An Exploration on Reinforcement Learning Algorithm

N. Vimala, Assistant Professor,
KanchiMamunivar Government Institute for Post Graduate Studies and Research (Autonomous)
Puducherry

Abstract - Nowadays researches are carried on the machine learning concept and trying to make the computer to act as a human with multiple capabilities of knowledge. Reinforcement learning is one of the classifications of the machine learning categories such as supervised learning and unsupervised learning. Various algorithms where used for reinforcement learning on the basis of on-policy and off-policy on temporal difference algorithm (TDA).The major algorithms Monte Carlo, dynamic programming, Q-Learning algorithm, Sarsa, and function approximation method are taken for exploration. Monte Carlo method for reinforcement learning learns directly from the experience without the knowledge of Markov Decision Processing (MDP). Dynamic programming solves complex problem by breaking into sub problem and combine the sub problems toa overall problem. Q-Learning is valued-based learning algorithm which is used to find the optimal action-selection policy using a function; the function finds the best action for each state of problem. SARSA algorithm is based on the policy which derives a different policy and finally function approximation helps to find the value of a state or an action without prior knowledge or past experience. Various problems are taken with research point of view are analyzed and applied on the algorithms and they are optimized.

Keywords: Reinforcement learning, TDA, Monte Carlo, dynamic programming, Q-Learning, Sarsa, Function approximation method, MDP

1. INTRODUCTION

Reinforcement learning (RL) is concerned with how software agents have to take actions in an environment in order to maximize some of the notion of cumulative rewards.Reinforcement learning, the first dimension is a trial and error problem why because it is like a child trying to stand by itself without anyone's help, it does not know what is standing and how to stand, it tries many times by itself but finally stands, and gets the reward.

The second dimension of reinforcement learning concerns with the problem of optimal control and the solution using value functions and dynamic programming techniques [1]. The first and second dimension put together in 1980 emerged as reinforcement learning, RL the founding father is Richard Sutton.

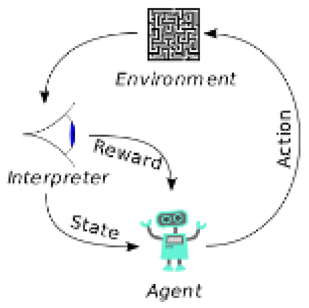


Figure 1. Reinforcement Learning[2]

The temporal difference (TD) method is a model that is designed based on the prediction made by the previous occurrence of the model and the model is further modified to the current occurrence of the situation. Hence this model is termed as secondary reinforcement learning as it has been paired with the primary reinforcement learning as it has the similar properties of reinforcement learner. This method is similar to backpropagation in neural network as the training data are trained and the weights are changed until it reduces the error and gets the exact value. The TD models are modified until a model for prediction is defined. This type of learning defined the animal psychology and it was recognized afterwards in Samuel's checkers players game.The temporal difference and optimal control where brought together in 1989 and was led to the developed of Q-learning.

Q-learning was introduced by Chris Watkins[3] in 1989.Q-means Quality learning. Q-learning isa model-free reinforcement learning algorithm. The goal of Q-learning is to learn aoff policy using greedy approach, which tells a software agent what action to take under what environment. It does not require a model of the environment, and it can handle problems with stochastic transitions and rewards, without requiring adaptations.

State-action-reward-state-action(SARSA) was proposed by Rich Sutton and is an algorithm for learning a Markov decision process policy. Rummery and Niranjan proposed with the name ‘Modified Connectionist Q-Learning’ (MCQ-L)[4]

A SARSA agent interacts with the environment and updates the policy based on actions taken, hence this is also known as an on-policy learning algorithm. The Q value for a state-action is updated by an error, adjusted by the learning rate alpha. Q values represent the possible reward received in the next time step for taking action a in state s, plus the discounted future reward received from the next state-action observation. This method is similar to backpropagation in neural network. Watkin's Q-learning may be applied to SARSA[5].

The state-action which represents a new feature that occurs in the problem increases with no prior knowledge and for every state-action, the tabular size increases it remains unsuitable. The main idea is to find the value of state-action where the agent goes the optimum path and by collecting the maximum rewards. Due the shortcoming in the tabular method. Each and every state-action which is a new feature, a value of estimation is calculated which is not the exact true value of state-action. The computation of the approximation is called the Function Approximates. There are many different types of Function Approximates performs computation and do not perform learning.

2. ALGORITHMS OF RL

2.1. Temporal Difference Algorithms

The temporal difference [6] is an agent learning from environment through episodes with no prior knowledge of the environment. There are three algorithms TD(1), TD(0) and TD(λ)

2.1.1. TD(1) Algorithm

Each and every episode is similar to Monte Carlo.

1. Take a random walk in an environment and accumulates the immediate reward(R) at a given point (time,t+1) with discount(γ)

2. Calculate the discount sum, Dt

$$D_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + .. \gamma^{T-1} R_T$$

3. Episode state value V(St) is calculated adding the state value and subtracting the sum of discount Dt from the prior estimate value, the TD error and adjust with α.

$$V(S_t)=V(S_t) + \alpha\,(D_t - V(S_t))$$

4. Steps 1, 2, 3 are repeated for each episode and the value is estimated based on the previous episode value.

This algorithm is also called Monte Carlo update.

2.1.2. TD (0) Algorithm

Algorithm TD (0) is similar to TD(1), the only difference is in calculating the state value of the episode, only calculates the immediate reward R(t+1) and plus the discount of the estimated value of only one step ahead V(St+1)

$V(S_t)=V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

This is called as bootstrapping. TD(0) is more bias then TD(1)

2.1.3. TD(λ) Algorithm

TD(λ) algorithm is implemented at the end of the episode in two different views either forward or backward. In backward view the state value is credited based on eligibility traces (ET) and λ and γ are used to discount the traces.

$$E_0(s) = 0$$

$$E_t(s) = \gamma \lambda E_{t-1}(s) + 1(S_t = s)$$

The value is updated based on ET and TD error δt

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

$$V(s) = V(s) + \alpha \delta_t E_t(s)$$

2.2 Dynamic Programming Algorithm

Given a complete model with an environment, MDP of all the specifications. DP can successfully find an optimal policy for the agent to follow.

1.Break the problem into sub-problem that can be solved

2.Solutions to the sub-problems are taken together and find the optimal solutions to the overall problem

3.To solve the MDP, Evaluate the policy to find out how good the arbitrary policy is by calculating the state-value function.

4.Improvement in the policy, if the newly calculated state-value function is high, this policy will be better and shows improvement in the policy.

5.Policy iteration = policy evaluation +
policy improvement

6.Finallygives the optimal policy.DP works well only when the model of theenvironment is known and is a model based RL.

2.3 Q-Learning Algorithm

Q-learning is an off policy RL algorithm that seeks to find the best action to take in the given current state[7]. It'soff-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy is not needed. More be more specific, q-learning seeks to learn a policy that maximizes the total reward.

1.Create a q-table or a matrix of [state, action] and initialize with zeroes

2.The agent considers the q-table as reference table and selects the best action based on the q-value.

3.The agent interacts with the environment and updates the state-action value.

3.1.The agent interacts the environment by referring the table value and chooses the maximum value to perform the action called **exploiting**.

3.2.It interacts by choosing the random value in order to perform the action called **exploring.**By which the agent explores by discovering new states.

4.Updates the table for new state and action and ends when an episode is done.

4.1.Agent starts in a state (s1) takes an action (a1) and receives a reward (r1)

4.2.Agent selects action by referencing Q-table with highest value (max) OR by random (epsilon, ε)

4.3.Update q-values

$$\mathbf{Q(s_t,a_t)=Q(s_t,a_t)+\alpha(r_{t+1}+\gamma max_a Q(s_{t+1},a) - Q(s_t,a_t))}$$

5.Finally it converges the q-value and learns the optimal q-value or q-star(Q*)

5.1 Convergence takes place by adjustment made on by the q-values based on the difference between the discounted new values and the old values. We discount the new values using gamma and we adjust our step size using learning rate (α).

Q-learning helps the agent to select the best actions that maximize the total reward.

2.4 SARSA Algorithms

In SARSA algorithm, the learning agent can have both on-policy and off-policy. It is slightly similar to Q-learning which is only off-policy. The only difference in the algorithm is the updation value of Q.

$$\mathbf{Q(s_t,a_t)=Q(s_t,a_t)+\alpha(r_{t+1}+\gamma Q(s_{t+1},a_t) - Q(s_t,a_t))}$$

the update equation for SARSA depends on the current state, current action, reward obtained, next state and next action. This observation lead to the naming of the learning technique as SARSA which symbolizes the tuple (s, a, r, s', a').

1.Intialize Q(s,a), for all $s \in \delta$, $a \in A(s)$ and

Q(terminal-state) = 0

2. For each episode repeat

Initialize S

Choose A from S using policy derived

```
from Q
        For each step of episode repeat
        Take action A, observe R, S’
        Choose A’ from S’ using policy
derived from Q
        $Q(S,A)=Q(S,A)+\alpha(R+\gamma Q(S’,A’)-$
        $Q(S,A))$
S=S’; A = A’
Until S is terminal
```

2.5 Function Approximation Algorithms

```
1:Initialize w = 0, k = 1
2: loop
3: Sample k-th episode (sk,1, ak,1,rk,1,sk,2, . . . ,sk,Lk ) given π
4: for t = 1, . . . ,Lk do
5:     if First visit to (s) in episode k then
6:         Gt(s) = PLk j=t rk,j
7:         Update weights:
8:     end if
9: end for
10: k = k + 1
11: end loop
```

Function approximation idea – generalization and efficiency • Gradient descent approximation to estimate value/Q functions • gradient descent to optimize the optimal Q-function directly

3. Comparison between the Algorithms

All the algorithms TD, DP, Q-Learning, SARSA and Function Approximation are taken for comparison and are tabulated based on the type of action it performs, the policy on or off and the value used for calculation to proceed with the algorithm.

Algorithm	Type of action	Policy	Value used
TD	State-action	Off	Episode value
DP	Environment with MDP Tabular method	Off	Max. /Min
Q-Learning	State-action-reward-state	Off	Q-value
SARSA	State-action-reward-state-action	On	Q-value
Function Approximator	State-action	Off	Q-function

Table 1 – Comparison on the Algorithms

In Q-learning algorithm is the only on-policy algorithm remaining are off-policy, TD is based on episode value, the rewards gained at the end of the episode based on the previous episode and reward gained can be altered and function approximation is done in TD. The remaining algorithm is based on Q-values.

4. Experimental Implementation

The n-armed bandit problem are taken for experimental analysis on the algorithms. This problem in which a fixed limited set of resources my be allocated between competing choices in a way that maximizes their expected gain, when each choice's properties are only partially known at the time of allocation, and may become better undertook as time passes or by allocating resources to the choice[8].

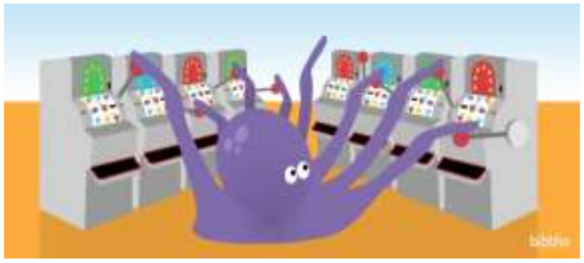
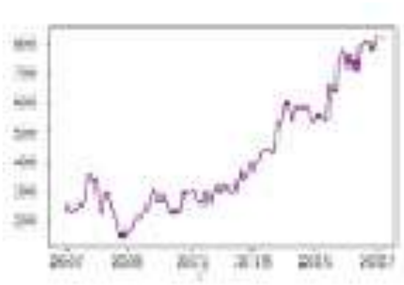


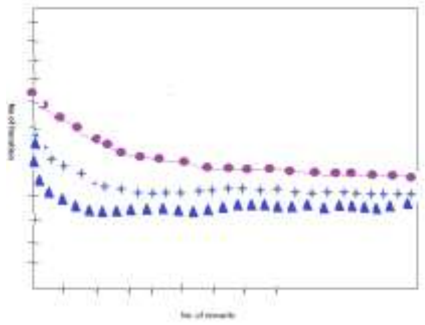
Figure 2 - Multi-armed Bandit Problem[9]

In DP algorithm based on the number of rewards gained and the number of exploration the chart is constructed.



Result - I

In TD algorithm is given by No. of rewards Vs. No.of iteration for n-armed bandit problem is given by



Result - II

In Q-learning and SARAS the chart is constructed based on the comparison between the algorithmand it was found that the SARSA performance was better than Q-Learning.

5. Conclusion and Future Enhancement

Now we are all still at the beginning of RL. This must be the next revolution in the field of machine learning, even though it is trail-and-error basis this provides an interesting spin introduced in the machine learning and making further future glow. This RL makes the machine to learn gradually and finally learns in a very speedy way with lot of rewards by exploring and exploitation. Many new algorithms can be designed and for further optimization.

References

1.VanOtterlo, M.; Wiering, M. (2012). Reinforcement learning and markov decision processes. Reinforcement Learning. Adaptation, Learning, and Optimization. 12. pp. 3–42. doi:10.1007/978-3-642-27645-3_1. ISBN 978-3-642-27644-6.

2.https://en.wikipedia.org/wiki/File:Reinforcement_learning_diagram.svg

3.Watkins, C.J.C.H. (1989), Learning from Delayed Rewards(PDF) (Ph.D. thesis), Cambridge University

4.Online Q-Learning using Connectionist Systems" by Rummery&Niranjan (1994).

5. Wiering, Marco; Schmidhuber, Jürgen (1998-10-01). "Fast Online Q(λ)" (PDF). Machine Learning. 33 (1): 105–115. doi:10.1023/A:1007562800292. ISSN 0885-6125.

6.Sutton, Richard S. "Learning to Predict by the Methods of Temporal Differences." *Machine Learning*, vol. 3, no. 1, 1988, pp. 9–44., doi:10.1007/bf00115009.

7.Wikipedia "Q-Learning"

8. Berry, Donald; A Fristedt, Bert(198). Bandit problems: Sequential allocation of experiments, Monographs on Statistics and Applied Probability, London; Chapman & Hall, ISBN 978-0—412-24810-8

9.https://miro.medium.com/max/3088/1*5q0Mihf29fftuXpKWWX2uA.png