DEEP BANANA EATER – A DEEP Q-NETWORK REINFORCEMENT LEARNING AGENT

JULIAN WERGIELUK

ABSTRACT. This short note provides a concise description of the model architecture and learning algorithms of the agent developed in this project. We also report learning performance of the agent and provide a list of possible future model improvements.

1. DESCRIPTION OF THE LEARNING ALGORITHM

The algorithm used to solve the problem posed in this project closely resembles the algorithm proposed in [2].

The agent is trained over a given number of episodes labeled with $n = 1, \cdots, 1800$. Each episode is divided into turns $t = 1, \cdots, 300$. The state space $\mathscr{S}$ of the environment is continuous and given by $\mathscr{S} = \mathbb{R}^{37}$, whereas the action space $\mathscr{A} = \{0, 1, 2, 3\}$ is discrete and independent of the current environment state (i.e. all four actions are available irrespective of the current state). The state-action value function $Q_\theta : \mathscr{S} \times \mathscr{A} \mapsto \mathbb{R}$ maps a state-action pair to an estimated expected discounted cumulative reward generated by an agent following a greedy policy π_θ derived from Q_θ. The parameter vector θ determining the state-action value function Q_θ takes values in a finite-dimensional vector space $\mathbb{R}^p$.

We count the turns across the episodes of a training session using the variable $T = 300(n-1) + t$. At each turn T during the training time the agent chooses an action using an ε-greedy policy derived from $Q_{\theta(T)}$ and $\varepsilon = \varepsilon(T) = \exp(-T\lambda)$ with constant decay rate $\lambda = 0.00002$ (see Figure 2).

Also, at each turn $t < 300$, the experience tuple $(S_t, A_t, R_{t+1}, S_{t+1})$ is saved to the replay buffer. If the turn number t is divisible by 4 and the replay buffer contains more than $B = 128$ saved tuples, the Q function parameters θ are updated as follows: The algorithm samples B experience tuples (without replacements) from the replay buffer, and for each tuple (S_0, A_0, R_1, S_1) the following loss value $L_\theta(S_0, A_0, R_1, S_1)$ is calculated

$$L_\theta(S_0, A_0, R_1, S_1) = \left[R_1 + \gamma \max_{a \in \mathscr{A}} Q_\theta(S_1, a) - Q_\theta(S_0, A_0)\right]^2.$$

Assume that for each $(s, a) \in \mathscr{S} \times \mathscr{A}$ the function $\theta \mapsto Q_\theta(s, a)$ is differentiable. It follows that the function $\theta \mapsto L_\theta(S_0, A_0, R_1, S_1)$ is also differentiable. We can calculate the (total) derivative $\frac{dL}{d\theta}$ and use the ADAM optimizer to update the parameter vector θ with the aim of minimizing the loss L_θ. We perform this minimization step for each experience tuple in the sampled experience batch.

March 24, 2019

TABLE 1. List of hyperparameters and their values

Hyperparameter	Value
Learning rate of the ADAM optimizer	0.0005
Q-network update frequency	every 4 turns
Replay buffer size	100,000
Batch size	128
Discount factor (γ)	0.99
ε	Decays from 1 to 0 with the decay rate $\lambda = 0.00002$.

To ensure the validity of the minimization step, we use a deep neural network with two fully connected layers to represent the function Q. The parameter vector θ contains the weight and bias parameters of that neural network.

The neural network architecture is as follows:

```
QNet(
  (_net): Sequential(
    (0): Linear(in_features=37, out_features=96, bias=True)
    (1): ReLU()
    (2): Linear(in_features=96, out_features=96, bias=True)
    (3): ReLU()
    (4): Linear(in_features=96, out_features=4, bias=True)
  )
)
```

2. TRAINING ANALYSIS

Despite its simplicity, the agent described in this report is able to solve the environment after completing less than 1000 episodes. In fact, in a training session depicted in Figure 1 the agent surpasses the average cumulative score of 13 around episode 900 and reaches the average cumulative reward of almost 16 at the end of the training session.

3. IDEAS FOR FUTURE WORK

The training performance of the agent could be improved by implementing some or all of the “rainbow” improvements summarized in [1]. In particular, the prioritized experience replay is easy to implement and would likely lower the number of training episodes needed to reach the threshold of 13 reward points.

REFERENCES

[1] Matteo Hessel et al. “Rainbow: Combining improvements in deep reinforcement learning”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[2] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836.

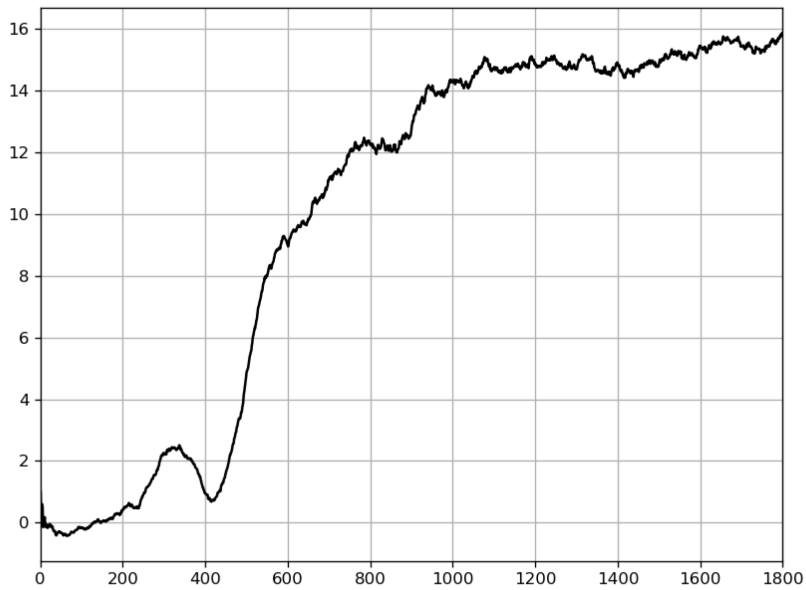

FIGURE 1. Rolling window average of the cumulative reward for each episode in a training session consisting of 1800 episodes. The threshold average cumulative reward of 13 is reached around episode 900.

Email address: `julian.wergieluk@risklab.com`

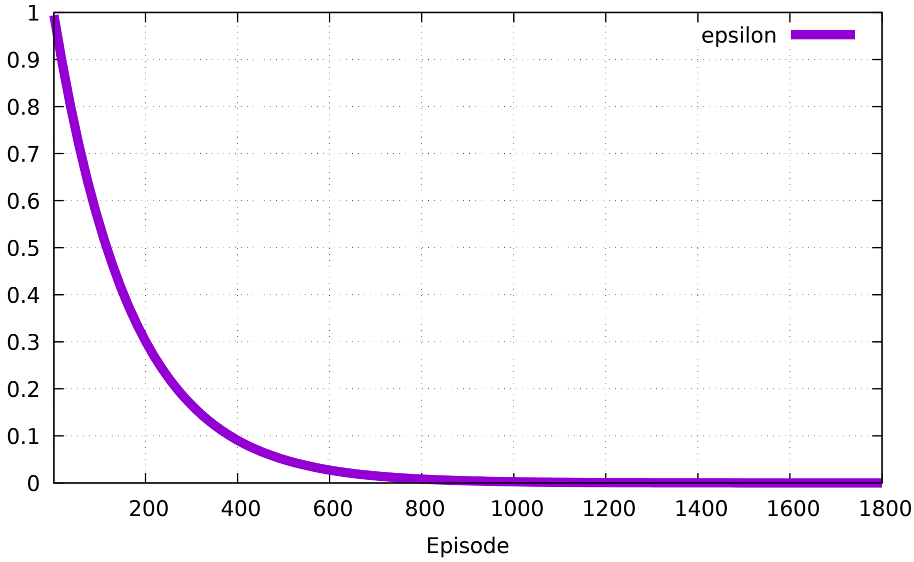

FIGURE 2. Decay of the ε parameter as function the training episode number.