Bachelor Thesis Review

Faculty of Mathematics and Physics, Charles University

Thesis author	Una Adilović
Thesis title	Playing a 3D Tunnel Game Using Reinforcement Learning
Year submitted	2023
Study program	Computer Science
Specialization	Artificial Intelligence
Review author	Milan Straka — Reviewer
Department	Institute of Formal and Applied Linguistics

Overall

	good	OK	poor	insufficient
Assignment difficulty	X	X		
Assignment fulfilled		X		
Total size *... text and code, overall workload*	X	X		

The thesis presents a 3D tunnel game designed by the author (Chapter 1) and implemented in Godot (the implementation is described in Chapter 2); I was able to compile and run the game without any problems.

However, the main focus of the work is to implement and evaluate agents learning how to play the game using tabular reinforcement learning. The thesis describes basic tabular reinforcement learning algorithms (Chapter 4), the state used to represent the game and the overall experimental settings (Chapter 3), the agent implementation (Chapter 5), and finally detailed experiments. Based on the user documentation, I managed to train several agents easily.

The thesis text is written in English and is of good quality. The scale of the thesis, especially the experimental part, is more than sufficient, and the author demonstrated independent, experimental work in the area of reinforcement learning.

I recommend the thesis to be defended.

Remarks and Suggestions

- The states do not include speed, Hans battery level, whether Hans is sick, etc. That makes the environment only partially observable, and it creates various problems (for example, as mentioned in Section 7.7, some policy might be optimal for low speed only). Some discussion and/or experiments about what to include in the state might be useful.
- From the text, I was not sure how large is the variance of different runs of the same experiment (but my feeling was that it can be large). For example, it might be interesting to generate the graphs from Figure 7.1 three times (each time with different 5 random seeds) and plot the results – then the reader could see if, for example, the relative ordering of the methods is always the same or not.

(continues on the next page)

- Page 21, is there any reason why is the first-visit variant of Monte Carlo used? In my experience, the every-visit tends to work better (of course, you would need to handle identical consecutive states before the same obstacle). Consider for example getting twice into contact with a rotavirus – if the second contact happens in the same state as the first, it will be ignored (note that if being sick would be included in the state, it would be different).
- Connected to the previous point, while not running the policy on a repeated state is sound for evaluation, I am not sure it works well during training. First, the ε-greedy policy should be used (so you could perform different actions in the same state), you should discount rewards obtained in the individual steps individually, and usually one performs one update for every time step (if every-visit is used). Compare this to the "oscillation" effect described in Chapter 7.7 – there you perform policy prediction very frequently, compared to a case when a "forward" action is selected.
- There are two reasons why the experiments with discounting do not work. First, discounting of 0.85 is very large (for example, in the frequently-used Atari environment, larger values between 0.99 and 0.999 are used). Second, the implementation of discounting is not correct (see remarks to the code).

Thesis Text

	good	OK	poor	insufficient
Form *... language, typography, references*	X	X		
Structure *... context, goals, analysis, design, evaluation, level of detail*		X		
Problem analysis		X		
Developer documentation		X		
User Documentation	X			

The thesis text consists of 54 pages and is well written, with only a few minor errors in English ("exedingly" instead of "exceedingly", "its" instead of "it is", a few commas).

To me, the weakest part is Chapter 4 describing existing tabular reinforcement learning algorithms, containing a few mistakes and inaccuracies.

From the typographical point of view, all vector graphic figures (diagrams, graphs) are included as bitmaps, which looks blurry (both in the electronic and in the printed version).

Remarks and Suggestions

- Page 8 and others, the you use references in double brackets like

  ... the Godot Engine (Linietsky [2021]).

  which should be typeset using the `\citep` command as

  ... the Godot Engine [Linietsky, 2021].
- Page 11, the horizontal divider after the last algorithm is on the next page, which does not look good.

(continues on the next page)

- Page 21, the on-policy and off-policy is described as

  On-policy learning means that the agent is using the same behavior policy to collect samples as it is using to improve the value function. Off-policy learning, on the other hand, means that the agent is using a different policy to collect samples than the policy it is using to improve the value function.

  However, the policy connected with the value function (the target policy) is *estimated* by the value function, not that the value function is improved using it. Note that on page 22, the off-policy is in fact described correctly.

- Page 21, the returns are defined in non-discounted form, even if the discount factor γ as already been defined on page 20 and all algorithms 1-5 compute the discounted return.

- Page 21, you state that

  While Monte Carlo methods are guaranteed to converge to an optimal policy with an infinite number of samples, . . .

  To my best knowledge, that is not true. Sutton and Barto (2020) state on page 99 that

  Convergence to this optimal fixed point seems inevitable as the changes to the action-value function decrease over time, but has not yet been formally proved. In our opinion, this is one of the most fundamental open theoretical questions in reinforcement learning. . .

  Maybe you confused Monte Carlo prediction (computing a value function for a fixed policy, which is guaranteed to converge) with Monte Carlo control (finding the optimal policy).

- Pages 23-24, in the description of SARSA, Q-learning, and Expected SARSA, you state that (citing from SARSA)

  the agent learns the value of a state-action pair Q(S',a) by estimating the expected return over all possible actions from state S'

  However, the agent learns the value of $Q(s, a)$ by considering actions a' in the state s' *following* the state s, i.e., a state resulting in performing the action a in state s. Consequently, the algorithms are very hard to understand from the text only.

- Page 23, you state about SARSA that

  This makes it well-suited for control problems, where the goal is to find an optimal policy.

  However, as an on-policy algorithm, it does not find the optimal policy, but an optimal ε-greedy policy, which might be considerably different to the optimal policy. The decay of ε to zero helps (because then the optimal ε-greedy policy converges to the optimal policy), but this is not discussed in Section 5.3.1 where ε decay is described.

- Page 24: *Expected SARSA can be seen as a compromise between SARSA and Q-learning*

  The Q-learning is in fact a special case of Expected SARSA, where the target policy is chosen to be greedy policy with respect to current action-state value estimate Q.

(continues on the next page)

- Page 28, the parentheses in the first formula are really small, use either `\left` and `right`, or `\big`, `\Big`, dots.
- Page 28, the $*$ symbol used in the formula is not scalar multiplication, but denotes *convolution operation*. Use `\cdot` for scalar multiplication.
- Pages 32-33, you discuss the same reason for setting `dists=1` twice in two consecutive paragraphs.
- Page 35, in *To produce the experiments ..., two scripts were written - one for ...*, the *en dash* should be used (`--` in LaTeX) instead of a hyphen.
- Page 52, I really liked the following comment, it makes the text enjoyable and live:
  with some sentiment ***`env=[Triangles]`*** *was chosen as this was the first environment on which the behaviour was noticed.*

Thesis Code

	good	OK	poor	insufficient
Design *... architecture, algorithms, data structures, used technologies*		X		
Implementation *... naming conventions, formatting, comments, testing*		X		
Stability	X			

Remarks and Suggestions

- The computation of discounting is not correct. Consider the following snippets:

```
var move = game.agent.move(game.state.get_state(), game.score.get_score(), (game.num_of_ticks * 33) / 1000.0)
...
func move(state, score, num_of_ticks):
    ...
        update_dicts(last_state_action, new_state_action, R, num_of_ticks)
    ...
    prev_time = (num_of_ticks * 33) / 1000.0
    ...
...
func update_dicts(last_state_action, new_state_action, R, curr_time, terminal = false):
    ...
    var new_gamma = pow(GAMMA,curr_time - prev_time)
```

  During the `prev_time` assignment, the $\frac{t \cdot 33}{1000}$ formula is applied for the second time; I believe you should use just `prev_time = num_of_ticks`.
- While the project is implemented in Godot 3.2.3, the scene `demo.tscn` was saved in Godot 4 and cannot be open using Godot 3.
- Page 26, Figure 5.1: it is surprising that Double Q-learning is not an instance of a `TDAgent`. While I understand that it is different by requiring two sets of action-value estimates, it *is* a TD agent. This suggest that what you call `TDAgent` class is in fact some specialization of a TD agent, and is either named inappropriately or not general enough. A better design would either make `TDAgent` general enough to encompass also the Double Q-learning, or there could be a specific subtype for the SARSAs and Q-learning.

(continues on the next page)

- Page 29: According to Algorithm 1, `G` in Monte Carlo implementation should be computed as `G = R + pow(GAMMA, ...) * G`. Is there any reason why you also discount `R`?
- Page 29: You suddenly use a non-constant learning rate α, while a fixed learning rate was used in algorithms 2-5. Is there any rationale behind your choice? I would be worried that for TD algorithms, such a learning rate could be too low (especially for Double Q-learning it might be one of the factors why it performs worse for larger initial optimistic value).
- Page 30, when reading the `get_update` methods, it is unclear what is the difference between `q[...]` and `Q(...)`, and which should be used when. If you include the method bodies in the thesis text, it should be possible to understand them.
- Page 31: Implementing Double Q-learning by copy-pasting the same code twice (with just `q1` and `q2` swapped) is definitely suboptimal.

Overall grade Very Good
Award level thesis No

Date Signature