Reinforcement Learning

Policy Gradient Methods

U Kang
Seoul National University

In This Lecture

- Policy approximation
- Policy gradient theorem
- REINFORCE: MC policy gradient
- Actor-critic methods

Overview

- So far all the methods have been action-value methods; they learned the values of actions and then selected actions based on their estimated action values
- Their policies would not even exist without the action-value estimates
- Instead, we consider methods that instead learn a parameterized policy which can select actions without consulting a value function
- A value function may still be used to learn the policy parameter, but is not required for action selection
- We use the notation $\theta \in R^{d'}$ for the policy's parameter vector
- Thus we write $\pi(a|s, \theta) = \Pr\{A_t = a|S_t = s, \theta_t = \theta\}$ for the probability that action a is taken at time t given that the environment is in state s at time t with parameter θ
- If a method uses a learned value function as well, then the value function's weight vector is denoted $w \in R^d$ as usual, as in $\hat{v}(s, w)$

Overview

- We consider *policy gradient* methods for learning the policy parameter based on the gradient of some scalar performance measure $J(\theta)$ with respect to the policy parameter
- These methods seek to maximize performance, so their updates approximate gradient ascent in J:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

- $\widehat{\nabla J(\theta_t)}$ is a stochastic estimate whose expectation approximates the gradient of the performance measure with respect to its argument θ_t
- All methods that follow this general schema we call policy gradient methods, whether or not they also learn an approximate value function
- Methods that learn approximations to both policy and value functions are often called actor–critic methods, where 'actor' is a reference to the learned policy, and 'critic' refers to the learned value function, usually a state-value function

Outline

➡ ☐ **Policy Approximation and its Advantages**

☐ The Policy Gradient Theorem

☐ REINFORCE: MC Policy Gradient

☐ REINFORCE with Baseline

☐ Actor-Critic Methods

☐ Policy Gradient for Continuing Problems

☐ Policy Parameterization for Continuous Actions

☐ Conclusion

Policy Approximation and its Advantages

- In policy gradient (PG) methods, the policy can be parameterized in any way, as long as $\pi(a|s,\theta)$ is differentiable with respect to its parameters θ; i.e., $\nabla_\theta \pi(a|s,\theta)$ exists and is finite
- In practice, to ensure exploration we generally require that the policy never becomes deterministic (i.e., $\pi(a|s,\theta) \in (0,1)$, for all s, a, and θ
- We introduce the most common parameterization for discrete action spaces and point out the advantages it offers over action-value methods
- Policy-based methods also offer useful ways of dealing with continuous action spaces (discussed later in this lecture)

Policy Approximation and its Advantages

- If the action space is discrete and not too large, then a natural and common kind of parameterization is to form parameterized numerical preferences $h(s, a, \theta) \in R$ for each state–action pair
- The actions with the highest preferences in each state are given the highest probabilities of being selected, for example, according to an exponential soft-max distribution:

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

- The action preferences $h(s, a, \theta)$ can be parameterized arbitrarily
  - E.g., ANN (as in AlphaGo)
  - E.g., linear: $h(s, a, \theta) = \theta^T x(s, a)$, where $x(s, a) \in R^{d'}$ is a feature vector

Policy Approximation and its Advantages

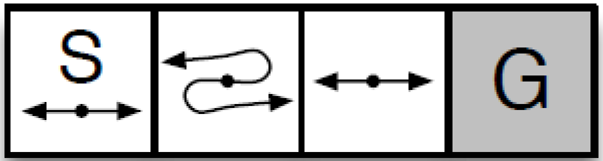
- Advantages of parameterized policies w/ soft-max preferences

- 1) The approximate policy can approach a deterministic policy, whereas with ϵ-greedy action selection over action values there is always an ϵ probability of selecting a random action

- 2) It enables the selection of actions with arbitrary probabilities
  - In problems with significant FA, the best approximate policy may be stochastic
  - E.g., in card games with imperfect information the optimal play is often to do two different things with specific probabilities, such as when bluffing in Poker
  - Action-value methods have no natural way of finding stochastic optimal policies, whereas policy approximating methods can

Example: Short Corridor with Switched Actions

- Reward: −1 per step
- Actions: right and left. Note that in the second state the roles of right and left are reversed, so that right moves to the left and left moves to the right.
- The problem is difficult because all the states appear identical under the FA; in particular, we define $x(s, right) = [1,0]^T$ and $x(s, left) = [0,1]^T$, for all s
- An action-value method with ϵ-greedy action selection is forced to choose between just two policies: choosing right with high probability $1 - \epsilon/2$ on all steps or choosing left with the same high probability on all time steps
- If ϵ = 0.1, then these two policies achieve a value (at the start state) of less than −44 and −82, respectively

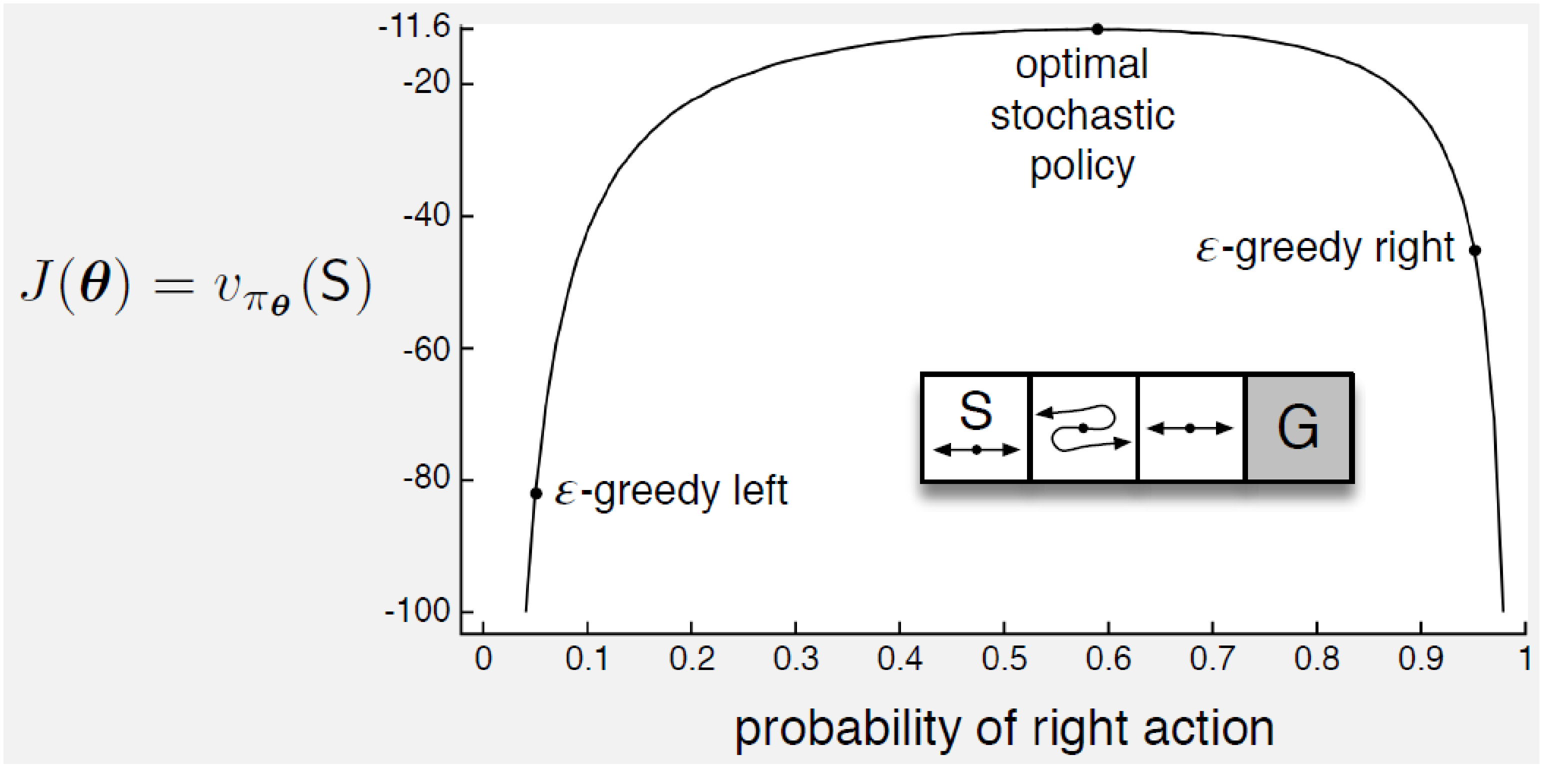


Sutton and Barto, Reinforcement Learning, 2018

Example: Short Corridor with Switched Actions

- A method can do significantly better if it can learn a specific probability with which to select right
- The best probability is about 0.59, which achieves a value of about −11.6

Sutton and Barto, Reinforcement Learning, 2018



Policy Approximation and its Advantages

- Advantages of parameterized policies w/ soft-max preferences

- 3) Compared to action-value parameterization, the policy may be a simpler function to approximate
  - Problems vary in the complexity of their policies and action-value functions
  - For some, the action-value function is simpler and thus easier to approximate
  - For others, the policy is simpler. In this case a policy-based method will typically learn faster and yield a superior asymptotic policy

- 4) The choice of policy parameterization is sometimes a good way of injecting prior knowledge about the desired form of the policy into the RL system
  - This is often the most important reason for using a policy-based learning method

Policy Approximation and its Advantages

- Advantages of parameterized policies w/ soft-max preferences

- 5) It naturally handles continuous action spaces
  - Action-value methods cannot (e.g., ϵ-greedy)

Outline

- ✓ Policy Approximation and its Advantages
- ☐ **The Policy Gradient Theorem**
- ☐ REINFORCE: MC Policy Gradient
- ☐ REINFORCE with Baseline
- ☐ Actor-Critic Methods
- ☐ Policy Gradient for Continuing Problems
- ☐ Policy Parameterization for Continuous Actions
- ☐ Conclusion

The Policy Gradient Theorem

- There is also an important theoretical advantage of policy parameterization over ϵ-greedy action selection

- With continuous policy parameterization the action probabilities change smoothly as a function of the learned parameter, whereas in ϵ -greedy selection the action probabilities may change dramatically for an arbitrarily small change in the estimated action values, if that change results in a different action having the maximal value

- Largely because of this, stronger convergence guarantees are available for policy-gradient methods than for action-value methods

The Policy Gradient Theorem

- For the episodic case, we define the performance measure as the value of the start state of the episode
- We can simplify the notation without losing any meaningful generality by assuming that every episode starts in some particular (non-random) state s_0
- Then, in the episodic case we define performance as

$$J(\theta) \doteq v_{\pi_\theta}(s_0)$$

- where v_{π_θ} is the true value function for π_θ, the policy determined by θ

The Policy Gradient Theorem

- Policy gradient theorem provides an analytic expression for the gradient of performance with respect to the policy parameter

- Policy gradient theorem

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

- In the episodic case, the constant of proportionality is the average length of an episode, and in the continuing case it is 1, so that the relationship is actually an equality

- Intuition of policy gradient
  - The contribution of each parameter to the value function is proportional to the change of average action value, which is related to the change of action probability by the parameter

Outline

- ✓ Policy Approximation and its Advantages
- ✓ The Policy Gradient Theorem
- ➡ ☐ **REINFORCE: MC Policy Gradient**
- ☐ REINFORCE with Baseline
- ☐ Actor-Critic Methods
- ☐ Policy Gradient for Continuing Problems
- ☐ Policy Parameterization for Continuous Actions
- ☐ Conclusion

REINFORCE: MC Policy Gradient

- Recall our overall strategy of stochastic gradient ascent requires a way to obtain samples such that the expectation of the sample gradient is proportional to the actual gradient

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

- The sample gradients need only be proportional to the gradient because any constant of proportionality can be absorbed into the step size α
- The policy gradient theorem gives an exact expression proportional to the gradient; all that is needed is some way of sampling whose expectation equals or approximates this expression

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

- The right-hand side of the policy gradient theorem is a sum over states weighted by how often the states occur under the target policy π; if π is followed, then states will be encountered in these proportions

REINFORCE: MC Policy Gradient

- Thus,

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
$$= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta)\right]$$

- We could stop here and instantiate our stochastic gradient-ascent algorithm as

$$\theta_{t+1} \doteq \theta_t + \alpha \sum_a \hat{q}(S_t,a,w) \nabla \pi(a|S_t,\theta)$$

- where $\hat{q}$ is some learned approximation to q_π
- This algorithm, called an all-actions method because its update involves all of the actions, is promising and deserving of further study
- But our current interest is the classical REINFORCE algorithm (Willams, 1992) whose update at time t involves just A_t, the one action actually taken at time t

REINFORCE: MC Policy Gradient

- In REINFORCE,

$$\nabla J(\theta) = \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\theta) q_\pi(S_t,a) \frac{\nabla\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}\right]$$

$$= \mathbb{E}_\pi\left[q_\pi(S_t,A_t)\frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}\right] \quad \text{(replacing } a \text{ by the sample } A_t \sim \pi)$$

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

- The final expression in brackets is exactly what is needed, a quantity that can be sampled on each time step whose expectation is equal to the gradient
- Using this sample to instantiate our generic stochastic gradient ascent algorithm yields the REINFORCE update:

$$\theta_{t+1} \doteq \theta_t + \alpha G_t \frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

REINFORCE: MC Policy Gradient

$$\theta_{t+1} \doteq \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

- Intuition of REINFORCE
  - Each increment is proportional to the product of a return G_t and a vector, the gradient of the probability of taking the action actually taken divided by the probability of taking that action
  - The vector is the direction in parameter space that most increases the probability of repeating the action A_t on future visits to state S_t
  - The update increases the parameter vector in this direction proportional to the return, and inversely proportional to the action probability
  - The former makes sense because it causes the parameter to move most in the directions that favor actions that yield the highest return
  - The latter makes sense because otherwise actions that are selected frequently are at an advantage (the updates will be more often in their direction) and might win out even if they do not yield the highest return

REINFORCE: MC Policy Gradient

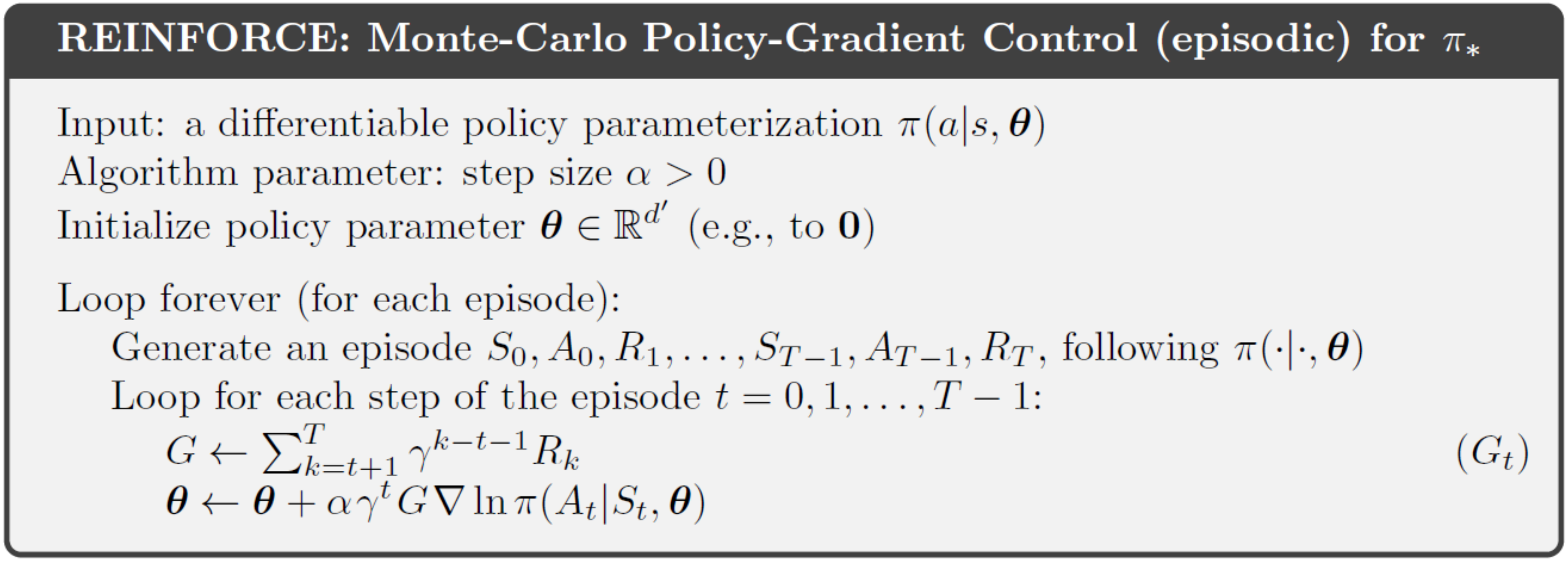
REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

```
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 $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                    $(G_t)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
```

- Note that REINFORCE uses the complete return from time t, which includes all future rewards up until the end of the episode
- REINFORCE is an MC algorithm and is well defined only for the episodic case with all updates made in retrospect after the episode is completed

Sutton and Barto, Reinforcement Learning, 2018

REINFORCE: MC Policy Gradient



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

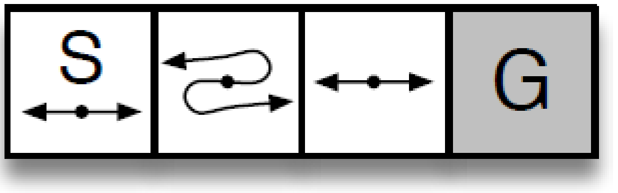
```
Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

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 $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                    $(G_t)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
```

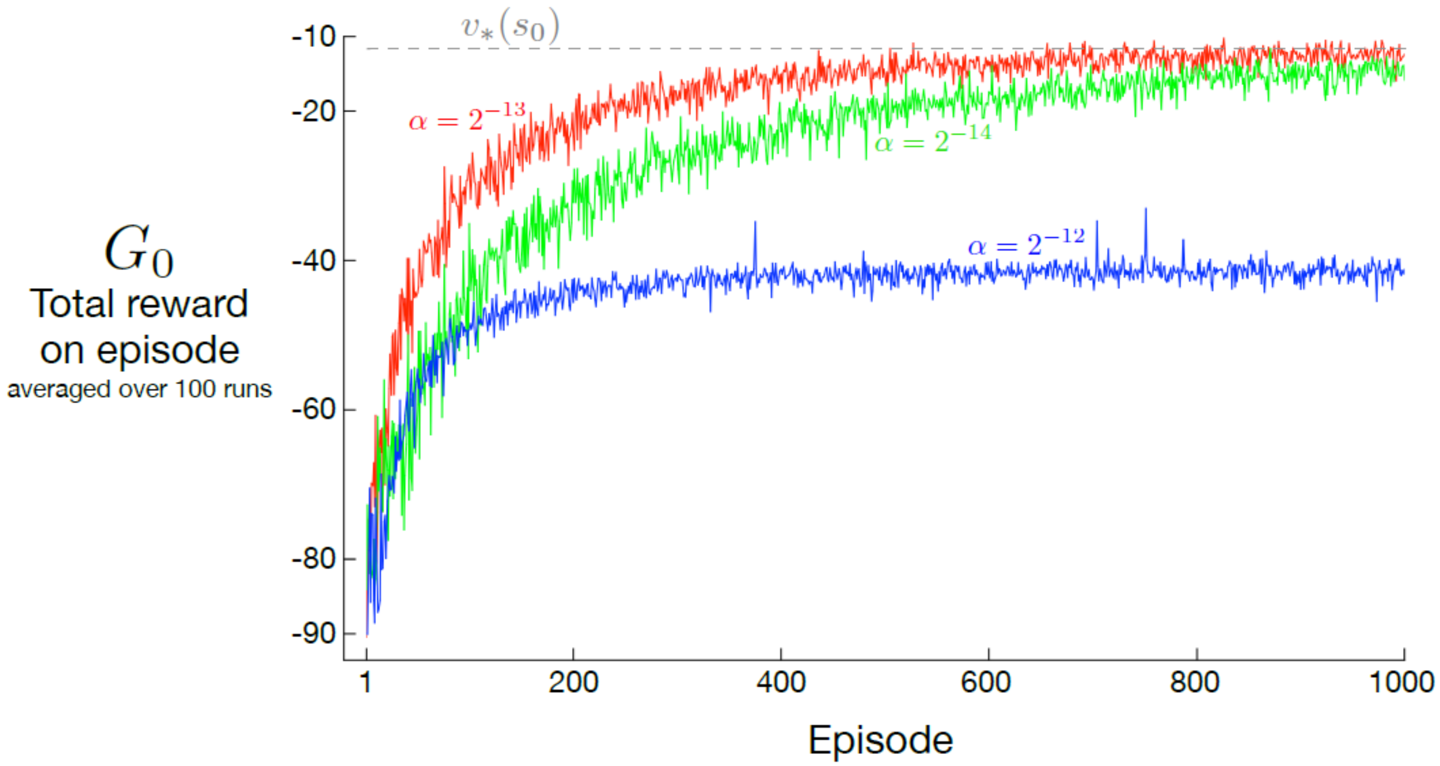
- $\nabla \ln \pi(A_t|S_t, \theta_t) = \frac{\nabla \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$ is called the *eligibility vector*
- Also note γ in the algorithm; all of the ideas go through in the discounted case

Sutton and Barto, Reinforcement Learning, 2018

REINFORCE: MC Policy Gradient

- REINFORCE on the short-corridor gridworld





Sutton and Barto, Reinforcement Learning, 2018

REINFORCE: MC Policy Gradient

- As a stochastic gradient method, REINFORCE has good theoretical convergence properties
- By construction, the expected update over an episode is in the same direction as the performance gradient
- This assures an improvement in expected performance for sufficiently small α, and convergence to a local optimum under standard stochastic approximation conditions for decreasing α
- However, as an MC method REINFORCE may be of high variance and thus produce slow learning
    - REINFORCE with baseline addresses the problem

Outline

- ✓ Policy Approximation and its Advantages
- ✓ The Policy Gradient Theorem
- ✓ REINFORCE: MC Policy Gradient
- ➡ ☐ **REINFORCE with Baseline**
- ☐ Actor-Critic Methods
- ☐ Policy Gradient for Continuing Problems
- ☐ Policy Parameterization for Continuous Actions
- ☐ Conclusion

REINFORCE with Baseline

- The policy gradient theorem can be generalized to include a comparison of the action value to an arbitrary baseline $b(s)$:

$$\nabla J(\theta) \propto \sum_a \mu(s) \sum_a \big(q_\pi(s,a) - b(s)\big) \nabla\pi(a|s,\theta)$$

- The baseline can be any function, even a random variable, as long as it does not vary with a; the equation remains valid because the subtracted quantity is zero

$$\sum_a b(s)\nabla\pi(a|s,\theta) = b(s)\nabla\sum_a \pi(a|s,\theta) = b(s)\nabla 1 = 0$$

- The policy gradient theorem with baseline can be used to derive a new version of REINFORCE that includes a general baseline:

$$\theta_{t+1} \doteq \theta_t + \alpha\big(G_t - b(S_t)\big)\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

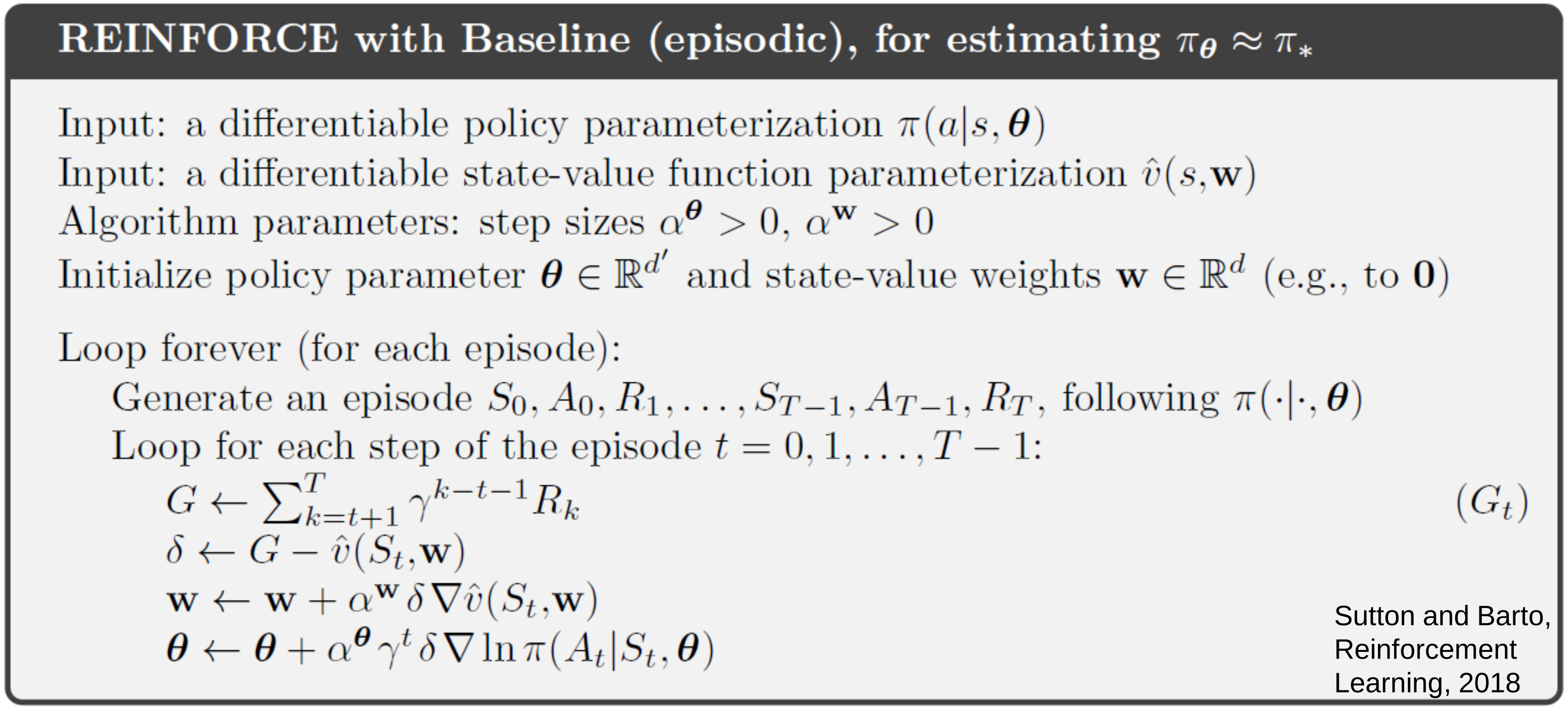
REINFORCE with Baseline

$$\theta_{t+1} \doteq \theta_t + \alpha\big(G_t - b(S_t)\big)\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

- Because the baseline could be uniformly zero, this update is a strict generalization of REINFORCE
- In general, the baseline leaves the expected value of the update unchanged, but it can have a large effect on its variance
- The baseline can significantly reduce the variance (and thus speed the learning)
- The baseline should vary with state in MDPs
- In some states all actions have high values and we need a high baseline to differentiate the higher valued actions from the less highly valued ones; in other states all actions will have low values and a low baseline is appropriate

REINFORCE with Baseline

- One natural choice for the baseline is an estimate of the state value, $\hat{v}(S_t, w)$, where $w \in R^m$ is a learned weight vector
- Because REINFORCE is an MC method for learning the policy parameter θ, it seems natural to also use an MC method to learn the state-value weights w



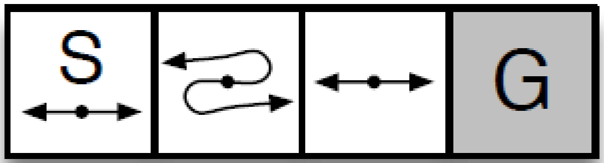
REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

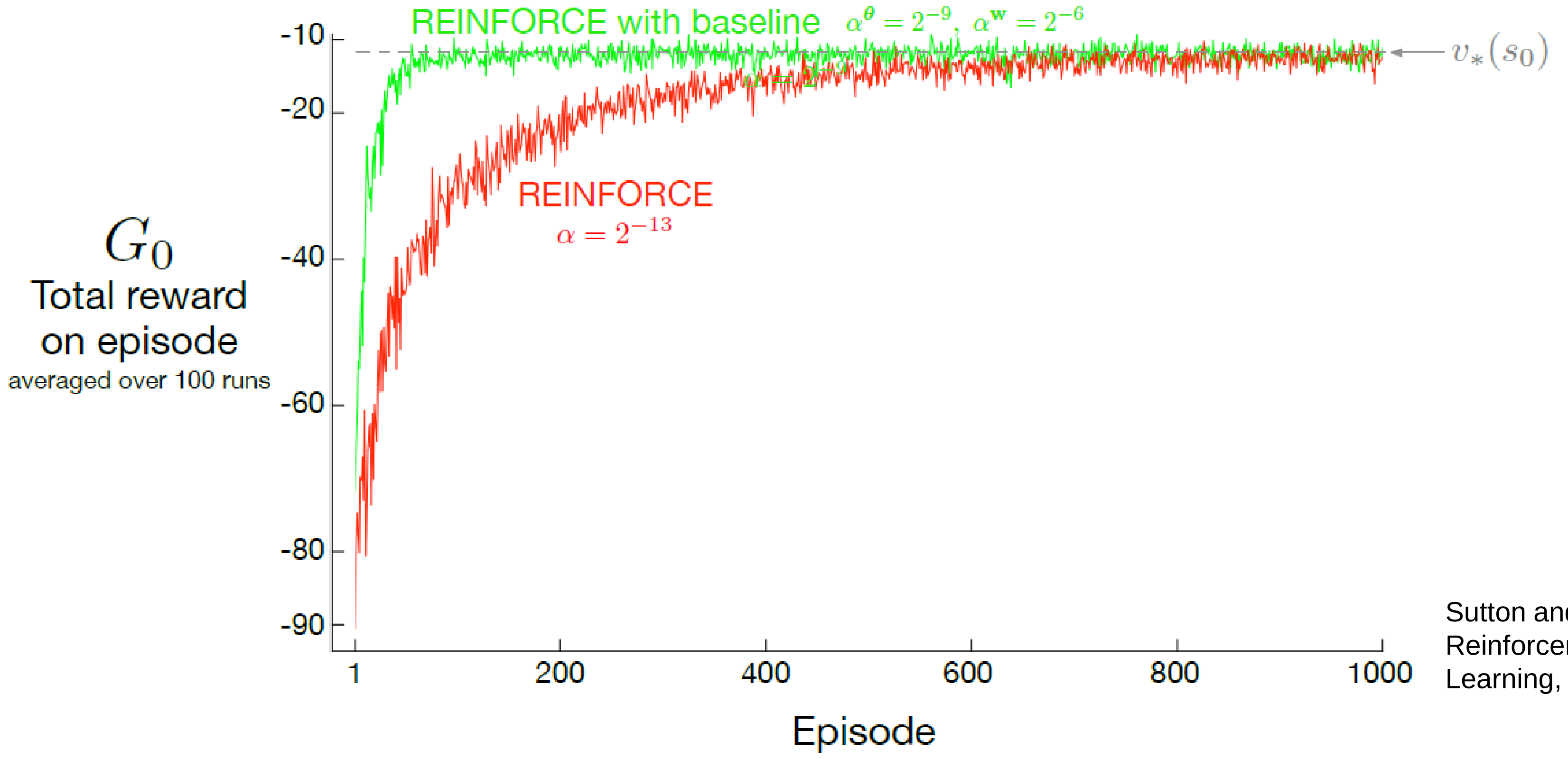
```
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    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 $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                    $(G_t)$
        $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$
```

Sutton and Barto, Reinforcement Learning, 2018

REINFORCE with Baseline

- REINFORCE with/ without a baseline on the short-corridor gridworld





Sutton and Barto, Reinforcement Learning, 2018

Outline

- ✓ Policy Approximation and its Advantages
- ✓ The Policy Gradient Theorem
- ✓ REINFORCE: MC Policy Gradient
- ✓ REINFORCE with Baseline
- ☐ **Actor-Critic Methods**
- ☐ Policy Gradient for Continuing Problems
- ☐ Policy Parameterization for Continuous Actions
- ☐ Conclusion

Actor-Critic Methods

- Actor-critic methods
  - Learn approximations to both policy and value functions
  - Critic: updates action-value function parameter
  - Actor: updates policy parameter in direction suggested by critic

- Although the REINFORCE-with-baseline method learns both a policy and a state-value function, we do not consider it to be an actor–critic method
  - Its state-value function is not used for bootstrapping (updating the value estimate for a state from the estimated values of subsequent states), but only as a baseline for the state whose estimate is being updated

- The bias introduced through bootstrapping is often beneficial because it reduces variance and accelerates learning

Actor-Critic Methods

- REINFORCE with baseline is unbiased and will converge asymptotically to a local minimum, but like all MC methods it tends to learn slowly (produce estimates of high variance) and to be inconvenient to implement online or for continuing problems

- With TD methods we can eliminate these inconveniences, and through multi-step methods we can flexibly choose the degree of bootstrapping

- In order to gain these advantages in the case of policy gradient methods we use actor–critic methods with a bootstrapping critic

Actor-Critic Methods

- First consider one-step actor–critic methods, the analog of the TD methods such as TD(0), Sarsa(0), and Q-learning
- The main appeal of one-step methods is that they are fully online and incremental, yet avoid the complexities of eligibility traces
- They are a special case of the eligibility trace methods, and not as general, but easier to understand
- One-step actor–critic methods replace the full return of REINFORCE with the one-step return (and use a learned state-value function as the baseline) as follows:

$$\begin{aligned}
\theta_{t+1} &\doteq \theta_t + \alpha\left(G_{t:t+1} - \hat{v}(S_t, w)\right)\frac{\nabla\pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \\
&= \theta_t + \alpha\left(R_{t+1} + \gamma\hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)\right)\frac{\nabla\pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \\
&= \theta_t + \alpha\delta_t\frac{\nabla\pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}.
\end{aligned}$$

Actor-Critic Methods

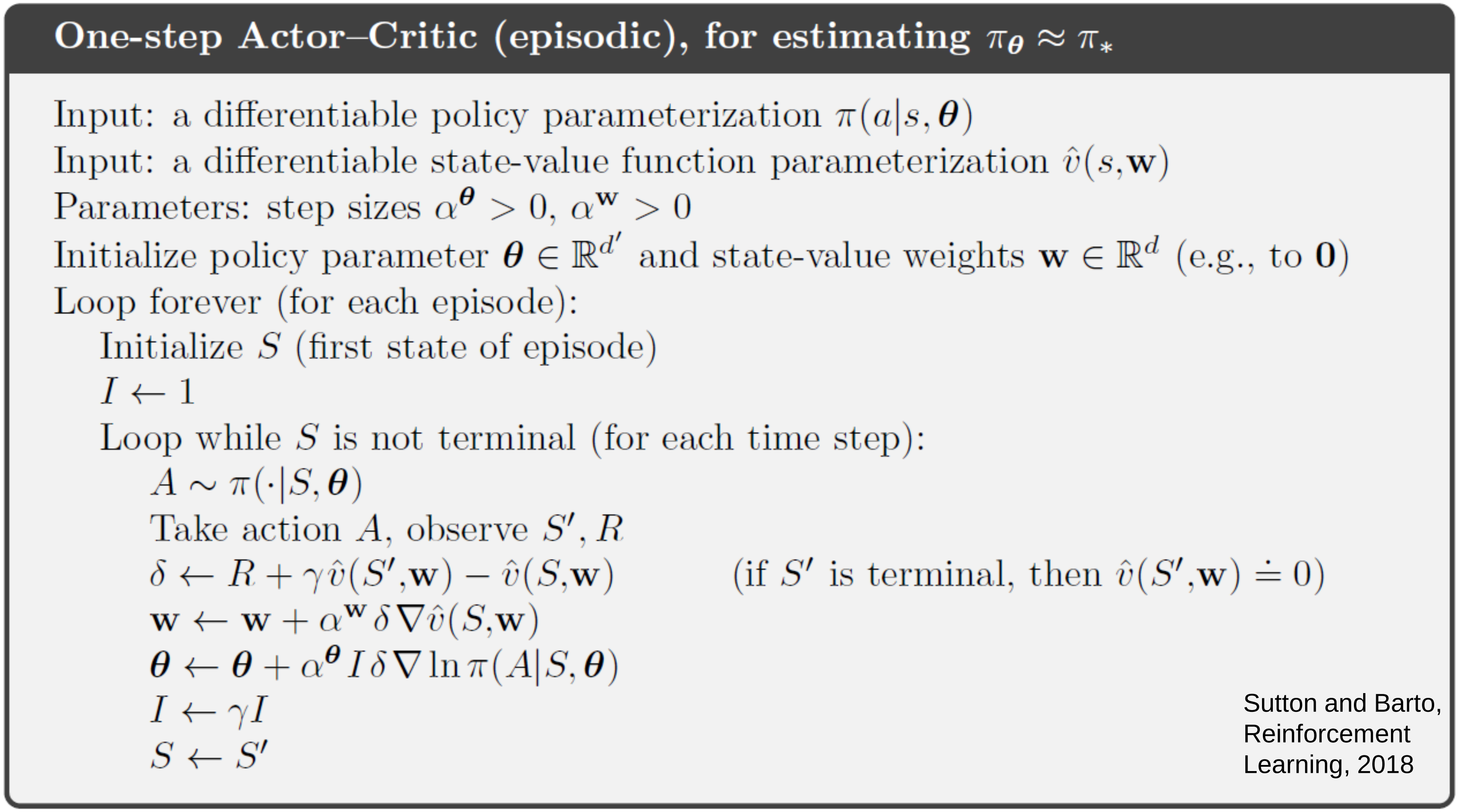
- One-step actor–critic methods replace the full return of REINFORCE with the one-step return (and use a learned state-value function as the baseline) as follows:

$$\begin{aligned}\theta_{t+1} &\doteq \theta_t + \alpha\left(G_{t:t+1} - \hat{v}(S_t, w)\right)\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \\ &= \theta_t + \alpha\left(R_{t+1} + \gamma\hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)\right)\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \\ &= \theta_t + \alpha\delta_t\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}.\end{aligned}$$

- Actor-critic methods
  - Learn approximations to both policy and value functions
  - Critic: updates action-value function parameter
  - Actor: updates policy parameter in direction suggested by critic

Actor-Critic Methods

- The natural state-value-function learning method to pair with this is semi-gradient TD(0)



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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\,\delta\,\nabla\hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\,I\,\delta\,\nabla\ln\pi(A|S,\boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
```

Sutton and Barto, Reinforcement Learning, 2018

Actor-Critic Methods

- The generalizations to the forward view of n-step methods and then to a λ-return algorithm are straightforward. The one-step return $G_{t:t+1}$ is replaced by $G_{t:t+n}$ and G_t^λ
- The backward view of the λ-return is also straightforward, using separate eligibility traces for the actor and critic

Actor-Critic Methods

- Reminder: eligibility trace
  - Offline λ-return: $w_{t+1} \doteq w_t + \alpha\left[G_t^\lambda - \hat{v}(S_t, w_t)\right]\nabla\hat{v}(S_t, w_t), \qquad t = 0, \dots, T-1$
  - (Online) TD(λ):

$$z_{-1} \doteq 0,$$

$$z_t \doteq \gamma\lambda z_{t-1} + \nabla\hat{v}(S_t, w_t), \qquad 0 \leq t \leq T$$

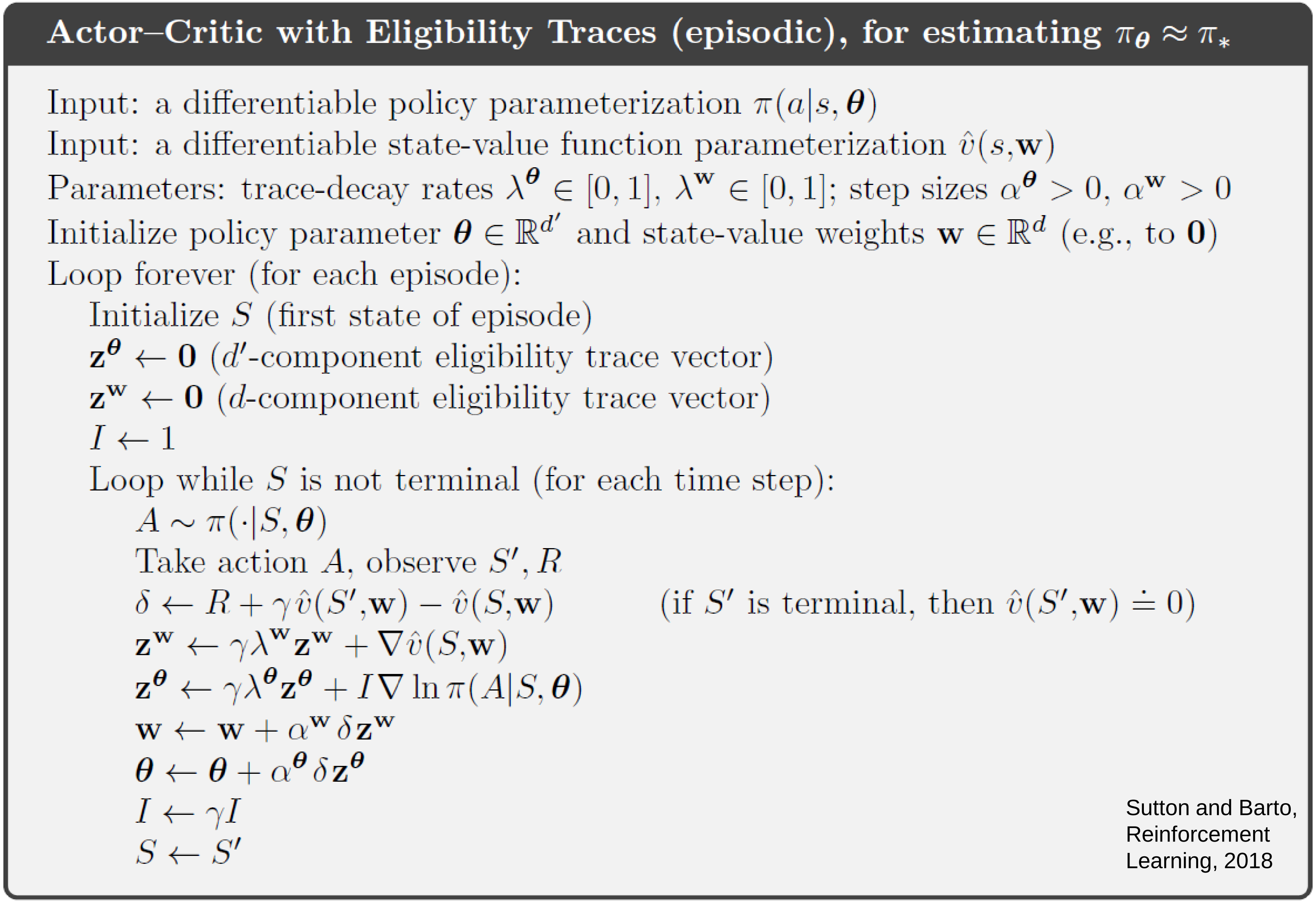
$$\delta_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$$

$$w_{t+1} = w_t + \alpha\delta_t z_t$$

  - Offline one-step actor-critic method:

$$\theta_{t+1} \doteq \theta_t + \alpha(R_{t+1} + \gamma\hat{v}(S_{t+1}, w) - \hat{v}(S_t, w))\frac{\nabla\pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}$$

Actor-Critic Methods



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 $\hat v(S',\mathbf{w}) \doteq 0$)
        $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat v(S,\mathbf{w})$
        $\mathbf{z}^{\boldsymbol\theta} \leftarrow \gamma\lambda^{\boldsymbol\theta}\mathbf{z}^{\boldsymbol\theta} + I\nabla \ln\pi(A|S,\boldsymbol\theta)$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\,\delta\,\mathbf{z}^{\mathbf{w}}$
        $\boldsymbol\theta \leftarrow \boldsymbol\theta + \alpha^{\boldsymbol\theta}\,\delta\,\mathbf{z}^{\boldsymbol\theta}$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
```

Sutton and Barto, Reinforcement Learning, 2018

Outline

- ✓ Policy Approximation and its Advantages
- ✓ The Policy Gradient Theorem
- ✓ REINFORCE: MC Policy Gradient
- ✓ REINFORCE with Baseline
- ✓ Actor-Critic Methods
- ➡ ☐ **Policy Gradient for Continuing Problems**
- ☐ Policy Parameterization for Continuous Actions
- ☐ Conclusion

Policy Gradient for Continuing Problems

- For continuing problems w/o episode boundaries we need to define the average rate of reward per time step:

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

- where μ is the steady-state distribution under π, $\mu(s) = \lim_{t\to\infty} \Pr\{S_t = s | A_{0:t}{\sim}\pi\}$, which is assumed to exist and to be independent of S_0 (an ergodicity assumption)
- This is the special distribution under which, if you select actions according to π, you remain in the same distribution:

$$\sum_s \mu(s) \sum_a \pi(a|s,\theta) p(s'|s,a) = \mu(s'), \qquad for\ all\ s' \in S$$

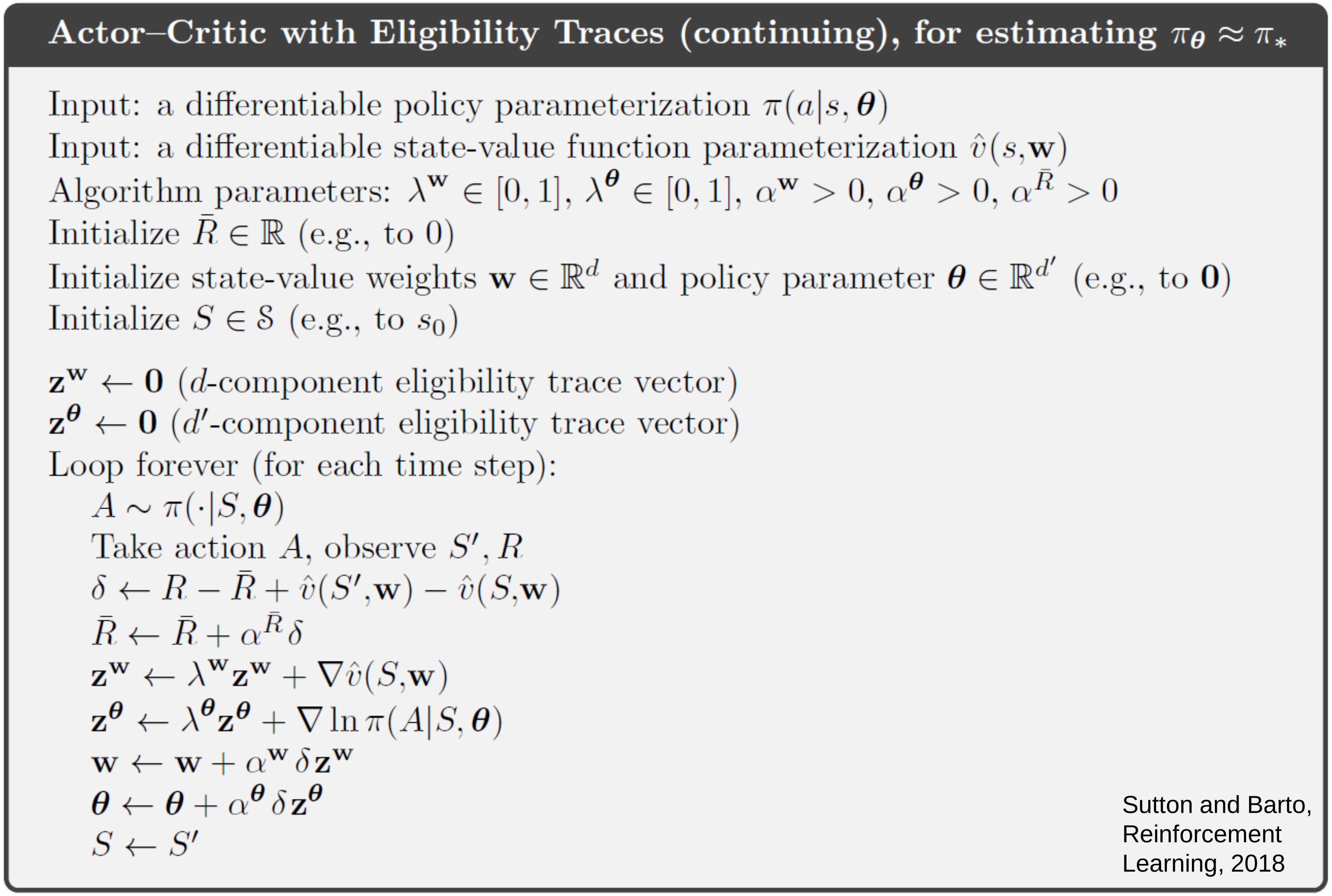
Policy Gradient for Continuing Problems

- Naturally, in the continuing case, we define values, $v_\pi(s) = E_\pi[G_t|S_t = s]$ and $q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$, with respect to the differential return:

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

- With these alternate definitions, the policy gradient theorem as given for the episodic case remains true for the continuing case

Policy Gradient for Continuing Problems



Actor–Critic with Eligibility Traces (continuing), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

```
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: $\lambda^{\mathbf{w}} \in [0,1]$, $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\alpha^{\mathbf{w}} > 0$, $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\bar{R}} > 0$
Initialize $\bar{R} \in \mathbb{R}$ (e.g., to 0)
Initialize state-value weights $\mathbf{w} \in \mathbb{R}^d$ and policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to $s_0$)

$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ ($d$-component eligibility trace vector)
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ ($d'$-component eligibility trace vector)
Loop forever (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action $A$, observe $S', R$
    $\delta \leftarrow R - \bar{R} + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \alpha^{\bar{R}}\delta$
    $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\,\delta\,\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\,\delta\,\mathbf{z}^{\boldsymbol{\theta}}$
    $S \leftarrow S'$
```

Sutton and Barto, Reinforcement Learning, 2018

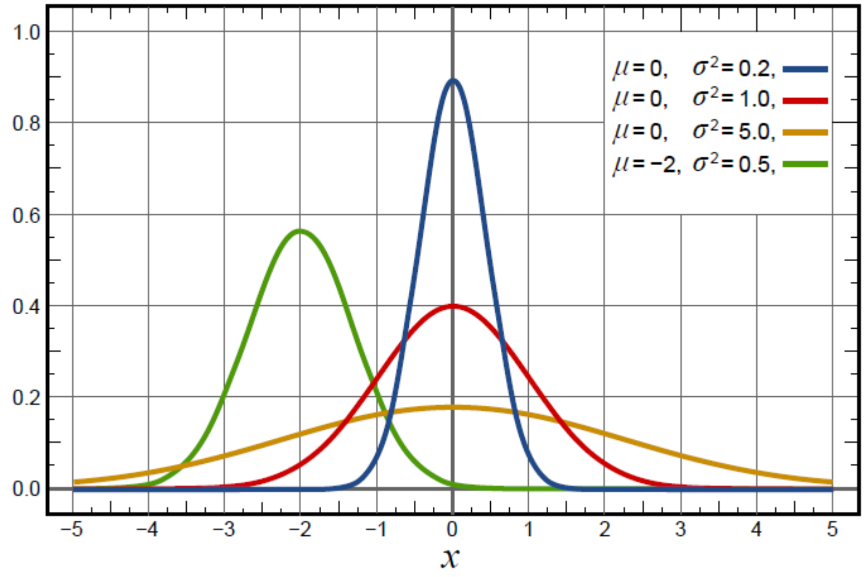
Outline

- ✓ Policy Approximation and its Advantages
- ✓ The Policy Gradient Theorem
- ✓ REINFORCE: MC Policy Gradient
- ✓ REINFORCE with Baseline
- ✓ Actor-Critic Methods
- ✓ Policy Gradient for Continuing Problems
- ☐ **Policy Parameterization for Continuous Actions**
- ☐ Conclusion

Policy Parameterization for Continuous Actions

- Policy-based methods offer practical ways of dealing with large actions spaces, even continuous spaces with an infinite number of actions
- Instead of computing learned probabilities for each of the many actions, we instead learn statistics of the probability distribution
- E.g., the action set might be the real numbers, with actions chosen from a normal (Gaussian) distribution

$$p(x) \doteq \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



Sutton and Barto, Reinforcement Learning, 2018

Policy Parameterization for Continuous Actions

- To produce a policy parameterization, the policy can be defined as the normal probability density over a real-valued scalar action, with mean and standard deviation given by parametric FAs that depend on the state. That is,

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

- where $\mu: S \times R^{d'} \rightarrow R$ and $\sigma: S \times R^{d'} \rightarrow R^+$ are two parameterized function approximators

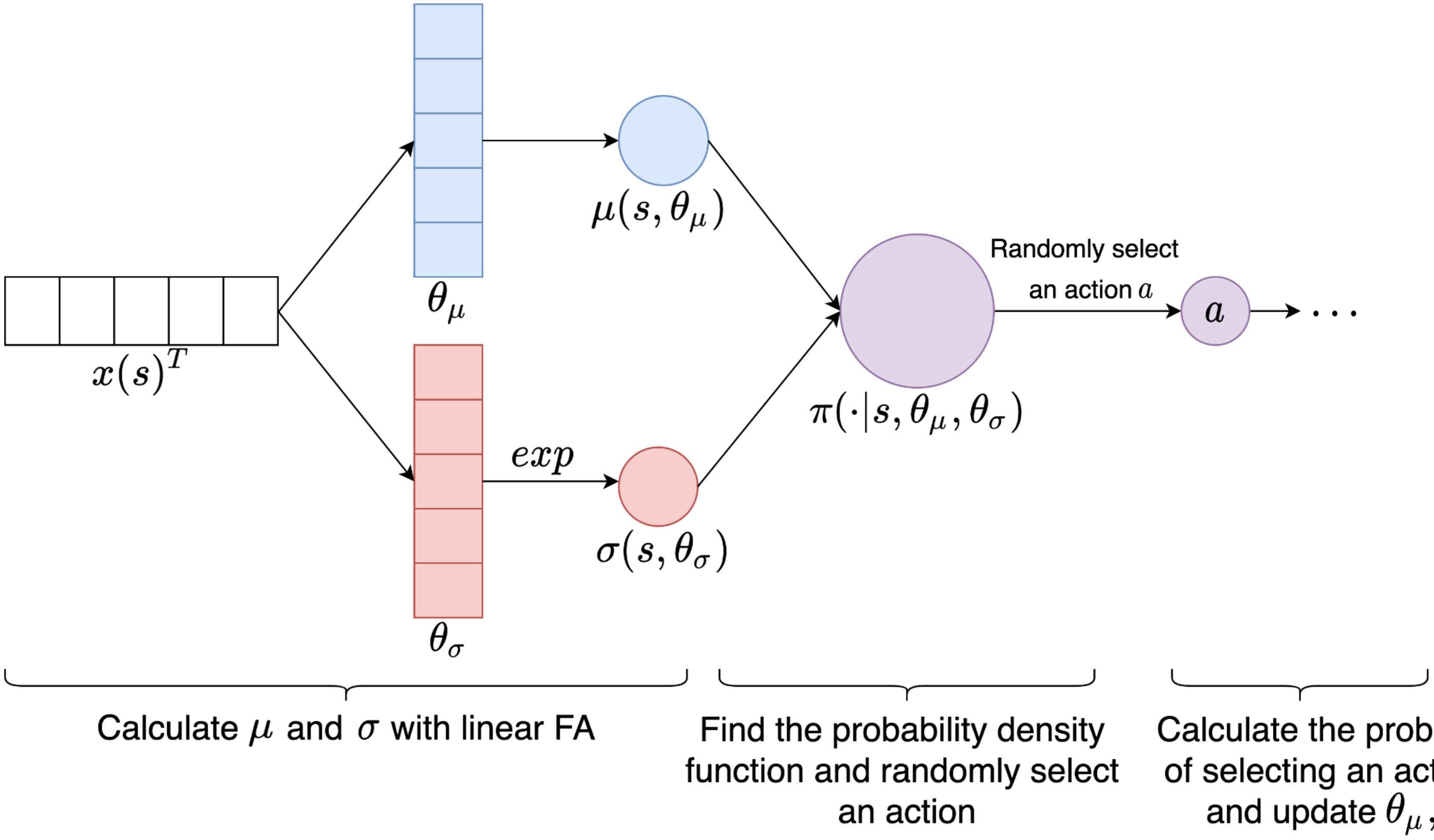
Policy Parameterization for Continuous Actions

- To complete the example we need only give a form for these approximators
- For this we divide the policy's parameter vector into two parts, $\theta = [\theta_\mu, \theta_\sigma]^T$, one part to be used for the mean and one part for the standard deviation
- The mean can be approximated as a linear function
- The standard deviation must always be positive and is better approximated as the exponential of a linear function

$$\mu(s,\theta) \doteq {\theta_\mu}^\top x_\mu(s) \qquad \text{and} \qquad \sigma(s,\theta) \doteq \exp\left({\theta_\sigma}^\top x_\sigma(s)\right)$$

- where $x_\mu(s)$ and $x_\sigma(s)$ are state feature vectors

Policy Parameterization for Continuous Actions



θ_μ
$\mu(s, \theta_\mu)$
$x(s)^T$
exp
θ_σ
$\sigma(s, \theta_\sigma)$
$\pi(\cdot|s, \theta_\mu, \theta_\sigma)$
Randomly select an action a
a
$\cdots$
Calculate μ and σ with linear FA
Find the probability density function and randomly select an action
Calculate the probability of selecting an action a and update $\theta_\mu, \theta_\sigma$

Outline

- ✓ Policy Approximation and its Advantages
- ✓ The Policy Gradient Theorem
- ✓ REINFORCE: MC Policy Gradient
- ✓ REINFORCE with Baseline
- ✓ Actor-Critic Methods
- ✓ Policy Gradient for Continuing Problems
- ✓ Policy Parameterization for Continuous Actions
- ➡ ☐ **Conclusion**

Conclusion

- We considered methods that learn a parameterized policy that enables actions to be taken without consulting action-value estimates
- We have considered policy-gradient methods which update the policy parameter on each step in the direction of an estimate of the gradient of performance with respect to the policy parameter

Conclusion

- Advantages of policy-gradient methods
  - They can learn specific probabilities for taking the actions (finds stochastic policies)
  - They can learn appropriate levels of exploration and approach deterministic policies asymptotically (due to stochastic policy representation)
  - They can naturally handle continuous action spaces
  - All these things are easy for policy-based methods but awkward or impossible for ϵ-greedy methods and for action-value methods in general
  - On some problems the policy is just simpler to represent parametrically than the value function
  - Theoretical advantage over action-value methods in the form of the policy gradient theorem, which gives an exact formula for how performance is affected by the policy parameter

Conclusion

- Disadvantages of policy-gradient methods
  - Typically converge to a local minimum, rather than a global optimum
  - Slow convergence, since policy improvement is performed in small steps

- REINFORCE: follows directly from the policy gradient theorem
- REINFORCE with baseline: adding a state-value function as a baseline in REINFORCE reduces its variance without introducing bias
- Actor-critic methods
  - Learn approximations to both policy and value functions
  - Critic: updates action-value function parameter
  - Actor: updates policy parameter in direction suggested by critic
  - Using the state-value function for bootstrapping introduces bias but is often desirable for the same reason that bootstrapping TD methods are often superior to MC methods (substantially reduced variance)

Questions?