Stabilizing Q-Learning with Linear Architectures for Provable Efficient Learning

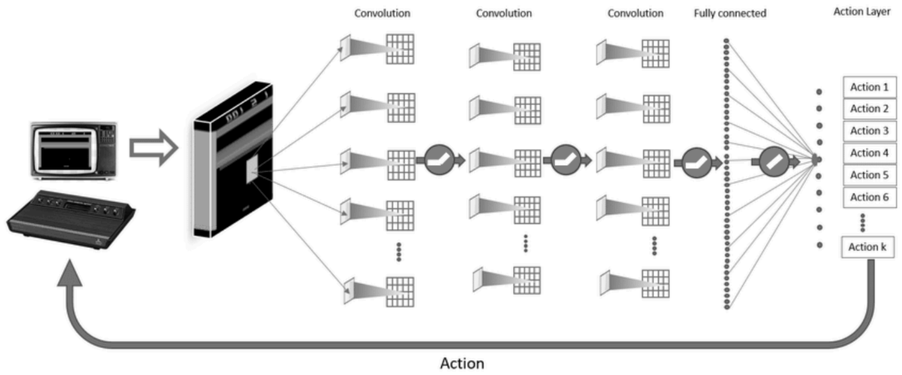
Andrea Zanette, Martin J. Wainwright

Q-Learning

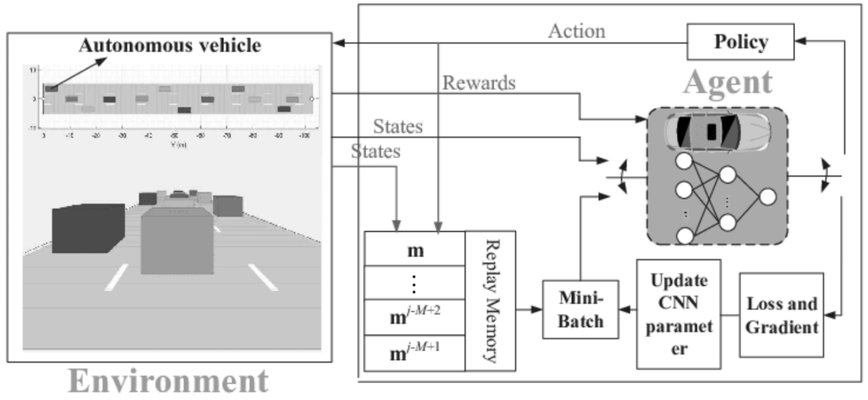
Core RL algorithm

Based on controlling the projected Bellman error

Practically successful even in large applications



Mnih et al, 2015

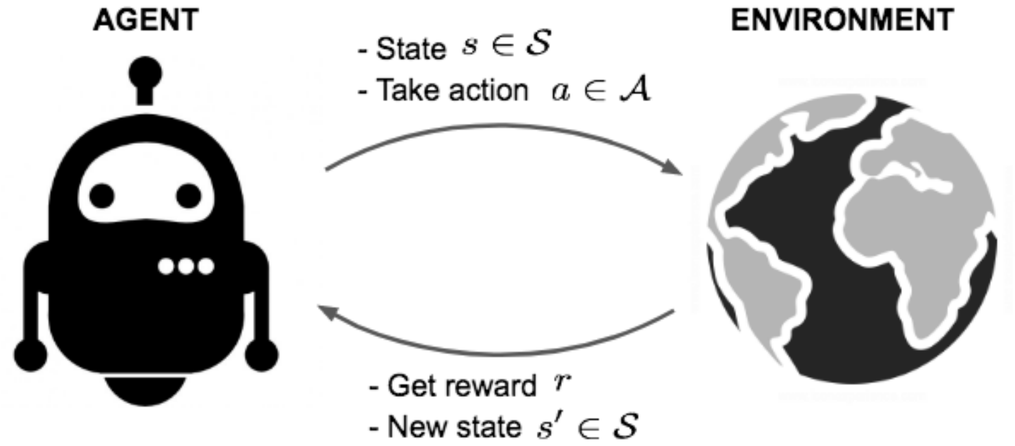


Kiran et al, 2020

Q-Learning with function approximation uses
experience replay, target networks

Reinforcement Learning Exploration

Setting

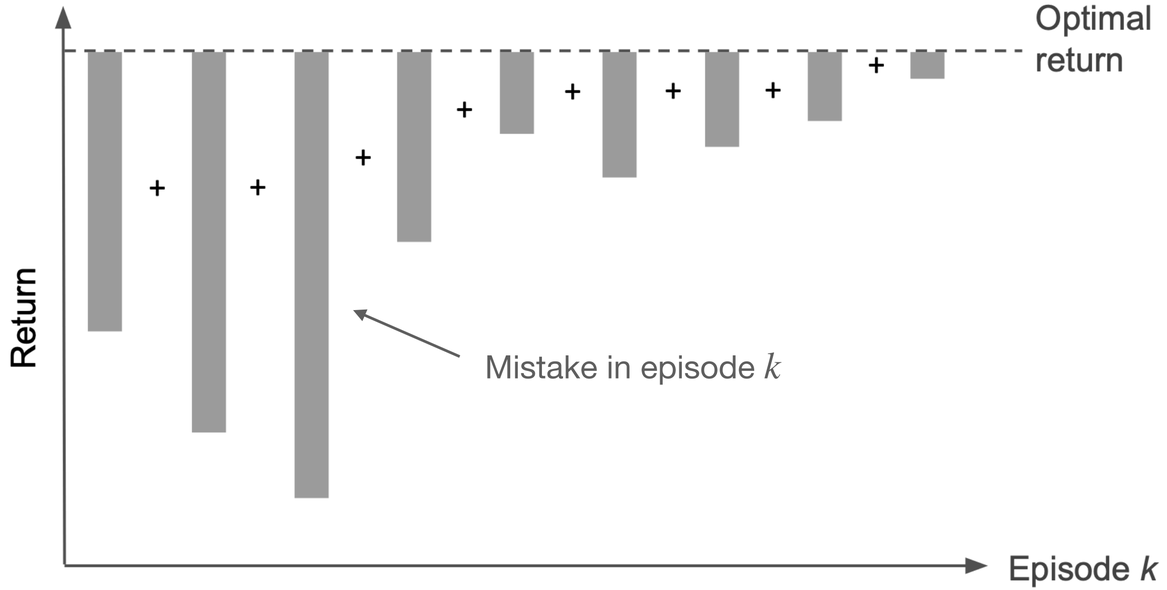


Goal:
Minimize Cumulative Regret

$$\text{Regret} = \sum_{k=1}^{K} V^{\star} - V^{\pi_k}$$

Optimal return

Agent's return



Q-Learning with Linear Function Approximation

Parametric Approximation

$$Q_w(s,a) = \phi(s,a)^\top w$$

Update

$$w^+ = w - \alpha[Q_w(s,a) - r - \gamma \max_{a'} Q_w(s',a')] \nabla Q_w(s,a)$$

Learning rate

Current parameter

TD error

Gradient of Q-function

Issue: not a stochastic gradient update

1. `Gradient' is not really a gradient of any loss
2. Non-stationary (s, a, r, s') data distribution

Stabilizing Q-Learning with Linear Architectures

1) **Exploration Bonus**

2) **Target Network**

Update

$$w^{+} = w - \alpha[Q_w(s,a) - r - \gamma \max_{a'} Q_w^{tar}(s',a')]\nabla Q_w(s,a)$$

4) **Learning Rate**

3) **Replay Mechanism**

Target Network

Target Network
Ensures Q-Learning
Minimizes a Loss

$$\mathscr{L}(w) = \sum_{s,a,r,s'} [Q_w(s,a) - r - \gamma \max_{a'} Q_w^{tar}(s',a')]^2$$

Fixed Bellman backup

Periodic Updates
(slower timescale)

$$Q_w^{tar} \leftarrow Q_w$$

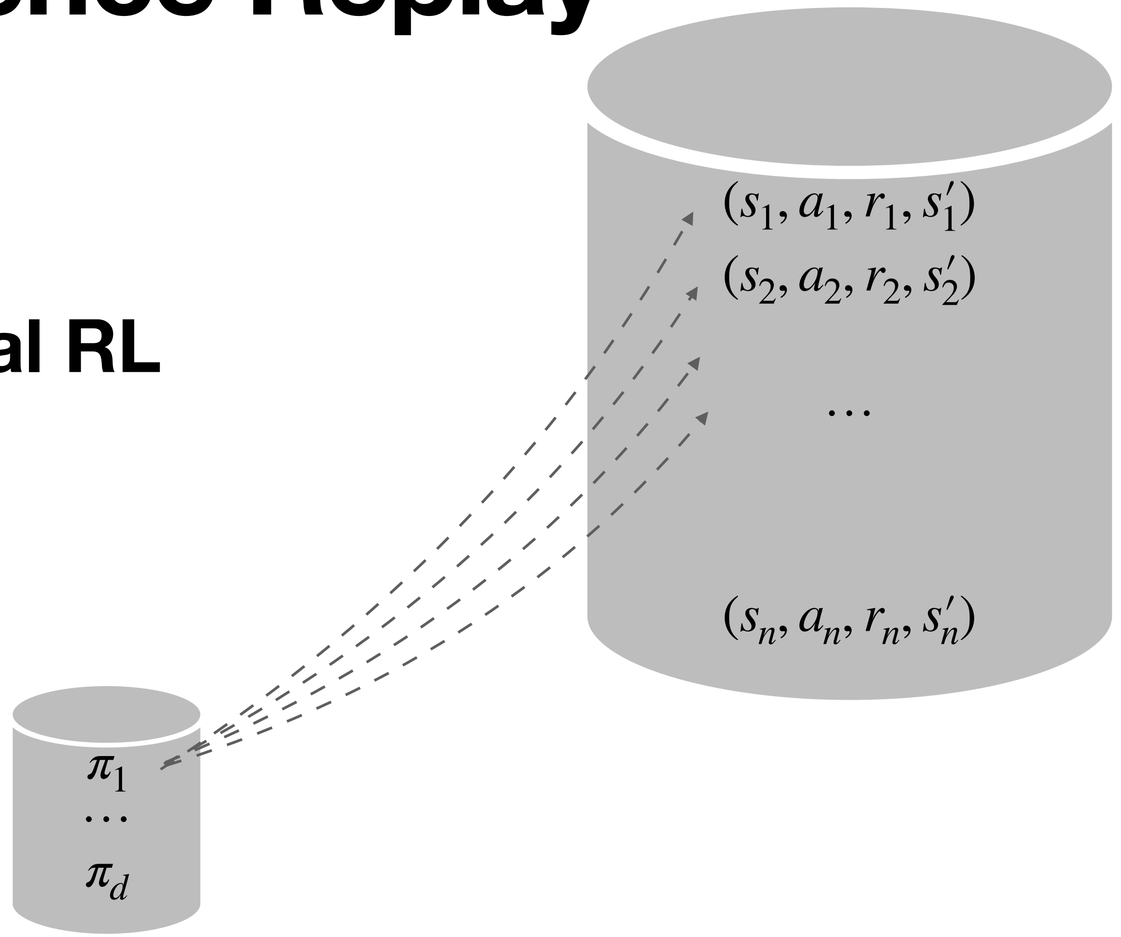
Policy Replay for Experience Replay

Experience replay

essential for state-of-the-art empirical RL (e.g., Mnih et al., 2015)

Policy replay

same effect but limited memory



Main Result

Q-Learning with Linear Function Approximation is provably efficient

Cumulative Regret

$$\widetilde{O}(H^3 d^{3/2}\sqrt{K})$$

Horizon

Feature dimension

Episodes

Total Memory Used

$$\widetilde{O}(d^3 H^2)$$

Per-Step Computations

$$\widetilde{O}(d^2 A)$$

Actions

Thank you for your Attention!