A Comparative Analysis of Value Function Approximation Approaches in Reinforcement Learning

Zhiwei Han
Chair of Data Processing
Faculty of Electrical Engineering and Information Technology
Technical University of Munich
Arcisstr. 21, Munich, 80333
Email: hanzw356255531@icloud.com

Abstract*—In this paper, a comparison of three value function approximation based reinforcement learning algorithms is presented to evaluate their performance on solving continuous state space control problem. They are GQ(λ), $\ell 1$ norm regularized off-policy TD(λ) and online selective kernel-based TD(λ). For convenience, these algorithms will be called GQ(λ), RO-TD(λ) and OSK-TD(λ) in rest of the paper. The comparison standard concentrates mainly on the total reward, convergence and efficiency of the algorithms. The classic TD(λ) framework Q-Learning is applied and adapted by compared algorithms in parameter learning. Note, other RL framework choice like Sarsa(λ) could also be possible, but won't be discussed here. GQ(λ), which minimize the mean-squre projected bellman error(MSPBE) with gradient descend, is a very powerful and straightforward learning method under small scale or well-defined discrete space setting condition. However, because of the curve of dimensionality, the algorithm could be less computational efficient as the increasing number of state. To overcome this issue, one $\ell 1$ regularity term was inserted to the objective function in RO-GQ(λ) algorithm. The sparsity generated in the learned parameters helps significantly reduce the computational complexity of RO-GQ(λ) and therefore it has a better performance in these problems. Alternatively, OSK-TD(λ) provides another solution for sparsification, which is less computational complex and can be runned online. OSK-TD includes two procedures, a dictionary consists of feature vector is online generated and refreshed with a criteria for whether and how to add a new feature in dictionary. In the second step, a kernel selective function β(s) selects the best matched feature vector in dictionary for the selective kernel-based Q-function representation.

***Index Terms*—reinforcement learning, Q-Learning, function approximation, sparsification, slective kernel-based value function**

I. Introduction

Reinforecement learning (RL) approaches to classic predict and control problems base on dynamic programming (DP) and markov decision process (MDP) model, which proofed to be useful but computational expensive in high dimensional case. Sutton and Barto [1] proposed temporal difference (TD) learning framework and its variations as the extension of DP. One of its nice properties is the combination with value function approximation and it makes the algorithm possible to learn from complicated learning tasks.

But there are still two problems about value function approximation mainteined unsovled. In practical large scale problems, smart value function approximation strategy is alwags required to reduce the computational complexity because large scale RL problems e.g. Go within continous/action spaces suffer from the curse of dimensionality, which means the computational complexity increases exponentially with growing number of states and actions. This problem arises as a main subproblem in RL and is generally considered as the most important step in RL algorithm development. There are serveral value function approximation techniques developed in recent researches, e.g. linear function approximation [1] [2], regression tree [3] and kernel-method [4][5], which can deal with a specific set of problems. More generally, after deep learning [6] concept and deep neural network were introduced into RL, the perforemance has been significantly improved since Alpha Go of Deep Mind, a company owned by Google, beated one of the best human Go players, Lee Sedong. Furthermore, the learning agent equipped with deep Q-network (DQN) [7] based Q function Approximator was also turned out to be smarter than human player in some Atri-games. Regularization could be another solution to this problem, it plays not only an important role in preventing overfitting and is also considered as an ideal feature selection technique for in algorithms with value function approximation thanks to the sparsity generated by $\ell 1$ norm reguarization[8] [9]. In this paper, the tabular algorithm RO-TD(λ) with $\ell 1$ norm reguarization is compared with the algorithm OSK-TD(λ) [10], which equipped with another feature selection strategy, in term of the efficency of sparsification.

Another widely concerned issure is convergence of the value function approximation. As Tsitsiklis [2] pointed out that even though the linear value function approximation converges in most online learning problem setting, but an algorithm could still diverge if either of the conditions is not satisfied, when a off-line learning is perform, because states are sampled from distributions independent of the dynamics of the Markov chain of interst, or a nonlinear value function approximation is used. We will also point out this kind of view later in the experiment. It's a servere problem in RL application, however, because of the great needs, value function approximation was still applied in many off-line TD algorithms like Q-learning at that time while ignoring the potential risk of divergent value function approximation. Sutton [1][11] solved off-line learning problem of off-policy learning by introducing a new object function, *mean-squre projected bellman error* (MSPBE) and performing stochastic gradient mehtod an it.

In this paper, we consider up-to-date algorithms GQ(λ) [12], RO-GQ(λ) [9] based on the traditional tabular algorithm framework (without value function approximation), kernel-based feature selective algorithm OSK-TD(λ) [10] and their comparison. The rest of this paper is organized as flollows. In Section II, an introduction on MDPs as well as the TD learning framework is given. III presents the used techniques in the algorithms to be compared. An algorithmetic introduction and comparison between three algorithm is proposed in the section IV. And in section V the experimental result on the classic RL problem settings Mountain Car and Carto Car are provided to visulization the effectiveness of compared algorithm. At the VI section gives a conclussion and future work.

II. RL and Background

A. Framework and Notation

In this paper, we consider infinite markov decision process with discrete state space $\mathcal{S}$ and discrete action space $\mathcal{A}$. A MDP can be defined as tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, r, γ), where $\mathcal{P}$ represents the transition probability matrix and it is also defined as the one-step dynamics of enviroment by given state and action. [1] Given any state and action, s and a, the probability of each possible state transition to next state s', is a function $Pr : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$

$$p(s'|s, a) = Pr\{S_{t+1} = s'|S_t = s, A_t = a\} \tag{1}$$

Similarly, with any current state and action, s and a, and after taking action a the agent reaches the next state s', the expected value of the next reward of tuple (s, a, s') is a function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ and $\gamma \in (0, 1)$ is the discount factor.

$$r(s, s', a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s'] \tag{2}$$

In RL, main task is either directly to find a convergent value function through like dynamic programming etc., or indirectly approximate a value function (action-value function) through linear value function approximation etc.. A value function maps from each state to a number representation of value function obtained if a policy is exactly followed. Policy can also be analysed and evaluated according its sum of discounted value function. More formally,

$$V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\inf} \gamma^t r_{t+1}] \tag{3}$$

$$P^\pi(s, s') = \sum_{a\in\mathcal{A}} \pi(s, a)P(s'|s, a) \tag{4}$$

$$R^\pi(s) = \sum_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} \pi(s, a)p(s'|s, a)r(s, a, s') \tag{5}$$

If V^π, R^π are vectorized with length $|\mathcal{S}|$ and the dimension of P^π is $|\mathcal{S}| \times |\mathcal{S}|$, so the famous $Bellman\ Equation$ can be written in form,

$$V^\pi = TV^\pi = R^\pi + \gamma P^\pi V^\pi \tag{6}$$

, where T is the bellman operator and V^π is the fix point of bellman operation.

In rest of this paper, value function is replaced of action-value function, which maps each state and action, instead of only state, to a number representation, as we mainly focus on control problem.

B. Tabular version of Q(λ)

The basic idea of temporal difference learning is looking back to the previous state, and correcting the previous value or action-value function with experience (Bellman Error). The update of value function is defined as following,

$$V(s_t) = V(s_t) + \alpha\delta(s_t) \tag{7}$$

, where α is the learning rate and δ is the bellman error, which defined as

$$\delta(s_t) = R_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{8}$$

As one of the most important extension of TD algorithm, the off-policy learning procedure Q-Learning[13], whose update form is,

$$\begin{aligned} Q(s_t, a_t) = Q(s_t, a_t) + \alpha[R_{t+1}+ \\ \gamma \max_{a\in A} Q(s_{t+1}, a) - Q(s_t, a_t)] \end{aligned} \tag{9}$$

, directly approximates Q^*, the optimal action-value function, independent of the policy being followed.

Based on the TD method, eligibility keeps track on several prior states and how the action-value function should be updated when a new observation is revceived. More precisely, eligibility is a temporal record of occured events e.g. states or actions, and propagates the attenuated TD error back then finally updates the parameters of the previous states. The famous combinations of Q-learning and eligibility trace are Watkinss Q(λ) [13] and Pengs Q(λ) [14] for different update rules of eligibility trace.

The tabular version of Q(λ) is a special case of linear action-value function approximation based Q(λ). With a sparse feature vector ϕ for a given state and action pair, only one entry in learned parameter vector θ is going to be chosen when doing a vector multiplication. And this entry serves exactly as the Q-Function for the given feature vector.

$$Q(s,a) = \theta^\top \phi(s,a) \tag{10}$$

So a vectorise and parallel version implementation of traditional Q(λ) is possible and the update rule of TD error δ and eligibility trace e in form of Q-Learning from backward view is following,

$$\delta_t = R + \gamma \max_a \theta^\top \phi(s_{t+1}, a) - \theta^\top \phi(s_t, a_t) \tag{11}$$

$$e_t = \alpha\lambda e_t + \phi_t \tag{12}$$

Note, here we assume the transitions between states are ergodic, for a nonergodic case, replacing trace technique will be helpful. Finally, the update rule of parameter θ is given as following,

$$\theta_{t+1} = \theta_t + \alpha\delta e \tag{13}$$

III. GQ(λ)

One natural RL approach to learning task can be turned into an optimization problem, where we seek to minimize a predefined objective function by updating its parameter . The objective function should reflect how well (or how badly) the value function is approximated and ideally it can be written in form of weighted square norm as *mean square error* (MSE),

$$\text{MSE}(\theta) := ||V_\theta - V||_D^2, \tag{14}$$

where D is a weight matrix, whose diagnal entries are a weight scalars $\{d_1, d_2, ..., d_n\}$. In TD methods, the idea is to replace the MSE with *mean-square Bellman error* (MSBE) [15] [16] how closely the approximate value function satisfies the Bellman equation.

$$\text{MSBE}(\theta) := ||V_\theta - TV_\theta||_D^2. \tag{15}$$

This is the objective function used by the most important prior effort to develop gradient-descent algorithms. But the prior algorithms like TD, LSTD, and GTD don't sufficiently converge to the minimum of MSE as bellman operator typically is irrespective of value function approximation structure. Hence, it is not be able to represent value function approximation for any θ, although it follows the dynamics of Markov Decision Process (MDP).
By introducing the project operator in linear approximation architecture

$$\Pi = \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D, \tag{16}$$

which can project any value function to the subspace of linear architecture, we obtain the most accurate function approximator of v in the subspace of linear architecture. In other words

$$\Pi v = V_\theta \text{ where } \theta = \arg\min_\theta ||V_\theta - v||_D^2. \tag{17}$$

After the projection operator was introduced, all algorithms mentioned before find or converge to a fixpoint of the composed projection and Bellman operators, which is a value of θ^* such that

$$V_{\theta^*} = \Pi T V_{\theta^*} \tag{18}$$

and named as TD fixpoint. The objective function used in this paper is very similar to the *mean-square projected Bellman error* (MSPBE)

$$\text{MSPBE}(\theta) = ||V_\theta - \Pi T V_\theta||_D^2, \tag{19}$$

deviated from this fixpoint.

As an extension version of the TDC [11], GQ(λ) minimizes its objective function by iteratively updating the parameter with eligibility traces and multi-step reward. Its objective function is a λ-step weighted version of the MSPBE and denoted as

$$\begin{aligned}
&J(\theta)\\
&= \|Q_\theta - \Pi T_\pi^{\lambda\beta} Q_\theta\|_D^2\\
&= (T_\pi^{\lambda\beta}Q_\theta - Q_\theta)^\top \Pi^\top D \Pi (T_\pi^{\lambda\beta}Q_\theta - Q_\theta)\\
&= (\Phi^\top D(T_\pi^{\lambda\beta}Q_\theta - Q_\theta))^\top (\Phi^\top D\Phi)^{-1}\Phi^\top D(T_\pi^{\lambda\beta}Q_\theta - Q_\theta)\\
&= \mathbb{E}_\pi[\delta^{\lambda\beta}\phi]^\top \mathbb{E}[\phi\phi^\top]^{-1}\mathbb{E}[\delta^{\lambda\beta}\phi],
\end{aligned} \tag{20}$$

at the fix point θ [11] [12], where the identity $\Pi^\top D \Pi = D\top\Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D$ is used and $g^{\lambda\beta\rho}$, $\delta^{\lambda\beta\rho}$ stand for λ-return and λ-step TD error, respectively,

$$\begin{aligned}
&g_t^{\lambda\beta\rho} = r_{t+1} + \beta_{t+1}e_{t+1}\\
&+ (1-\beta_{t+1})[(1-\lambda_{t+1})\theta^\top\bar{\phi}_{t+1} + \lambda_{t+1}\rho_{t+1}g_{t+1}^{\lambda\beta\rho}]\\
&\text{and}\\
&\delta_t^{\lambda\beta\rho} = g_t^{\lambda\beta\rho} - \theta^\top\phi_t,
\end{aligned} \tag{21}$$

and,

$$\bar{\phi}_t = \sum_a \pi(s_t,a)\phi(s_t,a) \text{ and } \rho_t = \frac{\pi(s_t,a)}{b(s_t,a)} \tag{22}$$

After taking the negative gradient of objective function as the optimization direction,

$$\begin{aligned}
&-\frac{1}{2}\nabla MSPBE(\theta) = -\frac{1}{2}\nabla J(\theta)\\
&= -\mathbb{E}_b[(\nabla g^{\lambda\beta\rho} - \phi)\phi^\top]\mathbb{E}_b[\phi\phi^\top]^{-1}\mathbb{E}_b[\delta^{\lambda\beta\rho}\phi]\\
&\approx \mathbb{E}[\delta^{\lambda\beta\rho}\phi] - \mathbb{E}_b[\nabla g^{\lambda\beta\rho}\phi^\top]\omega,
\end{aligned} \tag{23}$$

where a second modifiable parameter $\omega \in \mathbb{R}^n$ is used to avoid caculating the inverse matrix and double sampling [1],

$$\omega \approx \mathbb{E}_b[\phi^\top\phi]^{-1}\mathbb{E}_b[\delta^{\lambda\beta\rho}\phi], \tag{24}$$

, finally the forward and backward view update of $GQ(\lambda)$ are given as the proof in [12]

$$\begin{aligned}
&\text{Forward:}\\
&\theta_{t+1} = \theta t + \alpha_{\theta,t}\left(\delta_t^{\lambda\beta\rho}\phi_t - \nabla g_t^{\lambda\beta\rho}\phi_t^\top\omega_t\right)\\
&\omega_{t+1} = \omega_t + \alpha_{\omega,t}\left(\delta_t^{\lambda\beta\rho} - \omega_t^\top\phi_t\right)\phi_t\\
&\text{Backward:}\\
&\theta_{t+1} = \theta_t + \alpha_{\theta,t}[\delta_t e_t - \kappa_{t+1}(e_t\top\omega_t)\bar{\phi}_{t+1}]\\
&\omega_{t+1} = \omega_t + \alpha_{\omega,t}[\delta_t e_t - (\omega^\top\phi_t)\phi_t],
\end{aligned} \tag{25}$$

where

$$\kappa_t = (1-\beta_t)(1-\lambda_t) \text{ and } e_t = \phi_t + (1-\beta_t)\lambda_t\rho_t\phi_{t-1}. \quad (26)$$

Then we give the pseudocode of GQ(λ)

Algorithm 1: GQ(λ)

```
1  Initialization $\boldsymbol{\theta}$ arbitrarily and $\boldsymbol{\omega} = 0$
2  repeat (for each episode)
3      Initialize $\boldsymbol{e} = 0$
4      $S \leftarrow$ initial state of episode
5      repeat(for each step of episode)
6          $A \leftarrow$ action selected by policy $b$ in state $S$
7          Take action A, observe next state, $S^{'}$
8          $\bar{\phi} \leftarrow 0$
9          For all $a \in \mathcal{A}(s)$:$\bar{\phi} \leftarrow \bar{\phi} + \pi(S^{'}, a)\phi(S^{'}, a)$
10         $\rho = \frac{\pi(S,A)}{b(S,A)}$
11         GQlearn($\phi(S, A)$, $\bar{\phi}$, $\lambda(S^{'})$,$\gamma(S^{'})$, $r(S, A, S^{'})$, $\rho$,
           $I(S)$)
12         $S \leftarrow S^{'}$
13     until $S^{'}$ is terminal;
14 until;
```

Algorithm 2: GQlearn($\phi, \bar{\phi}, \lambda, \gamma, R, \rho, I$)

```
1 $\delta \leftarrow R + \lambda\boldsymbol{\theta}^\top\bar{\phi} - \boldsymbol{\theta}^\top\phi$
2 $\boldsymbol{e} \leftarrow \rho\boldsymbol{e} + I\phi$
3 $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\delta\boldsymbol{e} - \gamma(1-\lambda)(\omega^\top\boldsymbol{e})\bar{\phi}))$
4 $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \alpha\eta(\delta\boldsymbol{e} - (\boldsymbol{\omega}^\top\phi)\phi)$
5 $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e}$
```

IV. RO-GQ(λ)

RO-GQ(λ) is a novel $\ell 1$ regularized off-policy convergent TD-learning algorithm with low computational complexity, which is able to learn sparse representations of value function by performing $\ell 1$ regularization in learning procedure. The algorithmic framework of RO-TD integrates two key schemas: off-policy convergent gradient TD methods and a convex-concave saddle-point formulation of non-smooth convex optimization ($\ell 1$ regularization term in objective function), which enables first-order solvers and feature selection using online convex regularization.

A. Proximal Gradient and Saddle-Point First-Order Algorithms

Minimization of a composite objective function is useful in enforcing various structure , such as sparsity in case of $\ell 1$ regularization on machine learning task. As well known property, $\ell 1$ regularization is able to generate sparse representation optimization problem but can also lead to not-differentiability of objective function. Here we introduce proximal gradient method[17], which can be shown to be an efficient solution to non-differentiable convex problem caused by subdifferential of $\ell 1$ regularization.
The basic form of proximal mapping associated with a convex function h is defined as:

$$prox_h(x) = \arg\min_u(h(u) + \frac{1}{2}||u-x||^2)l \quad (27)$$

In case of $\ell 1$ regularity term in objective function, which can be written in its general form $h(x) = \rho||x||_1(\rho > 0)$, the proximal operator is turned out to be a entry-wise soft-threshold operator denoted as $S_\rho(\cdot)$ and we obtain:

$$\begin{aligned} &prox_h(x)_i \\ &= S_\rho(x_i) \\ &= max(x_i - \rho, 0) - max(-x_i - \rho, 0), \end{aligned} \quad (28)$$

where i is the index and ρ is the threshold. When the optimization problem is

$$x^* = \arg\min_{x \in \mathcal{X}}(f(x) + h(x)), \quad (29)$$

where $f(x)$ is a convex and differentiable loss function and $h(x)$ is a convex but non-differentiable regularization term. After performing proximal gradient method, the iterative update of x is

$$x_{t+1} = prox_{\alpha_t h}(x_t - \alpha_t \nabla f(x_t)). \quad (30)$$

where α_t is a step-size factor. And meanwhile it's also cheap for computing.

B. Convex-concave Saddle-Point First Order Algorithms

C. Objective Function

D. RO-GQ(λ) Design

Algorithm 3: RO-GQ(λ)

```
   Let $\pi$ be some fixed policy of an MDP $M$, and let
   the sample set $S = s_i, r_i, s_{ii=1}^{'N}$. Let $\phi$ be some fixed
   basis.
1  repeat(for each step of episode)
2      Compute $\phi_t, \phi_t^{'}$ and TD error
       $\delta_t = (r_t + \gamma\phi_t^{'\top}\theta_t) - \phi_t^\top\theta_t$
3      Compute $y_t^\top A_t, A_t x_t - b_t$ in Equation (), ()
4      Compute $x_{t+1}, y_{t+1}$ as in Equation ()
5      Set $t \leftarrow t + 1$
6  until $t = N$;
7  Compute $\bar{x_N}, \bar{y_t}$ as in Equation () with $t = N$
```

V. OSK-TD

A. Online Kernel-Based Sparcification

B. Kernel Selective Function

C. Kernel-Based Action-Value Function

VI. Experiment

VII. Conclusion

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[2] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE transactions on automatic control*, vol. 42, no. 5, pp. 674–690, 1997.

[3] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 503–556, 2005.

[4] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, 2007.

[5] D. Ormoneit and Ś. Sen, "Kernel-based reinforcement learning," *Machine learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] J. Z. Kolter and A. Y. Ng, "Regularization and feature selection in least-squares temporal difference learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 521–528.

[9] B. Liu, S. Mahadevan, and J. Liu, "Regularized off-policy td-learning," in *Advances in Neural Information Processing Systems*, 2012, pp. 836–844.

[10] X. Chen, Y. Gao, and R. Wang, "Online selective kernel-based temporal difference learning," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 12, pp. 1944–1956, 2013.

[11] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 993–1000.

[12] H. R. Maei and R. S. Sutton, "Gq (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces," in *Proceedings of the Third Conference on Artificial General Intelligence*, vol. 1, 2010, pp. 91–96.

[13] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[14] J. Peng and R. J. Williams, "Incremental multi-step q-learning," *Machine Learning*, vol. 22, no. 1-3, pp. 283–290, 1996.

[15] L. Baird *et al.*, "Residual algorithms: Reinforcement learning with function approximation," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 30–37.

[16] L. C. Baird III, "Reinforcement learning through gradient descent," Ph.D. dissertation, US Air Force Academy, US, 1999.

[17] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*. Mit Press, 2012.