
An Experimental Comparison Between Temporal Difference and Residual Gradient with Neural Network Approximation

Shuyu Yin¹, Tao Luo^{2,3}, Peilin Liu¹, Zhi-Qin John Xu^{2*},

¹ School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University

² School of Mathematical Sciences, Institute of Natural Sciences, MOE-LSC
and Qing Yuan Research Institute, Shanghai Jiao Tong University

³ CMA-Shanghai

{shuyu.yin, luotao41, liupeilin, xuzhiqin}@sjtu.edu.cn

Abstract

Gradient descent or its variants are popular in training neural networks. However, in deep Q-learning with neural network approximation, a type of reinforcement learning, gradient descent (also known as Residual Gradient (RG)) is barely used to solve Bellman residual minimization problem. On the contrary, Temporal Difference (TD), an incomplete gradient descent method prevails. In this work, we perform extensive experiments to show that TD outperforms RG, that is, when the training leads to a small Bellman residual error, the solution found by TD has a better policy and is more robust against the perturbation of neural network parameters. We further use experiments to reveal a key difference between reinforcement learning and supervised learning, that is, a small Bellman residual error can correspond to a bad policy in reinforcement learning while the test loss function in supervised learning is a standard index to indicate the performance. We also empirically examine that the missing term in TD is a key reason why RG performs badly. Our work shows that the performance of a deep Q-learning solution is closely related to the training dynamics and how an incomplete gradient descent method can find a good policy is interesting for future study.

1 Introduction

In recent years, many Deep Reinforcement Learning (DRL) applications appear in gaming [Mnih et al., 2013, Silver et al., 2017, Vinyals et al., 2019], recommendation system (Deng et al. [2021]), combinatorial optimization (Bello et al. [2016], Khalil et al. [2017]), etc. These successful applications reveal the huge potential of DRL. Temporal Difference (TD, Sutton [1988]) method is an important component for constructing DRL algorithms, which mainly appears in deep Q-learning-like algorithms, for example Mnih et al. [2013], Van Hasselt et al. [2016], Wang et al. [2016], Hessel et al. [2018] deal with problems with discrete action space. TD method is extended to continuous action space by using actor-critic framework (Gu et al. [2016]). In deep Q-learning, the state-action value function Q is parameterized by a hypothesis function, e.g., neural network. To optimize the parameters, the loss function is defined by the mean squared error between the prediction Q and its true value. However, the true value is untraceable during the training, which is often estimated by Monte-Carlo method or bootstrapping method based on the Bellman equation. When we use a bootstrapping method to estimate the true value, there are two different optimization methods emerge. In the first one, we plugin the estimation of the true value to the loss function. In this case, the

*Corresponding author: xuzhiqin@sjtu.edu.cn.

estimation also contains the state-action value function of some states, therefore, it also depends on the parameters. We then perform gradient descent on the loss function, where the gradient is performed on the prediction and the estimation. This is an exact gradient descent method, known as Residual Gradient (RG) method (Baird [1995], Duan et al. [2021]). In the second one, also known as the TD method, we plugin the estimation of the true value after we perform the gradient descent on the loss function. That is, the gradient is not operated on the true value but only on the prediction one. An interesting question arises that why the gradient descent, so popular in deep learning, is not commonly used in deep Q-learning, while the TD method, an incomplete gradient descent method, prevails. A more important issue is what special characteristics of deep Q-learning, different from training a neural network in supervised learning, make TD popular in deep Q-learning?

In this work, we are trying to answer the above two questions. We conduct experiments in four different environments and we only consider deterministic environments, including Grid World, Cart Pole, Mountain Car, and Acrobot. We use a neural network to parameterize the state-action value function. Also, we follow the off-policy scheme, which uses a fixed data set to train the neural network, the data sets are sampled before training. In order to further reduce randomness, we use the whole set of data to do gradient descent. Our contribution is summarized as follows.

- TD can learn better policy compared to the RG method in our setting. The goodness is twofold: first, the policy learned by TD has a larger accumulated reward. Second, the neural network is more robust and trained by TD than RG when introducing perturbation on parameters.
- Small Bellman residual error does not indicate a good policy. This shows a key difference between reinforcement learning and supervised learning: the loss in supervised learning can indicate the performance of the model. However, the loss in reinforcement learning only indicates the convergence of the model, not the performance.
- In the simple problems we considered, policies can be divided into good and bad regimes according to state value when Bellman residual error is small. Both TD and RG can achieve small losses, however, we empirically find that the RG method is more likely to learn a policy in a bad regime.

The first contribution points out that TD is better than RG, which answers why TD is more popular than RG. The second contribution shows a special characteristic of deep Q-learning, i.e., a small loss does not indicate good performance, which is an answer to the second question. The third contribution further analyzes the special characteristics of deep Q-learning. Our work makes a step towards a better understanding of the learning process of deep Q-learning.

2 Related works

We reviewed several basic empirical comparisons and theoretical analyzes between TD and RG. The RG method was officially proposed by Baird [1995]. It is an alternative approach to solving Bellman residual minimization problem. However, Baird [1995] does not give enough comparison of the performance between RG and TD. Schoknecht and Merke [2003] use spectrum radius analysis to prove that TD converges faster than RG under synchronous, linear function approximation and one-hot embedding settings. Because the proof depends on a linear model and synchronous structure, it is hard to extend their argument into the deep Q-learning setting. Li [2008] use techniques in stochastic approximation and prove that given N training samples, if the model is trained for N steps, RG would have a smaller total Bellman residual error, which is defined as the summation of the Bellman residual error with single training data in each step. This theorem is proved under the asynchronous and linear function approximation settings, which is also hard to extend to deep Q-learning problems. Zhang et al. [2019] combine RG and Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. [2015]) to construct Bi-Res-DDPG method. This combination can improve the performance in some Gym and Atari environments. With modification on RG, it is likely to improve RG in training RL models. To our best knowledge, the only work that compares TD and RG in the deep Q-learning setting is Saleh and Jiang [2019]. This work analyzes the performance of RG in a stochastic environment, while our work compares RG and TD in the deterministic environment.

3 Preliminaries

3.1 Basic concepts

Markov Decision Process is defined as $\mathcal{M}(S, A, f, r, \gamma)$, where S is the state space, $s \in S$ is a state, A is the action space and it is a finite set, $a \in A$ is an action, $f : S \times A \rightarrow S$ is the state transition function, $r : S \times A \rightarrow \mathbb{R}$ is the reward function and $\gamma \in [0, 1)$ is the discount factor. Reward function can be divided into two types: step reward $r_s(s, a)$ represents the reward between two non-terminal states and terminal reward $r_T(s, a)$ represents the reward between non-terminal state and terminal state. We defined the state-action value function as $Q : S \times A \rightarrow \mathbb{R}$ and state value function $V(\cdot) = \max_{a \in A} Q(\cdot, a)$. Given training data set $D_N = \{z_i\}_{i=1}^N = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^N$ where $s_i, s'_i \in S, a_i \in A, f(s_i, a_i) = s'_i$ and $r_i = r(s_i, a_i)$, and use θ to parameterize state-action value function. The empirical Bellman residual loss function is defined as

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (Q_{\theta}(s_i, a_i) - r(s_i, a_i) - \gamma \max_{a' \in A} Q_{\theta}(s'_i, a'))^2. \quad (1)$$

The gradient of this loss function with respect to parameter θ is

$$\nabla \mathcal{L}_{\text{true}} = \frac{1}{N} \sum_{i=1}^N (Q_{\theta}(s_i, a_i) - r(s_i, a_i) - \gamma \max_{a' \in A} Q_{\theta}(s'_i, a')) (\nabla Q_{\theta}(s_i, a_i) - \gamma \nabla \max_{a' \in A} Q_{\theta}(s'_i, a')), \quad (2)$$

The RG method updates parameter θ by the following rule:

$$\theta \leftarrow \theta - \eta \nabla \mathcal{L}_{\text{true}}, \quad (3)$$

which uses the exact gradient form for the RG method, so-called true gradient. η is the learning rate. On the other hand, for the TD method, the parameter is updated by semi gradient as

$$\nabla \mathcal{L}_{\text{semi}} = \frac{1}{N} \sum_{i=1}^N (Q_{\theta}(s_i, a_i) - r(s_i, a_i) - \gamma \max_{a' \in A} Q_{\theta}(s'_i, a')) \nabla Q_{\theta}(s_i, a_i), \quad (4)$$

$$\theta \leftarrow \theta - \eta \nabla \mathcal{L}_{\text{semi}}. \quad (5)$$

3.2 Experiment setting

We conduct experiments on the deterministic environments including discrete state and continuous state settings. There are four environments in total: one grid world environment and three gym classic control environments (Brockman et al. [2016]): Mountain Car, Cart Pole, and Acrobot. The project of these environments is under MIT license, we can use it for research purpose. We read these codes through, there are no personal information or offensive content in these codes. The geometric structure of the Grid World environment is shown in Figure 1. Traps and the goal are terminal states. The action set is {Up, Down, Left, Right}. Only reaching the goal can bring a positive reward, step into a trap leading to a penalty and each step has a small penalty. If the agent hit the boundary, e.g., the agent goes up when it is in the first row, it will stay at its current state. The number represents the state index.

For the Grid World environment, discount factor $\gamma = 0.95$, the terminal reward for the goal state is +1, the terminal reward for trap states is -1, step reward is -0.01. For the Mountain Car environment, discount factor $\gamma = 0.93$, terminal reward is +1, step reward is -1. For the Cart Pole environment, discount factor $\gamma = 0.87$, terminal reward is -5, step reward is +1. For the Acrobot environment, discount factor $\gamma = 0.98$, the terminal reward is 0, step reward is -1. In order to keep the training stability of the TD method, the discount factors can be randomly chosen as long as they are not very close to one to ensure both training algorithms converge.

To control the randomness in exploration, we trained the model with fix data set under the off-policy scheme. For the Grid World environment, the data set is generated by the Cartesian product of non-terminal states and all actions. For the other three Gym environments, the data set is sampled with a random seed, the random seed is used to choose random actions and control the initial state of

State 0	State 1	State 2	State 3	State 4
State 5	State 6	State 7	State 8	State 9
State 10	State 11	State 12	State 13	State 14

Figure 1: The geometric setting of the Grid World environment. The yellow circle represents the starting point, the black circles represent traps, and the blue rectangle represents the goal.

the environment at the beginning of the sampling process. For the Cart Pole environment, we collect samples from the first 100 epochs of random play with random seed 100, the total size of the data set is 2184. For the Mountain Car environment, we collect samples from the first 3 epochs of random play with random seed 550, the total number of data set is 14826. For the Acrobot environment, we collect samples from the first 10 epochs of random play with random seed 190, the total number of data set is 13771. We use the whole set of data to do gradient descent.

Also, we only change the parameter initialization in different trials and keep other hyper-parameters the same, like learning rate, training epoch, and optimization algorithms. In each trial, we use both TD and RG to train the neural network models with the same initial parameter values and other training hyper-parameters. All experiments use Adam as an optimizer. For the Grid World environment, the training step is 30000 and for the other environments, the training step is 50000. The initial learning rate of the Grid World environment is 10^{-4} and 10^{-5} for the other environments. The learning rate decays to its 85% and 75% for each 3000 steps for the Grid World environment and the other environments, respectively. We use an NVIDIA GeForce RTX 3080 GPU to train the model. We use a four-layer fully connected neural network to parameterize the state-action value function Q , the hidden unit number in each layer is 512, 1024, 1024, and use ReLU as an active function.

In addition, for the pre-terminal state, which is the state before the terminal state, we do not force it by a fixed value during the training. We limit our discussion around this setting in this work, however, the methodology of analyzing the regime of policy can be applied to wider settings.

4 TD performs better compared to RG

4.1 TD learns a better policy

In this subsection, we aim to compare TD and RG by using the accumulated reward, which is often used as an indicator for performance in practical tasks. We trained two algorithms in the Grid World environment 80 times and in other environments 10 times. In Figure 2, the line represents the mean accumulated reward, and the region around it represents the 95% confidence interval. Comparing the result in the figures, the TD method clearly learns a better policy during training in all four environments. Besides, the accumulated reward of RG in Figure 2 (a) decreases along with training, which indicates policies in some trials learned by RG step into a bad regime, where the loss function contradicts the goodness of policy.

4.2 Models learned by TD are more robust than RG

In supervised learning, the learned model is not only concerned with the precision of prediction but also the robustness of the solution. The robustness in the following examples is defined as the decrement of the accumulated reward after parameter perturbation.

We use the following equation to generate a noisy parameter vector θ' with a noise scale α :

$$\theta' = \theta + \alpha \times \mathcal{N}(0, 1), \quad \alpha > 0, \quad (6)$$

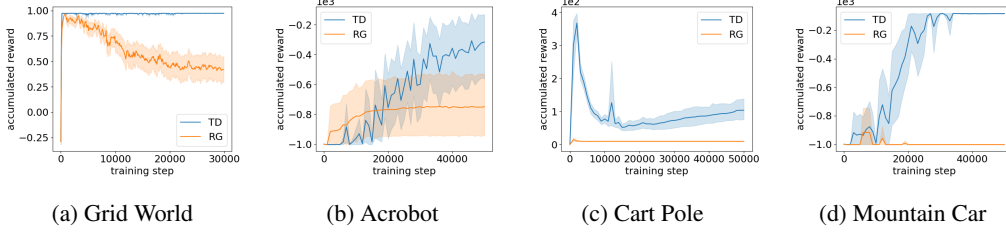


Figure 2: Histogram of the accumulated rewards of different environments for TD (blue) and RG (orange). The abscissa represents the training step, the ordinate represents the accumulated reward. Each line represents the mean value of their accumulated rewards and the transparent regions around it represent the 95% confidence interval. (a) contains 80 trials with different initialization, while each of (b, c, d) contains 10 trials.

In Figure 3, all the policies obtained by TD have higher accumulated rewards than RG at almost all the noise scales. Specifically, in (a), the model learned by TD almost does not change when the perturbation is small, which indicates that the TD method is much more robust in the Grid World environment.

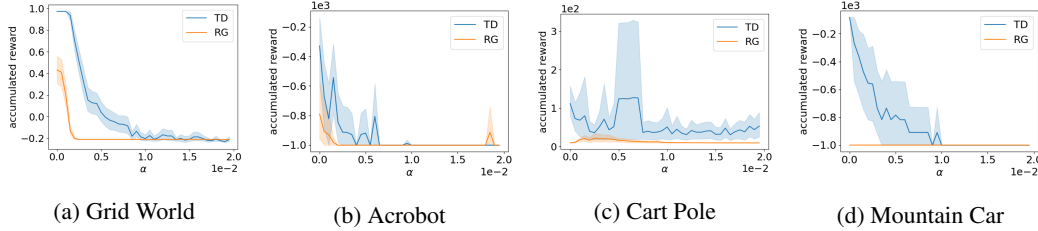


Figure 3: Performance comparison between TD and RG with perturbation on parameters. In all figures, the abscissa represents the scale α of the perturbation, and the ordinate represents the accumulated reward of the current policy. The noise is obtained by (6). At almost all α 's, models learned by TD have higher accumulated rewards, that is, TD is more robust.

In this section, we conclude that TD can empirically be better and more robust than RG in deep Q-learning. This is the answer to the question of why TD is more popular than RG. Next, we are going to study the special characteristics of deep Q-learning, different from supervised learning.

5 Analysis of the result with TD and RG

We assume both the TD method and RG method can achieve a small Bellman residual error. Under this assumption, we have

$$Q(s, a) \approx r(s, a) + \gamma \max_{a' \in A} Q(s', a'), \quad (7)$$

for any $(s, a, s', r) \in D_N$. If we choose $a = \arg \max_{a'' \in A} Q(s, a'')$ and use $V(\cdot)$ to represent $\max_{a \in A} Q(\cdot, a)$, we will have

$$V(s) \approx r(s, \arg \max_a Q(s, a)) + \gamma V(s'). \quad (8)$$

5.1 The RG method can obtain a random policy with a small loss

In this subsection, we would show that the loss function in deep Q-learning does not indicate the performance of the trained model, which is stark different from supervised learning.

Figure 4 (a) shows the policy learned by RG, the red triangle represents the action with maximum state-action value. Clearly, (a) shows a bad policy. However, the loss in Figure 4 (e) is around 10^{-9} , which is a very small number. By comparing the good policy learned by TD in Figure 4 (b) and the loss around 10^{-5} in (f), it is easy to notice that lower Bellman residual error does not indicate a better

policy. The following analysis will show that there are distinct regimes of the state value where the Bellman residual can all be small but the performance of policy can be very different.

Intuitively speaking, for each non-terminal state of the Grid World problem, we want there exists a path from the state to the pre-terminal state of the goal, where the state value monotonically increases and the pre-terminal state is the state before the terminal state. This means if a state is closer to the goal, it should have a higher state value. For a pre-terminal state s and corresponding terminal state s' , we have

$$Q(s, a) = r_T(s, a) + \gamma \max_{a' \in A} Q(s', a'). \quad (9)$$

Since $r_T(s, a)$ is positive, it does not have to require the highest state value for the terminal state. Similarly, the policy in the Mountain Car and Acrobot problem follows the same analysis because the goal of these two problems is to go to the terminal states similar to the Grid World. Mathematically, this means given a data tuple $z = (s, a, s', r)$, where $a = \arg \max_{a' \in A} Q(s, a')$ and s' is closer to the goal but not a terminal state, we want the following inequality holds

$$V(s) \approx r_s(s, \arg \max_a Q(s, a)) + \gamma V(s') < V(s') \quad (10)$$

$$\Rightarrow V(s') > \frac{r_s(s, \arg \max_a Q(s, a))}{1 - \gamma}. \quad (11)$$

For example, we consider the Grid World problem, we have $\gamma = 0.95$, $r_s = -0.01$, then, we obtain

$$V(s') > \frac{-0.01}{1 - 0.95} = -0.2. \quad (12)$$

To ensure the monotonicity, it requires all the state values except for the terminal state and the initial state greatly larger than -0.2 , that is, the regime for a good policy. Otherwise, it is a bad regime that Bellman residual is small but the policy is bad. Figure 4 (c) shows an RG example in the bad regime. For all states except terminal states, the value is around -0.2 in the bad regime with a small Bellman residual error shown in (e) and a bad policy shown in (a). However, for the TD case in (d), all the non-terminal state values are greatly larger than -0.2 , which indicates this value function is in a good policy regime. Consistent with theoretical analysis, we obtain a good policy in (b).

We then look into how these state values evolve during the training process for the two training methods. For the RG method, as shown in Figure 4 (g), the goal state almost monotonically decreases with the training step and achieves -1.26 , the lowest value among all state values. When the Bellman equation holds, all the other non-terminal state values can be computed as -0.2 . On the contrary, for the TD method, as shown in Figure 4 (h), all state value functions increase with the training step initially. Taken together, studying the dynamic behavior may be a key to understanding why these two algorithms take the same initial values to different regimes.

Statistically, we conduct 80 trials in the Grid World environment with different initialization, and we find that 38 out of 80 times the RG method learns a bad policy and TD learns all good policies.

In addition, to show the superiority of TD compared to RG, we use the model learned by RG as initialization and train this model with the TD method for 30000 steps. As an example shown in Figure 5 (a, b), TD learns a good policy and state values satisfy the monotonicity in general. The TD method jumps out of the bad policy regime learned by the RG method and enters the good policy regime where all the state values are greatly larger than -0.2 . Figure 5 (c, d) represents the loss and state value dynamics during training. At the beginning of these two figures, the loss and state value increase dramatically, which may represent the neural network model jumping out of the bad policy regime.

5.2 In Cart Pole, the RG method always learns a bad policy

The Cart Pole problem is different from the Grid World problem, which is trying to prevent the agent from going to terminal states. Therefore, the regime of good policies is also different. In order to simplify the following discussion, we define the terminal state set as

$$S_T = \{s \mid \text{for all } s \in S \text{ and } s \text{ is terminal state.}\}, \quad (13)$$

the pre-terminal state set as

$$S_{pT} = \{s \mid s \in (s, \arg \max_{a' \in A} Q(s, a'), r) \text{ where } s' \in S_T\}, \quad (14)$$

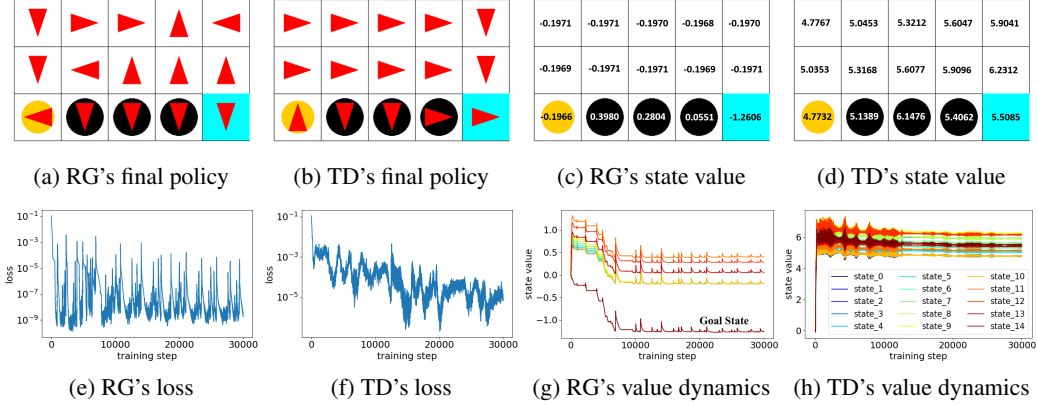


Figure 4: Experiment results comparison between TD and RG in the Grid World problem with the same initial parameter value and training hyper-parameters. Comparing (a, e) with (b, f), RG’s loss is around 10^{-9} and RG learns a bad policy, TD’s loss is around 10^{-5} and TD learns a good policy. This means lower loss does not indicate a better policy. In (c, g), state value is in the bad regime. However, in (d, h), state value is in the good regime. Besides, in (d), because the goal state has a reward $+1$, it can have a lower value than the previous state.

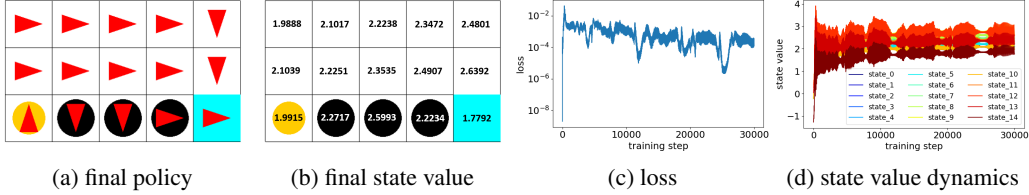


Figure 5: The result of the model that is continuously trained by TD from the model trained by RG in Figure 4 (a). (a, b) show the policy and state value after using TD to train for 30000 steps. As we can see, it jumps out of the bad policy and learns a good policy. (c, d) show the loss and state value dynamics during training. There is an impulse at the beginning of these two figures, which could represents jumping out of old solution.

and the other state set as

$$S_O = S \setminus (S_T \cup S_{pT}). \quad (15)$$

We also define the mean state value of the terminal state as

$$V_T = \frac{1}{|S_T|} \sum_{s \in S_T} \max_{a \in A} Q(s, a), \quad (16)$$

mean state value of pre-terminal state value as

$$V_{pT} = \frac{1}{|S_{pT}|} \sum_{s \in S_{pT}} \max_{a \in A} Q(s, a), \quad (17)$$

and mean state value of the other state set as

$$V_O = \frac{1}{|S_O|} \sum_{s \in S_O} \max_{a \in A} Q(s, a), \quad (18)$$

where $|S|$ represents the element number of set S .

One way to keep the agent alive is to let the agent move in a loop. To construct a loop, the state value in the loop should not increase or decrease monotonically, all the state values should be equal. This

means for a given data tuple $(s, \arg \max_a Q(s, a), s', r)$ in the loop, we should have

$$V(s) \approx r_s(s, \arg \max_a Q(s, a)) + \gamma V(s') = V(s') \quad (19)$$

$$\Rightarrow V(s') = \frac{r_s(s, \arg \max_a Q(s, a))}{1 - \gamma} = \frac{1}{1 - 0.87} \approx 7.6923. \quad (20)$$

Also, we want for all $s \in S_{pT}$ to have $V(s) \ll 7.6923$, so the agent does not easily go to these states. Therefore, a small terminal state value is preferred according to Bellman equation. However, the Cart Pole problem S_O may contain states that are not in the loop. To ensure the agent does not leave the loop, those states in S_O and not in the loop should have lower state values, smaller than 7.6923, leading to $V_O < 7.6923$. Also, we want $V_{pT} \ll 7.6923$ and a small V_T . This is a good regime of policies, otherwise, it is highly likely a bad regime.

In Section 4.1, we have shown that RG always learns a bad policy in the Cart Pole problem in Figure 2 (c). Besides, RG also has a lower loss shown in Figure 6 (c). This is because the model learned by RG falls into a bad regime of policies. Figure 6 shows V_O , V_{pT} and V_T of RG and TD for all ten experiments in the Cart Pole problem. In Figure 6 (a), nine of ten times V_{pT} and V_T obtained by RG is larger than 7.6923. Although the remaining one has $V_{pT} < 7.6923$, we find that most state values are in a bad regime. For example, in 100 random play sequence data, we count the pre-terminal state value and terminal state value of each play predicted by the trained model. We find that 24 and 66 times the pre-terminal state values and the terminal state values are larger than 7.6923 in 100 random play sequence data, respectively. Too large terminal state value in the Cart Pole Problem indicates RG learns a bad policy. On the contrary, for all the models learned by TD shown in 2 (b), V_{pT} are around -30, V_T are less than -30, which is a pretty small number, and V_O are around 7. Models learned by TD are in the good policy regime.

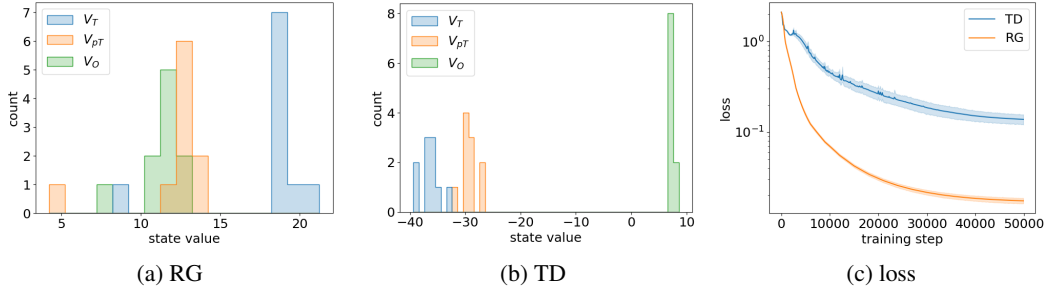


Figure 6: Histogram of V_O , V_{pT} and V_T for ten experiments in the Cart Pole problem with TD and RD training and the training loss for ten experiments. In (a), V_{pT} and V_T obtained by RG are larger than 7.6923 for nine times, so the agent will go to the terminal state following the increasing state values. On the contrary in (b), V_{pT} and V_T obtained by TD are pretty small, all around -30, so the agent will not go to the terminal state easily. In (c), the line represents the mean loss, and the region around it is the 95% confidence interval. The mean loss for TD is 0.138 and 0.017 for RG. Same as the Grid World problem, RG learns a bad policy with a lower loss. Too large terminal state value in the Cart Pole problem indicates the model learned by RG achieves a bad policy.

6 Learning dynamics of backward-semi gradient

The mathematical difference between TD and RG is their gradient form. We define the difference between $\nabla \mathcal{L}_{\text{true}}$ and $\nabla \mathcal{L}_{\text{semi}}$ as backward-semi gradient

$$\nabla \mathcal{L}_{\text{backward-semi}} = \nabla \mathcal{L}_{\text{true}} - \nabla \mathcal{L}_{\text{semi}} \quad (21)$$

$$= \frac{1}{N} \sum_{i=1}^N (Q_{\theta}(s_i, a_i) - r(s_i, a_i) - \gamma \max_{a' \in A} Q_{\theta}(s'_i, a')) (-\gamma \nabla \max_{a' \in A} Q_{\theta}(s'_i, a')). \quad (22)$$

We used the backward-semi gradient to update the model parameters in the Grid World environment. To ensure the convergence, we change the initial learning rate to 10^{-6} and it decays to its 65% for

each 3000 steps. Figure 7 (a), all the state values decrease, and in (b) the loss is increasing in the later stage. As shown in Figure 7 (c), the highest and the lowest final state values are traps (the terminal states with negative reward, black circle), and goal (the terminal state with a positive state, blue rectangle), respectively. As in the above analysis, a low estimation of terminal state in Grid World or a high estimation of terminal state in Cart Pole indicates that state values are in a bad policy regime. Moreover, TD does not use the gradient at the terminal state to update the parameter, therefore, the backward-semi gradient dominates the value change in terminal states.

The backward-semi gradient is the difference between TD with RG. The experiment in Figure 7 confirms that this backward-semi gradient is the key that RG often fails.

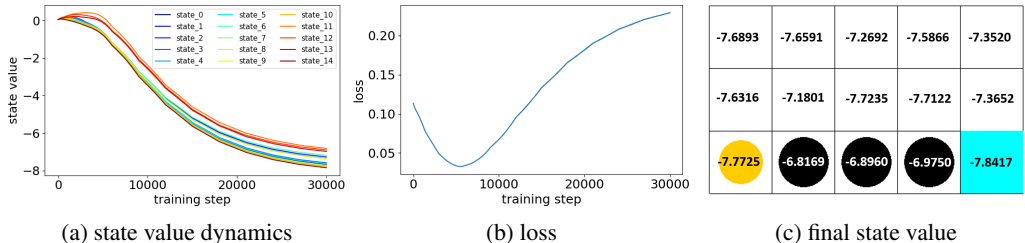


Figure 7: State values are obtained by using $\nabla \mathcal{L}_{\text{backward-semi}}$ as a gradient to train the neural network model in the Grid World environment. (a) shows all the state values decreasing along with training. (b) shows the training loss and it is increasing in the latter stage, which means $\nabla \mathcal{L}_{\text{backward-semi}}$ does not minimize Bellman residual error. (c) shows that the three penalty states, which have a negative reward, have the top three values among all the states, and the goal state, which has a positive reward, has the lowest value.

7 Conclusion

In this paper, we are trying to compare the performance of the TD and RG methods in a Deep Q-learning setting and analyze the reason that makes RG perform badly. We show their performance in four different environments, TD outperforms RG in all environments. Then we introduce noise to parameters to show that models learned by TD are more robust. We also analyze two examples that represent two types of problems and conclude that RG learns a bad policy because it finds the solution in a bad regime where the Bellman residual error is small while the policy is bad.

There is an important fact that emerges from this work: a low Bellman residual is not enough for judging the goodness of a given policy, and the state value should also be considered. This is a crucial difference between reinforcement learning and supervised learning wherein supervised learning loss represents its performance. This also inspires us, if we want to talk about generalization in reinforcement learning, we need to consider more indicators other than Bellman residual error.

This work also raises a cautionary tale about stochastic gradient descent (SGD) training, which is RG in this work. The extensive success of deep learning has formed an empirical belief that SGD or its variants are powerful to train neural networks to obtain good generalization. However, this work points out that in deep Q-learning with neural network approximation, SGD may have some serious drawbacks. Our work on the analysis of the backward-semi gradient also serves as an empirical basis for further theoretical analysis.

There are still some questions unsolved: why backward-semi gradient has such learning dynamics? what is the function of the negative terminal state in the Grid World example? What is the relation between reward function design and performance? We will try to answer these questions in the following works.

References

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

- Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. Unified conversational recommendation policy learning via graph-based reinforcement learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1431–1441, 2021.
- Yaqi Duan, Chi Jin, and Zhiyuan Li. Risk bounds and rademacher complexity in batch reinforcement learning. In *International Conference on Machine Learning*, pages 2892–2902. PMLR, 2021.
- Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pages 2829–2838. PMLR, 2016.
- Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.
- Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.
- Lihong Li. A worst-case comparison between temporal difference and residual gradient with linear function approximation. In *Proceedings of the 25th international conference on machine learning*, pages 560–567, 2008.
- Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- Ehsan Saleh and Nan Jiang. Deterministic bellman residual minimization. In *Proceedings of Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*, 2019.
- Ralf Schoknecht and Artur Merke. Td (0) converges provably faster than the residual gradient algorithm. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 680–687, 2003.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.
- Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.
- Shangdong Zhang, Wendelin Boehmer, and Shimon Whiteson. Deep residual reinforcement learning. *arXiv preprint arXiv:1905.01072*, 2019.