From Policy Gradient to Actor-Critic methods

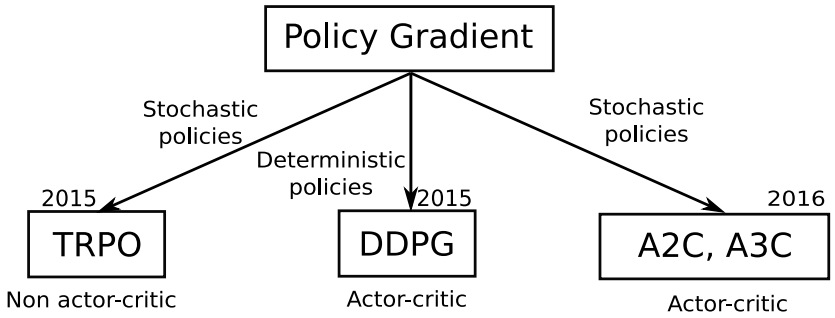
Advantage Actor Critic

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

A2C, A3C

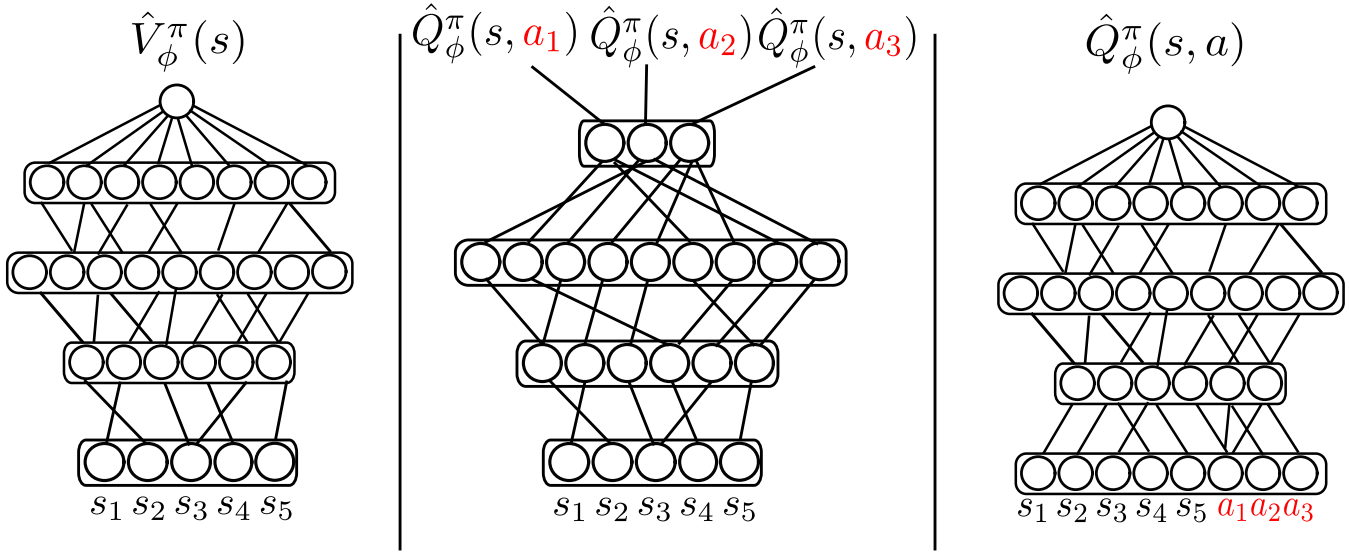
Advantage Actor Critic (A2C)



- A crucial move from Policy Gradient methods to Actor-Critic methods
- The earliest actor-critic algorithm of the deep RL era using stochastic policies
- It directly derives from the basic Policy Gradient method
- The critic is learned using bootstrap, which makes it an actor-critic algorithm
- The A2C paper focuses more on A3C, an asynchronous version where several agents generate data without using a replay buffer
- A2C can be seen as a simplified version of A3C with a single agent

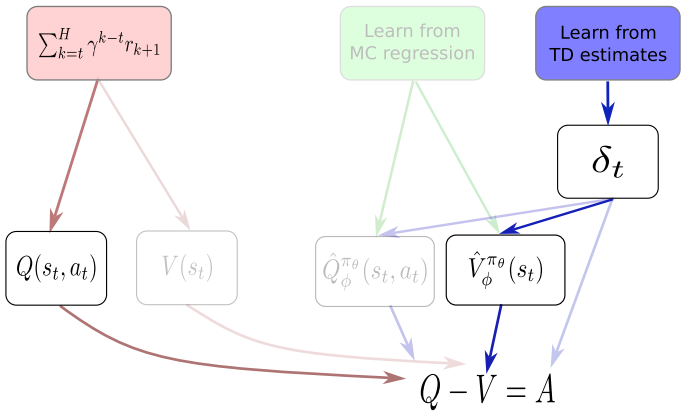
Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. (2016) Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*

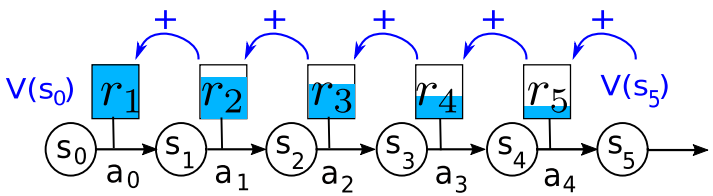
Choice of a V critic



- Main point: By contrast with $Q(s, a)$, $V(s)$ can be estimated in the same way irrespective of using discrete or continuous actions
- $\hat{V}^{\pi}_{\boldsymbol{\phi}}$ is smaller, but not necessarily easier to estimate (implicit max over actions)
- Temporal difference error: $\delta = [r(\mathbf{s}_t) + \gamma V^i_{\boldsymbol{\phi}}(\mathbf{s}_{t+1}) - V^i_{\boldsymbol{\phi}}(\mathbf{s}_t)]$
- Standard update rule: $V^{i+1}_{\boldsymbol{\phi}}(\mathbf{s}_t) \leftarrow V^i_{\boldsymbol{\phi}}(\mathbf{s}_t) + \alpha\delta$

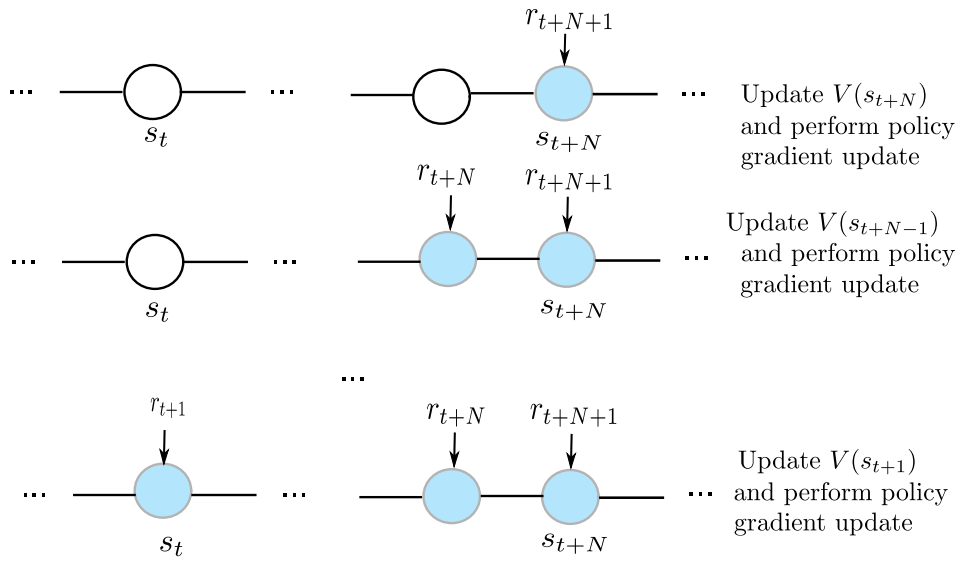
Advantage function calculation





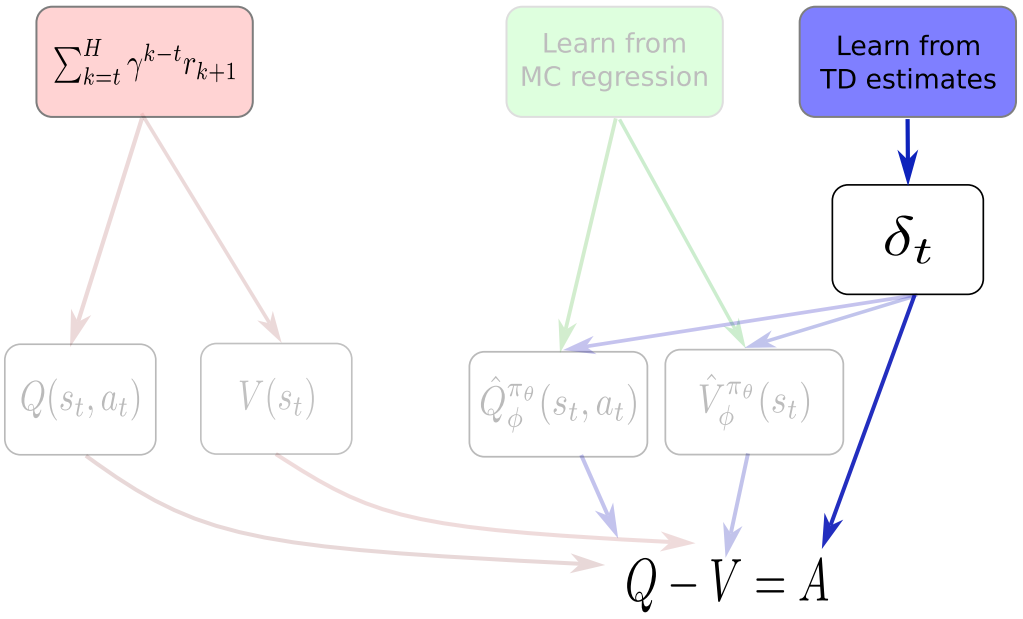
- To perform policy gradient updates, one needs to compute $\hat{A}_{\phi}(\mathbf{s}_t, \mathbf{a}_t)$
- By definition, $A(\mathbf{s}_t, \mathbf{a}_t) = Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)$
- A2C computes the advantage with $\hat{A}_{\phi}(\mathbf{s}_t, \mathbf{a}_t) = R_t(\mathbf{s}_t) - V_{\phi}(\mathbf{s}_t)$
- $R_t(\mathbf{s}_t) = \sum_{i=0}^{N-1} \gamma^i r_{t+i+1} + \gamma^N V_{\phi}(\mathbf{s}_{t+N})$ is the return of the current N-step trajectory from state $\mathbf{s}_t$
- $R_t(\mathbf{s}_t)$ can be seen as an approximate of $Q(\mathbf{s}_t, \mathbf{a}_t)$ computed along one trajectory
- Actually, computed on N steps rather than the full trajectory

N-step updates



- The agent performs N steps in the environment (or less if the episode stops earlier in the episodic case) before each update
- At each update, the agent has collected up to N states and rewards
- It can update the value of the last state using the last reward, the value of the second last step with two rewards
- And so on up to the first state of the current collection
- It updates both the critic and the policy at each update
- Straightforward in BBRL

Alternative implementation



- With this approach, the advantage can be computed out of current step information only
- Makes it possible to add a replay buffer and make it more off-policy
- In *bbrl_algos*, use the GAE version

Policy Gradient updates

- The standard Policy Gradient update is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(.)}[\nabla_{\boldsymbol{\theta}}[\log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)})] \hat{A}_{\boldsymbol{\phi}}(\mathbf{s}_t, \mathbf{a}_t)]$$

- But to favor exploration, A2C adds an entropy term to the gradient calculation
- Thus the policy update rule is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(.)}[\nabla_{\boldsymbol{\theta}}[\log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)})(R_t - V_{\boldsymbol{\phi}}(\mathbf{s}_t)) - \beta \mathcal{H}(\pi_{\boldsymbol{\theta}}(\mathbf{s}_t))]]$$

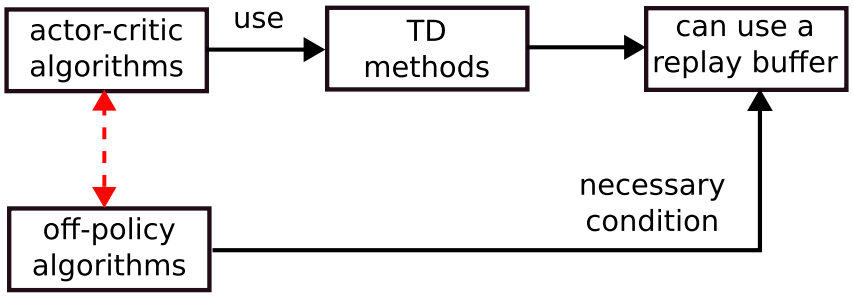
- where $\mathcal{H}(\pi_{\boldsymbol{\theta}}(\mathbf{s}_t))$ is the entropy of policy $\pi_{\boldsymbol{\theta}}$ at state $\mathbf{s}_t$.
- Note that A2C adds entropy in the update of the actor, but outside the critic, whereas SAC adds it in the critic target, which has a deeper impact.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A. Abbeel, P. et al. (2018) Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*

Summary: main distinguishing features

- To perform policy gradient, you need the advantage function
- Computes the advantage function as value function minus the return of the current N-step trajectory
- Adds entropy regularization to favor exploration in the gradient calculation step
- Uses N-step updates
- Does not use a replay buffer
- Note that A2C is Actor-Critic, but on-policy, so one cannot equate Actor-Critic and off-policy

Off-policy and actor-critic



- Because AC algorithms use a TD mechanism, they perform one-step (or n-step) updates
- Performing such updates is a necessary condition for using a replay buffer
- Thus AC algos often use a replay buffer
- DDPG, TD3 and SAC are AC algos, they use a replay buffer and they are said off-policy
- Thus AC algos are often said off-policy
- NB: A2C and A3C are AC algos but do not use a replay buffer (they could)
- There is no equivalence between being off-policy and being actor-critic

Any question?

Send mail to: Olivier.Sigaud@upmc.fr

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018).

Soft actor-critic algorithms and applications.
arXiv preprint arXiv:1812.05905.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016).

Asynchronous methods for deep reinforcement learning.
In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org.