Autonomous Agents
Report Assignment 2: Single Agent Learning

Authors:

Agnes van Belle *(10363130)*,
Maaike Fleuren *(10350470)*,
Norbert Heijne *(10357769)*,
Lydia Mennes *(10333843)*

October 5, 2012

1 Introduction

This report has been written for the Master Artificial Intelligence course Autonomous Agents. This report will contain the answers, motivation and explanation for our implementations of the tasks we had to accomplish in our second assignment for this course. These tasks were centered around the topic of ‘Single Agent Learning’.

1.1 The environment

In all tasks there is assumed to be a grid world (of 11×11) with a predator and a prey in it. The agents can both move one tile forward each iteration. The direction they take (or if they move at all) is affected by probabilities (their policies). If they move over the edge of the grid they end up at the opposing side of the grid. The prey will never step onto the predator. We are focused on improving the decisions of one agent, the predator.

1.2 The state space representation

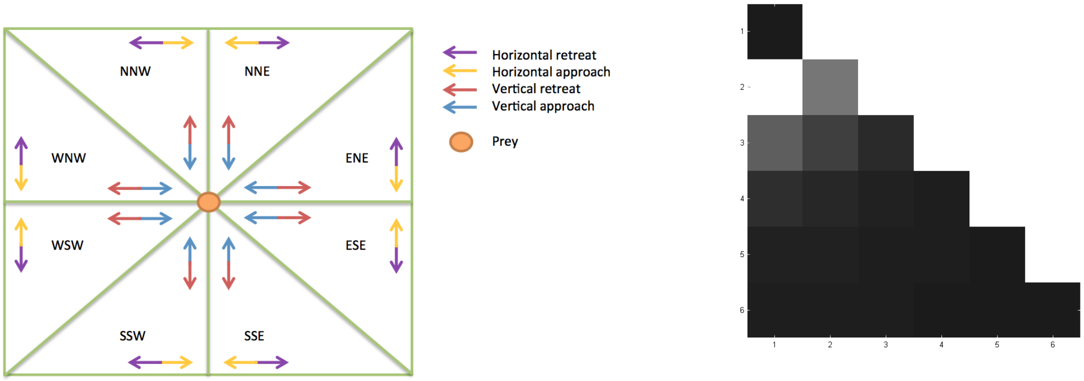
In the experiments described the first report, we initially used a state space that was an intuitive, yet cumbersome representation. We referred to that state space representation as the ‘default’ state space. The amount of states that was used in the default state space was $(11 \times 11) \times (11 \times 11) = 121 \times 121 = 14641$. We then changed the state space representation to a more efficient one, referred to as the ‘efficient’ state space, which led to a reduction of 697 times less states, resulting in just 21 different states.

In this assignment, we used this efficient state space representation for the learning algorithms. To give a good understanding of our learning algorithms, which were built on the efficient state space representation, we will once again explain how this representation works.

Figure **??** illustrates that there is a symmetry in the default state space, and thus that there were relatively much values redundantly computed. By using this symmetry in the default state space a much smaller state space was achieved.

Each state represents a distance between the prey and predator. These are represented in the lower left diagonal of a matrix, in which the x-axis is the relative horizontal distance in the MDP and the y-axis the relative vertical distance in the MDP. This matrix is shown in Figure **??**. Combinations of positions of prey and predator for which the horizontal and vertical distances are equal are now treated equivalent.

Also two combinations for which the horizontal distance in one equals the vertical distance in the other and vice versa are considered equal. In order to navigate through this state space different actions are required. These are: *horizontal retreat, horizontal approach, vertical retreat, vertical approach*, as illustrated in Figure **??**, and of course the action *wait*. When interacting with the environment these actions are converted into corresponding actions in the real world. This only requires the relative direction of the prey (which is always located at the centre, regardless of its coordinates) with respect to the predator. This is computed by using the difference in location of the prey and predator on the x- and y-axis.



(a) The 11×11 grid divided into eight symmetric pieces, with the corresponding possible moves which are also symmetric.

(b) Colormap of V-values, the brighter the color the higher the corresponding V-value. The prey is always located on the (1, 1) coordinate in this state representation.

Figure 1: Illustration of the symmetry and corresponding values of the new state space representation

1.3 Implementation details

This report will not be about our exact code and implementation details. However, a class diagram of our code is provided in Appendix **??**.

2 Learning algorithms

As mentioned before, we will use the same environment as in the previous assignment. But in this assignment, we will assume the learning scenerio: the agent does not know the transition probabilities, nor the reward structure. On a very high level there are two ways to come to a good solution in this setting: learning the model, and do planning again (model based learning), or not learn the model, and directly try to learn a high-reward policy (model-free learning). In this assignment we will focus on the latter.[1]

2.1 (M) Q-Learning

Q-Learning is an off-policy temporal-difference control algorithm. Temporal-difference methods can learn directly from raw experience without a model of the environment's dynamics. Furthermore, it updates estimates based in part on other learned estimates, without waiting for a final outcome.[2]

While the distinguishing feature of on-policy methods is that they estimate the value of a policy while using it for control, these two functions are separated in off-policy methods. The behavior policy, used to generate behavior, and the estimation policy, that is evaluated and improved, may in fact be unrelated. This separation is an advantage because the estimation policy may be deterministic (e.g., greedy), while the behavior policy can continue to sample all possible actions.[3] Its simplest form, *one-step Q-learning*, is

[1] Roijers (2012) *Assignments Autonomous Agents* p. 6

[2] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 133

[3] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 126

defined by[4]:

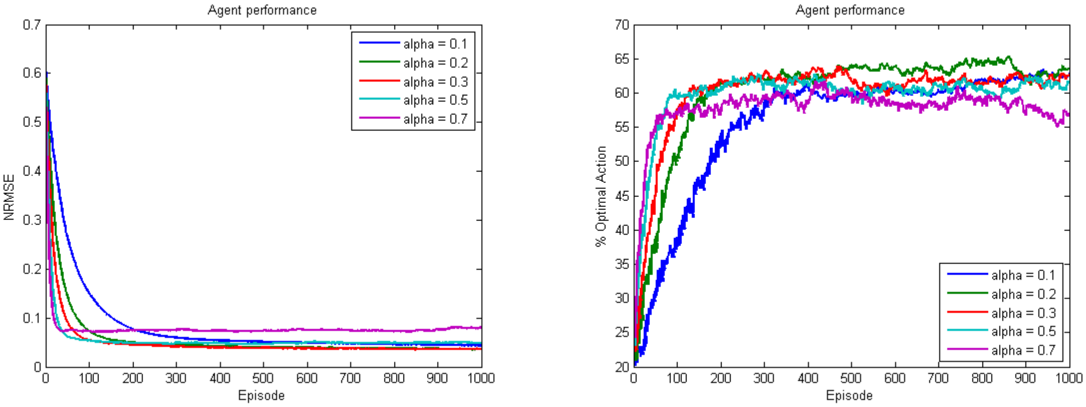
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

For this assignment, we implemented Q-learning and used it for our predator agent. We used ϵ-greedy action selection, which behaves greedily most of the time, but with probability ϵ, instead select an action at random, uniformly, independently of the action-value estimates.[5] In this case, we used $\epsilon = 0.1$. We initiated the values of our Q-learning table optimistically with a value of 15 for all cells in the table.

Figure **??** shows plots on the performance of the agent over time for different α (learning rate) and figure **??** for different γ (discount factor) using ϵ-greedy action selection.

A few observations can be made on these results. The values of α higher than 0.5 seem to have an adverse effect on the agent's performance for both the NRMSE (Normalized-Root-Mean-Squared-Error) and the percentage of optimal actions measures. As the α increases from 0.1 to 0.5, the performance increases over first 100 episodes for both measures, which is not the case for α=0.7 as seen in figure **??**.

The values of 0.1, 0.5 and 0.7 for γ seem to have an early advantage over the value 0.9 for γ in beginning of the runs as seen in figure **??**. This can be explained by the fact that a lower γ makes the state action pair values fluctuate much faster than a higher γ, so that early on the values might be better, but as the number of episodes increases, the values of the state action pairs drops further which in turn increases the NRMSE. However, looking at the percentage of optimal actions, the performance is on par with γ=0.9. Which means that even though the values might be low, the resulting policy would perform just as well.



(a) A comparison of the performance of the agent based on the resulting Normalized-Root-Mean-Squared-Error of the agent's state action pairs and the the estimated true state action pairs.

(b) A comparison of the performance of the agent based on the resulting percentage of best actions in the state space action pairs $\max_a Q(s_t, a_t)$ of the agent that are also present as the best actions in the estimated true state action pairs.

Figure 2: Plots on the performance of the agent over time for different values of the learning rate α using ϵ-greedy action selection. Averaged over 100 runs with 1000 episodes each with $\epsilon = 0.1$ and the discount factor $\gamma = 0.9$.

2.2 (M) Experiment with ϵ and optimistic initialization

The values chosen to evaluate were the combinations of a low initialization value (neutral initialization), a high initialization value (optimistic initialization), a low value for ϵ (close to greedy policy) and a high value for ϵ (exploratory/random policy). From these results we can make a few observations.

[4] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 148

[5] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 28

2.3 (SC) Softmax action selection instead of ϵ-greedy

For this exercise we will try to illustrate the difference between ϵ-greedy and softmax. These two methods are both for action selection and this will most likely have an impact on the way that the state action pairs are explored. ϵ-greedy will have a fixed chance of trying to explore a random action for the current state. For softmax the greedy action is still given the highest selection probability, but all the others are ranked and weighted according to their value estimates.[6]

Through the following plots in figure **??** we will show the effects of the different values of τ and ϵ on exploration. The performance of both algorithms seem to be on par with each other, and show very similar behavior. however Softmax explores more than ϵ-greedy, in both the beginning and later on except in the random policies as shown in figures **??** and **??**, and because of this exploratory property Softmax's model seems to converge faster with higher values of τ as show in figure **??** compared to figure **??**. Another thing to note is that for higher values of τ that which would be considered more a random policy than a greedy policy, the Softmax action selection visits all of the unique state action pairs sooner than the ϵ-greedy action selection.

2.4 Other ways to do learning

2.4.1 (SC) On-policy Monte Carlo Control

As mentioned in section **??**, on-policy methods try to evaluate or improve the policy that is used to make decisions.[7] Monte Carlo estimation can be used to approximate optimal policies, which results in Monte Carlo control. It proceeds according to the idea of generalized policy iteration, where one maintains both an approximate policy and an approximate value function. The value function is repeatedly altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved with respect to the current value function.[8] In this case, our policy moves towards a softmax policy, which was previously explained in section **??**. Furthermore, it is necessary to use either exploring starts or and ϵ-soft policy because making the policy greedy prevents further exploration of nongreedy actions. Since an assumption of exploring starts is an unlikely assumption an ϵ-soft policy is used in this case: softmax.

On Policy Monte Carlo Control differs from other methods in the fact that the agent does not learn until the end of an episode. Such an episode can be regarded as a sample which is then used to update the Q-values of the encountered state-action pairs. Learned information can therefore not be used until the next episode.

There are a number of parameters used in the implementation of this learning method. The first parameter is τ for the softmax policy, determining how much the agent will explore. The number of runs which determines how much the agent will learn. The initialization for the Q-values which is still optimistic and the discount factor λ.

How the number of steps needed to catch a prey changes over the number of runs for different parameter settings can be seen in figures **??** and **??**. The resulting policy can be found in Appendix **??**.

Explain the difference with other learning methods theoretically, and compare them using informative graphs.

2.4.2 (SC) Off-Policy Monte Carlo Control

Off-policy Monte Carlo control follows the behavior policy while learning about and improving the estimation policy, which makes it different from other learning methods.[9] The behavior policy is a ϵ-soft policy and therefore keeps exploring, while the estimation policy can be a deterministic greedy policy. In order to transfer the learned information from the behavior policy to the estimation policy importance sampling is

[6] Suttion, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 30

[7] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 122

[8] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 118

[9] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 126

used, since the probability of taking an action in one policy is not the same as in the other.

In this specific case the behavior policy is a softmax policy learned with the on-line Monte Carlo method, using 600 runs, a discount factor of 0.9 and τ is set to 0.9.

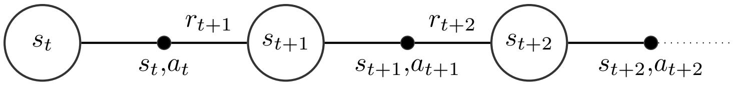
How the number of steps per episodes changes dusring learning in this implementation can be seen in figure **??**. The resulting policy can be found in Appendix **??**.

2.4.3 (SC) Sarsa

For an explanation of on-policy methods and temporal difference learning, please take a look at section **??** *or* **??***, and section* **??** *respectively.*

In contrast to the Q-learning method, Sarsa is an on-policy temporal-difference control algorithm. Just like Q-learning it learns an action-value function rather than a state-value function, but for an on-policy method like Sarsa we must estimate $Q^{\pi}(s, a)$ for the current behavior policy π and for all states s and actions a.[10]

In Sarsa we consider transitions from state-action pair to state-action pair, and learn the value of state-action pairs. An alternating sequence of of states and state-action pairs forms an episode:



After every transition from a nonterminal state s_t an update is done:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]$$

If s_{t+1} is terminal, then $Q(s_{t+1}, a_{t+1})$ is defined as zero. As you can see, this update uses every element that makes up a transition from one state-action pair to the next $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, from which the name Sarsa originates.

Explain the difference with other learning methods theoretically, and compare them using informative graphs.

3 Conclusion

[10] Sutton, Barto (1998) *Reinforcement Learning: An Introduction* Cambridge, Massachusetts: The MIT press. p. 145

Appendices

¡¡¡¡¡¡¡ HEAD

A Class Diagram

Appendices

B Policy of On Line Monte Carlo

The policy resulting from On Line Monte Carlo learning can be found in Table **??**

C Policy of Off Line Monte Carlo

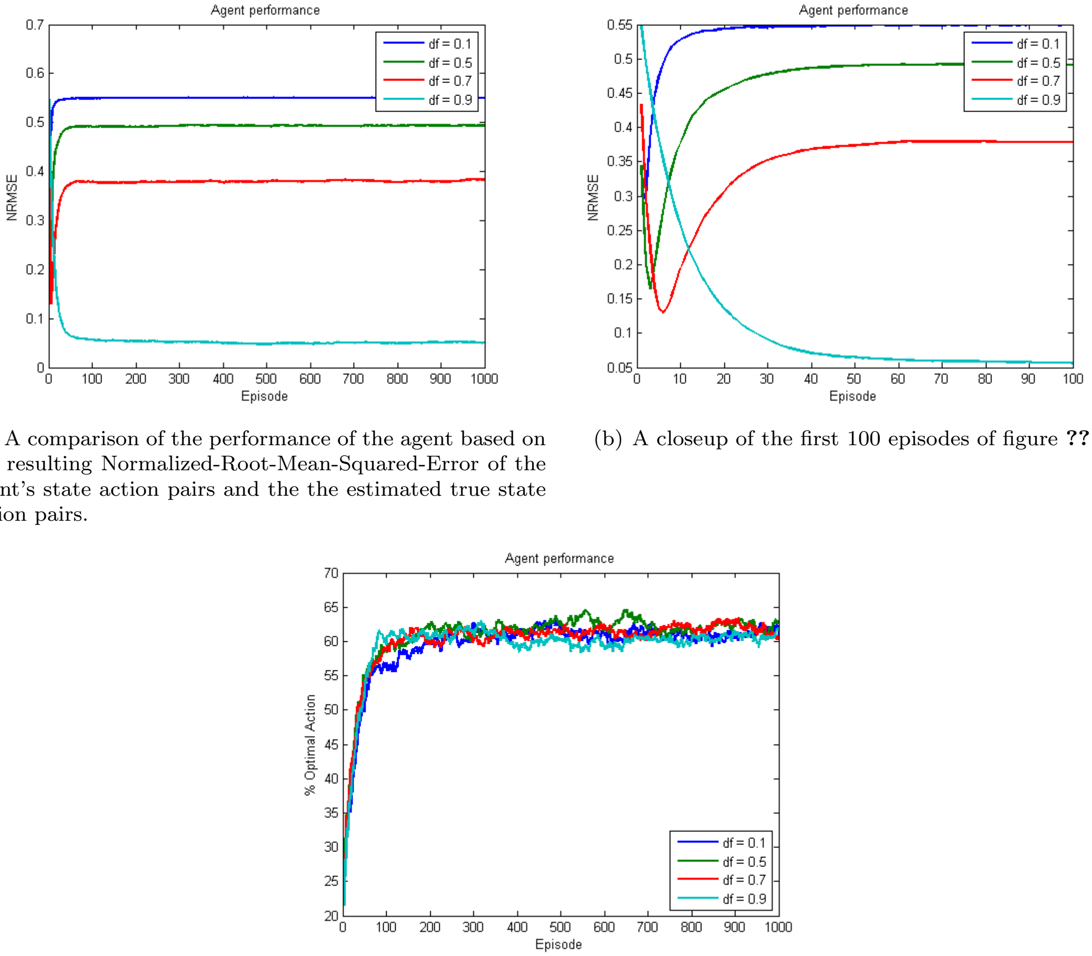
The policy resulting from Off Line Monte Carlo learning can be found in Table **??** =======

Table 1: The policy resulting from On Line Monte Carlo learning method with τ is 0.9 and discount factor is 0.9 after 500 episodes

	0	1	2	3	4	5	6	7	8	9	10
0	U 0,189 R 0,220 D 0,207 L 0,194 W 0,190	U 0,176 R 0,239 D 0,234 L 0,165 W 0,186	U 0,181 R 0,250 D 0,263 L 0,132 W 0,173	U 0,149 R 0,241 D 0,317 L 0,119 W 0,175	U 0,120 R 0,132 D 0,421 L 0,118 W 0,209	U 0,097 R 0,078 D 0,575 L 0,046 W 0,203	U 0,120 R 0,118 D 0,421 L 0,132 W 0,209	U 0,149 R 0,119 D 0,317 L 0,241 W 0,175	U 0,181 R 0,132 D 0,263 L 0,250 W 0,173	U 0,176 R 0,165 D 0,234 L 0,239 W 0,186	U 0,189 R 0,194 D 0,207 L 0,220 W 0,190
1	U 0,165 R 0,234 D 0,239 L 0,176 W 0,186	U 0,137 R 0,276 D 0,282 L 0,135 W 0,169	U 0,118 R 0,335 D 0,250 L 0,107 W 0,190	U 0,070 R 0,359 D 0,349 L 0,069 W 0,152	U 0,050 R 0,357 D 0,424 L 0,040 W 0,128	U 0,036 R 0,046 D 0,752 L 0,129 W 0,037	U 0,050 R 0,040 D 0,424 L 0,357 W 0,128	U 0,070 R 0,069 D 0,349 L 0,359 W 0,152	U 0,118 R 0,107 D 0,250 L 0,335 W 0,190	U 0,137 R 0,135 D 0,282 L 0,276 W 0,169	U 0,176 R 0,165 D 0,234 L 0,239 W 0,186
2	U 0,132 R 0,263 D 0,250 L 0,181 W 0,173	U 0,107 R 0,250 D 0,335 L 0,118 W 0,190	U 0,083 R 0,299 D 0,433 L 0,074 W 0,111	U 0,056 R 0,403 D 0,344 L 0,048 W 0,150	U 0,004 R 0,301 D 0,550 L 0,054 W 0,090	U 0,003 R 0,003 D 0,944 L 0,048 W 0,003	U 0,004 R 0,054 D 0,550 L 0,301 W 0,090	U 0,056 R 0,048 D 0,344 L 0,403 W 0,150	U 0,083 R 0,074 D 0,433 L 0,299 W 0,111	U 0,118 R 0,107 D 0,250 L 0,335 W 0,190	U 0,132 R 0,181 D 0,250 L 0,263 W 0,173
3	U 0,119 R 0,317 D 0,241 L 0,149 W 0,175	U 0,069 R 0,349 D 0,359 L 0,070 W 0,152	U 0,048 R 0,344 D 0,403 L 0,056 W 0,150	U 0,012 R 0,672 D 0,110 L 0,007 W 0,199	U 0,031 R 0,092 D 0,822 L 0,052 W 0,002	U 0,053 R 0,001 D 0,941 L 0,001 W 0,004	U 0,031 R 0,052 D 0,822 L 0,092 W 0,002	U 0,012 R 0,007 D 0,110 L 0,672 W 0,199	U 0,056 R 0,048 D 0,344 L 0,403 W 0,150	U 0,069 R 0,070 D 0,359 L 0,349 W 0,152	U 0,119 R 0,149 D 0,241 L 0,317 W 0,175
4	U 0,118 R 0,421 D 0,132 L 0,120 W 0,209	U 0,040 R 0,424 D 0,357 L 0,050 W 0,128	U 0,054 R 0,550 D 0,301 L 0,004 W 0,090	U 0,052 R 0,822 D 0,092 L 0,031 W 0,002	U 0,019 R 0,977 D 0,001 L 0,003 W 0,001	U 0,001 R 0,144 D 0,855 L 0,000 W 0,000	U 0,019 R 0,003 D 0,001 L 0,977 W 0,001	U 0,031 R 0,052 D 0,822 L 0,092 W 0,002	U 0,054 R 0,004 D 0,301 L 0,550 W 0,090	U 0,040 R 0,050 D 0,357 L 0,424 W 0,128	U 0,118 R 0,120 D 0,132 L 0,421 W 0,209
5	U 0,046 R 0,575 D 0,078 L 0,097 W 0,203	U 0,129 R 0,752 D 0,046 L 0,036 W 0,037	U 0,048 R 0,944 D 0,003 L 0,003 W 0,003	U 0,001 R 0,941 D 0,001 L 0,053 W 0,004	U 0,000 R 0,855 D 0,144 L 0,001 W 0,000	Prey	U 0,001 R 0,000 D 0,855 L 0,144 W 0,000	U 0,001 R 0,053 D 0,001 L 0,941 W 0,004	U 0,048 R 0,003 D 0,003 L 0,944 W 0,003	U 0,129 R 0,036 D 0,046 L 0,752 W 0,037	U 0,046 R 0,097 D 0,078 L 0,575 W 0,203
6	U 0,132 R 0,421 D 0,118 L 0,120 W 0,209	U 0,357 R 0,424 D 0,040 L 0,050 W 0,128	U 0,301 R 0,550 D 0,054 L 0,004 W 0,090	U 0,092 R 0,822 D 0,052 L 0,031 W 0,002	U 0,977 R 0,001 D 0,003 L 0,019 W 0,001	U 0,855 R 0,144 D 0,001 L 0,000 W 0,000	U 0,977 R 0,019 D 0,003 L 0,001 W 0,001	U 0,092 R 0,031 D 0,052 L 0,822 W 0,002	U 0,301 R 0,004 D 0,054 L 0,550 W 0,090	U 0,357 R 0,050 D 0,040 L 0,424 W 0,128	U 0,132 R 0,120 D 0,118 L 0,421 W 0,209
7	U 0,241 R 0,317 D 0,119 L 0,149 W 0,175	U 0,359 R 0,349 D 0,069 L 0,070 W 0,152	U 0,403 R 0,344 D 0,048 L 0,056 W 0,150	U 0,672 R 0,110 D 0,007 L 0,012 W 0,199	U 0,822 R 0,092 D 0,031 L 0,052 W 0,002	U 0,941 R 0,001 D 0,053 L 0,001 W 0,004	U 0,822 R 0,052 D 0,031 L 0,092 W 0,002	U 0,672 R 0,012 D 0,007 L 0,110 W 0,199	U 0,403 R 0,056 D 0,048 L 0,344 W 0,150	U 0,359 R 0,070 D 0,069 L 0,349 W 0,152	U 0,241 R 0,149 D 0,119 L 0,317 W 0,175
8	U 0,250 R 0,263 D 0,132 L 0,181 W 0,173	U 0,335 R 0,250 D 0,107 L 0,118 W 0,190	U 0,299 R 0,433 D 0,074 L 0,083 W 0,111	U 0,344 R 0,403 D 0,056 L 0,048 W 0,150	U 0,550 R 0,301 D 0,004 L 0,054 W 0,090	U 0,944 R 0,003 D 0,003 L 0,048 W 0,003	U 0,550 R 0,054 D 0,004 L 0,301 W 0,090	U 0,344 R 0,048 D 0,056 L 0,403 W 0,150	U 0,299 R 0,083 D 0,074 L 0,433 W 0,111	U 0,335 R 0,118 D 0,107 L 0,250 W 0,190	U 0,250 R 0,181 D 0,132 L 0,263 W 0,173
9	U 0,239 R 0,234 D 0,165 L 0,176 W 0,186	U 0,276 R 0,282 D 0,135 L 0,137 W 0,169	U 0,250 R 0,335 D 0,118 L 0,107 W 0,190	U 0,349 R 0,359 D 0,070 L 0,069 W 0,152	U 0,424 R 0,357 D 0,050 L 0,040 W 0,128	U 0,752 R 0,046 D 0,036 L 0,129 W 0,037	U 0,424 R 0,040 D 0,050 L 0,357 W 0,128	U 0,349 R 0,069 D 0,070 L 0,359 W 0,152	U 0,250 R 0,107 D 0,118 L 0,335 W 0,190	U 0,276 R 0,137 D 0,135 L 0,282 W 0,169	U 0,239 R 0,176 D 0,165 L 0,234 W 0,186
10	U 0,220 R 0,207 D 0,194 L 0,189 W 0,190	U 0,234 R 0,239 D 0,176 L 0,165 W 0,186	U 0,263 R 0,250 D 0,181 L 0,132 W 0,173	U 0,317 R 0,241 D 0,149 L 0,119 W 0,175	U 0,421 R 0,132 D 0,120 L 0,118 W 0,209	U 0,575 R 0,078 D 0,097 L 0,046 W 0,203	U 0,421 R 0,118 D 0,120 L 0,132 W 0,209	U 0,317 R 0,119 D 0,149 L 0,241 W 0,175	U 0,263 R 0,132 D 0,181 L 0,250 W 0,173	U 0,234 R 0,165 D 0,176 L 0,239 W 0,186	U 0,220 R 0,189 D 0,194 L 0,207 W 0,190

Table 2: The estimation policy resulting from Off Line Monte Carlo learning method after 150 episodes of learning. The behavior policy that is used is generated by On Line Monte Carlo learning with τ is 0.9 and discount factor is 0.9 after 600 episodes

	0	1	2	3	4	5	6	7	8	9	10
0	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000
1	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000
2	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000
3	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000
4	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000
5	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	Prey	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000
6	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000
7	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000
8	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 1,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000
9	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 0,000 W 1,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000
10	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 0,000 D 0,000 L 1,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 1,000 R 0,000 D 0,000 L 0,000 W 0,000	U 0,000 R 1,000 D 0,000 L 0,000 W 0,000

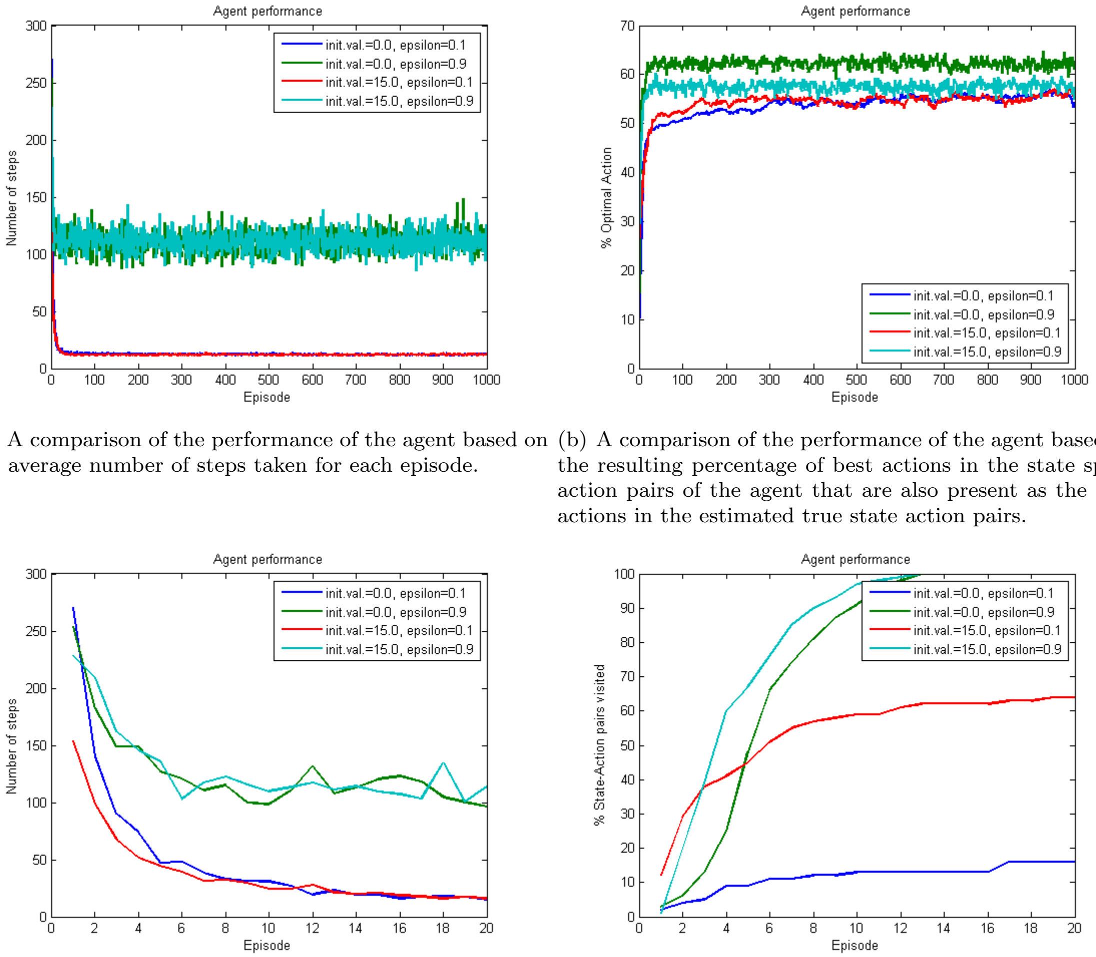


(a) A comparison of the performance of the agent based on the resulting Normalized-Root-Mean-Squared-Error of the agent's state action pairs and the the estimated true state action pairs.

(b) A closeup of the first 100 episodes of figure **??**.

(c) A comparison of the performance of the agent based on the resulting percentage of best actions in the state space action pairs of the agent that are also present as the best actions in the estimated true state action pairs.

Figure 3: Plots on the performance of the agent over time for different values of the discount factor γ using ϵ-greedy action selection. Averaged over 100 runs with 1000 episodes each with $\epsilon = 0.1$ and the learning rate $\alpha = 0.5$.



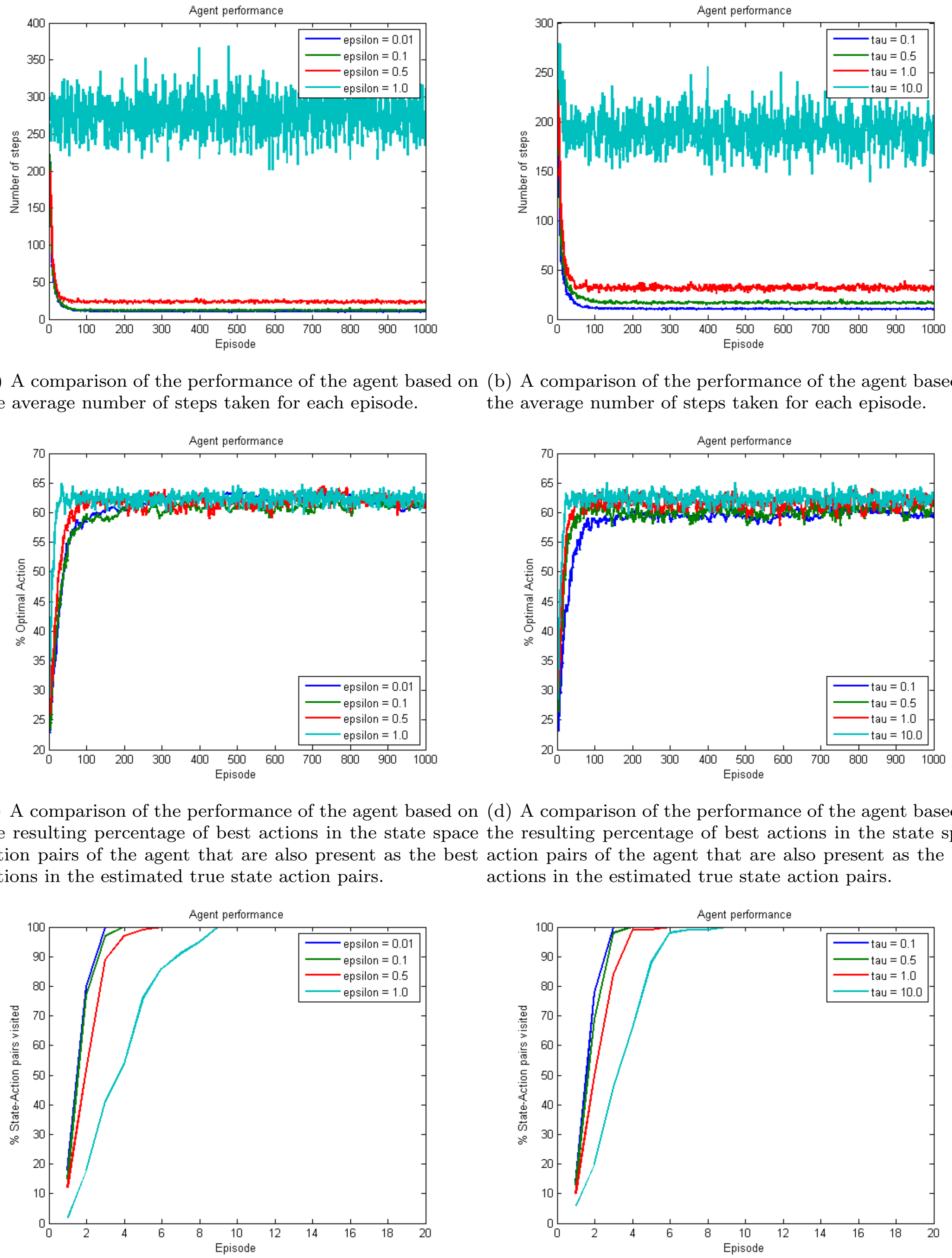
(a) A comparison of the performance of the agent based on the average number of steps taken for each episode.

(b) A comparison of the performance of the agent based on the resulting percentage of best actions in the state space action pairs of the agent that are also present as the best actions in the estimated true state action pairs.

(c) A closeup of the first 20 episodes of figure **??**.

(d) A comparison of the performance of the agent based on the percentage of state action pairs visited after each episode.

Figure 4: Plots on the performance of the agent over time for different values of ϵ and initialization values using ϵ-greedy action selection. Averaged over 100 runs with 1000 episodes each with $\gamma = 0.9$ and the learning rate $\alpha = 0.5$.



(a) A comparison of the performance of the agent based on the average number of steps taken for each episode.

(b) A comparison of the performance of the agent based on the average number of steps taken for each episode.

(c) A comparison of the performance of the agent based on the resulting percentage of best actions in the state space action pairs of the agent that are also present as the best actions in the estimated true state action pairs.

(d) A comparison of the performance of the agent based on the resulting percentage of best actions in the state space action pairs of the agent that are also present as the best actions in the estimated true state action pairs.

(e) A comparison of the performance of the agent based on the percentage of state action pairs visited after each episode.

(f) A comparison of the performance of the agent based on the percentage of state action pairs visited after each episode.

Figure 5: Plots on the performance of the agent over time for different values of ϵ using ϵ-greedy action selection (figures **??**, **??** and **??**) and for different values of τ using the Softmax action selection (figures **??**, **??** and **??**). Averaged over 100 runs with 1000 episodes each with $\gamma = 0.9$, the learning rate $\alpha = 0.5$ and the initialization value at 15.

Figure 6: Number of steps for each episode as the agent learns for different values of the discount factor. The values are averaged over 50 trials where the agent learns for 100 runs. The value of τ is set to 0.9

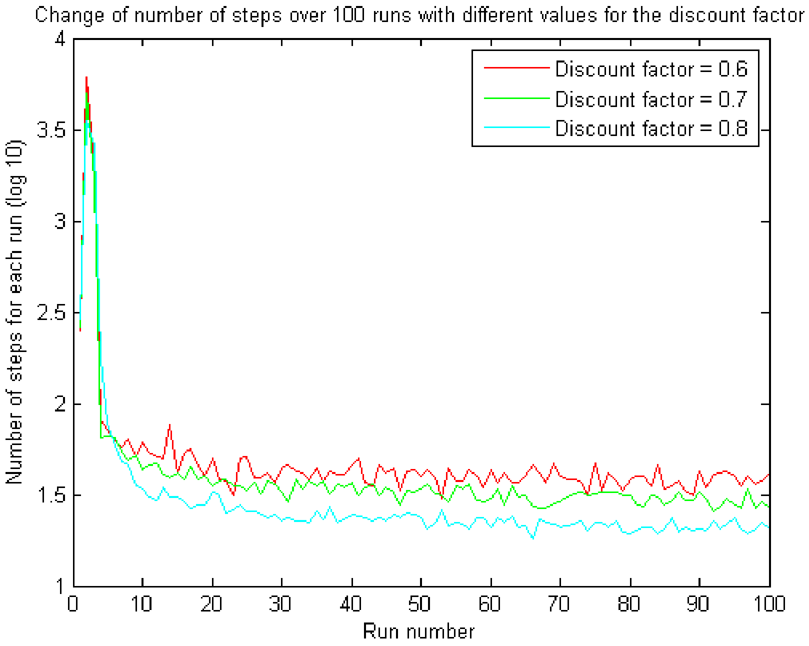


Figure 7: Number of steps for each episode as the agent learns for different values of τ. The values are averaged over 50 trials where the agent learns for 100 runs. The value of the discount factor is set to 0.8

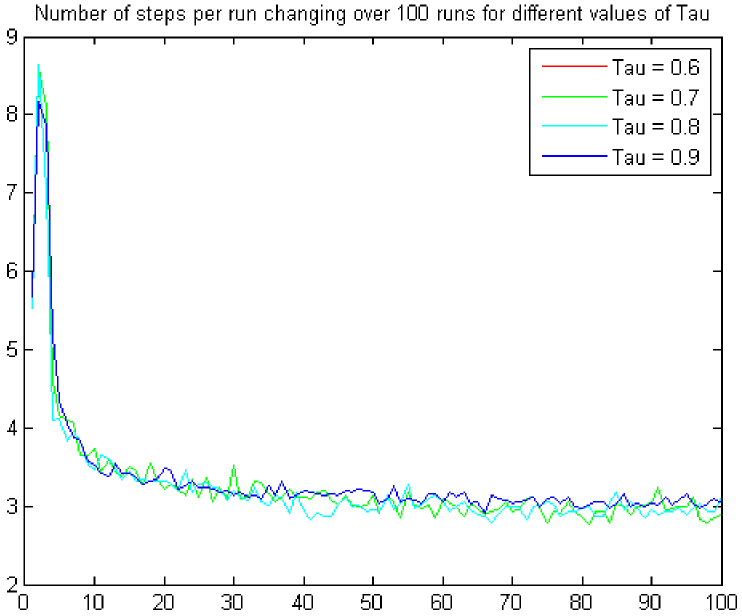
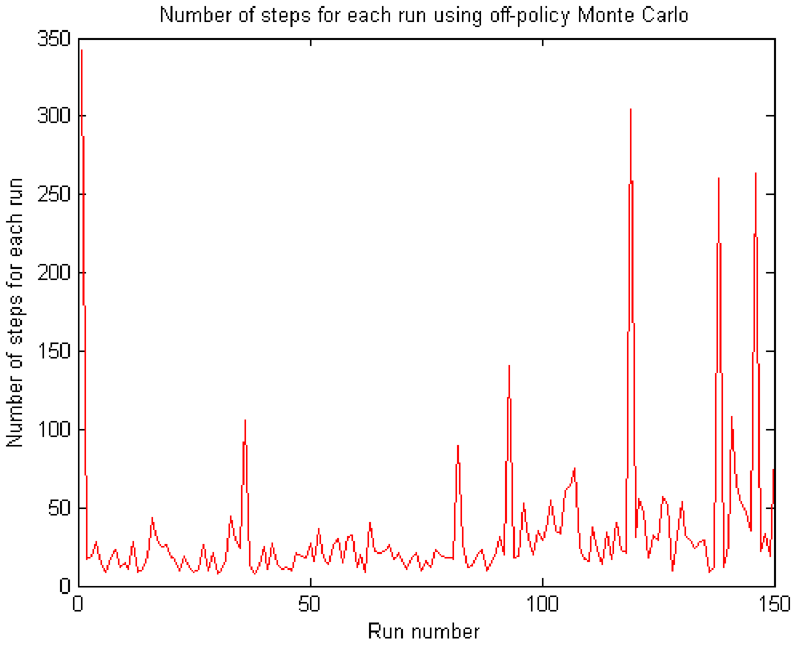


Figure 8: The number of steps for each episode averaged over 5 trials where the agent learns for 150 runs



D Class diagram

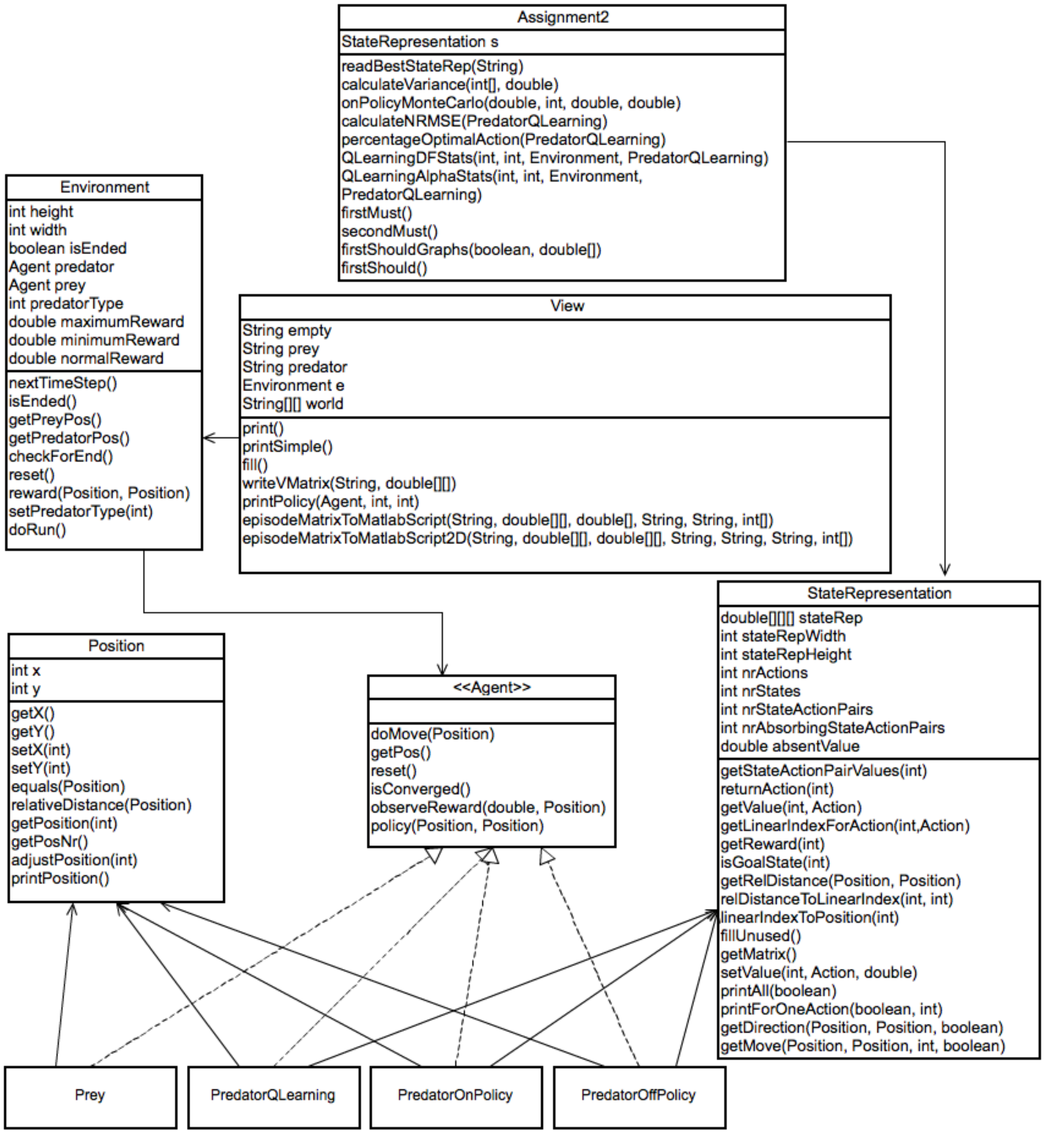

Assignment2
StateRepresentation s
readBestStateRep(String)
calculateVariance(int[], double)
onPolicyMonteCarlo(double, int, double, double)
calculateNRMSE(PredatorQLearning)
percentageOptimalAction(PredatorQLearning)
QLearningDFStats(int, int, Environment, PredatorQLearning)
QLearningAlphaStats(int, int, Environment, PredatorQLearning)
firstMust()
secondMust()
firstShouldGraphs(boolean, double[])
firstShould()
Environment
int height
int width
boolean isEnded
Agent predator
Agent prey
int predatorType
double maximumReward
double minimumReward
double normalReward
nextTimeStep()
isEnded()
getPreyPos()
getPredatorPos()
checkForEnd()
reset()
reward(Position, Position)
setPredatorType(int)
doRun()
View
String empty
String prey
String predator
Environment e
String[][] world
print()
printSimple()
fill()
writeVMatrix(String, double[][])
printPolicy(Agent, int, int)
episodeMatrixToMatlabScript(String, double[][], double[], String, String, int[])
episodeMatrixToMatlabScript2D(String, double[][], double[][], String, String, String, int[])
Position
int x
int y
getX()
getY()
setX(int)
setY(int)
equals(Position)
relativeDistance(Position)
getPosition(int)
getPosNr()
adjustPosition(int)
printPosition()
<<Agent>>
doMove(Position)
getPos()
reset()
isConverged()
observeReward(double, Position)
policy(Position, Position)
StateRepresentation
double[][][] stateRep
int stateRepWidth
int stateRepHeight
int nrActions
int nrStates
int nrStateActionPairs
int nrAbsorbingStateActionPairs
double absentValue
getStateActionPairValues(int)
returnAction(int)
getValue(int, Action)
getLinearIndexForAction(int,Action)
getReward(int)
isGoalState(int)
getRelDistance(Position, Position)
relDistanceToLinearIndex(int, int)
linearIndexToPosition(int)
fillUnused()
getMatrix()
setValue(int, Action, double)
printAll(boolean)
printForOneAction(boolean, int)
getDirection(Position, Position, boolean)
getMove(Position, Position, int, boolean)
Prey
PredatorQLearning
PredatorOnPolicy
PredatorOffPolicy

Figure 9: A class diagram of our code. For clarity purposes, not all methods and attributes of the classes are shown.

¿¿¿¿¿¿¿ klassediagram erbij