Neural Modelling Exercise 2 - Reinforcement Learning

Noël Kury

1 Theoretical Background

This exercise centers on instrumental conditioning, a form of learning in which the organism's behavior affects the likelihood of the behavior being repeated in the future. It involves learning the consequences of various actions and optimizing the behavior to maximize the cumulative reward. A strong focus hereby is for one the policy evaluation with the goal to determine an approximation of the reward under a given policy and on the other hand the policy improvement which is used to help finding an optimal policy and adjusting the policy to favor actions that lead to higher expected returns. The actor-critic algorithm incorporates both of these concepts and is being implemented in this exercise.

First, we focus on policy iteration which uses temporal difference learning to determine the cumulative future reward $v(u)$ at state u when using a fixed policy $m(u)$. The following procedure describes the prediction update of one state $v(u)$ in a given iteration.

Algorithm 1 (Policy Evaluation)
Require: *state $u \in \{1, 2, ..., n\}$, policy $m(u)$, action a, $r^a(u)$ reward after choosing action a that brings us to state u', learning rate ϵ*
1. *Choosing action a at state u according to policy $m(u)$*
2. $\delta \leftarrow r^a(u) + v(u') - v(u)$
3. $v(u) \leftarrow v(u) + \epsilon\delta$

Employing this prediction update over multiple trials will give us suitable approximations for the prediction $v(u)$ for each state u. This algorithm will later have the role of the critic in the actor-critic model.

The policy improvement refers to the process of optimizing the current policy to increase the expected cumulative reward. The policy $m(u)$ at a given state u describes the likelihood of which action to take at the current state. The particular action value for action b in state u gets denoted $m^b(u)$. The final probability of taking action b of all N_u actions in state u is described by

$$P_u(b) = \frac{\exp(\beta m^b)}{\sum_{i=1}^{N_u} \exp(\beta m^i)}. \tag{1}$$

Here β describes the rate of exploration within the algorithm with a small β corresponding with a more exploratory algorithm. The update of the action values $m(u)$ follows the following procedure.

Algorithm 2 (Policy Improvement)
Require: *state $u \in \{1, 2, ..., n\}$, policy $m(u)$, action a, $r^a(u)$ reward after choosing action a that brings us to state u', actor learning rate ϵ_A, stochasticity parameter β, decay rate ϵ_d*
1. *Choosing action a at state u according to policy $m(u)$*
2. $\delta \leftarrow r^a(u) + v(u') - v(u)$
3. $m^a(u) \leftarrow (1 - \epsilon_d)m^a(u) + \epsilon_A\delta$
4. *For all actions $b \neq a$:* $m^b(u) \leftarrow (1 - \epsilon_d)m^b(u)$

The actor-critic algorithm uses the policy evaluation algorithm (critic) to estimate the expected cumulative reward of each state. These approximations are used by the actor which follows essentially the policy improvement algorithm to update the policy for the purpose of maximizing the total reward.

2 Reproduction of the figures

The algorithms are now applied to a simple task of sequential action choices which is illustrated by the following figure.

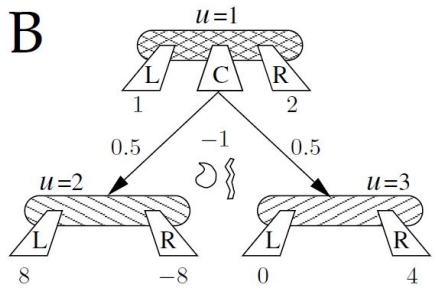


Figure 1: Decision tree consisting of the states $u \in \{1, 2, 3\}$ and multiple decisions in each state leading to different states and rewards.

Applying the policy evaluation to this task over the period of 1000 repetitions of 1000 trials, a learning rate of $\epsilon = 0.2$ and an unbiased policy (s.t. $m(u) = 0\ \forall u$) results in the following results.

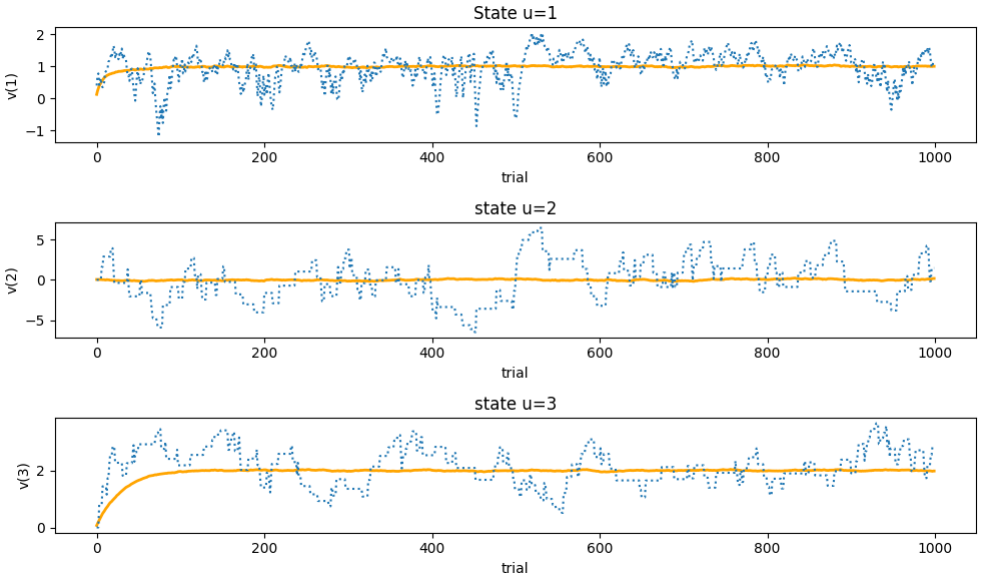


Figure 2

We can see that after the initial guess of no reward at all, the expected reward $v(1)$ averages approximately a reward of 1, $v(2) \to 0$, $v(3) \to 2$. These results can be explained by the unbiased policy that results in the prediction being the average total future reward:

$$v(3) = \frac{1}{2}(0 + 4) = 2$$
$$v(2) = \frac{1}{2}(8 - 8) = 0$$
$$v(3) = \frac{1}{3}\left(1 + 2\left(-1 + \frac{1}{2}\left(v(2) + v(3)\right)\right)\right) = 1$$

Applying the actor-critic model (with hyperparameters $\epsilon = 0.2, \epsilon_A = 0.075, \epsilon_d = 0, \beta = 1$) to the task, now also adjusting the policy for optimal reward leads to the following results.

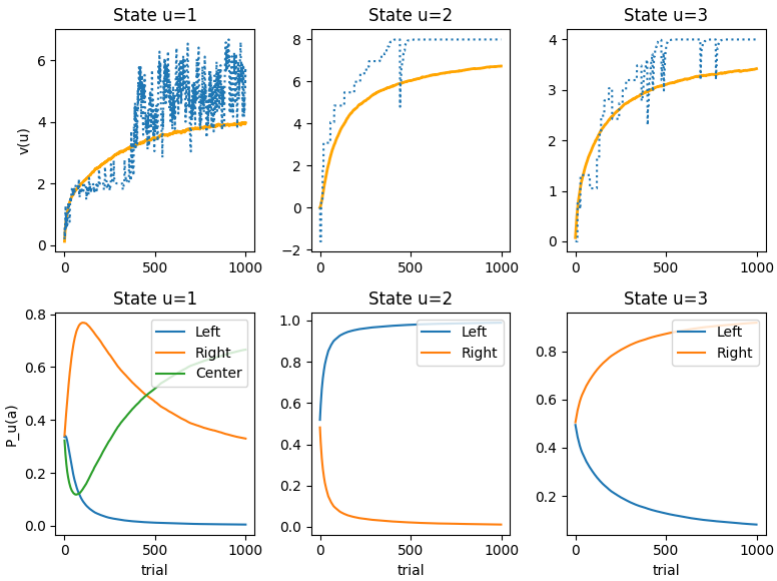


Figure 3

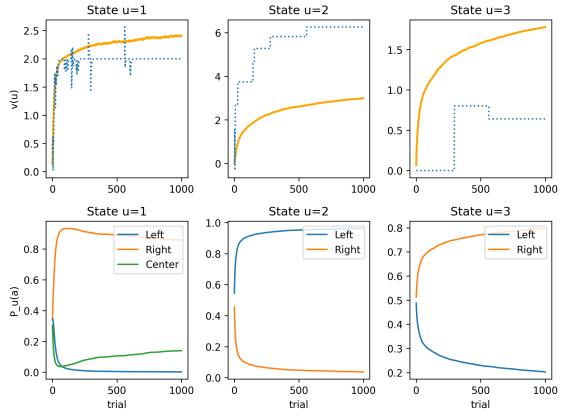
We can see that the algorithm learns the choices with the maximal cumulative future reward. Therefore, we see a monotonous increase in predicted future reward in the first row in all three states. In the second row we can see the algorithm adjusting the action values in a way that maximizes the total future reward. For example in the beginning of the learning trials of state 1 the model learns to avoid the action "Left" and "Center" because they correspond with a smaller immediate reward as the choice "Right" but then investigates the "Center" option over the period of more trials over an extended number of times and learns the ability to make the best subsequent choices in states two and three. These choices lead to a cumulative reward bigger than the immediate reward associated with selecting "Right" in the initial state.

3 Influence of Hyperparameters

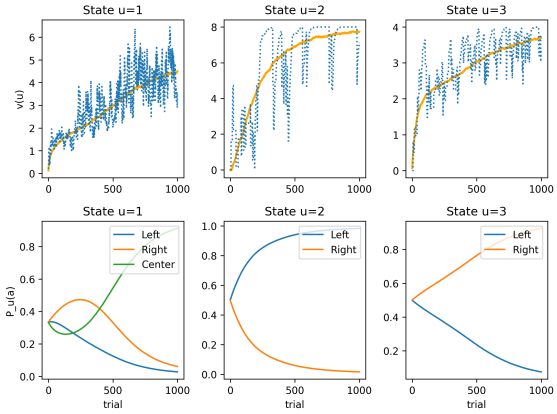
The example from the previous section can also help to illustrate the influence of the hyperparameters of the actor-critic algorithm. The hyperparameters from the previous section act as a reference value and changes in the hyperparameters are noted below the plot. Starting of with the actor learning rate ϵ_A that plays a crucial part in determining to which extend the current action values are updated based on the temporal difference. The update rule

$$m^b(u) \leftarrow (1 - \epsilon_d)m^b(u) + \epsilon_A \delta_{ab} \delta \tag{2}$$

can be seen as a gradient ascent step in the direction of a higher reward. A high ϵ_A results rapid updates and quick adaptations but comes with the risk of instability, whereas a small ϵ_A tends to result in more gradual updates and more stability.



(a) $\epsilon_A = 0.2$

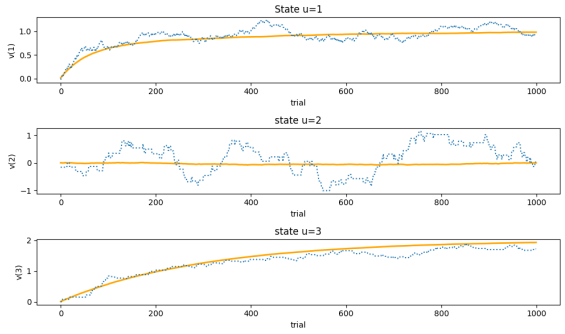


(b) $\epsilon_A = 0.01$

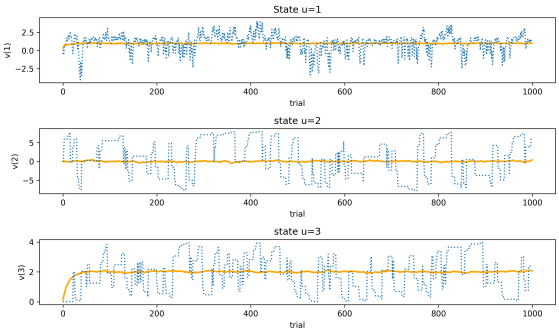
In figure (a) with a big value for ϵ_A, the algorithm fails to learn the optimal action values that bring maximal reward. The reason for that could be that the algorithm is too focused on recent experiences and therefore does not investigate the states two and three sufficiently and stays predominantly with the initial decision "Right" in the state one over course of all 1000 trials. in figure (b) with a small ϵ_A we can see a slower convergence compared to figure 3.

The effect of the hyperparameter ϵ on the critic can best be seen by a comparison with figure 2.

A big ϵ can lead to a faster convergence but brings the risk of high fluctuations within each repetition of a single trial. A small ϵ leads to a smoother fit and avoids big fluctuations, but converges slower.

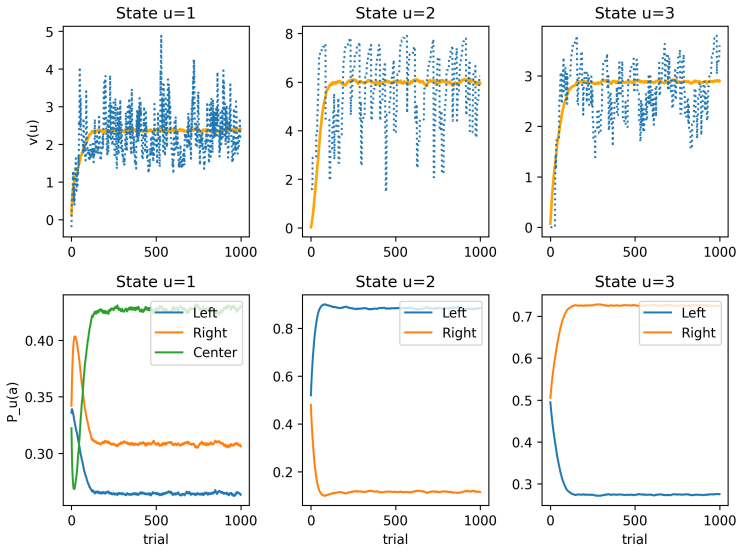


(a) $\epsilon = 0.02$



(b) $\epsilon = 0.5$

In the actor ϵ_d plays a role in determining the weight given to the current action preferences relative to the update term. In the term that can be seen in (1), the current action preferences $m^b(u)$ get multiplied by $(1 - \epsilon_d)$ resulting that we give a small weight to the current action preferences when ϵ_d is big and conversely for a small ϵ_d. This can lead to fast adaptation and the risk of instability or slower adaptation and a smaller risk of instability. When having ϵ_d too big we also risk that we fail to make sufficient learning progress and do not take the past experiences enough into account, which can be seen in the following plot.



(a) $\epsilon_d = 0.1$

Lastly, there also exists the hyperparameter β that controls the rate of exploration within the algorithm. As it can be seen in (1), because of the exponential growth in the nominator, a large β results in the probabilities being more focused on the largest action value. This makes the algorithm more deterministic and less exploratory. On the other side with a small β the algorithm becomes more uniform, less sensitive to the differences in the action values and therefore more exploratory.