Reinforcement Learning Applications in Macroeconomics

Topic 1: Basic Concepts

Authors: Ivy X. Yang

Department of Economics
University of Miami

January 9, 2025

UNIVERSITY OF MIAMI
MIAMI HERBERT
BUSINESS SCHOOL

Purpose of this note

- Understanding how Reinforcement Learning works. - **Topic 1: Basic Concepts.**
- Introduce classical and frontier algorithms in Reinforcement Learning. - **Topic 2: Frontier Algorithms and Application.**
- Discuss multibandit learning. - **Topic 3: Sampling and Adaptive Learning.**
- Discuss multi-agent learning. - **Topic 4: Multi-agent RL and equilibrium.**
- Propose potential directions in which reinforcement learning can contribute to empirical macroeconomic research.
  -**Topic 5: Shocks in the environment and the dynamic back to equilibrium.**
  -**Topic 6: Sub-optimal Choices and Distortion.**
  -**Topic 7: RL with other Machine Learning Methods for Simulation.**

Roadmap

1. Definitions in RL
    - Environment
    - Action
    - Reward
    - States

2. A Simple Q-learning Example

References

Reference: Sutton and Barto (2018)

- *"Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision making. It is distinguished from other computational approaches by its emphasis on learning by an agent from direct interaction with its environment, without requiring exemplary supervision or complete models of the environment."*

Reference: Dimitri P. Bertsekas (2020)

- *"RL can be viewed as the art and science of sequential decision-making for large and difficult problems, often in the presence of imprecisely known and changing environment conditions."*

Reference: Benjamin Moll (2024)

- *"RL ideas seem to me a promising direction for developing alternative approaches to rational expectations about equilibrium prices in heterogeneous-agent models."*

Basic Concepts

Major concepts in Q-learning:

1. Environment - state variables not for agents, which are usually high-dimensional.
2. Action - choice variables.
3. Reward - utility, payoffs, any objective function economists want to maximize or minimize.
4. States - state variables of the agent.

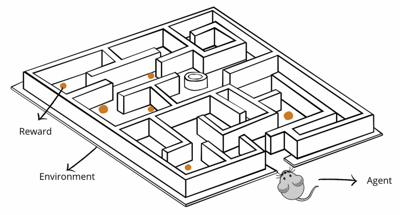
The main difference between Q-learning and value function iteration is how to cope with the transition dynamics. The value function takes the expectation for the future value, but Q-learning updates the quality function with randomization to detect its interaction with the environment.

Basic Concepts

Solving a finite Markov decision problem:

- **Dynamic Programming**
  - requires full information about the environment, so the transitional probability is explicit.
- **Monte Carlo Methods**
  - requires a definitive end of the trial, but do not need an explicit reward function form.
  Not suitable for macroeconomic questions.
- **Temporal Difference Updating** $\star$ - Minsky (1961),Arthur Samuel (1959)

Exploration: RL evaluates the accumulated reward for actions rather than guiding actions with the expected maximized rewards, so it needs to explore possible actions.



Basic Concepts

k-Armed Bandit problem:

In Sutton and Barto's definition, the k-armed bandit is defined as a trigger for k different expected or mean immediate rewards. - In this case, the agent needs to choose!

↓

The action brings in uncertainty from its interaction with the environment.

Suppose that the value at time step t is $Q_t(a^k)$, we do not know what the updated Q-value would be, as there is uncertainty in the environment that we may not know anything or only part of it.

⋆ A single-bandit problem is actually about learning the environment.

⋆ A k-bandit problem is actually about learning the joint impact by choices and the environment.

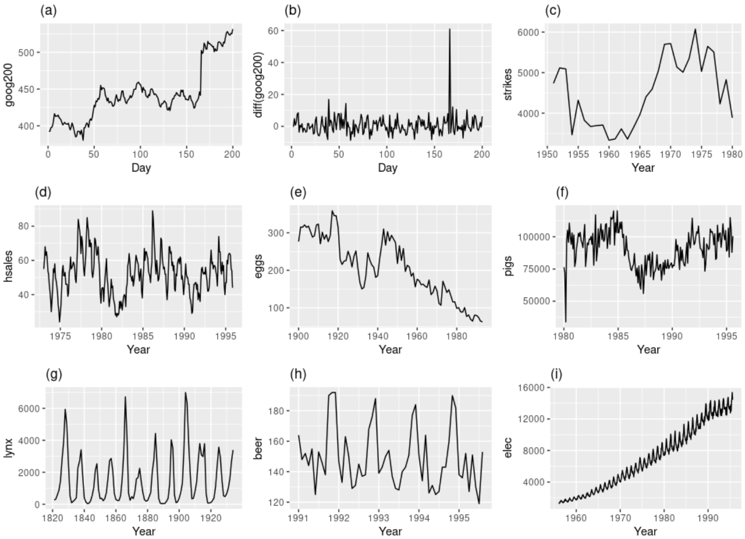
This concept is also developing in experimental economics, see this talk by Dr.Susan Athey: https://www.youtube.com/watch?v=I6GyDWh8kfw

Environment

Macroeconomic shocks and environment
In macroeconomics, we usually care about how an aggregate shock affects agents. This means how the change in the environment affects the reward of agents, and thus distorts their actions and consequently affects their state in the language of RL.

Learn the environment
Agents estimate the rewards of actions and make choices (actions) based on the estimated rewards. The rewards contain information about the unobserved environment, each reward is sent back to the agent as a result of the joint effect of the action they take and the environment.

Environment - Stationary & Nonstationary environment



Value of Actions - Stationary Bandit problem

Estimate the value of action a

- By accumulative rewards:

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

The value of an action depends on the average historical rewards it induced.

- By incremental updating:
  Suppose action a has been selected by n times.

$$\begin{aligned} Q_{n+1}(a) &= \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) = \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + (n-1)Q_n\right) = \frac{1}{n}\left(R_n + nQ_n - Q_n\right) = Q_n(a) + \frac{1}{n}\left[R_n - Q_n(a)\right] \end{aligned}$$

The accumulative reward method is unbiased only if the environment is stationary.

Value of Actions - Non-stationary Bandit problem

Estimate the value of action *a*

- By incremental updating:

$$Q_{n+1}(a) \doteq Q_n(a) + \alpha \left[R_n - Q_n(a) \right]$$

where $\alpha \in (0, 1]$ is a step-size parameter, while the step size is $\frac{1}{n}$ in the stationary environment. α defines how much the agent predicts relies on the most recent reward.

The general idea is to update the quality function is:

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}[\text{Target} - \text{OldEstimate}]$$

Greedy Action

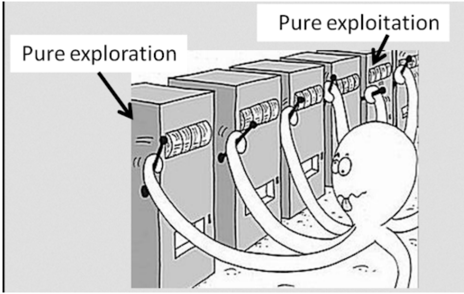
Solving a multi-bandit problem

- Exploitation: Choosing the action that maximized the value function in the past action:

$$A_t \doteq \underset{a}{\operatorname{argmax}} Q_t(a)$$

- Exploration: Choosing a random action to move forward.

ϵ-greedy: define a value $\epsilon \in [0,1]$, such that at each time step, the agent takes an exploitation action with probability ϵ, and takes an exploration action with probability $1 - \epsilon$. - ϵ could die out (shrink with a defined rate) with more iterations are taken.



Greedy Action

Question: Why do we explore?

- get rid of the local optimal solution
- learn more about the reward generated by both the dynamic environment and the actions (or choices in our macroeconomic definition).

What is in the back of this algorithm is the Law of Large Numbers.

- This is why we can use the past average reward to estimate the value of actions when the environment is stationary.

Update Rules

Reference link

Temporal-Difference: $V_t \leftarrow V_t + \alpha * E_t = V_t + \alpha * \left(\underbrace{r_{t+1} + \gamma * V_{t+1}}_{\text{true reward} V^*} - V_t \right)$

where γ is the time discount rate, as β in macro.

- **State Action Reward State Action (SARSA):**

  $$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right)\right]$$

  SARSA is the on-policy rule. - carry what we have for the last term (either exploration or exploitation result forward).

- **Q-learning:**

  $$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right) + \alpha\left[R_{t+1} + \gamma \max_a Q\left(S_{t+1}, a\right) - Q\left(S_t, A_t\right)\right]$$

  Q-learning is the off-policy rule - carry the choice that maximizes the quality function no matter what type (explore or exploit).

Update Rules

- Code for SARSA iteration:

```
    Q[s, a] = Q[s, a] + alpha * (reward - Q[s, a]
            + gamma * np.max([Q[s_next, a_next]
            for a_next in range(action_space)]))
```

- Code for Q-learning iteration:

```
    Q[s, a] = Q[s, a] + alpha * (reward - Q[s, a]
            + gamma * np.max([Q[s_next, a_next]
            for a_next in range(action_space)]))
```

Reference link

Reward function

In economics, we have the natural reward function - the objective function.

But it could be more flexible...

More flexible reward function $\star$

In RL, the reward function has a more broad definition. It is not necessarily the same as the utility that takes risk aversion. It could be a myopic reward, as households have not learned the environment, where risk comes from, enough. So, your risk aversion could be updated as you learn more. So, we can just use consumption at time t c_t as a reward when learning.

This idea takes both the knowledge of the adaptive learning process and the heterogeneity of learning ability, which is attached to behavioral economics.

When the utility function is risk-aversion type, it means the expected utility of any consumptions is less than the utility of the expected consumptions. In RL process, the uncertainty comes from the environment, households form expectations not on utility but on the state.

Reward is a function of state

In the RL, the reward is not necessarily a function of the state. In economics, as we have theoretical-based assumptions and models, we can write the reward of a function of state variables. This is indirect is you have a high-quality data on the state variables in the learning step.

Examples:

- Households' problem:

$$R(c) = u(c) = u((1+r)a - a')$$

Given the budget constraint:

$$a' = a(1+r) - c$$

- Firms' problem:

$$R(k,l) = \pi(k,l) = f(k,l) - (wl + rk)$$

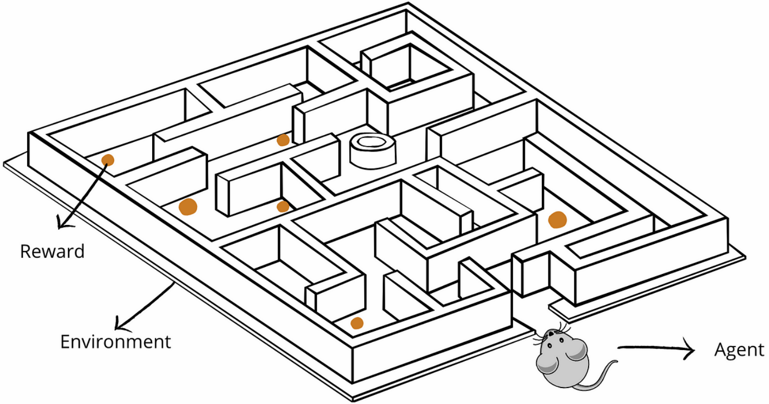
This function is on both state variables and choice variables.

Reward is a function of choice (action)

One virtue of RL is we can get rid of some economic assumptions to simulate. If we relax the assumption of the form of the budget constraints, we can feed the reward function directly as the choice variable (consumption $u(c)$). If we further relax the assumption of the utility function form, the reward could be the choice variable c directly.

States of agents

The states are important for transitional dynamics and are also important in the economic sense. The agent-specific state variable defines the transitional probability. - It may not depict your reward, but it depicts the position of the mouse in the maze, which attach to next step that may or may not have good reward.



States of agents

The transition dynamics are crucial in the VFI; while in the QFI, the convergence could be achieved with the random draw (i.e., by ϵ-greedy).

- **Value function iteration**

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[\sum \gamma^k R_{t+k+1} \mid S_t = s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right], \quad \text{for all } s \in \delta^\infty \end{aligned}$$

- **Quality function iteration**

$$\begin{aligned} q_*(s, a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*\left(S_{t+1}, a'\right) \mid S_t = s, A_t = a\right] \\ &= \sum_{a',r} p\left(s', r \mid s, a\right)\left[r + \gamma \max_{a'} q_*\left(s', a'\right)\right] \end{aligned}$$

Value function and Quality function

$$q_*(s, a) = \mathbb{E}\left[R_{t+1} + v_*\left(S_{t+1}\right) \mid S_t = s, A_t = a\right]$$

States of agents - transitional dynamics

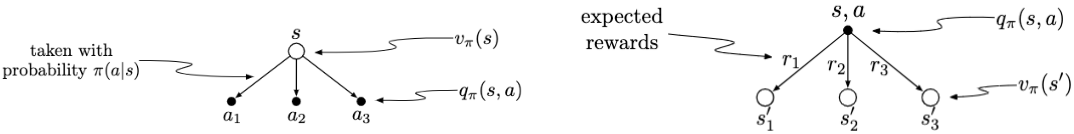


Figure: $v^*(s)$ and $q^*(s,a)$

with the definition of a discrete probability distribution:

$$p\left(s', r \mid s, a\right) \doteq \Pr\left\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\right\}$$

and policy function:

$$\pi(a \mid s)$$

States of agents - transitional dynamics

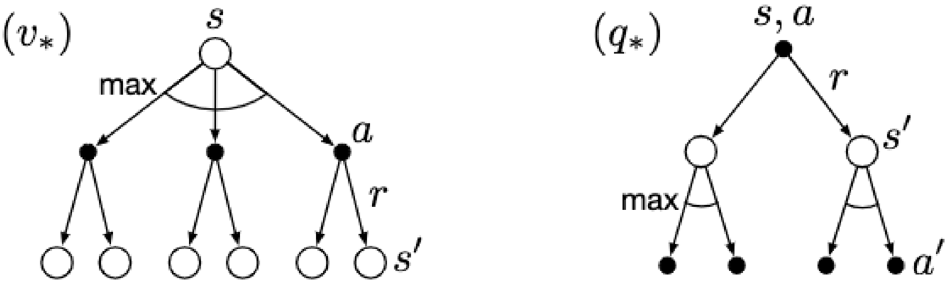


Figure 3.4: Backup diagrams for v_* and q_*

TD-learning: SARSA iteration

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

```
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in S^+$, $a \in A(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$
Loop for each episode:
    • Initialize $S$
    • Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
    • Loop for each step of episode:
        • Take action $A$, observe $R, S'$
        • Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        • $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        • $S \leftarrow S'$; $A \leftarrow A'$
Until $S$ is terminal
```

TD-learning: Q-learning iteration

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

```
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in S^+$, $a \in A(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$
Loop for each episode:
    • Initialize $S$
    • Loop for each step of episode:
        • Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-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
```

The End