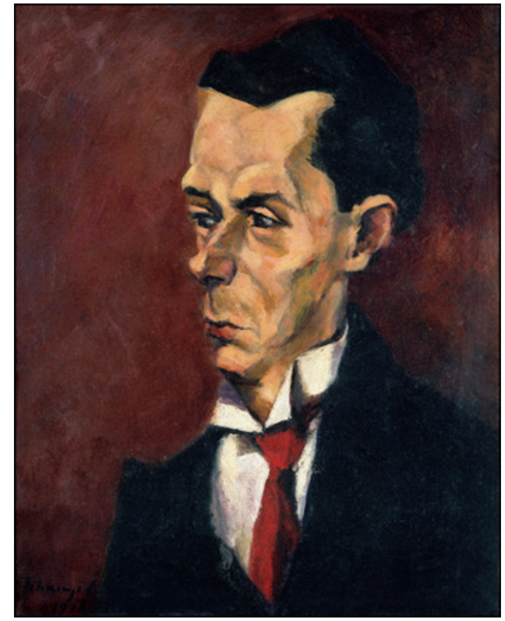
Actors from the Comédie Française, by Antoine Watteau, 1720. Public Domain, https://commons.wikimedia.org/w/index.php?curid=15418670

# Lecture 36: Actor-critic deep reinforcement learning



Mark Hasegawa-Johnson

April 2022



The Critic, by Lajos Tihanyi. Oil on canvas, 1916. Public Domain, https://commons.wikimedia.org/w/index.php?curid=17837438

# Outline

- Two approaches to solving an MDP
- Two approaches to deep reinforcement learning
- Combining the two, in order to solve the problems with either

# Solving an MDP

Remember that, if you know $P(s'|s,a)$, you can solve for the optimum policy $\pi(s)$. This is done by solving Bellman's equation:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s') \qquad \forall s, s'$$

- Bellman's equation is N nonlinear equations in N unknowns (N is the number of states). In general, the only way to solve it is by exhaustively testing every possible policy ($O\{d^N\}$ computations where d is the number of possible actions).

# Two approaches to solving an MDP

We've learned two practical algorithms for solving an MDP:

1. Value Iteration: focuses on finding $U(s)$
2. Policy Iteration: focus on finding $\pi(s)$

# Two approaches to solving an MDP

Value Iteration: focuses on finding $U(s)$

- Initialize with the value of a length-0 path: $U_0(s) = 0$
- Iterate by finding the best value of a length-t path:

$$U_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_{t-1}(s)$$

# Value Iteration

$$U_2(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U_1(s')$$

$U_2(s)$ ($\gamma = 1$)

| | | | |
|---|---|---|---|
| −0.08 | −0.08 | +0.75 | |
| −0.08 | | −0.08 | |
| −0.08 | −0.08 | −0.08 | −0.08 |

$\sum_{s'} P(s'|s,\mathrm{down})U_1(s')$

| | | | |
|---|---|---|---|
| −0.04 | −0.04 | +0.06 | |
| −0.04 | | −0.14 | |
| −0.04 | −0.04 | −0.04 | −0.04 |

$\sum_{s'} P(s'|s,\mathrm{up})U_1(s')$

| | | | |
|---|---|---|---|
| −0.04 | −0.04 | +0.06 | |
| −0.04 | | −0.14 | |
| −0.04 | −0.04 | −0.04 | −0.81 |

$\sum_{s'} P(s'|s,\mathrm{left})U_1(s')$

| | | | |
|---|---|---|---|
| −0.04 | −0.04 | −0.04 | |
| −0.04 | | −0.04 | |
| −0.04 | −0.04 | −0.04 | −0.14 |

$\sum_{s'} P(s'|s,\mathrm{right})U_1(s')$

| | | | |
|---|---|---|---|
| −0.04 | −0.04 | +0.79 | |
| −0.04 | | −0.81 | |
| −0.04 | −0.04 | −0.04 | −0.14 |

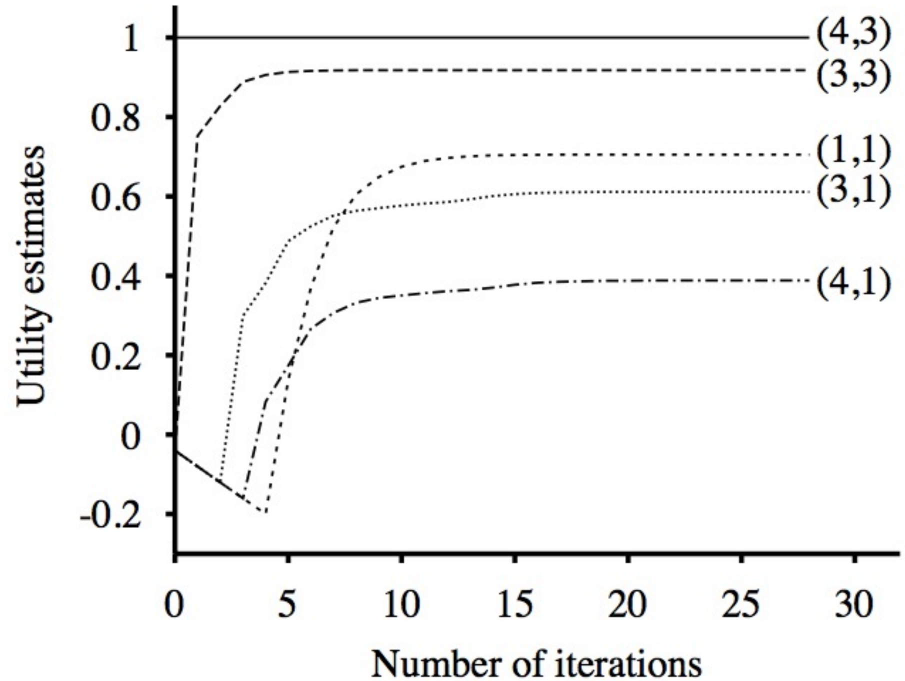
$U_1(s)$

| | | | |
|---|---|---|---|
| −0.04 | −0.04 | −0.04 | |
| −0.04 | | −0.04 | |
| −0.04 | −0.04 | −0.04 | −0.04 |

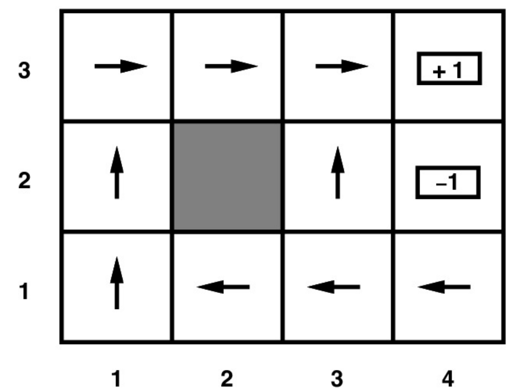
# Value iteration

Optimal utilities with discount factor 1
(Result of value iteration)

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 3 | 0.812 | 0.868 | 0.918 | + 1 |
| 2 | 0.762 | | 0.660 | –1 |
| 1 | 0.705 | 0.655 | 0.611 | 0.388 |




Final policy



# Two approaches to solving an MDP

Policy Iteration: focus on finding $\pi(s)$

– Initialize with a completely arbitrary initial policy, e.g.:

$$\pi_0(s) = \text{Left}$$

– Iterate:

- Policy evaluation: find out the value of each state under current policy:

$$U^\pi(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) U^\pi(s')$$

- Policy improvement: change the action, in each state, to improve value:

$$\pi(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s'|s, a) U^\pi(s)$$

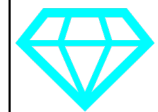
# 1. Policy Evaluation:

$$U^{\pi_0}(s) = R(s) + \gamma \sum_{s'} P(s'|s,\pi_0(s))U^{\pi_0}(s')$$

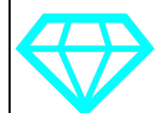
…write it in matrix form:

$$\begin{bmatrix} U^{\pi_0}(1) \\ \vdots \\ U^{\pi_0}(N) \end{bmatrix} = \begin{bmatrix} R\,(1) \\ \vdots \\ R\,(N) \end{bmatrix} + \gamma \begin{bmatrix} P(1|1,\pi(1)) & \cdots & P(N|1,\pi(N)) \\ \vdots & \ddots & \vdots \\ P(1|N,\pi(1)) & \cdots & P(N|N,\pi(N)) \end{bmatrix} \begin{bmatrix} U^{\pi_0}(1) \\ \vdots \\ U^{\pi_0}(N) \end{bmatrix}$$

$\pi_0(s)$:

| | | | |
|---|---|---|---|
| $\rightarrow$ | $\rightarrow$ | $\rightarrow$ |  |
| $\rightarrow$ | | $\rightarrow$ |  |
| $\rightarrow$ | $\rightarrow$ | $\rightarrow$ | $\rightarrow$ |

…and solve it: $U^{\pi_0(s)}(s)$:

| | | | |
|---|---|---|---|
| $+0.50$ | $+0.69$ | $+0.74$ |  |
| $-0.65$ | | $-0.90$ |  |
| $-1.40$ | $-1.44$ | $-1.39$ | $-1.40$ |

# 2. Policy Improvement:

$$U^{\pi_0}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_0(s)) U^{\pi_0}(s')$$

$$\pi_1(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s, \pi_0(s)) U^{\pi_0}(s')$$

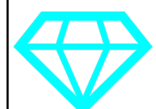
$\pi_0(s)$:

| | | | |
|---|---|---|---|
| $\rightarrow$ | $\rightarrow$ | $\rightarrow$ |  |
| $\rightarrow$ | | $\rightarrow$ |  |
| $\rightarrow$ | $\rightarrow$ | $\rightarrow$ | $\rightarrow$ |

$\pi_1(s)$

| | | | |
|---|---|---|---|
| $\rightarrow$ | $\rightarrow$ | $\rightarrow$ |  |
| $\uparrow$ | | $\uparrow$ |  |
| $\uparrow$ | $\rightarrow$ | $\uparrow$ | $\uparrow$ |

$U^{\pi_0(s)}(s)$:

| | | | |
|---|---|---|---|
| $+0.50$ | $+0.69$ | $+0.74$ |  |
| $-0.65$ | | $-0.90$ |  |
| $-1.40$ | $-1.44$ | $-1.39$ | $-1.40$ |

# Outline

- Two approaches to solving an MDP
- Two approaches to deep reinforcement learning
- Combining the two, in order to solve the problems with either

# Two approaches to deep reinforcement learning

- Deep Q learning: train a network to estimate Q(s,a)
  - Like value iteration: we focus on Q(s,a), which is closely related to U(s)
  - Big problem: Q(s,a) is very noisy, needs lots of smoothing
- Imitation learning: train a network to imitate a human being
  - Like policy iteration: focus directly on estimating $\pi(s)$
  - Big problem: the only way to train this is by imitating a human!

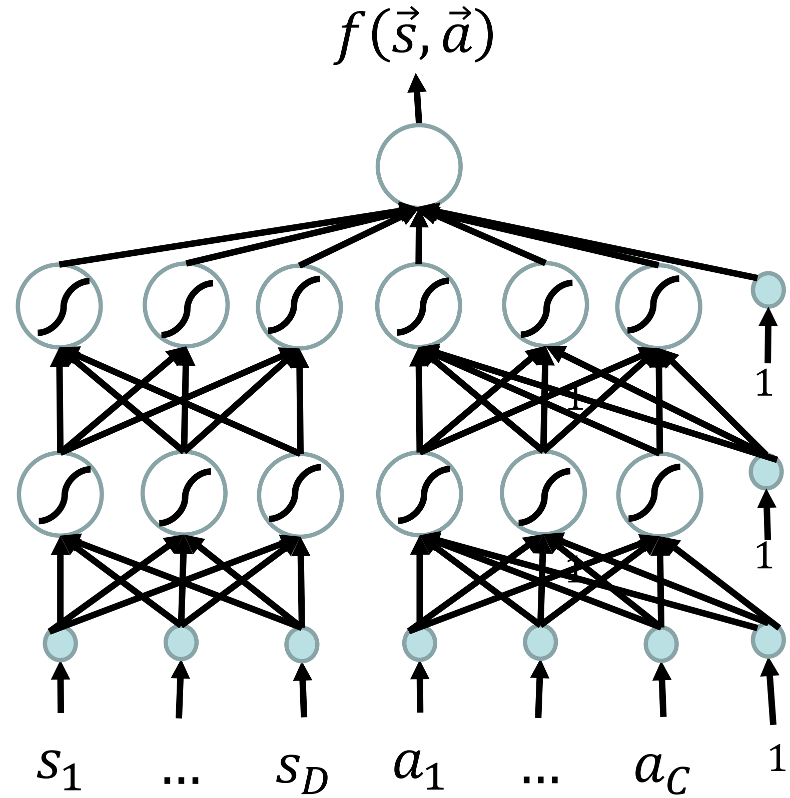
# Deep Q learning

Train the neural network weights in order to minimize the mean-squared error:

$$\mathcal{L} = \frac{1}{2} E[(f(\vec{s}, \vec{a}) - Q_{local}\,(\vec{s}, \vec{a})\,)^2]$$

$Q_{local}\,(\vec{s}, \vec{a})$ is the estimated value of the current action:

$$Q_{local}(\vec{s}_t, \vec{a}_t) = R_t(\vec{s}_t) + \gamma \max_{\vec{a}'} f(\vec{s}_{t+1}, \vec{a}')$$



# Imitation Learning

If we have $|A|$ possible, actions, $1 \leq a \leq |A|$, we could train the network to learn a hidden layer $h(s)$ so that:

$$\pi_a(s) = \frac{\exp(w_a^T h(s))}{\sum_{k=1}^{|A|} \exp(w_k^T h(s))} = P(A = a|S = s)$$

Meaning “the probability that the best action is a.”

# Two approaches to deep reinforcement learning

- Deep Q learning: train a network to estimate Q(s,a)
  - Like value iteration: we focus on Q(s,a), which is closely related to U(s)
  - Big problem: Q(s,a) is very noisy, needs lots of smoothing
- Imitation learning: train a network to imitate a human being
  - Like policy iteration: focus directly on estimating $\pi(s)$
  - Big problem: the only way to train this is by imitating a human!

# Outline

- Two approaches to solving an MDP
- Two approaches to deep reinforcement learning
- Combining the two, in order to solve the problems with either

# The Actor-Critic Algorithm

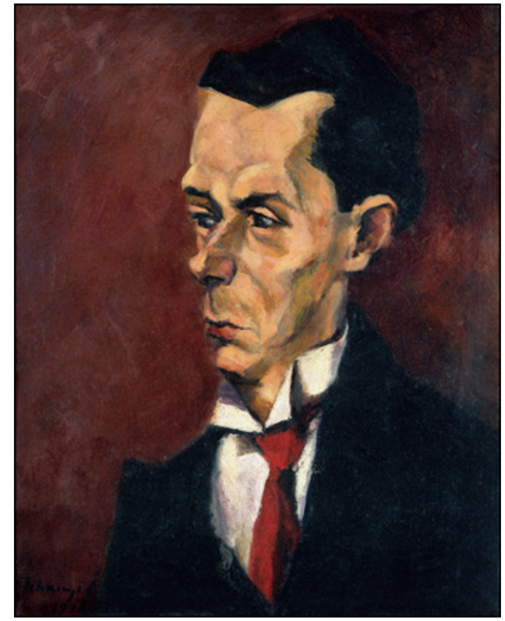
- Deep Q-learning gives us a network Q(s,a) which is very noisy, so we don't really want to trust it
- A policy network can directly estimate $\pi(s)$. The only problem is that we have no way to train it, unless we imitate human behavior.

Actors from the Comédie Française, by Antoine Watteau, 1720. Public Domain, https://commons.wikimedia.org/w/index.php?curid=15418670

# Actor-critic algorithm

So let's train two neural nets!

- $Q_t(s,a)$ is the **critic**, and is trained according to the deep Q-learning algorithm (MMSE).
- $\pi_a(s)$ is the **actor**, and is trained to satisfy the critic



The Critic, by Lajos Tihanyi. Oil on canvas, 1916. Public Domain, https://commons.wikimedia.org/w/index.php?curid=17837438

# The Actor-Critic Algorithm

Main idea:

- The **actor** is a policy network that decides what action to perform:

  $\pi_a(s) =$ Probability that $a$ is the best action in state $s$

- The **critic** is a deep Q-learning network that estimates the quality of that action ($Q(s,a)$).

  $Q(s,a) =$ Expected sum of future rewards if $(s,a)$

- The critic is noisy, so they don't get to decide the action. Instead, we only use the critic to help us to train the actor.

# The Actor-Critic Algorithm

$$\pi_a(s) = \text{Probability that } a \text{ is the best action in state } s$$

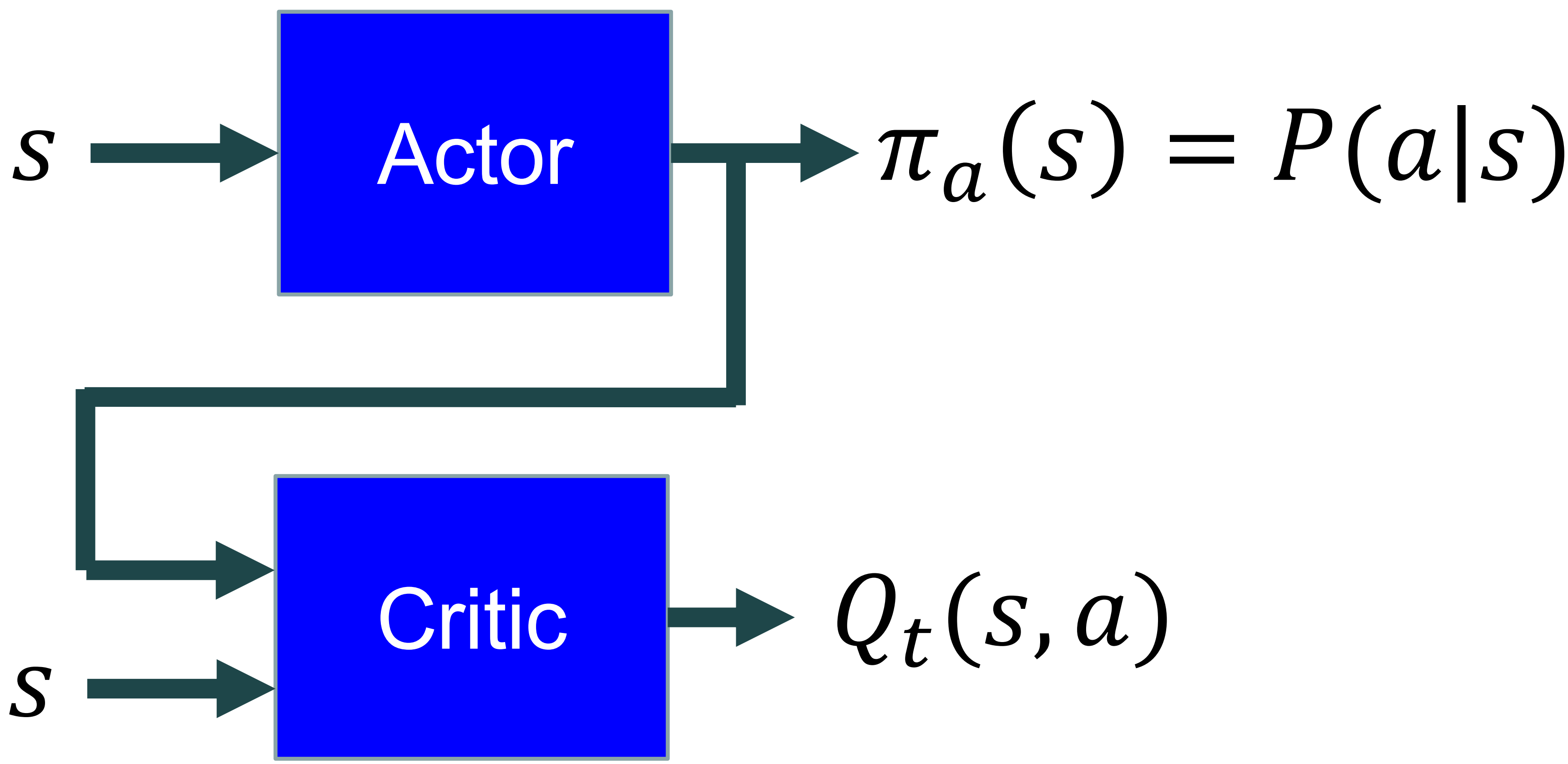
$$Q(s,a) = \text{Expected sum of future rewards if } (s,a)$$

- The critic is noisy, so they don't get to decide the action. Instead, we only use the critic to help us to train the actor.

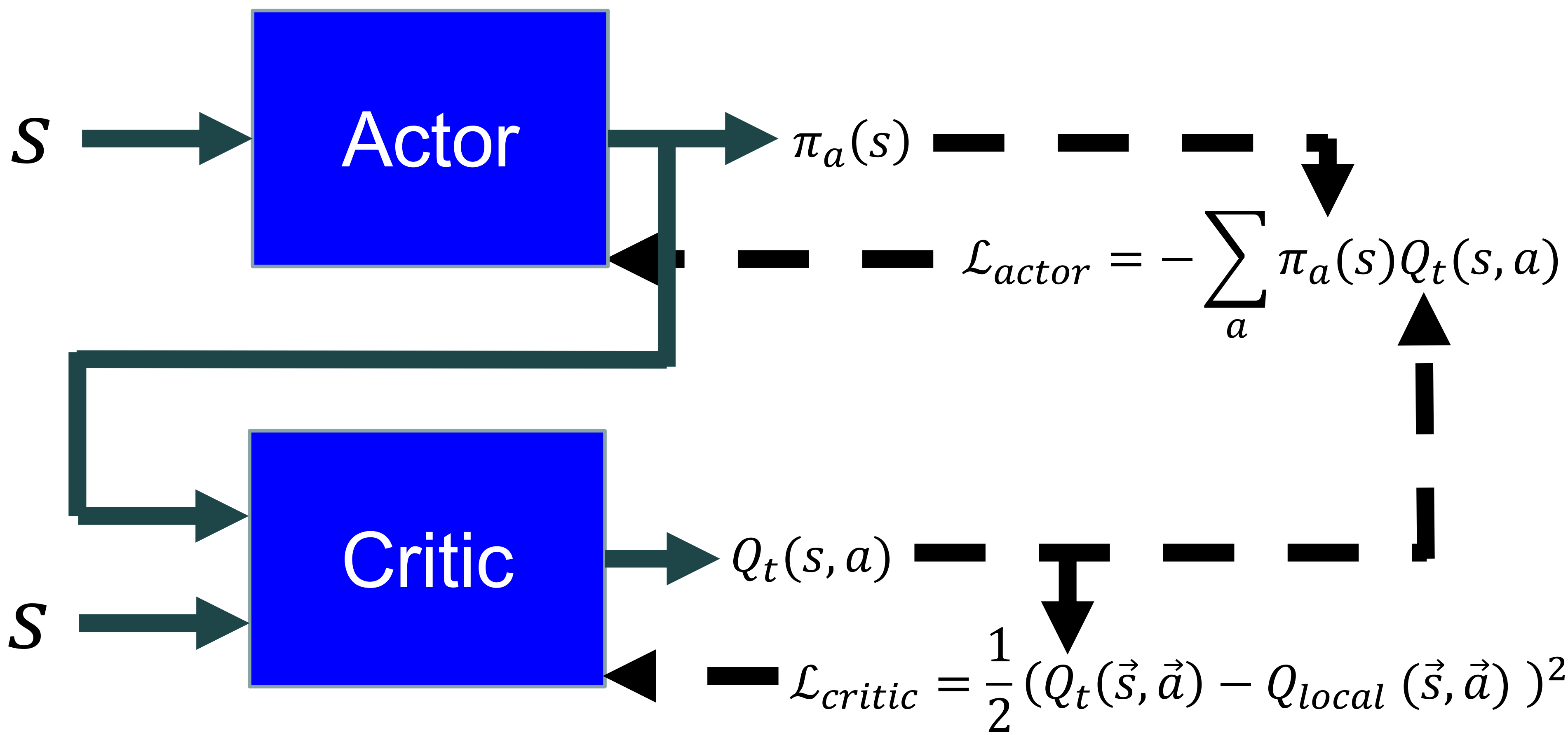
$$\mathcal{L} = -\sum_a \pi_a(s) Q(s,a)$$

- The training loss = negative expected sum of future rewards given action $a$, averaged over the probability with which the actor chooses action $a$.

# The Actor-Critic Algorithm: Forward-Prop



# The Actor-Critic Algorithm: Back-Prop



# Asynchronous advantage actor-critic (A3C)

TORCS car racing simulation video

Mnih et al. Asynchronous Methods for Deep Reinforcement Learning. ICML 2016

# Overview: All of the Model-Free Reinforcement Learning Algorithms You've Learned

- Policy learning: learn $\pi(s)$ directly
  - Imitation learning
- Q-learning: learn $Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a)U(s')$
  - Table-based: TD, SARSA
  - Deep Q-learning
- Actor-Critic: learn both