Reinforcement Learning

A precursor

Where does it fall in the ML sphere?

- Somewhere between supervised and unsupervised learning.
- The algorithm is only informed of whether an approach is correct or not.
- Searching for the correct approach involves trial and error.

The Reinforcement Learning (RL) Framework

Any RL algorithm involves:

- An **agent**: the entity that learns to solve a problem.
- An **environment**: where and what the agent learns.
- A **reward function**: a way to quantify the effectiveness of an agent's strategy.

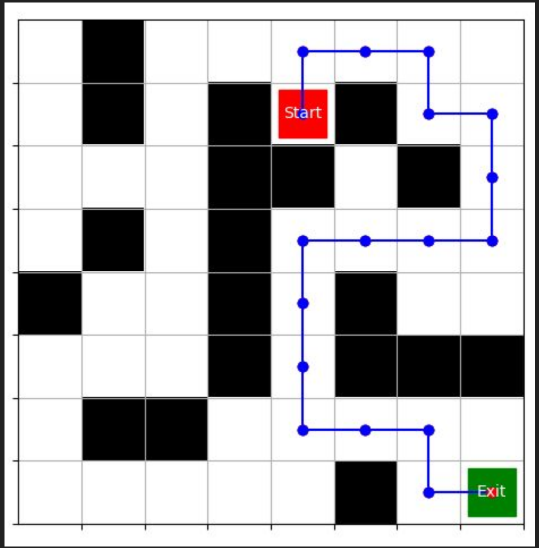

Figure 1: The **agent** must navigate the maze (**environment**) to reach the exit (**reward**)

The Reinforcement Learning (RL) Framework

State and Action Space

- An agent knows about its current input (**state**) and the possible things it can do (**actions**).
- The RL algorithm maps **states** to **actions** in order to maximise the **reward function**.
- The **state** and **action** spaces define the set of all possible states and actions.
- There is a clear motivation in efficiency to reduce these spaces as much as possible without oversimplifying the problem.

The Reinforcement Learning (RL) Framework

State and Action Space

- Often the connection between **action** and **reward** is not instantaneous.
- Algorithms must accommodate this delay by estimating the total **reward** based on a given **action**.

Figure 2: With white as the **agent**, what move (**action**) gives us the best chance of checkmate (final **reward**)?

The Reinforcement Learning (RL) Framework

Reward Function

- Maps a **state** and **action** to a numerical **reward** (positive or negative).
- The **reward** only specifies the goal, not how to achieve it.
- It can be tempting to add **sub-goals** in an attempt to better lead the agent to the total goal or reward.
- One must be careful in the design of **sub-goals** as the agent may converge on a local maxima where only the sub-goals are achieved and not the total goal.

The Reinforcement Learning (RL) Framework

Reward Function

- RL tasks are often **episodic**.
- After reaching a definite endpoint, the algorithm is restarted.
- The agent retains state/action/reward information from previous episodes.
- In this scenario the total reward is clearly defined.

The Reinforcement Learning (RL) Framework

Reward Function

- In contrast to **episodic** tasks, **continual** tasks do not have a definite endpoint.
- The agent must continually learn and improve its performance.
- Here the idea of a total/final **reward** is not clearly defined, making reward prediction more difficult than the previous case.
- How can we unify these two tasks in a single framework?

The Reinforcement Learning (RL) Framework

Discounting

- **Discounting** attempts to unify **episodic** and **continual** tasks by weighting future reward predictions by the confidence in that prediction.
- The less certain a prediction is, the more it should be discounted.
- A simple discounting method is to discount predictions based on their distance into the future.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+(k+1)}$$

The Reinforcement Learning (RL) Framework

Action Selection

- Given the current state the agent will compute a value (the expected reward) for each action.
- A simple way to estimate the reward is to compute the average reward that has been received in the past.
- This is commonly denoted as $Q_{s,t}(a)$, where
  - s - state
  - a - action
  - t - number of times that the action has been taken before in this state

The Reinforcement Learning (RL) Framework

Action Selection

Three common strategies using the average reward prediction $Q_{s,t}(a)$ are,

- **Greedy** - pick the action with the highest $Q_{s,t}(a)$
- **ϵ-greedy** - similar to above but with some small probability ϵ that another action is picked.
- **Soft-max** - A refinement of ϵ-greedy. Uses the soft-max function to decide the next action selection.

The Reinforcement Learning (RL) Framework

Policy

- Instead of using a fixed **action selection** strategy, RL algorithms typically vary the selection strategy.
- This is done by learning a **policy**.
- The optimal **policy** for each **state** will describe the best **action**.
- The agent's **policy** describes the combination of exploration and exploitation.
- Depending on the **state**, **policy** and **reward** the agent can,
  - Exploit a past **action** that performed well, or
  - Explore a new **action** in an attempt to further maximise reward.

The Reinforcement Learning (RL) Framework

Markov Decision Processes

- Tasks can also be separated depending on whether the current optimal **action** is dependent on past **actions** and **states**.
- Sometimes the past is necessary to decide the next **action**.
- In other cases the current **state** provides sufficient information to decide the next **action**.

The Reinforcement Learning (RL) Framework

Markov Decision Processes

- Problems where action selection is independent of the past are **Markov Decision Processes**.
- The state is said to be a **Markov State**.

$$P(r_t = r',\, s_{t+1} = s' \,|s_t,\, a_t)$$

Thanks for listening