Rainbow Q-Learning in Jelly Bean World

Thomas Jiralerspong, Flemming Kondrup, Sheheryar Parvaz

GROUP 025

Abstract

The Jelly Bean World is a novel environment proposed as a testbed for never-ending learning, where an agent aims to maximize rewards by capturing various items in a two-dimensional gridworld. Advances in model-free Deep Reinforcement Learning (DRL) have led to Rainbow, an approach that combines various highly-performing Q-Learning algorithms. In this work, we implement Rainbow and include features such as short-term memory and prioritized replay buffer. We show that using Rainbow in this environment leads to high performance and high sample efficiency. Furthermore, we evaluate our model's adaptability by testing it on alternative environments.

1. Introduction

The Jelly Bean World (Platanios et al. (2020)) was recently proposed for potentially never-ending learning in a two-dimensional gridworld. In this environment, the agent has limited perception of its environment and aims to maximize rewards received from various items. We investigate the use of reinforcement learning for effective learning within the Jelly Bean World environment. Specifically, we implement Rainbow (Hessel et al. (2017)), an approach that combines various efficient Q-Learning algorithms together.

2. Methods

2.1. Rainbow Q-Learning

Q-Learning aims to estimate the value of taking an action a from a state s, known as $Q(s,a)$. When the number of states is intractable, it becomes impractical to store in a table the Q-values for all state-action pairs. We can however use a function approximator to estimate the Q-value of any state-action pair. The Deep Q Network (DQN) ((Mnih, 2015)) algorithm combines Q-Learning with deep neural networks to handle complex RL problems. Since the introduction of the DQN, various efforts have been made to improve the model. This includes Double DQN which aims to reduce overestimation (van Hasselt et al. (2015)), Dueling DQN which splits the Q-values into a value function $V(s)$ and an advantage function $A(s,a)$ (Wang et al. (2015)) and more. Nonetheless, each of these approaches has both advantages and drawbacks. Rainbow (Hessel et al. (2017)) adresses this as it aims to combine these various approaches leading to a state-of-the-art Q-Learning model.

2.2. Implementation

We first implemented a DQN approach from scratch using the pseudocode discussed in class but found limited performance. Considering the reputation of Rainbow, we decided to attempt implementing it and used the code by Park (2019) as a foundation. This section gives more details on our approach adapting Rainbow to Jelly Bean World.

2.2.1. State Space

The agent's perception of its environment is limited to 3 components: *(1) vision* - a 15x15 RGB image of surroundings, *(2) feature vector* - a vector with binary information extracted from vision and *(3) scent* - the scent perceived by an agent, which depends on the objects near its location. Considering the feature vector contained all the information that would be received in vision in a simplified manner, we chose to disregard vision, as its inclusion would not be beneficial and would increase computational requirements. Thus, our state space consists of a scent vector of size 3 and a feature vector of size 900.

2.2.2. Adding Short-Term Memory

Since our state space does not take motion in account, it only tells our agent how far it is from a given object and disregards whether it is getting closer or further. To add this dimension, we include scent and features from previous timesteps, as well as the actions taken at those timesteps. We expect this to be particularly beneficial for scent, since its value is purely a measure of proximity and not direction. Additionally, memory in terms of feature vectors may give information about out of visual range objects. For example, if the agent sees 2 apples at opposite corners of its visual field, it can move to get the first and then use its memory to go back and get the second one.

2.2.3. Prioritized Replay Buffer

Experience replay (Lin (1992)) stores previous interactions to later use them in training, stabilizing training of the value function. Nonetheless, this replays previous interactions at the same frequency as they were experience, disregarding their relevance. Prioritized experience replay (Schaul et al. (2015)) addresses this by replaying more important interactions leading to more efficient training. We include prioritized replay buffer in our model.

2.2.4. Parameter search

We performed a thorough search for both parameters and architectures in order to find an optimal model. In terms of learning rates, we tried 1e-04, 1e-05, 1e-06 and found 1e-05 to be most beneficial. As mentioned in the previous section, we added a short-term memory to our model. We investigated what the optimal value for the number of frames to remember would be, exploring options in the range of 1, 5, 10, 20, 100 and found 5 to lead to the best performance. Furthermore, we experimented with the amounts of nodes per layer with values 128, 356, 512 and 1024 and found 512 nodes to lead to the best performance.

3. Results

3.1. Final Performance

We first investigate overall performance of our various models (see Table 1).

Table 1: Final reward of our model compared to rainbow without prioritized buffer, rainbow without history and a random policy on 5 seeds for 50 episodes

Model	Score
Rainbow Final Model	**797.18**
Rainbow without prioritized buffer	387.70
Rainbow without history	36.00
Random policy	14.75

We find that our model heavily outperforms the random policy, justifying it's placement in the leader-board. We find that Rainbow with a simple replay buffer is far outperformed by the one with prioritized buffer, and that rainbow without the short-term memory (history) performs close to equal to a random policy. These results highlight the potential of rainbow in Jelly Bean World and the importance of short-term memory and prioritized replay buffer.

3.2. Sample efficiency

We next consider sample efficiency, critical as it permits to train and deploy models more efficiently. We investigate performance over 500 000 time steps (see Figure 2).

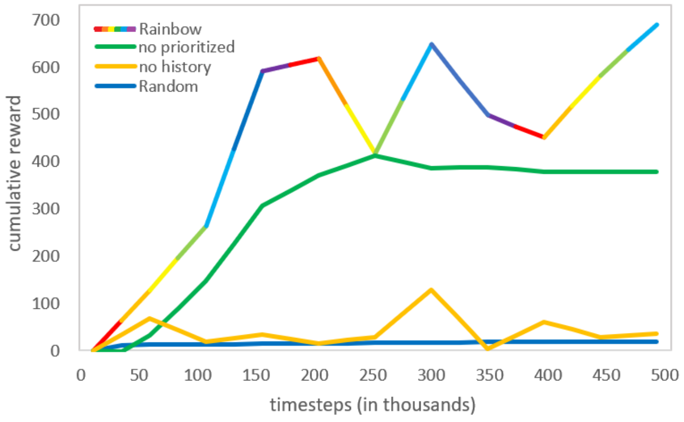

Figure 1: Cumulative reward received with Rainbow, Rainbow without prioritized buffer, Rainbow without history and a Random policy. Rainbow outperforms all alternatives in terms of final performance and sample efficiency

We find that Rainbow outperforms all alternative approaches. Removing the prioritized replay buffer leads to some loss in performance, slower increase in reward and a lower final reward. Removing the short-term memory (history) highly impede performance, with this policy often being comparable in performance to a random policy, highlighting the importance of memory for decision making in an environment like Jelly Bean World.

3.3. Adaptability

We performed two experiments to evaluate the adaptability of our model. Since the main goal of our model is to maximize positive rewards and minimize negative ones, we wanted to evaluate how swapping the rewards of items would affect performance. We first switch rewards for apples and jelly beans, thus -1 for collecting an apple and +1 for a jelly bean. In the second experiment, we wanted to evaluate switching the two positive rewards, and thus assigned +0.1 for collecting an apple and +1 for collecting a banana.

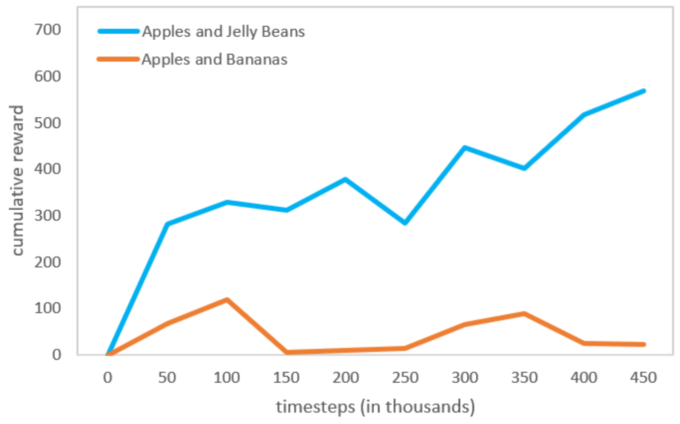

Figure 2: Cumulative reward received in the two experiments, switching rewards for apples with the ones of jelly beans and of bananas. We observe that our model quickly adapts to the apples and jelly beans being switched but faced difficulty when apples and bananas are switched

We find that our model quickly adapts in the first experiment with apples and jelly beans. Interestingly, the model performs a lot better compared to switching apples and bananas. We have a few hypotheses as to why this may be: (1) Our model may focus exclusively on apples and mostly ignore bananas, since their rewards are a lot less beneficial. In this case, when switching apples and bananas, our model may not explore sufficiently to get apples and learn this new pattern, (2) switching apples and jelly beans may lead to a much bigger shock, pushing our model to quickly change, whereas in the second experiment, it may still receive some positive reward and not explore alternatives as much.

4. Conclusion

In this work, we propose the use of Rainbow for sequential decision making in the Jelly Bean World environment. We show it leads to both high sample efficiency and final performance. Furthermore, we evaluate the relevance of including a short-term memory and a prioritized replay buffer, and show they both lead to improved performance. Finally, we evaluated our models adaptability to novel environments. Future works could consider alternative approaches to Rainbow and potentially bench-marking them in terms of performance and sample efficiency.

References

Matteo Hessel, Joseph Modayil, Hado van Hasselt, and et al. Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298, 2017. URL http://arxiv.org/abs/1710.02298.

LJ Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching, 1992. URL https://doi.org/10.1007/BF00992699.

V. et al. Mnih. Human-level control through deep reinforcement learning, 2015.

Curt Park. rainbow-is-all-you-need, 2019. URL https://github.com/Curt-Park/rainbow-is-all-you-need/.

Emmanouil Antonios Platanios, Abulhair Saparov, and Tom Mitchell. Jelly bean world: A testbed for never-ending learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Byx_YAVYPH.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015. URL https://arxiv.org/abs/1511.05952.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015. URL https://arxiv.org/abs/1509.06461.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2015. URL https://arxiv.org/abs/1511.06581.