Autonomous Agents Assignment 2

Tobias Stahl 10528199
Spyros Michaelides 10523316
Ioannis Giounous Aivalis 10524851
Francesco Stablum 6200982

October 2, 2013

1 Introduction

This report is based on the implementation of an assignment exercise in which a predator is trying to catch a prey in a 2-dimensional environment, being unknown to the predator. This will be attempted using Temporal Difference learning methods (TD), a combination of Monte Carlo and dynamic programming. TD learning methods can learn directly from experience, rather than having to rely on a model of the environment. TD methods can update action state values estimates based on other learned estimates, without having to wait for the final outcome.

2 Algorithms

In this part of the report the used algorithms are introduced on the basis of their Pseudocode description.

2.1 Q-Learning

The initial exercise of this lab assignment is to implement Q-learning, a temporal-difference (TD) learning algorithm. This algorithm is to be used by the predator agent to catch the prey.
Q-learning is an off-policy TD control algorithm. An off-policy TD algorithm is one in which the estimated value functions can be updated using hypothetical actions, without having actually executed the actions themselves. Using this approach the algorithm can separate exploration from control, meaning the agent could learn through the environment without

necessarily having had the explicit experience. The steps involved can be summarised into the following Algorithm 1

Algorithm 1 Q-learning

Initialise $Q(s, a)$ arbitrarily
for all Episodes **do**
  Initialise s
  for all Steps of episodes **do**
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$;
  end for
  Until s is terminal
end for

Where:

- α is the learning rate. Setting it to a high value will force learning to occur faster, whereas for a low value it will occur slower.
- $max_{a'}$ is the maximum reward reachable in the state s'
- γ is the value which gives future rewards less worth than immediate ones

2.2 Q-learning using ϵ-greedy

When the Q-learning algorithm to selects an action, there needs to be some form of trade-off between selecting the action with the highest estimated reward so far, and the the rest of the actions available. Limiting the action selection policy to only the best action learnt so far, would mean potentially losing out on a better action in the future, being in a given state. To satisfy this trade-off, ϵ-greedy policy uses a (in this example a small) probability of ϵ to select randomly between all between all the actions available in a given state, excluding the most optimal one so far. In turn, (as in this scenario ϵ is small), the most optimal action is chosen with a much larger probability,

$1-\epsilon$, most of the time, giving the policy a tendency to exploit the best action so far most of the time, but not lose out on potentially better actions, which could be found through exploration of the other actions.

Based on the task at hand, the predator should directly learn a high reward policy without learning a model, since the agent is not supposed to know not know the transition probabilities, nor the reward structure of the environment. It is assumed that convergence should occur as long as all state action pair values continue to be updated using a certain policy (in this case ϵ-greedy.

2.3 Softmax Action-Selection Policy

In this section, a different action selection policy will be used in Q-learning instead of ϵ-greedy. ϵ-greedy policy satisfies the exploration/exploitation variance which is desired to be used in the Q-learning algorithm, although the way in which it achieves this could be a disadvantage in scenarios where the least favourable action has a much worse pay-off than the e.g., the second-best one. When the algorithm explores with a probability ϵ, it does not do this by taking into consideration the performance of the individual non-best actions themselves. The probability distribution between which non-best action is selected, is uniformly distributed and therefore, each one is as likely to be chosen as the rest. To optimise the performance of the Q-learning algorithm, it could be more efficient to make the selection among the non-best actions by weighing them according to their action-value estimates, thus increasing probability of selection for the higher action-value estimates, and hence making a more guided (by value) exploration. Algorithm 2 differs from ϵ-greedy in Operation 5.
Where:

- τ is a positive parameter called *temperature*. A high temperature leads the actions to be almost equiproble, low tempreature values cause a greater difference with $\tau \rightarrow 0$ beeing the same as greedy action selection.

2.4 Sarsa

Sarsa, like Q-learning is a temporal difference algorithm, meaning that it compares temporally successive predictions. Unlike Q-learning though,

Algorithm 2 Softmax

1: Initialise $Q(s, a)$ arbitrarily
2: **for all** Episodes **do**
3: Initialise s
4: **for all** Steps of episodes **do**
5: Choose Choose a with probability $\frac{\exp^{Q_t(a)/\tau}}{\sum \exp^{Q_t(b)/\tau}}$
6: Take action a, observe r, s'
7: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$
8: $s \leftarrow s'$;
9: **end for**
10: Until s is terminal
11: **end for**

Sarsa is an on-policy TD method. In on-policy TD learning the algorithm learns the value of the policy that is used to make the decisions, meaning directly through experience. This is in contrast to off-policy where value functions are not updated solely on experienced actions. The action selection policies previously discussed in Q-learning are also applicable for use in Sarsa. Again, there is the choice of specifying the trade-off between exploitation/exploration by setting the ϵ parameter when using ϵ-greedy, or using the softmax policy.

Algorithm 3 Sarsa

Initialise $Q(s, a)$ arbitrarily
for all Episodes **do**
Initialise s
Choose a from s using derived from Q (e.g., ϵ-greedy)
for all Steps of episodes **do**
Take action a, observe r, s'
Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
$s \leftarrow s'$; $a \leftarrow a'$;
end for
Until s is terminal
end for

3 Experiments

This section describes the properties of the system the experiments were tested on and give an overview of the achieved results including plots to visualize them.

3.1 System Properties

The experiments were performend on a ...

3.2 Experiment 1

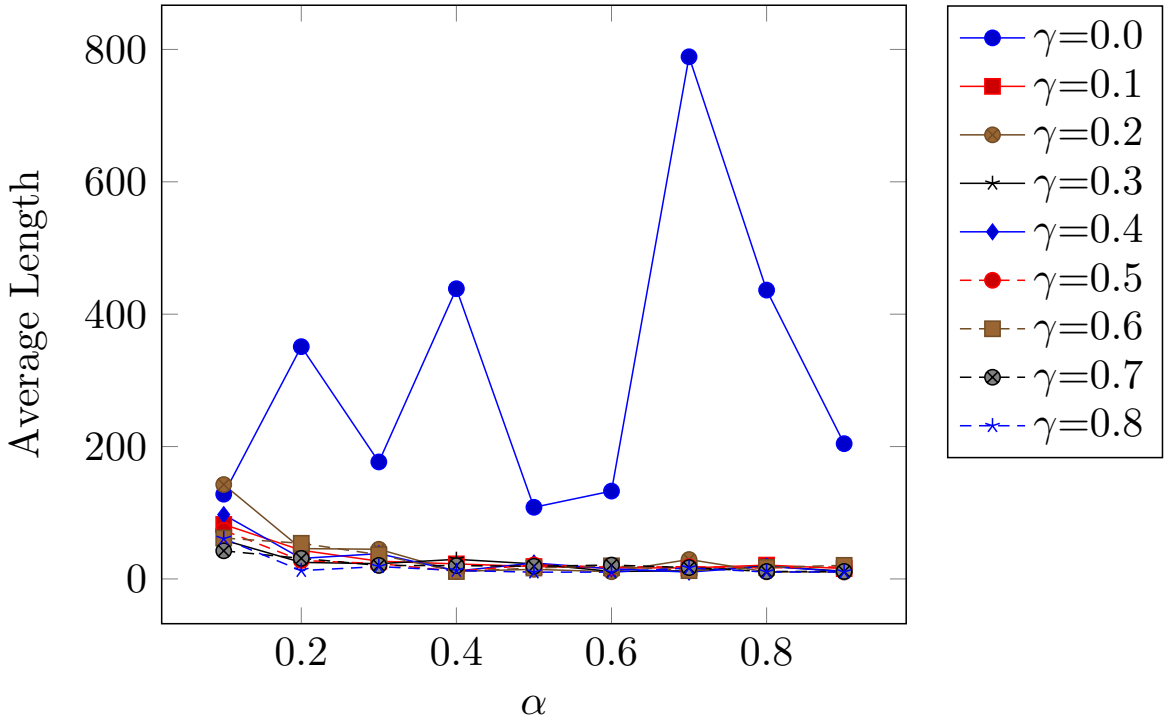
3.2.1 Hypotheses

The first experiment aims to measure the performance of the predator catching the predator with different learning rates α and different discount factors γ. Therefor the average performance of 100000 simulations for each α and γ is taken into account. This number is chosen, since a high number of simulations ensures higher precison. The reduced state space with 11x11 states is chosen, in order to save computational time. The predator learns for an episode count of 10000, before the simulations are executed.
The expected outcome is that high learning rates tend to always replace the current value with the new estimates and converge quickly, while a small learning rate value leads to a slow convergence and seems to trust the current estimate. [2]

	Discount factor γ								
α	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**
0.1	128.0169	82.3094	142.6086	58.6978	97.1353	73.4194	61.7706	42.4856	60.8453
0.2	350.9369	43.3417	45.8161	24.9713	31.0176	27.8658	53.8418	31.0789	12.5948
0.3	176.7514	26.7957	44.8313	23.0337	38.1784	23.1881	36.6555	20.3898	18.6524
0.4	438.4551	22.7017	11.4753	29.597	11.9084	11.5321	11.2164	20.088	12.6738
0.5	108.2213	18.7378	14.4834	22.7681	24.034	21.244	16.4981	19.1462	10.2268
0.6	132.7185	16.7787	10.8735	11.2989	15.3218	17.7085	19.8306	20.9615	10.315
0.7	788.898	16.5484	29.3294	12.2979	10.2584	18.4839	12.5068	17.1235	16.6028
0.8	436.3914	20.8921	10.7558	19.3525	19.0	16.6532	17.4	11.2239	10.4383
0.9	204.3642	15.7251	10.4158	11.5867	11.1238	18.2433	20.532	11.292	10.4699

Table 1: Average length of episode, the predator needs to catch the prey with different learning rates α and discount factors γ

3.2.2 Results



3.2.3 Interpretation

3.2.4

3.3 Experiment 2

3.3.1 Hypotheses

The second experiment observes the impact of ϵ in the ϵ-greedy action selection policy and the initialization of the Q-table. ϵ determines the exploration rate, with values close to zero favouring exploitation and values close to 1 prefering exploration.
In order to test this, the constant learning rate = 0.8 and the constant discount factor = 0.9 are chosen, since these values had the best performance in the previous experiment. The learning episode count is 1000 episodes and the number of simulations is 100.
In the previous experiment the Q-Table was initialized optimistically, with values higher than the actual reward to receive, which encourages an early exploration, since any actual reward is less than the actual reward. Changing this value to a pessimistically initalization on the other hand inspires exploitation.

3.3.2 Results

	Initial Q-Value									
ϵ	**30**	**25**	**20**	**15**	**10**	**5**	**0**	**-5**	**-10**	**-15**
0.0	18.97	11.37	10.62	10.33	11.48	10.29	11.65	11.62	11.46	11.9
0.1	11.03	10.34	10.52	10.56	11.46	11.05	13.22	11.8	11.62	11.3
0.2	12.09	10.25	12.25	11.48	11.51	10.48	12.94	11.53	16.42	11.85
0.3	10.42	11.32	11.03	10.34	11.18	11.89	13.8	11.5	12.08	12.24
0.4	11.58	10.37	11.07	11.46	11.51	21.99	11.8	11.67	11.59	20.06
0.5	10.55	11.16	10.51	10.46	27.92	11.63	16.61	11.87	12.97	20.32
0.6	10.46	12.54	10.22	11.47	10.33	10.19	11.03	40.66	11.28	11.67
0.7	10.63	10.13	10.42	11.42	10.26	12.25	11.87	10.61	11.76	11.67
0.8	10.68	10.72	15.35	10.38	11.48	10.41	11.51	27.56	16.68	11.81
0.9	10.39	10.21	13.31	10.6	11.31	10.38	11.19	11.47	11.41	22.84

Table 2: Average length of episode, the predator needs to catch the prey, with different exploration rate ϵ and initial Q-Values

3.3.3 Interpretation

3.3.4

3.4 Experiment 3

3.4.1 Hypotheses

In this experiment the difference between the ϵ-greedy and the softmax action selection policy is researched. The ϵ-greedy action selection chooses all actions, except the action with the highest estimated reward so far, with the same probability, while the softmax approach weights the other actions and chooses actions with more pay-off with a higher probability.
Assuming that

3.4.2 Results

3.4.3 Interpretation

3.4.4

3.5 Experiment 4

3.5.1 Hypotheses

3.5.2 Results

3.5.3 Interpretation

3.5.4

4 Conclusion

References

[1] Richard S. Sutton and Andrew G. Barto , *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts

[2] Eyal Even-Dar and Yishay Mansour , *Learning Rates for Q-learning*. Journal of Machine Learning Research 5 (2003)