Semi-Discretized sampling for Determinantal Point Processes

Gregory de Salaberry Seljak
Department of Computer Science
ENS Paris-Saclay
de.gregory@ens-paris-saclay.fr

Baptiste Abeles
Department of Computer Science
ENS Paris-Saclay
jean.abeles@ens-paris-saclay.fr

Abstract

[2] show striking results for the application of Determinantal Point Processes (DPP) as applied to Monte Carlo (MC) numerical integration methods, achieving convergence in $O\left(N^{-\frac{1}{2}\left(1+\frac{1}{d}\right)}\right)$ for N nodes. This comes at the cost of a burdensome sampling complexity $\mathcal{O}(N^3)$. We discuss the challenges of sampling from continuous-valued DPPs and compare some basic sampling algorithms. We specifically draw attention to the failure of loose intuition in simulating DPPs, and show a weakness of [6] in a simple setting.

1 Introduction and Background

General methods for numerical integration work by choosing N representative points at which to evaluate an objective function f over a measure η. However, the choice of representative points to draw suffers from a trade-off between (qualitatively defined) *discrepancy* and *diversity*. Non-stochastic quadrature techniques are in some sense low-discrepancy, meaning that they basically guarantee a representative coverage of supp(η). Even in the best case of a Lipschitz integrand, the non-vanishing gaps between nodes will eventually lead to systematic errors while sampling from the objective function, giving rise to the well-documented *curse of dimensionality*. Even sophisticated methods with good performance in 1-d such as Gaussian quadrature show MSE rates on the order of $O\left(N^{-1/d}\right)$, making them scale poorly with dimension.

The purely Monte Carlo method in its many forms (MCMC, etc.) aims to distribute points as $iid.$ draws of η. These draws are diverse in the sense that the samples are completely uncorrelated, but high-discrepancy in the sense that samples tend to clump in some areas and be sparse in others. While this quantitatively solves the curse of dimensionality, the relative inefficiency of having discrepant samples leads to errors of the order of $O\left(N^{-\frac{1}{2}}\right)$.

It is thus intuitive that the most modern techniques which improve on this per-node error rate are those which balance the intra-sample diversity of the non stochastic techniques with the inter-sample diversity of the stochastic techniques. The best known example in this line of literature is certainly Quasi Monte Carlo, and in particular Scrambled Nets [19]. In contrast with QMC, the Determinantal Point Process MC demonstrated by [2] draws samples in a way which is genuinely stochastic, but favours samples with some form of low discrepancy. Some authors refer to this balance as "structured Monte Carlo" [5]. While the guarantees of [2] are enticing and have proven to be computationally tractable [10], the cost of generating sample nodes limits the utility to a relevant but tightly-defined set of problems where evaluating the objective f is very costly, or the isotropic properties of the proposal distribution are favorable [18]. There is an obvious interest then in approximating DPPs quickly in order to capture some of the low-order properties of the distributions, such as in [6] and [8]. We review some of the reasoning of the semi-discretized approximate sampling of [6] and implement their method directly.

Submitted to Bayesian Machine Learning class MVA Spring 2023

1.1 Determinantal Point Processes

The modern[1] popular use of Determinantal Point Processes for obtaining diverse samples in machine learning goes back at least to 2012 [13]. A DPP $\Xi(K, \mu)$ is a random variable which has value in the *power set* of space Ω. This requires a modification of the usual notion of density, as test functions are no longer well-defined on the output of Ξ. Instead, density is replaced by the notion of *intensities* $\rho_{(k)} : \Omega^k \to \mathbb{R}_+$, defined as:

For any $n-$family of mutually disjoint subsets $(D_i)^n \subset \Omega$,

$$\mathbb{E}\left[\prod_i^n \kappa(D_i)\right] = \int_{D_1 \ldots} \int_{D_n} \rho_{(k)}(x_1, \ldots x_k) \prod_i^k d\mu$$

Where κ is the counting measure. The analogy with defining a density as $\mathbb{E}[\mathbb{1}_A] = \int_A p(x) d\nu$ is clear given that this is exactly the case for ρ_1

For hermitian K, the process $\Xi(K, \mu)$ exists if and only if $K : \Omega^2 \to \mathbb{C}$ is positive semidefinite, and the eigenvalues of the embedding operator Σ over μ lie in the interval $[0, 1]$

$$\Sigma : L^2 \to \mathcal{H} \qquad f \mapsto \int K(\cdot, y) f(y) d\mu, \qquad \text{where } \mathcal{H} \text{ is the RKHS of } K$$

Note that $\rho_{(k)}$ is constant under permutations of its arguments. Among the intensity functions $\rho_{(k)}$, of particular interest is the second-order intensity $\rho_2(x, y)$, which expresses the pairwise interactions in the point process. In its normalized form,

$$g(x, y) = \frac{\rho_2(x, y)}{\rho_1(x)\rho_1(y)} \qquad \cong \qquad \frac{p(X_j = y \,|\, X_i = x)}{p(X_i = x)} = \frac{p(X_i = x \,|\, X_j = y)}{p(X_j = y)} \tag{1}$$

Conditioning on the output having at least two points, this expression can be interpreted as a ratio of marginalized densities for a given point in the sample. When g (equivalently, K) is a function of the distance $|x - y|$, the process is *stationary* and g can be simply interpreted as the pairwise repulsion (stronger for small values of g) or attraction (for values of g greater than 1) effects. This is not directly applicable here, but recent advances in sampling techniques will likely prove useful in such cases[15]: see [4] for a complete discussion. However, a decomposition of the $g(x, y)$ in [2] as stationary and non-stationary components would show the varying strength of repulsion between nodes (stationary effect) depending on the distance to the zeros of the OPEs (non-stationary effect).

In principle, the realization of a DPP can be any subset of Ω in its power set $P(\Omega)$. In practice, and for means of practical comparison among Monte Carlo methods, we restrict ourselves to *projection DPPs*, alternatively referred to as elementary DPPs[14]. Realizations of projection DPPs only take values in the $N-$sized elements $P_N(\Omega) \subset P(\Omega)$. They are however very restrictive in the admissible kernel K_N and associated measure μ; a projection DPP requires that precisely N of the eigenvalues of K_N with respect to μ be 1, and the others be 0. Under these conditions, the distribution of the point sets from Ξ can be used directly to induce a joint probability distribution q over the points $(x_0, \ldots x_N)$.

$$q(x_0, \ldots x_N) \cong \frac{1}{N!} \det\left[k(x_i, x_j)\right]_{ij} \prod_i^N \mu(x_i) \tag{2}$$

Example 1.

Consider $\Omega = [-1, 1]^d$, and let K_ℓ be the linear kernel. Then $K_\ell(x, y)$ admits an eigendecomposition

$$K_\ell(x, y) = \sum_i^d x_i x_j \qquad \mu(dx) = \frac{3}{2^d} dx$$

The eigenvalues of K_ℓ are exactly $\lambda_{1 \le i \le d} = 1$ and $\lambda_{i > d} = 0$, and the eigenfunctions are the coordinate projections of the space Ω

[1] Post-formalization of DPPs by Macchi (1975) [17] and dissolution of the USSR (1991)

1.2 Results of Bardenet and Hardy (2016)

The primary result of Bardenet and Hardy (2016) is that for the importance-sampling based estimator $\hat{f}_N \widetilde{=} \int_{I_d} f d\mu$ using N nodes of a measure η with smooth, compactly-supported density wrt. lebesgue, the standard error decays in $O(N^{\frac{1}{2}\left(1+\frac{1}{d}\right)})$

$$N^{\frac{1}{2}\left(1+\frac{1}{d}\right)}\left(\hat{f}_N - \int_{I_d} f d\eta\right) \xrightarrow{N\to\infty} \mathcal{N}(0,\sigma^2) \tag{3}$$

While the original implementation makes effective use of the orthogonalized chain rule given by [12], the best-case complexity is $\mathcal{O}(N^3)$. Hardy and Bardenet propose DPPMC use-cases where overall complexity is dominated by evaluating the objective function f, which encompasses a relevant but non-generic set of problems[2] This barrier for utility would be softened if DPPs could be sampled faster at the cost of precision. Sampling from continuous DPPs remains one of the foremost challenges in effective implementation of DPPMC and is thus the main focus of the rest of the paper.

2 Approximate DPP Sampling

Unlike continuous-valued DPPs, sampling from finite sample space DPPs
(having value in $\{S \subset \Omega : |\Omega| < \infty\}$) has been the focus of much literature since [13]: see, for instance, [7][16]. The distribution induced by a projection DPP (equation 2) suggests a decomposition into the ambient measure $\prod_i^N \mu(dx) = \mu(dX)$ and particle-interaction effect $\det[k(x_i,x_j)]_{ij}$. It is thus natural to try to decompose sampling into a two-step algorithm: A continuous, non-determinantal point sampling step, and then a finite determinantal one which exploits existing finite-DPP techniques. We seek to find a DPP sampler which is at least as good as $\mathcal{O}(N^3)$ through more generic methods than those of the determinantal chain rule of [12].

$$p_{dpp}(X) \widetilde{=} \det[K(x_i,x_j)]_{ij} p_{\Pi\mu}(X)$$

2.0.1 Direct Monte Carlo sampler

Algorithm 1 Direct Monte Carlo Sampling Pseudocode

let $N \in \mathbb{N}$, and $\xi \in \mathbb{N}$

1. draw ξ iid. point sets $X_i \sim \prod_j^N \mu_j(x_j)$
2. Calculate the Gram matrix $L_i = [K_N(x_p,x_q)]_{pq}$ for each X_i
3. Sample discrete distribution $P(X = X_i) = \det L_i \frac{1}{\sum_j \det L_j}$

In the asymptotic case of $\xi \to \infty$, Algorithm 1 gives exactly (2); in the reverse case where $\xi = 1$, Algorithm 1 gives exactly $\prod_i \mu_i$. Algorithm 1 suffers from enormous complexity costs in steps 2 and 3.

It is easy to imagine other kinds of familiar Monte Carlo sampling methods that would modify algorithm 1, such as Markov Chain MC. However, these methods share a common weakness in requiring that the distribution function p_{DPP} be computed ξ times (ex. in MCMC, ξ is the number of steps). Given the cost of computing the determinant, this family of algorithms is impractical.

2.0.2 Pseudo-Gibbs Monte Carlo Sampler

A more clever approach than algorithm 1 in applying a determinant-valued distribution to a finite subsample of $P_N(\Omega)$ is to overdraw the base samples X_i and downsample from the $N-$permutations therein according to the discrete DPP $[K_N(x_(\sigma(i),x_\sigma(j))$. If ρN samples are initially drawn, the subsample space will be grow with $\binom{\rho N}{N}$. However, in this crude implementation these proposal samples are obviously strongly correlated and sampling is not particularly fast at $\mathcal{O}\left((N\rho)^3\right)$.

Algorithm 2 Pseudo-Gibbs Monte Carlo Pseudocode

let $N \in \mathbb{N}$, $\rho \in \mathbb{R}_+$ with $\rho > 1$

1. Draw $N\rho$ iid. points $x_i \sim \prod_j^N \mu$

2. Calculate the oversampled Gram matrix $\mathbf{L} = [K_N(x_p, x_q)]_{pq}$ of all points

3. Sample a submatrix among the principal minors of $\mathbf{L}$ according to the associated discrete $k - DPP$

2.1 Core Set Sampler (Choromanski 2020)

Algorithm 3 Core Set Sampler Pseudocode

let , $N \in \mathbb{N}$, $M \in \mathbb{N}$ with $M \geq N$, $\rho \in \mathbb{R}_+$ with $\rho > M$

1. Draw ρN iid. points $x_i \sim \prod_j^N \mu$

2. Assign each point x_i to a *core set* $\rfloor_j$ using kernel clustering and calculate cluster mean c_j
3. Calculate the cluster-mean Gram matrix $\mathbf{L} = [c_i^\top c_j]_{ij}$ of all clusters

4. Sample a submatrix among the principal minors of $\mathbf{L}$ according to the associated $k - DPP$
5. Subsample uniformly among the elements of the selected clusters c_i

The sampling mechanism of algorithm 3 in [6][2] is a much more practical version of algorithm 2, but implements the efficient sampling described by [16]. They claim that similarity in points is essentially transitive; $iid.$ points from μ come in clusters, and the permutations in algorithm 2 which have the greatest diversity among these clusters will dominate the discrete distribution (in fact, their particular implementation *imposes* this clustering and uses it to skip step 2). The `CoreDPP` algorithm of Jegelka et al. reduces this redundancy by summarizing each of M clusters as a single representative in the discrete kernel matrix L, selecting among the *clusters* based on determinants, and then uniformly sampling individuals among the chosen clusters. We follow the implementation procedure of [6], but we exchange their Gaussian mixture approximation step for our step 2 as uniform sampling is implemented immediately. We claim that in the given setting, our implementation cannot perform worse numerically than the approximation $\mathcal{P}_{\text{EM}}$ they suggest.

3 Simulation Results

The proposed sampling methods are tested on the degree$-\mathbf{b} = 10$ Christoffel-Darboux polynomial kernel for Legendre polynomials in dimension $d = 2$. These parameters are chosen arbitrarily to suit the computational power available.

3.1 Determinant Distribution

The obvious first example for confirming the correctness of the algorithms is to see how well the histogram of drawn samples matches the density function. This is a necessary but not sufficient condition for empirically confirming correctness. Results are seen in attached figure 1. The distribution of the density function $f_K(X) = \det[K(x_i, x_j)]_{ij}$ with respect to the uniform measure does not admit an obvious theoretical distribution, so a high-count (1×10^5) Monte Carlo approximation is made instead (green figures at the bottom). The imprecision in this count accounts for some of the shared variance seen on the right-side figures. The algorithms each show varying degrees of agreement with the expected linear curve (grey line). The lack of agreement beyond a certain point is attributed to the modest number of computations, not the properties of the distributions (the space of determinants is compact).

[2] Algorithm 1 in their text

3.2 Boundary Errors

Suppose, in the best case, that the disagreements shown in figure 1 are a result of insufficient sample sizes. We examine the actual point sets that are given by algorithm 3 in figures 2-3, and the cumulative point plots in figure 4. We emphasize that the fundamental difference between determinantal sampling and typical Monte Carlo sampling is the non-interchangeability (non-independence) of points between sampling iterations. Agreement in the histograms of figure 4 should then be considered as well to be a necessary-but-not-sufficient condition for approximation. It is clear from this figure that there is significant difference between the coreset approximation sampler and the DPPy reference sampler. Observing the effect of increasing the parameter ρ from 100 to 1000 shows little improvement.

4 Discussion and Conclusion

4.1 Choromanski (2020)

It is not clear that the comparison made in [6] of being a "much faster *DPP sampler* than Gautier et al. 2019" are accurate given the lack of guarantees for asymptotic convergence to a true DPP distribution[3]. A fundamental problem faced by this algorithm is the ambiguity of how samples should be taken near the boundaries of the domain. The k-means cluster step introduced in our Algorithm 3 is meant to allow for a non-negligible density of particles right up to the domain boundary at any cost in complexity (the hypothesis posed in the original text regards this as an an unnecessary extra step). Even in the best case, our numerical results show unconvincing agreement. Even worse, the sampling from Gaussian-mixture approximation used in [6] is not clearly defined for non-vanishing distributions. In a higher-dimensional space, such as is the case with the tested Evolutionary Models in [6], the effect may become negligible, which explains their relative success in numerical experiments.

4.2 Utility of approximate DPPs beyond integrands

Our findings reflect the guiding principles of DPP use in Monte Carlo methods: DPPs are powerful but very constrained and expensive tools, and that approximation techniques are presented *caveat emptor*. Alarmingly, poor implementation can even perform worse than conventional Monte Carlo under $iid.$ even in more sophisticated approaches with appropriate domain (see [11] regarding [1]).

Approximative DPP algorithms might prove to be useful not so much as IS-proposal samplers in themselves but instead as an auxiliary tool to Bardenet and Hardy's DPP-IS. [2] show that the DPP-IS algorithm improves in σ (eq. 3) when the integrand is close to a coordinate plane in the RKHS of K, but demonstrating that this condition has been met is non-trivial. A use case for approximate DPP would be for users to test out different choices of kernel as a way of guessing which ones induce an RKHS that is favourable to a family of integrands before implementing a full-scale version of DPP-IS.

4.3 Developments in Continuous Stationary-DPP Sampling

The field of exact continuous-valued DPP sampling techniques is also showing active development. Spectral decomposition of the Gaussian kernel and demonstration of its use in DPP-IP by [3] is bound to draw attention as users will be more familiar with the properties of Gaussian kernels, and stationary DPPs come with a host of established literature. Recalling the effect of stationarity from section 1, we comment that it is not obvious that a stationary DPP will perform favourably in the context of Monte Carlo on a compact domain, as the measure-discrepancy of nodes bunching near the domain edges is unlikely to improve. In the extreme case, repulsion induces a packing problem over the compact domain which is bound to recreate some of the problems mentioned in section 1 about generalized quadrature. We speculate that one of the easier parts of manipulating DPP-IS is to guess how repulsive the distribution should behave on the domain. As an example, consider functions which are flat (but not necessarily vanishing) near the boundary. Looking at $g_x(y-x) = g(x,y)$ from eq. 1, the most generically useful kernels for DPP-IS in that case will probably have lower values of g_x around 0 when x is near the domain boundary and will take more neutral values when x is near the center.

[3] Despite citing [14], no mention is made of elementary or projective restrictions; it might better be referred to as *DPP-like* sampler.

References

[1] Raja Hafiz Affandi, Emily B. Fox, and Ben Taskar. Approximate inference in continuous determinantal point processes, 2013. URL https://arxiv.org/abs/1311.2971.

[2] Rémi Bardenet and Adrien Hardy. Monte carlo with determinantal point processes, 2016. URL https://arxiv.org/abs/1605.00361.

[3] Nicholas P Baskerville. A novel sampler for gauss-hermite determinantal point processes with application to monte carlo integration, 2022. URL https://arxiv.org/abs/2203.08061.

[4] Christophe Ange Napolé on Biscio and Frédéric Lavancier. Quantifying repulsiveness of determinantal point processes. *Bernoulli*, 22(4), nov 2016. doi: 10.3150/15-bej718. URL https://doi.org/10.3150%2F15-bej718.

[5] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, and Yunhao Tang. Structured monte carlo sampling for nonisotropic distributions via determinantal point processes, 2019. URL https://arxiv.org/abs/1905.12667.

[6] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, and Yunhao Tang. Practical nonisotropic monte carlo sampling in high dimensions via determinantal point processes. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1363–1374. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/choromanski20a.html.

[7] Michal Derezinski, Daniele Calandriello, and Michal Valko. Exact sampling of determinantal point processes with sublinear time preprocessing. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/fa3060edb66e6ff4507886f9912e1ab9-Paper.pdf.

[8] Guillaume Gautier, Rémi Bardenet, and Michal Valko. Zonotope hit-and-run for efficient sampling from projection dpps. 2017. doi: 10.48550/ARXIV.1705.10498. URL https://arxiv.org/abs/1705.10498.

[9] Guillaume Gautier, Guillermo Polito, Rémi Bardenet, and Michal Valko. Dppy: Sampling dpps with python. 2018. doi: 10.48550/ARXIV.1809.07258. URL https://arxiv.org/abs/1809.07258.

[10] Guillaume Gautier, Rémi Bardenet, and Michal Valko. On two ways to use determinantal point processes for monte carlo integration. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1d54c76f48f146c3b2d66daf9d7f845e-Paper.pdf.

[11] Philipp Hennig and Roman Garnett. Exact sampling from determinantal point processes, 2016. URL https://arxiv.org/abs/1609.06840.

[12] J. Ben Hough, Manjunath Krishnapur, Yuval Peres, and Bálint Virág. Determinantal Processes and Independence. *Probability Surveys*, 3(none):206 – 229, 2006. doi: 10.1214/154957806000000078. URL https://doi.org/10.1214/154957806000000078.

[13] Alex Kulesza. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, 2012. doi: 10.1561/2200000044. URL https://doi.org/10.1561%2F2200000044.

[14] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. pages 1193–1200, 01 2011.

[15] Frédéric Lavancier and Ege Rubak. On simulation of continuous determinantal point processes, 2023. URL https://arxiv.org/abs/2301.11081.

[16] Chengtao Li, Stefanie Jegelka, and Suvrit Sra. Efficient sampling for k-determinantal point processes, 2015. URL https://arxiv.org/abs/1509.01618.

[17] Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975. ISSN 00018678. URL http://www.jstor.org/stable/1425855.

[18] Adrien Mazoyer, Jean-François Coeurjolly, and Pierre-Olivier Amblard. Projections of determinantal point processes, 2019. URL https://arxiv.org/abs/1901.02099.

[19] Art B. Owen. Scrambled net variance for integrals of smooth functions. *The Annals of Statistics*, 25(4):1541 – 1562, 1997. doi: 10.1214/aos/1031594731. URL https://doi.org/10.1214/aos/1031594731.

Appendix - Computation Notes

Computations were performed on 11th Gen Intel(R) Core(TM) i5-11300H and used packages DPPy[9] and pyDPP [6]

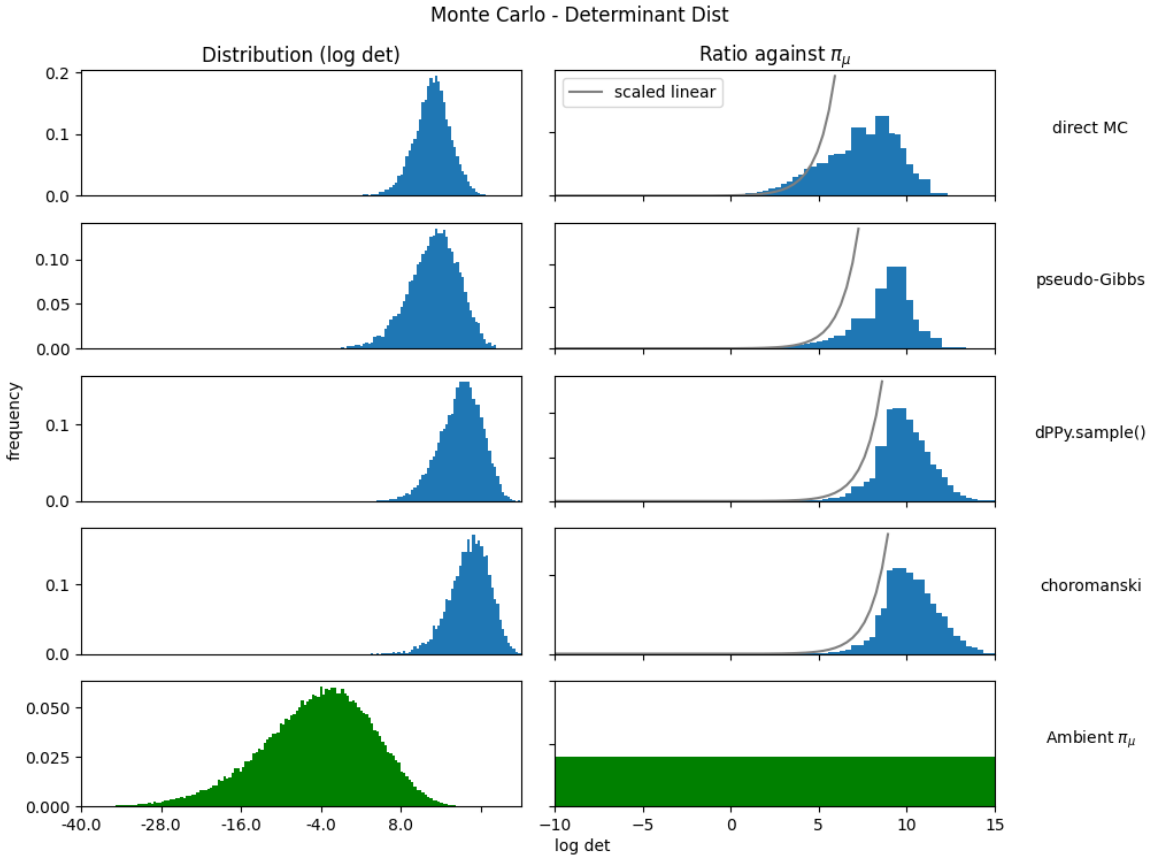

Figure 1: Left: Empirical distribution of determinants $\det[K(x_i, x_j)]$ under the given sampling mechanism. Right: Ratio of empirical density with respect to the empirical density under uniform distribution. A linear function is plotted along the expected ratio of distributions (grey). Each trial is run with 10^4 MC samples, except for the reference uniform trial (green) which is run with 10^5.

Hyperparameters in each algorithm:

1. $\rho = 20 * N$
2. $\rho = 10 * N$
3. n/a
4. $\rho = 100 * N, M = 10N$

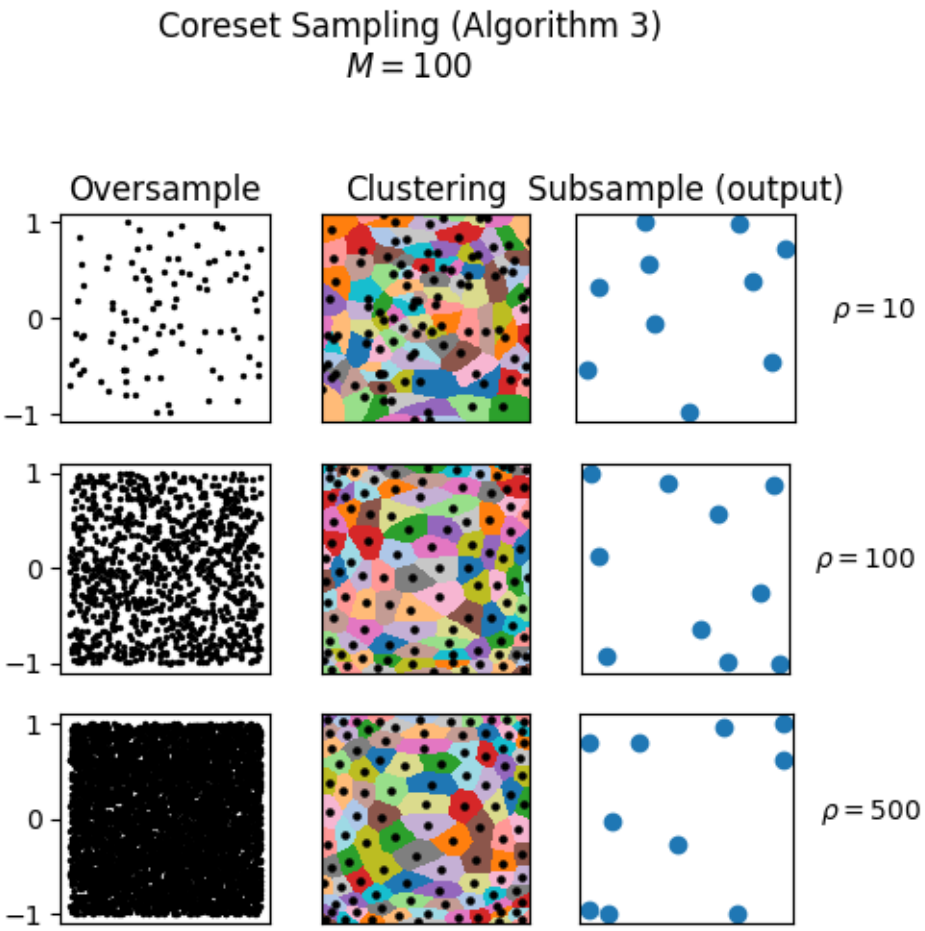

Figure 2: Coreset sampling algorithm for different values of hyperparameter ρ.
Left: overdrawn samples on the domain.
Center: Results of kernel M-means clustering with the partition of each c_i shaded in. The black dot "cluster representatives" are only for a visual guide as the actual mean lies in feature space.
Right: The output of the algorithm.

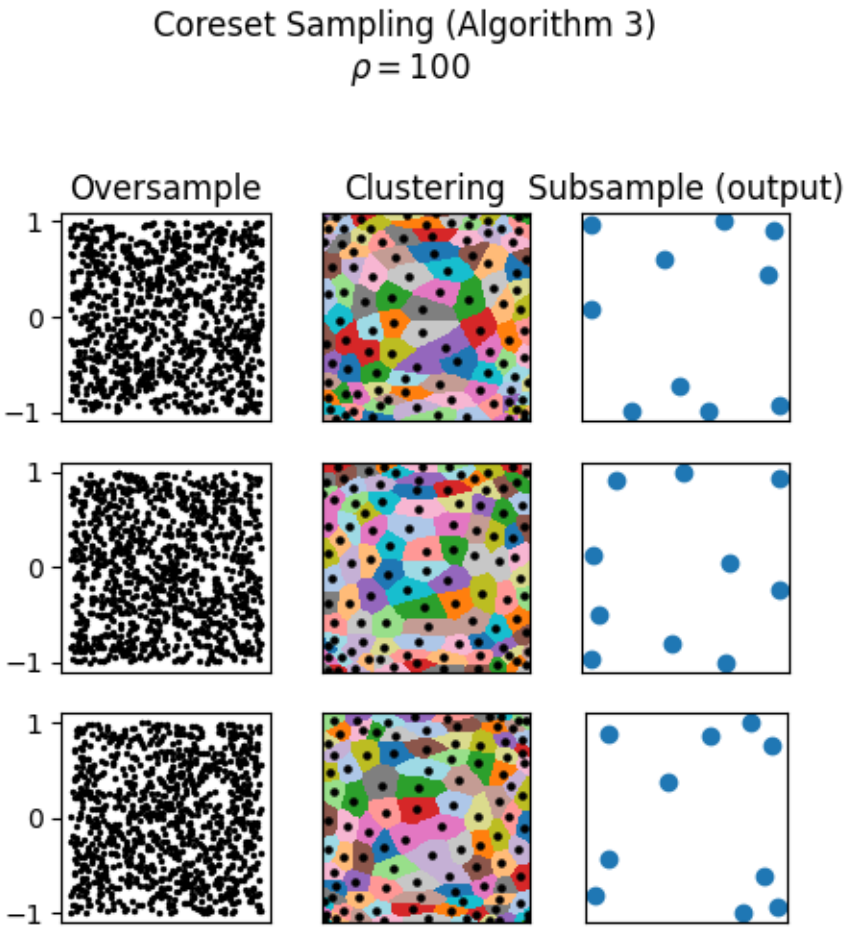

Figure 3: Coreset sampling algorithm for fixed draws of fixed hyperparameter ρ

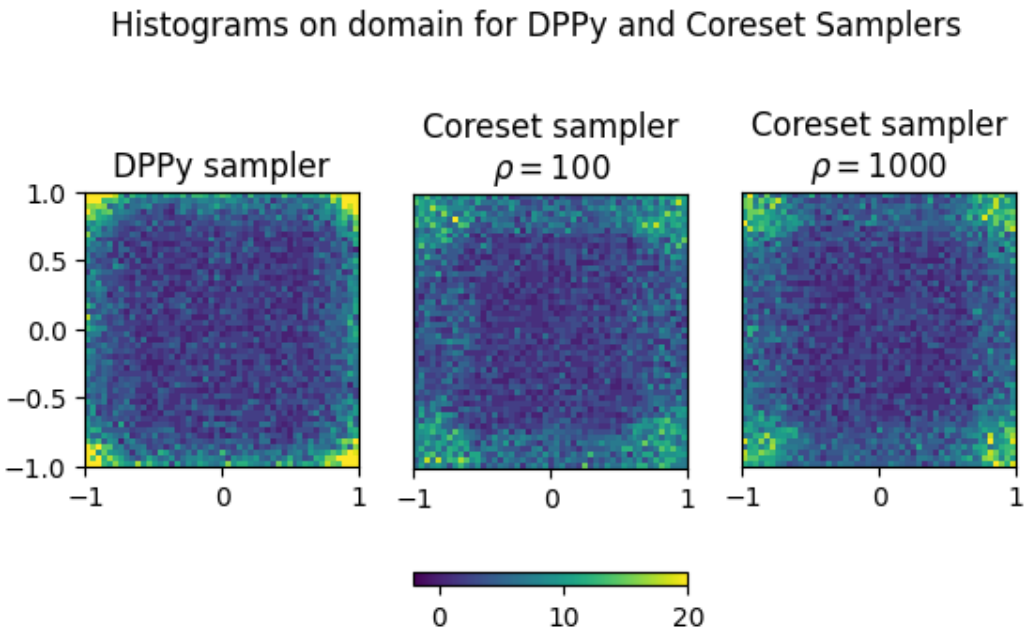

Figure 4: Cumulative heatmap for all points in 10^3 draws of $10-$point sets $(N = 10)$