

Reinforcement Learning

Lecture 8: Policy gradient methods

Robert Lieck

Durham University



Lecture covers chapter 13 in Sutton & Barto [1] and examples from David Silver [2]

1 Policy-based methods

- definition
- characteristics
- deterministic vs stochastic policies

2 Policy gradients

- gradient-based estimator
- Monte Carlo REINFORCE

3 Actor-critic methods

- definition
- algorithm
- function approximation error in AC methods
- extensions

Definition: policy-based methods

Last week, we used a function approximator to estimate the value function:

$$\hat{v}(s, \mathbf{w}) \approx v_{\pi}(s),$$

and for control we estimated Q :

$$\hat{q}(s, a, \mathbf{w}) \approx q_{\pi}(s, a).$$

This week we will estimate policies:

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

Given a state, what's the distribution over actions?

Example: what's the optimal policy?



Policy-based RL characteristics

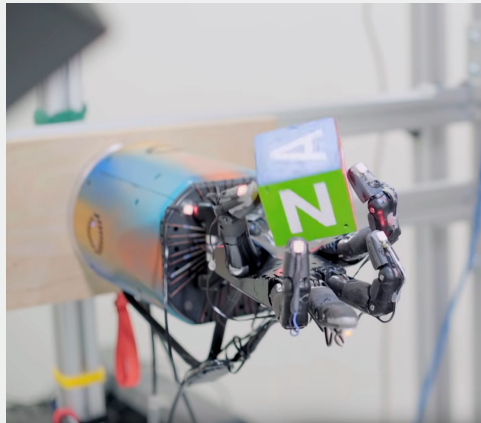
This approach has the following **advantages**:

- Can be more efficient than calculating the value function
- Better convergence guarantees
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

And the following **disadvantages**:

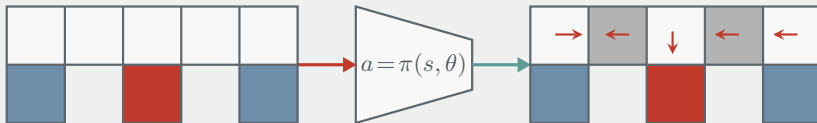
- Converges on local rather than global optimum
- Inefficient policy evaluation with high variance

Example: continuous action spaces

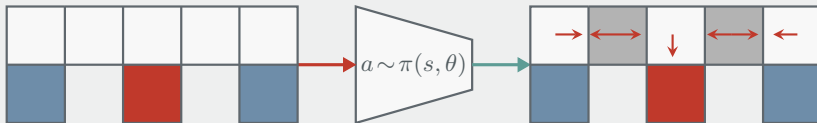


Example: deterministic vs stochastic policies

Deterministic policy for feature vectors describing the walls around a state:



Stochastic policy:





Problem: ∇ through stochastic decisions

We **estimate the advantage** $\hat{A}(a, s)$ (or value for higher variance) for state s under a stochastic policy $\pi_\theta(a | s)$ **by sampling from the policy** (MC estimate).

Typically π follows a categorical distribution (softmax) or a Gaussian for continuous action spaces. We could use zeroth-order optimisation that does not require gradients (stochastic annealing, genetic algorithms, simplex downhill etc.), but this does not work well in high-dimensional spaces (e.g. with deep learning).

To optimise the policy, we need the gradient, but **sampling breaks backpropagation!**

Solution: log-derivative trick

$$\begin{aligned} & \nabla_\theta \mathbb{E}_{\pi_\theta(a|x)} [\hat{A}(a, s)] \\ &= \nabla_\theta \sum_a \pi_\theta(a | x) \hat{A}(a, s) \\ &= \sum_a \pi_\theta(a | x) \frac{\nabla_\theta [\pi_\theta(a | x) \hat{A}(a, s)]}{\pi_\theta(a | x)} \\ &= \sum_a \pi_\theta(a | x) \hat{A}(a, s) \frac{\nabla_\theta \pi_\theta(a | x)}{\pi_\theta(a | x)} \\ &= \sum_a \pi_\theta(a | x) \hat{A}(a, s) \nabla_\theta \log \pi_\theta(a | x) \\ &\approx \frac{1}{n} \sum_{a \sim \pi_\theta(a|x)} \hat{A}(a, s) \nabla_\theta \log \pi_\theta(a | x) \end{aligned}$$



Definition: Monte Carlo REINFORCE

REINFORCE estimates the return in the previous equation by using a Monte Carlo estimate [3].

- Initialise some arbitrary parameters θ
- Iteratively sample episodes
- Calculate the complete return from each step
- For each step again, update in the gradient times the sample return

Algorithm: Monte Carlo REINFORCE

PyTorch example: ↗

```
# initialise  $\theta$  with random values  
 $\pi = \text{PolicyNetwork}(\theta)$ 
```

```
while(True):
```

```
    # sample episode following  $\pi$   
     $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T \sim \pi$ 
```

```
    for  $t$  in range( $T - 1$ ):
```

```
         $G_t \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k$ 
```

```
         $\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla \ln \pi(A_t | S_t, \theta)$ 
```

Definition: actor-critic methods

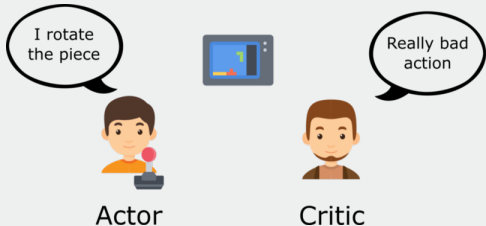
We combine policy gradients with action-value function approximation, using two models that may (optionally) share parameters.

- We use a **critic** to estimate the Q values:

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

- We use an **actor** to update the policy parameters θ in the direction suggested by the critic.

Example: Actor critic





Definition: actor-critic

Putting this together, actor-critic methods use an approximate policy gradient to adjust the actor policy in the direction that maximises the reward according to the critic:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \log \pi_{\theta}(a|s) q_{\mathbf{w}}(s, a)$$

Algorithm: Actor-Critic (PyTorch)

```
# initialise  $s, \theta, \mathbf{w}$  randomly
# sample  $a \sim \pi_{\theta}(a|s)$ 
for  $t$  in range( $T$ ):
    sample  $r_t$  and  $s'$  from environment( $s, a$ )
    sample  $a' \sim \pi_{\theta}(a'|s')$ 
     $\theta \leftarrow \theta + \alpha q_{\mathbf{w}}(s, a) \nabla_{\theta} \ln \pi_{\theta}(a|s)$  # update actor
     $\delta_t = r_t + \gamma q_{\mathbf{w}}(s', a') - q_{\mathbf{w}}(s, a)$  # TD error
     $\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta_t \nabla_{\mathbf{w}} q_{\mathbf{w}}(s, a)$  # update critic
     $a \leftarrow a', s \leftarrow s'$ 
```

Function approximation error in actor-critic

In practice, tricks are needed to reduce function approximation in actor-critic methods. TD3 [4]:

- Dual critic networks + actor network
- Adds $\epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$ to actions
- Delayed policy updates, replay buffer



Algorithm: TD3 [4]

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ

Initialize target networks $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$

Initialize replay buffer \mathcal{B}

for $t = 1$ **to** T **do**

 Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$,

$\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'

 Store transition tuple (s, a, r, s') in \mathcal{B}

 Sample mini-batch of N transitions (s, a, r, s') from \mathcal{B}

$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$

$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$

 Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$

if $t \bmod d$ **then**

 Update ϕ by the deterministic policy gradient:

$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$

 Update target networks:

$\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i$

$\phi' \leftarrow \tau \phi + (1 - \tau) \phi'$

end if

end for



Extensions

This has introduced the foundations, hopefully now you have a good platform to read about the extensions to this.

1. **Recommended further study (papers & code):** [↗](#)
2. **Recommended further study (theory & STAR):** [↗](#)

Extensions include:


- Advantage actor critic (A3C & A2C) [5]
- Experience replay & prioritised replay [6]
- Proximal policy optimisation [7]
- Addressing Function Approximation Error in Actor-Critic Methods (TD3) [4] - **recommended starter**



Summary

In summary:

- Policy gradients open up many new extensions
- Choose extensions to reduce variance to stabilise training
- And extensions to reduce overestimation bias of value function
- Consider regularisation to encourage exploration
- Going off-policy gives better exploration
- Its possible for the actor and critic to share some lower layer parameters, but be careful about it
- Experience replay can increase sample efficiency (where simulation is expensive)

- [1] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction (second edition). Available online  MIT press, 2018.
- [2] David Silver. Reinforcement Learning lectures. <https://www.davidsilver.uk/teaching/>. 2015.
- [3] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: Machine learning 8.3-4 (1992), pp. 229–256.
- [4] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: International Conference on Machine Learning. PMLR. 2018, pp. 1587–1596.
- [5] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: International conference on machine learning. 2016, pp. 1928–1937.
- [6] Ziyu Wang et al. "Sample efficient actor-critic with experience replay". In: arXiv preprint arXiv:1611.01224 (2016).



- [7] John Schulman et al. "Proximal policy optimization algorithms". In: arXiv preprint arXiv:1707.06347 (2017).
- [8] Shixiang Gu et al. "Q-prop: Sample-efficient policy gradient with an off-policy critic". In: arXiv preprint arXiv:1611.02247 (2016).