

Lecture 7: Actor-Critic Algorithms

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering
Nanjing University

Nov. 5th, 2020

Table of Contents

- 1 Improving the policy gradient with a critic
- 2 Policy evaluation – fit the value function
- 3 The actor-critic algorithm
- 4 Actor-critics with n -step returns and eligibility traces

Today's lecture

- Improving the policy gradient with a critic
- The policy evaluation problem
- Discount factors
- The actor-critic algorithm
- Goals
 - Understand how policy evaluation fits into policy gradients
 - Understand how actor-critic algorithms work

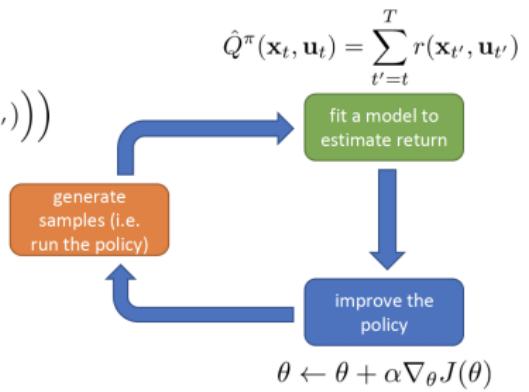
Recall: policy gradients

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \right) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}^\pi$$

“reward to go”



Improving the policy gradient

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\underbrace{\sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})}_{\text{"reward to go"}} \right)$$

“reward to go”

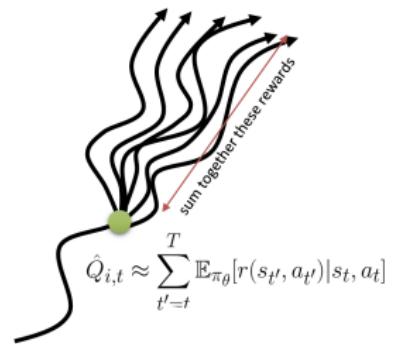
$$\hat{Q}_{i,t}$$

$\hat{Q}_{i,t}$: estimate of expected reward if we take action $\mathbf{a}_{i,t}$ in state $\mathbf{s}_{i,t}$

can we get a better estimate?

$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_{\theta}} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$: true *expected* reward-to-go

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$



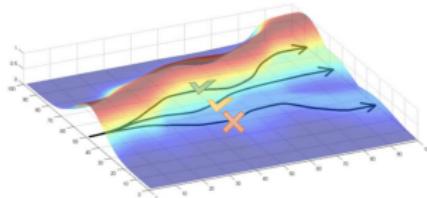
Recall: Reducing variance - Baselines

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(\tau) [r(\tau) - b]$$

a convenient identity

$$\pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) = \nabla_{\theta} \pi_{\theta}(\tau)$$

$$b = \frac{1}{N} \sum_{i=1}^N r(\tau)$$



- But... are we allowed to do that?

$$\begin{aligned}\mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(\tau) b] &= \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) b d\tau = \int \nabla_{\theta} \pi_{\theta}(\tau) b d\tau \\ &= b \nabla_{\theta} \int \pi_{\theta}(\tau) d\tau = b \nabla_{\theta} 1 = 0\end{aligned}$$

- Subtracting a baseline is unbiased in expectation!
- Average reward is not the best baseline, but it's pretty good!

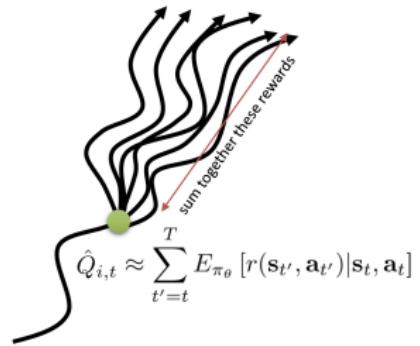
What about the baseline?

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]: \text{true expected reward-to-go}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) (Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) - V(\mathbf{s}_{i,t}))$$

$$b_t = \frac{1}{N} \sum_i Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \quad \text{average what?}$$

$$V(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} [Q(\mathbf{s}_t, \mathbf{a}_t)]$$



Recall: Relationship between Q and V

- State value function:

$$v_{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right]$$

- Action value function:

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right]$$

- **What is the relationship between $v_{\pi}(s)$ and $Q_{\pi}(s, a)$?**

Recall: Relationship between Q and V

$$v_{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right]$$

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right]$$

$$\begin{aligned} v_{\pi}(s) &= \mathbb{E}_{\pi} \left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right] \\ &= \sum_a \pi(a|s) \mathbb{E}_{\pi} \left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \\ &= \sum_a \pi(a|s) Q_{\pi}(s, a) = \mathbb{E}_{a \sim \pi}[Q_{\pi}(s, a)] \end{aligned}$$

State value function & Action value function

- Action value function $Q(s, a)$: total reward from taking a in s

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')] \end{aligned}$$

- State value function $v(s)$: total reward from s

$$v_\pi(s) = \mathbb{E}_{a \sim \pi(a|s)}[Q_\pi(s, a)]$$

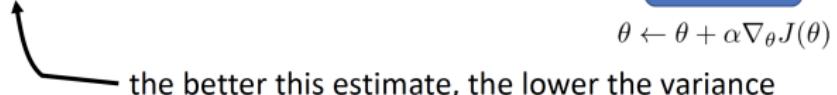
The “advantage” function

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$: total reward from taking \mathbf{a}_t in \mathbf{s}_t

$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$: total reward from \mathbf{s}_t

$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$: how much better \mathbf{a}_t is

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$



the better this estimate, the lower the variance

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\underbrace{\sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) - b}_{\text{unbiased, but high variance single-sample estimate}} \right)$$

unbiased, but high variance single-sample estimate

Value function fitting

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$$

$$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

fit what to what?

Q^π, V^π, A^π ?

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \underbrace{E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}[V^\pi(\mathbf{s}_{t+1})]}_{V^\pi(s_{t+1})}$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$$

let's just fit $V^\pi(\mathbf{s})$!

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) = r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

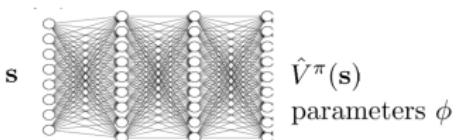
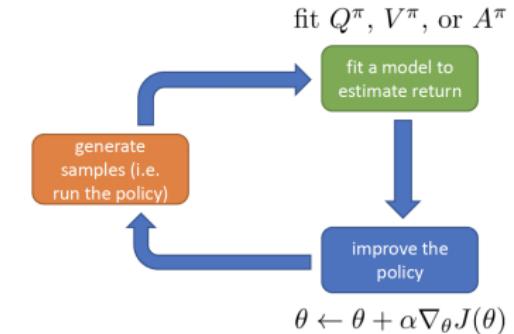


Table of Contents

- 1 Improving the policy gradient with a critic
- 2 Policy evaluation – fit the value function
- 3 The actor-critic algorithm
- 4 Actor-critics with n -step returns and eligibility traces

Recall: Policy evaluation in dynamic programming

- Compute the state-value function v_π for an arbitrary policy π

$$\begin{aligned}v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma v_\pi(s')]\end{aligned}$$

- If the environment's dynamics are completely known
 - In principle, the solution is a straightforward computation

Recall: Policy evaluation in Monte Carlo

- Considering Monte Carlo methods for learning the state-value function for a given policy
 - $v_\pi(s)$: the expected return—expected cumulative future discounted reward—starting from s
 - Estimate $v_\pi(s)$ from **experience**: simply average the returns observed after visits to s
 - As more returns are observed, the average should converge to the expected value

$$\begin{aligned}v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\&= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]\end{aligned}$$

Monte-Carlo evaluation with function approximation

$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^T E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t]$$

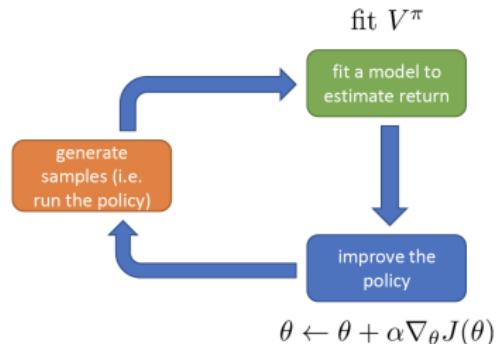
$$J(\theta) = E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}[V^\pi(\mathbf{s}_1)]$$

how can we perform policy evaluation?

Monte Carlo policy evaluation (this is what policy gradient does)

$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

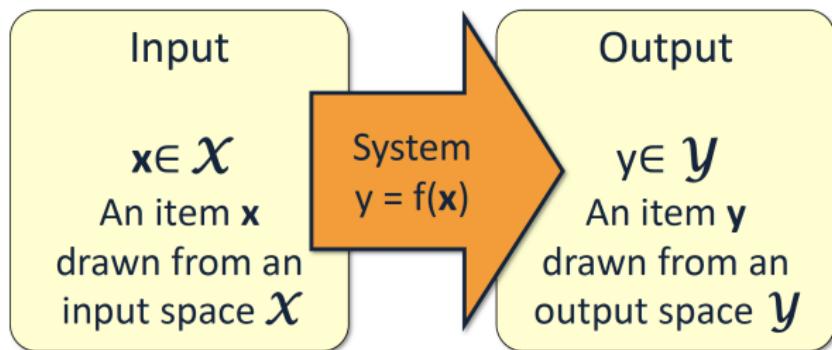
$$V^\pi(\mathbf{s}_t) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \quad (\text{requires us to reset the simulator})$$



$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

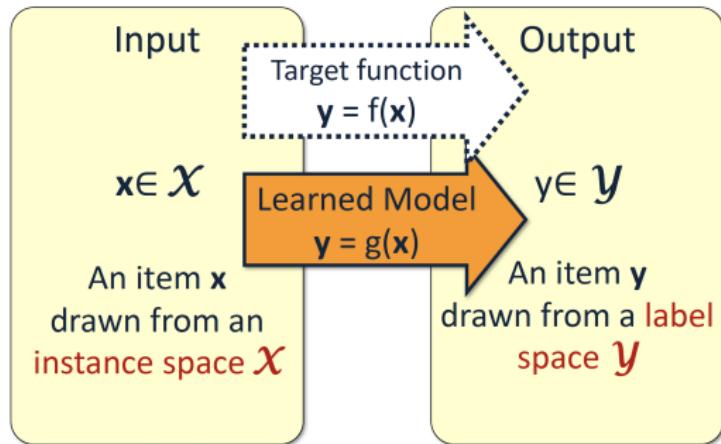


Aside: Regression in supervised learning



- We consider systems that apply a function $f(\cdot)$ to input items x and return an output $y = f(x)$
- In supervised learning, we deal with systems whose $f(\cdot)$ is learned from samples (x, y)

Aside: Regression in supervised learning



- We need to choose what kind of model we want to learn
 - Linear model, nonlinear model...
 - Parametric model, nonparametric model...
 - Decision trees, neural networks, Gaussian processes...

Monte-Carlo evaluation using supervised regression

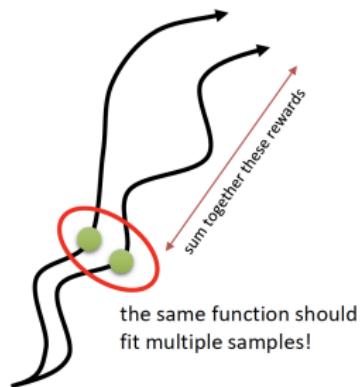
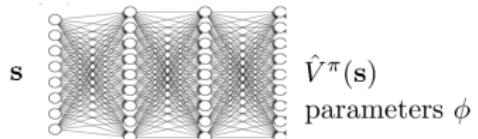
$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

not as good as this: $V^\pi(\mathbf{s}_t) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$

but still pretty good!

training data: $\left\{ \left(\mathbf{s}_{i,t}, \underbrace{\sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})}_{y_{i,t}} \right) \right\}$

supervised regression: $\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$



Recall: Policy evaluation in temporal-difference learning

- MC and TD in common
 - Use experience to solve the prediction problem, update their estimate V of v_π for the non-terminal state S_t occurring in that experience
- MC: **must wait** until the return following the visit is known (end of an episode)

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \end{aligned}$$

- TD: need to wait only until the next time step, bootstrapping

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \end{aligned}$$

Can we do better? – From MC to TD evaluation

$$\begin{aligned}v_{\pi}(s) &= \mathbb{E}_{\pi}[G_t | S_t = s] \\&= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\&= \mathbb{E}_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s] \\&= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_{\pi}(s')]\end{aligned}$$

- **MC**: The expected G_t is not known, a sample return is used in place of the real expected return
- **DP**: The true v_{π} is not known, and the current estimate $V(S_{t+1})$ is used instead
- **TD**: It samples the expected values R_{t+1} , and it uses the current estimate $V(S_{t+1})$ instead of the true v_{π}
 - Combine the sampling of MC with the bootstrapping of DP

TD policy evaluation with function approximation

ideal target: $y_{i,t} = \sum_{t'=t}^T E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{i,t}] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + V^\pi(\mathbf{s}_{i,t+1}) \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \underbrace{\hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})}$

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$ directly use previous fitted value function!

training data: $\left\{ \underbrace{\left(\mathbf{s}_{i,t}, r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1}) \right)}_{y_{i,t}} \right\}$

supervised regression: $\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$

sometimes referred to as a “bootstrapped” estimate

Policy evaluation examples

TD-Gammon, Gerald Tesauro 1992

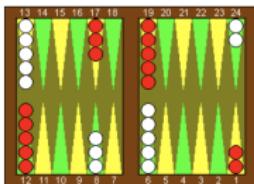


Figure 2. An illustration of the normal opening position in backgammon. TD-Gammon has sparked a near-universal conversion in the way experts play certain opening rolls. For example, with an opening roll of 4-1, most players have now switched from the traditional move of 13-9, 6-5, to TD-Gammon's preference, 13-9, 24-23. TD-Gammon's analysis is given in Table 2.

AlphaGo, Silver et al. 2016

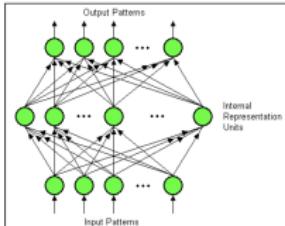


Figure 1. An illustration of the multilayer perceptron architecture used in TD-Gammon's neural network. This architecture is also used in the popular backpropagation learning procedure. Figure reproduced from [9].

reward: game outcome

value function $\hat{V}_\phi^\pi(\mathbf{s}_t)$:

expected outcome given board state

reward: game outcome

value function $\hat{V}_\phi^\pi(\mathbf{s}_t)$:

expected outcome given board state

Table of Contents

- 1 Improving the policy gradient with a critic
- 2 Policy evaluation – fit the value function
- 3 The actor-critic algorithm
- 4 Actor-critics with n -step returns and eligibility traces

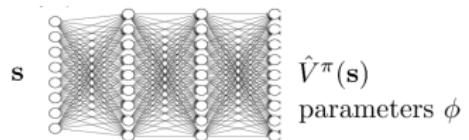
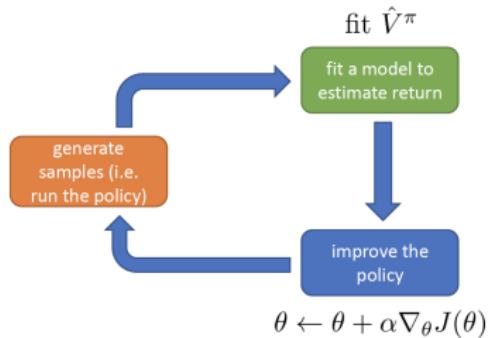
An actor-critic algorithm

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$$



$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t]$$

Recall: Discount rate $\gamma \in [0, 1]$

- Assume that: $0 \leq r_{min} \leq r \leq r_{max} \leq \infty$
- Without discount factor: unbounded

$$\begin{aligned} V(s_t) &= \mathbb{E}[r_t + r_{t+1} + r_{t+2} + \dots] \\ &\geq r_{min} + r_{min} + r_{min} + \dots \\ &= \infty \end{aligned}$$

- With discount factor: bounded

$$\begin{aligned} V(s_t) &= \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots] \\ &\leq r_{max} + \gamma r_{max} + \gamma^2 r_{max} + \dots \\ &= \frac{r_{max}}{1 - \gamma} \end{aligned}$$

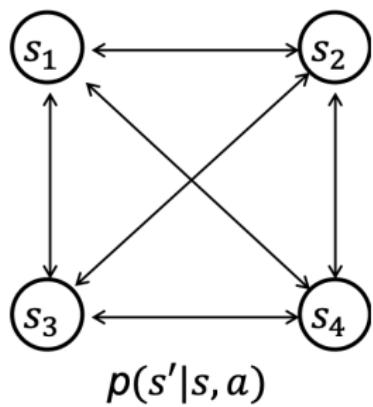
Recall: Discount rate $\gamma \in [0, 1]$

- The expected **discounted** return
 - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots = \sum_{k=1}^{\infty} \gamma^k R_{t+k+1}$
- The discount rate determines the present value of future rewards: a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately
- $\gamma \rightarrow 0$, the agent is “myopic”, only maximizing immediate rewards
 - Akin to supervised learning that maximizes the log-likelihood of each sample, $\log p(y_i|x_i)$
- $\gamma \rightarrow 1$, the agent is “farsighted”, taking future rewards into account
- Returns at successive time steps are related to each other

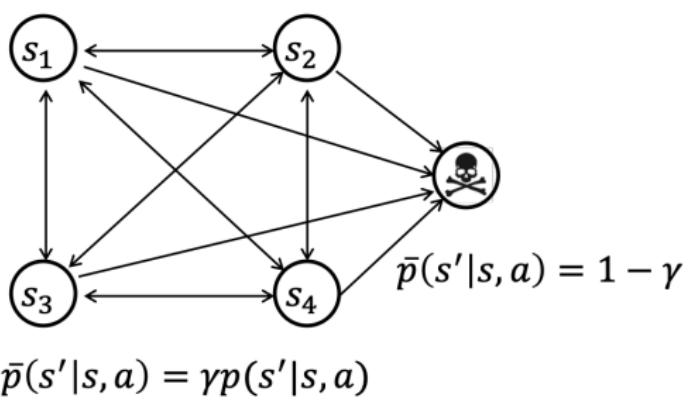
$$\begin{aligned} G_t &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Recall: γ changes the MDP

Without discount:



With discount:



Aside: discount factors in actor-critics

$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$$

what if T (episode length) is ∞ ?

\hat{V}_ϕ^π can get infinitely large in many cases



episodic tasks



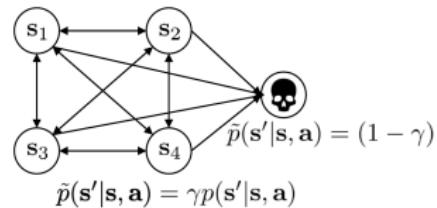
continuous/cyclical tasks

simple trick: better to get rewards sooner than later

$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$$

↑
discount factor $\gamma \in [0, 1]$ (0.99 works well)

γ changes the MDP:



Discount factors for policy gradients

$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$$

with critic:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \underbrace{\left(r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1}) - \hat{V}_\phi^\pi(\mathbf{s}_{i,t}) \right)}_{\hat{A}^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})}$$

what about (Monte Carlo) policy gradients?

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$$

Actor-critic algorithms (with discount)

batch actor-critic algorithm:

- 
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
 2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
 3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
 2. update \hat{V}_ϕ^π using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$
 3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$
 4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Batch-mode (offline) vs. online

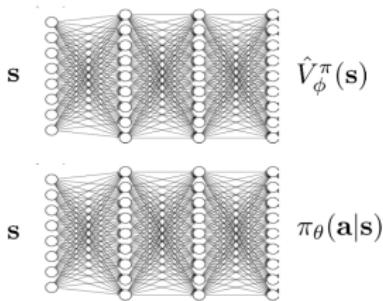
- Batch-model (offline) algorithms
 - Collect a batch of samples using some policy
 - Fit the state- or action-value function iteratively
- Online algorithms
 - Take some action to collect one sample
 - Fit the value function
 - Iteratively alternate the above two steps

Architecture design

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update \hat{V}_ϕ^π using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

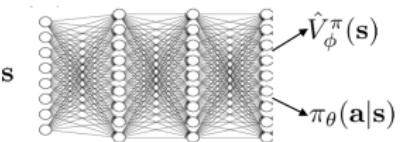
two network design



+ simple & stable

- no shared features between actor & critic

shared network design

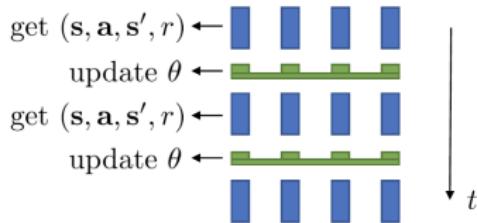


Online actor-critic in practice

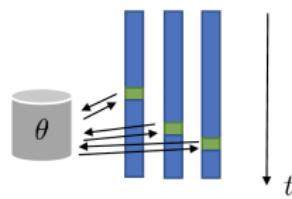
online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update \hat{V}_ϕ^π using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}') \leftarrow$ works best with a batch (e.g., parallel workers)
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a}) \leftarrow$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

synchronized parallel actor-critic



asynchronous parallel actor-critic



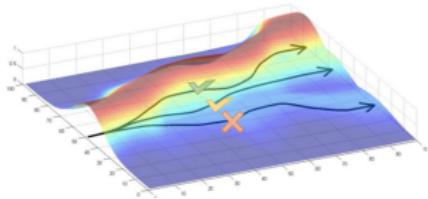
Recall: Reducing variance - Baselines

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(\tau) [r(\tau) - b]$$

a convenient identity

$$\pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) = \nabla_{\theta} \pi_{\theta}(\tau)$$

$$b = \frac{1}{N} \sum_{i=1}^N r(\tau)$$



- But... are we allowed to do that?

$$\begin{aligned}\mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(\tau) b] &= \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) b d\tau = \int \nabla_{\theta} \pi_{\theta}(\tau) b d\tau \\ &= b \nabla_{\theta} \int \pi_{\theta}(\tau) d\tau = b \nabla_{\theta} 1 = 0\end{aligned}$$

- Subtracting a baseline is unbiased in expectation!
- Average reward is not the best baseline, but it's pretty good!

Recall: Analyzing the variance

can we write down the variance?

$$\text{Var}[x] = E[x^2] - E[x]^2$$

$$\nabla_{\theta} J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) (r(\tau) - b)]$$

$$\text{Var} = E_{\tau \sim \pi_{\theta}(\tau)} [(\nabla_{\theta} \log \pi_{\theta}(\tau) (r(\tau) - b))^2] - E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) (r(\tau) - b)]^2$$


this bit is just $E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)]$
(baselines are unbiased in expectation)

$$\begin{aligned} \frac{d\text{Var}}{db} &= \frac{d}{db} E[g(\tau)^2 (r(\tau) - b)^2] = \frac{d}{db} (E[g(\tau)^2 r(\tau)^2] - 2E[g(\tau)^2 r(\tau)b] + b^2 E[g(\tau)^2]) \\ &= -2E[g(\tau)^2 r(\tau)] + 2bE[g(\tau)^2] = 0 \end{aligned}$$

$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]} \quad \leftarrow \quad \text{This is just expected reward, but weighted by gradient magnitudes!}$$

Critics as state-dependent baselines

Actor-critic: $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+1}) - \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t}) \right)$

+ lower variance (due to critic)
- not unbiased (if the critic is not perfect)

Policy gradient: $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) - b \right)$

+ no bias
- higher variance (because single-sample estimate)

can we use \hat{V}_{ϕ}^{π} and still keep the estimator unbiased?

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) - \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t}) \right)$$

+ no bias
+ lower variance (baseline is closer to rewards)

Table of Contents

- 1 Improving the policy gradient with a critic
- 2 Policy evaluation – fit the value function
- 3 The actor-critic algorithm
- 4 Actor-critics with n -step returns and eligibility traces

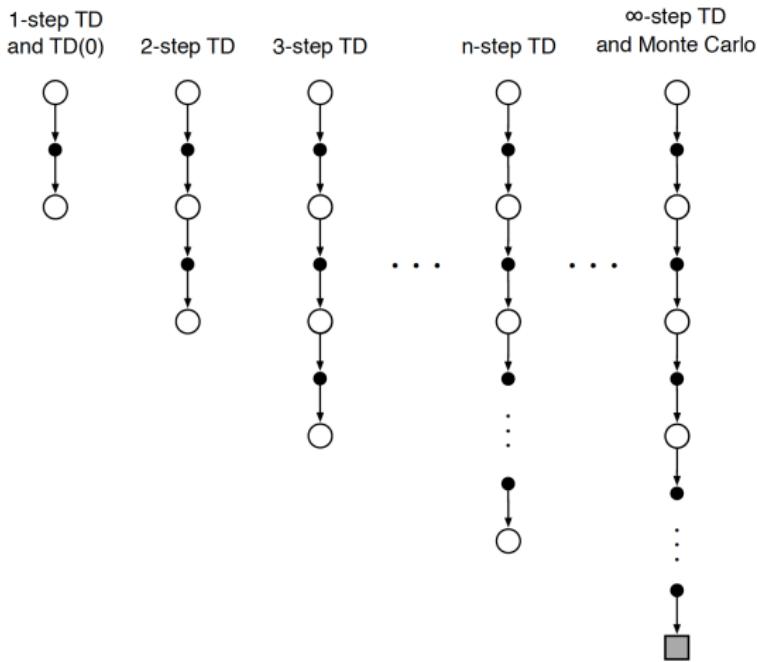
n -step bootstrapping: Combine MC and one-step TD

- Neither MC or one-step TD is always the best, we generalize both methods so that one can shift from one to the other smoothly as needed to meet the demands of a particular task
- One-step TD: In many applications, one wants to be able to update the action very fast to take into account anything that has changed
- However, bootstrapping works best if it is over a length of time in which a significant and recognizable state change has occurred

$n = 1$	n -step TD	$n = \infty$
$\text{TD}(0)$	\leftrightarrow	MC

n -step TD prediction

- Perform an update based on an intermediate number of rewards, more than one, but less than all of them until termination



Recall: MC and TD(0) updates

- In MC updates, the target is the **complete return**

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t+1} R_T$$

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha[\textcolor{red}{G_t} - V(S_t)] \\ &= V(S_t) + \alpha[\textcolor{red}{R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t+1} R_T} - V(S_t)] \end{aligned}$$

- In TD(0) updates, the target is the **one-step return**

$$G_{t:t+1} = R_{t+1} + \gamma V(S_{t+1})$$

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha[\textcolor{red}{G_{t:t+1}} - V(S_t)] \\ &= V(S_t) + \alpha[\textcolor{red}{R_{t+1} + \gamma V(S_{t+1})} - V(S_t)] \end{aligned}$$

n -step TD update rule

- For n -step TD, set the target as the **n -step return**

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- All n -step returns can be considered approximations to the complete return, truncated after n steps and then corrected for the remaining missing terms by $V(S_{t+n})$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_{t:t+n} - V(S_t)]$$

$$= V(S_t) + \alpha[R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}) - V(S_t)]$$

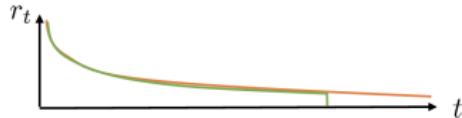
Actor-critics with n -step returns

$$\hat{A}_C^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}_\phi^\pi(s_{t+1}) - \hat{V}_\phi^\pi(s_t)$$

$$\hat{A}_{MC}^\pi(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}_\phi^\pi(s_t)$$

- + lower variance
- higher bias if value is wrong (it always is)
- + no bias
- higher variance (because single-sample estimate)

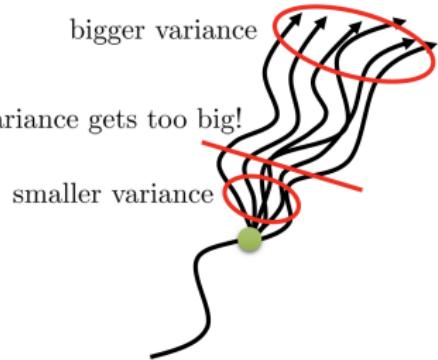
Can we combine these two, to control bias/variance tradeoff?



cut here before variance gets too big!

$$\hat{A}_n^\pi(s_t, a_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}_\phi^\pi(s_t) + \gamma^n \hat{V}_\phi^\pi(s_{t+n})$$

choosing $n > 1$ often works better!



Eligibility traces: unify/generalize TD and MC

- Almost any TD method can be combined with eligibility traces to obtain a more general method that may learn more efficiently
 - e.g., the popular $\text{TD}(\lambda)$ algorithm, λ refers the use of an eligibility trace
 - Produce a family of methods spanning a spectrum that has MC methods at one end ($\lambda = 1$) and one-step TD methods at the other ($\lambda = 0$)
- Eligibility traces offer an elegant algorithmic mechanism with significant computational advantages (compared to n -step TD)
 - Only a single trace vector is required rather than a store of the last n feature vectors
 - Learning also occurs continually and uniformly in time rather than being delayed and then catching up at the end of the episode
 - Learning can occur and effect behavior immediately after a state is encountered rather than being delayed n -steps

The λ -return

- How to interrelate TD and MC?
 - e.g., average one-step and infinite-step returns, $G = (G_t + G_{t:t+1})/2$
 - An update that averages simpler component updates is called a **compound update**
- The TD(λ) algorithm can be understood as one particular way of averaging n -step updates

$$\begin{aligned} G_t^\lambda &= (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \\ &= (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t \end{aligned}$$

Backup diagram for TD(λ)

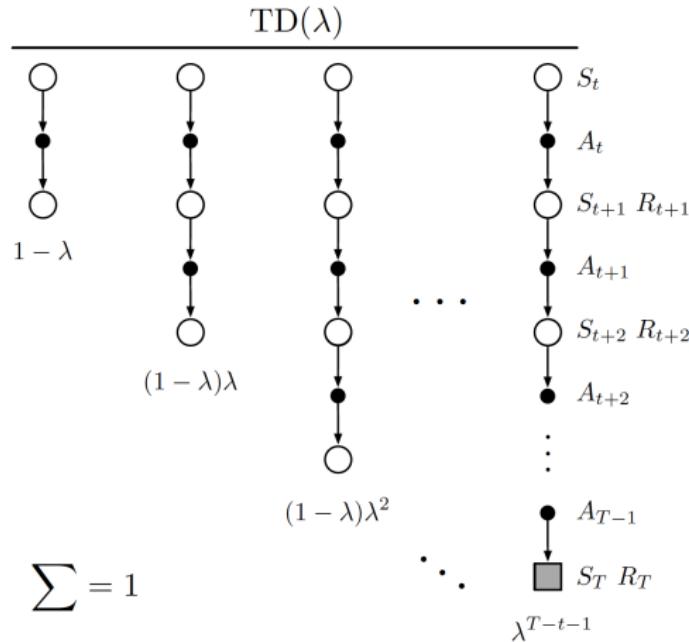


Figure 12.1: The backup diagram for TD(λ). If $\lambda = 0$, then the overall update reduces to its first component, the one-step TD update, whereas if $\lambda = 1$, then the overall update reduces to its last component, the Monte Carlo update.

The weight distribution

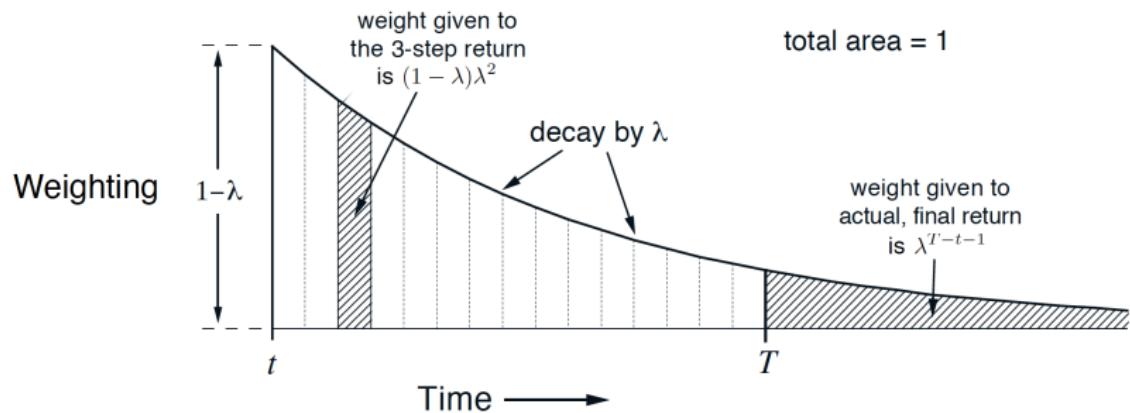
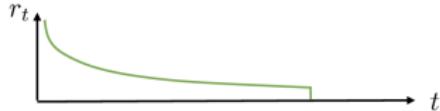


Figure 12.2: Weighting given in the λ -return to each of the n -step returns.

Generalized advantage estimation: Actor-critics with eligibility traces



Do we have to choose just one n?

Cut everywhere all at once!

$$\hat{A}_n^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}_\phi^\pi(\mathbf{s}_t) + \gamma^n \hat{V}_\phi^\pi(\mathbf{s}_{t+n})$$

$$\hat{A}_{\text{GAE}}^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{n=1}^{\infty} w_n \hat{A}_n^\pi(\mathbf{s}_t, \mathbf{a}_t)$$

Weighted combination of n-step returns

How to weight?

Mostly prefer cutting earlier (less variance)

$$w_n \propto \lambda^{n-1}$$

exponential falloff

$$\hat{A}_{\text{GAE}}^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma((1-\lambda)\hat{V}_\phi^\pi(\mathbf{s}_{t+1}) + \lambda(r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \gamma((1-\lambda)\hat{V}_\phi^\pi(\mathbf{s}_{t+2}) + \lambda r(\mathbf{s}_{t+2}, \mathbf{a}_{t+2}) + \dots))$$

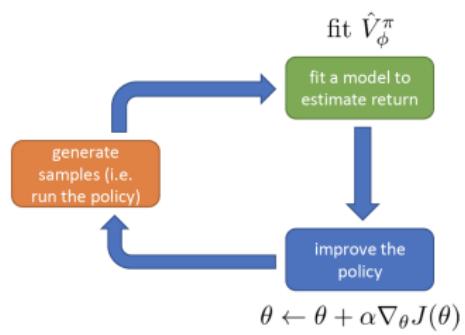
$$\hat{A}_{\text{GAE}}^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{\infty} (\gamma \lambda)^{t'-t} \delta_{t'}$$

$$\delta_{t'} = r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{t'+1}) - \hat{V}_\phi^\pi(\mathbf{s}_{t'})$$

similar effect as discount!

Review

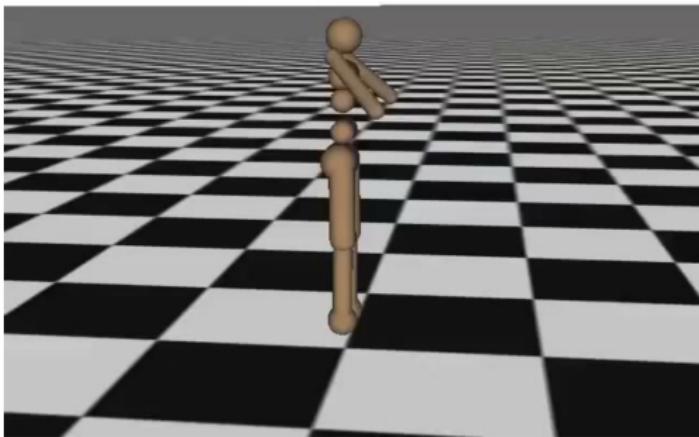
- Actor critic algorithms
 - Actor: the policy
 - Critic: value function
 - Reduce variance of policy gradient
- Policy evaluation
 - Fitting value function to policy
- Discount factors
 - Bound the value function
 - Also a variance reduction trick
- Actor critic algorithm design
 - One network (with two heads) or two networks
 - Batch mode, or online (+ parallel)
- State-dependent baselines
 - Another way to use the critic
 - Can combine: n -step returns or eligibility traces



Actor-critic examples

- High dimensional continuous control with generalized advantage estimation (Schulman, Moritz, L., Jordan, Abbeel '16)
- Batch-mode actor-critic
- Blends Monte Carlo and function approximator estimators (GAE)

Iteration 0



Actor-critic examples

- Asynchronous methods for deep reinforcement learning (Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu, 2016)
- Online actor critic, parallelized batch
- n -step returns with $n = 4$
- Single network for actor and critic

Learning objectives of this lecture

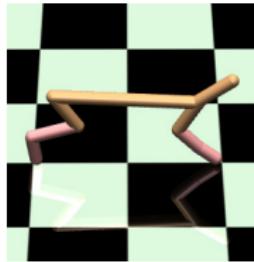
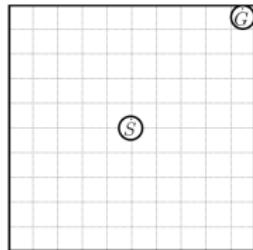
- You should be able to...
 - Extend policy gradient methods to actor-critic algorithms
 - Use policy evaluation to fit the critic, i.e., the value function
 - Be able to implement the basic actor-critic algorithm
 - Know the actor-critics with n -step returns
 - Know the actor-critics with eligibility traces, i.e., generalized advantage estimation

Actor-critic suggested readings

- Lecture 6 of CS285 at UC Berkeley, **Deep Reinforcement Learning, Decision Making, and Control**
 - <http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-6.pdf>
- Classic papers
 - Sutton, McAllester , Singh, Mansour (1999). **Policy gradient methods for reinforcement learning with function approximation:** actor critic algorithms with value function approximation
- DRL actor-critic papers
 - Mnih , Badia , Mirza, Graves, Lillicrap , Harley, Silver, Kavukcuoglu (2016). **Asynchronous methods for deep reinforcement learning:** A3C parallel online actor-critic.
 - Schulman, Moritz, L., Jordan, Abbeel (2016). **High dimensional continuous control using generalized advantage estimation:** batch mode actor-critic with blended Monte Carlo and function approximator returns
 - Gu, Lillicrap , Ghahramani , Turner, L. (2017). **Q-Prop: sample efficient policy gradient with an off-policy critic:** policy gradient with Q-function control variate
 - Tuomas Haarnoja, et al. (2018). **Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.**

Homework 4

- Study the actor-critic algorithm in detail
- Implement the actor-critic algorithm on problems 1 & 2
 - Problem 1: the point maze navigation, continuous state-action space
 - Problem 2: the MuJoCo HalfCheetah, make the robot run forward
 - Compare actor-critic with policy gradient
- Write a report introducing the algorithms and your experimentation
 - Explanations, steps, evaluation results, visualizations...
 - Submit the code and the report to yuanyang@smail.nju.edu.cn



THE END