Introduction to Q-Learning with Gymnasium

What is Reinforcement learning?

Reward based learning!

If it does a good job we give +1 and if it does bad we give it -1

For a given environment agent learns how to interact with it through rewards

Something to do with rewards

Like you train a pet dog and it learns from treats

Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning process that focuses on decision making by autonomous agents. An autonomous agent is any system that can make decisions and act in response to its environment independent of direct instruction by a human user. Robots and self-driving cars are examples of autonomous agents. In reinforcement learning, an autonomous agent learns to perform a task by trial and error in the absence of any guidance from a human user.[1] It particularly addresses sequential decision-making problems in uncertain environments, and shows promise in artificial intelligence development.

Reinforcement learning (RL) is a machine learning (ML) technique that trains software to make decisions to achieve the most optimal results. It mimics the trial-and-error learning process that humans use to achieve their goals. Software actions that work towards your goal are reinforced, while actions that detract from the goal are ignored.

RL algorithms use a reward-and-punishment paradigm as they process data. They learn from the feedback of each action and self-discover the best processing paths to achieve final outcomes. The algorithms are also capable of delayed gratification. The best overall strategy may require short-term sacrifices, so the best approach they discover may include some punishments or backtracking along the way. RL is a powerful method to help artificial intelligence (AI) systems achieve optimal outcomes in unseen environments.

Where is it used?

Google Deepmind's AlphaGo : AlphaGo - Google DeepMind

OpenAI's Hide & seek multi agent : Emergent tool use from multi-agent interaction | OpenAI

RL in sustainable agriculture : [PDF] Reinforcement Learning for Sustainable Agriculture | Semantic Scholar

FinRL : AI4Finance-Foundation/FinRL: FinRL: Financial Reinforcement Learning. 🔥

OpenAI's agent beats 5 dota 2 world champions: OpenAI Five defeats Dota 2 world champions | OpenAI

Deepmind agents plays Atari games : Agent57: Outperforming the human Atari benchmark - Google DeepMind

OpenAI ChatGPT : Introducing ChatGPT | OpenAI

RL in Robotics : Reinforcement Learning in Robotics: A Survey | SpringerLink

Simplified Keywords

1. **State(S):** Information that is available from environment based on which a action can be taken.
2. **Action(A):** something that can cause state change or remain same.
3. **Reward(R):** An event indicating positive or negative state, based on result of combination of A & S
4. **Environment(E):** Problem statement, a entity that provides has a state and a reward associated with it.
5. **Agent(A):** Interacts with Agent through “Actions”, getting a new state and a reward.
6. **Policy:** a sequence of [S, A, R, S’], associating each state with an action increasing long term reward
7. **Model Free:** Agent directly interact with environment to learn a policy
8. **Model based:** Agent has minimal to no interaction with environment but has a model that accurately simulate the model and agent tries to learn through simulations
9. **MDP:** Markov Decision Process, is a mathematical decision framework.
10. **Markov Property:** Decision is based on only current state no historical action, state should impact your current decision.
11. **Exploration vs Exploitation**

Q-Learning

***Q*-learning** is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations.

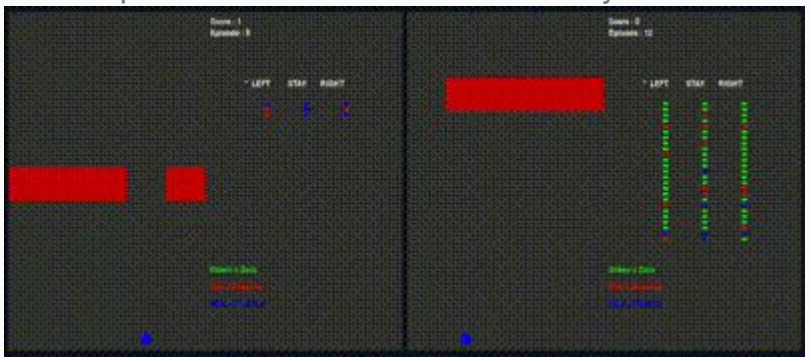
	A0	A1
S0	13.4	10.2
S1	-3.2	5.4
S2	20.0	12.1
S3	1.1	0.9

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

$$\left(\text{The New Action Value} = \text{The Old Value}\right) + \text{The Learning Rate} \times \left(\text{The New Information} - \text{the Old Information}\right)$$

- **Q-Value:** “Quality” of a given state action pair, it is derived from “value state v(s)” from MDP
- **Gamma:** Discount factor for future potential Quality state, simply how much should agent trust the future state
- **Alpha:** Learning rate, how much the new information affect the current Q-value
- **Reward:** reward received after agent took an action ***a(t)*** at ***s(t)***
- **Temporal Difference:** A technique to update Q-values in every step by making small changes to Q value of current state-action pair

Q-value are stored in a 2D matrix where each cell represents a state, action pair’s Quality, agent using Q-formula updates these cell values after every action to update this matrix



In simplest form,

Simple DQN: For continuous state space, we can leverage ANN to create a simple DQN network that tries to approximate Q-values for a given action space.

We store the state, action, reward, next_state information in a replay buffer and apply them at end of each or batch episodes.

Where we try to make network adapt to underlying MDP (or simply put Q distribution of environment) by using Q-Formula.

[Reinforcement Learning (DQN) Tutorial — PyTorch Tutorials 2.5.0+cu124 documentation]

What about non-discrete states, what if there are too many states?

Gymnasium

Thank You