Fixed-point algorithms for learning determinantal point processes

Zelda Mariet ZELDA@CSAIL.MIT.EDU
Suvrit Sra SUVRIT@MIT.EDU
Massachusetts Institute of Technology, Cambridge, MA 02139 USA

Abstract

Determinantal point processes (DPPs) offer an elegant tool for encoding probabilities over subsets of a ground set. Discrete DPPs are parametrized by a positive semidefinite matrix (called the DPP kernel), and estimating this kernel is key to learning DPPs from observed data. We consider the task of learning the DPP kernel, and develop for it a surprisingly simple yet effective new algorithm. Our algorithm offers the following benefits over previous approaches: (a) it is much simpler; (b) it yields equally good and sometimes even better local maxima; and (c) it runs an order of magnitude faster on large problems. We present experimental results on both real and simulated data to illustrate the numerical performance of our technique.

1. Introduction

Determinantal point processes (DPPs) arose in statistical mechanics, where they were originally used to model fermions (Macchi, 1975). Recently, they have witnessed substantial interest in a variety of machine learning applications (Kulesza, 2013; Kulesza and Taskar, 2012).

One of the key features of DPPs is their ability to model the notion of diversity while respecting quality, a concern that underlies the broader task of subset selection where balancing quality with diversity is a well-known issue—see e.g., document summarization (Lin and Bilmes, 2012), object retrieval (Affandi et al., 2014), recommender systems (Zhou et al., 2010), and sensor placement (Krause et al., 2008).

DPPs are also interesting in their own right: they have various combinatorial, probabilistic, and analytic properties, while involving a fascinating set of open problems (Lyons, 2003; Hough et al., 2006; Kulesza, 2013).

Within machine learning DPPs have found good use—see for instance (Gillenwater et al., 2014); (Kulesza and Taskar, 2011b); (Kulesza and Taskar, 2011a); (Affandi et al., 2014); (Affandi et al., 2103); (Affandi et al., 2013); (Gillenwater et al., 2012). For additional references and material we refer the reader to the survey (Kulesza and Taskar, 2012).

Our paper is motivated by the recent work of Gillenwater et al. (2014), who made notable progress on the task of learning a DPP kernel from data. This task is conjectured to be NP-Hard (Kulesza, 2013, Conjecture 4.1). Gillenwater et al. (2014) presented a carefully designed EM-style procedure, which, unlike several previous approaches (e.g., (Kulesza and Taskar, 2011b;a; Affandi et al., 2014)) learns a full DPP kernel nonparameterically.

One main observation of Gillenwater et al. (2014) is that applying projected gradient ascent to the DPP log-likelihood usually results in degenerate estimates (because it involves projection onto the set $\{X : 0 \preceq X \preceq I\}$). Hence one may wonder if instead we could apply more sophisticated manifold optimization techniques (Absil et al., 2009; Boumal et al., 2014). While this idea is attractive, and indeed applicable, e.g., via the excellent MANOPT toolbox (Boumal et al., 2014), empirically it turns out to be computationally too demanding; the EM strategy of Gillenwater et al. (2014) is more practical.

We depart from both EM and manifold optimization to develop a new learning algorithm that (a) is simple, yet powerful; and (b) yields essentially the same log-likelihood values as the EM approach while running significantly faster. In particular, our algorithm runs an order of magnitude faster on larger problems.

The key innovation of our approach is a derivation via a fixed-point view, which by construction ensures positive definiteness at every iteration. Its convergence analysis involves an implicit bound-optimization iteration to ensure monotonic ascent.[1] A pleasant byproduct of the fixed-point approach is that it avoids any eigenvalue/vector computations, enabling a further savings in running time.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

[1] The convergence analysis in this version of the paper improves upon our original submission, in that our proof is now constructive and requires weaker assumptions.

1.1. Background and problem setup

Without loss of generality we assume that the ground set of N items is $\{1, 2, \ldots, N\}$, which we denote by $\mathcal{Y}$. A (discrete) DPP on $\mathcal{Y}$ is a probability measure $\mathcal{P}$ on $2^{\mathcal{Y}}$ (the set of all subsets of $\mathcal{Y}$) such that for any $Y \subseteq \mathcal{Y}$, the probability $\mathcal{P}(Y)$ verifies $\mathcal{P}(Y) \propto \det(L_Y)$; here L_Y denotes the principal submatrix of the *DPP kernel* L induced by indices in Y. Intuitively, the diagonal entry L_{ii} of the kernel matrix L captures some notion of the importance of item i, whereas an off-diagonal entry $L_{ij} = L_{ji}$ measures similarity between items i and j. This intuitive notion provides further motivation for seeking DPPs with non-diagonal kernels when there is implicit interaction between the observed items.

The normalization constant for the measure $\mathcal{P}$ follows upon observing that $\sum_{Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I)$. Thus,

$$\mathcal{P}(Y) = \frac{\det(L_Y)}{\det(L+I)}, \qquad Y \subseteq \mathcal{Y}. \tag{1.1}$$

DPPs can also be given an alternative representation through a *marginal kernel* K that captures for a random $Y \sim \mathcal{P}$ and every $A \subseteq \mathcal{Y}$, the marginal probability

$$\mathcal{P}(A \subseteq Y) = \det(K_A). \tag{1.2}$$

It is easy to verify that $K = L(L+I)^{-1}$, which also implies that K and L have the same eigenvectors and differ only in their eigenvalues. It can also be shown (Kulesza, 2013) that $\mathcal{P}(Y) = |\det(K - I_{Y^c})|$, where I_{Y^c} is a partial $N \times N$ identity matrix with diagonal entries in Y zeroed out.

Both parameterizations (1.1) and (1.2) of the DPP probability are useful: Gillenwater et al. (2014) used a formulation in terms of K; we prefer (1.1) as it aligns better with our algorithmic approach.

1.2. Learning the DPP Kernel

The learning task aims to fit a DPP kernel (either L or equivalently the marginal kernel K) consistent with a collection of observed subsets. Suppose we obtain as training data n subsets $(Y_1, \ldots, Y_n)$ of the ground set $\mathcal{Y}$. The task is to maximize the likelihood of these observations. Two equivalent formulations of this maximization task may be considered:

$$\max_{L \succeq 0} \sum_{i=1}^{n} \log\det(L_{Y_i}) - n \log\det(I + L), \tag{1.3}$$

$$\max_{0 \preceq K \preceq I} \sum_{i=1}^{n} \log\big(|\det(K - I_{Y_i^c})|\big). \tag{1.4}$$

We will use formulation (1.3) in this paper. Gillenwater et al. (2014) used (1.4) and exploited its structure to derive a somewhat intricate EM-style method for optimizing it. Both (1.3) and (1.4) are nonconvex and difficult optimize. For instance, using projected gradient on (1.4) may seem tempting, but projection ends up yielding degenerate (diagonal and rank-deficient) solutions which is undesirable when trying to capture interaction between observations—indeed, this criticism motivated Gillenwater et al. (2014) to derive the EM algorithm.

We approach problem (1.3) from a different viewpoint (which also avoids projection) and as a result obtain a new optimization algorithm for estimating L. This algorithm, its analysis, and empirical performance are the subject of the remainder of the paper.

2. Optimization algorithm

The method that we derive has two key components: (i) a fixed-point view that helps obtain an iteration that satisfies the crucial positive definiteness constraint $L \succeq 0$ by construction; and (ii) an implicit bound optimization based analysis that ensures monotonic ascent. The resulting algorithm is vastly simpler than the previous EM-style approach of Gillenwater et al. (2014).

If $|Y| = k$, then for a suitable $N \times k$ indicator matrix U we can write $L_Y = U^* L U$, which is also known as a *compression* (U^* denotes the Hermitian transpose). We write $U_i^* L U_i$ interchangeably with L_{Y_i}, implicitly assuming suitable indicator matrices U_i such that $U_i^* U_i = I_{|Y_i|}$. To reduce clutter, we will drop the subscript on the identity matrix, its dimension being clear from context.

Denote by $\phi(L)$ the objective function in (1.3). Assume for simplicity that the constraint set is open, i.e., $L \succ 0$. Then any critical point of the log-likelihood must satisfy

$$\begin{aligned} &\nabla\phi(L) = 0, \quad \text{or equivalently} \\ &\sum\nolimits_{i=1}^{n} U_i \left(U_i^* L U_i\right)^{-1} U_i^* - n\,(I+L)^{-1} = 0. \end{aligned} \tag{2.1}$$

Any (strictly) positive definite solution to the nonlinear matrix equation (2.1) is a candidate locally optimal solution.

We solve this matrix equation by developing a fixed-point iteration. In particular, define

$$\Delta := \tfrac{1}{n} \sum\nolimits_{i=1}^{n} U_i \left(U_i^* L U_i\right)^{-1} U_i^* - (I+L)^{-1},$$

with which we may equivalently write (2.1) as

$$\Delta + L^{-1} = L^{-1}. \tag{2.2}$$

Equation (2.2) suggests the following iteration

$$L_{k+1}^{-1} \leftarrow L_k^{-1} + \Delta_k, \qquad k = 0, 1, \ldots. \tag{2.3}$$

A priori there is no reason for iteration (2.3) to be valid (i.e., converge to a stationary point). But we write it in this form to highlight its crucial feature: starting from an initial $L_0 \succ 0$, it generates positive definite iterates (Prop. 2.1).

Proposition 2.1. *Let $L_0 \succ 0$. Then, the sequence $\{L_k\}_{k\geq 1}$ generated by (2.3) remains positive definite.*

Proof. The proof is by induction. It suffices to show that

$$L \succ 0 \implies L^{-1} + \Delta \succ 0.$$

Since $I + L \succ L$, from the order inversion property of the matrix inverse map it follows that $L^{-1} \succ (I+L)^{-1}$. Now adding the matrix $\frac{1}{n}\sum_{i=1} U_i \left(U_i^* L U_i\right)^{-1} U_i^* \succeq 0$ we obtain the desired inequality by definition of Δ. □

A quick experiment reveals that iteration (2.3) *does not converge* to a local maximizer of $\phi(L)$. To fix this defect, we rewrite the key equation (2.2) in a different manner:

$$L = L + L\Delta L. \tag{2.4}$$

This equation is obtained by multiplying (2.2) on the left and right by L. Therefore, we now consider the iteration

$$L_{k+1} \leftarrow L_k + L_k \Delta_k L_k, \quad k = 0, 1, \ldots. \tag{2.5}$$

Prop. 2.1 in combination with the fact that congruence preserves positive definiteness (i.e., if $X \succeq 0$, then $Z^* X Z \succeq 0$ for any complex matrix Z), implies that if $L_0 \succ 0$, then the sequence $\{L_k\}_{k\geq 1}$ obtained from iteration (2.5) is also positive definite. What is more remarkable is that contrary to iteration (2.3), the sequence generated by (2.5) monotonically increases the log-likelihood.

While monotonicity is not apparent from our intuitive derivation above, it becomes apparent once we recognize an implicit change of variables that seems to underlie our method.

2.1. Convergence Analysis

Theorem 2.2. *Let L_k be generated via (2.5). Then, the sequence $\{\phi(L_k)\}_{k\geq 0}$ is monotonically increasing.*

Before proving Theorem 2.2 we need the following lemma.

Lemma 2.3. *Let $U \in \mathbb{C}^{N\times k}$ $(k \leq N)$ such that $U^*U = I$. The map $g(S) := \log\det(U^* S^{-1} U)$ is convex on the set of positive definite matrices.*

Proof. Since g is continuous it suffices to establish midpoint convexity. Consider therefore, $X, Y \succ 0$ and let

$$X \# Y = X^{1/2}(X^{-1/2} Y X^{-1/2})^{1/2} X^{1/2}$$

be their geometric mean. The operator inequality $X\#Y \preceq \frac{X+Y}{2}$ is well-known (Bhatia, 2007, Thm. 4.1.3). Hence,

$$\begin{aligned}\left(\tfrac{X+Y}{2}\right)^{-1} &\preceq (X\#Y)^{-1} = X^{-1}\#Y^{-1}\\ U^*\left(\tfrac{X+Y}{2}\right)^{-1} U &\preceq U^*(X^{-1}\#Y^{-1})U\\ &\preceq (U^*X^{-1}U)\#(U^*Y^{-1}U),\end{aligned}$$

where equality follows from (Bhatia, 2007, Thm. 4.1.3), and the final inequality follows from (Bhatia, 2007, Thm. 4.1.5)[2] Since $\log\det$ is monotonic on positive definite matrices and since $\det(A\#B) = \sqrt{\det A}\sqrt{\det B}$, it then follows that

$$\begin{aligned}\log\det\left(U^*\left(\tfrac{X+Y}{2}\right)^{-1}U\right) \leq\ & \tfrac{1}{2}\log\det(U^*X^{-1}U)\\ &+ \tfrac{1}{2}\log\det(U^*Y^{-1}U),\end{aligned}$$

which proves the lemma. □

Now we are ready to prove Theorem 2.2.

Proof (Thm. 2.2). The key insight is to consider $S = L^{-1}$ instead of L; this change is only for the analysis—the actual iteration that we implement is still (2.5).[3]

Writing $\psi(S) := \phi(L)$, we see that $\psi(S)$ equals

$$\begin{aligned}&\tfrac{1}{n}\sum\nolimits_i \log\det(U_i^* S^{-1} U_i) - \log\det(S^{-1} + I)\\ &= \log\det(S) + \tfrac{1}{n}\sum\nolimits_i \log\det(U_i^* S^{-1} U_i)\\ &\quad - \log\det(I + S).\end{aligned}$$

Let $h(S) = \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I+S)$, and $f(S) = \log\det(S)$. Clearly, f is concave in S, while h is convex is S; the latter from Lemma 2.3 and the fact that $-\log\det(I+S)$ is convex. This observation allows us to invoke iterative bound-optimization (an idea that underlies EM, CCCP, and other related algorithms).

In particular, we construct an *auxiliary function* ξ so that

$$\begin{aligned}\psi(S) &\geq \xi(S, R), \qquad \forall S, R \succ 0,\\ \psi(S) &= \xi(S, S), \qquad \forall S \succ 0.\end{aligned}$$

As in (Yuille and Rangarajan, 2003), we select ξ by exploiting the convexity of h: as $h(S) \geq h(R) + \langle \nabla h(R),\, S - R\rangle$, we simply set

$$\xi(S, R) := f(S) + h(R) + \langle \nabla h(R) \,|\, S - R\rangle\,.$$

Given an iterate S_k, we then obtain S_{k+1} by solving

$$S_{k+1} := \operatorname{argmax}_{S\succ 0}\ \xi(S, S_k), \tag{2.6}$$

which clearly ensures monotonicity: $\psi(S_{k+1}) \geq \psi(S_k)$.

Since (2.6) has an open set as a constraint and $\xi(S,\cdot)$ is strictly concave, to solve (2.6) it suffices to solve the necessary condition $\nabla_S \xi(S, S_k) = 0$. This amounts to

$$S^{-1} = (I + S_k)^{-1} + \tfrac{1}{n}\sum\nolimits_i S_k^{-1} U_i (U_i^* S_k^{-1} U_i)^{-1} U_i^* S_k^{-1}.$$

Rewriting in terms of L we immediately see that with $L_{k+1} = L_k + L_k\Delta_k L_k$, $\phi(L_{k+1}) \geq \phi(L_k)$ (the inequality is strict unless $L_{k+1} = L_k$). □

[2] For an explicit proof see (Sra and Hosseini, 2015, Thm. 8).

[3] Our previous proof was based on viewing iteration (2.5) as a scaled-gradient-like iteration. However, we find the present version more transparent for proving monotonicity.

Theorem 2.2 shows that iteration (2.5) is well-defined (positive definiteness was established by Prop. 2.1). The fixed-point formulation (2.5) actually suggests a broader iteration, with an additional step-size a:

$$L_{k+1} = L_k + aL_k\Delta_k L_k. \tag{2.7}$$

Above we showed that for $a = 1$ ascent is guaranteed. Empirically, $a > 1$ often works well; Prop. A.1 presents an easily computable upper bound on feasible a. We conjecture that for all feasible values $a \geq 1$, iteration (2.5) is guaranteed to increase the log-likelihood.

Moreover, all previous calculations can be redone in the context where $L = F^*WF$ for a fixed feature matrix F in order to learn the weight matrix W (under the assumption that S^*S is invertible), making our approach also useful in the context of feature-based DPP learning.

Pseudocode of our resulting learning method is presented in Algorithms 1 and 2. For simplicity, we recommend using a fixed value of a (which can be set at initialization).

Algorithm 1 Picard Iteration

Input: Matrix L, training set T, step-size $a > 0$.
for $i = 1$ **to** maxIter **do**
  $L \longleftarrow$ FixedPointMap(L, T, a)
  if stop(L, T, i) **then**
    break
  end if
end for
return L

Algorithm 2 FixedPointMap

Input: Matrix L, training set T, step-size $a > 0$
$Z \longleftarrow 0$
for Y **in** T **do**
  $Z_Y = Z_Y + L_Y^{-1}$
end for
return $L + aL(Z/|T| - (L + I)^{-1})L$

2.2. Iteration cost and convergence speed

The cost of each iteration of our algorithm is dominated by the computation of Δ, which costs a total of $O(\sum_{i=1}^{n} |Y_i|^3 + N^3) = O(n\kappa^3 + N^3)$ arithmetic operations, where $\kappa = \max_i |Y_i|$; the $O(|Y_i|^3)$ cost comes from the time required to compute the inverse $L_{Y_i}^{-1}$, while the N^3 cost stems from computing $(I + L)^{-1}$. Moreover, additional N^3 costs arise when computing $L\Delta L$.

In comparison, each iteration of the method of Gillenwater et al. (2014) costs $O(nN\kappa^2 + N^3)$, which is comparable to, though slightly greater than $O(n\kappa^3 + N^3)$ as $N \geq \kappa$. In applications where the sizes of the sampled subsets satisfy $\kappa \ll N$, the difference can be more substantial. Moreover, we do not need any eigenvalue/vector computations to implement our algorithm.

Finally, our iteration also runs slightly faster than the K-Ascent iteration, which costs $O(nN^3)$. Additionally, similarly to EM, our algorithm avoids the projection step necessary in the K-Ascent algorithm (which ensures $K \in \{X : 0 \preceq X \preceq I\}$). As shown in (Gillenwater et al., 2014), avoiding this step helps learn non-diagonal matrices.

We note in passing that similar to EM, assuming a non-singular local maximum, we can also obtain a local linear rate of convergence. This follows by relating iteration (2.5) to scaled-gradient methods (Bertsekas, 1999, §1.3) (except that we have an implicit PSD constraint).

3. Experimental results

We compare performance of our algorithm, referred to as *Picard iteration*[4], against the EM algorithm presented in Gillenwater et al. (2014). We experiment on both synthetic and real-world data.

For real-world data, we use the baby registry test on which results are reported in (Gillenwater et al., 2014). This dataset consists in $111,006$ sub-registries describing items across 13 different categories; this dataset was obtained by collecting baby registries from `amazon.com`, all containing between 5 and 100 products, and then splitting each registry into subregistries according to which of the 13 categories (such as "feeding", "diapers", "toys", etc.) each product in the registry belongs to. (Gillenwater et al., 2014) provides a more in-depth description of this dataset.

These sub-registries are used to learn a DPP capable of providing recommendations for these products: indeed, a DPP is well-suited for this task as it provides sets of products in a category that are popular yet diverse enough to all be of interest to a potential customer.

3.1. Implementation details

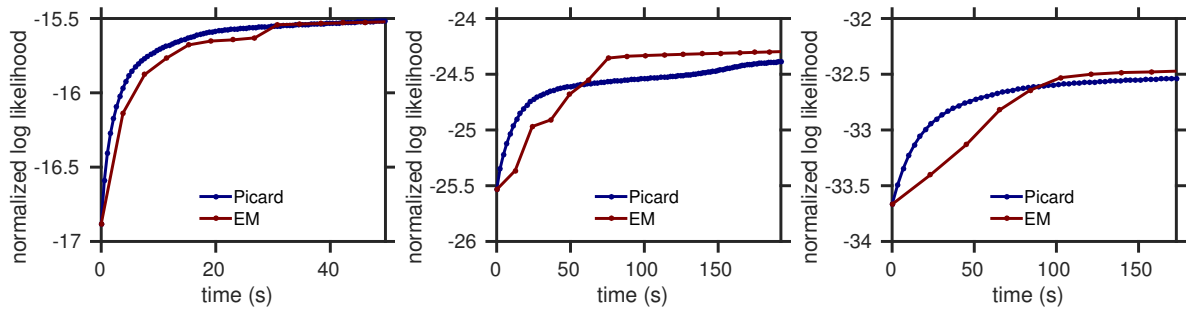
We measure convergence by testing the relative change $\frac{|\phi(L_{k+1}) - \phi(L_k)|}{|\phi(L_k)|} \leq \varepsilon$. We used a tighter convergence criterion for our algorithm ($\varepsilon_{\text{pic}} = 0.5 \cdot \varepsilon_{\text{em}}$) to account for the fact that the distance between two subsequent log-likelihoods tends to be smaller for the Picard iteration than for EM.

The parameter a for Picard was set at the beginning of each experiment and never modified as it remained valid throughout each test[5]. In EM, the step size was initially set to 1 and halved when necessary, as per the algorithm described in (Gillenwater et al., 2014); we used the code of Gillenwater et al. (2014) for our EM implementation[6].

[4] Our nomenclature stems from the usual name for such iterations in fixed-point theory (Granas and Dugundji, 2003).

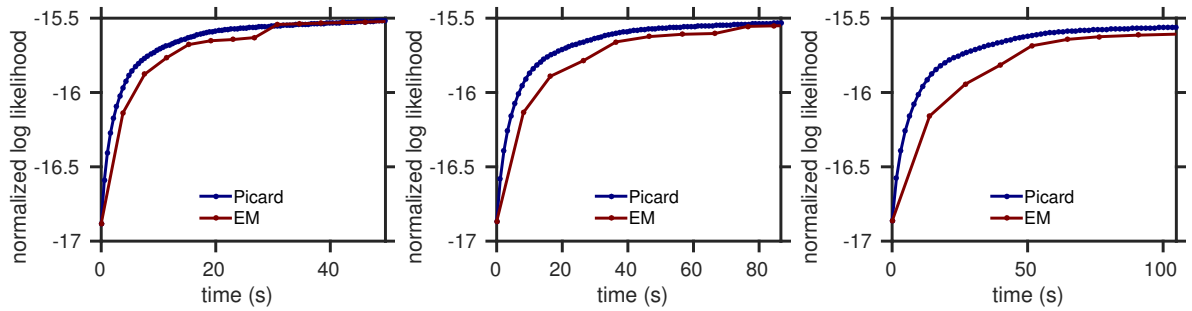
[5] Although it was not necessary in our experiments, if the parameter a becomes invalid, it can be halved until it reaches 1.

[6] These experiments were run with MATLAB, on a Linux Mint



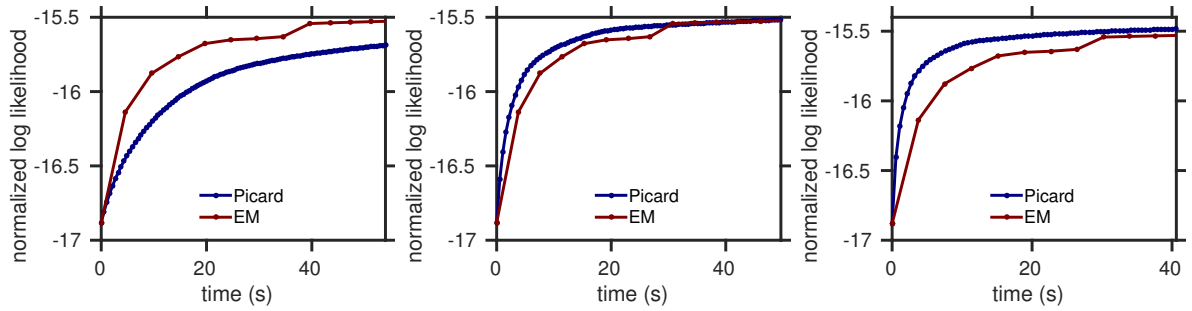
(a) $N = 50$ (b) $N = 100$ (c) $N = 150$

Figure 1. Normalized log-likelihood as a function of time for various set sizes N, with $n = 5000$ and $a = 5$ using the `BASIC` random distribution.



(a) $n = 5000$ (b) $n = 10,000$ (c) $n = 15,000$

Figure 2. Normalized log likelihood as a function of time for various numbers of training sets, with $N = 50$ and $a = 5$ using the `BASIC` random distribution.



(a) $a = 1$ (b) $a = 5$ (c) $a = 10$

Figure 3. Normalized log likelihood as a function of time for different values of a, with $N = 50$ and $n = 5000$ using the `BASIC` random distribution.

3.2. Synthetic tests

In each experiment, we sample n training sets from a base DPP of size N, then learn the DPP using EM and the Picard iteration. We initialize the learning process with a random positive definite matrix L_0 (or K_0 for EM) drawn from the same distribution as the true DPP kernel.

system, using 16GB of RAM and an i7-4710HQ CPU @ 2.50GHz.

Specifically, we used two matrix distributions to draw the true kernel and the initial matrix values from:

- `BASIC`: We draw the coefficients of a matrix M from the uniform distribution over $\left[0, \sqrt{2}\right]$, then return $L = MM^\top$ conditioned on its positive definiteness.
- `WISHART`: We draw L from a Wishart distribution with N degrees of freedom and an identity covariance

matrix, and rescale it with a factor $\frac{1}{N}$.

Figures 1, 2 and 3 show the log-likelihood as a function of time for different parameter values when both the true DPP kernel and the initial matrix L_0 were drawn from the BASIC distribution. Tables 1 and 2 show the final log-likelihood and the time necessary for each method to reach 99% of the optimal log likelihood for both distributions and parameters $n = 5000$, $a = 5$.

As shown in Figure 1, the difference in time necessary for both methods to reach a good approximation of the final likelihood (as defined by best final likelihood) grows drastically as the size N of the set of all elements $\{1, 2, \ldots, N\}$ increases. Figure 2 illustrates the same phenomenon when N is kept constant and n increases.

Finally, the influence of the parameter a on convergence speed is illustrated in Figure 3[7]. Increasing a noticeably increases Picard's convergence speed, as long as the matrices remain positive definite during the Picard iteration.

Table 1. Final log-likelihoods and time necessary for an iteration to reach 99% of the optimal log likelihood for both algorithms when using BASIC distribution for true and initialization matrices (training set size of 5,000, $a = 5$).

	LOG-LIKELIHOOD		RUNTIME TO 99%	
	PICARD	EM	PICARD	EM
$N = 50$	-15.5	-15.5	17.3s	30.7s
$N = 100$	-24.4	-24.2	143s	75.5s
$N = 150$	-32.5	-32.5	40.7s	84.0s
$N = 200$	-40.8	-41.2	51.1s	1,730s
$N = 250$	-45.7	-46.0	99.1s	2,850s

Table 2. Final log-likelihoods and time necessary for an iteration to reach 99% of the optimal log likelihood for both algorithms when using WISHART distribution for true and initialization matrices (training set size of 5,000, $a = 5$).

	LOG-LIKELIHOOD		RUNTIME TO 99%	
	PICARD	EM	PICARD	EM
$N = 50$	-33.0	-33.1	0.2s	2.0s
$N = 100$	-66.2	-66.2	0.5s	3.6s
$N = 150$	-99.2	-99.3	0.8s	5.2s
$N = 200$	-132.1	-132.4	1.2s	8.9s
$N = 250$	-165.1	-165.7	2.5s	11s

The greatest strength of the Picard iteration lies in its initial rapid convergence: the log-likelihood increases significantly faster for the Picard iteration than for EM. Although for small datasets EM sometimes performs better, our algorithm provides substantially better results in shorter timeframes when dealing with larger datasets.

[7]In the cases where $a > 1$, a safeguard was added to check that the matrices returned by our algorithm were positive definite.

Overall, our algorithm converges to 99% of the optimal log-likelihood (defined as the maximum of the log-likelihoods returned by each algorithm) significantly faster than the EM algorithm for both distributions, particularly when dealing with large values of N.

Thus, the Picard iteration is preferable when dealing with large ground sets; it is also very well-suited to cases where larger amounts of training data are available.

3.3. Baby registries experiment

We tested our implementation on all 13 product categories in the baby registry dataset, using two different initializations:

- the aforementioned Wishart distribution
- the data-dependent moment matching initialization (MM) described in (Gillenwater et al., 2014)

In each case, 70% of the baby registries in the product category were used for training; 30% served as test. The results presented in Figures 4 and 5 are averaged over 5 learning trials, each with different initial matrices; the parameter a was set equal to 1.3 for all iterations.

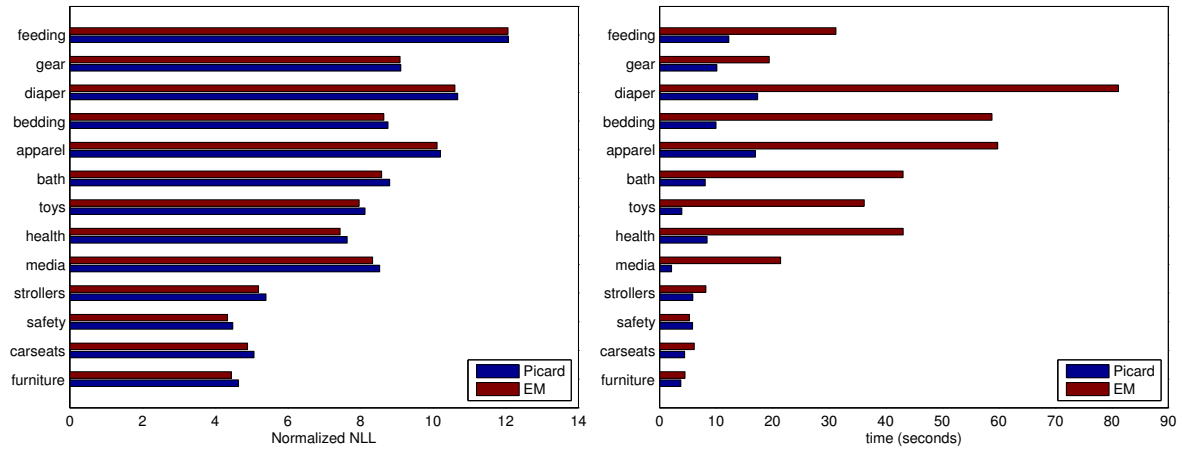
Table 3. Comparison of final log-likelihoods for both algorithms; relative closeness between Picard and EM: $\delta = |\phi_{\text{em}} - \phi_{\text{pic}}|/\phi_{\text{em}}$.

CATEGORY	δ (WISHART)	δ (MM)
FURNITURE	4.4E-02	1.2E-03
CARSEATS	3.7E-02	7.6E-04
SAFETY	3.3E-02	8.0E-04
STROLLERS	3.9E-02	3.0E-03
MEDIA	2.3E-02	2.4E-03
HEALTH	2.6E-02	7.4E-03
TOYS	2.0E-02	5.9E-03
BATH	2.6E-02	2.9E-03
APPAREL	9.2E-03	4.3E-03
BEDDING	1.3E-02	7.6E-03
DIAPER	7.2E-03	5.3E-03
GEAR	2.3E-03	9.0E-03
FEEDING	4.9E-04	2.1E-03

Similarly to its behavior on synthetic datasets, the Picard iteration provides overall significantly shorter runtimes when dealing with large matrices and training sets. As shown in Table 3, the final log-likelihoods are very close (on the order 10^{-2} to 10^{-4}) to those attained by the EM algorithm.

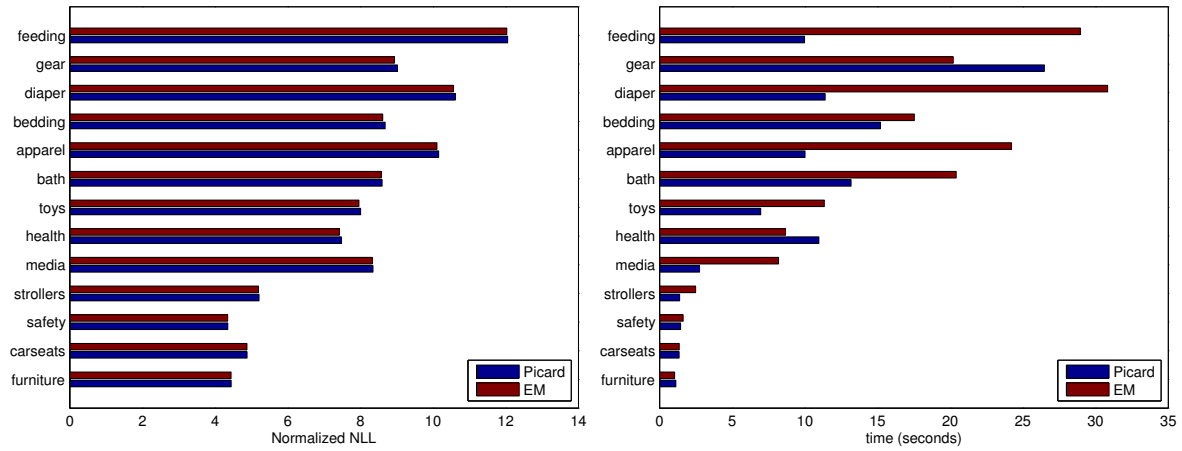
Using a moments-matching initialization leaves Picard's runtimes overall unchanged (a notable exception being the 'gear' category). However, EM's runtime decreases drastically with this initialization, although it remains significantly longer than Picard's in most categories.

The final log-likelihoods are also closer when using moments-matching initialization (see Table 3).



(a) Final negative log-likelihood (b) Runtime

Figure 4. Evaluation of EM and the Picard iteration on the baby registries dataset using Wishart initialization.



(a) Final negative log-likelihood (b) Runtime

Figure 5. Evaluation of EM and the Picard iteration on the baby registries dataset using moments-matching initialization.

4. Conclusions and future work

We approached the problem of maximum-likelihood estimation of a DPP kernel from a different angle: we analyzed the stationarity properties of the cost function and used them to obtain a novel fixed-point Picard iteration. Experiments on both simulated and real data showed that for a range of ground set sizes and number of samples, our Picard iteration runs remarkably faster that the previous best approach, while being extremely simple to implement. In particular, for large ground set sizes our experiments show that our algorithm cuts down runtime to a fraction of the previously optimal EM runtimes.

We presented a theoretical analysis of the convergence properties of the Picard iteration, and found sufficient conditions for its convergence. However, our experiments reveal that the Picard iteration converges for a wider range of step-sizes (parameter a in the iteration and plots) than currently accessible to our theoretical analysis. It is a part of our future work to develop more complete convergence theory, especially because of its strong empirical performance.

In light of our results, another line of future work is to apply fixed-point analysis to other DPP learning tasks.

ACKNOWLEDGMENTS

Suvrit Sra is partly supported by NSF grant: IIS-1409802.

A. Bound on a

Proposition A.1. *Let L, U_i, and Δ be as defined above; set $Z = \frac{1}{n}\sum_i U_i \left(U_i^* L U_i\right)^{-1} U_i^*$. Define the constant*

$$\gamma := \max\{\lambda_{\min}(LZ), 1/\lambda_{\max}(I+L)\}. \tag{A.1}$$

Then, $0 \leq \gamma \leq 1$ and for $a \leq (1-\gamma)^{-1}$ the update

$$L' \leftarrow L + aL\Delta L$$

ensures that L' is also positive definite.

Proof. Let $Z = \frac{1}{n}\sum_{i=1}^{n} U_i \left(U_i^* L U_i\right)^{-1} U_i^*$.

To ensure $L + aL\Delta L \succ 0$ we equivalently show

$$\begin{aligned}
&L^{-1} + a\Big(\tfrac{1}{n}\sum_{i=1}^{n} U_i \left(U_i^* L U_i\right)^{-1} U_i^* - (L+I)^{-1}\Big) \succ 0 \\
&\implies I + aL^{1/2} Z L^{1/2} \succ aL\left(L+I\right)^{-1} \\
&\implies (1-a)I + a(I+L)^{-1} + aL^{1/2} Z L^{1/2} \succ 0 \\
&\qquad\qquad \text{(as } L(L+I)^{-1} = I - (I+L)^{-1}\text{)} \\
&\implies (1-a) + a\lambda_{\min}((I+L)^{-1} + L^{1/2} Z L^{1/2}) > 0.
\end{aligned}$$

This inequality can be numerically optimized to find the largest feasible value of a. The simpler bound in question can be obtained by noting that

$$\begin{aligned}
&\lambda_{\min}((I+L)^{-1} + L^{1/2} Z L^{1/2}) \\
&\geq \max\{\lambda_{\min}(LZ), 1/\lambda_{\max}(I+L)\} = \gamma.
\end{aligned}$$

Thus, we have the easily computable bound for feasible a:

$$a \leq \frac{1}{1-\gamma}.$$

Clearly, by construction $\gamma \geq 0$. To see why $\gamma \leq 1$, observe that $(I+L) \prec I$, so that $\lambda_{\min}((I+L)^{-1}) < 1$. Further, block-matrix calculations show that $Z \preceq L^{-1}$, whereby $\lambda_{\min}(L^{1/2} Z L^{1/2}) \leq \lambda_{\min}(I) = 1$. □

References

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

R. Affandi, A. Kulesza, E. Fox, and B. Taskar. Nyström approximation for large-scale Determinantal Point Processes. In *Artificial Intelligence and Statistics (AISTATS)*, 2013.

R. Affandi, E. Fox, R. Adams, and B. Taskar. Learning the parameters of Determinantal Point Process kernels. In *International Conference on Machine Learning*, 2014.

R. Affandi, E. Fox, and B. Taskar. Approximate inference in continuous Determinantal Point Processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2103.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

R. Bhatia. *Positive Definite Matrices*. Princeton University Press, 2007.

N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. URL http://www.manopt.org.

J. Gillenwater, A. Kulesza, and B. Taskar. Near-optimal MAP inference for Determinantal Point Processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

J. Gillenwater, A. Kulesza, E. Fox, and B. Taskar. Expectation-Maximization for learning Determinantal Point Processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

A. Granas and J. Dugundji. *Fixed-point theory*. Springer, 2003.

J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág. Determinantal processes and independence. *Probability Surveys*, 3(206–229):9, 2006.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 9:235–284, 2008.

A. Kulesza. *Learning with Determinantal Point Processes*. PhD thesis, University of Pennsylvania, 2013.

A. Kulesza and B. Taskar. k-DPPs: Fixed-size Determinantal Point Processes. In *International Conference on Maachine Learning (ICML)*, 2011a.

A. Kulesza and B. Taskar. Learning Determinantal Point Processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2011b.

A. Kulesza and B. Taskar. *Determinantal Point Processes for machine learning*, volume 5. Foundations and Trends in Machine Learning, 2012.

H. Lin and J. Bilmes. Learning mixtures of submodular shells with application to document summarization. In *Uncertainty in Artificial Intelligence (UAI)*, 2012.

R. Lyons. Determinantal probability measures. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 98(1):167–212, 2003.

O. Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1), 1975.

S. Sra and R. Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM Journal on Optimization*, 25(1):713–739, 2015.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Comput.*, 15(4):915–936, Apr. 2003. ISSN 0899-7667.

T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.