Approximate Q-Learning: An Introduction

Deepshikha Pandey
Department of Computer Science and Engineering
Jaypee Institute of Engineering & Technology, Guna, INDIA
deepshikha1213@rediffmail.com

Punit Pandey
Department of Computer Science and Engineering
Jaypee Institute of Engineering & Technology, Guna, INDIA
pandey02_bit@rediffmail.com

***Abstract*—This paper introduces an approach to Q-learning algorithm with rough set theory introduced by Zdzislaw Pawlak in 1981. During Q-learning, an agent makes action selections in an effort to maximize a reward signal obtained from the environment. Based on reward, agent will make changes in its policy for future actions. The problem considered in this paper is the overestimation of expected value of cumulative future discounted rewards. This discounted reward is used in evaluating agent actions and policy during reinforcement learning. Due to the overestimation of discounted reward action evaluation and policy changes are not accurate. The solution to this problem results from a form Q-learning algorithm using a combination of approximation spaces and Q-learning to estimate the expected value of returns on actions. This is made possible by considering behavior patterns of an agent in scope of approximation spaces. The framework provided by an approximation space makes it possible to measure the degree that agent behaviors are a part of ("covered by") a set of accepted agent behaviors that serve as a behavior evaluation norm.**

Keywords- Reinforcement Learning, Q-learning Algorithm, Overestimation of Q-Values, Approximate Q-learning Algorithm.

I. INTRODUCTION

Q-Learning algorithm is a model free off policy reinforcement learning algorithm. In reinforcement learning selection of an action is based on the value of its state using some form of updating rule. An agent chooses that action which has maximum reward obtained from its environment. The reward signal may be positive or negative depends on the environment. Two different forms of Q-learning [2] method are considered in this article as a part of study of reinforcement learning in real-time. First, a conventional Q-learning method is considered, where a Q-value evaluates whether things have gotten better or worse than expected as a result of an action selection in the previous state. A temporal difference (TD) error term delta (δ) is computed to evaluate an action previously selected. An estimated Q-value in the current state is than determined using δ.Agent actions are generated using the maximum Q-values. Agent actions with lower TD error tend to be favored. The second form of Q-learning method is defined in context of overestimation. In this new form of Q-learning method we include Rough set. The contribution of this article is to introduce an Approximate Q- learning method to optimize the performance of an agent.

This paper has the following organization. A brief introduction about the Reinforcement Learning is given in section II. Basic concepts from rough set are briefly presented in section III. Section IV describes about the conventional Q-learning reinforcement method. Section V describes the problem related to Q-Learning. Approximate Q-learning method based on rough set is presented in section VI. Some result & plots based on the implementation of Q-learning Algorithm is given in section VII.

II. REINFORCEMENT LEARNING

Reinforcement learning (RL) is a goal directed learning methodology that is used to learn the agents. In Reinforcement learning [1,3,5,6,7,8] the algorithm decide what to do and how to map situations to actions so that we maximize a numerical reward signal. The learner is not advised which actions to take, but instead it discover which actions provide the maximum reward signal by trying them. Reinforcement learning is defined by characterizing a learning problem. Any algorithm that can able to solve the defined problem, we consider to be a reinforcement learning algorithm. The key feature of reinforcement learning is that it explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment. All reinforcement learning agents [5,10] have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments. In reinforcement learning agent prefer to choose actions that it has tried in the past and found to be effective in producing maximum reward. The agent has to exploit based on what it already knows in order

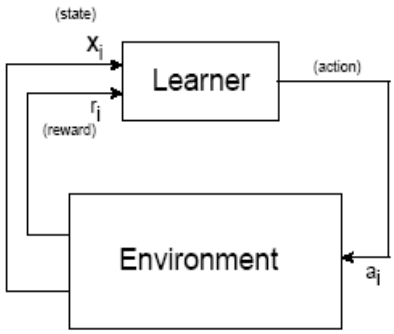

Fig .1:.Reinforcement learning

978-0-7695-3977-5/10 $26.00 © 2010 IEEE
DOI 10.1109/ICMLC.2010.38

to obtain reward and at the same time it also has to explore in order to make better action selection in future. Reinforcement learning has four elements [7,8] policy, reward function, value function and model of environment. Model of the environment is an optional element because reinforcement learning also supports the model free algorithm like Q-learning. A policy for our agent is a specification of what action to take for every Input. In some cases policy may be a simple function or look-up table or sometime it can be a extensive computation. The policy is the core of a reinforcement learning agent because it alone is sufficient to take the decision on further action.

III. BASIC CONCEPT OF ROUGH SET

This section describes about rough set theory that provide a foundation for projecting rewards for actions by collections of agents. The rough set approach introduced by Zdzis law Pawlak [1, 3, 4, 9] provides a ground for concluding to what degree a set of equivalent behaviors is a part of a set of behaviors representing a standard. An overview of rough set theory and applications is given in [4, 9]. For computational reasons, a syntactic representation of knowledge is provided by rough sets in the form of data tables. . A data (information) table IS is represented by a pair (U, A), where U is a non-empty, finite set of elements and A is a non-empty, finite set of attributes (features), where a : U −>V a for every a $\in$ A. For each B $\subseteq$ A, there is associated an equivalence relation IndIS(B) such that IndIS(B) = {(x, x') $\in$ U2| $\forall$ a $\in$ B.a(x) = a(x')}. Let U/IndIS(B) denote a partition of U, and let B(x) denote a set of B-indiscernible elements containing x. B(x) is called a block, which is in the partition U/IndIS(B). For X $\subseteq$ U, the sample X can be approximated from information contained in B by constructing a B-lower and B-upper approximation denoted by B*X and B*X, respectively, where B*X = {x$\in$ U|B(x) $\subseteq$ X} and B*X = {x $\in$ U|B(x) $\cap$ X$\neq$ Φ }. The B-lower approximation B*X is a collection of sample elements that can be classified with full certainty as members of X using the knowledge represented by attributes in B. By contrast, the B-upper approximation B*X is a collection of sample elements representing both certain and possible uncertain knowledge about X. Whenever B*X is a proper subset of B*X, i.e., B*X $\subset$ B*X, the sample X has been classified imperfectly, and is considered a rough set.

IV. Q-LEARNING METHOD

Q-learning is a form of model-free reinforcement learning [2]. It works by incrementally updating the expected values of actions in states. For every possible state, every possible action is assigned a value which is a function of both the immediate reward for taking that action and the expected reward in the future based on the new state that is the result of taking that action [2,10]. This is expressed by the one-step Q-update equation.

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

Where Q is the expected value of performing action a in state s; s is the state vector; a is the action vector; r is the reward; α is a learning rate which controls convergence and γ is the discount factor. The discount factor makes rewards earned earlier more valuable than those received later. This method learns the values of all actions, rather than just finding the optimal policy. This knowledge is expensive in terms of the amount of information that has to be stored, but it does bring benefits. Q-learning is exploration insensitive, any action can be carried out at any time and information is gained from this experience

Q-learning Algorithm:
Q-learning is a reinforcement learning method where the learner builds incrementally a Q-function[2,10] which attempts to estimate the discounted future rewards for taking action from given states. The output of Q-function for state s and action a is shown in equation (1). Q-values are usually stored in look up table. Formally a Q-learning can be described as follows:

Algorithm:
Initialize Q(s, a) arbitrarily
Repeat (for each episode)
Initialize s
Repeat (for each step of episode):
Choose a from s using policy derived from Q
Take action a, observe r, s'
Q(s, a) = Q(s, a) + α [r+ γ maxa' Q(s', a') - Q(s, a)]
s← s' ;
Until s is terminal

V. OVERESTIMATION IN Q-LEARNING

Thrun and Schwartz[10,11] describes the problem in Q-learning. Q-Learning is used for learning optimal policies in Markovian sequential decision tasks. Q-Learning perform this by incrementally learning a function Q(s, a) which it uses to evaluate the utility of performing action *a* in state *s.* More specifically, assume the agent observes during learning that action *a* executed at state *s* resulted in the state *s'* and some immediate reward r_s^a . The whole observation is performed to update Q Value:

$$Q(s,a) = r_s^a + \gamma \max_{a'} Q(s',a') \quad (2)$$

Here γ is a *discount factor* (0< γ <1), used to give a preference for rewards reaped sooner in time. At any time, the Q-values suggest a policy for choosing actions—namely the one which, in any state, chooses action *a which* maximizes Q(s, a). It has been shown that repeated application of this update equation eventually yields Q-values that give rise to a policy which maximizes the expected cumulative discounted reward. However, such results only apply when the Q-values are stored precisely, *e.g.*, by a look-up table. Let us assume that the currently stored Q-values, denoted by Q_{approx}, represent some implicit

target values Q_{target}, corrupted by a noise term $Y_{s'}^{a'}$ which is due to the approximated values.

$$Q^{approx}(s',a') = Q^{target}(s',a') + Y_{s'}^{a'} \qquad (3)$$

Here the noise is modeled by the family of random variables $Y_{s'}^{a'}$ with zero mean. Clearly, this noise causes some error on the left-hand side of Eq. (2), denoted by the random variable Z_s.

$$Z_s = r_s^a + \gamma \max_{a'} Q^{approx}(s',a') - \left(r_s^a + \gamma \max_{a'} Q^{target}(s',a')\right)$$
$$= \gamma\left(\max_{a'} Q^{approx}(s',a') - \max_{a'} Q^{target}(s',a')\right) \qquad (4)$$

The key observation underlying our analysis is that *zero-mean noise* $Y_{s'}^{a'}$ may easily result in Z_s with *positive mean* i.e.

$$E[Y_{s'}^{a'}] = 0 \forall a' \xrightarrow{often} E[Z_s] > 0 \qquad (5)$$

Due to the approximation, some of the Q-values might be too small, while others might be too large. The max operator, however, always picks the largest value, making it particularly sensitive to overestimations. If several Q-values are alike and their error intervals overlap, one is likely to overestimate the correct Q-value for some action, which will make the max operator overestimate as well. In short, max causes overestimation because it does not preserve the zero-mean property of the errors of its operands.

VI. APPROXIMATE Q-LEARNING

This section introduces what is known as Approximate Q-learning (RQ) method. In fact, common variation includes additional factors varying the amount of credit assigned to action taken. The Approximate Q-learning method calculates Q-values as shown below:

$Q(s,a) = Q(s,a) + \alpha [r + \gamma^n [\max_{a'} Q(s',a') - (1-\gamma) r'] - Q(s,a)]$ (6)

Where 'n' is number of episodes and r' is rough inclusion which is calculated as follows:

$$r' = \frac{1}{|B|}\sum V(B_a(x), B_*D) \qquad (6)$$

Where V(Ba(X), B*D) is calculated as:

$$V(B_a(x), B_*D) = \frac{|B_a(x) \cap B_*D|}{|B_*D|}$$

In Q-learning as we move from one episode to another, overestimation increases due to the max operator on Q (s', a'). 'γ' also enhance these overestimated values. This systematic overestimation affects the learning ability of agent. Therefore, in Approximate Q-learning we make two changes:

- We subtract a factor: (1- γ) r' from max Q (s', a'). This modification helps us to bring the overestimated values closer to standard values. In other words, it helps to minimize the error from approximated Q-values.
- We replace γ by γ^n. This modification helps us to improve performance of Q-learning by decreasing overestimated Q-values.

In Approximate Q-learning, reference or standard used is average rough inclusion (r') which implicitly measures to what extent action ***a*** has been selected relative to state **s**.

The Approximate Q-Learning algorithm is as follows

Modified Algorithm:

Initialize Q(s, a) arbitrarily
Repeat (for each episode)
Initialize s
Repeat (for each step of episode):
Choose a from s using policy derived from Q
Take action a, observe r, s'
$Q(s,a) = Q(s,a) + \alpha [r + \gamma^n [\max_{a'} Q(s',a') - (1-\gamma) r'] - Q(s,a)]$
S← s';
Calculate r' with help of equation 6
Until s is terminal

VII. CONCLUSION

The results reported in this paper suggest that average rough inclusion considered in the context of an approximation space used to minimize the error from approximated Q-values.

The results of experiments reported in this paper indicate that the Approximate Q-learning method performs better than the conventional Q-learning method. We have implemented both form of Q-Learning Algorithm and results is shown by plots (see fig 3 & fig4) in which upper line shows the performance of Approximate Q-learning and the lower line shows the performance of conventional Q-learning. The lower line shows that our learning will decrease and it will collapse after several episodes. The upper line describes about the performance induced by the

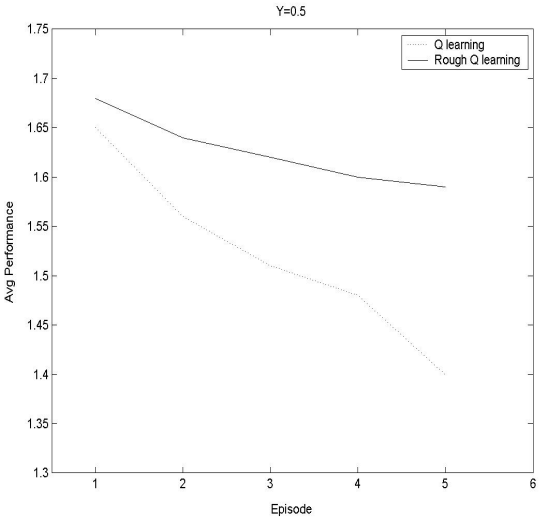

Fig 2: Average Performance For Q-learning Methods for γ =0.5

rough-inclusion that is better than the lower line. Ultimately, it is important to consider ways to evaluate the behavior of an intelligent system as it unfolds. Behavior decision tables constantly change during the life of an intelligent system because of changing proximate causes and changing rewards of corresponding action responses. As a result, there is a need for a cooperating system of agents to gain, measure, and share knowledge about changing behavior patterns. Part of the future work in this research will include a study of various reinforcement algorithms in the context of approximation spaces useful in on-line learning by agents.

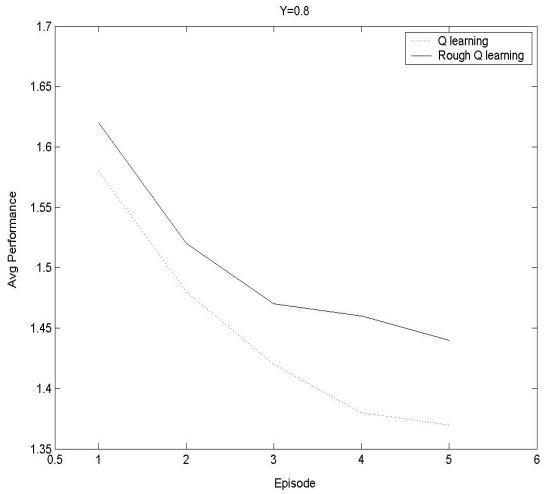

Fig 3: Average Performance For Q-learning Methods γ=0.8

After observation of plots we can say that as a consequence of this overestimation, as well as of the discounting typically used in reinforcement learning, performance of Q-learning is constantly decreases and the learner is expected to fail to learn an optimal policy after several episodes.

REFERENCES

[1] J.F.Peters. "Rough Ethology: Towards a biologically inspired study of collective behavior in Intelligent System with Approximation Spaces", Transactions on Rough Sets- III, LNCS 3400, 153-174, 2005, Springer-Verlag Berlin Heidelberg 2005.

[2] Christopher J.C.H. Watkins and Peter Dayan ,"Technical Note Q,-Learning", Centre for Cognitive Science, University of Edinburgh, Scotland Machine Learning, 8, 279-292 (1992)

[3] J.F. Peters, C. Henry, S. Ramanna, "Rough Ethograms: Study of Intelligent System Behavior", New Trends in Intelligent Information Processing and Web Mining (IIS05), Gda´nsk, Poland, June 13-16 (2005),117-126.

[4] Z. Pawlak, "Rough sets", International J. Comp. Inform. Science, 11, 1982, 341–356.

[5] J.F.Peters,K.S.Patnaik,P.K.Pandey,D.Tiwari, "Effetc of temperature on swarms that learn", In Proceeding of IASCIT-2007,Hyderabad,INDIA

[6] P.K.Pandey,D.Tiwari, " Temperature variation on Q-Learning",In Proceeding of RAIT in FEB 2008,ISM Dganbad

[7] L.P. Kaelbling, M.L. Littman, A.W. Moore, "Reinforcement learning: A survey", Journal of Artificial Intelligence Research, 4, 1996, 237-285.

[8] R.S. Sutton, A.G. Barto, "Reinforcement Learning: An Introduction ", (Cambridge, MA: The MIT Press, 1998).

[9] L. Polkowski, Rough Sets. Mathematical Foundations. (Heidelberg: Springer-Verlag, 2002).

[10] C. Gaskett, Q-Learning for Robot Control. Ph.D.Thesis, Supervisor: A.Zelinsky, Department of Systems Engineering, The Australian National University, 2002.

[11] Thrun. S.and Schwartz.A.(1993),Issues in using function approximation for reinforcement learning, in Proceeding of the 1993 Connectionist Models Summer School,Erblaum Associates.Nj.