Autonomous Agents

Assignment 3: Multi Agent Planning and Learning 19 October, 2012

By:
Paris Mavromoustakos
Georgios Methenitis
Patrick de Kok
Marios Tzakris

Introduction

In this last assignment, we added multiple predators to the environment, while making the prey intelligent, thus harder to catch. Our previous implementations already considered the prey to be an agent, for that reason we only added minor changes to our prey's functions, allowing it to use learning methods. The prey will now move equiprobably towards all directions, with a chance to trip (stay in the previous position), otherwise one predator would never be able to catch it.

What we also changed is that the prey and predator(s) move simultaneously, in a single timestep. That means, the agents can be next to eachother and swap positions, without considering their next move to be ”safe” or not.

Lastly, we now consider this implementation to be a zero-sum markov game, because the predators will receive a -10 reward if they crash into eachother, while the prey will be receiving a +10 reward. Beside that, the predators will receive +10 for catching the prey which gets -10 for getting caught.

Exercise 1

In the first excercise, we use the 11x11 grid where we add one prey and multiple predators. The program requests the number of predators as input from the user, initializes the prey at position (5,5) and puts the predators in random positions.

The agents then move randomly on the grid until two predators move into the same position (Collision) or a predator catches the prey (Catch). Those are the two possible and unique absorbing states. We should note that, if two predators collide and the prey is caught in the same timestep, the predators' collision is more ”important” and defines the reward distribution.

Table 1: Multiple Agents' Random Implementation

Number of Predators	Total No of Collisions	Total No of Catches	Average Step No until Catch
1	0	500	185
2	175	325	59
3	255	245	27
4	292	208	15

As we can see in the table above, adding more predators to this particular implementation does reduce the number of steps needed to catch the prey but dramatically increases the number of collisions between predators.

Exercise 2

In the second exercise, we made both prey and predators ”smarter” by implementing the Q-Learning algorithm on both agent types. That means that all the agents save and use a Q(s,a) table, from which they choose the next action using ϵ-greedy action selection.