Artificial Intelligence

Christos Dimitrakakis

June 5, 2024

Outline

Aims

This course will focus on algorithms and models for Artificial Intelligence. We will concentrate mainly on the decision making, rather than the learning, side of artificial intelligence. Learning is already addressed in statistics courses, as well as the machine learning course in the final year.

Philosophy

The philosophy of this course is as follows:

- We give example problems.
- We use theory to explain and generalise from those examples to general problems.
- We describe algorithms to *solve* general problems.
- We implement algorithms to solve the specific examples.

In general, the course will start from the simplest problems and slowly progress to the more complex ones.

How-to

Lectures

- All the relevant course content is given in-class.
- The slides only contain a summary.
- More details in the reference books.

Assignments

- Assignments are obligatory but not marked.
- You have a free pass on not handing in two assingments

Projects in teams of 2-3 people (60% of grade)

1. Problem formulation [Project proposal]
2. Problem formalisation
3. Selection of algorithms [Mid-term report]
4. Method evaluation.
5. Summarise findings and discuss alternatives to 1-3. [Final report]

Written exam (40% of grade)

Intelligence

What makes something intelligent?

- Thoughts?
- Behaviour?

Why do we need artificial intelligence?

- To do something a human can do (better? worse?)
- To do something a human cannot do?
- To help humans do something more efficiently?

Decisions and learning

Algorithms as thoughts

- Incorporating new information
- Deciding what to do.

Learning: Using information to update knowledge.

- Situational awareness: what is going on right now?
- Epistemic knowledge: How does the world work.

Decision making and interaction

- Recommend a route.
- Buy and sell stocks.
- Make a move in chess.

To the moon!

Goal: See if humans can live on the moon.

Learning

- Orbital mechanics: What are the equations of motion?
- Trajectory following: Where is the spaceship right now?

Decisions

- How much should we spend?
- How much payload/fuel/time?
- What is a good trajectory for a given payload/fuel?
- How can the spaceship follow the trajectory?
- How can the spaceship land on the moon?

The view from statistics

Optimal decisions

- Problem-dependent
- Require defining a cost- or utility function
- The optimal solution has the lowest cost or maximal utility

Optimisation: algorithms for optimal solutions

- Discrete optimisation.
- Linear optimisation.
- Non-linear optimisation.

Statistics and Machine Learning

- How to learn from data and interactions.
- Summarising knowledge into a model.
- Using the knowledge to make decisions.

Agent-environment interface

Agent

- Obtains stimuli/observations x_t
- Generates actions/decisions a_t

Environment

- Reacts to agent's actions
- Generates observations

The mind/body interface

- The body can be seen as part of the mind's environment

Policy and history

- The agent's next action a_{t+1} depends on previous observation's and actions.
- The policy is implemented through an algorithm

Examples

- Mazes
- Algebraic manipulation
- Chess game
- Poker game
- City driving
- Navigation assistant
- Space exploration

Environment components

We generally consider dynamic environments, so at time t:

- s_t: state of the environment
- x_t: observation of the environment by the agent
- a_t: actions taken by the agent

Example: Mazes

- s_t: the location of the agent in the maze
- x_t: What the agent observes (exact location, or just surroundings?)
- a_t: Direction in which the agent moves

Policies

- Policies determine the behaviour of the agent.
- They define what the agent does at any given time.

Reactive policies

We allow agents to randomise. The simplest agent choose actions only depending on the current observation:

$$\pi(a_t|x_t) \qquad \text{(the probability with which the agent takes action } a_t\text{)}$$

Deterministic (reactive) policies

Then for each x_t the same action $f(x_t)$ is always taken, so that $\pi(a_t = f(x_t)|x_t) = 1$.

Adaptive policies

The action taken may change over time, depending on what happened in the past:

$$\pi(a_t \mid x_t, a_{t-1}, x_{t-1}, \dots, a_1, x_1)$$

Example policies

Reactive maze policy

- Ordered actions $A = \{\mathrm{Up}, \mathrm{Right}, \mathrm{Down}, \mathrm{Left}\}$
- Take action $a_{t+1} = a_t$ unless there is a wall in front.
- If there is a wall, take the next action, $a_{t+1} = a_t + 1$. (where $+$ cycles over the 4 actions)

Problems with this policy

- Can it solve any maze?
- Why yes/no?
- What can we do to make sure that the agent visits every point of the maze?

Example: taking an exam

High-level policy

- Study for exam
- Prepare exam materials
- Get to exam on time
- Write

Mid-level policy for getting to the exam:

- Check starting time.
- Check location.
- Select transport option
- Set alarm clock.
- Go to the exam.

Low-level policy: Go to the exam.

- Get dressed
- Pick up things
- Get transport

Example: Planning a trip

There are three train routes from Neuchatel to Luzerne

- Neuchatel 6:58-IC-7:57 Olten 8:07-RE-8:55 Luzern (18 CHF)
- Neuchatel 7:01-S-7:52 Bern 8:00-IR-9:01 Luzern (22 CHF)
- Neuchatel 7:26-IC-8:18 Olten 8:30-IR-9:05 Luzern (26 CHF)

Criteria for choosing

- Price
- Train type
- Crowdedness
- Length of time

Planning the trip

- Is "go through Bern" enough of a plan?
- What about delays or cancellations?

Multiple levels of actions

- Which route to use, and fallbacks.
- What to pack
- How to get to the station

Hierarchical control

High-level planner
Selecting plans for the low-level controller

Low-level controller
Selects actions for each plan selected by the high-level planner

Learning and memory

Belief state

- Memory
- A summary of the agent's knowledge
- The state in a state machine
- The contents of the tape and read/write heads on a Turing machien.

Belief transitions

- A (possibly randomised) function $f : B \times A \times X \to B$

$$\beta_{t+1} = f(\beta_t, a_t, x_t)$$

- $\beta_t \in S$: Belief at time t.
- $a_t \in A$: Action at time t
- $x_t \in X$: Observation at time t.
- f is implemented by the agent's algorithm

Memory in the maze example

Known maze, known location

- Agent observes everything.
- No memory required.

Known maze, unknown co-ordinates

- Agent only observes immediate surroundings.
- Memory keeps track of location.

Unknown maze, known co-ordinates

- Agent only observes immediate surroundings.
- Memory keeps track of maze layout
- Agent always knows its exact co-ordinates.

Unknown maze and co-ordinates

- Agent only observes immediate surroundings.
- Memory keeps track of maze layout and co-ordinates

Goals as a design principle

- Easy to define
- Can be too vague.

Example: mazes

- Assign "goal" to a maze location
- The agent should find the way to the goal.

Example: exams

- Goal: pass the exam
- The agent should find a strategy so that it passes the exam!

Utility as a design principle

- Hard to define.
- Can be too specific.

Example: mazes

- Prefer shortest path to longer ones to the goal.
- More complicated if we have intermediate goals.

Example: exams

- Prefer higher grades than lower grades.
- Prefer to study less than more

Preferences

Types of rewards

- For e.g. a student: Tickets to concerts.
- For e.g. an investor: A basket of stocks, bonds and currency.
- For everybody: Money.

Preferences among rewards

For any rewards $x, y \in R$, we either

- (a) Prefer x at least as much as y and write $x \preceq^* y$.
- (b) Prefer x not more than y and write $x \succeq^* y$.
- (c) Prefer x about the same as y and write $x \eqsim^* y$.
- (d) Similarly define $\succ^*$ and $\prec^*$

Utility and Cost

Utility function

To make it easy, assign a utility $U(x)$ to every reward through a utility function $U : R \to \mathbb{R}$.

Utility-derived preferences

We prefer items with higher utility, i.e.

- (a) $U(x) \geq U(y) \Leftrightarrow x \succeq^* y$
- (b) $U(x) \leq U(y) \Leftrightarrow y \succeq^* x$

Cost

It is sometimes more convenient to define a cost function $C : R \to \mathbb{R}$ so that we prefer items with lower cost, i.e.

- $C(x) \geq C(y) \Leftrightarrow y \succeq^* x$

Decision making as an optimisation problem

How can we find the decision maximising utility / minimising cost?

Choice of the utility function

Designer input

- The AI designer selects the utility (or goals)
- The choice is not always obvious!

The value-alignment problem

- The designer selects a utility they think is the best choice
- However, their choice results in unintended behaviour
- Example: Autonomous vehicles

The value-alignment in populations

- Not everybody wants the same thing.
- We need to design fair policies.

Multi-agent problems

External agents

- Have their own utility/goals
- Are partly rational

Designed agents

- We can choose their utility/goals
- Computation/Optimality trade-off

Goals versus preferences

Maze-solving

- How should we define the utility/cost of every path?
- Is an additive utility sufficient?

Exam taking

- What if you say you want to perform super-well in the exam?
- How can set up the decision problem of how to study for the exam?

Exercises (From AI3e, 2.7)

- 1. Representations
- 2. Top-level controller.
- 3. Obstacle avoidance.
- 4. Robot trap.
- 10. Autonomous cars: driver preferences

Assignments (From AI3e, 2.7)

- 5. Moving targets
- 7. Sensing
- 8. Batteries
- 9. Which functions?
- 11. Autonomous cars: state of the art.