The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games

Paper

- https://arxiv.org/pdf/2103.01955.pdf

Introduction

- Other multiagent algorithms: MADDPG, value decomposed q learning
- Revisits Proximal Policy Optimization (PPO) for multi agent settings

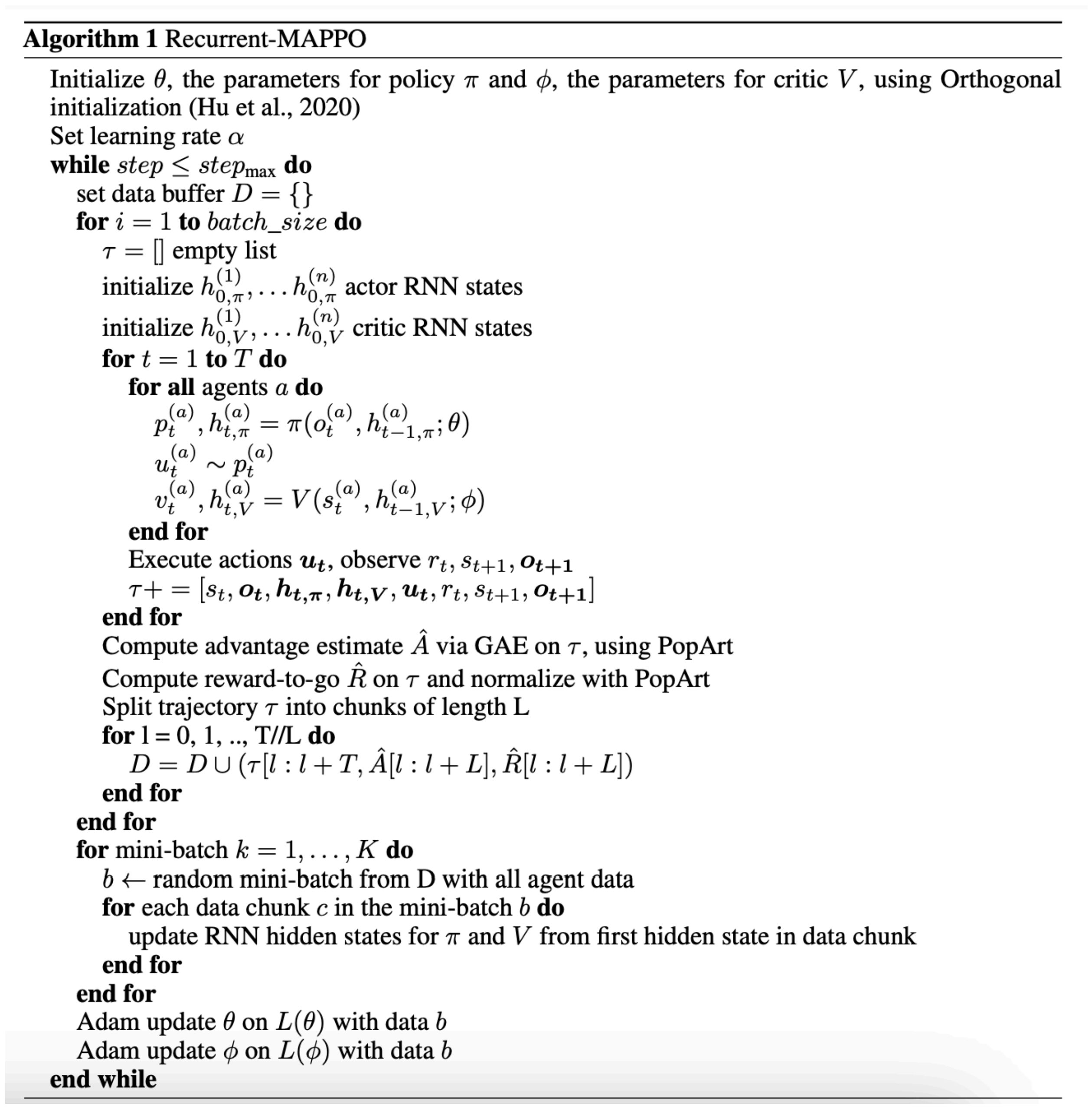
Related work

- MARL algorithms
    - Centralized learning - directly learn single policy to produce joint actions of all agents
    - Decentralized learning - each agent optimizes its reward independently
    - Centralized Training and Decentralized execution - in between
        - Value decomposition methods

Preliminaries

- Decentralized partially observable Markov decision processes (DEC-POMDP) with shared rewards
- <S, A, O<, R, P, n, γ>
    - S: state stpace
    - A: shared action space for each agent i
    - O_i = O(s;i): local observation for each agent i at global state s
    - P(s'|s , A)
    - A= (a1… an_
    - R(s,A)shared reward function
    - γ: discount factor r
    - Agents use policy

Mappo Details

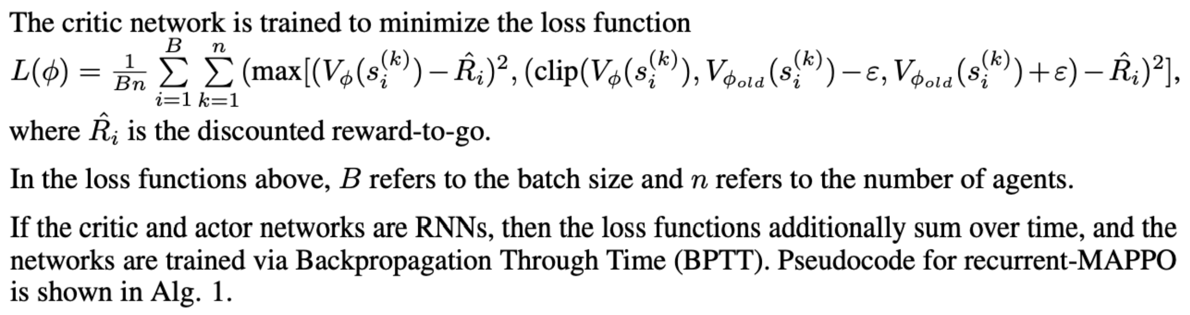


Algorithm 1 Recurrent-MAPPO

```
Initialize $\theta$, the parameters for policy $\pi$ and $\phi$, the parameters for critic $V$, using Orthogonal initialization (Hu et al., 2020)
Set learning rate $\alpha$
while $step \leq step_{\max}$ do
    set data buffer $D = \{\}$
    for $i = 1$ to $batch\_size$ do
        $\tau = []$ empty list
        initialize $h^{(1)}_{0,\pi}, \dots h^{(n)}_{0,\pi}$ actor RNN states
        initialize $h^{(1)}_{0,V}, \dots h^{(n)}_{0,V}$ critic RNN states
        for $t = 1$ to $T$ do
            for all agents $a$ do
                $p^{(a)}_t, h^{(a)}_{t,\pi} = \pi(o^{(a)}_t, h^{(a)}_{t-1,\pi}; \theta)$
                $u^{(a)}_t \sim p^{(a)}_t$
                $v^{(a)}_t, h^{(a)}_{t,V} = V(s^{(a)}_t, h^{(a)}_{t-1,V}; \phi)$
            end for
            Execute actions $\boldsymbol{u_t}$, observe $r_t, s_{t+1}, \boldsymbol{o_{t+1}}$
            $\tau += [s_t, \boldsymbol{o_t}, \boldsymbol{h_{t,\pi}}, \boldsymbol{h_{t,V}}, \boldsymbol{u_t}, r_t, s_{t+1}, \boldsymbol{o_{t+1}}]$
        end for
        Compute advantage estimate $\hat{A}$ via GAE on $\tau$, using PopArt
        Compute reward-to-go $\hat{R}$ on $\tau$ and normalize with PopArt
        Split trajectory $\tau$ into chunks of length L
        for l = 0, 1, .., T//L do
            $D = D \cup (\tau[l : l + T, \hat{A}[l : l + L], \hat{R}[l : l + L])$
        end for
    end for
    for mini-batch $k = 1, \dots, K$ do
        $b \leftarrow$ random mini-batch from D with all agent data
        for each data chunk $c$ in the mini-batch $b$ do
            update RNN hidden states for $\pi$ and $V$ from first hidden state in data chunk
        end for
    end for
    Adam update $\theta$ on $L(\theta)$ with data $b$
    Adam update $\phi$ on $L(\phi)$ with data $b$
end while
```

- Mappo trains 2 separate neural networks
    - Actor network with parameters θ
    - Value function network (critic) with parametsrs ϕ

  The actor network is trained to maximize the objective $L(\theta) = [\frac{1}{Bn} \sum_{i=1}^{B} \sum_{k=1}^{n} \min(r^{(k)}_{\theta,i} A^{(k)}_i, \text{clip}(r^{(k)}_{\theta,i}, 1 - \epsilon, 1 + \epsilon) A^{(k)}_i)] + \sigma \frac{1}{Bn} \sum_{i=1}^{B} \sum_{k=1}^{n} S[\pi_\theta(o^{(k)}_i))]$, where $r^{(k)}_{\theta,i} = \frac{\pi_\theta(a^{(k)}_i | o^{(k)}_i)}{\pi_{\theta_{old}}(a^{(k)}_i | o^{(k)}_i)}$. $A^{(k)}_i$ is computed using the GAE method, S is the policy entropy, and σ is the entropy coefficient hyperparameter.
-


The critic network is trained to minimize the loss function

$$L(\phi) = \frac{1}{Bn}\sum_{i=1}^{B}\sum_{k=1}^{n}(\max[(V_\phi(s_i^{(k)}) - \hat{R}_i)^2, (\text{clip}(V_\phi(s_i^{(k)}), V_{\phi_{old}}(s_i^{(k)}) - \varepsilon, V_{\phi_{old}}(s_i^{(k)}) + \varepsilon) - \hat{R}_i)^2],$$

where $\hat{R}_i$ is the discounted reward-to-go.

In the loss functions above, B refers to the batch size and n refers to the number of agents.

If the critic and actor networks are RNNs, then the loss functions additionally sum over time, and the networks are trained via Backpropagation Through Time (BPTT). Pseudocode for recurrent-MAPPO is shown in Alg. 1.

-
- Trained via backpropogation through time https://d2l.ai/chapter_recurrent-neural-networks/bptt.html
- Generalized Advantage Estimation
    -

Facors influential to ppo performance

- Value normalization
- Input representation to value function
    - “the fundamental difference between many multi-agent CTDE PG algorithms and fully decentralized PG methods is the input to the value network”
    - Options
        - Concatenation Of local observations
        - Environment provided global state
        - Agent specific global state
        - Feature pruned agent specific global state
- PPo clipping
    - PPO clipped importance ratio and value loss
    - Larger clipping values means larger updates to policy and value function ‘
    - Small clipping values → agent’s policy less likey to change per episdoe
- Ppo batch size
    - Shouldnt trian it for too long

Parameter sharing

- All agents share the same networks
- Recurrent data chunk length

Death masking

- When an agent dies these agent specific features become zero

https://arxiv.org/abs/2304.09870

Main takeaways