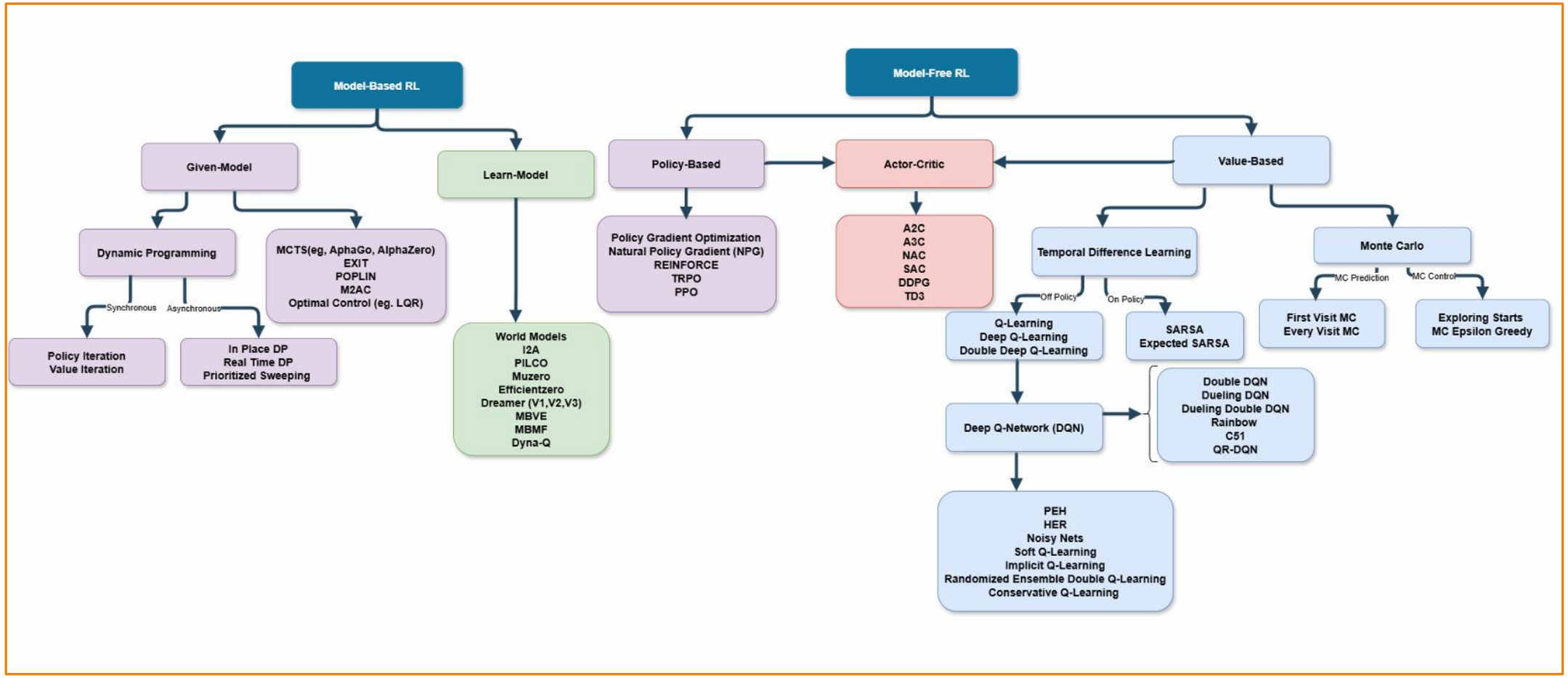
Reinforcement Learning

Fateme Taroodi



Fateme Taroodi

Model-free RL algorithms

Model-free RL algorithms learn an optimal **policy** (i.e., a mapping from states to actions) **without explicitly learning a model of the environment** (i.e., transition probabilities and reward functions). The agent interacts with the environment and learns from experience.

Types of Model-Free RL
Model-free RL methods can be categorized into:
1.Value-Based Methods – Learn the value of actions and derive a policy from it.
    1. Example: **Q-learning, Deep Q-Network (DQN)**
2.Policy-Based Methods – Directly learn the policy without estimating value functions.
    1. Example: **REINFORCE, PPO, TRPO**
3.Actor-Critic Methods – Combine value-based and policy-based methods.
    1. Example: **A2C, DDPG, TD3, SAC**

Advantages of Model-Free RL
✓ Works well in **unknown environments** where transition dynamics are complex or difficult to model.
✓ Can handle **high-dimensional** environments (e.g., games, robotics, finance).
✓ Does not require learning an explicit model, making it **simpler** and often **more sample-efficient** in low-data environments.
Disadvantages of Model-Free RL
❌ Requires a large number of **interactions** with the environment to learn an optimal policy.
❌ **Exploration is inefficient**, leading to longer training times.
❌ Struggles with **long-horizon tasks** (e.g., learning to drive a car).

Model-based RL algorithms

Model-based RL algorithms **learn a model of the environment** by estimating **transition probabilities** $P(s'|s,a)$ and **reward function** $R(s,a)$.This model is then used for **planning** (e.g., predicting future states and rewards) to improve decision-making.

Types of Model-Based RL

1.Planning-Based Methods – Use a learned model to plan ahead before taking actions.

1. Example: **Value Iteration, Policy Iteration**

2. Model-Based Deep RL – Combines deep learning with environment models to improve sample efficiency.

1. Example: **AlphaZero, MuZero, Dreamer, World Models**

Advantages of Model-Based RL

✓ More **sample-efficient** (requires fewer interactions with the real environment).
✓ Can use **simulations** to improve learning without collecting real data.
✓ Enables **long-horizon planning**, making it useful for complex tasks.

Disadvantages of Model-Based RL

❌ Requires **learning an accurate model**, which can be challenging.
❌ If the model is inaccurate, the policy learned may be **suboptimal** (model bias).
❌ More computationally expensive than model-free RL.

Advantages of Model-Based RL

MBRL offers several advantages, making it a compelling choice for specific applications:

•**Sample Efficiency:** Since the agent learns and plans using a model, fewer interactions with the environment are required, which is crucial for real-world problems where interactions can be expensive or slow.

•**Better Generalization:** The learned model allows agents to generalize better to unseen situations because they are trained to predict and adapt to new scenarios.

•**Long-Term Planning:** MBRL facilitates long-term planning by allowing agents to simulate long sequences of actions and their outcomes, which is valuable for tasks like robotic control or game-playing.

Challenges in Model-Based RL

While MBRL holds promise, it comes with its own set of challenges:

•**Model Accuracy:** Building an accurate model of a complex environment is challenging. Any inaccuracies in the model can lead to suboptimal policies and degraded performance.

•**Computational Complexity:** MBRL requires both learning the model and planning over it, which can be computationally intensive, especially in large, high-dimensional environments.

•**Exploration vs. Exploitation Dilemma:** The agent may over-rely on its learned model, leading to limited exploration of the environment, which might hinder the discovery of optimal solutions.

Dynamic Programming is a problem-solving method used to **break complex problems into smaller, simpler subproblems. Instead of solving the same subproblem multiple times, it stores the results of these subproblems and reuses them when needed.** This saves time and makes the solution more efficient. It is often used in optimization problems, where you need to find the best solution among many possibilities. Examples include finding the shortest path, solving knapsack problems, or calculating Fibonacci numbers. In this article, we will cover all about dynamic programming, from basics to advanced concepts, in a simple and easy-to-understand way.

Dynamic programming algorithms solve a category of problems called planning problems. Here in given the complete model and specifications of the environment (MDP), we can successfully find an optimal policy for the agent to follow. It contains **two main steps**

Break the problem into subproblems and solve it. **Solutions to subproblems are cached or stored for reuse to find an overall optimal solution to the problem at hand.**

A state S_t is *Markov* if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

Transition step, represented by a tuple (s, a, s', r)

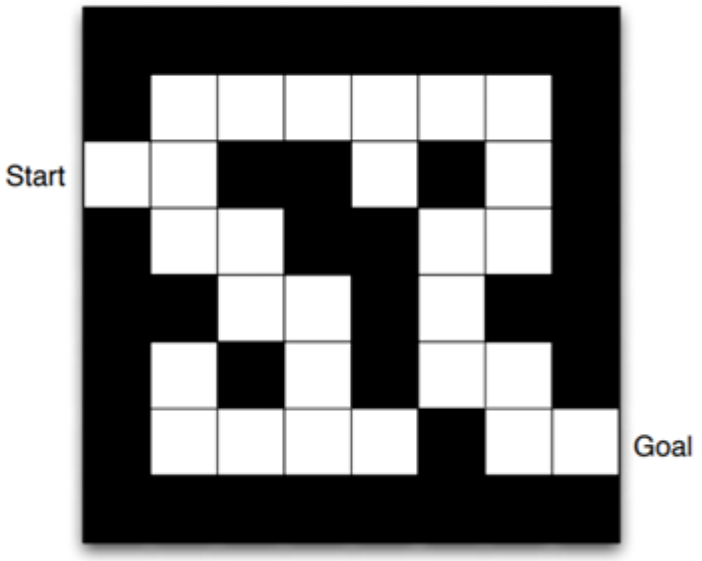
$$P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$$

State-transition function

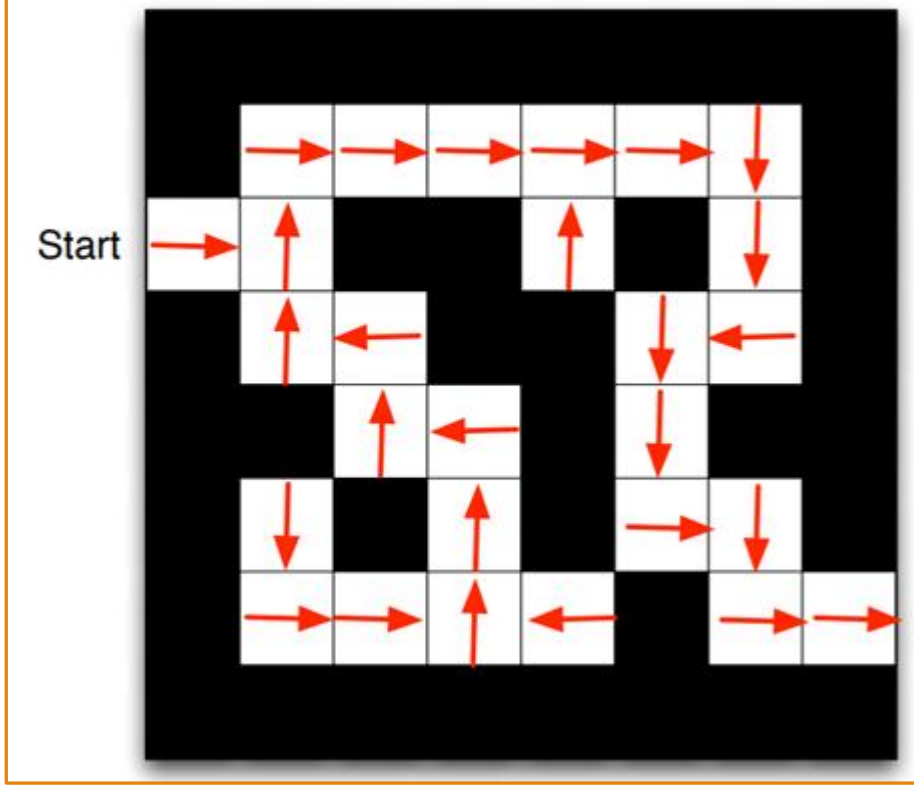
$$P_{ss'}^{a} = P(s'|s,a) = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} P(s', r|s, a)$$

Reward function

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



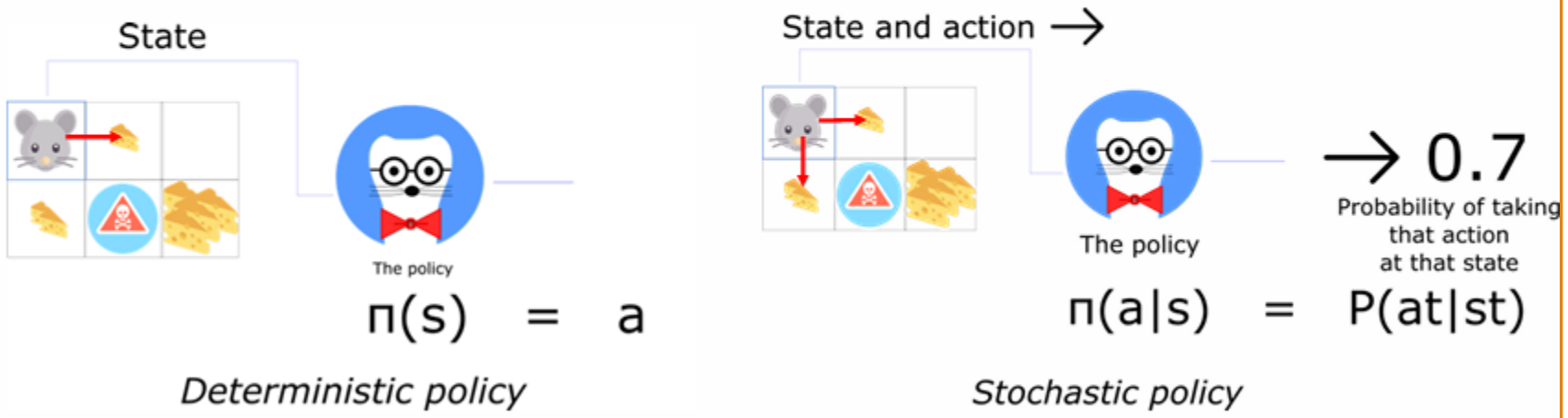
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location



Policy

Deterministic: $\pi(s) = a$.

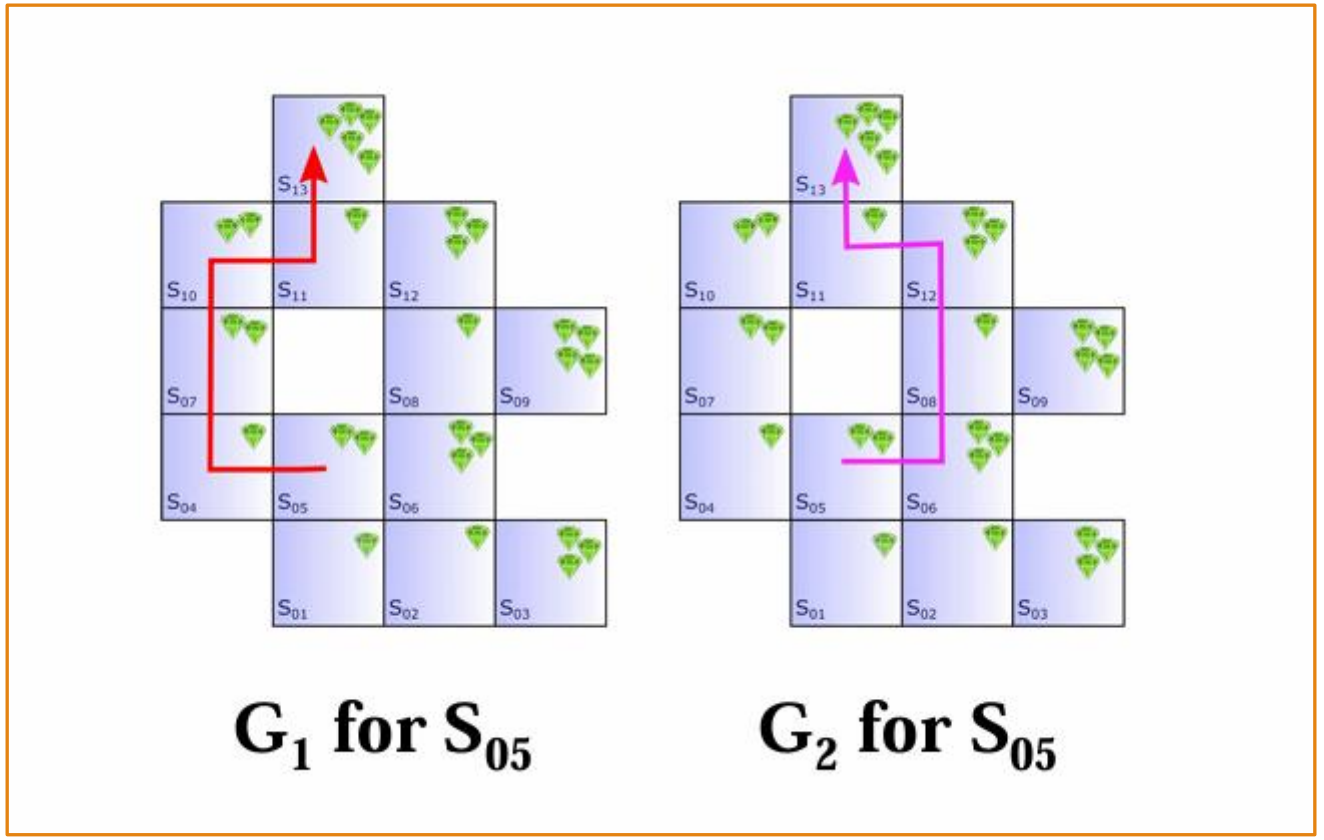
- Deterministic: $\pi(s) = a$.
- Stochastic: $\pi(a|s) = \mathbb{P}_\pi[A = a|S = s]$.



Value function measures the goodness of a state or how rewarding a state or an action is by a prediction of future reward.

return $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$

For Episode: $S_1, A_1, R_2, S_2, A_2, \ldots, S_T$


S13
S10
S11
S12
S07
S08
S09
S04
S05
S06
S01
S02
S03
G1 for S05
S13
S10
S11
S12
S07
S08
S09
S04
S05
S06
S01
S02
S03
G2 for S05

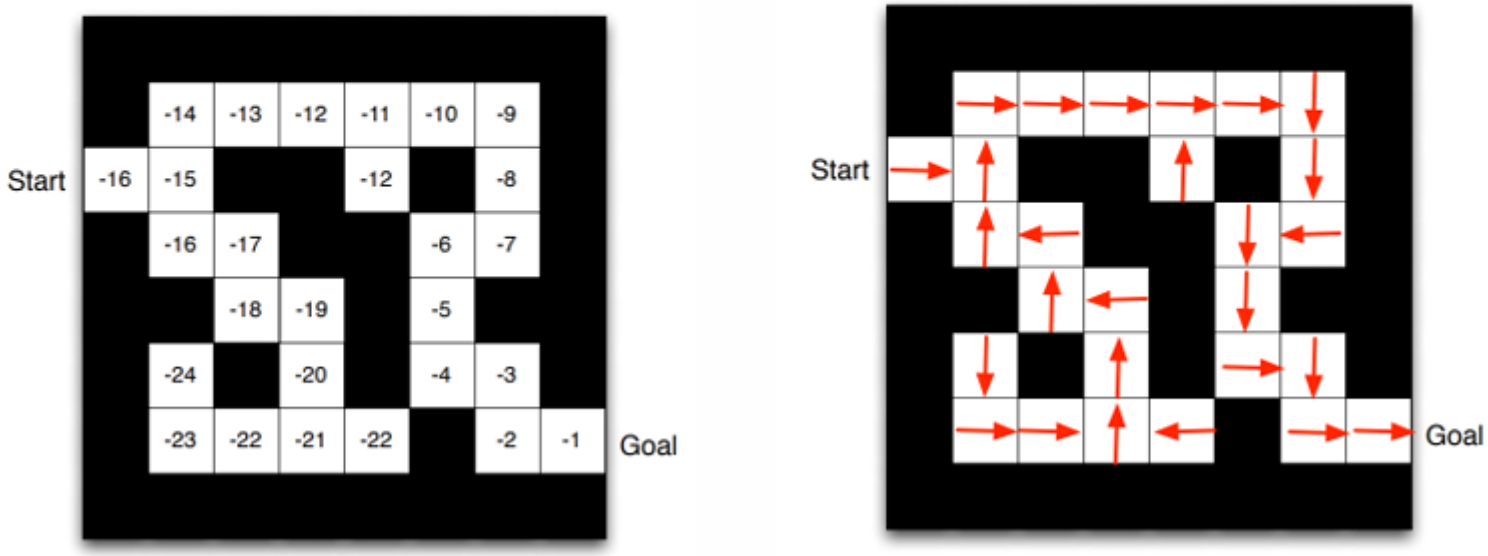
Discounting(why?)

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$\gamma = 0 \longrightarrow \gamma = 1$$

state-value

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



state-value

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

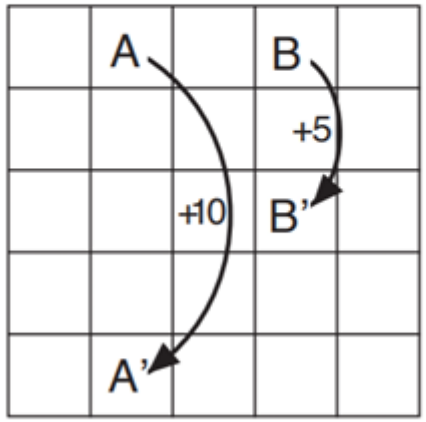
action-value

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

Problems in RL

- *Pb1*: Estimating v_π or q_π is called policy evaluation or, simply, prediction
  - Given a policy, what is my expected return under that behaviour?
  - Given this treatment protocol/trading strategy, what is my expected return?
- *Pb2*: Estimating v_* or q_* is sometimes called control, because these can be used for policy optimisation

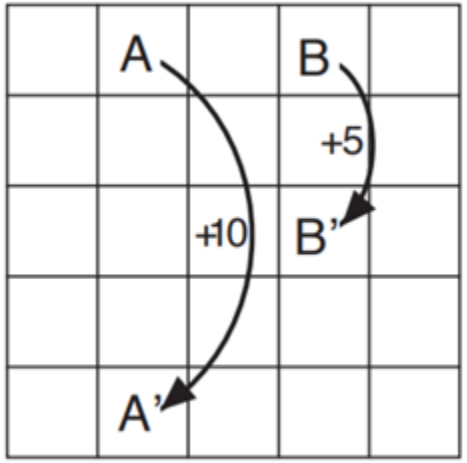
Using Random Policy you can predict value-state



Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

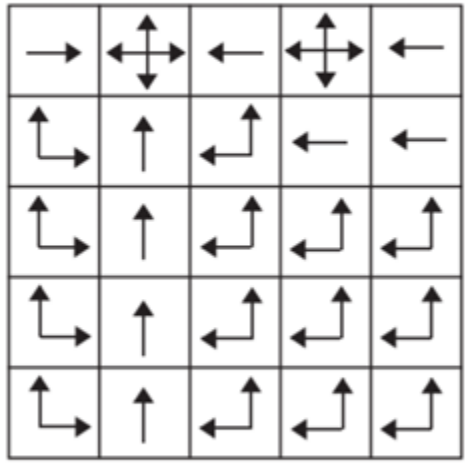
Having value-state you can Control your agent



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) v_*

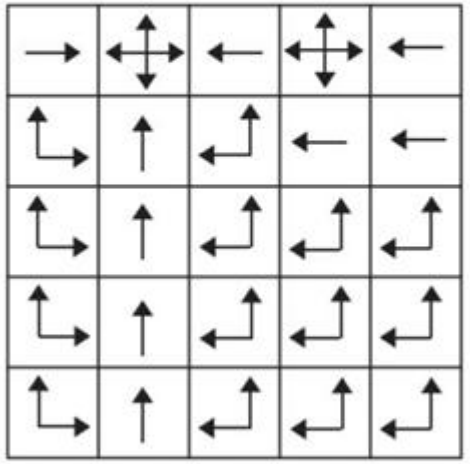


c) π_*

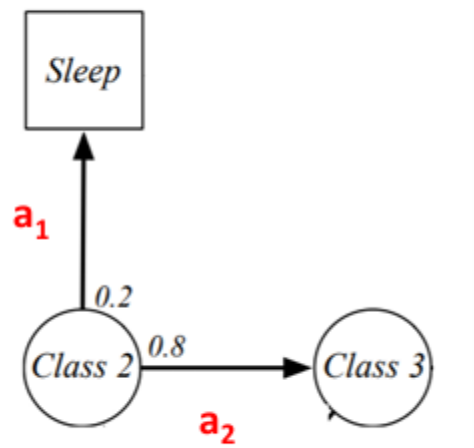
Having V^* we have $policy^*$

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) v_*



c) π_*



$$Q_{\Pi}(class2, a_1) = 1$$

$$Q_{\Pi}(class2, a_2) = 3$$

$$V_{\Pi}(class2) = 0.2 \times Q_{\Pi}(class2, a_2) + 0.8 \times Q_{\Pi}(class2, a_2)$$

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

state-value

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

action-value

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

The optimal value function produces the maximum return:

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

Can we search over policies?

Bellman Equation

Fateme Taroodi

Bellman Equations

$$
\begin{aligned}
V(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]
\end{aligned}
$$

Bellman Equations

$$
\begin{aligned}
V(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s]
\end{aligned}
$$

Bellman Equations

$$
\begin{aligned}
V(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&\overset{?}{=} \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s]
\end{aligned}
$$

Bellman's Equations

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

$$= \sum_{s'} \sum_{r} p(s', r|s, \pi(a|s))[r + \gamma v_\pi(s')].$$

$$= \sum_{a} \pi(a|s) \sum_{s'} \sum_{r} p(s', r|s, a)[r + \gamma v_\pi(s')].$$

Bellman Equations

$$
\begin{aligned}
Q(s,a) &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a \sim \pi} Q(S_{t+1}, a) \mid S_t = s, A_t = a]
\end{aligned}
$$

Bellman Equations

$$
\begin{aligned}
Q(s,a) &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a \sim \pi} Q(S_{t+1}, a) \mid S_t = s, A_t = a]
\end{aligned}
$$

Optimal state-value

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

Form known policy we can find V_*