Introduction to AI – Part B (BKI122): Course Handbook

Introduction and Overview

Part A of Introduction to AI teaches the introductory theory of the basic concepts in Artificial Intelligence, such as search, representation, and machine learning, and applies that theory using paper exercises to 'toy' problems. The aim of Introduction to AI Part B, is to take that knowledge, implement in a computer and use it to solve 'real-world' problems. Implementing and experimenting with the methods yourself gives you a more direct experience of their operation and makes the ideas more concrete.

To make the course more fun the 'real-world' problems we will be solving are all those faced by a small yellow circle (ᗧ) trying to find enough food to survive in a world inhabited by deadly ghosts. Obviously, we can't go around killing ᗧ all the time so we will use a computer simulation of this environment, commonly called **PACMAN :-)** ᗧ (see <*en.wikipedia.org/wiki/**Pac-Man***>) ᗧ faces a number of challenges which the AI techniques taught in Intro2AI A can be used to solve, such as finding the best path to the food, optimally avoiding the ghosts to stay alive and identify the type of ghost so he can better avoid it. The assignments given during this course will lead you through how to help ᗧ survive as long as possible. The course cumulates with a 'survival horror' competition where ᗧ must survive as long as possible, with the team with longest living ᗧ declared the winner!

The teaching environment we are using is based on the one used at UC Berkeley for there CS188 introductory AI courses (see <www-inst.eecs.berkeley.edu/~cs188/pacman/pacman.html>). We have modified this environment slightly to fit better with our course, but it remains largely similar to the orginal one. **Note: an important aspect of this environment is that it is based on the Python programming language (see <*www.python.org*>).** However, the amount of Python needed to complete the course is small, so we will teach you a **minimal subset** of Python needed in the first week.

Structure

The course runs for 7 weeks with two 2hr practical sessions every week on Tuesdays and Thursdays in the Terminal room SP A.-1.55. The first 5 weeks consist of assignments where you will work in small groups (2-3 students) to implement and evaluate one of the basic AI methods to solve a 'real-world' problem. Solutions to these assignments must be submitted via. Blackboard before the start of the following weeks lecture (Tuesday 13:45). After the assignments have been graded, you will be given short feedback on your submission. In addition an **example solution** to the assignment will be made available after the assignment deadline, i.e. after the start of the following Tuesday lecture.

The final 3 weeks are for a final project and competition where you take all the knowledge you have learned to solve a more complex problem. The groups different solutions are compared in a competition.

In contrast to Intro2AI-A all stages of this course are very hands-on where you will work at a computer from a template system and problem specification and are guided through the process of generating a solution in a step-by-step manner. The final assignment stage is less guided, but both the instructors and Teaching Assistants will be available for to help with any questions you have during the scheduled practical sessions, and **by appointment** at other times.

Grading

As with Intro2AI A 10% (1 mark) is available for simply attending the practical sessions or making a reasonable attempt at the assignments. The basic assignments themselves are then further graded for quality to contribute 50% (5 marks) to the final grade. For some weeks there are also ***Extra Credit*** assignments. Successfully completing **two (2)** (or more ;-)) of these extra-credit assignments gives another 10% (1 mark). The final project makes up the remaining 30% (3 marks) of the final grade.

The detailed mark breakdown for each of the assignments is given in the detailed assignment description for each week, which will be uploaded to Blackboard (Note: below we provide only a sketch of the assignment topic). However, generally half (50%) of the assignment marks will be given for a **working** solution, and half (50%) for the clarity of the code and comments. (Thus a badly written, but working code can obtain 50% of the available marks, or a clearly written but non-working code will also obtain 50% of the available marks).

Resits

You can resit the final assignment, by individually re-submitting the assignment, however the maximum score achievable for this resit is 70% of the assignment points, i.e. 30*.7=21%.

Due to the availability of example answers, a different procedure is used to resit the attendance and assignments. To resit these course components you must submit an solution to a new assignment which will be made available **after** the final competition. Note: this assignment will require approximately 1ec of effort, i.e. 1weeks work full time, but due to the deadline for the Binding Study Advice, you will only have 1 week in which to do it!

Course Schedule

Week 1: Python tutorial

In this practical session we will go through some simple tutorials to teach you the minimal subset of Python which you will need to complete the later assignments. (Note: If you can already program in Python then you may move on to the 1st assignment.)
N.B. The Python code we are using **only** works with versions 3.X. Thus if working on your personal computer ensure you are using this version.

Week 2: PyCharm and debugging

In this week we will go into some more detail about the object-orientated aspects of Python, how you can use the PyCharm IDE effectively, and how to do efficient debugging.

Week 2: Single agent search (single goal) – get the food

is hungry but tired, he knows there is food here somewhere (he can smell it). How can he get to the food as efficiently as possible?

Here we start with a simple Pacman environment where there are **no ghosts** and only a single goal food pellet. Your assignment is to implement and compare different search methods so can can find the food pellet in the fewest moves possible.

<u>Assignments:</u>
Implement DFS, BFS, A* (Also Lowest-cost-first aka Uniform-cost, Best-first?) by filling in the empty parts

of 'search.py' and using pre-programmed data structures (state, fringe stack/queue, closed list etc.)
Testing:
On predefined mazes of various sizes where goal always is (1,1) and start = (n,n), i.e. opposite corner.

Extra Credit:

The PACMAN world is full of corridors. Clearly it is stupid for ᗧ to go part way down a corridor and then turn round (when there are no ghosts). However, the naïve search method used above checks if turning around is a reasonable option after every step down the corridor. Potentially, a lot of search effort could be saved by representing the problem such that a single action moves ᗧ the whole way down a corridor rather than every individual step. (Replacing a single move by a sequence in this way is a form of problem abstraction, which is commonly used to speed up search – particularly in computer games where the graph of cross-roads and moves between them is called the way-point graph – see e.g. <*www.aiwisdom.com/ai_pathfinding.html*>.)

Assignment:
Modify the A* method above to work such that actions move between cross-roads in the map (i.e. points where ᗧ has more than 2 moves available).
Testing:
On predefined mazes of different size where goal always is (1,1) and start = (n,n), compare the performance in terms of numbers of nodes expanded by the simple and abstracted version of the problem.

Week 3: Single agent search (multiple goals) – get all the food

ᗧ Is **still** hungry and tired, but this time there seems to be lots of food! and not a ghost in sight! Your assignment is to modify the simple A* method from the previous lecture to work when ᗧ wants to visit lots of equally rewarding goals as efficiently as possible (i.e. in the minimum number of moves).

Assignment:
(1) Modify your A* agent from last week to work when there are multiple goals.

Extra Credit:

Some of the food-pellets are much bigger than the others, how would you modify your solution above to deal with this issue?

Assignment:
(1) Modify your multi-goal A* agent to work when different goals have different possible rewards.

Week 4: Multi-agent (Adversarial) search – beat the Ghosts

Oh No! Now there is a ghost 👻 around who for some reason is trying to kill ᗧ! ᗧ needs some way of planning to achieve his goals of eating the pellets, whilst avoiding the ghosts. As we don't know how clever the 👻 are, for now we will assume they try to act optimally to get ᗧ.

Assignments:
(1) Implement a Minimax agent for ᗧ so it can cope with avoiding the 👻.
(2) Implement alpha-beta pruning for your agent to improve its search efficiency.

Extra Credit:

Now there are 2, 3, 4 ghosts. Extend you solution so ᗧ can cope where when there are an arbitrary number of ghosts.

Assignments:

(1) Adapt the MinimaxAgent to cope with more than 1 👻.
(2) Add alpha-beta pruning to your multi-adversary agent..

Week 5,6,7: Survival Competition Assignment

ᗧ now has all the pieces needed to make a successful PACMAN agent. In the final 2 weeks of the course you will put all the pieces together, i.e. search and machine learning, to build an effective PACMAN playing agent. In the final assignment competition the agents from different groups will compete in a 'game-play' track to see which teams agent can survive the longest in the PACMAN world.

Note: the competition world has 2 additional features not mentioned so far:

1) Multiple levels: if ᗧ survives one level (by eating all the pellets) he will move on to the next level. Later levels will be **harder** due to having either larger maps and/or more 👻.

2) CPU limits : the time available for ᗧ to compute the optimal move will be limited

<u>Assignment:</u>
(1) You have 2 weeks to build the best PACMAN agent you can. We will provide you examples of the types of levels your agent will encounter.

Week 8: Assignment Deadline & Competition

The final assignment and competition will take place on the last day of the course. On this day we plan to run the competition on the beamer and you can invite your friends to see the action. Each groups agent will be run through all the levels of increasing difficulty until ᗧ dies. ᗧ's total survival time and number of pellets eaten in the last level completed are used as the group score to decide on the winner.

As well as submitting your PACMAN agent, you must submit a short written report (2-3 pages) explaining the methods used by your agent.

Resit Assignment: Machine learning – identify the Ghosts

After a while realizes the are actually **not very clever**. In fact, after careful empirical observation he notes that they only adopt one of 3 strategies to find him:

1. **Seeker** : this always tries to move directly towards current location.
2. **Tracker** : this forever runs round the same 'racetrack' of locations on the map.
3. **Random** : this just moves randomly at any time.

knows that if he can identify which type of he is trying to beat, then he can exploit it's stupidity to make his own life much easier. Unfortunately, he doesn't (yet) know how to identify which type of ghost is which. Fortunately, has been recording his experiences for a while now and has a large database of ghost movements and there identified type, (i.e. seeker, tracker, or random). also knows that possibly this thing called machine learning can use this database to learn some quick rules for identifying the type.

Assignments:
(1) Given a database of the last 20 and positions along with types, use ID3 to learn a set of rules to identify the ghost type.
(2) identification with the 'raw' position features seems more complex than it should be. Design some new features computed from the 'raw' features which make the machine learning problem easier[1].

Extra Credit:

Now can identify the type of , he can predict what they will do in the future. As the no longer move optimally to defeat and are completely deterministic no longer needs to use adversarial search but can treat the as simply dynamic part of the environment.

Assignment:
(1) Modify your multi-goal A* solution from Week 3, to include the predicted movements of a single ghost of a known type (and known race-track for the tracker ghost).

[1] This process of preprocessing features to make the learning problem easier is a basic first step in applying machine learning to most problems which is sometimes called feature engineering.