Agents in a Multiplayer Snake Environment

Anonymous Authors

***Abstract*—A short summary of your project. You should change also the title, but do *not* enter any author names or anything that unnecessarily identifies any of the authors. It is suggested you use a similar structure (sections, etc.) as demonstrated in this document, but you can make the section headings more descriptive if you wish. Of course *you should delete all the text in this template and write your own*! – this text simply provides detailed instructions/hints on how to proceed.**

I. Introduction

Describe what you did. Provide access to your anonymized code[1].

Note that results should be reproducible using the technologies from the labs (i.e., Python, and selecting among Scikit-Learn, OpenAI Gym, TensorFlow, PyGame, ...).

Do not change the formatting (columns, margins, etc). Hint: shared tools like `http://sharelatex.com/` and `http://overleaf.com/` are great tools for collaborating on a multi-author report in latex. If you wish to use Word, base it on the IEEE template[2] and convert to `pdf` for submission.

II. Background and Related Work

A. Reinforcement Learning

1) Learning: **Q-learning algorithm** [1] [2] [5] [3]

$$Q^{t+1}(s_t, a_t) = (1-\alpha)Q^t(s_t, a_t) + \alpha(r_t + \gamma \max_b Q^t(s_{t+1}, b))$$

SARSA algorithm [1] [2] [5] [4]

$$Q^{t+1}(s_t, a_t) = (1-\alpha)Q^t(s_t, a_t) + \alpha(r_t + \gamma Q^t(s_{t+1}, a_{t+1}))$$

2) Exploitation vs. Exploration: **ϵ-greedy exploration** [1] [2] [5] With probability ϵ, the agent chooses a random action to explore and learn the consequences.

Selection with softmax operator [5]

$$P\left(a_t = a\right) = \frac{\exp\left(Q^t(s_t, a)/\tau\right)}{\sum_b \exp\left(Q^t(s_t, b)\right)/\tau}$$

Elaborate (in your own words) the background material required to understand your work. It should cover a subset of the topics touched upon in the course. You are encouraged to cite topics in lectures, e.g., structured output prediction in , book chapters, e.g., Chapter 9 from , or articles from the literature, e.g., . Basically, you should prepare the reader to understand what you are about to present in the following sections. Eq. (1) shows a random equation.

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}\in\{0,1\}} p(\mathbf{y}|\mathbf{x}) \tag{1}$$

III. The Environment

Describe your environment, either one you adapted/borrowed from somewhere, or designed yourself. Convince the reader that it is an interesting and/or challenging environment (could it potentially have real-world use or is based on real-world data? Or simply to provide an interesting/fun/challenging problem to tackle. In particular you should outline the particular challenges it poses as a RL problem.

IV. The Agent

We designed two different agents for this environment to compare different strategies. The *Reinforcement Learning* (*RL*) agent is model-free, whereas the *Minimax* agent is based on the exploration of the different possibilities.

A. The Reinforcement Learning Agent

This model-free agent uses either SARSA or Q-Learning to learn the model and the consequences of its actions. It also uses either ϵ-greedy exploration or the Softmax method to tackle the dilemma between exploitation and exploration.

To perceive its environment without having too many states to learn, we use 12 inputs. When the input is bounded, if the real value is greater than this or doesn't exist, it is assigned the greatest value.

1) 0 if the head of the snake touches the right wall, 1 otherwise.
2) 0 if the head of the snake touches the upper wall, 1 otherwise.
3) 0 if the head of the snake touches the left wall, 1 otherwise.
4) 0 if the head of the snake touches the lower wall, 1 otherwise.
5) The distance between the square located to the right of the head and the closest candy (between 0 and 7).
6) The distance between the square located to the top of the head and the closest candy (between 0 and 7).
7) The distance between the square located to the left of the head and the closest candy (between 0 and 7).
8) The distance between the square located to the bottom of the head and the closest candy (between 0 and 7).
9) The distance between the square located to the right of the head and the closest square of another snake (between 0 and 3).
10) The distance between the square located to the top of the head and the closest square of another snake (between 0 and 3).
11) The distance between the square located to the left of the head and the closest square of another snake (between 0 and 3).

[1] Our code is available here: http://anonymouslinktoyourcode.zip

[2] https://www.ieee.org/publications_standards/publications/conferences/2014_04_msw_a4_format.doc

12) The distance between the square located to the bottom of the head and the closest square of another snake (between 0 and 3).

These inputs allows the snake to have a local vision of the candies, a local but smaller vision of the other snakes, and feel when it touches a wall.

We also give rewards after each action done by the snake.

- 1 if the snake ate a candy.
- -10 if the snake died.
- 0 otherwise.

The agent will easily learn to look for candies and to avoid walls. It will also learn to avoid other snakes.

Finally, we use the following parameters:

- $\epsilon = 0.1$ in the case of ϵ-greedy exploration.
- $\tau = 0.1$ in the case of Softmax exploration.
- $\alpha = 0.2$. A learning rate so high is strong enough because the inputs and the best actions to take are highly correlated.
- $\gamma = 0.9$. We don't use 1 to make it clear that the best path is always the shortest, but it is near 1 since we care about the future almost as much as the present.

B. *The Minimax Agent*

V. Results and Discussion

To measure how well the two agents perform in our environment, we use several measurements. We introduce the *Random* agent, which chooses at every step a random direction. We simulate the games on a grid of size 30, with 10 candies on the map and the two adversarial agents we want to compare.

A. *Performance of your Agent in your Environment*

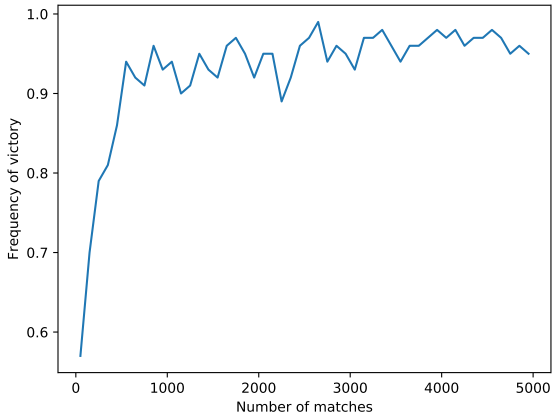

Fig. 1. The learning curve of the RL agent against a random one.

1) Performance of Learning: To analyze the learning time of the *RL* agent, we simulate it against the *Random* agent. At the beginning, the *RL* agent knows nothing. We run 5000 games, and we compute the frequency of victories for the *RL* agent for every chunk of 100 games. The results are presented on Figure 1. As we can see, with only a few hundreds games, this agent is far better than the *Random* one.

2) Performance of the agents: We simulated 1000 games between each pair of agents. The results are presented in Table I. When the sum of the scores isn't equal to 1000, it means that games ended because both agents died at the same time.

As can be seen, the *Minimax* agent is nearly unbeatable, no agent scored a point against it. However, it scored 933 against the *Random* agent, which means that it was killed 77 (while killing the other one). Even though the *RL* dies 4 times out of 1000 without killing the *Random* agent, it performs better with respect to its own score. In 991 out of 1000 games, it manages to stay alive while killing its opponent, which means it learned from experience to avoid going too close to its opponent, whereas the $emphMinimax$ agent thinks its opponent will take the best action. This hypothesis is clearly false for the *Random* agent.

TABLE I
Results of the games.

Game	**Random**	**Minimax**
RL	991 - 4	0 - 174
Minimax	933 - 0	

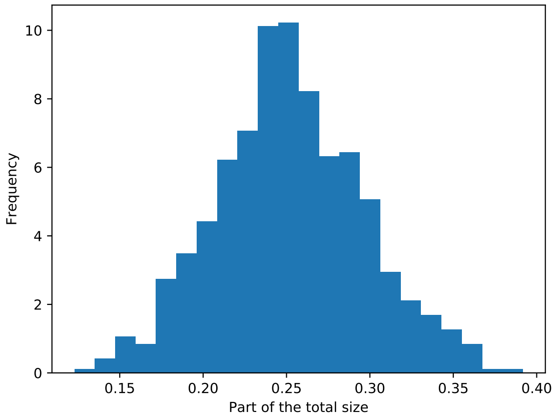

Fig. 2. Distribution of the proportion of the size of the *RL* agent for the 776 games ended after 1000 iterations.

B. *Performance of your Agent in the ALife Environment*

We only tried to deploy our *RL* agent in the ALife Environment since it is model-free. Deploying the *Minimax* agent would have require to create a totally new model, which would have made it a different agent. When using the same technics (Q-Learning or SARSA, ϵ-greedy or Softmax exploration), the results aren't as satisfactory as in the Snake world. This environment is much more complicated, and the inputs and outputs are analogous.

We chose to tackle these issues by discretizing the inputs. We don't use the energy input. We multiply each other input (the color sensors) by 5, and round them to the lower integer. To keep the number of actions small, we only allow rotations of $-\frac{\pi}{2}$, 0, $\frac{\pi}{2}$ and pi.

After a few minutes of simulation, it seems that the herbivore have understood that they have to eat the plants. Unfortunately, this timeframe doesn't seem sufficient enough for the insects to learn other lessons.

VI. Conclusion and Future Work

This section summarizes the paper: Your environment and agent, its strength and its weaknesses. Also remark about what would be the next steps you would take if you or someone else were to continue/extend this project. Note that for the initial submission you are limited strictly to 4 pages (double column), *not including references*. An extra page will be allowed for final submission (after the initial reviews).

References

[1] N. Tziortziotis. Lecture IV - Introduction to Reinforcement Learning. *INF581 Advanced Topics in Artificial Intelligence*, 2018.

[2] N. Tziortziotis. Lecture V, part II - Approximate and Bayesian Reinforcement Learning. *INF581 Advanced Topics in Artificial Intelligence*, 2018.

[3] Watkins, Christopher & Dayan, Peter. (1992). Technical Note: Q-Learning. Machine Learning. 8. 279-292. 10.1007/BF00992698.

[4] A. Rummery, G & Niranjan, Mahesan. (1994). On-Line Q-Learning Using Connectionist Systems. Technical Report CUED/F-INFENG/TR 166.

[5] Sutton, R. S. & Barto, A. G. 1998 Reinforcement learning: an introduction. Cambridge, MA: MIT Press.