Policy Gradient and Actor-Critic Algorithms[1]

[1] Sections 6.1: Yang&Ying

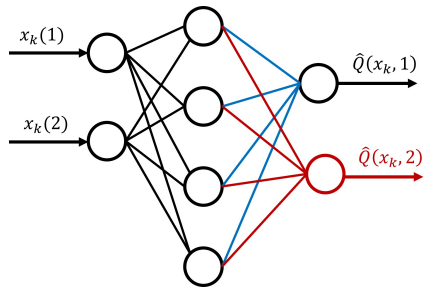
Q-learning follows value iteration

$$\tilde{Q}(i,u) \leftarrow \tilde{Q}(i,u) + \beta_k \left(r(i,u) + \alpha \max_v \tilde{Q}(j,v) - \tilde{Q}(i,u) \right)$$

Q-learning follows value iteration

$$\tilde{Q}(i,u) \leftarrow \tilde{Q}(i,u) + \beta_k \left(r(i,u) + \alpha \max_v \tilde{Q}(j,v) - \tilde{Q}(i,u) \right)$$

Works well when the action space is finite.



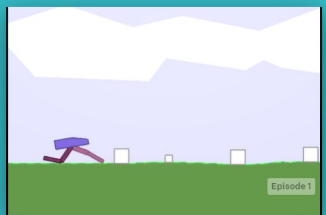
Policy Iteration

How about applications with infinite (countable or uncountable) action spaces?

BipedalWalkerHardcore-v2

Hardcore version with ladders, stumps, pitfalls. Time limit is increased due to obstacles. Reward is given for moving forward, total 300+ points up to the far end. If the robot falls, it gets -100. Applying motor torque costs a small amount of points, more optimal agent will get better score. State consists of hull angle speed, angular velocity, horizontal speed, vertical speed, position of joints and joints angular speed, legs contact with ground, and 10 lidar rangefinder measurements. There's no coordinates in the state vector.

VIEW SOURCE ON GITHUB



RandomAgent on BipedalWalkerHardcore-v2

Policy Iteration

How about applications with infinite (countable or uncountable) action spaces?

Policy Iteration

Given policy μ_k,

- policy evaluation to compute V_{μ_k}

$$V_{\mu_k}(i) = E[r(i,u)] + \alpha \sum_j P_{ij}(u) V_{\mu_k}(j)$$

- policy improvement to obtain a new policy μ_{k+1}

Deep Actor-Critic Algorithms

Actor-critic: generalized policy-iteration

- Critic: estimate the value of the current policy (Q-function in general, because the actor needs the action values)
  TD(λ), double-Q, clipped double-Q, etc.
- Actor:
  - ϵ-greedy based on the current Q-function
  - policy-gradient

Actor-Critic

Parameterized Policy

$\pi_w(u|x)$: probability of choosing action u in state x. The policy (probability) is parameterized by w.

Examples

Finite action space: Gibbs policy (Softmax Policy or Boltzman exploration)

$$\pi_w(a|x) = \frac{\exp\left(w^T \phi(x,a)\right)}{\sum_{u \in A} \exp\left(w^T \phi(x,u)\right)}$$

where $\phi(x,u)$ is the feature vector.

Examples

Finite action space: Gibbs policy (Softmax Policy or Boltzman exploration)

$$\pi_w(a|x) = \frac{\exp\left(w^T \phi(x,a)\right)}{\sum_{u \in A} \exp\left(w^T \phi(x,u)\right)}$$

where $\phi(x,u)$ is the feature vector.

Continuous action space: Gaussian policy

$$\pi_w(a|x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma_w(x))}} \exp\left\{-(a-\mu_w(x))^T \Sigma_w^{-1}(x)(a-\mu_w(x))\right\}$$

where $\mu_w(x) = \phi^T(x)w$ and $\Sigma_w(x) = \sigma^2 I$.

Policy Gradient

$$w \leftarrow w + \beta \nabla_w V_w(x_0)$$

Recall that

$$V_w(x_0) = \sum_u Q_w(x_0, u) \pi_w(u|x_0).$$

Policy Gradient Theorem

$$\begin{aligned}\nabla_w V_w(x_0) =& E\left[\sum_{k=0}^{\infty} \alpha^k \nabla_w \log \pi_w(u_k|x_k) Q_w(x_k, u_k)\right] \\ =& E_{x\sim\rho_w, u\sim\pi_w(u|x)}\left[Q_w(x,u)\nabla_w \log \pi_w(u|x)\right],\end{aligned}$$

where

$$\rho_w(x) = (1-\alpha)\sum_{k=0}^{\infty} \alpha^k P\left(x_k = x\right),$$

called discounted state distribution.

Policy Gradient Algorithms

$\nabla_w \log \pi_w(u|x)$: score function

Examples

- Gibbs policy:

$$\nabla_w \log \pi_w(u|x) = \phi(x,u) - E_{\pi_w}[\phi(x,\cdot)]$$

- Gaussian policy:

$$\nabla_w \log \pi_w(u|x) = \frac{(u-\mu(x))\phi(x)}{\sigma^2}$$

where $\mu(x) = \phi^T(x)w$

Policy-Gradient Theorem

- Recall $\pi_w(u|x)$ denotes a randomized (or deterministic) policy with parameter w,

$$\pi_w(u|x) = \mathbb{P}(\text{action} = u|\text{state} = x)$$

- Recall that

$$V_w(x_0) = E_{u_k \sim \pi_w(u_k|x_k)} \left[\sum_{k=0}^{\infty} \alpha^k r(x_k, u_k) \right].$$

Policy-Gradient Theorem

Proof

Assume the initial state $x_0 = i$.

$$V_w(i) = \sum_u Q_w(i,u)\pi_w(u|i)$$

$$\nabla V_w(i) = \sum_u \left(\nabla_w Q_w(i,u)\pi_w(u|i) + Q_w(i,u)\nabla_w \pi_w(u|i)\right).$$

Policy-Gradient Theorem

$$\begin{aligned}\sum_u \nabla_w Q_w(i,u)\pi_w(u|i) &= \sum_u \nabla_w \left(E[r(i,u)] + \alpha \sum_j P_{ij}(u) V_w(j) \right) \pi_w(u|i) \\ &= \alpha \sum_u \left(\sum_j P_{ij}(u) \nabla_w V_w(j) \right) \pi_w(u|i) \\ &= \alpha E_{x_0=i, u_0 \sim \pi_w(u|x_0)} \left[\nabla_w V_w(x_1) \right]\end{aligned}$$

$$\begin{aligned}\sum_u Q_w(i,u) \nabla_w \pi_w(u|i) &= \sum_u Q_w(i,u) \left(\nabla_w \log \pi_w(u|i) \right) \pi_w(u|i) \\ &= E_{x_0=i, u_0 \sim \pi_w(u|x_0)} \left[Q_w(i,u) \nabla_w \log \pi_w(u|i) \right]\end{aligned}$$

Policy-Gradient Theorem

Proof

$$\begin{aligned}\nabla V_w(i) &= \sum_u \left(\nabla_w Q_w(i,u)\pi_w(u|i) + Q_w(i,u)\nabla_w \pi_w(u|i)\right) \\ &= E_{x_0=i,u_0\sim\pi_w(u|x_0)}\left[Q_w(i,u)\nabla_w \log \pi_w(u|i)\right] + \alpha E_{x_0=i,u_0\sim\pi_w(u|x_0)}\left[\nabla_w V_w(x_1)\right] \\ &= E_{x_0=i,u_k\sim\pi_w(u|x_k)}\left[\sum_{k=0}^{\infty} \alpha^k \nabla_w \log \pi_w(u_k|x_k) Q_w(x_k,u_k)\right]\end{aligned}$$

REINFORCE

REINFORCE uses the Monte-Carlo method to estimate the Q-function

$$Q_w(x_k,u_k) \approx \sum_{\tau=k}^{T-1} \alpha^{\tau-k} r(x_\tau,u_\tau),$$

and uses a single episode as the estimate.

REINFORCE (Williams (1988, 1992))

Generate an episode $x_0, u_0, x_1, u_1, \cdots, x_{T-1}, u_{T-1}, x_T$ by following policy π_w. After the episode is generated, it takes T steps to update w. For each step k of the episode, update w as follows:

$$w \leftarrow w + \beta \alpha^k \nabla_w \log \pi_w(u_k|x_k) \sum_{\tau=k}^{T-1} \alpha^{\tau-k} r(x_\tau, u_\tau).$$

Continue to the next episode with the updated w.

Policy-Gradient Theorem (Continued)

$$
\begin{aligned}
\nabla V_w(i) &= E_{x_0=i,u_k\sim\pi_w(u|x_k)}\left[\sum_{k=0}^{\infty}\alpha^k\nabla_w\log\pi_w(u_k|x_k)Q_w(x_k,u_k)\right] \\
&= \sum_{k=0}^{\infty}\sum_{x,u}\alpha^k\nabla_w\log\pi_w(u|x)Q_w(x,u)P(x_k=x,u_k=u|x_0=i) \\
&= \sum_{x,u}\nabla_w\log\pi_w(u|x)Q_w(x,u)\sum_{k=0}^{\infty}\alpha^k P(x_k=x,u_k=u|x_0=i) \\
&= \sum_{x,u}\nabla_w\log\pi_w(u|x)Q_w(x,u)\sum_{k=0}^{\infty}\alpha^k P(x_k=x|x_0=i)\pi_w(u|x) \\
&= \frac{1}{1-\alpha}\sum_{u,x}\nabla_w\log\pi_w(u|x)Q^{\pi_w}(x,u)\left((1-\alpha)\sum_{k=0}^{\infty}\alpha^k P\left(x_k=x|x_0=i\right)\right)\pi_w(u|x) \\
&= \frac{1}{1-\alpha}\mathbb{E}_{x\sim\rho_w,u\sim\pi_w(u|x)}\left[\nabla_w\log\pi_w(u|x)Q_w(x,u)\right].
\end{aligned}
$$

References

- Chapter 4.4 of Csaba Szepevari, *Algorithms for Reinforcement Learning,* Morgan Claypool, 2010.
- Ronald J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. Machine Learning, 8(3):229-256, 1992.