

Low-Rank and Temporal Smoothness Regularization on Value-Based Deep Reinforcement Learning

Edward Yam



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-218

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-218.html>

August 18, 2022

Copyright © 2022, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**Low-Rank and Temporal Smoothness Regularization
on Value-Based Deep Reinforcement Learning**

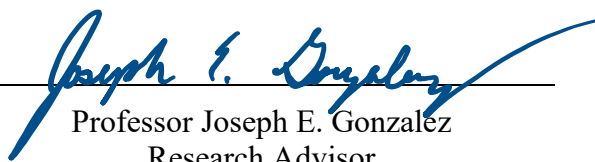
by Edward Yam

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,
University of California at Berkeley, in partial satisfaction of the requirements for the
degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

Committee:



Professor Joseph E. Gonzalez
Research Advisor

8/18/2022

(Date)

* * * * *



Professor Michael Mahoney
Second Reader

8/16/2022

(Date)

Low-Rank and Temporal Smoothness Regularization on Value-Based Deep Reinforcement Learning

Edward Yam

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
edwardyam (at) berkeley (dot) edu

Abstract

Value-based deep reinforcement learning is an efficient methodology to solve Markov Decision Process (MDP) problems, where a deep neural network work learns a value function, which represents the estimation of future total reward after taking a possible action. Also, it allows reinforcement learning agents to make decisions based on a value function. Existing research has shown that the value function in many reinforcement learning tasks has a low-rank hidden structure.

Harnessing the implicit low-rank structure of value functions, we can design and test two different neural network regularization methods to emphasize and train for these properties. First, value functions can be decomposed into a low rank and a sparse component. Second, value functions should have similar outputs for states in adjacent time steps, a property of temporal smoothness. We show that the enforcement of temporal smoothness is implicitly low-rank regularization of the state-action function. As well, we empirically find that models that use these regularization methods outperformed their baseline counterparts with the best performance occurring when both regularization methods are used.

Contents

1	Introduction and related work	3
1.1	Markov decision process (MDP)	3
1.2	Value Iteration	3
1.3	Deep Q-learning (DQN) in training	3
1.4	Low-rank property of value function	4
1.5	Temporal continuity of reinforcement learning (RL) tasks	4
1.6	Related Work	4
2	Methodology	5
2.1	Low-rank and sparse regularization (LRSR) on value function	5
2.2	Temporal continuity regularization (TSR) on value function	6
3	Experiment results	6
3.1	Low-rank and Sparse Regularization (LRSR) on Value Function	6
3.2	Temporal continuity regularization (TSR) on value function	7
4	Discussion: Temporal continuity and nuclear norm minimization	7
5	Conclusion and future work	9

1 Introduction and related work

1.1 Markov decision process (MDP)

A Markov Decision Process (MDP) can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \mu, P, r, h, \gamma)$ where \mathcal{S} and \mathcal{A} are the sets of all possible states and actions, respectively. μ is the initial distribution of state s_0 . P is a probabilistic transition function, and r is a reward function. P gives the probability of transitioning into state s' from taking action a at the current state s , and is often denoted $P(s'|s, a)$. r gives a scalar value indicating the immediate reward received for taking action a at the current state s and is denoted $r(s, a)$. h is the horizon, the number of state transitions that will occur for an agent. $\gamma \in (0, 1)$ is the discounting factor.

MDPs provide a good framework for modeling environments that have uncertainty. One example is self-driving cars. Cars-based agents can be incentivized to reach a particular destination safely and MDP's probabilistic transition function can help model uncertainty states, such as other cars' future actions.

1.2 Value Iteration

The goal is to create the optimal policy π^* . To solve an MDP, we compute a policy π^* that if followed, maximizes the total rewards given $(\mathcal{S}, \mathcal{A}, \mu, P, r, h, \gamma)$. The optimal state value function $V^*(s)$ is the expected value of all future rewards starting from s given the optimal policy.

$$V^{\pi^*}(s) = \mathbb{E}_{\{s_i \sim P(s_i | s_{i-1}, \pi^*(s_{i-1}))\}_{i \in [1:h]}} \left[\sum_{n=0}^h \gamma^n r(s_n, a_n) \right] \quad (1)$$

However, the optimal policy and state value function are unknown. This must be learned. In value iterations, optimal policies can be trained by using expectation-maximization based algorithms [1]. The values of an approximate V are calculated by sampling P to get (s_i, s_{i+1}, a_i) . Then $V^\pi(s)$ can be formulated as the following to be solved as a system of equations in the case of \mathcal{S} being discrete state space. Then the policy would change to maximize the right side of equation 2.

$$V^\pi(s_i) = \gamma \sum_{s_j \in \mathcal{S}} V^\pi(s_j) \cdot P(s_j, \pi(s_i) | s_i) \quad (2)$$

The emphasis in this paper will be placed on the optimal state-action value function $Q(s, a)$, which is the expected value when starting in state s , taking action a , and following actions dictated by π^* . Mathematically, it obeys the Bellman recursion

$$Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(s' | s, a)} \left[\max_{a' \in \mathcal{A}} Q(s', a') \right] \quad (3)$$

1.3 Deep Q-learning (DQN) in training

Deep Q-learning (DQN) is an efficient and expressive methodology for solving MDP problems. The goal of Q-learning is to train a deep neural network to represent $Q(s, a)$ and obtain policy $\pi(s) = \arg \max_{a \in \mathcal{A}} Q(s, a)$. During training, the model collects a batch of $|\mathcal{B}|$ samples and collects $\{(s_t^{(i)}, r_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)})\}_{i=1}^{|\mathcal{B}|}$, and forms the following updating targets.

$$y^{(i)} = r_t^{(i)} + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}^{(i)}, a'; \theta) \quad (4)$$

$Q(s_{t+1}^{(i)}, a'; \theta)$ is the learned neural network, called the value network, to represent the value function $Q(s, a)$ where θ denotes the parameters of the neural network which characterizes the network. The value network is then updated by taking a gradient step for the loss function.

$$\sum_{i=1}^{|\mathcal{B}|} (y^{(i)} - Q(s_t^{(i)}, a_t^{(i)}; \theta))^2 \quad (5)$$

1.4 Low-rank property of value function

Many works in reinforcement learning or control community shows the value function $Q(s, a)$ has low-rank property in a variety of tasks ([2], [3], [4]). For example, [2] shows that, if we denote the value function in a matrix form, $Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$, called the Q table, the matrix Q can be decomposed as $Q = L + E$, where L is a low-rank matrix and E is a sparse matrix. Intuitively, this means the Q table is composed mainly of the low-rank component L with a sparse noise E . Similarly, [4] shows that for most of the Atari games, the value function has a low-rank structure. Also, leveraging the knowledge that the value function has low-rank structure, [4] design a method using low-rank matrix reconstruction to enforce low-rank structure of the trained Q network through modifying the fitting target of the Q network ($y^{(i)}$ in eq. 5), leading to better performance on more than half of the Atari games. This can not be done with the state value function $V(s)$.

1.5 Temporal continuity of reinforcement learning (RL) tasks

Many RL tasks satisfy temporal smoothness, indicating that good action choices are usually similar across states of adjacent time steps. For example, [5] built a robust model-based RL algorithm by assuming temporal continuity of RL tasks. Having this understanding of the RL task, we believe that, during RL training, if we incentivize our RL model to have a temporal smoothness prediction, our RL agent could potentially have better performance.

The intuition of the benefit of having temporally smooth predictions is illustrated in the following example. In the Pong Atari game, when the ball is falling toward the baseline, typical RL agents tend to move the paddle to catch the ball at the last moment and catch the ball "just in time." A human player, however, tends to move the paddle earlier to the location where the ball is predicted to land and "waits" there before the ball reaches the paddle. Human play can be considered temporally smooth because the action to move the paddle earlier is more consistent across several time frames.

Intuitively, the behavior of the RL agent is riskier since the paddle is moved to catch the ball just in time, so any wrong decision during the process when the paddle is moved can lead to missing the ball. If we incentivize our RL model to have temporally smooth predictions, the agent will perform more like humans and better training.

Fundamentally, enforcing temporal continuity of value function is similar to enforcing low-rank structure. Low-rank property implies that the value function has a similar output for certain different but similar states, while temporal continuity implies the value function has a similar output for states of nearby time steps. This is expounded in section 4, where we analyze the theoretical relationship between temporal continuity and low-rank regularization.

1.6 Related Work

Regularization terms have been widely adopted in neural network training to prevent overfitting (such as L1 and L2 regularization terms discussed in [6]). Furthermore, when the target neural network we aim to train is assumed to be of low rank, low-rank regularization (LRR) methods are widely used to enforce low-rank property of the learned function, which has achieved great success in many data analysis tasks ([7] provides a comprehensive review of the applications of LRR).

Recently, there are also some works on the regularization of value functions for RL. For example, [8] proposed a functional regularization approach to increase training stability of value-based reinforcement learning so that stable training results can be generated even without a target network, leading to improvements in sample efficiency and performance across a range of Atari and simulated robotics environments. As well, [9] proposes a method that uses PARAFAC decomposition to create a low-rank matrix. Also, [10] proposed a value-based RL algorithm that uses mutual-information regularization to optimize a prior action distribution for better performance and exploration.

Several recent works focus on applying low-rank regularization by using low-rank matrix estimation as detailed in [20, 21]. The major modification to the vanilla RL learning algorithm is that Q_B is

Table 1: Common low-rank regularization terms

Type	$\mathcal{R}(Q_B)$
Nuclear norm	$\sum_{i=1}^k \sigma_i$
Log nuclear norm (LNN) [11]	$\sum_{i=1}^k \log(\sigma_i + 1)$
Elastic-Net Regularization of Singular Values [12]	$\sum_{i=1}^k (\sigma_i + \gamma \sigma_i^2)$
Schatten- p norm [13, 14]	$\sum_{i=1}^k \sigma_i^p$
Truncated Nuclear Norm [15]	$\sum_{i=r+1}^k \sigma_i$
Partial Sum Nuclear Norm [16]	$\sum_{i=r+1}^k \sigma_i$
Weighted Nuclear Norm [17, 18, 19]	$\sum_{i=r+1}^k w_i \sigma_i$

further processed. Values of Q_B are procedurally or randomly removed then a low-rank matrix estimation algorithm to applied to reconstruct the matrix. [20] further proposes an asymptotically sample-efficient method for learning assuming an optimal policy of rank r and that the Q function is Lipschitz continuous over the state-action space; temporal smoothness is Lipschitz continuity stochastically enforced during batch training. Aswell, [21] further proposes an uncertainty-aware algorithm that selectively removes highly certain values before the low-rank matrix estimation of the Q_B matrix.

Lastly, several papers use low-rank factorization for the feature selection of linearly encoded states of MDP models such as [22] [23].

2 Methodology

2.1 Low-rank and sparse regularization (LRSR) on value function

Define $Q_B \in \mathbb{R}^{|\mathcal{B}| \times |\mathcal{A}|}$ as the output of the value network given a batch of states $\{(s_t^{(i)})\}_{i=1}^{|\mathcal{B}|}$ as input. In other words, if we denote $f : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$ as the value function that inputs a state and outputs the expected total future reward for each action in the action space in vector form, then

$$f(s) = [Q(s, a_0) \quad Q(s, a_1) \quad \cdots \quad Q(s, a_{|\mathcal{A}|})] \quad (6)$$

$$Q_B = \begin{bmatrix} f(s_t^{(1)})^T \\ f(s_t^{(2)})^T \\ \cdots \\ f(s_t^{(|\mathcal{B}|)})^T \end{bmatrix} \quad (7)$$

Following this definition, [2] shows that, in many RL tasks, we can decompose matrix Q_B into the sum of a low-rank component and a sparse component. If we have the full matrix of $Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$. We can have any row represented by a small subset of the rows of Q .

$$f(s) \approx \sum_{i=0}^k f(s_{a_i}) \quad (8)$$

However, it is not computationally feasible for arbitrarily large state spaces. Instead, we can leverage this idea by replacing the original value network, $Q(s, a; \theta)$, with the sum of two jointly trained neural networks $L(s, a; \theta_L)$ and $E(s, a; \theta_E)$. In this case, the loss function in equation 5 becomes the following.

$$\sum_{i=1}^B (y^{(i)} - L(s_t^{(i)}, a_t^{(i)}; \theta_L) - E(s_t^{(i)}, a_t^{(i)}; \theta_E))^2 + \lambda_R R(L_B) + \lambda_S S(E_B) \quad (9)$$

Here λ_R and λ_S are hyper-parameters regularizing different parts of the network, $L_B, E_B \in \mathbb{R}^{|\mathcal{B}| \times |\mathcal{A}|}$ are the output of the L and E network respectively given $\{(s_t^{(i)})_{i=1}^{|\mathcal{B}|}\}$ as input. As well $R(L_B)$ is the rank of L_B , and $S(E_B)$ is the sparsity of E_B , and θ_L and θ_E denotes the parameter of the neural network L and E respectively.

The RL policy obtained from this setting becomes the following.

$$\pi(s) = \operatorname{argmax}_{a \in \mathcal{A}} L(s, a; \theta_L) + E(s, a; \theta_E) \quad (10)$$

Because the rank function is not differentiable, the low-rank component L_B was regularized using nuclear norm as it is the convex envelope of the rank function. Also, the sparse component was regularized using L_1 norm. The neural networks $L(s, a; \theta_L)$ and $E(s, a; \theta_E)$ shared the same convolutional and initial fully connected layers then split for the final layers.

2.2 Temporal continuity regularization (TSR) on value function

It has been shown that many RL tasks, especially Atari games, have temporal smoothness, we can design a neural network regularization term such that the value network has a similar output for adjacent states. In this case, the loss function in equation 5 becomes

$$\sum_{i=1}^B (y^{(i)} - Q(s_t^{(i)}, a_t^{(i)}; \theta))^2 + \lambda \|Q_B - Q'_B\|_1 \quad (11)$$

where λ is a hyperparameter controlling how heavy the network is regularized, $Q_B \in \mathbb{R}^{|\mathcal{B}| \times |\mathcal{A}|}$ is the output of the value network given $\{(s_t^{(i)})_{i=1}^{|\mathcal{B}|}\}$ as input, $Q'_B \in \mathbb{R}^{|\mathcal{B}| \times |\mathcal{A}|}$ is the output of the value network given $\{(s_{t+1}^{(i)})_{i=1}^{|\mathcal{B}|}\}$ as input, and $\|\cdot\|_1$ is the L_1 norm. Another regularization method we can try is

$$\sum_{i=1}^B (y^{(i)} - Q(s_t^{(i)}, a_t^{(i)}; \theta))^2 + \lambda * \|\tilde{Q}_B - \tilde{Q}'_B\|_1 \quad (12)$$

where \tilde{Q}_B and \tilde{Q}'_B are Q_B and Q'_B respectively with each row standardized (i.e., making the mean zero and standard deviation one). Intuitively, the regularization term prevents the value network from having drastically different predictions for current and the next observation, such that the RL policy will choose similar actions for adjacent time steps.

3 Experiment results

3.1 Low-rank and Sparse Regularization (LRSR) on Value Function

The result of the experiments for LRSR are illustrated in Figure 1. The blue curve is the learning curve for normal, single output, unregularized DQN (using equation 5 as the loss function). The orange curve is the setting where the network has two outputs. One is regularized to have low-rank output and the other is regularized to have sparse output. To be more specific, it uses equation. 9 as the loss function with $\lambda_R = 10^{-4}$ and $\lambda_S = 10^{-4}$. We can see from the plot that decomposing the network output into a low-rank and a sparse component does yield higher performance.

The green curve is the learning curve where the network has two outputs. One of them is unregularized while the other is regularized to have sparse output. To be more specific, it uses equation. 9 as the loss function with $\lambda_R = 0$ and $\lambda_S = 10^{-3}$. Surprisingly, the model works particularly well even without low-rank regularization. Compared with a normal, single output, unregularized DQN in the blue curve, this setting simply added another output in addition to the unregularized output, where the additional output is regularized to be sparse. The high performance of this setting was

unexpected. Although, uncertain it may be the result of heavy regularization on sparsity causing the network to revert back to a regular feed-forward neural network. However, that hypothesis is not well substantiated because it does train better than the baseline neural network. Perhaps the additional output is modeling the noise on the value function, or, perhaps the additional output gives the network additional parameters to better fit the data.

To show that the regularization (i.e., λ_R and λ_S in equation 9) is really affecting the network performance, a baseline was run with the setting there λ_R and λ_S are both zero. The learning curve of this setting is illustrated as the red curve in Figure 1. We can see that the performance of this setting is much worse than all others.

For each setting in Figure 1, the models were trained with at least 3 different random seeds. For each model obtained, the model’s performance was evaluated every 1000 time steps. During the evaluation, 100 episodes are run. The average performance for all episodes across all models is then calculated to plot Figure 1. Therefore, each data point in Figure 1 is an average score across at least 300 episodes.

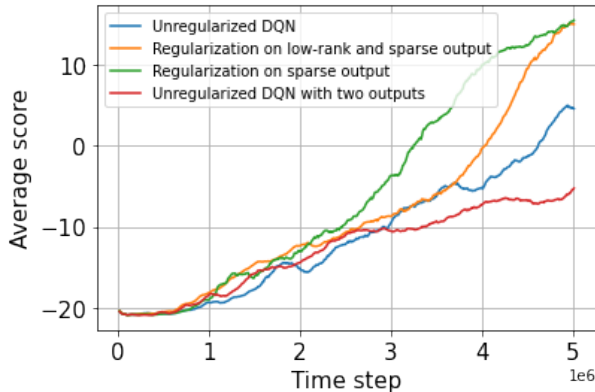


Figure 1: Learning curve of DQN with different regularization methods

3.2 Temporal continuity regularization (TSR) on value function

The performance of TSR versus normal, and unregularized DQN has been tested. Using DQN with TSR regularization term as equation 11 and modifying the hyperparameter λ , the learning curves are illustrated in figure 2. Normal, unregularized, DQN is the comparison baseline where the λ in equation 11 is set as zero. $\lambda=1e-2$ and $\lambda=1e-3$ are the learning curves where λ is set as 10^{-2} and 10^{-3} respectively.

From the experiments, we do see that adding the regularization term with $\lambda = 10^{-3}$ leads to slightly better performance. The intuition is that, due to the noise of the RL environment and the high variance nature of DQN, adding a certain level of regularization according to the domain knowledge of the environment helps learn the value function better. The case where $\lambda = 10^{-2}$ is likely to be caused by the regularization being too heavy.

In addition to testing the regularization method in equation 11, we also tested the standardized version of the regularization method as equation 12. However, the performance is similar or slightly worse than that of equation 11 so is omitted from this report.

Similar to Figure 1, each learning curve in Figure 2 are average performance across models trained by at least 3 different random seeds where 100 episodes are evaluated for each model.

4 Discussion: Temporal continuity and nuclear norm minimization

In this section, we will derive an upper bound of the nuclear norm of the Q table and shows that, when the temporal smoothness regularization term is minimized, the nuclear norm, which is the convex approximation of the rank, is also minimized.

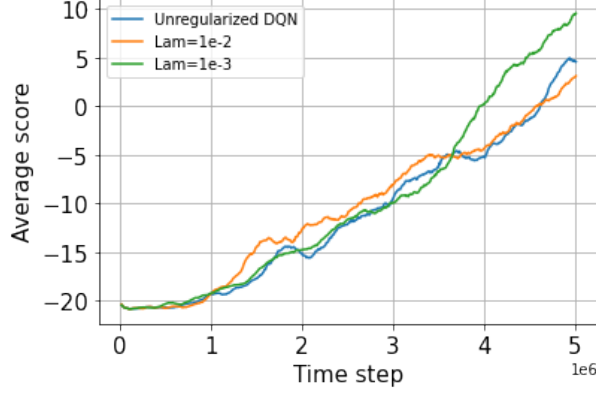


Figure 2: Learning curve of DQN with different λ for TSR

Assumption 1: Denote $f : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$ as the value function which inputs a state and outputs the total expected future reward after taking each action over the action space in a vector form. For any state $s \in \mathcal{S}$ reachable in the RL environment where $s' \in \mathcal{S}$ is any state reachable from s within one time step. Our assumption is that the norm of the difference of the output vectors is at most ϵ .

$$\|f(s) - f(s')\|_2 \leq \epsilon \quad (13)$$

Intuitively, the RHS of equation 13 is the difference between the output of the value function of adjacent states, which is what TSR regularization term is trying to minimize in through equation 11.

Under Assumption 1, due to the triangular inequality, given any reachable state \hat{s} ,

$$\|f(s_0) - f(\hat{s})\|_2 \leq \sum_{s_i} \|f(s_i) - f(s_{i+1})\|_2 \leq h\epsilon \quad (14)$$

where s_0 is the initial state in the MDP and h is the maximum number of time steps before the MDP terminates.

Moreover, by equation 15 and the triangular inequality, given any two reachable states s and s' ,

$$\|f(s) - f(s')\|_2 \leq \|f(s_0) - f(s)\|_2 + \|f(s_0) - f(s')\|_2 \leq 2h\epsilon \quad (15)$$

Lemma 1, for any matrix M ,

$$\|M\|_*^2 = \left(\sum_{i=1}^r \sigma_i\right)^2 \leq r \sum_{i=1}^r \sigma_i^2 = r \|M\|_F^2 \quad (16)$$

where r is the rank of M , σ_i is the i -th singular value of M . The first equality is due to the definition of nuclear norm. The inequality is due to Cauchy-Schwarz inequality. The second equality is due to the property of Frobenius that $\sum_{i=1}^r \sigma_i^2 = \|M\|_F^2$.

If we define $L \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ as a matrix with rank 1 and

$$L = \begin{bmatrix} f(s_t^{(1)})^T \\ f(s_t^{(1)})^T \\ \dots \\ f(s_t^{(1)})^T \end{bmatrix} \quad (17)$$

Under Assumption 1, we can decompose Q_B as

$$Q_B = L + S \quad (18)$$

where S is the difference of Q_B and L . Note that $Q_B, L, S \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ and Q_B is defined in equation 7

If Assumption 1 holds, by equation 15, 16, and 18, we know that

$$\|S\|_F = \|Q_B - L\|_F = \sum_{i=1}^{|\mathcal{B}|} \|f(s_t^{(i)}) - f(s_t^{(1)})\|_2 \leq 2|\mathcal{B}|T\epsilon \quad (19)$$

Also,

$$\|L\|_* = \sigma = \|L\|_F \quad (20)$$

where σ is the only singular value of the rank 1 matrix L . The second equality holds since $\sum_{i=1}^r \sigma_i^2 = \|M\|_F$ for any matrix.

Therefore,

$$\begin{aligned} \|Q_B\|_* &\leq \|L\|_* + \|S\|_* = \|L\|_F + \|S\|_* \\ &\leq \|L\|_F + \text{rank}(S)^{0.5} \cdot \|S\|_F \\ &\leq \|L\|_F + |\mathcal{B}|^{0.5} \cdot \|S\|_F \\ &\leq \|L\|_F + 2|\mathcal{B}|^{1.5}T\epsilon \end{aligned} \quad (21)$$

The first inequality is due to the triangular inequality. The equality is due to equation 20. The second inequality is due to Lemma 1. The last inequality is due to equation 19.

As temporal smoothness is applied, the output difference of the value function of adjacent states are minimized, therefore, ϵ is minimized. Equation 21 shows that when ϵ is minimized, the nuclear norm of the Q table, $\|Q_B\|_*$ is also implicitly minimized and when ϵ is zero, Q_B is reduced to a rank 1 matrix.

5 Conclusion and future work

In this paper, two new ideas were introduced to regularize the value function in Deep Q-learning to have low rank and sparsity regularization and temporal smoothness regularization.

For both ideas, the experiment results showed better performance than baseline after hyper-parameter tuning. However, high variance is observed in the result of our experiments. For example, running the experiments several times with different random seeds has produced different results. Ideally, to conclude the effect on Deep Q-learning, the average result of more experiment trials need to be observed while we currently only run the training with three different random seeds for each setting. Also, the same method should be adopted to multiple Atari games to test whether the method generalizes. The future work of this paper would be to conduct extensive experiments to further evaluate these regularization methods.

References

- [1] Moon, T. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.
- [2] Ong, H. Y. Value function approximation via low-rank models. *CoRR*, abs/1509.00061, 2015.
- [3] Alora, J. I., A. A. Gorodetsky, S. Karaman, et al. Automated synthesis of low-rank control systems from sc-ltl specifications using tensor-train decompositions. In *55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016*, pages 1131–1138. IEEE, 2016.
- [4] Yang, Y., G. Zhang, Z. Xu, et al. Harnessing structures for value-based planning and reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [5] Samejima, K., K. Katagiri, K. Doya, et al. Multiple model-based reinforcement learning for nonlinear control. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 89(9):54–69, 2006.
- [6] Krogh, A., J. A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, R. Lippmann, eds., *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pages 950–957. Morgan Kaufmann, 1991.
- [7] Hu, Z., F. Nie, R. Wang, et al. Low rank regularization: A review. *Neural Networks*, 136:218–232, 2021.
- [8] Piché, A., J. Marino, G. M. Marconi, et al. Beyond target networks: Improving deep q-learning with functional regularization. *CoRR*, abs/2106.02613, 2021.
- [9] Rozada, S., A. G. Marques. Tensor and matrix low-rank value-function approximation in reinforcement learning. *ArXiv*, abs/2201.09736, 2022.
- [10] Grau-Moya, J., F. Leibfried, P. Vrancx. Soft q-learning with mutual-information regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [11] Peng, C., Z. Kang, H. Li, et al. Subspace clustering using log-determinant rank approximation. In L. Cao, C. Zhang, T. Joachims, G. I. Webb, D. D. Margineantu, G. Williams, eds., *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 925–934. ACM, 2015.
- [12] Kim, E., M. Lee, S. Oh. Elastic-net regularization of singular values for robust subspace learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 915–923. IEEE Computer Society, 2015.
- [13] Nie, F., H. Huang, C. H. Q. Ding. Low-rank matrix recovery via efficient Schatten p -norm minimization. In J. Hoffmann, B. Selman, eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012.
- [14] Mohan, K., M. Fazel. Iterative reweighted algorithms for matrix rank minimization. *J. Mach. Learn. Res.*, 13:3441–3473, 2012.
- [15] Hu, Y., D. Zhang, J. Ye, et al. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(9):2117–2130, 2013.
- [16] Oh, T. H., H. Kim, Y. Tai, et al. Partial sum minimization of singular values in RPCA for low-level vision. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 145–152. IEEE Computer Society, 2013.
- [17] Gu, S., L. Zhang, W. Zuo, et al. Weighted nuclear norm minimization with application to image denoising. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 2862–2869. IEEE Computer Society, 2014.

- [18] Gu, S., Q. Xie, D. Meng, et al. Weighted nuclear norm minimization and its applications to low level vision. *Int. J. Comput. Vis.*, 121(2):183–208, 2017.
- [19] Zhong, X., L. Xu, Y. Li, et al. A nonconvex relaxation approach for rank minimization problems. In B. Bonet, S. Koenig, eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 1980–1987. AAAI Press, 2015.
- [20] Shah, D., D. Song, Z. Xu, et al. Sample efficient reinforcement learning via low-rank matrix estimation. *CoRR*, abs/2006.06135, 2020.
- [21] Sang, T., H. Tang, J. Hao, et al. Uncertainty-aware low-rank q-matrix estimation for deep reinforcement learning. In J. Chen, J. Lang, C. Amato, D. Zhao, eds., *Distributed Artificial Intelligence*, pages 21–37. Springer International Publishing, Cham, 2022.
- [22] Behzadian, B., M. Petrik. Low-rank feature selection for reinforcement learning. In *ISAIM*. 2018.
- [23] Song, Z., R. E. Parr, X. Liao, et al. Linear feature encoding for reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.