COMPARATIVE ANALYSIS OF SOFTMAX AND UPPER CONFIDENCE BOUND IN GAME-PLAYING AGENTS FOR FLAPPY BIRD

M. Zainal Arifin, M. Baharuddin Yusuf

Universitas Negeri Malang, Jl. Semarang No.5, Sumbersari, Kota Malang, Jawa Timur 65145
*Corresponding author, email: arifin.mzainal@um.ac.id

doi: 10.17977/um068.v5.i2.2025.2

Keywords
Q Learning
Reinforcement Learning
Ucb
Softmax
Flappy Bird.

Abstract
Training agents is an intriguing research topic due to human limitations in maintaining consistent performance, particularly in the game Flappy Bird. This study compares action selection methods, namely Softmax and Upper Confidence Bound (UCB), to enhance agent performance in action selection. Testing was conducted using both methods in a Flappy Bird environment based on Gymnasium. Evaluation was performed using metrics such as average score, highest score, average steps, and Q-value. The final results indicate that Softmax tends to explore early in training but achieves convergence toward the end, whereas UCB tends to exploit early, leading to stagnant scores. Based on t-test results, no significant difference was found in the performance of the two action selection methods. This study provides guidance on selecting action selection methods for agents in simple games.

1. Introduction

An agent can observe its environment through sensors and perform actions based on those observations. Agents can be physical robots, software, or even characters in a game. Each time an agent observes its environment, it produces an observation known as a percept. This perception is then processed through an agent function that determines the agent's next action. The implementation of this agent function in code is referred to as the agent program.

A game-playing agent (GPA) observes a game environment. Agents may not continuously operate alone; sometimes, two or more agents can operate together, depending on their objectives. Agents can also work in groups to achieve their goals. However, for a single agent, clear rules and fewer actions in a game simplify the research process.

Flappy Bird is a game created by Vietnamese developer Dong Nguyen. Released in 2013 by his company, Gears Studio (also known as dotGears), the game initially gained little attention. However, in 2014, its popularity surged due to its simple yet challenging gameplay. In Flappy Bird, players control a small, round, yellow pixel-art bird that must navigate through vertically aligned pipe obstacles. The bird can jump to pass through gaps by tapping the screen. The game ends if the bird touches a pipe or the ground, requiring players to jump to progress continuously. The ultimate goal is to achieve the highest score possible.

The mechanics of Flappy Bird are straightforward: tap to jump. Still, this simplicity can lead to boredom once players become accustomed to the game, resulting in errors due to fatigue and increasing the likelihood of failure. Given these human limitations, creating an agent to control the bird in Flappy Bird is an appealing research topic. Unlike humans, agents do not tire and have the potential to achieve significantly higher scores.

Numerous algorithms have been applied to develop Flappy Bird agents. Ardiansyah [1] implemented Q-learning with backpropagation, achieving 92% faster learning time and 94% less memory usage than standard Q-learning. [2] used a Deep Q-Network to train an agent with 300,000

training iterations, resulting in scores of 1456.6 for easy mode, 2598 for medium mode, and 73.45 for hard mode.

Q-learning is a viable algorithm for creating agents in video games, but training such agents is time-consuming. The value function is critical to agent training, which involves action selection based on the chosen method's calculations. The epsilon-greedy method exploits actions with the highest Q-value too quickly, neglecting the uncertainty of actions that might yield better Q-values in the future. Alternative value function methods, such as Softmax and Upper Confidence Bound (UCB), can address epsilon-greedy's shortcomings and accelerate agent training.

Softmax selects actions based on the probability of Q-values, favouring actions with higher Q-values while still allowing lower Q-value actions to be chosen. However, Softmax relies heavily on the tau parameter: a high tau value encourages exploration, while a low tau value leans toward exploitation.

UCB balances exploration and exploitation differently, assigning incentives or bonus points to actions with low selection frequency to encourage their selection. However, UCB requires additional computations to account for action selection uncertainty, which can slow the agent training process and make it less suitable for dynamic environments.

This study compares Softmax and UCB's performance in developing a Flappy Bird agent. Both methods are ideal for comparison because they rely on parameters to balance exploration and exploitation, making them suitable for evaluating action selection in this context.

After preparing all necessary materials and code, experiments were conducted using the Q-learning algorithm to compare the action selection methods Softmax and UCB. The comparison evaluated variables such as exploration frequency, exploitation level, the number of pipes the agent could pass, convergence speed, agent reward scores, and how quickly the agent achieved optimal rewards.

2. Method

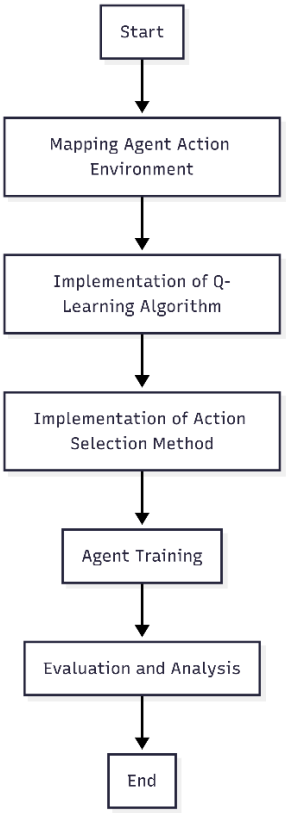

Figure 1. Research Flowchart

Based on the research flow diagram in Figure 1, the study proceeded as follows:

Agent Environment Preparation: This included all assets required for the study. The preparation was facilitated using the Gymnasium library, an API for reinforcement learning with diverse environments. The Flappy Bird environment was sourced from a Gymnasium contribution by GitHub user markub3327. Note that the Gymnasium does not directly manage this environment, but it is a community contribution.

Q-Learning Implementation: After setting up the agent environment with the Gymnasium library, Q-learning was implemented, including initialising the Q-table, updating Q-values, and applying two-step Q-learning, where rewards were calculated after two steps. The action space consisted of two actions: do nothing and flap. The state space comprised 12 states: horizontal position relative to the previous pipe, vertical position relative to the top of the previous pipe, vertical position relative to the bottom of the last pipe, horizontal position relative to the next pipe, vertical position relative to the top of the next pipe, vertical position relative to the bottom of the next pipe, horizontal position relative to the pipe after the next, vertical position relative to the top of the pipe after the next, vertical position relative to the bottom of the pipe after the next, the agent's vertical position, the agent's vertical velocity, and the agent's rotation. Discretisation was applied to each state, converting continuous values into 25 discrete values to improve computational efficiency and algorithm performance.

Action Selection Method Implementation: The Softmax and UCB action selection methods were implemented. During training, the agent selected one method per session. The UCB constant and Softmax tau parameters were adjusted using a linear decay pattern, where values decreased as episodes progressed. The learning rate and discount factor were also determined.

Agent Training: Training was conducted after implementing Q-learning, Q-table updates, two-step Q-learning, state discretisation, and the Softmax and UCB action selection methods. Training continued until convergence was achieved.

Evaluation: Once convergence was reached, agents' performance using Softmax and UCB was evaluated. Results were compiled into tables, analysed, and subjected to a t-test to compare the average rewards from each group's training.

Data Collection: The study used primary data collected through experiments. Data consisted of changes in independent variables recorded post-experiment.

Research Design: A quasi-experimental design was employed, with both groups receiving identical initial conditions. The study divided agents into two groups: one using Softmax and the other using UCB. Both groups used two-step Q-learning as the base algorithm and applied the Markov Decision Process as the framework for action selection.

Independent Variables: The action selection method and its respective parameters were the manipulated variables. While epsilon-greedy is commonly used, this study manipulated Softmax (tau parameter) and UCB (constant c). The learning rate and discount factor were also adjusted to optimise convergence.

Dependent Variables: The dependent variables were the average reward and the number of episodes required to achieve convergence, determined by stable reward acquisition per episode.

Data Collection Method: Data was collected through direct observation, generating primary data and logs from experiments. Independent and dependent variables were automatically recorded digitally during each iteration.

Analysis: Descriptive statistical analysis provided an overview of the collected data. Data from direct observations were tabulated, and a comparison was made between the Softmax and UCB groups. Metrics compared included average score, highest score, average steps, and average Q-value, as shown in Table 1.

Table 1. Comparison of Action Selection Method Performance

Metric	Softmax	UCB
Average Score	Average score for Softmax	Average score for UCB
Highest Score	Highest score for Softmax	Highest score for UCB
Average Steps	Average steps for Softmax	Average steps for UCB
Average Q-Value	Average Q-value for Softmax	Average Q-value for UCB

3. Results and Discussion

3.1. Softmax Testing Results

Table 2.Performance of Softmax Action Selection Method

Episode	Average Score	Highest Score	Average Steps	Average Q-Value
1000	-7.48	-0.9	50	-10.49
2000	-7.48	2.7	50	-13.16
3000	-7.49	2.7	50	-13.27
4000	-7.4	2.7	50	-13.11
5000	-7.38	2.7	50	-12.92
6000	-7.5	2.7	50	-13.28
7000	-7.42	2.7	50	-13.25
8000	0.62	4.4	33.78	1.74
9000	2	4.4	31	8.15
10000	2	4.4	31	8.17

Softmax is a technique that selects actions based on the probability of Q-values. This study shows that Softmax performs comparably to UCB. Based on t-test results, no significant difference was found between Softmax and UCB.

The experimental results in Table 2 indicate that Softmax heavily favoured exploration from the start until nearly the end of training (episodes 0–8000), with scores changing rapidly before shifting to exploitation in the final episodes (8000–10,000).

According to [3], in the Atari 2600 game, Softmax action selection outperformed epsilon-greedy by 4%–10%, avoiding random exploration. In Flappy Bird, Softmax was explored extensively but achieved performance equivalent to UCB by the end of training, with an average score of 2.

3.2. UCB Testing Results

Table 3.Performance of the UCB Action Selection Method

Episode	Average Score	Highest Score	Average Steps	Average Q-Value
1000	2.04	4.7	31.46	6.9
2000	2	4.7	31	8.17
3000	2	4.7	31	8.17
4000	2	4.7	31	8.17
5000	2	4.7	31	8.17
6000	2	4.7	31	8.17
7000	2	4.7	31	8.17
8000	2	4.7	31	8.17
9000	2	4.7	31	8.17
10000	2	4.7	31	8.17

Upper Confidence Bound (UCB) is another action selection technique in reinforcement learning that accounts for Q-value uncertainty. This study shows that UCB performs equivalently to Softmax, with no significant difference in t-test results.

The experimental results in Table 3 show that UCB was overly quick and unstable when selecting between the two actions in Flappy Bird. By episode 1000, UCB had already reached optimal performance and rapidly shifted to exploitation mode until the end of training.

According to [4], UCB performs better with 10 action options, as described in Figure 2.4, than epsilon-greedy. However, it struggles in environments with large state spaces. In the simpler Flappy

Bird environment with only two actions, UCB exhibited premature exploitation despite a sufficiently large constant c for exploration, resulting in stagnant scores until the end of training.

3.3. Comparison of Softmax and UCB Performance

Table 4.Comparison of Softmax and UCB Action Selection Performance

Episode	Average Score				Highest Score			
Softmax	*UCB*	*Softmax*	*UCB*	*Softmax*	*Softmax*	*UCB*	*Softmax*	*UCB*
1000	-7.48	2.04	-0.9	4.7	50	50	-10,49	-10,49
2000	-7.48	2	2.7	4.7	50	50	-13,16	-13,16
3000	-7.49	2	2.7	4.7	50	50	-13,27	-13,27
4000	-7.4	2	2.7	4.7	50	50	-13,11	-13,11
5000	-7.38	2	2.7	4.7	50	50	-12,92	-12,92
6000	-7.5	2	2.7	4.7	50	50	-13,28	-13,28
7000	-7.42	2	2.7	4.7	50	50	-13,25	-13,25
8000	0.62	2	4.4	4.7	33,78	33,78	1,74	1,74
9000	2	2	4.4	4.7	31	31	8,15	8,15
10000	2	2	4.4	4.7	31	31	8,17	8,17

3.4. T-Test Results

A t-test was conducted to compare the average score, highest score, average steps, and average Q-value between the UCB and Softmax methods. The results showed no significant difference between Softmax and UCB agents, with a p-value of 0.994 and a t-statistic of 0.008. Since the p-value is greater than 0.05, the hypothesis of a significant difference was rejected.

4. Conclusion

Training a Flappy Bird agent involved comparing two action selection methods, Softmax and Upper Confidence Bound, using two-step Q-learning as the base algorithm. Softmax and UCB achieved an average score of 2 and an average of 31 steps per episode. However, UCB achieved a slightly higher maximum score of 4.7 compared to Softmax's 4.4. UCB exhibited premature exploitation early in training despite a sufficiently large constant c, resulting in stagnant scores. In contrast, Softmax was explored extensively early on but achieved performance equivalent to UCB by the end of training.

References

[1] A. Ardiansyah and E. Rainarli, "Implementasi Q-Learning dan Backpropagation pada Agen yang Memainkan Permainan Flappy Bird," J. Nas. Tek. Elektro Dan Teknol. Inf. JNTETI, vol. 6, no. 1, Feb. 2017, doi: 10.22146/jnteti.v6i1.287.

[2] Y. Li, "Deep Reinforcement Learning for 2D Flappy Bird Game," SHS Web Conf., vol. 144, p. 03007, 2022, doi: 10.1051/shsconf/202214403007.

[3] A. K. Shakya, G. Pillai, and S. Chakrabarty, "Reinforcement learning algorithms: A brief survey," Expert Syst. Appl., vol. 231, p. 120495, Nov. 2023, doi: 10.1016/j.eswa.2023.120495.

[4] Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, Second edition, in progress. 2015. [Online]. Available: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

[5] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," IEEE Access, vol. 7, pp. 133653–133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[6] L. Meng, R. Gorbet, and D. Kulic, "Memory-based Deep Reinforcement Learning for POMDPs," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic: IEEE, Sep. 2021, pp. 5619–5626. doi: 10.1109/IROS51168.2021.9636140.

[7] P. Swazinna, S. Udluft, D. Hein, and T. Runkler, "Comparing Model-free and Model-based Algorithms for Offline Reinforcement Learning," Jan. 14, 2022, arXiv: arXiv:2201.05433. doi: 10.48550/arXiv.2201.05433.

[8] R. S. Sutton and A. G. Barto, Reinforcement learning: an introduction, Second edition. In the Adaptive Computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.

[9] R. S. Sutton, "Learning to predict by the methods of temporal differences," Mach. Learn., vol. 3, no. 1, pp. 9–44, Aug. 1988, doi: 10.1007/BF00115009.

[10] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," 2013, arXiv. doi: 10.48550/ARXIV.1312.5602.

[11] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," Adv. Appl. Math., vol. 6, no. 1, pp. 4–22, Mar. 1985, doi: 10.1016/0196-8858(85)90002-8.

[12] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A Theoretical Analysis of Deep Q-Learning," Feb. 24, 2020, arXiv: arXiv:1901.00137. Accessed: Nov. 04, 2024. [Online]. Available: http://arxiv.org/abs/1901.00137

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[14] L. Zhong, "Comparison of Q-learning and SARSA Reinforcement Learning Models on Cliff Walking Problem," in Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023), vol. 180, B. H. Ahmad, Ed., in Advances in Intelligent Systems Research, vol. 180. , Dordrecht: Atlantis Press International BV, 2024, pp. 207–213. doi: 10.2991/978-94-6463-370-2_23.

[15] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," Neural Comput., vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/neco_a_01199.

[16] T. Vu and L. Tran, "FlapAI Bird: Training an Agent to Play Flappy Bird Using Reinforcement Learning Techniques," Apr. 08, 2020, arXiv: arXiv:2003.09579. Accessed: Oct. 27, 2024. [Online]. Available: http://arxiv.org/abs/2003.09579

[17] Z. He, Y. Zhang, and D. Zhao, "Flappy Bird Game Based on Reinforcement Learning Q-Learning Algorithm," Highlights Sci. Eng. Technol., vol. 34, pp. 222–225, Feb. 2023, doi: 10.54097/hset.v34i.5475.

[18] K. Yang, "Using DQN and Double DQN to Play Flappy Bird," in Proceedings of the 2022 International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2022), vol. 7, N. Radojević, G. Xu, and D. D. H. K. H. Md Mansur, Eds., in Atlantis Highlights in Intelligent Systems, vol. 7. , Dordrecht: Atlantis Press International BV, 2023, pp. 1166–1174. doi: 10.2991/978-94-6463-010-7_120.

[19] Y. Guo, "Enhancing Flappy Bird Performance With Q-Learning and DQN Strategies," Highlights Sci. Eng. Technol., vol. 85, pp. 396–402, Mar. 2024, doi: 10.54097/qrded191.

[20] Louis-Samuel Pilcer, A. Hoorelbeke, and A. D'andigne, "Playing Flappy Bird with Deep Reinforcement Learning," 2018, doi: 10.13140/RG.2.2.13159.96165.

[21] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," Neural Netw., vol. 107, pp. 3–11, Nov. 2018, doi: 10.1016/j.neunet.2017.12.012.