Basic things about reinforcement learning

Ziyu Zhang

CCNU Overseas Study Service Center, Central China Normal University, Wuhan, Hubei, China, 430079

1095395278@qq.com

Abstract. Artificial Intelligence has been a very popular topic at present, machine learning is also one of the main algorithms in AI, which consisted of Supervised learning, Unsupervised learning and Reinforcement learning, and Supervised learning and Unsupervised learning have been relatively mature. Reinforcement learning technology has a long history, it wasn't until the late '80s and early' 90s that reinforcement learning became widely used in artificial intelligence, machine learningGenerally, Reinforcement learning is a process of trial and error, agent will choose to make an action according to the feedback from the environment, this step will repeat a lot of times until it find the best policy, mapping it to reality, it can help human to fulfill some missions which are nearly impossible before. However, there is still some potential problems in Reinforcement learning. This essay compares some basic algorithms related to RL to help reader to have a basic understanding of RL and propose some exsiting defects about it.

Keywords: Reinforcement learning，Markov Desicion Process，Machine Learning

1. Introduction

Reinforcement Learning has become more and more popular at present, it has many applications in reality, for example, many games like Go, Dota2, Starcraft, Atari and so on, RL algorithm has surpass the performance of many humans in those games. However, pure RL has many defects, so that there many improved and novel algorithms on the basis of RL. Such as Deep Reinforcement Learning which combines Reinforcement Learning and Deep Learning to solve the problem of too large state space. This paper will later introduce the basic framework of RL, its typical model Markov Decision Process, the difference between model-based RL and model-free RL, fundamental concept of Q-learning and its combination with DL which is Deep Q Network. There are some unsolved problems and questions and also some proposals in the end.

2. Reinforcement learning

Reinforcement learning is one method of machine learing, it is for describing and solving problems in which an agent learns strategies to maximise rewards or achieve specific goals during its interaction with the environment. RL can be seen as the process of trial and error, it is consisted of 6 main elements: Agent, environment, state, action, reward and policy.

1. Agent:The ontology of reinforcement learning, you could regard it as the brain of human. The target of agent is to get the maximum cumulative reward by iteracting with the environment by following the policy[1].

2. Environment:All content that interacts with the agent is called environment, in another word, everything unless agent itself is environment.

3. State:A data used to represent environment, the state set is the sum of all possible environment, and its total quantity is always very large.

4. Action:Some of the actions that the agent is allowed to do according to the state and policy, similar to the commands sent by the human brain to the body. For example, if you are playing Mario game, moving to the left, moving to the right and jump are the actions which agen can do.

5. Reward:If agent does an action close to the target, it will be rewarded and the opposite will be punished, and the reward can be received immediately or delayed[2].

6. Policy:The policy is represented by π, The policy is the core of the algorithm and is by itself suffificient enough to determine the behaviours of the agent[3].

The agent gives an action at to the environment, and it receives a reward rt which could be positive or negative from the environment and the state transfer from st to st+1, The new state may or may not be different from the previous state, depending on the action that was taken[3] . The agent will tend to do the action which could obtain positive reward until it gets the policy which has total maximum reward.

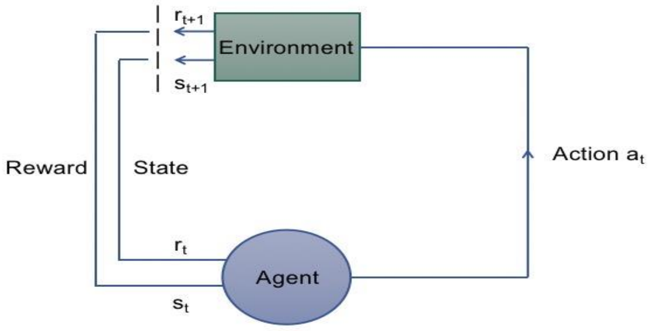

Fig1: The working process of RL

3. Markov Decision Process

When a RL problem satisfies the Markov property, for example, the future depends only on the current state and action, but not on the past, then it could be modelled as a Markov Decision Process(MDP) [4]. MDP is consisted of 5 main parts, there are State space(S), Action space(A), Transition possibility(P), Reward function(R) and Discount factor(γ). {S, A, P, R, γ}

S contains all the possible state S={s_1, s_2, s_3, . . . , s_t}.

A contains all the possible action which agent could do. A={a_1, a_2, a_3, . . . , a_t}, A|s represents set of all actions in state s.

P(s'|s, a) means doing action a in state s, the possibility of transfering from state s to state s'.

R(s, a) represents the expected reward of doing action a in state s.

γ could reduce the proportion of a reward, because sometimes the further reward is not as great as the reward at present. For an example, the 100 dollars today is worth more than the 100 dollars after 100 years because of inflation.

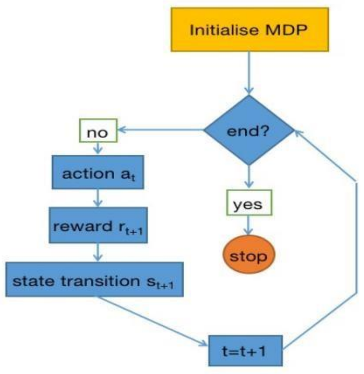

Fig2: Process of MDP

As you can see in Fig2, the system will first initialize MDP and randomly produces π_0 and initialize state s_0 normally, then the agent will choose an action a_0 according to the π_0, next the environment will give back a reward r_1 and the state updates to s1, finally system will ask you to stop it or not. This is one episode and it will repeat a lot of times to form a Markov chain(s_0, a_0, $r_1 \rightarrow s_1$, a_1, $r_2 \rightarrow s_2$, a_2, $r_3 \rightarrow . . . \rightarrow s_{T-1}$, a_{T-1}, $r_T \rightarrow s_T$(ending state)), then we could calculate the total reward of this sequence, which is denoted by Gt.

$G_t = r_1 + \gamma_1 r_2 + \gamma_2 r_3 + . . . + \gamma_{T-1} r_T$

We could estimate the quality of policy π0 through the value of Gt, the objective of MDP is to find the best policy with biggest G_t, which could solve the problem perfectly in reality.

G_t is the total reward by following a policy, it could be calculated by value function Vπ(s) (expected return) which represents the total maximum cumulative reward in state s while following policy π. Another similar function is action-value function Qπ(s, a), it means the expected return when taking action a in the state s. These two functions are both used in temporal difference (TD) learning method[5]. The aim of TD is find the optimal policy by using some functions and algorithms such as Q-learning, TD error function and approximation function to keep optimizing and updating Vπ(s) and Qπ(s, a) step by step.

4. Model-based and Model-free

However, the agent sometimes do not know what next state will be after it did an action and what reward it will get, this means the agent do not have the ability to construct the model of environment, in a word, the agent cannot learn to build a model of environment. This is model-free RL, some common algorithm like Q-learning and TD are in the branch of model-free RL.

The opposite one is model-based RL, which means the agent knows what state will be and what reward it will get with an action so that it is able to learn from that and build a environment model. An example from the reality is the AlphaZero which is the algorithm proposed by Deepmind. In chess, AlphaZero surpassed the world champion program Stockfish in just four hours of training and won the world champion program Elmo in Shogi with model-based RL. Although model-based RL is used wider than model-free RL normally, in some realistic environment we do not know much about the environment or we do not know what will happen if we do an action just like the agent in model-free RL, so model-free is tend to finish the task in this situation.

Overall, model-based RL try to model the environment and then based on model to choose the most suitable policy. Model-free RL try to learn the optimal policy in one step.

5. Q-Learning

Q-Learning is one algorithm of RL, the targer of Q-Learning is quite similar with RL, which is to find the optimal policy by using action-value function Q to approximate optimal action-value functioin Q[2]. Q-Learning will store the value of action-value functioin for combining each possible action with each state to form a table which called Q-table. Then the agent will choose the action with biggest reward in a given state, and then state is updated and agent will keep doing this according to the Q-table until the ending state.

However, this can bring a problem that is the agent will always choose the same action in a given state. As a result, some actions will never be taken, hence the value of Q cannot update. This problem can be solved by the ε-greedy, agent will randomly takes the action according to the value of ε. This ensures that each value of Q will be updated with enough time steps[5].

There is a little question which is how to deal with the situation that agent want to take two or even more actions in one given state, then how to form Q-table with two actions in one block?For example, if the agent is playing Super Mario game, and it want to jump over an obstacle which is in the right but it need to jump and move to right simultaneously so that can pass it.

6. Deep Reinforcement Learning

DRL is a novel technology which proposed by Deepmind, it combines reinforcement learning with deep learning, which made RL can receive high-dimensional input and produce high-dimensional output as well. Take Deep Q Network(DQN) which combines Q-learning with DL for an example, it can solve the problem of consuming too much time to search the corresponding state in a large Q-table[4]. If the agent is playing Super Mario, you can input the picture on the screen, turn the picture into a feature vector with a convolutional layer, and then mapping the feature to an output vector with fully connected layers. There will be three-dimensional output as Super Mario has 3 actions in total, which are “left”, “right” and “up”. The outputs are the Q values of there actions, agent will take the action with highest expected reward(Q value) according to the principle of Q-learning and the state updates, it will keep repeating this step to find the best action in the given state until game over.

7. Shortcomings of RL

In RL, the destination of an agent is to get maximum total reward, therefore if we want to solve some realistic problem while the agent reach its target, we have to set a good reward function[2]. However, it can be knotty sometimes. For example, if the agent is playing chess, it will be rewarded if it capture the opposite chess pieces, so that the agent maybe try to capture all the opposite chess pieces to get high total reward, but this is getting away from our goal which is to capture the check, maybe we need to dampen the reward when agent capture other pieces rather than the check. Or we could juat let the agent to learn and update a appropriate reward function. However, even we set a good reward function, it can be local optima[6]. It just like if you have taste a good restaurant, you may never go to another restaurant, even other restaurant has more delicious food. That is local optima. Other than that, the effciency of RL is still very slow today, it always need a lot of samples and very time consuming[6].

In most of cases, the agent do not know the target at the first, it just know to do the action which could get good reward. Taking Tic-Tac-Toe for an instance, the agent is not told the rule that who first to connect 3 markers into a line will win the game, it just keep attmpting any actions randomly for many times until it win once that it knows the rule. This initial step which to understand the rule costs some time, maybe we could just give the agent some reasonable priori conditions rather than a random one so that to save some time.

In addition, RL is normally used in game especially easy game such as Go game or Tic-Tac-Toe, because the environment of a game is unchanged in most of cases and those easy game have relatively small number of states. It is hardly to see RL used in memory and reasoning games, because it has Markov property, which means the reward rt+1 at time t+1 is only according to the state st and action at at the last time t, you could imagine that when you play a memory game, you will sometimes need to use the information from last or earlier state which you have undergone, that might be a problem for agent in RL, because it need to do many actions to back to that state and it might cost a lot of time especially when the state set is very large, it is pretty difficult for agent to find the right state.

The auther’s personal suggestion is record each state and add an alternative choice which could return to any recorded states. This might cause more amount of calculation but it could prevent agent fail to reach the right state.

8. Conclusion

Although RL has got a great success, such as AlphaGo, AlphaZero and OpenAI-Five, it still has some problems and limitations which wait to solve. The low sample efficieny, local optima, overfitting and so on. And you will find that most of RL algorithm are used in game area, it is also important to develop a wider field.

References:

[1] Max Fischer. Using Reinforcement Learning for Games with Nondeterministic State Transitions. 2019.
[2] Chen A, Dewan T, Trivedi, et al. The Use of Reinforcement Learning in Gaming The Breakout Game Case Study. 2020.
[3] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction. A Bradford Book, 2018.
[4] Li Y. Deep Reinforcement Learning: An Overview[J]. 2017.
[5] João Crespo and Andreas Wichert. Reinforcement learning applied to games. 2020.
[6] Alex Irpan. Deep Reinforcement Learning Doesn't Work Yet. 2018.