Implementation of Advantage Weighted Actor Critic Algorithm

Naga Sai Nikhita Sattiraju
nagasainikhita@vt.edu

ABSTRACT

This project aims at implementing an advanced algorithm called AWAC, Advantage weighted Actor Critic algorithm , which is discussed in [3] This project will attempt to understand the advantages and explore disadvantages (if any) of this algorithm. The main goal of this algorithm is to accelerate online reinforcement learning using offline datasets, which makes it a very useful tool for using reinforcement learning more efficient. But this task is also an extremely complicated and difficult task. The approach discussed in the paper aims to navigate through this task by efficiently handling its challenges with accumulation of error while bootstrapping, stemming from data inefficiency and excessive conservative on line learning.

Author Keywords

Authors' choice; of terms; separated; by semicolons; include commas, within terms only; required.

INTRODUCTION

Reinforcement Learning collects new dataset for every policy learned, and requires large and diverse dataset to be efficient in real world applications. If instead, reinforcement learning were able to have its models pre-trained on these already existing datasets collected from other models, then it would become easy to navigate through a online environment. These offline datasets might be consisting of different trajectories, states, actions , rewards associated, and all this data can help in extract useful information for our RL model before heading out in the environment and efficiency of our reinforcement learning model will increase manifold.

This task is not an easy one to accomplish, for it is riddled with challenges all along. The challenges are ,

1. **Data optimality:** Previous methods attempt to pre train models using RL demonstrations by using imitation learning, while it is a simplest approach, it is not most efficient because of its dependency on the assumption that data obtained is optimal, and disregards online fine tuning during RL stage.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. **Bootstrap Error in Offline Learning with Actor-Critic Methods** In off policy actor critic algorithms , the target action(i.e in online environment) is drawn from a policy obtained in the offline dataset, this would lead to inaccurate target action value function $Q(s',a')$ and accumulation of error.
3. **Excessively Conservative Online Learning:** It is highly tricky and difficult process when offline RL algorithms when attempts to fit an accurate behavioural model to online incoming data. When the behavior models are inaccurate or unable to model new data well, constrained optimization becomes too conservative,resulting in limited improvement with fine-tuning

AWAC: ADVANTAGE WEIGHTED ACTOR CRITIC ALGORITHM

Problem setup :

This algorithm intends to use offline data and fine tune it during online learning building on 'Actor-Critic' algorithm. To achieve this implementation, two actors are used, each having their own significance in the following phases .
The two phases involved in this algorithm are:

- Offline phase
- Online phase

To achieve this implementation, two actors are used, which are indicative of (π_β,π_θ) in the algorithm

Offline phase

During this phase, AWAC algorithm collects sample data from an offline dataset and trains its critic(policy evaluator, Q_ϕ) and actor(policy improvement), which here is π_β for a said number of steps.

π_β also updates its second actor π_θ as shown in the algorithm, which is during this phase.

Online phase

During online phase, π_θ, which was updated during offline phase, is used to learn new trajectories, which are again merged into the buffer.

The policy learnt during offline phase(π_θ) cautiously adapts to online data by choosing new actions closer to the policy learnt during offline phase, which here is π_β.

Thus, this algorithm provides a unique way to adapt to the online environment using offline data for its training.

BACKGROUND

Considering a standard reinforcement learning notation, with states s and action a, policy $\pi(a|s)$, rewards $r(s,a)$ and dynamics $p(s'|s,a)$. The discounted return is defined as $R_t = \sum_{i=t}^{T} \gamma^i r(s_i, a_i)$ for a discount factor γ and horizon T which might be infinite. Value function, $v_\pi(s) = \mathbb{E}_p \pi(T)[R_t|s]$
The following bellman equation can also be written using the bellman operator on the right side of the above equation.

$$B^\pi Q(s,a) = r(s,a) + \gamma_\pi(s'|s,a)[_p(a',s')[Q^\pi(s',a')]] \quad (1)$$

We know that by iterating $Q^{k+1} = B^\pi Q^k$, Q^k converges to Q^π

During Policy evaluation, Critic $Q^\pi(s,a)$, state action value function, is estimated for current policy, the actor, π. Since we would use function approximation , critic update can be written as minimizing the bellman error with respect to Q function parameters ϕ_k:

$$\phi_k = \underset{\phi}{argmin} \mathbb{E}_D[(Q^\pi(s,a) - y)^2],\ y = r(s,a) + \gamma \mathbb{E}_{s',a'}[Q_{\phi_{k-1}}(s',a')] \quad (2)$$

Since from the above equation and the definition of advantage (below), we can say that updating $Q^\pi(s,a)$ is same as updating $A^{\pi_k}(s,a)$.

Here $A^{\pi_k}(s,a)$ is known as advantage, the following expression help us understand more about it. $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ where $Q^\pi(s,a)$ is the state value function of that state, meaning, it evaluates the value of the action taken in that state for a given policy, and where as $V^\pi(s)$ is the over all value of a state under the policy π.

Policy improvement step however is the most defining component of this algorithm. While training π_β during offline phase, it also updates π_θ using

$$\pi_{k+1} = \underset{\pi \in \Pi}{argmax}\, \mathbb{E}_{a\sim\pi(\cdot|s)}[A^{\pi_k}(s,a)]\ \ s.t\ D_{KL}(\pi(\cdot|s)||(\pi_\beta(\cdot|s)) \leq \varepsilon \quad (3)$$

Having advantage weighted critic helps us get closer to the desired optimal policy. D_{KL} in the above equation is Kullback Liebler divergence used to keep the policy closer to the policy observed before.

Given the previous off policy advantage estimate, A^{pi_k} and buffer distribution π_β. It is shown that we can determine a analytic solution to the previous expression for actor while enforcing the constraint implicitly rather than explicitly learning a behavioral model[1]

$$\pi_{k+1} = arg\ \underset{\pi\in\Pi}{max}\ \mathbb{E}_{a\sim\pi(.|s)}[A^{\pi_k}(s,a)]$$

$$\text{s.t} \quad D_{KL}(\pi(\cdot|s)||\pi_\beta(\cdot|s)) \leq \varepsilon$$

$$\int_a \pi(a|s)da = 1$$

Enforcing the KKT conditions we get the above equations in this form.

$$\mathscr{L}(\pi,\lambda,\alpha) = \mathbb{E}_{a\sim\pi(.|s)}[A^{\pi_k}(s,a)] + \lambda(\varepsilon - D_{KL}(\pi(\cdot|s)||\pi_\beta(\cdot|s))) + \alpha\left(1 - \int_a \pi(a|s)da\right).$$

Differentiating with respect to π

$$\frac{\partial\mathscr{L}}{\partial\pi} = A^{\pi_k}(s,a) - \lambda\log\pi_\beta(a|s) + \lambda\log\pi(a|s) + \lambda - \alpha$$

Setting $\frac{\partial\mathscr{L}}{\partial\pi}$ to zero and solving for π gives the closed form solution to this problem:

$$\pi^*(a|s) = \frac{1}{Z(s)}\pi_\beta(a|s)\ exp\left(\frac{1}{\lambda}A^{\pi_k}(s,a)\right)$$

where $Z(s)$ is the normalizing partition function

$$Z(s) = \int_a \pi_\beta(a|s) exp\left(\frac{1}{\lambda}A^{\pi_k}(s,a)da\right)$$

projecting the solution into space of parametric policies, we get

$$arg\ \underset{\theta}{min}\ \underset{\rho_{\pi_\beta}(s)}{\mathbb{E}}[D_{KL}(\pi^*(\cdot|s)||\pi^\theta(\cdot|s))] =$$

$$arg\ \underset{\theta}{min}\ \underset{\rho_{\pi_\beta}(s)}{\mathbb{E}}\left[\underset{\pi^*(\cdot|s)}{\mathbb{E}}[-\log\ \pi_\theta(\cdot|s)]\right]$$

policy could be projected in any direction, choosing the forward direction has unique advantage. It allows us to optimize θ as maximum likelihood problem with an expectation over data $s,a \sim \beta$. Rather than sampling actions from the policy, that may be out of distribution for the Q function.

Hence the policy update is

$$\theta_{k+1} = \underset{\theta}{argmax}\ \underset{s,a\sim\beta}{\mathbb{E}}\left[log\pi_\theta(a|s)\frac{1}{z(s)}exp\left(\frac{1}{\lambda}A^{\pi_k}(s,a)\right)\right] \quad (4)$$

where $Z(s) = \int_a \pi_\beta(a|s)exp(\frac{1}{\lambda}A^{\pi_k}(s,a))$ is the normalizing partition function

This way AWAC enforces constraint by incorporating implicit learned behavior in a non parametric way. This aids in actor update, using supervised learning, where the targets are learned by re weighting the state action pairs observed in the online dataset by the predicted advantages from the learned critic, without explicitly learning any parametric behavioural model thereby avoiding excessive conservative online learning drawback to arise. the following equation is an expression for actor update.

ALGORITHM

```
Dataset D=(s,a,s',r)$_j$
Initialize buffer $\beta = D$
Initialize $\pi_\theta, Q_\phi$
for iteration i=1,2,.. do
    Sample batch $(s,a,s',r) \sim \beta$;
    Calculate y;
    $y = r(s,a) + \gamma \mathbb{E}_{s',a'}[Q_{\phi_{i-1}}(s',a')]$;
    Update $\phi$ using
    $\phi_i = \underset{\phi}{arg\ min}\ \mathbb{E}_D[(Q_\phi(s,a) - y)^2]$
    Update $\theta$ using
    $\theta_{k+1} =$
      $\underset{\theta}{arg\ max} \underset{s,a\sim\beta}{\mathbb{E}}\left[\log \pi_\theta(a|s) \frac{1}{Z(s)} exp\left(\frac{1}{\lambda} A^{\pi_k}(s,a)\right)\right]$;
    if $i >$ number of offline steps then
        $\tau_1, \dots \tau_K \sim \rho_{\pi_\theta}(\tau)$;
        $\beta \leftarrow \beta \cup \{\tau_1, \dots \tau_K\}$
    end
end
```

Algorithm 1: Advantage Weighted Actor-critic

ALGORITHMIC IMPLEMENTATION

- **Step 1** Implemented Actor critic algorithm using Open AI's discrete environment called 'LunarLander-v2' to create offline data using function approximation.

- **Step 2** Offline data collected in the above step is used to implement the aforementioned algorithm 1 by following a similar Actor-critic flavor differing only in policy improvement using the same environment called 'LunarLander-v2

- **Step 3** Main method

1	number of total episodes	5000
2	number of offline steps	3000
3	gamma	0.99
4	alpha(actors)	5e-6
5	beta (critic)	5e-5
6	lambda	5e-6
7	sample size	100

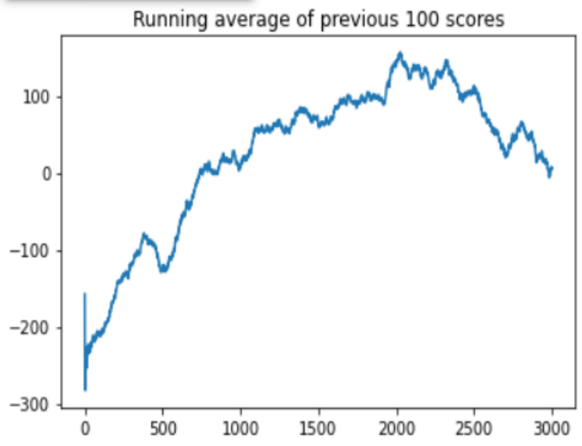
Hyper parameters

- **step 4** parameters for our agent

1	fully connected layer 1 size	2048
2	fully connected layer 2 size	1536
3	input dimensions	8
4	number of actions	4

Agent's parameters

RESULTS



During the offline learning,it took aproximately 1000 episodes for the algorithm to optimize, however after 2000 steps we can see that the performance dipped again maybe due to the 'catastrohic-forgetting' phenomenon with off policy lgorithms. It is the tendency of an artificial neural network to completely and abruptly forget previously learned information upon learning new information[2]

CONCLUSION

This project attempted to completely understand the derivation of solution to the equations for critic update and actor update, and implement this approach. However due to data corruption, further results could not be published.

ACKNOWLEDGMENTS

References

[1] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *arXiv preprint arXiv:1801.01290* (2018).

[2] MichaelMcCloskey and Neal J.Cohen. *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*. https://www.sciencedirect.com/science/article/pii/S0079742108605368?via%3Dihub. 1989.

[3] Ashvin Nair et al. *Accelerating Online Reinforcement Learning with Offline Datasets*. 2020. arXiv: 2006.09359 [cs.LG].