VRIJE UNIVERSITEIT BRUSSEL

Computational Game Theory
Assignment 3

Deadline: 16 December 2019

Submission Instructions

- We have organized a Github classroom for this assignment. Use this link `https://classroom.github.com/a/88GrpFBJ` to accept the assignment and to create inside the classroom your own private repository. Commit there your code and report (in PDF form), before the deadline.
- Follow this naming convention for your PDF report: *Lastname_Firstname_studentNo_affiliation.pdf*.
- Late submissions will be penalized (-1 point/day).
- This is an **individual** assignment, sharing code is considered plagiarism and will be punished. Any other online sources used should be declared in the report. Plagiarism will not be tolerated.
- If you have any questions send an email to: `roxana@ai.vub.ac.be`.

1 N-Armed Bandit

For this exercise you will have to implement the N-Armed bandit problem using each of the following action selection methods:

- Random
- ϵ-Greedy with parameter ϵ
- Softmax with parameter τ

All Q_{a_i} are initialized to 0. The rewards are subject to noise according to a normal distribution with mean $Q^*_{a_i}$ and standard deviation σ_i, $\quad \forall i = 1, \ldots, N$.

You are provided with 5 instances of reward distributions (Tables 1-5) for the N-Armed bandit problem. You only have to select **one** of the tables, according to the following criteria:

- Identify the last number of your VUB/ULB enrolment number
- If the number is 0 or 9, select Table 1
- If the number is 1 or 8, select Table 2
- If the number is 2 or 7, select Table 3
- If the number is 3 or 6, select Table 4
- If the number is 4 or 5, select Table 5

Let $N = 4$, $Q^*_{a_i}$ and σ_i as defined in your selected table.

Action	$Q^*_{a_i}$	σ_i
action #1	2.3	0.9
action #2	2.1	0.6
action #3	1.5	0.4
action #4	1.3	2

Table 1: Last number is 0 or 9.

Action	$Q^*_{a_i}$	σ_i
action #1	1.2	1.1
action #2	0.4	0.6
action #3	1	0.8
action #4	0.3	2

Table 2: Last number is 1 or 8.

Action	$Q^*_{a_i}$	σ_i
action #1	1.3	0.9
action #2	1.1	0.6
action #3	0.5	0.4
action #4	0.3	2

Table 3: Last number is 2 or 7.

Action	$Q^*_{a_i}$	σ_i
action #1	2.4	0.9
action #2	1.3	2
action #3	1	0.4
action #4	2.2	0.6

Table 4: Last number is 3 or 6.

Action	$Q^*_{a_i}$	σ_i
action #1	2.1	1.2
action #2	1.1	0.8
action #3	0.7	2
action #4	1.9	0.9

Table 5: Last number is 4 or 5.

Plotting

For each of the following exercises, run your simulation for 1000 time steps and plot the following graphs:

- One combined plot of the cumulative average reward *over time* for each algorithm (One graph).
- One plot per arm showing the $Q^*_{a_i}$ of that action along with the *actual* Q_{a_i} estimate *over time* (all action selection methods should be depicted on the same plot!) (One graph per arm).
- For each algorithm, plot a histogram showing the number of times each action is selected (One graph per action selection strategy).

Do **NOT** add any other graphs than the ones specified here.
See Section 2.2 of "Reinforcement Learning, An Introduction" by Sutton and Barto for a more comprehensive explanation of the problem[1].

Remarks:

- You can run each experiment (i.e., the 1000 time steps for each action selection method) for multiple trials and then average over the results (plotting the obtained mean together with an error measurement, such as standard deviation or standard error).
- **Provide comments and explanations for each of the obtained plots/results! Provide answers to all the questions explicitly asked in the following exercises!**

Exercise 1.1

After selecting the appropriate table, run the following algorithms and provide the requested plots:

- Random
- ϵ-Greedy with parameter $\epsilon = 0$
- ϵ-Greedy with parameter $\epsilon = 0.1$
- ϵ-Greedy with parameter $\epsilon = 0.2$
- Softmax with parameter $\tau = 1$
- Softmax with parameter $\tau = 0.1$

Comment on the results. Which algorithms arrive close to the best arm after 1000 iterations? Which one arrives there faster? Explain your findings.

Exercise 1.2

Re-run Exercise 1 doubling the standard deviations of each arm, and comment on the plots and results. **Discuss the results.** Is this problem harder or easier to learn? Does the performance of the algorithms change significantly? Which of the above performs best now?

[1] http://incompleteideas.net/book/RLbook2018.pdf

Exercise 1.3

Re-run Exercise 1 with two additional algorithms, and plot their results. The new algorithms have a time-varying parameter, which depends on the iteration number $t = 1, \ldots, 1000$ as follows:

- ϵ-Greedy with parameter $\epsilon(t) = 1/\sqrt{t}$
- Softmax with parameter $\tau = 4 * \frac{1000-t}{1000}$

Discuss the results. Are these algorithms better than the ones with a fixed parameter?

2 Stochastic Reward Game

	b_1	b_2	b_3
a_1	$\mathcal{N}(11, \sigma_0^2)$	$\mathcal{N}(-30, \sigma^2)$	$\mathcal{N}(0, \sigma^2)$
a_2	$\mathcal{N}(-30, \sigma^2)$	$\mathcal{N}(7, \sigma_1^2)$	$\mathcal{N}(6, \sigma^2)$
a_3	$\mathcal{N}(0, \sigma^2)$	$\mathcal{N}(0, \sigma^2)$	$\mathcal{N}(5, \sigma^2)$

Table 6: Stochastic Climbing Game

Table 6 shows the stochastic climbing game. It is a cooperative game, in which two agents must learn to maximize the score they receive from the game. Each has three available actions, and the rewards given to both agents depend on the actions of both agents, and are stochastic. After every interaction, for the chosen combination of actions, the reward received by both agents is sampled from a normal distribution with the mean presented in the above table and a standard deviation of σ_0 (for the joint action $< a_1, b_1 >$), σ_1 (for the joint action $< a_2, b_2 >$) or σ (for all the other possible joint actions). Implement two types of Joint-Action learners in the above problem. The first type with simple Boltzmann action selection with a fixed temperature of your choice. The second type with a more advanced multi-agent action selection heuristic of your choice (optimistic boltzmann, FMQ, ...). Recall that in Joint-Action learning, you learn about the quality of joint actions (9 in total in this setting), and you often have to maintain beliefs about the other agents strategy. An explanation of these concepts can be found here: [1, 2]. Further useful references are [3, 4, 5, 6].

2.1 Plotting

Consider the following three cases:

(a) $\sigma_0 = \sigma_1 = \sigma = 0.2$

(b) $\sigma_0 = 4$ and $\sigma_1 = \sigma = 0.1$

(c) $\sigma_1 = 4$ and $\sigma_0 = \sigma = 0.1$

For each case, play two agents of the first type for 5000 episodes and do the same for two agents of the second type. After learning, plot the following on the same figure, for each of the three cases:

- Average collected reward per episode versus the episode number for each of the two types (you are encouraged to run each experiment for at least 50 trials and average over the results).

You should have three plots (one for each case), each with two lines (one for each type). Using these plots, you can compare the two algorithms and decide which of the two types learns best in each setting. You can smooth the results using a moving average for readability. **Discuss your results!**

2.2 Discussion

Discuss the following:

1. How will the learning process change if we make the agents independent learners?
2. How will the learning process change if we make the agents always select the action that according to them will yield the highest reward (assuming the other agent plays the best response)?

NOTE: The choice of the programming language is free. Save yourself work by creating an abstraction for the number of actions and all parameters, so you can re-use your code. **Do not forget to always explain your results!**

References

[1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, (s 746):752, 1998.

[2] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In *Adaptive Agents and Multi-Agent Systems II*, pages 119–131. Springer, 2005.

[3] S. Kapetanakis, D. Kudenko, and M. Strens. Learning of coordination in cooperative multi-agent systems using commitment sequences. *Artificial Intelligence and the Simulation of Behavior*, 1(5), 2004.

[4] A. Nowé, P. Vrancx, Y. De Hauwere, M. Wiering, and M. van Otterlo. Reinforcement learning: state-of-the-art. *Game Theory and Multi-agent Reinforcement Learning*, pages 441–470, 2012.

[5] K. Verbeeck, A. Nowé, J. Parent, and K. Tuyls. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Autonomous Agents and Multi-Agent Systems*, 14(3):239–269, 2007.

[6] K. Verbeeck, A. Nowé, M. Peeters, and K. Tuyls. Multi-agent reinforcement learning in stochastic single and multi-stage games. In *Adaptive Agents and Multi-Agent Systems II*, pages 275–294. Springer, 2005.