Determinantal Reinforcement Learning with Techniques to Avoid Poor Local Optima

Takayuki Osogami and Rudy Raymond

1 Introduction

Reinforcement learning for multiple collaborative agents is important and has many practical applications. Consider a team that consists of different types of players. In many cases, to win a game, each type of player must master highly relevant actions that are necessarily different from the other types of player. For example, players of a defensive team should guard relevant and diverse areas or relevant and different types of players of the other team. This is also the case when controlling many similar robots to perform a task that cannot be performed by a single one. Training them in a way they take different and relevant actions can lead to faster and better convergence to optimal collaboration.

Typical approaches in reinforcement learning, which let each agent take actions independently of other agents, are therefore not effective. Formulating the learning as handling the combination of actions as if it is an action of a hypothetical agent can lead to searching in an exponentially larger action spaces that grow with the number of agents and therefore does not scale well.

In Osogami and Raymond (2019), we have proposed the use of the determinant of a matrix to approximate the action-value function in reinforcement learning that takes into account both relevance and diversity in a natural manner. When each action of an agent in a particular state is characterized by a feature vector, the length of the feature vector corresponds to the relevance of that action at that state. Meanwhile, the angle between two feature vectors represents the diversity between the two corresponding actions at that state. A set of feature vectors then comprises a parallelotope whose volume is determined by the lengths (i.e., relevances) and the angles (i.e., diversities) of

T. Osogami (✉) · R. Raymond
IBM Research—Tokyo, Tokyo, Japan
e-mail: osogami@jp.ibm.com

R. Raymond
e-mail: rudyhar@jp.ibm.com

J. Cheng et al. (eds.), *Proceedings of the Forum "Math-for-Industry" 2018*,
Mathematics for Industry 35, https://doi.org/10.1007/978-981-16-5576-0_5

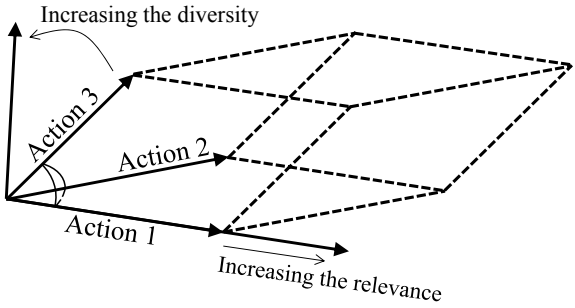

Fig. 1 The logarithm of the squared volume of the parallelepiped defined by the feature vectors of actions represents the value of the combination of those actions. The volume of the parallelepiped is determined by the length of vectors (their relevances) and their angles (their diversities). The volume can be increased by increasing relevance, diversity, or both

the feature vectors. An example of a parallelotope from three feature vectors in three dimensions is shown in Fig. 1. The figure also illustrates two ways the volume can be increased: increasing relevance and increasing diversity. The squared volume of the parallelotope can be computed from the determinant of the Gram matrix of the feature vectors. More specifically, the value of a combination of relevant and diverse actions at a state can be computed from the logarithm of the determinant (log-determinant) of the principal submatrix of a positive semidefinite matrix (kernel), where the principal submatrix is specified by the actions, and the kernel depends on the state.

In Osogami and Raymond (2019), we have derived efficient learning rules of determinantal SARSA (state-action-reward-state-action algorithm). Namely, we approximate the action-value function of N agents with an $N \times N$ kernel matrix whose effective dimension is $K \ll N$, so that at each iteration the action-value function can be updated with the additional $O(K^3)$ computational complexity. Determinantal SARSA has been shown to find nearly optimal policies approximately ten times faster than baseline approaches (Sallans and Hinton 2001, 2004; Heess et al. 2017; Sallans 2002), where free energy of a restricted Boltzmann machine (RBM) is used as a functional approximator. In this paper, we add theoretical insights and experiments on how to avoid policies with poor local optima by techniques based on derivation of the learning rules of determinantal SARSA.

2 Determinantal SARSA

In this section, we review determinantal SARSA, which we have proposed in Osogami and Raymond (2019). Determinantal SARSA considers the setting with a team of agents under central control. In the following, an agent team refers to the team of agents, and a team action refers to the combination of their actions. At each time t, the agent team makes an observation o_t. Let $\mathbf{z}_t \equiv \xi(a_{t-1}, r_t, o_t)$ represent the (features of) observation at time t, which may include the preceding team action a_{t-1}

and reward r_t in addition to o_t. Let $\mathbf{z}_{\leq t}$ denote the observations up to t. Depending on what has been observed (i.e., $\mathbf{z}_{\leq t}$), the agent team takes a team action a_t. Let $\mathbf{x}_t \equiv \psi(a_t) \in \{0, 1\}^N$ be a binary representation of a team action a_t (e.g.,, $\mathbf{x}_t$ may indicate which subset of N possible actions is taken by the agent team). The agent team then obtains reward r_{t+1}, and the environment changes its state. The agent team then makes a partial observation o_{t+1} of the environment and chooses the next team action a_{t+1}, and this process is continued. The goal of the agent team is to sequentially choose team actions so that the cumulative reward is maximized.

Given that the agent team has $\mathbf{z}_{\leq t}$, performs the action with the binary representation $\mathbf{x}$, and acts according to the policy under consideration, determinantal SARSA seeks to learn the Q (action value) function $Q_\theta(\mathbf{z}_{\leq t}, \mathbf{x})$ so that it best approximates the expected cumulative reward. By learning the Q function, one can identify the action that is optimal at a given state when one follows the policy under consideration from the next state. This allows one to iteratively improve the policy under consideration.

In determinantal SARSA, the Q function is assumed to have the following form:

$$Q_\theta(\mathbf{z}_{\leq t}, \mathbf{x}_t) \equiv \alpha + \log\det \mathbf{V}(\mathbf{x}_t)\, \mathrm{Diag}(\exp(\mathbf{d}_t(\phi)))\, \mathbf{V}(\mathbf{x}_t)^\top. \tag{1}$$

Here, $\mathbf{V}$ is an arbitrary $N \times K$ matrix for $0 < K \leq N$, and $\mathbf{V}(\mathbf{x}_t)$ denotes the matrix consisting of a subset of the rows of $\mathbf{V}$ in a way that the rows of $\mathbf{V}(\mathbf{x}_t)$ are indexed by the elements that are one in $\mathbf{x}_t$. Also, $\mathrm{Diag}(\cdot)$ denotes the diagonal matrix formed with a given vector, $\mathbf{d}_t(\phi)$ is a time-varying K-dimensional vector, and its exponentiation is elementwise. Here, $\mathbf{d}_t(\phi)$ should be considered as a time-series model, with parameter ϕ, that outputs a K-dimensional vector. Also, $\mathbf{d}_t(\phi)$ should be differentiable with respect to ϕ to allow end-to-end learning. Examples of such $\mathbf{d}_t(\phi)$ include recurrent neural networks (Hausknecht and Stone 2015), vector autoregressive models, and dynamic Boltzmann machines (Osogami and Otsuka 2015; Osogami 2017).

To intuitively understand the form of Q_θ in (1), consider the case where $\mathbf{V}$ is the identity matrix of order $K = N$. In this case, Q_θ is reduced to

$$Q_\theta(\mathbf{z}_{\leq t}, \mathbf{x}_t) = \alpha + \mathbf{d}_t(\phi)^\top \mathbf{x}. \tag{2}$$

If the i-th element of $\mathbf{x}$ indicates whether the i-th action is taken by an agent, the value of a team action is the sum of the values of individual actions without consideration of diversity, where $\mathbf{d}_t(\phi)$ represents the value (relevance) of individual actions at time t. With a non-identity $\mathbf{V}$, determinantal SARSA can take into account the diversity in actions.

Determinantal SARSA learns all of the parameters $\theta \equiv (\alpha, \mathbf{V}, \phi)$ in an end-to-end manner. Specifically, at each iteration, determinantal SARSA updates θ according to

$$\theta \leftarrow \theta + \eta \left(r_{t+1} + \rho\, Q_\theta(\mathbf{z}_{\leq t+1}, \mathbf{x}_{t+1}) - Q_\theta(\mathbf{z}_{\leq t}, \mathbf{x}_t)\right) \nabla_\theta Q_\theta(\mathbf{z}_{\leq t}, \mathbf{x}_t), \tag{3}$$

where we need the gradient $\nabla_\theta Q_\theta$. In Osogami and Raymond (2019), we have shown that the gradient can be represented in a computationally convenient form as follows:

Algorithm 1 Determinantal SARSA (Osogami and Raymond 2019)[1]

1: **Input:** Discount factor ρ; learning rate η; initial θ
2: Take initial team-action a_0; $\mathbf{x}_0 \leftarrow \psi(a_0)$
3: **for** $t = 0, 1, \ldots$ **do**
4: Get r_{t+1} and observe o_{t+1}; $\mathbf{z}_{t+1} \leftarrow \xi(a_t, r_{t+1}, o_{t+1})$
5: Take team-action a_{t+1}; $\mathbf{x}_{t+1} \leftarrow \psi(a_{t+1})$
6: $\mathbf{D}_t \leftarrow \mathrm{Diag}(\exp(\mathbf{d}_t(\phi)))$
7: Update $\mathbf{d}_t(\phi)$ to $\mathbf{d}_{t+1}(\phi)$ with $\mathbf{z}_{t+1}$
8: $\mathbf{D}_{t+1} \leftarrow \mathrm{Diag}(\exp(\mathbf{d}_{t+1}(\phi)))$
9: $Q_t \leftarrow \alpha + \log\det \mathbf{V}(\mathbf{x}_t)\, \mathbf{D}_t\, \mathbf{V}(\mathbf{x}_t)^\top$
10: $Q_{t+1} \leftarrow \alpha + \log\det \mathbf{V}(\mathbf{x}_{t+1})\, \mathbf{D}_{t+1}\, \mathbf{V}(\mathbf{x}_{t+1})^\top$
11: $\Delta_t \leftarrow r_{t+1} + \rho\, Q_{t+1} - Q_t$
12: $\alpha \leftarrow \alpha + \eta\, \Delta_t$
13: $\mathbf{V}(\mathbf{x}_t) \leftarrow \mathbf{V}(\mathbf{x}_t) + 2\,\eta\,\Delta_t\, (\mathbf{V}(\mathbf{x}_t)^+)^\top$
14: $\phi \leftarrow \phi + \eta\, \Delta_t\, \mathrm{diag}\left(\mathbf{V}(\mathbf{x}_t)^+\, \mathbf{V}(\mathbf{x}_t)\right)\, \nabla_\phi \mathbf{d}_t(\phi)$
15: **end for**

$$\nabla_\alpha Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = 1 \tag{4}$$

$$\nabla_{\mathbf{V}(\bar{\mathbf{x}})} Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = \mathbf{0} \tag{5}$$

$$\nabla_{\mathbf{V}(\mathbf{x})} Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = 2\,(\mathbf{V}(\mathbf{x})^+)^\top \tag{6}$$

$$\nabla_\phi Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = \mathrm{diag}\left(\mathbf{V}(\mathbf{x})^+\, \mathbf{V}(\mathbf{x})\right)\, \nabla_\phi \mathbf{d}_t(\phi) \tag{7}$$

where $\mathbf{V}(\mathbf{x})^+$ denotes the pseudo-inverse of $\mathbf{V}(\mathbf{x})$, $\mathrm{diag}(\cdot)$ denotes the vector formed with the diagonal elements of a given matrix, and $\bar{\mathbf{x}} \equiv 1 - \mathbf{x}$ elementwise.

Algorithm 1 gives a pseudocode of determinantal SARSA. In each iteration of the for-loop starting at Step 3, after getting reward r_{t+1} and making an observation o_{t+1} in Step 4, one takes a team action a_{t+1} in Step 5. Steps 6–8 compute the diagonal matrix $\mathbf{D}_t \equiv \mathrm{Diag}(\exp(\mathbf{d}_t(\phi)))$ by using the time-series model $\mathbf{d}_t(\phi)$, whose state is updated in Step 7 on the basis of the input $\mathbf{z}_t$. These diagonal matrices are then used in Steps 9–12 to compute the TD error Δ_t. The parameters $\theta \equiv (\alpha, \mathbf{V}, \phi)$ are then updated in Steps 12–14. In Step 14, the gradient $\nabla_\phi \mathbf{d}_t(\phi)$ depends on the particular time-series model under consideration. It is easy to estimate the computational time to update parameters at each iteration of determinantal SARSA. Since the rank of $\mathbf{V}$ is at most K, the computational complexity of the pseudo-inverse $\mathbf{V}(\mathbf{x}_t)^+$ and the computation of $\log\det \mathbf{V}(\mathbf{x}_t)\mathbf{D}_t\mathbf{V}(\mathbf{x}_t)^\top$ is $O(K^3)$.

Note that one may use determinantal SARSA to the fully observable case by letting $\mathbf{D}_t \equiv \mathrm{Diag}(\exp(\mathbf{d}_t(\phi)))$ in (1) be static but depend on the fully observed Markovian state s_t at time t. For example, one may use a feedforward neural network $\psi(\cdot)$ that maps a state s_t into a K-dimensional feature vector $\mathbf{d} = \psi(s_t)$, which then defines $\mathbf{D}_t = \mathrm{Diag}(\exp(\mathbf{d}))$. With $\mathbf{D}_t$ alone, the state can only influence the values

[1] Here, typographical errors in [10] are corrected by adding "$\top$" in Step 9–10 and removing "$^-$" in Step 13. In Step 13–14, the $\mathbf{V}(\mathbf{x}_t)^+$ is the pseudo inverse that will be slightly modified for improving the stability of Determinantal SARSA.

of individual actions, while we also allow the state to affect the diversity measure as well through $\mathbf{V}$.

In Step 2 and Step 5 of Algorithm 1, we need to choose team actions in consideration of the tradeoff between exploration and exploitation. One of popular approaches is Boltzmann exploration. In Osogami and Raymond (2019), we have shown that, for determinantal SARSA, the Boltzmann exploration with unit temperature reduces to sampling from a determinantal point process, which allows efficient (in time polynomial in N) sampling (Kulesza and Taskar 2012; Qiao et al. 2016; Kulesza and Taskar 2011). The Boltzmann exploration with general temperature for determinantal SARSA requires sampling from annealed determinantal distributions (Wachinger and Golland 2015; Belabbas and Wolfe 2009), for which practical sampling algorithm as Markov Chain Monte Carlo is available (Kang 2013; Gillenwater 2014).

3 Avoiding Poor Local Optima

In the experiments of Osogami and Raymond (2019), we have occasionally observed that determinantal SARSA is trapped into poor local optima. We hypothesize that this can happen because of the low-rank kernel approximation and the pseudo-inverse updates. Here, we show how we can avoid such poor local optima.

Our formulation of (1) assumes that the rank of the following $N \times N$ positive semidefinite matrix (kernel)

$$\mathbf{L}_t \equiv \mathbf{V}\,\mathrm{Diag}(\exp(\mathbf{d}_t(\phi)))\,\mathbf{V}^\top \tag{8}$$

is K. This is achieved by the use of the $N \times K$ matrix $\mathbf{V}$. However, depending on the initial values of $\mathbf{V}$, determinantal SARSA may fall into the situation where the rank of $\mathbf{V}$ (and its submatrix $\mathbf{V}(\mathbf{x})$) becomes smaller than K. Once determinantal SARSA is trapped into such $\mathbf{V}$s, it cannot search for parameters on the larger subspace. Namely, with (6), the parameters $\mathbf{V}(\mathbf{x})$ are updated by adding the terms that are proportional to $\left(\mathbf{V}(\mathbf{x})^+\right)^\top$. However, because of the property of the pseudo-inverse, we can observe that

$$\mathrm{Range}\,(\mathbf{V}(\mathbf{x})) = \mathrm{Range}\left(\left(\mathbf{V}(\mathbf{x})^+\right)^\top\right), \tag{9}$$

where $\mathrm{Range}\,(\mathbf{A})$ is the range or the image of a matrix $\mathbf{A}$. Thus, determinantal SARSA may not be able to reach optimal solutions from some initial values. This explains why determinantal SARSA can be trapped into local optima.

The above observation leads to mitigation techniques by keeping the rank of $\mathbf{V}(\mathbf{x})$ to be K during the computation. This can be achieved heuristically by adding some noises to $\mathbf{V}$ in the initialization and to the pseudo-inverse $\mathbf{V}(\mathbf{x})^+$ in each iteration.

First, we propose to initialize $\mathbf{V}$ using random special orthogonal matrices (Stewart 1980), $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A}$ is $N \times N$, and $\mathbf{B}$ is $K \times K$. Specifically, we initialize $\mathbf{V}$ as

$$\mathbf{V} = \mathbf{A}\,\Sigma\,\mathbf{B}^\top + \mathcal{E}, \tag{10}$$

where Σ is an $N \times K$ rectangular diagonal matrix in which every diagonal element is one, $\mathcal{E}$ is a random $N \times K$ matrix (in our experiments, and we will sample each element independently according to the uniform distribution with support $[-0.01, 0.01]$). Namely, the matrix $\mathbf{V}$ is initialized in a way such that each of its singular values is one with small noise $\mathcal{E}$. This particular initialization plays a rather important role in avoiding convergence to poor local optima.

Next, let

$$\mathbf{V} = \mathbf{A}\,\Sigma\,\mathbf{B}^\top \tag{11}$$

be the singular-value decomposition of $\mathbf{V}$. Then, instead of the pseudo-inverse

$$\mathbf{V}(\mathbf{x})^+ = \mathbf{B}\,\Sigma^{-1}\,\mathbf{A}^\top \tag{12}$$

in (6), we let determinantal SARSA use the following "noisy" pseudo-inverse:

$$\mathbf{V}(\mathbf{x})^+ \approx \mathbf{B}\left((1-\varepsilon_t)\,\Sigma^{-1} + \varepsilon_t\,\mathbf{M}\right)\mathbf{A}^\top, \tag{13}$$

where $\mathbf{M}_{ij} = \delta_{ij}$. We gradually reduce the magnitude of the noise ε_t over the iteration t.

4 Experiments

In our experiments, we evaluate the performance of determinantal SARSA with and without the new method of avoiding poor local optimal. We conduct our experiments on the blocker task (Sallans and Hinton 2001, 2004; Heess et al. 2017; Sallans 2002), for which we have shown in Osogami and Raymond (2019) that Determinantal SARSA outperforms baseline methods (Sallans and Hinton 2001, 2004; Heess et al. 2017; Sallans 2002). We closely follow the instances considered in Heess et al. (2017) and Osogami and Raymond (2019). All of the experiments are carried out with Python implementation on a workstation having 48 GB of memory and a 4.0 GHz CPU.

In the blocker task, we control an agent team, consisting of three agents, in a collaborative manner, where the goal is to let one of the agents reach the end zone, while two blockers hinder the agents. The field is a grid of four rows and seven columns. The three agents start at uniformly random positions in the top row. The two blockers, each occupies three consecutive squares, start at uniformly random positions in the bottom row. At each time step, each agent can move one step in one of the four directions or stay. After all of the agents take actions, each blocker moves one step to the right or to the left if doing so can block an agent; otherwise, the blocker stays. If one of the agents reaches the end zone, the agent team receives $+1$ reward for that time step. Otherwise, the agent team incurs -1 reward per time

step. See Heess et al. (2017) and Osogami and Raymond (2019) for more details about the exact settings. Note that the blocker task is performed on a fully observable environment. In Osogami and Raymond (2019), we also evaluate determinantal SARSA with stochastic policy tasks, where the environment is partially observable.

As we have done in Osogami and Raymond (2019), we represent the team action a_t by a $4 \times 7 = 28$ dimensional binary vector $\mathbf{x}$, where each element indicates whether an agent occupies a particular square after taking the team action a_t ($(\mathbf{x}_t)_i = 1$) or not. Likewise, we set $\mathbf{D}_t \equiv \mathbf{I}$ and $\alpha = 0$. Hyperparameters of determinantal SARSA are set as in Osogami and Raymond (2019).

Figure 2 shows the performance of determinantal SARSA without the use of the technique of avoiding poor local optima. Specifically, the average reward per step is evaluated for every 40,000 steps (and for every 4000 steps during the initial 40,000

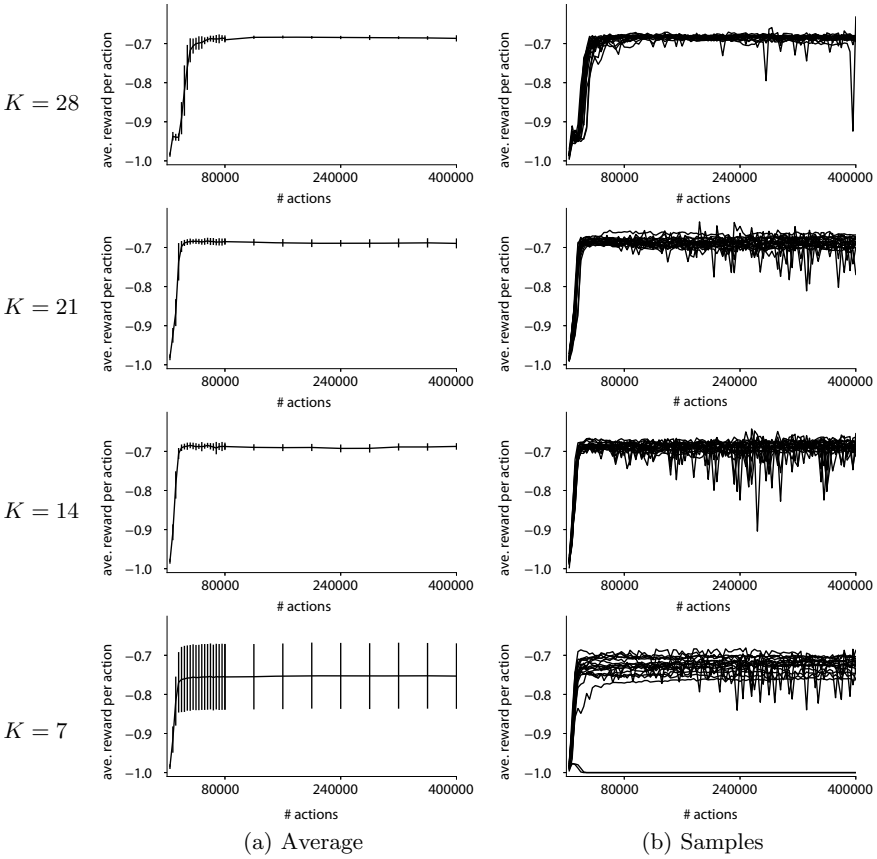

Fig. 2 Performance of determinantal SARSA with various ranks of kernels, where the rank K is indicated in the leftmost column. The panels in (**a**) show the mean and the standard deviation, over 20 runs, of the average reward per action. The panels in (**b**) show the average reward per action for each of the 20 runs

steps). The rank K of the kernel is varied as indicated in each row. Figure 2a shows the mean and the standard deviation, over 20 runs, of the average reward per action. Figure 2b shows the average reward per action for each of the 20 runs.

We can observe in Fig. 2a that reducing rank K has only a small impact on the average performance for $K \geq 14$. However, a significant degradation in performance

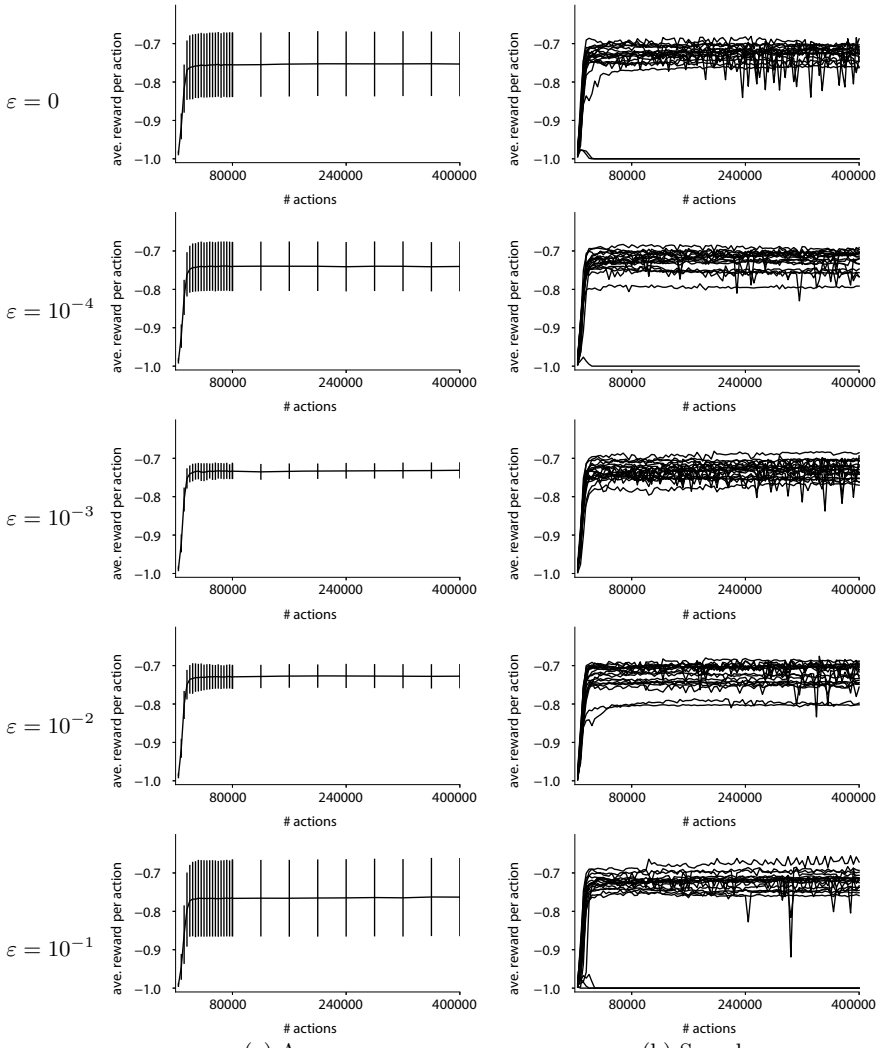

Fig. 3 Performance of determinantal SARSA with noisy initialization and pseudo-inverse when $K = 7$, where the magnitude ε of the noise at every 10,000-th iteration is indicated in the leftmost column. The panels in (**a**) show the mean and the standard deviation, over 20 runs, of the average reward per action. The panels in (**b**) show the average reward per action for each of the 20 runs

is observed with $K = 7$. Figure 2b shows that determinantal SARSA sometimes converges to poor local optima with $K = 7$. In particular, it has converged to the average reward of -1 (the lowest possible average reward) in two out of 20 runs.

Figure 3 shows the performance of determinantal SARSA with $K = 7$ when the technique of mitigating poor local optima is applied. Here, the magnitude of the noise at the t-th iteration (ε_t in (13)) is defined to be

$$\varepsilon_t = \varepsilon^{\frac{t}{10^4}}. \tag{14}$$

Namely, $\varepsilon_0 = 1$ and $\varepsilon_{10^4} = \varepsilon$, where various values of ε are tested as indicated in the leftmost column in Fig. 3. It suggests that determinantal SARSA can avoid convergence to poor local optima by the use of noisy gradient with appropriate magnitude of noise (specifically, $10^{-3} \leq \varepsilon \leq 10^{-2}$).

5 Conclusion

In Osogami and Raymond (2019), we have introduced determinantal SARSA, which uses the determinant of a matrix so that both diversity and relevance of team actions can be taken into account in reinforcement learning. Determinantal SARSA has been shown to substantially outperform existing methods proposed for coping with high-dimensional action space in multi-agent reinforcement learning. Determinantal SARSA can effectively deal with exponentially large team action space. When there are 2^N possible team actions, determinantal SARSA has at most $O(N^3)$ computational complexity and can have smaller complexity by assuming a low rank structure.

However, we find that determinantal SARSA with low-rank kernels can result in poor local optima. In this paper, we have proposed techniques of noisy initialization and noisy pseudo-inverse to avoid the poor local optima in determinantal SARSA. The results of numerical experiments support the effectiveness of the proposed techniques.

Acknowledgements This work was supported by JST CREST Grant Number JPMJCR1304, Japan.

References

Belabbas MA, Wolfe PJ (2009) On landmark selection and sampling in high-dimensional data analysis. Philos Trans R Soc: Math Phys Eng Sci 367:4295–4312

Gillenwater J (2014)Approximate inference for determinantal point processes. Ph.D. thesis

Hausknecht M, Stone P (2015) Deep recurrent Q-learning for partially observable MDPs. In: Sequential decision making for intelligent agents: papers from the AAAI 2015 fall symposium, pp 29–37

Heess N, Silver D, Teh YW (2013) Actor-critic reinforcement learning with energy-based policies. In: Proceedings of the 10th European workshop on reinforcement learning, vol 24., Edinburgh, Scotland, pp 45–58

Kang B (2013) Fast determinantal point process sampling with application to clustering. Adv Neural Inf Process Syst 26:2319–2327

Kulesza A, Taskar B (2011) k-DPPs: fixed-size determinantal point processes. In: Proceedings of the 28th international conference on machine learning, pp 1193–1200

Kulesza A, Taskar B (2012) Determinantal point processes for machine learning. Now Publishers Inc., Hanover, MA, USA

Osogami T (2017) Boltzmann machines for time-series. Technical report RT0980, IBM Research—Tokyo

Osogami T, Otsuka M (2015) Seven neurons memorizing sequences of alphabetical images via spike-timing dependent plasticity. Sci Rep 5:14149

Osogami T, Raymond R (2019) Determinantal reinforcement learning. In: Proceedings of the 33nd AAAI conference on artificial intelligence, pp 4659–4666

Qiao M, Xu RYD, Bian W, Tao D (2016) Fast sampling for time-varying determinantal point process. ACM Trans Knowl Discovery Data 11(Article No. 8)

Sallans B (2002) Reinforcement learning for factored Markov decision processes. Ph.D. thesis

Sallans B, Hinton GE (2001) Using free energies to represent Q-values in a multiagent reinforcement learning task. Adv Neural Inf Process Syst 13:1075–1081

Sallans B, Hinton GE (2004) Reinforcement learning with factored states and actions. J Mach Learn Res 5:1063–1088

Stewart GW (1980) The efficient generation of random orthogonal matrices with an application to condition estimators. SIAM J Numer Anal 17(3):403–409

Wachinger C, Golland P (2015) Sampling from determinantal point processes for scalable manifold learning. In: Proceedings of the international conference on information processing in medical imaging, pp 687–698