Autonomous Agents

Assignment 1: Single Agent Planning 21 September, 2012

By:
Paris Mavromoustakos
Georgios Methenitis
Patrick de Kok
Marios Tzakris

Introduction

The purpose of this first assignment was to implement a reinforcement learning task, of which the problem satisfies the Markov property. Such a task is known as a Markov Decision Process (MDP). To achieve this we used one prey, which is part of the environment and behaves in a known probabilistic way, and one agent, the predator, whose goal is to catch the prey. When that happens the episode ends. The environment we used is an 11 by 11 toroidal grid. The task assumed that the entire MDP is known to our agent. The agent could thus determine the optimal policy even before interacting with the environment.

To implement the assignment we used the programming language Java. The results are gathered from a laptop with a Intel Core 2 CPU with both cores running at 1.83 GHz. The laptop has 2 GB of RAM memory.

Exercise 1

In the first part, we simulated the environment while keeping the policies of the predator and the prey separate. The environment has the ability to hold one or more predators and one or more prey. The predator starts from position $\langle 0, 0\rangle$ and the prey from $\langle 5, 5\rangle$. Both have an action set consisting of moving north, south, west, east, or staying stationary, respectively represented by the symbols $\uparrow$, $\downarrow$, $\leftarrow$, $\rightarrow$ and $\circ$.

The predator choses one of these actions randomly, where each has an equal chance to be picked. The prey choses $\circ$ with probability 0.8, and the rest of the actions are equally probable chosen, with a probability of 0.05. However, if the prey directly ends up in the same area as the predator, that action is disabled. In such a case, the probability for that action is set to 0, and the other moving actions get a probability of $0.2/3 = 0.0\bar{6}$.

Each episode of the simulation ends when the predator catches the prey. In the case that there are multiple prey and predators the episode ends when there is no living prey in the simulation environment. In this first assignment we held all of our experiments with one predator and one prey. In table 1 we present the overall statistics for this part of the assignment.

Exercise 2

In order to evaluate the random policy we implemented Policy Evaluation, computing the state-value function $V^\pi(S)$ and determined the value for each state. For every state $s \in S$, we iteratively compute

$$V^\pi(S) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')]$$

Table 1: Statistics of the simulation where both the predator and prey move randomly. The statistics are gathered over 100 runs.

Average steps:	279.7
Max no. of steps:	1350
Min no. of steps:	22
Mean over 100 runs:	279.7
Stdev over 100 runs:	255.1

This is done according the algorithm in Figure 4.1 of Sutton and Barto.

First, in the naive approach we computed the state-value function for every state in S_{11^4}. After Policy Evaluation with $\gamma = 0.8$ has converged, we got the following state-value combinations:

$$V^{\pi}(S)(\{\texttt{Predator}\,\langle 0,0\rangle\,,\texttt{Prey}\,\langle 5,5\rangle\}) = 0.0008$$
$$V^{\pi}(S)(\{\texttt{Predator}\,\langle 2,3\rangle\,,\texttt{Prey}\,\langle 5,4\rangle\}) = 0.8900$$
$$V^{\pi}(S)(\{\texttt{Predator}\,\langle 2,10\rangle\,,\texttt{Prey}\,\langle 10,0\rangle\}) = 0.8900$$
$$V^{\pi}(S)(\{\texttt{Predator}\,\langle 10,10\rangle\,,\texttt{Prey}\,\langle 0,0\rangle\}) = 0.8817$$

We have also implemented Policy Evaluation for the reduced state spaces S_{6^2} and S_{21}. The values for above states are for all state spaces the same. However, we noticed that the algorithm runs significantly faster. Table 2 presents the performance results we had performing this algorithm to smaller state spaces than the original one. We set the maximum error boundary (θ in the pseudocode) to zero in order to let our implementation produce more accurate results. This may cause an increase in our implementation's execution time.

Table 2: Policy Evaluation algorithm performance.

State Space	**D. Factor**	**Iterations**	**Time of Execution (ms.)**
S_{11^4}	0.9	202	245608
	0.7	73	94660
	0.5	46	57754
	0.1	23	29467
S_{6^2}	0.9	215	821
	0.7	72	369
	0.5	44	283
	0.1	16	114
S_{21}	0.9	215	475
	0.7	72	280
	0.5	41	195
	0.1	16	50

Exercise 3

When naively exploring the state space S, you will have to compute the value for all possible states, depending on four variables (the coordinates of both the predator and prey), each of which can have 11 different values. This results in $11^4 = 14641$ different states. Because of the toroidal structure of the world, the state $\{\texttt{Predator}\left\langle x,y\right\rangle, \texttt{Prey}\left\langle x',y'\right\rangle\}$ is equivalent to $\{\texttt{Predator}\left\langle x+a,y+b\right\rangle, \texttt{Prey}\left\langle x'+a,y'+b\right\rangle\}$; there is no special square in the given world which has features that distinguish it from other squares. From this follows that we can fix either x,y or x',y' to a certain value. By doing so, we have removed two degrees of freedom, and reduced the space to only $11^2 = 121$ unique states. Note that, when the position of the predator and prey are seen as vectors, this represents the difference vector $\texttt{Predator}\left\langle x,y\right\rangle - \texttt{Prey}\left\langle x',y'\right\rangle$ or $\texttt{Prey}\left\langle x',y'\right\rangle - \texttt{Predator}\left\langle x,y\right\rangle$, when respectively x',y' or x,y are fixed. We have chosen to fix x',y', and with $\texttt{P}_{11^2}\left\langle x,y\right\rangle$ we denote the equivalence class of states for which $\{\texttt{Predator}\left\langle x+a,y+b\right\rangle, \texttt{Prey}\left\langle a,b\right\rangle\}$ holds.

For the means of computing $V(\cdot)$, we can shrink the state space even more. Because of the symmetric nature of the world, the predator can catch the prey (and thus collect its reward) just as easily from any side. This means that the following values correspond:

$$\begin{aligned} V(\texttt{P}_{6^2}\left\langle x,y\right\rangle) &= V(\texttt{P}_{11^2}\left\langle x,y\right\rangle) \\ &= V(\texttt{P}_{11^2}\left\langle -x,y\right\rangle) \\ &= V(\texttt{P}_{11^2}\left\langle x,-y\right\rangle) \\ &= V(\texttt{P}_{11^2}\left\langle -x,-y\right\rangle) \\ &= V(\texttt{P}_{11^2}\left\langle x+6\lambda_1,y+6\lambda_2\right\rangle \end{aligned}$$

Through this, we only have to compute $V(\cdot)$ for $6^2 = 36$ different states. The equivalent states under $\texttt{P}_{6^2}$ are represented on an 11 by 11 grid in figure 1. The grid represents the space of $\texttt{P}_{11^2}$. A minimal, continuous area of this world is highlighted in red.

But even this representation contains symmetry. This symmetry has its roots in the fact the predator likes to know best; how far away am I from the prey, when I move to this new spot? Given $\texttt{P}_{11^2}\left\langle x,y\right\rangle$, it can compute its Euclidean distance towards the prey $d(\texttt{P}_{11^2}\left\langle x,y\right\rangle) = d(\texttt{P}_{6^2}\left\langle x,y\right\rangle) = \sqrt{x^2+y^2}$. For each move, it can recompute this distance, and based on this, it will assign a value. There are only $\sum_{i=1}^{6} i = 21$ different distances, as the Euclidean distance is symmetrical in x and y. The equivalent states under $\texttt{P}_{21}$ are represented on an 11 by 11 grid in figure 2. The grid represents the space of $\texttt{P}_{11^2}$. A minimal, continuous area of this world is highlighted in red.

We have implemented Policy Evaluation, Value Iteration and Policy Iteration for the naive, 11^4-sized state space, for the square 6^2-sized state space, and the triangular state space of 21 states. We refer to these spaces by S_{11^4}, S_{6^2} and S_{21} respectively.

Exercise 4

To find an optimal policy we computed the optimal value function V^* for all states. First of all, we had to find the value function for the naive state space S_{11^4}. For all $s \in S$,

$$V^\pi(S) = \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')]$$

Figure 3 presents the value table that was computed with the value iteration algorithm for a discount factor $\gamma = 0.7$ and $\theta = 0$. The policy always taking the action with the highest expected value is depicted in figure 4.

We also implemented the Value Iteration algorithm on the reduced state spaces S_{6^2} and S_{21}, acquiring the same values as above. However, we noticed that the algorithm runs significantly faster. Table 3 presents the performance results we had performing this algorithm to smaller state spaces than the original one. We set θ variable to zero in order our algorithm to produce more accurate results. This may cause an increase in the execution time of the Value Iteration algorithm.

Figure 1: Equivalent states under $\mathtt{P}_{62}$ are labeled with the same number. The predator is assumed to be at $\langle 0, 0\rangle$. The states indicate the position of the prey. A single repetition of this set is highlighted in red.

0	1	2	3	4	5	5	4	3	2	1
6	7	8	9	10	11	11	10	9	8	7
12	13	14	15	16	17	17	16	15	14	13
18	19	20	21	22	23	23	22	21	20	19
24	25	26	27	28	29	29	28	27	26	25
30	31	32	33	34	35	35	34	33	32	31
30	31	32	33	34	35	35	34	33	32	31
24	25	26	27	28	29	29	28	27	26	25
18	19	20	21	22	23	23	22	21	20	19
12	13	14	15	16	17	17	16	15	14	13
6	7	8	9	10	11	11	10	9	8	7

Figure 2: Equivalent states under $\mathtt{P}_{21}$ are labeled with the same number. The predator is assumed to be at $\langle 0, 0\rangle$. The states indicate the position of the prey. A single repetition of this set is highlighted in red.

0	1	2	3	4	5	5	4	3	2	1
1	6	7	8	9	10	10	9	8	7	6
2	7	11	12	13	14	14	13	12	11	7
3	8	12	15	16	17	17	16	15	12	8
4	9	13	16	18	19	19	18	16	13	9
5	10	14	17	19	20	20	19	17	14	10
5	10	14	17	19	20	20	19	17	14	10
4	9	13	16	18	19	19	18	16	13	9
3	8	12	15	16	17	17	16	15	12	8
2	7	11	12	13	14	14	13	12	11	7
1	6	7	8	9	10	10	9	8	7	6

Figure 3: The value function V^* generated by Value Iteration, with the predator fixed at $\langle 5, 5 \rangle$, discount factor $\gamma = 0.7$, and maximum error boundary $\theta = 0$. The terminal state is represented by the symbol ∞.

0.44	0.61	0.85	1.18	1.65	2.13	1.65	1.18	0.85	0.61	0.44
0.61	0.84	1.18	1.67	2.35	3.09	2.35	1.67	1.18	0.84	0.61
0.85	1.18	1.67	2.35	3.32	4.50	3.32	2.35	1.67	1.18	0.85
1.18	1.67	2.35	3.32	4.69	6.55	4.69	3.32	2.35	1.67	1.18
1.65	2.35	3.32	4.69	6.55	10.00	6.55	4.69	3.32	2.35	1.65
2.13	3.09	4.50	6.55	10.00	∞	10.00	6.55	4.50	3.09	2.13
1.65	2.35	3.32	4.69	6.55	10.00	6.55	4.69	3.32	2.35	1.65
1.18	1.67	2.35	3.32	4.69	6.55	4.69	3.32	2.35	1.67	1.18
0.85	1.18	1.67	2.35	3.32	4.50	3.32	2.35	1.67	1.18	0.85
0.61	0.84	1.18	1.67	2.35	3.09	2.35	1.67	1.18	0.84	0.61
0.44	0.61	0.85	1.18	1.65	2.13	1.65	1.18	0.85	0.61	0.44

Table 3: Value Iteration algorithm results.

State Space	**D. Factor**	**Iterations**	**Time of Execution (ms.)**
S_{11^4}	0.9	29	37774
	0.7	26	34423
	0.5	24	31848
	0.1	19	24883
S_{6^2}	0.9	22	229
	0.7	19	162
	0.5	16	126
	0.1	11	88
S_{21}	0.9	22	93
	0.7	19	70
	0.5	17	67
	0.1	11	60

Exercise 5

To iteratively improve the policy we implemented Policy Iteration $V^{\pi'}$ for all states in all three state representations, S_{11^4}, S_{6^2} and S_{21}. First, for all $s \in S$, we are doing a slightly modified version of Policy

Figure 4: The policy based on the state-value function. For every state, the policy choses either the action leading to the absorbing/terminal state (which has a transition probability of 1), or the action with the highest expected value. The highest expected value is computed by $\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$.

↓→	→	↓→	↓	↓	↓	↓	↓	←↓	←	←↓
↓	↓→	↓→	↓→	↓	↓	↓	←↓	←↓	←↓	↓
↓→	↓→	↓→	↓→	↓	↓	↓	←↓	←↓	←↓	←↓
→	↓→	↓→	↓→	↓→	↓	←↓	←↓	←↓	←↓	←
→	→	→	↓→	↓→	↓	←↓	←↓	←	←	←
→	→	→	→	→	○	←	←	←	←	←
→	→	→	↑→	↑→	↑	←↑	←↑	←	←	←
→	↑→	↑→	↑→	↑→	↑	←↑	←↑	←↑	←↑	←
↑→	↑→	↑→	↑→	↑	↑	↑	←↑	←↑	←↑	←↑
↑	↑→	↑→	↑→	↑	↑	↑	←↑	←↑	←↑	↑
↑→	→	↑→	↑	↑	↑	↑	↑	←↑	←	←↑

Table 4: Policy Iteration algorithm performance.

State Space	**D. Factor γ**	**Iterations**	**Execution Time (ms.)**
S_{11^4}	0.9	9	356222
	0.7	8	152240
	0.5	7	103164
	0.1	8	61771
S_{6^2}	0.9	7	6221
	0.7	6	2010
	0.5	6	1005
	0.1	5	353
S_{21}	0.9	5	3594
	0.7	5	1134
	0.5	5	581
	0.1	5	202

Evaluation. After that, Policy Improvement is executed. The procedure iterates with the same order until a stable policy for each state will be derived. A policy is stable when, after a number of iterations of this algorithm, the policy converges into one that is optimal and cannot be changed to the better. This is implemented as presented in Figure 4.3 of Sutton and Barto. Table 4 presents the result we had with our implementation of Policy Iteration, for different values for discount factor γ. The maximum error boundary θ was set at 0.

Conclusion

We have managed to reduce the state space from 11^4 states which should be processed separately to 11^2, 6^2 and $\sum_{i=1}^{6} i$ equivalence classes of states. We have implemented Policy Evaluation, Value Iteration and Policy Iteration for three of the four state space representations S_{11^4}, S_{6^2} and S_{21}. The runtime of all algorithms is heavily reduced by running on the smallest state space.

The algorithms provide all equivalent values for the value function, which is expected. This produces the same optimal policy, when maximizing over the value function.