
Group Actor-Critic - A Hybrid Approach For Cooperative Multi-agent RL

Tram Tran

University of California, Berkeley

Abstract

This paper explores cooperative multi-agent reinforcement learning (MARL) with a team reward. Many real-world problems require a team of agents to cooperatively optimize a single joint reward. Although, in theory, many of these problems could be treated using a centralized approach by reducing them to single-agent reinforcement learning (RL) problems to find a centralized policy, it is usually impractical to do so. Cooperative MARL, which has achieved great progress in recent years, provides a promising approach to these problems. However, as MARL deals with multiple agents learning concurrently, it suffers from several challenges in addition to those that arise in single-agent RL. The standard paradigm to address MARL challenges is centralized critic with fully decentralized actors, in which the goal is to find decentralized individual agent policies. Following this paradigm, many MARL methods require a fully centralized critic Q_{total} that conditions on the global state and joint action. As a result, these methods become impractical when there are more than a handful of agents. Value-Decomposition Networks (VDN) [1] addresses this issue by decomposing Q_{total} as the sum of individual critic Q_{agent} . This linear decomposition enables scalable learning but induces a limitation on the representational capacity of the centralized critic Q_{total} . This paper presents a hybrid approach between fully centralized and fully decentralized approaches mentioned above. The idea is to split agents into groups and operate group-wise - Each group has a centralized policy conditions on concatenated observations of agents within the group. The benefit of operating as groups is that we can decompose Q_{total} as the sum of the groups' critic Q_{group} . Each group critic is a non-linear combination of within-group agents' critic Q_{agent} . Therefore, by decomposing Q_{total} into Q_{group} , we create a non linear decomposition of Q_{total} . As a result, Q_{total} could represent a richer class of value functions. For this idea to work, agents within a group have to be able to communicate with each other. To evaluate this idea, we use advantage actor-critic method and perform experiments on both partially and fully observable environments. The actor outputs actions for all agents within the group, and the critic is the estimation of the group value function V_{group} . Additionally, in partially observable environment, we use rnn actors and critics - each group's actor and critic have a lstm cell for memory. From our experiments, we found that our method converges faster than the fully decentralized approach and is comparable to centralized method in environments with homogeneous agents.

1 Introduction

Cooperative MARL is a promising approach to help address a variety of cooperative multi-agent problems such as coordination of self driving cars, coordination of robot swarms, or optimizing the productivity of a system composed of many interacting parts.

Although many of these problems could be treated in a centralized fashion by reducing them to single-agent reinforcement learning (RL) problems, single-agent RL methods require a central processing center which has some drawbacks such as lack of scalability and single point of failure (SPOF). Therefore, some local autonomy by learning decentralized policies which condition only on a subset of observations is necessary.

The simplest approach to learn decentralized policies is to use single-agent RL methods for each agent to optimize for the team reward as in *independent Q-learning* [2]. However, this method may not converge because agents can not distinguish other agents from the environment and so are faced with changing environment dynamics problem.

Another approach is to learn a centralized critic and use it to guide the learning of individual agent policies, an approach taken by *MADDPG*[3] and *COMA*[4]. However, learning a fully centralized critic is impractical if there are many agents. A way to improve the scalability of this approach is to learn a centralized but factorized critic as in *value decomposition networks* (VDN) [1]. VDN decomposes the centralized critic Q_{total} as the sum of individual action-value critic Q_{agent} . This linear decomposition structure enables scalable but severely limits the complexity of functions that Q_{total} can represent.

The approaches discussed so far either find a single fully centralized policy or multiple fully decentralized policies, a policy for each agent. In this paper, we present a hybrid approach that find policies between the two extremes - we split agents into groups and find a policy for each group. To find the group policies, we use multi-agent actor-critic framework and learn a centralized but factorized critic as in VDN. By learning decentralized policies at group level, many issues of both fully centralized approaches and fully decentralized approaches are addressed. The lack of scalability can be alleviated by increasing the number of groups. The SPOF problem in centralized approach can occur within each group, but we can create redundancy by having each agent keeps a copy of their group policy. The representational limitation from linear decomposition in VDN is alleviated because we decompose V_{total} as the sum of the groups' critics V_{group} . Each group value function is a non-linear combination of the individual agent value functions V_{agent} of agents within that group. Therefore, by decomposing V_{total} into V_{group} , we create a non linear decomposition of V_{total} and, in turns, increase the representation ability of V_{total} . However, note that agents within a group have to be able to communicate with each others for this approach to work.

2 Related Works

Independent Q-learning (IQL) [2] trains action-value functions for each agent independently using Q-learning. Inspired from IQL,[5] uses DQN to train the action-value function. Many methods use communication during execution such as CommNet [6], BicNet [7]. Some methods such as COMA [4], MADDPG [3] and [8] use centralized learning with fully decentralized paradigm. [9] and [10] explore individual reward shaping such that reward related more to agent observations.

3 Background

A fully cooperative multi-agent tasks can be modelled as a Dec-POMDP [(Oliehoek et al., 2016] consisting of $G = \langle I, S, A, P, R, Z, O, n, \gamma \rangle$. $s \in S$ is the true state of the environment. At each time step, each agent $i \in I$ receives an observation $o_i \in O$ according to observation function $Z(s, i)$ and selects an action $a_i \in A$, forming a joint action $\mathbf{a} \in \mathbf{A} \equiv A^n$. This causes a transition to a next state s' according to the transition function $P(s'|s, \mathbf{a})$ and a reward $r = R(s, \mathbf{a})$ shared by all agents. $\gamma \in [0, 1]$ is a discount factor.

VDN The main assumption in VDN is that the joint action-value function can be additively decomposed into value functions across agents

$$Q_{total}((h^1, h^2, \dots, h^d), (a^1, a^2, \dots, a^d)) \approx \sum_{i=1}^d \tilde{Q}_i(h^i, a^i)$$

where the \tilde{Q}_i depends on each agent's local observations. Note that \tilde{Q}_i is relative and by itself does not estimate an expected return. The loss function for VDN is

$$\mathcal{L}(\theta) = \sum_{i=1}^b (y_i - Q_{total}(\mathbf{h}, \mathbf{a}))^2$$

where $y_i = r + \gamma \max_{\mathbf{a}'} Q(\mathbf{h}', \mathbf{a}')$

Actor-Critic The actor, which is the policy, is trained using a gradient that depends on a critic, which usually is an approximation of the value function [11]. The actor's gradient is:

$$\nabla J(\theta) = E_{s_{0:T}, a_{0:T}} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a|s) R_t \right]$$

where R_t can be replaced by many expressions such as $R_t = \sum_{i=t}^T \gamma^{i-t} r_i$ or with the temporal difference (TD) error $R_t = r_t + \gamma V(s_{t+1}) - V(s)$

4 Method

Our method follows the standard paradigm of cooperative MARL - centralized training and decentralized execution, and uses actor-critic framework. Each group has its own actor and critic. Each group's actor outputs actions for all agents within the group given the group's local concatenated observations. Each group's critic is learned using only the group's local observations and guides the group's actor.

4.1 Train the critics

In training, we use a centralized but factorized critic as in VDN. The linear decomposition that VDN proposes is:

$$Q_{total}((h^1, h^2, \dots, h^d), (a^1, a^2, \dots, a^d)) \approx \sum_{i=1}^d \tilde{Q}_i(h^i, a^i)$$

Therefore, we get

$$V_{total}((h^1, h^2, \dots, h^d)) \approx \sum_{i=1}^d \tilde{V}_i(h^i) \quad (1)$$

where d is the number of agents. As linear decomposition of V_{total} limits the representational capacity of the function, we divide agents into groups so equation (1) becomes:

$$V_{total}((h^1, h^2, \dots, h^d)) \approx \sum_{g=1}^G \tilde{V}_g(h^g) \quad (2)$$

where G is the number of groups which is a hyperparameter. We learn \tilde{V}_g by backpropagating gradients from the following loss:

$$\mathcal{L}(\phi) = \sum_{i=1}^b \left\| V_{total}^\phi(\mathbf{h}_t) - y_i \right\|^2$$

where $y_i = r + \gamma V_{total}^\phi(\mathbf{h}_{t+1})$.



Figure 1: Snapshots of SMAC maps that we consider

4.2 Train the actors

After updating the group critics, we use \tilde{V}_g to derive the on-policy gradients for the group policies:

$$\nabla J(\theta_i) = E_{\pi_i} \left[\sum_{t=0}^T \nabla_{\theta_i} \log \pi_{\theta_i}(a_i|h_i) A(h_{i,t}, a_{i,t}) \right]$$

where $A(h_{i,t}, a_{i,t}) = r_t + \gamma \tilde{V}_i(h_{t+1}) - \tilde{V}_i(h_t)$ and $i \in [1, G]$. The reason that we can update each group policy independently is because of the linearly decomposed centralized critic [12]. Training requires some centralization, but during execution, only the learned group policies π_{θ_i} are used.

5 Experiments

In order to investigate the influence of dividing agents into groups, we run experiments on multiple environments and compare the average return of our method, VDN, and a centralized approach. As noted in the VDN paper, their approach is also applicable to policy gradient, therefore, we use the same implementation as our method but adjust the number of groups to N , the number of agents. We also use the same implementation for the centralized approach but adjust the number of groups to 1. For all experiments, we use groups of 2. We group agents in order using their ID number, agent 1 with agent 2, agent 3 with agent 4, and so on.... Moreover, we assume that each actor and critic is trained by an agent in the environment, and all agents have the same processing power. Therefore, we use the same neural net architect and hyperparameters for all actors and critics. The reason we impose this assumption is to avoid single point of failure problem and also ensure a fair comparison of the methods.

5.1 Decentralized StarCraft II Micromanagement - Partially Observable Environment

We perform some experiments on The StarCraft Multi-Agent Challenge (SMAC). We use combat scenarios where two groups, red and blue, of identical units are placed symmetrically on the map. The red group is controlled by RL policies. The blue group is controlled by a built-in StarCraft II AI, which uses handcrafted heuristics. The difficulty of the AI units is set to default which is *very difficult*. The action space of agents consists of *move[direction]* (four directions: north, south, east, or west), *attack[enemy_id]*, *stop* and *noop*. Dead agents can only take *noop* action while live agents cannot. The observation of an agent consists of *distance*, *relative x*, *relative y*, *health*, *shield*, and *unit_type* for each unit within the agent's sight range. An agent can use *attack[enemy_id]* only towards enemies in their *shooting range*. All agents have *sight range* larger than their *shooting range*. Agents can only observe other agents within their *sight range* that are not dead.

At each time step, the agents receive a shared positive reward from the hit-point damage dealt to the enemies or from killing the enemies. The maximum cumulative reward achievable in an episode is 20. We compare the methods on a set of maps (showed in figure 1) consists of 4 Marines (4m), 2 Stalkers and 2 Zealots (2s_2z), 3 Stalkers and 5 Zealots (3s_5z).

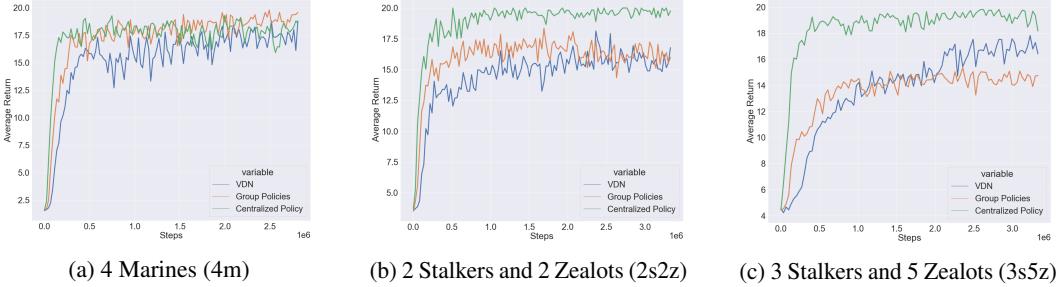


Figure 3: Average return on SMAC maps that we consider

5.2 Multi-Agent Particle Environments - Fully Observable Environment

We also perform an experiment using OpenAI’s multi-agent particle environment. We use Cooperative Navigation environment. In this environment, N agents must cooperate to reach a set of N landmarks, $N = 4$ in our experiment. Agents can observe the relative positions of other agents and landmarks and share a team reward based on how far any agent is from each landmark. Agents are penalized if they collide with other agents. Therefore, agents have to learn to cover all landmarks while avoiding collision.

6 Results

In order to evaluate each method’s performance, for every 20 iterations, we evaluate a batch of around 11 episodes and record the average return of each method.

6.1 SMAC - Partially Observable Environment

Showed in Figure 3

4m map In this map, the agents are homogeneous. The centralized policy achieves better returns earlier than all other methods and converges the fastest. An explanation is that centralized policy receives concatenated observations from all agents and controls all agents, so there is less uncertainty during training. Grouped policy achieves better returns earlier than VDN and converges to the best policy out of all three methods. This suggests that in simple environments, where all agents are homogeneous, the group policy can outperform centralized policy while converges faster than VDN.

2s2z map This map has heterogeneous agents. The centralized method is the strongest performer in this map. Group policy still achieves better returns earlier and converges faster than VDN. However, it converges to a sub optimal policy compared to the centralized method. This suggests that when complex behaviors are required, how we combine agents can affect its final policy.

3s5z map In this map, the agents are heterogeneous. The centralized method is still the strongest performer. Group policy still achieves better returns earlier than VDN. However, it converges to the least optimal policy out of all methods.

6.2 Multi-agent Particle Environment - Fully Observable Environment

Showed in figure 4

In fully observable environment, all methods converge to policies that are equally good. As for the convergence rate, the centralized method converges the fastest, then to ours and then VDN, which suggests that as the number of actors decreases, the convergence rate increases.

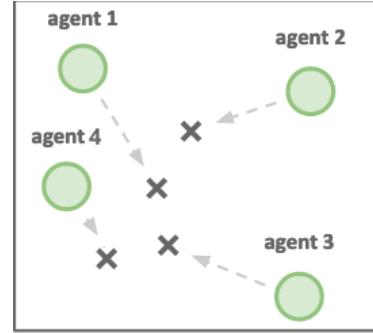


Figure 2: Cooperative Navigation

7 Future Work

Through our experiments, we think that dividing agents into groups is a promising approach. The number of groups and the number of agents in each group could be viewed as hyperparameters. Depending on the problems, we could tune these hyperparameters to get the optimal tradeoff in convergence rate and scalability. By decomposing the centralized critic and splitting agents into small groups, we address the problem of scalability. And by having more than one agent for each policy, we address the problem of convergence rate. However, our method is on-policy. With problems that have many agents, sample efficiency is also a tough issue that needs to be addressed. Using value-based methods or changing our implementation to off policy are possible approaches that should be investigated.

8 Conclusion

In this paper, we presented a novel group actor-critic method. We found that, for cooperative MARL, optimizing at group level is more scalable than centralized approach and could converge faster than at individual level.

9 Diagrams

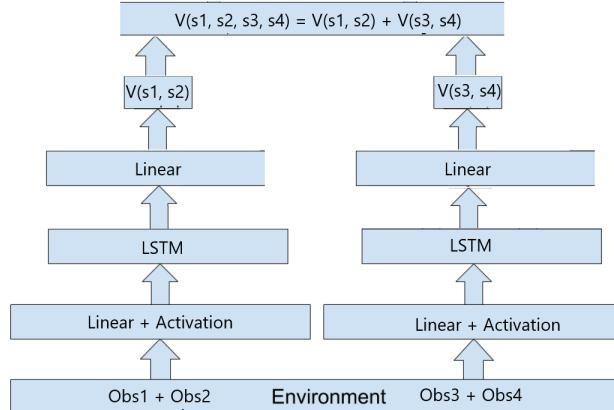


Figure 5: Centralized critic is sum of groups' critic

Modified from the diagram in VDN's paper

10 References

- [1] Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zam-baldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. *Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward*. In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, 2017.
- [2] an, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337, 1993.
- [3] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017
- [4] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
- [5] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 2017.
- [6] Sukhbaatar, S., Fergus, R., and others. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016
- [7] Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., and Wang, J. Multiagent Bidirectionally-CoordinatedNets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. *arXiv preprint arXiv:1703.10069*, 2017
- [8] Gupta, J. K., Egorov, M., and Kochenderfer, M. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems*, pp. 66–83. Springer, 2017.
- [9] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the Thirteenth International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2014.
- [10] A. Eck, L. Soh, S. Devlin, and D. Kudenko. Potential-based reward shaping for finite horizon online POMDP planning. *Autonomous Agents and Multi-Agent Systems*, 30(3):403–445, 2016.
- [11] Sutton, R. S.; McAllester, D. A.; Singh, S. P.; Mansour, Y.; et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, 1057–1063.
- [12] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, Chongjie Zhang. DOP: OFF-POLICY MULTI-AGENT DECOMPOSED POLICY GRADIENTS. *arXiv preprint arXiv:2007.12322*, 2020