On-policy Control with Approximation

Mohammad **Dehghani**

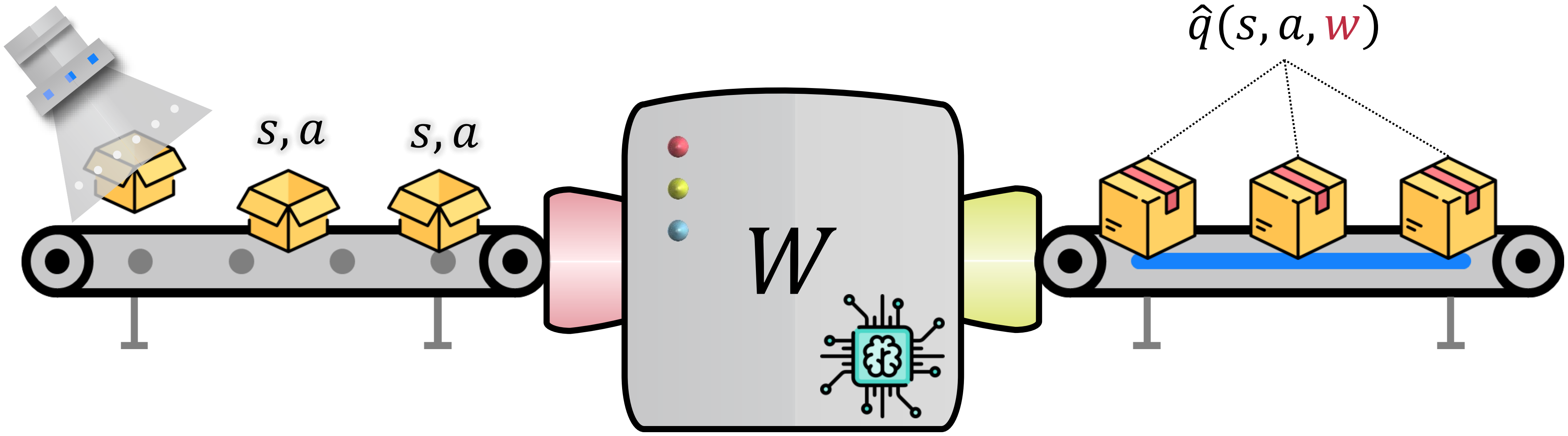
Mechanical and Industrial Engineering Department

Northeastern University

Apr-20

Introduction

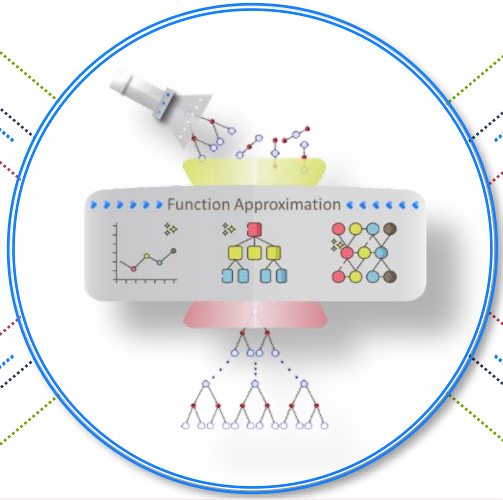
Action values function approximation: Graphical representation



$$\hat{q}(s, a, w) \approx q_\pi(s, a)$$

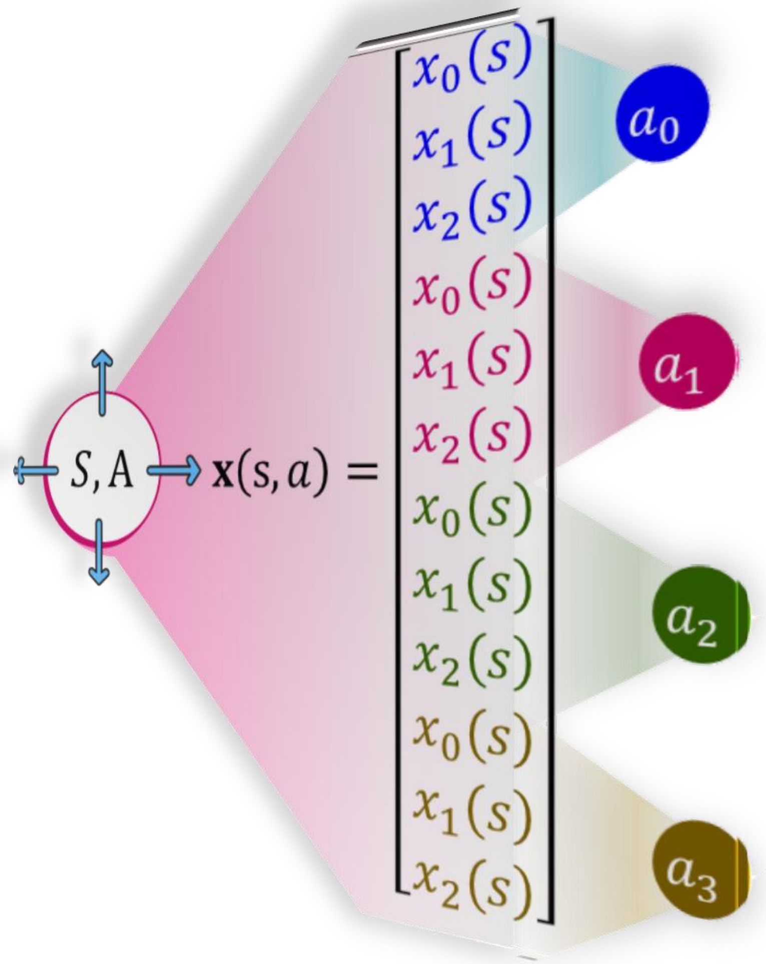
Action values function approximation: Mathematical representation

$$v_\pi(s) \approx \hat{v}(s, w) \doteq \mathbf{w}^{\mathrm{T}}\mathbf{x}(s)$$



$$q_\pi(s, a) \approx \hat{q}(s, a, w) \doteq \mathbf{w}^{\mathrm{T}}\mathbf{x}(s, a)$$

Action Values
Feature
Representation



Action Values Feature Representation

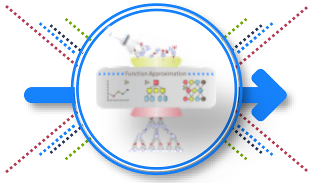
Value functions: Tabular setting

State values

State	Values
S_1	$v(S_1)$
S_2	$v(S_2)$
S_3	$v(S_3)$
S_4	$v(S_4)$
$\vdots$	$\vdots$

Action values

State	Action	Values
S_1	A_1	$q(S_1, A_1)$
	A_2	$q(S_1, A_2)$
	A_3	$q(S_1, A_3)$
	A_4	$q(S_1, A_4)$
S_2	A_1	$q(S_2, A_1)$
	A_2	$q(S_2, A_2)$
	A_3	$q(S_2, A_3)$
	A_4	$q(S_2, A_4)$
$\vdots$	$\vdots$	$\vdots$



Approximation

Values
$\hat{q}(S_1, A_1, \mathbf{w})$
$\hat{q}(S_1, A_2, \mathbf{w})$
$\hat{q}(S_1, A_3, \mathbf{w})$
$\hat{q}(S_1, A_4, \mathbf{w})$
$\hat{q}(S_2, A_1, \mathbf{w})$
$\hat{q}(S_2, A_2, \mathbf{w})$
$\hat{q}(S_2, A_3, \mathbf{w})$
$\hat{q}(S_2, A_4, \mathbf{w})$
$\vdots$

Action Values Feature Representation

Stacking the features

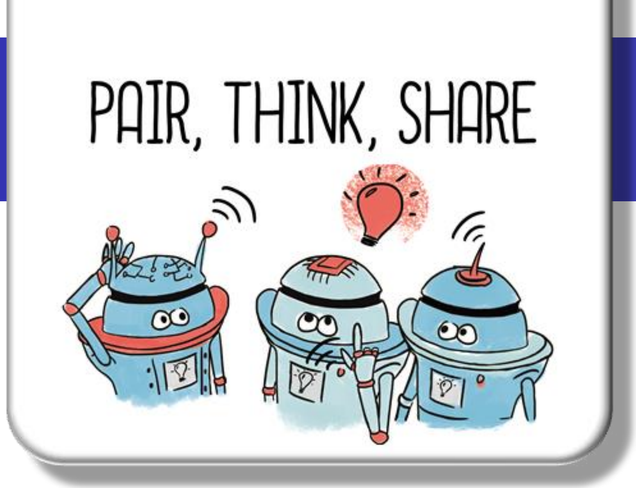
S

$$\mathbf{x}(s) = \begin{bmatrix} x_0(s) \\ x_1(s) \\ x_2(s) \end{bmatrix}$$

S, A

$$\mathbf{x}(s,a) = \begin{bmatrix} x_0(s) \\ x_1(s) \\ x_2(s) \\ x_0(s) \\ x_1(s) \\ x_2(s) \\ x_0(s) \\ x_1(s) \\ x_2(s) \\ x_0(s) \\ x_1(s) \\ x_2(s) \end{bmatrix} \begin{matrix} \\ a_0 \\ \\ \\ a_1 \\ \\ \\ a_2 \\ \\ \\ a_3 \\ \\ \end{matrix}$$

$$\mathbf{x}(s,a_0) = \begin{bmatrix} x_0(s) \\ x_1(s) \\ x_2(s) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathrm{x}(s,a_1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ x_0(s) \\ x_1(s) \\ x_2(s) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{x}(s,a_2) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ x_0(s) \\ x_1(s) \\ x_2(s) \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{x}(s,a_4) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ x_0(s) \\ x_1(s) \\ x_2(s) \end{bmatrix}$$



1

S_0 $\mathbf{x}(s_0) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

$\mathcal{A}(s)$ $\mathcal{A}(s) = \{a_0, a_1, a_2, a_3\}$

$$\mathbf{w} = \begin{bmatrix} 0.20 \\ -0.7 \\ 0.87 \\ 0.00 \\ 0.33 \\ 0.02 \\ -0.19 \\ -2.11 \\ 0.08 \\ 0.00 \\ 0.98 \\ 1.80 \end{bmatrix}$$

Stacking the features

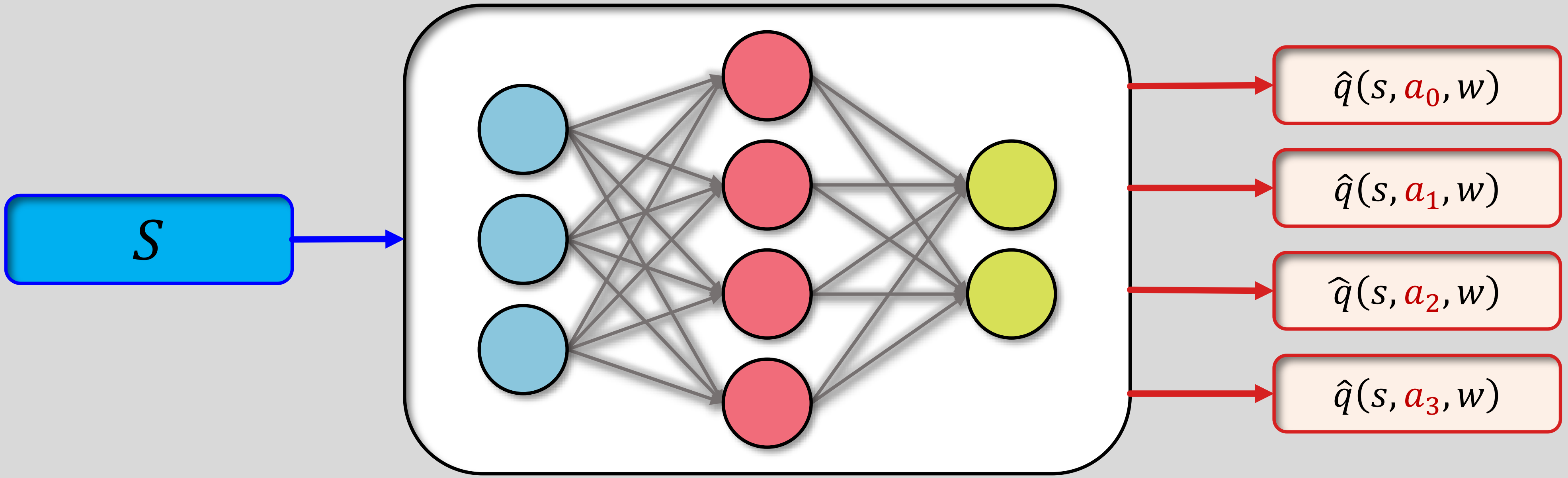
- Feature representation has to represent all actions for a given state. Using stacking the features, we can extend the original state feature vector and activate the features corresponding to that action.

Calculate action values

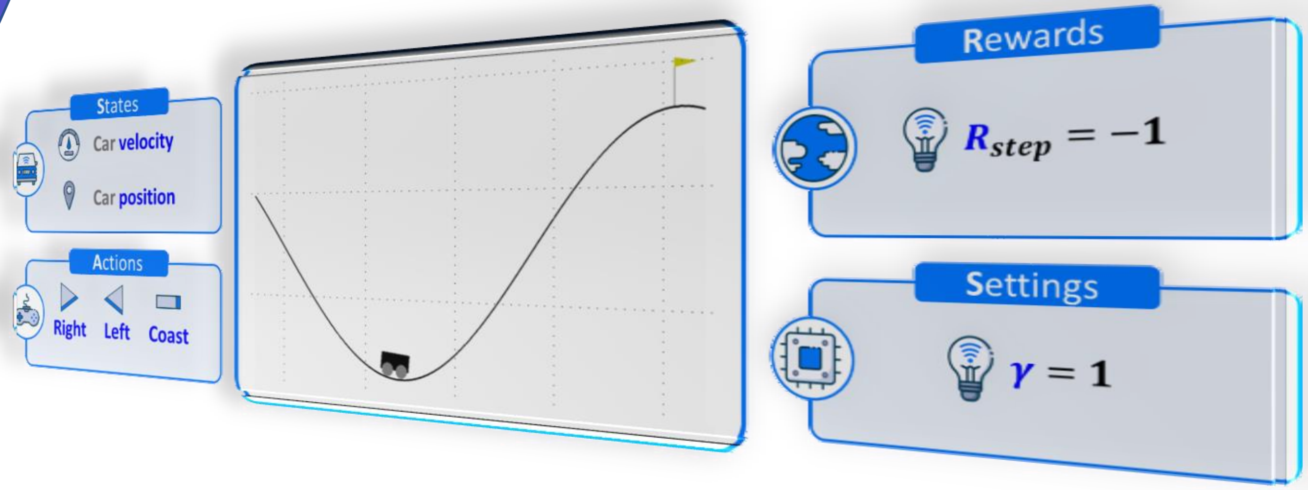
Form a stacking feature vector for each action, and then calculate action values $\hat{q}(s, a, w) \doteq \mathbf{w}^{\mathrm{T}}\mathbf{x}(s, a)$

Action Values Feature Representation

Neural Networks for action



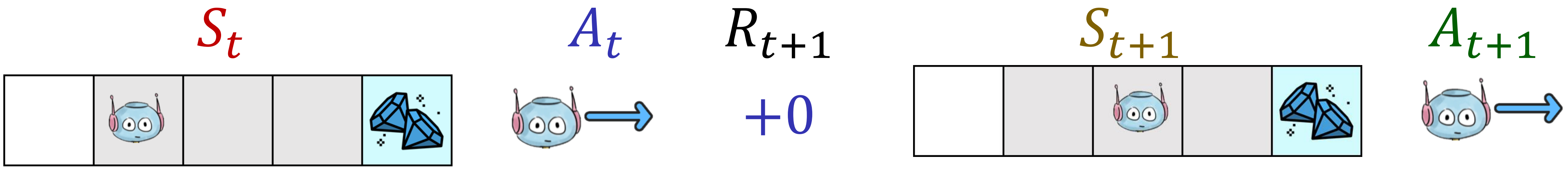
Episodic Sarsa with Fn Approx.

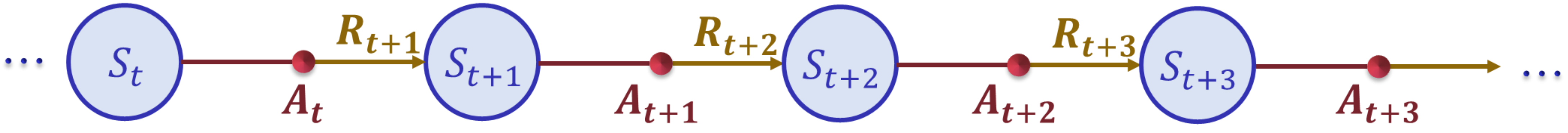


Episodic Sarsa with Function Approximation

Review: Sarsa Definition

S_t	A_t	R_{t+1}	S_{t+1}	A_{t+1}
S	A	R	S	A





Sarsa Update Equation

- **The theorems assuring the convergence of state values under TD(0) also apply to the corresponding algorithm for action values**

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

  - This update is done after every transition from a nonterminal state S_t

Episodic Sarsa with Function Approximation

Pseudocode: Semi-gradient Sarsa

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

1 Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

2 Loop for each episode:
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'

3 If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[R - \hat{q}(S, A, \mathbf{w}) \right] \nabla \hat{q}(S, A, \mathbf{w})$
      Go to next episode

4 Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[R + \gamma \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w}) \right] \nabla \hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

1 **Initialization**

2 **Sarsa Loop**

3 **Terminal State**

4 **Sarsa Update**

Episodic Sarsa with Function Approximation

Example: Mountain Car

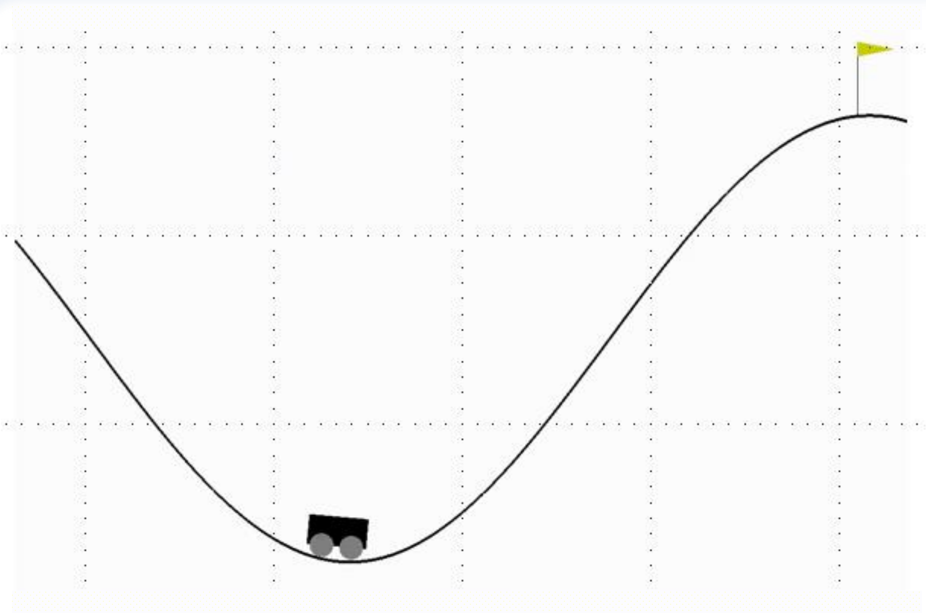
States

Car **velocity**

Car **position**

Actions

Right **Left** **Coast**



Rewards

$R_{step} = -1$

Settings

$\gamma = 1$

- **The difficulty is that gravity is stronger than the car's engine**
  - The only solution is to first move away from the goal and up the opposite slope on the left.
  - Then, by applying full throttle the car can build up enough inertia to carry it up the steep slope even though it is slowing down the whole way.

Feature Representation: Mountain Car

States

Car position

$$x_{t+1} \doteq bound[x_t + \dot{x}_{t+1}]$$

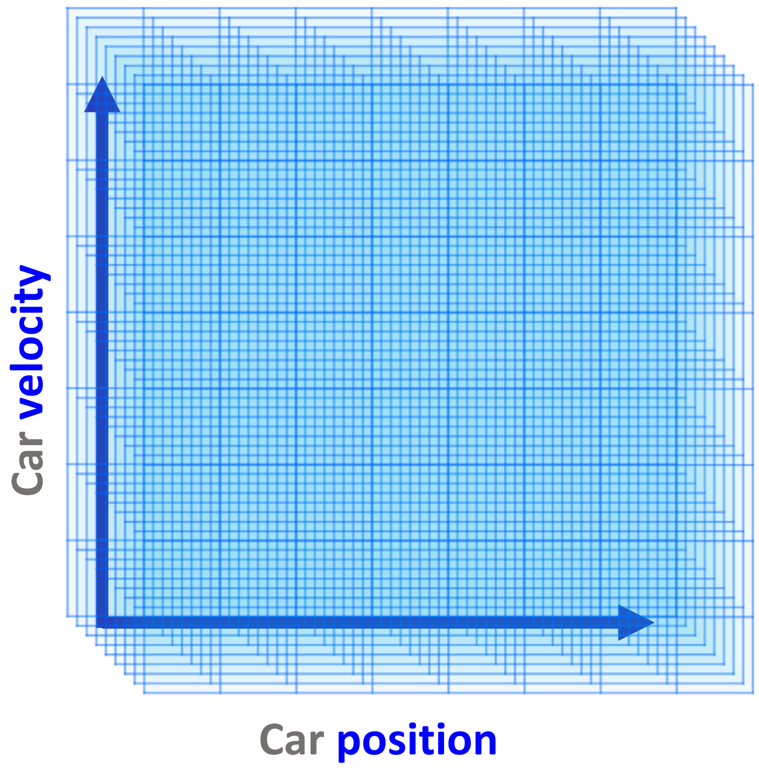
$$-1.2 \leq x_{t+1} \leq 0.5$$

$$x_0 \in [-0.6, -0.4]$$

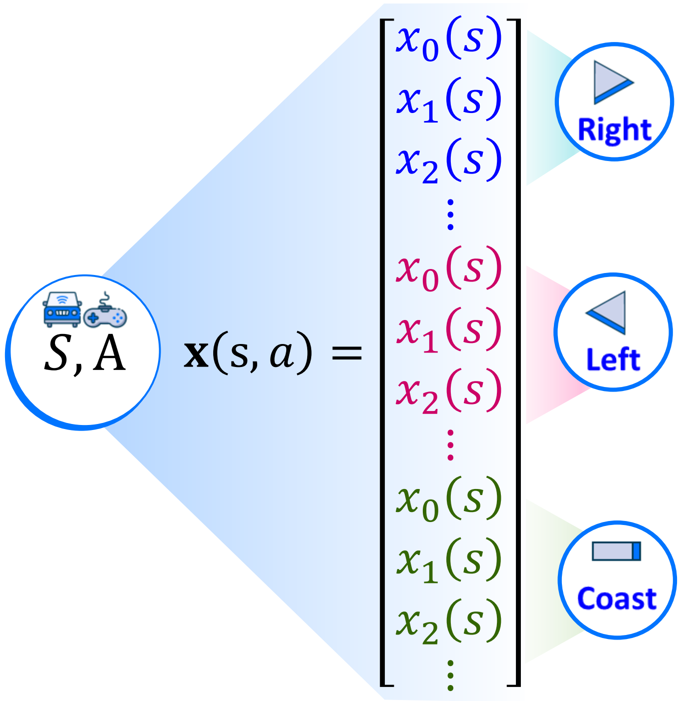
Car velocity

$$\dot{x}_{t+1} \doteq bound[\dot{x}_t + 0.001A_t - 0.0025\cos(3x_t)]$$

$$-0.07 \leq \dot{x}_{t+1} \leq 0.07$$



$$8 \times 8 \times 8$$



$$\mathbf{x}(s,a) = \begin{bmatrix} x_0(s) \\ x_1(s) \\ x_2(s) \\ \vdots \\ x_0(s) \\ x_1(s) \\ x_2(s) \\ \vdots \\ x_0(s) \\ x_1(s) \\ x_2(s) \\ \vdots \end{bmatrix}$$

$$\hat{q}(s,a,\mathbf{w}) \doteq \mathbf{w}^{\mathrm{T}} x(s,a) = \sum_{i=1}^{d} w_i \cdot x_i(s,a)$$

Episodic Sarsa with Function Approximation

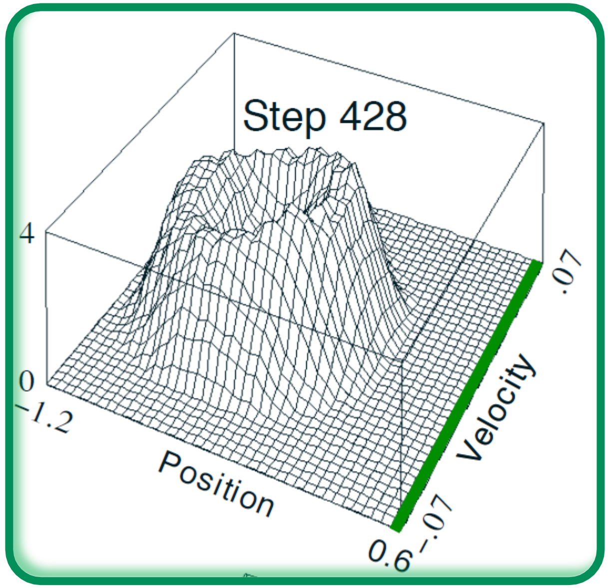
State Value Representation: Mountain Car

- **This example has a two-dimensional continuous-valued state.**
  - Ideally, it is desirable to plot the value function for every single state; but practically, it is not viable.
  - The number of states is an uncountably infinite, therefore, representing the state value for all possible states is not feasible.

- **How we can represent state values?**
  - **We can sample state values.**
    - We'll use the max value in each sample state.

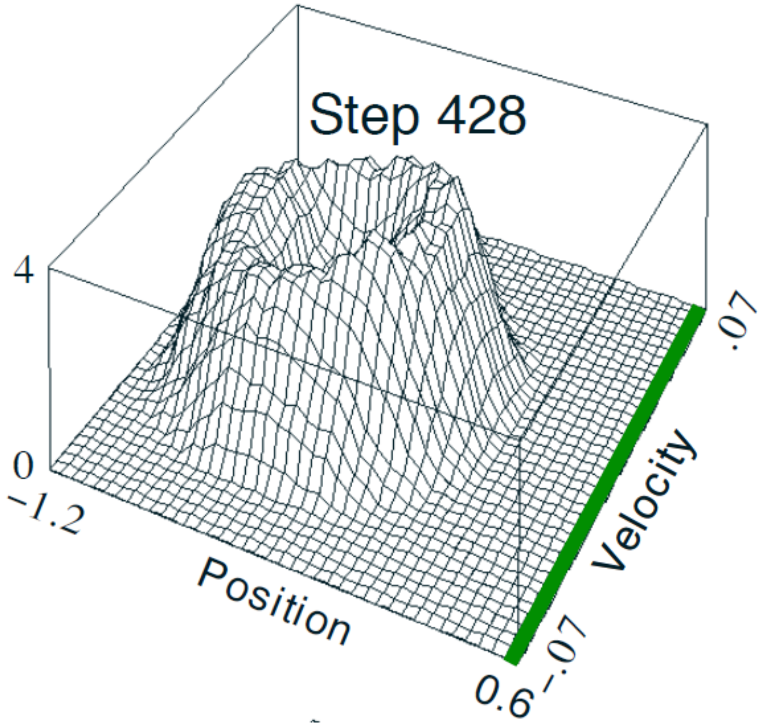
cost-to-go function

$$-\max_{a} \hat{q}(s, a, \mathbf{w})$$



  - This number is the **cost-to-go function**, indicating the number of steps the agent assumes it will take to escape under its greedy policy.

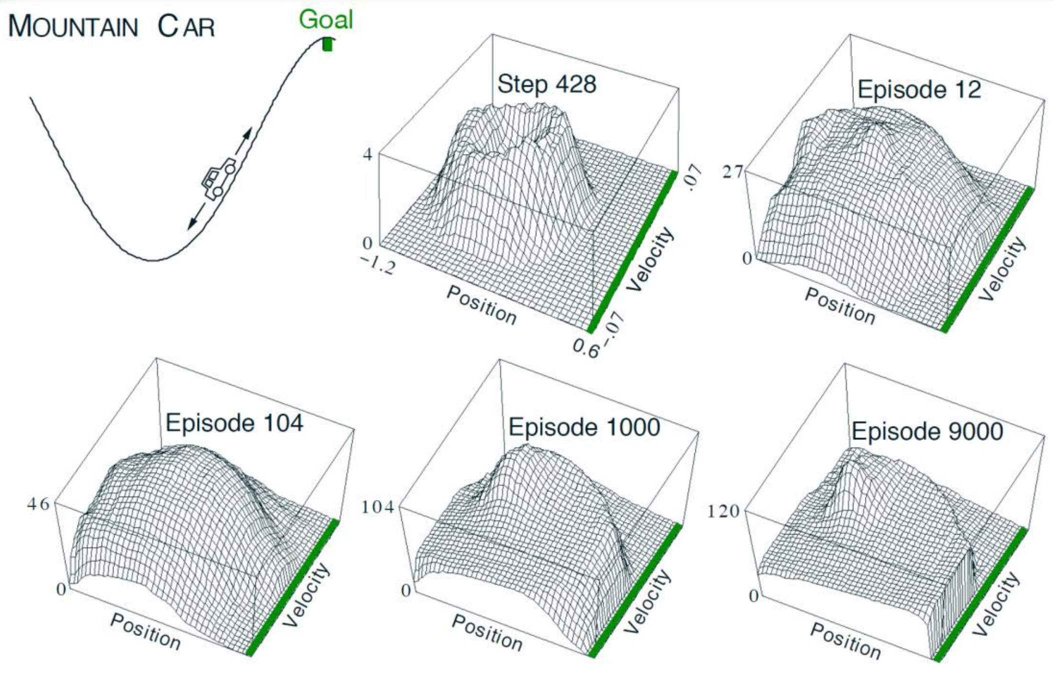
- **The initial action values were all zero, which was optimistic**
  - **We initialize the weights to zero.**
    - All true values are negative in this task), causing extensive exploration to occur even though the exploration parameter, ϵ, was 0.
    - The agent can act greedily without any additional random exploration.

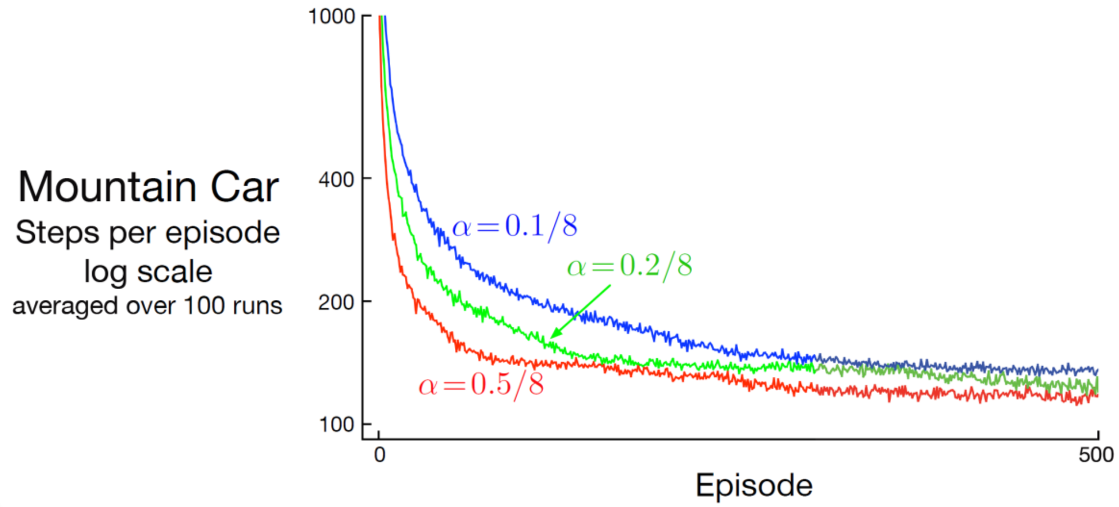


- **"Step-428" Results**
  - **At this time not even one episode had been completed**
    - However, the car has oscillated back and forth in the valley, following circular trajectories in state space.

Episodic Sarsa with Function Approximation

Results: Mountain Car





Episodic Sarsa with Function Approximation

Off-Policy updates

Sarsa

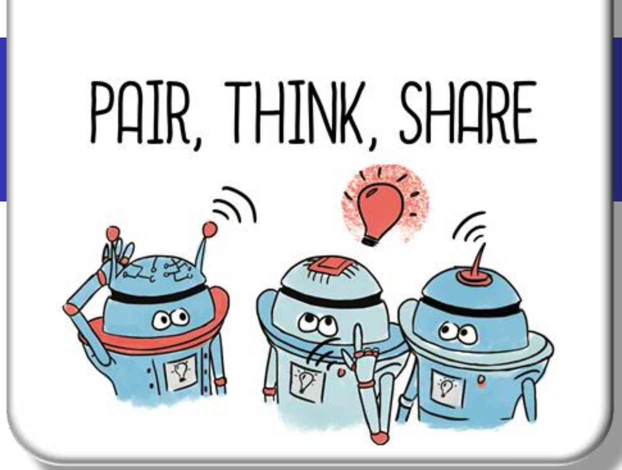
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

Expected Sarsa

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \sum_{a'} \pi\,(a'|S_{t+1}) Q(S_{t+1}, a') - Q(S_t, A_t)\right]$$

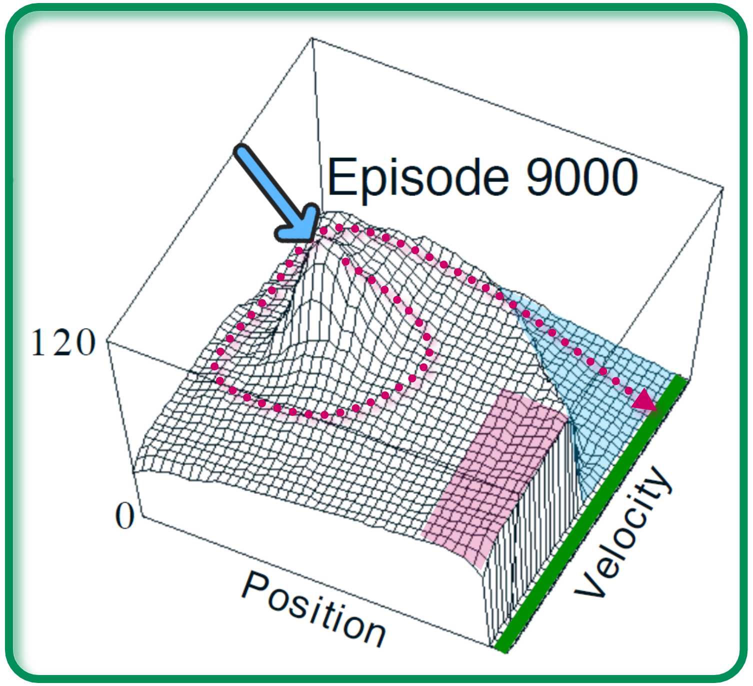
Q-Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right]$$



The Mountain Car example

- This figure shows the states values at the end of Episode 9000



1. What states are corresponding to the blue arrow?
2. What is your intuition about the blue highlighted states?
3. What is your intuition about the pink highlighted states?
4. What is indicated by the dashed-line?

Episodic Sarsa with
Expected Sarsa and Q-Learning

Q-learning

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Big(R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}, a', \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})\Big) \nabla \hat{q}(S_t, A_t, \mathbf{w})$$

Sarsa

$$\mathbf{w} \leftarrow \mathbf{w} + \boldsymbol{\alpha}\,[R_{t+1} + \boldsymbol{\gamma}\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})]\nabla\hat{q}(S_t, A_t, \mathbf{w})$$

Expected Sarsa

$$\mathbf{w} \leftarrow \mathbf{w} + \boldsymbol{\alpha}\,[R_{t+1} + \gamma \sum_{a'} \pi(a'|S_{t+1})\hat{q}(S_{t+1}, a', \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})]\nabla\hat{q}(S_t, A_t, \mathbf{w})$$

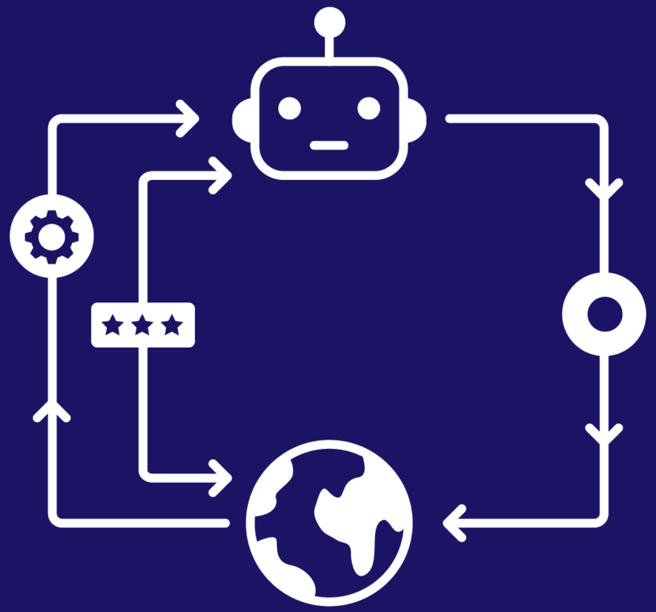
W

Update rules

Q-Learning

$$\mathbf{w} \leftarrow \mathbf{w} + \boldsymbol{\alpha}\,[R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}, a', \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})]\nabla\hat{q}(S_t, A_t, \mathbf{w})$$

$$\hat{q}(s, a, w) \approx q_\pi(s, a)$$



Summary