Learning from zero: how to make consumption-saving decisions in a stochastic environment with an AI algorithm [*]

Rui (Aruhan) Shi [†]

22 August 2022

Abstract

I model the learning behaviours of a representative household in a stochastic optimal growth environment. I model explicitly this learning agent's decision-making strategy, their subjective belief, and their memory (and how the memory is used to update beliefs). This agent makes exploratory consumption-saving decisions facing income shocks. I impose a temporary shock and a permanent change to the agent's income process. Their behaviours facing changes are adaptive and follow the qualitative predictions of existing economic theories. Depends on how exploratory the agent is, their quantitative behaviours are different and lead to heterogeneous welfare. I also compare behaviours between AI agents and a rational expectations agent.

JEL Codes: C45, D83, D84, E21, E70

Keywords: decision-making, learning from experience, expectation formation, exploration, deep reinforcement learning, bounded rationality, stochastic optimal growth

[*]A previous version of this paper was circulated as CESifo working paper No. 9255. I am grateful for the advice and constant support of Prof. Roger Farmer and Prof. Herakles Polemarchakis. I appreciate the financial support from Warwick University and the many conversations with colleagues at the Rebuilding Macroeconomics project. I am grateful for the helpful discussions and guidance from colleagues at the Bank of England, which also inspired the choice of the DDPG algorithm I used in this exercise. I thank participants at the Society for Computational Economics 27th International Conference, CES 2021 annual conference and the 2021 CESifo Area Conference on Macro, Money, and International Finance. All remaining errors are mine.

[†]PhD candidate at the Department of Economics, University of Warwick. Email: a.shi@warwick.ac.uk.

1 Introduction

How do economic agents form expectations? Empirical work, such as expectation surveys, points to the importance of a realistic and sophisticated framework to model belief formation[1]. I propose a model for a bounded rational agent in a stochastic optimal growth environment. I take a sophisticated learning set up from the literature on artificial intelligence (AI) and use it to build on the economic literature of adaptive expectations.

An agent with adaptive expectations[2] is assumed to be as smart as an econometrician, and updates their belief by running a regression. In this, they are distinct from the rational expectations hypothesis[3] (Muth, 1961; Lucas, 1972; Sargent, 1972). The latest AI advancements show how an artificial agent learns to complete a (real-life) task. The central algorithms, deep reinforcement learning, combine learning through trail-and-error and optimal control algorithms. The artificial agent builds intelligent entities to mimic human intelligence.[4]

I model the following components of a belief formation process:

- an interactive process for the AI agent to collect experience
- a memory to store past experience
- a replay process to recall past experience and sample from the memory
- a flexible function approximation for the decision-making strategy and the value function that reflects the evolving beliefs that are updated using the memory sample.

This approach to modelling[5] is empirically motivated. It allows for the method to be tailored to different sampling strategy, memory size, and different economic environments. It is flexible enough to have multiple agents with different stages of learning.

I start the learning process at the beginning, so the AI agent does not know the underlying economic structure, and is not aware of its own preference. The agent has to gather experience by interacting

[1]See for example Malmendier and Nagel (2016), D'Acunto et al. (2021) and Malmendier (2021).

[2]See, for example, Bray (1982), Marcet and Sargent (1989), Sargent (1993) and Evans and Honkapohja (2001).

[3]A utility-maximising agent is assumed to know the underlying economic structure and form model-consistent beliefs.

[4]Not only are reinforcement learning algorithms being widely applied in computer science and AI research, they are also connected to neural scientific research. Botvinick et al. (2020) argue that reinforcement learning has the potential to explain neural mechanisms of learning and decision-making. Perhaps the most impactful research to date is the empirical work that establishes the link between phasic dopamine release and a reinforcement learning algorithm reward-prediction error signal (Niv, 2009). The reward-prediction error signal, i.e., temporal different error, is introduced in the methodology section.

[5]Details are in the methodology section.

with the environment. It does not know the consequence of an action in a state, and it learns by trying different alternatives.

Some agents are adventurous and want to try new things whereas other agents prefer options that they have already tested before. The exploration mechanism generates these different learning behaviours in this method. It is intuitively appealing[6] and more importantly, allows for adaptability in a changing environment.

I present two simulation experiments and results. The first shows the learning behaviours of AI agents. With uncalibrated parameters, I show that the agents' decisions converge towards the optimal solution through learning simulations. Optimal means the rational expectations solution in the same economic model. However, the agent's decision does not converge fully to the optimal solution due to the exploration feature.

The second shows how the agent adapts to a changing environment. I impose a change in the underlying stochastic process of the model. I show the adaptive behaviours of AI agents, in particular, the importance of the exploration feature in generating different decision sequences with welfare implications.

The next section is a literature review. In Section 3 I introduce AI technologies. In Section 4 I discuss the economic model and the design of the AI agent. In Section 5 I present simulation experiments and results, and Section 6 is the conclusion.

2 Literature

Herbert Simon defines bounded rationality and introduces his approach to adopting AI in decision making.[7] He focuses more on the decision-making process than on its outcome. AI that he suggests is on the heuristic search and problem solving by recognition (Simon, 2016).

Sargent (1993) describes how he combines AI and macroeconomic modelling. He aims to find symmetry between econometricians and economic agents. The agent has a learning ability based

[6]People are different in terms of how exploratory they are with their environment. Some people prefer to try unknown options.

[7]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).

on a decision rule, and at the limit the agent converges to a rational expectations equilibrium. Agents behave like professional scientists or econometricians, and use methods of scientific inference in collecting information and forming their expectations. Sargent argues that this literature is important because it looks at the transition behaviours and dynamics in a learning process. The combination of AI and economics executed here builds on both Sargent's and Simon's views.

The literature on applications of deep reinforcement learning algorithms in economics is growing quickly. Charpentier et al. (2020) provide some economic frameworks that could be applied with deep reinforcement learning techniques. The possibilities range from economic modelling to applications in operations research and game theory. They advocate for reviewing economic and financial problems using deep reinforcement learning techniques.

Chen et al. (2021) adopt a deep reinforcement learning algorithm to solve a monetary model. Hill et al. (2021) use a deep reinforcement learning algorithm to solve heterogeneous agent models. They believe the method should be in the central banks' toolboxes.

Hinterlang and Tänzer (2021) apply a deep reinforcement learning method to find the optimal monetary policy reaction function with a dual mandate. Kuriksha (2021) modifies a deep reinforcement learning algorithm to make it applicable and easier to implement for a large number of households. He studies households' consumption-saving decisions as well. Curry et al. (2022) apply a deep reinforcement learning algorithm in a Markov game setup and study whether an equilibrium can be reached.

In this paper, in addition to introduce the mechanism of modelling bounded rationality through the AI technology, I study the role of exploration. I focus on how exploration mechanisms affect learning agents' beliefs and welfare in a changing economy. This has not been studied explicitly in the economics literature, and it is an important mechanism in modelling agents' behaviours during structural changes, and in multi-agent learning systems in future studies.

3 AI Technologies and Reinforcement Learning

Deep reinforcement learning (RL) is built on reinforcement learning, and uses artificial neural networks (ANNs) for function approximations. This has the advantage of learning in environments with high-dimensional state and action spaces[8]. Notable developments include teaching AI agents (using DRL algorithms) to play Go and Atari games, and to learn speech recognition.

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

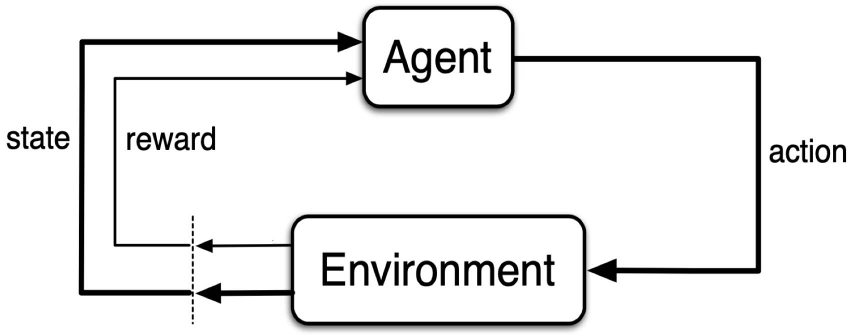
3.1 Reinforcement Learning: a primer

As a branch of machine learning, reinforcement learning (RL) is different from both supervised and unsupervised machine learning methods. It does not require exemplary supervision, or unlabelled data for classification. Training data in RL is generated by an agent directly interacting with its environment.

RL combines two ideas. The first is the psychology of animal learning through trail and error. The second is the solution method for optimal control problems, such as dynamic programming (Sutton and Barto, 2018).

RL algorithms solve Markov Decision Processes (MDPs). An MDP, as described by figure 1, is 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.

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



Source: Sutton and Barto (2018)

At each time step t, an RL agent receives some representation of the environment's state (a random variable) 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, partly as a consequence of its action, the agent receives a numerical reward based on a reward function, $r_t = r(s_t, a_t)$, 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

$$p(s_{t+1}|s_t, a_t) \equiv Pr\{s_{t+1}|s_t, a_t\} \tag{3.1}$$

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.

The RL agent's behaviours are defined by a policy, which can be either stochastic or deterministic. If an agent follows 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 a deterministic action for a realised state s_t.

Expected returns are defined 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). It is denoted as $Q^\mu(s_t, a_t)$, which shows the expected return after taking an action a_t in state s_t and thereafter following policy μ[9].

$$Q^\mu(s_t, a_t) \equiv \tilde{E}[R_t|s_t, a_t] \tag{3.2}$$

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. $\tilde{E}$ denotes the subjective expectation of this agent. Equation 3.2 also follows the Bellman recursive relationship,

$$Q^\mu(s_t, a_t) = r(s_t, a_t) + \beta \tilde{E} Q^\mu(s_{t+1}, a_{t+1}), \tag{3.3}$$

where $a_{t+1} = \mu(s_{t+1})$.

An optimal action-value function, defined as

$$Q^*(s_t, a_t) \equiv \max_\mu Q^\mu(s_t, a_t), \tag{3.4}$$

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.

[9]It can also be defined in terms of a stochastic policy π. Given that the algorithm adopted later follows a deterministic policy, here only the value function in terms of a deterministic policy μ is introduced.

At the outset, a RL agent does not know anything about the environment. This means it does not know how a reward r_t is generated. The agent gets to know its own preference through trying out different options and observing the rewards. The agent also does not know the true probabilities, i.e., $p(s_{t+1}|s_t, a_t)$ in its environment. It forms a subjective belief based on its past experience, and uses this belief to guide future decisions.

A RL agent's task is to make decisions, i.e., a policy π or μ, that maximises its expected returns. This expectation is a subjective belief of this agent, and need not be based on the true probabilities of the underlying processes.

3.1.1 Exploration vs Exploitation

One of the challenges in solving a RL problem 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 to obtain a reward, but it also has to explore to make better selections in the future.

The agent will fail at the task if they pursue either exploration nor exploitation exclusively. They must try a variety of actions and progressively favour those that represent an improvement. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward.[10] In reality, we learn about what we like or dislike through trying out different options, which is similar to how learning is modelled in this paper.

3.1.2 Deep Reinforcement Learning Algorithms

Deep RL algorithms refer to the use of deep artificial neural networks (ANNs) for function approximation.[11]. There are broadly two types of algorithms, action-value function based algorithms, and policy gradient based algorithms.

The Deep Q network algorithm is an example of the action-value function based approach, and is first introduced by Mnih et al. (2013). It is capable of human-level performance on many Atari

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

[11]Deep RL 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 the 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, Deep RL 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).

video games using pixels for input. However, while the deep Q network algorithm solves problems with high-dimensional state 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.

The deep deterministic policy gradient (DDPG) algorithm, introduced by Lillicrap et al. (2015), is an example of a policy-gradient based approach. The advantage is that both the policy function and the action value function are explicitly learnt. It is also capable of learning in environments with continuous action space.

4 Methodology

In this section, I first show how an economist would usually model the environment. I then show how it is adapted to the setting of an AI learning agent.

4.1 The Model: an economist's approach

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

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{4.5}$$

subject to

$$c_t + k_{t+1} = z_t y_t \tag{4.6}$$

$$y_t = k_t^{\alpha} \tag{4.7}$$

$$c_t \geq 0 \tag{4.8}$$

$$k_{t+1} \geq 0 \tag{4.9}$$

for all t.

c_t, k_t, y_t are consumption, capital investment, and output produced in period t. In this exercise, I use capital investment and saving interchangeably. Period utility $u(\cdot)$ 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

$$z_t = e^{\mu + \rho ln(z_{t-1}) + \epsilon_t} \tag{4.10}$$

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,[12] it is a natural starting point for showing 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 expectations counterpart, which is a product of the analytical solution.

4.1.1 Optimisation under Rational Expectations

The Bellman equation for this problem is:

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta E_t v(k_{t+1}, z_{t+1})\} \tag{4.11}$$

The solution[13] is:

$$k_{t+1} = \alpha \beta z_t k_t^{\alpha} \tag{4.12}$$

The value function following this policy is

[12]To name a couple, the real business cycle model and the incomplete market model.

[13]See appendix for detailed derivation.

$$
\begin{aligned}
v^*(k,z) = \frac{1}{1-\beta}\left[log(1-\alpha\beta) + \frac{\alpha\beta}{1-\alpha\beta}log\alpha\beta + \frac{\beta\mu}{(1-\alpha\beta)(1-\beta\rho)}\right] \\
+ \frac{\alpha}{1-\alpha\beta}logk + \frac{1}{(1-\alpha\beta)(1-\beta\rho)}logz
\end{aligned}
\tag{4.13}
$$

4.2 The Model: an AI approach

I apply and modify the DDPG algorithm for the current economic setting. Chen et al. (2021) also adopts this algorithm in their study of learnability of the rational expectations equilibrium in different policy regimes.

State, action and reward

I define bounded and compact state space and action space. The state space represents the environment, and the action space is the choice set of the learning agent.

Table 1 presents the components of RL and how they are represented in the economic environment. The state is represented by the total resource available each period. The RL agent chooses how much it wishes to consume out of the total available resource (second row of table 1). Once an action is made, the agent receives a reward. The reward is determined according to the state and action of a particular period, and it acts as a stimulus signal to show if the agent likes or dislikes a particular choice of action in a given state. The next state is a combination of whatever is saved plus the new stochastic income.

If the agent chooses to consume all in the previous period, it risks zero consumption this period (if it receives no income this period). This leads to a minimum reward to the agent (because it would not be happy to be hungry).

Table 1: RL components and the economic environment

Terminologies	**Description**	**Representation in the economic environment**
State, s_t	A random variable from a state space, $s_t \in \mathcal{S}$	total goods available to consume at period t, $z_t k_t^\alpha$
Actions, a_t	A random variable from an action space, $a_t \in \mathcal{A}$	proportion of the total goods that the agent is willing to consume at period t
Rewards, r_t	A function of state and action	utility at period t, $ln(c_t)$ where $c_t = a_t z_t k_t^\alpha$
Next State, s_{t+1}	A random variable from a state space	total goods available to consume at period $t+1$, where $k_{t+1} = (1 - a_t) z_t k_t^\alpha$
Policy function, $\mu(s\|\theta^\mu)$	A mapping from state to action, $\mu : \mathcal{S} \to \mathcal{A}$	Approximated by a neural network, ie., actor network; parameterised by θ^μ to be updated during learning
Value function, $Q(s, a\|\theta^Q)$	the ‘expected’ (subjective belief) return of taking an action in a state	Approximated by a neural network, ie., critic network; parameterised by θ^Q to be updated during learning

Policy function

The classification of the agent’s decision-making centre (or brain) involves two parameterised artificial neural networks (the last two rows of table 1).

The RL agent's decision is approximated by a parameterised ANN, i.e., $\mu(s|\theta^{\mu})$. It maps a realised state to an action given parameter θ^{μ}.

Given that an AI agent, at the beginning of a learning process, knows nothing about what action constitutes a high reward and lifetime utilities[14], 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 explores its action space, an exploration policy μ' is constructed by adding a noise process $\mathcal{N}$ to the actor policy.

$$\mu'(s_t) = \mu(s_t|\theta^{\mu}) + \sigma_t \mathcal{N}_t. \tag{4.14}$$

$\mathcal{N}_t$ is sampled from a discretised Ornstein-Uhlenbeck process.[15]

σ_t represents the scale of noise $\mathcal{N}_t$, and it represents how exploratory an agent is. It can take any value between 0 and 1. It can also vary through time. Early in the learning process the exploration is likely to be high, since the agent needs to try out more options to gather information. With time the agent learns what action could bring high reward, and may reduce their level of exploration.

In this exercise, $\sigma_t > 0$ for all t, and this means that the agent always explores. In a non-stationary environment, that is, an environment that constantly changes, the agent can observe when a state-action pair no longer attracts a high reward and adapt accordingly.

Value function

A good policy function is determined by the expected return of following a policy. The expected return is determined by the value function. It is approximated by another ANN, i.e.,$Q^{\mu}(s, a|\theta^{Q})$. Assume there are two policies, μ_1 and μ_2, μ_1 is more desirable if $Q^{\mu_1}(s, a|\theta^{Q}) > Q^{\mu_2}(s, a|\theta^{Q})$. That is to say, given the same realised state and action pair, the better policy produces a higher expected return, which is measured by the value function Q. θ^{Q} contains the subjective belief of this agent, and it evolves over time as it learns more in the environment.

Memory

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

[15]This noise could be sampled from an uncorrelated Gaussian process or a correlated Ornstein-Uhlenbeck process. It is likely that a learning agent explores its environment in a correlated manner, and thus an Ornstein-Uhlenbeck process is followed.

The interactive experience is saved in the agent's memory. Taking period t as an example, the experience includes (s_t, a_t, r_t, s_{t+1}). The agent in this paper is assumed to never forget.

The RL agent takes a random sample from the memory, which is used to update the policy and the value functions. Other sampling methods can be incorporated, for example, sampling from the most recent experience, or the most impactful experience.

4.2.1 Full Algorithm and Sequence of Events

The full algorithm[16] and the sequence of events follow three main steps:

Step I: Initialisation

- Set up an environment. This includes a bounded and compact state space; a bounded and compact action space; a reward function that takes the argument of two random variables (state and action) and generates a reward, and state transition dynamics, i.e., an evolution of how the current state and action lead to the next state.
- Set up two neural networks: an actor network $\mu(s|\theta^\mu)$ takes the argument of the state and generates an action; a critic network $Q(s, a|\theta^Q)$ takes the argument of a state-action pair and generates an expected return.
- θ^μ and θ^Q represent the parameters of the two networks. 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 RL 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. A mini-batch refers to a sample from the memory.
- Define the total number of episodes E.

[16]The use of DDPG algorithm comes from discussions with colleagues at the Bank of England.

- Define a simulation period of T for each episode, where $T > N$. The higher the episodes, the longer the learning periods.[17]

For each episode, loop step II and III.

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

- The agent observes a state, i.e., $s_t = z_t k_t^\alpha$, which is the total goods available to consume at period t. 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 (actor network) and an exploration noise.
- Execute action a_t, and observe a reward, which is derived based on the utility function, i.e., $r_t = ln(c_t)$ and the next state realisation is $s_{t+1} = z_{t+1} k_{t+1}^\alpha$.
- Store a transition (s_t, a_t, r_t, s_{t+1}) in memory $\mathcal{B}$.

Step III: Training the AI agent (when 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 memory $\mathcal{B}$.
- Calculate a value y_i for each transition i following

$$y_i = r_i + \beta Q^\mu(s_{i+1}, \mu(s_{i+1}|\theta^\mu)|\theta^Q) \tag{4.15}$$

  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}.
- Obtain $Q(s_i, a_i|\theta^Q)$ from the critic network with input state-action pair (s_i, a_i)
- Calculate the average loss for this sample of N transitions

$$L = \frac{1}{N} \sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2. \tag{4.16}$$

[17]In the DRL literature, an AL agent is usually set to learn a particular task or an Atari game. An episode means re-starting the game or task, and it ends with a terminal state (i.e., the end result of a game). In an economic environment, however, a clear terminal state can be difficult to specify. Therefore, the concept of episodes only correlates to how long an agent has been learning.

- Update the critic network with the objective of minimising the loss function L.[18]
- For the policy function, i.e., the actor network, the objective is to maximise its corresponding value function. This means a value function $Q(s, a|\theta^Q)$ that follows a particular policy. In other words, the input action a of Q function is from the policy μ, $a = \mu(s|\theta^\mu)$. Define the objective function as,

$$J(\theta^\mu) = Q^\mu(s_i, \mu(s_i|\theta^\mu)|\theta^Q). \tag{4.17}$$

- This objective function could also be rephrased as minimising $-J(\theta^\mu)$. Update the actor network parameters with the objective of minimising $-J(\theta^\mu)$.[19]

4.3 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	3.0
Autoregressive factor ρ	0
Discount Factor β	0.99
Learning Rate η	actor network $1e-4$; critic network: $1e-3$
Exploration Level σ_t	0.3

A learning rate parameter is used for an ANN in the process of updating weights[20].

[18]This involves applying back propagation and gradient descent procedures.

[19]Similar to the critic network, the specific steps of updating ANN's parameters by minimising an objective function involve back propagation and gradient descent.

[20]For more information on the purpose of a learning rate parameter, please see Appendix.

Exploration level is measured by the scale of noise σ_t, as specified in equation (4.14). In this case, the exploration level reduces over time from the full scale of value 1, and 0.3, as shown in the last row of table 2, is the minimum level of exploration a learning agent has.

The exploration parameter represents how an AI agent gathers information. Different exploration levels also mean that agents can have different past experiences living in the same environment. An AI agent can be more or less adventurous in exploring available actions and their outcomes.

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 to 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 approach to information collection and processing is consistent with empirical evidence on learning from experience and use-dependent brain by Malmendier and Nagel (2016), D'Acunto et al. (2021) and Malmendier (2021).

5 Experiments and Results

Using several experiments, in this section I highlight three key results: 1. Learning from zero, the AI agents can reach a stage[21], in which their behaviours facing shocks support the permanent income hypothesis argued by Friedman (1957). 2. AI agents are adaptive to changes in the environment in real time, and the results in this section provide plausible transition dynamics. 3. When AI agents are different in terms of how much they explore the environment (i.e., how they collect information), their transition behaviours are different facing environmental changes, which leads to welfare distinctions.

In this section, the changes in an environment are introduced through the stochastic process z_t in the economy. Recall equation (4.10)

[21]It may not be the stage of full rationality. For comparisons between AI learning agents and a rational expectations agent, please see section 6.

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

More specifically, I position three AI agents in the same environment with changes in the stochastic process. The agents are different only in terms of how much they explore their environment, i.e., their exploration levels are different. 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.

5.1 Learning from Zero

To illustrate the learning process of an AI agent, I take the example of a baseline agent (i.e., follows parameter in table 2), and show the difference in its simulated behaviours before and after learning.

Figure 2: Simulated consumption paths during learning

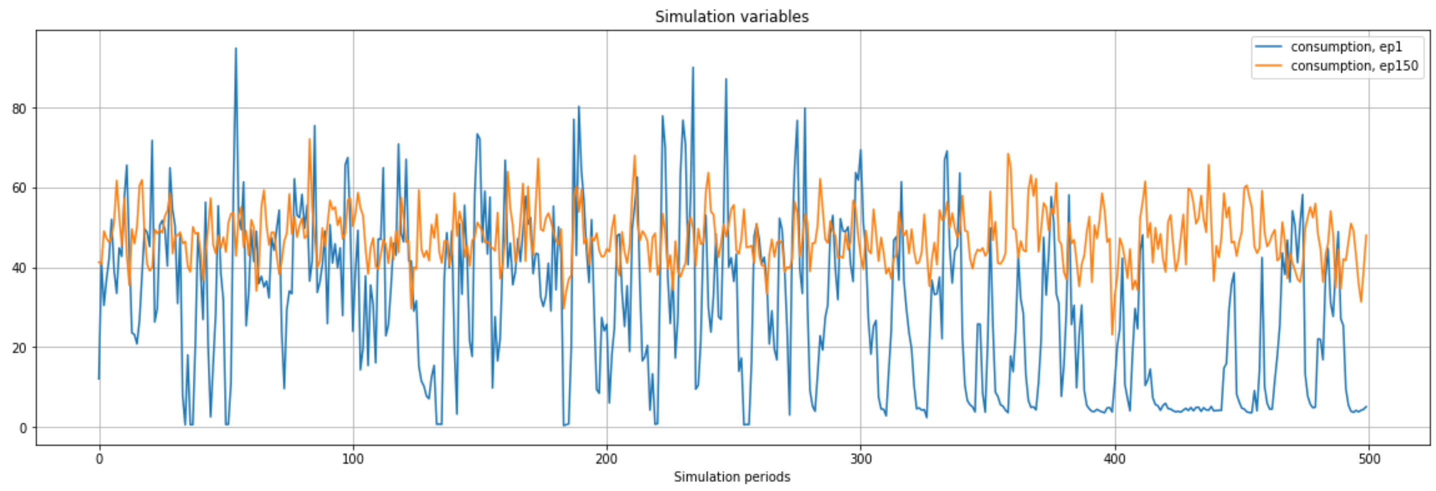


Figure 2 plots this AI agent's consumption paths at the beginning of a learning process (labelled ep1) and towards the end of a learning process (labelled ep150). The x-axis plots simulation periods.

It shows that at the beginning of a learning process, this agent's consumption choices (denoted by the blue line) are more volatile than when it has been learning in the environment after many periods (orange line). This shows that the agent does not know what is desirable in its choice set, and thus takes many random actions, which also corresponds to a high exploration level at the beginning of a learning process. After learning for many periods, its decisions are more centred.

The agent generates and learns from its own experience, and this leads to a long learning period. In this case, it is 150 episode with 5000 iterations for each episode. The agent can reach the state that is denoted by the orange line quicker, through varying algorithm parameters.

5.2 Transitory Shock

Figure 3: The stochastic process with a transitory shock

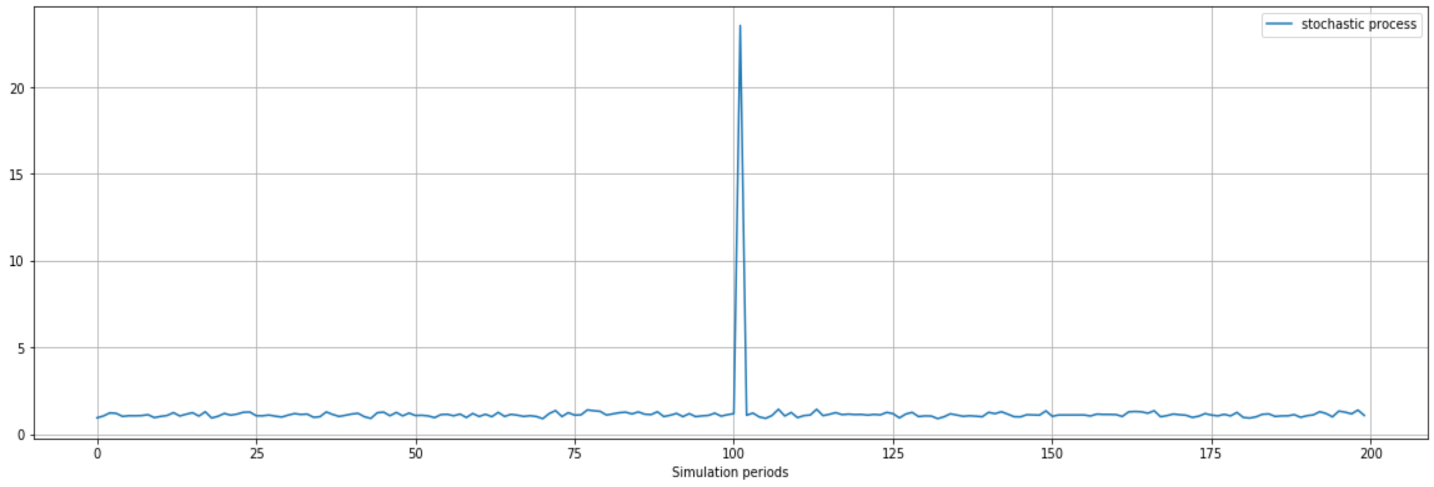


Figure 4: AI agents' consumption paths facing a transitory shock

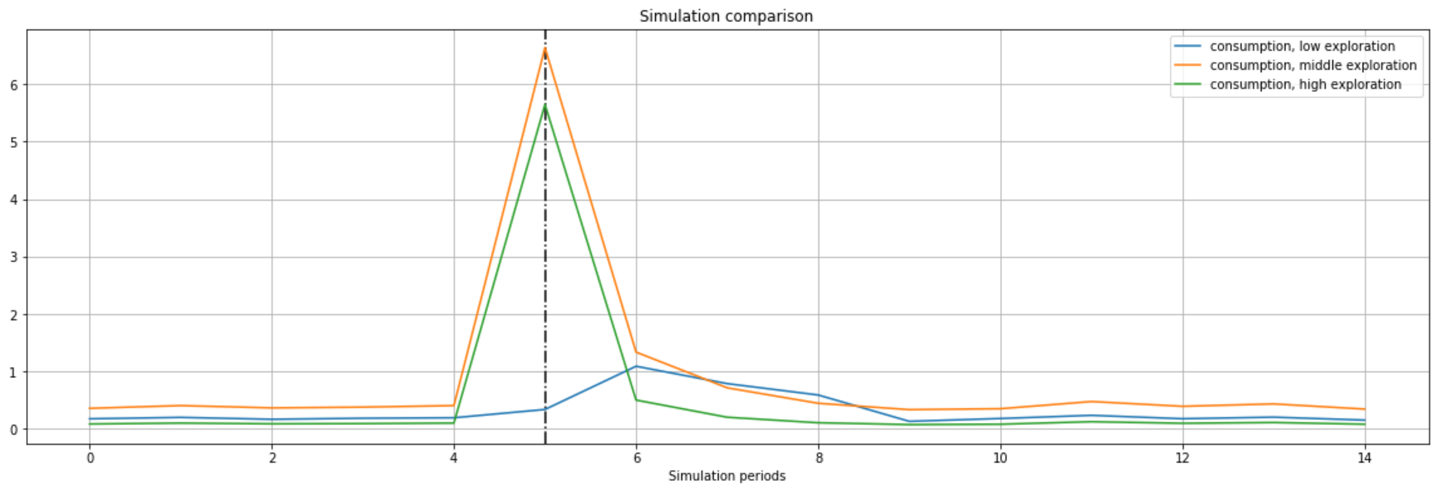


Figure 3 shows the stochastic process in this environment. The process follows $z_t = e^{0.1+\epsilon_t}$ except for simulation period 100 where $z_t = e^{3+\epsilon_t}$. ϵ_t follows a normal distribution with $N(0, 0.1)$. The x-axis represents simulation periods.

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. Before the shock hits, all three agents reach a stage of a smooth consumption path through learning.

When the shock hits, all three agents with different exploration levels exhibit similar overall consumption behaviours, namely consumption increases with the positive shock and reverts 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 is evidence that with higher exploration, an agent is more alert to changes in the environment and is quicker to respond.

The low-exploration agent responds with a lag, as shown by the blue line. The magnitude of their response is also correlated to their exploration levels. A low-exploration agent responds in a slower manner and with less magnitude than the other two agents.

Figure 5: AI agents' utilities facing a transitory shock

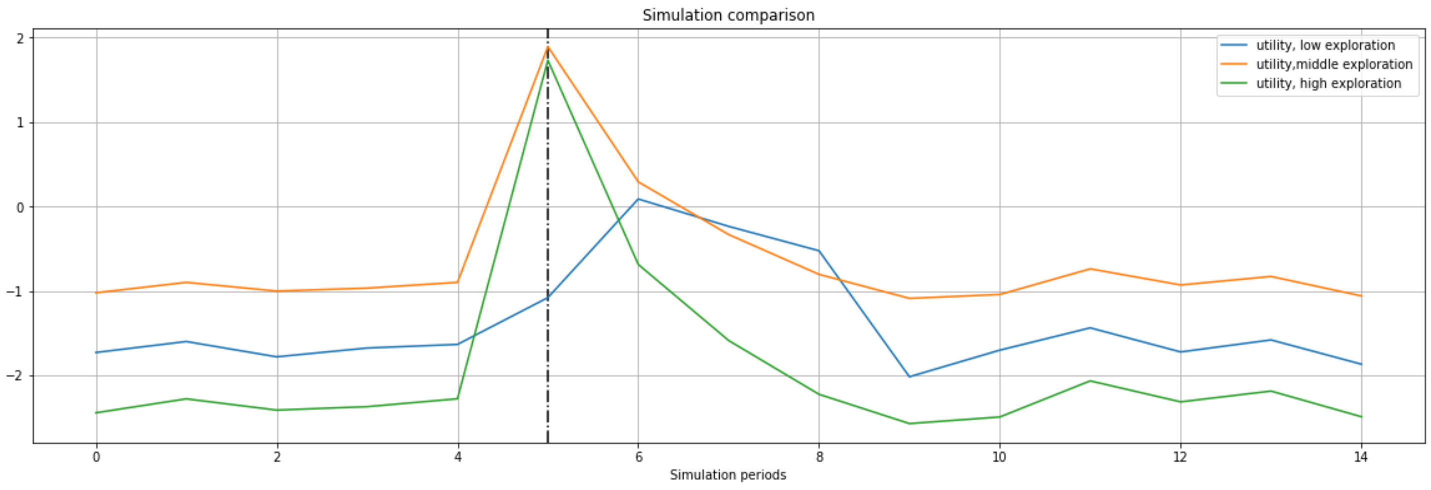


Figure 5 is a plot of all three agents' utility as a measurement of their welfare. The high-exploration agent (green line) does better than the low-exploration agent throughout the transitory shock.

However, this does not hold for periods before and after the transitory shock. Their overly adventurous nature leads to high excess investment (very low consumption) and thus a lower utility level than the middle agent.

The middle agent (orange line in figure 5) has the highest utility, and balances exploration and exploitation of existing knowledge. If the agent reduces their level of exploration, as shown by the blue line in figure 5, they sacrifice their welfare in favour of cautious behaviour and only try actions that they have tested before.

This transitory shock can be interpreted as a positive productivity shock. With a positive productivity shock there is a temporary increase in consumption. 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?

5.3 Permanent Shock

Figure 6: The stochastic process with a permanent change

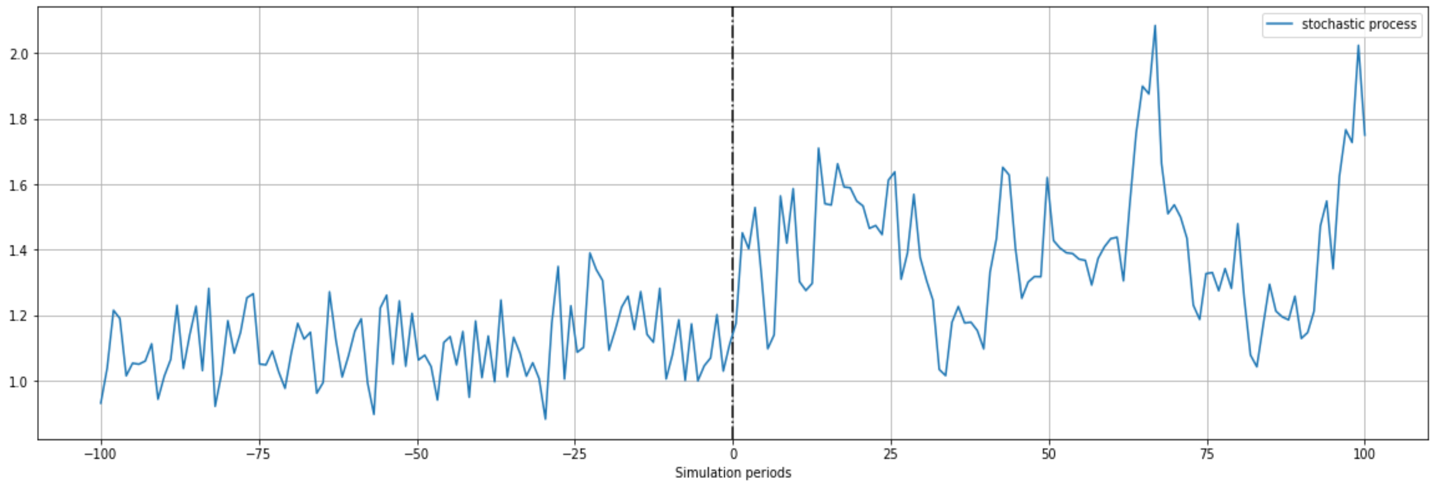


Figure 7: Learning agent in an environment with a permanent change

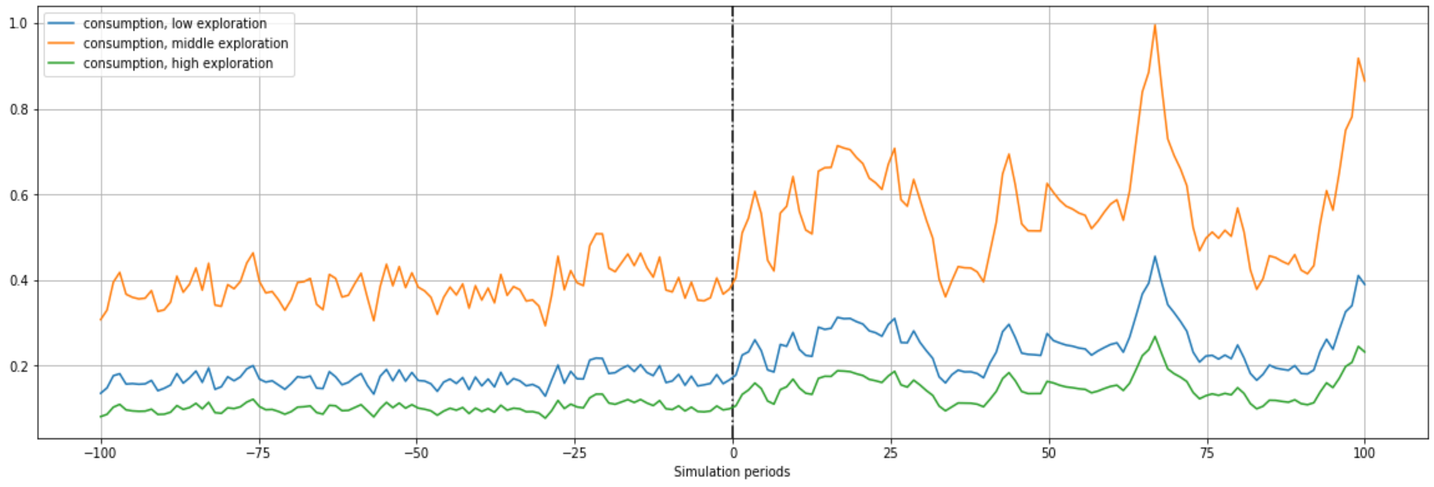


Figure 6 shows the stochastic process changes from $z_t = e^{0.1+\epsilon_t}$ to $z_t = e^{0.1+0.7lnz_{t-1}+\epsilon_t}$. The black dash line indicates the period when the change happens.

Given the permanent change of the 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 is consistent with Milton Friedman's permanent income hypothesis, and a permanent income change (rather than a temporary one) drives the change in a consumer's consumption smoothing behaviour (Friedman, 1957).

Figure 8: Low exploration learning agent in an environment with a permanent change

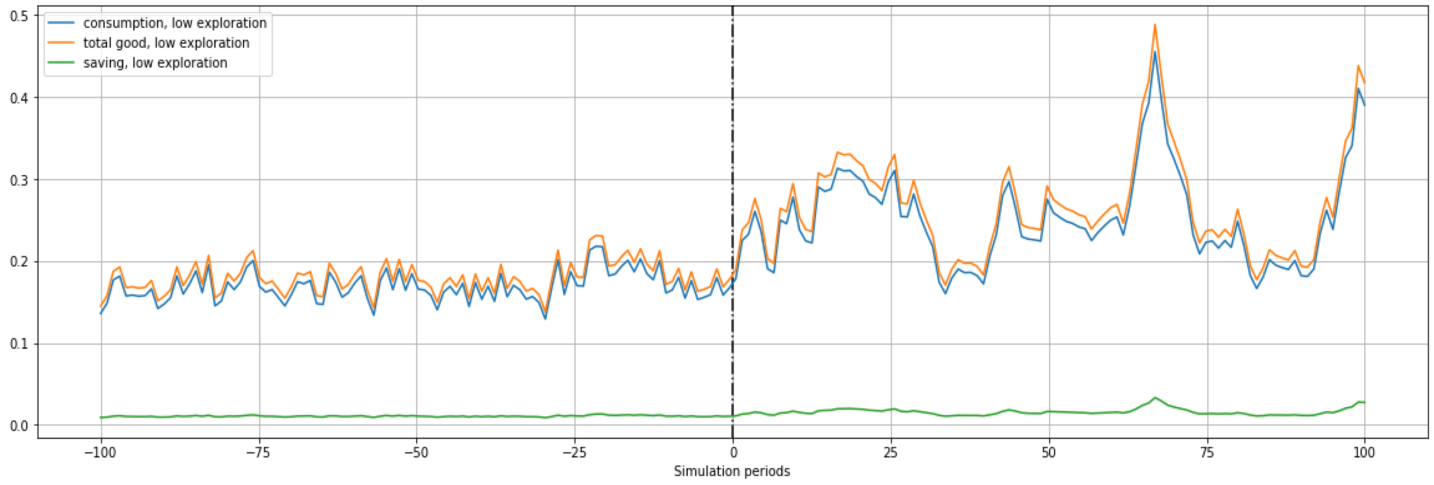


Figure 9: Middle exploration learning agent in an environment with a permanent change

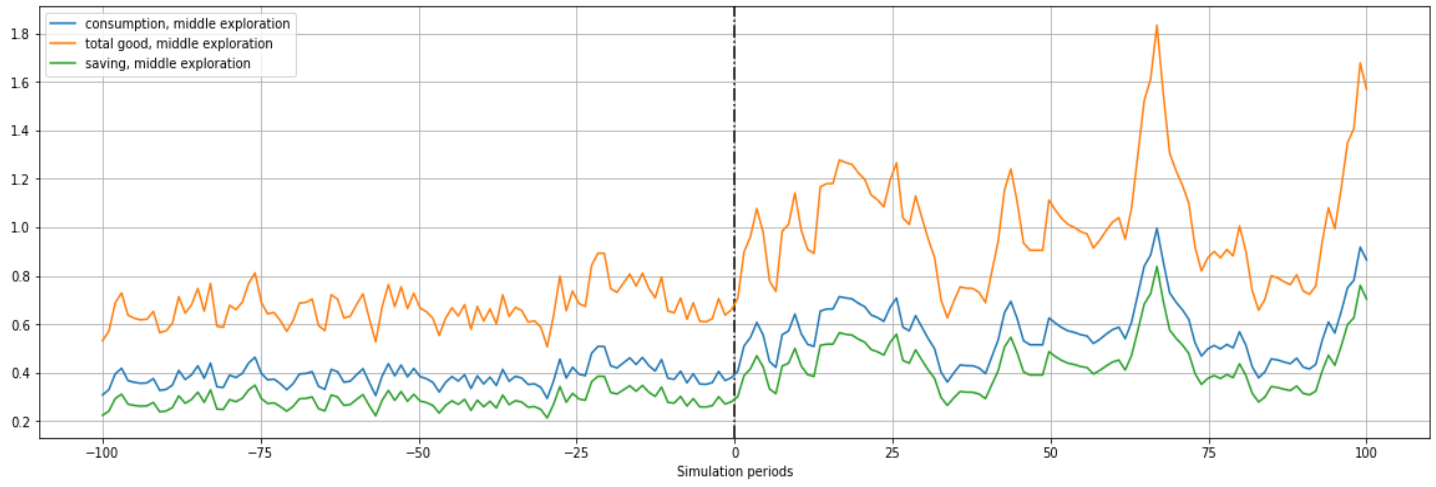


Figure 10: High exploration learning agent in an environment with a permanent change

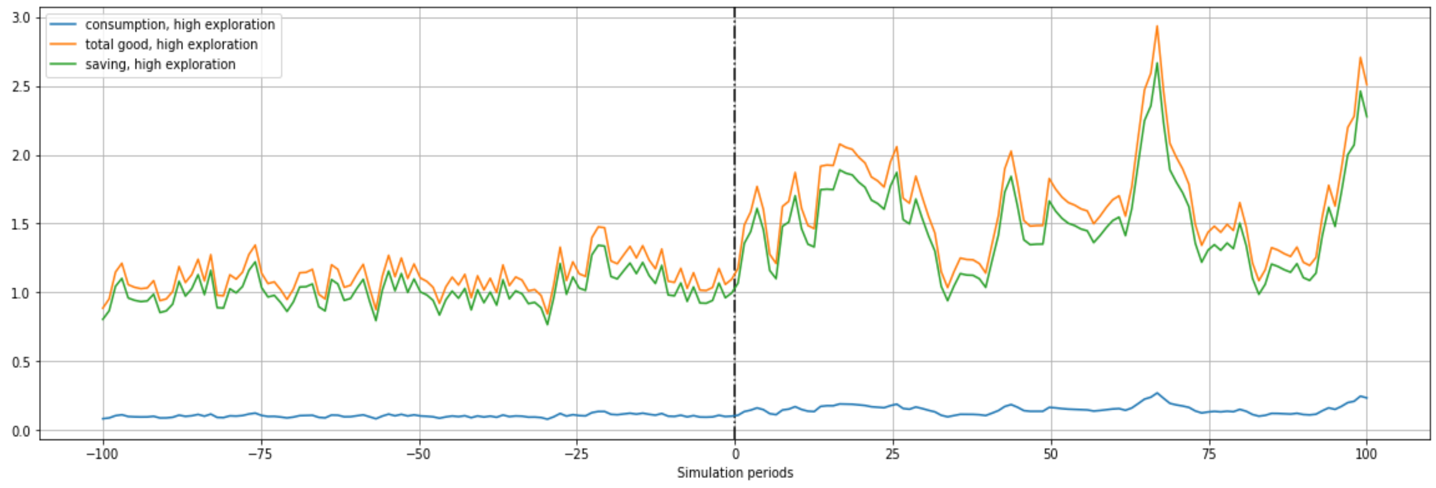


Figure 8 to 10 plot simulated series of consumption, investment, and total available resource for three types of AI agent with different exploration levels. With different exploration levels, the agent's saving behaviours are very different. The agents are living in the same environment with the same underlying stochastic shocks and the identical permanent shift in the shock process.

Figure 8 to 10 show that all three agents exhibit permanent shift of behaviours after the change in the underlying stochastic process. In addition to this adaptability result, I make other observations from this experiment.

Before the shock hits (i.e., to the left of the vertical black line), three agents stabilise around different levels of total resource series. This is mainly due to their different saving behaviours generated by their different exploration level (given that they face the same stochastic shocks).

The low exploration agent stabilises around a level of total output of 0.18 as shown in figure 8. This is 0.6 for the middle exploration agent in figure 9, and 1.0 for high exploration agent in figure 10.

The proportion of total good available the agent saves also differ. Figure 8 shows that the low exploration agent consumes almost all of their available resource each period. This is reversed in figure 10. The high-exploration agent invests nearly all their available resource each period, which partially explains their highest level of output among all the agents. The middle-exploration agent takes the middle ground, as shown in figure 9. Further investigations required to learn the reason behind this differences. One possible reason is that the high exploration agent needs to maintain a high level of total available resource to sustain its highly explorative nature of trying different actions, i.e., different consumption-saving levels. This result can be further linked to empirically observed heterogeneous saving behaviours in relation to wealth differences, which is left for future research.

The different welfare outcomes from these behaviours are shown in figure 11.

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

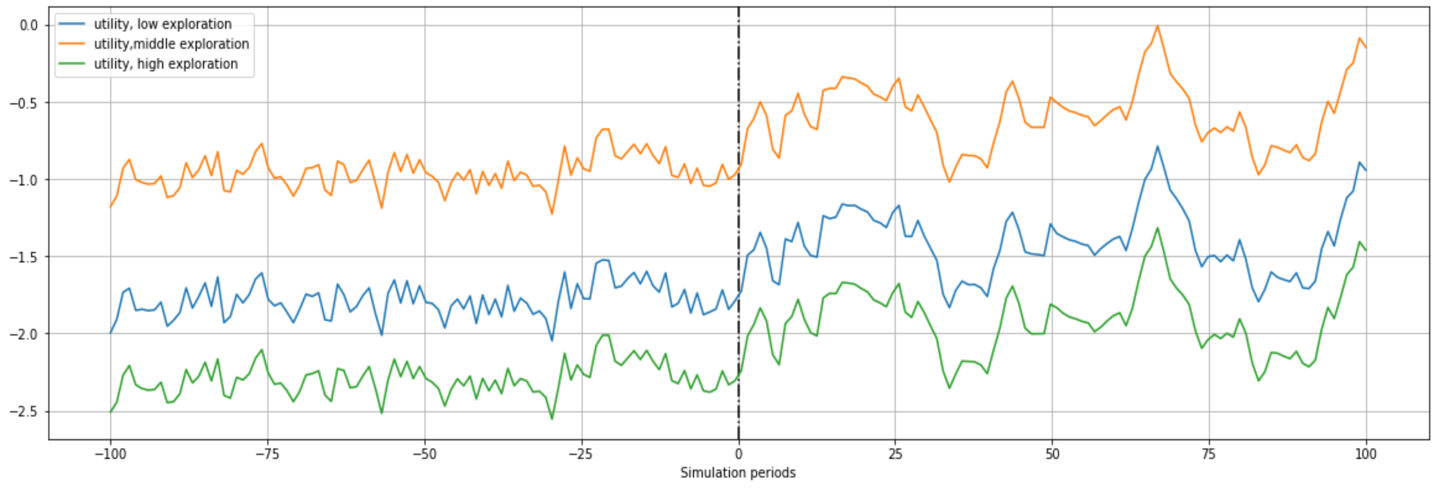


Figure 11 shows all three agents' utility in an environment with a permanent change in the stochastic process. As anticipated, the middle-exploration agent balances exploring the unknown action space and exploiting the known domain to achieve the highest utility. However, the agents with low and high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

6 Comparisons with One Agent Under the Rational Expectations Hypothesis

AI agents' belief formation processes are different from an economic agent under a rational expectations assumption or an econometric learning agent. In this section, I compare the AI agent's behaviour with that of a rational expectations agent (RE agent). I first compare the learned/approximated policy function with the analytical solution of the given economic model. I then compare the simulated path of the AI agent and the RE agent, and show how their consumption decisions differ.

Figure 12: Approximated policy functions

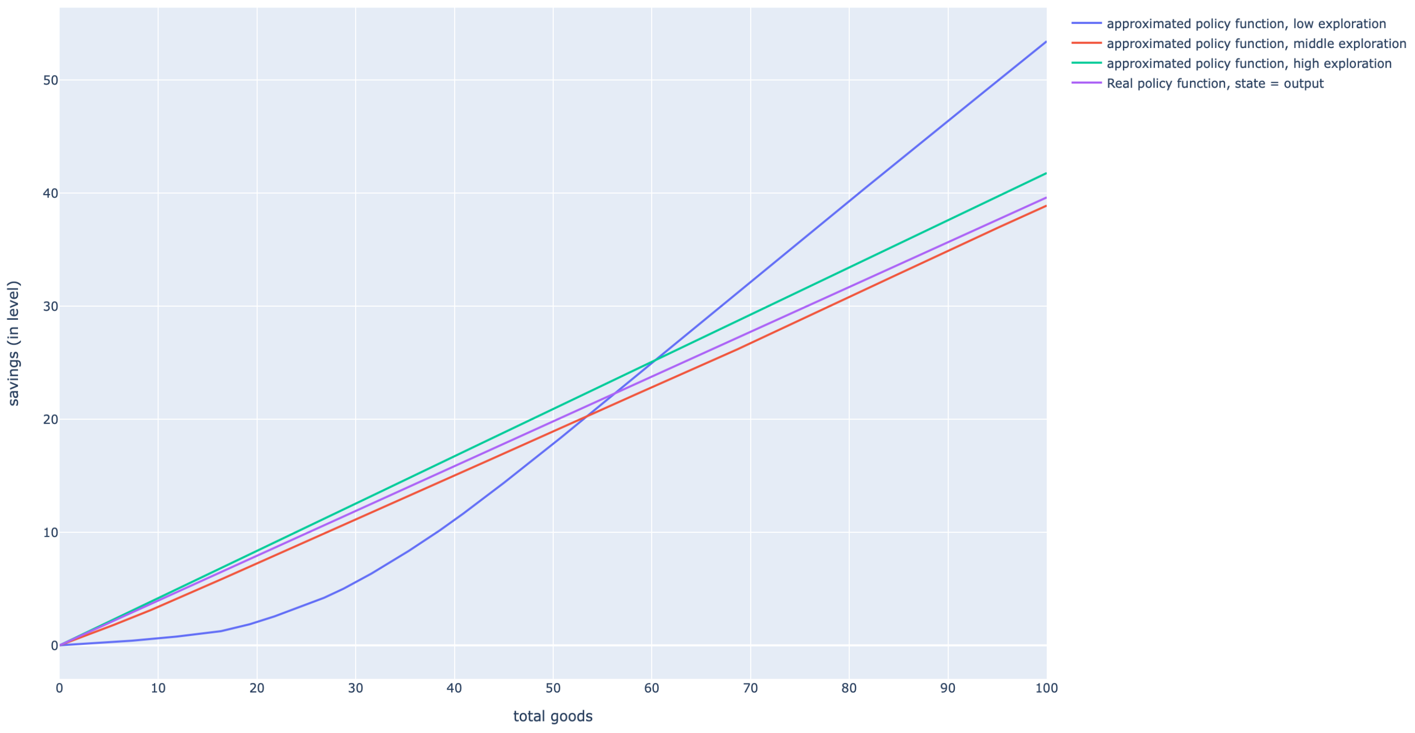


Figure 13: Distance metrics

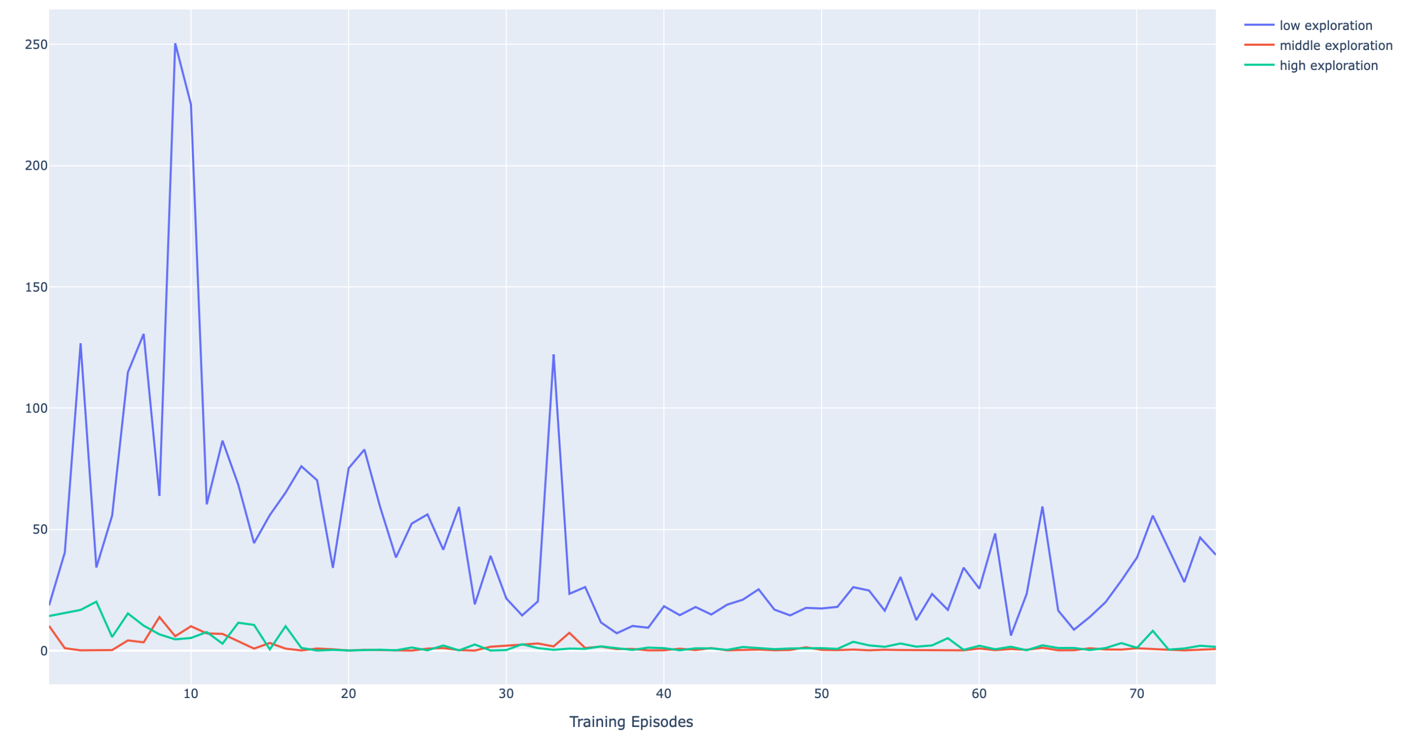


Figure 12 plots the approximated policy functions for three AI agents (with different exploration levels) and the analytical solution policy of this stochastic optimal growth problem (i.e., equation 4.12). All three AI agents have been learning for the same number of periods. The figure shows that the AI agents' learned 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 different if the agent is overly cautious or adventurous in choosing their actions (blue and green lines).

Figure 13 plots a distance metric that is calculated between the analytical solution policy and the approximated policies at each episode as illustrated by equation (6.18). 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 the total number of grids. d_e denotes the distance between the analytical solution and the approximated policy function at episode number e.

$$d_e = \frac{1}{G}\sum_{g=1}^{G}(k_g^* - k_g)^2 \tag{6.18}$$

In figure 13, 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 it does for the high and low exploration agents.

Figure 14: AI agent vs RE agent consumption paths

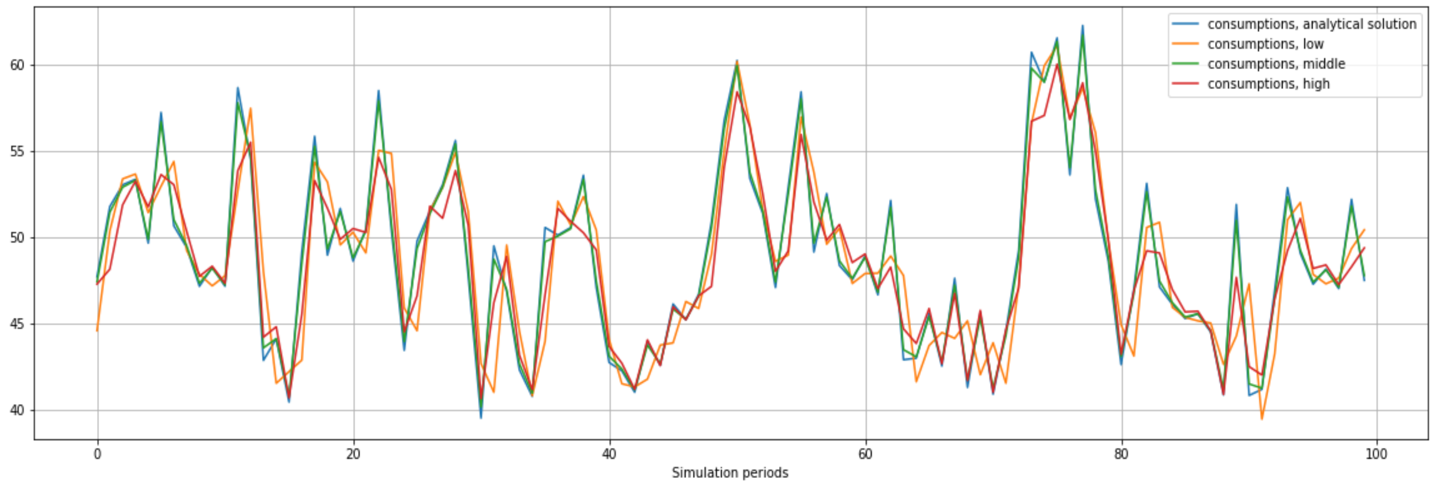


Figure 14 plots simulation comparisons among three AI agents and the RE agent in the same stochastic environment. All agents follow the same initial condition and behave following their respective policies in subsequent periods. It shows that the middle-exploration agent is doing almost as well as the RE agent in terms of the consumption level, which determines agents' welfare. More importantly, it illustrates that through the learning structure proposed in this exercise, AI agents have the ability to make decisions that are very close to what a rational agent would do, although their decisions need not be identical. This is due to constant exploration by AI agents.

7 Summary and Future Work

In this paper, I show how an economic agent learns to make decisions in an unknown environment and how it adapts to changes in the underlying stochastic process of the economy. The agent learns adaptively through a deep reinforcement learning algorithm.

This agent does not follow the rational expectations hypothesis. It also does not follow a pre-specified decision or forecast rule that is similar to econometric learning agents. The AI agent is born without knowing what is feasible and desirable in its choice set. It does not know if and how its actions have an impact on the environment. I model how this agent interacts with its environment and gains experience. The experience is stored in the agent's memory, and is used to update the agent's evolving subjective beliefs. The agent's decision-making strategy is formed and adjusted based on its evolving subjective belief.

I use a stochastic optimal growth model where an economic agent needs to make consumption-saving decisions to maximise its lifetime utilities. Through several simulation experiments, I show three sets of results.

First, the AI agent can learn to make consumption-saving decisions in a stochastic environment with very limited information. Its learned policy function can also be similar to that of a rational expectations agent.

Second, by imposing a transitory shock and a permanent change on the underlying stochastic process of the economy, I show that AI agents are adaptable. In particular, a transitory shock leads to a temporary response from the agent and no permanent change. Whereas the permanent change in the stochastic process leads to a sustained shift in agents' consumption behaviours.

Last but not least, to highlight the purpose and novelty of the exploration parameter, I run all experiments on three AI agents that differ in their levels of exploration. This yields differences in their past experience and information collected. This generates different consumption-saving behaviours. Three agents also stabilise around different levels of wealth even though they live in the same environment facing the same stochastic shocks. With a high level of exploration, the agent is overly adventurous, and sacrifices their welfare for an unknown/untested action. With a low 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.

This work proposes a framework to model bounded rationality. It is flexible and can be tailored to different economic environments, memory sampling strategies, and sample sizes. It supports further studies on important economic questions that include structural breaks, regime changes, and multi-agent learning. Owing to artificial neural networks, the agent's decision-making centre does not need to follow a pre-specified functional form, and it is adaptable to the evolving subjective belief of an AI agent, which aligns with the empirical evidence of learning with a use-dependent brain by Malmendier (2021).

References

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`.

M. Botvinick, J. X. Wang, W. Dabney, K. J. Miller, and Z. Kurth-Nelson. Deep reinforcement learning and its neuroscientific implications. *CoRR*, abs/2007.03750, 2020. URL `https://arxiv.org/abs/2007.03750`.

M. Bray. Learning, Estimation, and Stability of Rational Expectations. *Journal of Economic Theory*, 26:318 – 339, 1982.

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.

M. Curry, A. Trott, S. Phade, Y. Bai, and S. Zheng. Finding general equilibria in many-agent economic simulations using deep reinforcement learning. *arXiv preprint arXiv:2201.01163*, 2022.

F. D'Acunto, U. Malmendier, J. Ospina, and M. Weber. Exposure to grocery prices and inflation expectations. *Journal of Political Economy*, 129(5):1615–1639, 2021. doi: 10.1086/713192. URL `https://doi.org/10.1086/713192`.

G. W. Evans and S. Honkapohja. *Learning and Expectations in Macroeconomics.* Princeton University Press, 2001.

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

E. Hill, M. Bardoscia, and A. Turrell. Solving heterogeneous general equilibrium economic models with deep reinforcement learning. *arXiv preprint arXiv:2103.16977*, 2021.

N. Hinterlang and A. Tänzer. Optimal monetary policy using reinforcement learning. 2021.

A. Kuriksha. An economy of neural networks: Learning from heterogeneous experiences. *arXiv preprint arXiv:2110.11582*, 2021.

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.

R. E. Lucas. Expectations and the neutrality of money. *Journal of Economic Theory*, 4(2): 103–124, 1972. ISSN 0022-0531. doi: https://doi.org/10.1016/0022-0531(72)90142-1. URL https://www.sciencedirect.com/science/article/pii/0022053172901421.

U. Malmendier. Exposure, experience, and expertise: Why personal histories matter in economics. Working Paper 29336, National Bureau of Economic Research, October 2021. URL http://www.nber.org/papers/w29336.

U. Malmendier and S. Nagel. Learning from Inflation Experiences *. *The Quarterly Journal of Economics*, 131(1):53–87, 2016. ISSN 0033-5533. doi: 10.1093/qje/qjv037. URL https://doi.org/10.1093/qje/qjv037.

A. Marcet and T. J. Sargent. Convergence of least-squares learning in environments with hidden state variables and private information. *Journal of Political Economy*, 97(6):1306–1322, 1989. ISSN 00223808, 1537534X. URL http://www.jstor.org/stable/1833240.

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.

J. F. Muth. Rational expectations and the theory of price movements. *Econometrica*, 29(3): 315–335, 1961. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1909635.

Y. Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3): 139–154, 2009. ISSN 0022-2496. doi: https://doi.org/10.1016/j.jmp.2008.12.005. URL https://www.sciencedirect.com/science/article/pii/S0022249608001181. Special Issue: Dynamic Decision Making.

T. Sargent. Rational expectations and the term structure of interest rates. *Journal of Money, Credit and Banking*, 4(1):74–97, 1972. URL https://EconPapers.repec.org/RePEc:mcb:jmoncb:v:4:y:1972:i:1:p:74-97.

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 the Analytical Solution to the Stochastic Optimal Growth Model

Given the Bellman equation:

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta E_t v(k_{t+1}, z_{t+1})\} \tag{1.19}$$

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

$$v(k_t, z_t) = \max_{k_{t+1} \in \Gamma(k_t)} \{log(z_t k_t - k_{t+1}) + \beta E_t(A + Blog(k_{t+1}) + Dlog(z_{t+1}))\} \tag{1.20}$$

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

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

The F.O.C:

$$\frac{\partial v(k_t, z_t)}{\partial k_{t+1}} = 0 \rightarrow -\frac{1}{z_t k_t - k_{t+1}} + \beta \frac{B}{k_{t+1}} = 0 \tag{1.23}$$

$$k_{t+1} = \frac{\beta B}{1 + \beta B} z_t k_t \tag{1.24}$$

Apply the envelope theorem

$$\frac{B}{k_t} = \frac{z_t}{z_t k_t - k_{t+1}} \rightarrow B = \frac{z_t k_t}{z_t k_t - k_{t+1}} \tag{1.25}$$

k_{t+1} can then be derived as

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

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

$$A + Blog(k_t) + Dlog(z_t) = log(z_t k_t - \beta z_t k_t) + \beta(A + Blog(\beta z_t k_t) + D\mu) \tag{1.27}$$

$$A = \frac{1}{1-\beta}(log(1-\beta) + \frac{\beta}{1-\beta}log\beta + \frac{\beta\mu}{1-\beta}) \tag{1.28}$$

$$B = D = \frac{1}{1-\beta} \tag{1.29}$$

The value function is thus

$$v^*(k,z) = \frac{1}{1-\beta}\{log(1-\alpha\beta) + \frac{\alpha\beta}{1-\alpha\beta}log\alpha\beta + \frac{\beta\mu}{1-\alpha\beta}\} + \frac{\alpha}{1-\alpha\beta}logk + \frac{1}{1-\alpha\beta}logz \tag{1.30}$$

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

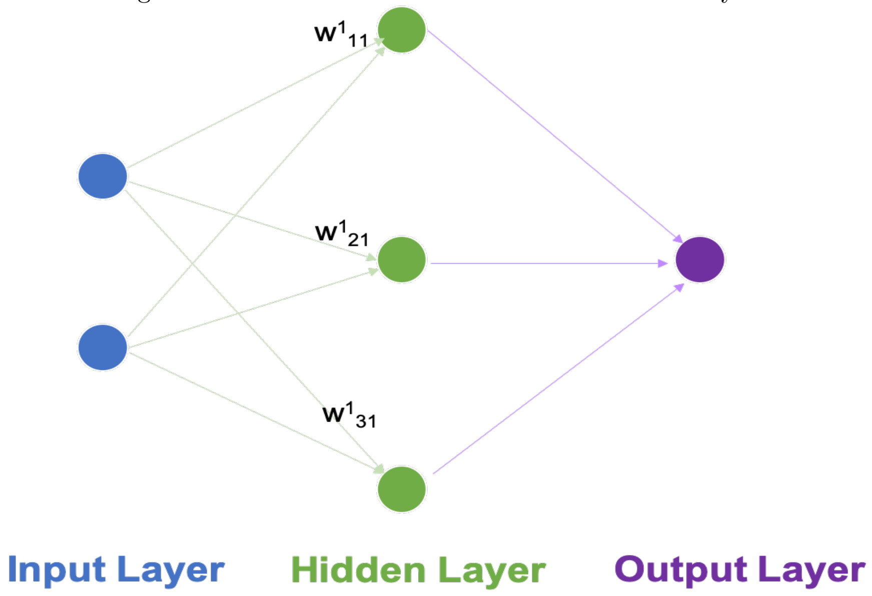
$$v^*(k,z) = \frac{1}{1-\beta}\{log(1-\alpha\beta) + \frac{\alpha\beta}{1-\alpha\beta}log\alpha\beta + \frac{\beta\mu}{(1-\alpha\beta)(1-\beta\rho)}\} + \frac{\alpha}{1-\alpha\beta}logk + \frac{1}{(1-\alpha\beta)(1-\beta\rho)}logz \tag{1.31}$$

Policy function is

$$k_{t+1} = \alpha\beta z_t k_t^{\alpha} \tag{1.32}$$

A.2 How do ANNs learn?

Figure 15: A feedfoward network with one hidden layer



Source: author's own construction

Figure 15 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 in 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 need to be learned 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 of 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 applies 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).

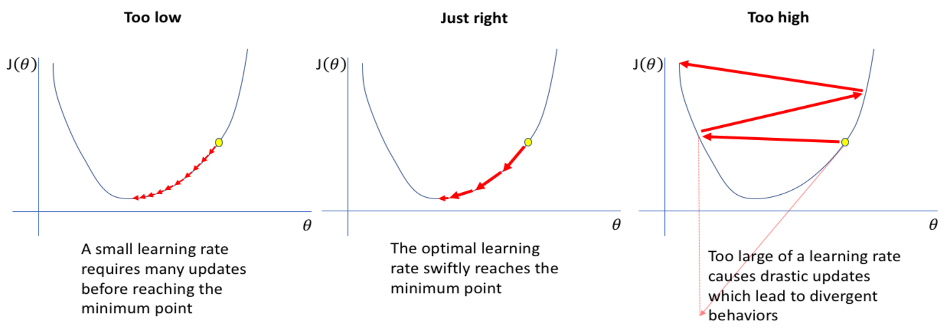
How do ANNs learn and update their parameters? They use the following procedures. Given some training dataset and an ANN model, pass the data forward through the neural network to obtain an output/prediction. Compare this output to a target value/goal. Calculate the loss between the predicted value and the target value. To make sure the neural network learns and updates 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. Specifically, 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, in order to update each weight of the network.

$$w'_{ij} = w_{ij} - \eta \frac{\partial Error}{\partial w_{ij}}, \tag{1.33}$$

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. The new weight w'_{ij} is a combination of the current weight w_{ij} and some weighted term $\eta \frac{dError}{dw_{ij}}$. η denotes learning rate, a hyperparameter. It represents how quickly weights are updated in response to the error contribution term. The higher the learning rate, the quicker the update is.

Figure 16: Learning rates


Too low
J(θ)
θ
A small learning rate requires many updates before reaching the minimum point
Just right
J(θ)
θ
The optimal learning rate swiftly reaches the minimum point
Too high
J(θ)
θ
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Source: jeremyjordan

To clearly illustrate the purpose of η, figure 16 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 a long time before it reaches the optimal solution.

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 the line and to the minimum point, which is similar to what figure 16 illustrates.

B Additional Result

Figure 17: Loss of the policy neural network

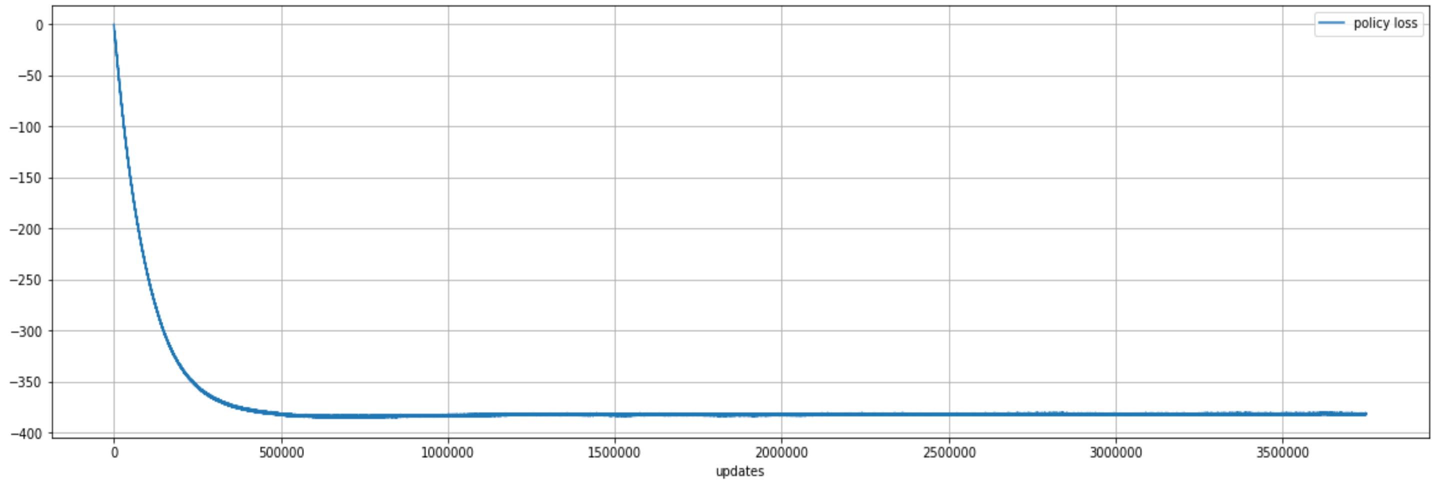


Figure 17 plots the loss of the policy network during this learning process. It plots the gradual reduction of the loss through learning. In other words, the agent makes more decisions that generate high rewards through learning.