1 - Introduction to Deep Reinforcement Learning

Overview:

- What is Deep Reinforcement Learning, how are rewards the central idea?
- The three approaches of Reinforcement Learning
- What the “Deep” in Deep Reinforcement Learning means

Idea behind RL is that an agent will learn from environment by interacting with it and receiving rewards for performing actions. Essentially, it’s a computational approach of learning from action.

The Reinforcement Learning Process

1. Agent receives a **state S0** from the **Environment** (such as the first frame of a game (state) from the game (environment))
2. Based on that **state S0**, agent takes an **action A0** (such as moving right)
3. Environment transitions to a new state s1 (new frame)
4. Environment gives some **reward R1** to the agent (not dead: +1)

Basically, we have a loop of **state**, **action**, and **reward**

Goal of the agent is to maximize expected cumulative reward. This is written as:

$$G_t = R_{t+1} + R_{t+2} + \ldots$$

Which is equivalent to:

$$G_t = \sum_{k=0}^{T} R_{t+k+1}$$

But there is an issue with simply adding the rewards like that. Rewards that happen sooner (in the beginning of the game) are more probable to happen because they are more predictable than the long term future reward.
To solve this we need to discount rewards.

We define a discount rate called gamma, which must be between 0 and 1.

- The larger the gamma the smaller the discount. This means the learning agent cares more about the long term reward
- The smaller the gamma, the bigger the discount. This means the agent cares more about the short term reward.

So, discounted cumulative expected rewards is

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} where \gamma \in [0, 1)$$

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$$

TL;DR - each reward will be discounted by gamma to the exponent of the time step. As the time step increases, the enemy gets closer to us (in the cat, mouse, cheese example) so the future reward is less and less probable to happen.

Episodic or Continuing Tasks

A task is an instance of a RL problem. There are two types of tasks: episodic and continuous.

Episodic Task
Episodic tasks have a starting and ending point (**terminal state**). This creates an episode: a list of states, actions, rewards, and new states.

Continuing Task
Continuing tasks are tasks that continue forever. They don't have a terminal state. As a result, the agents to learn how to choose the best actions and simultaneously interact with the environment. An example of this is an agent that

does automatic stock trading. For this task there is no starting and terminal state. The agent will keep running until we decide to stop him.

Monte Carlo vs TD Learning Methods

There are two ways of learning:
- Collecting the rewards **at the end of the episode** and then calculating the **maximum expected future reward** (Monte Carlo Approach)
- Estimating **the rewards at each step** (Temporal Difference Learning)

Monte Carlo
When the episode ends (so agent reaches a terminal state), the agent **looks at the total cumulative reward** to see how well it did. In Monte Carlo approach, rewards are only **received at the end of the game**.

Then we start a new game with this added knowledge. The **agent will make better decisions with each iteration.**

$$\underset{\text{Maximum expected future reward starting at that state}}{\underline{V(S_t)}} \leftarrow \underset{\text{Former estimation of maximum expected future reward starting at that state}}{\underline{V(S_t)}} + \underset{\text{learning rate}}{\alpha}[\underset{\text{Discounted cumulative rewards}}{\underline{G_t}} - \underline{V(S_t)}]$$

So in the mouse, cheese, cat game, an entire game is an episode. At the end of the episode we have a list of state, actions, rewards, and new states. The agent will sum the total rewards Gt (to see how well it did). Then it will update V(st) based on the formula above. Then start a new game with this new knowledge.

By running more and more episodes, the agent will learn to play better and better.

Temporal Difference Learning
TD Learning, on the other hand, will not wait until the end of the episode to update its value estimation (the function V) for the nonterminal states St occurring at that experience. This method is called TD(0) or **one step TD**, as it updates the value function V after any individual step.

Monte Carlo $$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

TD Learning $$V(S_t) \leftarrow \underbrace{V(S_t)}_{\text{Previous estimate}} + \alpha[\underbrace{\underbrace{R_{t+1}}_{\text{Reward t+1}} + \underbrace{\gamma V(S_{t+1})}_{\text{Discounted value on the next step}}}_{\text{TD Target}} - \underbrace{V(S_t)}_{}]$$

TD methods only wait until the next time step to update the value estimates. At time t+1 they immediately form a TD target (an estimation) using the observed reward Rt+1 and the current estimate V(St+1)

More on value functions: they are state-action pair functions that estimate how good a particular action will e in a. Given state, or what the return for that action is expected to be.

Exploration/Exploitation trade off

- Exploration is finding more information about the environment
- Exploitation is exploiting known information to maximize the reward.

It’s easy to fall into a trap where known information leads to a local maximum for the cumulative reward, that may be minuscule compared to the cumulative maximum given more info about environment. Thus, there is a tradeoff between exploration and exploitation that we need to define a rule for. There are different ways to handle it.

Three Approaches to Reinforcement Learning

- Value based
- Policy-based
- Model-based

Value Based

In value-based RL, the goal is to optimize the value function V(s) ~ s is a state

The value function is a function that tells us the maximum expected future reward the agent will get at each state.
Value of each state is the total amount of the reward an agent can expect to accumulate over the future, starting at that state.

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right]$$

Expected

Reward discounted

Given that state

The agent will use this value function to select which state to choose at each step. The **agent takes the state with the biggest value**.

Policy Based

In policy-based RL, we directly optimize the policy function pi(s) without using a value function.

The **policy** is what defines the agent behavior at a given time.

$$a = \pi(s)$$

Above equation translates to action = policy(state)

Thus, we learn a policy function so we can map each state to the east corresponding action.

There are two types of policy:

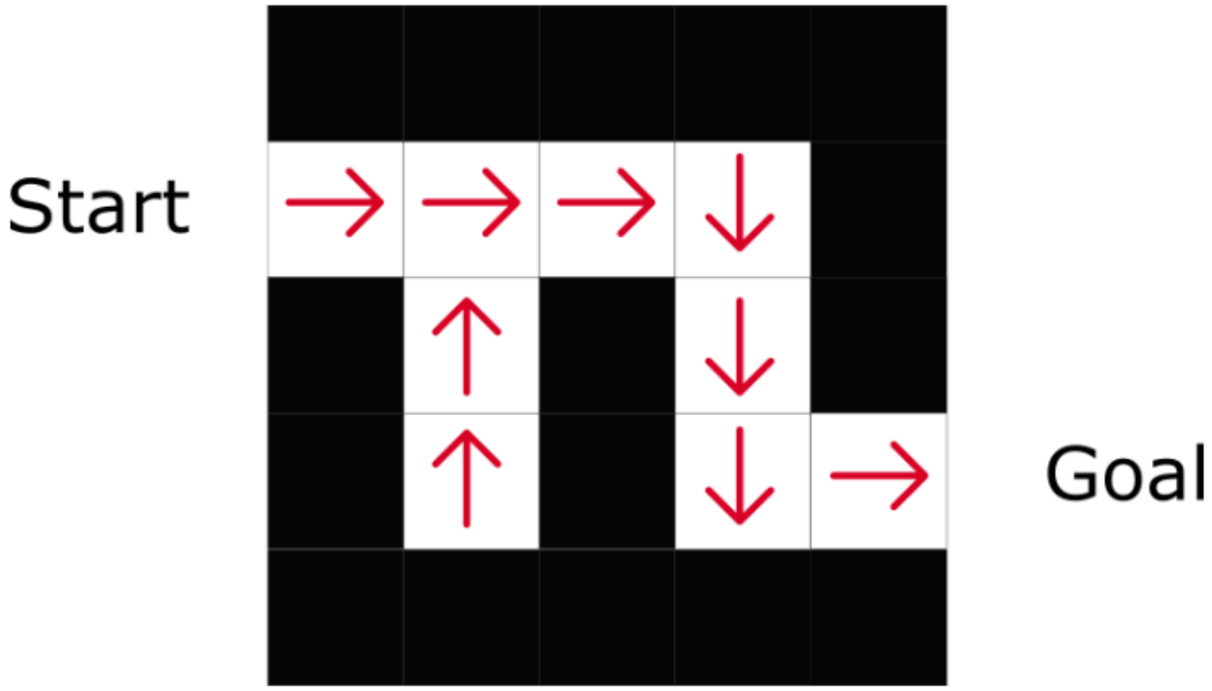
- **Deterministic**: a policy at a given state will always return he same action
- **Stochastic**: output a distribution probability over actions

Stochastic policies can be written like this:

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

This translates to the probability of a certain action given a state s is what that state maps to in the distribution probability over all actions (big P).

In the maze example, the policy directly indicates the best action to take for each step:



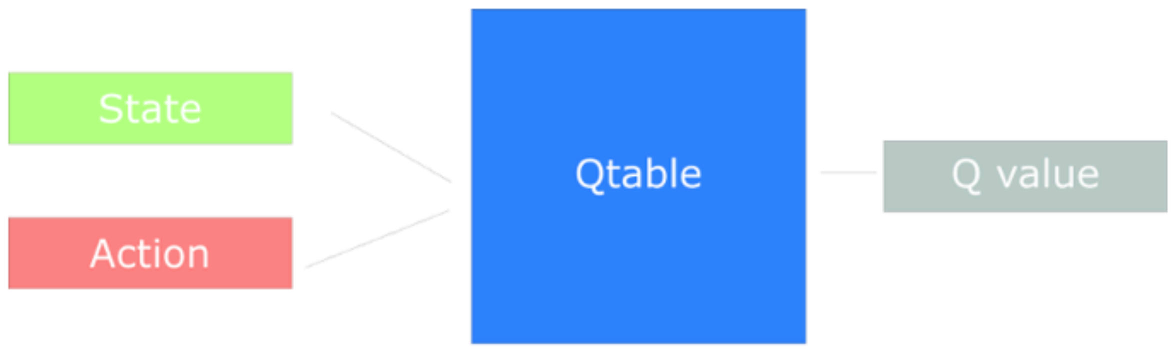
Model Based
The principle of model based RL is modeling the environment. This means that we create a model of the behavior of the environment.

The problem with this is that each environment will need a different model representation. So this is not a big focus.

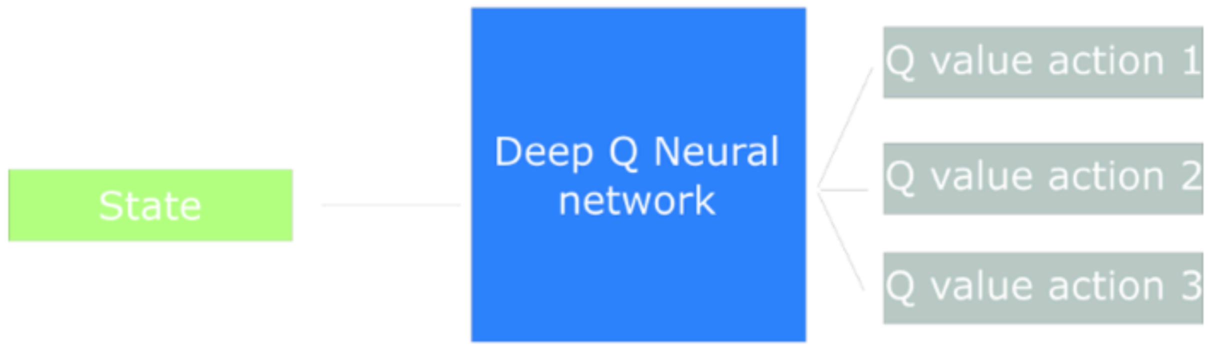
Introducing Deep Reinforcement Learning

“Deep” Reinforcement Learning simply introduces deep neural networks to solve reinforcement learning problems - hence the name “deep”

Q-Learning is classic reinforcement learning, while Deep Q-Learning is RL enhanced with deep neural networks. The difference between these approaches is summarized below:



Q* learning



Deep Q* learning

In the first approach, a traditional algorithm is used to create what is called a “Q Table”. This Q table helps us find what action to take for each state.

In the second approach, a neural network is used to approximate the reward based on state (q value).

—— Learning Legs meeting notes:
Natural gradient vs normal gradient