Chapter 10: On-policy Control with Approximation

Seungjae Ryan Lee

endtoend.ai

Episodic 1-step semi-gradient Sarsa

- Approximate action values (instead of state values)

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\Big]\nabla\hat{q}(S_t, A_t, \mathbf{w}_t).$$

- Use Sarsa to define target

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)\Big]\nabla\hat{q}(S_t, A_t, \mathbf{w}_t).$$

- Converges the same ways as TD(0) with same error bound

$$\overline{\mathrm{VE}}(\mathbf{w}_{\mathrm{TD}}) \;\leq\; \frac{1}{1-\gamma}\min_{\mathbf{w}}\overline{\mathrm{VE}}(\mathbf{w}).$$

endtoend.ai

Control with Episodic 1-step semi-gradient Sarsa

- Select action and improve policy using an ε-greedy action w.r.t. $\hat{q}(S_t, a, \mathbf{w}_t)$

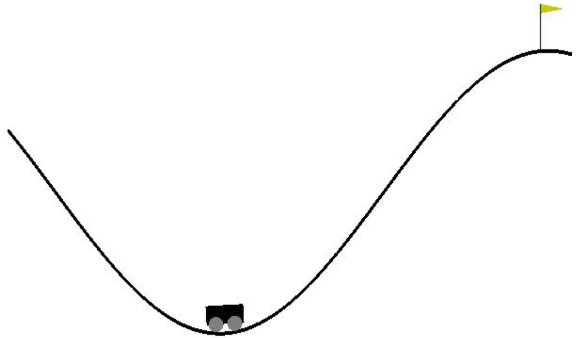
Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

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., $\mathbf{w} = \mathbf{0}$)

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'
        If S' is terminal:
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
            Go to next episode
        Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$

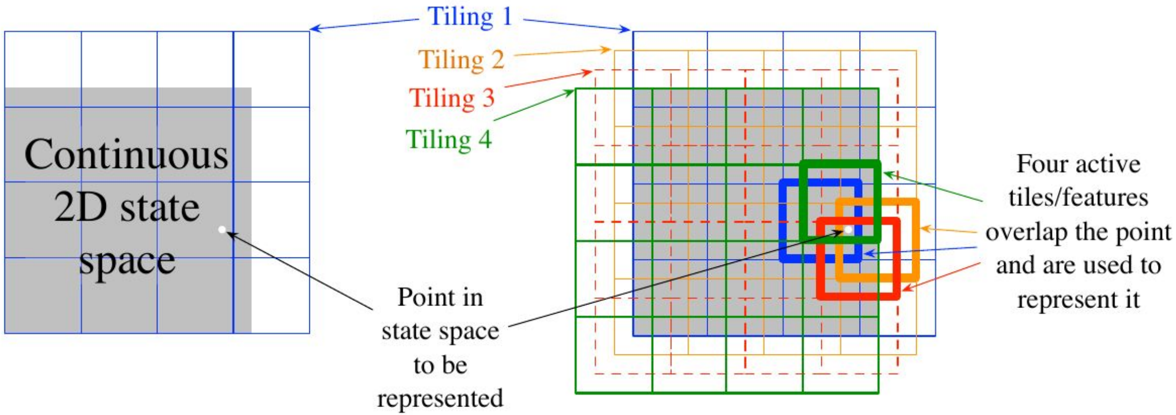
Mountain Car Example

- Task: Drive an underpowered car up a steep mountain road
  - Gravity is stronger than car's engine
  - Must swing back and forth to build enough inertia
- State: position x_t , velocity $\dot{x}_t$
- Actions: Forward (+1), Reverse (-1), No-op (0)
- Reward: -1 until the goal is reached



Approximation for Mountain Car

- *Tile coding* used to select binary features (8 tiles)

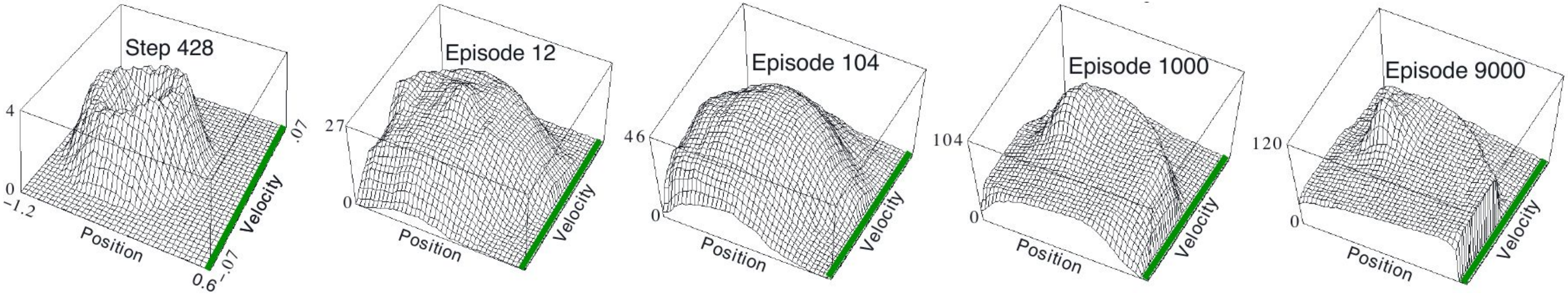


$$\hat{q}(s, a, \mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s, a) = \sum_{i=1}^{d} w_i \cdot x_i(s, a),$$

endtoend.ai

Results of Mountain Car

- Plot the *cost-to-go* function: $-\max_a \hat{q}(s, a, \mathbf{w})$
- Initial action values set to 0
    - Very optimistic



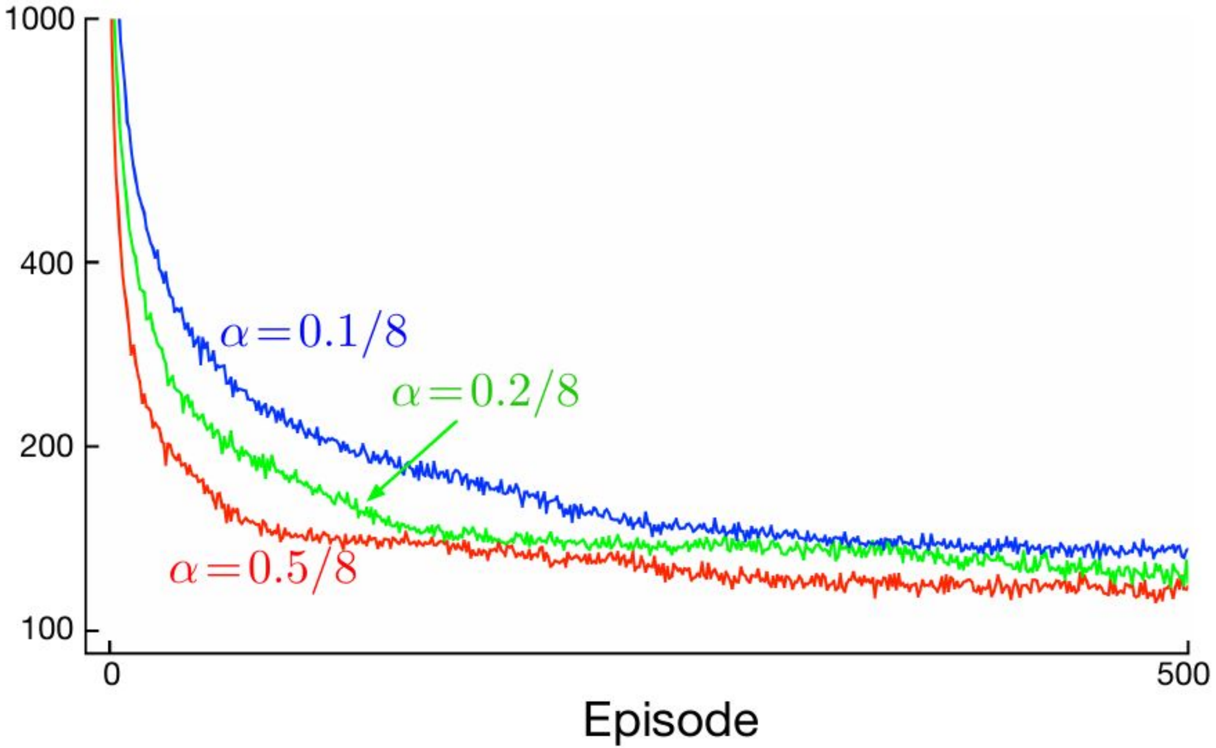
Results of Mountain Car

Mountain Car

Steps per episode

log scale

averaged over 100 runs



endtoend.ai

Episodic n-step Semi-gradient Sarsa

- Use n-step return $G_{t:t+n}$ as the update target

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)\Big]\nabla\hat{q}(S_t, A_t, \mathbf{w}_t).$$

$$\downarrow$$

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha\left[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})\right]\nabla\hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}),$$

Episodic n-step Semi-gradient Sarsa in Practice

Episodic semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_* \text{ or } q_\pi$

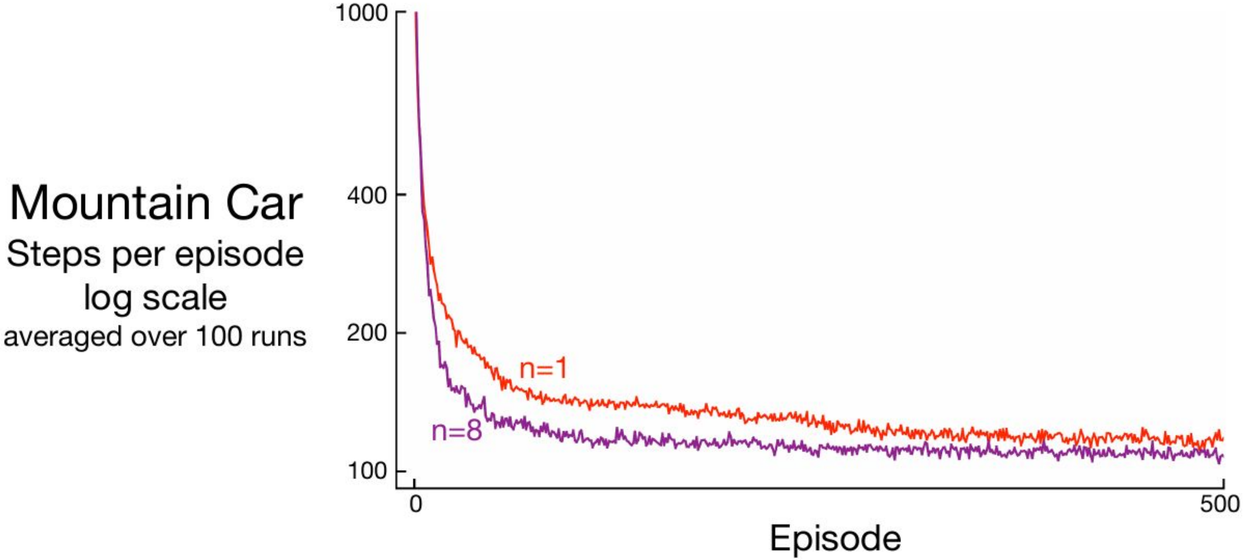
Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$, a positive integer n
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    Select and store an action $A_0 \sim \pi(\cdot|S_0)$ or ε-greedy wrt $\hat{q}(S_0, \cdot, \mathbf{w})$
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$:
    | If $t < T$, then:
    | Take action A_t
    | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    | If S_{t+1} is terminal, then:
    | $T \leftarrow t + 1$
    | else:
    | Select and store $A_{t+1} \sim \pi(\cdot|S_{t+1})$ or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
    | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    | If $\tau \geq 0$:
    | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
    | If $\tau + n < T$, then $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ $(G_{\tau:\tau+n})$
    | $\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[G - \hat{q}(S_\tau, A_\tau, \mathbf{w})\right] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
    Until $\tau = T - 1$

endtoend.ai

Episodic n-step Semi-gradient Sarsa Results

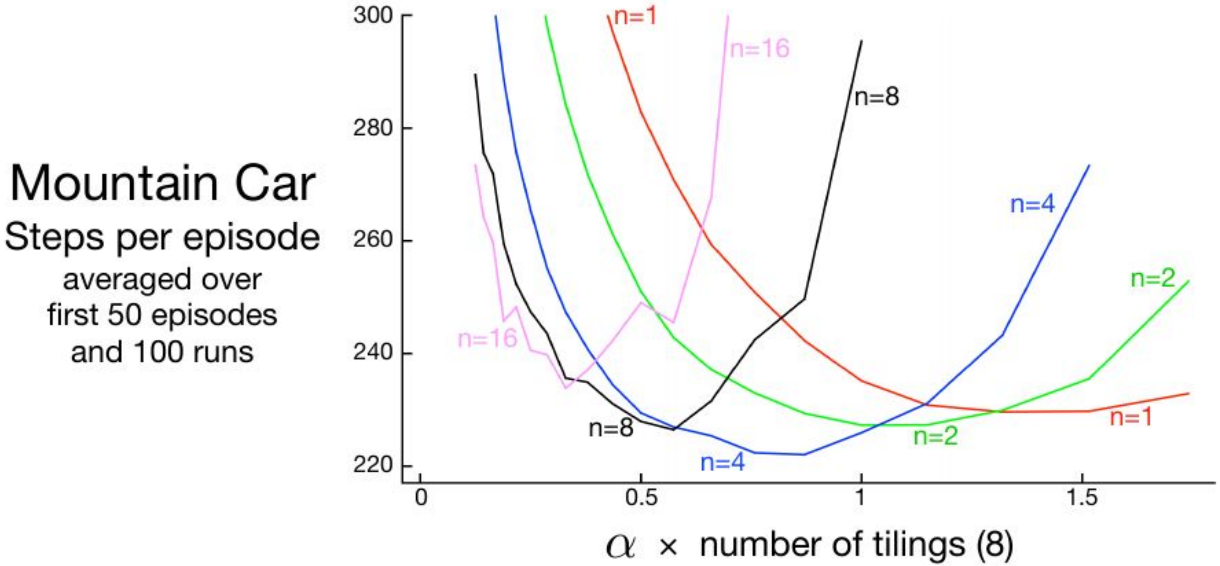
- Faster learning
- Better asymptotic performance



endtoend.ai

Episodic n-step Semi-gradient Sarsa Results

- Best performance for intermediate values of n-step



endtoend.ai

Average Reward Setting

- Quality $r(\pi)$ of policy π defined by the average reward following policy π
- Continuing tasks without discounting

$$
\begin{aligned}
r(\pi) &\doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] \\
&= \lim_{t\to\infty} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi], \\
&= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a) r,
\end{aligned}
$$

Differential Return and Value Functions

Differential Return: differences between rewards and average reward

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots.$$

Differential Value Functions: Expected differential returns

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$
$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

Bellman Equations

- Remove all γ
- Replace rewards with difference of rewards

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - r(\pi) + v_\pi(s') \Big],$$

$$q_\pi(s,a) = \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a') \Big],$$

$$v_*(s) = \max_a \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - \max_\pi r(\pi) + v_*(s') \Big], \text{ and}$$

$$q_*(s,a) = \sum_{r,s'} p(s', r \,|\, s, a) \Big[r - \max_\pi r(\pi) + \max_{a'} q_*(s', a') \Big]$$

Differential semi-gradient Sarsa

- Same update rule $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t \nabla\hat{q}(S_t, A_t, \mathbf{w}_t)$, with differential TD error
- Original TD error:

$$\delta_t \doteq R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)$$

- **Differential TD error**:

$$\delta_t = R_{t+1} - \bar{R}_{t+1} + \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)$$

Differential semi-gradient Sarsa

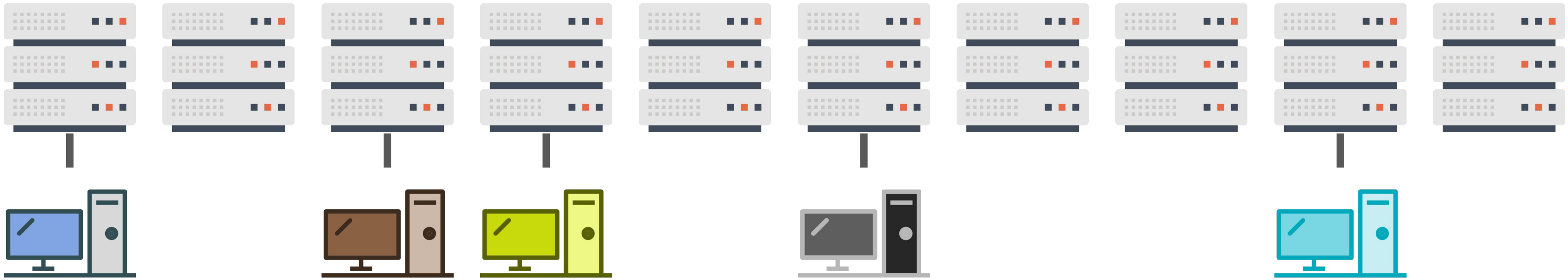
Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$

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

Initialize state S, and action A
Loop for each step:
    Take action A, observe R, S'
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\underline{\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})}$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

Access-Control Queuing Example

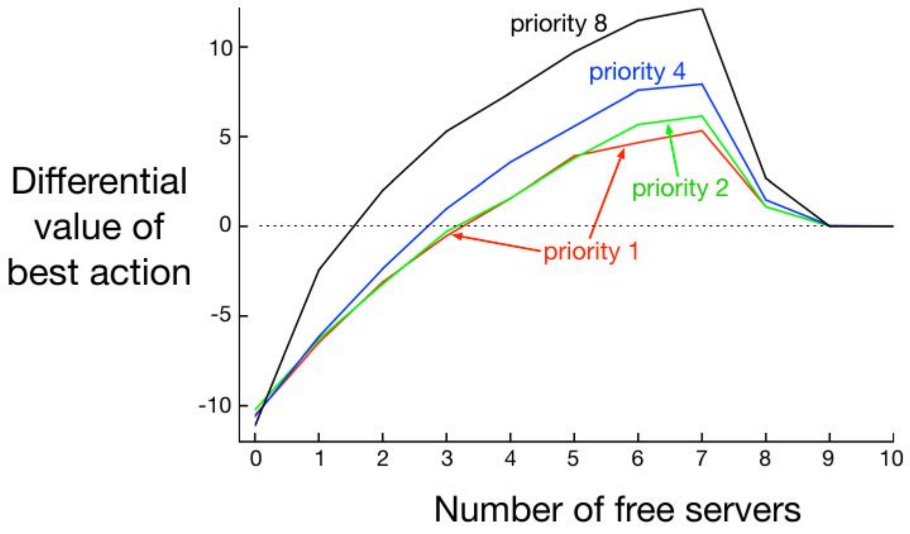
- Agent can grant access to 10 servers
    - Agent can accept or reject customers
- Customers arrive at a single queue
    - Customers have 4 different priorities, randomly distributed
    - Pay a reward of 1, 2, 4, or 8 when granted access to a server
- A busy server is freed with some probability

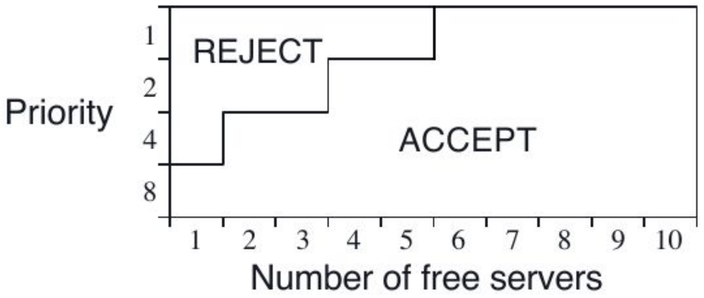


endtoend.ai

Access-Control Queuing Results

- Tabular solution with differential semi-gradient Sarsa





n-step Semi-gradient Sarsa

- Use n-step return
    - $G_{t:t+n} \doteq R_{t+1} - \bar{R}_{t+1} + R_{t+2} - \bar{R}_{t+2} + \ldots + R_{t+n} - \bar{R}_{t+n} + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1})$

$$\delta_t \doteq R_{t+1} - \bar{R}_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)$$

$$G_{t:t+1} - \hat{q}(S_t, A_t, \mathbf{w})$$

$$\downarrow$$

$$\delta_t \doteq G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w})$$

endtoend.ai

Thank you!

Original content from

- Reinforcement Learning: An Introduction by Sutton and Barto

You can find more content in

- github.com/seungjaeryanlee
- www.endtoend.ai