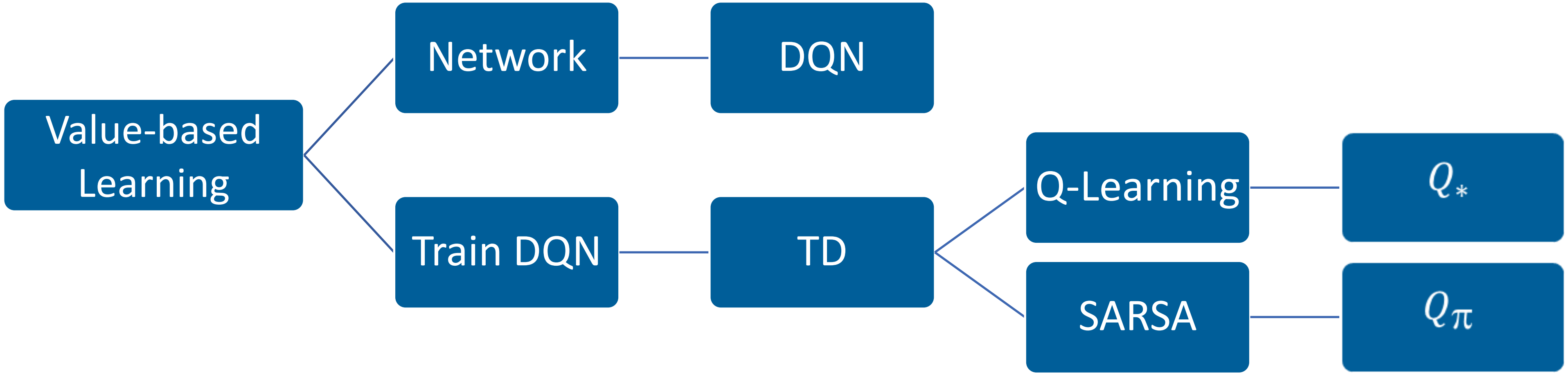
澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

Reinforcement Learning

GongYing 2023.11.24

Contents


Value-based Learning
Network
DQN
Train DQN
TD
Q-Learning
SARSA
Q_*
Q_π

Background

Goal of RL: Maximize sum of rewards.

How: Take the best action, basing on Q*(s,a).

$$a_t = argmax_a Q(s_t, a, \boldsymbol{w})$$

Problem: Get to know Q*(s,a).

DQN

DQN: Nueral network named Deep Q Network, denoted as $Q(s,a;\mathbf{w})$.

Goal: Use $Q(s,a;\mathbf{w})$ to approximate $Q^*(s,a)$.

state → conv → feature → dense → **Q**

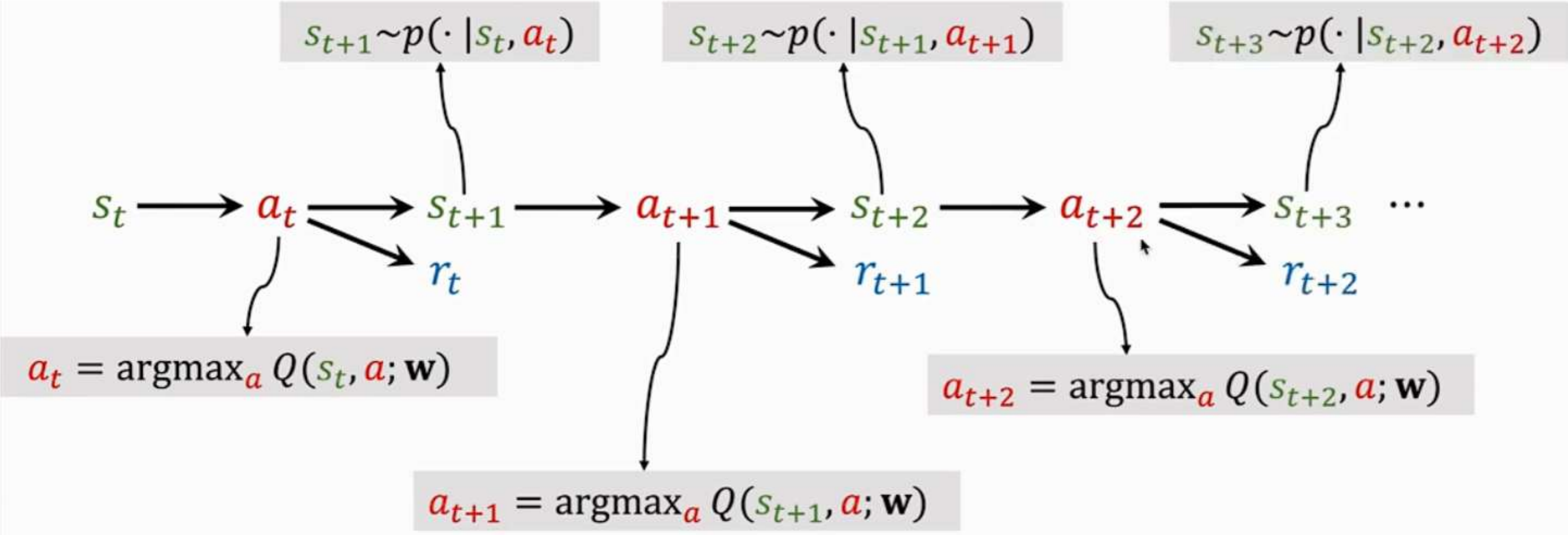
DQN

Example: If action space is {“left”, “up”, “right”}, then **Q** is a

3-dimensional vector $\begin{pmatrix} Q(s_t,\text{"}left\text{"};\boldsymbol{w}_t) \\ Q(s_t,\text{"}up\text{"};\boldsymbol{w}_t) \\ Q(s_t,\text{"}right\text{"};\boldsymbol{w}_t) \end{pmatrix}$

DQN

Apply DQN in a game



TD – Q-Learning

TD: Temporal difference, used to train DQN, including Q-Learning & SARSA.

Q-Learning aims to approximate Q*, while **SARSA** aims to approximate Q_π.

TD – Q-Learning

Observe transition: (s_t, a_t, r_t, s_{t+1})

$$\hat{q} = Q(s_t, a_t; \boldsymbol{w})$$

TD target: $y_t = r_t + \gamma * \max_a Q(s_{t+1}, a; \boldsymbol{w})$ → *Bellman Optimality Equation*

TD error: $\delta_t = Q(s_t, a_t; \boldsymbol{w}) - y_t$

Loss: $\delta_t^2/2$

Gradient: $\frac{\partial \delta_t^2/2}{\partial \boldsymbol{w}} = \delta_t * \frac{\partial Q(s_t, a_t; \boldsymbol{w})}{\partial \boldsymbol{w}}$

Update: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha * \delta_t * \frac{\partial Q(s_t, a_t; \boldsymbol{w})}{\partial \boldsymbol{w}}$

TD – SARSA

SARSA: StateActionRewardStateAction

$$(s_t, a_t, r_t, s_{t+1}, a_{t+1})$$

SARSA is used to approximate Q_π.

TD – SARSA

Observe transition: $(s_t,\ a_t,\ r_t,\ s_{t+1},\ a_{t+1})$

$$\hat{q} = q(s_t, a_t; \boldsymbol{w})$$

TD target: $y_t = r_t + \gamma * \mathrm{q}(s_{t+1},\ a_{t+1}; \mathbf{w})$

TD error: $\delta_t = q(s_t, a_t; \boldsymbol{w}) - y_t$

Loss: $\delta_t^2/2$

Gradient: $\frac{\partial \delta_t^2/2}{\partial \boldsymbol{w}} = \delta_t * \frac{\partial q(s_t, a_t; \boldsymbol{w})}{\partial \boldsymbol{w}}$

Update: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha * \delta_t * \frac{\partial q(s_t, a_t; \boldsymbol{w})}{\partial \boldsymbol{w}}$

On-policy & off-policy

On-policy: Behavior policy is the same with target policy.

Off-policy: Behavior policy is **not** the same with target policy.

Behavior policy: Used to control agent and gain experience.

Target policy: A certain policy that RL aims to find.

For example, Q-Learning is off-policy while SARSA is on-policy.

Experience replay

Meaning: Save interaction records (experience) into arrays and reuse them later to train the agent.

Advantages: Elimination of relevance and faster convergence speed.

Only applies to off-policy solutions.

Thank you.