An Introduction To Reinforcement Learning Algorithm

Reinforcement learning(RL) primarily was used in electrical engineering in the 1960s, due to the limitation of computer computing power, there are some theoretical algorithms that cannot be implemented to use in industrial production and application. However, people truly know and realize how powerful artificial intelligence is because the AI program AlphaGo beat the human player Lee Sedol who is one of the strongest players in the history of Go.

Figure 1: Documentary from Netflix for AlphaGo vs. Lee Sedol

Moreover, RL is widely applied in autonomous driving(Shalev-Shwartz et al., 2016), game AI (Li et al, 2022), Natural language processing (NLP)(Luketina et al, 2019), and so on.

What is reinforcement learning?

Briefly, RL is about how intelligent agents ought to make sequential actions in an unknown environment through trial and error.

Markov decision process(MDP)

Typically, we can frame all RL tasks as MDPs. By the Markov property, each state is independent of the other states, then we basically state the environment in form of a Markov

decision process.

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^{a}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

Figure 2: Mathematical definition of MDP

In the MDP, an agent at a state from state spaces can play an action from action spaces. Then the system transitions to another state according to an unknown probability, while returning an immediate reward. Although the learning agent does not know the transition probabilities at the beginning, the goal we expected of the agent is to maximize a reward signal return after playing for a while. There will be a following-up question: How does the whole RL system work through those elements?

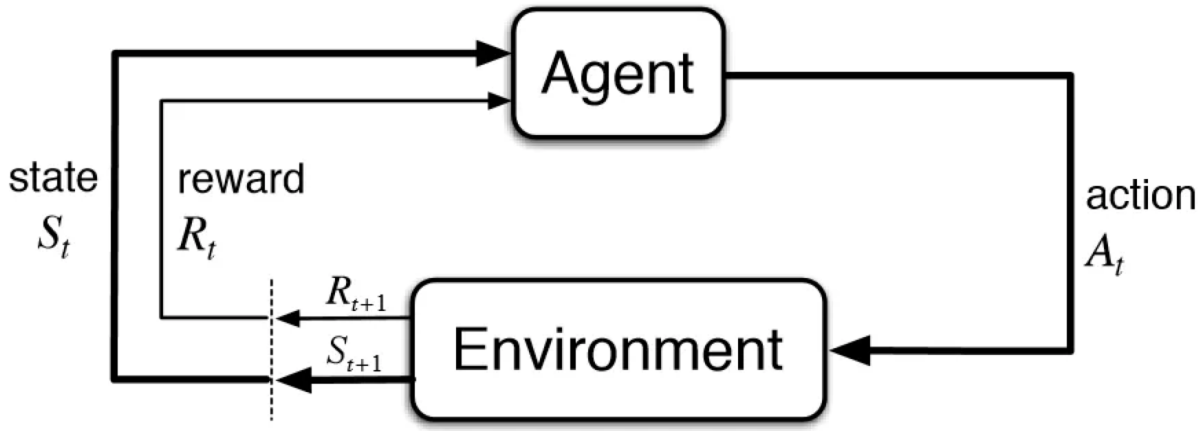

Figure 3: The agent–environment interaction in a Markov decision process.

There are main important components in reinforcement learning from the picture:

1. Agent: the agent is the main player in reinforcement learning and is responsible for making decisions and learning from experience in the environment.
2. Environment: the environment provides the state of the system, defines the actions that the agent can take, and provides a reward signal based on the agent's actions.
3. Reward Signal: the reward signal is the key feedback mechanism in reinforcement learning that allows the agent to learn from experience and improve its policy over time.

There are some key and potential elements that are not shown in the picture.

4. Policy: The policy is the mapping from states to actions that the agent takes. The policy defines the behavior of the agent in the environment and is updated over time as the agent receives feedback in the form of rewards.
5. Value Function: the value function is an important component of reinforcement learning that provides the agent with a way to estimate the expected long-term reward of its actions and to guide its decision-making.
6. Model: Optionally, the environment could be model-free, but different model-based environments have a significant impact on the performance of the reinforcement learning system

With all the components in the RL, that is time to combine them together to solve the tasks in real life by RL algorithms.

Why develop and research reinforcement learning algorithms?

Through the RL topic, scientists and engineers develop a lot of algorithms to solve and improve the hard problem for humans in actual production. This wiki article will talk about the popular and classic RL algorithms as an introduction to RL algorithms.

- policy gradients

  As we all know, the way of gradient descent is broadly applied in kinds of machine learning and deep learning to optimize the performance of algorithms. For RL, researchers develop policy gradients that have been widely used in various topics, such as Atari games(J. Schulman et al. 2015, V. Mnih et al. 2016), Real-world robots(S. Levine, 2015). There is a picture showing how the policy gradients work:

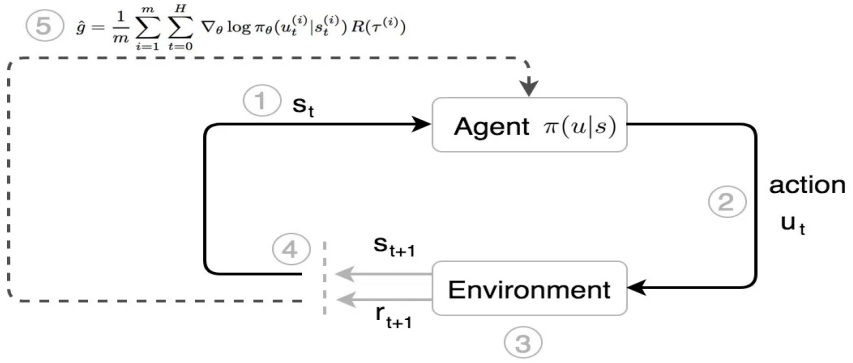

Figure 4: The Policy Gradients objective function.

- Q-learning
- Model-based RL

(will talk more about this in this part)

Challenges in Deep Reinforcement Learning:

We have talked a lot about existing research results in RL algorithms that help humans in production and life. However, there are some problems and challenges with core algorithms in reinforcement learning for us to care for and research in the future.

- Does the RL algorithm converge?

  On the engineering side, Stability is the ability of the algorithm to converge to the optimal policy. The whole convergence progress is affected by hyperparameters. Consequently, tuning hyperparameters in the process to select the optimal values for the RL algorithm's parameters is the key point. However, it is very hard to devise stable RL algorithms since we cannot run hyperparameter sweeps in the real world. People could do more research here and develop more stable algorithms that are less sensitive to hyperparameters, which essentially makes RL a viable tool for real-world problems.

- How long does the RL algorithm take to converge?

  Simply, it depends on the sample complexity. The efficiency of RL algorithms is a big challenge that is worth spending more time researching. For example, the RL algorithms applied in the self-driving system in the practice may need to wait for a long lag to finish running, then make a deal for changes in the environment, if the sample complexity is too high, and it may cause security risks. People need to design faster algorithms and develop better-modeled based RL algorithms in the future.

- After the RL algorithm converges, does it generalize?

  The topic of scaling and generalization is another direction challenge in RL. We need RL algorithms to effectively learn in problems with larger state and action spaces or more

complex environments with the ability to make problems scalable. After that with a larger scale, we hopefully make the RL algorithms more generalization, which could transfer knowledge from one problem to another. Overall, addressing these challenges is crucial for the success of RL algorithms in real-world problems, although the real world is not so simple.

Reference

- J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P.Abbeel. “Trust Region Policy Optimization”. (2015).
- V. Mnih, A. P. Badia, M. Mirza, A. Graves, T.P.Lillicrap, et al. “Asynchronous methods for deep reinforcement learning”. (2016).
- S. Levine*, C. Finn*, T. Darrell, P. Abbeel. “End-to-end training of deep visuomotor policies”. (2015).
- V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I.Antonoglou, et al. “Playing Atari with Deep Reinforcement Learning”. (2013). D. Kalashnikov et al. “QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation”. (2018).
- Li, Quanyi, Zhenghao Peng, Haibin Wu, Lan Feng, and Bolei Zhou. "Human-AI Shared Control via Frequency-based Policy Dissection." arXiv preprint arXiv:2206.00152 (2022).
- Luketina, Jelena, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. "A survey of reinforcement learning informed by natural language." arXiv preprint arXiv:1906.03926 (2019).
- Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.