Practical 1 - Reinforcement Learning

Multi-agent learning 2017-2018

Sven Luehof (4235991) & Michiel van Heusden (4173309)

December 8, 2017

Problem 1

a) The main advantage of using a learning parameter instead of averaging rewards is that the algorithm will continue learning from new rewards. In a dynamic environment, when the rewards to certain actions can change, this is an important property. Using a learning rate comes at the cost of the importance of the old rewards more quickly. A more important disadvantage is that an appropriate learning parameter α has to be chosen for each implementation.

When using averaging the effect of a new value will become less important at a rate depending on the amount of previous rewards obtained. The overall effect of a new value v when $n-1$ rewards have been obtained is:

$$effect = v \cdot \frac{1}{n}$$

When n becomes large this effect will turn negligible:

$$\lim_{n \to \infty} \frac{1}{n} = 0$$

Thus after a while the algorithm will stop learning; when the reward values of a certain action change the algorithm will no longer receive an optimal payoff.

Using a learning parameter this problem does not exist, as the weight of each new value v is constant:

$$effect = v \cdot \alpha$$

The rate at which old values become less relevant is much higher. The effect of a value v after n steps will only be:

$$effect = v \cdot (1 - \alpha)^{n-1}\alpha$$

This effect becomes irrelevant very quickly; after only nine new actions the effect is:

$$effect = v \cdot (1 - \alpha)^{9}\alpha$$

Assuming $\alpha = 0.10$ this means the effect is $v \cdot 0.90^9 \cdot 0.10 = v \cdot 0.039$, whereas the effect of a value when averaging at the same time is $effect = v \cdot \frac{1}{n} = v \cdot 0.10$. After only ten actions the importance of the first value is 2.6 times higher when averaging.

The speed at which old values become irrelevant depends on the value α. As a result one needs to carefully choose an appropriate value for α for the task at hand.

In order to show the effect of α on the speed at which a strategy emerges when using a learning parameter an experiment has been designed. This was done using the program provided by the University of Utrecht, written by Maaike Burghorn and Wouter van 't Hof[1]. This program implements reinforcement learning agents, with their own internal expected payoffs per action. The algorithm used in this experiment was the optimistic initial values algorithm. The other parameters were `nrAgents` = 10, `nrMachines` = 3, `initialValues` = 5. The `alpha` parameter varied (`alpha` = 0.05, 0.10, 0.25, 0.50, 0.75, 1.00). Another series of tests were run as well, using the same parameters but instead of using a learning rate the rewards were averaged.

During this experiment a strategy was said to have been converged when the action distribution of the agents has converged; meaning the number of agents per slot machines has remained the same for at least 50 ticks. Each value α, and the averaging algorithm, was tested 100 times. The complete test results can be found in the additional files, which will be provided upon request[2]. The means and averages of the ticks until convergence can be found in table 1.

The results of this tests can be found in table 1. From this table is can be seen that α has a clear effect on the rate of convergence. When

[1]The framework can be downloaded at: .

[2]Send an email to m.p.a.vanheusden@uu.nl

Algorithm	$\alpha = 0.05$	$\alpha = 0.10$	$\alpha = 0.25$	$\alpha = 0.50$	$\alpha = 0.75$	$\alpha = 1.00$	Averaging
mean	546.5	312.7	144.2	102.5	78.20	53.50	69.93
σ^2	154.1	86.34	55.65	23.18	35.51	35.12	69.98

Table 1: The amount of ticks required until the system converged; parameters: nrAgents = 10, nrMachines = 3, initialValues = 5.

compared to averaging rewards it can be seen in figure 1, that a lower α converges slower than when averaging, whilst having a higher α it converges quicker.

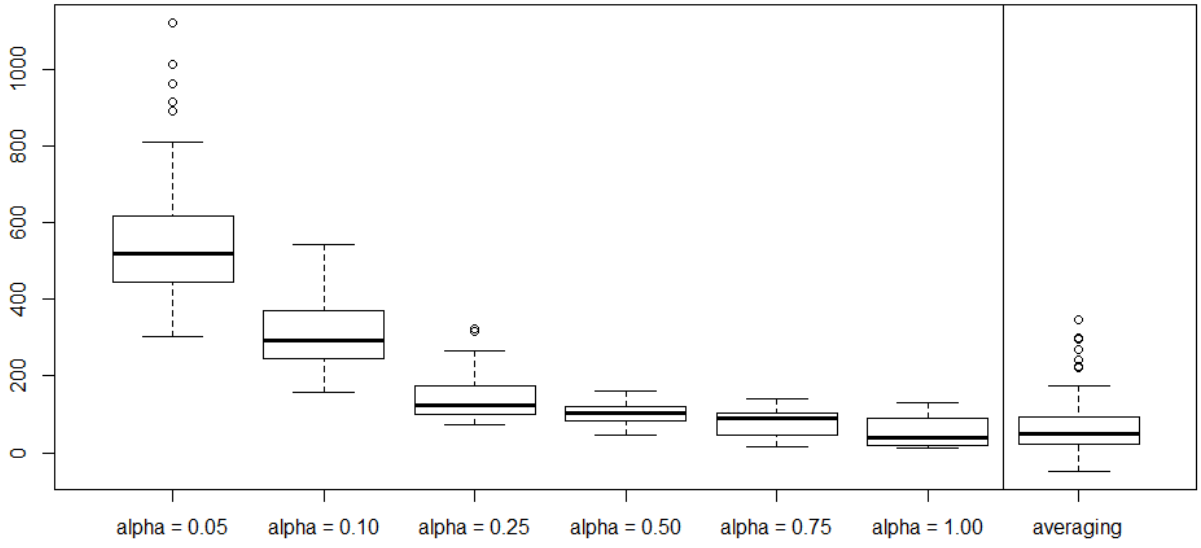

Figure 1: Box plots of the effects of a differing learning value, compared to averaging the rewards

Problem 2

a) Optimistic starting values is a greedy search algorithm; it selects the action with the highest expected pay-off. Due to the nature of greedy

algorithms one can assume that after some time a strategy will converge. This strategy is to choose the slot machine with the highest expected reward. As shown in question the expected rewards per slot machine will eventually converge. Therefore one can assume that each agent will eventually have one machine with a higher payoff than the others and thus will always be playing this machine.

However, the starting values will play part in both speed an quality of the converged strategy. Whenever the starting values are lower than the true expected value of all machines, each agent will pick a machine at random, and assign a higher expected value to it. The values of the other machines will still be lower. Thus will the first selected machine be the one with the highest reward.

If the starting value is higher than the true lowest expected value, the action distribution will always converge as well. Due to the fact that the initial value is the same for each machine, a machine will be chosen at random. If the chosen machine is the one with the lowest true expected value, its expected payoff will become lower than the initial values. This lowers its expected payoff below that of the other machines. If the first machine was not the one with the lowest true expected value, the agent will assign a new higher value to it. Thus the agent will only play the initially chosen machine, thus a strategy converges.

If the initial value is higher than the highest true expected value of all machines, each machine will be tested. The algorithm will choose a random machine and assign a value lower than the initial value to it. The next tick a different machine will be chosen, as it has a higher expected payoff. This process is repeated until the expected values of the machines converge and a strategy emerges.

As more machines have to be visited before a strategy converges, it will take longer for convergence. This allows one to assume that a lower initial value corresponds with a shorter time action distribution converges. In order to prove this an experiment, similar to the one used before, was created. The same program was used and the optimistic starting value algorithm was chosen again. The other parameters were: `nrAgents` = 10, `nrMachines` = 3, `alpha` = 0.10. This time around the initial values varied (`initial values` = 0, 1, 2, 5, 10). The test was

Initial Values	0	1	2	5	10
Mean	2.150	250.7	271.1	293.7	316.1
Standard Deviation	0.592	87.55	91.43	80.87	90.99

Table 2: The amount of ticks required until the system converged; parameters: nrAgents = 10, nrMachines = 3, $\alpha = 0.10$.

conducted 100 times per initial value.

The results were as expected: the tests with an initial value lower or equal to the lowest true expected payoff (0) converged instantly. The tests with a starting value that was lower than the highest, yet higher than the lowest true expected payoff, converged much slower. However, when the initial value was higher or equal than the highest true expected payoff it did not take much longer for a strategy to emerge. The required amount of ticks can be seen in 2. When the starting value became greater than 1 the increase in convergence time was not significant, as can seen in 2. The full test data will be given on request [3].

b) Two problems that are associated with optimistic initial values are as follows: using optimistic initial values a strategy will not always converge, a second problem is that the algorithm will not always chose an optimal strategy. Both of these problems can be proven informally.

The proof of the first one is quite straightforward. Assume a multi-armed bandit problem, with an infinite amount of slot machines. Each of these machines has a bounded expected payoff value ($\forall i : 0 < E[m_i] < p \in \mathbb{R}$). If the initially assigned value is larger than p, one can expect each slot machine will be visited once before a strategy converges. Thus, since there is an infinite amount of slot machines, it takes infinite steps before the strategy converges. Therefore optimistic initial values does not necessarily converge.

The second problem, getting stuck in a non-optimal strategy, is attributed to the lack of exploration. The optimistic initial values algorithm is an example of an exploitative algorithm; it does not explore, but only exploits the best option. This allows an agent using this algorithm to be stuck in a local optimal strategy instead of a global optimal

[3] Send an email to m.p.a.vanheusden@uu.nl

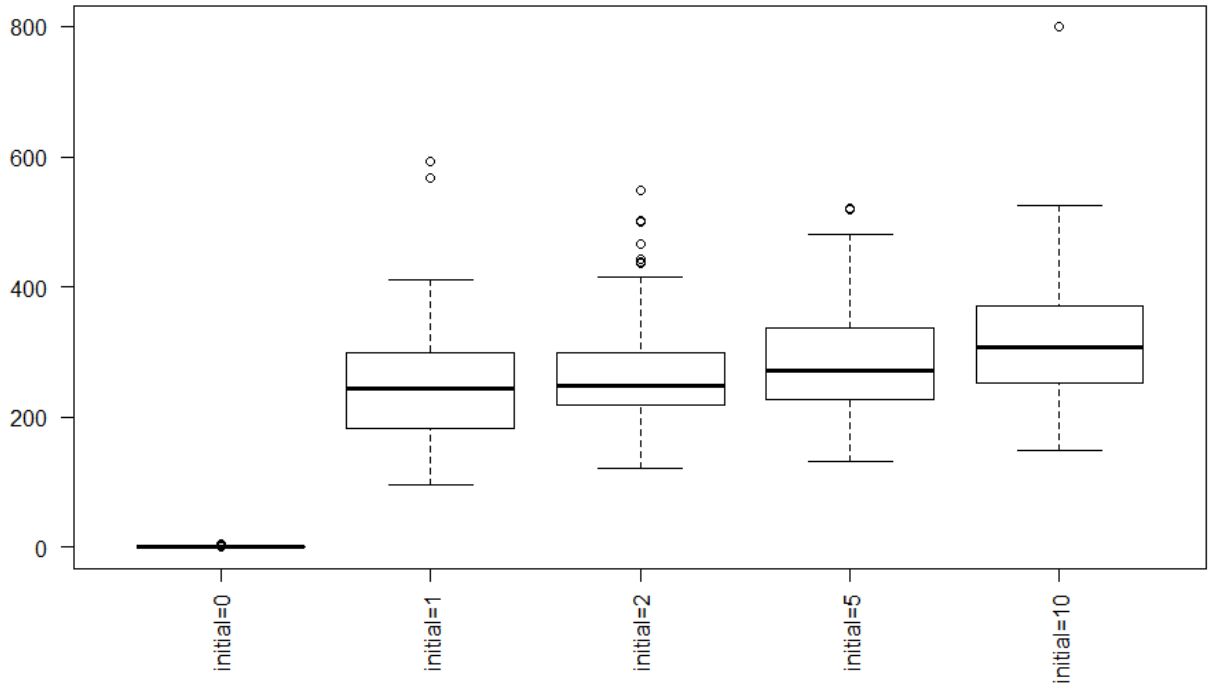

Figure 2: Box plots of the effects of a differing initial value on the rate of convergence.

strategy. By using the optimistic initial values algorithm, with the parameters set to nrAgents $= 10$, nrMachines ≥ 3, $\alpha = 0.10$, initial values $= 0$, a good example is generated. In the first step a random slot machine is chosen, as all the expected values are the same. If the agent chooses a machine with a higher true expected payoff than its initial value, this machine will have a higher expected payoff than the rest. Which results in the agent only choosing said slot machine, getting stuck in a sub-optimal strategy.

Problem 3

a) ϵ-greedy is an algorithm that has a 'willingness to explore' parameter ϵ. This parameter indicates the chance for the algorithm to choose to explore a random action. Leaving a $1 - \epsilon$ chance to take the best currently known action. As this is a non-deterministic algorithm, one can assume that the action distribution never converges.

In order to proof that the action distribution for ϵ-greedy never converged we once again conducted tests. The parameters for the epsilon-greedy algorithm were $\alpha = 0.10$, nrAgents = 10, nrMachines = 3, initialValues = 5, and the ϵ varied (ϵ = 0.05, 0.10, 0.25, 0.50, 0.75, 1.00). The results of these tests were conclusive and very decisive: not a single run truly converged.

b) The main disadvantage of a high *exploration* rate is that the agent will often choose a random action. This may result in finding a new optimal action, however, it may also result in obtaining a lower achieved reward, than if the agent had chosen the optimal slot machine. The negative effects of having a high exploration rate are very similar to the negative effects of having a high exploitation result.

Having a high *exploitation* rate has a different disadvantage, but the result is the same: the agent will receive a lower than optimal payoff. When an agent has too high an exploitation rate it will never explore, always choosing one of the first actions. This has as result that many actions go unexplored, some of which may be better than what the agent is perceiving as optimal.

A possible solution for this problem is lowering the exploration rate as the algorithm goes through its cycles. At the start of the run one wants the agents to have a relatively high exploration rate, to enable them to find the optimal slot machine. When it has become likely that an optimal strategy has been found, the exploration rate should be decreased. Decreasing the exploration rate ensures that the agents will choose the optimal action more often. It might even be useful to eventually let the exploration rate become 0. This way a strategy converges in which the agents will choose an action with the highest expected reward.

Problem 4

a) The softmax algorithm is an algorithm that explores. However, unlike ϵ-greedy, it explores depending on the perceived worth of actions. The chance for each possible action is determined by the Gibbs (or: Boltzmann) distribution. This function uses a temperature parameter τ to influence the weights. With a larger temperature value, there is a larger chance to explore, as can be seen in equation 1. Here an example is shown where the actions have the following values: $A_1 = 1, A_2 = 2$ and $\tau_1 = 0.1$, $\tau_2 = 0.2$.

$$\frac{e^{A_1/\tau_1}}{e^{A_1/\tau_1} + e^{A_2/\tau_1}} = \frac{e^{10}}{e^{10} + e^{20}} < \frac{e^{A_1/\tau_2}}{e^{A_1/\tau_2} + e^{A_2/\tau_2}} = \frac{e^{5}}{e^{5} + e^{10}} \tag{1}$$

Since this is the case one can argue that, despite softmax being an exploratory algorithm, it can converge with very low values of τ. In order to proof this another experiment was run. The softmax algorithm was used with parameters `nrAgents` = 10, `nrMachines` = 3, `alpha` = 0.10. The τ value varied (`tau` = 0.05, 0.10, 0.25, 0.50, 0.75, 0.10).

However, despite having a very low τ none of the runs truly converged. For low τ values, however, there were clear bands ordered by the true expected payoff of the slot machines. With higher values of τ this effect disappeared.

b) The softmax algorithm, when exploring, chooses options with a higher expected reward more often than the ϵ-greedy algorithm, because softmax assigns higher probabilities to options yielding a higher expected reward, unlike ϵ-greedy. Scenarios in which softmax action selection would perform better than ϵ-greedy is any scenario in which the non-optimal slot machines have varying expected reward and the chance the softmax algorithm chooses the optimal slot machine is at least as high as in the ϵ-greedy algorithm. In general, in such a scenario the softmax algorithm would achieve a higher final accumulated reward, because it chose the slot machine with higher expected reward more often.

Problem 5

When the rewards of the slot machines are no longer stationary, it means that there also no longer is a single stationary optimal slot machine. Therefore it becomes necessary to increase the willingness to explore and/or learning rate to allow the algorithm to find out whether or not another slot machine has gotten the highest expected reward.

To cope with these changing rewards α should be increased, so that the agent can adapt faster to new rewards. The same principle applies to ϵ, in hte ϵ-greedy algorithm the willingness to explore parameter ϵ should be increased, so the agent will keep on trying different slot machines. In order to increase the exploration in the softmax algorihtm, the temperature parameter τ should be increased as well. This way options with a lower expected reward will be chosen more often. Finally, the initial values do not require any change. The initial values should remain high, so the agents will still explore the slot machines. However, as long as these values stay high, changing these values will not significantly influence the agent‘s action selection.