MC Control
SARSA
Q-Learning
Function Approximation for V_π

TD

Prof. Subrahmanya Swamy

Model-Free Setting so far..

- Prediction (To find V_π for a given π)
  - MC First-Visit
  - MC Every-Visit
  - TD
  - N-step TD
  - MC Off-Policy π "b"

GPI. I.P.E.

- "Control" (To find π^* of an MDP)
  - GPI with MC
  - GPI with TD

Model Known: Policy Iteration (PI)

Policy Iteration

$$\pi_0 \rightarrow V_{\pi_0} \rightarrow \pi_1 \rightarrow V_{\pi_1} \rightarrow \pi_2 \rightarrow V_{\pi_2} \rightarrow \ldots$$

Policy Evaluation **Policy Improvement**

Repeat till
$\pi_{k+1} = \pi_k$
$\Downarrow$
$\pi_k = \pi_*$

- *Policy Evaluation:* $V_{k+1}(s) = R_s^{\pi} + \sum_{s'} P_{ss'}^{\pi} V_k(s')$

Repeat till $V_{k+1} = V_k$

- *Policy Improvement:* $\pi_{i+1}(s) := argmax_a \; R_s^a + \sum_{s'} P_{ss'}^a V_{\pi_i}(s')$

Policy Evaluation is stopped when $V_{k+1} \approx V_k$

Model Unknown: Generalized Policy Iteration (GPI)

PI → GPI

V_{π_0}

$$\pi_0 \rightarrow \cancel{Q_{\pi_0}} \rightarrow \pi_1 \rightarrow Q_{\pi_1} \rightarrow \pi_2 \rightarrow Q_{\pi_2} \rightarrow \ldots$$

~~I.P.E~~ MC/TD

Approx. Policy Evaluation

ϵ-greedy Policy Improvement

one Episode

1. Q_{π_0} is evaluated
2. ε-greedy PI "$Q(s,a)$"
3. I.P.E is replaced with MC/TD.

$V_{k+1} \approx V_k$.

When to stop the Approx. Policy Evaluation?

MC: After one episode ✓

TD: After one time-step ✓

MC (every visit) GPI : Pseudo Code

$$\pi_0 \rightarrow Q_{\pi_0} \rightarrow \pi_1 \rightarrow Q_{\pi_1} \rightarrow \pi_2 \rightarrow Q_{\pi_2} \rightarrow \ldots$$

one episode

"**MC-PE**" **for one episode**

ϵ −greedy PI

- Initialize $Q(s,a) = 0, \;\; \forall\,(s,a)$ Q_{π_0}
- **Repeat** for each *episode*:
  - $\pi(s) = \epsilon$-greedy w.r.t. $Q(s,a)$
  - Generate an episode following π : $S_0\,, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots, S_T$

    G_t
  - **Repeat** for each *time-step* ***t*** in the episode:
    - Compute $G_t = \sum_{i=t+1}^{T} \gamma^{i-t-1} R_i$
    - Update $Q(S_t, A_t) = Q(S_t, A_t) + \alpha\,(G_t - Q(S_t, A_t))$
- Output: $\pi^* = greedy\;(Q)$

$Q(S_0, A_0) \approx G_t$

MC

$Q_{new}(S_0, A_0) = Q_{old}(S_0, A_0) + \alpha\,[G_t - Q_{old}(S_0, A_0)]$

SARSA for π^*: Pseudo Code

$$\pi_0 \rightarrow Q_{\pi_0} \rightarrow \pi_1 \rightarrow Q_{\pi_1} \rightarrow \pi_2 \rightarrow Q_{\pi_2} \rightarrow \ldots$$

TD-PE for "one time-step" | **ϵ −greedy PI**

GPT

SARSA

S_t A_t R_{t+1} S_{t+1} A_{t+1}

$S_0, A_0, R_{t+1}, S_{t+1}, A_{t+1}$

- Initialize $Q(s,a) = 0, \ \forall (s,a)$
- **Repeat** for each "*episode*":
  - Initialize S_0 randomly
  - Sample $A_0 \sim$ "ϵ-greedy w.r.t. $Q(S_0, a)$"
  - **Repeat** for each *time-step* ***t*** in the episode:
    - Take action A_t and observe R_{t+1} and S_{t+1}
    - Sample action $A_{t+1} \sim$ "ϵ−greedy w.r.t. $Q(S_{t+1}, a)$"
    - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha\,(\underbrace{R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})}_{q_t} - Q(S_t, A_t))$
- Output: $\pi^* = greedy\ (Q)$

What ϵ and α to use?

π^*

$\varepsilon \longrightarrow$

"k^{th} episode"

$\pi^*(s) = \arg\max_a$ "Q^*"(s,a)

greedy w.r.t Q.

$\varepsilon_k = 1/k$

$\alpha_k = 1/k$

① $\varepsilon_k = \frac{1}{k} \to 0$

$\Rightarrow$

② $\sum_k \alpha_k = \infty$, $\sum_k \alpha_k^2 < \infty$.

$\alpha_k = \frac{1}{k}$

$\sum_k \frac{1}{k} = \infty$ $\quad$ $\sum_k \frac{1}{k^2} < \infty$.

MC

SARSA

TD

GPI.

GPI

Q-learning

Q-Learning

PI $\rightarrow$ GPI.

VI.

- SARSA: Based on Policy Iteration
- Q-Learning: Based on Value Iteration (Asynchronous)
- Value Iteration:

$$Q^*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a V^*(s') \qquad V^*(s) = \max_a Q^*(s,a)$$

$$= R_s^a + \gamma \sum_{s'} P_{ss'}^a \left(\max_{a'} Q^*(s',a')\right)$$

$$\Rightarrow Q_{k+1}(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} Q_k(s',a')$$

$Q_k(s,a)$

Repeat till. $Q_{k+1} = Q_k$.

$$\rightarrow \arg\max_{(s,a)} \left| Q_k(s,a) - Q_{k+1}(s,a) \right|$$

(s,a) ✓ $\qquad \arg\max_s \left| V_{k+1}(s) - V_k(s) \right|$

Q-Learning

Value Iteration.

- SARSA: Based on Policy Iteration
- Q-Learning: Based on Value Iteration (Asynchronous)
- Value Iteration:

$R_s^a = E[R_{t+1} | S_t = s, A_t = a]$

$$Q^*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a V^*(s')$$

$$= R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} Q^*(s',a')$$

- Q-Learning:

$$Q_{new}(S_t, A_t) = Q_{old}(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q_{old}(S_{t+1}, a') - Q_{old}(S_t, A_t))$$

Which state and action pair should be updated in Q-learning? (s, a)

- Random policy
- $\epsilon - greedy\ w.r.t\ Q$

$(S_t, A_t) \rightarrow \ldots\ldots$

Q-Learning: Pseudo Code

- Initialize $Q(s,a) = 0, \forall(s,a)$
- **Repeat** for each episode:
  - Initialize S_0 randomly
  - **Repeat** for each time-step $\boldsymbol{t}$ in the episode:
    - Sample action $A_t \sim \epsilon$-greedy w.r.t. $Q(S_t, a)$
    - Take action A_t and observe R_{t+1} and S_{t+1}
    - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right)$
- Output: $\pi^* =$ "$greedy\ (Q)$"

A_{t+1}

off-policy

Q^*

SARSA Vs Q-Learning

"ϵ"-greedy

Optimal π^*.

- **SARSA**
  - On-Policy
  - Based on Policy Iteration
  - Converges to the best among "ϵ"-soft policies if fixed ϵ = "0.1" is chosen
  - Converges to π^* if ϵ is decreased to zero with time

$\epsilon_k = \frac{1}{k}$

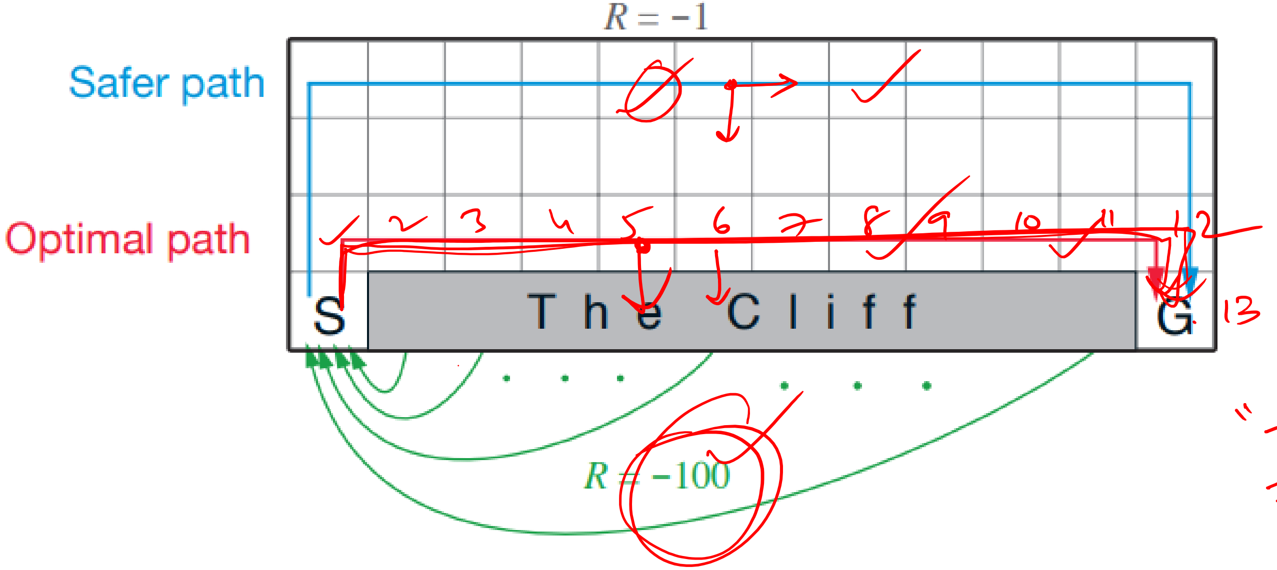
Q^*

ϵ-greedy (S_t, A_t)

- **Q-Learning**
  - Off-Policy
  - Based on Value Iteration
  - Converges to π^* even for fixed ϵ = 0.1

SARSA Vs Q-Learning: Example

$\varepsilon = 0.1$ ε-greedy(Q)



- SARSA: Learns Safer path
- Q-Learning: Learns Optimal path

ε-greedy

- **Aim:** To go in the shortest path from the **S**tart state to the **G**oal state
- Reward of -100 for transition into the Cliff region
- Reward of -1 for every other transition

SARSA Vs Q-Learning: Example

ε-greedy

- SARSA: Learns Safer path
- Q-Learning: Learns Optimal path

- Why does Q-learning have a worse reward, although it learned the optimal policy?

Function Approximation for Large State spaces

Prof. Subrahmanya Swamy

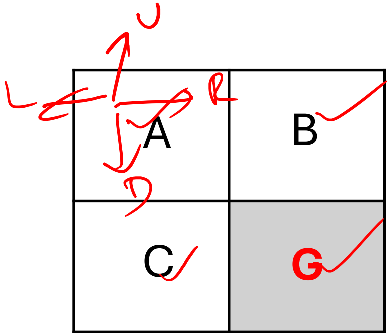
Large State Spaces in Real-time Applications

Go game

$\sim 10^{170}$ states > # Atoms in the universe

Tabular Methods

- Small state/action space: $Q(s,a)$ Table maintained for each (s,a) explicitly



Q-Table has 4 states x 4 actions = 16 entries

→ Q("s", a) =

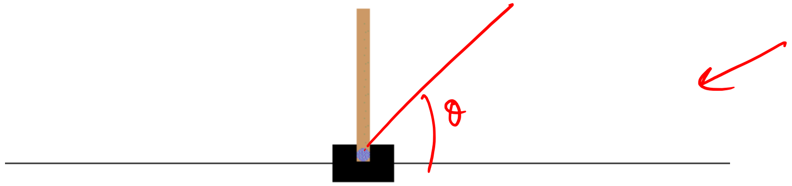
	U	L	D	R
A				
B				
C				
G				

16 Entry table. 10^{170}

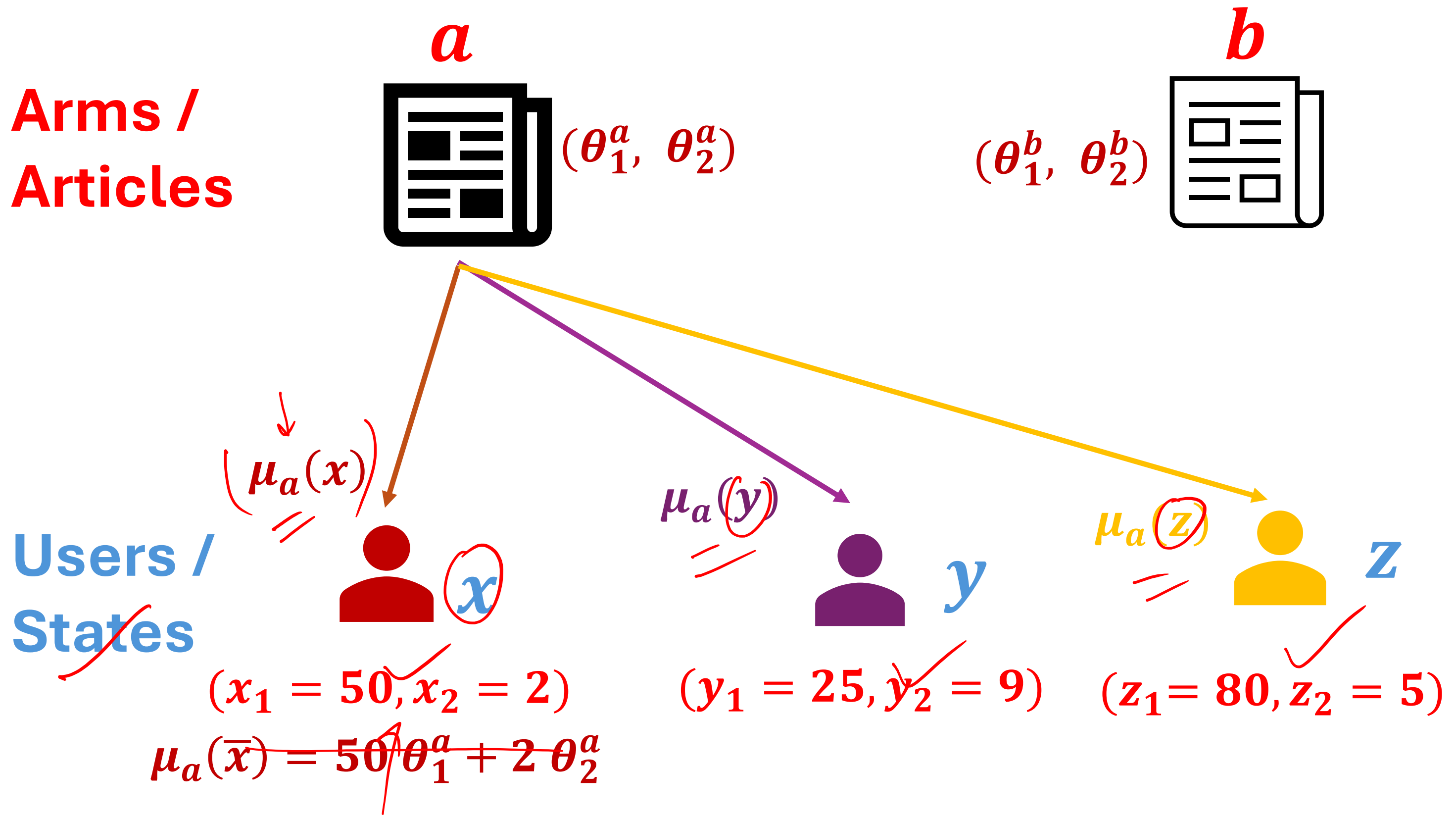
- Large state/action spaces: Not feasible!
- Continuous state/action spaces: Not feasible!

Fn Approx

Contextual Bandits



Bandits + Supervised Learning



- User (state) represented by features such as age, income
  $\bar{x} = (x_1, x_2)$
- Model the expected reward for user $\bar{x}$ for pulling arm a as
  $\mu_a(\bar{x}) = \theta_1^a x_1 + \theta_2^a x_2$

Features and Function Approximation

features

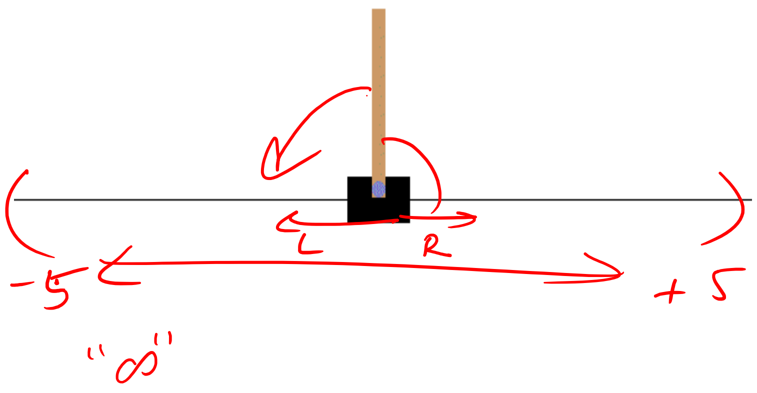
Cartpole: The goal is to balance the pole by applying forces in the left and right direction

State Features $s = (s_1, s_2, s_3, s_4)$

$V_\pi(s)$

$Q(s,a)$

State $\bar{s}$	Min	Max
Cart Position s_1	-4.8	4.8
Cart Velocity s_2	-Inf	Inf
Pole Angle s_3	~ -0.418 rad (-24°)	~ 0.418 rad (24°)
Pole Angular Velocity s_4	-Inf	Inf



Value fn Approx $V_\theta(s) \approx s_1\theta_1 + s_2\theta_2 + s_3\theta_3 + s_4\theta_4$

$Q(s,a)$

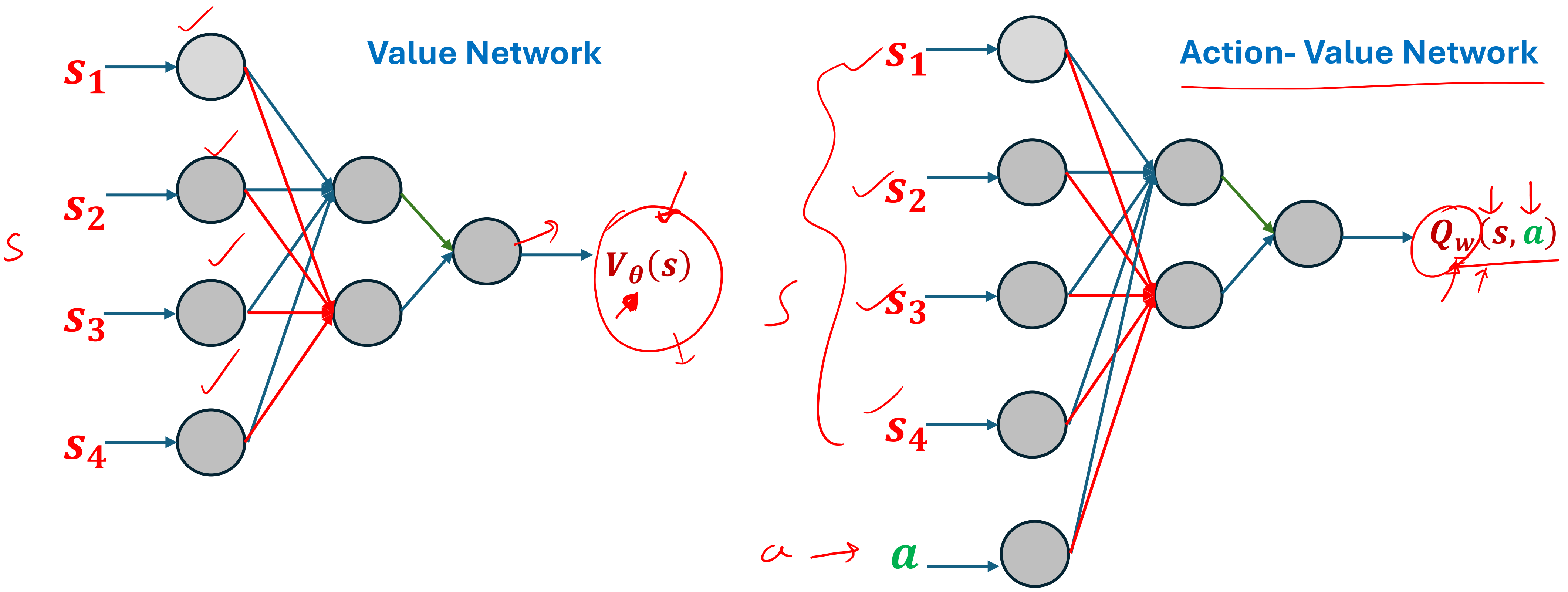
a Action Features

0:	Push the cart to the LEFT
1:	Push the cart to the RIGHT

Linear "w"

Q fn Approx $Q_w(s,a) \approx s_1 w_1 + s_2 w_2 + s_3 w_3 + s_4 w_4 + a\, w_5$

Non-Linear Function Approximation



Neural Network-based Function Approximation

How to find the weights of the neural network?

Function approx. for V_π

π

$$\boxed{\theta^* = \text{argmin}_\theta\ \mathbb{E}_\pi\left[\left(V_\pi(s) - V_\theta(s)\right)^2\right]}$$

S_3

$S_1 \rightarrow V_\pi(S_1)$

$S_2 \rightarrow V_\pi(S_2)$

$\theta_{new} = \theta_{old} - \alpha \cdot \frac{\partial L}{\partial \theta}$

Stochastic Gradient descent:

$$\theta_{new} = \theta_{old} + 2\,\alpha\,\left(V_\pi(s) - V_\theta(s)\right) \nabla V_\theta(s)$$

RL

Challenge:

- "$V_\pi(s)$ unknown"
- No training data available

MC Function approx. for V_π

MC|TD

$$\boxed{\theta^* = \operatorname{argmin}_\theta \ \mathbb{E}_\pi\left[\left(V_\pi(s) - V_\theta(s)\right)^2\right]}$$

$V_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right],$

$s \longrightarrow S_T \rightarrow$ "G_t"

$$\theta_{new} = \theta_{old} + 2\,\alpha\left(V_\pi(s) - V_\theta(s)\right)\nabla V_\theta(s)$$

SGD

$V_\pi(s) \approx G_t$ starting from state s

$$\text{“}\theta_{new}\text{''} = \theta_{old} + 2\,\alpha\left(G_t - V_\theta(s)\right)\nabla V_\theta(s)$$

TD Function approx. for V_π

"TD"

π Tab

$R_{t+1} + \gamma V_\theta(S_{t+1})$

$$\theta_{new} = \theta_{old} + 2\alpha\left(V_\pi(s) - V_\theta(s)\right)\nabla V_\theta \qquad \text{SGD}$$

$\underline{\text{MC}}$: $\underline{\underline{G_t}}$

Q_π

$=$

Q^*.

$$V_\pi(s) = \underset{\pi}{E}\left[G_t \mid S_t = s\right]$$

$$= E_\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right]$$

$$\approx R_{t+1} + \gamma \underline{\underline{V_\pi}}(S_{t+1})$$

$$\approx \boxed{R_{t+1} + \gamma V_\theta(S_{t+1})}$$