MC Control ✓
SARSA
Q-Learning
Function Approx for V_π

GPI

Prof. Subrahmanya Swamy

PI

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

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

Model Known: Policy Iteration (PI)

GPI

Policy Iteration

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

ML/TD

Q_{π}

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

$$\pi_{i+1}(s) = \underset{a}{argmax}\; Q^*(s, a)$$

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

GPI

Model Unknown: Generalized Policy Iteration (GPI)

IPE π_0

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

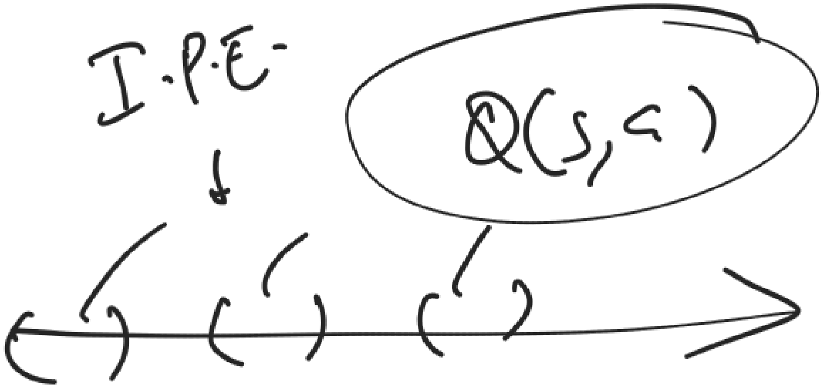
Approx. Policy Evaluation

ϵ −greedy Policy Improvement

When to stop the Approx. Policy Evaluation?

MC: After one episode

TD: After one time-step



MC (every visit) GPI : Pseudo Code

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

MC-PE for one episode ↑ (to $\pi_0 \rightarrow Q_{\pi_0}$)

ϵ −greedy PI ↑ (to $Q_{\pi_0} \rightarrow \pi_1$)

```
• Initialize $Q(s,a) = 0,\ \forall (s,a)$
• Repeat for each episode:
    • $\pi(s) = \epsilon$-greedy w.r.t. $Q(s,a)$
    • Generate an episode following $\pi : S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots, S_T$
    • 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: $\pi^* = greedy\ (Q)$
```

f.V.

G_0

$(A, left)$ ↑

$(A, left)$

π

new sample $Q(S_0, A_0) \approx G_0$

↓

$Q_{new}(S_0, A_0)$

$= Q_{old}(S_0, A_0)$

$+ \alpha \left(G_0 - Q_{old}(S_0, A_0) \right)$

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

Q

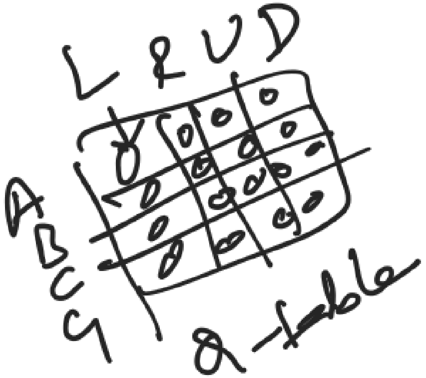
```
- Initialize $Q(s,a) = 0, \ \forall (s,a)$
- **Repeat** for each *episode*:
    - Initialize $S_0$ randomly
    - Sample $A_0 \sim \epsilon$-greedy w.r.t. $Q(S_0, a)$
    - **Repeat** for each *time-step* $t$ in the episode:
        - Take action $A_t$ and observe $R_{t+1}$ and $S_{t+1}$
       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

What ϵ and α to use?

- Sufficient conditions on α and ϵ to ensure MC/SARSA-based GPI converges to π^*
- k – episode number
- $\epsilon_k \rightarrow 0$ as k $\rightarrow \infty$
- $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$

Example:

In the k^{th} episode, $\boxed{\epsilon_k = \frac{1}{k} \text{ and } \alpha_k = \frac{1}{k}}$ will satisfy the sufficient conditions.



$$V_\pi(S) = E_\pi\left[G_t \mid S_t = S\right]$$

$$\approx G_t$$

$$E\left[\max_{a'} Q^k(S_{t+1}, a') \,\middle|\, \begin{matrix} S_t = s \\ A_t = a \end{matrix}\right]$$

$$\approx$$

Q-Learning

$E[f(x)] \simeq f(x_i)$

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

$$E[R_{t+1} \mid S_t = s, A_t = a] \qquad Q^*_{k+1}(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a V^*(s')$$

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

- 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?

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

P. Iteration

$$\pi_0 \xrightarrow{P.E.} V_{\pi_0} \xrightarrow{P.I.} \pi_1$$

V. Iteration

$$\boxed{V^*(s) = \max_a Q^*(s,a)}$$

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

$$V_{k+1}(s) = \max_a R_s^a + \gamma \sum_{s'} P_{ss'}^a V_k(s')$$

$V_{k+1} = V_k$

$V^* =$

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 $t$ 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 $S_{t+1}$
        • $Q(S_t, A_t) = Q(S_t, A_t) + \alpha\,(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
• Output: $\pi^* = greedy\,(Q)$
```

SARSA Vs Q-Learning

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

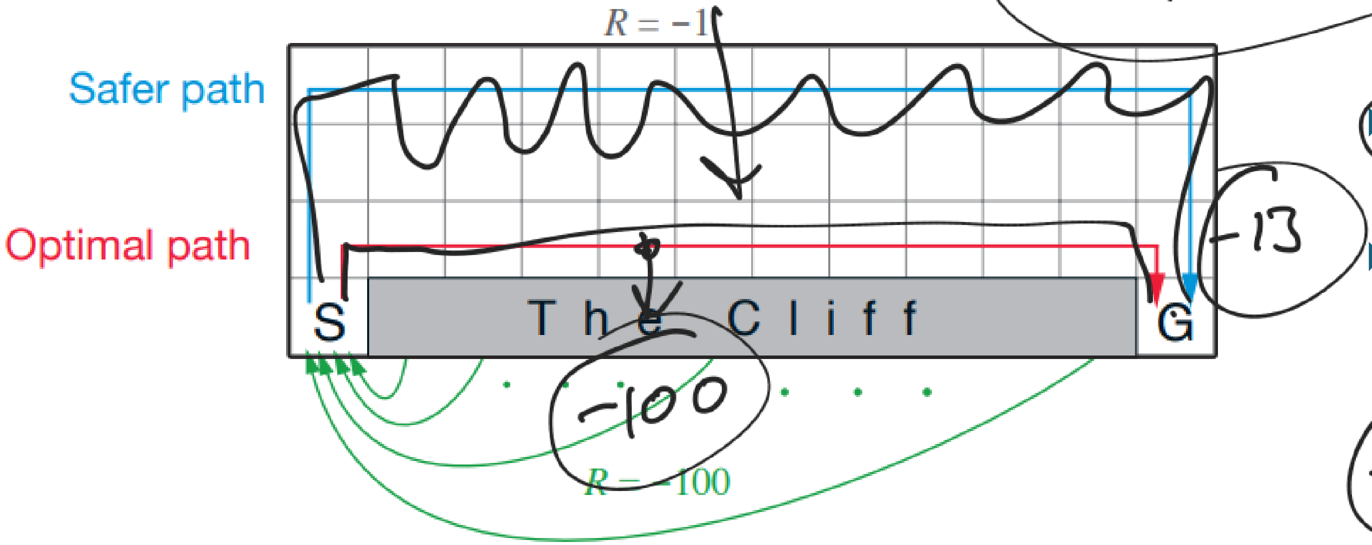
$\boxed{\varepsilon = 0.1}$

$\boxed{\varepsilon_k = \frac{1}{k}}$ $\varepsilon_k \to 0$ as $k \to \infty$

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

SARSA Vs Q-Learning: Example

$\varepsilon = 0.1$ ✓



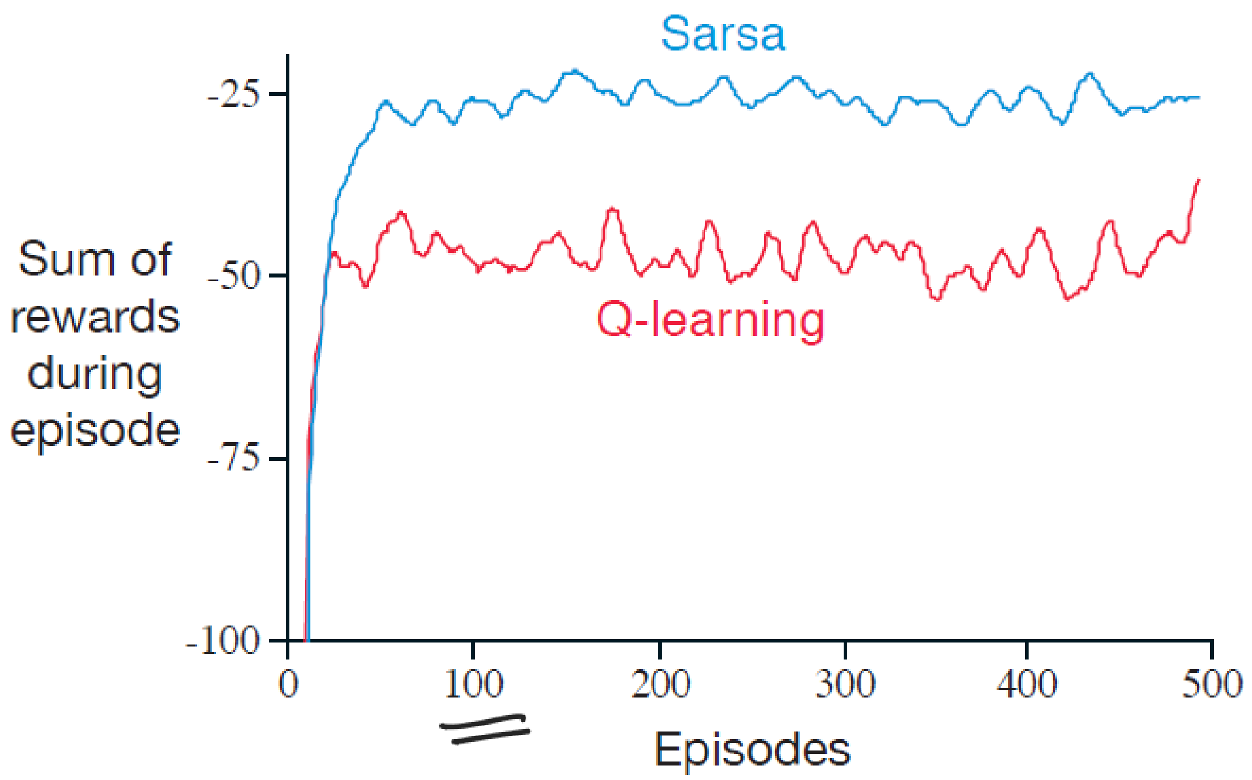
- SARSA: Learns Safer path
- Q-Learning: Learns Optimal path

-13
Red.

-17
Blue

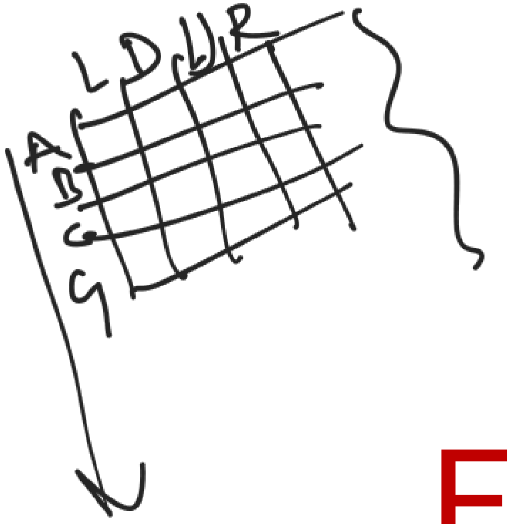
- **Aim:** To go in the shortest path from the **S**tart state to the **G**oal state
- Reward of -100 for transition into the Cliff region
- Reward of -1 for every other transition

SARSA Vs Q-Learning: Example



- 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

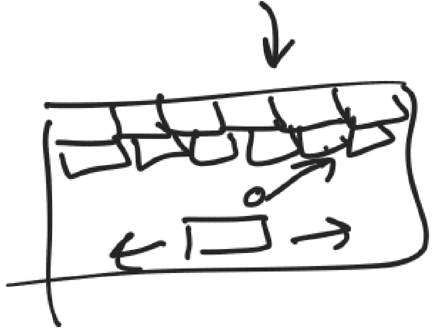
Prof. Subrahmanya Swamy

Large State Spaces in Real-time Applications

Go game

Deep

Alpha Go

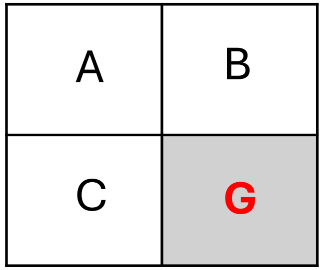


100 x 100

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

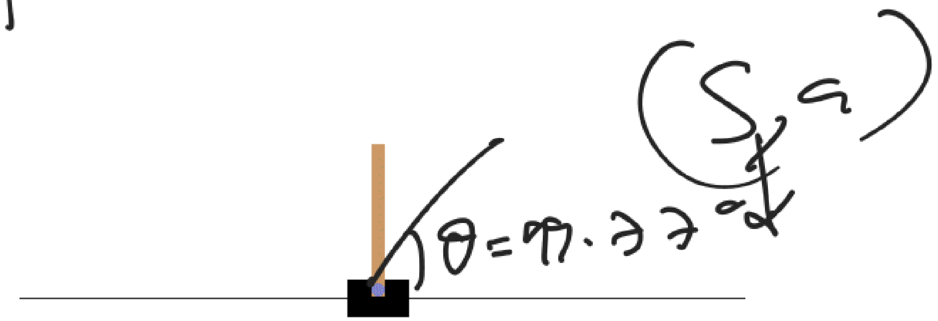
Tabular Methods

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

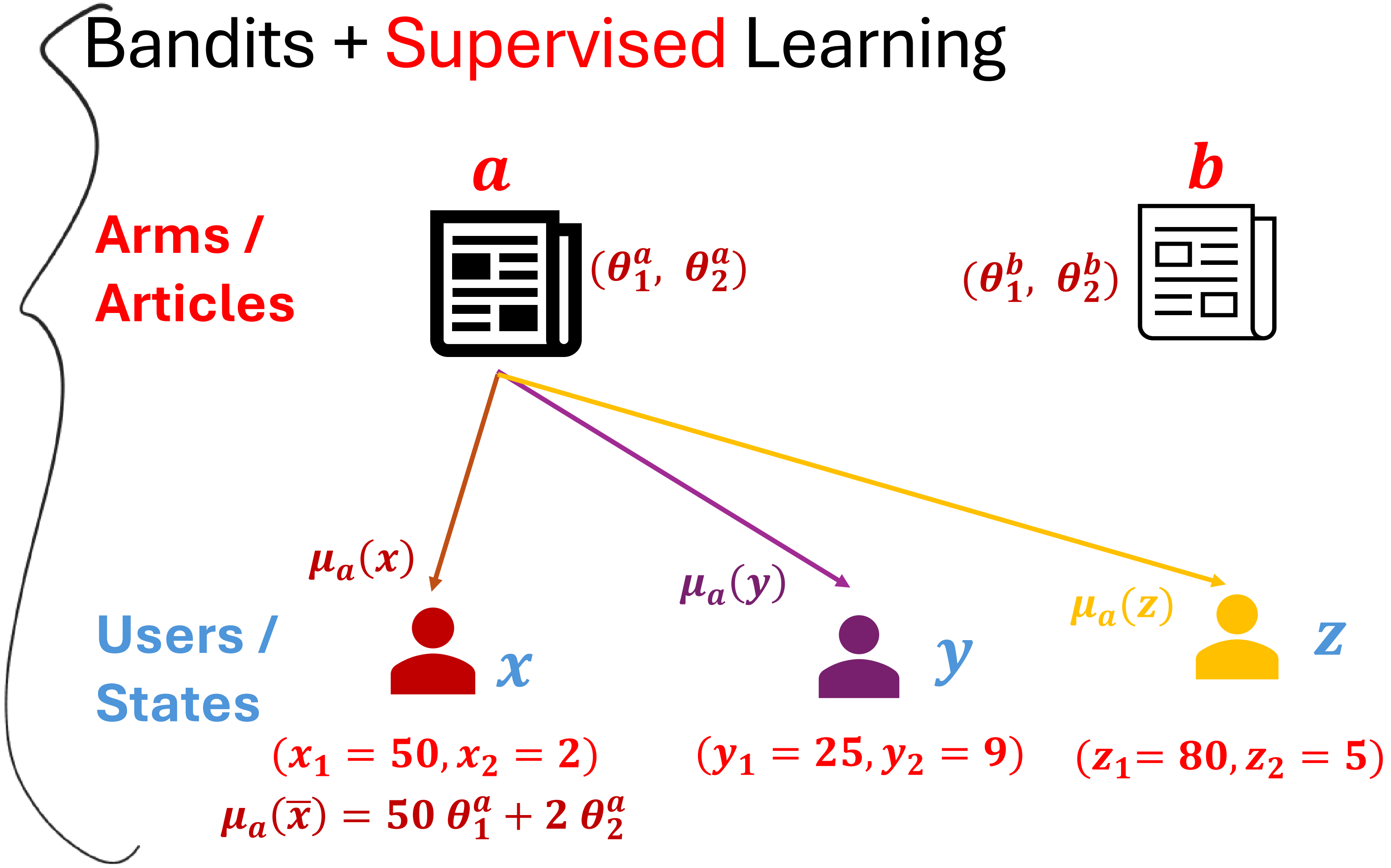


Q-Table has 4 states x 4 actions = 16 entries

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



Bandits + Supervised Learning



Arms / Articles

a (θ_1^a, θ_2^a)

b (θ_1^b, θ_2^b)

$\mu_a(x)$ $\mu_a(y)$ $\mu_a(z)$

Users / States

x y z

$(x_1 = 50, x_2 = 2)$ $(y_1 = 25, y_2 = 9)$ $(z_1 = 80, z_2 = 5)$

$\mu_a(\bar{x}) = 50\ \theta_1^a + 2\ \theta_2^a$

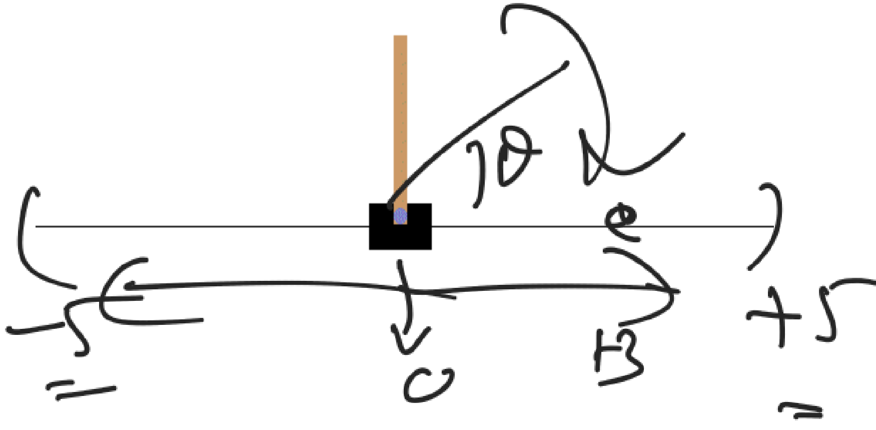
- User (state) represented by features such as age, income $\bar{x} = (x_1, x_2)$
- Model the expected reward for user $\bar{x}$ for pulling arm a as $\mu_a(\bar{x}) = \theta_1^a x_1 + \theta_2^a x_2$

Features and Function Approximation

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

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

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



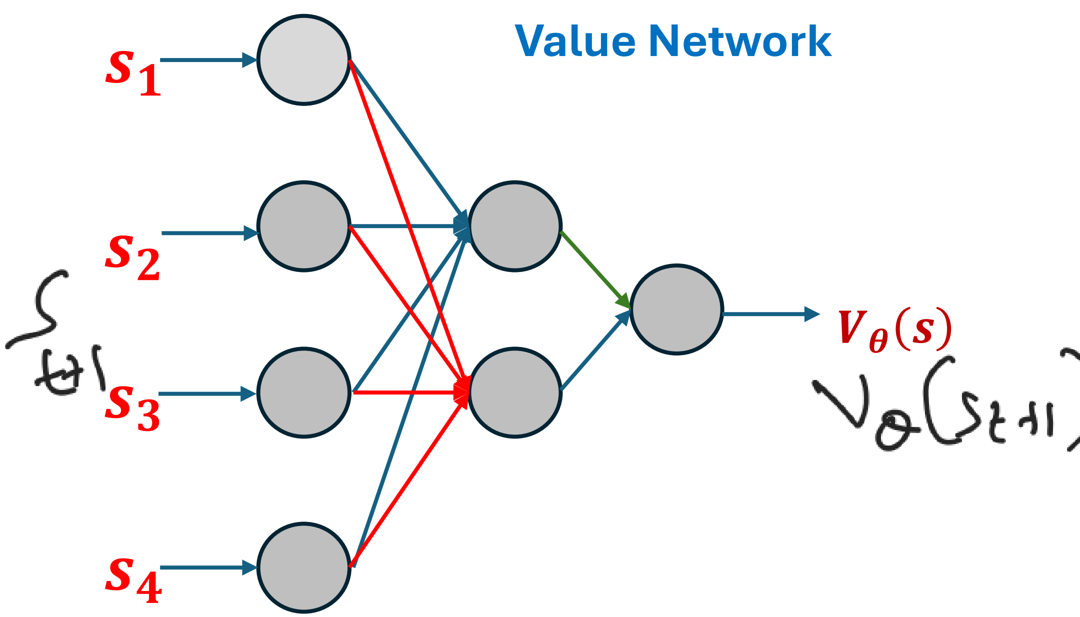
Value fn Approx $V_\theta(s) \approx s_1\theta_1 + s_2\theta_2 + s_3\theta_3 + s_4\theta_4$

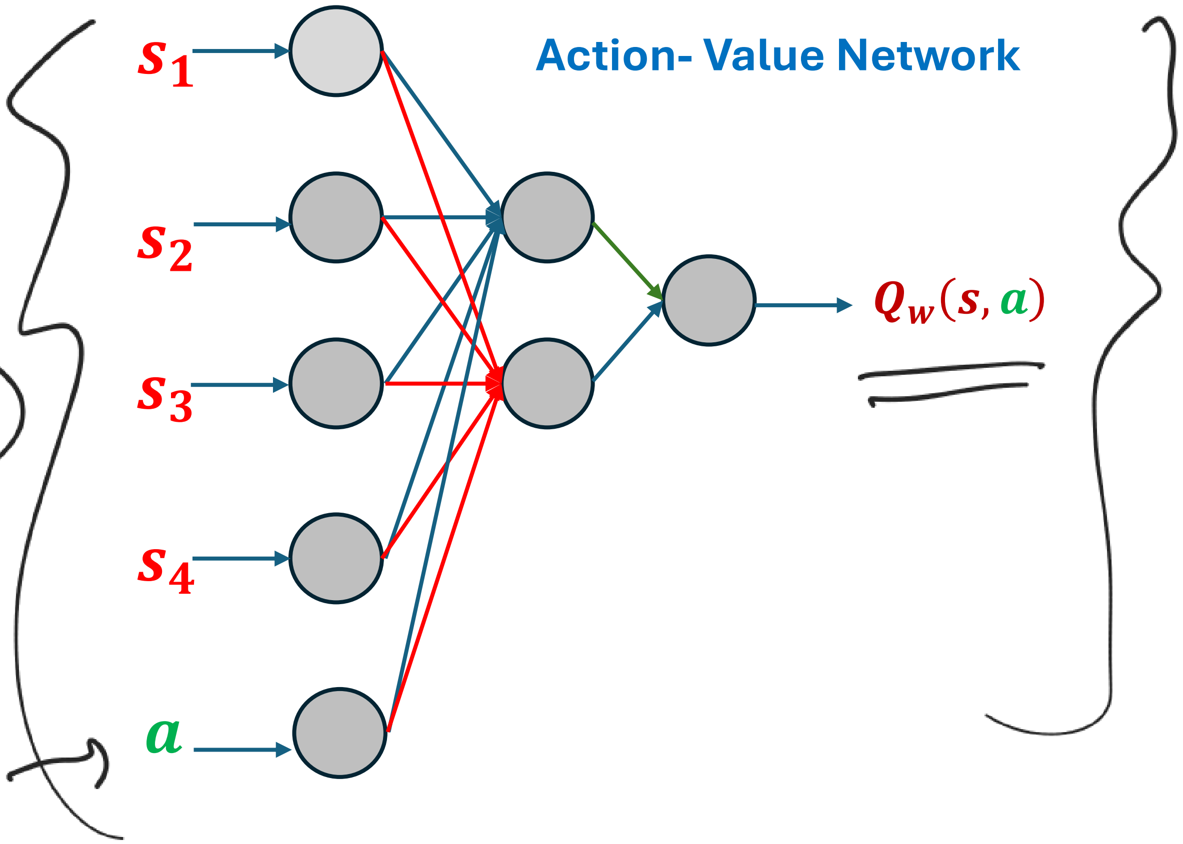
a Action Features

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

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

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

$S_1 \rightarrow V_\pi(S_1)$

$S_2 \rightarrow V_\pi(S_2)$?

Stochastic Gradient descent:

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

Challenge:

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

MC Function approx. for V_π

$(\theta_\theta - \theta^*)^2$

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

$E[G_t | S_t = s]$

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

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

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

TD Function approx. for V_π

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

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

$V_\pi(s) \sim R_{t+1} + \gamma V_\pi(S_{t+1})$

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

S_t R_{t+1}

$$\theta_{new} = \theta_{old} + 2\,\alpha\left(R_{t+1} + \gamma\,\underline{\underline{V_\theta(S_{t+1})}} - V_\theta(s)\right)\nabla V_\theta(s)$$