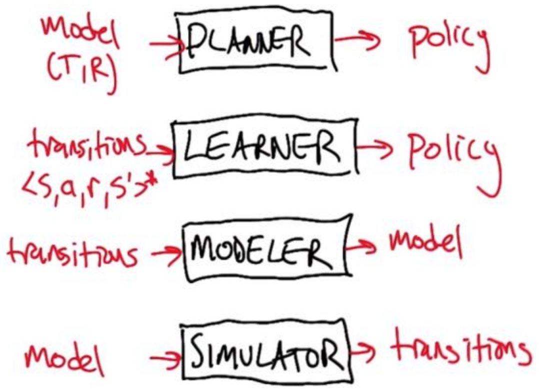
RL02. Reinforcement Learning

Introduction:

- In Markov Decision Process, our input is a model consisting of a transition function T and a reward function R, and the intended output is to compute the policy π (Planning).
- In Reinforcement Learning, the inputs are transitions (Initial state, action, reward, result state, …), and the intended output is to "learn" the policy π.
- Reinforcement Learning is about "reward maximization".



Approaches to RL:

- Policy Search Algorithm: Mapping states to actions. The problem with this approach is that the data doesn't reveal what actions need to be takes (Temporal Credit Assignment problem).
- Value Function based Algorithms: Mapping states to values. Learning values from states is quite direct, but turning this into a policy might be done in some sense using an $argmax$.
- Model-based Algorithm: Mapping (states & actions) to (next state & reward). Turning this into a utility function can be done using Bellman equations, then using $argmax$ to get the policy. This is computationally indirect.

More about Value Functions:

- The utility of a state (the long-term value of that state) is the reward of that state plus the discounted reward of the future. The reward of the future is the expectation over all possible next states given an action multiplied by the utility of each future state:

$$U(s) = R(s) + \gamma max_a \sum_{s'} T(s,a,s')U(s')$$

- The policy at state s, considering all the actions we can take to leave that state, we iterate over all the possible next states and calculate the probability of the utility of landing in a specific state:

$$\pi(s) = argmax_a \sum_{s'} T(s,a,s')U(s')$$

- Another function is the Q-function:

$$Q(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') max_a Q(s',a')$$

  This is the utility of leaving state s via action a, which is the reward of state s plus the discounted expected value of taking action a multiplied by the value of the optimum action in state s'.
- Now, $U(s)$ and $\pi(s)$ can be defined via $Q(s,a)$:

$$U(s) = max_a Q(s,a)$$
$$\pi(s) = argmax_a Q(s,a)$$

- Estimating the value of $Q(s,a)$ is called Q-Learning.

$Q-$ Learning:

- Q-Learning is estimating the value of $Q(s,a)$ based on transitions and rewards:

$$Q(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') max_a Q(s',a')$$

  - The issue is we don't have $R(s)$ and $T(s,a,s')$ since we're not in an MDP setting.
- Given a transition $< s,a,r,s' >$:

$$\hat{Q}(s,a) = (1-\alpha)\hat{Q}(s,a) + \alpha(r + \gamma\, max_{a'} \hat{Q}(s',a'))$$

$$\hat{Q}(s,a) \leftarrow^{\alpha_t} r + \gamma\, max_{a'} \hat{Q}(s',a')$$

  - $\hat{Q}(s,a)$ is an estimate of the Q-function that we'll update by a learning rate α_t in the direction of the immediate reward r plus the estimated value of the next state.

Estimating Q from Transitions:

- The goal is to estimate this function:

$$Q(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') max_a Q(s',a')$$

- We use the above equation to do this estimation. We're computing the expected value of the right-hand term:

$$\begin{aligned}\hat{Q}(s,a) &\leftarrow^{\alpha_t} r + \gamma\, max_{a'} \hat{Q}(s',a') \\ &= \mathbb{E}[r + \gamma\, max_{a'} \hat{Q}(s',a')] \\ &= R(s) + \gamma\, \mathbb{E}_{s'}[max_{a'} \hat{Q}(s',a')]\end{aligned}$$

- The distribution of the expected values over the next states ($\mathbb{E}_{s'}$) of the maximum estimated value ($max_{a'}$) of the estimated Q value of the next state ($\hat{Q}(s', a')$). This distribution is actually determined by the transition function, so we end up with this:

$$Q(s, a) = R(s) + \gamma \sum_{s'} T(s, a, s') max_a Q(s', a')$$

- The only issue with this analysis is that the value of $\hat{Q}(s', a')$ is actually changing over time. But it turns out that this update rule can solve the Markov Decision Processes.

- $Q -$ Learning Convergence:
  1. $\hat{Q}$ starts anywhere.
  2. $\hat{Q}(s, a) \xleftarrow{\alpha_t} r + \gamma\, max_{a'} \hat{Q}(s', a')$
  3. $\hat{Q}(s, a)$ will converge to $Q(s, a)$, which solves the Bellman Equation.
  4. This is only true if we visited all the (s, a) pairs (infinity!)
  5. Rules:

$$\sum_t \alpha_t = \infty$$
$$\sum_t {\alpha_t}^2 < \infty$$
$$s' \sim T(s, a, s')$$
$$r \sim R(s)$$

Choosing Actions:

- We have three question:
  - How to initialize $\hat{Q}$?
  - How α_t will decay?
  - How to choose actions?
- Method to choose actions:
  - Always choose a single specific action: We don't learn anything, and we violate the 4th rule of $Q -$ Learning (visiting all (s, a) pairs).
  - Choose randomly: We don't learn.
  - Use our estimate $\hat{Q}$ to choose actions. We can have some issues if we poorly initialized $\hat{Q}$ so that it always prefers a specific action over all the other actions. So, we initialize $\hat{Q}$ so that:

$$\bigvee_s \hat{Q}(s, a_0) = \uparrow\uparrow\uparrow$$
$$\bigvee_{s, a \neq a_0} \hat{Q}(s, a) = \downarrow\downarrow\downarrow$$

  - Random restarts: It will take a lot of time to get to the optimum answer.
  - Simulated Annealing, on the other hand, can facilitate a random but faster approach, while exploring the whole space:

$$\hat{\pi}(s) = argmax_a \hat{Q}(s, a) \qquad with\ probability\ 1 - \varepsilon$$

$\varepsilon -$ Greedy Exploration:

- In a GLIE approach, Greedy within the Limit with Infinite Exploration (or basically a decayed epsilon), two things happen:
  1. $\hat{Q} \rightarrow Q$ (Learning).
  2. $\hat{\pi} \rightarrow \pi^*$ (Using what we learned).
- Exploration-Exploitation dilemma:
  - Exploration is about getting the data that you need (learning) and exploitation is about stop learning and actually using what you've already learned.
  - The dilemma exists because there is only one agent interacting with the world with conflicting actions.