Science and Technology Facilities Council

Hartree Centre

Fundamentals of Reinforcement Learning

Nam Nguyen – AI Group – Hartree Centre – STFC

Session introduction

- Goal: Use RL to solve real-life problems.
  - How to formulate a real-life problem as an RL problem?
  - How to implement an RL problem as a Gymnasium environment? Formulating an RL problem as a Gymnasium environment is beneficial as it allows us to use a range of helpful packages.
  - What is Deep Reinforcement Learning? What goes on under the hood of a Deep Reinforcement Learning algorithm?
  - Understanding is important but no need to reinvent the wheels. Can we make use of pre-implemented RL algorithms to solve our problems?

Session outline

- I present some slides and demonstrate with Jupyter notebooks ~ 40 mins
- You get your hands dirty and do some coding ~ 45 mins
- Concluding remarks ~ 5 mins

Paying with coins

- Problem:
  - We have 3 types of coins in our pocket: £1, 5p, and 1p.
  - We need to pay a (randomly chosen) amount between 1p and £10 with our coins.
  - We want the amount we pay to be within 1p of the amount due.
  - We also want to minimise the number of coins we use for each payment (nobody wants 500 1p coins).
- Approach:
  - Translate this problem to an RL problem.
  - Train an RL agent that does the paying for us.

RL problem design

- A sequential decision process: we pay one coin at a time until the payment is complete.
- Observation: a real number from -1 to 10 representing in pounds the amount of money we are still due. E.g., 0.01 means 1p.
- Action: a discrete number from 1 to 3 describing which coin we want to pay with.
- Reward:
  - If the amount of left to pay is less 1p and more than -1p (we overpay less than 1p), we return a reward = 100.
  - Else, we return a reward = -1
- Termination condition: the amount still due is less than 1p.

RL problem implementation

- Live coding in Jupyter notebook

Deep reinforcement learning

- Deep RL differs from traditional RL in that it approximates the policy function (or other learned functions) using a neural network.
- Deep RL is useful when there are too many states and/or actions in our problem and, thus, impossible to learn the value of each state/action pair individually.
- Once we have found the optimal neural network architecture and their optimal parameters, we can generalise from seen states to unseen states.

REINFORCE

- Approximate the policy with a NN
- Identify the policy objective function
- Optimise the NN parameters to maximise the policy objective function
  - Collect data
  - Update NN parameters using gradient ascent

REINFORCE implementation

- Live coding in Jupyter notebook

Stable baselines 3

- stable-baselines3 is a popular package of implementations of reinforcement learning algorithms.
- To demonstrate how to use stable-baselines3, we shall use it to solve the Gymnasium Cartpole environment.
- We shall also use stable-baselines3 to solve our Paying with coins RL problem.

Stable baselines 3

- Live coding in Jupyter notebook

Solve paying with coins

- Live coding in Jupyter notebook

Why our agent fails?

- Generalise but incorrectly. When the amount due is 0.98 for example, because this value is so close to 1 (or even 0.99), our agent suggests giving £1 coin. However, this will make us over pay. Instead, we should use 9 10p coins and 8 1p coins.
- To fix this, one suggestion is to discretise the observation space into buckets of width 0.01. The idea is that payment in the same bucket requires the same solution. Putting payments in buckets avoid incorrect generalisation.

Why our agent fails?

- Generalise but incorrectly. When the amount due is 0.98 for example, because this value is so close to 1 (or even 0.99), our agent suggests giving £1 coin. However, this will make us over pay. Instead, we should use 9 10p coins and 8 1p coins.
- To fix this, one suggestion is to discretise the observation space into buckets of width 0.01. The idea is that payment in the same bucket requires the same solution. Putting payments in buckets avoid incorrect generalisation.

Q learning solves paying with coins

- Live coding in Jupyter notebook

Gymnasium wrapper

- We implement Q learning so we can add a discretise step. How can we apply a "discretise" step to pre-implemented algorithms?
- Gymnasium wrapper allows us to create a modified instance of the environment (e.g., discretise its observation space) without having to re-code the original environment.
- Live coding in Jupyter notebook.

Paying with coins 2

- Problem:
  - We have 3 types of coins in our pocket: 51p, 5p, and 3p.
  - We need to pay a (randomly chosen) amount between 3p and £10 with our coins.
  - We want the amount we pay to be within 1p of the amount due.
  - We also want to minimise the number of coins we use for each payment (nobody wants 500 1p coins).
- Approach:
  - Translate this problem to an RL problem.
  - Train an RL agent that does the paying for us.

You do it time!

- You have the complete
  - REINFORCE implementation for solving the Cart Pole problem.
  - Q learning implementation for solving the Cart Pole problem
  - Code that uses Stable baselines 3 to solve the Cart Pole problem.
- You have the partial
  - Paying with coin gymnasium implementation
  - Q learning implementation for solving the paying with coin problem
  - Code that uses Stable baselines 3 to solve the paying with coin problem
  - Discretise observation wrapper for the paying with coin gymnasium environment
  - Code that uses Stable baselines 3 on top of the wrapper to solve the paying with coin problem
- Task:
  - Fill in the partial code
  - Implement and solve the paying with coins 2 problem

Session summary

- Understand what goes under the hood of both tabular and deep reinforcement learning algorithms.
- Implement tabular algorithms in a toy gymnasium environment.
- Explore the effects of hyperparameters on learning rate and performance.
- Understand how to cast a real-life problem into an reinforcement learning problem.
- Understand how implement as a real-life problem as a custom gymnasium environment.
- Understand how to use custom code or pre-implemented packages of RL algorithms to solve problems.

Science and Technology Facilities Council

Hartree Centre

Thank you

Abbie Keats abbie.keats@stfc.ac.uk
Nam Nguyen nam.nguyen@stfc.ac.uk