Introduction to AI – Part B (BKI122a): 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 it 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 trying to find enough food to survive in a world inhabited by deadly ghosts. Obviously, we can't go around killing all the time so we will use a computer simulation of this environment, commonly called **PACMAN** (see <en.wikipedia.org/wiki/Pac-Man>).
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 survive as long as possible. The course cumulates with a 'survival horror' competition where must survive as long as possible, with the team with longest living declared the winner!

The teaching environment we are using is based on the one used at UC Berkeley for their CS188 introductory AI courses (see < http://ai.berkeley.edu/project_overview.html >). We have modified this environment slightly to fit better with our course, but it remains largely similar to the original one. **Note: an important aspect of this environment is that it is based on the Python programming language (**see <www.python.org>, which is new to you. However, the amount of Python needed to complete the course is small; in the first weeks we will teach you a **minimal subset** of Python which should be sufficient.

● Structure

The course runs for 11 weeks, usually with two 2hr practical sessions per week (on Tuesdays and Thursdays in the computer room SP A.-1.55). But one week is free (May holidays) and two weeks only have a Tuesday session (due to Ascension Day c.q. Dies Natalis on the Thursdays), leaving 8 "full week equivalents". See the course schedule on BB for details.

The first 8 weeks running (6 "full week equivalents") 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 Tuesday session in the following week (except for the first assignment, which can be submitted one week later, together with the second one). 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 submission deadline.

The final 3 weeks are for a final project where you take all the knowledge you have learned to solve a more complex problem. The different solutions of the groups 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, you get 10% (1 point) of the final grade for simply attending the practical sessions and making a reasonable attempt at the assignments. The weekly assignments themselves are then further graded for quality to contribute 60% to the final grade. The final project makes up the remaining 30% of the final grade.
For some weeks there are also ***Extra Credit*** assignments. For successfully completing **two (2)** of these extra-credit assignments over the whole course (or more ☺) you get an extra point in your final grade (unless that makes your final grade higher than 10, in that case you get a 10 with honours).

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 topics). Generally speaking, 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, and 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 weekly assignments. You can resit **one** weekly assignment (with the lowest grade) by submitting a solution to a new assignment which will be made available **after** the final competition. Note: this assignment will require approximately 2 days 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 (April 14 & 16): Python basics

In these practical sessions 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 next assignment.)
N.B. The code we are using **only** works with Python version 3.4. Thus if working on your own computer ensure you are using this version.

○ Week 2 (April 21 & 23): PyCharm and debugging

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

○ Week 3 (April 28 & 30): 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 with the least possible effort?

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

- Extra Credit:

The PACMAN world is full of corridors. Clearly it is stupid for  to go part way down a corridor and then turn round (when there are no ghosts). However, the naïve search methods used above check 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 the whole way down a corridor rather than just doing one 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>.

- *Week 4: May holidays*

has to do without you this week. Let's hope he's still alive and kicking when you get back.

- Week 5/6 (May 12 & 19): Single agent search (multiple goals) – get all the food

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

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?

- Week 6/7 (May 21 & 26): Multi-agent (Adversarial) search – beat the Ghost

Oh No! Now there is a ghost around who for some reason is trying to kill ! 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 .

Extra Credit:

It gets worse and worse, now there are 2, no, 3, no, **many** ghosts. Extend you solution so can cope when there are an arbitrary number of ghosts.

- Week 8 (June 2 & 4): Machine learning – identify the Ghosts

Let's relieve 's stress a bit by turning back to there only being one ghost at a time, giving him the chance to reflect upon ghost intelligence. 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 's 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 its 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 their 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.

Extra Credit:

Now can identify the type of , he can predict what they will do in the future, i.e. they are completely deterministic. As the s no longer move optimally to defeat So no longer needs to regard the s as adversaries that move optimally to defeat him (using adversarial search), but he can treat the s as simply a dynamic part of the environment, using a modified version of the earlier (in week 4) implemented multi-goal A* search that includes the predicted movements of a single ghost of a known type (and a known race-track for the tracker ghost).

- Week 9/10/11 (June 9, 11, 16, 18, 23) : Survival Competition Assignment

 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.
You have 2.5 weeks to build the best PACMAN agent you can. We will provide you with examples of the types of levels your agent will encounter.

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

1) Multiple levels: if  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  s.
2) CPU limits : the time available for  to compute the optimal move will be limited.

As well as submitting the code for your PACMAN agent, you must submit a short written report (2-3 pages) explaining the methods used by your agent. Deadline for submission is June 24, 23:59.

- Week 11 (June 25) : 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 group's agent will be run through all the levels of increasing difficulty until  dies.  '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.