Deep Deterministic Policy Gradient

Anirudh Goyal, Rosemary Nan Ke

Continuous actions ?

- Most interesting problems fall in this category.
- If you discretize your action space naively,
  - Curse of dimensionality problem as before.
  - Throws away valuable information regarding the geometry of action domain

Policy gradient

- Instead of estimating the value or action-value function. Directly parameterize the policy
- Used for continuous action spaces
- Stochastic policy $\boldsymbol{\mu}$(a|s) - probability over actions
- Seems many examples with high reward for good actions, and many examples with negative reward for bad actions.

Policy gradient

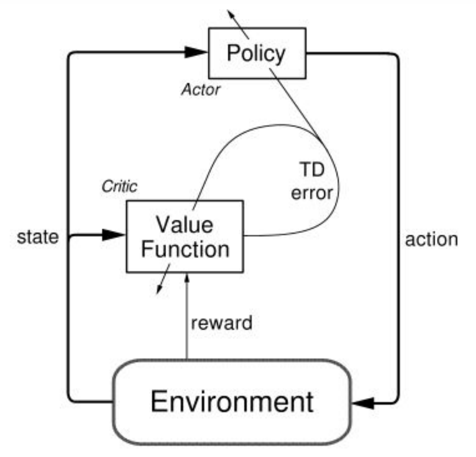
- Optimize the policy end-to-end by computing noisy estimates of the gradient of the expected reward of the policy and then updating the policy in the gradient direction.
- Traditionally, PG methods have assumed a stochastic policy.

Policy gradient

- Issues:
  - Getting one reward signal at the end of the episode, of interaction makes it impossible to ascertain which action was good (Credit assignment problem)
  - Curse of dimensionality

Actor Critic Algorithms

- Policy function - Actor ,Value function - critic
- Actor: Input state s, output action a
- Critic: Input state s, reward r, output TD error difference



Off-Policy v/s On-Policy

Off-Policy

- Generally employ a separate behaviour policy that is independent of policy being improved upon, behaviour policy is used to simulate trajectories.
- **Key benefit - Behaviour policy can operate by sampling all actions, whereas estimation policy can be greedy (i.e deterministic)**
- Q policy is an off policy algorithm
- On policy algorithms directly use the policy that is being estimated to sample trajectories during training.

Model free algorithms

- No effort to learn the underlying dynamics that govern how an agent interacts with environment.
- In case, environment -> discrete state space, agent -> discrete number of actions, a model of the dynamics of the environment is 1 step transition operator. $T(s(t+1) \mid s(t), a(t))$
- For high dimensional state space, this is extremely expensive!!
- Hence, Model free algorithm directly estimate the policy or value function, (more computationally efficient)
- Be wary, Using a bad approximation of environment would bring you misery.

Deep Deterministic Policy Gradient

- Uses a stochastic behaviour policy for good exploration.
- Estimates a deterministic target policy.
- Utilize a form of policy iteration: they evaluate the policy and then follow policy gradient.
- Actor critic algorithm as well.
- In continuous space, finding the greedy policy requires an optimization of a_t at every time sleep. Use Actor critic based on DPG algorithm

Theorem. *(Deterministic Policy Gradient Theorem). Suppose the Markov Decision Process satisfies the appropriate conditions (see [3]). These imply that* $\nabla_{\theta^\mu}\mu(s|\theta^\mu)$ *and* $\nabla_a Q(s,a|\theta^Q)$ *exist and that the deterministic policy gradient exists. Then,*

$$\begin{aligned}\nabla_{\theta^\mu}\mu &\approx \int_S \rho^{\mu'}(s_t)\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_t}ds \\ &= \mathbb{E}_{\mu'}\left[\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_t}\right]\end{aligned}$$

Proof. For a greedy stochastic policy $\mu(a|s,\theta)$ over a continuous action space, a global maximization step is required at every time step. Rather, we employ a deterministic policy $\mu(s|\theta)$ and update the policy parameters by moving them in the direction o the gradient of the action-value function. We take an expectation to average over the suggested directions of improvement from each state w.r.t. the state distribution under the target policy μ', given by $\rho^{\mu'}(s)$.

$$\theta^\mu_{k+1} = \theta^\mu_k + \alpha\mathbb{E}_{\mu'^k}\left[\nabla_\theta Q(s,\mu(s|\theta^\mu_k)|\theta^Q_k)\right].$$

By applying the chain rule,

$$\theta^\mu_{k+1} = \theta^\mu_k + \alpha\mathbb{E}_{\mu'^k}\left[\nabla_a Q(s,a|\theta^Q_k)|_{a=\mu(s|\theta^\mu_k)}\nabla_\theta\mu(s|\theta^\mu_k)\right].$$

DDPG

- Uses 2 networks
  - Actor - Input is current state, output is single real value representing the action chosen from a continuous action space.
  - Critic's output is simply the estimated Q value of the state and action given by actor (policy)
  - Deterministic policy gradient theory provides the update rule for the weights of the actor network.
  - Critic network is updated from the gradients obtained from the TD error signal.

Key Conspirators for divergence

- **Problem 1 -** Training your policy for 1000's of temporally correlated simulated trajectories leads to the introduction of enormous amounts of variance in your approximation of true Q function. TD error signal is excellent in compounding the variance by bad predictions over time.
- **Problem - 2 -** Directly updating the actor and critic weights with gradients obtained from TD error, can cause your learning algorithm to diverge.

Solutions - Lessons from Success of DQN

- NN for RL Assume, that samples are iid, but where the samples are generated from exploring sequentially in an environment, this assumption not holds.
- AS DQN, we use replay buffers, a technique called experience reply to address this issue.
- Using the set of target networks to generate the targets for your TD error computation regularizes your learning algorithm and increases stability.
- Targets network slowly change that greatly improve the stability of the training.

Challenge- 2

- Advantage of off policy algorithm is that you can treat the exploration problem independent of learning algorithm
- Construct an exploration policy by adding noise sampled from a noisy process.
- Use an ornstein-uhlenbeck process to generate temporally correlated exploration for exploration efficiency with inertia.

References

- Continuous control with deep reinforcement learning
- Deterministic Policy Gradient Algorithms
- Prioritized experience replay https://arxiv.org/abs/1511.05952