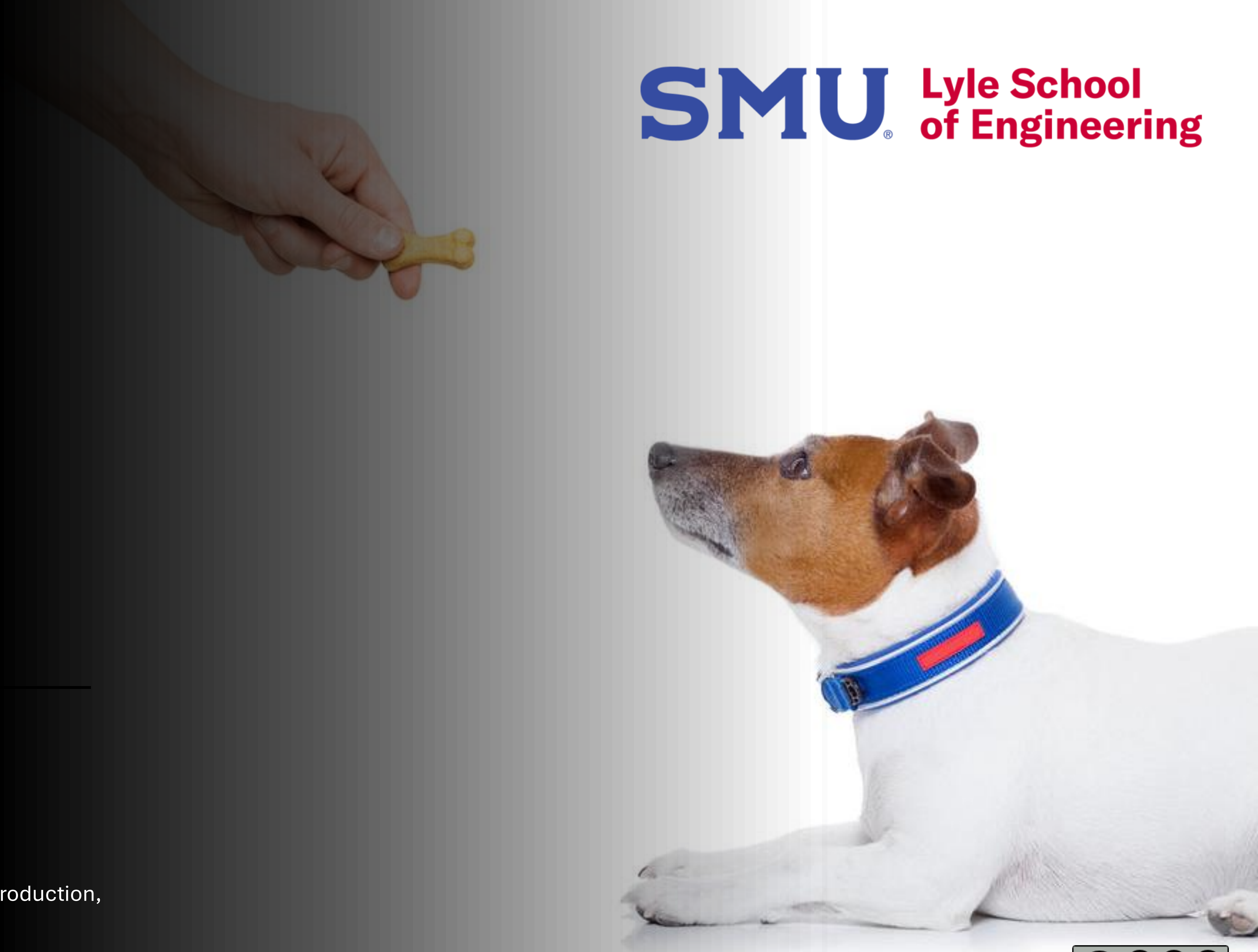
Reinforcement Learning

Policy Gradient Methods

Sutton/Barto* Chapter 13



Michael Hahsler, SMU

With figures from Sutton/Barto*

*Sutton and Barto, Reinforcement Learning: An Introduction, 2nd edition, MIT Press, Cambridge, MA, 2018

Topics of this Course

- Introduction to reinforcement learning
- Markov decision processes
- Part I: Tabular Methods
    - Dynamic programming
    - Monte Carlo methods
    - Temporal-difference learning
    - Multi-step bootstrapping
    - Planning and learning with tabular methods
- **Part II: Approximate Solution Methods**
    - Prediction and Control using Approximation
    - Eligibility Traces
    - **Policy Gradient Methods**
- Part III: Modern RL Methods
    - Deep Reinforcement Learning
    - Current Applications

Summary of Notation

General

X	capital letters: random variables
x, p	lower-case letters: realizations of random variables or scalar functions
$\boldsymbol{w}$	Bold lower-case letters: real-valued vectors (even if random variables)
$\boldsymbol{W}$	bold capitals: matrices
α	Greek letters: parameters (vectors if in bolt)
$\Pr\{X = x\}$	probability that a random variable X takes on the value x
$X \sim p$	random variable X selected from distribution $p(x) = \Pr\{X = x\}$
$\mathbb{E}[X]$	expectation of a random variable X, i.e., $\mathbb{E}[X] = \sum_x p(x)x$
$\text{argmax}_a\, f(a)$	a value of action a at which $f(a)$ takes its maximal value

Value Function

G_t	return (cumulative reward) following time t
$G_{t:h}$	return from t to h (discounted and corrected)
$v_\pi(s)$	value of state s under policy π (expected return)
$v_*(s)$	value of state s under the optimal policy
$q_\pi(s, a)$	value of taking action a in state s under policy π
$q_*(s, a)$	value of taking action a in state s under the optimal policy
V, V_t	array estimates of state-value function v_π or v_*
Q, Q_t	array estimates of action-value function q_π or v_*

MDP

s, s'	states
a	an action
r	a reward
$\mathcal{S}$	set of all (nonterminal) states, $\mathcal{S}^+$ are all states
$\mathcal{A}(s)$	set of all actions available in state s
γ	discount-rate parameter
t	discrete time step
T	final time step of an episode (a.k.a. horizon)
A_t	random variable for the action at time t
S_t	random variable for the state at time t
R_t	random variable for the reward at time t
$p(s', r \mid s, a)$	probability of transition to state s' and receiving reward r, from state s taking action a.
$p(s' \mid s, a)$	probability of transition to state s' fom state s taking action a.
$r(s, a)$	expected immediate reward from state s after action a.
$r(s, a, s')$	expected immediate reward from state s to s' with action a.
$\pi(a \mid s)$	probability of taking action a in state s under stochastic policy π
$\pi(s)$	action taken in state s under deterministic policy π

Parametrized Policies

Learn policies directly

Learn a Parameterized Policy

- **Up to now**: Learn parameters for an approximate action value function $\hat{q}$ and then extract a greedy policy.
- **New idea**: Learn parameters for a policy directly.

Parameterized policy:

$$\pi(a, | s, \boldsymbol{\theta}) = Pr\{A_t = a \mid S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}, \quad \boldsymbol{\theta} \in \mathbb{R}^d$$

- Learn $\boldsymbol{\theta}$ given a performance measure $J(\boldsymbol{\theta})$ using approximate gradient ascent.
- Update rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}$$

where$\widehat{\nabla J(\boldsymbol{\theta}_t)} \in \mathbb{R}^d$ is a stochastic estimate of the gradient.

Parametrized Policy Example

Requirements

- $\pi(a, | s, \boldsymbol{\theta})$ needs to be **differentiable** with respect to $\boldsymbol{\theta}$ so that $\nabla\pi(a, | s, \boldsymbol{\theta})$ exist and is finite.
- π needs to stay soft to guarantee **exploration** and convergence.

Popular for a small, discrete action space: **Soft-max in action preferences**

1. Define a numerical preference function:

   $$h(s, a, \boldsymbol{\theta}) \in \mathbb{R}$$

   Some options:
   - ANN with $\boldsymbol{\theta}$ as network weights
   - Linear in features $h(s, a, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{x}(s, a)$

2. Give the action with the highest preference in each state the highest probability:

   $$\pi(a, | s, \boldsymbol{\theta}) = \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}}$$

Advantages over ϵ-greedy Action Selection

- **Stochastic policies**: Soft-max defined stochastic policies (i.e., each action has a probability). The policy will be driven to the optimal stochastic policy which can also approach a deterministic policy.

  Note: value-action methods cannot learn stochastic policies and are often stuck with learning an ϵ-greedy policy!

- **Smoothness**: Parameterized policy changes action probability smoothly while ϵ-greedy policies jump when a different action becomes the maximum.

- **Prior knowledge** about the form of the policy can be incorporated into the way the policy is parameterized. This can improve learning significantly.

Policy Gradient Theorem

- We define the performance measure as the true value of following π_θ from the start state s_0

$$J(\boldsymbol{\theta}) \overset{\text{def}}{=} v_{\pi_\theta}(s_0)$$

- The **policy gradient theorem** gives us an analytical expression for the gradient of the with respect to the policy parameters:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

On-policy distribution under π

Depends on estimates for q

Derived from the def. of the policy.

- The theorem also provides a **strong convergence guarantee**!

Policy: $\pi(a|s,\boldsymbol{\theta}) = \dfrac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}}$

Gradient: $\nabla\ \ln \pi(s|a,\boldsymbol{\theta}) = h(s,a) - \sum'_a h(s,a')\pi(s|a',\boldsymbol{\theta})$

REINFORCE Update

Normal stochastic gradient-ascent algorithm update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \, \nabla J(\boldsymbol{\theta}_t)$$

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$ (replace s by the sampled $S_t \sim \mu$)

$$= \mathbb{E}_\pi \left[\sum_a \pi(A_t|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$ (replace a by the sampled $A_t \sim \pi$)

$$= \mathbb{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]$$ (because $\mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t)$)

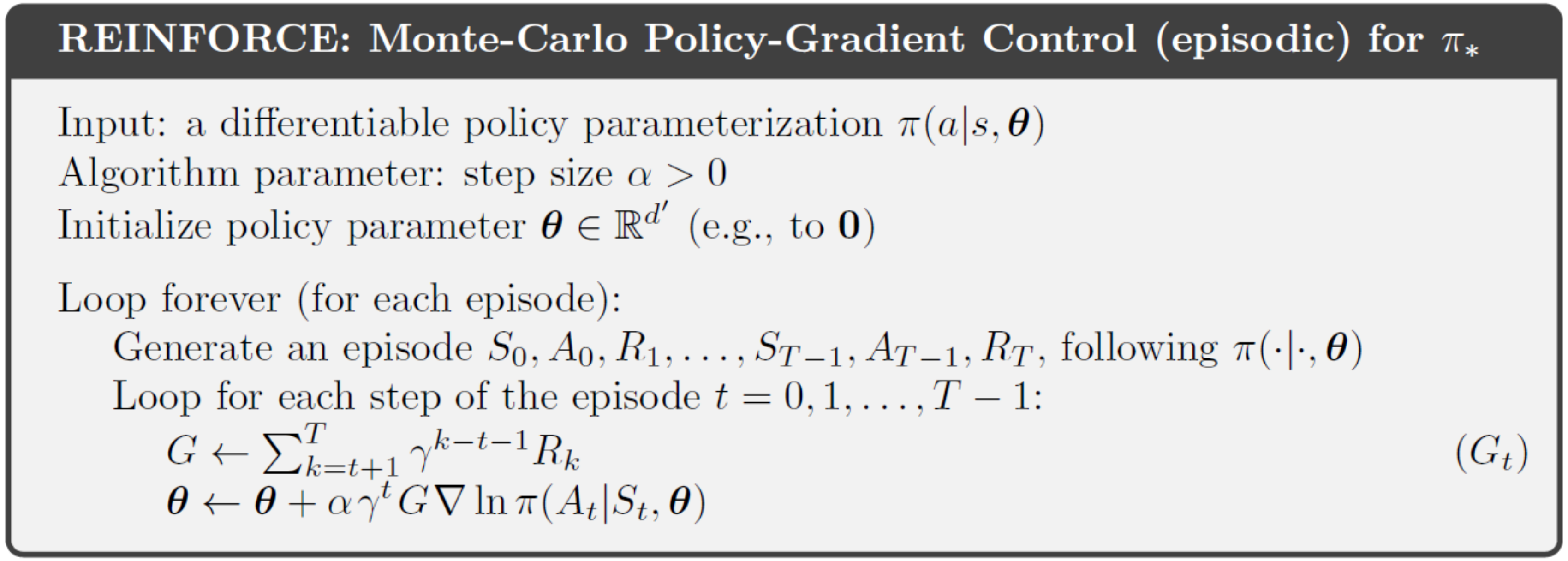
$$= \mathbb{E}_\pi [G_t \ln \nabla \pi(A_t|S_t,\boldsymbol{\theta})]$$ (because $\nabla \ln x = \frac{\nabla x}{x}$)

We can now sample G_t, A_t and S_t using π

MC Policy Gradient

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \underbrace{\mathbb{E}_\pi[G_t \ln \nabla \pi(A_t|S_t, \boldsymbol{\theta})]}_{\text{Proportional to } \nabla J(\boldsymbol{\theta})}$$

A_t, S_t and an unbiased estimate of G_t under the current π can be obtained via MC.



REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

```
Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                                        $(G_t)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
```

Actor-Critic Methods

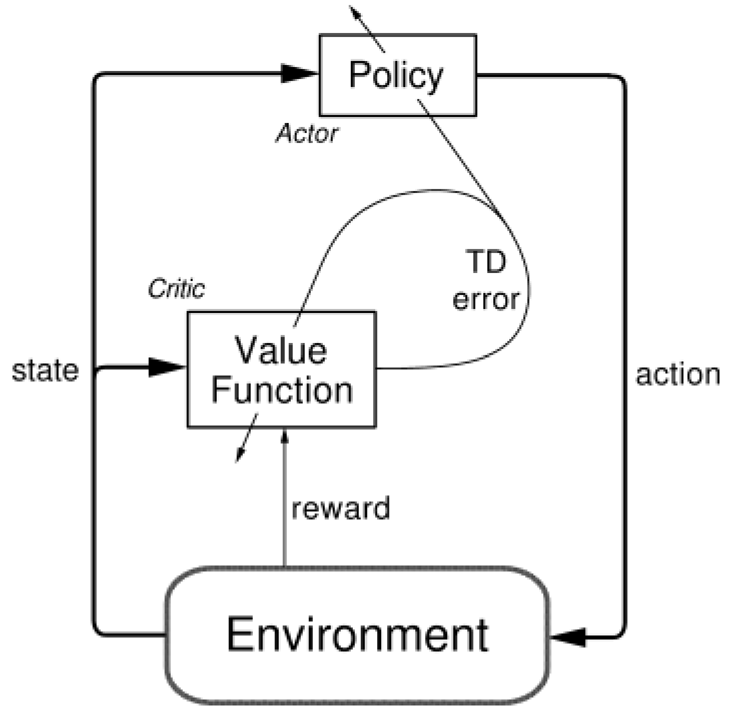
Combining parameterized policies with parameterized value functions

Actor-Critic Methods

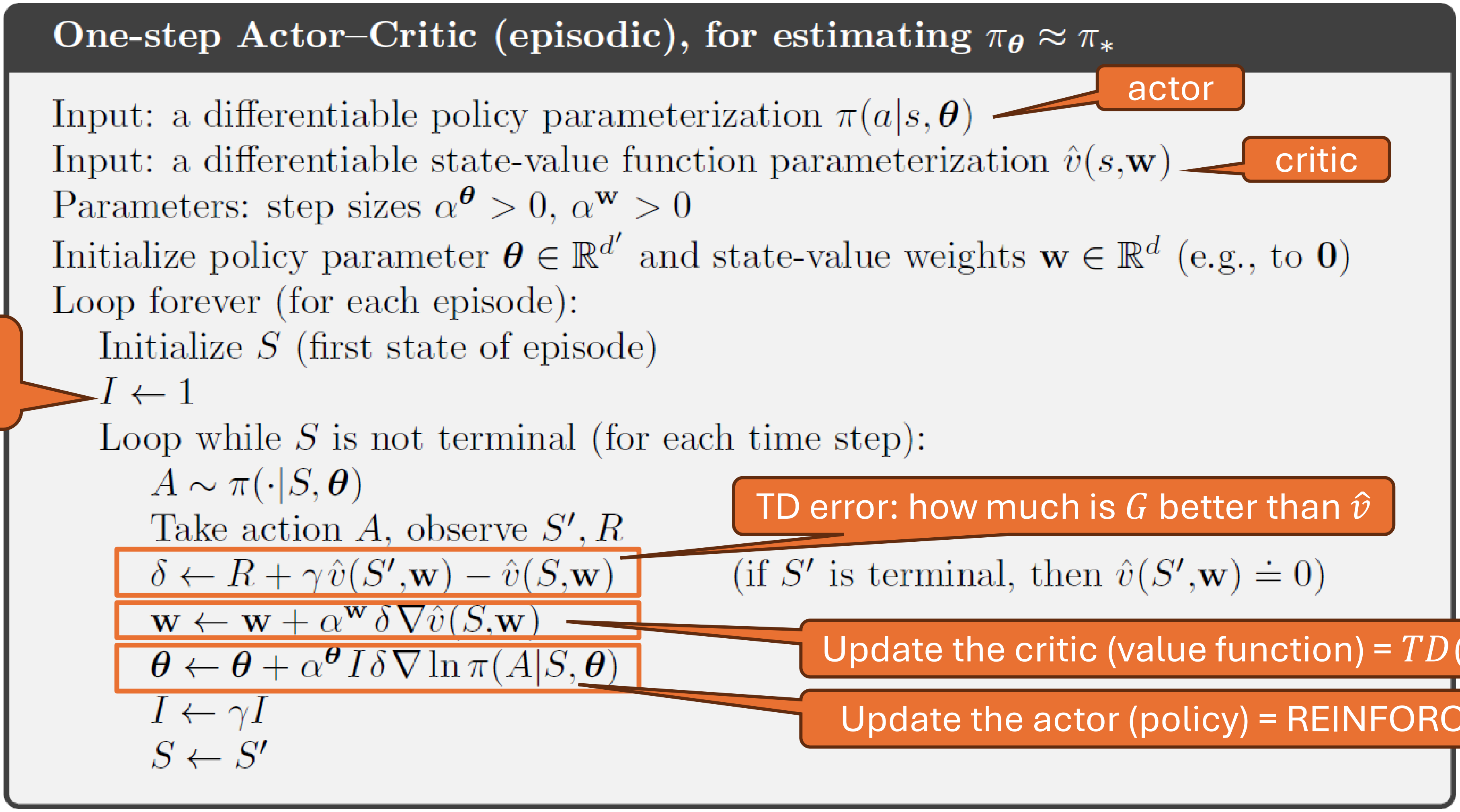
- **Idea**: estimate a parameterized policy and a parameterized value function at the same time.
- **Actor**: estimates the policy and decides on actions.
- **Critic**: estimates the TD error using its value function estimate

$$\delta_t = \underbrace{R_{t+1} + \gamma V(S_{t+1})}_{G_t} - V(S_t)$$

- The critic judges the actor's action by comparing the received reward to the expected difference in value.
- The TD error is used to update both approximations.
- A positive TD error indicates the last action performed better than expected and the tendency to pick it should be strengthened, whereas if the TD error is negative, it suggests the tendency should be weakened.



One-Step Actor-Critic Algorithm



One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

```
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize $S$ (first state of episode)
    $I \leftarrow 1$
    Loop while $S$ is not terminal (for each time step):
        $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
        Take action $A$, observe $S', R$
        $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$        (if $S'$ is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
```

actor

critic

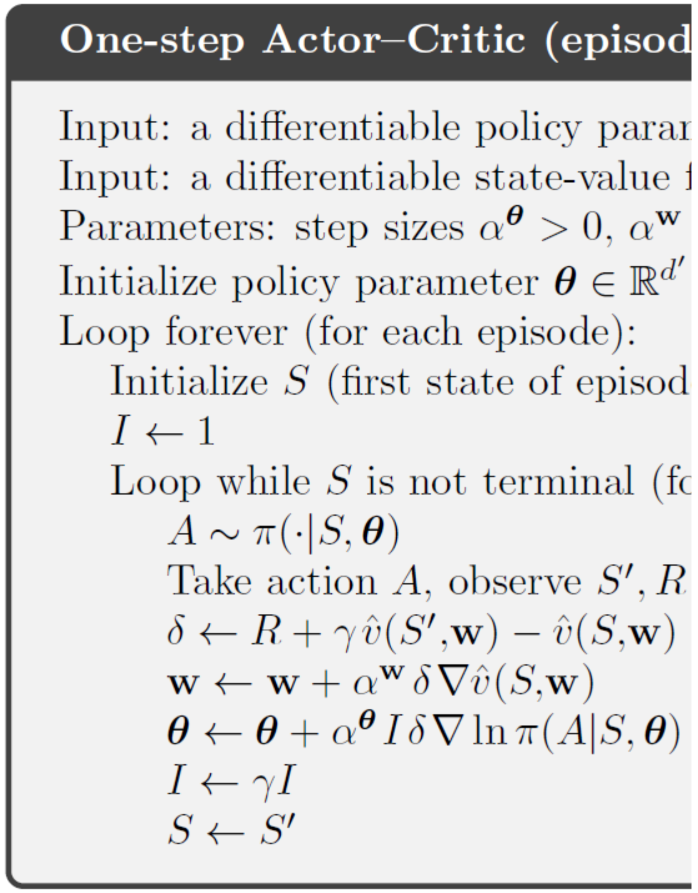
Takes care of discounting γ^t

TD error: how much is G better than $\hat{v}$

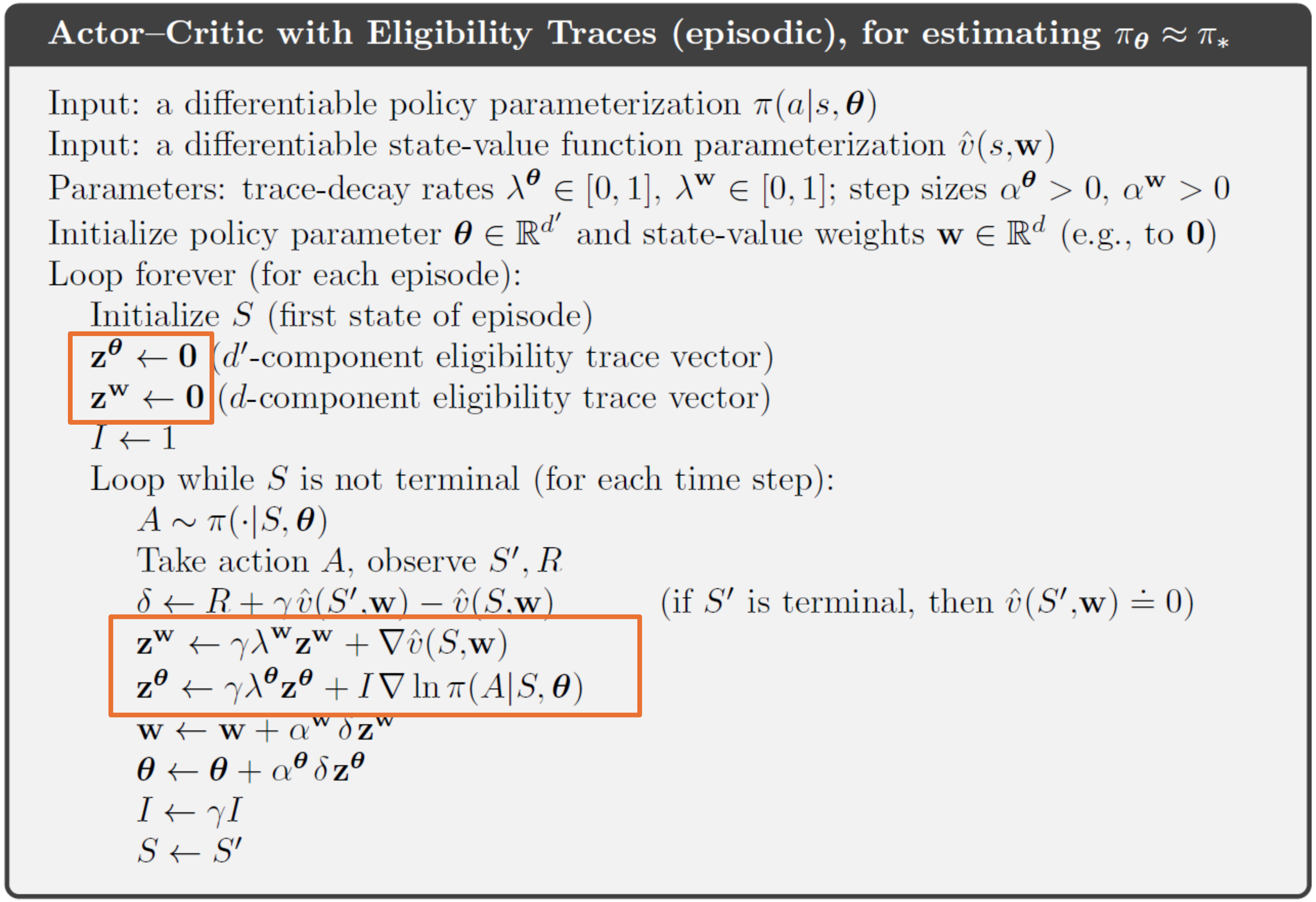
Update the critic (value function) = $TD(0)$

Update the actor (policy) = REINFORCE

Actor-Critic Methods with Eligibility Traces



```
One-step Actor–Critic (episod
Input: a differentiable policy parar
Input: a differentiable state-value f
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}}$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$
Loop forever (for each episode):
    Initialize $S$ (first state of episod
    $I \leftarrow 1$
    Loop while $S$ is not terminal (fo
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action $A$, observe $S', R$
        $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
```



```
Actor–Critic with Eligibility Traces (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$
Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize $S$ (first state of episode)
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ ($d'$-component eligibility trace vector)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ ($d$-component eligibility trace vector)
    $I \leftarrow 1$
    Loop while $S$ is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action $A$, observe $S', R$
        $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$        (if $S'$ is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma \lambda^{\mathbf{w}} \mathbf{z}^{\mathbf{w}} + \nabla \hat{v}(S,\mathbf{w})$
        $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma \lambda^{\boldsymbol{\theta}} \mathbf{z}^{\boldsymbol{\theta}} + I \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{z}^{\mathbf{w}}$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \mathbf{z}^{\boldsymbol{\theta}}$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
```

Policy Parameterization for Continuous Actions

- Soft-max in action preferences works for a small number of actions.
- For continuous actions: learn parameters for a distribution.

Example: choose a continuous actions from a Gaussian distribution:

$$\pi(a|s,\boldsymbol{\theta}) = \frac{1}{\sigma(s,\boldsymbol{\theta})\sqrt{2\pi}} \exp\left(-\frac{\left(a-\mu(s,\boldsymbol{\theta})\right)^2}{2\sigma(s,\boldsymbol{\theta})^2}\right)$$

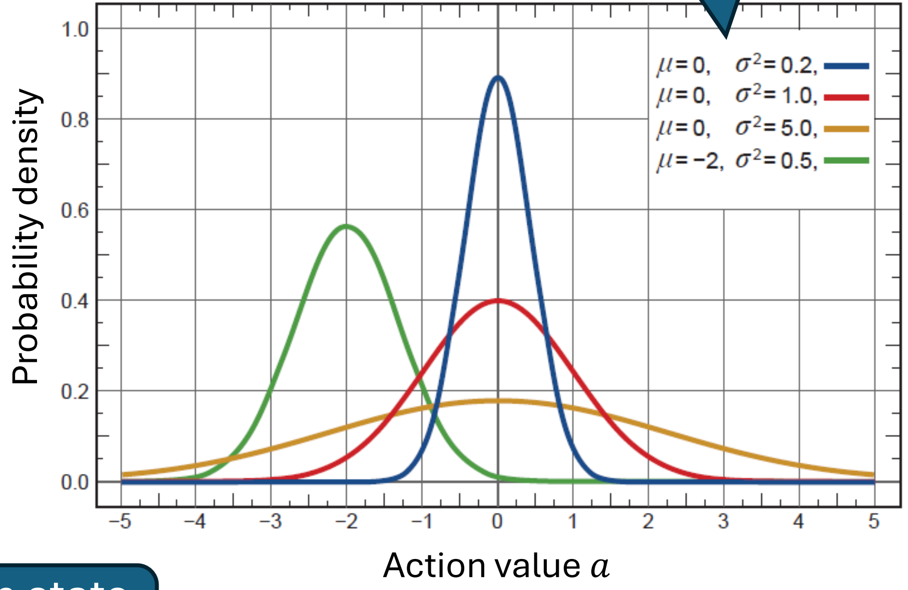
where often

$$\mu(s,\boldsymbol{\theta}) = \boldsymbol{\theta}_\mu^\top \boldsymbol{x}_\mu(s)$$
$$\sigma(s,\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}_\sigma^\top \mathbf{x}_\sigma(\mathrm{s}))$$

Is learned with $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\boldsymbol{\mu}}, \boldsymbol{\theta}_{\boldsymbol{\sigma}}]$

Linear in state features

Depend on state and learned parameters θ



Summary

- Learning a parameterized policy without action-values estimates has many advantages:
  - Learn probabilities for taking an action (stochastic policies).
  - Can handle continuous action spaces.
  - Policy gradient theorem provides strong performance bounds.

- In general, policy gradient methods, including actor-critic, have a different set of strengths and weaknesses compared to action-value methods, and the best choice depends on the problem.