Posudek bakalářské práce

Matematicko-fyzikální fakulta Univerzity Karlovy

Autor práce	Una Adilović
Název práce	Playing a 3D Tunnel Game Using Reinforcement Learning
Rok odevzdání	2023
Studijní program	Computer Science
Specializace	Artificial Intelligence

Autor posudku	Adam Dingle	**Role**	Supervisor
Pracoviště	KSVI		

Prosím vyplňte hodnocení křížkem u každého kritéria. Hodnocení *OK* označuje práci, která kritérium vhodným způsobem splňuje. Hodnocení *lepší* a *horší* označují splnění nad a pod rámec obvyklý pro bakalářskou práci, hodnocení *nevyhovuje* označuje práci, která by neměla být obhájena. Hodnocení v případě potřeby doplňte komentářem. Komentář prosím doplňte všude, kde je hodnocení jiné než *OK*.

K celé práci	lepší	OK	horší	nevyhovuje
Obtížnost zadání	X	X	☐	☐
Splnění zadání	☐	X	☐	☐
Rozsah práce *… textová i implementační část, zohlednění náročnosti*	☐	X	X	☐

Una's goal was to implement a 3D video game, then to write agents that can play the game using tabular reinforcement learning methods such as Monte Carlo or temporal difference learning. This was a somewhat challenging topic.

The game is nicely designed, and its implementation using the Godot engine seems solid. Una also implemented agents that use five different discrete learning algorithms, and I believe that these work correctly. I think she chose a reasonable representation for discrete game states.

Una has performed many experiments with her agents. It seems that these experiments were moderately successful: sometimes (but not always) the agents were able to learn a winning policy in environments with various subsets of game features. She found that the agents' success was quite sensitive to hyperparameter choices. It's not clear that her search for ideal hyperparameter values was especially systematic.

Some of the thesis text reads like a draft, not a finished thesis. In some places the text is not especially clear. There are various typographic and grammatical errors. I have the impression that some of the text was written in a hurry, and I think it could have been improved a lot with a couple more weeks of polishing and editing.

Textová část práce	lepší	OK	horší	nevyhovuje
Formální úprava *… jazyková úroveň, typografická úroveň, citace*	☐	X	X	☐
Struktura textu *… kontext, cíle, analýza, návrh, vyhodnocení, úroveň detailu*	☐	X	X	☐
Analýza	☐	X	X	☐
Vývojová dokumentace	☐	X	☐	☐
Uživatelská dokumentace	☐	X	☐	☐

In the introduction, Una states that she wants to use tabular reinforcement learning by quantizing the continuous game environment into a set of discrete states. This seems like a reasonable approach. However, has anyone done this for any other video game before, or is this a novel approach? Some more background would be valuable here.

Also, Una does not mention other possible approaches to learning, such as function approximation via neural networks. Some researchers have successfully used these methods to learn to play video games, but Una doesn't talk about that at all. Does she think that continuous methods would work well for her game? Has she chosen the discrete approach only because it's easier, or for other reasons? It would be nice to have at least some discussion of this.

For the most part, I think chapters 1 through 3 (Game Design, Implementation of the Game, Structure of the Experimental Setting) look pretty good. It's nice that Una has screenshots of the game and pictures of all the trap types. Her description of her object-oriented architecture seems reasonable.

In 3.2.1 "Command-line option descriptions", the description of the important parameter `agentSeedVal` could be a bit clearer. It would be better to say explicitly that this is a seed for a random number generator (and perhaps also to mention that the learning agents actually use randomness only for epsilon actions).

In the same section, the description of the "database" option has the text "we mean storing the final policy of the agent inside the Agent_databases and score of each game and the final policy inside Command outputs." If the final policy is stored in both places, then why is the Agent_databases directory necessary?

Chapter 4 (Applied Algorithms) aims to explain the various learning algorithms (such as Monte Carlo learning and Sarsa) that are used in the thesis. This is mostly a summary of algorithms that are described e.g. in the Sutton/Barto reinforcement learning textbook. The text here mostly seems fine.

In 4.1 "Monte Carlo", in the pseudocode in Algorithm 1 it seems not quite right that Returns(s, a) (which is a list of numbers) is initialized to the initial optimistic value (which is a single number). It would be clearer to say e.g. that Return(s, a) is initialized to a list containing just the initial optimistic value.

Chapter 5 (Implementation of the Agents) describes the implementation of the algorithms in Chapter 4. Mostly it seems OK.

Section 5.3.1 "Common parameters and behaviors" doesn't specify the default values of the parameters `initOptVal`, `eps`, and `epsFinal`. The formula at the end of this section would be clearer if the text explained that one tick is (approximately) 33 milliseconds long since

the game runs at a rate of 30 frames per second.

The first paragraph of section 5.3.2 "Monte Carlo Agent" contains a few sentences ("This calculation is performed at each state transition during the episode. To clarify, instead of calculating a new move...") that seem to repeat information that was given in the the last two sentences in the first paragraph of section 5.3.1 ("The agent will not request a new move until the state has changed...")

Chapter 6 (Testing and Plotting) describes Una's experimental approach, the Python scripts that she used to run tests and generate plots, and the format of the generated plots.

In section 6.1 "Conducting the experiments" the first paragraph is a bit confusing. It says "this shift did not yield significant results, and the same behaviour could likely be achieved by increasing the number of games..." What does "the same behaviour" mean here? If the switch to level 15 did not yield significant results, then why was it kept for future experiments?

In the same section, the second paragraph ("However, for obstacles...") and third paragraph ("After addressing...") contain basically the same information. It looks like these are two different versions of what was intended to be a single paragraph.

In this section Una explains her experimental methodology. From her discussion it's not obvious that she explored the space of hyperparameter values systematically enough. For example, she writes that she used `epsFinal` = 0.0001, and justifies this by saying "The rationale for using a low `epsFinal` value is that towards the end, the agent almost exclusively exploits the current policy". That makes sense. But just how low should it be? Did she try values such as .001, or .00001? There is no mention of that, and no graph showing how the performance of any agent varies with `epsFinal`.

Similarly, she writes "`initOptVal` of 20.0 and 100.0 was promising in most experiments", and "the `initOptVal` being either 20.0 or 100.0". It seems a bit odd to use only these two specific values. There's no graph showing how performance varies with `initOptVal` for any agent, so it's not evident that Una has found a value for this parameter that is even locally optimal.

Certainly the outcome of any experiment may depend a lot on the number of games that were played. Regarding the number of games, Una writes "The general practice was to choose n 10 to 15 times greater than the number of obstacles the environment used for a particular experiment." But in the experiments with just 1 trap type (described in section 7.1) she ran 150 games, which is 150 times greater than the number of obstacles (i.e. 1). This discrepancy is unclear.

Furthermore, she writes that choosing this value of n "was found to be sufficient for facilitating the training of the agents." Was it really sufficient? That's not clear, because in many of her experiments the agents did not reliably learn to win the game. For example, in her experiments with just one trap type she chose n = 150, and in most cases the agents learned to win the game some, but not all, of the time. With n = 300 or n = 500, might they have learned to win the game all the time? It's not evident from her text that she tried these larger numbers of episodes.

In section 6.2 "Interpreting the plots" Una explains the format of the plots that appear in the thesis text.

In this section she writes "parameters such as the number of seeds in the averaged data line or the size of the smoothing window will be clearly specified for each plot mentioned in the rest of the chapter". It would be clearer if these parameters were specified above each plot, rather than in the surrounding text.

Section 6.3.1 "Testing" explains the Python script that Una used to run here tests. The text here uses the term "variables" to describe the parameters `n`, `stoppingPoint` and so on that the user can provide. I think the term "parameters" or "options" might be clearer.

This section doesn't explain that test.py will write both into the `Command_outputs` directory (which is mentioned elsewhere in the text) and also into a directory `Tests_output`. It would be better if the text explained why both these directories are needed.

This section also does not mention that inside test.py the default value of `stoppingPoint` is 10, which differs from the default value of 25 that was mentioned in section 3.2.1 "Command-line option descriptions". So presumably all the experiments in the thesis were actually run with `stoppingPoint` = 10, though that is not clear from the text.

In the script test.py, it's awkward that if I want to specify a single value for a parameter then I must specify a range, e.g. [100.0, 200.0, 100.0] for the single value 100.0. It would be nicer if I could specify either a single value or a range. This would be a very easy change.

It's nice that the framework stores outputs in filenames that include most of the experimental parameters (e.g. 'agent=DoubleQLearning,agentSpecParam=[eps=0.20,epsFinal=0.00010,gam=1.00,...'). However, the number of games that were run is not recorded in the filename. That seems unfortunate, because if I want to run experiments with various numbers of games (e.g. 100, 300, 1000), it seems that I need to remove or rename all the existing files each time I want to change the number of games.

Section 6.3.2 "Plotting" says "The number of seed values included and which agents are displayed depend solely on the accuracy of the files in the Command outputs folder". The word "accuracy" seems odd here - what does it mean, exactly?

In the script plots.py, the option name "option" is completely vague, and its values "1" and "2" aren't very meaningful. Since option=2 generates a combined graph with data averaged over multiple seeds, I think a better name for this option would be "combine". Then if the user specified "--combine" then the script could generate a combined graph, otherwise not.

Chapter 7 (Experiments) presents the results of experiments with the learning agents in a series of environments, ranging from those containing only a single trap type to the full game with traps, creatures (bugs and viruses), and energy tokens.

Section 7.1 "Individual traps environments" begins with a set of 10 graphs showing the performance of the 5 agents in each of the 10 single-trap environments. There is no winning line on these graphs, so to the reader it may not be clear how successful these agents were. However, in a previous section Una did explain that a typical winning score is

about 500. In some graphs, the average score of some agents is an integer fraction of that number. For example, in the graph "Env: Hex" several agents have an average score close to 100 points. Probably that's because in 1 of 5 random seeds the agent learned an optimal policy (scoring about 500 points per game), and in the other 4 seeds it did not (scoring almost no points per game). However, Una does not discuss this interpretation at all.

In the same section, Una writes "for most trap/agent pairs, we managed to find at least one combination of hyperparameters where the agent found an optimal or a policy that won some games but not consecutively, across multiple seeds." It would be better to list these hyperparameter values in a table; their values might be interesting, and the reader might want to repeat the experiments to confirm that this assertion is true. As the text is written, the reader must simply take it at face value that these hyperparameter values exist.

Section 7.2 "Traps environment" presents the results of experiments in an environment with all trap types, but no creatures (bugs/viruses) or tokens.

In this section, above Figure 7.6 Una writes that the experiments "yielded a policy that managed to win enough times to have an average score between 100 and 200 or more". This makes it sound if the individual policies in the figure ran many times and had average scores in that range. However, from the text below the figure it seems that the were the average scores in experiments in which the policy was constantly changing, and the figure only shows the final policies in these experiments, which only ran once at the end of the experiments. I think the difference between the assertion above the table and the experimental reality is confusing.

Section 7.3 "Tokens environment" shows the results of experiments in a relatively easy environment containing only energy tokens. Una writes that "the average performance of all agents is commendable", but that doesn't seem so clear: none of the agents apparently achieved a winning score of 500 by the end of the experiment. If this environment is so easy, why didn't all the agents win all the time? The number of episodes that Una chose here (20) seems pretty small. Would the agents do better with e.g. 100 episodes?

Section 7.4 "Individual Bugs and Viruses environments" discusses agent performance in environments containing only a single bug or virus type. In these environments most agents did not learn a policy that was close to optimal, especially in the environments in which shooting was enabled. The number of episodes in these experiments (50 without shooting, 80 with shooting) seems relatively small. Would a larger number of episodes improve performance?

Section 7.5 "Bugs and Viruses environments" discusses environments containing multiple bug or virus types and/or energy tokens.

Figure 7.14 "Bugs and Viruses with and without shooting comparison" is confusing. According to the text, two of these plots depict the Bugs environment, and two depict the Viruses environment. And apparently in two of them shooting is disabled, and in two shooting is enabled. However, the plots are not labelled with environment names, and the text does not clearly say in which plots shooting is enabled. The last sentence does say "with the right plot showing an average and winning scores approximately 10 times higher than on the left, as a consequence of the ability to shoot", which implies that shooting is

enabled in the plots on the right (as in previous sections), however it would be clearer to state this explicitly.

Section 7.6 "Full game environment" presents arguably the most important results in the thesis, i.e. the agents' performance in the full game. It appears that in the full game the ExpectedSARSA agent performed the best, and on average was able to learn a policy that scored about 100 points per game (whether or not shooting was enabled).

Figure 7.16 "Catastrophic forgetting experiment" is a bit confusing. The plots in this figure have the same hyperparameter values as in Figure 7.15 "Full game experiment", except that the initial epsilon value is 0.2. So is this a variation of the experiment of Figure 7.15, in which the initial epsilon is different but everything else is the same? The text below the figure says "The policy it learned in the first couple of thousand games was far from optimal, but it achieved a better score than the policies used after the sudden drop at approximately 2500 games." That seems to imply that the plots in this graph are showing results with only a single seed value, rather than the average of 10 seed values as in Figure 7.15, but the text doesn't state this explicitly so it's unclear.

Figure 7.17 shows that with some particular random seed value the ExpectedSARSA agent was able to learn a policy that won hundreds of games. What was that seed value? The text doesn't specify it, so the reader can't repeat the experiment and must just take it on faith that such a seed value actually exists.

Section 7.7 "Interesting behaviors" first explains that learning experiments performed best when there was no reward discounting at all, i.e. when gamma = 1.0.

After that, the text explains that the agents sometimes learned to stay at the boundary between two adjacent discrete rotations. The text claims that this was a consequence of the fact that the agents chose a new action only when the discrete state changed. The behavior is interesting, but it's not obvious to me that this is a consequence of this particular architectural choice. If the agents chose a new action at every time tick, then they would choose epsilon actions much more frequently than in the current implementation, since they would make many more decisions per second. Otherwise, I think their learning behavior would be the same as in the existing implementation: for any policy, the action is a function of the current state, so if the state did not change then the action would not change. So even if agents chose a new action at each time tick, perhaps they would still be able to learn an optimal policy, which might involve wobbling between two rotations.

The Conclusion states that "the ExpectedSARSA algorithm performed moderately or exedingly well in all environments". That does not appear to be true: in some of the individual bug/virus experiments described in section 7.4, ExpectedSARSA did quite poorly compared to other agents. The conclusion also states that "All algorithms displayed adequate performance in environments with individual trap types". That statement doesn't seem fully justified by the results in section 7.1 "Individual trap environments". In particular, in Una's discussion there she explicitly mentions some trap/agent pairs (e..g MonteCarlo with the Hex trap) where she was unable to find hyperparameters that let the agent win any games.

In many places the thesis says that certain algorithms outperformed others in various experiments. There is not much analysis of why this is the case, i.e. what it is about these

algorithms that made them better or worse suited for particular environments. In particular, why is ExpectedSarsa so much better than the other algorithms in general?

The conclusion does not say much about future directions for this work. It says "Future research could explore ways to influence these random actions to potentially achieve better outcomes.", but that's pretty vague. It would be good to at least mention other possible discrete (tabular) reinforcement learning algorithms, and also those using continuous function approximation, e.g. neural networks.

Throughout the work, in various places the text seems like a draft, not a finished thesis. Here are some examples of typos and English errors in the text:

- In 3.2.1 "Command-line option descriptions", in the description of the "database" option the directory names "Agent_databases" and "Command_outputs" should be in a fixed-width font. The phrase "can be used start another session" is missing a word.

- In 3.2.2 "Running the program", "we recommended to use" should be "we recommend to use".

- In 5 "Implementations of the Agents" the word "agents" is duplicated.

- In 6 "Testing and Plotting", the phrase "the plots what will be used" is ungrammatical; "what" should be "that" or "which".

- In 6.1 "Conducting the experiments", the second sentence "the primary concern..." doesn't begin with a capital letter. In the third paragraph it seems wrong that "Bug", "virus" and "token" are in a fixed-width font, since these are not literal parameter names. "it should mentioned" should be "it should be mentioned". "how we shoose" should be "how we choose".

- In 6.3.1 "Testing", "agentsSeed" should be "agentSeedVal". Also, the fixed-width text "[min,max(not inclusive),step]" looks odd, since this is not text that the user should enter literally.

- In 7.1 "Individual traps environments", there's a phrase "where the agent found an optimal or a policy..." that doesn't make sense.

- In 7.4 "Individual Bugs and Viruses environments", the first two sentences are poorly worded. The first sentence talks about the "performance" of "obstacles", but obstacles don't have a performance; agents do. In the second sentence, the pronoun "they" is unclear. The phrase "always avoid the these obstacles" evidently has an extra word.

- In 7.5 "Bugs and Viruses environments", the phrase "the performance of MonteCarlo agent shown environments" is ungrammatical.

- In the Conclusion, "exedingly" should be "exceedingly".

- In the Bibliography, the first/last names of the authors of the textbook "Artificial Intelligence: A Modern Approach" are swapped. The year given for the textbook (2010) is wrong; it should be 2021. "4 edition" should be "4th edition".

Implementační část práce	lepší	OK	horší	nevyhovuje
Kvalita návrhu *… architektura, struktury a algoritmy, použité technologie*	☐	X	☐	☐
Kvalita zpracování *… jmenné konvence, formátování, komentáře, testování*	☐	X	☐	☐
Stabilita implementace	X	☐	☐	☐

The implementation of the Space Run game itself seems quite nice. The game looks good, runs quickly and is fun to play. I think that Godot and GDScript were good choices for this game. Una used Blender to design custom 3D models for the creatures in the game, which was impressive. It's great that the game has 10 different trap types, which provide an interesting and varied environment for reinforcement learning.

It's also nice that the program can create obstacles and shift tunnels forward dynamically, which allows the game to simulate an infinite series of tunnels while still only having a fixed number of objects in memory at any one time.

The program has a command-line interface that can run a series of games automatically and accept various options that configure the game. It allows the game to be controlled by any agent class that implements a couple of abstract methods (e.g. move, init). All of this is good.

The implementation of the learning agents themselves seems fine. It's nice that Una implemented the agents using a class hierarchy, which allows a lot of code to be shared between them.

The implementation also contains a couple of Python scripts that can produce graphs from data that was generated by training runs. These seem to work well, though I've suggested a couple of improvements to them in my longer review of the thesis text above.

Celkové hodnocení Velmi dobře (2)
Práci navrhuji na zvláštní ocenění Ne

Datum 29.5.2023 **Podpis**