Reinforcement agent fitness in an cyclic environment including a dynamical predator with a variable time sensor

L.P. Janssens
s2989344
LIACS
Leiden University
The Netherlands
s2989344@vuw.leidenuniv.nl

In this paper, I will discuss two reinforcement agents, one with sensory input of time, and one that does not. I will consider a deterministic environment with a cyclical dynamical predator and an objective the agent wants to reach. The reinforcement agents should avoid the predator at all times and thus learn the pattern of where the predator is at a certain time step. The agent with sensory input of time can learn directly at what time the predator is at a certain place. However, the agent without sensory input of time, cannot. It must thus learn via the number of occurrences the predator is at a certain state. First, I am going to discuss how I set up the environment. This is done by defining what states the environment consists of and what the actions there are. Then, I am going to discuss how the agents operate and the difference in mental representations of the environment in each of the agents, and the approach of combinations of actions to get to the objective, using SARSA or Q-Learning as updating function. Lastly, I will compare the average cumulative rewards of rats having a sense of time and rats that do not have a sense of time and compare a complex representation where the predators are deterministic over a simpler mental representation where the predators are not deterministic.

Introduction

The principle behind reinforcement learning is learning via the emergence of a stimulus-response habit (Melnikoff et al., 2021). Thorndike (1911) came up with the law of effect, which states that the link from the stimulus to the response is strengthened when a response in the presence of a stimulus is followed by a reward. These associations with rewards or punishments are then processed back into the memory of the agent and are reused once it is back in a similar situation. In reinforcement learning, these habits are directly saved in the agents' memory, via a policy with fitness values. Learning is thus done model-freely since it reacts upon the reward received from doing an action in a certain state (Daw, 2012).

In this paper, I am going to make use of a dynamic environment, where certain punishing states move in a cycle. Agents with a sense of time can make use of the fact of when they have encountered the punishing state. This is paired with a more elaborate state space but allows for better cognitive mapping. The agents without a sense of time cannot make use of time and do not know the time when they have encountered a punishing state. This then also comes with a simpler state space, but limited cognitive mapping. I will be observing the difference in cognitive mapping as well as overall fitness. Furthermore, I will look into the difference between SARSA and Q-Learning as updating functions combined with having a sense of time or not.

A paper done by Wiering (2001) tests something similar to agents not having a sense of time. In their paper, there exists an environment containing wandering spiders. The agent that walks in this environment learns for several episodes the possible locations of the spiders by encountering them. This is a five-by-five area, with the spider nest in the center. The agent is only able to find possible locations, but not when the spider is there. Therefore, it is meant for the agent to handle uncertain information and thus learn which states are safe(r) and which states are more dangerous, to which it adapts its path. This is somewhat similar to the agents in this paper, since the agents without a sense of time can never be certain that a punishing state is at a certain position, but can find out that a punishing state could be at that position.

The outline of the paper is as follows. First, I will explain the method as to how I set up a cyclic environment with dynamic s_P states and agents. Next, I am going to discuss the results found for three various environments. I will do this by addressing their behavior and fitness in the environment and how this relates to the agents' updating method and their sense of time. Lastly, I summarize what I have observed and conclude. I end with a discussion and possible future work.

Method

Environment

Without an element of time, I specify an environment as a grid with size width w by height h. This environment is thus made up of $w \times h$ states. These states contain tags, which are coupled to values. The following tags are possible for a state to have:

- $'S'$: The starting state, which has a value of -1 and is signified by the color yellow.
- $'.'$: An empty state, which has a value of -1 and is signified by the color gray.
- $'P'$: A predator state, which has a value of -100 and is signified by the color red.
- $'R'$: A reward state, which has a value of 100 and is signified by the color green.
- $'X'$: A non-accessible state, which has a value of 0 and is signified by the color black.

Throughout the paper, I will address states with certain tags by s_{tag}, which refers to the aforementioned list of tags. A tag is also coupled with whether a state is a terminal state. A terminal state is a state where the agent is not able to make any more actions. For instance, a reward state s_R and a predator state s_P could be terminal states, such that, once the reward is obtained you do not continue the search, or once caught by a predator, no more searching can be done. Lastly, as briefly mentioned, every state has a collection of actions that can be done to move to the next state. An action consists of a tuple $(\Delta w, \Delta h)$, which defines the move from the current state to the next state. For example, an action that defines a move to the right in such an environment would be $(1, 0)$. Another action could be a move to itself, and thus standing still, which is defined as $(0, 0)$.

Cyclic environment

To make an environment cyclic, there has to be an element of time. This allows a certain state s to be at a position $(0 \leq x < w, 0 \leq y < h)$ at a specific time t. Time, in this case, adds a third dimension. An environment, therefore, is defined as a stack of t layers of w by h grids. An action in this environment is defined as a tuple $(\Delta w, \Delta h, t + 1)$, where it is definite that an action results in being in the next time step. Once the end of the stack of layers is reached, the time gets reset $t = 0$. In this way, you get a cyclic environment. Figure 1 shows an environment of $w = 3$, $h = 3$ and $t = 4$. In this environment, you see a s_P state moving from left to right and back. This action is cyclic and takes four steps to complete. These are four unique steps, even though the s_P state is twice in position (1,1). This is because, the action taken by both s_P states are not the same: at $t = 2$, the action taken is (1,0), while at $t = 4$ the action taken was (-1,0). It should also be noted when there are multiple s_P states present, that they can move into each other or through each other. In that case, only one s_P state is shown, and only once the points are reduced. The cycle ends when the orientation of all states along with their movement matches the orientation of the first time step along with their movement.

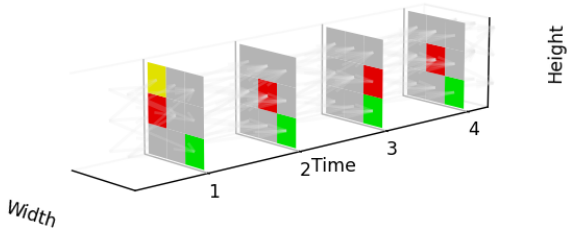

Figure 1

Cyclic environment of $w = 3, h = 3, t = 4$

Reinforcement use-ability

To make use of a three-dimensional environment, I will first flatten it down to a two-dimensional environment. I will do this by taking each layer, and removing the h dimension, such that the resulting environment's dimensions are $w \times h$ by t. An example is shown in figure 2. Here you can see how one grid is 'flattened' to form a one-dimensional grid. Stacking multiple grids transformed in the same way results in a two-dimensional grid. It is noteworthy that performing actions in this new environment is not visually trivial. Every action that was possible to perform in the three-dimensional environment, can still be performed in the two-dimensional environment. However, as you can see in figure 3, actions are translated as well. Instead of a tuple of three elements, as defined as $(\Delta w, \Delta h, t + 1)$ earlier, the tuple consists of two elements defined as $(t + 1, \Delta w + (\Delta h \times w_{old}))$.

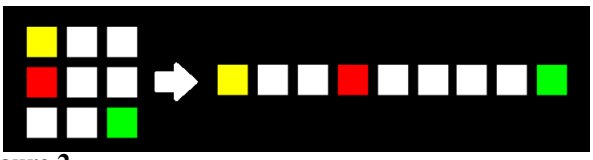

Figure 2

Grid of $w = 3, h = 3, t = 1$ *flattened into* $t = 1, w_{new} = w_{old} \times h$

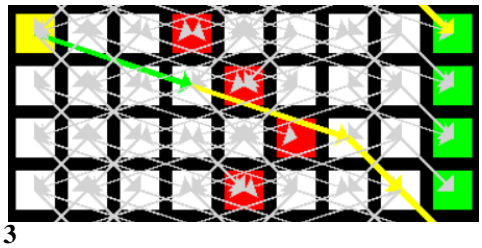

Figure 3

Environment of figure 1 from three-dimensional to two-dimensional representation, with, in gray the actions from some state to result in another state and with, in yellow the path of the current agent in the environment. Notice how there are no actions from states s_R and s_P.

Rats

Agents can be defined in the environment. In this case, these are called rats. Rats walk through this environment following a certain policy. A policy consists of a table of size $\#actions \times \#states$, such that there is a pair $Q(s,a)$ for each state s and each action a. On initialization, the $Q(s,a)$ values are set to random values[1]. However, this does not hold for terminal states. For every terminal state $s_{tag \in terminals}$, $Q(s_{tag \in terminals}, \cdot) = 0$.

Rats can make use of two different updating functions, SARSA or Q-Learning. These model-free updating functions are run for several *episodes*. In each episode, the $Q(s,a)$ values are adjusted accordingly to their performance. If the $Q(s,a)$ value contributed to getting to an s_R, the value will increase, whilst the opposite happens when it contributed towards an s_P state. Both Q-Learning, seen in 1, and SARSA, seen in 2, take three variables.

- A learning rate $\alpha \in (0,1]$, which is responsible for the swiftness of updating the $Q(s,a)$ values, dependent on the prediction error. A higher α means a bigger step size, and thus faster changes in the $Q(s,a)$ values. The opposite is true for a low α (Katahira, 2015).
- A discount factor γ, which is responsible for discounting the maximum expected future reward. This factor is constant and the calculations done with this variable are also constant, i.e. the obtained reward does not gradually decrease after many actions.
- And a small epsilon > 0. This is used in the ϵ-greedy policy (see equation 1) and has to do with the way an action is chosen in the updating functions.

$$action = \begin{cases} argmax_a(Q(s,a)), & \text{if } r \sim U([0,1]) > \epsilon. \\ random(a), & \text{if } r \sim U([0,1]) \leq \epsilon. \end{cases} \quad (1)$$

The epsilon states there is an ϵ chance the action that is chosen is a random action. Otherwise, it chooses a greedy action. For $\epsilon = 0$, the action picked is always greedy. Similarly, for $\epsilon = 1$, the action picked is always random. This simple algorithm is used for the rat to pick an action and is applied throughout the paper when talking about a rat picking an action and allows for exploration rather than sticking to a greedy approach.

The idea of Q-Learning is to learn predictions of the reward when being in state s and taking action a (Rummery and Niranjan, 1994). These values are stored in a so-called Q-function, which contains all $Q(s,a)$ for all combinations of states s and actions a. These predictions are continuously updated per episode by:

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(S',a) - Q(S,A)]$$

where R is the reward received for moving to the successor state S', $\gamma max_a Q(S',a)$ is the maximum expected value of the successor state, and $Q(S,A)$ the expected value of the current state. The square brackets combined yield the prediction error. If this is zero, the prediction is perfect and there is no need to learn. Q-Learning is an off-policy temporal difference control algorithm, where the behavioral policy is not the target policy (Sutton, 1988). This is because, in Q-Learning, an action is chosen and immediately executed, before planning the next action. This leads to more greedy behavior in combination with not planning (Sutton and Barto, 2018).

Algorithm 1: Q-Learning

```
Parameters: α ∈ (0,1], γ, ε > 0

FOR each episode:
    Initialize starting state s
    WHILE S is not terminal:
        Choose A from S using policy from ε–greedy
        Do action A
        S' ← state after doing action A in S
        R ← S_value
        Q(S,A) ← Q(S,A) + α[R + γmax_a Q(S',a) − Q(S,A)]
        S ← S'
    ENDWHILE
ENDFOR
```

SARSA, or modified connectionist Q-Learning, is an algorithm that is quite similar to Q-Learning (Rummery and Niranjan, 1994). However, the difference lies in the updating of the $Q(s,a)$ values.

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$$

The name SARSA implies State-Action-Reward-State-Action, which summarizes what is going on in this algorithm. The

[1] There is an option to set these values all to 0 as well, by specifying that *randomQ* = False

next action is planned before the $Q(s,a)$ value is updated. In this way, there is prior planning involved and the maximum expected value of the successor state is equal to $Q(S',A')$, which is the expected value of the successor state S' taking action A'. This is thus an on-policy temporal difference control algorithm where the behavioral policy is also the target policy. This leads to safer behavior, and somewhat less greedy behavior (Sutton and Barto, 2018).

Algorithm 2: SARSA

```
Parameters: α ∈ (0,1], γ, ε > 0

FOR each episode:
    Initialize starting state s
    Choose A from S using policy from ε−greedy
    WHILE S is not terminal:
        Do action A
        S' ← state after doing action A in S
        R ← S_value
        Choose A' from S' using policy from ε−greedy
        Q(S,A) ← Q(S,A) + α[R + γQ(S',A') − Q(S,A)]
        S ← S'
        A ← A'
    ENDWHILE
ENDFOR
```

The last parameter to explain is the parameter *senseTime*. *senseTime*, in essence, means that the rat knows what time step it starts in, and thus can use the knowledge that it has previously gained from being in that specific time step. It is randomized in which time step the rat starts, but it does have the ability to gain the knowledge of what time step it is in. This allows the rat to adjust its $Q(s,a)$ values with respect to its time step. A rat without a sense of time is not able to gain knowledge of what time step it is in. It also starts in a random time step but does not have the ability to gain knowledge of what time step it is in. This does not allow the rat to adjust its $Q(s,a)$ values with respect to the time it is in, and thus must learn in another way. This is done by not adjusting the $Q(s,a)$ value for a specific time step, but for all time steps. Because of the fact that time is not a relevant factor to the rat anymore, the mental representation of the environment changes. It must now learn on the basis of the percentages of the rat ending up in an s_P state when it does an action. For example, in the environment of figure 1, it must learn that the predator, wandering around in the middle row, is 25% of the time on the state in the left column, 25% of the time on the state in the right column and 50% of the time in the state in the middle. This differs from the rat that has a sense of time since the rat that has a sense of time can learn that a predator is 100% or 0% in a certain state at a certain time. This differs from how Wiering (2001) has done it. In their paper, literal probabilities are used to determine the reward received as well as used in calculating the policy values. This allows for gaining the exact knowledge of the agent as to how likely it is for a spider to be in the next state. In this paper, the probabilities are directly incorporated into the policy values. The agent does not directly know what the probability is of there being a punishing state, but there is a clear difference in the policy values between safe(r) states and more dangerous states.

Results

The code can be found in this GitHub repository. This repository contains an elaborate explanation of how to use the code. I suggest reading through this if a part is unclear.
In each experiment, I will be using algorithm 3 as the base. It works as follows: I start by setting up the environment using a folder. I define the actions to be up, down, left, right, and possibly standing still[2]. Then the environment is set up using *setup()*. Following this, I create a number of groups of rats. There are four selections: rats using SARSA as updating policy function with a sense of time, rats using SARSA as updating policy function without a sense of time, rats using Q-Learning as updating policy function with a sense of time, rats using Q-Learning as updating policy function without a sense of time. Each group contains 100 rats. Furthermore, every rat uses $\alpha = 0.1$, $\gamma = 0.9$, $\epsilon = 0.1$ and all $Q(s,a)$ values are initialized randomly. I train every rat for 500 episodes and compare every group's average cumulative rewards in a plot.

Algorithm 3: compareSenses()

```
env = Environment(folder,backToStartOnP=False)
env.defineActions([(0,1),(1,0),(-1,0),(0,-1)])
env.setup()

rats = 100
episodes = 500

# SARSA rats with sense of time
env.generateRats(rats, method="SARSA", args
    =(0.1,0.9,0.1), randomQ=True, senseTime=True)
# SARSA rats without sense of time
env.generateRats(rats, method="SARSA", args
    =(0.1,0.9,0.1), randomQ=True, senseTime=False)
# QLearning rats with sense of time
env.generateRats(rats, method="QLearning", args
    =(0.1,0.9,0.1), randomQ=True, senseTime=True)
# QLearning rats without sense of time
env.generateRats(rats, method="QLearning", args
    =(0.1,0.9,0.1), randomQ=True, senseTime=False)
# Train all rats
env.trainRats(episodes=episodes,all=True)

# Plot performances
env.plotPerformance([
    ({"method": "SARSA", "senseTime": True}, "SARSA
        with time"),
    ({"method": "SARSA", "senseTime": False}, "SARSA
        without time"),
    ({"method": "QLearning", "senseTime": True}, "
        QLearning with time"),
    ({"method": "QLearning", "senseTime": False}, "
        QLearning without time")
])
```

I will compare three environments, each different in its own way. The first environment has just one path, and an s_P state that every rat must go through. The second environment considers a wandering s_P state, which leaves room for the

[2] Applied in environments 1 and 3

choice of more danger versus less danger. The last environment considers a dangerous short path and a safer longer path, which does not include any danger.

Environment 1

The first environment clearly shows the problem of not having a sense of time. The environment is shown in figure 4. This environment shows three time steps, with one having no s_P state present. The rats should wait until the s_P state has disappeared and can safely cross. In this environment, the rats are allowed to stand still (meaning that action (0,0) is in the defined list of actions). The rats with a sense of time can directly learn at what times the s_P state is present and therefore learn to avoid them. The rats without a sense of time can only learn that the s_P state is 33% of the time present, but do not know with certainty when to cross safely. This result is immediately clear from figure 5. It is interesting to see how both groups of rats with a sense of time perform similarly, but the groups of rats without time perform quite differently. The rats without a sense of time using Q-Learning perform better than rats without a sense of time using SARSA. When visualizing the cognitive map[3] of the rats using SARSA and not having a sense of time, they are not moving from s_S, no matter what time step they are in. Unless the action picked is random, it moves away from s_S. Then again, if a greedy action is followed, it moves back to s_S. The only time the rat would try to pass the s_P state, it takes two random actions in sequence. The Q-Learning rats without a sense of time, however, move back and forth between the s_S and the neighboring state. As stated earlier, Q-Learning is stated to show more risky behavior. This however pays out, since the random action the rat takes 10% of the time could be the lucky move passing the s_P state and allowing it to reach the reward, or being unlucky but just losing 100 points instead of a continuous loss of points due to standing still or moving back and forth. The rats with a sense of time perform well. Both the SARSA and Q-Learning rats wait on s_S until they can safely cross. They do this by not stopping once in the process. This is straightforward since the rats with a sense of time can directly learn at what time the s_P state is present.

Environment 2

The second environment is the same environment seen in figure 1. In this environment, as explained earlier, there is a s_P state moving from left to right and back. This means that the s_P state has a 50% chance of being in the middle and 25% chance of the s_P state being at the left or right for some random time step. It should be noted that the rats cannot stand still in this environment. The results are shown in figure 6. Again, a clear significance is seen between the rats that do

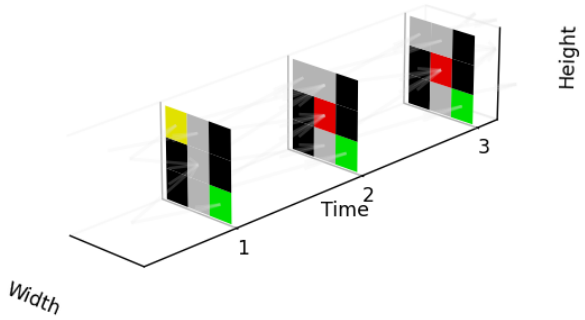

Figure 4

Environment 1. The environment consists of $t = 3$ timesteps, each consisting of a grid of $w = 3$ and $h = 3$. There is only one way to go to the reward, and that is crossing the middle column. The middle state of that column is 33% of the time clear.

have a sense of time and the rats that do not have a sense of time. The rats that do have a sense of time both perform well: they can avoid the s_P state quite well on average. The rats that do not have a sense of time, are performing worse. However, the rats do improve, as seen in the first 50 episodes. After examining the cognitive maps by both SARSA and Q-Learning rats without a sense of time, it is observed that they avoid the middle state of the middle row at all times. I assume they have learned (within 50 episodes), that it is safer to go via the left or the right sides than via the middle. For example, figure 7 visualizes the heat map of the rats using SARSA without a sense of time. You can see that the rats walk via the right side to the reward. The reason why some states are lightly blue-tinted is that this action is done based on the ϵ-greedy policy, where this is the result of a random move.

Environment 3

The last environment gives the rats a choice: either go for a dangerous path or take a safe path. The dangerous path considers a short path of around two actions to get to the reward, but there is a s_P state present 75% of the time. The safer path consists of a longer path of around nineteen actions but is safe 100% of the time. The rats are allowed to stand

[3] In the code, remove the return statement of the function to draw out the average rat of each group. This will show the model as well as an animation.

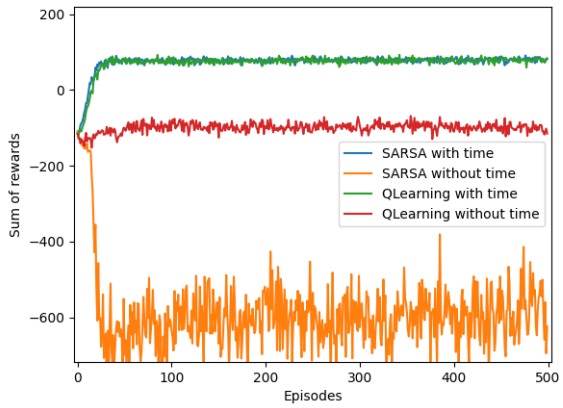

Figure 5

Environment 1 fitness. You can clearly see that the rats with a sense of time perform much better. You can also see a significant difference in the rats that do not have a sense of time using either SARSA (orange) or QLearning (red) as updating function

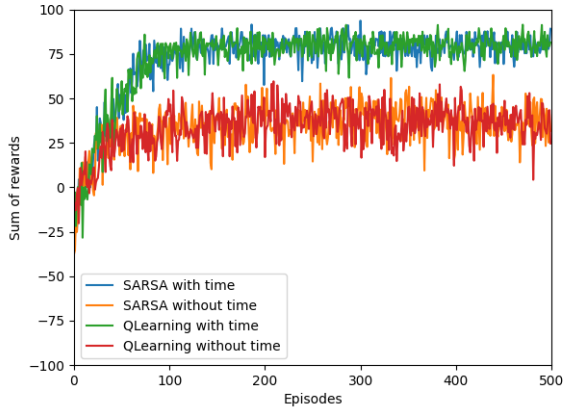

Figure 6

Environment 2 fitness. You can see that on average the rats with a sense of time perform better than the rats without a sense of time. However it is noted that the rats without a sense of time do learn in to take a safer path within the first 50 episodes.

still in this environment, which allows them to wait out the s_P state and time their movement. The environment is shown in figure 8.

The results, shown in figure 9 show that all rats perform quite similarly, yet the rats with a sense of time perform just a little better. This accounts for both SARSA and Q-Learning rats with a sense of time performing better than both SARSA

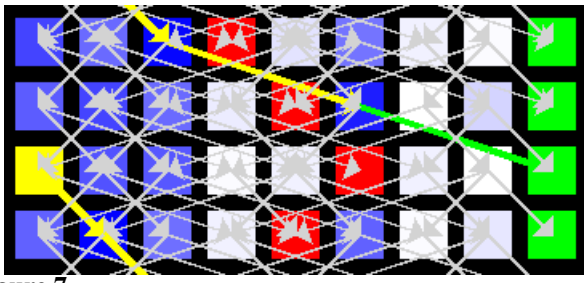

Figure 7

Environment 2 average route for rats without sense of time using SARSA as updating function. The blue tinted states represent a heat map of where the rats have gone to. The darker the blue, the more that state is walked in to. It is observed via the heatmap that they avoid the middle state of the middle row.

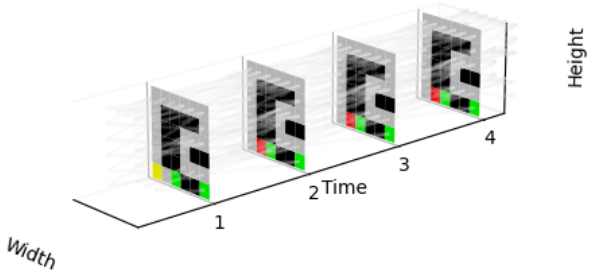

Figure 8

Environment 3. The environment consists of $t = 4$ timesteps, each consisting of a grid of $w = 6$ and $h = 6$. There is a dangerous approximately two-actions path with a s_P state present 75% of the time and a safe approximately nineteen-actions path with no s_P state present.

and Q-Learning rats without a sense of time. After observing their behavior in the environment, it is noted that the rats with a sense of time waiting for the right moment to cross the s_P state. This allows them to get to the reward in approximately two to five actions. The rats without a sense of time will never consider the dangerous option and will take the longer option around. They do however get to the reward in approximately nineteen actions. This, in essence, yields a good performance too. However, the rats without a sense of time will always perform more actions and thus lose more points. That is why the curves of the rats without a sense of time are close to the curves of the rats with a sense of time, but still just below it.

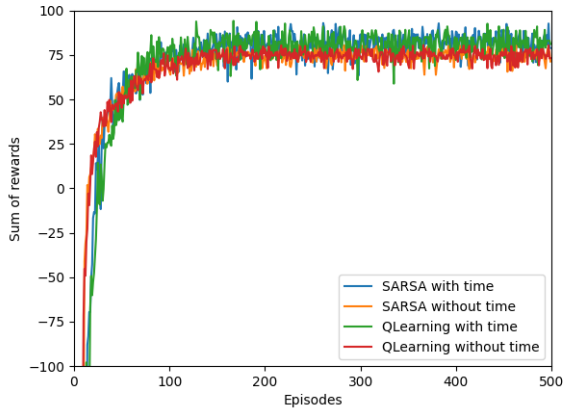

Figure 9

Environment 3 fitness. You can see that the rats with a sense of time perform better than the rats without a sense of time. They do not lie far apart, since the difference only consists of the extra points deducted from taking a longer path.

Discussion

I have covered the way I transform three-dimensional environments into two-dimensional environments. Furthermore, I defined what a rat is, and how the movement is controlled via a policy that is updated using SARSA or Q-Learning. Additionally, I have added instructions in the GitHub repository. Lastly, I tested whether the rats with a sense of time, and thus using a complex deterministic environmental representation, have better fitness than the average rats without a sense of time, and thus use a simple non-deterministic environmental representation.

I found out that rats without a sense of time perform worse in general. In the first environment, rats without a sense of time were terrible in timing when to cross the predator state, and thus failed. There was a difference to be seen between Q-Learning and SARSA, which comes down to the characteristics of Q-Learning, which is taking more risks, that at the end paid out. In the second environment, the rats without a sense of time performed worse as well. However, they did manage to learn to take a safer path instead of more dangerous options. Lastly, to finalize the finding in environment 2, I found out that if there is a 100% safe path present, the rats without a sense of time are not bothered to go a long way if it means that it is safe. This means that the stimulus-response habits formed show a cognitive mapping concerning a safer path.

The results found in this paper are in line with the results in Wiering (2001). The agents can establish a solution that minimizes the risk of making an action to an s_P state. As seen in this paper, the agents will choose the safest path over any more dangerous path if possible. As expected, if the agent is aware of there being an s_P state, the agent can avoid it, which allows the agent to take a more risky path.

For further research, one could look into an environment with a moving reward. This not just makes the agent time the s_P states right, but also finds out where the rewards are located at that specific moment. Could an agent be tricked into thinking the reward is in a certain state, but then get surrounded by s_P states? Furthermore, one could also look into a second agent whose goal is to find the first agent, in a dynamic environment with moving s_P states. How would the cognitive mapping look like of such an agent?

References

Daw, N. D. (2012). Model-based reinforcement learning as cognitive search : Neurocomputational theories.

Katahira, K. (2015). The relation between reinforcement learning parameters and the influence of reinforcement history on choice behavior. *Journal of Mathematical Psychology*, *66*, 59–69. https://doi.org/10.1016/j.jmp.2015.03.006

Melnikoff, D. E., Bargh, J. A., & Wood, W. (2021). Editorial: On the Nature and Scope of Habits and Model-Free Control. *Frontiers in Psychology*, *12*. https://doi.org/10.3389/fpsyg.2021.760841

Rummery, G., & Niranjan, M. (1994). On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*.

Sutton, R. S., & Barto, A. G. (2018, November 13). *Reinforcement learning, second edition: An introduction (adaptive computation and machine learning series)* (second edition). Bradford Books.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44. https://doi.org/10.1007/bf00115009

Thorndike, E. L. (1911). Animal intelligence; experimental studies. *Classics in the History of Psychology*. https://doi.org/10.5962/bhl.title.55072

Wiering, M. A. (2001). Reinforcement Learning in Dynamic Environments using Instantiated Information. *International Conference on Machine Learning*, 585–592. https://dspace.library.uu.nl/bitstream/1874/20824/2/wiering_01_reinforcement.pdf