Reproducibility report of "Federated Reinforcement Learning with Environment Heterogeneity"

Reproducibility Summary

Scope of Reproducibility

The original work by Hao Jin et al addressed two issues: how to learn a single policy with uniformly good performance in all n environments, and how to achieve personalization. The main difference from the existing FedRL work is that the n environments have different state-transition functions. They propose two algorithms, QAvg and PAvg, which are federated extensions of Q-Learning and policy gradient, and then analyze Their convergence and showed how environment heterogeneity affects the convergence. Besides, they also propose a heuristic approach for personalization in FedRL, where environment embeddings are used to capture any specific environment, and prove that such heuristic helps to achieve generalization of convergent policies to fit any unseen environment via adjusting the embeddings.

Methodology

Since the original paper's code is open source, with very less descriptions and guidance. I read through the whole codes and comprehend the functions and implementation methods of each part. During reproducibility, Since there is basically no problem in original codes, most of the codes are based on the original codes.

Results

According to the analysis method and content in the paper, I print the results and draw the tables like that in the article, and find that there are some difference from that shown in the article.

Put opinions

There exists some minor flaws in the codes, it will be more satisfied after improvement.

Communication with original authors

I did not contact the original authors of the publication.

1 Introduction

In recent years, reinforcement learning has made unprecedented progresses in solving challenging problems. Traditionally, when handling such problems, one

typically assumes that the environment has a fixed state transition. However, in some real-life applications, an agent is expected to simultaneously deal with different state transitions in multiple environments. However, there are still many problems in the implementation of RL in practical scenarios. For example, in multi-agent RL, agents not only interact with the environment, but also have complex interactive relationships with other agents.
Federated Learning (FL) enables training a global model without sharing the decentralized raw data stored on multiple devices to protect data privacy. In general, FL is a ML algorithmic framework that allows multiple parties to perform ML under the requirements of privacy protection, data security, and regulations. In FL architecture, model construction includes two processes: model training and model inference. It is possible to exchange information about the model between parties during training, but not the data itself, so that data privacy will not be compromised in any way. An individual party or multiple parties can possess and maintain the trained model. In the process of model aggregation, more data instances collected from various parties contribute to updating the model. As a last step, a fair value-distribution mechanism should be used to share the profits obtained by the collaborative model. The well-designed mechanism enables the federation sustainability.
The article study a Federated Reinforcement Learning (FedRL) problem in which $\boldsymbol{n}$ agents collaboratively learn a single policy without sharing the trajectories they collected during agent-environment interaction. While the environment heterogeneity means $\boldsymbol{n}$ environments corresponding to these $\boldsymbol{n}$ agents have different state transitions.

2 Scope of reproducibility

In the process of reproducibility, I found that there many codes and these codes are not completely different, instead there are different applications of several algorithms mentioned in the article. And after compiling and running one of them, I found it cost much time completing one example and showing the result. So I choose the representative one that actualize the algorithm Qavg and use it to solve random MDP and windy cliff problem. And on this basis, to analyze the impact of local update time E and environment heterogeneity on convergent performance.

3 Methodology

The algorithm QAvg is already explained in the article, which learns a value function like Q-Learning. But there are also differences from Q-Learning, it iteratively perform local updates on the agent side and global aggregation on the server side.

Given the state space S, action space A, and reward function R, QAvg learns an $|S| \times |A|$ table by alternating between local updates and global

aggregations. The local update is the calculation like follows.

$$Q_{t+1}^k(s,a) \leftarrow (1-\eta_t)\cdot Q_t^k(s,a) + \eta_t \cdot \Big[R(s,a) + \gamma \sum_{s'} \mathcal{P}_k(s'|s,a) \max_{a' \in \mathcal{A}} Q_t^k(s',a')\Big].$$

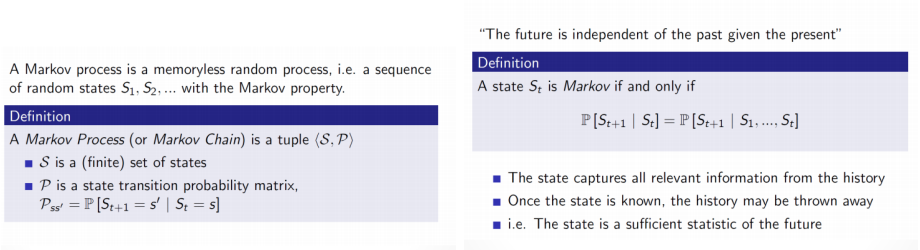
And global aggregation is like follows. through which Q tables are communicated and agents do not share their collected experience.

$$\bar{Q}_t(s,a) \leftarrow \frac{1}{n}\sum_{i=1}^{n} Q_t^i(s,a),\ \forall\, s,a;$$
$$Q_t^i(s,a) \leftarrow \bar{Q}_t(s,a),\ \forall\, s,a,k.$$

And the original article have already proved that QAvg converge to suboptimal solutions whose performance across the $\boldsymbol{n}$ environments are theoretically guaranteed. Sufficient theoretical basis and mathematical proof are given out in the article. If you are interested in them, you can find the article and read it, not tired in words here.

3.1 Model descriptions

MDP: Markov Decision Process formally describe an environment for reinforcement learning, where the environment is fully observable, that is the current state completely characterises the process.
And the following are also some basic descriptions and properties of Markov Decision Process.



For a Markov state s and successor state s', the *state transition probability* is defined by

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s',

$$\mathcal{P} = \text{from} \begin{bmatrix} & \text{to} & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix}$$

where each row of the matrix sums to 1.

The value function $v(s)$ gives the long-term value of state s

Definition

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right]$$

Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k+1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

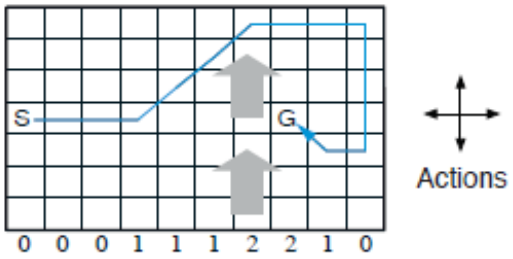
A Markov reward process is a Markov chain with values.

Definition

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

WindyCliff GridWorld: The windy gridworld is a simple example in the textbook. Now I introduce this to you, and it goes:
There is a standard grid world, each of them represents a state, with the start and goal states set on the grid. Your mission is to have a agent travel from the start point to the goal point successfully. Now this is a just standard grid world, if adding a crosswind running upward through the middle of the grid, it turns to be the windy grid world. For the beginning, the action space just contains four actions: up, down, right and left, but in the region of wind, the action will be shifted upward according to the level of the "wind", which varies from column to column.



For example, if you are just one cell to the right of the goal state, then the action left takes you to the cell just above the goal. The whole process is an undiscounted episodic task, with constant rewards of -1 until the goal state is reached.
To further consider the problem, you can think about what if expanding the action space to 8 actions? That is, on the basis of the previous, four actions are added: up left, up right, down left and down right. And what if the wind strength is stochastic instead of constant in every column?

3.2 Experimental setup and code

The original codes has high degree of completion, and there is basically no problem in implementation, thus there are few places to improve and perfect. Most of the codes used in reproducibility are based on the original codes.
The problem is that it time required for running is too long, running each files to get the similar results as in the article in the codes need more than 12 hours. It takes 12 hours for the short, and more than 80 even up to 180 hours for the long. As a result of which, running all the examples cost too much time and source.
After reading and understanding the source codes, in which I find that there exist some redundant computations or operations. For example, some calculations can be simplified, or some variables and computations are needed for reference and comparison. If only need the analysis for the algorithms mentioned in the article, we can remove the baseline operations like saving the data in the codes. By making some adjustments to the code, it helps simplify the code structure and accelerate the running.
Besides, functions used to draw the heat diagram like that in the article are not included. In order to draw the similar tables and show the objective values of FedRL at different iterations during the training of QAvgs with different E, I also modify the report function and add function used to print the heat diagram.

4 Results

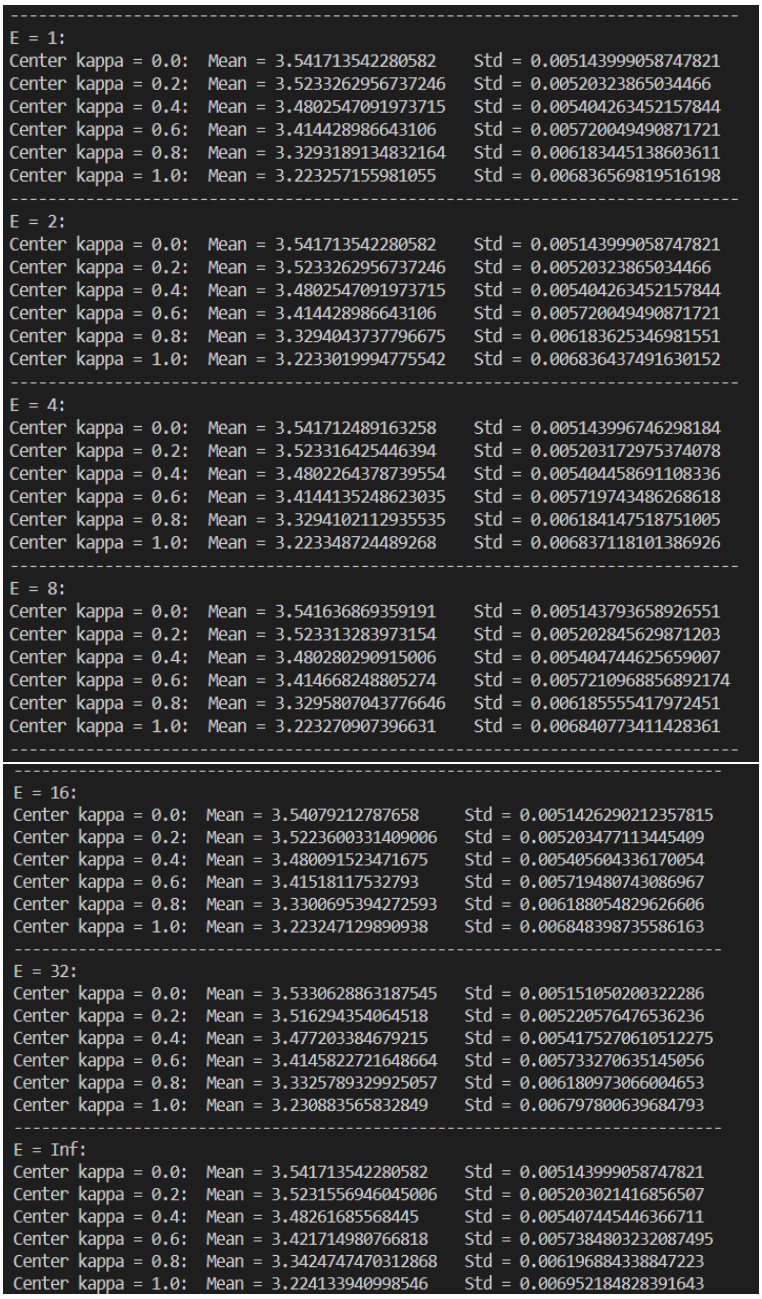
4.1 Results reproducing original paper

After running, the output of the file QAvg.py are as followed, which use QAvg to solve random MDP and analyze the impact of local update time E and environment heterogeneity on convergent performance.

The output of the code prints all the mean and standard deviation of different E. Then I compare them with the result shown in the article.

	RandomMDPs		
	QAvg	SoftPAvg	ProjPAvg
$\kappa = 0$	$35.42_{\pm 0.05}$	$35.15_{\pm 0.05}$	$34.97_{\pm 0.05}$
$\kappa = 0.2$	$35.23_{\pm 0.05}$	$34.97_{\pm 0.05}$	$34.92_{\pm 0.05}$
$\kappa = 0.4$	$34.80_{\pm 0.05}$	$34.58_{\pm 0.05}$	$34.54_{\pm 0.05}$
$\kappa = 0.6$	$34.14_{\pm 0.06}$	$34.02_{\pm 0.06}$	$34.02_{\pm 0.06}$
$\kappa = 0.8$	$33.29_{\pm 0.06}$	$33.25_{\pm 0.06}$	$33.38_{\pm 0.06}$

the result shown in the paper

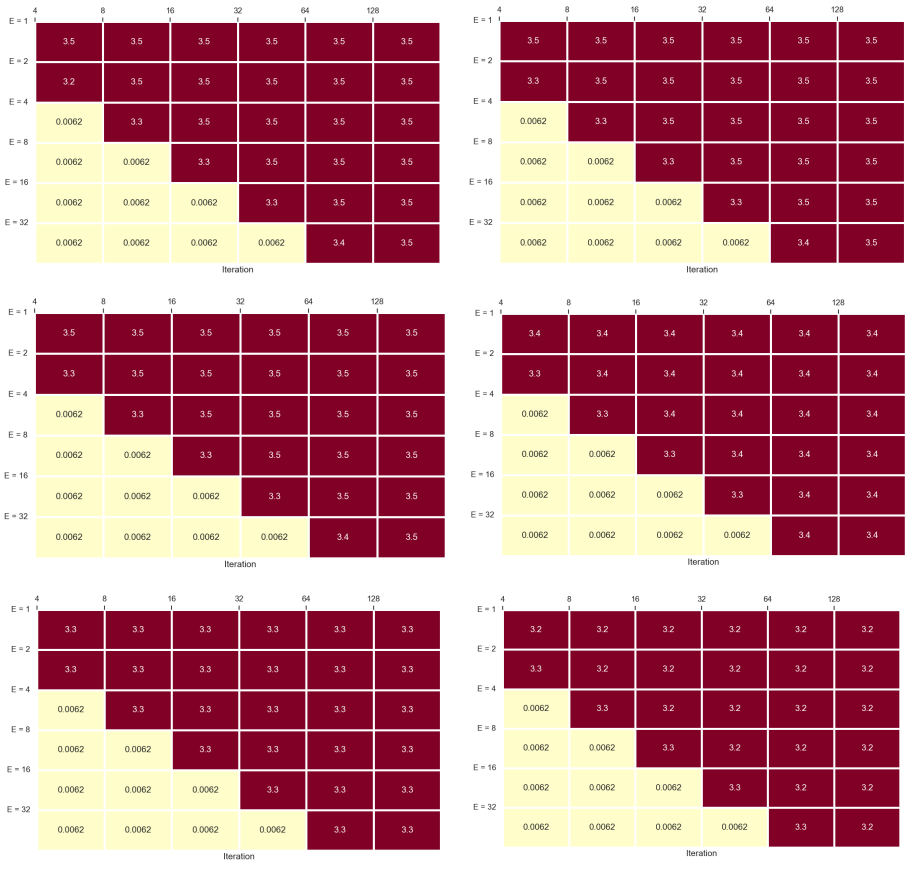


the output result of the code

Result given in the paper sets the **E** to 4, so I compare it with the output result with $\mathbf{E} = \mathbf{4}$. And find that all the mead and standard deviation are 10 times smaller than those in the article. And all results in reproducibility are

approximately 1/10 of the results in the paper, with very little error. The multiple relationship maybe caused by the setting of environment initial value and data. In general, the reproducibility of this table basically meets the expectation.

Besides, original article also draw a heat diagram to show the objective values of FedRL at different iterations during the training of QAvgs with different E. Unlike the paper use the data of QAvg solving the windy cliff, I use above data that are now available.
I draw a set of heat diagrams with different kappa, showing the objective values of FedRL at different iterations during the training of QAvgs with different E. The results are as followed.
From left to right, from top to bottom, the results of **kappa = 0, 0.2, 0.4, 0.6, 0.8, 1.0** are in order.



Because the data used are different, it is not possible to directly compare the results above and that in the paper. But it can also prove the theoretical result proposed in the article that the convergent **Q** table is free of **E** while

communication frequency affects the convergence speed, more specifically is that QAvg with larger **E** suffers from a lower convergence speed. As we all can see in the diagrams that the convergence value of different **E** is very similar, almost the same. As well as that the larger **E** is, the more iterations cost to reach convergence, meaning the slower convergence speed. And this result can be verified with different kappa, giving further proof for the theoretical result.

5 Put forward critical opinions

As mentioned above, the running time of each part of codes cost too much time. It will take several days if someone wants to reproduce the whole results the same as that shown in the paper. Some calculations in codes can be simplified, moreover, some operations for storing and updating can be removed since they do not matter when checking the results. And the original codes repeat the experiment for 16000 times, repeating each setting with 16000 random seeds. Maybe it is too large. Even if it is reduced, it is enough to verify the theory to be proved.
It can be said that the original codes is relatively perfect. If it can optimize the running time, the codes could strive for further improvement and be more satisfying.

6 Discussion

This time I reproduce the results in "Federated Reinforcement Learning with Environment Heterogeneity". Since there is basically no problem in implementation of original codes, most of the reproducing codes are based on original codes and I only make minor improvements.
Through this experience of reproducing, I learn much about solving the Federated Reinforcement Learning problem in which $\boldsymbol{n}$ agents collaboratively learn a single policy without sharing the trajectories they collected during agent-environment interaction. And environment heterogeneity is also important, which means n environments corresponding to these $\boldsymbol{n}$ agents have different state transitions.
I learn some new algorithms, including **QAvg** and **PAvg**, **DQNAvg** and **DDPGAvg**. Knowing about their principle and implementation methods, as well as their convergence and efficiency in application to the problems. I have benefited a lot.

The code is open source by the link:
(https://github.com/ShuaiShuyi/SYSU2022-Reinforcement-Learning-Final-assignment)

References

[1] Yang H . federated reinforcement 1eaming with environment heterogeneity[J]. 2022.

[2] Qi J , Zhou Q , Lei L , et al. Federated Reinforcement Learning: Techniques, Applications, and Open Challenges[J]. 2021.

[3] Zhuo H H , Feng W , Xu Q , et al. Federated Reinforcement Learning[C]// IEEE. 0.

[4] Reinforcement Learning: An Introduction (http://www.incompleteideas.net/book/the-book-2nd.html)

[5] Windy Gridworld: A simple implementation of Tabular RL (https://blog.csdn.net/WZXHello/article/details/115560022)