Reinforcement Learning

Satvik Gupta

September 23, 2023

Contents

Reinforcement Learning

The main idea is that we learn by interacting with our environment. RL tries to map situations to action. The agent's goal is to maximise a *reward*. We don't tell the agent which action to take, or even which action has higher reward (beyond an estimate). The agent must try different actions on its own and **learn.**

RL involves interaction between an active decision-making agent, and ts environment. The agent is always trying to seek a goal (or mazximize reward), even thought the environment is *uncertain.*

Exploration vs Exploitation

Exploration is any action that lets the agent discover new features about the environment, while exploitation means capitalizing on the information we have already learned.

If the agent continues to exploit only past experiences, it is likely to get stuck in a **suboptimal policy.**

On the other hand, if it continues to explore without exploiting, it **might never find a good policy.**

Elements of Reinforcement Learning

1. Policy
2. Reward Signal
3. Value Function
4. Model of the environment

Policy

Policy is how the agent behaves. It is a mapping of state to action, i.e, it tells the agent what action to perform, based on the state of the environment.

Reward Signal

Reward signal is the goal of the RL agent. Whenever the agent takes an action, the environment gives it a *reward*. The agent's goal is to maximize its **long term reward**.

Reward signal is the basis through which we change the policy. If the policy told us to do an action x, but that action resulted in a low (or negative) reward, then we may change the policy not to suggest x in that situation.

Reward may be a function of the state and the action taken.

Value Function

Reward signal gives us the immediate reward. Value function tells us the long term reward of a particular state. The *value* of a state is the total amount of reward an agent can accumulate in the future, if it **starts** at that state.

Model of Environment

Mimics the behavior of the environment. It lets the agent guess how the environment will react to a particular action. Using the model, the agent can predict the reward and next state, if it takes action x from its current state. Models are used for planning - i.e, the agent uses the model to consider many future situations before it actually experiences them.

Models are optional - not all RL agents will use a model. Those using a model are called **model-based**, and those without models are called **model-free**. Model-free methods are trial-and-error.

k-Armed Bandit

- We repeatedly get a choice among k actions.
- After each choice we get a reward, that depends on the action.
- We repeat this *time steps* times.
- We don't know action values for certain. We may have estimates.

At time t,

- **Action selected** – A_t
- **Reward received** – R_t
- **Value for action** a – $q_*(a)$ = expected reward given that a is selected.

$$q_*(a) = \mathbb{E}[R_T | A_t = a]$$

- **Estimated value of action** a – $Q_t(a)$

Ideally, $Q_t(a)$ should be as close to $q_*(a)$ as possible.

Greedy approach - always take the action with the highest Q value.

ϵ-greedy approach - At each step, with probability ϵ take a non-greedy action, i.e, take the action which does NOT have the highest estimated value.

> Methods that estimate the value of actions, and then use those estimates to choose which action to perform, are called **action-value methods**

A way to estimate $Q_t(a)$ is to average all the actual rewards received. This is the **sample-average** method

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}},$$

Figure 1: Estimation of Q_t

$1_{predicate}$ represents a random variable that is 1 when *predicate* is true, and 0 when it's false.

If denominator is 0, we can define $Q_t(a)$ as some fixed value, such as 0.

Incremental Updates

$$Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i$$

Instead of storing all the R_i values, we use incremental updating. The above formula can easily be reduced to :

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

This is a frequently occurring format in RL. General form:

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate]$$

StepSize is sometimes denoted by α or $\alpha_t(a)$

Non-stationary problems

Rewards may change during the course of the problem. These problems are called non-stationary. Incremental updating will not work for those. In these cases, we attach more weight to more recent rewards than rewards received a long time ago. For e.g, if a particular action gave us reward of 1 during the starting of the problem, but more recently it's been giving us reward of 10, our estimated reward of it should lean more towards 10.

We do this by fixing the value of α , instead of above where it changed according to n.

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

where $\alpha \in (0, 1]$ is constant. Q_{n+1} becomes a weighted average of all past rewards and the initial estimate Q_1

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

Recursing the formula, we get

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

Past rewards (with lower i) are multiplied with higher powers of a number that's less than 1, i.e, they get less weightage.

This is called **exponential recency-weighted average**

Optimal Initial Values

- Instead of setting all initial Q as 0, we can set them to an optimistically high value. This promotes exploration.
- In the 10-armed testbed, we chose $q_*(a)$ values from a distribution with mean 0 and variance 1. If we set initial Q as +5, this is very optimistic. Whatever the agent will do, it will always get a reward less than 5. Thus, it will try out other actions, which still have a estimated value of 5.
- All actions will be tried several times before the Q values converge.
- Optimistic greedy (with $\epsilon = 0$) can even perform better than realistic ($Q = 0$) ϵ-greedy.
- Not suited to non-stationary problem, since it only promotes exploration temporarily (during the start). If rewards keep changing, it cannot do anything at a later stage in the problem.
- More like a simple trick instead of a very useful method.

Gradient Bandit Algorithms

- Instead of estimated values, each action is given a *preference* $H_t(a)$. Larger preference means the action is more likely to be selected, but preference cannot be interpreted in terms of reward. Only relative preference of actions is important.

$$Pr\{A_t = a\} = \pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

$\pi_t(a)$ represents the probability of taking an action a at time t. Initially, $H = 0$ (or some other constant) for all a

On each step, after selecting A_t and receiving reward R_t,we update action preferences by:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

and

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a)$$

for all $a \neq A_t$

$\bar{R}_t$ is the average of all rewards received until (and including) time-step t. If the reward received in current step is higher than the average, the preference for A_t goes up, otherwise it goes down.