Exact sampling of determinantal point processes with sublinear time preprocessing

Michał Dereziński[*]
Department of Statistics
University of California, Berkeley
mderezin@berkeley.edu

Daniele Calandriello[*]
LCSL
Istituto Italiano di Tecnologia, Italy
daniele.calandriello@iit.it

Michal Valko
DeepMind Paris
valkom@deepmind.com

Abstract

We study the complexity of sampling from a distribution over all index subsets of the set $\{1, ..., n\}$ with the probability of a subset S proportional to the determinant of the submatrix $\mathbf{L}_S$ of some $n \times n$ positive semidefinite matrix $\mathbf{L}$, where $\mathbf{L}_S$ corresponds to the entries of $\mathbf{L}$ indexed by S. Known as a determinantal point process (DPP), this distribution is used in machine learning to induce diversity in subset selection. When sampling from DDPs, we often wish to sample *multiple* subsets S with small expected size $k \triangleq \mathbb{E}[|S|] \ll n$ from a very large matrix $\mathbf{L}$, so it is important to minimize the preprocessing cost of the procedure (performed once) as well as the sampling cost (performed repeatedly). For this purpose we provide DPP-VFX, a new algorithm which, given access *only* to $\mathbf{L}$, samples *exactly* from a determinantal point process while satisfying the following two properties: (1) its preprocessing cost is $n \cdot \mathrm{poly}(k)$, i.e., *sublinear* in the size of $\mathbf{L}$, and (2) its sampling cost is $\mathrm{poly}(k)$, i.e., *independent* of the size of $\mathbf{L}$. Prior to our results, state-of-the-art exact samplers required $\mathcal{O}(n^3)$ preprocessing time and sampling time linear in n or dependent on the spectral properties of $\mathbf{L}$. We furthermore give a reduction which allows using our algorithm for *exact* sampling from cardinality constrained determinantal point processes with $n \cdot \mathrm{poly}(k)$ time preprocessing. Our implementation of DPP-VFX is provided at https://github.com/guilgautier/DPPy/.

1 Introduction

Given a positive semi-definite (psd) $n \times n$ matrix $\mathbf{L}$, a determinantal point process $\mathrm{DPP}(\mathbf{L})$, also known as an L-ensemble, is a distribution over all 2^n index subsets $S \subseteq \{1, \ldots, n\}$ such that

$$\Pr(S) \triangleq \frac{\det(\mathbf{L}_S)}{\det(\mathbf{I} + \mathbf{L})},$$

where $\mathbf{L}_S$ denotes the $|S| \times |S|$ submatrix of $\mathbf{L}$ with rows and columns indexed by S. Determinantal point processes naturally appear across many scientific domains [Mac75, BLMV17, Gue83], and they have emerged as an important tool in machine learning [KT12] for inducing diversity in subset selection and as a variance reduction approach. DPP sampling has been successfully applied in core ML problems such as recommender systems [GKVM18, CZZ18, Bru18], stochastic optimization [ZKM17, ZÖMS19], data summarization [CKS+18], Gaussian processes [MS16, BRVDW19],

---

[*]Equal contribution.

	exact	DPP	k-DPP	first sample	subsequent samples
[HKP+06, KT11]	✓	✓	✓	n^3	nk^2
[AGR16]	✗	✗	✓	$n \cdot \text{poly}(k)$	$n \cdot \text{poly}(k)$
[LJS16b]	✗	✓	✗	$n^2 \cdot \text{poly}(k)$	$n^2 \cdot \text{poly}(k)$
[LGD18]	✓	✓	✗	n^3	$\text{poly}(k \cdot (1 + \|\mathbf{L}\|))$
[Der19]	✓	✓	✗	n^3	$\text{poly}(\text{rank}(\mathbf{L}))$
DPP-VFX **(this paper)**	✓	✓	✓	$n \cdot \text{poly}(k)$	$\text{poly}(k)$

Table 1: Comparison of DPP and k-DPP algorithms using the L-ensemble representation. For a DPP, k denotes the expected subset size. Note that $k \leq \text{rank}(\mathbf{L}) \leq n$. We omit log terms for clarity.

experimental design [DW17, DWH18], and many more. In these applications, we often wish to efficiently produce many DPP samples of small expected size[2] $k \triangleq \mathbb{E}[|S|]$ given a large matrix $\mathbf{L}$. Sometimes, the distribution is restricted to subsets of fixed size $|S| = k \ll n$, denoted k-DPP($\mathbf{L}$). [HKP+06] gave an algorithm for drawing samples from DPP($\mathbf{L}$) distributed exactly, later adapted to k-DPP($\mathbf{L}$) by [KT11], which can be implemented to run in polynomial time. In many applications, however, sampling is still a computational bottleneck because the algorithm requires performing the eigendecomposition of matrix $\mathbf{L}$ at the cost of $\mathcal{O}(n^3)$. In addition to that initial cost, producing many independent samples $S_1, S_2, \ldots$ at high frequency poses a challenge because the cost of each sample is at least linear in n. Many alternative algorithms exist for both DPPs and k-DPPs to reduce the computational cost of preprocessing and/or sampling, including many approximate and heuristic approaches. Contrary to approximate solutions, we present an algorithm which samples *exactly* from a DPP or a k-DPP with the initial preprocessing cost *sublinear* in the size of $\mathbf{L}$ and the sampling cost *independent* of the size of $\mathbf{L}$.

Theorem 1 *For a psd $n \times n$ matrix $\mathbf{L}$, let $S_1, S_2, \ldots$ be i.i.d. random sets from* DPP($\mathbf{L}$) *or from any k-DPP($\mathbf{L}$). Then, there is an algorithm which, given access to $\mathbf{L}$, returns*

- *a) the first subset, S_1, in:* $\quad n \cdot \text{poly}(k) \, \text{polylog}(n)$ *time,*
- *b) each subsequent S_i in:* $\quad \text{poly}(k)$ *time.*

We refer to this algorithm as the Very Fast and eXact DPP sampler, or DPP-VFX. Table 1 compares DPP-VFX with other DPP and k-DPP sampling algorithms. In this comparison, we feature the methods that provide strong accuracy guarantees. As seen from the table, our algorithm is the first exact sampler to achieve sublinear overall runtime. Only the approximate MCMC sampler of [AGR16] matches our $n \cdot \text{poly}(k)$ complexity (and only for a k-DPP), but for this method every next sample is equally expensive, making it less practical when repeated sampling is needed. In fact, to our knowledge, no other exact or approximate method (with rigorous approximation guarantees) achieves $\text{poly}(k)$ sampling time of the present paper.

Our method is based on a technique developed recently by [DWH18, DWH19] and later extended by [Der19]. In this approach, we carefully downsample the index set $[n] = \{1, ..., n\}$ to a sample $\sigma = (\sigma_1, ..., \sigma_t) \in [n]^t$ that is small but still sufficiently larger than the expected target size k, and then run a DPP on σ. As the downsampling distribution we use a *regularized* determinantal point process (R-DPP), proposed by [Der19], which (informally) samples σ with probability $\Pr(\sigma) \sim \det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma)$, where $\widetilde{\mathbf{L}}$ is a rescaled version of $\mathbf{L}$. We can summarize this approach as follows, where $|S| \leq t \ll n$,

$$\{1, ..., n\} \xrightarrow{\sigma \sim \text{R-DPP}} (\sigma_1, ..., \sigma_t) \xrightarrow{\widetilde{S} \sim \text{DPP}} S = \{\sigma_i : i \in \widetilde{S}\}.$$

The DPP algorithm proposed by [Der19] follows the same diagram, however it requires that the size of the intermediate sample σ be $\Omega(\text{rank}(\mathbf{L}) \cdot k)$. This means that their method provides improvement over [HKP+06] only when $\mathbf{L}$ can be decomposed as $\mathbf{X}\mathbf{X}^\top$ for some $n \times r$ matrix $\mathbf{X}$, with $r \ll n$. However, in practice, matrix $\mathbf{L}$ is often only *approximately low-rank*, i.e., it exhibits some form of eigenvalue decay but it does not have a low-rank factorization. In this case, the results of [Der19] are vacuous both in terms of the preprocessing cost and the sampling cost, in that obtaining *every* sample would take $\Omega(n^3)$. We propose a different R-DPP implementation (see DPP-VFX as Algorithm 1)

---

[2] To avoid complicating the exposition with edge cases, we assume $k \geq 1$. Note that this can be always satisfied without distorting the distribution by rescaling $\mathbf{L}$ by a constant, and is without loss of generality, as our analysis can be trivially extended to the case $0 \leq k < 1$ with some additional notation.

where the expected size of σ is $\mathcal{O}(k^2)$. To make the algorithm efficient, we use new connections between determinantal point processes, ridge leverage scores, and Nyström approximation.

Definition 1 *Given a psd matrix $\mathbf{L}$, its ith λ-ridge leverage score (RLS) $\tau_i(\lambda)$ is the ith diagonal entry of $\mathbf{L}(\lambda\mathbf{I} + \mathbf{L})^{-1}$. The λ-effective dimension $d_{\text{eff}}(\lambda)$ is the sum of the leverage scores, $\sum_i \tau_i(\lambda)$.*

An important connection between RLSs and DPPs is that when $S \sim \text{DPP}(\mathbf{L})$, the marginal probability of index i being sampled into S is equal to the ith 1-ridge leverage score of $\mathbf{L}$, and the expected size k of S is equal to the 1-effective dimension,

$$\Pr(i \in S) = \left[\mathbf{L}(\mathbf{I} + \mathbf{L})^{-1}\right]_{ii} = \tau_i(1), \qquad k \triangleq \mathbb{E}\big[|S|\big] = \text{tr}\big(\mathbf{L}(\mathbf{I} + \mathbf{L})^{-1}\big) = d_{\text{eff}}(1).$$

Intuitively, if the marginal probability of i is high, then this index should likely make it into the intermediate sample σ. This suggests that i.i.d. sampling of the indices $\sigma_1, ..., \sigma_t$ proportionally to 1-ridge leverage scores, i.e., $\Pr(\sigma_1 = i) \propto \tau_i(1)$, should serve as a reasonable and cheap heuristic for constructing σ. In fact, we can show that this distribution can be easily corrected by rejection sampling to become the R-DPP that we need. Computing ridge leverage scores exactly costs $\mathcal{O}(n^3)$, so instead we compute them approximately by first constructing a Nyström approximation of $\mathbf{L}$.

Definition 2 *Let $\mathbf{L}$ be a psd matrix and C a subset of its row/column indices with size $m \triangleq |C|$. Then we define the Nyström approximation of $\mathbf{L}$ based on C as the $n \times n$ matrix $\widehat{\mathbf{L}} \triangleq (\mathbf{L}_{C,\mathcal{I}})^\top \mathbf{L}_C^+ \mathbf{L}_{C,\mathcal{I}}$.*

Here, $\mathbf{L}_{C,\mathcal{I}}$ denotes an $m \times n$ matrix consisting of (entire) rows of $\mathbf{L}$ indexed by C and $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse. Since we use rejection sampling to achieve the right intermediate distribution, the correctness of our algorithm does not depend on which Nyström approximation is chosen. However, the subset C greatly influences the computational cost of the sampling through the rank of $\widehat{\mathbf{L}}$ and the probability of rejecting a sample. Since $\text{rank}(\widehat{\mathbf{L}}) = |C|$, operations such as multiplication and inversion involving the Nyström approximation will scale with m, and therefore a small subset increases efficiency. However, if $\widehat{\mathbf{L}}$ is too different from $\mathbf{L}$, the probability of rejecting the sample will be very high and the algorithm inefficient. In this case, a slightly larger subset could improve accuracy and acceptance rate without increasing too much the cost of handling $\widehat{\mathbf{L}}$. Therefore, subset C has to be selected so that it is both small and accurately represents the matrix $\mathbf{L}$. Here, we once again rely on ridge leverage score sampling which has been effectively used for obtaining good Nyström approximations in a number of prior works such as [AM15, CLV17, RCCR18].

While our main algorithm can sample only from the *random-size* DPP, and not from the *fixed-size* k-DPP, we present a rigorous reduction argument which lets us use our DPP algorithm to sample exactly from a k-DPP (for any k) with a small computational overhead.

Related work Prior to our work, fast exact sampling from generic DPPs has been considered out of reach. The first procedure to sample general DPPs was given by [HKP+06] and even most recent exact refinements [LGD18, Der19, Pou19], when the DPP is represented in the form of an L-ensemble, require preprocessing that amounts to an expensive $n \times n$ matrix diagonalization at the cost $\mathcal{O}(n^3)$, which is shown as the *first-sample* complexity column in Table 1.

Nonetheless, there are well-known samplers for very specific DPPs that are both fast and exact, for instance for sampling uniform spanning trees [Ald90, Bro89, PW98], which leaves the possibility of a more generic fast sampler open. Since the sampling from DPPs has several practical large scale machine learning applications [KT12], there are now a number of methods known to be able to sample from a DPP *approximately*, outlined in the following paragraphs.

As DPPs can be specified by kernels (L-kernels or K-kernels), a natural approximation strategy is to resort to low-rank approximations [KT11, GKT12, AKFT13, LJS16a]. For example, [AKFT13] provides approximate guarantee for the probability of any subset being sampled as a function of eigengaps of the L-kernel. Next, [LJS16a] construct *coresets* approximating a given k-DPP and then use them for sampling. In their Section 4.1, [LJS16a] show in which cases we can hope for a good approximation. These guarantees become tight if these approximations (Nyström subspace, coresets) are aligned with data. In our work, we aim for an *adaptive* approach that is able to provide a good approximation for *any* DPP.

The second class of approaches are based on Markov chain Monte-Carlo [MU49] techniques [Kan13, RK15, AGR16, LJS16b, GBV17]. There are known polynomial bounds on the mixing rates [DS91]

of MCMC chains with arbitrary DPPs as their limiting measure. In particular, [AGR16] showed them for cardinality-constrained DPPs and [LJS16b] for the general case. The two chains have mixing times which are, respectively, linear and quadratic in n (see Table 1). Unfortunately, for any subsequent sample we need to wait until the chain *mixes again*.

Neither the known low-rank approximations or the known MCMC methods are able to provide samples that are *exactly* distributed according to a DPP (also called *perfect sampling*). This is not surprising as having *scalable and exact* sampling is very challenging in general. For example, methods based on rejection sampling are *always exact*, but they typically scale poorly to high-dimensional data and are adversely affected by the spikes in the distribution [EVCM16], resulting in high rejection rate and inefficiency. Surprisingly, our method is based on both low-rank approximation (a source of inaccuracy) and rejection sampling (a common source of inefficiency). In the following section, we show how to obtain a *perfect DPP sampler* from a Nyström approximation of the L-kernel. Then, to guarantee efficiency, in Section 3 we bound the number of rejections, which is possible thanks to the use of *intermediate downsampling*.

2   Exact sampling using any Nyström approximation

Notation   We use $[n]$ to denote the set $\{1, \ldots, n\}$. For a matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ and index sets C, D, we use $\mathbf{B}_{C,D}$ to denote the submatrix of $\mathbf{B}$ consisting of the intersection of rows indexed by C with columns indexed by D. If $C = D$, we use a shorthand $\mathbf{B}_C$ and if $D = [m]$, we may write $\mathbf{B}_{C,\mathcal{I}}$. Finally, we also allow C, D to be multisets or sequences, in which case each row/column is duplicated in the matrix according to its multiplicity (and in the case of sequences, we order the rows/columns as they appear in the sequence). Note that with this notation if $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$ then $\mathbf{L}_{C,D} = \mathbf{B}_{C,\mathcal{I}}\mathbf{B}_{D,\mathcal{I}}^\top$.

As discussed in the introduction, our method relies on an intermediate downsampling distribution to reduce the size of the problem. The exactness of our sampler relies on the careful choice of that intermediate distribution. To that end, we use *regularized* determinantal processes, introduced by [Der19]. In the below definition, we adapt them to the kernel setting.

Definition 3   *Given an $n \times n$ psd matrix $\mathbf{L}$, distribution $p \triangleq (p_1, \ldots, p_n)$ and $r > 0$, let $\widetilde{\mathbf{L}}$ denote an $n \times n$ matrix such that $\widetilde{L}_{ij} \triangleq \frac{1}{r\sqrt{p_i p_j}} L_{ij}$ for all $i, j \in [n]$. We define $\text{R-DPP}_p^r(\mathbf{L})$ as a distribution over events $A \subseteq \bigcup_{k=0}^\infty [n]^k$, where*

$$\Pr(A) \triangleq \frac{\mathbb{E}_\sigma\left[\mathbf{1}_{[\sigma \in A]} \det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma)\right]}{\det(\mathbf{I} + \mathbf{L})}, \quad for \quad \sigma = (\sigma_1, \ldots, \sigma_t) \overset{\text{i.i.d.}}{\sim} p, \quad t \sim \text{Poisson}(r).$$

Since the term $\det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma)$ has the same form as the normalization constant of $\text{DPP}(\widetilde{\mathbf{L}}_\sigma)$, an easy calculation shows that the R-DPP can be used as an intermediate distribution in our algorithm without introducing any distortion in the sampling.

Proposition 1 (Der19, Theorem 8)   *For any $\mathbf{L}$, p, r, and $\widetilde{\mathbf{L}}$ defined as in Definition 3,*

$$if \quad \sigma \sim \text{R-DPP}_p^r(\mathbf{L}) \quad and \quad S \sim \text{DPP}(\widetilde{\mathbf{L}}_\sigma) \quad then \quad \{\sigma_i : i \in S\} \sim \text{DPP}(\mathbf{L}).$$

To sample from the R-DPP, our algorithm uses rejection sampling, where the proposal distribution is sampling i.i.d. proportionally to the approximate 1-ridge leverage scores $l_i \approx \tau_i(1)$ (see Definition 1 and the following discussion), computed using any Nyström approximation $\widehat{\mathbf{L}}$ of matrix $\mathbf{L}$. Apart from $\widehat{\mathbf{L}}$, the algorithm also requires an additional parameter q, which controls the size of the intermediate sample. Because of rejection sampling and Proposition 1, the correctness of the algorithm *does not depend* on the choice of $\widehat{\mathbf{L}}$ and q, as demonstrated in the following result. The key part of the proof involves showing that the acceptance probability in Line 6 is bounded by 1. Here, we obtain a considerably tighter bound than the one achieved by [Der19], which allows us to use a much smaller intermediate sample σ (see Section 3) while maintaining the efficiency of rejection sampling.

Theorem 2   *Given a psd matrix $\mathbf{L}$, any one of its Nyström approximations $\widehat{\mathbf{L}}$ and any positive q, DPP-VFX (Algorithm 1) returns $S \sim \text{DPP}(\mathbf{L})$.*

An important implication of Theorem 2 is that even though the choice of $\widehat{\mathbf{L}}$ affects the overall execution of the algorithm, it *does not affect the distribution of the output*. Therefore we can reuse

---

Algorithm 1 DPP-VFX sampling $S \sim \mathrm{DPP}(\mathbf{L})$

Input: $\mathbf{L} \in \mathbb{R}^{n \times n}$, its Nyström approximation $\widehat{\mathbf{L}}$, $q > 0$
1: Compute $l_i = \left[(\mathbf{L} - \widehat{\mathbf{L}}) + \widehat{\mathbf{L}}(\mathbf{I} + \widehat{\mathbf{L}})^{-1} \right]_{ii} \approx \Pr(i \in S)$
2: Initialize $s = \sum_i l_i, \quad z = \mathrm{tr}\left(\widehat{\mathbf{L}}(\mathbf{I} + \widehat{\mathbf{L}})^{-1} \right), \quad \widetilde{\mathbf{L}} = \frac{s}{q} \left[\frac{1}{\sqrt{l_i l_j}} L_{ij} \right]_{ij}$
3: **repeat**
4:     sample $t \sim \mathrm{Poisson}(q\, \mathrm{e}^{s/q})$
5:     sample $\sigma_1, \ldots, \sigma_t \overset{\text{i.i.d.}}{\sim} \left(\frac{l_1}{s}, \ldots, \frac{l_n}{s} \right)$,
6:     sample $Acc \sim \mathrm{Bernoulli}\left(\frac{\mathrm{e}^z \det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma)}{\mathrm{e}^{ts/q} \det(\mathbf{I} + \widehat{\mathbf{L}})} \right)$
7: **until** $Acc = \mathrm{true}$
8: sample $\widetilde{S} \sim \mathrm{DPP}(\widetilde{\mathbf{L}}_\sigma)$
9: **return** $S = \{ \sigma_i : i \in \widetilde{S} \}$

---

the same $\widehat{\mathbf{L}}$ to produce multiple independent samples $S_1, S_2, \ldots \sim \mathrm{DPP}(\mathbf{L})$, which we prove in Appendix A. This is particularly important because, as we will see in Section 3, DPP-VFX can generate successive samples $S_2, S_3, \ldots$ much more cheaply than the first sample S_1.

Lemma 1 *Let $C \subseteq [n]$ be a random set variable with any distribution. Suppose that S_1 and S_2 are returned by two executions of* DPP-VFX*, both using inputs constructed from the same $\mathbf{L}$ and $\widehat{\mathbf{L}} = \mathbf{L}_{\mathcal{I},C} \mathbf{L}_C^+ \mathbf{L}_{C,\mathcal{I}}$. Then S_1 and S_2 are (unconditionally) independent.*

Before proceeding with the proof, we highlight for clarity that the Poisson r.v. t in DPP-VFX (Algorithm 1, Line 4) has a different role than the the Poisson r.v. used in the definition of R-DPPs. As a result, the definition of $\widetilde{\mathbf{L}}$ (Line 2) is not exactly the one that would be suggested by Definition 3. In particular, DPP-VFX uses $q\, \mathrm{e}^{s/q}$ instead of just q as the mean parameter for t. The extra $\mathrm{e}^{s/q}$ factor is *necessary* to correct for the rejection sampling step in Line 6 during the sampling loop, see the proof below.

Proof of Theorem 2 We start by showing that the Bernoulli probability in Line 6 is bounded by 1. Note that this is important not only to sample correctly, but also when we later establish the efficiency of the algorithm. If we showed a weaker upper bound, say $c > 1$, we could always divide the expression by c and retain the correctness, however it would also be c times less likely that $Acc = 1$.

Since $\widehat{\mathbf{L}}$ is a Nyström approximation for some $C \subseteq \mathcal{I}$, it can be written as

$$\widehat{\mathbf{L}} = \mathbf{L}_{\mathcal{I},C} \mathbf{L}_C^+ \mathbf{L}_{C,\mathcal{I}} = \mathbf{B} \mathbf{B}_{C,\mathcal{I}}^\top \mathbf{L}_C^+ \mathbf{B}_{C,\mathcal{I}} \mathbf{B}^\top = \mathbf{B} \mathbf{P} \mathbf{B}^\top,$$

for any $\mathbf{B}$ such that $\mathbf{L} = \mathbf{B} \mathbf{B}^\top$, where $\mathbf{P} \triangleq \mathbf{B}_{C,\mathcal{I}}^\top \mathbf{L}_C^+ \mathbf{B}_{C,\mathcal{I}}$ is a projection, so that $\mathbf{P}^2 = \mathbf{P}$. Let $\widetilde{\mathbf{L}} \triangleq \widetilde{\mathbf{B}} \widetilde{\mathbf{B}}^\top$ where the ith row of $\widetilde{\mathbf{B}}$ is the rescaled ith row of $\mathbf{B}$, i.e. $\widetilde{\mathbf{b}}_i^\top \triangleq \sqrt{\frac{s}{q l_i}} \, \mathbf{b}_i^\top$. Then, we have

$$\frac{\det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma)}{\det(\mathbf{I} + \widehat{\mathbf{L}})} = \frac{\det(\mathbf{I} + \widetilde{\mathbf{B}}_{\sigma,\mathcal{I}}^\top \widetilde{\mathbf{B}}_{\sigma,\mathcal{I}})}{\det(\mathbf{I} + \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})} = \det\left((\mathbf{I} + \widetilde{\mathbf{B}}_{\sigma,\mathcal{I}}^\top \widetilde{\mathbf{B}}_{\sigma,\mathcal{I}})(\mathbf{I} + \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})^{-1} \right)$$

$$= \det\left(\mathbf{I} + (\widetilde{\mathbf{B}}_{\sigma,\mathcal{I}}^\top \widetilde{\mathbf{B}}_{\sigma,\mathcal{I}} - \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})(\mathbf{I} + \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})^{-1} \right)$$

$$\leq \exp\left(\mathrm{tr}\left((\widetilde{\mathbf{B}}_{\sigma,\mathcal{I}}^\top \widetilde{\mathbf{B}}_{\sigma,\mathcal{I}} - \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})(\mathbf{I} + \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})^{-1} \right) \right)$$

$$= \exp\left(\sum_{i=1}^t \frac{s}{q l_{\sigma_i}} \left[\mathbf{B}(\mathbf{I} + \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})^{-1} \mathbf{B}^\top \right]_{\sigma_i \sigma_i} \right) \cdot \mathrm{e}^{-z} = \mathrm{e}^{ts/q} \mathrm{e}^{-z},$$

where the last equality follows because

$$\mathbf{B}(\mathbf{I} + \mathbf{P} \mathbf{B}^\top \mathbf{B} \mathbf{P})^{-1} \mathbf{B}^\top = \mathbf{B}\left(\mathbf{I} - \mathbf{P} \mathbf{B}^\top (\mathbf{I} + \mathbf{B} \mathbf{P} \mathbf{B}^\top)^{-1} \mathbf{B} \mathbf{P} \right) \mathbf{B}^\top$$

$$= \mathbf{L} - \widehat{\mathbf{L}}(\mathbf{I} + \widehat{\mathbf{L}})^{-1} \widehat{\mathbf{L}} = \mathbf{L} - \widehat{\mathbf{L}} + \widehat{\mathbf{L}}(\mathbf{I} + \widehat{\mathbf{L}})^{-1}.$$

Thus, we showed that the expression in Line 6 is valid. Let $\widetilde{\sigma}$ denote the random variable distributed as σ is after exiting the repeat loop. It follows that

$$\Pr(\widetilde{\sigma} \in A) \propto \mathbb{E}_\sigma\left[\mathbf{1}_{[\sigma \in A]} \frac{\mathrm{e}^z \det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma)}{\mathrm{e}^{ts/q} \det(\mathbf{I} + \widehat{\mathbf{L}})}\right] \propto \sum_{t=0}^{\infty} \frac{(q\,\mathrm{e}^{s/q})^t}{\mathrm{e}^{q\,\mathrm{e}^{s/q}} t!} \cdot \mathrm{e}^{-ts/q}\, \mathbb{E}_\sigma\left[\mathbf{1}_{[\sigma \in A]} \det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma) \mid t\right]$$

$$\propto \mathbb{E}_{t'}\left[\mathbb{E}_\sigma\left[\mathbf{1}_{[\sigma \in A]} \det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma) \mid t = t'\right]\right] \quad \text{for } t' \sim \text{Poisson}(q),$$

which shows that $\widetilde{\sigma} \sim \text{R-DPP}_l^q(\mathbf{L})$ for $l = (\frac{l_1}{s}, \dots, \frac{l_n}{s})$. The claim follows from Proposition 1. ∎

3   Conditions for fast sampling

The complexity cost of DPP-VFX can be roughly summarized as follows: we pay a large one-time cost to precompute $\widehat{\mathbf{L}}$ and all its associated quantities, and then we pay a smaller cost in the rejection sampling scheme which must be multiplied by the number of times we repeat the loop until acceptance. We first show that if $\widehat{\mathbf{L}}$ is sufficiently close to $\mathbf{L}$ then we will exit the loop with high probability. We then analyze how accurate the precomputing step needs to be to satisfy this condition. Finally, we bound the overall computational cost, which includes also the final step where we sample S out of the intermediate sample σ using any off-the-shelf exact DPP sampler.

1) Bounding the number of rejections The following result presents the two conditions needed for achieving efficient rejection sampling in DPP-VFX. First, the Nyström approximation needs to be accurate enough, and second, the intermediate sample size (controlled by parameter q) needs to be $\Omega(k^2)$. This is a significant improvement over the guarantee of [Der19] where the intermediate sample size is $\Omega(\text{rank}(\mathbf{L}) \cdot k)$, which is only meaningful for low-rank kernels. The main novelty in this proof comes from showing the following lower bound on the ratio of the determinants of $\mathbf{I} + \mathbf{L}$ and $\mathbf{I} + \widehat{\mathbf{L}}$, when $\widehat{\mathbf{L}}$ is a Nyström approximation: $\det(\mathbf{I} + \mathbf{L})/\det(\mathbf{I} + \widehat{\mathbf{L}}) \geq \mathrm{e}^{k-z}$, where $z \triangleq \text{tr}(\widehat{\mathbf{L}}(\mathbf{I} + \widehat{\mathbf{L}})^{-1})$. Remarkably, this bound exploits the fact that any Nyström approximation of $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$ can be written as $\widehat{\mathbf{L}} = \mathbf{B}\mathbf{P}\mathbf{B}^\top$, where $\mathbf{P}$ is a projection matrix. Note that while our result holds in the worst case, in deployement, the conditions on $\widehat{\mathbf{L}}$ and on q can be considerably relaxed.

Theorem 3 *If the Nyström approximation $\widehat{\mathbf{L}}$ and the intermediate sample size parameter q satisfy*

$$\text{tr}\big(\mathbf{L}(\mathbf{I} + \mathbf{L})^{-1}\mathbf{L} - \widehat{\mathbf{L}}(\mathbf{I} + \mathbf{L})^{-1}\widehat{\mathbf{L}}\big) \leq 1 \qquad \text{and} \qquad q \geq \max\{s^2, s\},$$

then $\Pr(Acc = true) \geq \mathrm{e}^{-2}$. Therefore, with probability $1 - \delta$, Algorithm 1 exits the rejection sampling loop after at most $\mathcal{O}(\log \delta^{-1})$ iterations and, after precomputing all of the inputs, the time complexity of the rejection sampling loop is $\mathcal{O}\big(k^6 \log \delta^{-1} + \log^4 \delta^{-1}\big)$.

Proof We first bound the number of rejections, and then discuss the complexity of each loop iteration. Let σ be distributed as in Line 5. The probability of exiting the repeat loop at each iteration is

$$P \triangleq \mathbb{E}_\sigma\left[\frac{\mathrm{e}^z \det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma)}{\mathrm{e}^{ts/q} \det(\mathbf{I} + \widehat{\mathbf{L}})}\right] = \sum_{t=0}^{\infty} \frac{(q\,\mathrm{e}^{s/q})^t}{\mathrm{e}^{q\,\mathrm{e}^{s/q}} t!} \cdot \frac{\mathrm{e}^{z-ts/q}}{\det(\mathbf{I} + \widehat{\mathbf{L}})} \mathbb{E}_\sigma\big[\det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma) \mid t\big]$$

$$= \frac{\mathrm{e}^{q-q\,\mathrm{e}^{s/q}+z}}{\det(\mathbf{I} + \widehat{\mathbf{L}})} \sum_{t=0}^{\infty} \frac{q^t}{\mathrm{e}^q t!} \mathbb{E}_\sigma\big[\det(\mathbf{I} + \widetilde{\mathbf{L}}_\sigma) \mid t\big] = \mathrm{e}^{q-q\,\mathrm{e}^{s/q}+z} \frac{\det(\mathbf{I} + \mathbf{L})}{\det(\mathbf{I} + \widehat{\mathbf{L}})},$$

where the last equality follows because the infinite series computes the normalization constant of R-DPP$_l^q(\mathbf{L})$ given in Definition 3. If $s \geq 1$ then $q = s^2$ and the inequality $\mathrm{e}^x \leq 1 + x + x^2$ for $x \in [0, 1]$ implies that $q - q\,\mathrm{e}^{s/q} + z = s^2(1 - \mathrm{e}^{1/s} + 1/s) + z - s \geq -1 + z - s$. On the other hand, if $q = s \in [0, 1]$, then $q - q\mathrm{e}^{s/q} + z = (1 - \mathrm{e})s + z \geq -1 + z - s$. We proceed to lower bound the determinantal ratio. Here, let $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$ and $\widehat{\mathbf{L}} = \mathbf{B}\mathbf{P}\mathbf{B}^\top$, where $\mathbf{P}$ is a projection matrix. Then,

$$\frac{\det(\mathbf{I} + \mathbf{L})}{\det(\mathbf{I} + \widehat{\mathbf{L}})} = \frac{\det(\mathbf{I} + \mathbf{B}^\top\mathbf{B})}{\det(\mathbf{I} + \mathbf{P}\mathbf{B}^\top\mathbf{B}\mathbf{P})} = \det\big(\mathbf{I} - (\mathbf{B}^\top\mathbf{B} - \mathbf{P}\mathbf{B}^\top\mathbf{B}\mathbf{P})(\mathbf{I} + \mathbf{B}^\top\mathbf{B})^{-1}\big)^{-1}$$

$$\geq \exp\Big(\text{tr}\big((\mathbf{B}^\top\mathbf{B} - \mathbf{P}\mathbf{B}^\top\mathbf{B}\mathbf{P})(\mathbf{I} + \mathbf{B}^\top\mathbf{B})^{-1}\big)\Big)$$

$$= \exp\Big(\text{tr}\big(\mathbf{B}(\mathbf{I} + \mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\big) - \text{tr}\big(\mathbf{B}\mathbf{P}(\mathbf{I} + \mathbf{B}^\top\mathbf{B})^{-1}\mathbf{P}\mathbf{B}^\top\big)\Big),$$

where we can reformulate $k = \mathrm{tr}\big(\mathbf{B}(\mathbf{I} + \mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\big)$ and the other elements in terms of $\widehat{\mathbf{L}}$ and $\mathbf{L}$ as

$$\exp\Big(k - \mathrm{tr}\big(\mathbf{B}\mathbf{P}(\mathbf{I} + \mathbf{B}^\top\mathbf{B} - \mathbf{B}^\top\mathbf{B})(\mathbf{I} + \mathbf{B}^\top\mathbf{B})^{-1}\mathbf{P}\mathbf{B}^\top\big)\Big)$$
$$= \exp\Big(k - \mathrm{tr}(\widehat{\mathbf{L}}) + \mathrm{tr}\big(\mathbf{B}\mathbf{P}\mathbf{B}^\top\mathbf{B}(\mathbf{I} + \mathbf{B}^\top\mathbf{B})^{-1}\mathbf{P}\mathbf{B}^\top\big)\Big)$$
$$= \exp\Big(k - \mathrm{tr}(\widehat{\mathbf{L}}) + \mathrm{tr}\big(\mathbf{B}\mathbf{P}\mathbf{B}^\top(\mathbf{I} + \mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B}\mathbf{P}\mathbf{B}^\top\big)\Big)$$
$$= \exp\Big(k - \mathrm{tr}(\widehat{\mathbf{L}}) + \mathrm{tr}\big(\widehat{\mathbf{L}}(\mathbf{I} + \mathbf{L})^{-1}\widehat{\mathbf{L}}\big)\Big).$$

Putting all together,

$$\mathrm{e}^{q - q\,\mathrm{e}^{s/q} + z}\,\frac{\det(\mathbf{I} + \mathbf{L})}{\det(\mathbf{I} + \widehat{\mathbf{L}})} \geq \exp\Big(-1 + z - s + k - \mathrm{tr}(\widehat{\mathbf{L}}) + \mathrm{tr}\big(\widehat{\mathbf{L}}(\mathbf{I} + \mathbf{L})^{-1}\widehat{\mathbf{L}}\big)\Big),$$

and using the definitions $k = \mathrm{tr}\big(\mathbf{L}(\mathbf{I}+\mathbf{L})^{-1}\big)$, $s = \mathrm{tr}(\mathbf{L} - \widehat{\mathbf{L}} + \widehat{\mathbf{L}}(\widehat{\mathbf{L}}+\mathbf{I})^{-1})$, and $z = \mathrm{tr}(\widehat{\mathbf{L}}(\widehat{\mathbf{L}}+\mathbf{I})^{-1})$ on the middle term $z - s + k - \mathrm{tr}(\widehat{\mathbf{L}})$ we have

$$\mathrm{tr}\big(\widehat{\mathbf{L}}(\widehat{\mathbf{L}} + \mathbf{I})^{-1} + \mathbf{L}(\mathbf{I} + \mathbf{L})^{-1} - \mathbf{L} + \widehat{\mathbf{L}} - \widehat{\mathbf{L}}(\widehat{\mathbf{L}} + \mathbf{I})^{-1} - \widehat{\mathbf{L}}\big)$$
$$= \mathrm{tr}\big(\mathbf{L}(\mathbf{I} + \mathbf{L})^{-1} - \mathbf{L}\big) = \mathrm{tr}\big(\mathbf{L}(\mathbf{I} + \mathbf{L})^{-1} - \mathbf{L}(\mathbf{I} + \mathbf{L})^{-1}(\mathbf{I} + \mathbf{L})\big) = -\mathrm{tr}\big(\mathbf{L}(\mathbf{I} + \mathbf{L})^{-1}\mathbf{L}\big),$$

and therefore,

$$\mathrm{e}^{q - q\,\mathrm{e}^{s/q} + z}\,\frac{\det(\mathbf{I} + \mathbf{L})}{\det(\mathbf{I} + \widehat{\mathbf{L}})} \geq \exp\Big(-1 + \mathrm{tr}\big(\widehat{\mathbf{L}}(\mathbf{I} + \mathbf{L})^{-1}\widehat{\mathbf{L}}\big) - \mathrm{tr}\big(\mathbf{L}(\mathbf{I} + \mathbf{L})^{-1}\mathbf{L}\big)\Big),$$

and we obtain our condition. Thus, with probability $1 - \delta$, the main loop will be repeated $\mathcal{O}(\log \delta^{-1})$ times. We now quantify the cost of a single loop iteration. First note that since the number of samples t_i drawn from l in the ith iteration of the loop is a Poisson distributed random variable, a standard Poisson tail bound implies that with probability $1 - \delta$, all iterations will satisfy $t_i = \mathcal{O}(k^2 + \log \delta^{-1})$. Then, drawing the Poisson r.v. t_i (Line 4) requires $\mathcal{O}(k^2 + \log \delta^{-1})$ time. Drawing σ from the multinomial $\big(\frac{l_1}{s}, \dots, \frac{l_n}{s}\big)$ can be done by first sorting the l_i, which is a one-time $\mathcal{O}(n \log(n))$ cost, and then using specialized samplers which require $\mathcal{O}(1)$ time [BP12]. Therefore, the dominant cost is computing the determinant of the matrix $\mathbf{I} + \widetilde{\mathbf{L}}_\sigma$ in $\mathcal{O}(t_i^3)$, and the result follows. ∎

2) Bounding the precompute cost All that is left is to control the cost of the precomputation phase. We separate the analysis into two steps: how much it costs to choose $\widehat{\mathbf{L}}$ to satisfy the assumption of Theorem 3, and how much it costs to compute everything else given $\widehat{\mathbf{L}}$, see Appendix A.

Lemma 2 *Let $\widehat{\mathbf{L}}$ be constructed by sampling $m = \mathcal{O}(k^3 \log \frac{n}{\delta})$ columns proportionally to their RLS. Then, with probability $1 - \delta$, $\widehat{\mathbf{L}}$ satisfies the assumption of Theorem 3.*

There exist many algorithms to sample columns proportionally to their RLSs. For example, we can take the BLESS [RCCR18] with the following guarantee.

Proposition 2 (RCCR18, Theorem 1) *There exists an algorithm that with probability $1 - \delta$ samples m columns proportionally to their RLSs in $\mathcal{O}(nk^2 \log^2 \frac{n}{\delta} + k^3 \log^4 \frac{n}{\delta} + m)$ time.*

We can now compute the remaining preprocessing costs, given a Nyström approximation $\widehat{\mathbf{L}}$.

Lemma 3 *Given $\widehat{\mathbf{L}}$ with rank m, we can compute l_i, s, z, and $\widetilde{\mathbf{L}}$ in $\mathcal{O}(nm^2 + m^3)$ time.*

3) Bounding the overall cost We can now fully characterize the computational cost.

Theorem 1 (restated for DPPs only) *For a psd $n \times n$ matrix $\mathbf{L}$, let S_1, S_2 be i.i.d. random sets from $\mathrm{DPP}(\mathbf{L})$. Denote with $\widehat{\mathbf{L}}$ a Nyström approximation of $\mathbf{L}$ obtained by sampling $m = \mathcal{O}(k^3 \log(n/\delta))$ of its columns proportionally to their RLSs. If $q \geq \max\{s^2, s\}$, then w.p. $1 - \delta$, DPP-VFX returns*

a) *subset S_1 in:* $\mathcal{O}(nk^6 \log^2 \frac{n}{\delta} + k^9 \log^3 \frac{n}{\delta} + k^3 \log^4 \frac{n}{\delta})$ *time,*

b) *then, S_2 in:* $\mathcal{O}\big(k^6 \log \frac{1}{\delta} + \log^4 \frac{1}{\delta}\big)$ *time.*

Discussion The above result follows from the bounds on the precompute costs, on the number of iteratios, on the iterations cost, and on the fact that the final DPP sampling step $\widetilde{S} \sim \mathrm{DPP}(\widetilde{\mathbf{L}}_\sigma)$ requires $\mathcal{O}(t^3) \leq \mathcal{O}((k^2 + \log \delta^{-1})^3)$ time. Note however that due to the nature of rejection sampling, as long as we exit the loop, i.e., we accept the sample, the output of DPP-VFX is guaranteed to follow the DPP distribution for any value of m and q. In Theorem 1 we set $m = (\tilde{k}^3 \log \frac{n}{\delta})$ and $q \geq \max\{s^2, s\}$ to satisfy Theorem 3 and guarantee a constant acceptance probability in the rejection sampling loop, but this might not be necessary or even desirable in practice. Experimentally, much smaller values of m, starting from $m = \Omega(k \log \frac{n}{\delta})$ seem to be sufficient to accept the sample, while at the same time a smaller m greatly reduces the preprocessing costs. In general, we recommend to separate DPP-VFX in three phases. First, compute an accurate estimate of the RLS using off-the-shelf algorithms in $\mathcal{O}(nk^2 \log^2 \frac{n}{\delta} + k^3 \log^4 \frac{n}{\delta})$ time. Then, sample a small number m of columns to construct an explorative $\widehat{\mathbf{L}}$, and try to run DPP-VFX. If the rejection sampling loop does not terminate sufficiently fast, then we can reuse the RLS estimates to compute a more accurate $\widehat{\mathbf{L}}$ for a larger m. Using a simple doubling schedule for m, this strategy quickly reaches a regime where DPP-VFX is w.h.p. guaranteed to accept, resulting in faster sampling.

4   Reduction from DPPs to k-DPPs

We next show that with a simple extra rejection sampling step we can efficiently transform *any* exact DPP sampler into an exact k-DPP sampler.

A common heuristic to sample S from a k-DPP is to first sample S from a DPP, and then reject the sample if the size of S is not exactly k. As we show in this section, the success probability of this procedure can be improved by appropriately rescaling $\mathbf{L}$ by a constant factor α,

$$\text{sample} \quad S_\alpha \sim \mathrm{DPP}(\alpha \mathbf{L}), \quad \text{accept if } |S_\alpha| = k.$$

Note that rescaling the DPP by a constant α as above only changes the expected size of the set S_α, and not its distribution. Therefore, if we accept only sets with size k, we will be sampling exactly from our k-DPP. Moreover, if $k_\alpha = \mathbb{E}[|S_\alpha|]$ is close to k, the success probability will improve. With a slight abuse of notation, in the context of k-DPPs we will indicate with k the desired size of S_α, i.e., the final output size at acceptance, and with $k_\alpha = \mathbb{E}[|S_\alpha|]$ the expected size of the scaled DPP.

While the above rejection sampling heuristic is widespread, until now there has been no proof that this heuristic can provably succeed in few rejections We solve this open question with two new results. First, we show that for an appropriate rescaling $\alpha^\star$ we only reject $S_{\alpha^\star}$ roughly $O(\sqrt{k})$ times. Then, we show how to find such an $\alpha^\star$ with a $\widetilde{\mathcal{O}}(n \cdot \mathrm{poly}(k))$ time preprocessing step.

Theorem 4 *There exists constant $C > 0$ such that for any rank n psd matrix $\mathbf{L}$ and $k \in [n]$, there is $\alpha^\star > 0$ with the following property: if we sample $S_{\alpha^\star} \sim \mathrm{DPP}(\alpha^\star \mathbf{L})$, then $\Pr(|S_{\alpha^\star}| = k) \geq \frac{1}{C\sqrt{k}}$.*

The proof in Appendix B. relies on a known Chernoff bound for the sample size $|S_\alpha|$ of a DPP. When applied naïvely, the inequality does not offer a lower bound on the probability of any single sample size. However we show that the probability mass is concentrated on $O(\sqrt{k_\alpha})$ sizes. This leads to a lower bound on the sample size with the largest probability, i.e., the *mode* of the distribution. Then, it remains to observe that for any $k \in [n]$ we can always find $\alpha^\star$ for which k is the mode of $|S_{\alpha^\star}|$. We conclude that given $\alpha^\star$, the rejection sampling scheme described above transforms any $\mathrm{poly}(k)$ time DPP sampler into a $\mathrm{poly}(k)$ time k-DPP sampler. It remains to efficiently find $\alpha^\star$, which once again relies on using a Nyström approximation of $\mathbf{L}$.

Lemma 4 *If $k \geq 1$, then there is an algorithm that finds $\alpha^\star$ in $\widetilde{\mathcal{O}}(n \cdot \mathrm{poly}(k))$ time.*

While the existence proof of $\alpha^\star$ is general and based on simple unimodality, this characterization is not sufficient to control the mode of the DPP when $\alpha^\star$ is perturbed, as it happens during an approximate optimization of α. However, for DPPs the mode can be expressed as a Poisson binomial random variable [H$^+$56] based on the spectrum of $\mathbf{L}$, which can be controlled [D$^+$64] using the approximate spectrum of $\widehat{\mathbf{L}}$. While our analysis shows that in the worst case a much more accurate $\widehat{\mathbf{L}}$ is necessary compared to Theorem 3, in practice the same $m = \Omega(k \log \frac{n}{\delta})$ seems to suffice.
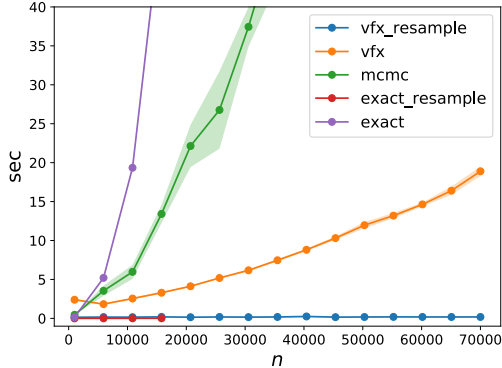
Figure 1: *First-sample* cost for DPP-VFX against other DPP samplers (n is data size).
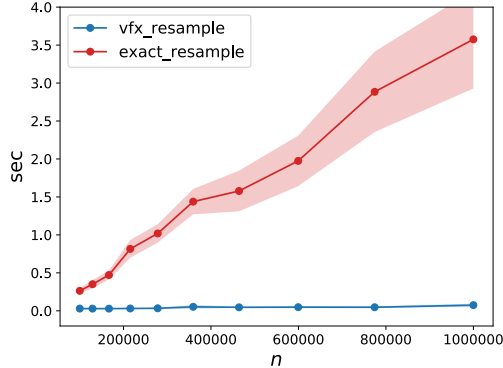
Figure 2: *Resampling* cost for DPP-VFX compared to the exact sampler of [HKP+06].

5 Experiments

In this section, we experimentally evaluate[3] the performance of DPP-VFX and compare it to exact sampling [HKP+06] and MCMC-based approaches [AGR16]. In particular, we are only interested in evaluating computational performance, i.e., how DPP-VFX and baselines scale with the size n of the matrix $\mathbf{L}$. Note that an inexact DPP sampler, e.g., MCMC samplers or previous approximate rejection samplers [Der19], also had another metric to validate, i.e., they had to empirically show that their samples were close enough to a DPP distribution. However, Section 2 proves that DPP-VFX's output is distributed *exactly* according to the DPP and therefore it is strictly equivalent to any other exact DPP sampler, e.g., the Gram-Schmidt [Gil14] or dual [KT12] samplers implemented in DPPy.

We use the infinite MNIST digits dataset (i.e., MNIST8M [LCB07]) as input data, where n varies up to 10^6 and $d = 784$. All algorithms, including DPP-VFX, are implemented in `python` as part of the DPPy library [GBV19]. All experiments are carried out on a 24-core CPU and fully take advantage of potential parallelization. For each experiment we report the mean and a 95% confidence interval over 10 runs of each algorithm's runtime.

This first experiment compares the time required to generate a first sample. We consider subsets of MNIST8M that go from $n = 10^3$ to $n = 7 \cdot 10^4$, i.e., the original MNIST dataset, and use an RBF kernel with $\sigma = \sqrt{3d}$ to construct $\mathbf{L}$. For the Nyström approximation we set $m = 10d_{\text{eff}}(1) \approx 10k$. While this is much lower than the $\mathcal{O}(k^3)$ value suggested by the theory, as we will see it is already accurate enough to result in drastic runtime improvements over exact and MCMC. Following the strategy of Section 4, for each algorithm we control[4] the size of the output set by rescaling the input matrix $\mathbf{L}$ by a constant $\alpha^\star$ such that $\mathbb{E}[|S_{\alpha^\star}|] = 10$. Results are reported in Figure 1.

Exact sampling is performed using eigendecomposition and DPPy's default Gram-Schmidt [Gil14] sampler. It is clearly cubic in n, and we could not push it beyond $n = 1.5 \cdot 10^4$. For MCMC, we enforce mixing by runnning the chain for nk steps, the minimum recommended by [AGR16]. However, for $n = 7 \cdot 10^5$ the MCMC runtime is 160s and exceeds the plot's limit, while DPP-VFX completes in 16s, an order of magnitude faster. Moreover, DPP-VFX rarely rejects more than 10 times, and the mode of the rejections up to $n = 7 \cdot 10^5$ is 1, i.e., we mostly accept at the first iteration.

Since in Figure 1, the resampling time of both exact and DPP-VFX are negligible, we investigate the cost of a second sample, i.e., of resampling, on a much larger subset of MNIST8M up to $n = 10^6$, normalized to have maximum row norm equal to 1. In this case it is not possible to perform an eigendecomposition of $\mathbf{L}$, so we replace RBFs with a linear kernel, for which the input $\mathbf{X}$ represents a factorization $\mathbf{L} = \mathbf{X}\mathbf{X}^\top$ and we use an exact dual sampler on $\mathbf{X}$ [KT12]. However, as Figure 2 shows, even with a special algorithm for this simpler setting the resampling process still scales with n. On the other hand, DPP-VFX's complexity (after preprocessing) is light years better as it scales only with k and remains constant regardless of n.

---

[3]The code used for these experiments is available at `https://github.com/LCSL/dpp-vfx`.

[4]For simplicity we do not perform the full k-DPP rejection step, but only adjust the expected size of the set.

Acknowledgements

References

[AGR16]   Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 103–115, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[AKFT13]   Raja Hafiz Affandi, Alex Kulesza, Emily Fox, and Ben Taskar. Nystrom approximation for large-scale determinantal processes. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 85–98, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

[Ald90]   David J Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM Journal on Discrete Mathematics*, 3(4):450–465, 1990.

[AM15]   Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 775–783, Montreal, Canada, December 2015.

[BLMV17]   Rémi Bardenet, Frédéric Lavancier, Xavier Mary, and Aurélien Vasseur. On a few statistical applications of determinantal point processes. *ESAIM: Procs*, 60:180–202, 2017.

[BP12]   Karl Bringmann and Konstantinos Panagiotou. Efficient sampling methods for discrete distributions. In *International Colloquium on Automata, Languages, and Programming*, pages 133–144. Springer, 2012.

[Bra14]   Petter Branden. Unimodality, log-concavity, real-rootedness and beyond. *Handbook of Enumerative Combinatorics*, 10 2014.

[Bro89]   A Broder. Generating random spanning trees. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 442–447. IEEE, 1989.

[Bru18]   Victor-Emmanuel Brunel. Learning signed determinantal point processes through the principal minor assignment problem. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7365–7374. Curran Associates, Inc., 2018.

[BRVDW19]   David Burt, Carl Edward Rasmussen, and Mark Van Der Wilk. Rates of convergence for sparse variational Gaussian process regression. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 862–871, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[CCL+19]   Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Gaussian process optimization with adaptive sketching: Scalable and no regret. In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 533–557, Phoenix, USA, 25–28 Jun 2019. PMLR.

[CKS⁺18] Elisa Celis, Vijay Keswani, Damian Straszak, Amit Deshpande, Tarun Kathuria, and Nisheeth Vishnoi. Fair and diverse DPP-based data summarization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 716–725, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[CLV17] Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Distributed adaptive sampling for kernel matrix approximation. In *AISTATS*, 2017.

[CZZ18] Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5622–5633. Curran Associates, Inc., 2018.

[D⁺64] John N Darroch et al. On the distribution of the number of successes in independent trials. *The Annals of Mathematical Statistics*, 35(3):1317–1321, 1964.

[Der19] Michał Dereziński. Fast determinantal point processes via distortion-free intermediate sampling. In *Proceedings of the 32nd Conference on Learning Theory*, 2019.

[DS91] Persi Diaconis and Daniel Stroock. Geometric Bounds for Eigenvalues of Markov Chains. *The Annals of Applied Probability*, 1991.

[DW17] Michał Dereziński and Manfred K. Warmuth. Unbiased estimates for linear regression via volume sampling. In *Advances in Neural Information Processing Systems 30*, pages 3087–3096, Long Beach, CA, USA, December 2017.

[DWH18] Michał Dereziński, Manfred K. Warmuth, and Daniel Hsu. Leveraged volume sampling for linear regression. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2510–2519. Curran Associates, Inc., 2018.

[DWH19] Michał Dereziński, Manfred K. Warmuth, and Daniel Hsu. Correcting the bias in least squares regression with volume-rescaled sampling. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

[EVCM16] Akram Erraqabi, Michal Valko, Alexandra Carpentier, and Odalric-Ambrym Maillard. Pliable rejection sampling. In *International Conference on Machine Learning*, 2016.

[GBV17] Guillaume Gautier, Rémi Bardenet, and Michal Valko. Zonotope hit-and-run for efficient sampling from projection DPPs. In *International Conference on Machine Learning*, 2017.

[GBV19] Guillaume Gautier, Rémi Bardenet, and Michal Valko. DPPy: Sampling determinantal point processes with Python. *Journal of Machine Learning Research - Machine Learning Open Source Software (JMLR-MLOSS)*, 2019.

[Gil14] Jennifer Ann Gillenwater. Approximate inference for determinantal point processes. 2014.

[GKT12] Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Discovering diverse and salient threads in document collections. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 710–720, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[GKVM18] Jennifer A Gillenwater, Alex Kulesza, Sergei Vassilvitskii, and Zelda E. Mariet. Maximizing induced cardinality under a determinantal point process. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6911–6920. Curran Associates, Inc., 2018.

[Gue83] A Guenoche. Random spanning tree. *Journal of Algorithms*, 4(3):214 – 220, 1983.

[H+56] Wassily Hoeffding et al. On the distribution of the number of successes in independent trials. *The Annals of Mathematical Statistics*, 27(3):713–721, 1956.

[HKP+06] J Ben Hough, Manjunath Krishnapur, Yuval Peres, Bálint Virág, et al. Determinantal processes and independence. *Probability surveys*, 3:206–229, 2006.

[Kan13] Byungkon Kang. Fast determinantal point process sampling with application to clustering. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 2319–2327, USA, 2013.

[KT11] Alex Kulesza and Ben Taskar. k-DPPs: Fixed-Size Determinantal Point Processes. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1193–1200, Bellevue, WA, USA, June 2011.

[KT12] Alex Kulesza and Ben Taskar. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA, 2012.

[LCB07] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

[LGD18] Claire Launay, Bruno Galerne, and Agnès Desolneux. Exact Sampling of Determinantal Point Processes without Eigendecomposition. *arXiv e-prints*, page arXiv:1802.08429, Feb 2018.

[LJS16a] Chengtao Li, Stefanie Jegelka, and Suvrit Sra. Efficient sampling for k-determinantal point processes. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1328–1337, Cadiz, Spain, 09–11 May 2016. PMLR.

[LJS16b] Chengtao Li, Stefanie Jegelka, and Suvrit Sra. Fast mixing markov chains for strongly rayleigh measures, dpps, and constrained sampling. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4195–4203, USA, 2016. Curran Associates Inc.

[Mac75] Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

[MS16] Zelda E. Mariet and Suvrit Sra. Kronecker determinantal point processes. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2694–2702. Curran Associates, Inc., 2016.

[MU49] Nicholas Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[Pou19] Jack Poulson. High-performance sampling of generic determinantal point processes. ArXive:1905.00165v1, 2019.

[PP14] Robin Pemantle and Yuval Peres. Concentration of lipschitz functionals of determinantal and other strong rayleigh measures. *Combinatorics, Probability and Computing*, 23(1):140–160, 2014.

[PW98] J G Propp and D B Wilson. How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms*, 27(2):170–217, 1998.

[RCCR18] Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco. On fast leverage score sampling and optimal learning. In *Advances in Neural Information Processing Systems 31*, pages 5672–5682. 2018.

[RK15] P Rebeschini and A Karbasi. Fast mixing for discrete point processes. In *Conference on Learning Theory*, pages 1480–1500, 2015.

[ZKM17] Cheng Zhang, Hedvig Kjellström, and Stephan Mandt. Determinantal point processes for mini-batch diversification. In *33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, 11 August 2017 through 15 August 2017*. AUAI Press Corvallis, 2017.

[ZÖMS19] Cheng Zhang, Cengiz Öztireli, Stephan Mandt, and Giampiero Salvi. Active mini-batch sampling using repulsive point processes. In *AAAI*, 2019.