Gradient Actor-Critic Algorithm under Off-policy Sampling and Function Approximation

Youngsuk Park

PhD Candidate, Stanford University

Dec 3, 2018
Outline

- RL introduction
- RL background
  - Class of RL algorithm
  - Modularity and scalablity of RL
- New actor-critic method: gradient actor-critic (GAC)
- Empirical studies
  - simple two-state examples
  - classic control problems
  - atari game and mojuco environment (next)
Consider the following interface

- agent’s goal is to select actions to maximize long-term rewards
  - long-term rewards is called value $V$
  - learn policy $\pi(\text{state}) = \text{action}$, rule of how to act on state
- how can agent achieve the goal efficiently?
  - cannot store/refer to all past history, e.g.) $\#\text{state} = 10^{170}$ in Go
  - use RL that has the collection of algorithms to find optimal policy
Q-learning is one of value-base methods

- predictor learns $Q(s, a)$ value, future rewards at state $s$ for action $a$

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \max_a Q(s', a) - Q(s, a)]$$

- control is determined by Q-value in prediction
- pros: online learning, etc
- cons: does not scale for continuous (high-dim discrete) actions space
Background: Policy Gradient Method

REINFORCE is one of policy gradient methods

- policy $\pi$ is parameterized with $\theta$, e.g.) $\pi(a \mid s; \theta) = \mathcal{N}(\theta^T \phi(s), 1)$
- learns policy parameter $\theta$

$$
\theta \leftarrow \theta + \beta \left( \sum_{i=t}^{\infty} r_i - b \right) \nabla \ln \pi
$$

where $b$ is some baseline

- no prediction/estimation of any value w.r.t $\pi$

- cons: have to wait long time (off-line), etc

- pros: scales well for continuous action space, etc
actor-critic methods is hybrid of value-based and policy gradient methods

- critic (in prediction) learns to estimate $V^\pi$, giving feedback to actor
- actor (in control) improves policy $\pi$ and generates actions
- overcomes weakness of previous two methods
  - scalable for continuous action space (vs. value-based)
  - online learning (vs. policy gradient)
- has two separate components
Background: Control with Exploration/Exploitation

- in control, exploration/exploitation can be important
  - just exploit via best policy learned so far (from history)
  - or maybe consider to explore more (for the better future)

Q) while exploring environment, can we still learn optimal policy?
  - yes, we can via off-policy learning!
  - behavior policy $\pi_b$ just generates actions, target policy $\pi_t$ is learned
Gradient Actor-Critic for Off-Policy

1Off-PAC

\[ \begin{align*}
(\text{critic}) & \quad w \leftarrow w + \alpha \rho \delta \phi(s) \\
(\text{actor}) & \quad \theta \leftarrow \theta + \beta \rho \delta \nabla \ln \pi \\
\text{- state feature} & \quad \phi(s), \text{ TD error } \delta = r(s, a) + \gamma w^T \phi(s') - w^T \phi(s) \\
\text{- ratio} & \quad \rho = \frac{\pi_t(a|s)}{\pi_b(a|s)}
\end{align*} \]

Gradient Actor-Critic for Off-Policy

- (new) gradient actor-critic (with parameter $\lambda$)

\[(\text{critic}) \quad w \leftarrow w + \alpha \rho \delta e^\lambda\]

\[(\text{actor}) \quad \theta \leftarrow \theta + \beta \rho \delta \psi^\lambda\]

- ratio $\rho = \frac{\pi_t(a_t | s_t)}{\pi_b(a_t | s_t)}$
- $e^\lambda$ is the combination of $(\phi(s_t), \ldots, \phi(s_0))$
- $\psi^\lambda$ is the combination of $\nabla \ln \pi(a_t | s_t), \ldots, \nabla \ln \pi(a_0 | s_0)$
Properties of Gradient Actor-Critic

- GAC allows bootstrap parameter $\lambda \in [0, 1]$
  
  \[
  \begin{align*}
  \text{(critic)} \quad & w \leftarrow w + \alpha \rho \delta e^\lambda \\
  \text{(actor)} \quad & \theta \leftarrow \theta + \beta \rho \delta \psi^\lambda
  \end{align*}
  \]

  where $\lambda$ decides how much remember/forget past features

- prove GAC converges to optimal for $\lambda = 1$

- show that Off-PAC can have bias (see in examples later)

- in practice, choose $\lambda = 1 - \epsilon$ for less variance but (potential) bias

  and

- prove its bias is within $O\left(\frac{\gamma}{(1-\gamma)^2} \epsilon\right)$
Examples 1: Short Corridor

- 4 corridors where 2nd corridor is abnormal
- agent can only distinguish goal or non-goal corridor
- optimal policy is stochastic with $\Pr(\text{action=right}) = 0.6$

- behavior policy is uniform-random, still learn optimal with $\lambda \approx 1$
- large biased solution for $\lambda < 0.8$
- note Q-learning cannot learn optimal
Examples 2: $\theta$ to $2\theta$ Counter example

- two state $s = 1, 2$
- optimal policy is taking action 1 for every state
- use the feature $\phi(s = 1) = 1$, $\phi(s = 2) = 2$, thus $V_\theta(s) = s\theta$

- with $\lambda \approx 1$, GAC learn optimal
- Off-PAC ($\lambda = 0$) fails
Examples 3: Mountain Car

- *continuous* state space (position, velocity) in $\mathbb{R}^2$
- discrete action space [left, stay, right]
- car moves according to dynamical system
- reward is $-1$ if it has not reached the goal yet
- behavior policy is uniform random (timesteps to reach $> 5000$)
- every 100 episodes, evaluate the performance of target policy
Examples 4: Pendulum

- *continuous* state (angle, angular velocity), represented by tilecoding
- *continuous* action (torque), modeled by Gaussian
- reward is based on position and velocity
- goal is to make pendulum stand
Examples 5: Mojuco and Atri Game (Next)

Figure: humanoid in Mojuco and atari game in Gym

- input is just pixel information
- need to use DL to represent state from input
Summary & Future Work

- RL agent has two components: prediction and control
- actor-critic is scalable on action and state space (under function approx.)
- off-policy (with target and behavior) can allow distributed learning
- GAC is (first) convergent actor-critic method under off-policy and function approximation
- we can warm-start with reasonable behavior
- next: apply GAC in mojuco and atari game environment that use DL to represent features