Solving a McCall Search Model via Reinforcement Learning

Trevor Gallen
Econ 64200

Fall 2022

Deliverables

- You should have a word/LaTeXdocument that has three sections:
  1. Discusses the model and answers the questions I pose throughout.
  2. Contains the tables and figures you will produce.
  3. Contains a discussion of your programming choices if you had to make any.
- You should have a Matlab file or set of files (zipped) that contain **all** your programs and raw data. There should be a file called "Main.M" that produces everything I need in one click.

1 Model

In this homework, we're going to try to solve a simple McCall search model with Reinforcement Learning.

Each period, agents wage up with a wage draw $w \sim \log \mathcal{N}(1,3)$, they have a choice whether or not to accept the offer or not. If they accept, they receive that $w^{accepted}$ draw every period for the rest of their life. If they reject, then (if they don't die) receive zero and get a new wage draw in the next period, discounted by a factor of β. The probability of death is $\alpha = 0.067$. Their problem is can therefore be written:

$$V(w) = \max_{accept,reject} \left\{ \frac{w}{1-(1-\alpha)\beta}, (1-\alpha)\beta V(w') \right\}$$

2 Problem

Set this problem up and solve it with a reinforcement learning agent in Matlab. Note that while the state is continuous, the decision is discrete. However, you may make a discrete decision continuous (round a sigmoid function, for instance), and approximate a continuous state with a discrete set of N levels. To solve this, you must therefore:

1. Set up an action space (accept/reject) and observation space (wage draw)
2. Set up neural networks (if actor critic, then a critic network that takes in one action and one state and spits out a scalar, and an action network that takes in a state and spits out an action)
3. Set up a reset function that starts your actor
4. Set up a step function that advances them one period, or terminates the problem if they accept (or die)

This can be as simple as creating the observation info (rlNumericSpec(dimensions)), the act info (rlFiniteSetSpec(discrete actions), set up the environment (rlFunctionEnv(obsInfo,actInfo,stepfcn,resetfcn)) and then setting up a default DQN agent (rlDQNAgent(obsInfo,actInfo)), telling it the discount factor (agent.AgentOptions.DiscountFactor = xxx), and training it train(agent,env,opt). **I leave it to you** how you solve it, this is merely a suggestion.
Once you've solved the problem, you should graph out:

- The value function of each action, contingent on draw (getValue(getCritic(agent,wage draw)))
- The action at each wage draw getAction(agent,wage draw)

3 Research

Change the model in some interesting way and report your results.