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

Background

Proximal policy optimization: on policy reinforcement learning algorithm

- Less utilized than off policy RL in multi agent settings
    - Belief that PPO is less sample efficient than off policy methods in multi agent systems -> less useful in resource- constrained settings
    - Common implementation and hyperparameter tuning practices when using PPO in single agent settings do not yield strong performance when transferred to multi agent settings

On policy RL such as IMPALA and PPO + distributed training

- DeepMind's AlphaStar surpassed professional level performance in StarCraft 2
- OpenAI five defeated world champion in Dota 2
- OpenAI demonstrated emergence of human like tool use agent behaviors via multi agent learning

Recent literature in multi agent reinforcement learning (MARL) focused on off- policy learning such as MADDPG and value decomposed q learning

Centralized methods: directly learn single policy to produce joint actions of all agents
Decentralized learning: each agent optimizes its reward independently, tackle general- sum games but suffer from instability even in simple matrix games

Centralized training and decentralized execution is between the above

- Adopt Actor critic structures and learn centralized critic which takes global information as input
- Value decomposition represent joint Q function as a function of agents' local q functions

Advances in off policy methods such as SAC led to consensus that policy gradient methods (PPO) are less sample efficient

Idea

PPO in cooperative multi agent settings does work: particle world environment, StarCraft, Google Research Football, Hanabi (minimal tuning or algorithm modifications)

- Competitive or superior results in final returns and sample efficiency
- Implementation and hyperparameter factors that are critical for performance (5) + suggestions how to do it + intuition why

Decentralized partially observable Markov decisions process (DEC- POMDP) with shared rewards

Policy and value function as two separate networks (CTDE)

- Value function used for variance reduction and only utilized during training -> take as input extra global information not present in agent's local observation

- PPO with centralized value function (MAPPO)
- Local inputs for both policy and value as IPPO

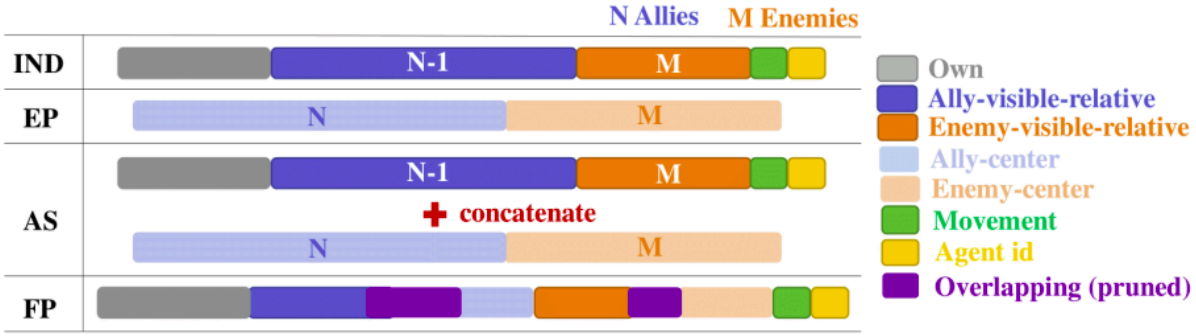
Parameter sharing- agents share both policy and value function parameters (improves efficiency of learning)

- Useful in benchmark environments with homogenous agents

Common implementation practices

- Generalized advantage estimation: advantage normalization and value clipping

1. Value normalization: standardize targets of value function by running estimates of average and standard deviation of value targets
    a. During value learning, value network regresses to normalized target values
    b. For GAE, use running average to denormalize output of value network so value outputs are properly scaled
2. Value function inputs: observing full global state can make value learning easier, accurate value function further improves policy learning through variance reduction
    a. Concatenation of local observations: concatenating all local agent observations (grows with number of agents and omits important global information that is unobserved)
    b. Environment provided global state: contain general global information but omits important local agent- specific information
    c. SMAC: combines concatenation of local observations and environment provided global state (overlap in information)
    d. Feature pruned agent specific global state which removes repeated features in SMAC



3. Training data usage: split training data into at most two mini batches and avoid mini batching in the majority of situations
    a. In multi agent, performance degrades when samples are re-used too often
    b. Using fewer epochs per update limits change in agents' policies improving stability of policy and value learning
4. Policy/ value clipping: clipped importance ratio and value loss to prevent policy and value functions from drastically changing between iterations

    a. Agents' policy change less per update improves learning stability at expense of learning speed
5. Batch size: critical point where less than this the performance is poor but more than this the sample efficiency is worse

Limitations

Discrete action spaces, homogenous agents, cooperative games