

Improved Methods based on Too Many Cooks

Xiaotian Yuan

Department of Informatics, King's College London, London, UK

xiaotian.yuan@kcl.ac.uk

Abstract. In the paper "too many cooks", the authors developed a method called Bayesian Delegation to enable human-like coordination by inferring the sub-tasks of others quickly. However, some limitations which could be solved to improve the whole programme still exist in the partial work of the code. First, the implement of sub-task in terms of efficient actions or which agent(s) should work on it is not specified. Second, the sub-tasks may be finished in many different orders since the ordering of sub-tasks is partial and so agents probably have blank time. Third, with the increasing running times the action of different state could be recorded to improve the efficiency of the programme. So, this paper will focus on these three issues and try to solve them. In this work, some solutions to these challenges using Deep Q-Learning (DQN) and Bayesian inference are proposed. In the DQN experiment, value approximation has good performance in the simple multi-agents' environment, the limitation that the implement of sub-task in terms of efficient actions or which agent(s) should work on it is specified in our model. Besides, in the Bayesian inference experiment, the code could give accurate action prediction with enough database, and also avoid some time consuming because the blank time between subtasks is saved.

Keywords: Coordination; Multi-agent Reinforcement Learning; Deep Q-Learning Network; Bayesian Inference.

1. Introduction

Currently, AI has made clear technological advancements in handling difficulties related to finances, healthcare, security, and many other domains, and its application prospects are expanding rapidly. Now that AI is achieving global commercialization, it has sparked significant changes in a number of different industries and helped speed up business digitization, modify the structure of the industrial supply chain, and increase the effectiveness of information use. At the same time, AI has entered the era of machine learning, and one of the mainstreams to implement machine learning is reinforcement learning. Reinforcement learning is a complex and vast field, and what distinguishes reinforcement learning from other branches of machine learning such as supervised and unsupervised learning is that reinforcement learning is centered on the concept of interactive learning. Reinforcement learning models (also known as agents) learn in interaction with the environment to maximize a reward function. In reinforcement learning, we don't teach the agent how to do things, it only defines what the agent is expected to achieve. Then, on the basis of specific experimental results, rewards are determined based on the success or failure of the agent. Reinforcement learning is a better tool for making decisions in complex environments. the central mathematical concept is that of the Markov decision process. This paper builds on the work of "too many cooks" and mentions that currently, agents cannot determine when they have accomplished their individual "share" of a group effort when they jointly plan for a single subtask for discussion [1]. This paper uses two models to measure how multiple agents divide works and try to overcome the three limitations mentioned in the abstract. The first model is based on the traditional DQN, builds the target network and evaluation network, and infers the task assignment of the agent; the second model is based on Bayesian inference, infers the agent's assignment of subtasks.

2. Materials and Methods

This paper aims to use two models to measure how the multi-intelligent agents divide work and overcome the limitations of the original programme. This paper uses the gym-cooking as the

environment, which has two agents and how to separate work to cook. This work refers to the paper “too many cooks” and create the environment to increase two functions. One of the functions uses deep Q-learning network to determine when agents divide the task, the priority of the subtask, which means in a subtask, agents do first, and the other is next. The agent who finishes his portion can depart from this subtask, or the agent does not wait for another agent. Another function uses Bayesian inference to infer how agents perform more than one task or which action should be chosen in the current state. As for this function, this work can measure more situations by dividing the work of subtasks, and improve the efficiency of the programme to some extent.

2.1 Deep Q-learning Network Method

The first model is deep Q-Learning network (DQN), and the paper measure the value approximation performance in this situation.

DQN is similar to Q-Learning, both are the algorithm according to the iteration of outcomes. However, in ordinary Q-learning, Q-Table is proper to be used as database in order to record the Q value of different state action pair in the low dimension with discrete states and actions space. If the high-dimensional continuous situation of states and actions space appears, it probably will find hard to still take this way and the Q-Table inaction space and the state is too large to use. So here the work will be turned into a function fitting problem instead of the original Q-table updating problem, and how to use the fitting function which is different from traditional Q-table to calculate the Q value is very important, and in this case the work can expect to receive similar outcome actions if similar states act as the input value. As a result, this work can integrate deep learning with reinforcement learning and consider that the deep neural network has a positive impact on the extraction of complicated features [2]. This becomes DQN. DQN is updated in the same way as Q-learning:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (1)$$

The loss function of DQN is as follows, θ indicates that the network parameters are mean square error loss:

$$L(\theta) = E[(\text{TargetQ} - Q(s, a; \theta))^2] \quad (2)$$

$$\text{TargetQ} = r + \gamma \max_{a'} Q(s', a'; \theta) \quad (3)$$

In this model the network has four layers. The first layer is the input layer, which has five neurons as the state input, the second and third layers are fully connected; they have fifty neurons. The last layer is the output layer, with four neurons representing the action [3-4].

Also, the Markov Decision Process for agents are used to make their actions. A mathematical model of sequential decision called the Markov Decision Process (MDP) is used to simulate the stochastic strategies and returns that agents in an environment with Markov features can realise in the system state. State, action, strategy, and reward are among the components of MDP, which is based on a collection of interacting objects, called agents and settings [5]. In the MDP simulation, the intelligent experience recognizes the present system state, carries out actions on the environment in accordance with the strategy, and therefore modifies the state of the environment while gaining rewards. Returns are the accumulated rewards over time [6].

Unlike traditional DQN, this paper has two networks; target network and evaluate the network. Furthermore, this paper utilizes the experience replay, which can help us train more efficiently. The work increases the reward degree and make the reward can converge very quickly [7].

2.2 Bayesian Inference Method

The second model is Bayesian Inference. A supervised learning approach called Naive Bayes applies Bayes' theorem under the "naive" assumption that given the value of the class variable, every pair of features is conditionally independent of one another. Given the class variable y and dependent feature vector $(x_1, \dots, x_n)$, it can express the posterior distribution using Bayes' theorem as follows [8]:

$$P(y | x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)} \quad (4)$$

Based on the naive conditional independence assumption that

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y) \quad (5)$$

For all i , this relationship is simplified to

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)} \quad (6)$$

Since $P(x_1, \dots, x_n)$ is constant given the input, the following classification rule is used:

$$P(y | x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i | y) \quad (7)$$

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i | y) \quad (8)$$

The code is based on Naive Bayes, it requires a database of running action in different state. Then with the underlying database, once the current state of each agent is received, the code will calculate the likelihood of every possible action and return the action with the highest likelihood.

To make sure the accuracy of the code, Laplacian correction is also added in the code. Laplacian correction means that the total number of categories in the training set is expressed in N ; The possible values of d_i attribute are expressed by N_i , so the formula for calculating the original prior probability $P(c)$ is [8]:

$$P(c) = \frac{Dc}{D} \quad (9)$$

Revised by Laplacian correction as:

$$P(c) = \frac{Dc + 1}{D + N} \quad (10)$$

In this case the code will weaken the bad effect which is caused by lack of database.

2.3 Experiment

The paper create the environment referring to the package gym of OpenAI [3]; we write the several main functions that env.step() env.reset(). As follow, the introduction of parameters of ENV.

There is Action_space, State_space, Observation.

Action_space = [0:cut tomato 1:cut lettuce 2:cooperate 3:mix] total 4 actions

State_space = [agent1(none, cut1, cut2,)
agent2(none, cut1, cut2,)
tomato(not cut, 1, 2)
lettuce(not cut, 1, 2)
salad(none, 1, 2)]

And the observation structure is the same as the State_space. The explanation of the actions in the Action_space. The agent has 4 acts; when the agent executes the cut tomato, the agent's state will become cut1, and the tomato's state will add "1", which means the number of tomatoes is ready. The act cut lettuce is the same as the act cut tomato. Still, the reward differs when the agent chooses to cooperate when another state is cut1 or cut2. When the agent act and cooperate with another agent, the state is cut1, and the reward is 0.1, while will cut2, the reward is 0.01. It can be distinct the efficiency of the cooperation, avoiding being trapped in constantly choosing the cooperation [9]. Then, when the agent wants to act mix, the tomato state and lettuce state need to be up to 1, and they will be minus one, and the salad state adds 1. Finally, the condition of finishing an episode is salad's state of more than two or equal to two, and this step deserves one reward.

When the environment is built, we then run the DQN model to get the details of each agents' action decision in every different state, this decision is inferred by MDP and trained by DQN model. With the increasing episode times, we could expect to receive better performance from the agents. Then we run the Bayesian inference model, this model focus on how to solve the connection between different tasks and sub-tasks, also it could be used to improve the efficiency of the programme [10]. We have noticed that in the programme although most time agents have to deal with different sub-tasks, however the number of the type of tasks is limited, so we could use a database to record the action and state pairs and then use the Bayesian inference to infer the action based on the database, instead of asking agents to try to handle duplicate routines. In this case we could find the model is able to help the programme save lots of time. Besides, we also notice that in the programme the order of cooking sub-tasks like cut tomato will not influence the tasks, so using Bayesian inference could make sure that agents could work on different cooking sub-tasks from different tasks.

3. Result

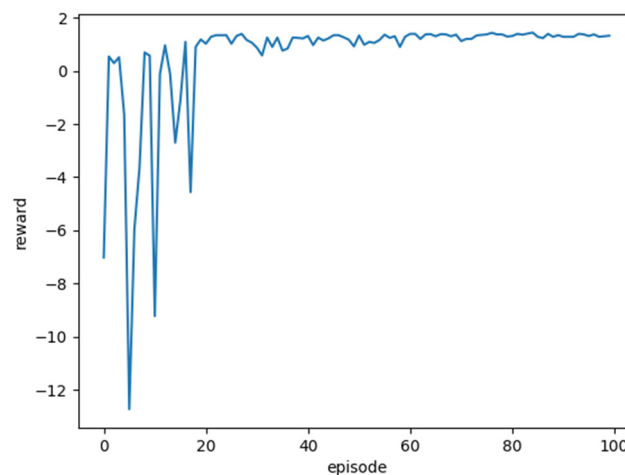


Fig 1. DQN model train network shows the relationship between episode and reward

It can notice that value approximation has good performance in the simple multi-intelligent agents' environment, it can converge quickly in 20 episodes (Fig.1). While in the last 20 episodes, the odds are up to 95%. Compared with the original programme, this model could overcome the limitation that the implement of sub-task in terms of efficient actions or which agent(s) should work on it is not specified, we also offer a new way for agents to make decision.

From the result, it can notice that Bayesian inference could have good efficiency for agents to handle duplicate routines (Table 1). Also, because it focusses on states and actions, it could allow the agents to work on different sub-tasks from different tasks. Compared with the original programme, this model could overcome the limitation that the sub-tasks may be finished in many different orders since the ordering of sub-tasks is partial and so agents probably have blank time, and it also improve the efficiency of the programme.

Table 1. Bayesian inference model shows the running time and action inference of an example state

Running times	5	10	100 or bigger
Running cost	0.0002s	0.00015	close to 0s

We built an environment to test the performance of DQN and Bayesian inference. At first, we amplified the dimension of Action_space and State_space, it has ten acts and nine states with coordination, and it is hard to converge in 1000 episodes. Due to the coordination, the complexity of space and time causes training an episode to need much time. Also, the amount of the valid data will have effects on the accuracy of the Bayesian inference of the action. In the future, an algorithm based on policy to solve these problems will be tried.

4. Conclusion

This research tries to solve the challenges mentioned in the paper “too many cooks” for the implementation of sub-task in terms of efficient actions and the ordering of sub-tasks. First, some models and functions are used to measure more situations that how agents perform more than one task. Second, the priority of the subtask is assigned when agents work jointly in one sub-task. We use the methods of DQN to overcome the limitation that sub-task in terms of efficient actions or which agent(s) should work on it is not specified, and this work also offers a new way for agents to make decision. Then this paper uses the methods of Bayesian Inference to overcome the limitation that the sub-tasks may be finished in many different orders since the ordering of sub-tasks is partial and so agents probably have blank time, and also improve the efficiency of the programme. Both models have achieved some good performance in the experiments. The programme “too many cooks” offers us a good example of the multi-agents works, and we believe multi-agents works situation will become more and more popular in all kind of AI works in the future. Multi-agents work together could help us to deal with much more complex issues, improve both effectiveness and efficiency of AI works.

References

- [1] Rose E. Wang, Sarah A. Wu James A. Evans, Joshua B. Tenenbaum, David C. Parkes, Max Kleiman-Weiner. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. 2020,561 (2): 392-403.
- [2] Ong H Y, Chavez K, Hong A. Distributed Deep Q-Learning. Computer Science, 2015.21(11), 54-63.
- [3] Forrest Ding. Deep Q-Network (DQN) [web log post].2020, 2(22) Retrieved from [https:// zhuanlan. zhihu. com/p/108286901](https://zhuanlan.zhihu.com/p/108286901).
- [4] Zhang Q, Lin M, Yang L T, et al. Energy-Efficient Scheduling for Real-Time Systems Based on Deep Q-Learning Model. IEEE Transactions on Sustainable Computing, 2017:1-11.

- [5] H Cruz-Suárez, G. Zacarías-Espinoza, V Vázquez-Guevara. A Version of the Euler Equation in Discounted Markov Decision Processes. *Journal of Applied Mathematics*, 2012(2):1-12.
- [6] Ohtsubo Y. Optimal threshold probability in undiscounted Markov decision processes with a target set. *Applied Mathematics & Computation*, 2004, 149(2):519-532.
- [7] Mehmood T, Kanwal A, Butt M M. Naive Bayes combined with partial least squares for classification of high dimensional microarray data. *Chemometrics and Intelligent Laboratory Systems*, 2022(222-):222.
- [8] Ning K, Sun T, Zhao H, et al. Naive Bayesian Classification Algorithm Based on Attribute Association[J]. *Computer Engineering*, 2018, 44:492-504.
- [9] Kanwal A, Mohyud-Din S T. Coupling of laplace transform and correction functional for wave equations. *World Journal of Modelling and Simulation*, 2013, 9(3):173-180.
- [10] Pino R, Scuseria G E. Laplace-transformed diagonal Dyson correction to quasiparticle energies in periodic systems. *The Journal of Chemical Physics*, 2004, 121(6):2553-2557.