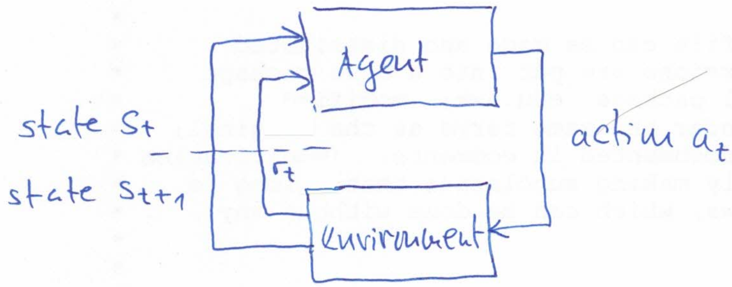
The shortest path to PPO

Reinforcement Learning.

An agent interacts with an environment to optimize an objective:



A trajectory is a sequence of experiences over an episode:

$$\tau = (s_0, a_0, r_0), (s_1, a_1, r_1), \dots$$

The environment provides the state transitions and the rewards. The state transition function has the Markov property and will be denoted by $p(s_{t+1}|s_t, a_t)$. The reward function will be denoted by $R(s_t, a_t, s_{t+1})$. The agent does not a priori know these two functions.

The agent has to learn a good policy function $\pi(s)$ which describes the probabilities for selecting available actions at state s:

$$\pi(a|s)$$

The objective of the agent is to find the policy which maximizes the expected discounted reward for trajectories generated according to this optimal policy

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right],$$

where $0 < \gamma \leq 1$ is the discounting factor.

The REINFORCE algorithm

The policy function $\pi_\theta(a|s)$ is parametrized by a NN with parameters θ. The objective is

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

The shortest path to PPO

To maximize the objective we perform gradient ascent on the policy parameters θ:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\pi_\theta).$$

The policy gradient can be rewritten by using the discounted sum of rewards from time t to the end of the trajectory

$$R_t(\tau) = \sum_{s=t}^{T} \gamma^{s-t} r_s$$

After a bit of algebra it could be shown that

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

The numerical estimate of the policy gradient is achieved with MC sampling:

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T} R_t(\tau_i) \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

Unfortunately this MC estimate has very-high variance.

Advantage Actor Critic (A2C) algorithm

In A2C a policy is reinforced with a signal from a learned value function. Here

Actor – learns a parametrized policy

Critic – learns a value function to evaluate state-action pairs

Some value functions are:

$$V^\pi(s) = \mathbb{E}_{s_0 = s, \tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \qquad \text{(state value)}$$

$$Q^\pi(s,a) = \mathbb{E}_{s_0=s, a_0=a, \tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r_t\right] \qquad \text{(State-action pair value)}$$

In A2C the critic learns the advantage function

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

The advantage function quantifies how much better or worse an action is than the average available action in state s.

The shortest path to PPO

Two expressions are typically used to estimate the advantage function. The n-step method uses the expression

$$A^{\pi}_{NSTEP}(s_t, a_t) \approx r_t + \gamma r_{t+1} + \ldots + \gamma^n r_{t+n} + \gamma^{n+1} \hat{V}^{\pi}(s_{t+n+1}) - \hat{V}^{\pi}(s_t)$$

The Generalized Advantage Estimation uses

$$A^{\pi}_{GAE}(s_t, a_t) \approx \sum_{\ell=0}^{\infty} (\gamma\lambda)^{\ell} \delta_{t+\ell}, \text{ where } \delta_t = r_t + \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t)$$

GAE is an exponentially weighted average of forward return advantages. The decay rate is controlled by the coefficient λ.

Both these methods assume that we have access to an estimate $\hat{V}^{\pi}$ for the state value function V^{π}. We parametrize $\hat{V}^{\pi}$ with ϕ and generate a target V^{π}_{tar} from the experiences an agent gathers. An n-step estimate is given by

$$V^{\pi}_{tar}(s_t) = r_t + \gamma r_{t+1} + \ldots + \gamma^n r_{t+n} + \gamma^{n+1} \hat{V}^{\pi}(s_{t+n+1}).$$

Alternatively we can use a MC estimate for V^{π}_{tar}

$$V^{\pi}_{tar}(s_t) = \sum_{s=t}^{T} \gamma^{s-t} r_s$$

To minimize the difference between $\hat{V}^{\pi}(s;\phi)$ and $V^{\pi}_{tar}(s)$ we use a regression loss such as MSE.

The gradient update for the Actor (policy) network is via

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_t \left[A^{\pi}_t(s_t, a_t) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right]$$

If the advantage $A^{\pi}_t(s_t, a_t) > 0$, then the probability $\pi_{\theta}(a_t|s_t)$ for the action is increased and if the advantage is negative the probability for the action a_t is decreased.

Proximal Policy Optimization (PPO)

A2C is susceptable to performance collapse: the agent starts generating poor trajectories which are then used to further train the policy. The surrogate objective of PPO avoids performance collapse by guaranteeing monotonic policy improvement.

Let a policy π be given and let π' be its next iteration after a parameter update. Notice that

$$\max_{\pi'} J(\pi') \iff \max_{\pi'} \big(J(\pi') - J(\pi) \big)$$

It could be shown that

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau \sim \pi'} \Big[\sum_{t=0}^{T} \gamma^t A^{\pi}(s_t, a_t) \Big],$$

where again

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \Big[\sum_{t=0}^{T} \gamma^t r_t \Big].$$

However in the equation for $J(\pi') - J(\pi)$ we have an expectation with respect to the updated policy π' which will be hard to work with. This difference could be approximated by using trajectories sampled from the old policy and adjusted with importance sampling weights:

$$J(\pi') - J(\pi) \approx \mathbb{E}_{\tau \sim \pi} \Big[\sum_{t=0}^{T} A^{\pi}(s_t, a_t) \frac{\pi'(a_t \mid s_t)}{\pi(a_t \mid s_t)} \Big] = J^{CPI}_{\pi}(\pi')$$

Here CPI stands for conservative policy iteration. Remarkably it could be shown that

$$\nabla_\theta J^{CPI}_{\theta_{old}}(\theta) \Big|_{\theta_{old}} = \nabla_\theta J(\pi_\theta) \Big|_{\theta_{old}}$$

so an optimization of the surrogate objective $J^{CPI}_{\pi}(\pi')$ uses the same gradient ascent updates as the objective $J(\pi')$.

To guarantee that $J(\pi') - J(\pi) \geq 0$ we will use the inequality

$$J(\pi') - J(\pi) \geq J^{CPI}_{\pi}(\pi') - C \Big\{ \mathbb{E}_t \big[KL(\pi'(a_t \mid s_t) \,\|\, \pi(a_t \mid s_t)) \big] \Big\}^{1/2}$$

The shortest path to PPO

In order to accept a change in a policy the estimated policy improvement $J^{CPI}_{\pi}(\pi')$ has to be greater than the maximum error $C\{\mathbb{E}_t[KL(\pi'(a_t|s_t))\|\pi(a_t|s_t))\}^{1/2}$

To implement this requirement in practice the simplest approach is to restrict how far the new policy can diverge from the old policy by imposing a trust region constraint

$$\mathbb{E}_t[KL(\pi'(a_t|s_t))\|\pi(a_t|s_t))] \leq \delta$$

The first version of PPO called PPO with adaptive KL penalty turns this constraint into an adaptive KL penalty. The objective to be maximized is

$$J^{KLPEN}(\theta) = \mathbb{E}_t[r_t(\theta)A_t - \beta KL(\pi_\theta(a_t|s_t)\|\pi_{\theta_{old}}(a_t|s_t))]$$

where
$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

Notice the expectation in the objective is over a single time step. β is an adaptive coefficient which controls the size of the KL penalty

The second version of PPO is PPO with clipped surrogate objective. The objective is

$$J^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t]$$

ε is the value which defines the clipping neighborhood $[r_t(\theta)-1] \leq \varepsilon$. When $r_t(\theta) \in [1-\varepsilon, 1+\varepsilon]$ both terms in the $\min(\cdot)$ are equal. This objective prevents parameter updates which could case large and risky updates to the policy.

Finally, entropy regularization could be added to encourage exploration through diverse actions

$$J^{CLIP}(\theta) \longrightarrow J^{CLIP}(\theta) - \beta H$$

The shortest path to PPO

PPO with clipping

Set:
1. $\beta \geq 0$, entropy regularization weight
2. $\varepsilon \geq 0$, the clipping variable
3. k, the number of epochs
4. N, the number of actors
5. M, the batch size
6. $\alpha_A \geq 0$, $\alpha_C \geq 0$ the Actor and Critic learning rates

Randomly initialize the actor and critic parameters θ_A, θ_C.

for $i = 1, 2, \dots$ do
  Set $\theta_{A\,old} = \theta_A$
  for actor $= 1, \dots, N$ do
    Run policy $\theta_{A\,old}$ for T time steps and collect the trajectories.
    Compute the advantages $A_1, \dots, A_T$ using θ_{old}.
    Calculate $V^{\pi}_{tar,1}, \dots, V^{\pi}_{tar,T}$ using the trajectory data
  end for
  Let batch with size NT consist of the collected trajectories, advantages, V^{π}_{tar}
  for epoch $= 1, \dots, k$ do
    for minibatch m in batch do
      calculate $r_m(\theta_A)$, $J_m^{CLIP}(\theta_A)$, H_m
      Calculate the policy objective $L_{POL}(\theta_A) = J_m^{CLIP}(\theta_A) - \beta H_m$
      Calculate value loss $L_{VAL}(\theta_C) = MSE(\hat{V}^{\pi}(s_m), V^{\pi}_{tar}(s_m))$
      Update the Actor parameters
        $\theta_A \to \theta_A + \alpha_A \nabla_{\theta_A} L_{POL}(\theta_A)$
      Update the Critic parameters
        $\theta_C \to \theta_C + \alpha_C \nabla_{\theta_C} L_{VAL}(\theta_A)$
    end for
  end for
end for