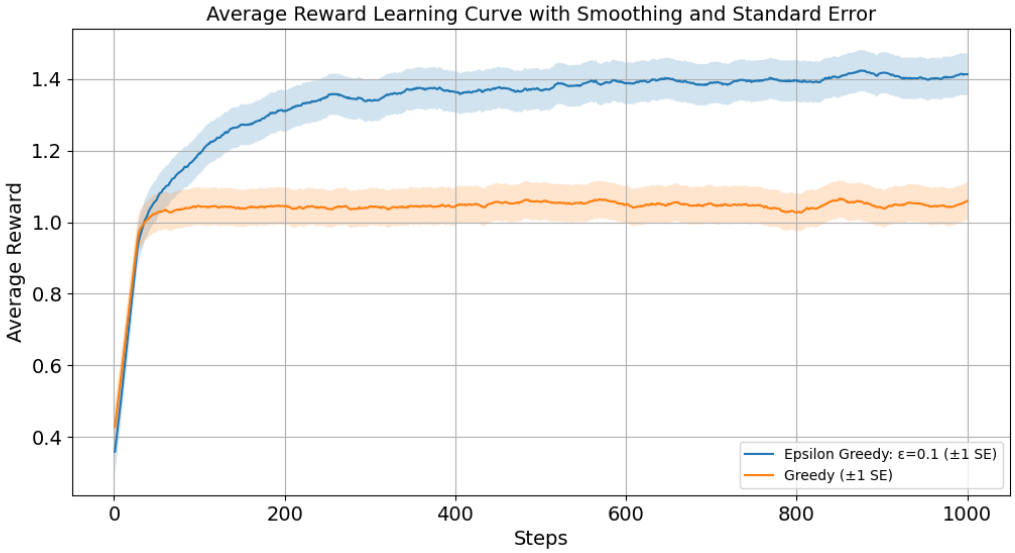

Figure 3.1: Results from the learning curves experiment, in which $\varepsilon-$greedy strategies were evaluated for $\varepsilon = 0$ (greedy) and $\varepsilon = 0.1$ over 1000 steps. The graph displays the average reward per step, which is also averaged across 500 independent runs. The shaded regions denote one standard error (± 1 SE) around the mean. The configuration with $\varepsilon = 0.1$ achieves higher average rewards with greater variance early on, stabilizing over time, while the greedy ($\varepsilon = 0$) method yields consistently lower rewards with less variability. Estimated value updates were done using the sample-average method. A one-dimensional convolution with a smoothing window of 50 has been applied to the average reward per step to ensure better visualization.

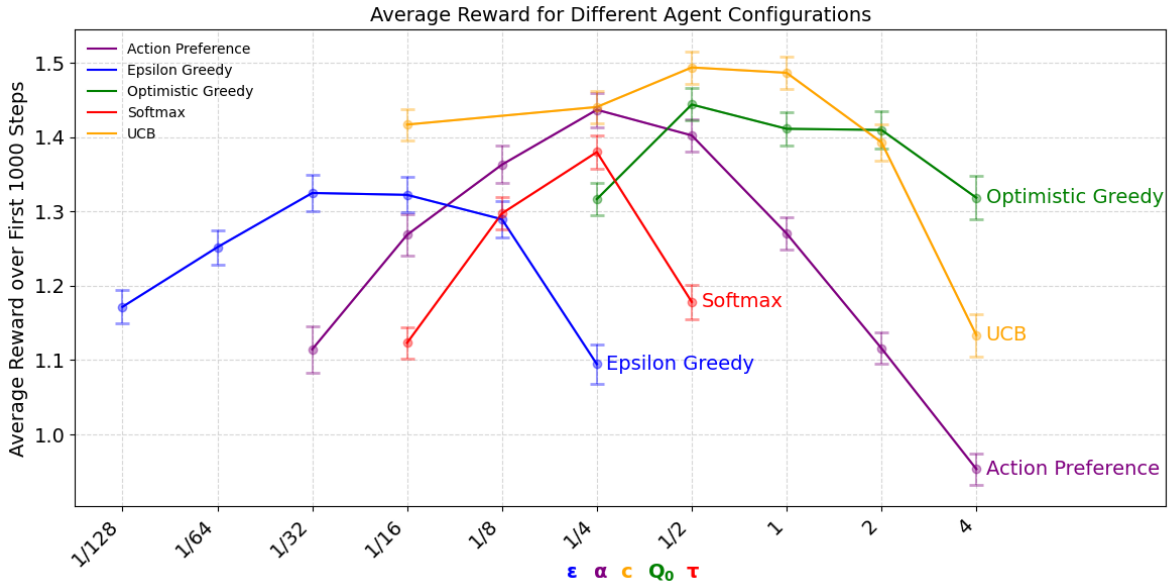

Figure 3.2: Results from the hyperparameter comparison experiments for various hyperparameter configurations. The plot shows the average reward over the first 1000 steps, averaged out over 500 independent runs. Algorithms that use Q-value updates used the sample-average method, except for optimistic greedy, which used the weight-averaged method with $\alpha = 0.1$. Error bars denote standard error across runs. The plot illustrates how varying exploration strategies and their hyperparameters affect performance, with Optimistic Greedy and UCB configurations achieving the highest rewards in this comparison.

4 Discussion

In this section, we analyze the performance of the algorithms in the two experiments in the following subsections.

4.1 Learning Curves

In the 10-armed Bandit environment, the agent with $\varepsilon = 0.1$ performs much better than the greedy ($\varepsilon = 0$) agent. This is in line with expectation as the greedy agent has no exploration whatsoever, which leads to suboptimal rewards in the long term, as observed in Figure 3.1. In contrast, the addition of a small exploration probability of $\varepsilon = 0.1$ utilizes random exploration to ensure that the arm with the highest reward can be found, and exploited more often, even though occasionally a random action is played.

4.2 Hyperparameter Comparison

In the hyperparameter comparison experiments, UCB ($c = 1/16, 1/4, 1/2, 1$) achieves the overall highest performance, as it is able to explore effectively by prioritizing actions with higher uncertainty and at the same time exploiting actions with high estimated values. Moreover, optimistic greedy also performs sufficiently well, achieving high rewards in several configurations. Its optimistic initialization drives exploration early and as learning progresses, the agent knows which arm should be continuously exploited.

For $\alpha = 1/4$, action preference has the same best overall performance as for the UCB agent with $c = 1/4$. However, this algorithm, along with Softmax, are the most sensitive to the change in parameter values, obtaining a high variance of mean rewards between hyperparameter configurations. It is also worth noting that the gradient bandit outperforms Softmax for all values. Moreover, both action preference and Softmax are data- and resource-heavy, especially compared to simpler algorithms like greedy with optimistic initialization and ε-greedy.

ε-greedy performs the worst overall, with its best configuration of $\varepsilon = 1/32$ being outperformed by several other configurations from all other algorithms. This is because its exploration mechanism is driven by a fixed probability ε, hence this exploration is uncontrolled and will always contain some noise. Lower ε values lead to insufficient exploration, while higher ones lead to too much random exploration, making the algorithm with fixed ε highly inflexible in the setting, compared to the rest.

5 Conclusion

This study examined several algorithms on the multi-armed bandit testbed: ε-greedy, greedy with optimistic initialization, UCB, Softmax, and Gradient Bandit. Through two experiments, we explored the strengths and limitations of each method by assessing their performance in terms of average reward under different conditions and configurations.

In the first experiment, we compared the learning curve of ε-greedy with $\varepsilon = 0$ and $\varepsilon = 0.1$ with the latter significantly outperforming its greedy counterpart. This highlights the importance of exploration, as ε-greedy's random exploration mechanism allows it to avoid exploiting a suboptimal action, unlike the purely exploitative greedy strategy.

In the second experiment, we compared the performance of the aforementioned algorithms for various hyperparameter configurations. Notably, UCB and greedy with optimistic initialization performed the best overall. UCB leveraged its uncertainty-based exploration mechanism to balance exploration and exploitation effectively, resulting in the highest rewards across configurations. Similarly, the optimistic greedy algorithm's initial high-value estimates promote exploration in the early stages, allowing the agent to discover and focus on high-reward actions. While Softmax and Gradient Bandit showed mixed performance due to their sensitivity to the hyperparameters, the ε-greedy algorithm performed the worst, because of its inflexibility to balance exploration and exploitation.

Overall, UCB and optimistic greedy proved to be the most suitable strategies for finite-horizon tasks like the 10-armed bandit.

Acknowledgements

We thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Hábrók high-performance computing cluster.

Conflicts

There was no conflict between the authors and all members contributed equally.

References

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.