Autonomous Agents 1
Assignment 1

By Group 4: Gieske, Gornishka, Loor, Radscum

November 8, 2014

Introduction

This report discusses a predator versus prey Markov Decision Process (MDP). In order to analyze this MDP, it was implemented. This MDP consists of an 11×11 toroidal grid. The predator and the prey are placed on the grid, after which the predator must catch the prey. Both can move vertically and horizontally across the grid as well as stay put until the next time step. Describing the movements about the grid as North, East, South, West and Wait, the policies of the predator and the prey are as follows:

	North	East	South	West	Wait
Predator	0.2	0.2	0.2	0.2	0.2
Prey	0.05	0.05	0.05	0.05	0.8

The predator and prey move about on the grid as specified but the policy. However, the prey does not move towards the predator. After the prey is caught, the episode ends and the game is reverted to starting positions. Catching the prey gives a reward of 10, 0 otherwise.

This implementation contains an execution of the game, policy evaluation, policy iteration and value iteration. The performance of these functions are analyzed in order to research the behaviour of the agents. The results of these functions are also compared with one another as part of analyzation.

Theory

Iterative policy evaluation

Iterative policy iteration is used compute the state-value function v_π for an arbitrary policy π. It is a stationary algorithm where the goal state and the arbitrary policy are static. In this case, it means that the goal state, the prey, remains on the same location. It uses the following algorithm as described in Barto and Sutton [source]:

Input π, the policy to be evaluated
Initialize an array v(s) = 0, for all s $\in S^+$
Repeat
$\Delta \leftarrow 0$
For each $s \in S$:
temp $\leftarrow$ v(s)
v(s) $\leftarrow \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v(s')]$
$\Delta \leftarrow max(\Delta, |temp - v(s)|)$
until $\Delta < \theta$(a small positive number)
Output v $\approx$ v(s)
Where:
$\pi(a|s)$ is an action chosen, given the state.
$p(s'|s,a)$ is a transition function.
$r(s,a,s')$ is a reward function.
$\gamma v(s')$ is the discounted state.

Policy improvement

Policy improvement is used to find an optimal, deterministic policy. This is, again a stationary function. Again, from Barto and Sutton [source]:

1. Initialization
   v(s) $n\mathcal{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all s $\in$ S
2. Policy evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in S$:
   temp $\leftarrow$ v(s)
   v(s) $\leftarrow \sum_{s'} p(s'|s,\pi(s))[r(s,a,s') + \gamma v(s')]$
   $\Delta \leftarrow max(\Delta, |temp - v(s)|)$
   until $\Delta < \theta$(a small positive number)
3. Policy improvement
   Policy stable $\leftarrow$ true
   For each s $\in$ S:
   temp $\leftarrow \pi(s)$
   v(s) $\leftarrow arg \max_a \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v(s')]$
   if $temp \neq \pi(s)$, policy stable $\leftarrow$ false

Value iteration

From Barto and Sutton [source]
Repeat
$\Delta \leftarrow 0$
For each s $\in$ S:

$temp \leftarrow v(s)$

$v(s) \leftarrow \max_a \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v(s')]$

$\Delta \leftarrow max(temp, |v(s)|)$

until $\Delta \leftarrow \theta$ (a small positive number)
Output a deterministic policy π, such that
$\pi(s) = arg \max_a \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v(s')]$

Implementation

The current implementation consists of the following classes:

Predator
This class implements the predator. It contains the policy and actions as described in the introduction.

Prey
This class implements the prey. It contains the policy and actions as described in the introduction.

Environment
This class implements the environment. This is a toroidal 11 $\times$ 11 grid. Objects, in this case the prey and predator, can be placed on this environment and move about.

Game
This class implements the game itself. It is possible to start the game. It also performs policy evaluation, policy iteration and value iteration.

Analysis

Simulator for the environment

Avarage run time	Standard deviation
296 rounds	286.580390118

Iterative policy evaluation

As described in the assignment, policy evaluation must be executed. Policy evaluation is a stationary algorithm which evaluates the agent's policy. The algorithm used is described in Barto and Sutton [source], section 4.1. By analyzing different cases for policy evaluation, the policy of the agent can be analyzed for improvement. It is expected that the policy evaluation values increase around the location of the prey. Therefore, if the agent moves in the direction of the increasing numbers on the grid, it will catch the prey. The locations of the predator are marked green and the locations of the prey are marked red to increase readability. The following cases have been analyzed, using a stationary prey:

Case	Predator	Prey
1	(0,0)	(5,5)
2	(2,3)	(5,4)
3	(2,10)	(10,0)
4	(10,10)	(0,0)

Starting with case 1, the following result is calculated:

Value grid in loop 32, Predator(0,0), Prey(5,5)											
Indices y\x	0	1	2	3	4	5	6	7	8	9	10
0	0.003357	0.005538	0.010435	0.018407	0.027837	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357
1	0.005538	0.009923	0.020607	0.040185	0.067041	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538
2	0.010435	0.020607	0.047865	0.105160	0.198928	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435
3	0.018407	0.040185	0.105160	0.265391	0.591645	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407
4	0.027837	0.067041	0.198928	0.591645	1.650667	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837
5	0.033106	0.085283	0.280822	0.991487	3.741537	2.850622	3.741537	0.991487	0.280822	0.085283	0.033106
6	0.027837	0.067041	0.198928	0.591645	1.650667	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837
7	0.018407	0.040185	0.105160	0.265391	0.591645	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407
8	0.010435	0.020607	0.047865	0.105160	0.198928	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435
9	0.005538	0.009923	0.020607	0.040185	0.067041	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538
10	0.003357	0.005538	0.010435	0.018407	0.027837	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357

In this case, it can be seen that moving in a diagonal direction gets the agent to the prey fastest. This takes ten steps. It can also be seen that this is the maximum distance between the predator and the prey.

Value grid in loop 32, Predator(2,3), Prey(5,4)											
Indices y\x	0	1	2	3	4	5	6	7	8	9	10
0	0.005538	0.010435	0.018407	0.027837	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357	0.003357
1	0.009923	0.020607	0.040185	0.067041	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538	0.005538
2	0.020607	0.047865	0.105160	0.198928	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435	0.010435
3	0.040185	0.105160	0.265391	0.591645	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407	0.018407
4	0.067041	0.198928	0.591645	1.650667	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837	0.027837
5	0.085283	0.280822	0.991487	3.741537	2.850622	3.741537	0.991487	0.280822	0.085283	0.033106	0.033106
6	0.067041	0.198928	0.591645	1.650667	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837	0.027837
7	0.040185	0.105160	0.265391	0.591645	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407	0.018407
8	0.020607	0.047865	0.105160	0.198928	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435	0.010435
9	0.009923	0.020607	0.040185	0.067041	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538	0.005538
10	0.005538	0.010435	0.018407	0.027837	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357	0.003357

With the agent starting at (2,3) and the prey located at (5,4), it will take the predator four steps to reach the prey.

Value grid in loop 32, Predator(2,10), Prey(10,0)											
Indices y\x	0	1	2	3	4	5	6	7	8	9	10
0	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837	0.027837	0.067041	0.198928	0.591645	1.650667
1	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407	0.018407	0.040185	0.105160	0.265391	0.591645
2	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435	0.010435	0.020607	0.047865	0.105160	0.198928
3	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538	0.005538	0.009923	0.020607	0.040185	0.067041
4	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357	0.003357	0.005538	0.010435	0.018407	0.027837
5	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357	0.003357	0.005538	0.010435	0.018407	0.027837
6	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538	0.005538	0.009923	0.020607	0.040185	0.067041
7	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435	0.010435	0.020607	0.047865	0.105160	0.198928
8	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407	0.018407	0.040185	0.105160	0.265391	0.591645
9	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837	0.027837	0.067041	0.198928	0.591645	1.650667
10	2.850622	3.741537	0.991487	0.280822	0.085283	0.033106	0.033106	0.085283	0.280822	0.991487	3.741537

With the agent starting at (2,10) and the prey at (10,0), it will take the agent four steps to reach the prey. This means that the distance between the predator and the prey is the same as in the previous case. The fact that the grid is toroidal makes this so.

Value grid in loop 32, Predator(10,10), Prey(0,0)											
Indices y\x	0	1	2	3	4	5	6	7	8	9	10
0	2.850622	3.741537	0.991487	0.280822	0.085283	0.033106	0.033106	0.085283	0.280822	0.991487	3.741537
1	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837	0.027837	0.067041	0.198928	0.591645	1.650667
2	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407	0.018407	0.040185	0.105160	0.265391	0.591645
3	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435	0.010435	0.020607	0.047865	0.105160	0.198928
4	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538	0.005538	0.009923	0.020607	0.040185	0.067041
5	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357	0.003357	0.005538	0.010435	0.018407	0.027837
6	0.033106	0.027837	0.018407	0.010435	0.005538	0.003357	0.003357	0.005538	0.010435	0.018407	0.027837
7	0.085283	0.067041	0.040185	0.020607	0.009923	0.005538	0.005538	0.009923	0.020607	0.040185	0.067041
8	0.280822	0.198928	0.105160	0.047865	0.020607	0.010435	0.010435	0.020607	0.047865	0.105160	0.198928
9	0.991487	0.591645	0.265391	0.105160	0.040185	0.018407	0.018407	0.040185	0.105160	0.265391	0.591645
10	3.741537	1.650667	0.591645	0.198928	0.067041	0.027837	0.027837	0.067041	0.198928	0.591645	1.650667

The agent starts at (10,10) and the prey is located at (0,0). It takes the agent two steps to catch the prey. Using the property that the grid is toroidal minimizes the distance between the agent and the prey. However, it is not possible for the agent to move diagonally in one step.

Predator	Prey	Value	Discount Factor	Iterations to converge
(0, 0)	(5, 5)	0.00335	0.8	33
(2, 3)	(5, 4)	0.19892	0.8	33
(2, 10)	(10, 0)	0.19892	0.8	33
(10, 10)	(0, 0)	1.65066	0.8	33

The table above proves that policy evaluation to converge always takes equally long for the same size of the grid. This makes sense, as the size of the grid has not changed.

Discount Factor	Iterations to converge
0.1	5
0.5	13
0.7	22
0.9	64

The discount factor appears to affect the number of iterations necessary to converge. This makes sense as the discount factor discounts the value of a state. A small discount value discounts the value of the state quite radically, leading to quick conversion. However, this quick conversion leaves many states with a random policy. This makes the convergence radical and most likely undesired. Using a higher discount value leads to more iterations before convergence. With a less radical discount, policy evaluation can be optimized in such a way that every state has a value. The discount factor should, however, not be too large. This will lead to faster convergence. Do note that in order to reach convergence, the discount factor must lie between 0-1.

Policy iteration

Policy iteration is a stationary algorithm to find the optimal policy. This algorithm exists of two steps: policy evaluation and policy iteration. Policy evaluation is demonstrated in the previous section. Policy iteration finds the optimal (deterministic) policy. The table below shows the results for policy iteration with the prey located at (5,5). As this algorithm first performs policy evaluation until convergence and then performs policy improvement, this algorithm is relatively slow and computationally expensive.

Policy Iteration Grid in loop 3, discount 0.8											
Indices y\x	0	1	2	3	4	5	6	7	8	9	10
0	3.7281 ES	4.6602 ES	5.8252 ES	7.2816 ES	9.1020 ES	11.3776 S	9.1020 WS	7.2816 WS	5.8252 WS	4.6602 WS	3.7281 WS
1	4.6602 ES	5.8252 ES	7.2816 ES	9.1020 ES	11.3776 ES	14.2220 S	11.3776 WS	9.1020 WS	7.2816 WS	5.8252 WS	4.6602 WS
2	5.8252 ES	7.2816 ES	9.1020 ES	11.3776 ES	14.2220 ES	17.7776 S	14.2220 WS	11.3776 WS	9.1020 WS	7.2816 WS	5.8252 WS
3	7.2816 ES	9.1020 ES	11.3776 ES	14.2220 ES	17.7776 ES	22.2220 S	17.7776 WS	14.2220 WS	11.3776 WS	9.1020 WS	7.2816 WS
4	9.1020 ES	11.3776 ES	14.2220 ES	17.7776 ES	22.2220 ES	27.7776 S	22.2220 WS	17.7776 WS	14.2220 WS	11.3776 WS	9.1020 WS
5	11.3776 E	14.2220 E	17.7776 E	22.2220 E	27.7776 E	22.2220 WENS	27.7776 W	22.2220 W	17.7776 W	14.2220 W	11.3776 W
6	9.1020 EN	11.3776 EN	14.2220 EN	17.7776 EN	22.2220 EN	27.7776 N	22.2220 WN	17.7776 WN	14.2220 WN	11.3776 WN	9.1020 WN
7	7.2816 EN	9.1020 EN	11.3776 EN	14.2220 EN	17.7776 EN	22.2220 N	17.7776 WN	14.2220 WN	11.3776 WN	9.1020 WN	7.2816 WN
8	5.8252 EN	7.2816 EN	9.1020 EN	11.3776 EN	14.2220 EN	17.7776 N	14.2220 WN	11.3776 WN	9.1020 WN	7.2816 WN	5.8252 WN
9	4.6602 EN	5.8252 EN	7.2816 EN	9.1020 EN	11.3776 EN	14.2220 N	11.3776 WN	9.1020 WN	7.2816 WN	5.8252 WN	4.6602 WN
10	3.7281 EN	4.6602 EN	5.8252 EN	7.2816 EN	9.1020 EN	11.3776 N	9.1020 WN	7.2816 WN	5.8252 WN	4.6602 WN	3.7281 WN

As described in the introduction, the agent can move North, South, East, West and Wait. To keep notation as clear and concise as possible, only the first letter of the optimal policy is printed. For clarity, the state 'Wait' is renamed to 'Hold' and is depicted as 'H' where applicable. Policy iteration shows that a state can make multiple optimal transitions. The optimal transitions all have the same probability of being chosen, while all other transition probabilities are set to zero. This creates an optimal, deterministic policy.

Predator	Prey	Value	Discount Factor	Iterations to converge
(0, 0)	(5, 5)	0.00335	0.8	2
(2, 3)	(5, 4)	0.19892	0.8	2
(2, 10)	(10, 0)	0.19892	0.8	2
(10, 10)	(0, 0)	1.65066	0.8	2

Discount Factor	Iterations to converge
0.1	2
0.5	2
0.7	2
0.9	2

Value iteration

Prey is located at (5, 5)

Value Iteration Grid in loop 8											
Indices y\x	0	1	2	3	4	5	6	7	8	9	10
0	0.000000	0.000000	0.000027	0.000168	0.001049	0.006554	0.001049	0.000168	0.000027	0.000000	0.000000
1	0.000000	0.000027	0.000168	0.001049	0.006554	0.040960	0.006554	0.001049	0.000168	0.000027	0.000000
2	0.000027	0.000168	0.001049	0.006554	0.040960	0.256000	0.040960	0.006554	0.001049	0.000168	0.000027
3	0.000168	0.001049	0.006554	0.040960	0.256000	1.600000	0.256000	0.040960	0.006554	0.001049	0.000168
4	0.001049	0.006554	0.040960	0.256000	1.600000	10.000000	1.600000	0.256000	0.040960	0.006554	0.001049
5	0.006554	0.040960	0.256000	1.600000	10.000000	0.000000	10.000000	1.600000	0.256000	0.040960	0.006554
6	0.001049	0.006554	0.040960	0.256000	1.600000	10.000000	1.600000	0.256000	0.040960	0.006554	0.001049
7	0.000168	0.001049	0.006554	0.040960	0.256000	1.600000	0.256000	0.040960	0.006554	0.001049	0.000168
8	0.000027	0.000168	0.001049	0.006554	0.040960	0.256000	0.040960	0.006554	0.001049	0.000168	0.000027
9	0.000000	0.000027	0.000168	0.001049	0.006554	0.040960	0.006554	0.001049	0.000168	0.000027	0.000000
10	0.000000	0.000000	0.000027	0.000168	0.001049	0.006554	0.001049	0.000168	0.000027	0.000000	0.000000

Discount Factor	Iterations to converge
0.1	1
0.5	7
0.7	7
0.9	8

Smarter state-space encoding

Conclusion

Files attached

Sources