MULTI-TASK MARL

Shenao Zhang

April 6, 2020

1 Background

1.1 Multi-agent RL

The main difference between MARL and single agent rl is non-stationarity of the multi agent environment. That is, if we independently optimize the policy of each agent while treating other agents as part of environment, other agents' policies are changing at the same time. To solve this problem, **centralized training** and **communication channel** are proposed.

Centralized training (and decentralized execution) means critic takes every agent's observation and action when training; when execution, we only need the policy network which only takes local observation. (Policy network is "actor" which is optimized by policy gradient, "critic" is to fit the actual reward for reducing variance.) Motivation: if we know the actions taken by all agents, the environment is stationary even as the policies change. *MADDPG* [1]: For every agent, learning an actor network and a centralized critic network. Action space is *contiguous*, so adopt *deterministic* policy gradient algorithms, which outputs a single value as action, instead of probability distribution in discrete action space.

Communication channel: communication means agents send their features to a center network, and then take the shared information (communication) as input during training and execution. *Multi-Actor-Attention-Critic* [2] share information between agents using a transformer-like structure at critic network, expecting to learn relationship between agents. *ATOC* (attentional communication model [3]): all agents share one centralized policy network, which means adjacent agents may have similar behaviors, however communication can increase the diversity of their strategies.

1.2 Multi-task RL

Meta-Learning Latent Variable Policies: sample latent variable as "skill" z, policy is then $\pi(a|s,z)$. Introducing latent variable facilitates the representation of several alternative solutions. SNN [4] first learns useful skills in a pre-training environment by sampling z from categorical distributions with uniform weights, and then leverages the acquired skills for learning faster in downstream tasks by learning task-specific latent space distributions. [5] learns "a diverse set of skills instead of just T separate solutions (one per task)", and expects high entropy of policy and high rewards simultaneously, which corresponds to several solutions for each task. [6] combines MAML and latent space variables to solve multi tasks.

Hierarchical policies: MLSA [7] is a metalearning approach for learning hierarchically structured policies, including shared sub-policies and master policy for different tasks. Specifically, a set of primitives are shared within a distribution of tasks, and are switched between by task-specific policies.
HIRO [8]: higher-level policy producing goals in terms of desired observations, lower-level policy is rewarded accordingly.

2 Multi-task MARL

2.1 Previous works in MARL algorithms:

- learn different policy for each agent can't generalize to different environments. Once number of agents changes, it needs to be trained from scratch again.
- learn the same policy for all agents. It is impossible for agents with similar observations (position) to explore and learn different strategies. Besides, such methods didn't show generalization ability.
- [9] defines relational MDP, using *class-based* local value subfunctions for each class. However, they *pre-defined a scope* for each object, and assumes equal relationships with other agents in this scope, zero relationship outside this scope. (They also assume that the value function can be well-approximated as a sum of local value subfunctions associated with the individual objects in the model.)
- recent work [10] proposed population-invariant **per-agent** actor and critic network (*transformer-like*). It first trains in environment with small number of agents, and evolve (cloning existing agents) to larger environments. Since the networks are per-agent ones, cloning and re-training are always needed and generalization ability is thus poor.

2.2 Motivation:

- As shown in [11], agents trained in larger environment (more agents) tend to have higher reward due to the trend of increasing exploration . However, [10] shows that directly training in large environment will be hard and sometimes fails. So here we would like to jointly train in environments with different number of agents and expect to learn generalized policies that cannot learnt in single tasks (e.g., low exploration with few agents).
- Per-agent policy cannot generalize well and naive same-policy-for-all (class-based) methods didn't perform well even in one task. So we use class-based methods and take advantage of **latent variable** to form multi-modal policy.
- **Relationships** between agents are important. [9] assumes equal relationships within a predefined scope. [10] needs to learn different relationships: two different relationships are required in critic network: observation embedding network and when calculating q value. Similarly, two relationships exist in policy network, thus leading to overfitting to current task (task includes environment with specific number of agents each role, world size, food range, number of landmarks).
- Another reason why [10] cannot generalize well is after cloning, they are assuming the relationships network (i.e., q and k in transformer) are similar in *different tasks*. However, the assumption doesn't hold when task is changing.

From above point **3** and **4**, we expect to learn a single **task-specific** relationship network that can be used in both actor and critic network to give reasonable and generalized relationship. From point **2**, we expect to sample different skills (latent code) for different agents.

So instead of directly model policy as $\pi_\theta(a|s, z_i)$, we integrate latent space code z and state s for generating task-specific relationship k, and output action distribution by $\pi_\theta(a|s, k)$. Similar to [4], for task $t \in T$, sample latent space skill $z \sim \text{Cat}(\psi(t))$, where $\psi(t)$ encodes task t, Cat is categorical distribution.

Intuitively, we expect to learn a multi-modal policy for all agents per role. That is, when knowing current task description, each agent will accordingly sample its own "intention" (latent space skill as mentioned before) in one episode and generate relationships with other agents each time step with local observation.

- **Task-specific latent space distribution can interprete task descriptions, and leads to generalization to unseen tasks.**
- **z can be seen as $\theta - \theta_i$, where i is ground truth parameter for agent i.**
- **Sampled "intention" encourages exploration and can learn multiple successful strategies, i.e., multi-modal policy (since we only learn one policy for many agents per role, in this way, agents with similar observation can still have different relationships, and thus different strategies).**

2.3 Formulation

- task t, local state (observation) s
- task encoder: $\psi(t)$
- latent code (skill) z: $z \sim \text{Cat}(\psi(t))$
- relationships k: $M_\phi(k|s,z)$
- policy: $\pi_\theta(a|s,k)$
- critic: $Q(s,a,k)$

Since we are using learnt relationships in both policy and critic networks, we fix the relationships when optimizing policy and critic, and needs additional reward when optimizing relation network and task encoder.

2.4 MI reward for relation network

Similar to [12], which unsupervised learns multiple skills in single agent RL task by regarding mutual information $\mathcal{I}(s';a|s)$ as policy reward, we adopt $\mathcal{I}(a;k|s) = \mathcal{I}(k;a|s)$ as reward of relation network M_ϕ. That is, how much we can know about the "ground truth" action a at state s of task t after giving the relationships k between agents, or symmetrically. So reward for task t is:

$$\begin{aligned}
R_t(s,k,a) &= \mathcal{I}(a;k|s) \\
&= \int p(k,s,a) \log \frac{p(a|s,k)}{p(a|s)} dadsdk \\
&= E_{k,s,a}[\log \frac{p(a|s,k)}{p(a|s)}] \\
&= E_{k,s,a}[\log \frac{\pi_\theta(a|s,k)}{p(a|s)}] + E_{k,s}[\mathcal{D}_{KL}(p(a|s,k)||\pi_\theta(a|s,k))] \\
&\geq E_{k,s,a}[\log \frac{\pi_\theta(a|s,k)}{p(a|s)}]
\end{aligned} \tag{1}$$

So we can maximize the above lower bound to optimize relation network. Next, we approximate $p(a|s)$.

$$\begin{aligned}
p(a|s) &= \int p(a|s,k)p(k|s)dk \\
&\approx \int \pi_\theta(a|s,k)p(k|s,z)p(z|t)dkdz
\end{aligned} \tag{2}$$

However, rather than fitting $p(k|s,z)$, we use a *deterministic* relationship encoder, similar as ones in deterministic policy gradient algorithms [13, 14] to solve contiguous action space control problems. That is, $k = M_\phi(s,z)$. Then we have:

$$\begin{aligned}
p(a|s) &= \int \pi_\theta(a|s,k)p(z|t)dz \\
&\approx \frac{1}{N}\sum_{i=1}^{N} \pi_\theta(a|s,k)|_{k=M_\phi(s,z_i),z_i \sim \text{Cat}(\psi(t))}
\end{aligned} \tag{3}$$

where N is number of samples per task, z_i is the ith sample from task t .
So the reward that generating relationship k w.r.t. state s is approximated by:

$$R_t(s,k,a) = \log \frac{N\pi_\theta(a|s,k)|_{k=M_\phi(s,z),z\sim\text{Cat}(\psi(t))}}{\sum_{i=1}^{N}\pi_\theta(a|s,k)|_{k=M_\phi(s,z_i),z_i\sim\text{Cat}(\psi(t))}} \tag{4}$$

However, given state s and relation k, the "ground truth" action a is unknown. We thus adopting *target policy* network, which is a previous version of current policy network (and also widely used when optimizing critic network for stabilizing training). Assume that several steps of optimizing θ of target policy π_θ will obtain a better policy $\pi_{\theta'}$ (which is current policy network). Then the desired action becomes $a \sim \pi_{\theta'}(\cdot|s,k)$. Reward now is:

$$R_t(s,k,a) = \log \frac{N\pi_\theta(a|s,k)|_{k=M_\phi(s,z),z\sim\text{Cat}(\psi(t)),a\sim\pi_{\theta'}(\cdot|s,k)}}{\sum_{i=1}^{N}\pi_\theta(a|s,k)|_{k=M_\phi(s,z_i),z_i\sim\text{Cat}(\psi(t)),a\sim\pi_{\theta'}(\cdot|s,k)}} \tag{5}$$

Intuitively, $R_t(s, k, a)$ indicates how well the policy network can correctly interpret current relationships (since the "ground truth" action also conditions on current relationships) compared with relationships generated by randomly sampled z.

Then optimize parameters of relation network ϕ and task embedding network ψ with $\nabla_\phi R$ and $\nabla_\psi R$ respectively for each task. Gumbel softmax replaces categorical distribution for bp, i.e., $z \sim$ gumbel_softmax$(\psi(t))$.

2.5 Optimizing task encoder

Recall that we expect each task to have its own skill set, from which each agent in that task sample a skill per episode. But the above reward cannot guarantee the diversity of skill sets, and will easily collapse into a single mode. So we also add a cycle consistent reconstruction loss for interpretability when optimizing task encoder:

$$\mathcal{L}_\psi = ||T - \psi'(\psi(T))||_2 \tag{6}$$

where ψ and ψ' is encoder and decoder respectively, T is task description list.

$\psi(T)$ will find an embedding space for task description so that $\psi'(T)$ can decodes the embedding to original task description. In this way, similar tasks (e.g., equally scaled environment) will have similar latent space probability distribution. $\mathcal{L}_\psi$ and $\nabla_\psi R$ are integrated together for optimizing ψ.

2.6 Pseudocode

Algorithm 1 Training Procedure

```
Input: $\Omega$ roles pf agents, a set of tasks $T$ with different (# total agents $A$, # agents per role $\{A_i\}_{1\leq i\leq\Omega}$, # landmarks, field of view, initial world size, initial landmark region)
Output: task $t$ specific latent code distribution $\psi(t)$, relation network $k = M_\phi(s, z)$, policy $\pi_{\theta_i}(a|s, k)$ for each role $1 \leq i \leq \Omega$, critic $Q_\alpha(s, a, k)$
initialize $P$ parallel environments, $T_{\text{update}} \leftarrow 0$
initialize replay buffer $\mathcal{D}_t$ for each task $t$
while total episodes number not reach do
  for task$t = 1 \dots T$ do
    Reset environment and get observation $s_j$ for each agent $j$
    while time steps in one episode not reach do
      sample skill $z_j \sim \text{gumbel\_softmax}(\psi(t))$ for each agent $j$, compute relationship $k_j = M_\phi(s_j, z_j)$
      select action $a_j \sim \pi_{\theta_i}(a_j|s_j, z_j)$ for agent $j$ with role $i$, get reward $R_j$, next observation $s_j^+$ and terminal symbol $d_j$
      store $(s_j, a_j, s_j^+, R_j, d_j, k_j)_{1\leq j\leq N}$ to replay buffer $\mathcal{D}_t$, $T_{\text{update}} = T_{\text{update}} + P$
      if $T_{\text{update}} \geq$ steps before update then
        update target parameters: $\pi_i^{\text{target}} = \pi_i$ and $Q^{\text{target}} = Q$
        sample minibatch with size $B$ from $\mathcal{D}_t$
        for update $= 1 \dots$ # optimization steps do
          update policy network using $\nabla_{\theta_i} J = \frac{1}{B}\sum_{b\in B} \nabla_{\theta_i} \log \pi_{\theta_i}(a_j^b|s_j^b, k_j^b)(Q_\alpha(s_j^b, a_j^b, k_j^b) - V)$
          update critic network by minizing $\mathcal{L}(\alpha) = \frac{1}{B}\sum_{b\in B}(y_j^b - Q_\alpha(s_j^b, a_j^b, k_j^b))^2$, where $y_j^b = R_j + \gamma Q^{\text{target}}(s_j^+, a_j^+, k_j^b), a_j^+ \sim \pi_i^{\text{target}}(\cdot|s_j^+, k_j^b)$
        end for
        compute desired action for target policy $\pi_i^{\text{target}}$ of each agent $j$, batch sample $b$: $a_{\text{target}} \sim \pi_{\theta_i}(\cdot|s_j^b, k_j^b)$
        sample latent code number $n = 1 \dots N$ for current task: $z_n \sim \text{gumbel\_softmax}(\psi(t))$
        update relation network $M_\phi$ and task encoder $\psi$ by:
            $\nabla_{\phi,\psi} J = \frac{1}{B}\sum_{b\in B} \nabla_{\phi,\psi} \log \frac{N\pi_i^{\text{target}}(a_{\text{target}}|s_j^b, k_j^b)}{\sum_{n=1}^{N} \pi_i^{\text{target}}(a_{\text{target}}|s_j^b, k_j^b)|_{k_j^b = M_\phi(s_j^b, z_n)}}$
        update task encoder $\psi$ by minimizing:
            $\mathcal{L}_\psi = ||T - \psi'(\psi(T))||_2$
        update target policy and target critic: $\pi_i^{\text{target}} = \pi_{\theta_i}$, $Q^{\text{target}} = Q_\alpha$
      end if
    end while
  end for
end while
```

References

[1] 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*, pages 6379–6390, 2017.

[2] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:1810.02912*, 2018.

[3] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in neural information processing systems*, pages 7254–7264, 2018.

[4] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[5] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[6] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018.

[7] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. META LEARNING SHARED HIERARCHIES. In *International Conference on Learning Representations*, 2018.

[8] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[9] Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational mdps. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1003–1010, 2003.

[10] Qian Long*, Zihan Zhou*, Abhinav Gupta, Fei Fang, Yi Wu†, and Xiaolong Wang†. Evolutionary population curriculum for scaling multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2020.

[11] Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural MMO: A massively multiplayer game environment for intelligent agents, 2019.

[12] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised skill discovery. In *International Conference on Learning Representations*, 2020.

[13] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.