Exact sampling of determinantal point processes with sublinear time preprocessing

Michał Dereziński [*1] **Daniele Calandriello** [*2] **Michal Valko** [3]

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 $n \times n$ p.s.d. 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 widely used in machine learning to induce diversity in subset selection. In practice, we often wish to sample multiple subsets S with small expected size $k = \mathbb{E}[|S|] \ll n$ 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). To that end, we propose a new algorithm which, given access to $\mathbf{L}$, samples exactly from a DPP while satisfying the following two properties: (1) its preprocessing cost is $n \cdot \mathrm{poly}(k)$ (sublinear in the size of $\mathbf{L}$) and (2) its sampling cost is $\mathrm{poly}(k)$ (independent of the size of $\mathbf{L}$). Prior to this work, state-of-the-art exact samplers required $O(n^3)$ preprocessing time and sampling time linear in n or dependent on $\mathbf{L}$'s spectrum.

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

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

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 accross many scientific domains (Macchi, 1975; Bardenet et al., 2017; Guenoche, 1983), while also being widely used as a tool in machine learning and recommender systems (Kulesza & Taskar, 2012) for inducing diversity in subset selection and as a variance reduction technique. In this context, we often wish to efficiently produce many DPP samples of small expected size $k := \mathbb{E}[|S|]$ given a large matrix $\mathbf{L}$. Sometimes the distribution is restricted to subsets of fixed size $|S| = k \ll n$, denoted k-$\mathrm{DPP}(\mathbf{L})$. Hough et al. (2006) gave an algorithm for drawing exact samples from $\mathrm{DPP}(\mathbf{L})$, later adapted to k-$\mathrm{DPP}(\mathbf{L})$ by Kulesza & Taskar (2011), 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 $O(n^3)$. 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 have been considered for both DPPs and k-DPPs to reduce the computational cost of preprocessing and/or sampling, including many approximate and heuristic approaches. In this paper we present an algorithm which samples *exactly* from a 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 $\mathrm{DPP}(\mathbf{L})$. Then, there is an algorithm which, given access to $\mathbf{L}$, returns:*

a) the first subset, S_1, in: $n \cdot \mathrm{poly}(k)\,\mathrm{polylog}(n)$ *time,*

b) each subsequent S_i in: $\mathrm{poly}(k)$ *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 focus on the methods that provide strong theoretical accuracy guarantees (see discussion in Section 1). As seen from the table, our algorithm is the first exact sampler to achieve sublinear overall runtime. Only the approximate MCMC sampler of Anari et al. (2016) matches our $n \cdot \mathrm{poly}(k)$ complexity (albeit for a k-DPP instead of a DPP), but for this method every next sample is equally expensive,

[*]Equal contribution [1]Department of Statistics, University of California, Berkeley [2]LCSL Istituto Italiano di Tecnologia, Italy [3]DeepMind Paris. Correspondence to: Michał Dereziński <mderezin@berkeley.edu>, Daniele Calandriello <daniele.calandriello@iit.it>.

Proceedings of the Workshop on Negative Dependence in Machine Learning at the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Full version appears as https://arxiv.org/abs/1905.13476.

	Exact	Variant	First sample	Subsequent samples
Hough et al. (2006)	✓	DPP	n^3	nk^2
Kulesza & Taskar (2011)	✓	k-DPP	n^3	nk^2
Anari et al. (2016)	✗	k-DPP	$n \cdot \mathrm{poly}(k)$	$n \cdot \mathrm{poly}(k)$
Li et al. (2016b)	✗	DPP	$n^2 \cdot \mathrm{poly}(k)$	$n^2 \cdot \mathrm{poly}(k)$
Launay et al. (2018)	✓	DPP	n^3	$\mathrm{poly}(k \cdot (1 + \lVert\mathbf{L}\rVert))$
Dereziński (2019)	✓	DPP	n^3	$\mathrm{poly}(\mathrm{rank}(\mathbf{L}))$
DPP-VFX **(this paper)**	✓	DPP	$n \cdot \mathrm{poly}(k)$	$\mathrm{poly}(k)$

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 $k \leq \mathrm{rank}(\mathbf{L}) \leq n$. We omit log-terms for clarity.

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 $\mathrm{poly}(k)$ sampling time of this paper. Our approach draws on the ideas of Dereziński (2019), particularly the notion of regularized determinantal point processes (R-DPP, discussed in Section 2). However, despite similarities, their algorithm does not come close to the time complexity of our approach except for some corner cases when the ensemble matrix $\mathbf{L}$ has extremely low rank.

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 Hough et al. (2006) and even most recent exact refinements (Launay et al., 2018; Dereziński, 2019; Poulson, 2019), 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 (Aldous, 1990; Broder, 1989; Propp & Wilson, 1998), which leaves the possibility of a more generic fast sampler open. Since the sampling from DPPs has several practical large scale machine learning applications (Kulesza & Taskar, 2012), 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 (Kulesza & Taskar, 2011; Gillenwater et al., 2012; Affandi et al., 2013; Li et al., 2016a). For example, Affandi et al. (2013) provides approximate guarantee for the probability of any subset being sampled as a function of eigengaps of the L-kernel. Next, Li et al. (2016a) construct *coresets* approximating a given k-DPP and then use them for sampling. In their Section 4.1, Li et al. (2016a) 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 (Metropolis & Ulam, 1949) techniques (Kang, 2013; Rebeschini & Karbasi, 2015; Anari et al., 2016; Li et al., 2016b; Gautier et al., 2017). There are known polynomial bounds on the mixing rates (Diaconis & Stroock, 1991) of MCMC chains with arbitrary DPPs as their limiting measure. In particular, Anari et al. (2016) showed them for cardinality-constrained DPPs and Li et al. (2016b) 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 (also called *perfect sampling*) according to a DPP. 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 do not scale with the dimension and are adversely affected by the spikes in the distribution (Erraqabi et al., 2016), 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 via Nyström approximation

Our method is based on a technique developed recently by Dereziński et al. (2018; 2019), and extended by Dereziński (2019). 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 Dereziński (2019), 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 $\mathbf{L}$. Overall, the approach is described in the diagram below, where $|S| \le t \ll n$:

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

The DPP algorithm proposed by Dereziński (2019) follows the same diagram, however it requires that the size of the intermediate sample σ be $\Omega(\mathrm{rank}(\mathbf{L})\cdot k)$. Therefore, this method provides improvement only when $\mathbf{L}$ can be decomposed as $\mathbf{X}\mathbf{X}^\top$ for some $n\times r$ matrix $\mathbf{X}$, with $k \le r \ll n$. Even in this special case, the sampling time of their algorithm can only match ours when $r = O(k)$. 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 Dereziński (2019) are vacuous in that sampling would take $\Omega(n^3)$. We propose a different R-DPP implementation (see Algorithm 1) where the expected size of σ is $O(k^2)$. To make the algorithm efficient, we draw on the connections between determinantal point processes, ridge leverage scores and Nyström approximations.

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 is defined as $d_{\text{eff}}(\lambda) = \sum_i \tau_i(\lambda)$.*

An important connection between RLSs and DPPs is that when $S \sim \mathrm{DPP}(\mathbf{L})$ 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:

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

Intuitively, if the marginal probability of i is high then this index should also likely make it into the intermediate sample σ. This suggests that i.i.d. sampling 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 i.i.d. distribution can be easily corrected by rejection sampling to become the R-DPP that we need. Computing ridge leverage scores exactly costs $O(n^3)$, so instead we compute their approximations (denoted l_i in Algorithm 1) by first constructing a Nyström approximation of $\mathbf{L}$.

Definition 2 *Let $\mathbf{L}$ be a psd matrix and C a subset with size $m = |C|$. Then the Nyström approximation of $\mathbf{L}$ based on C is the $n\times n$ matrix $\widehat{\mathbf{L}} := (\mathbf{L}_{C,\mathcal{I}})^\top\mathbf{L}_C^+\mathbf{L}_{C,\mathcal{I}}$.*

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.

Theorem 2 *Given a psd matrix $\mathbf{L}$, any of its Nyström approximations $\widehat{\mathbf{L}}$, and $q > 0$, Alg. 1 returns $S \sim \mathrm{DPP}(\mathbf{L})$.*

The key part of the proof involves showing that the acceptance probability in line 4 is bounded by 1. Here, we obtain a considerably tighter bound than the one achieved by Dereziński (2019), which allows us to use a much smaller intermediate sample σ. The bound implies that σ is distributed according to an R-DPP, which is sufficient to guarantee the exactness of the algorithm.

Remarkably, Theorem 2 requires no assumptions on the subset C that is used to produce the Nyström approximation $\widehat{\mathbf{L}}$. However, the choice of C greatly influences the computational cost of the sampling through the rank of $\widehat{\mathbf{L}}$ and the probability of rejecting a sample. Since $\mathrm{rank}(\widehat{\mathbf{L}}) = |C|$, 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 the cost of handling $\widehat{\mathbf{L}}$ by too much. 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 (Alaoui & Mahoney, 2015; Calandriello et al., 2017; Rudi et al., 2018).

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

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

3. Conditions for fast sampling

The complexity cost of Algorithm 1 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 the approximate $\widehat{\mathbf{L}}$ is sufficiently close to $\widehat{\mathbf{L}}$, then we will exit the loop with high probability. We then analyze how accurate the precomputing step needs to be to satisfy this condition.

Theorem 3 *If the Nyström approximation $\widehat{\mathbf{L}}$ satisfies:*

$$\operatorname{tr}\left(\mathbf{L}(\mathbf{I}+\mathbf{L})^{-1}\mathbf{L} - \widehat{\mathbf{L}}(\mathbf{I}+\mathbf{L})^{-1}\widehat{\mathbf{L}}\right) \leq 1,$$

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

The above theorem essentially states that as long as intermediate sample size (controlled by q) is sufficiently large and the approximation error of $\widehat{\mathbf{L}}$ is sufficiently small, then the acceptance probability in line 4 is constant. Therefore, the dominant operations in Algorithm 1, i.e., computing the determinant of $\mathbf{I} + \widetilde{\mathbf{L}}_\sigma$ and sampling from $\operatorname{DPP}(\widetilde{\mathbf{L}}_\sigma)$, need to be performed a constant number of times. Both of them take $O(q^3) = O(k^6)$ time by using standard techniques.

All that is left now is to control the cost of the precomputation phase. We will separate the analysis into two steps: how much it costs to choose $\widehat{\mathbf{L}}$ to satisfy the accuracy condition, and how much it costs to compute everything else given $\widehat{\mathbf{L}}$.

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

There exist many algorithms to sample columns proportionally to their RLS. For example, we can take the following result for the BLESS algorithm from Rudi et al. (2018).

Proposition 1 (Rudi et al., 2018) *There exists an algorithm that with probability $1-\delta$ samples m columns proportionally to their RLS in time $O(nk^2 \log^2\frac{n}{\delta} + k^3 \log^4\frac{n}{\delta} + m)$.*

Lemma 2 *Given $\mathbf{L}$ and an arbitrary $\widehat{\mathbf{L}}$ of rank m, computing l_i, s, z, and $\widetilde{\mathbf{L}}$ requires $O(nm^2 + m^3)$ time.*

Combining these results we fully bound DPP-VFX's cost.

Theorem 1 (restated) *For a psd $n \times n$ matrix $\mathbf{L}$, let $S_1, S_2, \ldots$ be i.i.d. random sets from $\operatorname{DPP}(\mathbf{L})$. Denote with $\widehat{\mathbf{L}}$ a Nyström approximation of $\mathbf{L}$ obtained by sampling $m = O(k^3 \log\frac{n}{\delta})$ of its columns proportionally to their RLS. Then, with probability $1-\delta$, Algorithm 1 returns:*

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

b) then, S_2 in $O\left(k^6 \log\frac{1}{\delta} + \log^4\frac{1}{\delta}\right)$ time.

Due to the nature of rejection sampling, as long as we exit the loop, i.e.,, we accept the sample, the output of Algorithm 1 is guaranteed to follow the DPP distribution for any value of m. In Theorem 1 we set $m = O(k^3 \log\frac{n}{\delta})$ 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 $m = O(k \log\frac{n}{\delta})$ 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 Algorithm 1 in three phases. First compute an accurate estimate of the RLS using off-the-shelf algorithms in $O(nk^2 \log^2\frac{n}{\delta} + k^3 \log^4\frac{n}{\delta})$ time. Then sample a small number m of columns to construct an explorative $\widehat{\mathbf{L}}$, and try to run Algorithm 1. If the rejection sampling loop does not terminate sufficiently fast, then we can re-use the RLS estimates to compute a more accurate $\widehat{\mathbf{L}}$ for a larger m. Using a simple doubling schedule for m, this procedure maintains its asymptotic complexity while greatly reducing runtime sampling in practice.

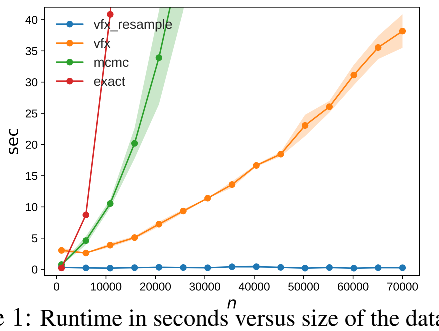

Figure 1: Runtime in seconds versus size of the dataset n.

4. Experiments

In this section we experimentally evaluate the performance of DPP-VFX compared to exact sampling (Hough et al., 2006) and MCMC-based approaches (Anari et al., 2016). In particular, since Section 2 proves that DPP-VFX samples exactly from the DPP, we are only interested in evaluating computational performance. This will be characterized by showing how DPP-VFX and baselines scale with n.

To construct $\mathbf{L}$ we downloaded the original MNIST digits dataset, where $n = 7 \cdot 10^4$ and $d = 784$, and use an RBF kernel with $\sigma = \sqrt{3d}$ to construct $\mathbf{L}$. All algorithms are implemented in `python`. For exact and MCMC sampling we used the `DPPy` library (Gautier et al., 2018), while for DPP-VFX we reimplemented BLESS (Rudi et al., 2018), and used `DPPy` to perform exact sampling on the intermediate subset. All experiments are carried out on a 24 core CPU and fully take advantage of potential parallelization.

The Nyström approximation is performed using BLESS, and, after computing an approximation of RLS and $d_{\text{eff}}(1)$ with BLESS, we set $m = 2d_{\text{eff}}(1) \approx 2k$. While this is much lower than the $O(k^3)$ value suggested by the theory, as we will see it is already accurate enough to result in drastic runtime improvements over exact and MCMC.

In Figure 1 we report our results for subsets of MNIST that grow in size. Exact sampling is clearly cubic in n so we could not push it beyond $n = 20{,}000$. For MCMC, we enforce mixing by runnning the chaing for only nk steps, as indicated by Anari et al. (2016). However, the runtime is still at least 10 times slower than DPP-VFX.

References

Affandi, R. H., Kulesza, A., Fox, E., and Taskar, B. Nystrom approximation for large-scale determinantal processes. In Carvalho, C. M. and Ravikumar, P. (eds.), *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pp. 85–98, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

Alaoui, A. E. and Mahoney, M. W. Fast randomized kernel ridge regression with statistical guarantees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pp. 775–783, Montreal, Canada, December 2015.

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

Anari, N., Gharan, S. O., and Rezaei, A. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In Feldman, V., Rakhlin, A., and Shamir, O. (eds.), *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 103–115, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

Bardenet, R., Lavancier, F., Mary, X., and Vasseur, A. On a few statistical applications of determinantal point processes. *ESAIM: Procs*, 60:180–202, 2017.

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

Calandriello, D., Lazaric, A., and Valko, M. Distributed adaptive sampling for kernel matrix approximation. In *AISTATS*, 2017.

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

Dereziński, M., Warmuth, M. K., and Hsu, D. Leveraged volume sampling for linear regression. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2510–2519. Curran Associates, Inc., 2018.

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

Diaconis, P. and Stroock, D. Geometric Bounds for Eigenvalues of Markov Chains. *The Annals of Applied Probability*, 1991.

Erraqabi, A., Valko, M., Carpentier, A., and Maillard, O.-A. Pliable rejection sampling. In *International Conference on Machine Learning*, 2016.

Gautier, G., Bardenet, R., and Valko, M. Zonotope hit-and-run for efficient sampling from projection DPPs. In *International Conference on Machine Learning*, 2017.

Gautier, G., Bardenet, R., and Valko, M. DPPy: Sampling determinantal point processes with Python. 2018.

Gillenwater, J., Kulesza, A., and Taskar, B. 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, pp. 710–720, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

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

Hough, J. B., Krishnapur, M., Peres, Y., Virág, B., et al. Determinantal processes and independence. *Probability surveys*, 3:206–229, 2006.

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

Kulesza, A. and Taskar, B. k-DPPs: Fixed-Size Determinantal Point Processes. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 1193–1200, Bellevue, WA, USA, June 2011.

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

Launay, C., Galerne, B., and Desolneux, A. Exact Sampling of Determinantal Point Processes without Eigendecomposition. *arXiv e-prints*, art. arXiv:1802.08429, Feb 2018.

Li, C., Jegelka, S., and Sra, S. Efficient sampling for k-determinantal point processes. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 1328–1337, Cadiz, Spain, 09–11 May 2016a. PMLR.

Li, C., Jegelka, S., and Sra, S. 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, pp. 4195–4203, USA, 2016b. Curran Associates Inc.

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

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

Poulson, J. High-performance sampling of generic determinantal point processes. ArXive:1905.00165v1, 2019.

Propp, J. G. and Wilson, D. B. 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.

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

Rudi, A., Calandriello, D., Carratino, L., and Rosasco, L. On fast leverage score sampling and optimal learning. In *Advances in Neural Information Processing Systems 31*, pp. 5672–5682. 2018.