Autonomous Agents
Report Assignment 2: Single Agent Learning

Authors:

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

October 3, 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

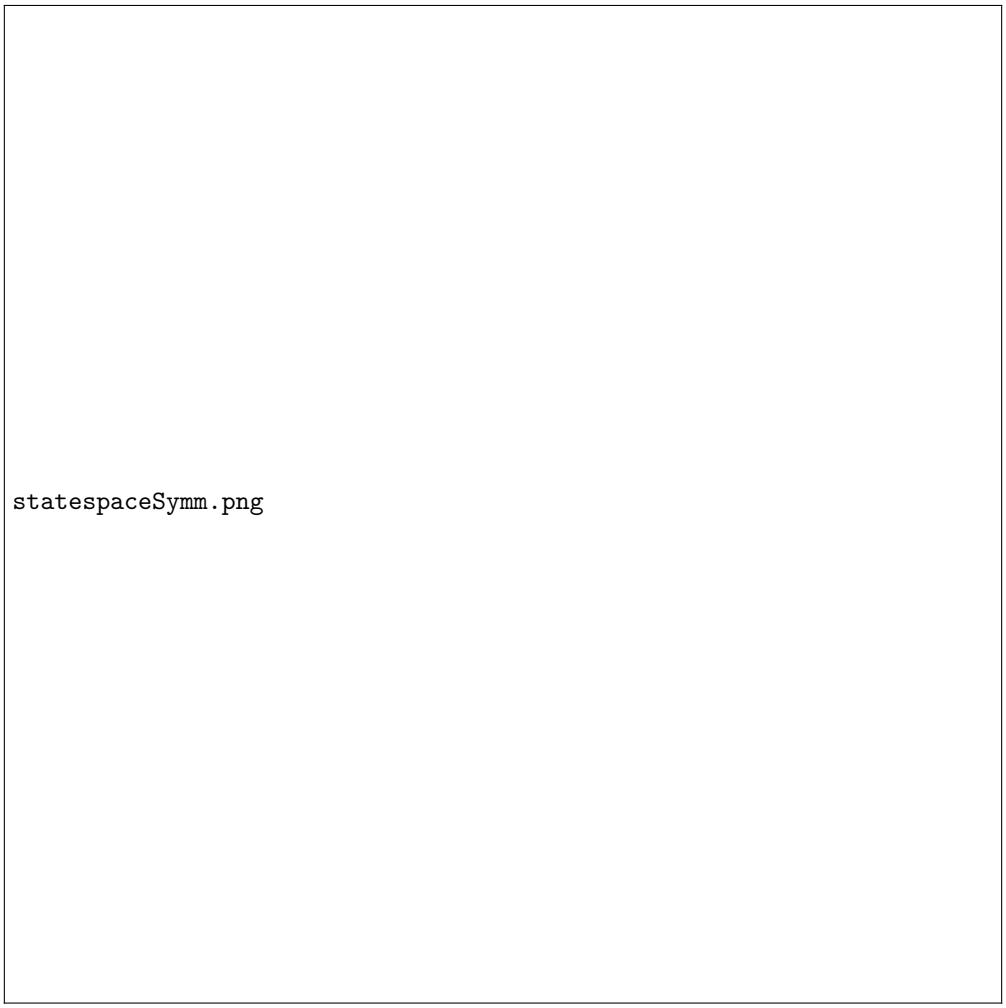

Figure 1: The 11×11 grid divided into eight symmetric pieces, with the corresponding possible moves which are also symmetric.

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: *approach horizontal, retreat horizontal, approach vertical, retreat vertical* and as before *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.

This reduction in the number of states has lead to output values which can be viewed in the appendix section **??**. The results show that the algorithm does not converge faster in terms of iterations with the efficient state space than with the default state space with $\theta = 0$ and $\gamma = 0.8$, the default state space required 111 iterations and the efficient state space required 107.

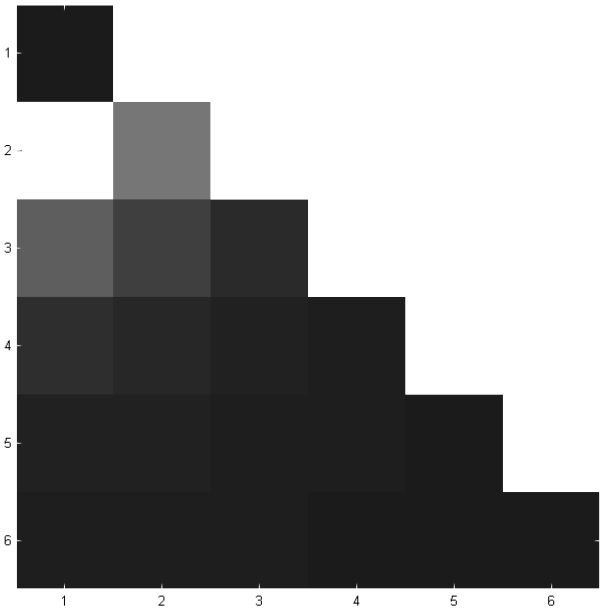

Figure 2: Colormap of the V-values resulting from Policy Evaluation for $\theta = 0$ and $\gamma = 0.8$
using the "efficient" state space representation. The brighter the color the higher the corresponding V-value. The prey is always located on the (1, 1) coordinate in this state representation.

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 **??**.

Appendices

A Class Diagram