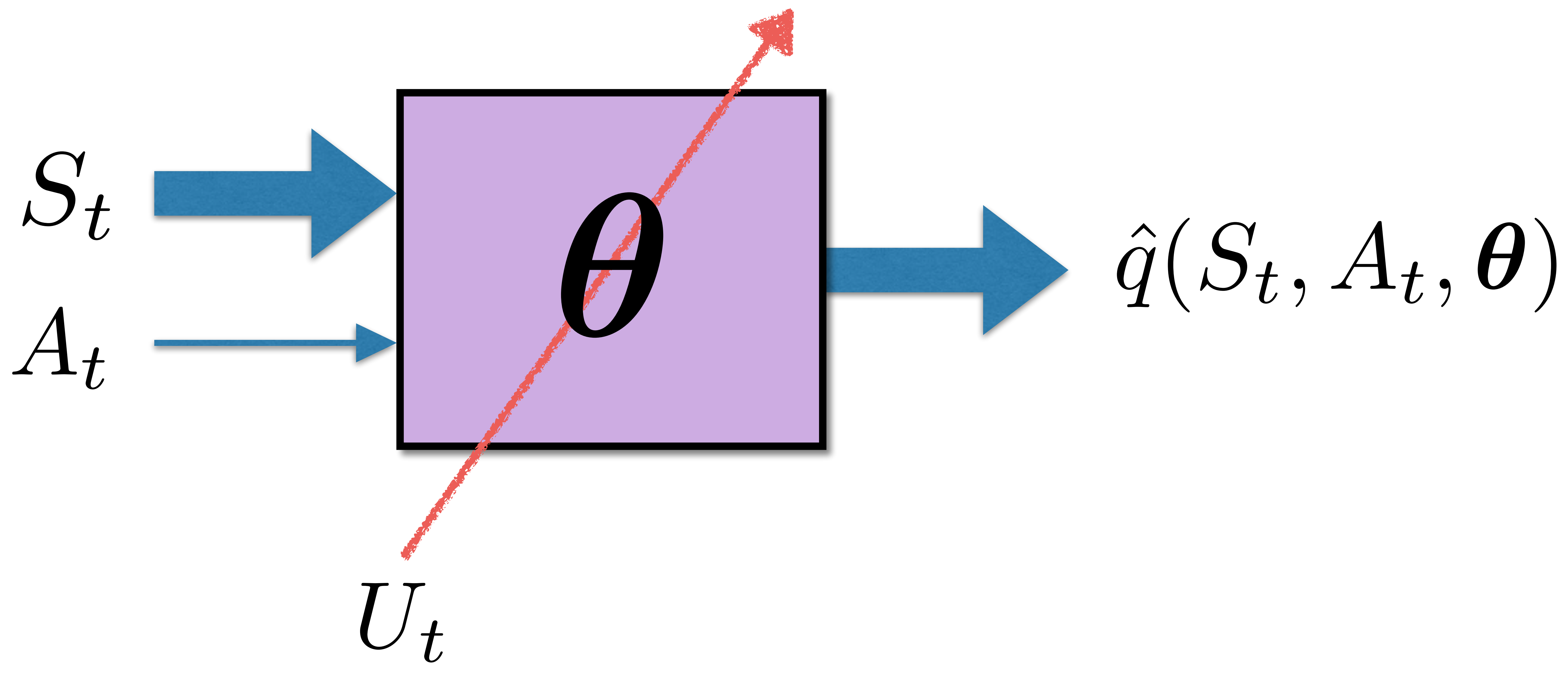
What we learned last time

- Value-function approximation by stochastic gradient descent enables RL to be applied to arbitrarily large state spaces
- Most algorithms just carry over the Targets from the tabular case
- With bootstrapping (TD), we don’t get true gradient descent methods
    - but the linear, on-policy case is still guaranteed convergent
    - and learning is *faster* with *n*-step methods (*n*>1), as before
- For continuous state spaces, coarse/tile coding is a good strategy

Chapter 10:

On-policy Control with Approximation

Value function approximation (VFA) replaces the table with a general parameterized form



On-policy Control with Approximation

- (Semi-)gradient methods carry over to control in the usual way
    - Mountain Car example
- *n*-step methods carry over too, with the usual tradeoffs
- A new *average-reward* setting,
  with *differential* value functions and *differential* algorithms
    - Queuing example (tabular)
- The discounting setting is deprecated

(Semi-)gradient methods carry over to control in the usual on-policy GPI way

- Always learn the action-value function of the current policy

- Always act near-greedily wrt the current action-value estimates

- The learning rule is the same as in Chapter 9:

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

update target, e.g., $U_t = G_t$ (MC) $\quad U_t = R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{\theta}_t)$ (Sarsa)

(Expected Sarsa) $U_t = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) \hat{q}(S_{t+1}, a, \boldsymbol{\theta}_t)$ $\quad U_t = \sum_{s',r} p(s', r|S_t, A_t) \Big[r + \gamma \sum_{a'} \pi(a'|s') \hat{q}(s', a', \boldsymbol{\theta}_t) \Big]$ (DP)

(Semi-)gradient methods carry over to control

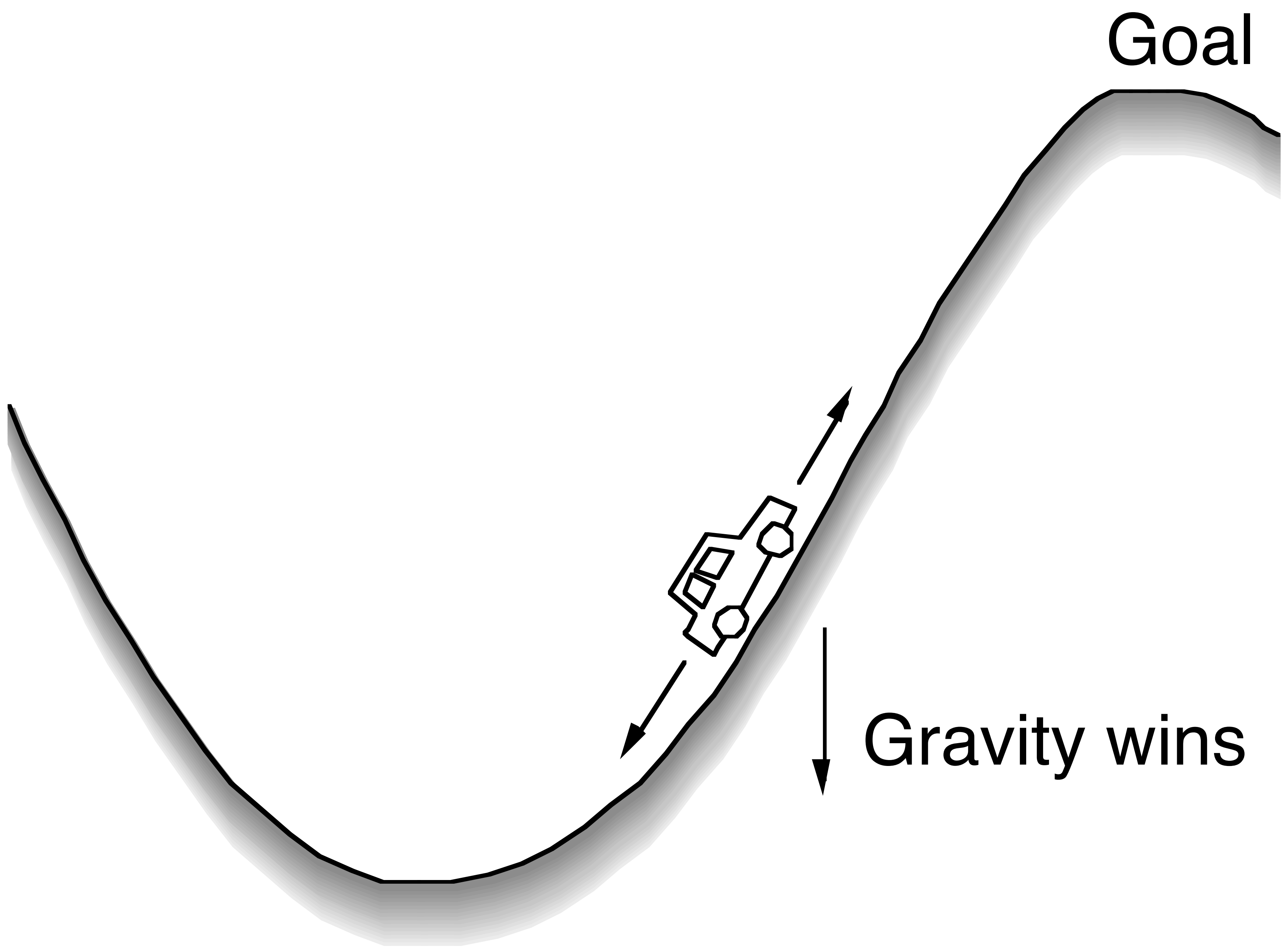
$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big[U_t - \hat{q}(S_t, A_t, \boldsymbol{\theta}_t)\Big]\nabla\hat{q}(S_t, A_t, \boldsymbol{\theta}_t)$$

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

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^n \to \mathbb{R}$

Initialize value-function weights $\boldsymbol{\theta} \in \mathbb{R}^n$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
    $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        If S' is terminal:
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R - \hat{q}(S, A, \boldsymbol{\theta})\big]\nabla\hat{q}(S, A, \boldsymbol{\theta})$
            Go to next episode
        Choose A' as a function of $\hat{q}(S', \cdot, \boldsymbol{\theta})$ (e.g., ε-greedy)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R + \gamma\hat{q}(S', A', \boldsymbol{\theta}) - \hat{q}(S, A, \boldsymbol{\theta})\big]\nabla\hat{q}(S, A, \boldsymbol{\theta})$
        $S \leftarrow S'$
        $A \leftarrow A'$

Example: The Mountain-Car problem



Minimum-Time-to-Goal Problem

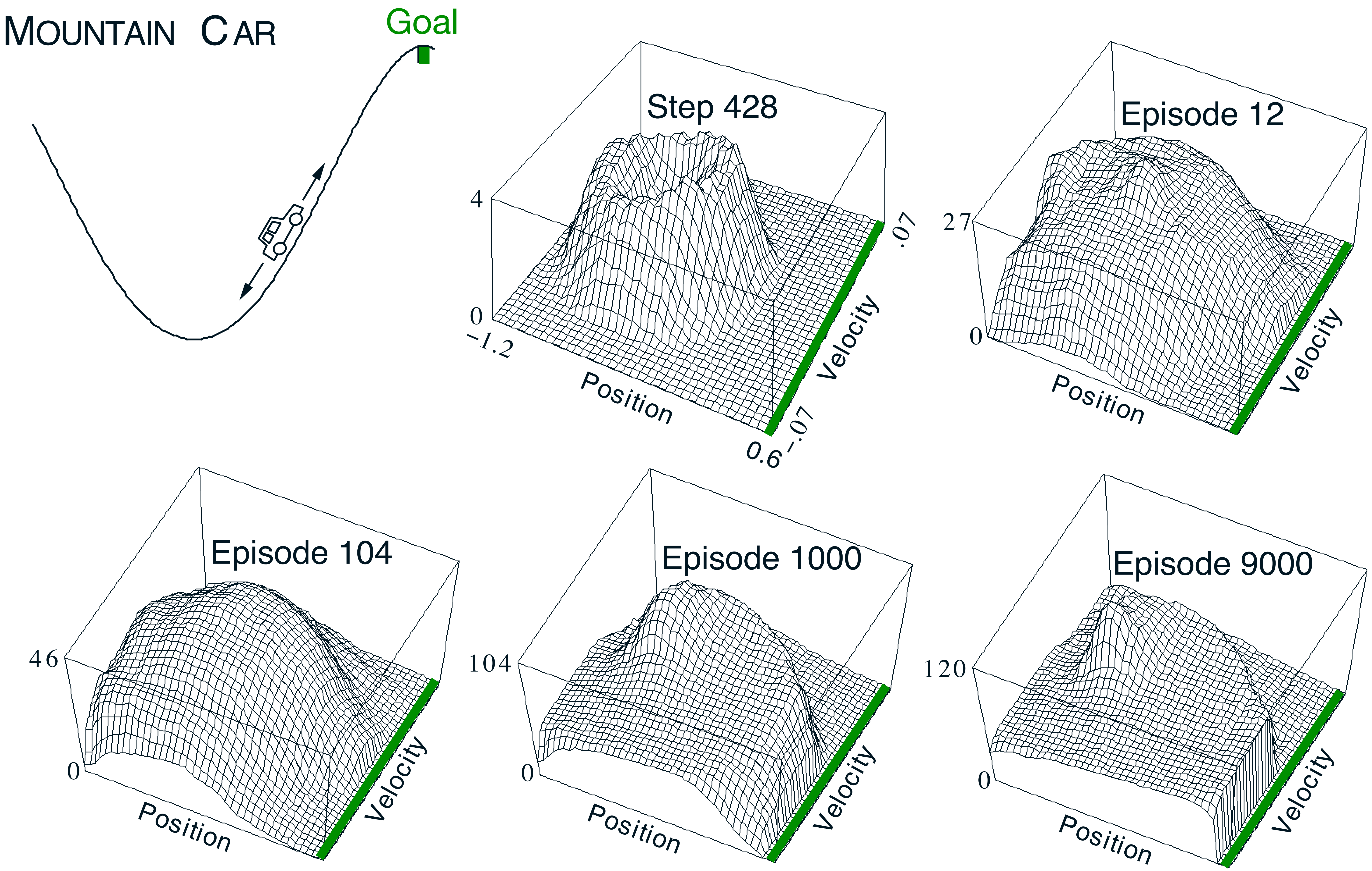
SITUATIONS:
car's position and velocity

ACTIONS:
three thrusts: forward, reverse, none

REWARDS:
always –1 until car reaches the goal

Episodic, No Discounting, γ=1

Values learned while solving Mountain-Car with tile coding function approximation

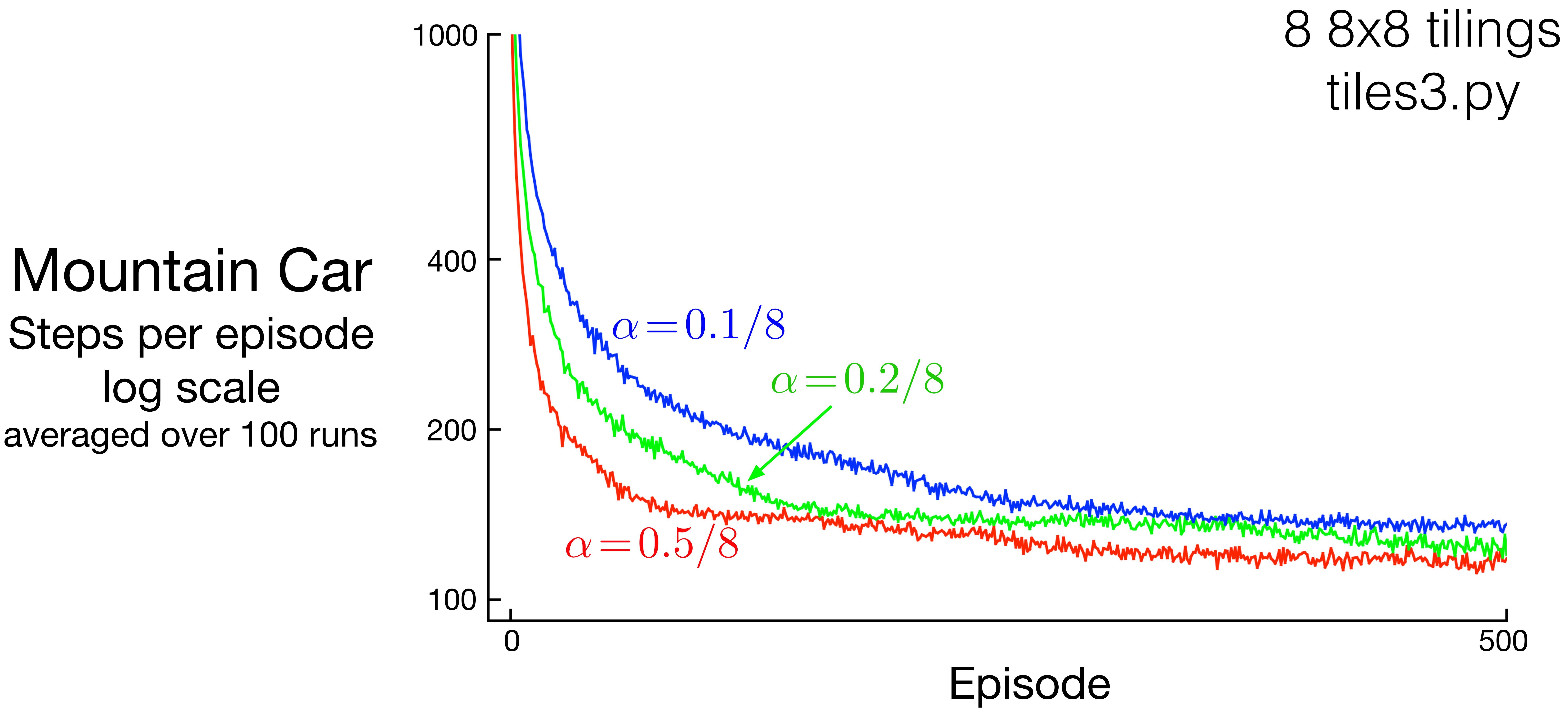


$$-\max_a \hat{q}(s, a, \boldsymbol{\theta})$$

Demo

Learning curves
for semi-gradient Sarsa with tile coding

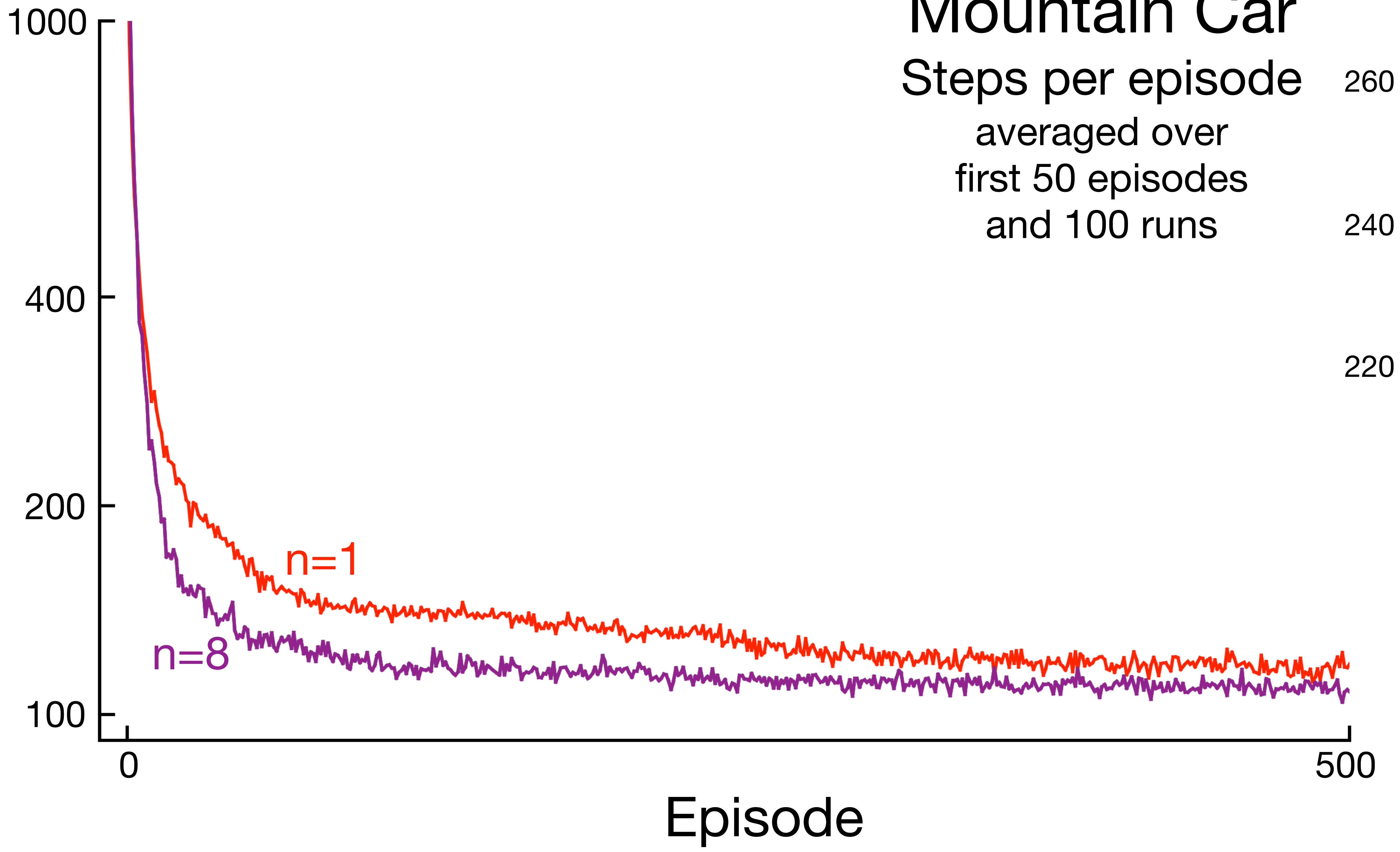
Mountain Car



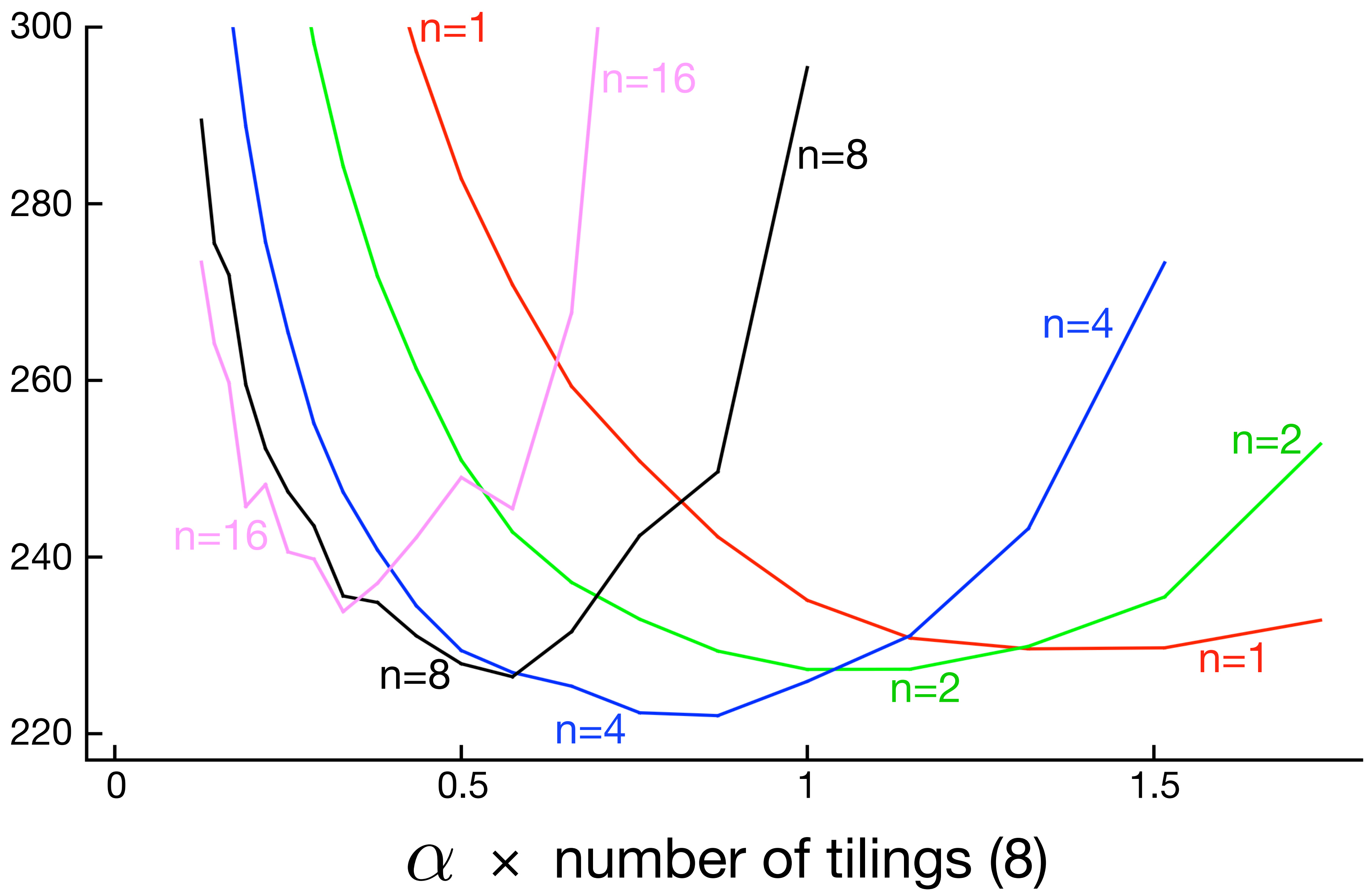
n-step semi-gradient Sarsa is better for n>1

$$\boldsymbol{\theta}_{t+n} \doteq \boldsymbol{\theta}_{t+n-1} + \alpha \left[G_t^{(n)} - \hat{q}(S_t, A_t, \boldsymbol{\theta}_{t+n-1}) \right] \nabla \hat{q}(S_t, A_t, \boldsymbol{\theta}_{t+n-1}), \qquad 0 \leq t < T$$

Mountain Car
Steps per episode
log scale
averaged over 100 runs



Mountain Car
Steps per episode
averaged over
first 50 episodes
and 100 runs



On-policy Control with Approximation

- (Semi-)gradient methods carry over to control in the usual way
  - Mountain Car example
- *n*-step methods carry over too, with the usual tradeoffs
- A new average-reward setting,
  with differential value functions and differential algorithms
  - Queuing example (tabular)
- The discounting setting is deprecated

On-policy Control with Approximation

- (Semi-)gradient methods carry over to control in the usual way
  - Mountain Car example
- *n*-step methods carry over too, with the usual tradeoffs
- A new average-reward setting,
  with differential value functions and differential algorithms
  - Queuing example (tabular)
- The discounting setting is deprecated

A new goal for continuing tasks: Maximizing average reward per time step

$$\text{Maximize } \eta(\pi) \doteq \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[R_t \mid A_{0:t-1} \sim \pi]$$
$$= \lim_{t\to\infty} \mathbb{E}[R_t \mid A_{0:t-1} \sim \pi],$$
$$= \sum_s d_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) r$$

assuming that these limits exist
is known as the *ergodicity* property

$d_\pi : \mathcal{S} \to [0,1]$ is the steady-state distribution under π,
also known as the on-policy distribution:

$$d_\pi(s) \doteq \lim_{t\to\infty} \Pr\{S_t = s | A_{0:t-1} \sim \pi\}$$

$\eta(\pi)$ is the *average* amount of *reward* received per time step

In the average reward setting, everything is new

$\cancel{\gamma}$

- Returns: $G_t \doteq R_{t+1} - \eta(\pi) \;+\; R_{t+2} - \eta(\pi) \;+\; R_{t+3} - \eta(\pi) \;+\; \cdots$

- Bellman Eqs:

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

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

prediction

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

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

control

- Update targets:

$$U_t \doteq R_{t+1} - \bar{R}_t + \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{\theta}) \quad \text{or} \quad U_t \doteq R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, \boldsymbol{\theta})$$

estimate of $\eta(\pi)$

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

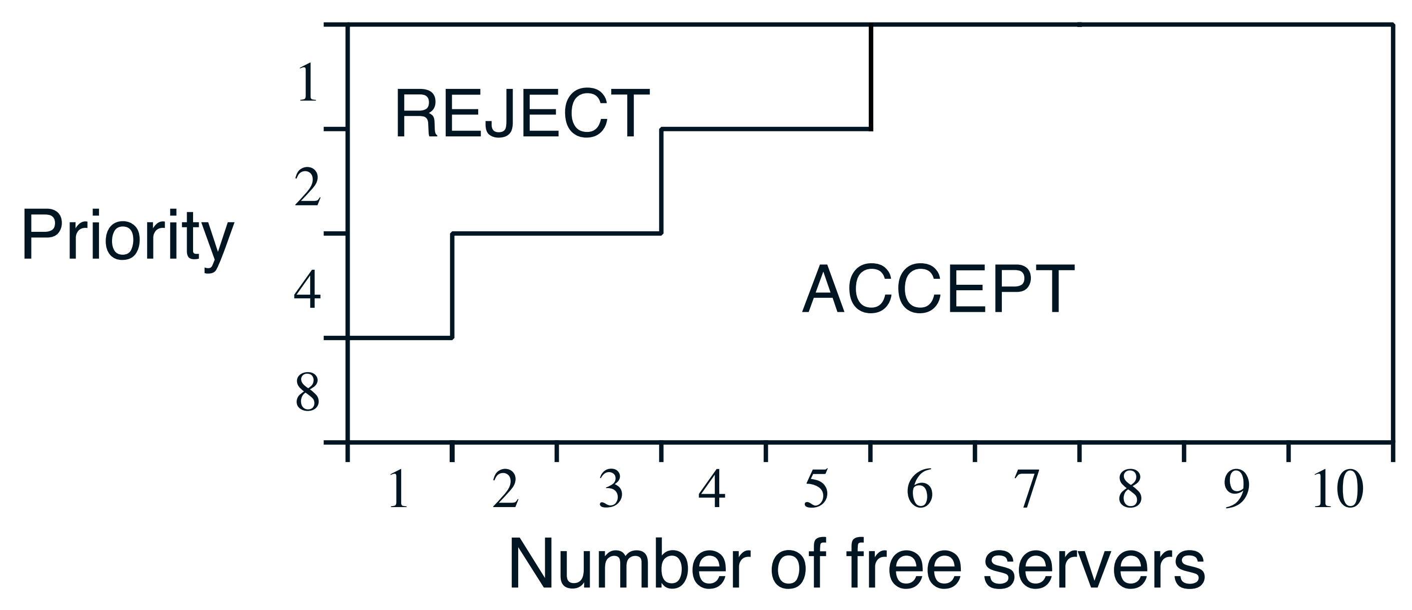
Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^n \to \mathbb{R}$
Parameters: step sizes $\alpha, \beta > 0$

Initialize value-function weights $\boldsymbol{\theta} \in \mathbb{R}^n$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Initialize average reward estimate $\bar{R}$ arbitrarily (e.g., $\bar{R} = 0$)
Initialize state S, and action A

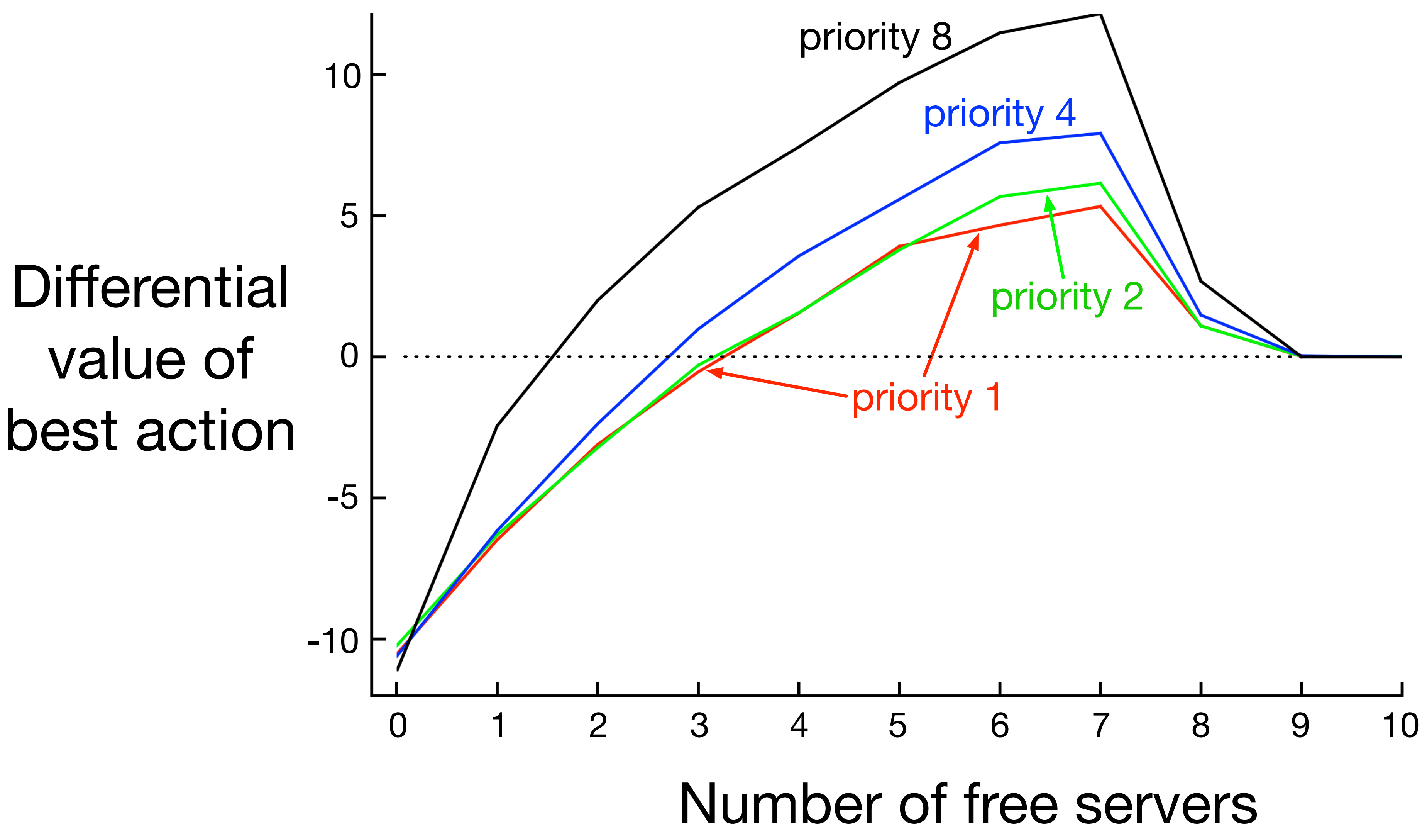
Repeat (for each step):
 Take action A, observe R, S'
 Choose A' as a function of $\hat{q}(S', \cdot, \boldsymbol{\theta})$ (e.g., ε-greedy)
 $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \boldsymbol{\theta}) - \hat{q}(S, A, \boldsymbol{\theta})$
 $\bar{R} \leftarrow \bar{R} + \beta\delta$
 $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\nabla\hat{q}(S, A, \boldsymbol{\theta})$
 $S \leftarrow S'$
 $A \leftarrow A'$

Example: The access-control queuing problem solved by tabular differential Sarsa

- Customers wait in line to be served by one of k=10 servers
- Customers pay rewards of 1, 2, 4, or 8 (depending on their priority) for being served
- On each step, the customer at the front of the queue is accepted (served), or rejected
- The queue never empties; new customers have random priorities
- Busy servers become free with probability p=0.06 on each step



POLICY



VALUE FUNCTION

$t = 2{,}000{,}000,\ \alpha = \beta = .01,\ \epsilon = .1,\ \bar{R}_t \approx 2.31$

Discounting is futile in continuing control settings with function approximation

- The problem statement is broken! The goal is broken!
- We can not longer give a useful ordering on policies
  - we can only order a few policies,
    those that dominate others in all states
- It would be OK if we could say what states we care about,
  but in the control case we can't
- Suppose we cared about states according to how often they occur? Surprisingly, discounting then becomes irrelevant!

The Futility of Discounting in Continuing Problems

Perhaps discounting can be saved by choosing an objective that sums discounted values over the distribution with which states occur under the policy:

$$\begin{aligned}
J(\pi) &= \sum_s d_\pi(s) v_\pi^\gamma(s) && \text{(where } v_\pi^\gamma \text{ is the discounted value function)} \\
&= \sum_s d_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma v_\pi^\gamma(s')\right] && \text{(Bellman Eq.)} \\
&= \eta(\pi) + \sum_s d_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \gamma v_\pi^\gamma(s') && \text{(from (10.5))} \\
&= \eta(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \sum_s d_\pi(s) \sum_a \pi(a|s) p(s'|s, a) && \text{(from (3.8))} \\
&= \eta(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') d_\pi(s') && \text{(from (10.6))} \\
&= \eta(\pi) + \gamma J(\pi) \\
&= \eta(\pi) + \gamma \eta(\pi) + \gamma^2 J(\pi) \\
&= \eta(\pi) + \gamma \eta(\pi) + \gamma^2 \eta(\pi) + \gamma^3 \eta(\pi) + \cdots \\
&= \frac{1}{1-\gamma} \eta(\pi).
\end{aligned}$$

The proposed discounted objective orders policies identically to the undiscounted (average reward) objective. We have failed to save discounting!

Conclusions

- Control is straightforward in the on-policy, episodic, linear case
- For the continuing case, we need the average-reward setting
    - which is a lot like just replacing R_t with $R_t - \eta(\pi)$ everywhere
    - where $\eta(\pi)$ is the average reward per step, or its estimate
- We should probably never use discounting as a control objective
- Formal results (bounds) exist for the linear, on-policy case
    - we get chattering near a good solution, not convergence