Reinforcement Learning: A Gentle Introduction

PUBH 7475/8475

Based on “Deep Reinforcement Learning in Action” by A. Zai & B. Brown

https://www.manning.com/books/deep-reinforcement-learning-in-action

Outline

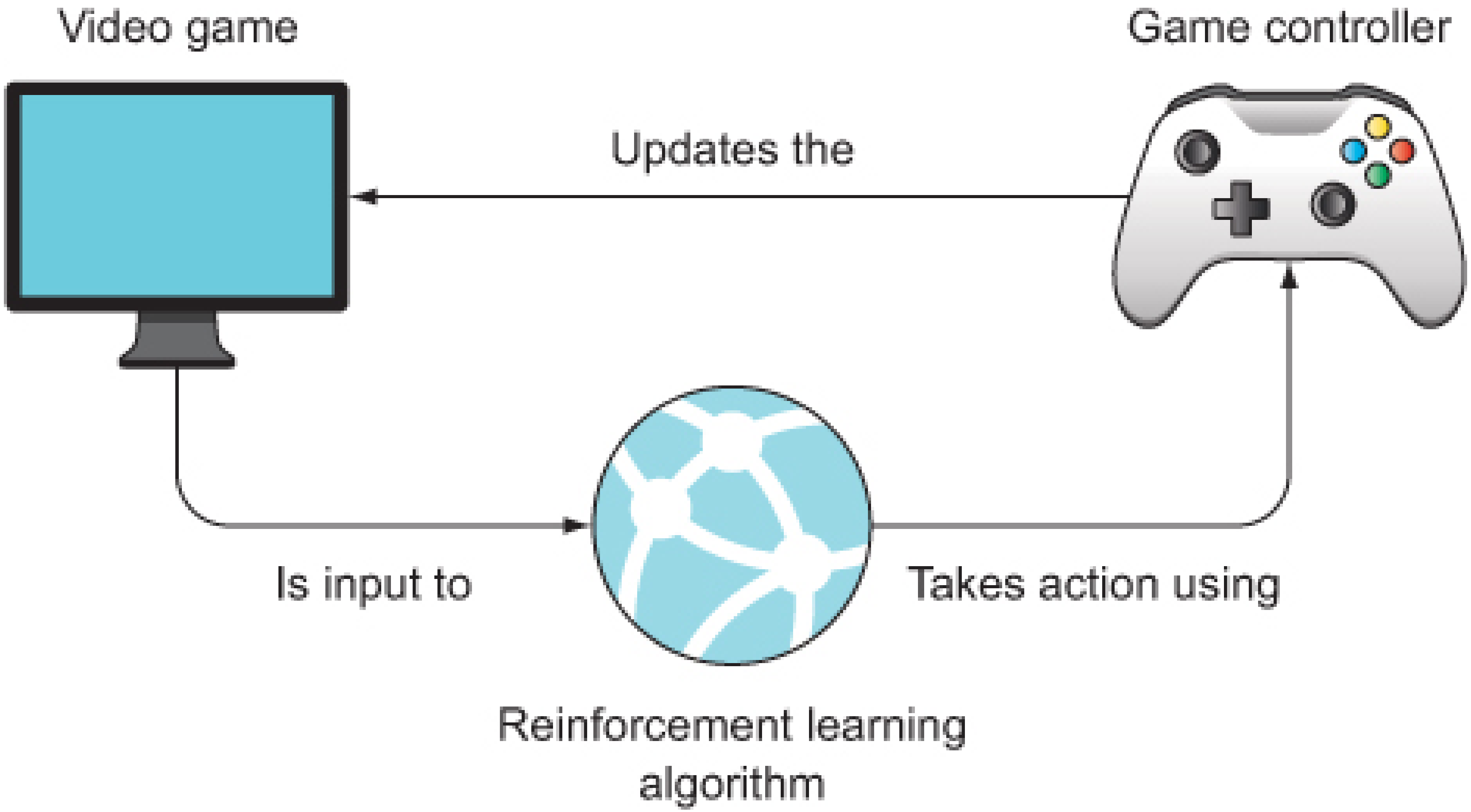
- What is RL?
- Q-Learning: DeepMind’s DQN
- Policy methods
- Actor-critic methods

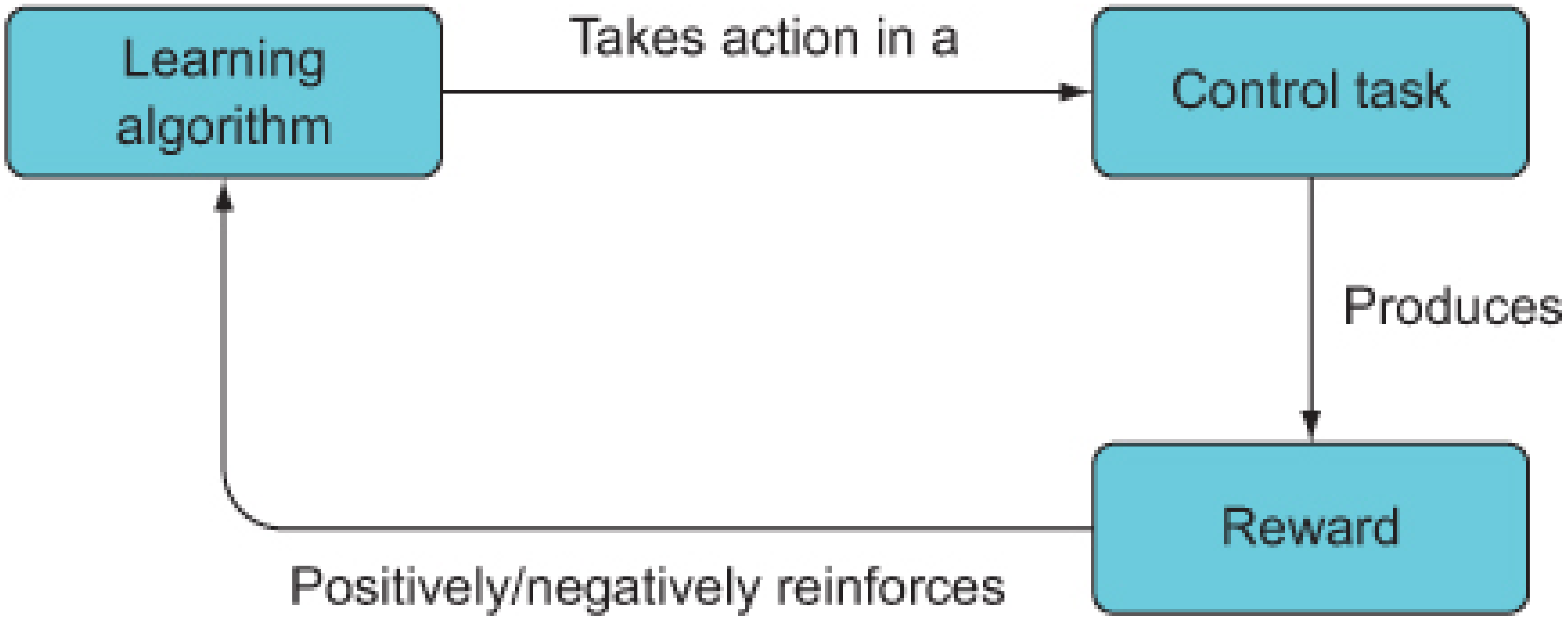
- General comments:

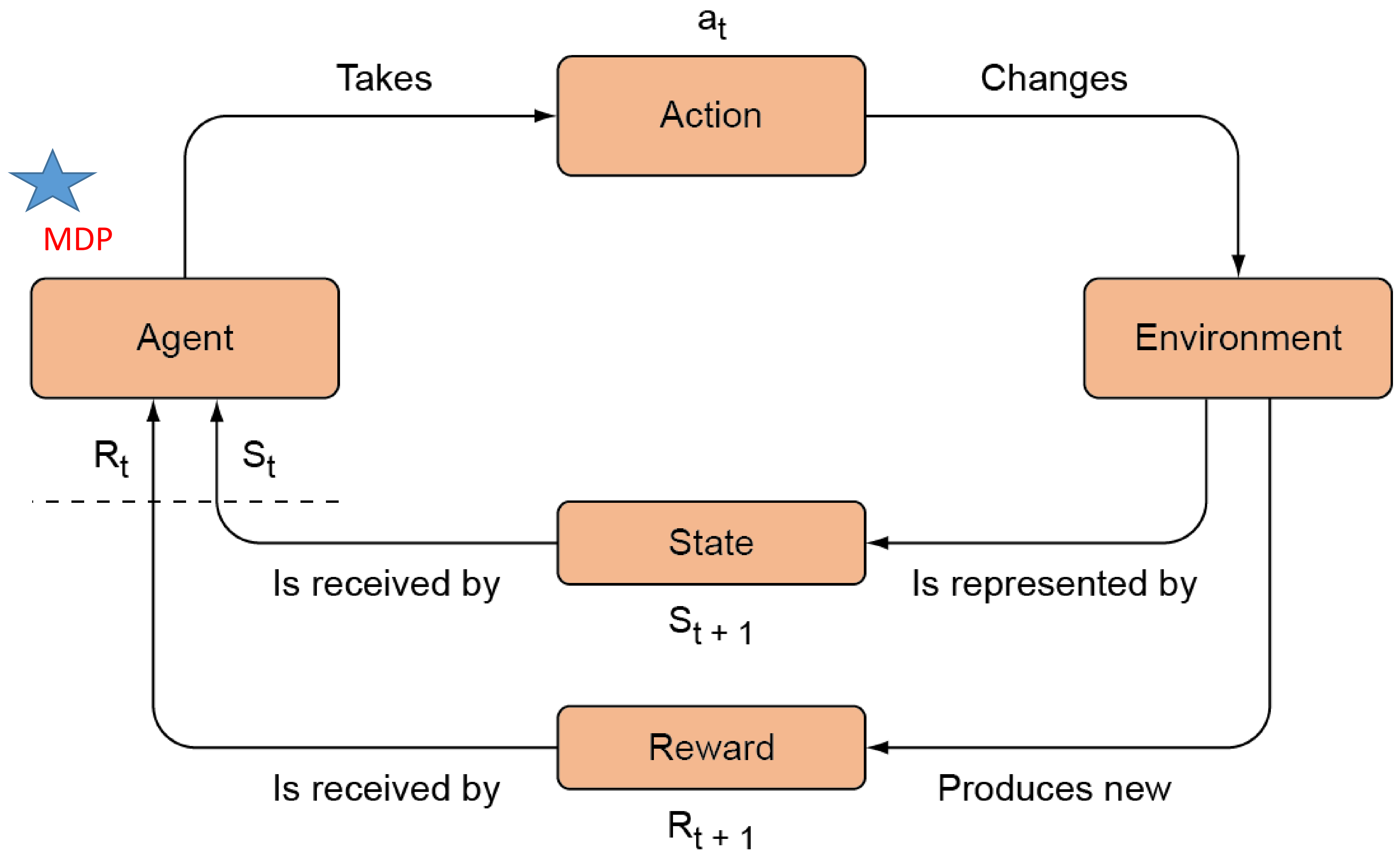
+/-: highly mathematical

+: most similar to human learning; impressive applications (e.g. DeepMind’s AlphaGo, AlphaZero); rapid development

-: time-consuming, not efficient; unstable, hard to train


Video game
Game controller
Updates the
Is input to
Takes action using
Reinforcement learning
algorithm


Learning algorithm
Takes action in a
Control task
Produces
Reward
Positively/negatively reinforces



a_t
Takes
Action
Changes
MDP
Agent
Environment
R_t
S_t
State
Is received by
Is represented by
S_{t+1}
Reward
Is received by
Produces new
R_{t+1}

The action-value (Q) function

Math	English
Q_π: $(s\|a) \rightarrow E(R\|a,s,\pi)$,	Q_π is a function that maps a pair, (s, a), of a state, s, and an action, a, to the expected reward of taking action a in state s, given that we're using the policy (or "strategy") π.

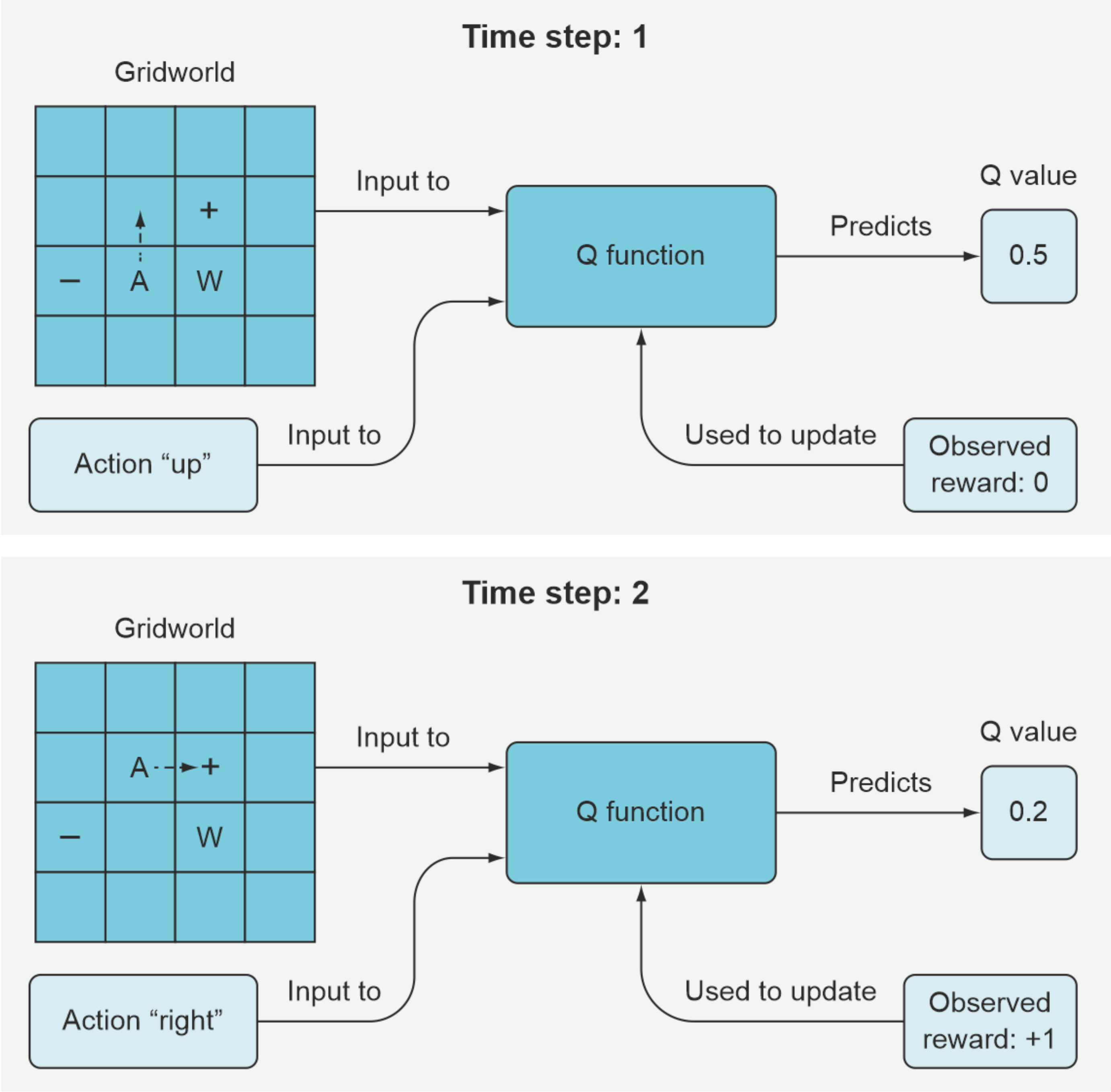
The policy function

Math	English
π; $s \rightarrow Pr(A\|s)$, where $s \in S$	A policy, π, is a mapping from states to the (probabilistically) best actions for those states.

The state-value function

Math	English
V_π: $s \rightarrow E(R\|s,\pi)$,	A value function, V_π, is a function that maps a state, s, to the expected rewards, given that we start in state s and follow some policy, π.

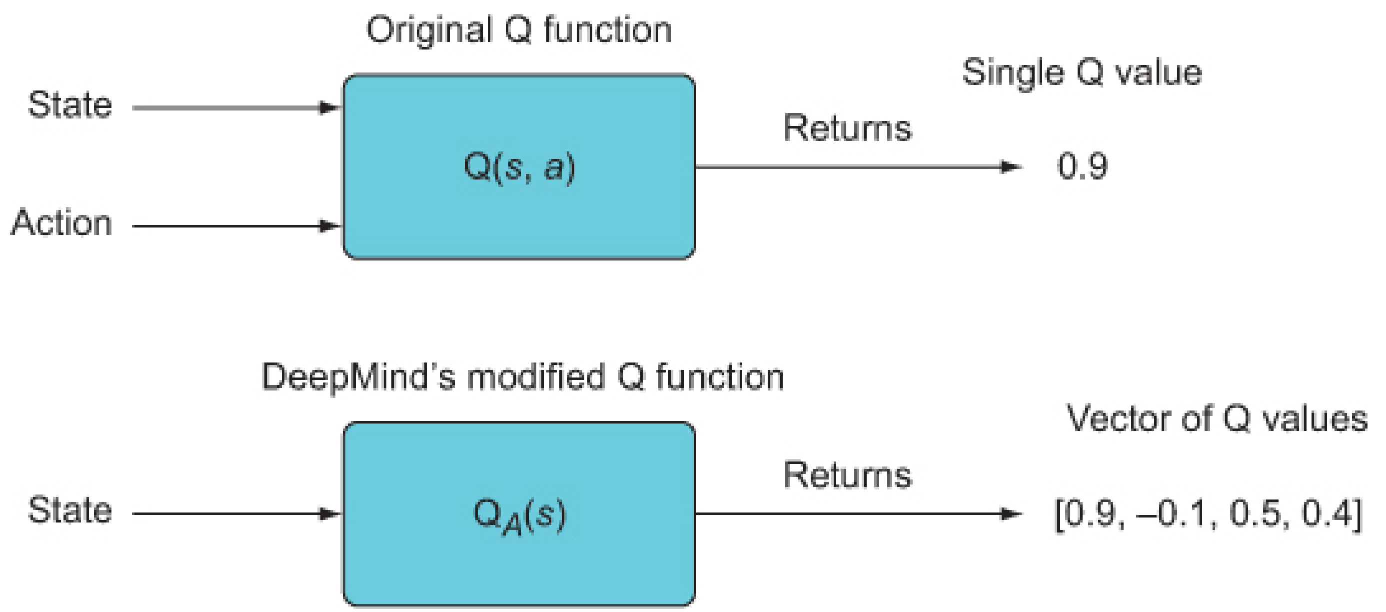
Q learning: learn the Q function

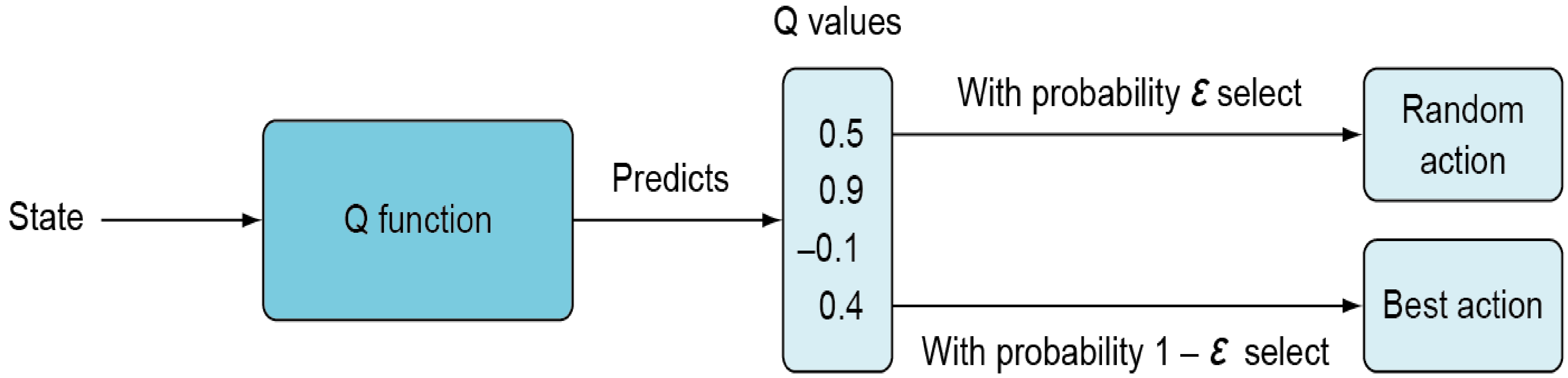


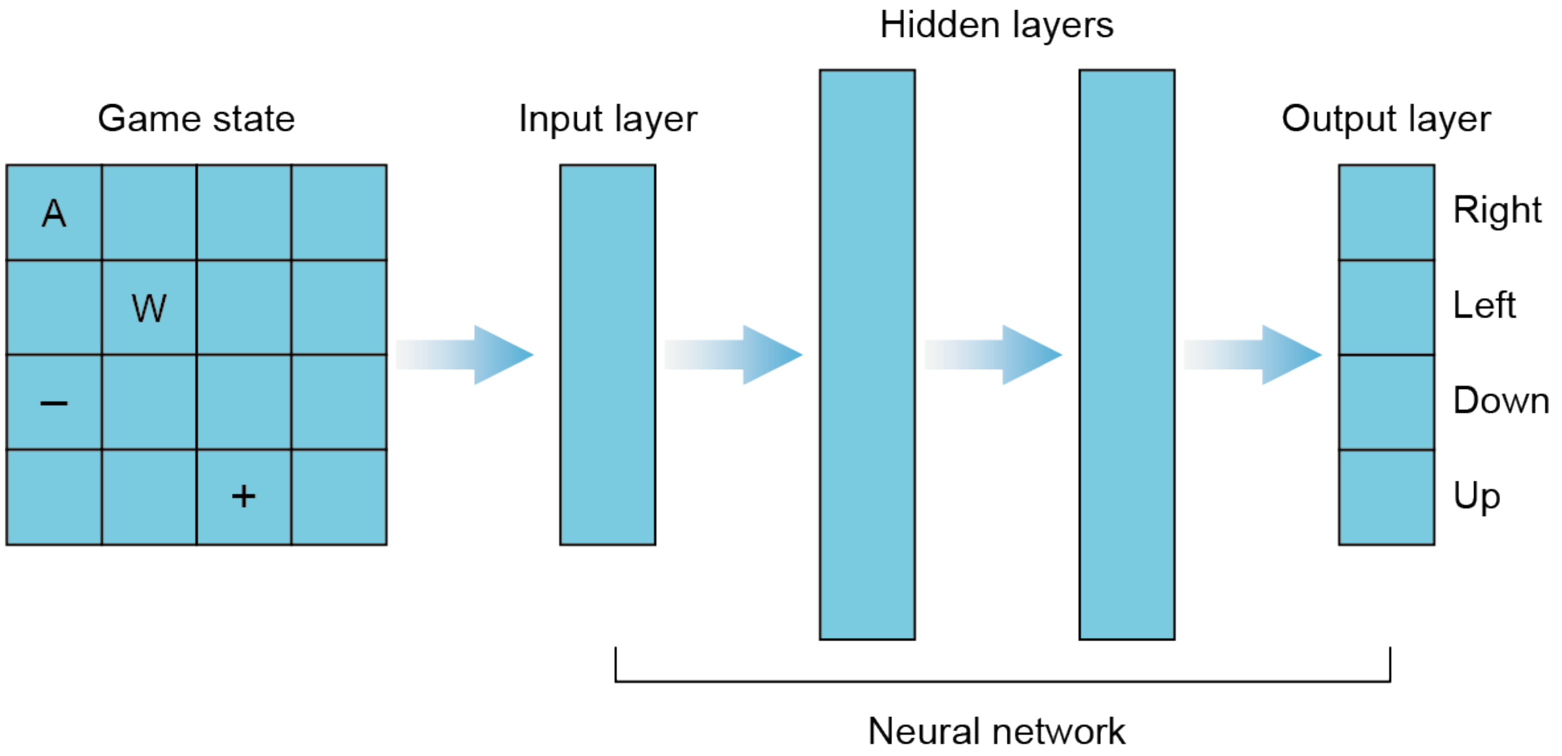
$$\overset{\substack{\text{Updated}\\ \text{Q value}\\ \downarrow}}{Q(S_t, A_t)} = \overset{\substack{\text{Current}\\ \text{Q value}\\ \downarrow}}{Q(S_t, A_t)} + \underset{\substack{\uparrow\\ \text{Step size}}}{\alpha}[\overset{\substack{\text{Observed}\\ \text{reward}\\ \downarrow}}{R_{t+1}} + \underset{\substack{\uparrow\\ \text{Discount}\\ \text{factor}}}{\gamma} \max \overset{\substack{\text{Max Q value}\\ \text{for all actions}\\ \downarrow}}{Q(S_{t+1}, a)} - Q(S_t, A_t)]$$

The Bellman equation:

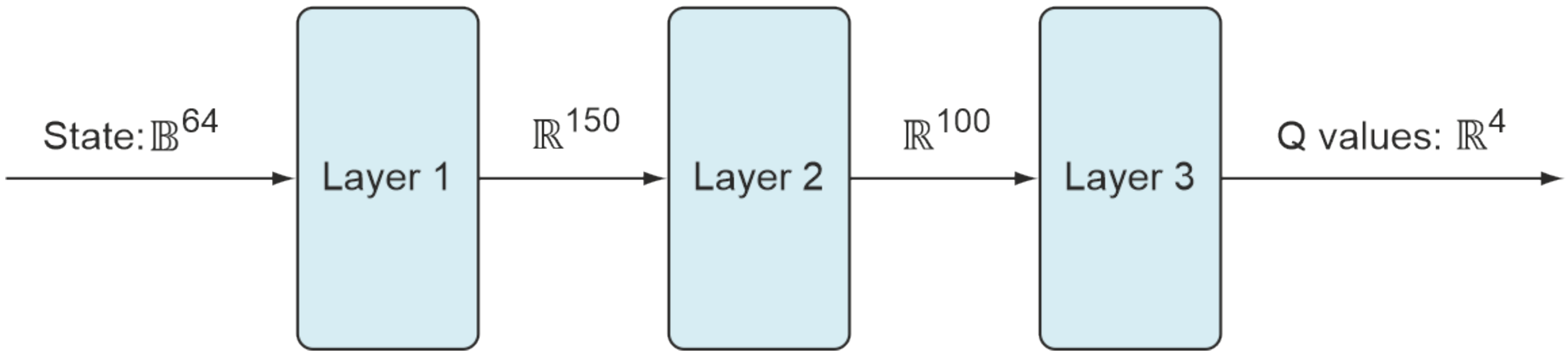
$$Q_\pi(s_t, a_t) \leftarrow r_t + \gamma \times max[Q_\pi(s_{t+1}, a)]$$

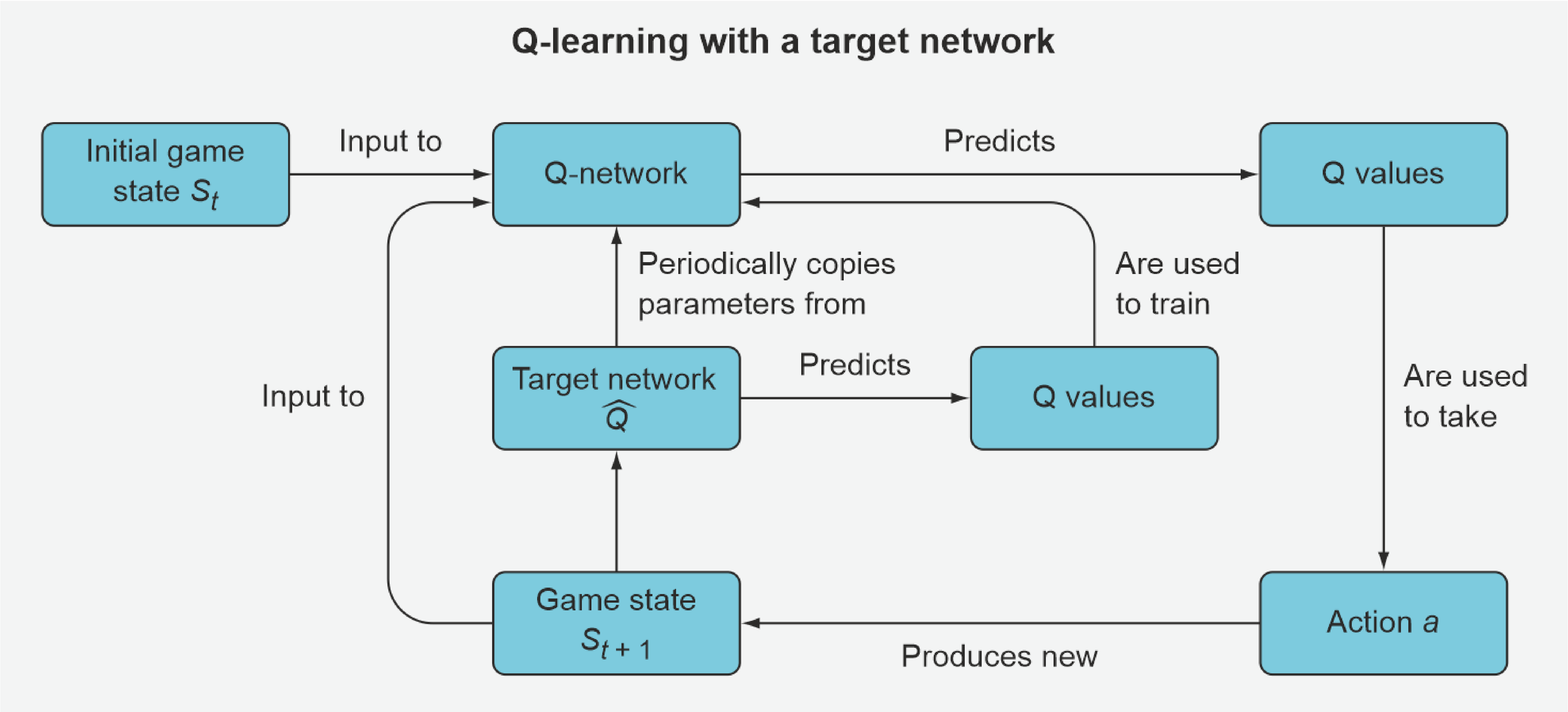

Original Q function
State
Action
Q(s, a)
Returns
Single Q value
0.9
DeepMind's modified Q function
State
$Q_A(s)$
Returns
Vector of Q values
[0.9, –0.1, 0.5, 0.4]


State
Q function
Predicts
Q values
0.5
0.9
–0.1
0.4
With probability ε select
Random action
With probability 1 – ε select
Best action



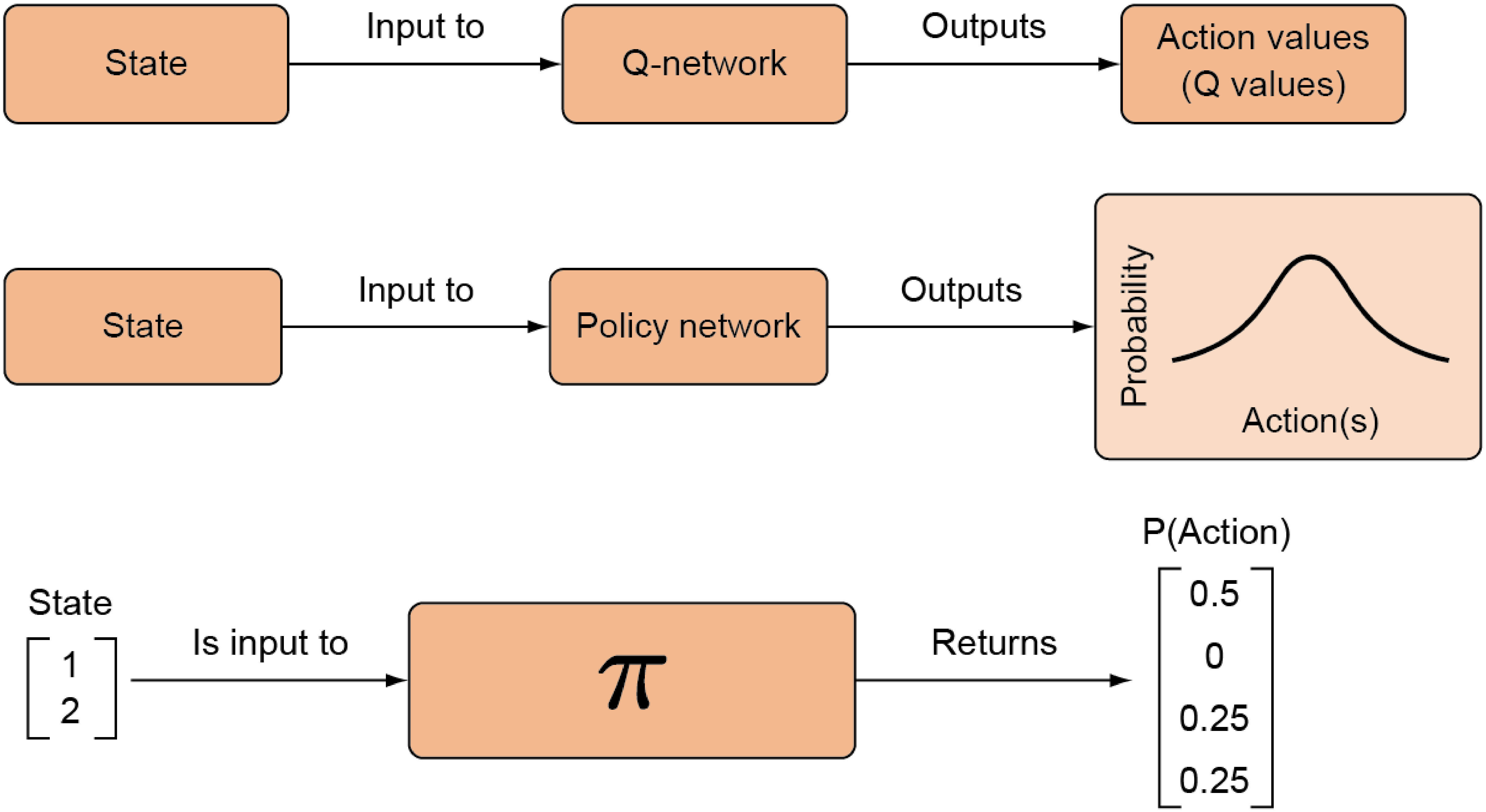
Deep Q-network



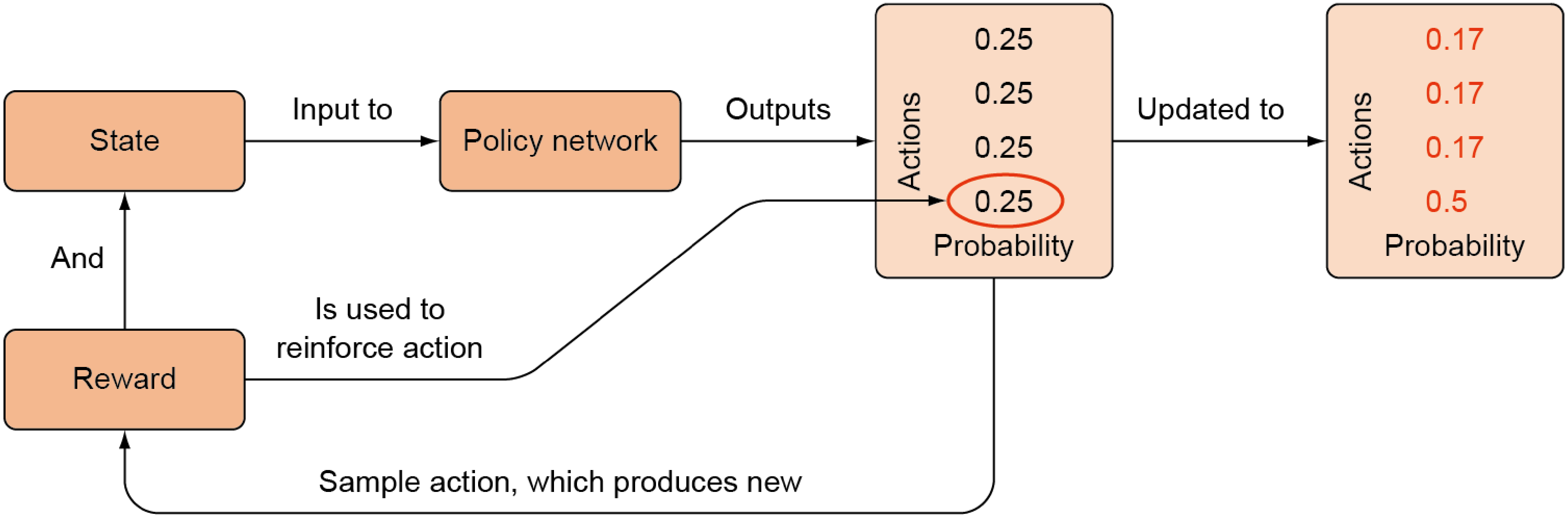


Before: $Q_{new} = R_t + d*\max(Q(S_{t+1}))$, not stable

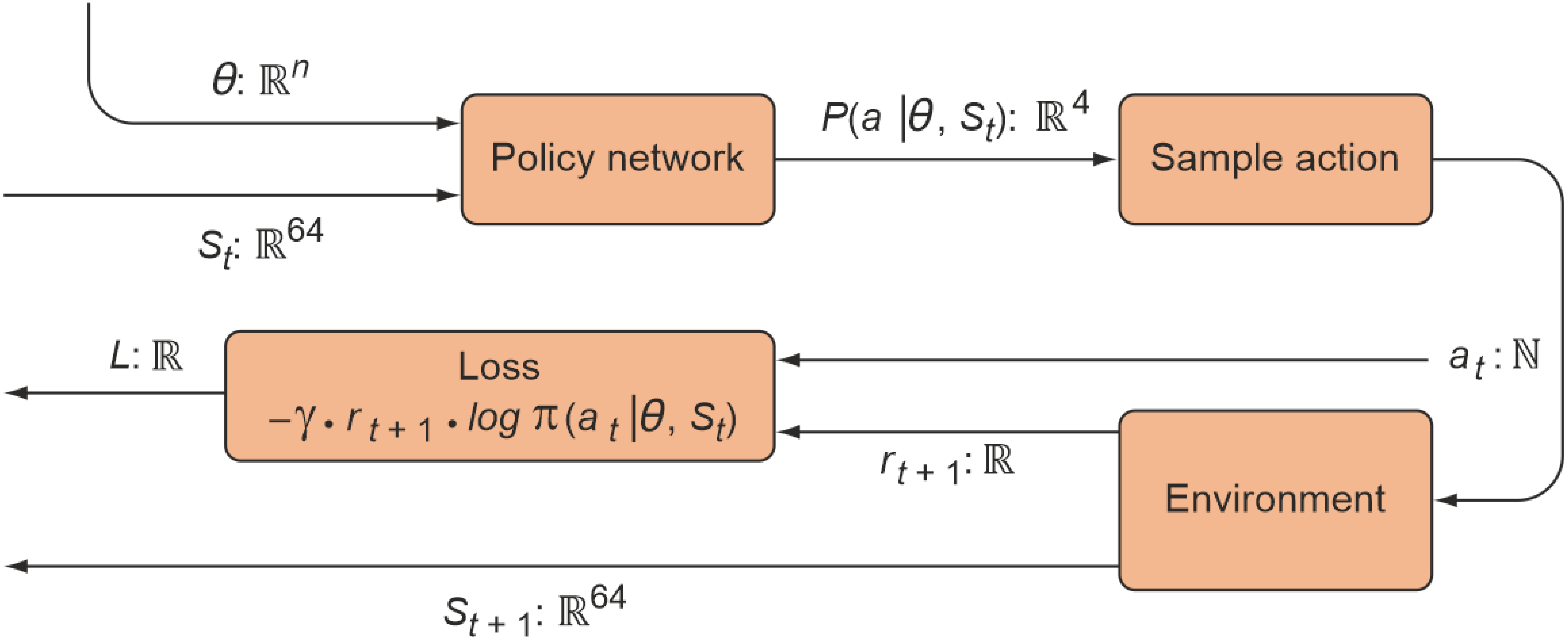
Now: $Q_{new} = R_t + d*\max(Q'(S_{t+1}))$


State
Input to
Q-network
Outputs
Action values
(Q values)
State
Input to
Policy network
Outputs
Probability
Action(s)
State
1
2
Is input to
π
Returns
P(Action)
0.5
0
0.25
0.25

Policy methods

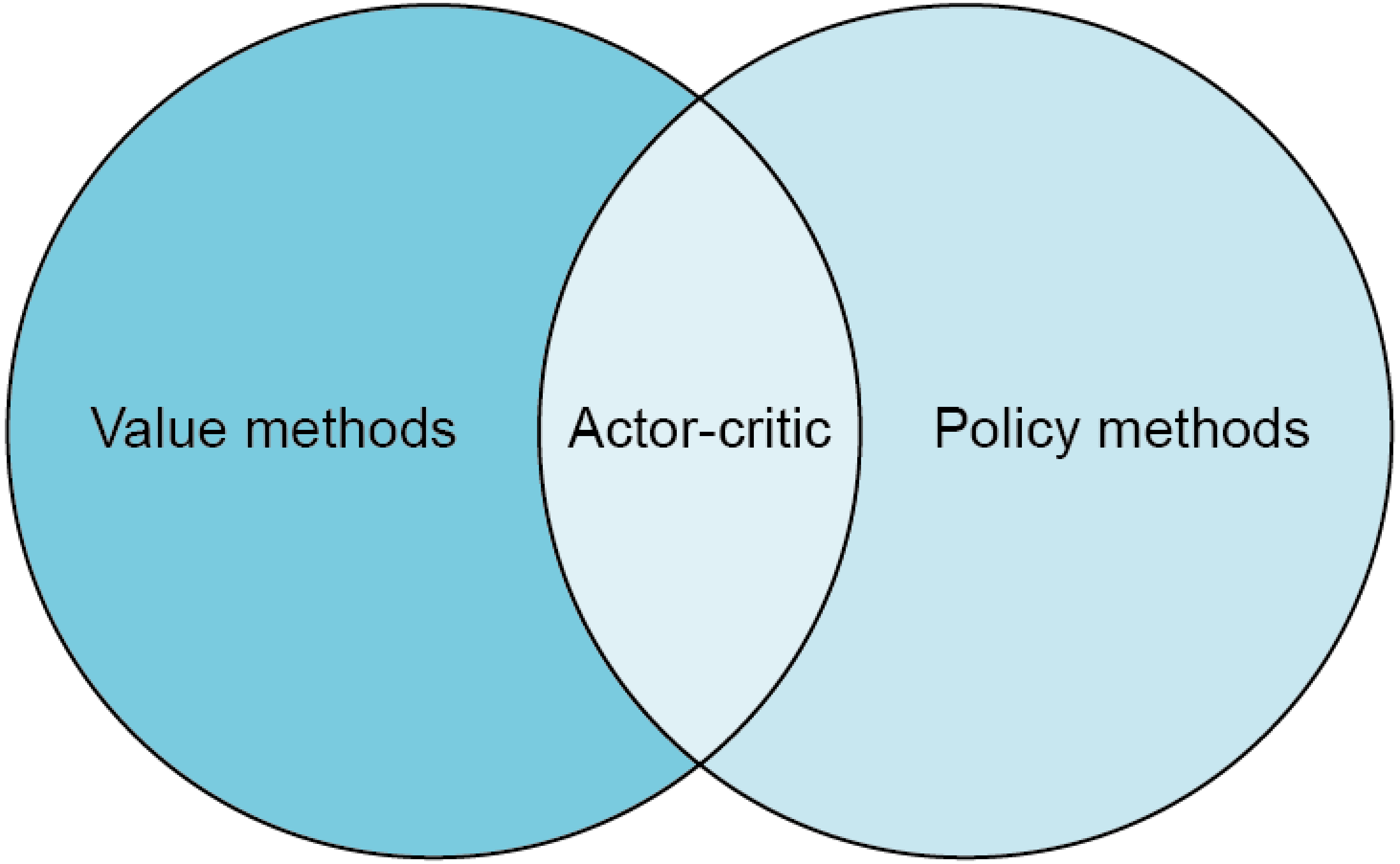


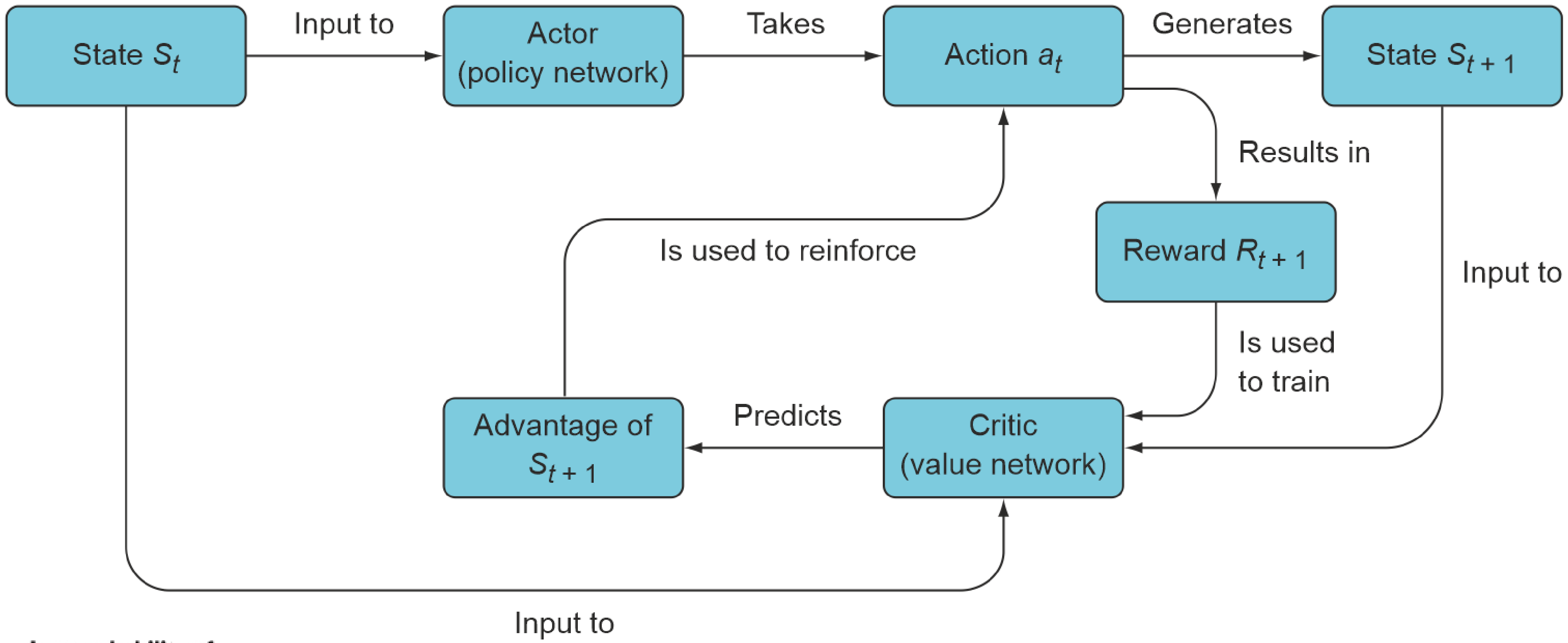
Why? Compared to Q-learning, generally regarded advantageous, even necessary (if the action space is continuous), but more mathematical…



θ: $\mathbb{R}^n$
Policy network
S_t: $\mathbb{R}^{64}$
$P(a \mid \theta, S_t)$: $\mathbb{R}^4$
Sample action
$a_t : \mathbb{N}$
Loss
$-\gamma \cdot r_{t+1} \cdot log\, \pi(a_t \mid \theta, S_t)$
L: $\mathbb{R}$
r_{t+1}: $\mathbb{R}$
Environment
S_{t+1}: $\mathbb{R}^{64}$

State-of-the-art:





Log probability of action given state **Return** **State value**

$$\text{Loss} = -log\,(\pi(a \mid S)) \cdot (R - V_{\pi}\,(S))$$