University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	Artificial Intelligence (H)				
Coursework Number					
Deadline	**Time:**	**4:30pm**	**Date:**	22/11 2017	
% Contribution to final course mark	20%		**This should take this many hours:**	20 (per student)	
Solo or Group ✓	**Solo**		**Group**	X	
Submission Instructions	Code and report must be submitted via Moodle.				
Who Will Mark This? ✓	**Lecturer** X		**Tutor** X		**Other**
Feedback Type? ✓	**Written** X		**Oral**		**Both**
Individual or Generic? ✓	**Generic**		**Individual** X		**Both**
Other Feedback Notes					
Discussion in Class? ✓	**Yes**		**No** X		
Please Note: This Coursework cannot be Re-Done					

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below. The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via
http://www.dcs.gla.ac.uk/socs-online for all coursework
UNLESS submitted via Moodle

Marking Criteria
See the following pages.

Artificial Intelligence (H) 2017-2018
Assessed Exercise - Group project
23/10 2017a

1 Problem Statement

The task is to design, implement and evaluate three different virtual agents which are (potentially) able to survive, collect rewards and reach a goal in at least one of three pre-specified types of Malmo missions. The missions are all similar except for the size of the problem and specific layout.

The missions are situated in a virtual, maze-like world with obstacles, intermediate rewards and a final goal which must be reached within a certain timeframe (which depends on the size of the the mission). The performance measure is pre-defined as the average cumulative reward collected combined with the average time required to reach the goal across repetitions of the same mission type.

You must consider three different types of agents: a senseless/random agent, a simple agent and a realistic agent based on the requirements listed in the next section. Your agents will be tested against ten random instances of the same mission type, i.e., you can not (successfully) hard-code the solution. You will have access to three specific training missions which are defined programmably (small, medium and large). As part of a **three person group** your are only required to focus on one mission size. **Groups of four** are expected to consider two levels of complexity and report on those for all agent types.

Your agents and findings should be documented in a report (using the provided Latex/Word templates) and in terms of the actual implementation/code.

2 Tasks

You are required to provide a design, implementation, and evaluation of three different agent types; each with its own set of specific requirements:

2.1 Task I: Senseless/Random agent

You should provide a solution for an agent without sensory input which takes random actions.

Hint: This agent is used as a naive baseline. A basic senseless/random agent is already provided for the first mission in myagents.py outlining the interface the solution should preferably support.

2.2 Requirements

Sensors: None
Actions: Discrete and noise free. An action takes at least 200 ms to carry out.

State-space: No prior knowledge (i.e. it has not got a map)
Rewards/goal: No prior knowledge (does not know where the goal and intermediate rewards are located, does not know that there is a time limitation)

2.3 Task II: Simple Agent

You should provide a virtual agent based on a tree/graph-search or hill-climbing strategy of your choice. You should justify its use for solving the particular the mission (e.g. using A* search with a suitable heuristic or simulated annealing) assuming that the task environment fully known and observable.

Hint: This agent is used as an ideal baseline. If you have attended the Lab sessions you will be able to reuse most of the code to solve this part without adding new code.

2.3.1 Requirements

Sensors: Oracle (i.e. you know the location of everything by querying the oracle sensor)
Actions: Discrete and noise free. An action should take at least 200 ms to perform.
State-space: Fully observable a priori
Rewards/goal: Fully known a priori (knows where the goal and intermediate rewards are located, knows that there is a time limitation)

2.4 Task III: Realistic Agent

You should provide a realistic, virtual agent making minimal assumptions about the state-space. The choice of agent type is up to you but the choice should be well justified and meet the list of requirements listed below. This agent will typically require a form of online agent capable of learning the required properties as it explores and exploits the environment (see e.g. AIMA Chapter 4.5 or 21). A critical element in the design of this agent is the way the agent explores the state-space (possibly based on experience).

Sensors: Oracle sensor with radius one (optionally: use vision sensors)
Action: Discrete and noisy. The requested action is only carried out correctly 90% of the time and 10% of the time being selected randomly from the remaining possible actions (see code template for a predefined transition model with these properties). An action should take a minimum of 20 ms to carry out.
State-space: No prior knowledge, but partially observable via the sensors/actions.
Rewards: No prior knowledge, but partial observable via sensors/actions.

Notice: You can restart and replay the same instance of the mission multiple times and maintain the knowledge obtained across repetitions. The reward should in this case be reported as a) the average across all restarts/repetitions and b) the single best solution (typically after a long learning phase).

Hint: When developing your agent it may be beneficial to extent the time available to the agent or lower the size of the mission until you are sure it works correctly and can solve the task.

3 Implementation

Your solution - containing the three different agents (along with any dependencies, except Malmo) - should be uploaded to Moodle as a zip-file containing the source code (e.g. executable notebooks or standalone files) and instructions on how to run them.

The executable files should (preferably) follow the input/output interface defined in the code template and you should implement your solution by extending the predefined agent classes AgentRandom, AgentSimple and AgentRealistic (you may want create new files, classes and functions etc).

Any code which must be executed for your agents to run, should be called as part of the main file myagents.py (including training and replays if required for e.g. learning agents).

4 Report

The the group report should be in the style of a scientific paper using one of the provided templates (Latex / Word). It should not be longer than 7 pages each with columns (excluding references and appendices). You must include an appendix briefly outlining the contribution of each team member in terms of analysis, design, implementation, evaluation and documentation. Additional appendices may be used to provide extra information to support the data and arguments in the main document, e.g., detailed simulation results. It should in itself not provide crucial information required to understand the principle of the solution and the outcome. You can include as many references as you see fit.

5 Assessment

The assessment is based on the degree to which your submission (code and report) addresses the following aspects:

Analysis [15%]
-Suitable introduction, motivation and PEAS analysis/characterization, etc.

Method/design [25%]
-Presentation of relevant theory and methods (for all your agents), including justification for choosing specific methods, modeling assumptions/abstractions in defining the agent, etc.

Implementation [20%]
-The code for all the agents (not in the report!) should be well-commented and follow general best-practices in software engineering. The report must contain a presentation of relevant aspects of the implementation, etc.

Evaluation / Test [25%]
-Must contain a suitable presentation of the evaluation principle, metrics and the obtai-

ned simulation results. This includes a comparison of the three different agent types across repetitions of e.g. the same mission instance, etc.

Discussion [10%]

Conclusion [5%]

The weighting of the senseless, simple and realistic agents for all marking criterion is 5,20 and 75 %, respectively.

Marking guide: An A grade submission would contain a well-structure and well-written report, citing proper literature, containing a proper analysis, show creative use of presented AI techniques (incl. Malmo) to solve important and interesting problems in agent design, and take account of practical constraints. The code would be well-commented with a logical structure and appropriate use of external toolboxes. A B grade might be due to poorer writing, less insightful use of algorithms, poorer presentation of the simulation results. A C grade would have some basic use of AI techniques but would have shortcomings in the implementation, testing, presentation or justification of choices. A D grade report would have the most basic use of AI techniques, but they would not be (well) justified, and might not be desirable.