Adaptive Epsilon Annealing with Rainbow DQN

Samuel Lin S.Lin25@bcp.org
Bellarmine College Preparatory

Abstract

Given recent developments in more advanced algorithms, there has been less focus on exploration vs. exploitation concerns in the context of epsilon-greedy strategies. This paper examines a different strategy regarding epsilon annealing for an epsilon-greedy approach when paired with methods utilized in Rainbow DQN. While it is generally accepted to decrease the value of epsilon during the entirety of training, our efforts were fueled by ideas of possible ways to improve the rate of convergence to an optimal policy. For our strategy, we altered the value of epsilon depending on the performance of randomly selected actions under an epsilon-greedy approach.

1. Introduction

Fundamentally, traditional reinforcement learning experiences a bottleneck regarding memory limits as the number of state-action pairs in any mildly complex environment greatly exceeds the memory capacity in modern day machines. Given vast improvements in computational power, neural networks have been greatly popularized in the usage of approximation methods in modern reinforcement learning algorithms. These approaches bypass the issues associated with memory and constitute what is known as deep reinforcement learning. For instance, the Deep Q-Network (DQN) approach (Mnih et al. 2015) expands upon the effectiveness of prior ideas from Q-Learning by replacing the Q-table array representation with 2 neural networks for approximation. Primarily, the main area of focus lies with Rainbow DQN, integrating multiple ideas and expansions on traditional DQN (Hessel et al. 2017).

2. Background

Reinforcement learning consists of various approaches to improving a policy. For this paper, we focus on off-policy learning algorithms, as with DQN.

2.1 Epsilon-Greedy

One of the fundamental concepts of reinforcement learning is the idea of the duel between exploration and exploitation, where an agent explores an environment to learn the optimal policy at the detriment of short-term rewards, and exploits it at the cost of potentially overlooking the best strategy due to lack of exploration. Finding a balance between exploration and exploitation is necessary for optimal learning (Sutton and Barto 1998). Specifically, ϵ-greedy approaches involve the exploration parameter ϵ, which defines the rate of exploration, and the probability of selecting a random (non-optimal) choice compared to the assumed optimal choice during training. Additionally, the value of ϵ is also commonly annealed across the training process in order to facilitate better results and to shift to an exploitative policy.

2.2 Rainbow DQN

Rainbow DQN combines multiple optimizations to the DQN algorithm. Notable changes include the integration of Deep Q-Network with Prioritized experience replay, or PER (Schaul et al. 2015), Double Q-Learning (van Hasselt 2015), and dueling architecture (Wang et al. 2016). These three factors alone allow for significant improvements in performance. Additionally, Rainbow DQN also replaces standard epsilon-greedy action selection strategies with noisy net implementations to reduce overfitting and potentially allow for faster optimizations in certain scenarios.

DQN

Traditional DQN uses two neural networks to approximate and minimize the Bellman error. This update is modeled by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma max_{\mathbf{a}} Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Additionally, the two networks are split into what are known as the target and online networks, introducing a hyperparameter dictating how many time steps there are in between which the online network is copied to the target network in order to tackle the notion of following a moving target.

Double DQN

Double DQN is a method focused on reducing overestimation of Q-values, through the updates of the online network being partially dependent on the target network. Let Q be online network and Q' be target network. This primary difference compared to regular DQN is represented through the expressions

$$a' = max_{\mathbf{a}} Q(s_{t+1}, a)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma Q'(s_{t+1}, a') - Q(s_t, a_t)]$$

While seemingly a minute change, this adjustment heavily improves the performance of agents in certain environments as mentioned in previous studies (van Hasselt et al. 2015).

Dueling DQN

Dueling DQN helps the agent better distinguish between different actions, allowing for much better learning as well as reduce overestimations. In order to accomplish this, the central idea behind the architecture lies in the introduction of two streams in the neural net, namely the advantage stream and the value stream. These two streams provide an intermediate step before being recomputed to produce the original Q-value layer as the output. Specifically, the value stream estimates the value for a given state and the advantage stream estimates how good each action is compared to each other. When determining the

Q-values, we compute the output layer using

$$Q(s,a) = V(s) + (A(s,a) - \tfrac{1}{|A|}\Sigma A(s,a))$$

The idea of using the mean instead of the maximum of the advantage stream was mainly targeted towards improving the stability of the optimization (Wang et al. 2016).

Prioritized Experience Replay

Perhaps one of the most significant improvements, PER helps to improve the quality of training by choosing to sample experiences that have a larger effect on the agent's learning. To accomplish this, each experience in the buffer is assigned a priority, which is determined based on the TD-error. Furthermore, the transitions are also selected through random sampling, with the probabilities of selecting each transition being dependent on their priority. In detail, the TD error is calculated as follows with a' and Q' defined earlier

$$\delta_t = r(s_t, a_t) + \gamma Q'(s_{t+1}, a') - Q(s_t, a_t)$$
$$p_t \leftarrow |\delta_t|$$

Note that this is when implemented with double DQN, and with normal DQN, the update would change to the form prior mentioned. The probabilities for selecting action i can therefore be computed as

$$P(i) = \frac{p_i^\alpha}{\Sigma_t p_t^\alpha}$$

However, using PER still introduces bias, as the distribution is non-uniform, therefore requiring the use of what are known as the importance-sampling weights (Schaul et al. 2015). These weights are calculated as follows

$$w_i = (\tfrac{1}{N} \cdot \tfrac{1}{P(i)})^\beta$$

We use this weight when we update our neural networks as it allows for stability and to help with learning.

Noisy Nets

Noisy nets provide an alternate method to previous epsilon-greedy approaches by adding noise to the neural networks, simulating the random factor in ϵ-greedy. When applying noise into the neural net, the following expression is used

$$y = (\mu^w + \sigma^w \odot \epsilon^w)x + \mu^b + \sigma^b \odot \epsilon^b$$

instead of the conventional

$$y = wx + b$$

when adjusting weights for neural networks. These changes allow for substantial improvements across many environments (Fortunato et al. 2018).

3. Related Works

Adaptive Epsilon

In a paper by Tokic (2010), he discusses the downsides of having a standard epsilon annealing strategy based on time in large state spaces, given that exploration is encouraged in unfamiliar environments, but standard epsilon annealing strategies disregard these concerns.

As an alternative, Tokic suggests an epsilon value dependent on the value-function error which is calculated as follows:

$$f(s, a, \sigma) = \frac{1-e^{\frac{-|\alpha \cdot TD-Error|}{\sigma}}}{1+e^{\frac{-|\alpha \cdot TD-Error|}{\sigma}}}$$

$$\epsilon_{t+1}(s) = \delta \cdot f(s_t, a_t, \sigma) + (1 - \delta) \cdot \epsilon_t(s)$$

Here, σ is defined as the *inverse sensitivity* and δ as another hyperparameter adjusting the influence of the selected action on exploration (Tokic 2010).

Proposed Method

In our approach, we draw similar ideas from the paper above regarding the usage of temporal difference in defining the quality and importance of each experience. Such priorities have already been calculated through our implementation of PER included in Rainbow DQN.

As explained prior, the TD-error associated with each transition helps determine the priority of training on select experiences. By assigning an additional attribute detailing whether each transition in the buffer was chosen through exploration or exploitation (random or greedy), we are facilitated with the means to determine whether the final sampling contains transitions that are classified as random, hinting at whether random experiences at the time are more valuable in the agent's learning.

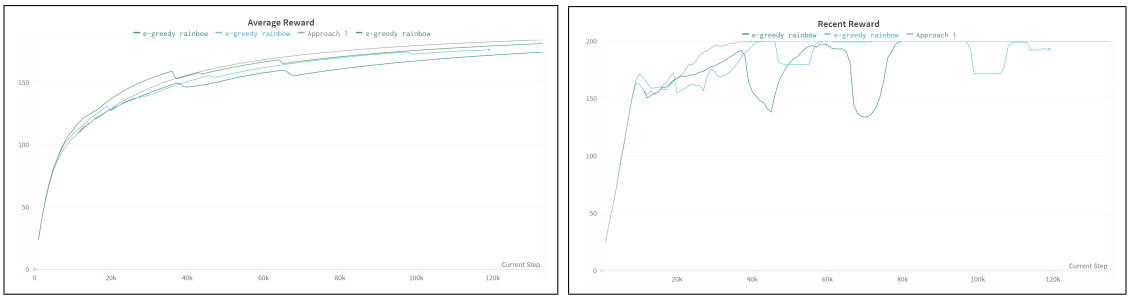
However, when determining the application of the proposed formula by Tokic (2010), the method is better suited for on-policy algorithms and computations, leading to us proposing an alternative. While aiming to maintain the dependency of epsilon on the TD-error, while simultaneously sampling multiple experiences at once each with different errors, we instead experimented with two different annealing methods. For each method we modeled off of a standard epsilon decay strategy, but differed in the implementation of an adaptive epsilon.

In the first case, we adapt the length of time for epsilon decay as this allows for slight fluctuations in the value of epsilon while still maintaining a overall decrease across an adjusted decay period. This approach introduces a new hyperparameter that we will name δ to fit previous conventions. Like before, δ will signify the impact of each action on the value of epsilon. However, contrary to the bounds of [0, 1) mentioned in the paper, our value of δ may assume any value ≥ 0, but for general tuning, one may prefer to tune their epsilon decay time steps prior to tuning the value of δ so as to assume more reasonable parameters.

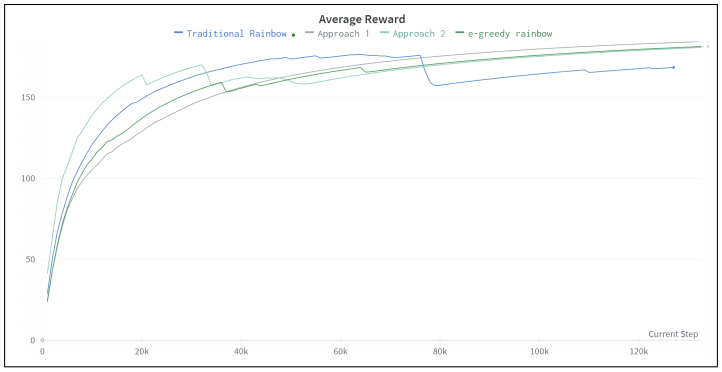
For our second approach, we experimented with a more direct epsilon computation with a secondary epsilon decay, acting as an upper bound for possible values of epsilon. For this method, we avoid introducing any new hyperparameters, and instead re-tune the initial value of epsilon, and the time of decay. This is due to the computation of epsilon being derived from a ratio determined by the priorities of transitions labeled as non-optimal (random) and the amount found in the final sampling being multiplied by the decaying epsilon to determine a temporary epsilon value for the next step (or an interval of steps). This method likely results in greater fluctuations in epsilon values, given the distributional nature of PER, but still serves to account for the issues mentioned prior with a permanent non-increasing epsilon annealing function.

4. Experimental Results

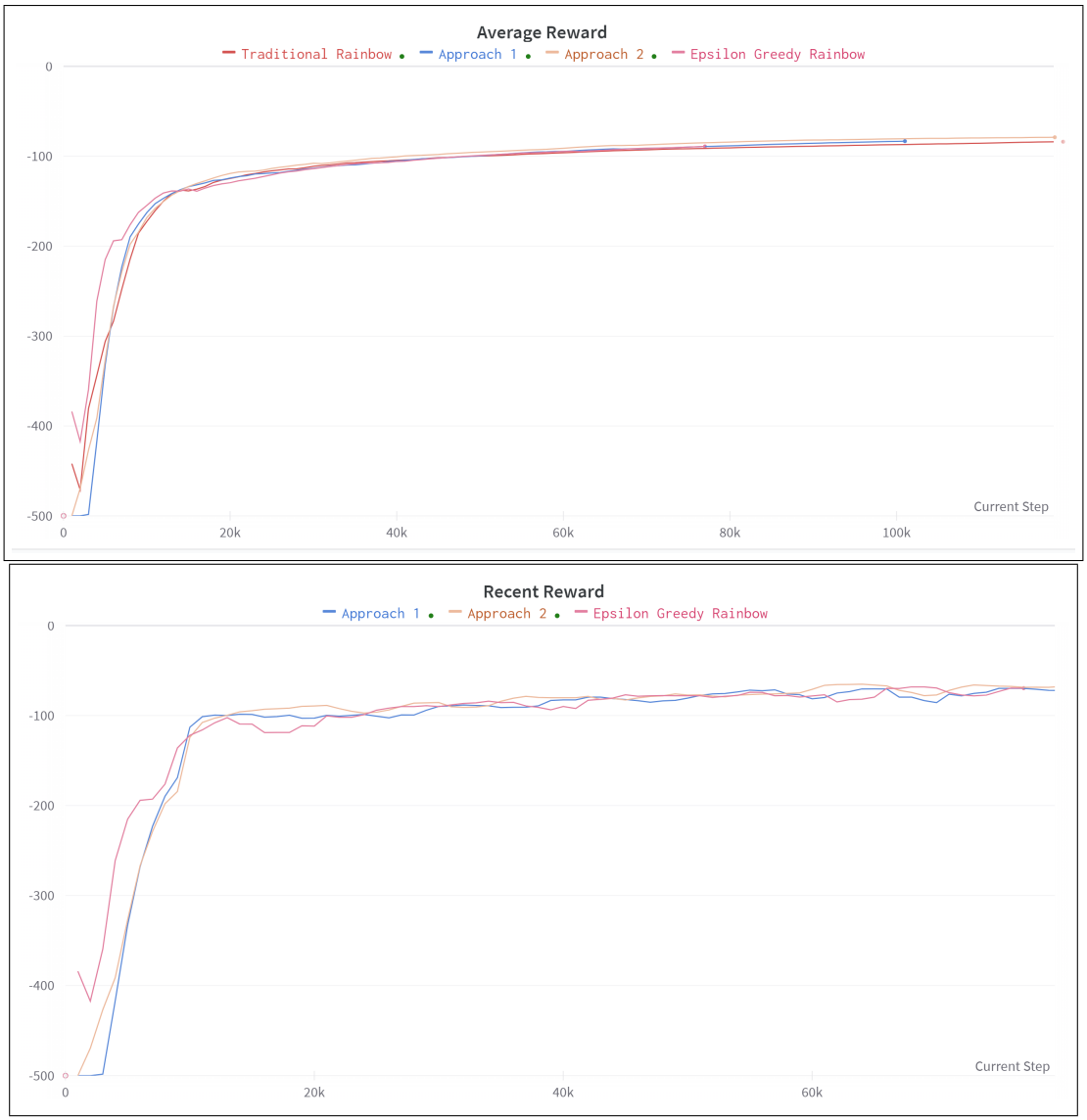
Due to limitations in computational power, we tested on two relatively basic environments. The first experiment was done on Cartpole-v0, and the second experiment was done on Acrobot-v1.



The graph on the left compares multiple baseline epsilon-greedy rainbow performances to an adaptive epsilon approach using the former implementation method. Two baseline trajectories branch out from distinct checkpoints along the adaptive epsilon approach, while the third starts from the beginning. From the graph, the adaptive epsilon approach appears to outperform the other baselines. However, this may also face some variance as sampling luck may alter the performance of these approaches. The graph on the right displays the average reward within the last 50 episodes, which accounts for the higher fluctuation in reward, but convergence is more apparent.



This final graph highlights an overall comparison between Traditional Rainbow DQN (Using Noisy Nets for exploration), e-greedy Rainbow DQN, our adaptive epsilon approach using the first implementation, and our adaptive epsilon approach using the second implementation.



The graphs above showcase the same plots as before, but instead work with the Acrobot-v1 environment.

4.1 Analysis

Cartpole

From the graphs, it is apparent that the first implementation outperforms the baseline

overall, while the second implementation experiences a much higher rate of growth earlier on, since random actions are more frequently prioritized, but eventually decreases to match the epsilon-greedy baseline. This behavior displays potential promise with implementing slightly altered methodologies to provide more consistent behavior during training.

When compared with the epsilon-greedy Rainbow baseline, the first approach shares a more consistent behavior with a gradual increase and quick convergence. On the other hand, the second approach faces similar traits as traditional Rainbow DQN utilizing Noisy Nets, as both methods achieve a rate of growth that is much larger than the others at the start. However, both methods seem to face the difficulty of stability, as they both experiences relatively large drops in average reward, suggesting that if better tuned, or better trajectories were sampled, both methods could likely outperform the others by a respectable margin. While comparing the two performances before their respective drops, it is clear that the second adaptive epsilon approach does in fact have a better trajectory, although the stability does appear to be slightly worse than Rainbow DQN with Noisy Nets.

Acrobot
As opposed to the graphs for Cartpole, from our testing, the second adaptive epsilon approach seemed to perform the best out of all the e-greedy approaches. Here, there is less visible distinction between the different approaches, but there are still general distinctions to be made between the different methods. Additionally, in this environment, training appears to be more stable than in the previous environment, allowing for more similarities between all methods.

4.2 Hyperparameters

One of the main flaws in the performances of both regular Rainbow DQN and the second approach was the aforementioned stability. Possible explanations of this behavior may be attributed to a relatively frequent update rate hyperparameter between the online and target network causing the idea of chasing a moving target. After tuning, despite this notion, the current update rate still seemed to perform the best in terms of trajectories compared to other more stable alternatives. Note that this factor is most likely the cause for the under-performance of traditional Rainbow DQN in comparison to e-greedy Rainbow, but allows for a much better initial trajectory for all approaches. Additional tuning of this parameter may allow more conclusive results in terms of comparison.

Other hyperparameters were tuned similarly to methods mentioned in other papers detailing similar DQN approaches. Another two hyperparameters that were notable are the two newly created parameters for the two types of implementations of an adaptive epsilon strategy. From initial testing, the first hyperparameter δ, values within a range of [0, 5] generally work the best as values beyond such ranges would result in better results if the normal epsilon decay were tuned instead, and a smaller value of δ was chosen. As for the second introduced hyperparameter generating an upper bound decay for epsilon, experiments reflect the overall better performance seen with generous restraints as the epsilon values usually converge to the minimum amount, prior to the upper bound converging.

5. Conclusion

By comparing the results between various methods involving DQN, as with Rainbow DQN, Rainbow DQN without Noisy Nets, and both implementations of an adaptive epsilon annealing strategy, we attempted to investigate potential improvements associated with integrating the performance of exploratory actions into adjusting the rate of exploration, ultimately leading to faster learning. This paper addresses some possible implementations, but there may be more sophisticated approaches associating the two factors which allow for better performance. Additionally, the environments presented in this paper were generally not suited for making the best use of such methods, as these adaptive exploratory strategies would likely excel in environments with larger action spaces, but with sequences of events necessary to fully explore deeper states in the environment.

For instance, given that an agent is required to perform a sequence of events to pass a stage in order to move onto a secondary portion of an environment, the value of epsilon would need to be sufficiently annealed in order to complete the first stage of an environment, as too high of a value for epsilon would likely lead to the agent never reaching past the first stage. However, this would likely hinder exploration in later unfamiliar stages, whereas adaptive epsilon approaches would likely allow for a faster learning rate in these secondary stages. Unfortunately, due to the scope of these ideal scenarios being too large for satisfactory tuning with our computational limits, we were unable to fully experiment with the potential of these approaches. In general, we believe there is still much to be explored with an adaptive exploration strategy, which may result in interesting alternatives for future research.

References

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. (2017). Noisy networks for exploration. *ArXiv*, abs/1706.10295.

Hasselt, H. V., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*.

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.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *CoRR*, abs/1511.05952.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Tokic, M. (2010). Adaptive epsilon-greedy exploration in reinforcement learning based on value difference. In *Deutsche Jahrestagung für Künstliche Intelligenz*.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H. V., Lanctot, M., and de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*.