Analyzing the effect of adaptive ε-greedy on Deep Q-learning in Atari game environments

Nasik Muhammad Nafi
Kansas State University
nnafi@ksu.edu

Abstract

Exploration exploitation dilemma is an important issue in learning algorithm. This paper presents a new approach to adaptive ε-greedy policy for better balancing between exploration and exploitation. This method is based on classic ε-greedy but it allows the value of ε to increase proportional to a predefined number of state transisions when the mean reward value goes through a certain reduction. We present experiments numerically comparing the two methods for the Breakout Atari Game environment. The adaptive ε-greedy method presents better performance as compared to the classic decreasing ε-greedy.

1 Introduction

Reinforcement learning is a form of machine learning in which an agent learns from its interaction with an environment to achieve a given goal. By interacting with the environment through actions taken, the agent receives numerical rewards. The agent's goal is to maximize the total number of rewards received.

A distinctive challenge that arises in reinforcement learning lies in balancing between exploration and exploitation. Exploitation means that the agent selects the action that has the highest average rewards. Exploration means that the agent selects an action randomly, regardless of the reward values obtained previous.

This paper introduces an adaptive ε-greedy method. This method is based on the classic ε-greedy method but allows the value of ε to be changed according to the rewards received from the environment. We numerically compared the two approaches for the Breakout Atari game. The adaptive ε-greedy presented superior performance to the classic ε-greedy.

The paper is organized as follows. Section 2 discusses related works. Section 3 presents the adaptive ε-greedy method. Section 4 presents the results of the experiments. Finally, Section 5 presents the conclusions and future work.

2 Related Works

In the ε-greedy policy the amount of exploration is globally controlled by a parameter, ε. Epsilon determines the randomness in action selections. No memorization of exploration specific data is required for this strategy. Only big issue is to determine which setting of ε leads to good results.

With ε-greedy, at each time step, the agent selects a random action with a fixed probability, $0 \leq \varepsilon \leq 1$, instead of selecting greedily one of the learned optimal actions with respect to the Q-function

$$\pi(s) = \begin{cases} \text{random action from } \mathcal{A}(s) & \text{if } \xi < \varepsilon \\ \text{argmax}_{a \in \mathcal{A}(s)} Q(s,a) & \text{otherwise,} \end{cases}$$

where $0 \leq \xi \leq 1$ is a uniform random number drawn at each time step. If epsilon is 1.0 then the actions are always random. If epsilon is 0.0 then the actions are always argmax for the Q-values. For $0 < \varepsilon < 1$ any random action is taken with probability ε

2.1 Decreasing-ε:

Decreasing- ε generally decreases the value of ε linearly from 1.0 to 0.1 over number of state transitions. Most of the cases this number of states is on around 1 million state transitions. This uses the idea of considering "time" in order to reduce the exploration probability, but what is known to be less related to the true learning progress. For example, why should an agent be less explorative in unknown parts of a large state space due to a time-decayed exploration rate?

2.2 Value-Difference Based Exploration

This method adapts the exploration parameter of ε-greedy in dependence of the temporal-difference error observed from value-function backups, which is considered as a measure of the agent's uncertainty about the environment in the preamble. y. For this, the proposed VDBE method extends ε-greedy [2] by adapting a state dependent exploration probability, ε(s), instead of the classical handtuning of this

globally used parameter. The key idea is to consider the TD-error observed from value-function backups as a measure of the agent's uncertainty about the environment, which directly affects the exploration probability.

The desired behavior is to have the agent more explorative in situations when the knowledge about the environment is uncertain, i.e. at the beginning of the learning process, which is recognized as large changes in the value function. On the other hand, the exploration rate should be reduced as the agent's knowledge becomes certain about the environment, which can be recognized as very small or no changes in the value function. For this, the following equations adapt such desired behavior according to a (softmax) Boltzmann distribution of the value-function estimates, which is performed after each learning step by

$$
\begin{aligned}
f(s,a,\sigma) &= \left| \frac{e^{\frac{Q_t(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} - \frac{e^{\frac{Q_{t+1}(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} \right| \\
&= \frac{1 - e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}}{1 + e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}} \\
&= \frac{1 - e^{\frac{-|\alpha \cdot \mathrm{TD-Error}|}{\sigma}}}{1 + e^{\frac{-|\alpha \cdot \mathrm{TD-Error}|}{\sigma}}} \\
\varepsilon_{t+1}(s) &= \delta \cdot f(s_t, a_t, \sigma) + (1 - \delta) \cdot \varepsilon_t(s) \ ,
\end{aligned}
$$

where σ is a positive constant called inverse sensitivity and δ ∈ [0, 1) a parameter determining the influence of the selected action on the exploration rate. An obvious setting for δ may be the inverse of the number of actions in the current state, δ = 1/ |A(s)| , which led to good results in our experiments. The resulting effect of σ is depicted in Figure 1. It is shown that low inverse sensitivities cause full exploration even at small value changes. On the other hand, high inverse sensitivities cause a high level of exploration only at large value changes. In the limit, however, the exploration rate converges to zero as the Q-function converges, which results in pure greedy action selection. At the beginning of the learning process, the exploration rate is initialized by $\varepsilon_{t=0}(s) = 1$ for all states.

2.3 Recent Adaptive ε-greedy

This adaptive ε-greedy method has two configurable parameters: l and f. This added to the classic ε-greedy exploration mode one adaptive action that can change the value of ε. Parameter l is used to set how many times to run the exploration mode before performing the adaptive action that changes the value of ε. Parameter f is used to regularize the calculated values of the rewards received to obtain an adequate value of the function that produces a new value for ε.

Algorithm 1 presents the algorithm used in the adaptive ε-greedy method. This algorithm is used to choose an action a when it is in a state s. It decides whether to perform exploitation or exploration mode. If it decides for the exploitation mode, it selects the action with highest average reward (A∗ t). If it decides for the exploration mode, it selects an action randomly. When the algorithm is in the exploration mode, it can perform an adaptive action, depending on its configuration, which modifies the value of ε.

1: $max_{prev} \leftarrow 0$
2: $k \leftarrow 0$
3: **if** normal distribution ≤ ε **then**
4: $max_{curr} \leftarrow Q_t(A_t^*)$
5: $k \leftarrow k + 1$
6: **if** $k = l$ **then**
7: $\Delta \leftarrow (max_{curr} - max_{prev}) * f$
8: **if** $\Delta > 0$ **then**
9: $\varepsilon \leftarrow sigmoid(\Delta)$
10: **else**
11: **if** $\Delta < 0$ **then**
12: $\varepsilon \leftarrow 0.5$
13: **end if**
14: **end if**
15: $max_{prev} \leftarrow max_{curr}$
16: $k \leftarrow 0$
17: **end if**
18: randomly selects an action
19: **else**
20: selects A_t^*
21: **end if**

The variables maxprev and k are static, being used to store the state of the algorithm. Variable k is used to count how many times the algorithm performs the exploration mode, after the last time it changed the value of ε. When the k variable reaches the limit specified in the l parameter, the algorithm decides whether to alter the value of ε. To do so, the difference (Δ) between the highest average rewards (maxcurr) and the highest previous rewards average (maxprev), obtained in the last time the Value of the variable k has reached the limit l. This difference is multiplied by the value of the parameter f to regularize its value. If the Δ value is greater than zero, a new value is calculated for ε. If the Δ value is less than zero,it means that after the last setting of a new value for ε, the algorithm has more often selected actions that are not optimal, and we must thus continue to search for optimal actions. For this, ε = 0.5 is defined. Ifthe Δ value equals zero,its current value remains. After setting the new value of ε, variable k is zeroed and the variable max-prev receives the value of the variable max-curr.

A new value for ε is defined by using a sigmoid function:

$$sigmoid(x) = \frac{1.0}{1.0 + \exp(-2 * x)} - 0.5.$$

According to this function, the value of ε can vary between 0.0 and 0.5. Empirically, it has been found that there is no significant gain if the value of ε is greater than 0.5.

3 Methodology

I developed a adaptive ε-greedy method which is based on the classic ε-greedy but allows the value of ε to vary in a controlled way throughout the execution. Changing the value of ε is accomplished by performing an adaptive action triggered through out the process. A new value for ε is calculated based on the rewards received from the environment.

3.1 Adaptive Approach

My strategy is to adaptively change the value of epsilon depending on the mean reward of a predefined last episodes. Here I have chosen 100 as the number of last predefined episode. I chose this value because values less than that gives more oscilation in the value of mean over that period.

If the value of mean reward over this last 100 episodes goes down 10% less than the mean reward value recorded at the previous 100 episodes, then I will increase the value of epsilon to allow more exploration. More exploration will increase its ability to explore more spaces in the search space and to correctly identify its target function.

Now the question is how we will calculate the new value of epsilon? How much we should increase or decrease the value of epsilon? In this regard I am using a similar notion of What we want to do when we do something wrong? We generally think if we can go backwards through these steps! And try to explore the other options. So here I am calculating the state trasitions happened in the last 100 episodes and increased the epsilon to value by that is linear in the no of state transitions. We can symmarize it as follows,

If (current_reward < prev_reward) //10% less
state_transitions = current_state_count – state-count-100-episode-ago
*new_epsilon = current-epsilon + state-transitions * delta*
Else
Decrease linearly until 0.1

Here, delta = (1.0 - 0.1)/ 1e6
That means delta is the amount of epsilon change per state transition. As we are decreasing epsilon linearly over 1 million state transitions.

If the reduction in mean reward over the 100 episodes is not large or even no reduction then the value of epsilon will be decreased linearly at every state transitions.

4 Experiments and Results

The proposed method is evaluated on the Breakout Atari game environment. This is an game with 896 highest reward. This game's action space has 6 actions named NO-OP, LEFT, RIGHT, FIRE, LEFT-FIRE, RIGHT_FIRE. In our case the reward value is bounded to [-1,1]. That means at every state transition agent is given at most 1 point even if it scores more.

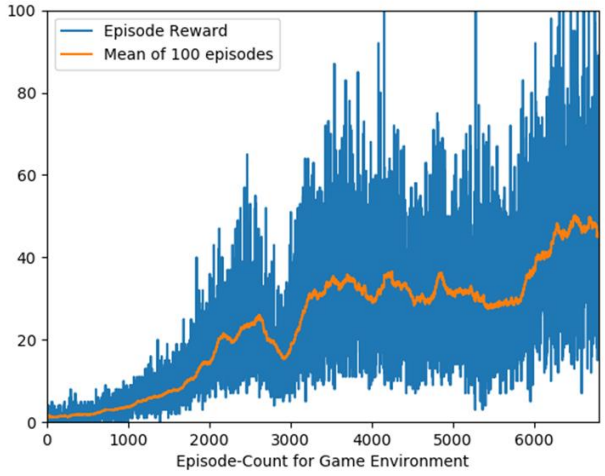
4.1 Experiment Setup

I executed all the training on a Quadro M4000 GPU enabled Machine with 30 GB RAM which runs on Ubuntu 16.04.
I trained both agent over 6500 episodes. Nearly 5-6 million state transitions occurred in these episodes.

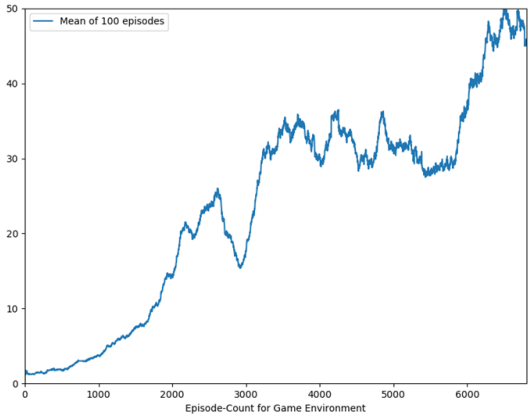
4.2 Evaluation Criteria

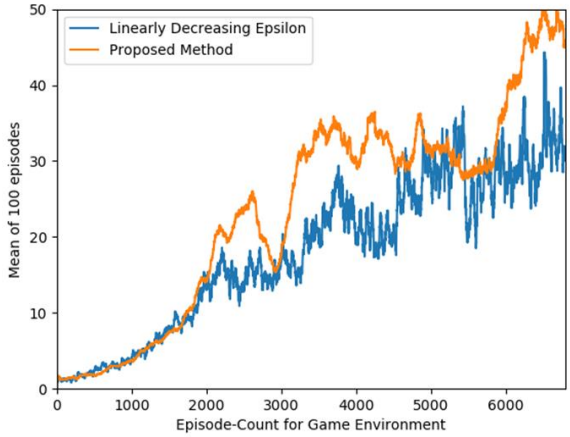
I used the HVass's Labs DQN implementation for the decreasing- **ε** policy. Here esilon value decreased linearly from 1.0 to 0.1 in 1 million state trannsitions. I compared the performance of the proposed approach against this approach. Mean reward over 100 episodes is used as the measure of performance.

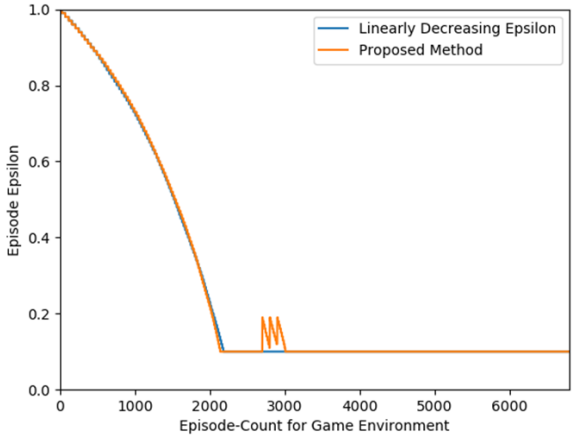
4.3 Results



Here I plotted mean reward over 100 episodes and every episode rewards against episode count. I executed this evaluation for 6802 operations. In this episodes it made a state transition of 6599876 and made a highest reward of 49.6. A closer view of the mean reward curve will give a more comprehensive understanding of the progress of mean reward. On the other hand, decreasing- **ε** based approach made a state transition of 5675949 and a highest reward of 44.1 in the this 6802 episodes. So both in terms of state explored and highest/average reward my approach is crearly performing well than the decreasing- **ε** approach.







5 Conclusion and Future Works

A distinctive challenge of the area of reinforcement learning is the balancing between exploration and exploitation. ε-greedy is a simple method for this balancing; however, the ε value is static or just linearly changing for some period. In this work, we introduce the adaptive ε-greedy method that allows the value of ε to be dynamic, adapting to the behavior of the environment. The experiments performed showed that the new approach outperforms the existing decresing - ε approach.

As future work, I intend to study the outcome of Training the agent over 1 million episodes or 50 million state transitions. I want to incorporate my strategy with the OpenAI Baseline code. Also I would like to check the performance of this approach on other Atari game evironments.

Acknowledgments

I would like to thank Dr. Hsu for his guidance and class materials. I would like to thank my fellow classmate Vahid for his continuous support to shape the projects into a success. Also I would like to thank Aditya who helped me to build a good understanding about the reinforcement learning process. Finally, I would like to thank HVass's lab for their DQN implementation and tutorial.

References

[Mnih, 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In Neural Information Processing Systems 2013

[Watkins et al., 1989] Watkins, C.: Learning from Delayed Rewards. PhD thesis, University of Cambridge, Cambridge, England (1989)

[Mnih, 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[Sutton,1998] Sutton, R. and Barto, A. (1998). Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.

[Tokic,2010] Michel Tokic "Adaptive ε-greedy Exploration in Reinforcement Learning Based on Value Differences "

[Tokic, 2011] Michel Tokic, and Gu¨nther Palm1 "Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax".

[Hvass,2017] Hvass Labs. https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/16_Reinforcement_Learning.ipynb