INTERNATIONAL MONETARY FUND

Deep Reinforcement Learning: Emerging Trends in Macroeconomics and Future Prospects

Tohid Atashbar and Rui (Aruhan) Shi

WP/22/259

IMF Working Papers **describe research in progress by the author(s) and are published to elicit comments and to encourage debate.** The views expressed in IMF Working Papers are those of the author(s) and do not necessarily represent the views of the IMF, its Executive Board, or IMF management.

2022
DEC

WORKING PAPER

© 2022 International Monetary Fund WP/22/259

IMF Working Paper
Strategy, Policy, and Review Department

Deep Reinforcement Learning: Emerging Trends in Macroeconomics and Future Prospects

Prepared by Tohid Atashbar and Rui (Aruhan) Shi

Authorized for distribution by Stephan Danninger
December 2022

***IMF Working Papers* describe research in progress by the author(s) and are published to elicit comments and to encourage debate.** The views expressed in IMF Working Papers are those of the author(s) and do not necessarily represent the views of the IMF, its Executive Board, or IMF management.

ABSTRACT: The application of Deep Reinforcement Learning (DRL) in economics has been an area of active research in recent years. A number of recent works have shown how deep reinforcement learning can be used to study a variety of economic problems, including optimal policy-making, game theory, and bounded rationality. In this paper, after a theoretical introduction to deep reinforcement learning and various DRL algorithms, we provide an overview of the literature on deep reinforcement learning in economics, with a focus on the main applications of deep reinforcement learning in macromodeling. Then, we analyze the potentials and limitations of deep reinforcement learning in macroeconomics and identify a number of issues that need to be addressed in order for deep reinforcement learning to be more widely used in macro modeling.

RECOMMENDED CITATION: T. Atashbar and R.A. Shi 2022. “ Deep Reinforcement Learning: Emerging Trends in Macroeconomics and Future Prospects”, IMF Working Papers, WP/22/259.

JEL Classification Numbers:	C63, C89, D85, D87
Keywords:	Reinforcement learning; Deep reinforcement learning; Artificial intelligence, RL; DRL; Learning algorithms; Macro modeling
Author’s E-Mail Address:	tatashbar@imf.org; ashi@imf.org

WORKING PAPERS

Deep Reinforcement Learning: Emerging Trends in Macroeconomics and Future Prospects

Prepared by Tohid Atashbar, and Rui (Aruhan) Shi

Contents

ANNEX

FIGURES

TABLE

Glossary

AC Actor-Critic

A2C Advantage Actor-Critic

A3C Asynchronous Advantage Actor Critic

ANNs Artificial Neural Networks

CDRL Causal Deep Reinforcement Learning

DL Deep Learning

DRL Deep Reinforcement Learning

DDPG Deep Deterministic Policy Gradients

DQN Deep Q Networks

MDP Markov Decision Processes

ML Machine Learning

RL Reinforcement Learning

PPO Proximal Policy Optimization

QRL Quantum reinforcement learning

SARSA State-Action-Reward-State-Action

TD Temporal Difference

TRPO Trust Region Policy Optimization

Introduction

Artificial intelligence (AI) technologies are widely used in many fields. Deep reinforcement learning (DRL) as a subset of AI is one of the most effective approaches to solve complex problems through interactive learning and has been widely used in many fields, including robotics, autonomous driving, computer vision, and natural language processing. DRL has made significant advances in recent years. These advances are driven by the combination of various methods in deep learning (DL) and reinforcement learning (RL), a branch of machine learning (ML) that deals with how agents can learn from the environment through interactions to maximize their reward. This combination can provide RL algorithms with more expressive power, making them more capable of learning and generalizing from data.

DRL models can learn from multi-dimensional and continuous state-action spaces, and non-stationary complex and changing environments, which make them more efficient. These models also theoretically have the ability to improve their own learning algorithms through a process known as meta learning—a machine learning technique that can learn how to improve learning which is also known as “learning to learn”, “learning from experience”, or “learning by doing”.

The application of reinforcement leaning and deep reinforcement learning in economics has been limited (but is gaining traction), mainly due to the difficulty in modeling the environment, designing optimal reward functions and solving the high dimensional economic models.

This paper discusses recent applications of deep reinforcement learning algorithms in economics while introducing the main theory of reinforcement learning (RL) and deep reinforcement learning (DRL)— in relatively non-technical terms. Then, with a focus on macroeconomics, it explores the prospects of this class of algorithms and their open issues.

Through discussing several use cases of deep reinforcement learning algorithms in economics[1], this paper aims to shed light on this class of algorithms: how it can be used to answer existing questions and encourage a new venue of research.

I. What is Reinforcement Learning and Deep Reinforcement Learning?

Reinforcement learning (RL) is a type of machine learning that allows agents to learn in an interactive environment (primarily through trial and error) by taking actions and receiving rewards for their actions. Deep reinforcement learning (DRL) is a type of reinforcement learning that uses deep neural networks to approximate its value or policy functions which are then used to guide the agent's decisions.

Reinforcement learning is different from both supervised learning and unsupervised learning in the literature of machine learning. Supervised learning requires a labelled training dataset so that it can be used to measure how well a supervised machine learns to predict the labelled output. Unsupervised learning methods are mainly for data without labels in order to classify or find a structure in the data. RL, however, does not require a training dataset as seen in supervised or unsupervised learning. It learns from the interactions with what is called an “environment”. The agent learns through a reward function, which is a feedback mechanism that tells

[1] Surveys on applications of machine learning techniques and AI technologies in economics. Mosavi et al (2020) focus on the applications of neural networks rather than reinforcement learning algorithms. Charpentier et al (2021) focus on economics and finance, however not many actual cases of applications are included in their paper. Athey (2018) discusses the impact of machine learning on economics and highlights the importance and caveats of using machine learning methods. She focuses on supervised and unsupervised machine learning methods without many discussions on reinforcement learning algorithms. Fisher (2018) provides a survey of applications of reinforcement learning techniques in financial markets.

the algorithm if it is doing the right thing and tries to find the best possible response in that environment. Rl algorithms generate simulation data from having RL agents actively interact with the environment and the simulated data or learning experience is used to solve the RL problems.

A. Brief History

Reinforcement learning has its roots in behaviorism and was first proposed as a way to explain animal learning behavior. RL is heavily connected to psychology and neuroscience. The early development of RL involves the works of Alan Turing, Norbert Wiener, and Richard Bellman. However, it wasn't until the 1970s that reinforcement learning began to be studied extensively by computer scientists.

Two segments of literature have played a significant role in the development of RL dating back to its infancy: the first one is the development of temporal-difference updating (learning from experience by using the difference between expected values and actual ones) and trail-and-error learning (in which the agent tries different actions in order to find the best action or maximum reward for a given situation), which has close links to the psychology literature on animals learning; the second is the literature of optimal control, which involves dynamic programming (Sutton and Barto, 2018).

In early artificial intelligence, researchers in engineering literature explored the idea of trail-and-error. Minsky (1954) and Farley and Clark (1954) were among the first two researchers who investigated the use of trail-and-error in computational techniques. Minsky (1961) started using the term "reinforcement" and "reinforcement learning". His paper discussed one central problem in reinforcement learning, the credit assignment problem, i.e., how do you distribute credit for success among the many decisions that may have been involved in producing it?

In the late 1950s and early 1960s, RL was more formalized in work by Bellman, which became the foundation of dynamic programming. Optimal control was first used to describe the problem of designing a controller to minimize a measure of the behavior of a dynamic system over time. Bellman (1957) was among the first to develop a solution to solve this problem, extending an early theory of Hamilton and Jacobi. In Bellman's approach, it used an optimal return function, known as the value function or Bellman equation, and the optimal control problems were solved by solving this equation. Klopf (1972) combined trial-and-error learning with temporal-difference learning. Moreover, he also linked trial-and-error learning to the psychology of animal learning. This was extended by Sutton and Barto (1981), in which they described learning rules as driven by changes in temporally successive predictions (hence temporal difference updating). They also developed a psychological model of classical conditioning based on temporal-difference learning.

In the 1970s and early 1980s, RL was combined with control theory and adaptive control to describe the behavior of agents. In the late 1980s, RL reached the machine learning community, where it was applied to problems such as simple mobile robots and chess. Watkin (1989) combined the temporal-difference learning and the optimal control literature in his development of Q-learning algorithm, which was the first modern RL algorithm.

In the early 1990s, RL was applied to simple problems such as balancing an inverted pendulum and playing backgammon. In 1995, Tesauro proposed the use of Temporal difference (TD) learning for playing the game of backgammon and showed that it could achieve human performance. Schultz et al (1997) explored the connections between temporal-difference learning and neuroscience. In the same year, Schmidhuber proposed an algorithm for learning complex control policies for high-dimensional robot control tasks. In the same year, Sutton and Barto proposed an algorithm called Sarsa, one of the most popular RL algorithms.

In the mid-1990s, RL was combined with function approximation, which allowed it to be applied to problems such as learning how to control a robot arm. In the late 1990s, RL was combined with artificial neural networks, which allowed it to be applied to problems such as learning how to play video games. In the early 2000s, RL was combined with evolutionary computation, which allowed it to be applied to problems such as learning how to control a robot arm.

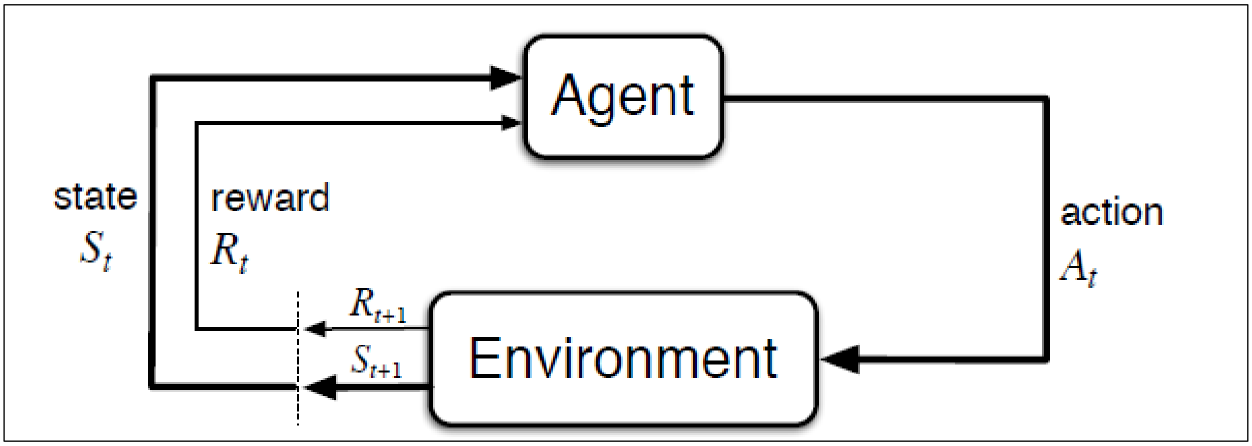
And finally in the late 2000s, RL was combined with deep learning, which allowed it to be applied to problems such as learning how to play video games or navigate 3D environments. Since then, deep RL has seen growing interest and has been applied to many tasks in robotics, healthcare, finance, and many other fields. Recent applications in the area of economics are discussed in the remainder of the paper.

B. Theory

B.1 A reinforcement learning problem and terminologies

Reinforcement Learning involves a class of algorithms that aims to solve Markov Decision Processes (MDPs). An MDP is a mathematical framework for modeling decision-making in situations where outcomes are uncertain; partly random and partly under the control of a decision-maker. MDP could be considered as a model that helps an agent take the best decision in any given state by considering the immediate and longer-term rewards. The environment in RL is usually modeled as a Markov Chain, which means that the next state of the environment only depends on the current state and not on the past states. This makes the problem much simpler to solve at the cost of simplifying learning processes in real world situations.

Figure 1. A Markov Decision Process (A Reinforcement Learning Problem)



Source: Sutton and Barto (2018, page48)

Figure 1 shows an MDP. Given a state that represents the environment, a RL agent makes an action, and the state transitions to the next in part due to the agent's action. The agent then receives a reward.

In a setting of an economic environment, taking a simple stochastic optimal growth model as an example, the representative household can be an RL agent. The state could be a realized exogenous shock and the agent's available resource, the action could be the decision on consumption-investment, and the reward could be the utility, which depends on consumption level. How state transitions depend on the underlying data generating process of the economy. In a comparable case, the state could be a combination of all the key factors that can be affected by households' decisions. The action space is the amount of investment in different sectors. The goal of a household is to maximize the cumulative utility from consumption. The RL agent can learn a near-optimal policy to allocate investment for different sectors to maximize the utility.

In a different illustration using the businesses as agents, the agent could be a firm that is trying to maximize its profits. The state could be the current market conditions, such as the prices of inputs and outputs, the level of demand, etc. The action space would be the different production levels that the firm can choose. The goal of the firm would be to learn a policy that would allow it to maximize its profits.

Table 1. Terminologies in Reinforcement Learning

Terminologies	Description
State, $s \in S$	A representation of an environment, drawn from the state space
Action, $a \in A$	Behavior of the RL agent, drawn from the action space
Reward, $R(s, a)$	A stimulus sent to the RL agent in part due to its action and the current state
Policy function, $\pi(s)$	Decision making strategy of the agent, a mapping from state to action (deterministic policy) or a distribution of actions (stochastic policy)
Value function, $Q(s, a)$	Expected cumulative rewards, a mapping from a state-action pair to the expected value

Table 1 summarizes the main terminologies used in RL. State is a representation of an environment, drawn from a state space. Action denotes the agent's choices, drawn from an action space. Reward is a function that depends on state and action, which is a stimulus to guide the agent's decision and learning. A policy function, can be either stochastic or deterministic, is the decision-making strategy of the RL agent. A stochastic policy returns a probability distribution of actions given states. A deterministic policy returns a single action given states. Action-value function, taking the input of state-action pair, gives the cumulative rewards (in expectation).

The goal of a RL agent is to find the optimal policy, i.e., π^*, that maximizes its sum of rewards, which is approximated by the value function, as described by equation (1).

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, Q(s, a) \qquad 1$$

The value function is a state-action pair that indicates how good it is for the agent to be in a given state and take a given action. The value function is also known as the state-action value function or the Q-function.

Solving for RL rules can involve both tabular methods and approximation methods. When the policy or/and value functions become difficult to track with tabular methods, function approximators are usually applied. Artificial neural networks (ANNs) are an example of such approximations. The resulting class of algorithms are classified as deep RL.

Exploration vs Exploitation

An RL agent faces a trade-off between exploitation and exploration. Exploration refers to the process of trying out different actions in order to find the best possible action which allows the agent to sample the environment, thereby providing information that helps the agent to find the optimal policy. Exploitation on the other hand refers to the process of taking the best action that is known and allows the agent to reap the maximum possible reward assuming that the agent has found the optimal policy.

The balance between exploration and exploitation is important because it determines how quickly the agent will find the best possible action and the optimal strategy for an agent to learn while maximizing the rewards. If the agent explores too much, it will not exploit the knowledge it has gained and will not learn anything useful. On the other hand, if it does not explore enough, it may never discover the optimal solution.

There are several ways to balance exploration and exploitation. One simple way to address the exploration-exploitation dilemma is to use an exploration strategy that encourages the agent to explore more in the beginning and then slows down the exploration as the agent learns more about the environment. This is known as an exploration schedule.

Similarly, one simple way to encourage exploration is to use an exploration bonus. This is an extra reward that the agent gets for exploring new states. This can be used to encourage the agent to try new things and to explore more of the state space.

Yang et al (2022) survey different existing exploration strategies for solving RL problems. Two main strategies include ϵ- greedy and noise perturbation.

*ϵ- greedy: W*ith probability ϵ, a RL agent chooses actions randomly (exploration), and with probability $1-\epsilon$, the agent chooses actions greedily (exploitation). It is a simple and easy to implement method and used in algorithms such as Q-learning[1]. It, however, can be inefficient in complex problems that involves large state-action spaces or continuous action spaces.

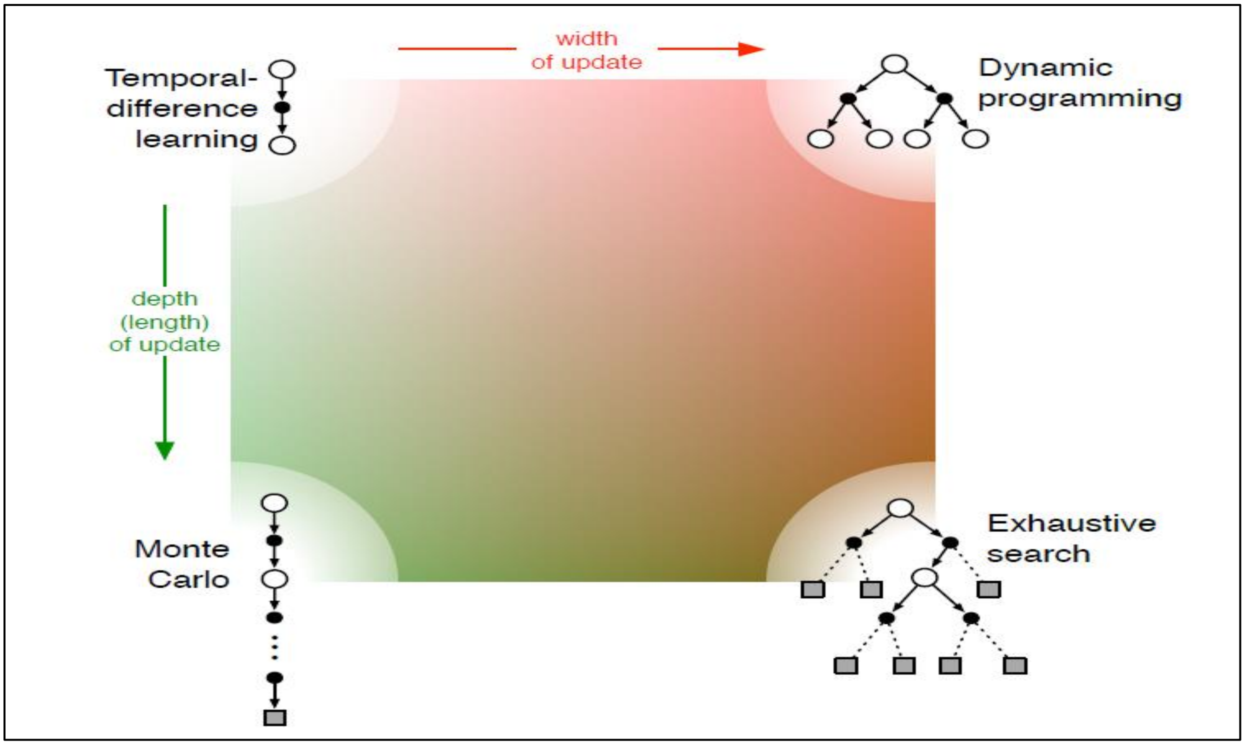
Noise perturbation: This involves adding a noise term to a deterministic policy, which could take the following form,

$$a_t = \pi(s_t) + \mathcal{N}_t \qquad 2$$

Equation (2) means that the action of a RL agent is the output of its deterministic policy plus a noise term to explore the actions space. The noise can be sampled from a Gaussian process or an autoregressive process. This strategy can be used for RL problems with continuous action spaces.

B.2 Main Algorithms

Figure 2. Comparison of RL and Other Methods



Source: Sutton and Barto (2018, page190).

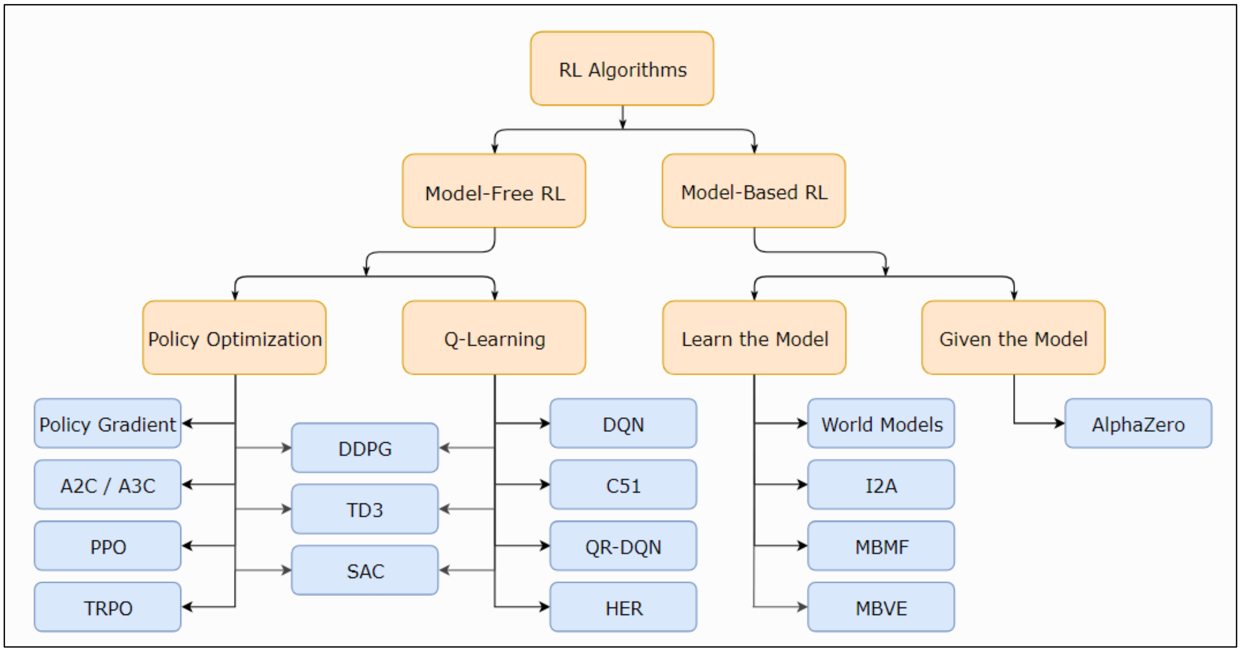
[1] Q-learning is an off-policy model-free learning algorithm, meaning that the agent does not need to follow the current policy in order to learn. This is advantageous because it allows the agent to explore different policies and find the one that leads to the highest rewards. The algorithm works by first observing the environment and then choosing an action based on a set of rules. The agent then receives a reward based on the action taken. The agent then uses this information to update the set of rules that it uses to choose actions.

As solution methods for MDPs, RL algorithms are similar to both dynamic programming and Monte Carlo updating. Dynamic programming algorithms are similar to reinforcement learning algorithms in that they both update a value function that is used to make decisions. However, the difference is that dynamic programming algorithms do this offline, while reinforcement learning algorithms do this online, meaning that they update the value function as they interact with the environment. Monte Carlo updating is also similar to reinforcement learning in that it updates a value function that is used to make decisions. However, the difference is that Monte Carlo updating does this by using samples of the environment, while reinforcement learning algorithms do this by interacting with the environment. One advantage of RL over dynamic programming algorithms lies in its ability to learn directly from raw meaning that it does not require a model of the environment. The advantage over Monte Carlo algorithms lies in its ability to learn without waiting for an episode to terminate.

Figure 2 compares RL methods with the existing methods of optimal control. It has two dimensions. Along the vertical dimension, it shows the depth of updating, and the degree of bootstrapping. RL methods usually update at each step, in contrast to Monte Carlo method that requires a full depth of history before updating the target value. Temporal difference learning refers to the update of target value with the current new information, while Monte Carlo learning refers to the update of target value with the entire trajectory. Exhaustive search methods refer to the update of target value with all possible trajectories.

Along the horizontal dimension is the width of updating. It shows whether the method is based on a sample trajectory of agent or expected updates. Expected updates mean that the method requires a fully specified transition dynamics, such as dynamic programming. RL methods, however, do not require all the transition information, and its methods are usually based on a sample trajectory.

Figure 3. RL Algorithm Overview



Source: *Part 2: Kinds of RL Algorithms — Spinning Up documentation (openai.com)*

Model-based vs Model-free Learning

RL algorithms can be broadly classified into model-based and model-free methods. Figure 3 shows an overview of some RL algorithms with two main branches: model-free and model-based. Model-based methods are learning algorithms that require a model of the environment (model of the world). They are also called planning algorithms. In contrast, Model-free methods are learning algorithms that do not require a model of the environment. They are also called learning from experience or trial-and-error learning algorithms.

A classic example of a model-based algorithm that is used in macroeconomics is dynamic programming, in which state transition probabilities are fully present. RL algorithms have the advantage of learning in a model-free environment, which is for an environment with unknown transition probability distributions. Many currently applied RL algorithms follow a model-free approach, e.g., temporal-difference, policy gradient and actor-critic. Advanced DRL applications commonly benefit from model-free algorithms, among them Deep Deterministic Policy Gradients (DDPG), Deep Q Networks (DQN), Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), Advantage Actor-Critic (A2C) and Asynchronous Advantage Actor Critic (A3C)

The model-based and model-free RL methods can be further classified into value-based and policy-based methods. Value-based methods are model-based or model-free methods that learn a “value function” that can be used to find the optimal action. Policy-based methods are methods that learn a “policy” that can be used to find the optimal action.

The model-based and model-free methods can also be classified into on-policy and off-policy methods. On-policy methods are learning algorithms that learn from the current policy. Off-policy methods are learning algorithms that can learn from any policy.

The criteria that determine the reasons for using one or the other learning algorithms in RL could be related to a variety of factors including the size of the state space, the computational constraints, or the types of problems to be solved. For example, model-free RL algorithms are more easily used for problems with large state spaces. Other factors that could affect the choice of learning algorithm include the amount of prior knowledge about the environment, the types of feedback available, and the types of reinforcement signals that are used.

Temporal-difference Updating: one step ahead

Temporal-difference updating is an important milestone in the field of RL. As Sutton and Barto (2018) argue, “if one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning”. They also argue that TD learning is powerful because it can learn from raw experience without a model of the environment and can bootstrap, which means it can learn from incomplete sequences. TD learning also has close connections to biology; it has been argued that the brain may use TD learning to do credit assignment (O'Reilly et al., 2012).

TD learning combines both the ideas of Monte Carlo and dynamic programming. “Temporal” refers to the fact that the agent is learning from intermediate rewards at every time step, rather than waiting for a terminal reward. "Difference" refers to the fact that the agent updates its values by making an estimate of the value of a state by bootstrapping from the value of the next state using the difference between the expected reward and the actual reward. This way of updating allows a RL agent to learn and update at every period. The update is done through calculating a TD-error term. A one step TD-error term is defined as follows.

$$R_{t+1} + \gamma\, Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

It shows that the TD-error term is composed of the reward R_{t+1} plus a discounted future value $\gamma\, Q(S_{t+1}, A_{t+1})$ minus the current value $Q(S_t, A_t)$.

There are two main types of TD learning: State–action–reward–state–action (SARSA) and Q-learning. In SARSA, the agent learns by taking an action, observing the next state and reward, and then taking another action. In Q-learning, the agent learns by taking an action, observing the next state and reward, and then taking the best action given the learning stage.

Action-Value approach: Q-learning

Action-value approach includes methods that focus on updating the action value function. RL agents in this approach first learn the values of actions and then select actions based on their estimated action values. By estimating the expected reward for each state-action pair, the agent can learn which actions are most likely to lead to the highest rewards, and then choose the best action accordingly. One prominent example of an action-value approach algorithm is Q-learning. Q refers to an action-value function that takes in a state and an action, and outputs a numeric reward. The goal of the algorithm is to find an optimal action-selection policy, i.e., a policy that maximizes the expected cumulative reward.

Q-learning algorithm was introduced by Watkins (1989)[2]. It uses the TD-updating introduced in the previous section. The central updating/learning equation is defined as,

$$Q(S', A') = \ Q(S, A) + \alpha[R + \gamma \ \max_{a} Q(S', A') - Q(S, A)] \qquad 3$$

Equation (3) says that the updated Q function, i.e., $Q(S', A')$, depends on the existing Q function, $Q(S, A)$, plus an updating term, which is the TD-error term.

In this case, Q function is an approximate of the optimal action-value function, independent of the policy being followed. Policy function here influences which state-action pairs are visited and updated. The main procedure in this algorithm is to map different pairs of state-action with a value that approximates the expected return. For a learning agent to find optimal behaviors, it needs to compare Q values for different state-action pairs, which can be costly and inefficient in cases with a large state and action space. It also faces difficulty in learning when the action space is continuous.

When artificial neural networks (ANNs) are used to approximate the action-value function, the resulting algorithm is called deep Q-learning (DQN). In fact, DQN is an algorithm that combines Q-learning with deep learning. In Q-learning, the action-value function is approximated using a table, meaning that it is essentially memorized and is represented as a set of values for every possible state-action pair. In contrast, in deep learning, the approximation is done using ANNs with many hidden layers in a neural network. DQN is an off-policy algorithm, meaning that it can learn from data that is not necessarily generated by the algorithm itself. DQN was first proposed and patented [3] by Google DeepMind researchers (Mnih et al., 2015). The algorithm was used to successfully train an agent to play a variety of Atari games.

Policy Gradient Approach

In a policy-function approach, a RL agent learns the policy function parameters. The learning is based on the gradient of a performance measure, call it $J(\theta)$, where θ denotes the policy function parameter. To maximize performance measure $J(\theta)$, the updating of θ is done through gradient ascent on the measure J.

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)} \qquad 4$$

In equation (4), $\widehat{\nabla J(\theta_t)} \in \mathbb{R}^d$ is a stochastic estimate whose expectation approximates the gradient of the performance measure with respect to its argument θ_t.

To put it simply, the agent starts with some initial policy function parameters. It then interacts with the environment, and at each step, it receives a state and chooses an action according to the policy function. After taking the action, it receives a reward and a new state. The agent then updates the policy function parameters using the gradient of the expected total rewards with respect to the policy function parameters. The process is repeated until the agent converges to an optimal policy function.

[2] The full algorithm is supplied in the annex.

[3] https://patentimages.storage.googleapis.com/71/91/4a/c5cf4ffa56f705/US20150100530A1.pdf

One example of a policy-gradient algorithm is REINFORCE. Its pseudocode algorithm is supplied in the annex (Sutton and Barto, 2018). REINFORCE uses Monte Carlo sampling to estimate the gradient of the expected return with respect to the policy parameters. REINFORCE can be used with any parametric policy, such as a neural network.

Reinforce has a number of advantages over other policy gradient algorithms. REINFORCE is relatively simple to implement and can be used with any parametric policy. REINFORCE is also efficient, as it only requires a single pass through the data to estimate the gradient. However, REINFORCE also has a number of disadvantages. REINFORCE is biased, as the gradient is estimated using Monte Carlo sampling. This can lead to poor performance on complex tasks. REINFORCE is also slow, as it requires a full pass through the data to estimate the gradient. Castro et al (2022) adopt the REINFORCE algorithm in searching for an optimal solution for a bank's participation in the high-value payment system. REINFORCE has been used in a variety of tasks, including robotic control, and natural language processing.

Actor-critic Approach

Actor critic in reinforcement learning refer to methods that learn a value function and a policy jointly, using the gradient of the value function to update the policy. It combines the central updating equations for both action-value functions, equation (3), and policy functions, equation (4). These methods are particularly effective in high dimensional environments. One advantage of using the value function gradient to directly update the policy is that it can reduce the variance of the gradient estimate, as compared to using the score function gradient. However, this may come at the expense of increased bias in the gradient estimate, mainly because the value function is being updated in the same direction as the current policy. Another advantage of critic algorithms is that they can learn faster than value-only methods, since they can make use of the information in the value function to guide the learning of the policy. A disadvantage of critic algorithms is that they can be unstable, since the gradient of the value function can be very sensitive to small changes in the environment or the policy. Another disadvantage is that they require more computational resources than value-only methods since they must solve an optimization problem at each step.

A popular actor-critic algorithm is the Deep Deterministic Policy Gradient (DDPG) algorithm, first introduced by Lillicrap et al (2015). DDPG is an extension of the deterministic policy gradient algorithm, which is itself an extension of the policy gradient algorithm. DDPG is an off-policy algorithm that can be used to learn a continuous control task in an environment with high dimensional state space. In this algorithm, the policy and action-value functions are approximated by their respective ANNs. DDPG has been successful in a number of continuous control tasks, including robotics tasks such as manipulation and locomotion. However, like other actor-critic algorithms, it can be difficult to tune the parameters of the networks. Additionally, DDPG is not guaranteed to converge to the optimal policy. Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) is a new algorithm that addresses some problems of DDPG including overestimation of Q-values.

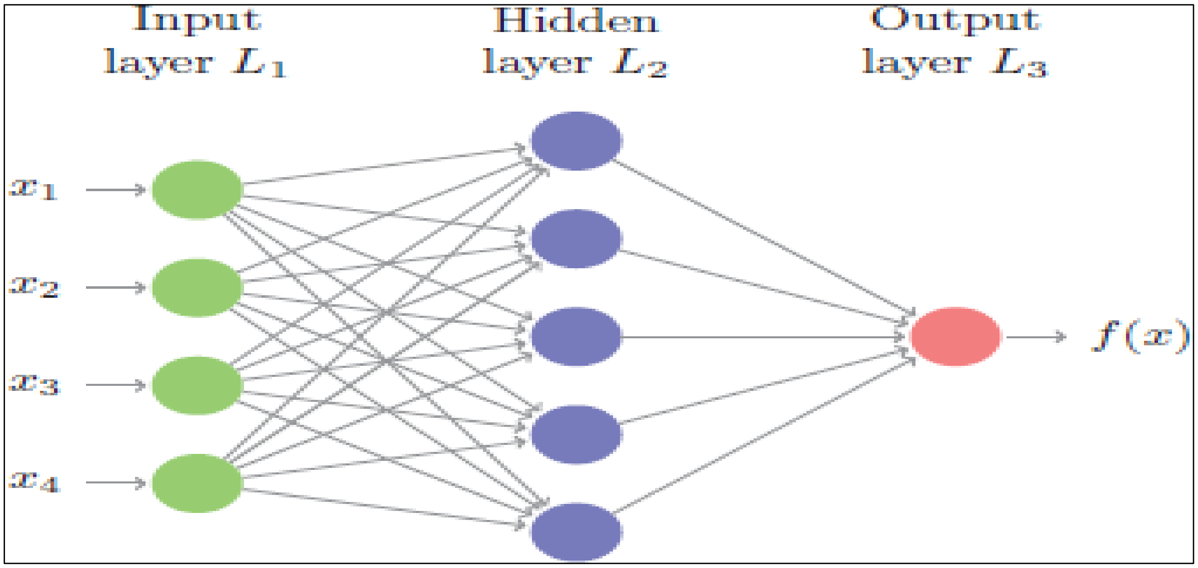
Advantage Actor-Critic (A2C) and Asynchronous Advantage Actor Critic (AC3) are other examples of actor-critic algorithms. AC2 is a modification of the original AC algorithm and employs a trust region model to allow faster and more stable training of the agent. The trust region refers to an area around the current policy that the agent does not want to explore too much to accept new actions. This allows for faster and more stable training of the agent. AC2 trains the actor and the critic simultaneously. The actor is trained to improve the quality of the action taken by the agent, whereas the critic is trained to evaluate the actions of the agent. A3C is an improvement over the A2C algorithm, as it uses an asynchronous and more efficient gradient descent method to update the parameters of the agent. A3C has been shown to outperform A2C in terms of both sample efficiency and learning speed.

B.3 RL and the Use of Artificial Neural Networks

Artificial neural networks (ANNs)[4] are widely used as nonlinear function approximators. Many RL algorithms use ANN function approximation, which sometimes is called neural reinforcement learning and produce fruitful results in the AI literature.

An ANN is a group of interconnected units that are named neurons, which resembles the components of nervous system and are used to process information. The output of the network is determined by the weights of the connections between the units. The pattern of weights is often determined by a learning algorithm. The learning algorithm adjusts the weights of the links in order to minimize the error between the output of the network and the desired output. A simple feedforward neural network is shown in Figure 3.

Figure 4. A Feedforward Deep ANN



Source: Feedforward Deep Learning Models · UC Business Analytics R Programming Guide (uc-r.github.io)

Figure 3. network has three layers. It is also known as a deep network, because of the presence of a hidden layer. Information (or data) flows from the input nodes (the x_s) and feeds through the neurons in the hidden layer. The filtered information is then passed through the output layer and produce an output $f(x)$.

ANNs are flexible and can approximate different nonlinear functions.

B.4 Deep RL Algorithms: Putting It All Together with Experience Replay and Core Structure

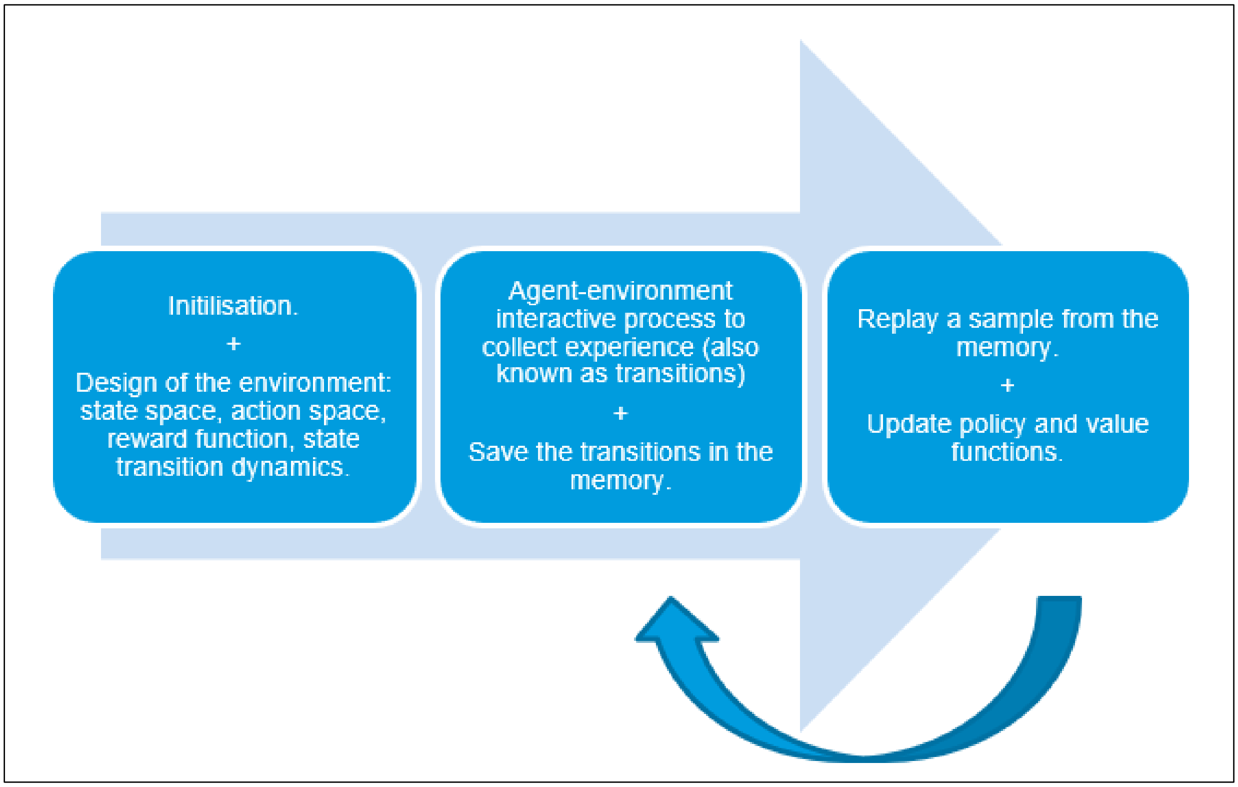
Most deep RL algorithms involve two main components: an agent-environment interactive process to collect experience, and a learning component to update policy and/or value functions. To update functions, a method called experience replay is used. Experience replay is a sampling method that stores samples from the agent's experience while it is still actively learning and replays them later during the training process. The idea behind experience replay is that by replaying past experiences, the agent can learn from them without being influenced by the current state of the environment. Experience replay is also used to break the correlation between consecutive samples, which can help the agent learn faster and more effectively.

[4] Goodfellow et al (2016) have an in-depth analysis and discussions on different architectures of ANNs, which are not discussed further here.

The flow of most deep RL algorithms is as follows:

- Initiate an empty memory
- Step up the environment; design the state and action space; specify the reward function with respect to state and action.
- Agent-environment interactive process to collect experience (also known as transitions) that involve state, action, next state, and reward.
- Save the transitions in the memory.
- Sample from the memory, and update either the action-value function or the policy functions or both (updating usually follows equation 3 and 4).
- The updated policy and value functions can be used to guide the RL agent's interactive process with the environment further.
- The agent-environment interactive process continues until the task is solved or a terminal state is reached, which is problem dependent.

Figure 5. Workflow of Deep RL Algorithms

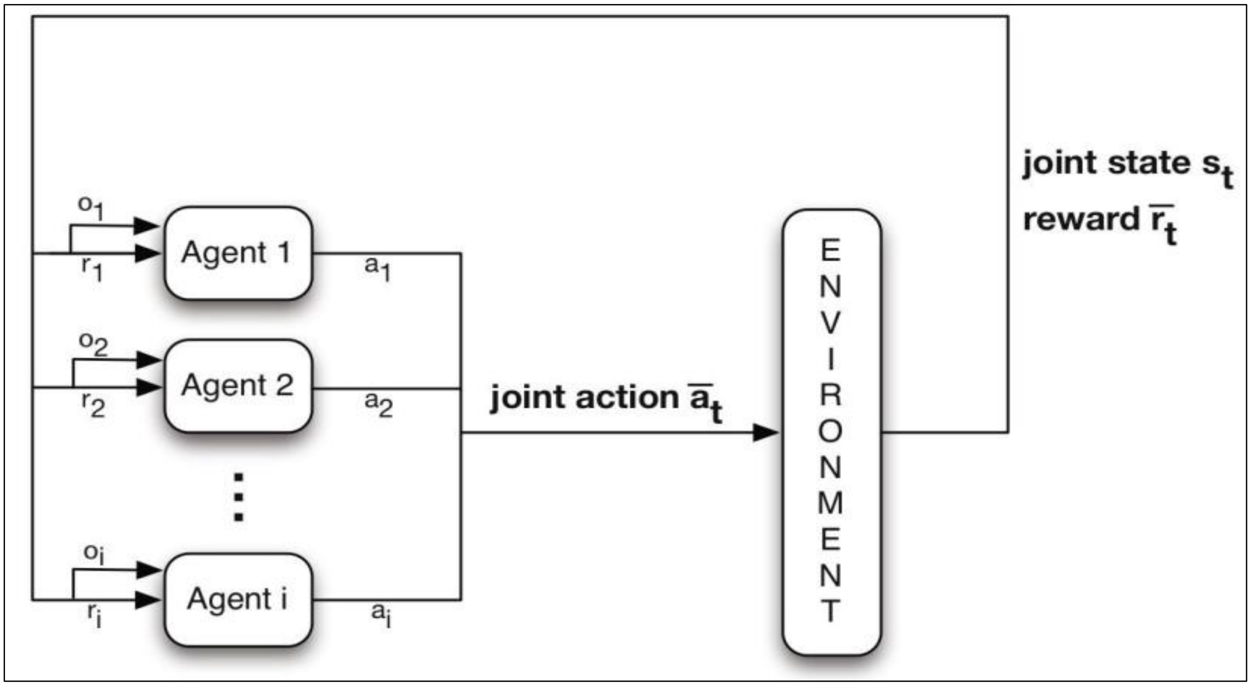

Initilisation.
+
Design of the environment: state space, action space, reward function, state transition dynamics.
Agent-environment interactive process to collect experience (also known as transitions)
+
Save the transitions in the memory.
Replay a sample from the memory.
+
Update policy and value functions.

Source: Authors' construction.

B.5 Multiagent Deep Reinforcement Learning

Many recent studies have shifted their focus from single-agent RL problems to multi-agent learning scenarios. In contrast to a single-agent RL problem (i.e., figure 1), figure 4 shows a visualization of a multi-agent RL problem.

Figure 6. Multiagent RL



Source : Nowe et al (2012)

In a learning scenario with multiple agents, the MDP is generalized to a stochastic game, or a Markov game. To provide a short description of a Markov game, denote n as the number of agents, S as a discrete set of environmental states, and A_i, where $i = 1,2, \dots, n$, as a set of actions for each agent i. Define the joint set of actions for all agents as $A = A_1 \times A_2 \times \dots \times A_n$. The value function of each agent is dependent on the joint action and joint policy.

Multi-agent learning can be viewed as a distributed optimization problem that requires agents to cooperate with each other to achieve a common goal. The problem of multi-agent learning is further complicated by the presence of non-cooperative agents in the environment who seek to deviate from the commonly agreed upon optimal policy in order to maximize their own rewards.

The key question in multiagent learning is how the agents interact with each other. The traditional approach to multiagent learning is to use a centralized training algorithm with a centralized evaluation function. However, this approach does not scale well to large numbers of agents and can be computationally expensive. An alternative approach is to use a decentralized training algorithm with a decentralized evaluation function. Decentralized training algorithms are more scalable and can be more efficient, but they often require more communication between the agents. A recent trend in multiagent learning is to use a hybrid approach that combines centralized and decentralized training. In this approach, the agents are trained using a centralized training algorithm, but the evaluation function is decentralized.

Multiagent systems can be used to solve complex tasks through the cooperation of individual agents(Nguyen et al., 2020). Zhang et al (2021) has a selected overview of currently used algorithms and applications for multiagent RL problems. Fudenberg and Levine (2007) discuss how multiagent learning can be applied in game theory. Curry et al (2022) look into how multiagent RL can be used in a real business cycle model with heterogenous agents categorized as 100 consumer-workers, 10 firms and a government. Hernandez-Leal et al (2020) provide a survey of multiagent deep RL and its critiques.

C. Recent Applications

Recent applications of deep reinforcement algorithms range from game playing to robotics and from finance to healthcare and biology.

The DeepMind's AlphaGo program, which defeated a professional Go player in 2016, used a combination of Deep Learning and Monte Carlo Tree Search. DeepMind's Deep Q-Networks (DQN) algorithm, an early version of which was used by AlphaGo, has been used to train artificial agents to play a wide variety of video games. DQN achieves human-level performance across many of these games. The advancement of deep RL algorithms does not only mark a major advancement in the field of computer science, but also has significant implications in other disciplines and in the real world. For example, the AI system AlphaFold[5] predicts 3D protein structure, which accelerates the field of biology. Moreover, inspired by psychology and neural science, AI algorithms also help inform and better understand neural structure of human brains and the psychology of decision making. Relatedly, in the field of behavioral marketing, deep RL algorithms have helped to better understand consumer preferences and find the optimal time to place marketing ads. Personalized recommendations systems have benefited from DRL (Cheremukhin, 2018). In finance, deep RL algorithms have been used to improve trading strategies (Pendharkar & Cusatis, 2018). In the medical field, RL algorithms have helped to develop agents for automated diagnosis and personalized therapy (Zhu et al., 2018). DRL AI algorithms also help minimize energy consumption in the large autonomous data [6] at Google. This AI system helps identify actions that minimize energy consumption while ensure safety constraints are satisfied. The identified set of actions are then sent back to the data center and being implemented. WaveNet is another real-world use of an AI algorithm, and it improves the speech of Google Assistant on Android devices and makes sound more natural[7].

The applications of DRL algorithms in economics and more specifically macroeconomics are fast growing and range from simple single-agent optimization to complex multi-agent settings, which will be discussed in detail in the following sections.

II. Economic Deep Reinforcement Learning: Applications and Emerging Trends in Macroeconomics

RL and DRL algorithms are relatively new to economics, however economists have been using comparable models for much longer. In Arthur (1991) or Barto and Singh (1991)—two reviews of reinforcement learning techniques in computational economics published thirty years ago—it is possible to find an old economic framework very similar to the one used in reinforcement learning (see, for example, the seminal thesis Hellwig (1973)), (in Charpentier et al., 2020). Hughes (2014) also analyzed the applicability of reinforcement learning to specific economic topics.

Application of deep RL algorithms in economics has seen some traction in recent years. Charpentier et al (2020) and Mosavi et al (2020), reviewing the application of deep and reinforcement learning in finance and economics, have cited some recent use cases (including risk analysis, portfolio management, insurance, pricing and bidding optimization). Deep RL has also been applied to some topics in game theory (see Nowe et al., 2012 and Rajeswaran et al., 2020), and mechanism design (Zhan et al., 2020 and Li et al., 2020).

Applications of deep RL algorithms in macroeconomics literature have focused on two areas. First, use of the algorithms as a solution method to find optimal policy or policy response function, such as Hinterlang and

[5] See https://www.deepmind.com/research/highlighted-research/alphafold

[6] See https://www.deepmind.com/blog/safety-first-ai-for-autonomous-data-centre-cooling-and-industrial-control

[7] See https://www.deepmind.com/blog/wavenet-launches-in-the-google-assistant

Tänzer (2022) and Covarrubias (2022). This area also extends to solving general equilibrium models, such as Chen et al (2021), Hill et al (2021), and Curry et al (2022). Moreover, Chen et al (2021) also study the learnability of rational expectation solutions in a general equilibrium model with multiple equilibria. The second area of focus is on modelling bounded rationality and transition dynamics, such as Shi (2021a) and Shi (2021b). It treats households or consumers as boundedly rational, and studies agents' different behaviors facing shocks in the economy. An in-depth discussion of these two types of DRL applications is provided in this section.

A. Finding Optimal Policies

Hinterlang and Tänzer (2022) make use of deep RL to find optimal interest rate reaction function that fulfils the inflation and output gap targets of central banks. They first use ANN to estimate transitions equations of an economy that is qualified by the three equation New Keynesian model in Rotemberg and Woodford (1997). In particular, they estimate the aggregate demand equation (i.e., the dynamic investment-saving curve), and the aggregate supply equation. They then use the estimated transition equations as a representation of the RL environment and train the central bank RL agent to find the optimal interest rate reaction function.

They assume that the transition equations do not change over time while the RL agent learns[1]. The central bank's reward is determined by how much it deviates from the inflation and output targets. They show that the transition equations estimated with ANNs match empirical data better than VAR models, and attribute this to the flexibility of ANNs to approximate non-linear functions. They also show that the optimal reaction function reached through RL outperform common rules used in the literature. They advocate that central bank should add RL algorithms to its toolkit in determining optimal monetary policy reaction functions.

Castro et al (2022) use RL to approximate the policy rules of banks participating in a high-value payments system. The objective of the agents is to learn a policy function for the choice of amount of liquidity provided to the system at the beginning of the day.

They use a real-time gross settlement payments system environment and solve for the best initial liquidity demand of banks through the REINFORCE algorithm. Each day is an episode in their environment. And each day is further divided into T intraday periods. The bank agent starts the day with an initial decision on initial liquidity given the available collateral. The bank then goes through the day by making decisions on sending a payment or not. At the end of the day, the borrowing from central bank for remaining payments can be calculated, and its associated borrowing costs. They make two assumptions of their RL agent. First, agents are risk neutral and will choose their initial liquidity to minimize the total liquidity cost. Second, we assume that the payment demand profile is common knowledge. They train two RL agents concurrently in the same environment, and they make independent actions that are not related to each other. For each training episode, they simultaneously generate multiple trajectories using each agent's current policy, where each trajectory is a complete path of state, action and reward.

Using the experience gathered through multiple trajectories, they update the policy function at the end of each episode. Individual choices have complex strategic effects precluding a closed form solution of the optimal policy, except in simple cases. They show that in a simplified two-agent setting, agents using RL learn the optimal policy that minimises the cost of processing their individual payments. They also illustrate that in more complex settings, both agents learn to reduce their liquidity costs.

Curry et al (2022) illustrate how deep RL with multiple agents can help discover stable solution that are ϵ-Nash equilibria for a meta-game over agent types. They apply this multi-agent deep RL system in real-business cycle models, with 100 worker-consumers, 10 firms, and one government who taxes and redistributes. The learnt meta-game ϵ-Nash equilibria are evaluated through approximate best-response analyses. They also show that the resulting learnt policies align with economic intuitions.

[1] The ANNs used to estimate transition equations should be updated regularly with latest data to take into account the possible behavioral and structural changes in the economy.

They base their research on three main issues faced with existing solution methods of dynamic general equilibrium models. First, linearization is normally required to solve dynamic general equilibrium models through Taylor expansion. However, this approach might be misleading and provide undesirable properties (when nominal interest rate is zero) as argued by Atolia et al (2010) and Boneva et al (2016). Second, formal solutions methods, such as backwards induction and dynamic programming methods, often cannot be solved explicitly and only characterize optimal policies implicitly. Third, curse of dimensionality is an often-encountered problem meaning that the number of states and actions increases exponentially as the number of dimensions increases. This occurs especially for models with a large number of agents. Their incentives are usually misaligned and pose a complex general-sum game, and it remains a challenge to select equilibria in general-sum games (Bai et al 2021).

One common challenge of multi-agent RL setting is that each agent faces a non-stationary environment. Agents are interdependent. Their actions will have an impact on other agents. The second challenge is that economic agents typically have private information that is not observed by other agents. Therefore, Curry et al (2022) states that joint learning can be very unstable in deep multiple agent RL, and moreover, individually trained policies easily get stuck in trivial equilibria where no labor or production occurs.

They develop multi-agent RL techniques that can find equilibria in both a closed and an open real business cycle models. The open RBC model involves a price-taking export market. The ϵ-Nash equilibria for the meta game over agent types are defined as a set of agent policies such that no agent type can unilaterally improve its reward by more than ϵ. They also show how different factors contribute to which equilibria can be found. These factors include world dynamics, initial conditions and policies, learning algorithm and lastly policy model class.

The RBC model application is abstracted in terms of a normal-form meta game between three players representing all the agent types, i.e., the consumer-workers, the firm, and the government. Their policy evaluation, i.e., if a policy is ϵ-Nash equilibrium for the meta game, is defined as follows. They train each agent type separately for a significant number of RL iterations and holding the policies of other agent types fixed. This is to observe if the training improves the agent's reward by more than ϵ. If it does not, the policy is treated at least a local equilibrium. They also discover that a meta-game best response, as learnt in their setup, may not be the best response for an individual player. There may also be free-riders that benefit from the collective performance while not putting in any effort themselves.

Johnson et al (2022) study multiagent RL in a microeconomic environment. RL agents learn to produce resources in a spatially complex world, trade them with one another, and consume those that they prefer. They show that the emergent production, consumption and pricing behaviours can be explained by the supply and demand shifts. They find that when price disparities emerge, some agents learn to take arbitrage opportunities for profit. Their work is part of a research program that aims to build human-like artificial general intelligence through multi-agent interactions in simulated societies. Their model incorporates heterogeneous tastes and physical abilities, and agents negotiate with one another as a grounded form of communication.

Other similar applications include, for example, Jirnyi and Lepetyuk (2011). They solve an incomplete market model with liquidity constraints through RL. Zheng et al (2020) study optimal tax policy in a gather-and-build economy using RL algorithm. Covarrubias (2022) studies the impact of oligopolistic competition on the transmission of monetary policy. He aims to solve for optimal strategies of firms using deep RL algorithms. Calvano et al. (2020) examine the likelihood that the pricing algorithms of multiple sellers trained with RL could result in collusion without the use of any coordination, a potentially challenging problem for competition policy. Igami (2020) examines the similarities and differences between structural estimation (as understood in econometrics) and dynamic programming and RL in the context of two-player, perfect-information, zero-sum games like chess and Go.

B. Bounded Rationality, Learning and Convergence

Learnability and convergence of the algorithm is studied by Chen et al (2021), adopting a monetary model with a representative agent that exhibits multiple equilibria. The representative agent learns in the economy with the DDPG algorithm. They show that the RL agent can locally converge to all the steady states described by the monetary model.

Learning behaviours of RL agents, and especially the exploration feature of RL algorithms are studied in Shi (2021a) and Shi (2021b). Shi (2021) studies the consumption-saving behaviors of RL agents in a stochastic growth environment. In particular, she looks into the discrepancies in learning behaviors when RL agents are different in terms of their exploration levels, and how this impact convergence of optimal policy. She also observes learning behaviors during both a transitory and a permanent income shock.

Shi (2021b) adopts the DDPG algorithm in a representative agent model with transaction cost of money demand. This environment is subject to monetary policy regime changes. Through simulations, she shows that different exploration leads to different beliefs of learning agents, which results in welfare distinctions. She also uses the experience and memory feature of RL algorithms and learns the impact of experience on beliefs and decisions. She finds that an agent who experienced unforeseen regime shifts adjusts better (in terms of rewards) than an agent who has never experienced a similar structural change. This is consistent to the empirical evidence that expectations are affected by past experience of agents (Malmendier and Nagel, 2016).

Chen et al (2021), Shi (2021a, 2021b) all study learning in the case of a single representative household. Hill et al (2021) and Curry et al (2022) look into cases of general equilibrium models with multiple learning agent. Hill et al (2021) show how to solve three rational expectations equilibrium models with discrete heterogenous agents instead of a continuum of agents or a single representative agent. These models are precautionary saving model; the interaction between a pandemic and the macroeconomy (an ‘epi-macro’ model), with stochasticity in health statuses; and a macroeconomic model which has global stochasticity, i.e., where the background is changing in a way that the agents are unable to predict.

III. Deep Reinforcement Learning in Macroeconomics: Prospects and Issues

Although deep RL algorithms have been applied in macroeconomics in recent years, they have been utilized in a limited scope. However, considering the potential of deep RL to provide solutions to high-dimensional complex problems, it seems reasonable to expect its usage in macroeconomics to exponentially grow in the coming years.

There are many areas that could be explored by taking inspirations from RL algorithms. This section discusses the protentional and some prospects of deep RL in the field of macroeconomics, provides instances of how the existing work at the intersection of DRL and macroeconomics could be extended, and offers a few suggestions for future research.

Subsequently, we will highlight some issues and challenges in the application of DRL in macroeconomic settings. This will include a discussion of the issues related to training and choice of algorithms, computational scaling of DRL models, the identifiability of heterogeneous agent models, the robustness of learning to changes in the environment, difficulty in modeling stochastic environments, and the curse of dimensionality. Other issues related to bias and inference also will be discussed in brief.

A. Prospects

The use of deep RL algorithms in macroeconomics has so far been restricted. But there are numerous topics that might be researched and further developed using RL algorithms. These include but are not limited to the following:

Forecasting: Forecasting is likely to be an area of interest to apply deep RL algorithms. ANNs have been used extensively in the past to forecast macroeconomic variables. Applications of RL in forecasting can also be further explored. Research has been done to predict energy usage (Liu et al., 2020) and stock trading decisions (Li et al., 2020). Most macro models consist of micro building blocks, and actors interact with each other. Could the deep RL algorithm, thus, be used to also predict strategic behaviors? This is linked to game theory and can be further explored in the context of economic policy making.

Macro simulation: RL can be used to simulate economies with many economic agents who can interact with each other. This is similar to Curry et al (2022). In such simulation exercises, agents' behaviors in terms of cooperative and competitive can be studied. Moreover, there is a theoretical chance that RL agents could, through learning, share information with each other in order to attain better welfare. Agent-based models (ABMs) and RL can be looked at together. Sert et al (2020) study social segregation behaviors through ABMs when agents are modelled with deep Q learning. The combination of RL and ABMs can provide an artificial environment for policy makers to observe private agents' behaviors and policy impacts.[2]

Rational expectations: RL algorithm can contribute to the understanding of learnability and equilibria-selection in rational expectation general equilibrium models. Chen et al (2021) study the local convergence properties of equilibria. Is there global convergence in models with RL agents? This has further policy implications. Focusing on one equilibrium and studying its local properties and policy response may thus not be sufficient. This is especially important given the increasing current aggregate uncertainties.

Transfer learning to model herding behaviour, spillovers or technology transfer: RL is closely related to transfer learning literature (for a survey see Zhu et al., 2020), which transfers knowledge from external expertise to increase efficiency of the learning process. This is closely related to learning in real life when agents take advice from others. Natural language processing can be a useful resource to collect knowledge and expert opinions. In macroeconomic models, many studies focus on a few countries or regions. Deep learning models are often data hungry. Transfer learning can be used here. That is, deep learning models trained on data from one country can be applied to another. In this way, some data requirements may be alleviated by the application of reinforcement learning. For example, a DRL model trained on data from region-I can be applied to predict economic recession in region-II based on the agents-environment interaction learned from the US data.

Meta learning: Sutton and Barto (2018) pointed out a challenge - representation learning/meta-learning. The main question was how we can use experience not just to learn about a given problem, but to learn inductive biases to benefit future learnings. Zhang et al. (2022) proposed a DRL strategy based on meta-learning. Through meta-learning, a meta-model is initially trained. A few update steps were used to fine-tune the meta-model in order to create submodels for the associated subproblems. The Pareto front is then constructed in accordance. In comparison to existing learning-based approaches, their method may significantly reduce the training time of numerous submodels. Due to the speed and flexibility of the meta-model, new submodels may be created to improve the quality and diversity of the solutions. Meta learning could be useful in specific economic planning settings where a number of subproblems need to be solved associated with a more general optimization problem. It also could be useful in cases where many independent tasks need to be learned in a short time, or the number of tasks can be increased over time, such as in the case of a changing environment.

Automatic hyperparameter optimization and automated model selection and construction: For any given macroeconomic problem, there may be a range of different models that can be applied. Automatic model selection methods that use DRL may be able to address this issue. Shang et al., (2019) proposes an automatic

[2] Song et al (2021) also calibrate an ABM with RL algorithm.

model selection framework based on reinforcement learning. The framework can be divided into two phases, one is data preprocessing stage, which uses meta-learning to process; the other is model selection framework, including feature preprocessing, feature selection and model selection. These aspects are the environment of reinforcement learning, through which a model with the highest accuracy can be chosen as the predictive model output.

Deep learning models can be used to automatically search (even brute force) and tune the hyperparameters of a neural network in order to optimize its performance (Agrawal, T., 2021). This could potentially be useful in situations where it is difficult or expensive to do manual tuning. Similarly, specific RL models could be designed to efficiently search or brute force the model space by trying different macro models on a changing environment or by adding and removing components and variables from models, keeping track of their performance, and automatically or semiautomatically selecting or constructing the most successful ones.

Barriga Rodriguez et al., (2018) present a prototype tool for automatic model repairing by using RL algorithms. They show that RL algorithms could potentially reach model repairing with human-quality without requiring supervision. Laredo et al., (2019) propose a modified micro-genetic deep learning algorithm that automatically and efficiently finds the most suitable neural network model for a given task. Zhu and Yuan (2007) use RL to automate recovery policy generation. In this sense, there’s a high chance that RL could also be used for the calibration of macroeconomic models.

Quantum reinforcement learning (QRL). Quantum reinforcement learning is a relatively new field of study that uses quantum computing to allow RL algorithms to learn faster and more efficiently. Combination of RL with quantum computing (see Dunjko et al., (2017); Wu et al., (2020); Wittek, P. (2014); Biamonte et al., (2017); Zhang & Ni (2020)) will help *automated macro modeling* (see above) and drastically improve this class of RL algorithms. Even if DRL models are currently constrained in terms of processing resources, it might not be the case in the coming years due to the advances in quantum computing (e.g., see Madsen et al., 2022.)

Super integrated policy frameworks: Macroeconomic models can tend to prescribe policies that are one-sided, or that focus on one particular goal. Relatedly, macroeconomic models are typically complicated and detailed in one area but oversimplified in the other sectors. Macromodels also could suffer from Lucas Critique that patterns derived from historical relationship are assumed to permanent – not capturing deeper underlying relationships. Modern macro models attempt to be microfounded and rely on the so-called "deep parameters" in order to circumvent the criticism. However, there is a significant chance that micro-foundations and utility functions in the models are not immune from Lucas Critique because even these deep parameters are not independent of the environments and institutional settings that themselves may be modified and influenced by policies or shocks in the models (Second round Lucan Critique). RL approach to macro modeling could improve our understanding of what macroeconomic models can do as well as their role in the policy-making by tackling above mentioned problems. First, in the RL models, the environment is not an exogenous object that is known to the model agents (or even model developers), but it is an object that is actively seen and learned by the agent, as well as something which could be modified by the agent. Second, RL models could bring an epistemic reform to macro understanding, taking into account not only goal-oriented or restrictive assumptions, but all factors that can influence individual economic decisions. Third, in addition to the previous two, there is no theoretical limitation on the agents' perception, except computational power, hence RL models could be highly granulated both in time and space and might serve as the foundation for future hyper interconnected and super integrated policy frameworks.

Interdisciplinary Integration (Towards Social science-integrated policy frameworks): RL models for macroeconomic studies, coupled with significant developments in natural language processing, could create a ground for a new generation of integrated policy frameworks in which diverse factors from other disciplines could be introduced and studied as the scenario, and agent profiles in an economic model could be more complex, sophisticated and agents' perception more inclusive. This evolution might even lead to a horizon shift and turn the discipline of macroeconomics from the science of gross national product (GNP) and gross domestic product (GDP)— a reduced and one-dimensional understanding of human decision making—into the macro manifest of a broader event, and the science of human welfare, which is a more inclusive, deep and holistic view of human behavior and wellbeing more akin to the social science of a society. The border between different branches of social science could be increasingly blurred, and ultimately, AI's push to use multiple disciplines to study social behavior could lead to a unified, integrated social science (and a likely integration-plus stage afterward incorporating human and natural sciences), instead of currently near-separate islands of social studies. Integrated social science frameworks could help us to better identify which policies are more effective in different social, institutional, cultural, religious, and political contexts or even to develop new policies that are tailored to the specific needs of a particular community or region, considering various contextual factors.

Combined NLP-DRL approaches: Recent advancements in NLPDRL (also called DRL4NLP) aim to use the strengths of both DRL and NLP for better text understanding. (See Wang et al., 2018. For a survey see Uc-Cetina et al., 2022). This could be useful also for macroeconomic modeling. For example, central banks release minutes of their meetings. This text can be scraped and an understanding of the minutes using DRL4NLP can then be integrated into a macroeconomic model. The same could be done for scraped news articles to predict economic activity. For another example, a Transformer model pretrained on news or search data can be fine-tuned on economic time-series data to improve predictions of economic activity (e.g., predicting changes in the consumer price index using texts produced in the search engine or social media).

Quasi experiments using Causal Deep Reinforcement Learning (CRL & CDRL): Causal RL is a new area of research which is the combination of RL and causal inference (See Gershman, 2017; Dasgupta et al., 2019; Grimbly, 2020; Gasse et al., 2021). In many cases, experiments are not possible to be conducted in macroeconomics or there is no possibility to conduct a counterfactual analysis mainly due to the lack of data. For example, it is not possible to do a controlled experiment to test the counterfactual scenario of a change in a policy on economic activity. RL can be used in this context. For example, one can use an RL algorithm to simulate different scenarios of a fiscal policy and observe the corresponding macroeconomic outcomes, even in the absence of data regarding the hypothetical reality.

B. Issues

Deep reinforcement learning algorithms face a wide range of challenges and issues (See Ding & Dong, 2020; Dulac-Arnold et al., 2020 and Du & Ding, 2021), some of which can be summarized as follows:

Training costs and scalability: Training with a deep RL algorithm requires a long computational time and high computational power. RL algorithms need to generate their own data through agent-environment interactive process, which means it requires a long simulation period to collect sufficient experience to solve a RL problem. A related issue is scalability. Scalability is the ability of an algorithm to scale up to large environments. This is important in RL because many real-world environments are too large to be handled by a single RL agent. However, scalability is often a difficult problem because RL algorithms often require a lot of data and computational resources. This is in part due to ANNs. ANNs work in high dimensional state and action spaces, however it remains to be a slow learning device and requires lengthy training dataset. This means that for incremental online learning settings, ANNs would struggle to learn rapidly. This issue becomes more prominent for multiagent RL problems. Transfer learning is one way to go to improve learning efficiency.

Reward function design: Designing a proper reward function is critical to successful training of a deep RL agent. A well-designed reward function should provide the agent with clear and consistent feedback signal so that it can learn the desired behavior. However, designing a reward function is often a difficult and tedious task. It is easy to design a function that would work for a simple task but would not work for a more complex task. In addition, it is often hard to design a reward function that would work for all agents in a multiagent setting, meaning that some agents may be rewarded for actions that are not beneficial for other agents.

Stability: Deep RL agents often suffer from instabilities during training. This is due to the fact that they are constantly trying to optimize a complex objective function and are often operating in highly stochastic environments. These instabilities can lead to the agent diverging from the optimal policy and never converging to a solution. Similarly, sensitivity to hyperparameters (e.g., batch size, exploration level, episode length) should also be addressed when applying these algorithms. Stability issues could be related to computational aspects of the algorithms as well as the design of the reward function or even the complexity of the environment.

Interpretability and choice of algorithms: Deep RL algorithms use ANNs to approximate policy and value functions. How to unpack and interpret an ANN remains an open question in the literature. There are many deep RL algorithms, and the total number is growing quickly. The choice of a particular algorithm depends on the question at hand, and its environment. This includes the design of the state and the action spaces, the reward function, as well as if one is interested in learning the parameters of an action-value function or a policy function or both.

Ergodicity: Ergodicity in reinforcement learning refers to the property of a Markov decision process that determines whether all states and actions are reachable from the initial state. The challenge of ergodicity is important because it determines whether or not an RL agent can learn from all possible experiences. A process is said to be ergodic if all states and actions are reachable. A process is said to be non-ergodic if there are some states or actions that are not reachable. Ergodicity is important because it determines whether or not an RL agent can learn from all possible experiences. If a process is not ergodic, then the agent may never encounter some states or actions, and thus will never be able to learn about them.

Credit assignment: Credit assignment is the process of determining which actions contribute to the goal and which actions do not. This can be a difficult task in complex environments where there may be many different ways to achieve the goal. In addition, credit assignment may be delayed in time, so that an action that contributes to the goal may not be apparent until after the goal is achieved. This can make it difficult for an RL agent to learn which actions are actually helpful in achieving the goal.

Sparse rewards: Sparse rewards are rewards that are only given occasionally. This can make it difficult for an RL agent to learn, because it may not receive enough feedback to learn the desired behavior.

Long-term planning: Long-term planning is the ability of an RL agent to plan for the future. This is important because it allows the agent to take into account the long-term consequences of its actions. However, long-term planning is often a difficult problem as it requires the agent to have a good understanding of the future and to be able to plan for it.

Bias: Bias is the tendency of an RL algorithm to learn a suboptimal policy. This is often caused by the design of the reward function or the architecture of the RL algorithm. Bias is a difficult problem because it can be hard to detect, and it can lead to suboptimal results.

Fairness: Deep RL agents are trained using a reinforcement learning algorithm, which means that they are constantly trying to maximize some reward function. However, this can lead to behavior that can be labeled as socially unfair, as the agents may prioritize their own rewards over the rewards of other agents. This can be a problem in multiagent settings, where the agents need to cooperate in order to achieve a common goal. Fairness is generally referred to the ability of an agent to balance its own actions with the actions of others. It could also refer to the ability of RL algorithm to treat all agents equally in a multi-agent setting. This is important because it allows us to avoid bias and discrimination. However, fairness is often a difficult problem because it is hard to know in advance how an RL algorithm will treat all agents.

Transparency: Deep RL agents are often opaque, and their decision-making process is difficult to understand. This lack of transparency can be problematic when the agents are deployed in real-world settings where it is important to understand why the agents made the decisions they did. For example, policy makers may want to understand why an economic RL model recommended a particular policy.

Benchmarking: It is difficult to compare the performance of different deep RL algorithms because there is a lack of standardized benchmarks. Deep RL algorithms are often compared on a small number of tasks, which might not be representative of the algorithm's true performance. In addition, the performance of a deep RL algorithm can vary greatly depending on the specific details of the implementation, such as the choice of hyperparameters.

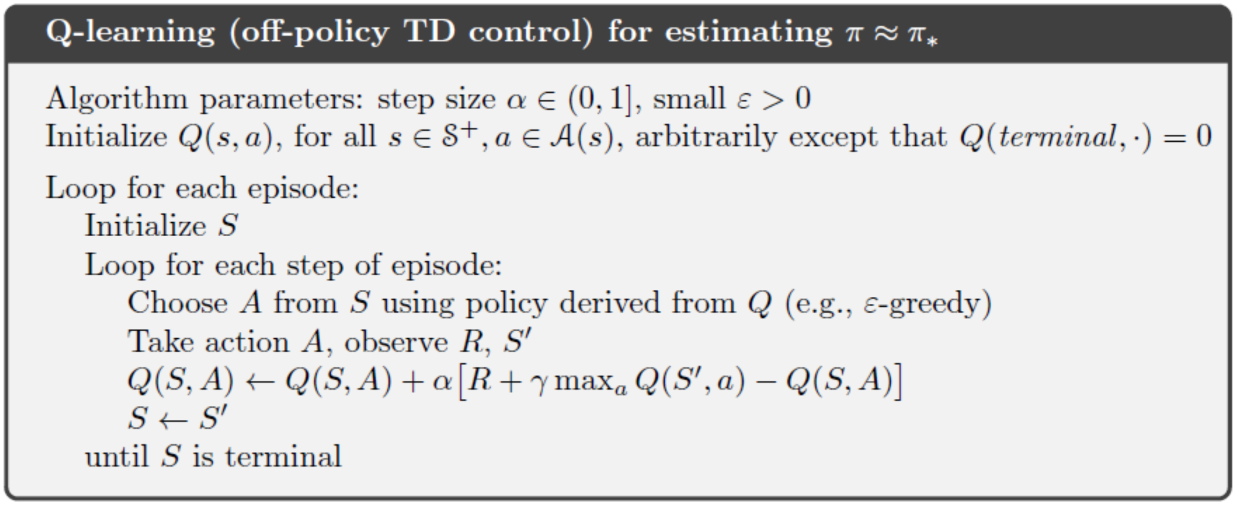
Safety: Safety is the property of an RL agent that ensures that it will not take any actions that would lead to undesirable outcomes. Safety issues could be critical mainly in industrial and health applications of RL, where an RL agent controls a physical action. However, it could also be an issue in other settings where the RL agent's actions or recommendations could have indirect negative consequences for human beings or health (e.g., resource allocation in an economic plan). Safety is often difficult to guarantee because of the stochastic nature of RL algorithms.

IV. Conclusion

The applications of artificial intelligence algorithms, mainly deep reinforcement learning, are growing quickly in economics. In recent years, deep reinforcement learning is used to solve a variety of economic tasks, such as optimal fiscal and monetary policy calibration, game theory, decision making with bounded rationality, and so on. In this paper, with the aim of providing a comprehensive review of the current status of deep reinforcement learning in macroeconomics, we first introduce the basic concepts and methods of deep reinforcement learning, and then go on to look at the recent applications of deep reinforcement learning in macro modeling. We found that the application of DRL models in macroeconomics has focused on two main areas. One area takes advantage of the model-free and flexibility features of deep RL algorithms and use them as solution methods for optimal policies. The second area focuses on adopting RL algorithms to model bounded rationality and it studies issues such as transition dynamics, equilibria learnability and convergence properties. We also give an overview of the potentials of DRL models for the macro models, including forecast improvement, macro simulation, transfer learning, automatic hyperparameter optimization and model construction, interdisciplinary macro models, and causal deep reinforcement learning, Finally, we discuss the challenges of deep reinforcement learning in economics and propose several avenues for future research.

Annex. Full Algorithms

Q-learning algorithm (off-policy) *source: Sutton and Barto (2018, page 131)*

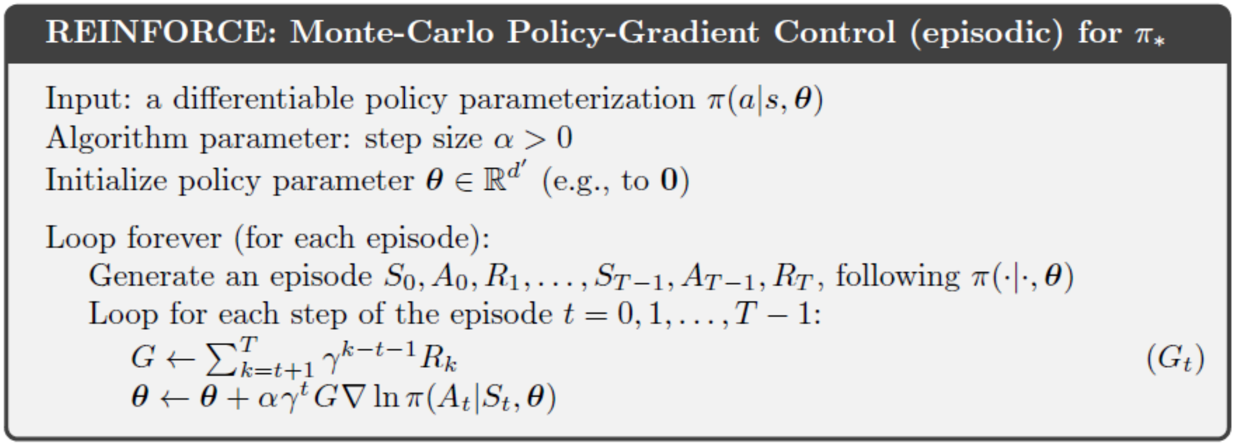


```
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until $S$ is terminal
```

Reinforce: Monte-Carlo *source: Sutton and Barto (2018, page 328)*



```
REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for $\pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                                  $(G_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
```

References

Agrawal, T. (2021). *Hyperparameter Optimization in Machine Learning: Make Your Machine Learning and Deep Learning Models More Efficient.* New York, NY, USA: Apress.

Arthur, W. B. (1991). Designing economic agents that act like human agents: A behavioral approach to bounded rationality. *The American economic review*, 81(2), 353-359.

Athey, S., 2018. The impact of machine learning on economics. In *The economics of artificial intelligence: An agenda* (pp. 507-547). University of Chicago Press.

Atolia, M., Chatterjee, S. and Turnovsky, S.J., 2010. How misleading is linearization? Evaluating the dynamics of the neoclassical growth model. Journal of Economic Dynamics and Control, 34(9), pp.1550-1571.

Bai, Y., Jin, C., Wang, H. and Xiong, C., 2021. Sample-efficient learning of stackelberg equilibria in general-sum games. *Advances in Neural Information Processing Systems*, *34*, pp.25799-25811.

Barriga, A., Rutle, A., & Heldal, R. (2018, October). Automatic model repair using reinforcement learning. In *MoDELS (Workshops)* (pp. 781-786).

Barto, A. G., & Singh, S. P. (1991). On the computational economics of reinforcement learning. In *Connectionist Models* (pp. 35-44). Morgan Kaufmann.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671), 195-202.

Boneva, L.M., Braun, R.A. and Waki, Y., 2016. Some unpleasant properties of loglinearized solutions when the nominal rate is zero. Journal of Monetary Economics, 84, pp.216-232.

Covarrubias, M. (2022), Dynamic Oligopoly and Monetary Policy: A Deep Reinforcement Learning Approach.

Charpentier, A., Elie, R., & Remlinger, C. (2021). Reinforcement learning in economics and finance. *Computational Economics*, 1-38.

Chen, M., Joseph, A., Kumhof, M., Pan, X., Shi, R. and Zhou, X., (2021) Deep reinforcement learning in a monetary model. *arXiv preprint arXiv:2104.09368*.

Cheremukhin, A. (2018). Personalized ad recommendation systems with deep reinforcement learning.

Curry, M., Trott, A., Phade, S., Bai, Y., & Zheng, S. (2022). Finding General Equilibria in Many-Agent Economic Simulations Using Deep Reinforcement Learning. arXiv preprint arXiv:2201.01163.

Dasgupta, I., Wang, J., Chiappa, S., Mitrovic, J., Ortega, P., Raposo, D., ... & Kurth-Nelson, Z. (2019). Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv*:1901.08162.

Ding, Z., & Dong, H. (2020). Challenges of reinforcement learning. In *Deep Reinforcement Learning* (pp. 249-272). Springer, Singapore.

Du, W., & Ding, S. (2021). A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. *Artificial Intelligence Review*, 54(5), 3215-3238.

Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S. and Hester, T., 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, *110*(9), pp.2419-2468.

Dunjko, V., Taylor, J. M., & Briegel, H. J. (2017, October). Advances in quantum reinforcement learning. In 2017 *IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 282-287). IEEE.

Fudenberg, D. and Levine, D.K., 2007. An economist's perspective on multi-agent learning. *Artificial intelligence*, *171*(7), pp.378-381.

Gasse, M., Grasset, D., Gaudron, G., & Oudeyer, P. Y. (2021). Causal reinforcement learning using observational and interventional data. *arXiv preprint arXiv:*2106.14421.

Gershman, S. J. (2017). Reinforcement learning and causal models. T*he Oxford handbook of causal reasoning,* 295.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep learning*. MIT press. https://www.deeplearningbook.org

Grimbly, St John. “Causal Reinforcement Learning: A Primer | Asking Why.” Asking Why., stjohngrimbly.com, 9 Dec. 2020, https://stjohngrimbly.com/causal-reinforcement-learning/.

Hernandez-Leal, P., Kartal, B. and Taylor, M.E., 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, *33*(6), pp.750-797.

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

Hinterlang, N., & Tänzer, A. (2021). Optimal monetary policy using reinforcement learning.

Hughes, N., 2014, November. Applying reinforcement learning to economic problems. In ANU Crawford Phd Conference.

Laredo, D., Qin, Y., Schütze, O., & Sun, J. Q. (2019). Automatic model selection for neural networks. *arXiv preprint arXiv:1905.06010.*Li, B., Xie, K., Huang, X., Wu, Y., & Xie, S. (2022). Deep Reinforcement Learning based Incentive Mechanism Design for Platoon Autonomous Driving with Social Effect. *IEEE Transactions on Vehicular Technology.*

Li, Y., Ni, P. and Chang, V., 2020. Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, *102*(6), pp.1305-1322.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, T., Tan, Z., Xu, C., Chen, H. and Li, Z., 2020. Study on deep reinforcement learning techniques for building energy consumption forecasting. *Energy and Buildings*, *208*, p.109675.

Madsen, L. S., Laudenbach, F., Askarani, M. F., Rortais, F., Vincent, T., Bulmer, J. F., ... & Lavoie, J. (2022). Quantum computational advantage with a programmable photonic processor. *Nature*, 606(7912), 75-81.

Malmendier, U. and Nagel, S., 2016. Learning from inflation experiences. *The Quarterly Journal of Economics*, *131*(1), pp.53-87.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mosavi, A., Faghan, Y., Ghamisi, P., Duan, P., Ardabili, S. F., Salwana, E., & Band, S. S. (2020). Comprehensive review of deep reinforcement learning methods and applications in economics. *Mathematics*, 8(10), 1640.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R.S., Abbeel, P., Levine, S. and Finn, C., 2018. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. arXiv preprint arXiv:1803.11347.

Nguyen, T.T., Nguyen, N.D. and Nahavandi, S., 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, *50*(9), pp.3826-3839.

Nowé, A., Vrancx, P. and Hauwere, Y.M.D., 2012. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning* (pp. 441-470). Springer, Berlin, Heidelberg.

Pendharkar, P. C., & Cusatis, P. (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, 103, 1-13.

Rajeswaran, A., Mordatch, I., & Kumar, V. (2020). A game theoretic framework for model based reinforcement learning. In *International conference on machine learning* (pp. 7953-7963). PMLR.

Sert, E., Bar-Yam, Y. and Morales, A.J., 2020. Segregation dynamics with reinforcement learning and agent based modeling. *Scientific reports*, *10*(1), pp.1-12.

Shang, D., Sun, H., & Zeng, Q. (2020, February). A Reinforcement-Algorithm Framework for Automatic Model Selection. In *IOP Conference Series: Earth and Environmental Science (Vol. 440, No. 2, p. 022060).* IOP Publishing.

Shi, R.A., (2021a) Learning from Zero: How to Make Consumption-Saving Decisions in a Stochastic Environment with an AI Algorithm. *arXiv preprint arXiv:2105.10099*.

Shi, R. A., (2021b) Can an AI agent hit a moving target?. *arXiv preprint arXiv:2110.02474*.

Song, B., Xiong, G., Yu, S., Ye, P., Dong, X. and Lv, Y., 2021, July. Calibration of agent-based model using reinforcement learning. In *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)* (pp. 278-281). IEEE.

Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.

Trott, A., Srinivasa, S., van der Wal, D., Haneuse, S. and Zheng, S., 2021. Building a foundation for data-driven, interpretable, and robust policy design using the ai economist. *arXiv preprint arXiv:2108.02904*.

Uc-Cetina, V., Navarro-Guerrero, N., Martin-Gonzalez, A., Weber, C., & Wermter, S. (2022). Survey on reinforcement learning for language processing. *Artificial Intelligence Review*, 1-33.

Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D. and Kavukcuoglu, K., 2017, July. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning* (pp. 3540-3549). PMLR.

Wang, W. Y., Li, J., & He, X. (2018, July). Deep reinforcement learning for NLP. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts* (pp. 19-21).

Wittek, P. (2014). Quantum machine learning: what quantum computing means to data mining. Academic Press.

Wu, S., Jin, S., Wen, D., & Wang, X. (2020). Quantum reinforcement learning in continuous action space. *arXiv preprint arXiv:2012.10711.*

Zhan, Y., & Zhang, J. (2020, July). An incentive mechanism design for efficient edge learning by deep reinforcement learning approach. In IEEE INFOCOM 2020-*IEEE Conference on Computer Communications* (pp. 2489-2498). IEEE.

Zhang, Y., & Ni, Q. (2020). Recent advances in quantum machine learning. *Quantum Engineering,* 2(1), e34.

Zhang, Z., Wu, Z., Zhang, H., & Wang, *J. (2022). Meta-Learning-Based Deep Reinforcement Learning for Multiobjective Optimization Problems. IEEE Transactions on Neural Networks and Learning Systems*.

Zheng, S., Trott, A., Srinivasa, S., Naik, N., Gruesbeck, M., Parkes, D.C. and Socher, R., 2020. The ai economist: Improving equality and productivity with ai-driven tax policies. *arXiv preprint arXiv:2004.13332*.

Zhu, F., Liao, P., Zhu, X., Yao, J., & Huang, J. (2018, August). Cohesion-driven Online Actor-Critic Reinforcement Learning for mHealth Intervention. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 482-491).

Zhu, Q., & Yuan, C. (2007, June). A reinforcement learning approach to automatic error recovery. In 37th Annual IEEE/IFIP *International Conference on Dependable Systems and Networks* (DSN'07) (pp. 729-738). IEEE.

Zhu, Z., Lin, K., & Zhou, J. (2020). Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*.

PUBLICATIONS

Deep Reinforcement Learning: Emerging Trends in Macroeconomics and Future Prospects
Working Paper No. **WP/2022/259**