Reinforcement learning

Fraida Fund

Contents

Reinforcement learning

Elements of RL (1)

- An *agent* acts in an *environment*
- The agent sees a sequence of *observations* about the environment
- The agent wants to achieve a *goal*, in spite of some *uncertainty* about the environment.

May need to consider indirect, delayed result of actions.

Elements of RL (2)

- The *state* of the agent at time t is S_t (from $s \in \mathcal{S}$)
- The agent chooses action A_t at time t (from $a \in \mathcal{A}$)
- The agent earns a reward R_t for its actions
- The next state is determine by current state and current action, using a (possibly stochastic) state transition function $\delta(s, a)$:

$$P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a]$$

Elements of RL (3)

Over interactions in T time steps, the agent takes a sequence of actions and observes next states and rewards.

This sequence of interactions is called a *trajectory*:

$$S_1, A_1, R_2, S_2, A_2, \ldots, S_T$$

What are all the things an agent might try to learn?

Elements of RL (4)

- the *policy* π is the agent's mapping from state to action (or probabilities of action)
- the environment sends a *reward* back to the agent, depending on its state and action (may be stochastic), and a *value function* describes expected total **future** reward from a state
- we may sometimes have/learn a *model* of the environment, which we can use to **plan** before or during interactions with the environment

Taxonomy of RL agents

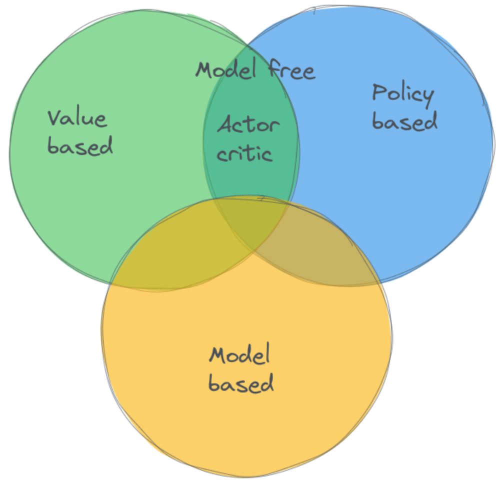

Figure 1: Taxonomy of RL agents.

- Policy-based: build an explicit representation of policy $\pi : S \to A$
- Value-based: try to learn what is the expected total reward for each state or state-action pair. Then there is an *implicit* policy: select the action that maximizes that.
- Actor-critic methods use both policy and value function learning.
- Model-based: uses either a known or learned model of the environment.
- Model-free: does not know or try to learn the model.
- (Model-free methods interact with the environment by trial-and-error, where model-based methods can plan for future situations by computation on the model.)

The optimization problem

Reward

Suppose the state transition function is

$$P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$$

the reward for a state-action will be

$$R(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} P(s', r|s, a)$$

The state transition function gives the probability of transitioning from state s to s' after taking action a, while obtaining reward r.

Policy

We want to find a *policy*, or a probability distribution over actions for a given state:

$$\pi(a|s) = \mathbb{P}_\pi[A = a|S = s]$$

Value function

Let future reward (**return**) from time t on be

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where the discount factor $0 < \gamma < 1$ penalizes future reward.

State-value

The state-value of a state s is the expected return if we are in the state at time t:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

Action-value

The action value of a state-action pair is

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

Relationship between Q and V

For a policy π, we can sum the action values weighted by the probability of that action to get:

$$V_\pi(s) = \sum_{a \in \mathcal{A}} Q_\pi(s, a)\pi(a|s)$$

Action advantage function

The difference between them is the action advantage:

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$$

"Taking this action in this state" vs. "getting to this state."

Optimal value function

The optimal value function maximizes the return (future expected reward):

$$V_*(s) = \max_\pi V_\pi(s)$$

$$Q_*(s, a) = \max_\pi Q_\pi(s, a)$$

Optimal policy

The optimal policy achieves the optimal value functions:

$$\pi_* = \arg\max_\pi V_\pi(s)$$

$$\pi_* = \arg\max_\pi Q_\pi(s, a)$$

i.e. $V_{\pi_*}(s) = V_*(s)$ and $Q_{\pi_*}(s, a) = Q_*(s, a)$.

Optimal policy breakdown

We can also think of it as the policy that maximizes current reward + discounted value of next state:

$$\pi_* = \arg\max_\pi r(s, a) + \gamma V^*_\pi(\delta(s, a))$$

How do we learn this policy?

- what is the loss function?
- what are the training samples?

Q learning

Q table

- Each row is an action
- Each column is a state
- $Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$
- Table stores current estimate $\hat{Q}(s, a)$

Iterative approximation

- start with random values
- observe state, then iteratively:
  - choose action a and execute,
  - observe immediate reward r and new state s'
  - update $\hat{Q}(s, a)$ using $r + \gamma \max_{a'} \hat{Q}(s', a')$
  - $s \leftarrow s'$

Exploration and exploitation

If we only take the best actions we know about, we might miss out on learning about other, better actions!

- **exploration**: take some action to find out about environment, even if you may miss out on some reward
- **exploitation**: take the action that maximizes reward, based on what you know about the environment.

ϵ-greedy policy

- With probability ϵ, choose random action
- With probability $1 - \epsilon$, choose optimal action