Actor-Critic Methods (A2C)

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Spring 2020

avereshc@buffalo.edu

April 8, 2020

*Slides are adopted from Deep Reinforcement Learning by Sergey Levine & Policy Gradients by David Silver
1 Types of RL Algorithms

2 Actor-Critic
Types of RL algorithms

\[
\theta^* = \arg \max_{\theta} R_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right]
\]

- Model-based RL:

- Value-based:

- Policy-gradient:

- Actor-critic:
Types of RL algorithms

\[ \theta^* = \arg \max_{\theta} R_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

- **Model-based RL**: estimate the transition model and then:
  - Use it for planning (no explicit policy)
  - Use it to improve a policy

- **Value-based**: estimate value function or Q-function of the current policy (no explicit policy)

- **Policy-gradient**: directly differentiate the objective

- **Actor-critic**: estimate value function or Q-function of the current policy, use it to improve the policy
Types of RL algorithms

\[ \theta^* = \arg \max_{\theta} R_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

- **Model-based RL**: estimate the transition model and then:
  - Use it for planning (no explicit policy)
  - Use it to improve a policy

- **Value-based**:

- **Policy-gradient**:
- **Actor-critic**: estimate value function or Q-function of the current policy, use it to improve the policy
Types of RL algorithms

\[ \theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

- **Model-based RL**: estimate the transition model and then:
  - Use it for planning (no explicit policy)
  - Use it to improve a policy
- **Value-based**: estimate value function or Q-function of the current policy (no explicit policy)
- **Policy-gradient**: 

Alina Vereshchaka (UB)  
CSE4/510 Reinforcement Learning, Lecture 20  
April 8, 2020
Types of RL algorithms

\[ \theta^* = \arg \max_{\theta} R_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

- **Model-based RL**: estimate the transition model and then:
  - Use it for planning (no explicit policy)
  - Use it to improve a policy

- **Value-based**: estimate value function or Q-function of the current policy (no explicit policy)

- **Policy-gradient**: directly differentiate the objective

- **Actor-critic**: 
Types of RL algorithms

\[ \theta^* = \arg \max_{\theta} R_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

- **Model-based RL**: estimate the transition model and then:
  - Use it for planning (no explicit policy)
  - Use it to improve a policy

- **Value-based**: estimate value function or Q-function of the current policy (no explicit policy)

- **Policy-gradient**: directly differentiate the objective

- **Actor-critic**: estimate value function or Q-function of the current policy, use it to improve the policy
Model-based Algorithms

- fit a model
  - learn $p(s_{t+1}|s_t, a_t)$

- improve the policy
  - a few options

- generate samples (i.e. run the policy)
Value Based Algorithms

Examples:
- Value-Iteration
- Q-Learning
- DQN

1. fit a model/estimate the return: \( \hat{V}(s) \) or \( \hat{Q}(s, a) \)
2. generate samples (i.e. run the policy)
3. improve the policy: set \( \pi(s) = \arg\max_a Q(s, a) \)
Direct Policy Gradient

\[ R_\tau = \sum_t r(s_t, a_t) \]

\[ \theta \leftarrow \theta + \alpha \nabla_\theta E[\sum_t r(s_t, a_t)] \]
Actor-critic: Value Function + Policy Gradients

- fit a model/estimate the return

- evaluate returns using $V$ or $Q$

- generate samples (i.e. run the policy)

- improve the policy

\[
\theta \leftarrow \theta + \alpha \nabla_{\theta} E \left[ \sum_t r(s_t, a_t) \right]
\]
Comparison: Sample Efficiency

- **Sample efficiency**: How many samples do we need to get a good policy?
**Sample efficiency**: How many samples do we need to get a good policy?

Most important questions: Is the algorithm off policy?

- **Off policy**: able to improve the policy without generating new samples from that policy
Comparison: Sample Efficiency

- **Sample efficiency**: How many samples do we need to get a good policy?
- Most important questions: Is the algorithm off policy?
  - **Off policy**: able to improve the policy without generating new samples from that policy
  - **On policy**: each time the policy is changed, even a little bit, we need to generate new samples
Comparison: Sample Efficiency

More efficient (fewer samples)
- model-based shallow RL
- model-based deep RL
- off-policy Q-function learning

Less efficient (more samples)
- actor-critic style methods
- on-policy policy gradient algorithms
- evolutionary or gradient-free algorithms
REINFORCE (Monte-Carlo Policy Gradient)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return $G_t$ as an unbiased sample of $Q^\pi(\theta)(s_t, a_t)$

$$\Delta \theta_t = \alpha G_t \nabla_{\theta} \log \pi(\theta)(s_t, a_t)$$

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \theta), \forall a \in A, s \in S, \theta \in \mathbb{R}^n$
Initialize policy weights $\theta$
Repeat forever:
Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$
For each step of the episode $t = 0, \ldots, T-1$:
$G_t \leftarrow$ return from step $t$
$\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_{\theta} \log \pi(A_t|S_t, \theta)$
Solution

 Policy Update: \[ \Delta \theta = \alpha * \nabla_\theta * (\log \pi(S_t, A_t, \theta)) * R(t) \]

 New update: \[ \Delta \theta = \alpha * \nabla_\theta * (\log \pi(S_t, A_t, \theta)) * Q(S_t, A_t) \]
Table of Contents

1 Types of RL Algorithms

2 Actor-Critic
Monte-Carlo policy gradient still has high variance

We can use a critic to estimate the action-value function:

$$Q_w(s, a) \approx Q_{\pi_\theta}(s, a)$$
Actor-Critic

- Monte-Carlo policy gradient still has **high variance**

- We can use a **critic** to estimate the action-value function:

\[ Q_w(s, a) \approx Q_{\pi_{\theta}}(s, a) \]

- Actor-critic algorithms maintain **two** sets of parameters
  - **Critic** Updates action-value function parameters \( w \)

Monte-Carlo policy gradient still has high variance

We can use a critic to estimate the action-value function:

\[ Q_w(s, a) \approx Q_{\pi_\theta}(s, a) \]

Actor-critic algorithms maintain two sets of parameters

- **Critic** Updates action-value function parameters \( w \)
- **Actor** Updates policy parameters \( \theta \), in direction suggested by critic
Actor-Critic

- Monte-Carlo policy gradient still has high variance
- We can use a critic to estimate the action-value function:

\[ Q_w(s, a) \approx Q_{\pi_\theta}(s, a) \]

- Actor-critic algorithms maintain two sets of parameters
  - Critic Updates action-value function parameters \( w \)
  - Actor Updates policy parameters \( \theta \), in direction suggested by critic
- Actor-critic algorithms follow an approximate policy gradient

\[ \nabla_\theta J(\theta) \approx E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a)Q_w(s, a)] \]
Actor-Critic

- Monte-Carlo policy gradient still has **high variance**

- We can use a **critic** to estimate the action-value function:

\[ Q_w(s, a) \approx Q_{\pi^\theta}(s, a) \]

- **Actor-critic algorithms maintain** *two* sets of parameters
  - **Critic** Updates action-value function parameters \( w \)
  - **Actor** Updates policy parameters \( \theta \), in direction suggested by critic

- Actor-critic algorithms follow an approximate policy gradient

\[
\nabla_{\theta} J(\theta) \approx E_{\pi^\theta} [\nabla_{\theta} \log \pi^\theta(s, a) Q_w(s, a)] \\
\Delta \theta = \alpha \nabla_{\theta} \log \pi^\theta(s, a) Q_w(s, a)
\]
Actor-Critic

I rotate the piece

Really bad action

Actor

Critic
Actor-Critic

- The **actor** is the policy $\pi_\theta(a|s)$ with parameters $\theta$ which conducts actions in an environment.

The critic computes value functions to help assist the actor in learning. These are usually the state value, state-action value, or advantage value, denoted as $V(s)$, $Q(s,a)$, and $A(s,a)$, respectively.
The **actor** is the policy $\pi_\theta(a|s)$ with parameters $\theta$ which conducts actions in an environment.

The **critic** computes value functions to help assist the actor in learning. These are usually the state value, state-action value, or advantage value, denoted as $V(s)$, $Q(s,a)$, and $A(s,a)$, respectively.
Actor-Critic
The critic is solving a familiar problem: policy evaluation.

How good is policy $\pi_\theta$ for current parameters $\theta$?
The critic is solving a familiar problem: policy evaluation

How good is policy $\pi_\theta$ for current parameters $\theta$?

To estimate, use any policy evaluation method:

- Monte-Carlo policy evaluation
- Temporal-Difference learning
- Least-squares policy evaluation
Estimating the TD Error

For the true value function $V_{\pi\theta}(s)$, the TD error $\delta_{\pi\theta}$

$$\delta_{\pi\theta} = r + \gamma V_{\pi\theta}(s') - V_{\pi\theta}(s)$$

So we can use the TD error to compute the policy gradient

$$\nabla_{\theta} J(\theta) = E_{\pi\theta}[\nabla_{\theta} \log \pi_{\theta}(s, a) \delta_{\pi\theta}]$$

In practice we can use an approximate TD error, that requires one set of parameters

$$\delta_{w} = r + \gamma V_{w}(s') - V_{w}(s)$$
For the true value function $V_{\pi\theta}(s)$, the TD error $\delta_{\pi\theta}$

$$\delta_{\pi\theta} = r + \gamma V_{\pi\theta}(s') - V_{\pi\theta}(s)$$
Estimating the TD Error

- For the true value function $V_{\pi\theta}(s)$, the TD error $\delta_{\pi\theta}$

$$\delta_{\pi\theta} = r + \gamma V_{\pi\theta}(s') - V_{\pi\theta}(s)$$

- is an unbiased estimate of the advantage function

$$\mathbb{E}_{\pi\theta}[\delta_{\pi\theta} | s, a] = \mathbb{E}_{\pi\theta} \left[ r + \gamma V_{\pi\theta}(s') | s, a \right] - V_{\pi\theta}(s)$$
Estimating the TD Error

- For the true value function $V_{\pi\theta}(s)$, the TD error $\delta_{\pi\theta}$
  \[ \delta_{\pi\theta} = r + \gamma V_{\pi\theta}(s') - V_{\pi\theta}(s) \]

- is an unbiased estimate of the advantage function
  \[
  \mathbb{E}_{\pi\theta}[\delta_{\pi\theta}|s, a] = \mathbb{E}_{\pi\theta}\left[ r + \gamma V_{\pi\theta}(s')|s, a \right] - V_{\pi\theta}(s) = Q_{\pi\theta}(s, a) - V_{\pi\theta}(s)
  \]
For the true value function $V_{\pi\theta}(s)$, the TD error $\delta_{\pi\theta}$

$$\delta_{\pi\theta} = r + \gamma V_{\pi\theta}(s') - V_{\pi\theta}(s)$$

is an unbiased estimate of the advantage function

$$E_{\pi\theta}[\delta_{\pi\theta}|s, a] = E_{\pi\theta}\left[r + \gamma V_{\pi\theta}(s')|s, a\right] - V_{\pi\theta}(s)$$

$$= Q_{\pi\theta}(s, a) - V_{\pi\theta}(s)$$

$$= A_{\pi\theta}(s, a)$$
For the true value function $V_{\pi\theta}(s)$, the TD error $\delta_{\pi\theta}$

$$\delta_{\pi\theta} = r + \gamma V_{\pi\theta}(s') - V_{\pi\theta}(s)$$

is an unbiased estimate of the advantage function

$$\mathbb{E}_{\pi\theta}[\delta_{\pi\theta} | s, a] = \mathbb{E}_{\pi\theta}
\left[r + \gamma V_{\pi\theta}(s') | s, a\right] - V_{\pi\theta}(s)
\quad = Q_{\pi\theta}(s, a) - V_{\pi\theta}(s)
\quad = A_{\pi\theta}(s, a)$$

So we can use the TD error to compute the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi\theta} [\nabla_\theta \log \pi_\theta(s, a) \delta_{\pi\theta}]$$
Estimating the TD Error

- For the true value function $V_{\pi \theta}(s)$, the TD error $\delta_{\pi \theta}$
  $$
  \delta_{\pi \theta} = r + \gamma V_{\pi \theta}(s') - V_{\pi \theta}(s)
  $$

- is an unbiased estimate of the advantage function
  $$
  \mathbb{E}_{\pi \theta}[\delta_{\pi \theta}|s,a] = \mathbb{E}_{\pi \theta}\left[r + \gamma V_{\pi \theta}(s')|s,a\right] - V_{\pi \theta}(s) = Q_{\pi \theta}(s,a) - V_{\pi \theta}(s) = A_{\pi \theta}(s,a)
  $$

- So we can use the TD error to compute the policy gradient
  $$
  \nabla_{\theta} J(\theta) = \mathbb{E}_{\pi \theta}[\nabla_{\theta} \log \pi_{\theta}(s,a) \delta_{\pi \theta}]
  $$

- In practice we can use an approximate TD error, that requires one set of parameters $w$
  $$
  \delta_{w} = r + \gamma V_{w}(s') - V_{w}(s)
  $$
One-step Actor–Critic (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \theta)$
Input: a differentiable state-value function parameterization $\hat{v}(s,w)$
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 0)
Loop forever (for each episode):
  Initialize $S$ (first state of episode)
  $I \leftarrow 1$
Loop while $S$ is not terminal (for each time step):
  $A \sim \pi(\cdot|S, \theta)$
  Take action $A$, observe $S', R$
  $\delta \leftarrow R + \gamma \hat{v}(S',w) - \hat{v}(S,w)$ (if $S'$ is terminal, then $\hat{v}(S',w) \approx 0$)
  $w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S,w)$
  $\theta \leftarrow \theta + \alpha^\theta I \delta \nabla \ln \pi(A|S, \theta)$
  $I \leftarrow \gamma I$
  $S \leftarrow S'$