Determinantal Reinforcement Learning

Takayuki Osogami, Rudy Raymond
IBM Research - Tokyo
Tokyo, Japan

Abstract

We study reinforcement learning for controlling multiple agents in a collaborative manner. In some of those tasks, it is insufficient for the individual agents to take relevant actions, but those actions should also have diversity. We propose the approach of using the determinant of a positive semidefinite matrix to approximate the action-value function in reinforcement learning, where we learn the matrix in a way that it represents the relevance and diversity of the actions. Experimental results show that the proposed approach allows the agents to learn a nearly optimal policy approximately ten times faster than baseline approaches in benchmark tasks of multi-agent reinforcement learning. The proposed approach is also shown to achieve the performance that cannot be achieved with conventional approaches in partially observable environment with exponentially large action space.

Introduction

We study the problem of learning to control multiple agents in a collaborative manner. For example, players of a team want to learn from their experience how to collaboratively play to win a game, or one might want to learn to control multiple robots to accomplish a task that cannot be handled by a single robot. For some of those multi-agent tasks, it is important for the agents to take not only individually relevant but also collectively diverse actions. For example, players of a defensive team should guard relevant and diverse areas (*i.e.*, zone defense) or relevant and diverse players of the other team (*i.e.*, man-to-man defense).

Even when we have central control, multi-agent reinforcement learning is hard due to exponentially many possible combinations of actions. The simple approach of handling the combination of actions as if it is an action of a hypothetical single agent (or a team) does not scale with increased number of agents. It is also undesirable to let each agent greedily take an action without consideration of others.

We propose to use the determinant of a matrix to approximate the action-value function in reinforcement learning where the combination of relevant and diverse actions tends to have high value (see Figure 1). Each action is characterized by a feature vector, whose length represents the relevance of that action at a state, and the angle between two feature vectors represents the similarity between two actions at that state. The feature vector depends on the state. A set of feature vectors then defines a parallelotope whose squared volume is given by the determinant of the Gram matrix of those feature vectors. More specifically, the value of a combination of actions at a state is given by the logarithm of the determinant (log-determinant) of a principal submatrix of a positive semidefinite matrix (kernel), where the principal submatrix is specified by the actions, and the kernel depends on the state.

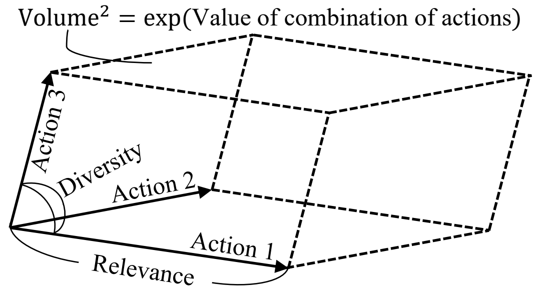

$\text{Volume}^2 = \exp(\text{Value of combination of actions})$
Action 3
Diversity
Action 2
Action 1
Relevance

Figure 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 proposed approach can deal with partial observability by letting the kernel depend on the history of observations (of non-Markovian states). The special case of a history-dependent diagonal kernel reduces to the standard approach of representing the action-value function with a time-series model such as a recurrent neural network (Hausknecht and Stone 2015), vector autoregressive models, and dynamic Boltzmann machines (Osogami and Otsuka 2015; Osogami 2017). Our approach can thus be seen as adding a differential "determinantal layer" to the output of a neural network.

We derive specific learning rules of the SARSA (state-action-reward-state-action) algorithm with log-determinant as a functional approximator. We refer to the resulting algorithm as Determinantal SARSA. We apply Determinantal SARSA to blocker tasks (fully observable environment) and stochastic policy tasks (partially observable environment). The experimental results suggest that Determinantal SARSA can find nearly optimal policy for blocker tasks approximately 10 times faster than baseline approaches (Sallans and Hinton 2001; 2004; Heess, Silver, and Teh 2013; Sallans 2002), where free energy of a restricted Boltzmann machine (RBM) is used as a functional approximator. Determinantal SARSA also finds nearly optimal policy for stochastic policy tasks, substantially outperforming those baseline approaches, due to its capability of effectively dealing with partial observability.

Our contributions can be summarized as follows. First, we introduce the idea of using the determinant in reinforcement learning to approximate the action-value function. Second, we derive specific learning rules of Determinantal SARSA. Finally, we demonstrate the effectiveness of Determinantal SARSA in reinforcement learning where relevant and diverse actions tend to have high value possibly in partially observable environment.

Related Work

Determinantal SARSA is motivated by determinantal point processes (DPPs) (Kulesza and Taskar 2012; Macchi 1975) and Free Energy SARSA (Sallans and Hinton 2001; 2004; Sallans 2002). Here, we discuss such prior work related to ours. Note that most of the prior work on multi-agent reinforcement learning (e.g., Gupta, Egorov, and Kochenderfer; Foerster et al. (2017; 2017)) except those that will be discussed in the following assumes factored action space, letting each agent choose an action independently although based on (partially) shared information. Such prior work is complementary to our approach of jointly choosing the actions, taking into account diversity.

A DPP defines a probability distribution over the subsets from a ground set. The probability of a subset is proportional to the determinant of a principal submatrix of a positive semidefinite matrix, where the submatrix is indexed by the items in the subset. A DPP thus assigns high probability to those subsets that have relevant and diverse items. DPPs have been used in machine learning applications, including recommendation of products (Gillenwater et al. 2014; Gartrell, Paquet, and Koenigstein 2017), summarization of documents or videos (Gong et al. 2014), hyper-parameter optimization (Kathuria, Deshpande, and Kohli 2016), and mini-batch sampling (Zhang, Kjellström, and Mandt 2017). DPPs have also been used for modeling neural spiking to better represent the negative correlation between neurons (Snoek, Zemel, and Adams 2013). DPPs, however, have never been used in reinforcement learning. We will see that a DPP naturally appears with Determinantal SARSA when we choose actions according to the standard approach of Boltzmann exploration.

Some of the steps of Determinantal SARSA are related to learning algorithms for DPPs. Specifically, we assume that the kernel has the structure similar to Low Rank DPP (Gartrell, Paquet, and Koenigstein 2017) and its extension to Dynamic DPP (Osogami et al. 2018). As a result, Determinantal SARSA involves the gradient of the log determinant that also appears in the learning algorithms in Gartrell, Paquet, and Koenigstein; Osogami et al. (2017; 2018). As we will see, however, the exact structure of our kernel is different from those studied in Gartrell, Paquet, and Koenigstein; Osogami et al. (2017; 2018). Specifically, our kernel has a smaller number of time-varying parameters and is more suitable for reinforcement learning where collecting a large amount of training data is relatively difficult. There also exists prior work on efficiently learning DPPs by assuming other structures in the kernel, such as tensor decomposition (Mariet and Sra 2016), that may be exploited in Determinantal SARSA as well.

Free Energy SARSA uses the free energy of an RBM as a functional approximator for multi-agent reinforcement learning, where the property of the RBM that allows efficiently sampling from a high dimensional space according to a Boltzmann distribution is exploited. There is also related work that uses expected energy of an RBM (Elfwing, Uchibe, and Doya 2016). While a DPP always represents negative correlation, an RBM can represent both negative and positive correlation. This flexibility of the RBM leads to greater generality but comes at the expense of greater computational complexity. Determinantal SARSA is suitable for the domain where a combination of relevant and diverse actions tends to have high value, because we can find a good value function from a limited space. Free Energy SARSA has been extended to an Actor Critic method (Heess, Silver, and Teh 2013). Determinantal SARSA may also allow such extension.

Determinantal SARSA

SARSA is a method of reinforcement learning and is usually applied to the settings with a single agent, but that single agent may be considered as a team of agents with central control, which is the setting we study. 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 observes the state s_t and takes a team-action a_t, depending on s_t. The agent-team then obtains reward r_{t+1} and transitions into state s_{t+1}, where the agent-team 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 in a way that cumulative reward is maximized.

SARSA seeks to learn a Q (action-value) function $Q(s, a)$, which represents the expected cumulative reward that can be obtained from a state s by taking action a at s and then act according to a policy under consideration. By learning the Q function, one can identify the action that is optimal at a given state when we follow the policy under consideration from the next state. This allows one to iteratively improve the policy under consideration.

More specifically, SARSA iteratively updates the Q function according to

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta\, \Delta_t \tag{1}$$

at each time $t+1$, where η is the learning rate, and Δ_t is the temporal difference (TD) error with a discount factor ρ for $0 \leq \rho \leq 1$:

$$\Delta_t \equiv r_{t+1} + \rho\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t). \tag{2}$$

SARSA usually assumes that the Markovian state s_t is observable. When s_t is not observable, a common practice is to let s_t represent (a feature vector of) the history of observations by time t.

Approximating Q Function Using Determinant

When the Q function is approximated with a function Q_θ with parameters θ, SARSA may update θ according to[1]

$$\theta \leftarrow \theta + \eta\, \Delta_t\, \nabla_\theta Q_\theta(s_t, a_t). \tag{3}$$

In Determinantal SARSA, we use determinant in Q_θ. Specifically, let $\mathbf{x}_t \equiv \psi(a_t) \in \{0,1\}^N$ be a binary representation of a team-action a_t. For example, $\mathbf{x}_t$ may indicate which subset of N possible actions is taken by the agent-team. 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 the partial observation o_t of s_t. Let $\mathbf{z}_{\le t}$ denote the observations by time t. We approximate the Q function with

$$Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}_t) \equiv \alpha + \log\det \mathbf{L}_t(\mathbf{x}_t), \tag{4}$$

where $\mathbf{L}_t$ is a positive semidefinite (hence symmetric) $N \times N$ matrix (kernel), which can vary over time t depending on $\mathbf{z}_{\le t}$, and $\mathbf{L}_t(\mathbf{x}_t)$ is the principal submatrix of $\mathbf{L}_t$ indexed by the elements that are 1 in $\mathbf{x}_t$ (*i.e.*, $\mathbf{L}_t(\mathbf{x}_t)$ is obtained from $\mathbf{L}$ by removing rows and columns, where the i-th row and column are removed iff the i-th element of $\mathbf{x}_t$ is 0 for any i). We define $\log\det \mathbf{L}_t(\mathbf{0}) = 0$, so that α determines the baseline value[2] at $\mathbf{x}_t = \mathbf{0}$.

For simplicity, we do not consider which actions should be assigned to which agents. When the agents are homogeneous, one may arbitrarily assign actions to agents once a subset is determined. For heterogeneous agents, one could consider the product space of the action spaces and choose one action from each action space.

We propose to represent $\mathbf{L}_t$ with the following form:

$$\mathbf{L}_t = \mathbf{V}\, \mathbf{D}_t\, \mathbf{V}^\top, \tag{5}$$

where $\mathbf{V}$ is an arbitrary $N \times K$ matrix for $0 < K \le N$, $\mathbf{D}_t$ is a diagonal matrix of order K with positive elements that can depend on $\mathbf{z}_{\le t}$. To ensure positivity, let

$$\mathbf{D}_t = \mathrm{Diag}(\exp(\mathbf{d}_t(\phi))) \tag{6}$$

for a K-dimensional vector $\mathbf{d}_t(\phi)$, where exponentiation is elementwise, and $\mathrm{Diag}(\cdot)$ denotes the diagonal matrix formed with a given vector. 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 a recurrent neural network and a vector autoregressive model.

To intuitively understand this $\mathbf{L}_t$, consider the case where $\mathbf{V}$ is the identity matrix of order $K = N$:

$$\begin{aligned}\log\det \mathbf{L}_t(\mathbf{x}) &= \log\det \mathbf{D}_t(\mathbf{x}) && (7)\\ &= \log \prod_{i:(\mathbf{x})_i=1} \exp(\mathbf{d}_t(\phi)_i) && (8)\\ &= \mathbf{d}_t(\phi)^\top \mathbf{x}. && (9)\end{aligned}$$

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 according to (3), where we need the gradient $\nabla_\theta Q_\theta$. The following theorem provides the $\nabla_\theta Q_\theta$ for the $Q_\theta(\mathbf{z}_{\le t}, \mathbf{x})$ in (4) that is used in Determinantal SARSA:

Theorem 1. *Consider the* $Q_\theta(\mathbf{z}_{\le t}, \mathbf{x})$ *in* (4) *with* $\mathbf{L}_t$ *as defined with* (5)-(6). *Let* $\mathbf{V}(\mathbf{x})$ *denote the matrix consisting of a subset of the rows of* $\mathbf{V}$ *in a way that the rows of* $\mathbf{V}(\mathbf{x})$ *are indexed by the elements that are one in* $\mathbf{x}$. *Let* $\mathbf{V}(\mathbf{x})^+$ *be the pseudo inverse of* $\mathbf{V}(\mathbf{x})$. *Let* $\bar{\mathbf{x}} \equiv 1 - \mathbf{x}$ *elementwise. Then*

$$\nabla_\alpha Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = 1 \tag{10}$$

$$\nabla_{\mathbf{V}(\bar{\mathbf{x}})} Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = \mathbf{0} \tag{11}$$

$$\nabla_{\mathbf{V}(\mathbf{x})} Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = 2\, (\mathbf{V}(\mathbf{x})^+)^\top \tag{12}$$

$$\nabla_\phi Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = \mathrm{diag}\left(\mathbf{V}(\mathbf{x})^+\, \mathbf{V}(\mathbf{x})\right)\, \nabla_\phi \mathbf{d}_t(\phi) \tag{13}$$

where $\mathrm{diag}(\cdot)$ *is the vector formed with the diagonal elements of a given matrix.*

Proof. Because $\nabla_\alpha Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}) = 1$ follows immediately from (4), we will prove (11)-(13). Let $\mathbf{U} \equiv \mathbf{V}(\mathbf{x})\sqrt{\mathbf{D}_t}$, where $\sqrt{\mathbf{D}_t} = \mathrm{Diag}(\exp(\mathbf{d}_t(\phi)/2))$. Then $\mathbf{L}_t(\mathbf{x}) = \mathbf{U}\,\mathbf{U}^\top$. The derivative of $\log\det \mathbf{L}_t(\mathbf{x})$ can be derived from the following equality for a generic matrix $\mathbf{Y}$:

$$\begin{aligned}\frac{\partial \log\det(\mathbf{Y}\,\mathbf{Y}^\top)}{\partial\theta} &= \sum_{i,j} \frac{\partial(\mathbf{Y}^\top)_{i,j}}{\partial\theta} \frac{\partial \log\det(\mathbf{Y}\,\mathbf{Y}^\top)}{\partial(\mathbf{Y}^\top)_{i,j}} \\ &= \sum_{i,j} \frac{\partial \mathbf{Y}_{j,i}}{\partial\theta} 2\,(\mathbf{Y}^+)_{i,j} && (14)\\ &= 2\,\mathrm{tr}\left(\frac{\partial \mathbf{Y}}{\partial\theta}\mathbf{Y}^+\right), && (15)\end{aligned}$$

where $\mathbf{Y}^+$ denotes the pseudo-inverse of $\mathbf{Y}$. Using (15), we can derive the derivative with respect to the (i,j)-th element of $\mathbf{V}$:

$$\frac{\partial}{\partial V_{i,j}} \log\det \mathbf{L}_t(\mathbf{x}) = 2\,\mathrm{tr}\left(\frac{\partial \mathbf{V}(\mathbf{x})}{\partial V_{i,j}}\sqrt{\mathbf{D}_t}\,\mathbf{U}^+\right). \tag{16}$$

Because $\sqrt{\mathbf{D}_t}$ is diagonal and invertible, we have

$$\mathbf{U}^+ = \sqrt{\mathbf{D}_t}^{-1}\,\mathbf{V}(\mathbf{x})^+. \tag{17}$$

When $x_i = 1$, by (16)-(17), we have

$$\begin{aligned}\frac{\partial}{\partial V_{i,j}} \log\det \mathbf{L}_t(\mathbf{x}) &= 2\,\mathrm{tr}\left(\frac{\partial \mathbf{V}(\mathbf{x})}{\partial V_{i,j}}\mathbf{V}(\mathbf{x})^+\right) && (18)\\ &= 2\,\mathrm{tr}\left(\mathbf{E}_{i',j}\,\mathbf{V}(\mathbf{x})^+\right), && (19)\end{aligned}$$

where $\mathbf{E}_{i',j}$ is the matrix whose elements are zero except that the (i',j)-th element is one, where i' is the row of $\mathbf{V}(\mathbf{x})$ corresponding to the i-th row of $\mathbf{V}$. Then we have

$$\frac{\partial}{\partial V_{i,j}} \log\det \mathbf{L}_t(\mathbf{x}) = 2\,\left((\mathbf{V}(\mathbf{x})^+)^\top\right)_{i',j}, \tag{20}$$

[1]When Q_θ is non-linear, convergence is not generally guaranteed for SARSA (Baird 1995; Boyan and Moore 1995). One may use more sophisticated techniques (Maei and Sutton 2010; Maei et al. 2009) for better convergence, but we choose the simple framework of SARSA in this paper.

[2]An extension to a time-varying α_t is straightforward, analogously to other parameters.

Algorithm 1 Determinantal SARSA

```
1: Input: Discount factor $\rho$; learning rate $\eta$; initial $\theta$
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:     $Q_t \leftarrow \alpha + \log\det \mathbf{V}(\mathbf{x}_t)\, \mathbf{D}_t\, \mathbf{V}(\mathbf{x}_t)$
10:    $Q_{t+1} \leftarrow \alpha + \log\det \mathbf{V}(\mathbf{x}_{t+1})\, \mathbf{D}_{t+1}\, \mathbf{V}(\mathbf{x}_{t+1})$
11:    $\Delta_t \leftarrow r_{t+1} + \rho\, Q_{t+1} - Q_t$
12:    $\alpha \leftarrow \alpha + \eta\, \Delta_t$
13:    $\mathbf{V}(\bar{\mathbf{x}}_t) \leftarrow \mathbf{V}(\bar{\mathbf{x}}_t) + 2\,\eta\,\Delta_t\, (\mathbf{V}(\mathbf{x}_t)^+)^\top$
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
```

which establishes (12). Notice that (11) follows from the fact that $\mathbf{L}_t(\mathbf{x})$ does not involve $\mathbf{V}(\bar{\mathbf{x}})$.

Next, consider the derivative with respect to the i-th element of $\mathbf{d}_t(\phi)$:

$$\frac{\partial}{\partial d_t(\phi)_i} \log\det \mathbf{L}_t(\mathbf{x})$$

$$= 2\,\mathrm{tr}\left(\mathbf{V}(\mathbf{x})\, \frac{\partial \sqrt{\mathbf{D}_t}}{\partial d_{t,i}}\, \sqrt{\mathbf{D}_t}^{-1}\, (\mathbf{V}(\mathbf{x}))^+\right) \tag{21}$$

$$= 2\,\mathrm{tr}\left(\mathbf{V}(\mathbf{x})\, \frac{1}{2}\, \exp(d_{t,i}/2)\, \mathbf{E}_{i,i}\, \sqrt{\mathbf{D}_t}^{-1}\, \mathbf{V}(\mathbf{x})^+\right) \tag{22}$$

$$= \mathrm{tr}\left(\mathbf{V}(\mathbf{x})\, \mathbf{E}_{i,i}\, \mathbf{V}(\mathbf{x})^+\right) \tag{23}$$

We thus have

$$\nabla_{\mathbf{d}_t(\phi)} \log\det \mathbf{L}_t(\mathbf{x}) = \mathrm{diag}\left(\mathbf{V}(\mathbf{x})^+\, \mathbf{V}(\mathbf{x})\right), \tag{24}$$

which then implies (13). □

Algorithm 1 summarizes Determinantal SARSA. In each iteration, after getting reward r_{t+1} and making an observation o_{t+1} in Step 4, we take a team-action a_{t+1} in Step 5. Steps 6-8 compute the diagonal matrix of (6) 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 9-11 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.

In practice, one may let $\mathbf{V} = \mathbf{I} + \mathbf{A}$ and learn $\mathbf{A}$ with L2 regularization such that the Frobenius norm of $\mathbf{A}$ tends to be small, which is expected to help avoid overfitting to limited training data. Notice that this does not lose generality, because $\mathbf{A}$ is arbitrary. Because $\mathbf{V} \to \mathbf{I}$ as $\mathbf{A} \to \mathbf{O}$, strong regularization reduces to the standard method of learning only the relevance $\mathbf{d}_t(\phi)$ of individual actions, ignoring diversity. We will use such L2 regularization in our experiments.

Choosing Actions with Determinantal SARSA

In Step 2 and Step 5 of Algorithm 1, we need to choose team-actions in consideration of the tradeoff between exploration and exploitation. Popular approaches include ε-greedy and Boltzmann exploration. The ε-greedy method chooses the action having the highest value with probability $1-\varepsilon$, which is generally intractable for high dimensional action space. In the following, we will see that the structure of log-determinant allows Boltzmann exploration that runs efficiently in practice.

In Boltzmann exploration, a team-action a having feature $\mathbf{x} = \phi(a)$ is chosen at time t with probability

$$\pi(\mathbf{x} \mid \mathbf{z}_{\le t}) = \frac{\exp(\beta\, Q_\theta(\mathbf{z}_{\le t}, \mathbf{x}))}{\sum_{\tilde{\mathbf{x}}} \exp(\beta\, Q_\theta(\mathbf{z}_{\le t}, \tilde{\mathbf{x}}))} \tag{25}$$

$$= \frac{\det \mathbf{L}_t(\mathbf{x})^\beta}{\sum_{\tilde{\mathbf{x}}} \det \mathbf{L}_t(\tilde{\mathbf{x}})^\beta}, \tag{26}$$

where β is the inverse temperature, and the summation with respect to $\tilde{\mathbf{x}}$ is over the binary feature vectors that correspond to all of the possible team-actions.

When $\beta = 1$ and the summation with respect to $\tilde{\mathbf{x}}$ is over all of the possible 2^N binary vectors, (26) is reduced to a DPP, which allows efficient (in time polynomial in N) sampling (Kulesza and Taskar 2012). The low rank structure ($K < N$ in (5)) can be exploited for further efficiency (Qiao et al. 2016). Also, when the size of the subset is restricted to a given constant k (*e.g.*, each action in the subset corresponds to the action of one of the k agents consisting of a team), (26) is reduced to a k-DPP (Kulesza and Taskar 2011).

The case with $\beta \neq 1$ has been studied as annealed determinantal distributions (Wachinger and Golland 2015; Belabbas and Wolfe 2009). This general case is less tractable than the DPP, but one can still draw samples via Markov Chain Monte Carlo (MCMC) methods. Namely, starting from a random binary vector $\mathbf{x}$, we iteratively choose a candidate vector $\mathbf{x}'$ and replace $\mathbf{x} \leftarrow \mathbf{x}'$ with acceptance probability: $\min\{1, (\det \mathbf{L}_t(\mathbf{x}')/\det \mathbf{L}_t(\mathbf{x}))^\beta\}$.

When each $\mathbf{x}$ corresponds to a candidate team-action $a = \phi^{-1}(\mathbf{x})$, such an MCMC method can be accelerated in a way similar to the the approach studied in Kang; Gillenwater (2013; 2014) for the DPP. In this case, we can choose the candidate vector $\mathbf{x}'$ in a way that it differs from $\mathbf{x}$ by only one bit, which may be sampled uniformly at random from $\{1, \ldots, N\}$. Then $\mathbf{L}(\mathbf{x})$ and $\mathbf{L}(\mathbf{x})'$ differs only by one rank, and the ratio of their determinants can be computed efficiently using the Schur determinant identity and rank-one update techniques, as shown in Kang; Osogami et al. (2013; 2018). The only difference from the case of the DPP is that the ratio of the determinant is powered to β in the acceptance probability.

In practice, one may be more exploratory than a DPP by mixing the DPP ($\beta = 1$) and the uniform distribution ($\beta \to 0$) with suitable probabilities. To be more exploitative, one may sample from the DPP with a suitably chosen number of times and select the one having the highest value. Because we can sample efficiently from a DPP, these heuristics allow efficient sampling, while trading off between exploration and exploitation. In our experiments, we use MCMC to sample with Boltzmann exploration, which we find tends to work better for our applications.

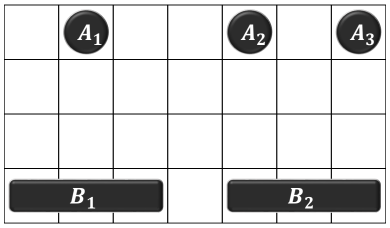


Figure 2: An example of the initial positions of the agents (A_1, A_2, A_3) and blockers (B_1, B_2) in the blocker task.

Experiments

We evaluate the performance of Determinantal SARSA on the blocker task (Sallans and Hinton 2001; 2004; Heess, Silver, and Teh 2013; Sallans 2002) and the stochastic policy task (Heess, Silver, and Teh 2013; Sallans 2002), which have been designed to evaluate the performance of multi-agent reinforcement learning methods. The blocker task is on a fully observable environment, while the stochastic policy task involves partial observability. We compare the proposed approach against the existing approaches to multi-agent reinforcement learning with non-factored action structures (*i.e.*, taking into account the value of a team-action rather than the value of the individual action of an agent) (Sallans and Hinton 2001; 2004; Heess, Silver, and Teh 2013; Sallans 2002). We closely follow the instances considered in Heess, Silver, and Teh (2013) and compare our results against those reported in Heess, Silver, and Teh (2013). All of the experiments are carried out with Python implementation on a workstation having 48 GB memory and 4.0 GHz CPU.

Blocker Task

In the blocker task, we seek to control an agent-team (consisting of three agents) in a collaborative manner so that one of the agents reaches the end zone, while two blockers hinder the agents. Figure 2 shows an example of the initial positions of the agents (A_1, A_2, A_3) and the blockers (B_1, B_2). The field is a grid of four rows and seven columns. Each agent starts at uniformly random positions in the top row. Each blocker, who occupies three squares, starts at uniformly random positions in the bottom row. The goal of the agent-team is to let one of the agents reach the end zone (bottom row) by avoiding the blockers.

At each step, each agent can move one step in one of the four directions or stay unmoved. 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 unmoved. If one of the agents reaches the end zone, the agent-team receives $+1$ reward for that step. Otherwise, the agent-team incurs -1 reward per step. See Heess, Silver, and Teh (2013) for more details about the exact settings.

For this fully observable task, we learn to control the agents via Determinantal SARSA with $\mathbf{D}_t \equiv \mathbf{I}$. Here, we represent the team-action a_t by a $4 \times 7 = 28$ dimensional binary vector $\mathbf{x}$, where each dimension corresponds to one of the 4×7 squares constituting the grid. Specifically, the i-th dimension indicates whether an agent occupies the i-th square after taking the team-action a_t ($(\mathbf{x}_t)_i = 1$) or not. We use a full rank kernel (*i.e.*, 28×28 matrix $\mathbf{V}$), which tends to perform best for this particular task. Also, α in (4) is omitted, because $\mathbf{x}_t = \mathbf{0}$ never appears in this task. In Boltzmann exploration, we sample a team-action according to the exact distribution of 26 without the use of MCMC, because exact sampling is tractable for this task, where the number of feasible team-actions is at most $5^3 = 125$ and can be much smaller in some states.

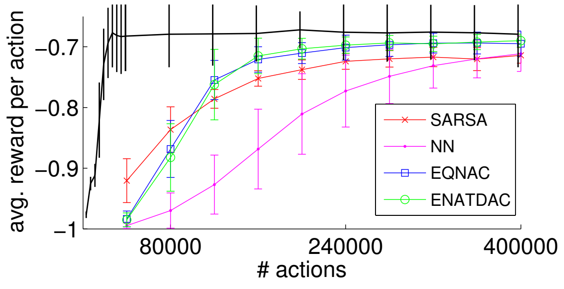


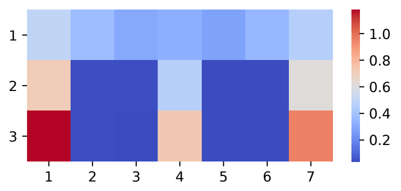
Figure 3: Performance of Determinantal SARSA (black curve) and baseline methods (colored curves) on the blocker task (the black curve is drawn on Figure 4.2 of Heess, Silver, and Teh (2013)). Each curve shows the mean and the standard deviation, over 20 runs, of the average reward per action.

Figure 3[3] shows the average reward per step, for every 40,000 steps (and for every 4,000 steps during the initial 40,000 steps of Determinantal SARSA), against the total number of steps for Determinantal SARSA (black) and baseline methods (colored) studied in Heess, Silver, and Teh (2013). The baselines are free-energy SARSA (SARSA), neural network (NN), energy-based Q-value natural actor critic (EQNAC), and energy-based natural TD actor-critic (ENATDAC). We find that the average reward for Determinantal SARSA already exceeds -0.7 during 20,000-24,000 steps, which is approximately 10 times faster than the baseline methods.

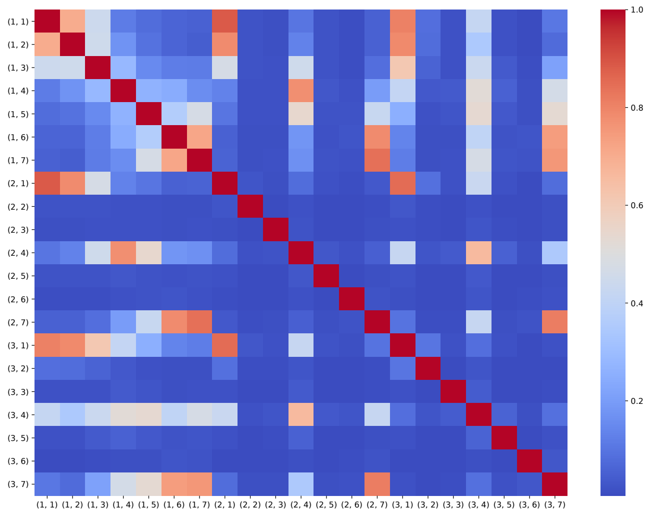
A potential weakness of Determinantal SARSA is relatively large standard deviation in the average reward. The average reward has stayed below -0.9 for one of the 20 runs. This suggests that Determinantal SARSA may be trapped in poor local optima, depending on initial conditions, where the initial values of $\mathbf{V}$ have been chosen by adding noise to each element of the identity matrix, where the noise is uniformly at random between -0.01 and 0.01.

While learning $\mathbf{L}$, Determinantal SARSA learns the qual-

[3]Here, we let the learning rate decrease over time with a "simple back-off strategy" (Dabney and Barto 2012), where the learning rate at step t is $\eta_t = \eta_0 \min\{1, 10^4/(t+1)\}$ with $\eta_0 = 10^{-3}$. The discount factor is set $\rho = 0.9$. In Boltzmann exploration, we let the inverse temperature β_t increase over time t: $\beta_t = (\beta_{10^4})^{t/10^4}$ with $\beta_{10^4} = 10.0$. These hyper-parameters are set as the values that give best performance for the initial 10,000 steps of one run, where the candidate values are $\eta_0 \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, $\rho \in \{0.9, 0.95, 1.0\}$, and $\beta_{10^4} \in \{1.0, 10.0, 100.0\}$.



(a) Learned quality



(b) Learned similarity

Figure 4: The quality of the positions (top) and the absolute value of their similarity (bottom) learned by Determinantal SARSA.

ity of each position and the similarity between those positions. In fact, the learned $\mathbf{L}$ can be decomposed into quality score $\mathbf{q} = \mathrm{diag}(\mathbf{L})$ and similarity measures $\mathbf{S}$, whose elements are $S_{i,j} = L_{i,j}/\sqrt{L_{i,i}\,L_{j,j}}$ (see (85)-(86) from Kulesza and Taskar (2012)).

Figure 4 shows the quality score and the similarity measure learned by Determinantal SARSA. The figure of the quality score corresponds to the top three rows of the grid field in Figure 2. Three positions in the third row ((3, 1), (3, 4), (3, 7)) are found to have high value, as an agent might reach the end zone in the next step from those positions. Learning quality score is not sufficient, because then all of the agents can head toward the same position to be blocked by a defender. The similarity measure shows that the positions that are close to each other (*e.g.*, (2, 1) and (3, 1)) are found to have high similarity. This makes agents tend to move toward the positions that not only have high quality but also are far away from each other, letting the three agents occupy (3, 1), (3, 4), and (3, 7), making it impossible for the defenders to block all of the agents. Note that the agent-team has learned such quality and similarity only from the experiences of taking actions, observing states, and earning reward. In particular, no geographical information about the positions is given.

Stochastic Policy Task

Next, we study three stochastic policy tasks, where an agent-team iteratively chooses team-actions from the space of N dimensional binary vectors, where $N = 5$ in Task 1, $N = 6$ in Task 2, and $N = 8$ in Task 3. States are unobservable but can be represented as binary vectors of N dimensions. When the team-action matches the state, the agent-team receives $+10$ reward, and the state transitions to another. Otherwise, the agent-team receives no reward, and the state stays unchanged. The state transitions in a deterministic and cyclic manner among two states in Task 1 and three states in Task 2-3.

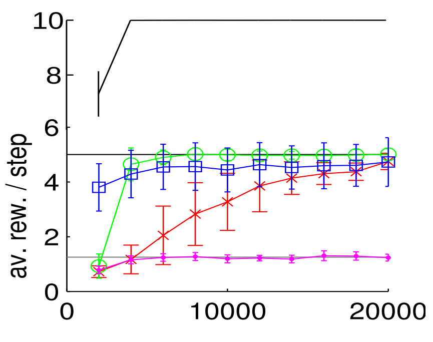
In stochastic policy tasks, it is important to take into account correlation between actions (bits): some pairs of bits should be selected together, and others should not. We use determinant to take into account "diversity", but it is non-trivial what "diversity" really means here. Determinantal SARSA learns a feature vector for each bit in a way that the bits that should not be selected together have similar feature vectors, and those that should be selected together have rather orthogonal feature vectors. We will see that this "diversity" helps in stochastic policy tasks.

We learn to control the agent-team via Determinantal SARSA. Here, the team-action a_t has the natural N-bit feature vector $\mathbf{x}_t$, and we let the observation at time t to be the previous team-action $\mathbf{z}_t = \mathbf{x}_{t-1}$. We then use full rank matrices ($K = M = N$), so that $\mathbf{V}$ and $\mathbf{D}_t$ are $N \times N$. Throughout, we use a lag-1 autoregressive model $\mathbf{d}_t(\phi) = \mathbf{b} + \mathbf{W}\,\mathbf{z}_t$ as our time-series model and set the discount factor as $\rho = 0$, because those settings are the simplest and generally perform well for stochastic policy tasks. In Boltzmann exploration, we run MCMC for 100 steps, starting from a uniformly random team-action (each bit is 1 with probability 0.5).

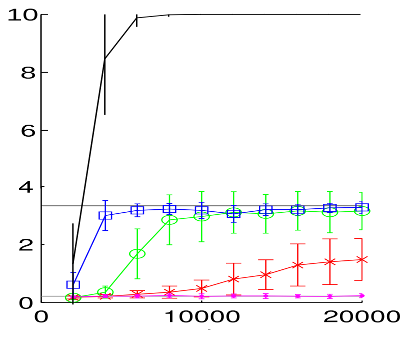
Figure 5[4] shows the average reward per step against the total number of steps. The black curves show the performance of Determinantal SARSA, and the colored curves show the performance of the four baseline methods studied in Heess, Silver, and Teh (2013). The color of each curve indicates the specific method, as is shown in the legend of Figure 3.

Overall, Determinantal SARSA performs substantially better than the baseline methods consistently for all of the three tasks, where the size of the action space is varied from $2^5 = 32$ (Task 1) to $2^8 = 256$ in (Task 3). In particular, Determinantal SARSA finds nearly optimal history-dependent policies within the steps that the best baseline method needs to find nearly optimal Markovian policies. The agent-team

[4]Here, the hyper parameters are set as the values that give best performance for the initial $n = 1,000$ iterations of one run for Task 1-2, but we let $n = 10,000$ for Task 3, which requires longer iterations to observe meaningful difference. Specifically, the learning rate is set as $\eta_0 = 0.005$ for Task 1-2 and $\eta_0 = 0.003$ for Task 3, which are then decreased after n iterations with the simple back-off strategy similar to the blocker task. We use L2 regularization as discussed immediately after Theorem 1, and the strength of the regularization is set as 0.2 for Task 1, 0.02 for Task 2, and 0.1 for Task 3. In Boltzmann exploration, we set the inverse temperature as $\beta = 20.0$ for Task 1 and 3 and $\beta = 32.0$ for Task 2.



(a) Task 1


(b) Task 2

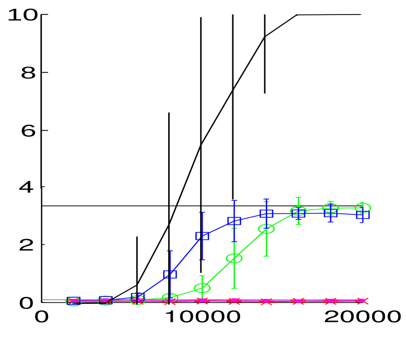

(c) Task 3

Figure 5: Performance of Determinantal SARSA (black curves) and baseline methods (colored curves) on the stochastic policy tasks (black curves are drawn on Figure 4.1 of Heess, Silver, and Teh (2013)). Each curve shows the mean and the standard deviation, over 20 runs, of the average reward per step (team-action) for every 2,000 steps (team-actions). Gray lines show the average reward per step for the random policy (The gray straight line for Task 1 was drawn at wrong values in Heess, Silver, and Teh (2013), which remains here.). Black straight lines show the average reward per step for the optimal Markovian policy or the optimal history-dependent policy.

obtains $+10$ reward every step with the optimal history-dependent policy, while the agent-team can obtain the reward with probability $1/|\mathcal{S}|$ with the optimal Markovian policy, where $|\mathcal{S}|$ is the number of hidden states.

Conclusion

We have proposed the approach of using the determinant of a matrix in reinforcement learning. Determinant plays the role of taking into account diversity in team-actions. When the matrix is diagonal, our approach reduces to the standard approach of factored action space, where multi-agents choose actions independently. Determinantal SARSA can deal with partial observability by using determinant with a time-series model, which can then be trained in an end-to-end manner. In our experiments, Determinantal SARSA substantially outperforms existing methods that have been proposed for dealing with high dimensional action space, which typically appears in multi-agent reinforcement learning.

The proposed 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. Notice that learning a Q function with Determinantal SARSA does not need to deal with a partition function, which is the source of computational bottleneck in the case of learning DPPs. As a result, the most computationally expensive step of learning a Q function with Determinantal SARSA is in computing the pseudo inverse of a $\kappa \times K$ matrix $\mathbf{V}(\mathbf{x})$, where κ is the number of 1 in $\mathbf{x}$ (e.g., number of agents), and K is the rank of the kernel, and these can be much smaller than N.

A DPP (or an annealed determinantal distribution) naturally appears with Determinantal SARSA when we choose team-actions according to the standard approach of Boltzmann exploration. We can thus exploit existing techniques for sampling from DPPs to choose team-actions from a combinatorially large space. However, the kernel $\mathbf{L}_t$ that we learn with Determinantal SARSA may be used to sample actions according to other distributions that induce diversity. This leads to non-standard exploration strategies but possibly with reduced sampling cost. For example, the learned $\mathbf{L}_t$ may be used with Matérn's hard-core processes (Matérn 1986; Haenggi 2012), where the actions are sampled independently according to their relevance and thinned (removed) on the basis of their similarity.

A focus of our study has been on learning a time-varying kernel $\mathbf{L}_t$ with reinforcement learning. Learning $\mathbf{L}_t$ means learning what actions are relevant and what actions are similar to each other, where these relevance and similarity can vary over time. We have demonstrated that one can learn those relevance and similarity only from observing states, taking team-actions, and getting rewards. In our framework, actions are considered to be similar to each other when a team-action consisting of those actions has lower value than what is expected from the individual relevance of those actions.

Finally, SARSA is only one method of reinforcement learning, and the approach of using determinant to approximate action-value functions or policies is largely applicable to other methods of reinforcement learning. Future work includes using determinant with more sophisticated methods of reinforcement learning for practical tasks of multi-agent reinforce learning such as sports games.

Acknowledgments

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

References

Baird, L. C. 1995. Residual algorithms: Reinforcement learning with function approximation. In *Proc. of the 12th International Conference on Machine Learning*, 30–37.

Belabbas, M.-A., and Wolfe, P. J. 2009. On landmark selection and sampling in high-dimensional data analysis. *Philo-*

sophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences 367:4295–4312.

Boyan, J. A., and Moore, A. W. 1995. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*. 369–376.

Dabney, W., and Barto, A. G. 2012. Adaptive step-size for online temporal difference learning. In *Proc. of the 26th AAAI Conference on Artificial Intelligence*, 872–878.

Elfwing, S.; Uchibe, E.; and Doya, K. 2016. From free energy to expected energy: Improving energy-based value function approximation in reinforcement learning. *Neural Networks* 84:17–27.

Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P.; Kohli, P.; and Whiteson, S. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proc. of the 34th International Conference on Machine Learning*, 1146–1155.

Gartrell, M.; Paquet, U.; and Koenigstein, N. 2017. Low-rank factorization of determinantal point processes. In *Proc. of the 31st AAAI Conference on Artificial Intelligence*, 1912–1918.

Gillenwater, J. A.; Kulesza, A.; Fox, E.; and Taskar, B. 2014. Expectation-maximization for learning determinantal point processes. In *Advances in Neural Information Processing Systems 27*. 3149–3157.

Gillenwater, J. 2014. *Approximate Inference for Determinantal Point Processes*. Ph.D. Dissertation.

Gong, B.; Chao, W.-L.; Grauman, K.; and Sha, F. 2014. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems 27*. 2069–2077.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative multi-agent control using deep reinforcement learning. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, 66–83.

Haenggi, M. 2012. *Stochastic Geometry for Wireless Networks*. Cambridge University Press.

Hausknecht, M., and 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*. 29–37.

Heess, N.; Silver, D.; and Teh, Y. W. 2013. Actor-critic reinforcement learning with energy-based policies. In *Proc. of the 10th European Workshop on Reinforcement Learning*, volume 24, 45–58.

Kang, B. 2013. Fast determinantal point process sampling with application to clustering. In *Advances in Neural Information Processing Systems 26*. 2319–2327.

Kathuria, T.; Deshpande, A.; and Kohli, P. 2016. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems 29*. 4206–4214.

Kulesza, A., and Taskar, B. 2011. k-DPPs: Fixed-size determinantal point processes. In *Proc. of the 28th International Conference on Machine Learning*, 1193–1200.

Kulesza, A., and Taskar, B. 2012. *Determinantal Point Processes for Machine Learning*. Hanover, MA, USA: Now Publishers Inc.

Macchi, O. 1975. The coincidence approach to stochastic point processes. *Advances in Applied Probability* 7:83–122.

Maei, H. R., and Sutton, R. S. 2010. GQ (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proc. of the 3rd Conference on Artificial General Intelligence*, 91–96.

Maei, H. R.; Szepesvári, C.; Bhatnagar, S.; Precup, D.; Silver, D.; and Sutton, R. S. 2009. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems 22*. 1204–1212.

Mariet, Z., and Sra, S. 2016. Kronecker determinantal point processes. In *Advances in Neural Information Processing Systems 29*. 2694–2702.

Matérn, B. 1986. *Spatial Variation*. Lecture Notes in Statistics. Springer, 2nd edition.

Osogami, T., and Otsuka, M. 2015. Seven neurons memorizing sequences of alphabetical images via spike-timing dependent plasticity. *Scientific Reports* 5:14149.

Osogami, T.; Raymond, R.; Goel, A.; Shirai, T.; and Maehara, T. 2018. Dynamic determinantal point processes. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*, 3868–3875.

Osogami, T. 2017. Boltzmann machines for time-series. Technical Report RT0980, IBM Research - Tokyo.

Qiao, M.; Xu, R. Y. D.; Bian, W.; and Tao, D. 2016. Fast sampling for time-varying determinantal point process. *ACM Transactions on Knowledge Discovery from Data* 11(Article No. 8).

Sallans, B., and Hinton, G. E. 2001. Using free energies to represent Q-values in a multiagent reinforcement learning task. In *Advances in Neural Information Processing Systems 13*. 1075–1081.

Sallans, B., and Hinton, G. E. 2004. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research* 5:1063–1088.

Sallans, B. 2002. *Reinforcement learning for factored Markov decision processes*. Ph.D. Dissertation.

Snoek, J.; Zemel, R.; and Adams, R. P. 2013. A determinantal point process latent variable model for inhibition in neural spiking data. In *Advances in Neural Information Processing Systems 26*. 1932–1940.

Wachinger, C., and Golland, P. 2015. Sampling from determinantal point processes for scalable manifold learning. In *Proc. of the International Conference on Information Processing in Medical Imaging*, 687–698.

Zhang, C.; Kjellström, H.; and Mandt, S. 2017. Balanced mini-batch sampling for SGD using determinantal point processes. In *Proc. of the 33rd Conference on Uncertainty in Artificial Intelligence*.