Artificial Intelligence Laboratory 1: Agents

DT8012 (HT16) Halmstad University

3rd of November 2016

Introduction

This lab is intended to allow you to practice design and implementation of intelligent agents, in two different application domains.

- Mobile robot: collecting energy (red blocks).
- Simplified poker game player.

You will implement (very simple) agents of four different types:

- ***Random agent***: acts randomly, disregarding any external input information.
- ***Fixed agent***: performs a sequence of pre-specified actions over and over, disregarding any external input information.
- ***Reflex agent***: selects actions based on the agent's current perception of the world, but not based on past perceptions.
- ***Agent with memory***: The agent can remember the sensor input, as well as own actions, from the past and utilizes this information to make better decisions about how to act.

Task 1: Mobile robot exploration

Introduction

You will program a mobile robot, called 'Pioneer P3-DX', to explore a simple environment. This robot has two wheels, driven by independent motors. It is also equipped with 16 ultrasonic sensors and an 'energy sensor' (which provides the relative position to the nearest red block). Your task is to implement different kinds of agents, each trying to collect energy from the environment. If you feel ambitious, you can also try to ensure that the robot doesn't crash into walls, doesn't aimlessly wander in circles, etc.

Environment Setup

The following software tools are required for Lab 1. Make sure you have installed them and verified that they work properly.

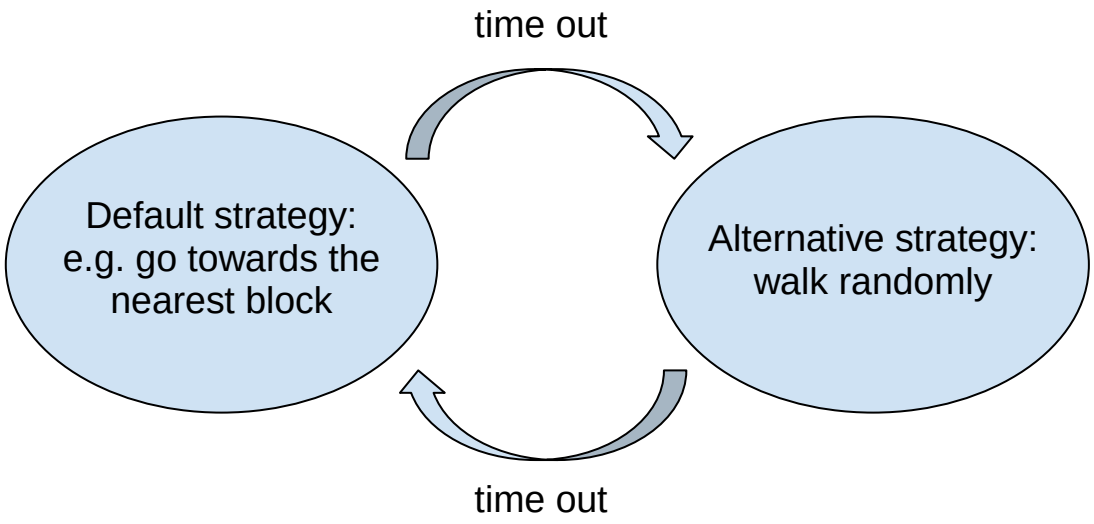
- V-REP: A virtual robot simulation platform (choose the free, educational version)
- Python: General-purpose, high-level programming language. If you do not have experience with Python programming, we suggest the following two resources:
    - Codecademy: Detailed python tutorial with web-integrated python interpreter
    - Pycharm: An intelligent python IDE

In the **`Lab1.zip`** file, you can find the following resources:

- **`Lab1-Agents-Task1-World.ttt`** file is a V-REP scene file. Load it in V-REP and try to get familiar with the GUI Interface. Several objects are defined in this scene. The main elements are the 'Pioneer' robot, 12 red blocks (objects to collect), outer and inner walls. Click items within 'Scene hierarchy' tree to inspect various objects and their properties.
    - The 'Pioneer' robot is set up to be controlled remotely by an external controller script.
- **`pioneer.py`** file is the remote python client that can act as the controller of the 'Pioneer' robot. In order for the script to work, three files from the V-REP's remoteAPI package need to be in the same folder as **`pioneer.py`**:
    - **`vrep.py`** and **`vrepConst.py`** from the directory `<PATH_TO_V-REP>\V-REP_PRO_EDU\programming\remoteApiBindings\python\python`
    - the appropriate remote API library, depending on your system **`remoteApi.dll`** (Windows), **`remoteApi.dylib`** (Mac) or **`remoteApi.so`** (Linux) from `<PATH_TO_V-REP>\V-REP_PRO_EDU\programming\remoteApiBindings\lib\lib`
- Run the simulation, by first clicking *start simulation* in V-REP and then running **`pioneer.py`** client script.
- Take a look at the code within **`pioneer.py`** and make sure you understand what it is doing, figure out the following (at least on a conceptual level):
    - Which two sensors of the robot have been used?
    - How to control the motion of the robot? How to assign different speeds to the two wheels?
    - How to get the nearest block's position? And how to collect it?
- Make sure you understand the following functions you will be using:
    - World.getSensoreReading(…)
    - World.setMotorSpeeds(…)
    - World.collectNearestBlock(...)
    - World.execute(…)

Tasks 1:

a. Implement an agent that takes random actions.
b. Implement an agent that performs exactly the same sequence of actions all the time, ignoring sensor input. Try to collect at least 1-2 red blocks.
c. Implement a simple reflex agent that takes sensor input and makes decisions based on the current sensor readings (no past information should be used).
d. Implement an agent that remembers past sensor readings and actions taken and utilizes it to achieve the goal in a more intelligent way. One type of the problem you probably encounter with the reflex agent is that it gets stuck at some part of the environment (e.g. in a corner). A very simple way to deal with this is to use a time counter for reaching the next block's position. If the robot is taking too much time to reach one of the blocks, it probably got stuck somewhere, and therefore it should change its strategy to deal with this (see figure below).



Task 2: Poker game agent

Introduction

This part of the lab requires you to implement three different agents to play a simplified poker game, the rules of the game are stated as follows:

- There will be two agents playing against each other within the game.
- We assume that coins/money are provided by the 'central bank', agents don't play with their own money and they have unlimited amount of money to play. The goal for each agent is to win as much money as possible.
- This is a simplified version of poker game and agents are only going to play with three cards. The possible hands are 'three of a kind', 'a pair' and the rest are 'high cards''.
- There will be 50 hands for each game.
- Each hand includes the following flow:
    - **Card dealing phase**: assigning a randomly generated hands (three cards) to each agent.
    - **Bidding phase (1-3):** agents decide how much money ($0-50) they want to bid into the pot. There are three bidding phases every hand.
    - **Showdown phase**: after three bidding phases, both agent show their hands and the agent with stronger hands gets the pot.
- After every 50 hands compute the sum of money each agent got.

Here is an example game flow of one hand:

- **Card dealing phase,** generate random hand (5 cards) to each agent:
    a. Agent 1 got a hand '4h, Ks, Kc'
    b. Agent 2 got a hand '5s, 5h 5c'
- **Bidding phase 1**, two agent decide the amount of money to bid:
    a. Agent 1 bids $20
    b. Agent 2 bids $30
- **Bidding phase 2**:
    a. Agent 1 bids $30
    b. Agent 2 bids $25
- **Bidding phase 3**:
    a. Agent 1 bids $5
    b. Agent 2 bids $45
- **Showdown phase**, both agents show their hand, agent 2 has a 'three of a kind' while agent 1 has only a pair of King. Therefore agent 2 wins and got the pot, which is $155.

Tasks 2:

a. Implement a random agent that bids randomly.
b. Implement a fixed agent.
c. Build the environment of the game, have the two agent play against each other. At this point you need to implement the following:
    i. game flow, according to the rules above.
    ii. hand identification and strength evaluation function for this simplified poker game. ***Lab1.zip*** contains a simple example that checks whether there is one pair in the hand.
    iii. sensor input for the agents: (1) hand of agent its own (during **Card dealing phase**) (2) hand of opponent (during **showdown phase**), (3) amount of money both agent bid (during **betting phase**).
    iv. a way of recording the results
d. Analyse the result of the game with random agent vs. fixed agent. Which agent is better? Why?
e. Implement a simple reflex agent and play it against random agent. The reflex agent should make better betting decisions based on the strength of its own hand.

Grading Criteria

- Pass: Complete all tasks (1a-d and 2a-e).
- Deadline is 2 weeks starting from the introduction session
- You submission should include:
    - Code
    - A short report (about 1 page of text) describing what you have done, observed and learnt
- Extra credits:
    - Your robot agent can collect more than half of the energy blocks.
    - Your robot can quite reliably avoid obstacles, i.e. it does not crash into walls.
    - You implement a poker agent (with memory) which makes betting decision based on its current hand strength and the amount of money opponent bet last round.
    - Have your poker agent with memory play against the three types of the agents 2a, 2b and 2e and make it
        - deduce which type of the agent it is playing
        - outplay other agents, by adapting its strategy to exploit the weaknesses of the fixed and reflex agents