Contemporary Engineering Sciences, Vol. 1, 2008, no. 1, 41 - 49

Q Learning in Context of Approximation Spaces

K. S. Patnaik

Dept. of Computer Science and Engineering
Birla Institute of Technology
Ranchi - 835215, India
ktosri@rediffmail.com

Shamama Anwar

Dept. of Computer Science and Engineering
Birla Institute of Technology
Ranchi - 835215, India
shamama3@gmail.com

Abstract

This paper introduces an approach to reinforcement learning by cooperating agents using a variation of the Q-learning method. Q-learning is a model free method i.e. in this method agent does not need to predict future condition. The framework provided by approximation space makes it possible to minimize the overestimation caused by approximated Q-values. Due to overestimation learning capability of the algorithm is not consistent. It is observed that under this condition the ability to take a particular action is decreased. Therefore, by using the Rough Q-learning method the performance of the algorithm increases. This is shown by comparing the average Q values for Q learning and Rough Q learning by means of plots.

Keywords: Q learning, Rising Q problem, Rough Sets, Rough Q Learning

1. Introduction

Q-learning is a form of model-free reinforcement learning [1] [2] [5]. 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[1][7]. This is expressed by the one-step Q-update equation

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

where Q is the expected value of performing action a in state s, 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

2. Q Learning Architecture

An overview of the architecture for Q learning is given in Fig 1 below [4]. The Elementary parts of the Q learning architecture are:

World: Q learning is based on model free mode of behavior i.e. the environment is continuously changing. Agent does not need to predict the future state [2].

Policy: The policy block represents a mapping from each state s and action a, to the probability of taking the action a when in state s [5].

Return Predictor: The Return predictor is the measure of the long term cumulative reward and is a function of action as well as state.

$$\mathbf{return(s,a) = E\{\sum \gamma^t\, r_{t+1} \mid s_0 = s,\ a_0 = a\}}$$

where γ is the discount factor and $0 \le \gamma \le 1$.

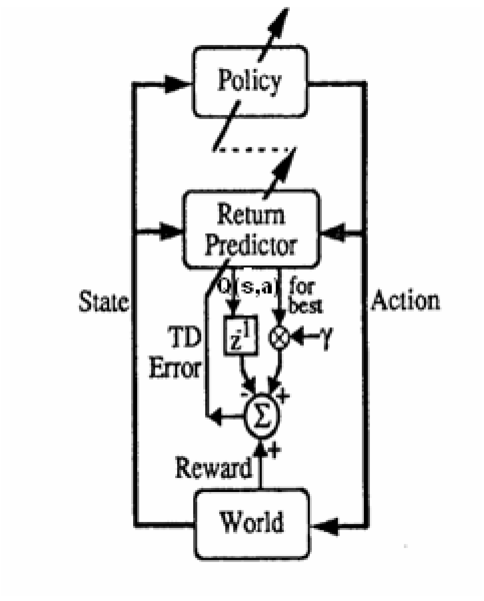

Fig 1: Q Learning Architecture

There are two outputs of the return predictor, one is the best predicted return for the state and the other is the return of the action actually selected. The best predicted return is then multiplied by gamma, which is done in the symbol indicated by $\otimes$. The output is **[γ max Q(s',a')].** z^{-1} indicates a one-time-step delay. The output of $\otimes$ and the actual Q(s,a) values are subtracted yielding **[γ max Q(s',a') – Q(s,a)].**

Reward: The reward r is the added to the output giving **r + [γ max Q(s',a') – Q(s,a)]**. Both these calculations are done in the ∑ block. Finally the return predictor and the current state influences the policy block and hence influence the next state to be selected and the final Q(s,a) equation is

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

3. The Q Learning Algorithm

```
Input : States, $s \in S$, Actions $a \in A(s)$, Initialize Q(s,a),$\alpha$, $\gamma$, $\pi$ to an
        arbitrary policy (non-greedy)
Output: Optimal action value Q(s,a) for each state-action pair
while True do
    for ($i = 0$; $i \leq$ # of episodes; $i++$) do
        Initialize s
        Choose a from s, using policy derived from Q
        Repeat(for each step of episode):
        Take action a; observe reward, r, and next state, s'
         $Q(s,a) = Q(s,a) + \alpha [r+ \gamma [ \max_{a'} Q (s',a') – Q(s,a)]$
        $s \leftarrow s'$; $a \leftarrow a'$;
        until s is terminal
    end
end
```

4. Rising Q Problem

Thrun and Schwartz [9] observed a fundamental flaw in Q-learning. Q-Learning is a technique for learning optimal policies in Markovian sequential decision tasks. It does 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 $\mathbf{r}_s^a$. This observation is employed to update Q:

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

Here γ is a *discount factor* ($0< \gamma <1$), 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 $\mathbf{Y}_{s'}^{a'}$ which is due to the approximated values.

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

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

$$\mathbf{Z_s = r_s^a + \gamma \max_a Q^{approx}(s', a') – [r_s^a + \gamma \max_a Q^{target}(s', a')]}$$
$$\mathbf{= \gamma [\max_a Q^{approx}(s', a')- \max_a Q^{target}(s', a')]} \quad \mathbf{(4)}$$

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

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

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. The consequence of this overestimation, as well as of the discounting typically used in reinforcement learning, performance of Q-learning constantly decreases and the learner is *expected* to fail to learn an optimal policy even after several episodes.

5. Rough Sets

This section briefly presents some fundamental concepts in rough set theory that provide a foundation for projecting rewards for actions by collections of cooperating agents. The rough set approach was introduced by Zdzis law Pawlak [6][8] 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 [8].

A data (information) table IS is represented by a pair *(U, A),* where *U* is a non-empty, finite set of objects called the universe and *A* is a non-empty, finite set of attributes (features), where $a : U \rightarrow V_a$ for every $a \in$ A. For each B $\subseteq$ A, there is associated an equivalence relation $IND_A(B)$ such that $IND_A(B) = \{(x, x') \in U^2 | \forall a \in B.\ a(x) = a(x')\}$ where $IND_A(B)$ is called the B-indiscernibility relation. If $(x, x') \in IND_A(B)$, then objects x and x' are indiscernible from each other by attributes from B. For A = (U,A) and B $\subseteq$ A, X $\subseteq$ U, we can approximate X using only the information contained in B' by constructing the B-lower and B-upper approximation denoted by B_*X and B^*X, respectively, where $B_*X = \{x \mid [x]_B \subseteq X\}$ and $B^*X = \{x \mid [x]_B \cap X \neq \Phi\}$. 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.

6. Rough Q Learning

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

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

Where 'n' is number of episodes and $\bar{v}$ is the average rough inclusion which is calculated as :

$$\bar{v} = \frac{1}{|B|} \sum_{B_j(x) \in B} v(B_j(x), B_* D)$$

Where $v(B_j(x), B_*D)$ is calculated as:

$$v(B_j(x), B_* D) = \frac{|B_j(x) \cap B_* D|}{|B_* D|}$$

. In Q learning as we move from one episode to another episode overestimation increases due to the max operator applied to Q(s',a'). γ also inhance this overestimated values. This systematic overestimation affects learning ability. Therefore, in Rough Q-learning we make two changes:

- We subtract a factor: $(1-\gamma)\ \bar{v}$ from max Q (s', a'). This modification helps us to bring the overestimated values closer to standrad values. In other words, it helps to minimizes 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.

7. Rough Q Learning Algorithm

```
Input : States, $s \in S$, Actions $a \in A(s)$, Initialize $Q(s,a), \alpha, \gamma, \pi$ to an
        arbitrary policy (non-greedy)
Output: Optimal action value Q(s,a) for each state-action pair
while True do
    for ($i = 0$; $i \le$ # of episodes; $i + +$) do
            Initialize s
        Choose a from s, using policy derived from Q
        Repeat(for each step of episode):
        Take action a; observe reward, r, and next state, s'
         $Q(s,a) = Q(s,a) + \bar{v}\ [r + \gamma^n\ [\ max_{a'}\ Q\ (s',a') - (1-\gamma)\ \bar{v}\ ] - Q(s,a)]$
        s ← s'; a ← a';
        until s is terminal
    end
end
```

8. Conclusion

The plots given below are for Q learning vs. Rough Q Learning for different Gamma values (0.1, 0.5, 0.9, 1).

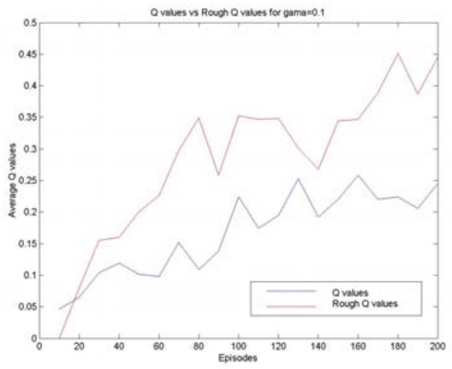

Fig 2: Q values vs Rough Q values for gamma=0.1

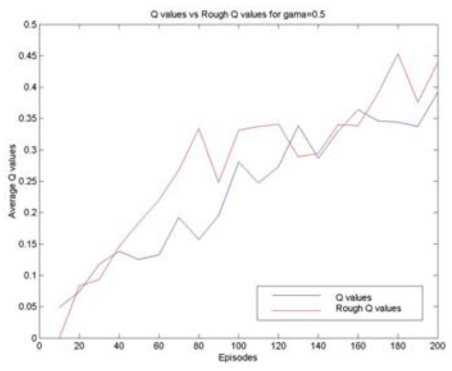

Fig 3: Q values vs Rough Q values for gamma=0.5

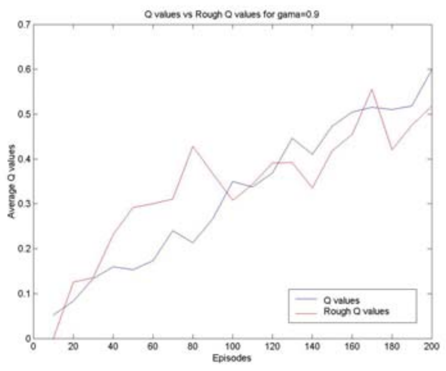

Fig 4: Q values vs Rough Q values for gamma=0.9

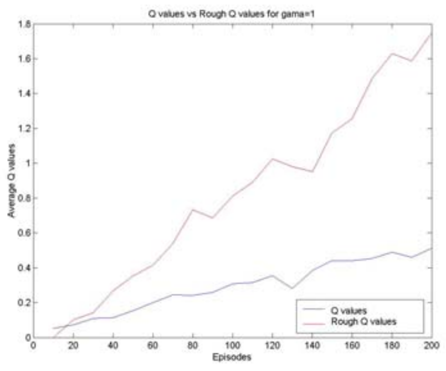

Fig 5: Q values vs Rough Q values for gamma=1

The results reported in this paper suggest that average rough inclusion considered is used to minimize the error from approximated Q-values and it indicates that the rough inclusion Q-learning method outperforms the conventional Q-learning method. The main results of this paper is shown by plots in which upper line shows the performance of Rough Q-learning and the lower line shows the performance of basic Q-learning. The lower line shows that the learning rate for Q learning becomes steady as the number of episodes increases and hence the learning capability does not increase.

Acknowledgement

The authors gratefully acknowledge the suggestions and constant support by Prof. J. F. Peters, Department of Electrical and Computer Engg., University of Manitoba, Winnipeg, Canada and insights from A Mustafi, Department of Computer Science and Engg., BIT, Ranchi, India.

References

[1] Christopher J. C. H. Watkins and Peter Dayan. Technical Note: Q-Learning. Machine Learning, 8 (1992), 279-292.

[2] Christopher J.C.H. Watkins. Learning from Delayed Rewards. PhD thesis, University of Cambridge,

[3] J. F. Peters, C. Henry, and S. Ramanna. Reinforcement Learning in Swarms that Learn. Published in the Proceedings of the 2005 IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, France, 19-22 September 2005, 400-406.

[4] J. F. Peters, C. Henry and S. Ramanna. Reinforcement Learning with Pattern Based Rewards. In proceeding of forth International IASTED Conference Computational Intelligence (CI 2005) Calgary, Alberta, Canada.

[5] 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, Springer-Verlag Berlin Heidelberg, 2005.

[6] J. Komorowski, Z. Pawlak, L. Polkowski, and A. Skowron. Rough Sets: A Tutorial.

[7] Richard. S Sutton. Reinforcement Learning Architectures. GTE Laboratories Incorporated, Waltham, MA

[8] R.S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. MIT Press, 1998.

[9] S. Thrun and A. Schwartz. Issues in Using Function Approximation for Reinforcement Learning. Proceedings of the Fourth Connectionist Models Summer School Lawrence Erlbaum Publisher, Hillsdale, NJ, 1993.

Received: January 3, 2008