Learning to make consumption-saving decisions in a changing environment: an AI approach *

Rui (Aruhan) Shi †

This draft: May 17, 2021

Abstract

This exercise offers an innovative learning mechanism to model economic agent's decision-making process using a deep reinforcement learning algorithm. In particular, this AI agent has limited or no information on the underlying economic structure and its own preference. I model how the AI agent learns in terms of how it collects and processes information. It is able to learn in real time through constantly interacting with the environment and adjusting its actions accordingly. I illustrate that the economic agent under deep reinforcement learning is adaptive to changes in a given environment in real time. AI agents differ in their ways of collecting and processing information, and this leads to different learning behaviours and welfare distinctions. The chosen economic structure can be generalised to other decision-making processes and economic models.

*I am deeply grateful to my supervisors Prof. Roger Farmer and Prof. Herakles Polemarchakis for their dedicated support and guidance. All remaining errors are mine.

†PhD candidate at the department of economics, University of Warwick. Email: a.shi@warwick.ac.uk.

1 Introduction

Both artificial intelligence (AI) and economics focus on decision-making of an intelligent entity. AI is broadly defined as intelligence demonstrated by machines, in contrast to the natural intelligence displayed by animals. AI technologies are widely adopted and successfully implemented in situations that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages. The essence of most economic problems is an economic agent's decision-making process. This exercise combines AI and economics and aims at building AI agents adopting cutting-edge computer science algorithms, deep reinforcement learning, to solve economic decision-making problems. I implement an AI algorithm in a stochastic optimal growth environment, with special focus on several novel features that this AI technology brings to economics. Notably, the economic agent under deep reinforcement learning is adaptive to a constantly changing environment owing to a special 'exploration' property of the algorithms.

This exercise offers an innovative learning mechanism to model economic agent's decision-making process using a deep reinforcement learning algorithm. Deep reinforcement learning, takes the middle ground of reinforcement learning and deep learning (i.e., deep artificial neural networks). Reinforcement learning is a class of algorithms that are motivated by how animals and humans learn in the real world, i.e., learning through reinforcing good/bad decisions based on some reward signals.[1] In this case, AI agents are born with no information about the fundamentals of the economy nor do they know their own preferences. They learn how to make optimal consumption decisions through trying out different consumption values and observe the consequence comes with these actions. In other words, this learning process first models how agent obtains information. The information is then processed through artificial neural networks. They are trained to maximum rewards over long run, or in economic terms, maximise their lifetime utilities. The AI agent's learning characteristics, governed by parameters within the learning algorithms, can also be adjusted with respect to particular questions. The chosen stochastic optimal growth model is a main building block for many macroeconomic models, and can be generalised to solve other decision-making processes.

This is not the first time that the two concepts are blended together. Sargent (1993) discusses his agenda in combining AI with macroeconomic modelling. He aims at finding a symmetry between econometricians and economic agents. Giving the agent a learning ability based on a decision rule, the agent, at limits, converges to a rational expectation equilibrium. In his case, agents behave like professional scientists or econometricians and use methods of scientific inference in collecting information and forming their expectation. He argues that this is an important line of literature because it looks at the transition behaviours and dynamics in a learning process. Earlier than Sargent, Herbert Simon defines the concept of bounded rationality and introduces his approach in adopting AI in making decisions.[2] Different from Sargent, Simon focuses more on the decision-making process rather than its outcome. AI that he suggests is on the heuristic search and problem solving by recognition (Simon, 2016).

[1] in reality, what differ us humans from a reinforcement learning agent is that rewards given by the reality are often not clear and understandable.

[2] He argues that bounded rationality denotes "the whole range of limitations on human knowledge and human computation that prevent economic actors in the real world from behaving the ways that approximate the predictions of economic theories: including the absence of a complete and consistent utility function for ordering all possible choices, inability to generate more than a small fraction of the potentially relevant alternatives, and inability to foresee the consequences of choosing the alternatives"(Simon, 2016).

The combination of AI and economics executed here is different from both Sargent's and Simon's views. The AI agent in this exercise does not follow a pre-specified decision rule. It must determine what is feasible (learning the causal rules of the world), and what is desirable (learning the utility-filled states of the world) by interacting with its environment and experimenting.

In the following sections, I first give an introduction of AI technologies, focusing on deep reinforcement learning and related literature. This is followed by the discussion of an economic environment and how I build the artificial economic agent in this specified environment. In the end, I present several experiments and corresponding results highlighting the novelty of combining economic modelling with deep reinforcement learning.

2 AI Technologies and Reinforcement Learning

Artificial intelligence or machine intelligence has been on the centre stage of computer science, and subsequently technological advancement for decades. The field of artificial intelligence, or AI, attempts not just to understand but also to build intelligent entities (Russell, 2020). It involves a wide range of machine learning techniques.[3] Machine learning is about learning from data and making predictions and/or decisions. It is broadly categorised as supervised, unsupervised, and reinforcement learning.[4]

At the core of AI technologies is the class of algorithms under deep reinforcement learning (DRL), which is the combination of reinforcement learning and artificial neural networks (ANNs). ANNs are used as function approximators, which help reinforcement learning to deal with environment settings with high-dimensional state and action spaces[5]. Notable developments include teaching AI agents (using DRL algorithms) to play Go and Atari games, and to learn speech recognition.

2.1 Reinforcement Learning: a primer

The early history of reinforcement learning (RL) has two main threads that were pursued independently before intertwining in modern RL. One thread concerns learning by trail and error, which originates in the psychology of animal learning. The second thread, which is familiar to most computational economists, concerns the problem of optimal control and its solution using value functions and dynamic programming. For the most part, this thread does not involve any learning. The threads come together in late 1980s (Sutton and Barto, 2018).

Modern RL[6] contains a series of algorithms aiming at solving Markov Decision Processes (MDPs). It is distinguished from other computational approaches by its emphasis on learning by an agent from direct interaction with its environment, without requiring exemplary supervision (e.g., supervised machine learning) or complete models of the environment (e.g., dynamic programming). RL uses the formal framework of MDPs to define the interaction between a learning agent and its environment in terms of states, actions, and rewards.

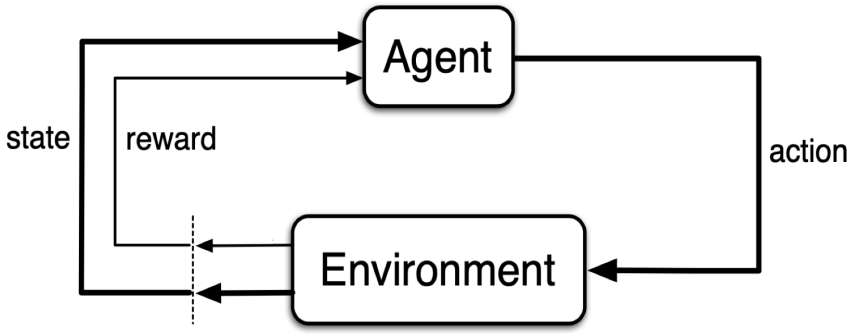
[3] For an in-depth review of AI, please see Russell (2020).

[4] In supervised learning, there are labelled data; in unsupervised learning, there are no labelled data. In reinforcement learning, in contrast to supervised learning and unsupervised learning, there are evaluative feedback (i.e., reward signals), but no supervised labels.

[5] An example is learning to play video games directly from raw pixels

[6] For a comprehensive review, please see Sutton and Barto (2018).

Figure 1: The agent-environment interaction in a reinforcement learning setting



Source: Sutton and Barto (2018)

An MDP, as described by figure 1, shows a process where given a state variable agent interacts with the environment and chooses an action, this leads to a reward signal for the agent and the current state transits to the next.

At each time step t, an RL agent receives some representation of the environment's state out of a state space, $s_t \in \mathcal{S}$, and on that basis selects an action out of an action space, $a_t \in \mathcal{A}(s_t)$. One time step later, in part as a consequence of its action, the agent receives a numerical reward, $r_t = r(s_t, a_t)$, and finds itself in a new state, s_{t+1}. The new state then feeds into another loop of the agent-environment interactive process. To describe how the state transits, a three-argument function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is defined as

$$p(s_{t+1}|s_t, a_t) \equiv Pr\{s_{t+1}|s_t, a_t\} \tag{2.1}$$

for all $s_t, s_{t+1} \in \mathcal{S}$ and $a_t \in \mathcal{A}(s_t)$. It shows the probability of transition to state s_{t+1} from state s_t, taking action a_t.

A RL agent's task is to learn to make a decision that maximises its expected returns. The decision-making strategy, or the RL agent's behaviour is defined by a policy. If an agent is following a stochastic policy π at time t, then $\pi(a_t|s_t)$ is the probability of choosing action a_t given a state s_t. If an agent is following a deterministic policy μ at time t, then $\mu(s_t)$ gives an action for a state s_t. Expected returns are described by a value function, which estimates how good it is for an agent to perform a given action in a given state (Sutton and Barto, 2018). Formally, it is denoted as $Q^\pi(s_t, a_t)$, which shows the expected return after taking an action a_t in state s_t and thereafter following policy π.

$$Q^\pi(s_t, a_t) \equiv \mathbb{E}[R_t|s_t, a_t] \tag{2.2}$$

where $R_t = \sum_{k=0}^{\infty} \beta^k r_{t+k}$, and it is the sum of discounted future rewards. $\beta \in [0, 1]$ represents a discount factor.

An Optimal action-value function, defined as

$$Q^*(s_t, a_t) \equiv \max_{\pi} Q^{\pi}(s_t, a_t) \tag{2.3}$$

for all $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}(s_t)$. For the state - action pair (s_t, a_t), this function gives the expected return for taking action a_t in state s_t and thereafter following an optimal policy.

The Bellman optimality equation for Q^* is

$$Q^*(s_t, a_t) = r(s_t, a_t) + \beta \mathbb{E} \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}). \tag{2.4}$$

This expresses the fact that the value of a state-action pair under an optimal policy must equal the expected return for the best action from that state.

The central assumption in RL is that its agent does not know how a reward r_t is generated and what transition dynamics $p(s_{t+1}|s_t, a_t)$ are. Instead, the agent must determine what is feasible (learning the causal rules of the world, $p(s_{t+1}|s_t, a_t)$), and what is desirable (learning the utility-filled states of the world, $r(s_t, a_t)$), by wandering through the environment and experimenting. The job of any RL agent is to learn about r, p, and Q as best as possible, so as to come up with a decision-making strategy, i.e., a policy π or μ, that maximises Q in states of relevance. This process gives a natural machinery to model bounded rationality, as argued by Abel (2019).

2.1.1 Exploration vs Exploitation

To solve a RL problem, one of the challenges that arises is the trade-off between exploration and exploitation. To obtain a lot of rewards, a RL agent must prefer actions that it has tried in the past and found to be effective in producing rewards. But to discover such actions, it has to try actions that it has not selected before. The agent has to exploit what it has already experienced in order to obtain reward, but it also has to explore in order to make better action selections in the future. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. The agent must try a variety of actions and progressively favour those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward.[7]

In reality, we only learn about what we like or dislike through trying out different options. This direct link of how AI agent learns and how we learns in real life makes it an attractive framework to model learning processes.

2.2 Deep Reinforcement Learning

To ensure RL algorithms can cope with large state space and non-linear value and policy functions, ANNs are combined with the RL algorithms. The resulting class of algorithms are called deep reinforcement learning[8]. Deep refers to the use of ANNs. The pioneer DRL algorithm is called

[7] Different RL algorithms have different ways of adding exploration.

[8] DRL algorithms have already been applied to a wide range of problems, such as robotics, where control policies for robots can now be learned directly from camera inputs in real world, succeeding controllers that used to be hand- engineered or learned from low-dimensional features of the robot's state. In a step towards even more capable agents, DRL has been used to create agents that can meta-learn ('learn to learn'), allowing them to generalise to complex visual environments they have never seen before (Arulkumaran et al., 2017).

deep Q network algorithm (Mnih et al., 2013), which is capable of human level performance on many Atari video games using unprocessed pixels for input. However, while deep Q network algorithm solves problems with high-dimensional observation spaces, it can only handle discrete and low-dimensional action spaces. Many tasks of interest have continuous (real valued) and high dimensional action spaces, including economic decision-making processes. To solve this issue, Lillicrap et al. (2015) introduces deep deterministic policy gradient (DDPG) algorithm, which is also the algorithm implemented in this exercise.

2.3 Applications of AI Technologies in Economics

The literature on applications of DRL in economics is scarce. A majority of the existing literature focuses on other machine learning methods. For example, Athey (2018) has a survey on the adaptations of machine learning techniques in economics with a particular focus on how machine learning can be used to enhance existing econometric methods. Deep learning (please note that deep learning is a component of but not the same as DRL) is used in stock market predictions. Lien Minh (2018) presents a framework for forecasting stock prices movements concerning financial news and sentiment dictionary. In another study, Go and Hong (2019) employ deep learning technique to forecast stock value streams while analysing patterns in stock price. Other applications of deep learning include but not limited to fraud detection in insurance industry, auction design, anti-money laundering in banking and online market. Deep learning is also adopted in forecasting macroeconomic indicators but these approaches require huge amounts of data and suffer from model dependency (Mosavi et al., 2020). Maliar et al. (2019) adopt deep learning to approximate Bellman function and then use supervised learning to train the neural network. Azinovic et al. (2020) apply deep neural networks to solve models with heterogeneity. Fernandez-Villaverde et al. (2020) also apply deep neural networks to solve high-dimensional dynamic programming problems.

It is apparent that most machine learning techniques are used in forecasting and predictions. A growing number of papers focus on using deep learning as a solution method for large-scale and heterogeneous economic models. Very few papers focus on applications of DRL algorithms in economics. Charpentier et al. (2020) provide some economic frameworks that could be applied with DRL techniques. This ranges from economic modelling to possible applications in operations research and game theory. They advocate that economic and financial problems would benefit from being reviewed using DRL techniques. Chen et al. (2021) adopt a deep reinforcement learning algorithm to solve a monetary model.

3 Methodology

In this section, I first describe an economic environment. I then show how the economic agent's learning process is modelled under an AI algorithm, namely DDPG.

3.1 Stochastic Optimal Growth Model

In a closed economy with one consumption/capital good, a representative consumer aims at maximising its lifetime utilities:

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{3.5}$$

subject to

$$c_t + k_{t+1} = z_t y_t \tag{3.6}$$

$$y_t = k_t^{\alpha} \tag{3.7}$$

$$c_t \geq 0 \tag{3.8}$$

$$k_{t+1} \geq 0 \tag{3.9}$$

for all t.

c_t, k_t, y_t are consumption, capital investment, and output produced in period t respectively. In this exercise, I use capital investment and saving interchangeably. Period utility $u()$ is increasing and strictly concave, i.e., $u' > 0$, and $u'' < 0$. β is the discount factor. Disturbance to the output, z_t, is a stochastic random variable, and takes the following form

$$z_t = e^{\mu + \rho ln(z_{t-1}) + \epsilon_t} \tag{3.10}$$

where ϵ_t takes a normal distribution, μ is a constant, and ρ is an autoregressive parameter.

I take a specific example of the stochastic optimal growth model with logarithmic utility and no capital depreciation. This specification is not a good representation of the real world, nor is it a model for policy experiments. However, it contains the central decision-making problem in economics, i.e., how to make consumption-investment decisions over a lifetime. As the foundation for many popular macroeconomic models[9], it is a natural starting point to show AI implementation in economic modelling. Moreover, this specification has an analytical solution, which can be used to show how an AI agent generates different and interesting behaviours compared to its rational expectation counterpart formulated by the analytical solution.

3.2 Optimisation under Rational Expectation

The bellman equation of this problem is as follows.

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta E_t v(k_{t+1}, z_{t+1})\} \tag{3.11}$$

The solution[10] to this problem is:

[9] To name a couple, real business cycle model, and incomplete market model.
[10] See appendix for detailed derivation.

$$k_{t+1} = \alpha\beta z_t k_t^{\alpha} \tag{3.12}$$

The value function following this policy is

$$\begin{aligned} v^*(k,z) = \frac{1}{1-\beta}\left[log(1-\alpha\beta) + \frac{\alpha\beta}{1-\alpha\beta}log\alpha\beta + \frac{\beta\mu}{(1-\alpha\beta)(1-\beta\rho)}\right] \\ + \frac{\alpha}{1-\alpha\beta}logk + \frac{1}{(1-\alpha\beta)(1-\beta\rho)}logz \end{aligned} \tag{3.13}$$

3.3 AI Implementation

The DRL algorithm adopted here is DDPG, first introduced by Lillicrap et al. (2015). To implement this algorithm, I first need to translate the economic model into RL components that are introduced in section 2.1, which are presented in table 1.

Table 1: RL components

RL components	**Economic environment**	**Description**
State	$z_t k_t^{\alpha}$	total goods available to consume at period t
Actions	a_t	proportion of the total goods that the agent is willing to consume at period t
Rewards	$ln(c_t)$	period utility at period t, where $c_t = a_t z_t k_t^{\alpha}$
Next State	$z_{t+1} k_{t+1}^{\alpha}$	total goods available to consume at period $t+1$, where $k_{t+1} = (1-a_t) z_t k_t^{\alpha}$
Policy function	- to be learnt	approximated by an ANN, called the actor network
Value function	- to be learnt	approximated by an ANN, called the critic network

An AI agent does not know what form of preference it has, nor the fundamentals of the economy. It must gather these information by taking an action, i.e., making a consumption-investment decision, each period. How it decides what action to take given each state depends on its policy function, approximated by the policy neural network. More specifically, the algorithm maintains a parameterised actor function $\mu(s|\theta^{\mu})$, which specifies the current policy by deterministically

mapping states to a specific action given some parameter θ^{μ}. Given that an AI agent, at the beginning of a learning process, knows nothing or very little about what action constitutes a high reward and lifetime utilities[11], it has to try many different actions at each state to have a good idea of what works best. This depends crucially on the agent's ability to explore its action space.

To make sure that the agent is exploring its action space, an exploration policy μ' is constructed by adding a noise process $\mathcal{N}$ to the actor policy

$$\mu'(s_t) = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t. \tag{3.14}$$

This noise could be sampled from an uncorrelated Gaussian process or a correlated Ornstein-Uhlenbeck (OU) process. Following Lillicrap et al. (2015), the original paper that constructed DDPG algorithm, $\mathcal{N}$ is sampled from a discretised Ornstein-Uhlenbeck (OU) process. There is a strain of literature in computer science solely focus on different exploration strategies to achieve the best performance for a given task. It is out of the scope of this current exercise, and not discussed in details.

3.3.1 Full Algorithm and Sequence of Events

The full algorithm[12] follows three steps:

Step I: Initialisation

- Sep up two neural networks: an actor network $\mu(s|\theta^{\mu})$ takes the argument of state and output an action; a critic network $Q(s, a|\theta^{Q})$ takes the argument of a state-action pair and output a value.
- θ^{μ} and θ^{Q} represent the parameters of the two networks respectively. Both are initialised randomly. Both parameters update during the learning process so that the networks will move towards the true policy and value functions.
- Define a replay buffer $\mathcal{B}$, which is a memory that stores information (called transitions in the DRL literature) collected by a DRL agent during the agent-environment interactive process. A transition is characterised by a sequence of variables (s_t, a_t, r_t, s_{t+1}).
- Define a length of N, which is the size of a mini-batch. Mini-batch refers to a sample from the memory.
- Define the total number of episodes E.
- Define a simulation period of T for each episode, where $T > N$.

For each episode, loop over step II and III.

Step II: The AI agent starts to interact with the environment.

[11] Lifetime utilities, i.e., the value function, is approximated by the other neural network, namely critic network.

[12] I lay out the skeleton of the DDPG algorithm that implemented in this exercise. There are further details, such as a 'soft update' method to ensure a stable neural network learning process which is not discussed here.

- The agent observes a state (total goods available to consume) $s_t = z_t k_t^{\alpha}$. It then selects an action (the proportion of the total goods that it is willing to consume) $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$ according to current policy and exploration noise.
- Execute action a_t, and observe a reward $r_t = ln(c_t)$ and the next state $s_{t+1} = z_{t+1} k_{t+1}^{\alpha}$.
- Store a transition (s_t, a_t, r_t, s_{t+1}) in the memory $\mathcal{B}$.

Step III: Training the AI agent/ the AI agent starts to learn for period $N \leq t \leq T$.

- Sample a random mini-batch of N transitions (s_i, a_i, r_i, s_{i+1}) from the memory $\mathcal{B}$.
- Calculate a value y_i for each transition i following

$$y_i = r_i + \beta Q^{\mu}(s_{i+1}, \mu(s_{i+1}|\theta^{\mu})|\theta^{Q}) \tag{3.15}$$

  for all $i \in N$, where $Q^{\mu}(s_{i+1}, \mu(s_{i+1}|\theta^{\mu})|\theta^{Q})$ is a prediction made by the critic network with state-action pair $(s_{i+1}, \mu(s_{i+1}|\theta^{\mu}))$, and $\mu(s_{i+1}|\theta^{\mu})$ is a prediction made by the actor network with input s_{i+1}. The series of y_i is similar to the right hand side of equation 2.4, i.e., the Bellman optimality equation.
- Obtain $Q(s_i, a_i|\theta^{Q})$ from the critic network with input state-action pair (s_i, a_i), which is the left hand side of equation 2.4.
- Calculate the average loss for this sample of N transitions

$$L = \frac{1}{N} \sum_i \big(y_i - Q(s_i, a_i|\theta^{Q})\big)^2. \tag{3.16}$$

- Update the critic network with the objective of minimising the loss function L.[13]
- One way of measuring the performance of a policy is through the value function, as defined by equation (2.2). Therefore, for the policy function, i.e., the actor network, the objective is to maximise the expected return, denoted as $J(\theta^{\mu})$

$$J(\theta^{\mu}) = Q^{\mu}(s_i, \mu(s_i|\theta^{\mu})|\theta^{Q}). \tag{3.17}$$

- This objective function could also be rephrased as minimising $-J(\theta^{\mu})$. Update the actor network parameters with the objective of minimising $-J(\theta^{\mu})$.[14]

[13] This process involves applying back propagation to calculate the amount of error to which each weight in an ANN contribute to. Then apply gradient descent to minimise the gradient of the loss function with respect to the weights of ANN. Please see appendix for further information.

[14] Similar to the critic network, the specific steps of updating ANN's parameters by minimising an objective function involves back propagation and gradient descent. Further information can be found in the appendix.

3.4 Parameters and Learning Agent's Characteristics

The main parameters are presented in table 2.

Table 2: Main Parameters

Parameters	Baseline Agent
Output elasticity of capital α	0.4
Shock location parameter μ	0.1
Autoregressive factor ρ	0
Discount Factor β	0.99
Learning Rate η	actor network $1e-4$; critic network: $1e-3$
Exploration Level	0.2

The neural network uses back propagation algorithm to compute how the error changes with respect to each weight

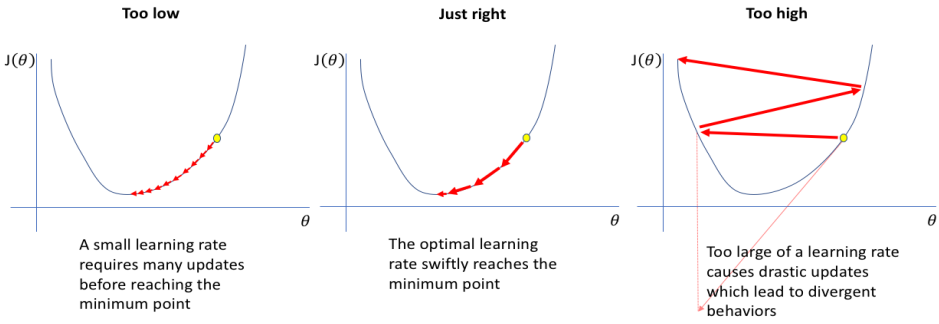
$$\frac{\partial Error}{\partial w_{ij}} \tag{3.18}$$

In order to update each weight of the network, use the update rule

$$w'_{ij} = w_{ij} - \eta \frac{\partial Error}{\partial w_{ij}} \tag{3.19}$$

The new weight w'_{ij} is a combination of the current weight w_{ij} and some weighted term $\eta \frac{dError}{dw_{ij}}$. η is the learning rate parameter. It governs how quickly a neural network updates its weights. The higher the learning rate, the quicker the update is.

Figure 2: Learning rates



Source: jeremyjordan

To clearly illustrate the purpose of η, figure 2 shows that when the learning rate is too high, it

causes a divergence and fails to reach the appropriate weights; with a very low learning rate, the learning process takes very long before it reaches the optimal solution.

Exploration level is measured by the noise attached to the action, as specified in equation (3.14). In this case, exploration level means the standard deviation of the noise process.

Exploration and learning rate parameters are crucial in this paper, not only because they aid the learning process of the neural networks but also because they add sophistication in the AI agent's learning behaviour, which opens up an unstudied path in economic modelling. Taking the exploration parameter as an example, I first illustrate the logic behind it and then back it up with several experiments in the next section.

The parameters can be interpreted as AI agents' different personalities/characteristics, which generate different learning behaviours with welfare distinctions. Exploration parameter represents how an AI agent gathers information. An AI agent can be more or less adventurous in exploring available actions (and what these actions lead to). With a higher exploration level, the agent is 'willing' to take actions that it has not previously tested, and thus increase the probability of finding a better action (measured by rewards). However, this could also be risky to the agent and it may be left in a worse place than before. With a low exploration level, an agent is unlikely to try anything new, and may never uncover the state-action pairs that contribute to high rewards. This characteristic also allows the AI agent to be alert of any changes in its environment. If a change occurs, for example the autoregressive parameter in z_t (recall that $z_t = e^{\mu+\rho ln(z_{t-1})+\epsilon_t}$) changes, an AI agent with the ability to explore will notice such changes and adjust its future actions (and hence policy function) accordingly. This way of information collection and processing makes it possible to build models with AI agents and experiment on the effects of regime changes without the assumption of full rationality.

4 Experiments and Results

To illustrate AI agents' abilities in adapting to changes in their environments, I conduct several simulation experiments. The changes in an environment are introduced through the stochastic process z_t in the economy, recall equation (3.10)

$$z_t = e^{\mu+\rho ln(z_{t-1})+\epsilon_t}.$$

More specifically, I run the following simulations.

- Transitory shock: in an environment with $z_t = e^{0.1+\epsilon_t}$, impose a one-time change to the mean and resulting $z_t = e^{3+\epsilon_t}$. Observe AI agents' consumption behaviours in relation to this transitory change.
- Permanent change: shift the stochastic process z_t from $z_t = e^{0.1+\epsilon_t}$ to $z_t = e^{0.1+0.7lnz_{t-1}\epsilon_t}$. Observe AI agents' consumption behaviours in relation to this permanent change.
- Impose a sequence of changes in the stochastic process to demonstrate that observed regularities in an environment with one shock/change carry through to cases when agents live in an environment with multiple changes.

In the following subsections, I present results for the aforementioned simulation experiments. All results are presented for three types of AI agents differing in their exploration levels (i.e., how they collect information), accentuating the importance of exploration parameter in generating different learning dynamics and determining the welfare of AI agents.

4.1 Transitory Shock

Figure 3: The stochastic process with a transitory shock

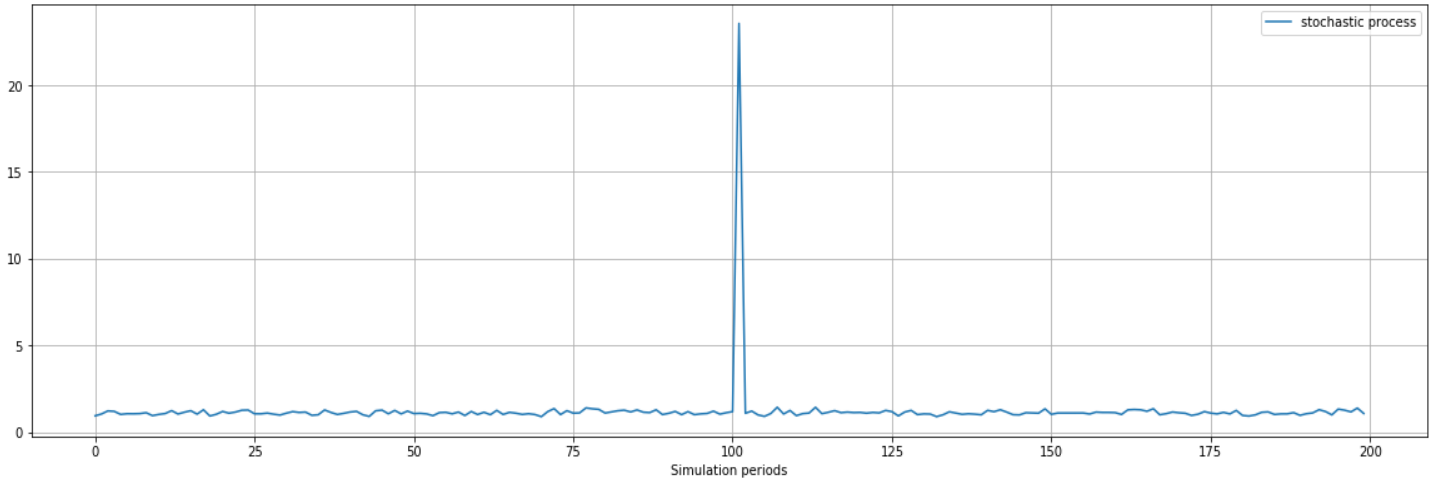


Figure 4: AI agents' consumption paths facing a transitory shock

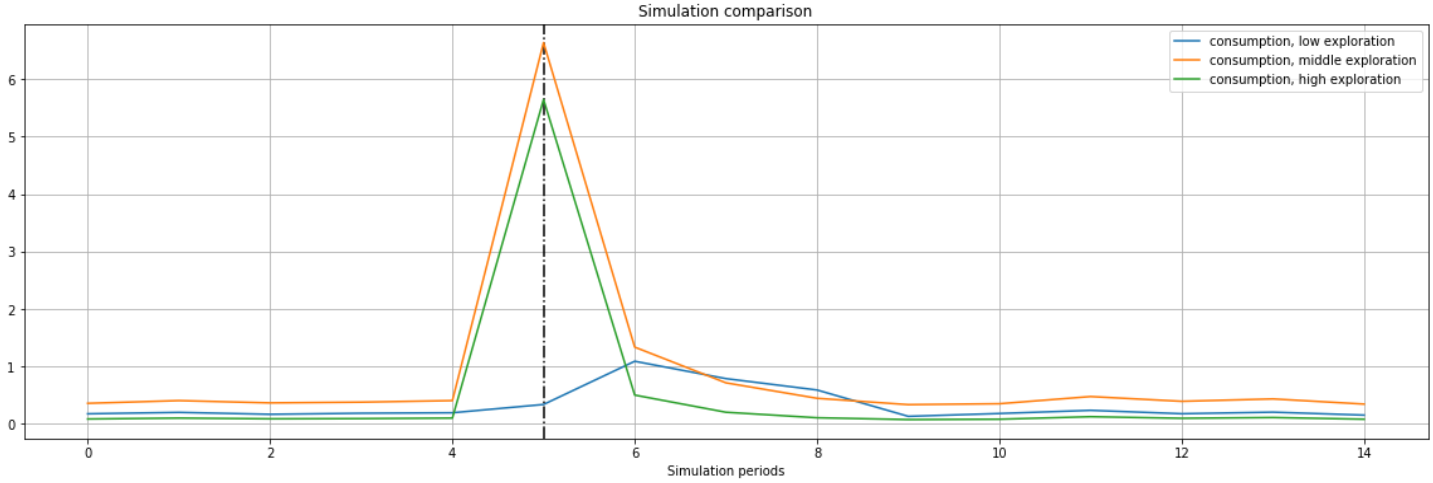


Figure 3 shows the stochastic process in this environment. The process follows $z_t = e^{0.1+\epsilon_t}$ except for simulation period 100 where $z_t = e^{3+\epsilon_t}$. ϵ_t follows a normal distribution with $N(0, 0.1)$. The x-axis represents simulation periods. To clearly show agents' responses, I plot simulation data for the 15 periods around the transitory shock. Hence the x-axis values of figure 4 and 5 are different from figure 3. Figure 4 plots AI agents' consumption paths in this environment. The transitory shock is unknown to them before it hits. The black vertical dash line represents the period when the positive transitory shock is realised. All three agents with different exploration level exhibit

similar overall consumption behaviours, namely consumption increases with the positive shock and revert back to the pre-shock level after a few periods. The timing and magnitude of their responses are different. The middle- and high-exploration agents respond more swiftly than the low exploration agent, which attests that with higher exploration, an agent is more alert to changes in the environment and hence responds quickly. The low-exploration agent responds with a lag, as shown by the blue line. The magnitude of their responses is also correlated to their respective exploration levels. Low-exploration agent responds in a slower and less prominent manner than the other two agents.

Figure 5: AI agents' utilities facing a transitory shock

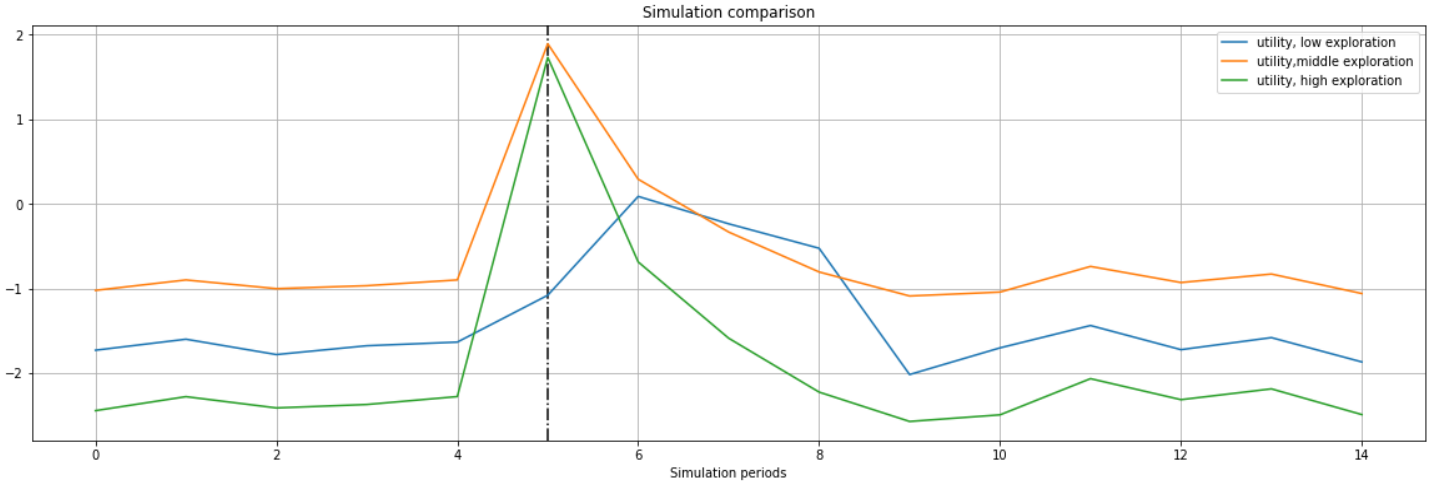


Figure 5 plots all three agents' utility as a measurement of their welfare. The green-line agent (high exploration) does better than the low-exploration agent during the transitory shock period. However, this does not hold for periods before and after the transitory shock. Its overly adventurous nature leads to a behaviour with high excess investment (very low consumption) and thus a lower utility level than the middle agent. The middle agent (orange line) has the highest utility, and balances exploration and exploitation of existing knowledge. If the agent reduces its level of exploration, as shown by the blue line in the figure, it sacrifices its welfare in support of its cautious behaviour and only try actions that it has tested before.

This transitory shock can be interpreted as a positive productivity shock. With a positive productivity shock, a temporary increase in consumption is seen. Interestingly, without any further assumptions, a lagged response can be generated simply through varying AI agents' exploration parameters (i.e., the low-exploration agent in blue in figure 4).

How would the agents respond in an environment with a permanent change?

4.2 Permanent Shock

Figure 6: The stochastic process with a permanent change

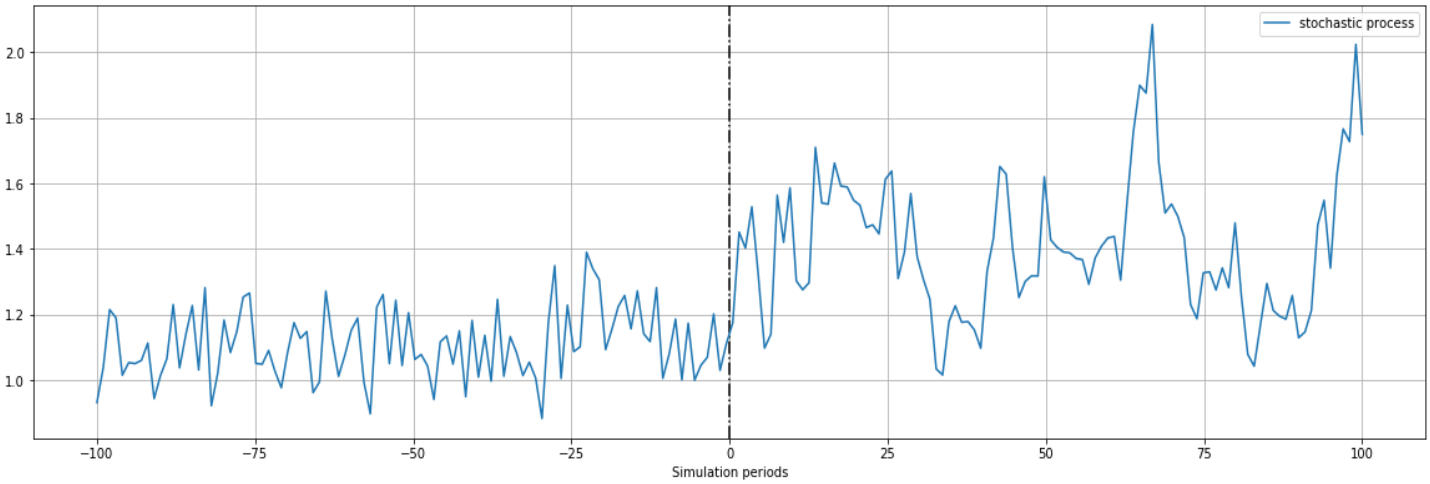


Figure 7: Learning agent in an environment with a permanent change

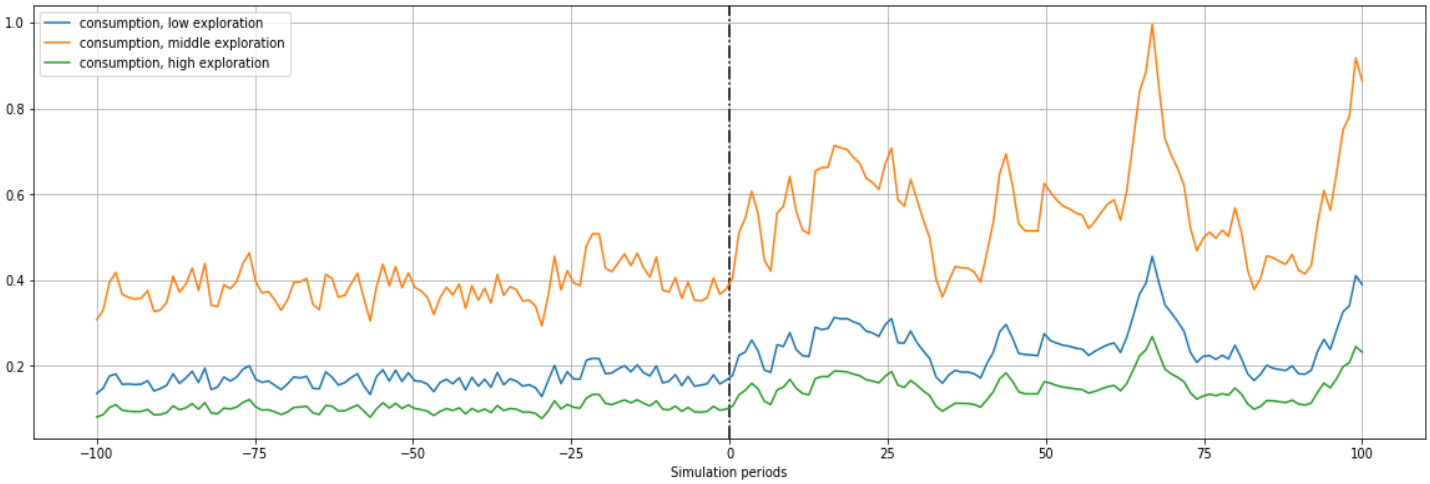


Figure 6 shows the stochastic process changes from $z_t = e^{0.1+\epsilon_t}$ to $z_t = e^{0.1+0.7lnz_{t-1}+\epsilon_t}$. The black dash line indicates the period when the change happens. Given this permanent change of stochastic process, all three AI agents modify their consumption levels permanently, as indicated by figure 7. AI agents' behaviours in environments with transitory and permanent changes echo with Milton Friedman's permanent income hypothesis, and that a permanent income change (rather than a temporary one) drives the change in a consumer's consumption smoothing behaviour (Friedman, 1957).

Figure 8: Learning agent in an environment with a permanent change

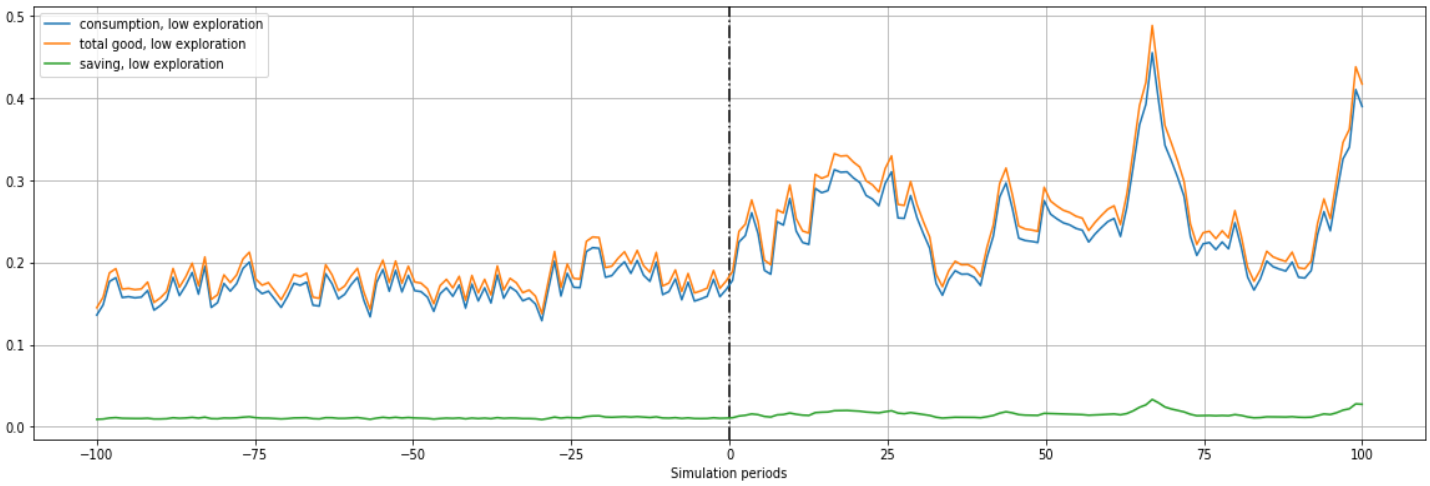


Figure 9: Learning agent in an environment with a permanent change

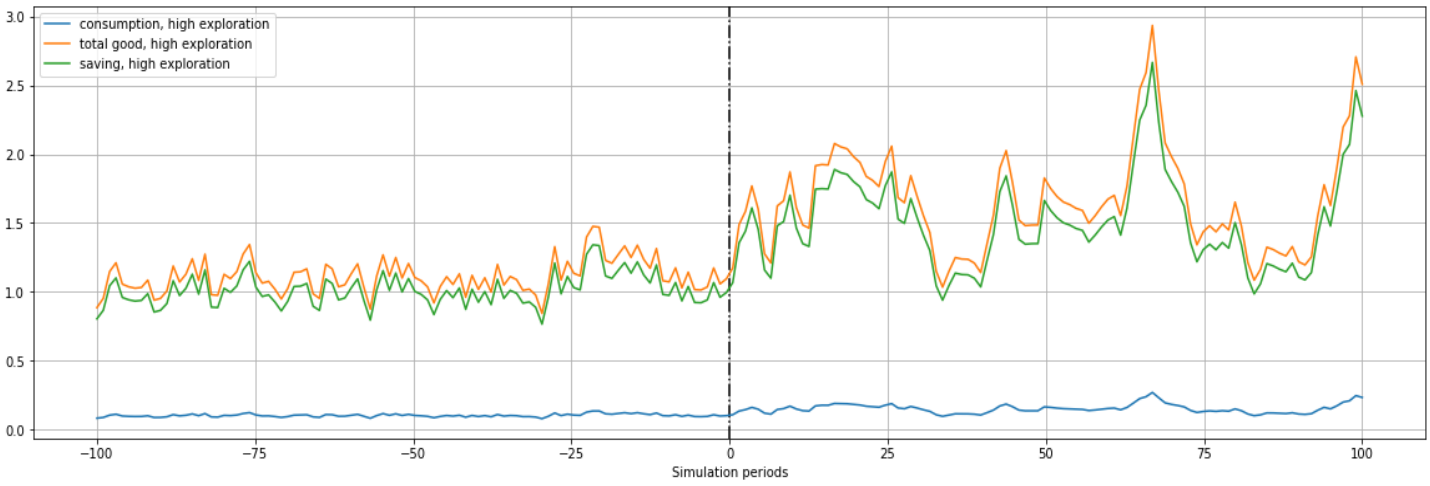


Figure 10: Learning agent in an environment with a permanent change

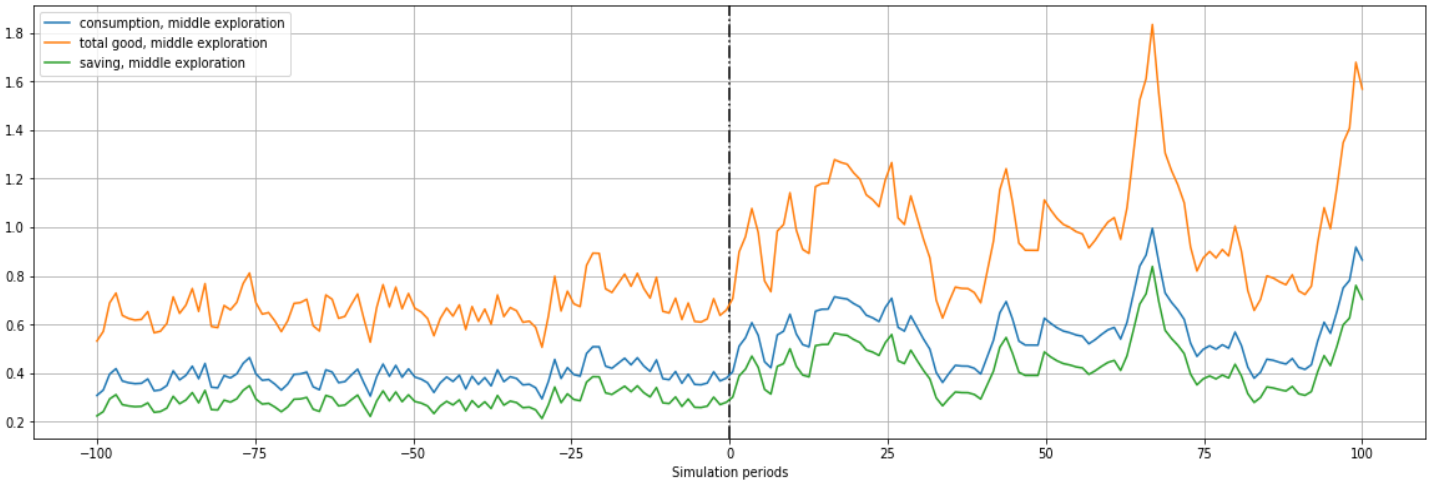


Figure 8 plots low-exploration agent's simulated series of consumption, investment, and total available resource each period. As clearly shown, it consumes almost all of its available resource each period. This is reversed in figure 9. The high-exploration agent invests nearly all its available resource each period. The middle-exploration agent takes the middle ground, as shown in figure 10. Their respective behaviours contribute to their welfare distinctions as shown in figure 11.

Figure 11: Learning agent in an environment with a permanent change

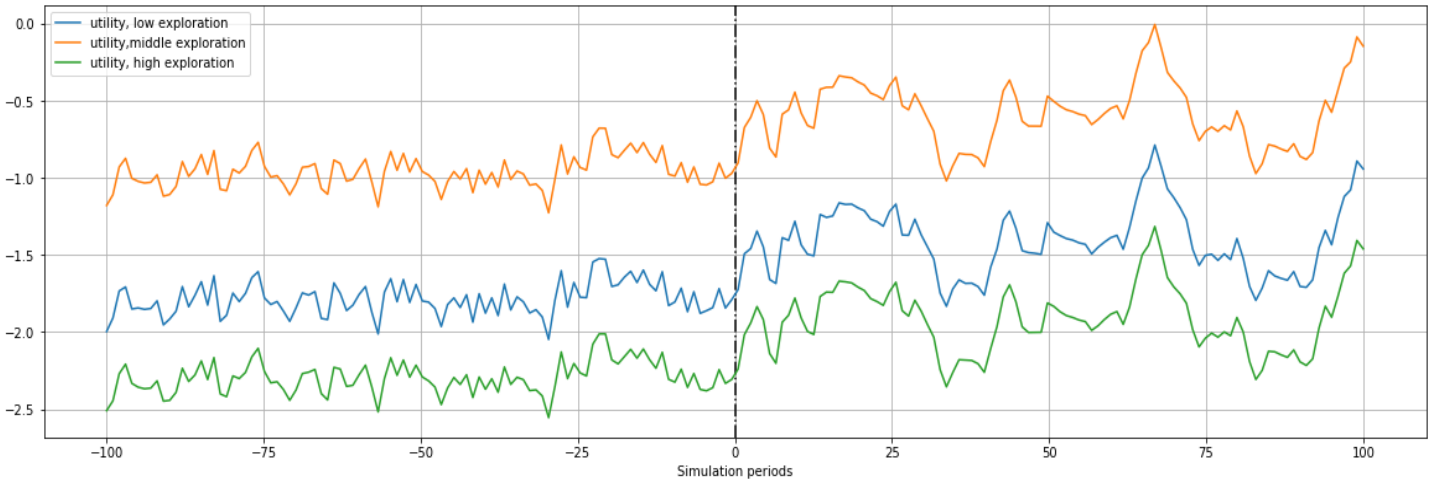
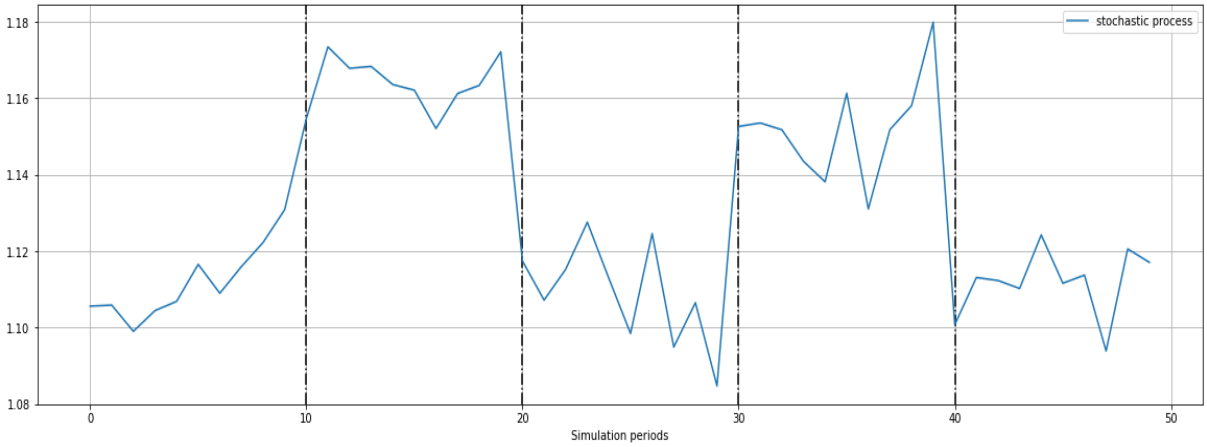


Figure 11 shows all three agents' utility in this environment with a permanent change in the stochastic process. As anticipated, the middle-exploration agent balances exploring the unknown action space and exploiting the known domain to achieve the highest utility. However, the agent with low and high level of exploration sacrifice their welfare to support their overly cautious or adventurous behaviours.

4.3 Multiple Changes in An Environment

Figure 12: Simulated consumption paths for all three agents



In this section, the three AI agents (low, middle, and high exploration) act in an environment with 4 permanent stochastic process changes (figure 12). The dotted vertical lines signal the timing when

the four changes happen. All agents live in the same environment with the same stochastic process but they do not interfere with each other's choices. All four permanent changes of the stochastic process are unknown to the AI agents. The purpose of this experiment is to show that AI agents can adapt to, not only one, but multiple unforeseen changes in the environment. Differing in their exploration level, all three agents behave as expected and similar to the results shown in section 4.1 and 4.2.

Figure 13: Simulated consumption paths for all three agents

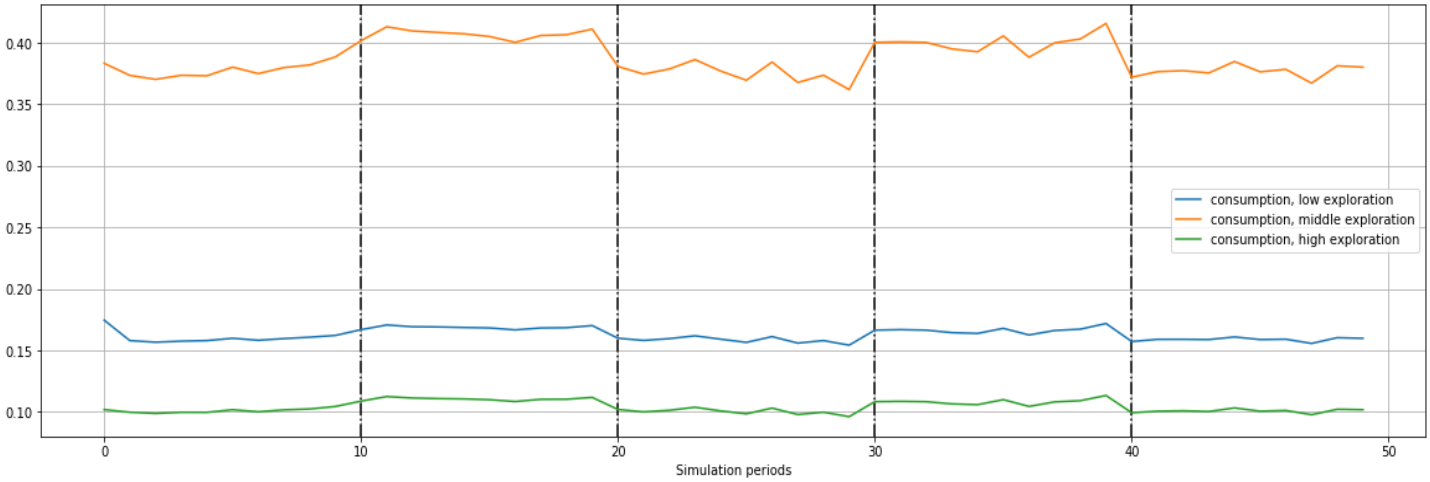


Figure 14: Simulated capital investment paths for all three agents

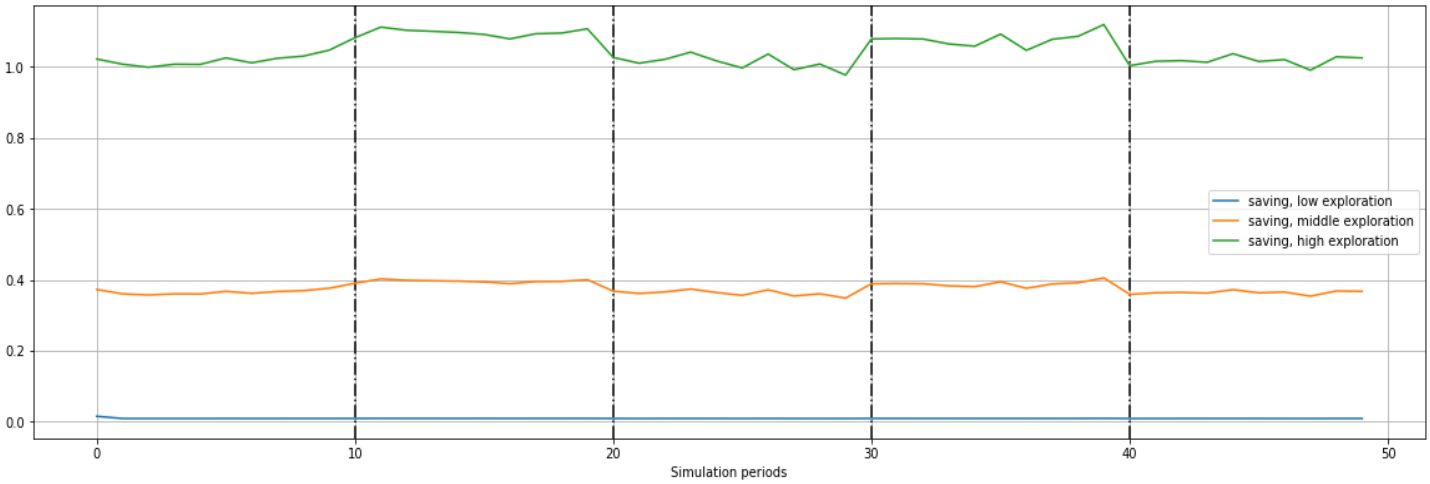


Figure 15: Rewards/utility for all three agents

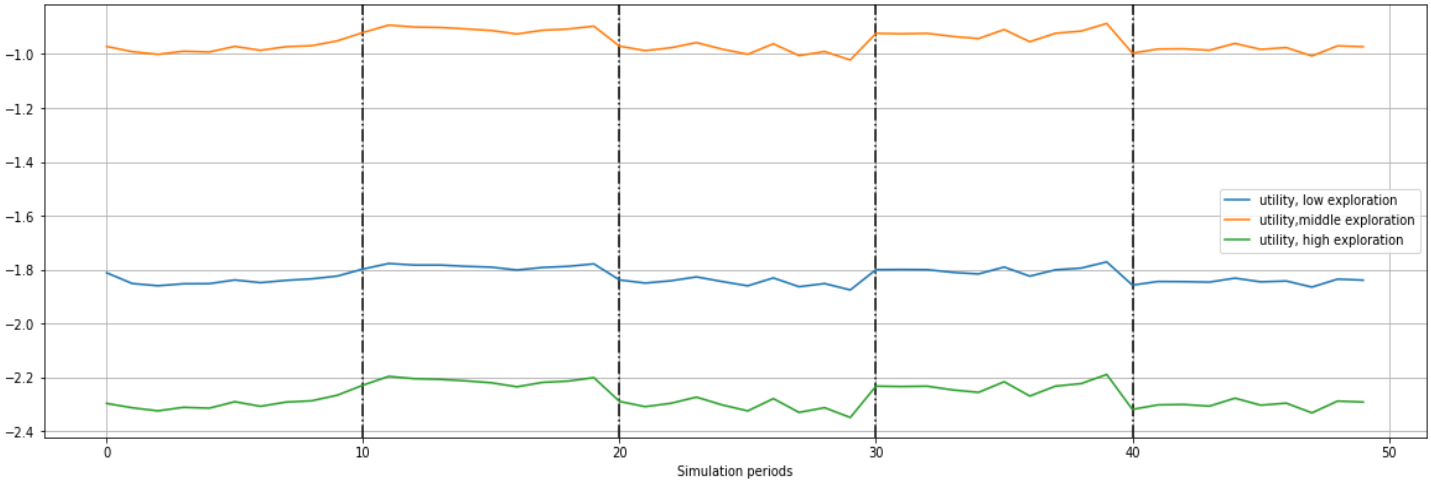


Figure 13 plots average consumption paths of all agents. Figure 14 plots average capital investment paths of all agents. The high-exploration agent saves the most, and this behaviour is consistent across all four changes in the stochastic process. The middle-exploration agent gains the highest welfare, as shown in figure 15.

One important issue is that exploration level is a relative term and subjective to a particular environment or a decision-making problem. A high-level exploration in this setting could mean a low-level one in another problem. Therefore, it is important, when applying this technique, to experiment with many different levels of exploration.

5 Comparisons with One Agent under Rational Expectation

AI agents in this paper are born in an environment that they have no prior information. These include state space, action space, reward function, and the fundamentals of an economy. They get to know all these information through interacting with an environment and maximise the rewards that they receive. Moreover, to make sure that they learn all the available choices given a particular state, the agent must explore, i.e., trying out unknown actions. All these behaviours differ from an economic agent under rational expectation assumption. In this section, I compare AI agent's behaviour with an agent under rational expectation assumption (RE agent). More specifically, I first compare the learnt/approximated policy function with the analytical solution of the given economic model. I then compare the simulated path of the AI agents and the RE agent, and show how their consumption decisions differ.

Figure 16: Approximated policy functions

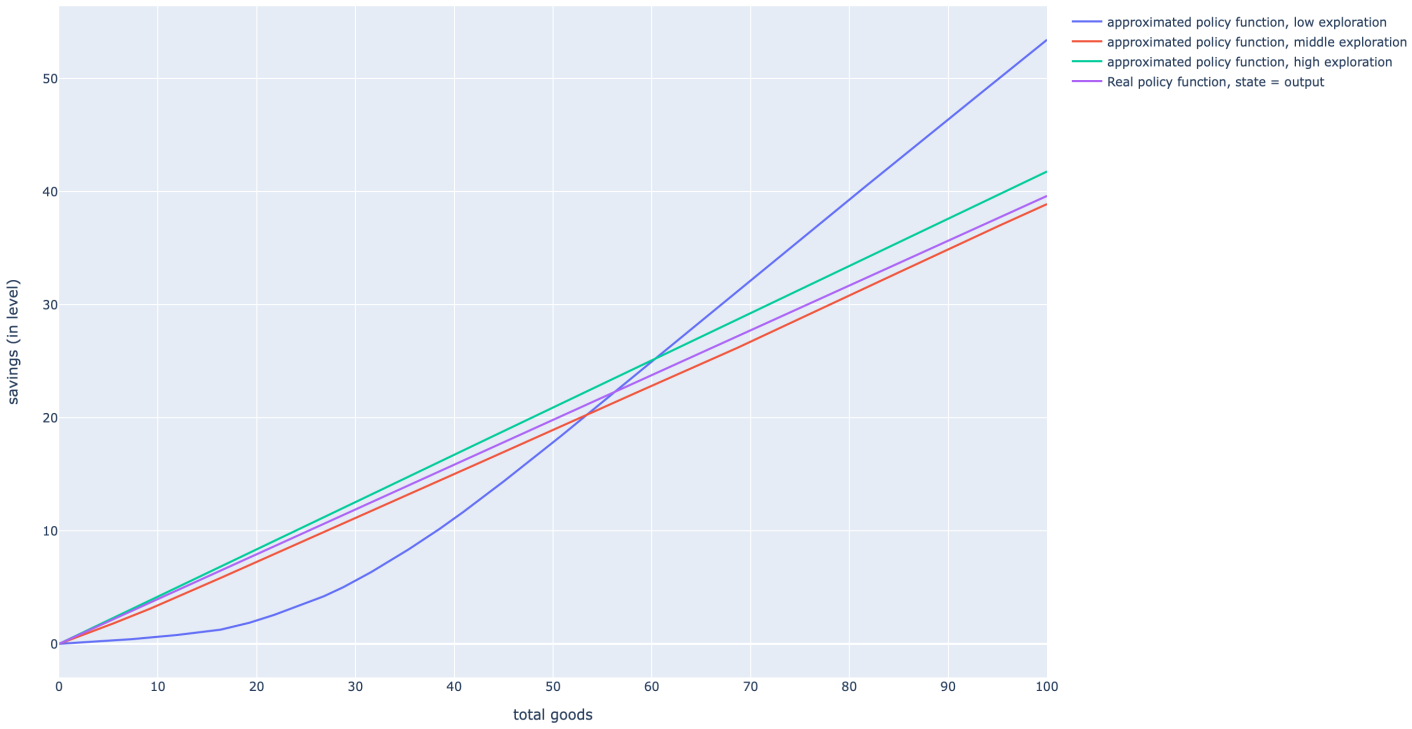


Figure 17: Distance metrics

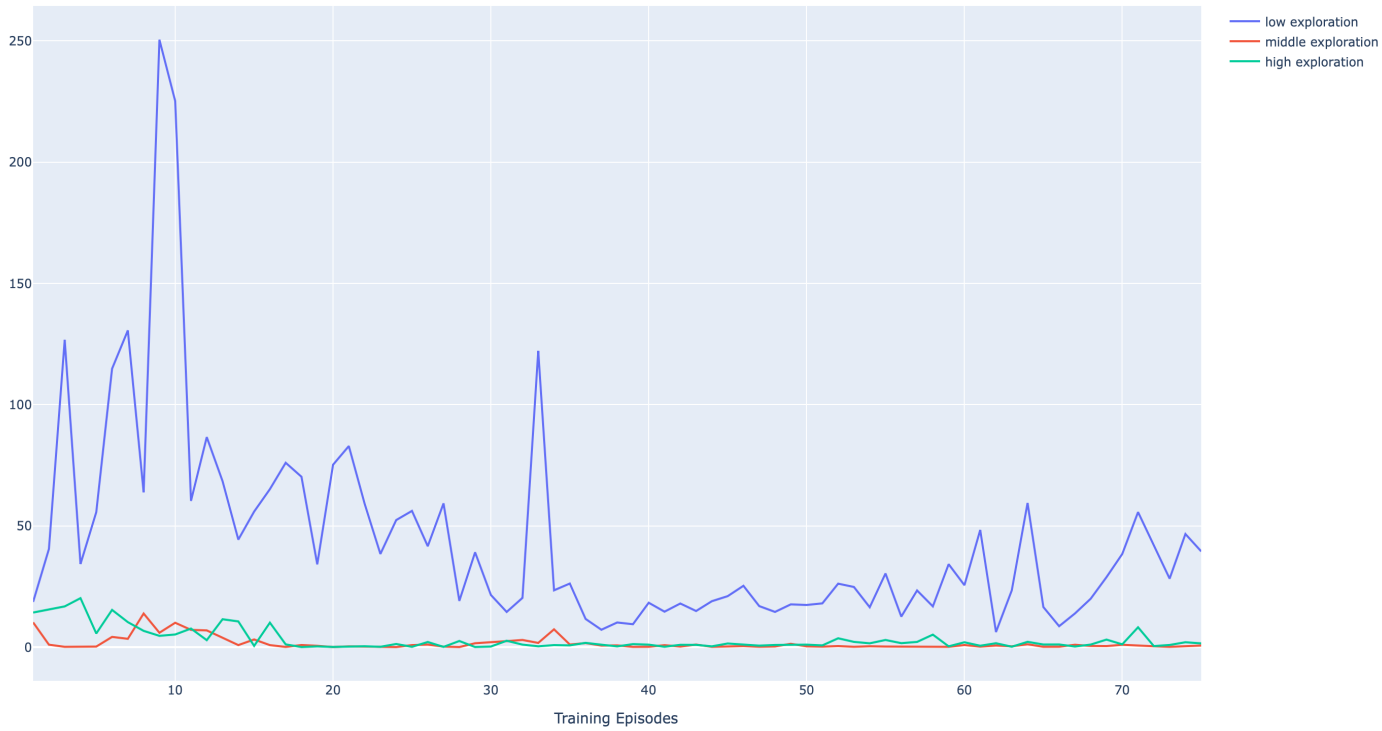


Figure 16 plots the approximated policy functions for three AI agents (with different exploration levels) and the analytical solution policy of this stochastic optimal growth problem (i.e., equation

3.12). All three AI agents have been learning for the same amount of periods. It shows that the AI agents' learnt policy functions can be very close to the solution of the problem with the middle exploration parameter (red line agent). The approximated policy can also be quite different if the agent is overly cautious or adventurous in choosing their actions (blue and green lines). Figure 17 plots a distance metrics calculated between the analytical solution policy and the approximated policies at each episode as illustrated by equation (5.20). k_g^* represents the analytical solution policy function value at a grid g, and k_g is the approximated policy function at the same grid. G denotes total number of grids. d_e denotes the distance between analytical solution and approximated policy function at episode number e.

$$d_e = \frac{1}{G}\sum_{g=1}^{G}(k_g^* - k_g)^2 \tag{5.20}$$

In figure 17, the x-axis denotes the number of training episodes, i.e., how long the AI agent has been living and learning in an environment, and the y-axis denotes the distance calculated. It shows that as the number of training episodes increases, the distance becomes smaller. In addition, the middle exploration agent learns the fastest, that is, the distance becomes smaller at earlier episodes than the high and low exploration agents.

Figure 18: AI agent vs RE agent consumption paths

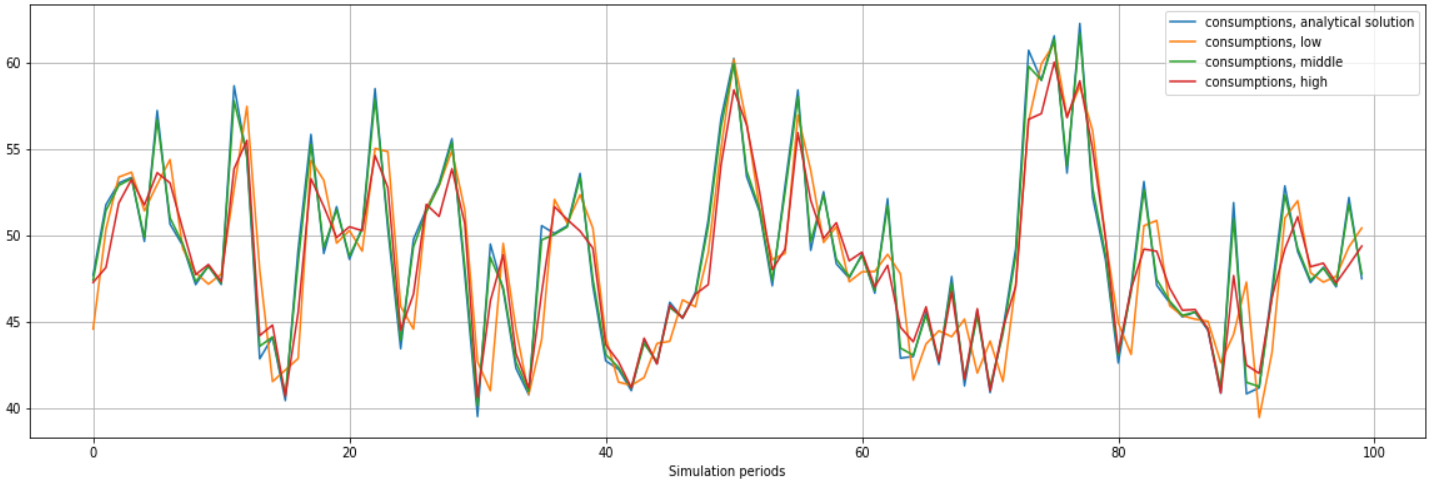


Figure 18 plots simulation comparisons among three AI agents and the RE agent in the same stochastic environment. All agents follow the same initial condition and behave following their respective policies in subsequent periods. It shows that the middle exploration agent is doing almost as well as the RE agent in terms of the consumption level, which determines agents' welfare.

6 Summary and Future Work

In this paper, I show how an economic agent learns to act and optimise in an unknown environment and how it could adapt to changes using an AI technique, a deep reinforcement learning algorithm. Through this algorithm, I model how the AI agent collects and processes information. I adopt

a version of the stochastic optimal growth model where an economic agent needs to make consumption-investment decisions to maximise its lifetime utilities. This agent, however, does not follow the rational expectation hypothesis. It does not know the fundamentals of the economy. It is physically constraint to take an optimal action because it does not possess any information on what is feasible or desirable in its choice set. The agent could only learn how to make an optimal decision through acquiring information by interacting with the environment for many periods and processing it through artificial neural networks.

Several experiments are conducted. To show how AI agents respond to permanent and transitory changes in the environment, I impose a transitory shock and a permanent change to the stochastic process. The AI agents' behaviours echo with the permanent income hypothesis, and that a transitory shock leads to a temporary response from the agent and nothing permanent. Whereas the permanent change in the stochastic process leads to a sustained shifts in agents' consumption behaviours. In addition, to highlight the purpose and novelty of the exploration parameter, I run all experiments on three AI agents differing in their levels of exploration and how they acquire information. With an appropriate exploration level, the agent achieves a high level of welfare measured by its utility. With a higher level of exploration, the agent is overly adventurous, and sacrifices its welfare for an unknown/untested action. With a lower level of exploration, the agent is too cautious and unwilling to try anything unheard of and thus is unable to fully explore and find actions that lead to high rewards. In the end, I show a comparison of policy functions and simulated behaviours between AI agents who have the ability to explore and the rational expectation agent. Their policy functions could be very similar given an appropriate exploration parameter. This is affirmed by their behaviours in a simulated environment.

This work provides a new possibility to model the decision-making process of an economic agent. It relaxes the rational expectation assumption, and models the learning behaviour of an artificial agent in terms of how it collects and processes information. The AI agent collects information through exploring the environment and processes information through artificial neural networks. It is highly relevant, and supports further studies on important economic questions that include structural breaks, regime changes, and multi-agent learning. The algorithm by design is able to handle models with large state and action spaces that are often useful in policy analyses. Owing to artificial neural networks, it thrives in environment with non-linear policy and value functions. Lastly but not least, I implement one out of a class of DRL algorithms, which is a fast-evolving literature at the frontier of AI technologies.

References

D. Abel. Concepts in Bounded Rationality: Perspectives from Reinforcement Learning. *Brown University Master thesis*, 2019.

K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *CoRR*, abs/1708.05866, 2017. URL `http://arxiv.org/abs/1708.05866`.

S. Athey. *The Impact of Machine Learning on Economics*, pages 507–547. University of Chicago Press, January 2018. doi: https://doi.org/10.7208/chicago/9780226613475.001.0001. URL `http://www.nber.org/chapters/c14009`.

M. Azinovic, L. Gaegauf, and S. Scheidegger. Deep equilibrium nets. 2020. `http://dx.doi.org/10.2139/ssrn.3393482`.

A. Charpentier, R. Elie, and C. Remlinger. Reinforcement learning in economics and finance. 2020.

M. Chen, A. Joseph, M. Kumhof, X. Pan, R. Shi, and X. Zhou. Deep reinforcement learning in a monetary model, 2021.

J. Fernandez-Villaverde, G. Nuno, and G. Sorg-Langhans. Solving high-dimensional dynamic programming problems using deep learning. 2020. `https://maximilianvogler.github.io/My_Website/Deep_Learning.pdf`.

M. Friedman. *A Theory of the Consumption Function*. Princeton University Press, 1957. `https://www.nber.org/books-and-chapters/theory-consumption-function`.

Y. Go and J. Hong. Prediction of stock value using pattern matching algorithm based on deep learning. pages 31–35, 2019.

D. Lien Minh. Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. pages 55392–55404, 2018.

T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv e-prints*, (arXiv), 2015.

L. Maliar, S. Maliar, and P. Winant. Will Artificial Intelligence Replace Computational Economists Any Time Soon? *CEPR Discussion Paper Series*, DP14024 v.3(3-4), 2019.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. 2013.

A. Mosavi, P. Ghamisi, Y. Faghan, P. Duan, and S. Shamshirband. Comprehensive review of deep reinforcement learning methods and applications in economics. Mar 2020. doi: 10.20944/preprints202003.0309.v1. URL `http://dx.doi.org/10.20944/preprints202003.0309.v1`.

P. Russell, Stuart J.; Norvig. *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall, 2020.

T. Sargent. *Bounded Rationality in Macroeconomics*. Oxford University Press, 1993.

H. A. Simon. *Behavioural Economics*, pages 1–9. Palgrave Macmillan UK, London, 2016. ISBN 978-1-349-95121-5. doi: 10.1057/978-1-349-95121-5_413-1. URL https://doi.org/10.1057/978-1-349-95121-5_413-1.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

A Appendix

A.1 Derivation of Analytical Solution of the Stochastic Optimal Growth Model

Given the bellman equation:

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta E_t v(k_{t+1}, z_{t+1})\} \tag{1.21}$$

Guess the value function to be the form $v(s) = A + Blog(k) + Dlog(z)$, and substitute to the right hand side of the bellman equation

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta E_t (A + Blog(k_{t+1}) + Dlog(z_{t+1}))\} \tag{1.22}$$

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta (A + Blog(k_{t+1}) + DE_t log(z_{t+1}))\} \tag{1.23}$$

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta (A + Blog(k_{t+1}) + D\mu)\} \tag{1.24}$$

The F.O.C:

$$\frac{\partial v(k_t, z_t)}{\partial k_{t+1}} = 0 \rightarrow -\frac{1}{z_t k_t - k_{t+1}} + \beta \frac{B}{k_{t+1}} = 0 \tag{1.25}$$

$$k_{t+1} = \frac{\beta B}{1 + \beta B} z_t k_t \tag{1.26}$$

Apply the envelop theorem

$$\frac{B}{k_t} = \frac{z_t}{z_t k_t - k_{t+1}} \rightarrow B = \frac{z_t k_t}{z_t k_t - k_{t+1}} \tag{1.27}$$

k_{t+1} can then be derived as

$$k_{t+1} = \beta z_t k_t \tag{1.28}$$

If the guessed form of value function is the solution then it must satisfy

$$A + Blog(k_t) + Dlog(z_t) = log(z_t k_t - \beta z_t k_t) + \beta (A + Blog(\beta z_t k_t) + D\mu) \tag{1.29}$$

$$A = \frac{1}{1 - \beta} (log(1 - \beta) + \frac{\beta}{1 - \beta} log\beta + \frac{\beta \mu}{1 - \beta}) \tag{1.30}$$

$$B = D = \frac{1}{1 - \beta} \tag{1.31}$$

The value function is thus

$$v^*(k,z) = \frac{1}{1-\beta}\{log(1-\alpha\beta) + \frac{\alpha\beta}{1-\alpha\beta}log\alpha\beta + \frac{\beta\mu}{1-\alpha\beta}\} + \frac{\alpha}{1-\alpha\beta}logk + \frac{1}{1-\alpha\beta}logz \quad (1.32)$$

When z_t follows an autoregressive process, i.e. $z_t = e^{\mu+\rho lnz_{t-1}+\epsilon_t}$, where ϵ_t follows a standard normal distribution, the analytical solution is derived analogously.

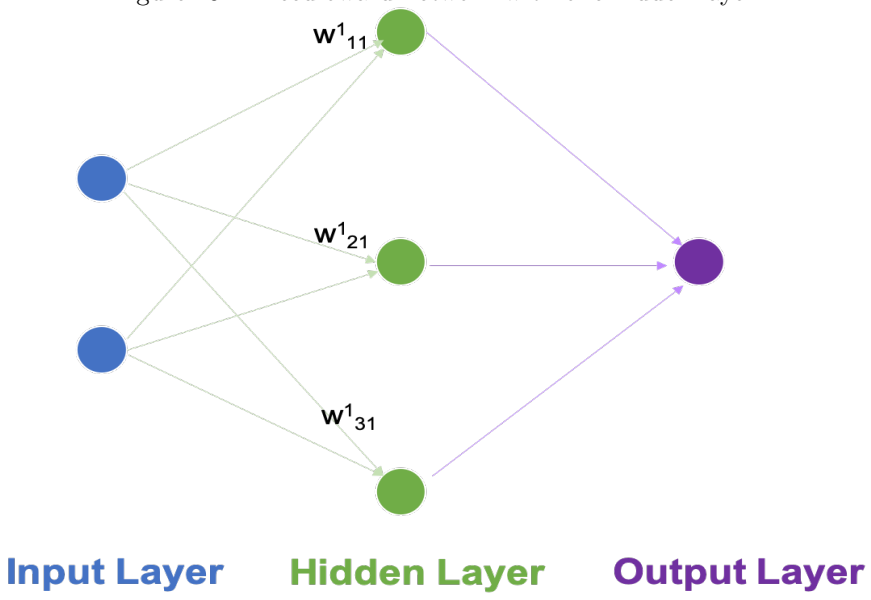
$$v^*(k,z) = \frac{1}{1-\beta}\{log(1-\alpha\beta) + \frac{\alpha\beta}{1-\alpha\beta}log\alpha\beta + \frac{\beta\mu}{(1-\alpha\beta)(1-\beta\rho)}\} + \frac{\alpha}{1-\alpha\beta}logk + \frac{1}{(1-\alpha\beta)(1-\beta\rho)}logz \quad (1.33)$$

Policy function is

$$k_{t+1} = \alpha\beta z_t k_t^{\alpha} \quad (1.34)$$

A.2 How do ANNs learn?

Figure 19: A feedfoward network with one hidden layer



Source: author's own construction

Figure 19 shows a feedforward network with one hidden layer. All the circles are neurons (or nodes) of this ANN. The first column is the input layer, and it has two nodes represented by blue. The middle column is the hidden layer, and it has three nodes in green. The last column is the output layer with one node in purple. The arrows represent directions of information/data flow. Feedforward means that the data flows forward from input to output layers. ws are weights, which needs to be learnt while training an ANN. The superscripts on ws represent the layer that the

weights are assigned to. For example, w_{21}^1 represents the weight of the first neuron in the input layer (the first layer) to the second neuron in the hidden layer (second layer).

I use the first node in the hidden layer as an example to show the information flow within a node. Assume that the input layer nodes output x_1 and x_2 respectively. The first node in the hidden layer then takes information from the previous layer as $w_{11}^1 x_1 + w_{21}^1 x_2$, and apply an overall bias. The output of this node becomes $\sigma(w_{11}^1 x_1 + w_{21}^1 x_2 + bias)$, where $\sigma()$ represents an activation function, and it can take many forms (e.g., sigmoid, logistics, and tanh functions).

How do ANNs learn and update their parameters? It takes the following procedures. Given some training dataset and an ANN model, pass the data forward through the neural network to obtain an output/prediction. Compare this output to a target value/goal. Calculate the loss between the predicted value and the target value. To make sure the neural network learns and update its parameters (including weights of each neuron and biases), a way to link each weight and the loss is needed. *Back propagation* is an algorithm to find such links. In particular, It finds partial derivatives of the loss function with respect to each weight and bias by applying chain rules.

These partial derivatives are called gradients in the literature. Given these gradients, parameters can be updated through gradient methods, such as *gradient descent*. In gradient descent, the update process looks as follows

$$w'_{ij} = w_{ij} - \eta \frac{\partial Error}{\partial w_{ij}}, \tag{1.35}$$

where w_{ij} represents weight of the j^{th} neuron to the i^{th} neuron in the next layer. $\frac{\partial Error}{\partial w_{ij}}$ is the partial derivative of the loss function with respect to the weight, and it states how much error the weight w_{ij} contributed. η denotes learning rate, a hyperparameter. It represents how quickly weights are updated in response to the error contribution term.

The objective of gradient descent is to minimise a loss function. At the point where the loss is minimised, the gradient is 0. Thus, gradient descent requires weights adjustment so as to move along on the line and go to the minimum point, which is similar to what figure 2 illustrates.