DS425 - Reinforcement learning assignment

2022-2023
174 POL

Context

Reinforcement learning is a type of machine learning that deals with training agents to make decisions in an environment. The agent learns by taking actions and receiving rewards or penalties, and it tries to maximise the total reward over time. In this assignment, you will implement or use a reinforcement learning algorithm to train an agent to solve a specific task. The task and the algorithm will be chosen by you, based on your interests and available resources. You will have to design the state space, the action space, the reward function, and the algorithm, and you will have to train and evaluate the agent using appropriate metrics. The goal of the assignment is to gain hands-on experience with reinforcement learning and to showcase your understanding of the concepts and the techniques. This assignments counts as one out of two evaluations for this course.

Some guidelines

- The expected outcome is a small paper in which you explain the problem setting and analysis, how you went about it and your conclusions. Additionally, a code repository or script that allows us to reproduce your results should be provided.
- Work in small groups (1-3), detail the contributions of each group member.
- Project timeline: by the beginning of March you should have seen the necessary theory and some applications to allow you to get started. The deadline of the project is the Friday of the week before the course exam (e.g. should the exam be scheduled on June 8, the deadline would be June 2).
- No specific programming language is imposed, you can use Julia, Matlab, Python or another language of your choice. However, clear instructions on how to run the code should be provided.
- Limit yourself to what we have seen in the course e.g. don't submit something using deep reinforcement learning, as neural networks are addressed only by the end of the course.
- Be honest about your work. If you reuse code or ideas from someone else, reference them in the proper way. Also, avoid submitting a copied tutorial as a finished piece of work. You can definitely reuse something that exists, but you should add a useful contribution or extension yourselves too.

Some pointers

- Ideas for simple applications (note that some of these could also be solved using search):
  - Tic-tac-toe: Train an agent to play the game of tic-tac-toe against a human or another agent.
  - Towers of Hanoi: Train an agent to play Towers of Hanoi.

  - Playing simple games: Multiple old-school 2-D games are suited for reinforcement learning (e.g. train an agent to play flappy bird).
  - Inventory management: Train an agent to make ordering decisions for a retail store.
  - Mountain Car: Train an agent to learn to climb a hill.
  - Cartpole: Train an agent to balance a pole on a cart in a more complicated environment.
  - Traffic lights control: Train an agent to control the traffic lights in an intersection to minimize wait times and maximize traffic flow.
  - Blackjack: Train an agent to play the card game of blackjack.
  - Path planning for a robot: Train an agent to plan the optimal path for a robot to move from one point to another in a 2D or 3D environment, while avoiding obstacles.
  - Job scheduling: Train an agent to schedule jobs on a set of machines in a factory or a data center, to minimize the makespan (the time required to complete all jobs) or the total flow time (the time each job spends in the system).
  - Energy management: Train an agent to manage the energy consumption of a building or a neighborhood, by controlling the heating, ventilation, and air conditioning, the lighting, and the appliances, to minimize the energy costs or the greenhouse gas emissions.
  - Recommender systems: Train an agent to recommend items to users, such as movies, books, or music, based on their preferences and history.
  - Portfolio management: Train an agent to manage a portfolio of assets, such as stocks, bonds, or commodities, based on their historical returns and volatilities (and maybe compare with the performance of the Markovitz mean-variance optimization we did in ES313).
  - Robotics manipulation: Train an agent to manipulate objects with a robotic arm or a robotic hand.
  - Self-driving cars: Train an agent to drive a car in a simulated environment.

- Useful references:
  - ReinforcementLearning.jl: a package for reinforcement learning research in Julia.
  - ACME: a framework for reinforcement learning in Python.
  - Matlab reinforcement learning toolbox: Matlab environment for reinforcement learning.
  - OpenSpiel: a framework for reinforcement learning in games. Has both a Julia and a Python wrapper. Can be used for inspiration.
  - Gymnasium: open source Python library for developing and comparing reinforcement learning algorithms (formerly run by OpenAI). Also includes lots of environments.
  - MiniGrid: a collection of discrete grid-world environments to conduct research on reinforcement learning, compatible with Gymnasium.

- If you are short on computational power, you can always try one of these options:
  - The RMA cluster
  - Google Colab, which can run both Python as well as Julia code.
  - A Kaggle notebook, which is restricted to Python and R
  - Amazon SageMaker, defaults to Python, but can also run Julia code