Minorization-Maximization for Learning Determinantal Point Processes

Anonymous authors
Paper under double-blind review

Abstract

A determinantal point process (DPP) is a powerful probabilistic model that generates diverse random subsets from a ground set. Since a DPP is characterized by a positive definite kernel, a DPP on a finite ground set can be parameterized by a kernel matrix. Recently, DPPs have gained attention in the machine learning community and have been applied to various practical problems; however, there is still room for further research on the learning of DPPs. In this paper, we propose a simple learning rule for full-rank DPPs based on a minorization-maximization (MM) algorithm, which monotonically increases the likelihood in each iteration. We show that our minorizer of the MM algorithm provides a tighter lower-bound compared to an existing method locally. In our experiments on both synthetic and real-world datasets, our method outperforms existing methods in most settings.

1 Introduction

A determinantal point process (DPP) is a probabilistic model that represents the occurrence probability of random subsets of a ground set. Initially, DPPs were originated in statistical mechanics to describe the probabilistic behavior of fermions (Macchi, 1975). In recent years, broader applications of DPPs have been developed in the machine learning community (Kulesza & Taskar, 2012).

An important feature of DPPs is the presence of negative dependence. Suppose A and B are subsets of a ground set, $P(A \cup B) < P(A)P(B)$ is possible when $P(\cdot)$ is defined as a DPP. This means that DPPs can take into account inter-element repulsion, which encourages the occurrence of diverse subsets. This feature aligns with a variety of machine learning applications, such as diversity-promoting image search (Kulesza & Taskar, 2011), recommender systems (Gillenwater et al., 2014), base station configuration for cellular networks (Miyoshi & Shirai, 2014), and random design regression (Dereziński et al., 2022).

A natural problem on DPPs is efficient learning of the parameters. Since a DPP defined on a finite ground set is parameterized by a positive semidefinite kernel matrix, the learning methods are roughly classified into three approaches: (a) assuming the kernel matrix is full-rank and having no additional structure (full-rank DPPs), (b) assuming the kernel matrix is low-rank (low-rank DPPs), or (c) assuming other tractable structure for the kernel matrix.

So far, some learning methods have been designed for full-rank DPPs. Gillenwater et al. (2014) pioneered the learning problem of DPPs; they developed an EM algorithm for full-rank DPPs. Mariet & Sra (2015) later proposed a fixed-point algorithm for full-rank DPPs. They derived a simple update rule for the kernel matrix and showed its monotonicity by finding its equivalence with a minorization-maximization (MM) algorithm. Their experiments also showed that the fixed-point algorithm is more efficient and stable than the EM algorithm.

Gartrell et al. (2017) introduced low-rank DPPs. Learning of low-rank DPPs involves gradient-based optimization. Mariet et al. (2019) proposed contrastive estimation as an alternative of the maximum likelihood estimation (MLE), while Osogami et al. (2018) incorporated temporal dynamics into low-rank DPPs. A Bayesian extension of low-rank DPPs was also proposed in (Gartrell et al., 2016).

In principle, without special structures, it is difficult to overcome the $\mathcal{O}(N^3)$ time complexity for full-rank DPPs and $\mathcal{O}(NK^2)$ for low-rank DPPs, where N is the size of the ground set and K is the rank of the kernel matrix. To go beyond these complexities, DPPs with special structure are developed, such as Kronecker DPPs (Mariet & Sra, 2016) and the "diagonal+special low-rank" structure (Dupuy & Bach, 2018).

Our study focuses on learning of full-rank DPPs. While full-rank DPPs are sometimes not suitable for problems with a large ground set, we often want to conduct an exact inference for small to medium-sized problems. For example, consider a virtual application of a DPP. The first step in the data analysis is to assess whether a DPP is appropriate for our task or not. Even if our final goal is to handle large data, we typically take a relatively small data collected provisionally during this assessment phase. In such a situation, we hope to utilize a ready-made learning algorithm that is hyperparameter-free, easily implementable, and well-behaved. However, the existing methods for full-rank DPPs have some difficulties; the EM algorithm (Gillenwater et al., 2014) internally requires optimization on a Stiefel manifold, making the learning procedure complicated and unstable. In (Mariet & Sra, 2015), the authors introduced a step size in order to accelerate the fixed-point algorithm, but this induces a violation of the condition that guarantees a monotone convergence property of the algorithm.

In this paper, we propose a simple yet powerful learning rule for full-rank DPPs based on the MM algorithm. Our method increases the log-likelihood monotonically and stably, and locally provides a tighter minorizer than the fixed-point algorithm. Furthermore, our minorizer is concave while the fixed-point algorithm maximizes a non-concave minorizer in the iteration. We also conduct experiments with both synthetic and real-world datasets and our method outperforms the existing methods in most settings.

In summary, our main contributions in this paper are:

- We present an easy-to-implement learning method for full-rank DPPs based on the MM algorithm. By the property of MM algorithms, our method monotonically increases the log-likelihood.
- We compare the tightness of the minorizers between the existing and proposed methods. The fixed-point algorithm for DPPs proposed in (Mariet & Sra, 2015) can also be viewed as an MM algorithm. Our result indicate that our minorizer locally provides a tighter lower-bound than the existing method. Moreover, our method provides a concave minorizer unlike the exsiting method.
- We conduct experiments to evaluate learning algorithms for full-rank DPPs using both synthetic and real-world datasets. Our empirical results show superiority of our method in convergence speed and stability.

2 Determinantal Point Processes

Let $k(\cdot,\cdot)$ be a kernel function on a ground set Ω. A determinantal point process (DPP) with the kernel function $k(\cdot,\cdot)$ is a point process on Ω whose joint intensities are formed as

$$\rho_n(x_1, x_2, \ldots, x_n) = \det(\boldsymbol{K}_{[n]}),$$

where $x_1, x_2, \ldots, x_n \in \Omega$ and $\boldsymbol{K}_{[n]} = (k(x_i, x_j))_{i,j=1}^n$ (Hough et al., 2009). The following theorem gives a sufficient condition for the existence and uniqueness of DPP:

Theorem 2.1 (Soshnikov (2000); Shirai & Takahashi (2000))**.** *Let $\mathcal{K}$ be a self-adjoint integral operator determined by a kernel function k and be of locally trace class. Then, the kernel function $k(\cdot,\cdot)$ determines a DPP if and only if all the eigenvalues of $\mathcal{K}$ are in* $[0, 1]$.

If the restriction of an operator $\mathcal{K}$ to an arbitrary compact subset of Ω is of trace class, $\mathcal{K}$ is said to be locally trace class. Roughly speaking, Theorem 2.1 states that a positive definite kernel $k(\cdot,\cdot)$ defines a DPP under appropriate scaling.

In the context of machine learning, DPPs on a finite ground set $\mathcal{Y} = \{1, 2, \ldots, N\}$ are typically considered. On the finite ground set $\mathcal{Y}$, a point process $P(\cdot)$ is a DPP with a kernel matrix $\boldsymbol{K} \in \mathbb{S}_+^N$ if

$$P(\mathcal{S} \subset \mathcal{A}) = \det([\boldsymbol{K}]_{\mathcal{S}})$$

for a random subset $\mathcal{A} \subset \mathcal{Y}$ drawn by P and an arbitrary $\mathcal{S} \subset \mathcal{Y}$. $[\boldsymbol{K}]_{\mathcal{S}} = (K_{ij})_{i,j\in\mathcal{S}} \in \mathbb{S}_+^{|\mathcal{S}|}$ denotes the principal submatrix of $\boldsymbol{K}$ and the kernel matrix $\boldsymbol{K}$ must be $\boldsymbol{O} \preceq \boldsymbol{K} \preceq \boldsymbol{I}$ from an analogy with the DPPs on a general ground set. A DPP on a finite ground set have an alternative representation called the $\boldsymbol{L}$-ensemble (Borodin & Rains, 2005), which defines the occurrence probability of a random subset $\mathcal{A} \subset \mathcal{Y}$ as

$$P_{\boldsymbol{L}}(\mathcal{A}) = \frac{\det([\boldsymbol{L}]_{\mathcal{A}})}{\det(\boldsymbol{L}+\boldsymbol{I})},$$

where $\boldsymbol{L} \in \mathbb{S}_+^N$ is a positive semidefinite kernel matrix. We can commute between $\boldsymbol{K}$ and $\boldsymbol{L}$ using the equation $\boldsymbol{K} = \boldsymbol{L}(\boldsymbol{L}+\boldsymbol{I})^{-1}$ or its inversion $\boldsymbol{L} = \boldsymbol{K}(\boldsymbol{I}-\boldsymbol{K})^{-1}$. In this paper, we develop a learning algorithm for $\boldsymbol{L}$.

3 Learning Algorithm

Given M samples $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M \subset \mathcal{Y}$, our goal is to solve MLE, that is, to find a maximizer of the log-likelihood

$$\begin{aligned} f(\boldsymbol{L}) &= \frac{1}{M}\sum_{m=1}^{M} \log\det([\boldsymbol{L}]_{\mathcal{A}_m}) - \log\det(\boldsymbol{L}+\boldsymbol{I}) \\ &= \frac{1}{M}\sum_{m=1}^{M} \log\det(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}\boldsymbol{U}_{\mathcal{A}_m}^\top) - \log\det(\boldsymbol{L}+\boldsymbol{I}), \end{aligned} \tag{1}$$

where $\boldsymbol{U}_{\mathcal{A}_m} \in \{0,1\}^{|\mathcal{A}_m|\times N}$ is a submatrix of $\boldsymbol{I}$ obtained by keeping the rows corresponding to the elements in $\mathcal{A}_m$.

3.1 MM Algorithm

A minorization-maximization (MM) algorithm is a powerful meta-algorithm for finding a local maximizer of a generally non-concave objective $f(\theta)$ (Hunter & Lange, 2004; Sun et al., 2017). The MM algorithm consists of two steps: (i) find a minorizer $g(\theta|\theta^{(t)})$ of $f(\theta)$ that satisfies

- $f(\theta) \geq g(\theta|\theta^{(t)})$
- $f(\theta^{(t)}) = g(\theta^{(t)}|\theta^{(t)})$

for all θ and $\theta^{(t)}$ within a feasible region. Then, (ii) maximize the minorizer $g(\theta|\theta^{(t)})$ with respect to θ and set $\theta^{(t+1)} = \arg\max g(\theta|\theta^{(t)})$. Repeating this process, we can obtain a sequence of the parameters $\{\theta^{(t)}\}_{t\geq 0}$ which monotonically increases the objective value, because

$$f(\theta^{(t+1)}) \geq g(\theta^{(t+1)}|\theta^{(t)}) \geq g(\theta^{(t)}|\theta^{(t)}) = f(\theta^{(t)}) \tag{2}$$

holds.

Since $\log\det(\cdot)$ is concave on $\mathbb{S}_{++}$, the objective function (1) is a combination of concave and convex functions. By lower-bounding the convex part $-\log\det(\boldsymbol{L}+\boldsymbol{I})$ with the first-order Taylor expansion around $\boldsymbol{L}^{(t)}+\boldsymbol{I}$, we have

$$-\log\det(\boldsymbol{L}+\boldsymbol{I}) \geq -\log\det(\boldsymbol{L}^{(t)}+\boldsymbol{I}) - \operatorname{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}(\boldsymbol{L}-\boldsymbol{L}^{(t)})\}, \tag{3}$$

which yields a choice for minorizing the objective (1). This method is referred to as the concave-convex procedure (CCCP) (Yuille & Rangarajan, 2001), a special case of MM algorithms. However, the minorizer derived by the CCCP has no closed-form maximizer in our case, therefore, we devise an easy-to-optimize alternative.

3.2 Proposed Algorithm

In the proposed minorizer of (1), the convex part is lower-bounded linearly by (3) and the concave part $\log\det([\boldsymbol{L}]_{\mathcal{A}_m}) = \log\det(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}\boldsymbol{U}_{\mathcal{A}_m}^\top)$ is also lower-bounded. The following proposition provides the concrete form of our proposed minorizer.

Proposition 3.1. *Let* $f(\boldsymbol{L})$ *be given by* (1) *and*

$$g(\boldsymbol{L}|\boldsymbol{L}^{(t)}) = -\frac{1}{M}\sum_{m=1}^{M} \mathrm{tr}\{\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top[\boldsymbol{L}^{(t)}]_{\mathcal{A}_m}^{-1}\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{L}^{-1}\} - \mathrm{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}\boldsymbol{L}\} + \zeta(\boldsymbol{L}^{(t)}), \tag{4}$$

where

$$\zeta(\boldsymbol{L}^{(t)}) = \frac{1}{M}\sum_{m=1}^{M}\Big\{\log\det(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top) + |\mathcal{A}_m|\Big\} - \log\det(\boldsymbol{L}^{(t)}+\boldsymbol{I}) + \mathrm{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}\boldsymbol{L}^{(t)}\}$$

is a constant term. Then, $f(\boldsymbol{L}) \geq g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ *and* $f(\boldsymbol{L}^{(t)}) = g(\boldsymbol{L}^{(t)}|\boldsymbol{L}^{(t)})$ *hold for any* $\boldsymbol{L}, \boldsymbol{L}^{(t)} \in \mathbb{S}_{++}^N$.

Proof. For $\boldsymbol{P} \succ 0$, the following matrix inequality holds (Sun et al., 2016; 2017):

$$\begin{aligned}(\boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^\top)^{-1} &\preceq \boldsymbol{R}_t^{-1}\boldsymbol{A}\boldsymbol{P}_t\boldsymbol{P}^{-1}\boldsymbol{P}_t\boldsymbol{A}^\top\boldsymbol{R}_t^{-1}, \\ \boldsymbol{R}_t &= \boldsymbol{A}\boldsymbol{P}_t\boldsymbol{A}^\top.\end{aligned} \tag{5}$$

Using the first-order Taylor expansion and (5), we have

$$\begin{aligned}\log\det(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}\boldsymbol{U}_{\mathcal{A}_m}^\top) &\geq |\mathcal{A}_m| + \log\det(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top) - \mathrm{tr}\{(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}\boldsymbol{U}_{\mathcal{A}_m}^\top)^{-1}\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top\} \\ &\geq |\mathcal{A}_m| + \log\det(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top) - \mathrm{tr}\{\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top[\boldsymbol{L}^{(t)}]_{\mathcal{A}_m}^{-1}\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{L}^{-1}\}.\end{aligned} \tag{6}$$

Combining the lower-bounds (3) and (6), we can construct the minorizer of $f(\boldsymbol{L})$ as (4). □

In order to obtain the maximizer of (1), we iteratively optimize the proposed minorizer $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ by solving the first-order optimality condition for $t = 1, \ldots, T$. Since $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ is concave because of the convexity of $\mathrm{tr}(\boldsymbol{X}^{-1})$ for $\boldsymbol{X} \succ 0$, a stationary point of $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ is also its global maximizer.

Proposition 3.2. *A global maximizer of* $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ *satisfies*

$$-\boldsymbol{L}(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}\boldsymbol{L} + \boldsymbol{Q}_M^{(t)} = \boldsymbol{O}, \tag{7}$$

where

$$\boldsymbol{Q}_M^{(t)} = \boldsymbol{L}^{(t)}\left(\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{U}_{\mathcal{A}_m}^\top[\boldsymbol{L}^{(t)}]_{\mathcal{A}_m}^{-1}\boldsymbol{U}_{\mathcal{A}_m}\right)\boldsymbol{L}^{(t)}.$$

Proof. Taking a derivative of the minorizer $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$, we have

$$\nabla_{\boldsymbol{L}} g(\boldsymbol{L}|\boldsymbol{L}^{(t)}) = \boldsymbol{L}^{-1}\boldsymbol{Q}_M^{(t)}\boldsymbol{L}^{-1} - (\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1} = \boldsymbol{O}, \tag{8}$$

for the optimality condition. By multiplying the both sides of (8) by $\boldsymbol{L}$, we can see that the stationary points of $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ satisfy (7). From the concavity of $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$, we obtain the result. □

The matrix quadratic equation (7) is a special case of the continuous algebraic Riccati equation (CARE):

$$\boldsymbol{A}^\top\boldsymbol{X} + \boldsymbol{X}\boldsymbol{A} - \boldsymbol{X}\boldsymbol{G}\boldsymbol{X} + \boldsymbol{Q} = \boldsymbol{O}, \tag{9}$$

where $\boldsymbol{X} \in \mathbb{S}^N$ is unknown, and $\boldsymbol{G}, \boldsymbol{Q} \in \mathbb{S}^N, \boldsymbol{A} \in \mathbb{R}^{N\times N}$ are fixed coefficient matrices. The CARE is well-studied in control engineering and is solvable by some numerical methods such as the Schur method

Algorithm 1: Minorization-Maximization (MM)

```
Data: Training set $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M\}$
Result: $\boldsymbol{L}$
Initialize $\boldsymbol{L}$, set $\varepsilon \geq 0$;
for $t = 1$ to $T$ do
    $\boldsymbol{A} \leftarrow \boldsymbol{O}$;
    $\boldsymbol{Q}_\varepsilon \leftarrow \boldsymbol{L}\left(\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{U}_{\mathcal{A}_m}^\top [\boldsymbol{L}]_{\mathcal{A}_m}^{-1} \boldsymbol{U}_{\mathcal{A}_m}\right)\boldsymbol{L} + \varepsilon\boldsymbol{I}$;
    $\boldsymbol{G} \leftarrow (\boldsymbol{L} + \boldsymbol{I})^{-1}$;
    $\boldsymbol{L} \leftarrow \mathrm{SolveCARE}(\boldsymbol{A}, \boldsymbol{Q}_\varepsilon, \boldsymbol{G})$ // Solve Equation (9)
end
```

(Laub, 1979) and Newton's method (Bini et al., 2011; Benner & Byers, 1998). It is worth noting that CARE solvers are available in most programming languages through packages for scientific computation; for example, `SciPy` in Python and `MatrixEquations.jl` in Julia.

In addition, we can confirm the following statement as a corollary of Proposition 3.2.

Corollary 3.1. *With the same notation as in Proposition 3.2 and a positive definite initial value* $\boldsymbol{L}^{(0)} \succ 0$*, we have* $\mathrm{rank}(\boldsymbol{L}^{(t)}) = \mathrm{rank}(\boldsymbol{Q}_M^{(t)})$ *for* $t = 1, 2, \ldots$.

Proof. Since $\boldsymbol{L}^{(0)}$ is positive definite, $(\boldsymbol{L}^{(0)} + \boldsymbol{I})^{-1}$ is non-singular. Therefore, from the optimality condition (7),

$$\mathrm{rank}(\boldsymbol{L}^{(1)}(\boldsymbol{L}^{(0)} + \boldsymbol{I})^{-1}\boldsymbol{L}^{(1)}) = \mathrm{rank}(\boldsymbol{L}^{(1)}) = \mathrm{rank}(\boldsymbol{Q}_M^{(1)}).$$

By applying similar operations recursively, the result can be confirmed. □

Corollary 3.1 says that if $\boldsymbol{Q}_M^{(t)}$ is degenerated, the solution of (7) must also be degenerated. This means that when $\boldsymbol{Q}_M^{(t)}$ is singular, the solution of (7) falls outside the feasible region $\mathbb{S}_{++}^N$. This problem arises when some elements of $\mathcal{Y}$ are never observed in the given data $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M$. To avoid this issue and stabilize numerical computation, we recommend to solve

$$-\boldsymbol{L}(\boldsymbol{L}^{(t)} + \boldsymbol{I})^{-1}\boldsymbol{L} + \boldsymbol{Q}_M^{(t)} + \varepsilon\boldsymbol{I} = \boldsymbol{O}$$

with a machine epsilon $\varepsilon > 0$ instead of (7). We note that the choice of the machine epsilon ε does not affect the estimate significantly; we use $\varepsilon = 10^{-10}$ throughout this paper. The procedure for the proposed MM-based learning is summarized in Algorithm 1.

3.3 Relation to the Existing Method

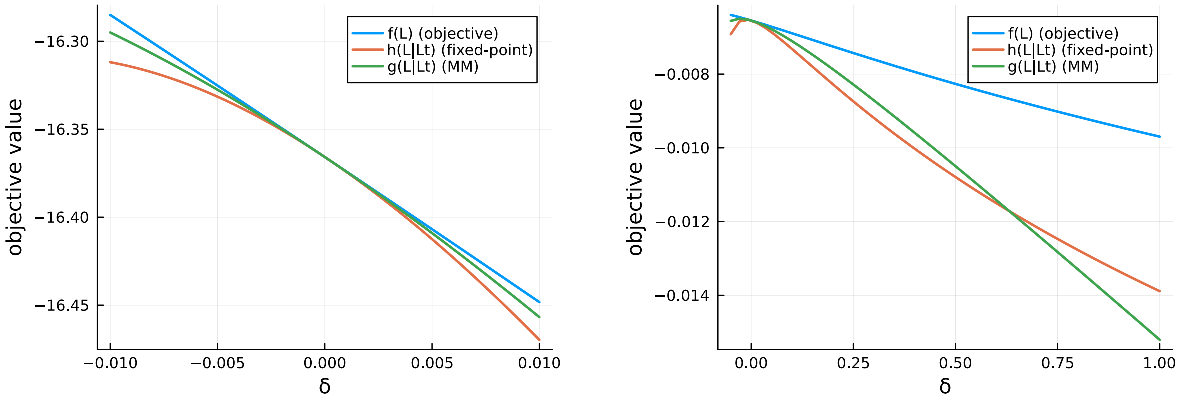
Mariet & Sra (2015) derived the following update rule to maximize (1) as a fixed-point algorithm:

$$\boldsymbol{L}^{(t+1)} = \boldsymbol{L}^{(t)} + a\boldsymbol{L}^{(t)}\nabla f(\boldsymbol{L}^{(t)})\boldsymbol{L}^{(t)}, \tag{10}$$

$$\nabla f(\boldsymbol{L}) = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{U}_{\mathcal{A}_m}^\top [\boldsymbol{L}]_{\mathcal{A}_m}^{-1} \boldsymbol{U}_{\mathcal{A}_m} - (\boldsymbol{L} + \boldsymbol{I})^{-1},$$

where $a > 0$ is a step size. For $a = 1$, they also show that the update rule (10) can also be regarded as an MM algorithm with the non-concave minorizer

$$\begin{aligned} h(\boldsymbol{L}|\boldsymbol{L}^{(t)}) = &-\frac{1}{M}\sum_{m=1}^{M} \mathrm{tr}\{\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top [\boldsymbol{L}^{(t)}]_{\mathcal{A}_m}^{-1} \boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{L}^{-1}\} \\ &- \log\det(\boldsymbol{L}) - \mathrm{tr}\{(\boldsymbol{L}^{(t)} + \boldsymbol{I})^{-1}\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)}\} + \xi(\boldsymbol{L}^{(t)}), \end{aligned} \tag{11}$$



(a) Neighborhood of $\boldsymbol{L}^{(t)}$.

(b) Non-neighborhood of $\boldsymbol{L}^{(t)}$.

Figure 1: Behavior of minorizers.

where $\xi(\boldsymbol{L}^{(t)})$ is a constant term and explicitly given in the appendix. Comparing (11) with (4), we can see that the lower-bounds for the first term in (1) are the same, and those for the second term only differ. With respect to these minorizers, the following proposition holds.

Proposition 3.3. *For $g(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ defined in (4) and $h(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ defined in (11), it holds that $g(\boldsymbol{L}|\boldsymbol{L}^{(t)}) \geq h(\boldsymbol{L}|\boldsymbol{L}^{(t)})$ for $\boldsymbol{L}$ in the neighborhood of $\boldsymbol{L}^{(t)}$.*

Proof. We have the following inequality:

$$g(\boldsymbol{L}|\boldsymbol{L}^{(t)}) - h(\boldsymbol{L}|\boldsymbol{L}^{(t)}) \geq \operatorname{tr}\{(\boldsymbol{L}^{(t)} + \boldsymbol{I})^{-1}(2\boldsymbol{L}^{(t)} - \boldsymbol{L} - \boldsymbol{L}^{(t)}\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)})\}, \tag{12}$$

where the derivation is shown in the appendix. If $\boldsymbol{L}$ is in the neighborhood of $\boldsymbol{L}^{(t)}$, meaning that $\boldsymbol{L}$ can be expressed as $\boldsymbol{L} = \boldsymbol{L}^{(t)} + \delta\boldsymbol{M}$ with a sufficiently small $\delta > 0$ and a symmetric matrix $\boldsymbol{M}$ whose eigenvalues are all in $[-1, 1]$, we have

$$2\boldsymbol{L}^{(t)} - \boldsymbol{L} - \boldsymbol{L}^{(t)}\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)} \approx \boldsymbol{O}. \tag{13}$$

The details of the derivation can be found in the appendix. Applying the approximation (13) to (12), we can conclude the proposition. ☐

The proposition 3.3 states that the proposed minorizer gives a tighter lower-bound of the objective than that of the existing method locally. This leads to a tighter leftmost inequality in (2), making it likely that the proposed method will produce better $\boldsymbol{L}^{(t+1)}$. Figure 1 shows the behavior of the minorizers in the neighborhood and non-neighborhood of $\boldsymbol{L}^{(t)}$. The proposed minorizer becomes looser as $\boldsymbol{L}$ moves farther away from $\boldsymbol{L}^{(t)}$, but the experimental results in Section 4 show that the proposed method converges faster in most cases. Note that the minorizer of the fixed-point algorithm is non-convex as seen in Figure 1(b). This implies that the fixed-point algorithm is possible to get trapped in poor stationary points of $h(\boldsymbol{L}|\boldsymbol{L}^{(t)})$.

3.4 Computational Costs

The total computational cost of our method is $\mathcal{O}(M\kappa^3 + N^3)$, where $\kappa = \max_m |\mathcal{A}_m|$. It is computed as follows; the computation of $\boldsymbol{Q}_M^{(t)}$ in (7) requires $\mathcal{O}(\sum_{m=1}^{M} |\mathcal{A}_m|^3 + N^3) = \mathcal{O}(M\kappa^3 + N^3)$ operations, including the evaluation of $[\boldsymbol{L}^{(t)}]_{\mathcal{A}_m}^{-1}$ for all $m = 1, 2, \ldots, M$ and the matrix multiplications of $N \times N$ matrices. The inversion $(\boldsymbol{L}^{(t)} + \boldsymbol{I})^{-1}$ and solving the CARE also cost $\mathcal{O}(N^3)$.

The computational complexity of our method is equal to that of the fixed-point algorithm (Mariet & Sra, 2015). Although our method incurs additional $\mathcal{O}(N^3)$ computations due to the CARE, the experimental results in Section 4 show faster convergence of our method in computational time. We note that the gradient-based learning of a low-rank factorized DPP also takes the same $\mathcal{O}(M\kappa^3 + N^3)$ per iteration if the factorization is full-rank (Gartrell et al., 2017; Osogami et al., 2018).

4 experiments

4.1 Experimental Settings

We evaluate performance of the learning methods for full-rank DPPs through experiments on synthetic and real-world datasets. We take the fixed-point algorithm (FP) (Mariet & Sra, 2015) and Adam (Kingma & Ba, 2015) as references. For Adam, we factorize the kernel matrix as $\boldsymbol{L} = \boldsymbol{V}\boldsymbol{V}^\top$ by $\boldsymbol{V} \in \mathbb{R}^{N\times N}$ and optimize $\boldsymbol{V}$ as with the low-rank DPPs (Gartrell et al., 2017; Osogami et al., 2018).

Mariet & Sra (2015) introduced a hyperparameter a that controls the learning speed of the fixed-point algorithm, but the convergence and monotonicity of the algorithm are guaranteed only when $a = 1$; therefore, we use $a = 1$. In Adam optimization, we employ the default values $\beta_1 = 0.999, \beta_2 = 0.9$ for the decay rates, $\eta = 10^{-9}$ for the learning rate, and $\epsilon = 10^{-8}$ for the machine epsilon[1].

We provide the following two initialization schemes with reference to (Mariet & Sra, 2015):

- `WISHART`: We sample an initial value from the Wishart distribution as $\boldsymbol{L}^{(0)} \sim \mathcal{W}(N, \boldsymbol{I})/N$.
- `BASIC`: We sample $v_{ij}^{(0)} \sim \mathcal{U}(0, \sqrt{2}/N)$ for $i, j = 1, 2, \ldots, N$ and initialize as $\boldsymbol{L}^{(0)} = \boldsymbol{V}^{(0)}\boldsymbol{V}^{(0)\top}$.

The `WISHART` initialization provides a near-identity matrix, while `BASIC` provides a unstructured matrix for $\boldsymbol{L}^{(0)}$.

In each experiment, we stop the learning when the criterion $\frac{|f(\boldsymbol{L}^{(t)}) - f(\boldsymbol{L}^{(t-1)})|}{|f(\boldsymbol{L}^{(t-1)})|} \leq \delta_{\text{tol}}$ is satisfied. We set $\delta_{\text{tol}} = 10^{-4}$ as the relative tolerance for all the experiments reported below. We implemented all the experiments in Julia, and all our experiments were run on a Linux Mint system with 32GB of RAM and an Intel Core i9-10900K CPU @ 3.70GHz.

4.2 Datasets

We compare the learning algorithms with the following three datasets.

Synthetic

We make true parameters as $\boldsymbol{L}^* = \boldsymbol{V}^*\boldsymbol{V}^{*\top}$ with $v_{ij}^* \sim \mathcal{U}(0, 10/N)$ for $i, j = 1, 2, \ldots, N$, and sample M realizations from the DPP $P_{\boldsymbol{L}^*}(\cdot)$. We consider three different problem sizes: $(N, M) = (32, 2{,}500)$, $(N, M) = (32, 10{,}000)$, and $(N, M) = (128, 2{,}500)$. Because the true parameters are constructed from the uniform distribution, they are likely to have no clear structure. Using this `Synthetic` dataset, we test the general applicability of our method.

In `Synthetic`, true parameters $\boldsymbol{L}^*$ are available; we assess goodness of estimation using not only log-likelihoods but also the von Neumann divergences $D_{\text{vN}}(\boldsymbol{L}, \boldsymbol{L}^*) = \operatorname{tr}(\boldsymbol{L}\log\boldsymbol{L} - \boldsymbol{L}\log\boldsymbol{L}^* - \boldsymbol{L} + \boldsymbol{L}^*)$, which is a kind of Bregman divergences for positive definite matrices.

Nottingham

We apply our method to the `Nottingham` dataset[2], which was used in (Boulanger-Lewandowski et al., 2012; Osogami et al., 2018). The dataset contains more than 1,000 folk tracks in the ABC format. We randomly pick 25 tracks and that yields $M = 6{,}364$ samples on average. The number of items, or keys, is $N = 88$.

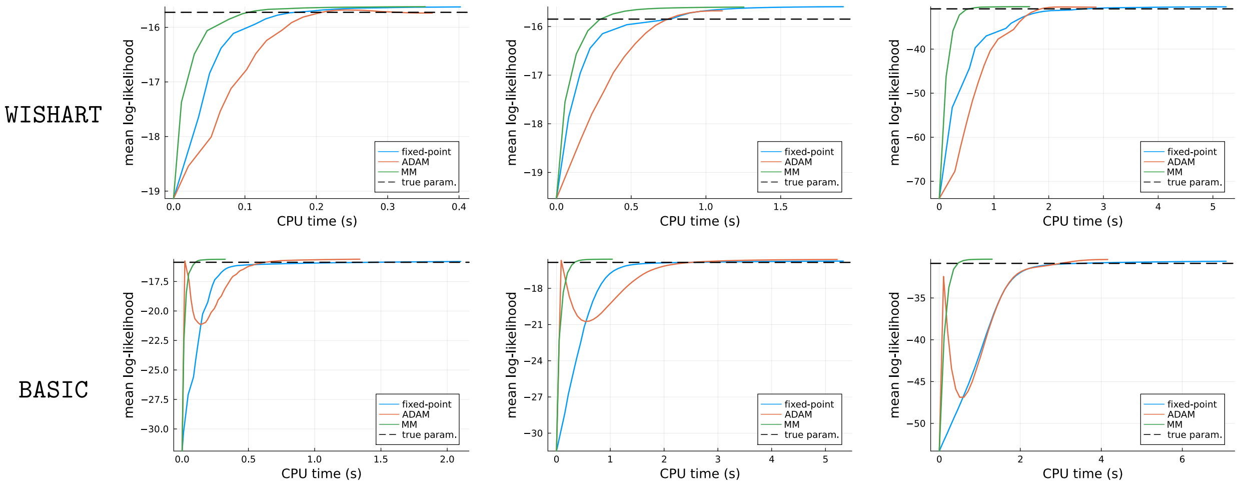
---

[1] Larger learning rates, such as $\eta \geq 10^{-7}$, failed to optimize in our experiments.

[2] Available at `https://abc.sourceforge.net/NMD/`.

Table 1: Final mean log-likelihoods, runtimes, and von Neumann divergences $D_{\mathrm{vN}}(\boldsymbol{L}, \boldsymbol{L}^*)$ of the `Synthetic` datasets. Each value is computed from the average or standard deviation of 30 trials.

		WISHART			BASIC		
Data Size	Method	Log-likelihood	Runtime (s)	vN div.	Log-likelihood	Runtime (s)	vN div.
$N = 32$ $M = 2{,}500$	FP	-15.58 ± 0.15	0.44 ± 0.04	$\mathbf{38.20} \pm 1.84$	-15.61 ± 0.21	1.72 ± 0.27	42.19 ± 4.19
	Adam	-15.66 ± 0.16	$\mathbf{0.37} \pm 0.15$	51.81 ± 4.12	$\mathbf{-15.46} \pm 0.20$	1.42 ± 0.11	55.27 ± 4.81
	MM	$\mathbf{-15.58} \pm 0.15$	0.43 ± 0.06	43.94 ± 2.62	-15.46 ± 0.20	$\mathbf{0.33} \pm 0.03$	$\mathbf{30.15} \pm 1.99$
$N = 32$ $M = 10{,}000$	FP	$\mathbf{-15.58} \pm 0.18$	1.57 ± 0.10	$\mathbf{38.03} \pm 2.00$	-15.72 ± 0.14	6.01 ± 0.65	41.61 ± 3.30
	Adam	-15.68 ± 0.17	1.54 ± 0.53	51.52 ± 3.68	-15.58 ± 0.14	5.48 ± 0.49	54.91 ± 4.40
	MM	-15.59 ± 0.18	$\mathbf{1.32} \pm 0.12$	43.95 ± 2.58	$\mathbf{-15.56} \pm 0.14$	$\mathbf{1.06} \pm 0.06$	$\mathbf{29.74} \pm 1.47$
$N = 128$ $M = 2{,}500$	FP	-30.14 ± 0.18	3.90 ± 0.39	$\mathbf{36.20} \pm 0.44$	-30.35 ± 0.19	6.78 ± 0.40	53.47 ± 1.80
	Adam	-30.18 ± 0.19	2.28 ± 0.48	51.18 ± 3.61	-30.12 ± 0.20	4.60 ± 1.35	60.33 ± 2.71
	MM	$\mathbf{-30.11} \pm 0.18$	$\mathbf{1.45} \pm 0.09$	44.68 ± 0.62	$\mathbf{-30.10} \pm 0.19$	$\mathbf{1.31} \pm 0.06$	$\mathbf{32.26} \pm 0.50$



(a) $N = 32, M = 2{,}500$ (b) $N = 32, M = 10{,}000$ (c) $N = 128, M = 2{,}500$

Figure 2: Learning curves of the `Synthetic` datasets.

In `Nottingham`, there is large disparity in the probability of each item appearing, with very low- and high-pitched keys being rarely used. Moreover, music theory prohibits certain key combinations within a chord. From these facts, the optimal $\boldsymbol{L}^*$ of the `Nottingham` dataset is expected to have unknown but particular structure.

Amazon Baby Registry

`Amazon baby registry` has served as a benchmark for learning methods of DPPs since (Gillenwater et al., 2014). It contains 13 categories of child care products, including "feeding" and "carseats," and on average, has $N = 71$ items and $M = 8{,}585$ samples, respectively. We run our experiment on each of the 13 categories to assess performance of the learning methods for medium-sized recommender systems.

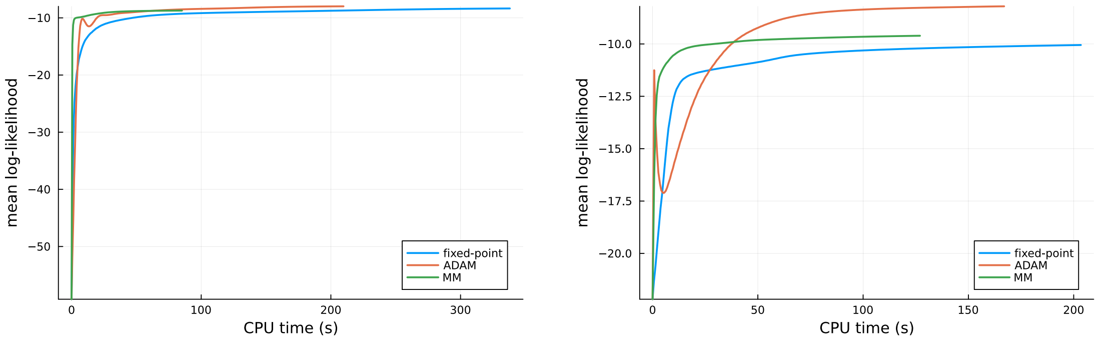
4.3 Experimental Results

Synthetic

The final mean log-likelihoods and runtimes of the `Synthetic` datasets are presented in Table 1. For each experiment, we conducted 30 trials with different $\boldsymbol{L}^*$ and $\boldsymbol{L}^{(0)}$ and calculated the average and standard deviation. As shown in Table 1, our method (MM) achieves the best runtimes for almost settings and

Table 2: Final mean log-likelihoods and runtimes of the `Nottigham` dataset. Each value is computed from the average or standard deviation of 30 trials.

	`WISHART`		`BASIC`	
Method	Log-likelihood	Runtime (s)	Log-likelihood	Runtime (s)
FP	-8.30 ± 0.22	46.58 ± 3.66	-10.14 ± 0.28	37.26 ± 7.47
Adam	$\mathbf{-7.98} \pm 0.76$	29.36 ± 8.28	$\mathbf{-8.89} \pm 2.96$	29.00 ± 11.91
MM	-9.51 ± 0.25	$\mathbf{24.27} \pm 6.66$	-9.59 ± 0.22	$\mathbf{21.56} \pm 4.24$



(a) `WISHART`

(b) `BASIC`

Figure 3: Learning curves of the `Nottingham` dataset.

also produces better final log-likelihoods. Furthermore, our method also produces the best von Neumann divergences D_{vN} with `BASIC` initialization and moderately performs with `WISHART` initialization. The results show good stablity of our method; the proposed algorithm is considered to be favorable in many situations.

In Figure 2, we show the learning curves for each setting. The fixed-point algorithm convergences stably yet slightly slow with the step size $a = 1$, which is the only setting with guaranteed convergence and monotonicity. The Adam optimizer may fall into poor local optima, depending on the initial value. On the other hand, the proposed MM algorithm consistently indicates monotonic and rapid convergence.

Nottingham

The results of the `Nottingham` dataset are presented in Table 2, and the learning curves are showed in Figure 3. Similarly to the `Synthetic` datasets, the convergence of the MM algorithm is remarkably rapid.

Under the `BASIC` initialization, the MM and fixed-point algorithms get stuck in a poor local optimum. Since the optimal $\boldsymbol{L}^*$ is considered to have a particular structure, the `BASIC` initialization may not be compatible with `Nottingham`. We can see that only Adam, which allows for a non-monotonic increase of an objective, escapes the local optimum.

Amazon Baby Registry

In Table 3, we show the results in all the 13 categories of `Amazon baby registry`. Overall, our algorithm achieves better log-likelihood values and outstanding convergence speeds in most categories. Especially, when the sample size is relatively large, such as $M > 10{,}000$, our algorithm outperforms with a convergence speed that is about five times faster than the fixed-point algorithm.

Although the convergences of the MM algorithm seems to be slow in some of the smaller categories in Table 3, that is not very significant. In these cases, the MM algorithm quickly reaches a near optimum value, but takes longer to meet the stopping criterion. By managing the stopping criterion, we may be able to stop its learning much earlier.

Table 3: Final mean log-likelihoods and runtimes of the `Amazon baby registry` dataset. Each value is computed from the average or standard deviation of 30 trials and initialized by `WISHART`.

Category	Method	Log-likelihood	Runtime (s)	Category	Method	Log-likelihood	Runtime (s)
Apparel	FP	-10.20 ± 0.00	26.25 ± 0.64	Gear	FP	-9.27 ± 0.00	30.41 ± 0.22
$N = 100$	Adam	-10.30 ± 0.12	8.39 ± 2.36	$N = 100$	Adam	-9.35 ± 0.14	10.00 ± 2.51
$M =$ 14,970	MM	$\mathbf{-10.18} \pm 0.00$	$\mathbf{5.00} \pm 0.38$	$M =$ 16,823	MM	$\mathbf{-9.24} \pm 0.00$	$\mathbf{3.78} \pm 0.23$
Bath	FP	-8.79 ± 0.00	27.92 ± 0.37	Health	FP	-7.59 ± 0.00	14.51 ± 0.29
$N = 100$	Adam	-8.92 ± 0.16	8.07 ± 2.48	$N = 62$	Adam	-7.63 ± 0.08	4.44 ± 1.32
$M =$ 14,542	MM	$\mathbf{-8.75} \pm 0.00$	$\mathbf{3.73} \pm 0.32$	$M =$ 14,057	MM	$\mathbf{-7.55} \pm 0.00$	$\mathbf{3.43} \pm 0.50$
Bedding	FP	-8.79 ± 0.00	34.02 ± 0.55	Media	FP	-8.56 ± 0.00	4.64 ± 0.12
$N = 100$	Adam	-8.92 ± 0.18	9.56 ± 2.64	$N = 58$	Adam	-8.58 ± 0.02	$\mathbf{1.43} \pm 0.43$
$M =$ 16,370	MM	$\mathbf{-8.77} \pm 0.00$	$\mathbf{6.89} \pm 1.54$	$M =$ 5,904	MM	$\mathbf{-8.52} \pm 0.02$	2.61 ± 0.94
Carseats	FP	-5.18 ± 0.06	5.08 ± 3.03	Safety	FP	-4.75 ± 0.16	10.95 ± 9.16
$N = 34$	Adam	-5.04 ± 0.20	$\mathbf{3.61} \pm 2.99$	$N = 36$	Adam	-4.70 ± 0.26	$\mathbf{3.70} \pm 3.66$
$M =$ 7,566	MM	$\mathbf{-5.02} \pm 0.06$	4.69 ± 1.57	$M =$ 8,892	MM	$\mathbf{-4.57} \pm 0.06$	7.35 ± 2.10
Diaper	FP	-10.71 ± 0.00	27.32 ± 0.18	Strollers	FP	-5.66 ± 0.06	5.03 ± 3.49
$N = 100$	Adam	-10.80 ± 0.10	9.05 ± 2.39	$N = 40$	Adam	-5.56 ± 0.20	$\mathbf{3.27} \pm 3.66$
$M =$ 16,759	MM	$\mathbf{-10.67} \pm 0.00$	$\mathbf{5.05} \pm 0.42$	$M =$ 7,393	MM	$\mathbf{-5.47} \pm 0.06$	6.52 ± 2.45
Feeding	FP	-12.17 ± 0.00	27.92 ± 0.19	Toys	FP	-8.10 ± 0.00	9.82 ± 0.18
$N = 100$	Adam	-12.27 ± 0.05	8.48 ± 2.03	$N = 62$	Adam	-8.15 ± 0.05	2.98 ± 0.83
$M =$ 19,001	MM	$\mathbf{-12.15} \pm 0.00$	$\mathbf{5.11} \pm 0.31$	$M =$ 10,073	MM	$\mathbf{-8.07} \pm 0.00$	$\mathbf{2.68} \pm 0.62$
Furniture	FP	-4.85 ± 0.14	5.31 ± 5.25				
$N = 32$	Adam	-4.84 ± 0.17	$\mathbf{1.38} \pm 1.73$				
$M =$ 7,093	MM	$\mathbf{-4.67} \pm 0.07$	4.66 ± 1.67				

5 Conclusion and Future Work

In this paper, we developed an efficient learning method for full-rank DPPs based on the MM algorithm. Compared with the existing methods, our algorithm has many advantages: it has guaranteed convergence and monotonicity, requires no bothersome hyperparameters, convergences rapidly and stably, and is easy to implement. Upon considering the performance of our algorithm, we revealed that our algorithm provides a locally tighter minorizer than the existing method. We also assessed the empirical performance of our method through experiments on both synthetic and real-world datasets, outperforming in terms of convergence speed and reaching a better estimate in most experimental settings.

We believe that our algorithm is a strong candidate for learning full-rank DPPs at the present moment, but there is still much future work to be done. First, we need to deepen our understanding of the performance of our method. Proposition 3.3 partially addresses this question, but we consider it is not enough. Second, scaling up our method for large N is a crucial issue. Several numerical algorithms for solving large-sized CARE (9) have been proposed based on some structure of a problem: low-rank structure and/or sparsity (Bini et al., 2011; Simoncini, 2016). On the other hand, our CARE (7) formed by full-rank and dense matrices, therefore, exploring a good CARE solver is considered to be an essential task. Third, we are also interested in extending our algorithm. For example, by adding a “proximity term” between $\boldsymbol{L}$ and $\boldsymbol{L}^{(t)}$ with the logdet divergence, which is a kind of Bregman divergence, we can generalize our CARE to solve (7) as

$$\frac{1}{\mu^{(t)}}\boldsymbol{L} - \boldsymbol{L}\left\{(\boldsymbol{L}^{(t)} + \boldsymbol{I})^{-1} + \frac{1}{\mu^{(t)}}\boldsymbol{L}^{(t)-1}\right\}\boldsymbol{L} + \boldsymbol{Q}_M^{(t)} = \boldsymbol{O},$$

where $\mu^{(t)} \in (0, \infty]$ is a step size at the t-th iteration. This generalization is related to the mirror descent (Nemirovsky, 1983; Beck & Teboulle, 2003), which is an extension of the proximal gradient method. Although this generalization does not directly improve the performance of our algorithm, such a perspective will surely play an important role in further understanding and extensions.

References

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

P. Benner and R. Byers. An exact line search method for solving generalized continuous-time algebraic Riccati equations. *IEEE Transactions on Automatic Control*, 43(1):101–107, 1998.

Dario Bini, Bruno Iannazzo, and B. Meini. *Numerical Solution of Algebraic Riccati Equations.* Society for Industrial and Applied Mathematics, 2011.

Alexei Borodin and Eric M. Rains. Eynard–Mehta Theorem, Schur Process, and their Pfaffian Analogs. *Journal of Statistical Physics*, 121(3):291–317, 2005.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, pp. 1881–1888. Omnipress, 2012.

Michał Dereziński, Manfred K. Warmuth, and Daniel Hsu. Unbiased estimators for random design regression. *Journal of Machine Learning Research*, 23(167):1–46, 2022.

Christophe Dupuy and Francis Bach. Learning Determinantal Point Processes in Sublinear Time. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pp. 244–257. PMLR, 2018.

Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. Bayesian Low-Rank Determinantal Point Processes. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 349–356. ACM, 2016.

Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. Low-Rank Factorization of Determinantal Point Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017.

Jennifer A Gillenwater, Alex Kulesza, Emily Fox, and Ben Taskar. Expectation-Maximization for Learning Determinantal Point Processes. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

J. Hough, Manjunath Krishnapur, Yuval Peres, and Bálint Virág. *Zeros of Gaussian Analytic Functions and Determinantal Point Processes*, volume 51 of *University Lecture Series.* American Mathematical Society, 2009.

David R Hunter and Kenneth Lange. A Tutorial on MM Algorithms. *The American Statistician*, 58(1): 30–37, 2004.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations.* arXiv, 2015.

Alex Kulesza and B. Taskar. K-DPPs: Fixed-Size Determinantal Point Processes. In *International Conference on Machine Learning*, 2011.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, 2012.

A. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Transactions on Automatic Control*, 24(6):913–921, 1979.

Odile Macchi. The Coincidence Approach to Stochastic Point Processes. *Advances in Applied Probability*, 7 (1):83–122, 1975.

Zelda Mariet and Suvrit Sra. Fixed-point algorithms for learning determinantal point processes. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2389–2397. PMLR, 2015.

Zelda Mariet, Mike Gartrell, and Suvrit Sra. Learning Determinantal Point Processes by Corrective Negative Sampling. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pp. 2251–2260. PMLR, 2019.

Zelda E. Mariet and Suvrit Sra. Kronecker Determinantal Point Processes. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Naoto Miyoshi and Tomoyuki Shirai. A Cellular Network Model with Ginibre Configured Base Stations. *Advances in Applied Probability*, 46(3):832–845, 2014.

Arkadi Nemirovsky. *Problem Complexity and Method Efficiency in Optimization.* Wiley-Interscience Series in Discrete Mathematics. New York: Wiley, 1983.

Takayuki Osogami, Rudy Raymond, Akshay Goel, Tomoyuki Shirai, and Takanori Maehara. Dynamic Determinantal Point Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

Tomoyuki Shirai and Yoichiro Takahashi. Fermion process and Fredholm determinant. In *Proceedings of the Second ISAAC Congress*, pp. 15–23. Springer, 2000.

V. Simoncini. Computational Methods for Linear Matrix Equations. *SIAM Review*, 58(3):377–441, 2016.

A Soshnikov. Determinantal random point fields. *Russian Mathematical Surveys*, 55(5):923–975, 2000.

Ying Sun, Prabhu Babu, and Daniel P. Palomar. Robust Estimation of Structured Covariance Matrix for Heavy-Tailed Elliptical Distributions. *IEEE Transactions on Signal Processing*, 64(14):3576–3590, 2016.

Ying Sun, Prabhu Babu, and Daniel P. Palomar. Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2017.

Alan L Yuille and Anand Rangarajan. The Concave-Convex Procedure (CCCP). In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

Appendix: Proof of Proposition 3.3

Derivation of Equation (12)

In (11), the constant term is given by

$$\begin{aligned}
&\xi(\boldsymbol{L}^{(t)}) \\
&\quad = \frac{1}{M}\sum_{m=1}^{M}\left\{\log\det(\boldsymbol{U}_{\mathcal{A}_m}\boldsymbol{L}^{(t)}\boldsymbol{U}_{\mathcal{A}_m}^\top) + |\mathcal{A}_m|\right\} \\
&\qquad + \log\det\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}\boldsymbol{L}^{(t)}\} + \mathrm{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}\}.
\end{aligned}$$

By the following inequality from the Taylor expansion

$$-\log\det(\boldsymbol{L}^{(t)}) \geq -\log\det(\boldsymbol{L}) - \mathrm{tr}\{(\boldsymbol{L}^{-1}(\boldsymbol{L}^{(t)}-\boldsymbol{L})\},$$

we have

$$\begin{aligned}
&g(\boldsymbol{L}|\boldsymbol{L}^{(t)}) - h(\boldsymbol{L}|\boldsymbol{L}^{(t)}) \\
&\quad = \mathrm{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}(\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)} - \boldsymbol{I} - \boldsymbol{L} + \boldsymbol{L}^{(t)})\} \\
&\qquad + \log\det(\boldsymbol{L}) - \log\det(\boldsymbol{L}^{(t)})
\end{aligned}$$

$$
\begin{aligned}
&\geq \operatorname{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}(\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)}-\boldsymbol{I}-\boldsymbol{L}+\boldsymbol{L}^{(t)})\}\\
&\quad -\operatorname{tr}\{\boldsymbol{L}^{-1}(\boldsymbol{L}^{(t)}-\boldsymbol{L})\}\\
&= \operatorname{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}(\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)}-\boldsymbol{L}+2\boldsymbol{L}^{(t)})\}\\
&\quad -\operatorname{tr}\{\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)}\}-N+N\\
&= \operatorname{tr}\{(\boldsymbol{L}^{(t)}+\boldsymbol{I})^{-1}(2\boldsymbol{L}^{(t)}-\boldsymbol{L}-\boldsymbol{L}^{(t)}\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)})\}.
\end{aligned}
$$

Derivation of Equation (13)

Let $\boldsymbol{L}=\boldsymbol{L}^{(t)}+\delta\boldsymbol{M}$, where $\delta>0$ is a sufficient small coefficient and $\boldsymbol{M}$ is a symmetric matrix whose all eigenvalues are in $[-1,1]$. Then, we can approximate the matrix inverse as $\boldsymbol{L}^{-1}=(\boldsymbol{L}^{(t)}+\delta\boldsymbol{M})^{-1}\approx\boldsymbol{L}^{(t)-1}-\delta\boldsymbol{L}^{(t)-1}\boldsymbol{M}\boldsymbol{L}^{(t)-1}$ by the Taylor expansion. Using this approximation, we have

$$
\begin{aligned}
&2\boldsymbol{L}^{(t)}-\boldsymbol{L}-\boldsymbol{L}^{(t)}\boldsymbol{L}^{-1}\boldsymbol{L}^{(t)}\\
&\quad = 2\boldsymbol{L}^{(t)}-(\boldsymbol{L}^{(t)}+\delta\boldsymbol{M})\\
&\qquad -\boldsymbol{L}^{(t)}(\boldsymbol{L}^{(t)}+\delta\boldsymbol{M})^{-1}\boldsymbol{L}^{(t)}\\
&\quad \approx 2\boldsymbol{L}^{(t)}-(\boldsymbol{L}^{(t)}+\delta\boldsymbol{M})\\
&\qquad -\boldsymbol{L}^{(t)}(\boldsymbol{L}^{(t)-1}-\delta\boldsymbol{L}^{(t)-1}\boldsymbol{M}\boldsymbol{L}^{(t)-1})\boldsymbol{L}^{(t)}\\
&\quad = \boldsymbol{O}.
\end{aligned}
$$