Exact sampling of determinantal point processes with sublinear time preprocessing

Michał Dereziński[†*], Daniele Calandriello[‡*], Michal Valko[§]
[†]UC Berkeley, [‡]LCSL - Istituto Italiano di Tecnologia, [§]DeepMind Paris *Equal contribution

In a nutshell

Determinantal point processes (DPP) are used in machine learning for **randomized selection of diverse subsets** to capture **negative dependencies** between samples.

Existing algorithms for DPP sampling are **expensive** and require the eigendecomposition of an $n \times n$ similarity matrix at $\mathbf{O(n^3)}$ cost.

Our contribution: first **exact** DPP sampler with $\widetilde{\mathbf{O}}(\mathbf{n})$ cost (the cost is **sublinear** in the n^2 size of the similarity matrix!)

Determinantal point processes

Goal: Given an $n \times n$ p.s.d. matrix $\mathbf{L}$, sample $S \subseteq \{1, ..., n\}$ from:

(variant 1) $\mathrm{DPP}(\mathbf{L})$: $\Pr(S) = \dfrac{\det(\mathbf{L}_S)}{\det(\mathbf{I}+\mathbf{L})}$,

(variant 2) k-$\mathrm{DPP}(\mathbf{L})$: above, restricted to $|S| = k$.

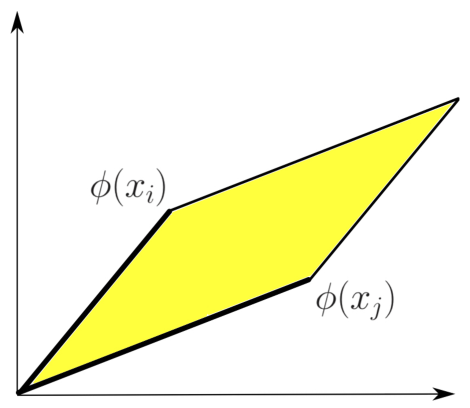
Similarity/Kernel interpretation

$\mathbf{L} = \left[\phi(x_i)^\top \phi(x_j)\right]_{ij}$ for a mappping $\phi : \mathcal{X} \to \mathbb{R}^m$

Determinant is *volume squared* in kernel space:

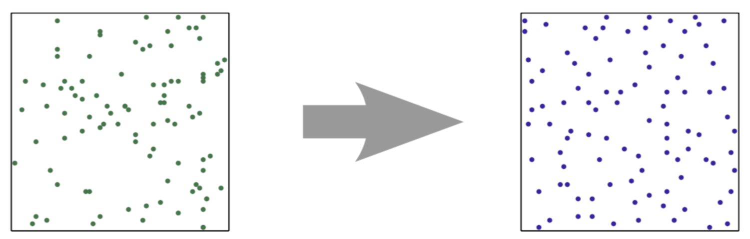
$$\det(\mathbf{L}_S) = \mathrm{Vol}^2\big(\{\phi(x_i) : i \in S\}\big)$$

Expected size $k = \mathbb{E}[|S|]$ is effective dimension of $\mathbf{L}$.



Applications: to learn more about DPPs, see [1]

- Recommender systems,
- Data summarization,
- Stochastic optimization,
- Gaussian processes.



uniform vs DPP

Existing DPP samplers

Exact sampler [2]

Eigendecompose $\mathbf{L}$: $O(n^3)$
performed only once

Volume sampling: $O(nk^2)$
performed for every sample

Cost of $S_1 \sim \mathrm{DPP}(\mathbf{X})$: $O(n^3)$
Cost of $S_2 \sim \mathrm{DPP}(\mathbf{X})$: $O(nk^2)$

MCMC sampler [3]

1. Start from $S \subseteq [n]$
2. Sample $i \in S$ and $j \notin S$
3. Swap w.p. $\frac{1}{2}\min\left\{1, \frac{\Pr(S-i+j)}{\Pr(S)}\right\}$

$\overset{\epsilon}{\sim}$: ϵ-close in total variation distance

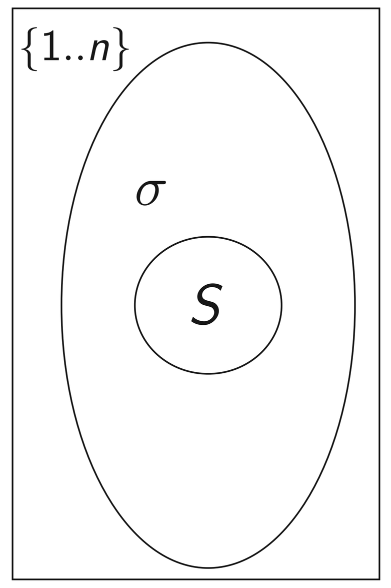
Cost of $S_1 \overset{\epsilon}{\sim} k$-$\mathrm{DPP}(\mathbf{X})$: $n \cdot \mathrm{poly}(k)$
Cost of $S_2 \overset{\epsilon}{\sim} k$-$\mathrm{DPP}(\mathbf{X})$: $n \cdot \mathrm{poly}(k)$

Distortion-free intermediate sampling

💡 Sample intermediate set σ out of $\{1..n\}$, then downsample S out of σ

🤔 How to ensure that intermediate σ *always* contains DPP sample S?

1. Use marginal inclusion probabilities of $S \sim \mathrm{DPP}(\mathbf{L})$
$$\Pr(i \in S) = \ell_i = \left[\mathbf{L}(\mathbf{I}+\mathbf{L})^{-1}\right]_{ii}$$
a.k.a. 1-ridge leverage scores, $\sum_{i=1}^n \ell_i = \mathbb{E}[|S|] = k$
2. Sample i.i.d. $\sigma_1, \ldots, \sigma_t \sim [\ell_1/k, \ldots, \ell_n/k]$



🙁 Ignores negative dependence: $S \nsubseteq \sigma$ possible

💡 Reject σ if $S \nsubseteq \sigma$! Sampling becomes *exact*!

🤔 **Key question:** how many rejections occur before we get a good σ?

Trade-offs: 1. How large should the intermediate sample σ be?
2. How accurate should the leverage score estimates be?

DPP-VFX: first sub-linear time exact DPP sampler

Thm. For any $\mathrm{DPP}(\mathbf{L})$ or k-$\mathrm{DPP}(\mathbf{L})$, we can sample

a) the first subset S_1 in: $n \cdot \mathrm{poly}(k)\,\mathrm{polylog}(n)$ time,
b) each successive S_i in: $\mathrm{poly}(k)$ time.

🙂 **Our answer:** we only suffer a constant number of rejections!

1. Size of the intermediate sample is $t = O(k^2)$, independent of n!
2. $l_i \approx \big(1 \pm O(1/k)\big)\ell_i$ leverage scores estimates are accurate enough
↳ efficient to compute with off-the-shelf Nyström approximations

🙂 Easier in practice: $l_i \approx (1 \pm 1/2)\,\ell_i$ enough for constant rejections

Comparison with existing DPP samplers

	exact	DPP	k-DPP	first sample	next sample
Hough et al. [2]	✓	✓	✗	n^3	nk^2
Kulesza and Taskar [1]	✓	✗	✓	n^3	nk^2
Anari et al. [3]	✗	✗	✓	$n \cdot \mathrm{poly}(k)$	$n \cdot \mathrm{poly}(k)$
Li et al. [4]	✗	✓	✗	$n^2 \cdot \mathrm{poly}(k)$	$n^2 \cdot \mathrm{poly}(k)$
DPP-VFX	✓	✓	✓	$n \cdot \mathrm{poly}(k)$	$\mathrm{poly}(k)$

Very Fast and eXact DPP sampler (DPP-VFX)

Input: $\mathbf{L} \in \mathbb{R}^{n\times n}$, its Nyström approximation $\widehat{\mathbf{L}}$

$l_i = \left[(\mathbf{L}-\widehat{\mathbf{L}}) + \widehat{\mathbf{L}}(\mathbf{I}+\widehat{\mathbf{L}})^{-1}\right]_{ii} \approx \ell_i$,
$s = \sum_i l_i$, $z = \mathrm{tr}\big(\widehat{\mathbf{L}}(\mathbf{I}+\widehat{\mathbf{L}})^{-1}\big)$, $[\widetilde{\mathbf{L}}]_{ij} = [\mathbf{L}]_{ij}/(s\sqrt{l_i l_j})$ } preprocessing

```
1: repeat
2:    sample t ~ Poisson(s^2 e^{1/s})
3:    sample σ_1, ..., σ_t ~ (l_1/s, ..., l_n/s),
4:    sample Acc ~ Bernoulli( e^{z-t/s} · det(I+L̃_σ)/det(I+L̂) )
5: until Acc = true
                                             } rejection sampling
6: sample S̃ ~ DPP(L̃_σ)
7: return  S = {σ_i : i ∈ S̃}
                                             } downsample
```

General reduction from k-DPP to DPP sampling

Folklore heuristic: k-DPP rejection sampler

$$\textbf{repeat} \quad S_\alpha \sim \mathrm{DPP}(\alpha\mathbf{L}) \quad \textbf{until} \quad |S_\alpha| = k$$

🤔 Rejecting is expensive. How often should we do it?

The folklore heuristic was right (with the right $\alpha^\star$)

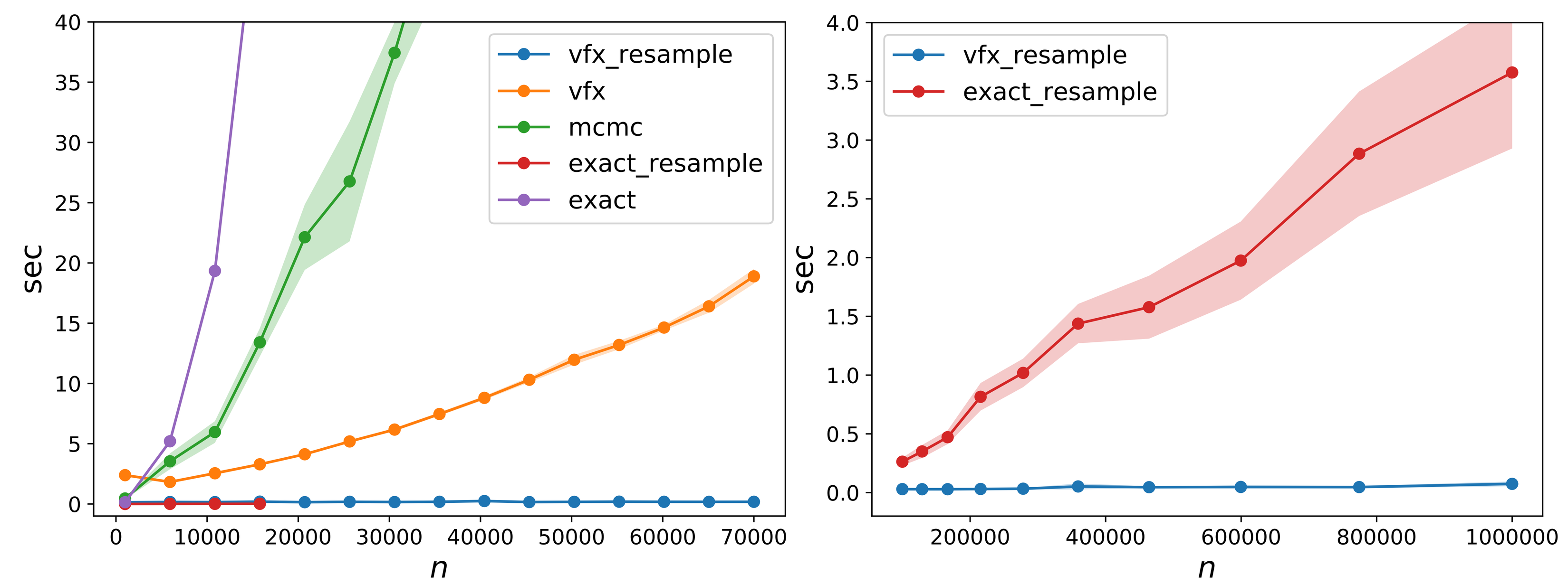
Thm. We can compute $\alpha^\star$ in $\widetilde{O}(n \cdot \mathrm{poly}(k))$ such that

$$\mathrm{Mode}(|S_{\alpha^\star}|) = k, \qquad \mathbf{P}(|S_{\alpha^\star}| = k) \geq \Omega(k^{-1/2})$$

inducing at most $O(\sqrt{k})$ rejections.

Experiments

Sampling from MNIST8M with $n \in \{1..10^6\}$



	DPP-VFX		eigendecomp. (exact)		MCMC	
	first	succ.	first	succ.	first	succ.
$n = 15000$	4	1	54 (10x)	1 (1x)	13 (4x)	13 (13x)
$n = 70000$	19	1	DNF	DNF	175 (9x)	175 (175x)

Runtime in seconds and corresponding (speedup), DNF = Did Not Finish

[1] A. Kulesza and B. Taskar. Determinantal Point Processes for Machine Learning. Now Publishers Inc., 2012 [2] J.B. Hough, M. Krishnapur, Y. Peres, B. Virag. Determinantal processes and independence. Probability Surveys, 2006 [3] N. Anari, S.O. Gharan, A. Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. COLT'16 [4] C. Li, S. Jegelka, S. Sra. Fast mixing markov chains for strongly rayleigh measures, dpps, and constrained sampling. NeurIPS'16