Report About Multi-Agent Reinforcement Learning in Stochastic Networked Systems

20302124 Zhuang Yangyang

7. januar 2023

1. Introduction

The report is mainly about some ideas after reading the paper Multi-Agent Reinforcement Learning in Stochastic Networked Systems. And I write some codes to implement the examples in the paper. The objective of multi-agent reinforcement learning (MARL) in a stochastic network of agents is to find localized policies that maximize the (discounted) global reward. In comparison to single-agent reinforcement learning (RL), MARL poses many challenges, chief of which is scalability. Even if each agent's local state/action spaces are small, the size of the global state/action space can be large, potentially exponentially large in the number of agents, which renders many RL algorithms such as Q-learning not applicable.

In this paper, we introduce a class of stochastic, non-local dependency structures where every agent is allowed to depend on a random subset of agents. The author proposed and analyzed a Scalable Actor Critic (SAC) algorithm that provably learns a near-optimal local policy in a scalable manner. Moreover, in the context of the Examples, the author applied the Scalable Actor Critic (SAC) algorithm with parameter $\kappa = 1$ to the example of the Wireless Networks, which can learn a policy that performs better than the benchmark, which is used for comparison.

2. Networked MARL

2.1 Theories

According to the paper, we considering a network of agents that are associated with an underlying undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1,2,\cdots,n\}$ denotes the set of agents and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ denotes the set of edges. The distance $d_{\mathcal{G}}(i,j)$ between two agents i and j is defined as the number of edges on the shortest path that connects them on graph $\mathcal{G}$. Each agent has a local state $s_i \in \mathcal{S}_i$ and local actions $a_i \in \mathcal{A}_i$,

where $\mathcal{S}_i$ and $\mathcal{A}_i$ are finite sets. Based on local state and action, we also define global state and action: $s = (s_1, \cdots, s_n) \in \mathcal{S} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ and $a = (a_1, \cdots, a_n) \in \mathcal{A} := \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$, which are the combination of all local states/actions. N_i^κ denotes the κ-hop neighborhood of agent i on $\mathcal{G}$, i.e., $N_i^\kappa := \{j \in \mathcal{N} \mid d_\mathcal{G}(i,j) \leq \kappa\}$, which is similar to the concept of neighborhood in the mathematical sense. Let $f(\kappa) := \sup_i |N_i^\kappa|$. For a subset $M \subseteq \mathcal{N}$, we use s_M/a_M to denote the tuple formed by the states/actions of agents in M.

Next, we first define the notion of active link sets L, which are directed graphs on the agents $\mathcal{N}$. L characterize the interaction structure among the agents. A active link set is a subset of $\mathcal{N} \times \mathcal{N}$ and a super set of $\{(i,i)|i \in \mathcal{N}\}$, which means that all the self-loops belong to an active link set. Generally speaking, $(j,i) \in L$ means agent j can affect agent i in the activate link set L. Therefore, given a certain i and a special active link set L, we can use $N_i(L) := \{j \in \mathcal{N} \mid (j,i) \in L\}$ to denote all agents j that affect i. According to the paper, we consider a pair of active link sets (L_t^s,L_t^r) denote the result independently drawn from some joint distribution $\mathcal{D}$ at each time step t, where L_t^s denotes the active link set in the state transition at time t and L_t^r denotes the reward at time t. Therefore, L_t^s/L_t^r deotes the dependence structure ofstate transition/reward at time t.

As mentioned above, we talk about transitions first. At time t, assuming the current state is $s(t)$, the action in $s(t)$ is $a(t)$ and the active link set is L_t^s, the next individual state $s_i(t+1)$ is independently generated and only depends on the state/action of the agents in $N_i(L_t^s)$, which can directly affect agent i in the active link set L_t^s. In other words, we have

$$P\left(s(t+1) \mid s(t), a(t), L_t^s\right) = \prod_{i \in \mathcal{N}} P_i\left(s_i(t+1) \mid s_{N_i\left(L_t^s\right)}(t), a_{N_i\left(L_t^s\right)}(t), L_t^s\right) \tag{1}$$

Then we talk about rewards. We associate a local reward function ri with each agent. At time t, it is a function of L_t^r and the state/action of agents in $N_i(L_t^r)$: $r_i\left(L_t^r, s_{N_i\left(L_t^r\right)}(t), a_{N_i\left(L_t^r\right)}(t)\right)$. And we also need a global reward to control all agents. The global reward $r(t)$ is defined to be the summation of the local rewards $ri(t)$: $r(t) = \sum_i r_i(t)$.

Finally, we talk about policy. Specifically, at time step t, given the global state $s(t)$, agent i adopts a local policy ζ_i parameterized by θ_i to decide the distribution of $a_i(t)$ based on the the states of agents in N_i^β. Our objective is to find a largest discounted global reward which is produced by all agents working cooperatively. Here giving the discounted global reward as follows:

$$J(\theta) = \mathbb{E}_{s \sim \pi_0}\left[\sum_{t=0}^{\infty} \gamma^t r(s(t), a(t)) \mid s(0) = s\right] \tag{2}$$

where π_0 is a given distribution on the initial global state, and $r(s(t),a(t))$ denotes the global reward at time t, which is defined as the sum of all local rewards at time t: $r(s(t),a(t)) = \sum_i r_i(s_i(t),a_i(t))$.

2.2 μ-decay Property

One of the core challenges for multi-agent reinforcement learning is that the size of the Q function is exponentially large in the number of agents. For example, assuming that each agent has m possible actions and there are n agents. Each of them influence each other. It is obviously that m^n state/action situations is obviously exponential large when calculating the Q function of each agent, most of which are from other agents.

As mentioned above, we noticed that the key to the exponential growth of the Q-function is that any two agents interact in pairs, which says that the local Q-function of each agent i is mainly decided by the states of the agents who are near i. But the strength of the dependence between different agents may be different in practical tasks. For this case, we can consider discarding some dependencies between agents, which means we have to truncate the agent dependencies. In other words, we need to know how dependencies decay. Recently, exponential decay has been shown to hold in networked MARL when the network is static, which is exploited to design a scalable RL algorithm. However, in stochastic network settings it is too much to hope for exponential decay in general, and so we introduce the more general notion of μ-decay here, where μ is a function that converges to 0 as κ tends to infinity. The case of exponential decay that has been studied previously corresponds to $\mu(\kappa) = \gamma^\kappa/(1-\gamma)$. For simplicity, we use $i \xrightarrow{L} j$ to denote $(i,j) \in L$ and denote $N_{-i}^\kappa := \mathcal{N} \backslash N_i^\kappa$.

Definition 2.1 For a function $\mu : \mathbb{N} \to \mathbb{R}^+$ that satisfies $\lim_{\kappa \to +\infty} \mu(\kappa) = 0$, the μ-decay property holds if for any policy θ and any $i \in \mathcal{N}$, the local Q function Q_i^θ satisfies $\left|Q_i^\theta(s,a) - Q_i^\theta(s',a')\right| \leq \mu(\kappa)$ for any $(s,a),(s',a')$ that are identical within N_i^κ, i.e. $s_{N_i^\kappa} = s'_{N_i^\kappa}, a_{N_i^\kappa} = a'_{N_i^\kappa}$

Intuitively, if the μ-decay property holds and $\mu(\kappa)$ decays quickly as κ increases, we can approximately decompose the global Q function as $Q^\theta(s,a) = \frac{1}{n}\sum_{i=1}^n Q_i^\theta(s,a) \approx \frac{1}{n}\sum_{i=1}^n \hat{Q}_i^\theta\left(s_{N_i^\kappa}, a_{N_i^\kappa}\right)$, where $\hat{Q}_i$ only depends on the states and actions within κ-hop neighborhood of agent i. It can be proved that the global Q function can be directly decomposed in the networked MARL model and that the error of such decomposition is provably small. The following Theorem 2.1 shows the relationship between the random active link sets and the μ-decay property. Let $j \xrightarrow{L} i$ denote $(i,j) \in L$ and let $N_{-i}^\kappa := \mathcal{N} \backslash N_i^\kappa$.

Theorem 2.1 Define L^a as the static active link set that contains all pairs (i,j) whose graph distance on $\mathcal{G}$ is less than or equal to β, which is the dependency of local policy. Let random variable $X_i(\kappa)$ denote the smallest $t \in \mathbb{N}$ such that there exists a chain of agents

$$j_0^a \xrightarrow{L_0^s} j_1^s \xrightarrow{L^a} j_1^a \xrightarrow{L_1^s} \cdots \xrightarrow{L_{t-1}^s} j_t^s \xrightarrow{L^a} j_t^a$$

, that satisfies $j_0^a \in N_{-i}^\kappa$ and $j_t^a \xrightarrow{L_t^r} i$. The μ-decay property holds for $\mu(\kappa) = \frac{1}{1-\gamma}\mathbb{E}\left[\gamma^{X_i(\kappa)}\right]$.

According to Theorem 2.1, we study the case where long range links do not exist in the following Corollary 2.2. Therefore, in this case, we are capable of obtaining an exponential decay property.

Corollary 2.2 (Exponential Decay). Consider a distribution $\mathcal{D}$ of active link sets that satisfies

$$P_{(L^s,L^r)\sim\mathcal{D}}\{(i,j)\in L^s\}=0, \text{ for all } i,j\in\mathcal{N} \text{ s.t. } d_{\mathcal{G}}(i,j)\geq\alpha_1,$$
$$P_{(L^s,L^r)\sim\mathcal{D}}\{(i,j)\in L^r\}=0, \text{ for all } i,j\in\mathcal{N} \text{ s.t. } d_{\mathcal{G}}(i,j)\geq\alpha_2.$$

Then, $\mathbb{E}\left[\gamma^{X_i(\kappa)}\right]\leq C\rho^{\kappa}$, where $\rho=\gamma^{1/(\alpha_1+\beta)}$, $C=\gamma^{-\alpha_2/(\alpha_1+\beta)}$. According to the Corollary 2.2, long range active links can occur, but with exponentially small probability with respect to their distance. In this case, we can obtain a near exponential decay property where $\mu(\kappa)=O\left(\rho^{\kappa/\log\kappa}\right)$ for some $\rho\in(0,1)$.

Theorem 2.3 (Near-Exponential Decay) Suppose the distribution $\mathcal{D}$ of active link sets satisfies

$$P_{(L^s,L^r)\sim\mathcal{D}}\{(i,j)\in L^s\cup L^r\}\leq c\lambda^{d_{\mathcal{G}}(i,j)}, \text{ for all } i,j\in\mathcal{N}$$

where $c\geq 1$, $1>\lambda>0$ are constants. If the largest size of the κ neighborhood in the underlying graph $\mathcal{G}$ can be bounded by a ploynomial of κ, i.e., there exists some constants $c_0\geq 1, n_0\in\mathbb{N}$ such that $|\{j\in\mathcal{N}\mid d_{\mathcal{G}}(i,j)=\kappa\}|\leq c_0(\kappa+1)^{n_0}$ holds for all i, then $\mathbb{E}\left[\gamma^{X_i(\kappa-1)}\right]\leq C\rho^{\kappa/(1+\ln(\kappa+1))}$ for some positive constant C and decay rate $\rho<1$.

2.3 A Scalable Actor Critic Algorithm

Motivated by the μ-decay property, the paper design a novel Scalable Actor Critic algorithm for networked MARL(Multi-Agent Reinforcement Learning) problem shown in Algorithm 1 which exploits the µ-decay result in the previous section. In the algorithm, we use the local trajectory $\left\{\left(s_{N_i^\kappa},a_{N_i^\kappa},r_i\right)\right\}$ to evaluate the local Q-functions under parameter $\theta(m)$, as the Critic part (from line 2 to line 7) directly interacts with the environments. As for the Actor part (from line 8 to line 9), we use the estimated local Q-functions to compute the partial derivative to update local parameter θ_i.

According to the Near-Exponential Decay, it is intuitive that the µ-decay property guarantees that good approximation error can be achieved even when κ is not large. Specifically, Algorithm 1 highly scalable. Instead of considering of all the other agents, each agent i needs only to query and store the information within its κ-hop neighborhood during the learning process, whose error is tolerable and measured. The parameter κ can be set to balance accuracy and complexity. Obviously, as κ increases, the error bound becomes tighter at the expense of increasing computation, communication, and space complexity.

Algorithm 1 Scalable Actor Critic

1: **for** $m = 0,1,2,...$ **do**

2: Sample initial global state $s(0)$ π_0.

3: Each node i takes actions $a_i(0) \sim \zeta_i^{\theta_i(m)}\left(\cdot \mid s_{N_i^\beta}(0)\right)$ to obtain the global state $s(1)$.

4: Each node i records $s_{N_i^\kappa}(0), a_{N_i^\kappa}(0), r_i(0)$ and initialize $\hat{Q}_i^0$ to be all zero vector.

5: **for** $t = 1,2,...,T$ **do**

6: Each node i takes action $a_i(t) \sim \zeta_i^{\theta_i(m)}\left(\cdot \mid s_{N_i^\beta}(t)\right)$ to obtain the global state $s(t+1)$.

7: Each node i update the local estimation $\hat{Q}_i$ with step size $\alpha_{t-1} = \frac{H}{t-1+t_0}$,

$$\hat{Q}_i^t\left(s_{N_i^\kappa}(t-1), a_{N_i^\kappa}(t-1)\right) =$$
$$(1-\alpha_{t-1})\,\hat{Q}_i^{t-1}\left(s_{N_i^\kappa}(t-1), a_{N_i^\kappa}(t-1)\right) + \alpha_{t-1}\left(r_i(t) + \gamma \hat{Q}_i^{t-1}\left(s_{N_i^\kappa}(t), a_{N_i^\kappa}(t)\right)\right),$$
$$\hat{Q}_i^t\left(s_{N_i^\kappa}, a_{N_i^\kappa}\right) = \hat{Q}_i^{t-1}\left(s_{N_i^\kappa}, a_{N_i^\kappa}\right) \text{ for } \left(s_{N_i^\kappa}, a_{N_i^\kappa}\right) \neq \left(s_{N_i^\kappa}(t-1), a_{N_i^\kappa}(t-1)\right).$$

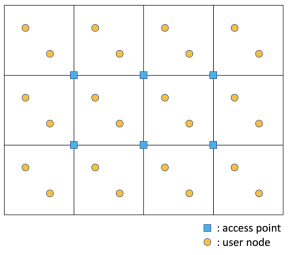
8: Each node i approximate $\nabla_{\theta_i} J(\theta)$ by

$$\hat{g}_i(m) = \sum_{t=0}^{T} \gamma^t \frac{1}{n} \sum_{j \in N_i^\kappa} \hat{Q}_j^T\left(s_{N_j^\kappa}(t), a_{N_j^\kappa}(t)\right) \nabla_{\theta_i} \log \zeta_i^{\theta_i(m)}\left(a_i(t) \mid s_{N_i^\beta}(t)\right)$$

9: Each node i conducts gradient ascent by $\theta_i(m+1) = \theta_i(m) + \eta_m \hat{g}_i(m)$.

3. Result of Example(A.1)

We will reproduce the Example(A.1) Wireless Networks. And we consider a wireless network with multiple access points shown in Figure 1. There are a set of user nodes in a wireless network, which are denoted as $U = \{u_1,u_2,...,u_n\}$ and share a set of access points $Y = \{y_1,y_2,...,y_m\}$ to denote the access points in the network. Each access point y_i is associated with a probability p_i of successful transmission.



Figur 1: An example of wireless network

Each user node is within the range of an access node. We stipulate that each user node can only access the access nodes within the range, which is a subset $Y_i \subset Y$. More specifically, each user node can send packets to an access point on the corner of its grid in Fig 1. We will build a MARL model based on this network. Each user node will be treated as an agent and the edge between u_i and u_j denote that u_i

and u_j may be conflict when sending packet. That is, $(u_i, u_j) \in \mathcal{E}$ if and only if $Y_i \cap Y_j \neq \emptyset$. Then we can build a graph $\mathcal{G}$ using $\mathcal{N}$ and $\mathcal{E}$.

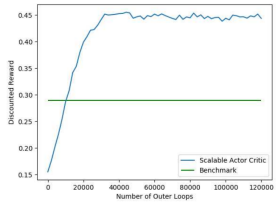
As for rewards in the wireless network, at each time step t, each user u_i receives a packet with initial life span d with probability q, which should be sent to the access point. Each user maintains a queue to cache the packets it receives. A packet will be removed from the queue at each time step if the packet is successfully sent to an access point or its remaining life span is 0. Otherwise, its life span will decrease by 1. Therefore, at each time step t, a user node ui can choose to send one of the packets in its queue to one of the access point $yi,t \in Y_i$. The packet from user i can be delivered successfully with probability $p_i \in U[0,1]$ only when no other user node sends packets to access point $y_{i,t}$ at time step t. After user node u_i successfully send a packet at time step t, it will receives a local reward of $r_{i,t} = 1$. Otherwise, u_i will receive $r_{i,t} = 0$. The objective is to find a policy that maximizes the global discounted reward under a discounted factor $0 \leq \gamma < 1$:

$$\mathbb{E}\left[\sum_{i=1}^{n}\sum_{t=0}^{\infty}\gamma^t r_{i,t}\right]$$

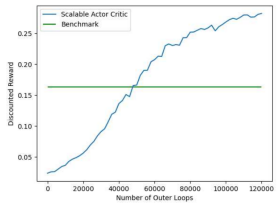
We then apply the MARL model to this wireless network. In order to indicate which packets are alive in the queue of each user node, we use a d-tuple $s_i = e_1, e_2, ..., e_d \in S_i := 0,1^d$ to denote the local state of user node i, where e_k indicates that there is a packet with remaining life span j in the queue of u_i. A local action of user node u_i is $2 - tuple(l,y)$, which means sending the packet with remaining life span $l \in \{1,2,\ldots,d\}$ to an access point $y \in Y_i$. It is noted that a user node ui performs an empty action if ui performs an action (l,y) when there is no packet with life span l in its queue. Specifically, in the setting of this example, the next local state of user node ui only depends on the current local states/actions in its 1-hop neighborhood, that is $\alpha_1 = 1$ in Corollary2.2. It is assumed that the action of each user node can be chosen only based on its current local state, that is $\beta = 1$. As mentioned above, we learn about the principles on the current model. Specifically, user node u_i only dependent on the user node that may connect to the same access point as u_i. Whether a user node can successfully send a packet depends only on a node adjacent to the graph $\mathcal{G}$. That is, the local reward of user u_i depends on the states/actions in its 1-hop neighborhood, which means $\alpha_2 = 1$ in Corollary 2.2.

The detailed setting we use is as follows. We consider the setting where the user nodes are located $h \times w$ grids (You can see it in Figure 1) and there are c user node in each grid. Initially, the life span of packet is seted as $d = 2$, the arrival probability $q = 0.5$, the discounted factor $\gamma = 0.7$. Each access point y_i can successfully transmit a packet with probability p_i, which is sampled uniformly randomly from [0,1]. Fianlly, we run the Scalable Actor Critic algorithm with parameter $\kappa = 1$ to learn a localized stochastic policy in the network which $(h,w,c) = (5,5,1)$ and $(h,w,c) = (3,4,2)$. For comparison, we use a benchmark based on the localized ALOHA protocol. Specifically, the benchmark policy works as following: At time step t, each user node u_i takes the empty action with a certain probability p'; otherwise, it sends the packet withthe minimum remaining life span to a random access point

in Y_i, with the probability proportionalto the successful transmission probability of this access point and inverse proportional to the numberof users sharing this access point. The results of the model from the original paper are shown in Figure 2 and Figure 3.

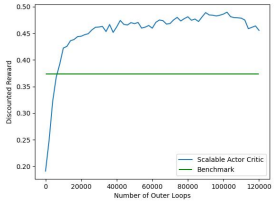


Figur 2: original result: Discounted reward of 5 × 5 grid, 1 user per grid.

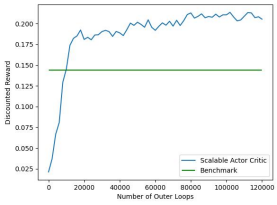


Figur 3: original result: Discounted reward of 3 × 4 grid, 2 user per grid.

Next, I will briefly describe how to reproduce the experiment. Firstly, we need to build the Wireless Networks in the example, which can generate the global reward according to the global states/actions of all user nodes and reach to the next global state. The environment can identify each access point with the grid where user nodes can access and how they can access. Secondly, we need to build the agents in the network which can update the state/action and local action according to the κ-hop neighbours and the Q-values estimated by the Critic part. In the Scalable Actor Critic algorithm, for each episode, given the global initialized state, each user node takes action to obtain the next global state according to the environment. And there will be a Critic part for each node and the details how the Critic part estimate the local Q-values, which mentioned in Algorithm 1 (line 7). Lastly, we set the parameters as mentioned. The outer loop for Scalable Actor Critic algorithm is seted as 120000, the count of the inner loop is 15. And the frequency of getting reward is seted as 2000. The results of the model from the my code are shown in Figure 4 and Figure 5.



Figur 4: my result: Discounted reward of 5 × 5 grid, 1 user per grid.



Figur 5: my result: Discounted reward of 3×4 grid, 2 user per grid.

By observing at Figure 2 to Figure 5, we see that the results using the algorithm(Scalable Actor Critic) are better than the baseline state(Benchmark) and the result is roughly the same as the original paper.

4. For More Analysis

To begin with, the structural formula of dependencies between different agents is relatively fixed. Specifically, they are all sampled with a certain probability D. For example, a more general modelthat the dependency structure between different agents will change over time. Obviously, the state transition and reward dependency structure of the agents depend on the current environment and the state of other agents. So, we can introduce parameters for active link sets as $L(s(t), a(t))$ and modify the Actor-Critic algorithm model and add a Director to tell each agent how to choose the appropriate dependency structure and appropriate other agents for training. For example, we can update our Director through the global state of the current time step and the state of all agents and use it to instruct an agent i how to choose an appropriate κ_i-neighborhood to update the local Q-function.

Furthermore, maybe we can introduce an neural network model for the assignment of the Q-values estimation like the Deep Q Network. The neural network model approximator of Q-values act as a gradually improved Critic, which takes the states/actions of the κ-hop neighbourhood as inputs and outputs what it estimates the Q-values. The loss function can be described by the local estimation of the Q-learning function. The neural network model updates its parameters through gradient descent. And each agent has a Critic to update the parameters of the local Q-values estimation through the gradient descent. The objective is to find localized policies that maximize the (discounted) global reward.

Lastly, the paper proposes a Scalable Actor-Critic algorithm for a stochastic network of agents. First of all, the authors introduce a new finite-time analysis of stochastic approximation. Then they show how stochastic approximation can be used for state aggregation. Finally they proposed the main algorithms and the theoretical results. The paper takes into account the problem of learning the optimal policies for stochastic network agents. The problem is interesting, and the authors show by examples that real-world problems can be cast into this framework. Going into details, the main difference between the setting proposed in this paper and the one of Qu et al.

is that the transitions and the rewards do not depend on static links between the agent, but that the links are sampled by a distribution . An interesting aspect that is not enough discussed in the paper is: what happens to the proposed algorithm if it is in the setting considered by Qu et al? Does it achieve better theoretical performance? The algorithm is a generalization of the one in [38] as author discussed in lines 337-340. In terms of the paper analysis, it show a stronger bound (see lines 371-373) than [38] under a different assumption to ensure sufficient exploration (Assumption 4.1). This is a contribution in the deterministic setting in addition to the generalization to the stochastic setting.

Code Linking

`https://github.com/sysu-yyxc/Final-Reinforcement-Learning`

References

NeurIPS 2021 · Yiheng Lin, Guannan Qu, Longbo Huang, Adam Wierman. Multi-Agent Reinforcement Learning in Stochastic Networked Systems