

Lecture #19

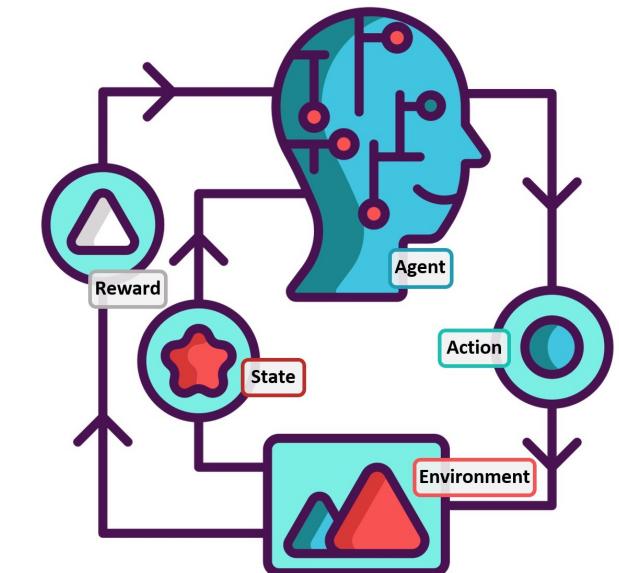
Policy Gradient

Methods: Actor Critic

Approaches & Batch

Methods

Gian Antonio Susto



Announcements before starting

- The google form is open!
- As stated previously, if you have conflicts (for example with other partial) you are allowed to be tested on the second part of the course material during the January exam

VFA: Chapter 9 & 10 Exam Material

Chapter 9

- 9.1, 9.2, 9.3, 9.4 (no proof of convergence of linear TD), 9.7
- 9.5: 9.5.3, 9.5.4
- 9.5: optional 9.5.1, 9.5.2, 9.5.5
- 9.6: optional
- Not recommend for a first study: 9.8, 9.9, 9.10, 9.11

Chapter 10

- 10.1, 10.2
- Optional: 10.3, 10.4

Btw, Chapter 11 (off-policy methods with approximation) is a nice reading, but not exam material!

Recap – The Mountain car problem

Example: Mountain Car problem

Was the usage of V truly necessary in this case?

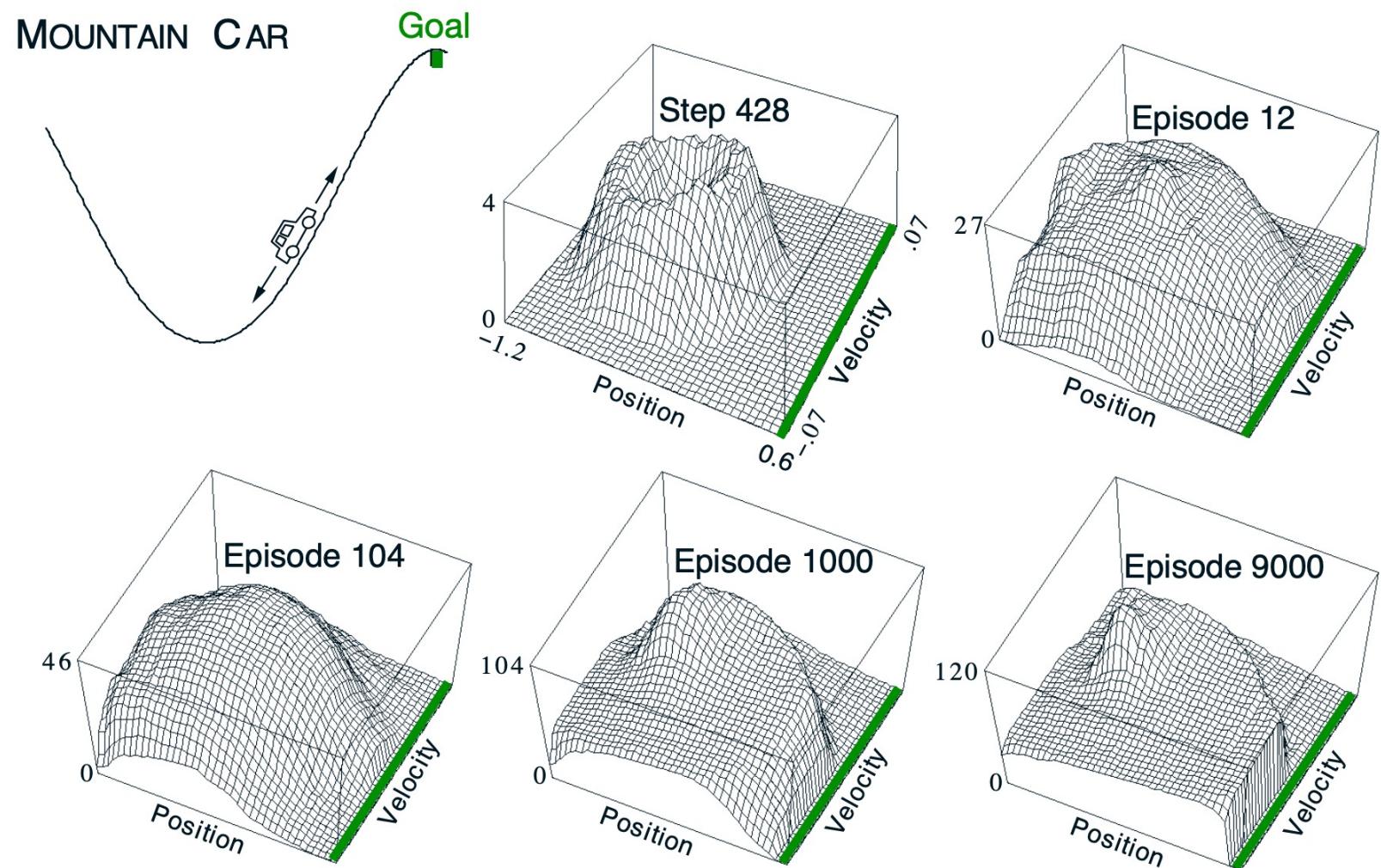
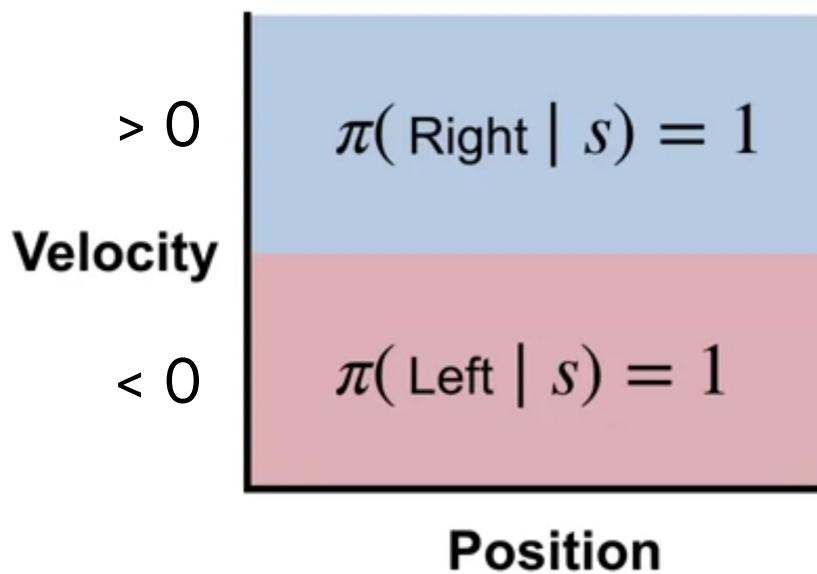
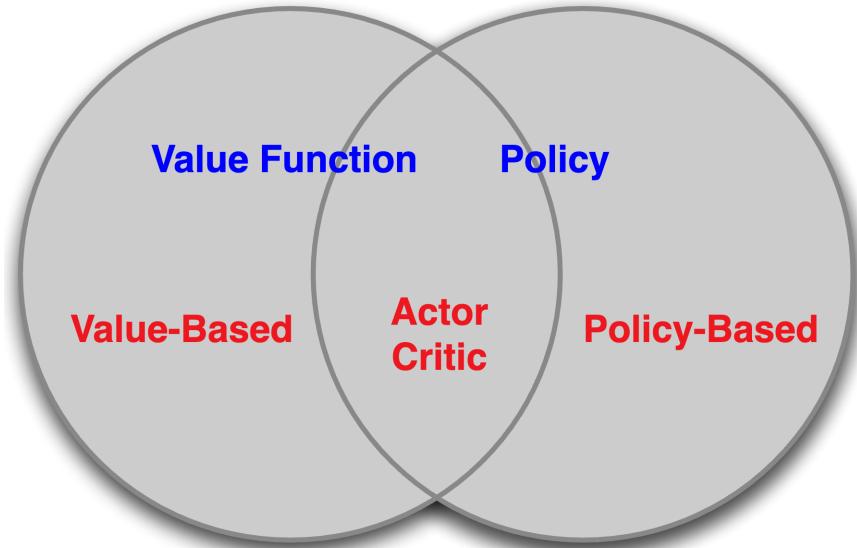


Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function ($-\max_a \hat{q}(s, a, \mathbf{w})$) learned during one run.

Recap – There are RL approaches that do not need a value function!



Up until now we have been dealing with Value-Based approaches!

That are approaches that do not need the definition of a Value function (V and/or Q) and directly act on the policy!

Crude taxonomy of model-free RL approaches:

Value Based

- Learnt Value Function
- Implicit policy(e.g. epsilon-greedy)

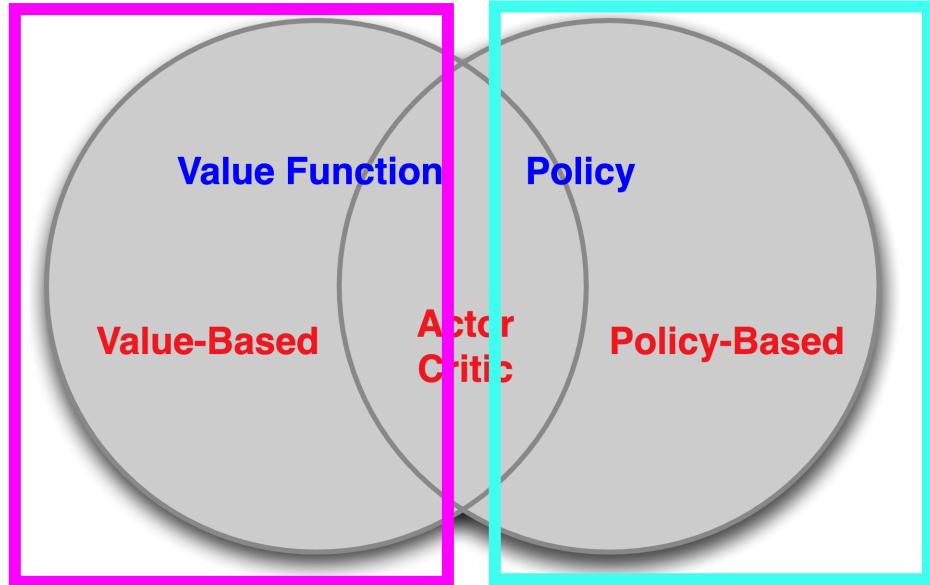
Policy Based

- No Value Function
- Learnt Policy

Actor-Critic

- Learnt Value Function
- Learnt Policy

Recap – There are RL approaches that do not need a value function!



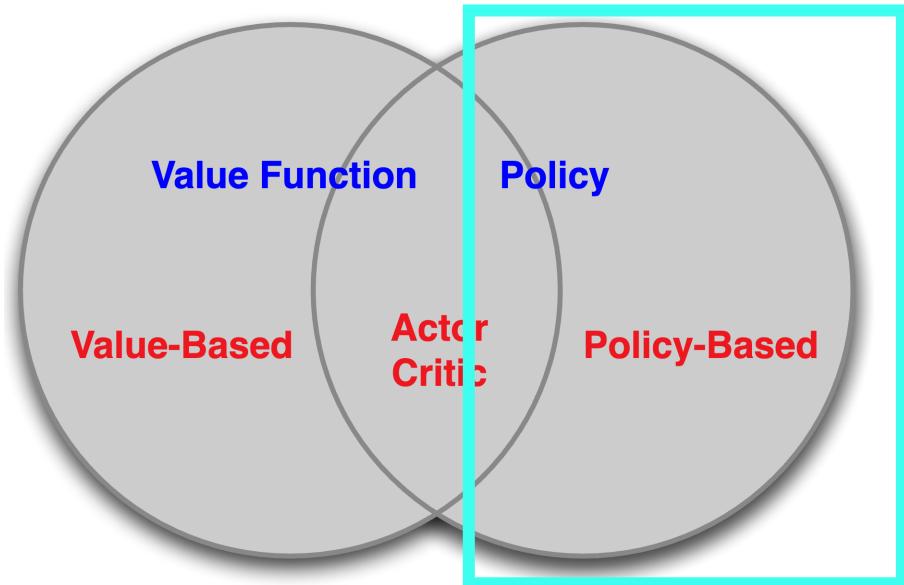
We will directly parametrize the policy and use data to find ‘better parameters’ (remember that π still need to be a distribution!)

$$V_{\theta}(s) \approx V^{\pi}(s)$$
$$Q_{\theta}(s, a) \approx Q^{\pi}(s, a)$$



$$\pi_{\theta}(s, a) = \mathbb{P}[a | s, \theta]$$

Recap – Policy-based approaches



Pros of policy-based approaches:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies (remember, in value-based there is always a 'max' operation involved)

Cons:

- Typically converge to a local rather than global optimum
- Evaluating how good a policy is, it is typically inefficient and high variance

Recap – A typical Policy parametrization

- We consider that the policy is differentiable with respect to its parameters θ
- The derivative $\nabla_{\theta}\pi(a|s, \theta)$ exists and it is always finite
- An example of policy function, the exponential softmax distribution ->

where $h()$ are parametrized numerical preferences
(similarly to what we have seen for example in Gradient Bandits)

$$\pi_{\theta}(s, a) = \mathbb{P}[a | s, \theta]$$

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

Recap – Policy Gradient

- Let $J(\theta)$ be any policy objective function
- Policy gradient algorithms search for a local maximum in $J(\theta)$ by ascending the gradient of the policy w.r.t. parameters θ

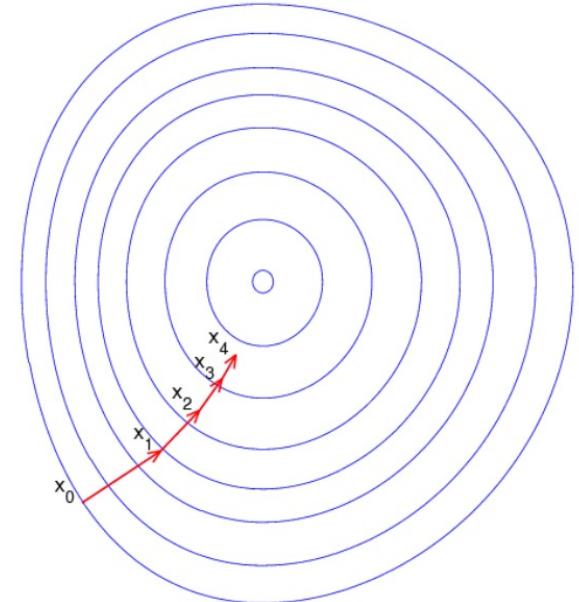
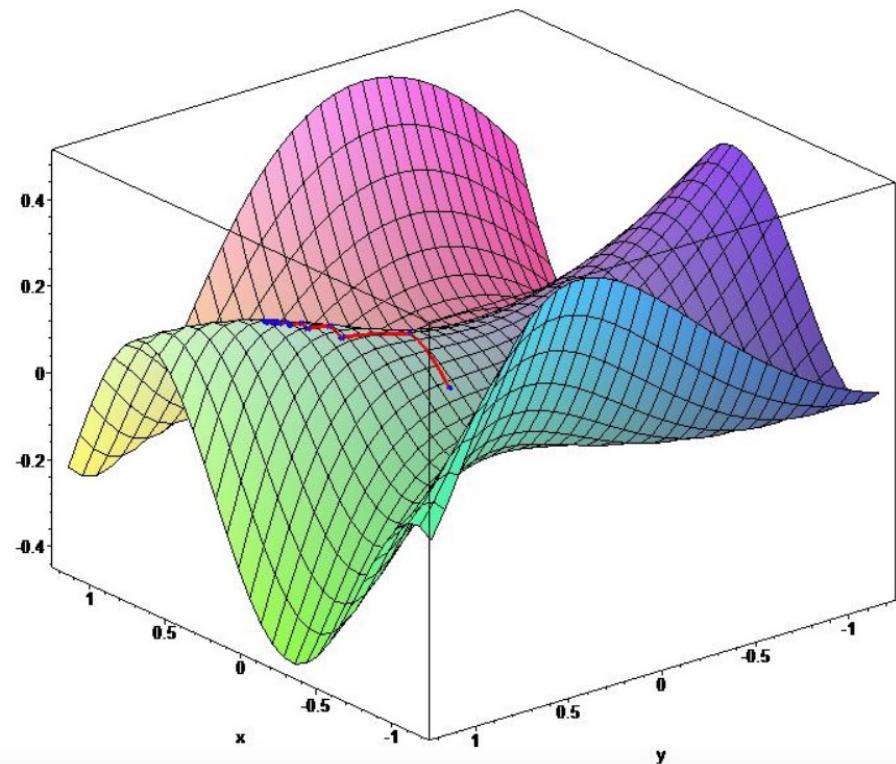
$$\theta_{t+1} = \theta_t + \alpha \Delta \theta$$

Step-size

$$\Delta \theta = \boxed{\alpha \nabla_{\theta} J(\theta)}$$

$$\nabla_{\theta} J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

Policy gradient



Recap – The Policy Gradient Theorem (sec. 13.2)

It can be shown – in the episodic case – that (see if curious the proof in the book)

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

Unfortunately, in stochastic gradient ascent this is not directly usable: we need a way to obtain samples (something that I can get from experience) such that the expectation of the sample gradient is proportional to the actual gradient of the performance measure as a function of the parameter

$J(\theta)$: performance measure

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

Recap – Towards REINFORCE: Monte Carlo Policy Gradient

We got rid of q (and mu) and we replace it with data coming from experience!

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



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



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



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



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

(because $\mathbb{E}_\pi[G_t|S_t, A_t] = q_\pi(S_t, A_t)$)

Recap – The REINFORCE update (A Monte Carlo approach)

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



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



$$\doteq \theta_t + \alpha G_t \boxed{\nabla_{\theta} \ln \pi(A_t | S_t, \theta_t)}$$

Score function

The update increases the parameter vector in a direction:

- (i) proportional to the return
- (ii) inversely proportional to the action probability

Recap – The REINFORCE algorithm (A Monte Carlo approach)

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

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Algorithm parameter: step size $\alpha > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot| \cdot, \theta)$

Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

$$\begin{aligned} G &\leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \\ \theta &\leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t | S_t, \theta) \end{aligned} \tag{G_t}$$

No V or Q is involved!

Recap – REINFORCE with baseline

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

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Input: a differentiable state-value function parameterization $\hat{v}(s, w)$

Algorithm parameters: step sizes $\alpha^{\theta} > 0, \alpha^w > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $w \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$

Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

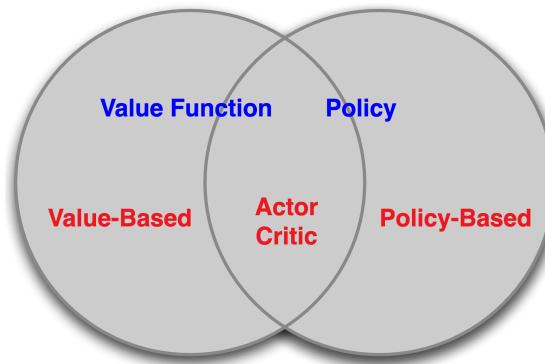
$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \quad (G_t)$$

$$\delta \leftarrow G - \hat{v}(S_t, w)$$

$$w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S_t, w)$$

$$\theta \leftarrow \theta + \alpha^{\theta} \gamma^t \delta \nabla \ln \pi(A_t | S_t, \theta)$$

Actor-Critic Approaches



Actor:

- Improve the policy
- Take care of the **control** task

Critic:

- Evaluate the policy
- Take care of the **prediction** task

Actor-Critic vs REINFORCE with baseline

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

- Because its state-value function is used only as a baseline, not as a ‘true’ critic.
- That is, it 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.
- **Bootstrapping introduces bias and an asymptotic dependence** on the quality of the function approximation
- The bias introduced through bootstrapping and reliance on the state representation is often beneficial because it reduces variance and accelerates learning
- REINFORCE with baseline is unbiased and will converge asymptotically to a local minimum, but like all Monte Carlo methods it tends to learn slowly (produce estimates of high variance) and to be inconvenient to implement online or for continuing problems.

Actor-Critic

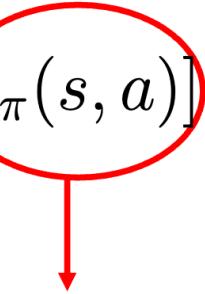
From the policy
gradient theorem:

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) q_\pi(s, a)]$$

Actor-Critic

From the policy
gradient theorem:

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) q_\pi(s, a)]$$

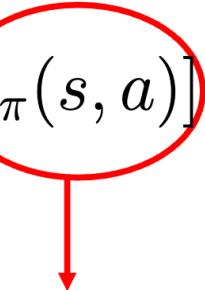


What about approximating it ? $\hat{q}_\pi(s, a, \mathbf{w}) \approx q_\pi(s, a)$

Actor-Critic

From the policy gradient theorem:

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) q_\pi(s, a)]$$



What about approximating it ? $\hat{q}_\pi(s, a, \mathbf{w}) \approx q_\pi(s, a)$

- We use a **critic** to estimate the action-value function
- Actor-critic algorithm maintain two sets of parameters:
 - **Critic** update action-value function parameters \mathbf{w}
 - **Actor** updates policy parameters θ in the direction suggested by the critic
- The **critic** is solving the prediction problem, and we can re-use what we already know (Monte Carlo, TD learning, TD(lambda))

QAC: (q) Actor-Critic

- Linear function approximation
$$\hat{q}_\pi(s, a, \mathbf{w}) = \mathbf{x}(s, a)^T \mathbf{w}$$
- Critic: TD(0)

Algorithm QAC

Initialize policy parameter θ and w

Algorithm parameter : step-size $\alpha, \beta > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\theta = \theta + \alpha \nabla_\theta \ln \pi(a|s, \theta) \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{w} = \mathbf{w} + \beta \delta \mathbf{x}(s, a)$$

$$a \leftarrow a', s \leftarrow s'$$



QAC: (q) Actor-Critic

- Linear function approximation
 $\hat{q}_\pi(s, a, \mathbf{w}) = \mathbf{x}(s, a)^T \mathbf{w}$
- Critic: TD(0)

Algorithm QAC

Initialize policy parameter θ and w

Algorithm parameter : step-size $\alpha, \beta > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\theta = \theta + \alpha \nabla_\theta \ln \pi(a|s, \theta) \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{w} = \mathbf{w} + \beta \delta \mathbf{x}(s, a)$$

$$a \leftarrow a', s \leftarrow s'$$

The book show you directly the TD(lambda) version of the algorithm



QAC: (q) Actor-Critic

- Linear function approximation
 $\hat{q}_\pi(s, a, \mathbf{w}) = \mathbf{x}(s, a)^T \mathbf{w}$
- Critic: TD(0)

Algorithm QAC

Initialize policy parameter θ and w

Algorithm parameter : step-size $\alpha, \beta > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\theta = \theta + \alpha \nabla_\theta \ln \pi(a|s, \theta) \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{w} = \mathbf{w} + \beta \delta \mathbf{x}(s, a)$$

$$a \leftarrow a', s \leftarrow s'$$

Actor - control



QAC: (q) Actor-Critic

- Linear function approximation
 $\hat{q}_\pi(s, a, \mathbf{w}) = \mathbf{x}(s, a)^T \mathbf{w}$
- Critic: TD(0)

Algorithm QAC

Initialize policy parameter θ and w

Algorithm parameter : step-size $\alpha \beta > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\theta = \theta + \alpha \nabla_\theta \ln \pi(a|s, \theta) \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{w} = \mathbf{w} + \beta \delta \mathbf{x}(s, a)$$

$$a \leftarrow a', s \leftarrow s'$$

Critic - prediction



QAC: (q) Actor-Critic

- Linear function approximation
 $\hat{q}_\pi(s, a, \mathbf{w}) = \mathbf{x}(s, a)^T \mathbf{w}$
- Critic: TD(0)

Algorithm QAC

Initialize policy parameter θ and w

Algorithm parameter : step-size $\alpha, \beta > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\theta = \theta + \alpha \nabla_\theta \ln \pi(a|s, \theta) \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{w} = \mathbf{w} + \beta \delta \mathbf{x}(s, a)$$

$$a \leftarrow a', s \leftarrow s'$$

Updates are done are each step! This version is a TD approach



QAC: (q) Actor-Critic

- Linear function approximation
 $\hat{q}_\pi(s, a, \mathbf{w}) = \mathbf{x}(s, a)^T \mathbf{w}$
- Critic: TD(0)

Algorithm QAC

Initialize policy parameter θ and w

Algorithm parameter : step-size $\alpha, \beta > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\theta = \theta + \alpha \nabla_\theta \ln \pi(a|s, \theta) \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{w} = \mathbf{w} + \beta \delta \mathbf{x}(s, a)$$

$$a \leftarrow a', s \leftarrow s'$$

Note that at each step, I am changing policy! Is a sort of value iteration...



Actor-Critic with a baseline: advantage function 1/2

- A good baseline is:

$$b(s) = v_{\pi_\theta}(s)$$

- Let us introduce the advantage function:

$$A_{\pi_\theta}(s, a) = q_{\pi_\theta}(s, a) - v_{\pi_\theta}(s)$$

in the case of optimal policy $A=0$ for optimal action in state s

- We can have a re-write the estimator of the gradient considering also the new baseline:

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) A_{\pi_\theta}(s, a)]$$

Actor-Critic with a baseline: advantage function 1/2

- A good baseline is:

$$b(s) = v_{\pi_\theta}(s)$$

- Let us introduce the advantage function:

$$A_{\pi_\theta}(s, a) = q_{\pi_\theta}(s, a) - v_{\pi_\theta}(s)$$

in the case of optimal policy $A=0$ for optimal action in state s

- We can have a re-write the estimator of the gradient considering also the new baseline:

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) A_{\pi_\theta}(s, a)]$$

What is the advantage in the case of a already according to the policy?

Actor-Critic with a baseline: advantage function 2/2

- The advantage function can significantly **reduce variance of policy gradient**
- The critic should really estimate the advantage function
- For example, by estimating both q and v
- Using two function approximators and two parameter vectors

$$\hat{v}_{\pi_\theta}(s, \mathbf{v}) \approx v_{\pi_\theta}(s)$$

$$\hat{q}_{\pi_\theta}(s, a, \mathbf{w}) \approx q_{\pi_\theta}(s, a)$$

$$A_{\pi_\theta}(s, a) = \hat{q}_{\pi_\theta}(s, a, \mathbf{w}) - \hat{v}_{\pi_\theta}(s, \mathbf{v})$$

- And updating both value functions (for example by TD learning)

Initialize policy parameter θ, w, v

Algorithm parameter : step-size $\alpha, \beta, \xi > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta_q = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\delta_v = r + \gamma \hat{v}_\pi(s', \mathbf{v}) - \hat{v}_\pi(s, \mathbf{v})$$

$$\theta = \theta + \alpha (\hat{q}_\pi(s, a, \mathbf{w}) - \hat{v}_\pi(s, \mathbf{v})) \nabla_\theta \ln \pi(a|s, \theta)$$

$$\mathbf{w} = \mathbf{w} + \beta \delta_q \nabla \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{v} = \mathbf{v} + \xi \delta_v \nabla \hat{v}_\pi(s, \mathbf{v})$$

$$a \leftarrow a', s \leftarrow s'$$

A2C

- Linear function approximation
- Critic: TD(0)

Initialize policy parameter θ , w, v

Algorithm parameter : step-size $\alpha, \beta, \xi > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta_q = r + \gamma \hat{q}_\pi(s', a', w) - \hat{q}_\pi(s, a, w)$$

$$\delta_v = r + \gamma \hat{v}_\pi(s', v) - \hat{v}_\pi(s, v)$$

$$\boxed{\theta = \theta + \alpha (\hat{q}_\pi(s, a, w) - \hat{v}_\pi(s, v)) \nabla_\theta \ln \pi(a|s, \theta)}$$

$$w = w + \beta \delta_q \nabla \hat{q}_\pi(s, a, w)$$

$$v = v + \xi \delta_v \nabla \hat{v}_\pi(s, v)$$

$$a \leftarrow a', s \leftarrow s'$$

Actor is the same...

Initialize policy parameter θ , w v

Algorithm parameter : step-size α β $\xi > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta_q = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\delta_v = r + \gamma \hat{v}_\pi(s', \mathbf{v}) - \hat{v}_\pi(s, \mathbf{v})$$

$$\theta = \theta + \alpha (\hat{q}_\pi(s, a, \mathbf{w}) - \hat{v}_\pi(s, \mathbf{v})) \nabla_\theta \ln \pi(a|s, \theta)$$

$$\mathbf{w} = \mathbf{w} + \beta \delta_a \nabla \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{v} = \mathbf{v} + \xi \delta_v \nabla \hat{v}_\pi(s, \mathbf{v})$$

$$a \leftarrow a', s \leftarrow s'$$

Critic is now composed by both a v and q components

Initialize policy parameter θ, w, v

Algorithm parameter : step-size $\alpha, \beta, \xi > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

 Sample action $a' \sim \pi(a'|s', \theta)$

$$\delta_q = r + \gamma \hat{q}_\pi(s', a', \mathbf{w}) - \hat{q}_\pi(s, a, \mathbf{w})$$

$$\delta_v = r + \gamma \hat{v}_\pi(s', \mathbf{v}) - \hat{v}_\pi(s, \mathbf{v})$$

$$\theta = \theta + \alpha (\hat{q}_\pi(s, a, \mathbf{w}) - \hat{v}_\pi(s, \mathbf{v})) \nabla_\theta \ln \pi(a|s, \theta)$$

$$\mathbf{w} = \mathbf{w} + \beta \delta_q \nabla \hat{q}_\pi(s, a, \mathbf{w})$$

$$\mathbf{v} = \mathbf{v} + \xi \delta_v \nabla \hat{v}_\pi(s, \mathbf{v})$$

$$a \leftarrow a', s \leftarrow s'$$

Let's try to simplify
the algorithm...

Estimating the advantage function

- Consider the TD error $\delta_{\pi_\theta} = r + \gamma v_{\pi_\theta}(s') - v_{\pi_\theta}(s)$
- The TD error is an unbiased estimate of the advantage function

$$\begin{aligned}\mathbb{E} [\delta_{\pi_\theta} | s, a] &= \mathbb{E} [r + \gamma v_{\pi_\theta}(s') | s, a] - v_{\pi_\theta}(s) \\ &= q_{\pi_\theta}(s, a) - v_{\pi_\theta}(s) \\ &= A_{\pi_\theta}(s, a)\end{aligned}$$

Estimating the advantage function

- Consider the TD error $\delta_{\pi_\theta} = r + \gamma v_{\pi_\theta}(s') - v_{\pi_\theta}(s)$
- The TD error is an unbiased estimate of the advantage function

$$\begin{aligned}\mathbb{E} [\delta_{\pi_\theta} | s, a] &= \mathbb{E} [r + \gamma v_{\pi_\theta}(s') | s, a] - v_{\pi_\theta}(s) \\ &= q_{\pi_\theta}(s, a) - v_{\pi_\theta}(s) \\ &= A_{\pi_\theta}(s, a)\end{aligned}$$

- Let's exploit this:

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) A_{\pi_\theta}(s, a)]$$



$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla_\theta \ln \pi(a|s, \theta) \delta_{\pi_\theta}]$$

Estimating the advantage function

- Consider the TD error $\delta_{\pi_\theta} = r + \gamma v_{\pi_\theta}(s') - v_{\pi_\theta}(s)$
- The TD error is an unbiased estimate of the advantage function

$$\begin{aligned}\mathbb{E} [\delta_{\pi_\theta} | s, a] &= \mathbb{E} [r + \gamma v_{\pi_\theta}(s') | s, a] - v_{\pi_\theta}(s) \\ &= q_{\pi_\theta}(s, a) - v_{\pi_\theta}(s) \\ &= A_{\pi_\theta}(s, a)\end{aligned}$$

- Let's exploit this:

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) A_{\pi_\theta}(s, a)]$$



$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla_\theta \ln \pi(a|s, \theta) \delta_{\pi_\theta}]$$

In practice we can use an approximator:

$$\delta_{\mathbf{v}} = r + \gamma \hat{v}(s', \mathbf{v}) - \hat{v}(s, \mathbf{v})$$

Initialize policy parameter θ, \mathbf{v}

Algorithm parameter : step-size $\alpha, \beta > 0$

Repeat (for each episode) :

 Initialize s and take an action $a \sim \pi(a|s, \theta)$

 Loop for each step of the episode $t = 0, \dots, T - 1$

 Observe reward r and new state s'

$$\delta_{\mathbf{v}} = r + \gamma \hat{v}(s', \mathbf{v}) - \hat{v}(s, \mathbf{v})$$

$$\theta = \theta + \alpha \delta_{\mathbf{v}} \nabla_{\theta} \ln \pi(a|s, \theta)$$

$$\mathbf{v} = \mathbf{v} + \beta \delta_{\mathbf{v}} \nabla \hat{v}(s, \mathbf{v})$$

$$s \leftarrow s'$$

Now the critic is simplified! We get only 1 ste of parameters

Critics at different time-scales

- Critic can estimate value function $v_{\pi_\theta}(s)$ from many targets at different time-scales. From last lectures :

- For MC, the target is the return G_t

$$\Delta \mathbf{v} = \alpha(\textcolor{red}{G_t} - \hat{v}(s, \mathbf{v}))\mathbf{x}(s)$$

- For TD(0), the target is the TD target $r + \gamma \hat{v}(s', \mathbf{v})$

$$\Delta \mathbf{v} = \alpha(\textcolor{red}{r + \gamma \hat{v}(s', \mathbf{v})} - \hat{v}(s, \mathbf{v}))\mathbf{x}(s)$$

- For forward-view TD(λ), the target is the λ - return G_t^λ

$$\Delta \mathbf{v} = \alpha(\textcolor{red}{G_t^\lambda} - \hat{v}(s, \mathbf{v}))\mathbf{x}(s)$$

- For backward-view TD(λ), we use eligibility traces

$$\delta_t = r_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{v}_t) - \hat{v}(s_t, \mathbf{v}_t)$$

$$e_t = \gamma \lambda e_{t-1} + \mathbf{x}(s_t)$$

$$\Delta \mathbf{v} = \alpha \delta_t e_t$$

Actors at different time-scales 1/2

- The policy gradient can also be estimated at many time-scales

$$\nabla J(\theta) = \mathbb{E}_\pi [\nabla \ln \pi(a|s, \theta) A_{\pi_\theta}(s, a)]$$

- Monte-Carlo policy gradient uses error from complete error

$$\Delta\theta = \alpha(\textcolor{red}{G_t} - \hat{v}(S_t, \mathbf{v})) \nabla_\theta \ln \pi(A_t|S_t, \theta)$$

- Actor-critic policy gradient uses the one-step TD error

$$\Delta\theta = \alpha(\textcolor{red}{r + \gamma \hat{v}(S_{t+1}, \mathbf{v})} - \hat{v}(S_t, \mathbf{v})) \nabla_\theta \ln \pi(A_t|S_t, \theta)$$

Actors at different time-scales 2/2

- Just-like forward-view TD(λ), we can mix over time-scales

$$\Delta\theta = \alpha(\textcolor{red}{G_t^\lambda} - \hat{v}(S_t, \mathbf{v}))\nabla_\theta \ln \pi(A_t|S_t, \theta)$$

where $\textcolor{red}{G_t^\lambda} - \hat{v}(S_t, \mathbf{v})$ is a biased estimate of advantage function

- Like backward-view TD(λ), we can also use eligibility traces
 - By equivalence with TD(λ), substituting $\mathbf{x}(s) = \nabla_\theta \ln \pi(A|S, \theta)$

$$\delta_t = r_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{v}_t) - \hat{v}(s_t, \mathbf{v}_t)$$

$$e_t = \gamma \lambda e_{t-1} + \nabla_\theta \ln \pi(a_t|s_t, \theta_t)$$

$$\Delta\theta = \alpha \delta_t e_t$$

- This update can be applied online, to incomplete sequences

Actors at different time-scales 2/2

- Just-like forward-view TD(λ), we can mix over time-scales

$$\Delta\theta = \alpha(\textcolor{red}{G_t^\lambda} - \hat{v}(S_t, \mathbf{v}))\nabla_\theta \ln \pi(A_t|S_t, \theta)$$

where $\textcolor{red}{G_t^\lambda} - \hat{v}(S_t, \mathbf{v})$ is a biased estimate of advantage function

- Like backward-view TD(λ), we can also use eligibility traces
 - By equivalence with TD(λ), substituting $\mathbf{x}(s) = \nabla_\theta \ln \pi(A|S, \theta)$

Does it make sense
to have actor and
critic at different
time-scales?

$$\delta_t = r_{t+1} + \gamma\hat{v}(s_{t+1}, \mathbf{v}_t) - \hat{v}(s_t, \mathbf{v}_t)$$

$$e_t = \gamma\lambda e_{t-1} + \nabla_\theta \ln \pi(a_t|s_t, \theta_t)$$

$$\Delta\theta = \alpha\delta_t e_t$$

- This update can be applied online, to incomplete sequences

Actors at different time-scales 2/2

- Just-like forward-view TD(λ), we can mix over time-scales

$$\Delta\theta = \alpha(G_t^\lambda - \hat{v}(S_t, \mathbf{v}))\nabla_\theta \ln \pi(A_t|S_t, \theta)$$

where $G_t^\lambda - \hat{v}(S_t, \mathbf{v})$ is a biased estimate of advantage function

- Like backward-view TD(λ), we can also use eligibility traces
 - By equivalence with TD(λ), substituting $\mathbf{v}(s) = \nabla_\theta \ln \pi(A|S, A)$

Does it make sense
to have actor and
critic at different
time-scales?

$$\begin{aligned}\delta_t &= r_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{v}_t) - \hat{v}(s_t, \mathbf{v}_t) \\ e_t &= \gamma \lambda e_{t-1} + \nabla_\theta \ln \pi(a_t|s_t, \theta) \\ \Delta\theta &= \alpha \delta_t e_t\end{aligned}$$

We can mix approaches, but
critic should be ‘faster’ (TD λ for
example) than the actor, since
the actor improves based on
the knowledge provided by the
critic!

- This update can be applied online, to incomplete sequences

Actor-critic: some considerations 1/2

- Approximating the policy gradient introduces bias
- A biased policy gradient may not find the right solution
- Luckily, if we choose value function approximation carefully, then we can avoid introducing any bias: we can still follow the exact policy gradient
- Book material:
 - 13.1
 - 13.2 (proof is optional)
 - 13.3
 - 13.4
 - 13.5 (TD-lambda version)
 - Optional 13.6, 13.7 -> continuous case (more on next slide)

Actor-critic: some considerations 2/2

- (Optional) Policy gradient for **continuous problems**: we define performance in terms of the average rate of reward per time step

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

- (Optional) Policy parametrization can be extended to continuous scenarios, for example

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

Policy Gradient: Chapter 13

- 13.1
- 13.2 (Proof of Policy Gradient Theorem Optional)
- 13.3
- 13.4
- 13.5**

Batch methods



Incremental vs. Batch approaches

- Up until now we have been dealing with incremental approaches: data and experience is collected ‘on-line’ in an incremental fashion
- Batch approaches instead store data during experience to achieve some benefits
- Batch approaches were a breakthrough in ATARI [1]
- Notes:
 - No dedicated chapter for batch approaches (some references in chapter 16)
 - We briefly talked about ‘batch’ in the past...

[1] Mnih et al., “Human-level control through deep reinforcement learning” (Nature, 2015)



Prediction: (batch) MC and TD – AB Example

Two states, no discounting, 8 episodes of experience:

A, 0, B, 0

$V(B) = 3/4$

B, 1

$V(A) = 0$ is the Monte Carlo solution

B, 1

-> MC converges to solution with minimum mean-squared error

B, 1

$V(A) = 3/4$ is the TD(0) solution

B, 1

-> TD(0) (implicitly) converges to solution of max likelihood

B, 1

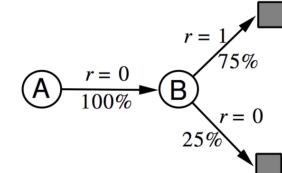
Markov model

B, 1

B, 1

B, 0

What is $V(A)$ and what is $V(B)$?



Batch methods in Reinforcement Learning

- Gradient descent is simple and appealing
- But it is not sample efficient (one of the reason we introduced off-policy)
- Batch methods seek to find the best fitting value function given the agent's experience (training data)

Batch methods in Reinforcement Learning

- Gradient descent is simple and appealing
- But it is not sample efficient (one of the reason we introduced off-policy)
- Batch methods seek to find the best fitting value function given the agent's experience (training data)

Intuition behind batch methods: *in general one of the requirement for SGD optimization is that the training data is **indipendent and identically distributed** and when an Agent interacts with the environment, the sequence of experience tuples can be highly correlated.*



*Learning algorithms that learn from each of these experiences tuples in **sequential order** runs the risk of getting swayed by the effects of this correlation*

Experience Replay

- We can improve this situation using a large buffer of past experience and sample training data from it (a random subset), instead of using latest experience
- This technique is called **replay buffer or experience buffer**. The replay buffer contains a collection of tuples (S, A, R, S') . The tuples are gradually added to the buffer as we are interacting with the environment*
- The act of sampling a small batch of tuples from the replay buffer in order to learn is known as **experience replay**
- In addition to breaking harmful correlations, experience replay allows us to learn more from individual tuples multiple times, recall rare occurrences, and in general make better use of our experience



*The simplest implementation is a buffer of fixed size, with new data added to the end of the buffer so that it pushes the oldest experience out of it

Buffer Reply: Least Squares Prediction

- Given value function approximation $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$
- and experience \mathcal{D} consisting of $\langle \text{state}, \text{value} \rangle$ pairs

$$\mathcal{D} = \{\langle s_1, v_1^\pi \rangle, \langle s_2, v_2^\pi \rangle, \dots, \langle s_T, v_T^\pi \rangle\}$$

- which parameters \mathbf{w} give the *best fitting* value function $\hat{v}(s, \mathbf{w})$

Buffer Reply: Least Squares Prediction

- Given value function approximation $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$
- and experience \mathcal{D} consisting of $\langle \text{state}, \text{value} \rangle$ pairs

$$\mathcal{D} = \{\langle s_1, v_1^\pi \rangle, \langle s_2, v_2^\pi \rangle, \dots, \langle s_T, v_T^\pi \rangle\}$$

- which parameters \mathbf{w} give the *best fitting* value function $\hat{v}(s, \mathbf{w})$
- **Least squares** algorithms find parameter vector \mathbf{w} minimising sum-squared error between $\hat{v}(s_t, \mathbf{w})$ and v_t^π

$$\begin{aligned} LS(\mathbf{w}) &= \sum_{t=1}^T (v_t^\pi - \hat{v}(s_t, \mathbf{w}))^2 \\ &= \mathbb{E}_{\mathcal{D}} [(v^\pi - \hat{v}(s, \mathbf{w}))^2] \end{aligned}$$

SGD with Experience Replay

Given experience \mathcal{D} consisting of $\langle state, value \rangle$ pairs

$$\mathcal{D} = \{\langle s_1, v_1^\pi \rangle, \langle s_2, v_2^\pi \rangle, \dots, \langle s_T, v_T^\pi \rangle\}$$

Repeat :

1. Sample state, value from experience

$$\langle s, v^\pi \rangle \sim \mathcal{D}$$

2. Apply stochastic gradient descent update

$$\Delta \mathbf{w} = \alpha (v^\pi - \hat{v}(s, \mathbf{w})) \nabla_{\mathbf{w}} \hat{v}(s, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta \mathbf{w}$$

SGD with Experience Replay

Given experience \mathcal{D} consisting of $\langle state, value \rangle$ pairs

$$\mathcal{D} = \{\langle s_1, v_1^\pi \rangle, \langle s_2, v_2^\pi \rangle, \dots, \langle s_T, v_T^\pi \rangle\}$$

Repeat :

1. Sample state, value from experience

$$\langle s, v^\pi \rangle \sim \mathcal{D}$$

2. Apply stochastic gradient descent update

$$\Delta \mathbf{w} = \alpha (v^\pi - \hat{v}(s, \mathbf{w})) \nabla_{\mathbf{w}} \hat{v}(s, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta \mathbf{w}$$

We are assuming the availability of the oracle...

SGD with Experience Replay **in practice**

- We do not know true values v_t^π
- In practice, our *training data* must use noisy or biased samples of v_t^π

LSMC Least Squares Monte-Carlo uses return

$$v_t^\pi \approx G_t$$

LSTD Least Squares Temporal-Difference uses TD target

$$v_t^\pi \approx R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w})$$

LSTD(λ) Least Squares TD(λ) uses λ -return

$$v_t^\pi \approx G_t^\lambda$$

Linear Least
Squares
Prediction
Algorithms

**Thank you!
Questions?**

**Lecture #19:
Policy Gradient Methods: Actor
Critic Approaches & Batch Methods**

Gian Antonio Susto

