A Greedy Approximation for k-Determinantal Point Processes

Julia Grosse
University of Tübingen,
Tübingen AI Center

Rahel Fischer
University of Tübingen,
Tübingen AI Center

Roman Garnett
University of Washington

Phlipp Hennig
University of Tübingen,
Tübingen AI Center

Abstract

Determinantal point processes (DPPs) are an important concept in random matrix theory and combinatorics, and increasingly in machine learning. Samples from these processes exhibit a form of self-avoidance, so they are also helpful in guiding algorithms that explore to reduce uncertainty, such as in active learning, Bayesian optimization, reinforcement learning, and marginalization in graphical models. The best-known algorithms for sampling from DPPs exactly require significant computational expense, which can be unwelcome in machine learning applications when the cost of sampling is relatively low and capturing the precise repulsive nature of the DPP may not be critical. We suggest an inexpensive approximate strategy for sampling a fixed number of points (as would typically be desired in a machine learning setting) from a so-called k-DPP based on iterative inverse transform sampling. We prove that our algorithm satisfies a $(1 - 1/e)$ approximation guarantee relative to exact sampling from the k-DPP, and provide an efficient implementation for many common kernels used in machine learning, including the Gaussian and Matérn class. Finally, we compare the empirical runtime of our method to exact and Markov-Chain-Monte-Carlo (MCMC) samplers and investigate the approximation quality in a Bayesian Quadrature (BQ) setting.

1 INTRODUCTION

Determinantal point processes (DPPs), introduced by Macchi (1975), are point processes whose joint intensity is proportional to the determinant of a positive definite kernel Gram matrix. Intuitively, this introduces a dependence between points in samples from such processes that gives them a repulsive property—points drawn from DPPs cover a space more regularly than uniform random samples; see Figure 1 (left vs. middle).

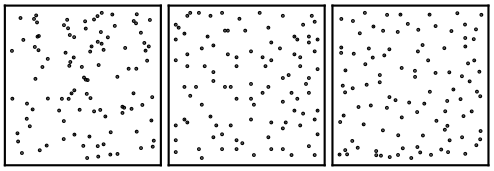

Figure 1: 100 random points on the interval $[0, 1]^2$ sampled uniformly at random (left), sampled from a k-DPP with square-exponential kernel with a length-scale of 0.1 (middle), and sampled from its greedy approximation that is the subject of this paper (right).

DPPs initially arose in the study of fermionic gases in physics and have since found application in other areas, such as random matrix theory (Mehta, 1991). A review of their statistical properties is provided by Soshnikov (2000). Meanwhile, DPPs have also received attention in machine learning and statistics. A reason for the growing interest in them is that they provide an elegant theoretical view on the notion of *exploration* that is of relevance across machine learning. In areas like active and reinforcement learning, as well as in numerical tasks like marginalization in graphical models, the basic challenge is that the algorithm should in some sense "probe" an input domain in a maximally informative way. The repulsive property of DPPs can help automatically guide such a procedure.

As we will review below, DPPs have a direct connection to the entropy of Gaussian process models, which closely ties them to many basic probabilistic algorithms in machine learning, e.g. in Bayesian optimization (Garnett, 2023; Nava et al., 2022; Kathuria et al., 2016; Wang et al., 2017), Bayesian quadrature (Bardenet and Hardy, 2016), kernel quadrature (Bel-

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

hadji et al., 2019) and Monte Carlo integration (Gautier et al., 2019a). DPPs have been used as diversity-inducing priors (e.g., Kulesza and Taskar, 2012) and found applications in several other areas, such as recommender systems (Wilhelm et al., 2018), clustering (Kang, 2013), neural network compression (Mariet and Sra, 2015), batch stochastic gradient descent (Zhang et al., 2017) or learning diverse generative models (Elfeki et al., 2019). In these applications, the number of points desired k is usually fixed by the users in advance, whereas in a DPP, this size is a random variable. To this end, we may consider a so-called k-DPP, which is a DPP conditioned to have fixed size. In machine learning, the process of sampling from a k-DPP is often used a subroutine rather than the end goal in itself. Therefore, it can then be desirable to trade-off accuracy in the sampling process for speed. This is the motivation for our contributions here.

After reviewing the definition of a k-DPP in §2, we start by characterizing the k-DPP as a point processes maximizing a natural score function. This score function consists of two components: One that ensures that the elements of a single point set sampled from the process are expected to be diverse. And another one that rewards a high entropy of the point processes itself and thereby ensures that multiple point sets sampled from the process are expected to be diverse themselves.

We then introduce the greedy strategy for approximate maximization of this score function. Our motivation to choose a greedy approach for the approximation is due to its success in the analogous non-stochastic setting. It is common practice in tasks that involve the (non-stochastic) exploration of a function or a domain, e.g. by maximizing information gain (Srinivas et al., 2009; Hennig and Schuler, 2012; Ma et al., 2018) or entropy (Sharma et al., 2015). Hence, we suspect it to be useful for our stochastic *sampling* setting, too. We continue with a theoretical analysis of the approximation quality, where we use tools from submodular optimization to show that the point process defined by our greedy sampling procedure achieves a near-optimal value of the introduced score function.

While a large amount of computational costs is saved by sampling greedily, the implementation costs of a single greedy decision can still be significant over large domains or continuous domains. However, we will show that the kernels typically encountered in machine learning are amenable to an efficient implementation. In §4, we present an analytic sampling scheme, by way of example on the popular square-exponential kernel, which we generalize to other common kernels, including the Matérn class. We conclude with an empirical study of the approximation quality in an applied setting in §5 and a runtime comparison to exact and MCMC sampling algorithms in §6.

2 DPPs

Let $\ell\colon \mathbb{X}\times\mathbb{X}\to\mathbb{R}$ be a symmetric positive semi-definite kernel over some compact Euclidean space $\mathbb{X}$. Given two sets $\mathrm{A} := [a_1,\dots,a_I], \mathrm{B} := [b_1,\dots,b_J] \subseteq \mathbb{X}$, the symbol $\mathrm{L}_{\mathrm{AB}} \in \mathbb{R}^{I\times J}$ is a matrix containing the elements $[\mathrm{L}_{\mathrm{AB}}]_{ij} = \ell(a_i, b_j)$. For our purposes, a k-DPP is a stochastic process, such that a sample of cardinality k, $\mathrm{X} := [x_1,\dots,x_k] \subset \mathbb{X}$ from the process has joint probability proportional to the determinant of the corresponding Gram matrix L_{XX}:

$$p_{k\text{-DPP}}(\mathbf{X}=\mathrm{X}) = Z\det \mathrm{L}_{\mathrm{XX}}. \tag{1}$$

Here, Z is a normalization constant, the existance of which can be shown via a general argument (Hough et al., 2006; Kulesza and Taskar, 2011). More precise definitions of DPPs and k-DPPs can be found in Soshnikov (2000); Hough et al. (2009); Bardenet and Hardy (2016), and Kulesza and Taskar (2011). They require a discussion of base measures and other properties of point processes, which unnecessarily complicate the exposition in our context. Kulesza and Taskar (2012) also provide a relatively complete introduction to discrete DPPs, where $\mathbb{X}$ is restricted to be a discrete space. In this work, we consider a finite discretization of a continuous space. In order to simplify exposition, we assume that $\mathbb{X}$ is a unit cube $[0,1]^D$ discretized into an equally spaced grid of size N. For a more general box constraint $\tilde{x}_d \in [a_d, b_d]$ for each $d = 1,\dots,D$, one can apply the linear transformations $x_d = {}^{(\tilde{x}_d - a_d)}/_{(b_d - a_d)}$.

3 GREEDY APPROXIMATION

We begin by outlining a connection between the k-DPP and the softargmax of the Gaussian differential entropy in §3.1. This relation serves as the motivation behind our choice of the score function. Then, we greedily maximize this score function, resulting in our greedy approximation of the k-DPP in §3.2. Finally we examine the theoretical properties of the approximation in §3.3.

3.1 Motivation

Consider an algorithm aiming to learn the function $f\colon \mathbb{X}\to\mathbb{R}$ by choosing k evaluation points ("designs") $\mathrm{X}_{1:k}$, using a Gaussian process prior $p(f) = \mathcal{GP}(\mu, \ell)$ with arbitrary mean function $\mu\colon \mathbb{X}\to\mathbb{R}$ and kernel function ℓ as above. The slicing notation $\mathrm{X}_{i:j}$ denotes the elements selected in steps $i,\dots,j$ (for $j < i$, $\mathrm{X}_{i:j} = \varnothing$). We allow for a stochastic and sequential policy π, defined over a product probability space

$(\mathbb{X}^k, 2^{\mathbb{X}^k}, \pi)$:

$$\pi(\mathbf{X}_{1:k}=\mathrm{X}_{1:k})=\prod_{i=1}^{k}\pi\big(\mathbf{X}_i=x_i|\mathbf{X}_{1:i-1}=[x_1,...,x_{i-1}]\big). \tag{2}$$

In many cases, the evaluation order does not matter, that is, one is only interested in the distribution over unordered sets of points $\mathrm{X}\in\mathcal{X}=\{\mathrm{X}\subseteq\mathbb{X}||\mathrm{X}|=k\}$. Every sequential policy π induces a random variable $\mathbf{X}$ over the discrete probability space $(\mathcal{X}, 2^{\mathcal{X}}, p_\pi)$, where p_π is obtained by summing over all permutations perm(X) of the elements in a set X:

$$p_\pi(\mathbf{X}=\mathrm{X}) = \sum_{\mathrm{X}'\in\mathrm{perm}(\mathrm{X})}\pi(\mathbf{X}_{1:k}=\mathrm{X}')\,. \tag{3}$$

Aiming to collect informative observations, assume the algorithm may randomly place evaluations $\mathrm{X}_{1:k}$ such that the differential entropy

$$h_{\mathrm{diff}}(f_{\mathrm{X}_{1:k}}) = \tfrac{1}{2}\log(2e\pi)^k\det\mathrm{L}_{\mathrm{X}_{1:k}\mathrm{X}_{1:k}} \tag{4}$$

of the corresponding multivariate Gaussian $f_{\mathrm{X}_{1:k}}\sim\mathcal{N}(\mu_{\mathrm{X}_{1:k}},\mathrm{L}_{\mathrm{X}_{1:k}\mathrm{X}_{1:k}})$ has a high value. To simplify notation, we drop constants and use

$$h(\mathrm{X}_{1:k}) = \log\det\mathrm{L}_{\mathrm{X}_{1:k}\mathrm{X}_{1:k}} \tag{5}$$

in the following. To be more precise, we require samples to be draws from the *softargmax* of h:

$$p_\beta(x_1,\ldots,x_N) = Z\exp\big(\beta\cdot h(\mathrm{X}_{1:N})\big)\,, \tag{6}$$

where $\beta>0$ is a constant. A policy π for sampling from p_β maximizes the following score function

$$\pi = \arg\max_\pi\ \beta\mathbb{E}_{\mathrm{X}\sim p_\pi}h(\mathrm{X})+\mathcal{H}(p_\pi)\,, \tag{7}$$

where $\mathcal{H}(p_\pi) := -\sum_{\mathrm{X}\in\mathcal{X}}p_\pi(\mathrm{X})\log p_\pi(X)$ is the Shannon entropy and $\mathbb{E}_{\mathrm{X}\sim p_\pi}h(\mathrm{X})$ is the expected value of the objective value h of the sampled subsets. To see this, consider the Kullback–Leibler divergence between p_π and p_β

$$D_{\mathrm{KL}}(p_\pi||p_\beta) = -\mathcal{H}(p_\pi)-\beta\mathbb{E}_{\mathrm{X}\sim p_\pi}h(\mathrm{X})+\log Z;$$
$$Z := \sum_{\mathrm{X}\in\mathcal{X}}\exp\beta h(\mathrm{X}),$$

and note that π achieves the minimium $D_{\mathrm{KL}}(p_\pi\,\|\,p_\beta)=0$ per definition of optimality. For $\beta=1$, one obtains the k-DPP associated with ℓ. This form reveals that a k-DPP is a smooth approximation to the argmax of the differential entropy in the sense that for $\beta\to\infty$, one recovers the exact argmax (and the mode of the k-DPP). The expression $H_\beta(\pi) := \beta\mathbb{E}_{\mathrm{X}\sim p_\pi}h(\mathrm{X})+\mathcal{H}(p_\pi)$ can be interpreted as a *softmax* corresponding to the *softargmax*. Intuitively, this score function rewards high diversity within samples, as well as high diversity between samples. For active learning, the resulting samples are useful, for example, in so far as the resulting empirical estimator $\mathbb{E}_N[f]$ for expectations of f (even if f is *not* a true sample from $\mathcal{GP}(\mu,\ell)$, or even an element of the RKHS associated with ℓ) converges at a rate dominating that of the Monte Carlo estimator as shown in Bardenet and Hardy (2016).

3.2 Method

Finding the exact argmax ($\beta\to\infty$) of the entropy h is known to be NP-hard, but a greedy approximation typically shows good practical performance and is also theoretically well understood (Sharma et al., 2015). The greedy approach to finding the set of points $\mathrm{X}_{1:k}$ with the highest entropy h consists in iteratively selecting the next point x_i by maximizing the marginal gain $\Delta_h(x|\mathrm{X}_{1:i-1}) := h(\mathrm{X}_{1:i-1}\cup\{x\})-h(\mathrm{X}_{1:i-1})$ in each step, i.e.

$$x_i = \operatorname*{argmax}_x\Delta_h(x|\mathrm{X}_{1:i-1})\,. \tag{8}$$

By taking the Schur complement of $\mathrm{L}_{\mathrm{X}_{1:i},\mathrm{X}_{1:i}}$, one obtains

$$\begin{aligned}\det(\mathrm{L}_{\mathrm{X}_{1:i},\mathrm{X}_{1:i}}) &= \det(\mathrm{L}_{\mathrm{X}_{1:i-1},\mathrm{X}_{1:i-1}})\\ &\quad\cdot\det(\mathrm{L}_{x_ix_i}-\mathrm{L}_{x_i\mathrm{X}_{1:i-1}}\mathrm{L}^{-1}_{\mathrm{X}_{1:i-1}\mathrm{X}_{1:i-1}}\mathrm{L}_{\mathrm{X}_{1:i-1}x_i}),\end{aligned}$$

and thereby the marginal gain simplifies to

$$\begin{aligned}&\Delta_h(x|\mathrm{X}_{1:i-1}) =\\ &\qquad\log\Big(\mathrm{L}_{xx}-\mathrm{L}_{x\mathrm{X}_{1:i-1}}\mathrm{L}^{-1}_{\mathrm{X}_{1:i-1}\mathrm{X}_{1:i-1}}\mathrm{L}_{\mathrm{X}_{1:i-1}x}\Big)\,.\end{aligned} \tag{9}$$

The term inside the logarithm will be known to readers experienced with Gaussian processes as the posterior variance of a Gaussian process regression model conditioned on function values at $\mathrm{X}_{1:i-1}$. We will refer to it as $\mathbb{v}_i(x)$ and in preparation for the derivations in §4, we introduce the shorthand $\mathrm{L}_{(i)} := \mathrm{L}_{\mathrm{X}_{1:i-1},\mathrm{X}_{1:i-1}}$ and re-formulate the posterior variance more explicitly as

$$\mathbb{v}_i(x) = \ell(x,x)-\sum_{a,b=1}^{i-1}\ell(x,x_a)\,\ell(x,x_b)[\mathrm{L}^{-1}_{(i)}]_{ab}\,. \tag{10}$$

In analogy to greedy optimization, we define the greedy approximation to the k-DPP by sampling iteratively from the softargmax of the marginal gain $\Delta_h(x\,|\,\mathrm{X}_{1:i-1})$:

$$\pi_{\mathrm{greedy}}(x\,|\mathrm{X}_{1:i-1}) = \frac{\exp\Delta_h(x\,|\,\mathrm{X}_{1:i-1})}{Z_{\mathrm{X}_{1:i-1}}}\propto\mathbb{v}_i(x). \tag{11}$$

$Z_{\mathrm{X}_{1:i-1}}$ denotes the normalizing constant, that depends on the previously selected points $\mathrm{X}_{1:i-1}$. Alternatively, one can view the greedy approximation to sampling from a k-DPP as the greedy approximation to the optimum of the score function $H(\pi) := \mathbb{E}_{\mathrm{X}_{1:k}\sim\pi} h(\mathrm{X}_{1:k}) + \mathcal{H}(\pi)$ since

$$\pi_{\text{greedy}}(x \,|\, \mathrm{X}_{1:i-1}) = \operatorname*{argmax}_{\pi(x|\mathrm{X}_{1:i-1})} \mathbb{E}_{\pi(x|\mathrm{X}_{1:i-1})}\big[\Delta_h(x \,|\, \mathrm{X}_{1:i-1})\big] + \mathcal{H}(\mathbf{X}_i \mid \mathbf{X}_{1:i-1} = \mathrm{X}_{1:i-1}) \quad (12)$$

is equivalent to Eq. (11) by the same argument based on the Kullback–Leibler divergence as in the previous section. Here, $\mathcal{H}(\mathbf{X}_i \mid \mathbf{X}_{1:i-1} = \mathrm{X}_{1:i-1})$ denotes the conditional Shannon entropy.

3.3 Approximation Guarantees

In this section we derive the approximation guarantee for the greedy optimization of the above score function H. It is based on a classic result from combinatorial optimization giving a $(1-1/e)$-approximation ratio between greedy and optimal maximization of monotone submodular set functions (Nemhauser et al., 1978). We extend this guarantee to a stochastic setting and show that it holds for our greedy k-DPP sampling algorithm. For technical reasons, we introduce a free parameter α during this analysis that we will later fix to a convenient value, leaving us with an approximation ratio that depends on k and the spectral properties of the kernel of the k-DPP.

The function $h(\mathrm{X}) = \log\det \mathrm{L}_{\mathrm{XX}}$ has two helpful characteristics, as pointed out by Krause et al. (2008) and Sharma et al. (2015). It is submodular, i.e. $\forall\, \mathrm{X}_1 \subseteq \mathrm{X}_2$ and $i \notin \mathrm{X}_2$, $h(\mathrm{X}_1 \cup \{i\}) - h(\mathrm{X}_1) \geq h(\mathrm{X}_2 \cup \{i\}) - h(\mathrm{X}_2)$. If additionally, the smallest eigenvalue $\lambda_{min}(\mathrm{L}) \geq 1$, the function h is also monotone, i.e. $\forall\, \mathrm{X}_1, \mathrm{X}_2$ with $\mathrm{X}_1 \subseteq \mathrm{X}_2 \subseteq \mathrm{X}, h(\mathrm{X}_1) \leq h(\mathrm{X}_2)$. In combination, monotonicity and submodularity bound the future change in the objective value h in subsequent steps by the previous change. Nemhauser et al. (1978) used this property to give an upper bound on the optimal objective value $h(\mathrm{O})$ based on $(1-1/e)^{-1}$ times the objective value $h(\mathrm{G})$ found with the greedy algorithm:

Theorem (Nemhauser et al. (1978)) *Given a monotone submodular function h, let G be the solution found with the greedy algorithm as defined in Eq. (8) and O be the optimal solution. It holds:*

$$(1-1/e)h(O) \leq h(G)\,.$$

In our case, where we treat the k-DPP as the soft-argmaximum of h, we derive an analogous statement in terms of the softmaximum instead of the maximum:

Theorem 1 *Let h be a submodular set function with $h(\varnothing) = 0$ and $\Delta_h(x \mid X) > (1/k)\log k!$ for all $X \subset \mathbb{X}$, $x \in \mathbb{X} \setminus X$. Assume $\mathbb{X}$ is finite. It holds*

$$(1-1/e)H(p_{\pi_{opt}}) \leq H(p_{\pi_{greedy}})\,,$$

with $H(p_\pi) = \mathbb{E}_{X\sim p_\pi}[h(X)] + \mathcal{H}(p_\pi)$ and π_{opt} being an optimal policy.

A full proof is included in Appendix 4. The idea is to first prove the desired results for the sequential, order-dependent policies, i.e. $(1-1/e)H(\pi_{\text{opt}}) \leq H(\pi_{\text{greedy}})$. This can be done by following the series of arguments in Nemhauser et al. (1978) with a replacement of sets by (ordered) set-valued random variables and additional care of the Shannon entropy terms. Then, we transfer the result to the final distribution over unordered sets by exploiting that the order-dependent bound holds for all order-dependent optimal policies, including the ("worst case") one with uniform distribution over all permutations.

In order for the latter step to work out, we introduce the additional requirement $\Delta_h(x \mid \mathrm{X}) > (1/k)\log k!$ for all $\mathrm{X} \subset \mathbb{X}$, $x \in \mathbb{X} \setminus \mathrm{X}$ regarding the slope of h. However, note that altering the slope of h can be done easily by scaling the kernel function with a constant value. To make this dependence explicit, we use $h_\alpha(\mathrm{X}) = \log\det \alpha \mathrm{L}_{\mathrm{XX}}$ instead of h for the following analysis. In optimization, as well as in the sampling case, the greedy strategy is invariant with respect to this change because the scale α cancels out when sampling proportionally to the posterior variance. The distribution of samples from a random-sized DPP, though, changes in general. In particular, increasing α increases the expected cardinality of the samples. But for k-DPPs, the scaling again does not matter. For them, we can therefore give the following approximation guarantee:

Corollary 1 *Running the algorithm $\pi_{greedy}(x \,|\, X_{1:i-1}) \propto \mathbb{v}_i(x)$ as introduced in Section 4 for k iterations on a finite grid is a $(1-1/e)$ approximation to the exact distribution $p_{k\text{-}DPP}$ of the corresponding k-DPP, in the sense of*

$$(1-1/e)H_\alpha(p_{k\text{-}DPP}) \leq H_\alpha(p_{\pi_{greedy}}),$$

with $H_\alpha(p) = \mathbb{E}_{X\sim p}[h_\alpha(X)] + \mathcal{H}(p)$, $h_\alpha(X) = \log\det(\alpha L_{XX})$ and $\alpha > k!^{1/k}/\lambda_{min}$, where λ_{min} is the smallest eigenvalue of the Kernel Gram matrix over the grid.

This result follows directly from Theorem 1. For details, see Appendix 4. By plugging in $\alpha = k!^{1/k}/\lambda_{min}$, rearranging the terms and additionally applying Strin-

gling's approximation for $k!$, the above inequality reads

$$\begin{aligned}(1-1/e)H_1(p_{k\text{-DPP}}) - H_1(p_{\pi_{\text{greedy}}}) \\ \leq (1/e)k\log(\alpha) \\ \leq (1/e)\big(k\log(k) - k + \mathcal{O}(\log k) + k\cdot\log\lambda_{\min}^{-1}\big)\end{aligned}$$

This form reveals that the tightness of the bound increases for a larger smallest eigenvalue $\lambda_{\min}^{-1}$ and a smaller number of points k. In the case of a DPP with random k, larger eigenvalues lead to a higher expected number of sampled points. Therefore, one possible intuition for this result is that in settings in which the sample space volume is not "very tightly filled" (i.e., if k is much less than the expected number of sampled points under the DPP), the problem of placing k self-avoiding points might become "easier" and the greedy approximation can be very close to the exact algorithm. The subtle differences between greedy and exact only matter if the volume is very "packed" (note that this does *not* imply that exact and approximate methods are similar to iid. samples in such "loose" cases. They still self-avoid).

A quantity closely related to the differential entropy h is the information gain. Since it is also known to be submodular and monotone, a $(1-1/e)$-approximation guarantee for greedy sampling from the softargmax of the information gain holds as well. Please refer to Appendix 4 for the corresponding statement. It is restricted to order-dependent sampling and requires a slight modification of the sampling scheme presented in §4 to take the noise term σ^2 into account. The modification does not impede the efficiency of the method and for $\sigma \twoheadrightarrow 0$, it corresponds to the algorithm introduced above.

Besides the $(1-1/e)$ approximation guarantees on monotone submodular set functions, it is also known that the greedy approach is optimal on matroidal structures, and Lyons (2003) pointed out the close relationship between matroids and orthogonal projection DPPs. This special kind of DPPs is charaterized by all eigenvalues of the correlation kernel matrix $\mathrm{K} = \mathrm{L}(\mathrm{I}+\mathrm{L})^{-1}$ being zero or one. The cardinality of the samples is then deterministic. For orthogonal projection DPPs, following the greedy approach and sampling iteratively from the posterior variance is known to result in the exact distribution (Hough et al., 2006).

As a final remark, note that the above statements claim nothing about the *similarity of the distributions* itself (e.g. in the sense of a total variation distance of the probability masses). Instead, they state that the two sampling distributions achieve a *similar performance in the task of generating diverse sets* – as quantified by the two entropy terms. While the latter is typically what one cares about in practical applications of DPP sampling, the former can be interesting future work to gain more theoretical insight into the elegant nature of DPPs.

4 EFFICIENT IMPLEMENTATION

From a computational perspective, sampling proportionally to the posterior variance $\mathbb{v}_i(x)$ in Eq. (11) poses two challenges. First, for general kernels ℓ, there is usually no analytic cumulative density function for them which is required to efficiently sample the next point. However, this problem is much less severe in machine learning, because our community enjoys freedom in the design of models and can thus choose kernels with convenient analytic properties. Choosing such a kernel and exploiting these properties directly yields an efficient approximation for the generation of samples from k-DPPs, even in high-dimensional domains.

Second, even if the kernel is analytically convenient, the calculation of the posterior variance $\mathbb{v}_i(x)$ involves the matrix inverse of $\mathrm{L}_{(i)}$. Given the inverse of $\mathrm{L}_{(i-1)}$ from the preceding step in the iterative sampling scheme, this inverse can be computed with complexity $\mathcal{O}\big((i-1)^2\big)$, using the matrix inversion lemma. Even so, the cost of drawing a sample of cardinality k remains cubic in k. This issue is directly connected to inference in Gaussian process regression models, and many approximations have been proposed over the past decade. Furthermore, in use cases like Bayesian optimization and quadrature, the number k of function evaluations is often low, and a decomposition of the Gram matrix is computed anyway. In such cases, the cubic cost can be unproblematic, and scaling to larger domains (high N and D) may be more important.

We provide an efficient implementation of the greedy principle for k-DPP sampling if the kernel ℓ is analytically integrable. To ease intuition, the derivations will be by way of example, using the popular square-exponential kernel $\ell_{\text{SE}}\colon \mathbb{R}^D \times \mathbb{R}^D \twoheadrightarrow \mathbb{R}$ over the real vector space

$$l_{\text{SE}}(x_a, x_b) = \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_a - x_b)_d^2}{\lambda_d^2}\right). \qquad (13)$$

To simplify things even further, we initially consider the univariate problem, $D = 1$, then generalize to arbitrary dimensionality. The resulting algorithm draws a k-sized sample at cost $\mathcal{O}(Dk^3\log(N))$. A general form of the algorithm is summarized in Algorithm 1 in Appendix 2. There, we also provide a more detailed runtime analysis and describe how to extend the scheme to other popular kernels, like the Matérn class.

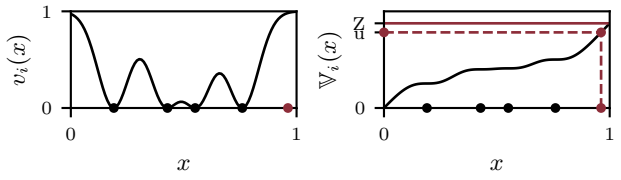

Figure 2: Sketch illustrating analytic sampling from a k-DPP in one dimension, using the square-exponential kernel (Eq.13). Left: Posterior variance $\mathbb{v}_i(x)$ after the 4th iteration ($i = 5$). Right: Inverse transform sampling from $\mathbb{v}_i(x)$, by computing the cumulative density $\mathbb{V}_i$ (black line), drawing a scaled uniform random sample u and finding the point x_i such that $\mathbb{V}_i(x_5) = u$, by interval bisection.

4.1 Sampling in One Dimension

We may draw samples using the classic form of computing a non-normalized cumulative density

$$\mathbb{V}_i(x) = \int_0^x \mathbb{v}_i(\tilde{x})\,\mathrm{d}\tilde{x}\,, \tag{14}$$

and transforming standard uniform random variables $u \sim \mathbb{U}\big[0, \mathbb{V}_i(1)\big]$, produced by a pseudo-random number generator, into exact samples from $\mathbb{v}_i$ (cf. Figure 2), by setting

$$x = \mathbb{V}_i^{-1}(u) = \{x \,|\, \mathbb{V}_i(x) = u\}\,. \tag{15}$$

For the univariate square-exponential exponential kernel Eq.(13), (10) can be re-written, using standard properties of the Gaussian function, as

$$\mathbb{v}_i(x) = 1 - \sum_{a,b=1}^{i-1} \exp\left(-\frac{(x-m_{ab})^2}{\lambda^2}\right) \cdot \underbrace{\exp\left(-\frac{(x_a-x_b)^2}{4\lambda^2}\right)}_{=:M_{(i),ab}} [\mathrm{L}_{(i)}^{-1}]_{ab}\,. \tag{16}$$

where $m_{ab} := {}^1\!/_2(x_a + x_b)$, and we have defined a matrix $\mathrm{M}_{(i)} \in \mathbb{R}^{(i-1)\times(i-1)}$. The variables $m, \mathrm{M}, \mathrm{L}^{-1}$ provide the statistics of the sample needed to draw the subsequent point. After x_i has been drawn, these three variables can be updated in $\mathcal{O}(k^2)$—we use the matrix inversion lemma to update $\mathrm{L}_{(i+1)}^{-1}$; the other two variables can be updated in $\mathcal{O}(k)$. The cumulative density is then

$$\mathbb{V}_i(x) = x - \frac{\sqrt{\pi}\lambda}{2} \sum_{a,b=1}^{i-1} \left[\mathrm{erf}\left(\frac{x-m_{ab}}{\lambda}\right) + \mathrm{erf}\left(\frac{m_{ab}}{\lambda}\right)\right] [\mathrm{M}_{(i)} \odot \mathrm{L}_{(i)}^{-1}]_{ab}\,. \tag{17}$$

Here, $\odot$ is the Hadamard (element-wise) product, and we have used $\mathrm{erf}(x) = -\,\mathrm{erf}(-x)$. Given a uniform random draw u, all that is left to do is to find x such that $\mathbb{V}_i(x) = u$. A straightforward, numerically robust, albeit not particularly elegant way to do so is by interval bisection, which takes $\frac{1}{D}\log(N)$ steps of costs $\mathcal{O}(k^2)$. A more elegant search strategy could be constructed using grid refinement methods similar to the popular Ziggurat algorithm of Marsaglia and Tsang (2000).

4.2 Multivariate Samples

For square-exponential exponential kernel k-DPPs in dimension $D > 1$, $\mathbb{v}_i(x)$ retains much of its structure. Equation (16) simply turns into (defining the elements of a new matrix $\mathbf{M} \in \mathbb{R}^{(i-1)\times(i-1)}$ analogous to M in Eq. (16)):

$$\mathbb{v}_i(x) = 1 - \sum_{a,b=1}^{i-1} \exp\left(-\sum_{d=1}^{D} \frac{(x-m_{ab})_d^2}{\lambda_d^2}\right) \cdot \underbrace{\exp\left(-\sum_{d=1}^{D} \frac{(x_a-x_b)_d^2}{4\lambda_d^2}\right)}_{=:\mathbf{M}_{(i),ab}} [L_{(i)}^{-1}]_{ab}\,. \tag{18}$$

The additional challenge in this multivariate case is to construct a parametrization of the cumulative density $\mathbb{V}_i$. This step, too, can be performed in an iterative fashion, drawing one coordinate of the sample point x_i after another (cf. Figure 3). Given that the first $d-1$ elements of x_i are given by $x_{i,1:d-1}$, the cumulative density associated with the d-th dimension is given by the sum rule:

$$\mathbb{V}_i(x_{i,d}|x_{i,1:d-1}) = \int_0^{x_{i,d}} \int \cdots \int_0^1 \mathbb{v}_i\big([x_{i,1:d-1}, \tilde{x}_{i,d}, \tilde{x}_{i,d+1:D}\big)\,\mathrm{d}\tilde{x}_{i,d} \prod_{\tilde{d}=d+1}^{D} \mathrm{d}\tilde{x}_{i,\tilde{d}}\,. \tag{19}$$

For the square-exponential exponential kernel, this works out to

$$\mathbb{V}_i(x_{i,d} = x \,|\, x_{i,1:d-1}) = x - \sum_{a,b=1}^{i-1} \left\{ \exp\left(-\sum_{r=1}^{d-1} \frac{(x-m_{ab})_r^2}{\lambda_r^2}\right) [\mathbf{M}_{(i)} \odot \mathrm{L}_{(i)}^{-1}]_{ab} \cdot \left(\mathrm{erf}\left(\frac{[x_i - m_{ab}]_d}{\lambda_d}\right) + \mathrm{erf}\left(\frac{[m_{ab}]_d}{\lambda_d}\right)\right) \frac{\sqrt{\pi}\lambda_l}{2} \cdot \left(\prod_{j=d+1}^{D} \left(\mathrm{erf}\left(\frac{[1-m_{ab}]_j}{\lambda_j}\right) + \mathrm{erf}\left(\frac{[m_{ab}]_j}{\lambda_j}\right)\right) \frac{\sqrt{\pi}\lambda_j}{2}\right)\right\}. \tag{20}$$

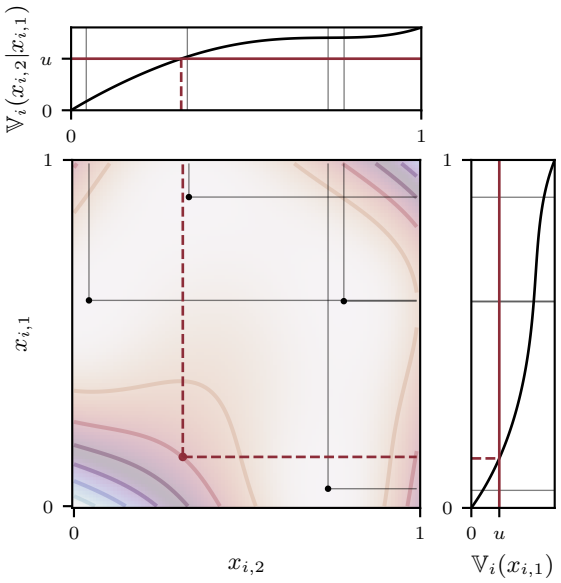

Figure 3: Drawing approximate k-DPP samples in two dimensions. Bottom left: Contour plot of the multivariate probability density $\mathbb{v}_i(x)$ with $(i = 5)$; preceding four samples as black points. Right: The first coordinate of the fifth sample is drawn first, from the marginal density along this dimension. Top: The second coordinate is then drawn by computing a cumulative density *conditioned* on the value of the first coordinate.

5 EXPERIMENTS

We conduct an empirical analysis of the approximation quality in an applied setting. If greedily sampled locations cover the domain almost as well as the exactly sampled ones, one would expect to learn a similar amount of information about a function (modeled with a Gaussian Process) evaluated at these locations. This should lead to little performance decrease in follow-up tasks, such as the integration of that function. To investigate this, we perform Bayesian Quadrature (BQ) of several benchmark functions with evaluation locations sampled from exact and approximate k-DPPs, as well as uniformly chosen locations. For the BQ we rely on the vanilla version from `emukit` [1] (Paleyes et al., 2023). As integrands we use the benchmark functions from The Virtual Library of Simulation Experiments [2] (Surjanovic and Bingham). We use a square-exponential kernel with a default lengthscale of 0.2 for all methods and benchmarks. All functions are integraded over the domain $\mathbb{X} = [0, 1] \times [0, 1]$. Evaluation locations are sampled from a regular 50×50 grid. For more details on the methods and hyperparameters, see Appendix 5. The code for all experiments in this paper is available at https://github.com/JuliaGrosse/GreedykDPPSampling.

Figure 4 shows the mean error between the true value $F := \int_{x \in \mathbb{X}} f(x)dx$ and the value $\hat{F}$ estimated with BQ for three of them. Figure 1 in Appendix 5 contains the results for all twelve benchmark functions. On the benchmarks, where there was a significant advantage of the DPP over the uniform distribution, the greedy version performed equally well to the DPP. This indicates that sampling from the exact k-DPP might not be very crucial in an application like this, as long as some repulsiveness is still present.

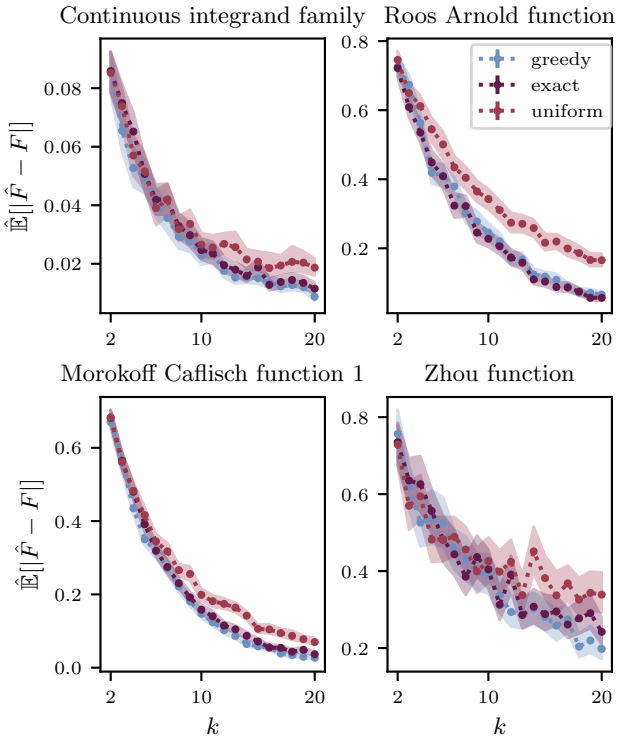

Figure 4: Results from BQ with evaluation locations sampled from an exact k-DPP, the greedy approximation and uniformly sampled locations. The plots show the mean error over 100 samples and 95% confidence intervals for the mean error as shaded areas.

6 RELATED WORK

The maximum entropy formulation we use as starting point for our derivations is prominent in reinforcement learning. For an introduction to *Maximum Entropy Reinforcement Learning* see e.g. (Levine, 2018). These methods are used to learn a policy π that satisfies the objective in equation 7 for an arbitrary function h. Our method differs from this line of work in that we do not learn π, but instead give an analytic approx-

[1] https://emukit.github.io/bayesian-quadrature/
[2] https://www.sfu.ca/~ssurjano/integration.html

imation for π from scratch for the specific choice of $h(X)$ being the Gaussian entropy itself (equation 5).

Regarding the theoretical analysis, the work from Djolonga et al. (2018) is closest to ours. They also prove a $(1 - 1/e)$ guarantee of a greedy algorithm on a log partition function. Instead of monotonicity and submodularity, they assume that h is a sum of $M^\natural$-concave functions. In addition, their greedy algorithm is a variational approximation and not analytically derived as in our case. The paper by Hough et al. (2006, Prop. 19) contains a special case of our greedy algorithm for orthogonal projection DPPs (where it is exact). A greedy approximation for maximum a posteriori inference in DPPs is suggested in Chen et al. (2018).

Chen et al. (2022) sample proportionally to the posterior variance of a GP – without the inverse transform sampling – in the context of quadrature and low-rank matrix approximations. They provide bounds on the expected trace $\mathbb{E}[tr(A-A_k)]$, where A_k is the sampled low-rank approximation of the matrix A. Epperly and Moreno (2023) further analysed the greedy sampling scheme – specifically for quadrature – and bound the approximation error on the integrand itself. Similarly, Huszár and Duvenaud (2012) and Adachi et al. (2022) studied the convergence properties of the method in the quadrature setting. However, our analysis in terms of the entropies draws a novel connection to k-DPPs, providing a theoretical justification for why it can be seen as an approximation of them.

Due to the large amount of recent literature on DPPs in general, we restrict the remainder of this section to an overview of exact and approximate sampling from k-DPPs only. Originally, exact samplers for generic k-DPPs were based on eigendecompositions of the entire $N \times N$ Kernel Gram matrix and thereby required $\mathcal{O}(N^3)$ time (Kulesza and Taskar, 2011). Derezinski et al. (2019) introduced DPP-VFX, an intermediate sampling method for k-DPPs. They first sample intermediate points from the marginal distributions of a random-sized DPP, and then repeatedly sample from a DPP restricted to the intermediate points until a set of size k is sampled. The algorithm has time complexity in $\mathcal{O}(N \cdot k^{10} + k^{15})$. The linear costs in N can be reduced to less than linear costs by another intermediate sampler named α-DPP (Calandriello et al., 2020), that does not require the computation of all marginals and additionally uses a more efficient reduction method from the DPP to the k-DPP. The resulting complexity is reported to be in $\mathcal{O}((\beta N \cdot k^6 + k^9)\sqrt{k})$, making α-DPP to the best of our knowledge the currently fastest exact sampler for k-DPPs. The constant $\beta \leq 1$ depends on the effective dimension d_{eff} of the matrix L, the sample size k, as well as the largest entry κ^2 in L and can be specified further to $\beta \leq \min\{k^2\kappa^2/d_{\text{eff}}(L), 1\}$.

Regarding approximate sampling, MCMC methods are popular. Anari et al. (2016) introduced one that runs in $\mathcal{O}(\text{poly}(k)\, n \log(n/\epsilon))$. Here, ϵ is a small constant determining the approximation quality of the MCMC samples. The approximation guarantees for MCMC only hold after the mixing time of $\mathcal{O}(n \cdot \text{poly}(k))$. Transitions steps in the Markov Chain take time polynomial in k. Rezaei and Gharan (2019) developed a k-DPP sampler for continuous domains. As such its runtime does not depend on N, however, it involves rejection sampling from the conditionals of the k-DPP, which can become expensive in k. If $k \leq e^{D^{1-c}}$ for some constant $0 \leq c \leq 1$, one can show that the time complexity is in $\mathcal{O}(D \log(1/\epsilon)) \cdot k^{\mathcal{O}(1/c)}$. Based on the asymptotic runtimes, the greedy algorithm in this paper compares favorably if exactness is not absolutely crucial and the domain is large or high-dimensional.

6.1 Runtime Comparison

We compare the runtime of the greedy algorithm with those of several state-of-the-art exact and approximate samplers (Derezinski et al., 2019; Calandriello et al., 2020; Anari et al., 2016) described above (Figure 6.1). For the baselines, we use the Python implementations available in DPPy[3] (Gautier et al., 2019b) with default parameters. We draw 100 samples with $k = 10$ and $k = 100$ points from a k-DPP with square-exponential kernel with lengthscale 0.01, respectively 0.001, on the interval $[0, 1]$. We run all methods for discretization size $N \in \{10^2, 10^3, 10^4, 5 \cdot 10^4, 6 \cdot 10^4, 7 \cdot 10^4, 10^5\}$. Runs taking longer then 100 seconds for $k = 10$ or 360 seconds for $k = 100$ were stopped, i.e. they do not appear in the figure. For $k = 10$, the experiment was repeated 5 times and for $k = 100$, 3 times. Additional results from a repetition of the experiment on $[0, 1]^3$ with lower discretization sizes are included in Appendix 5.The results agree with those from the asymptotic runtime analyses.

[3] https://github.com/guilgautier/DPPy

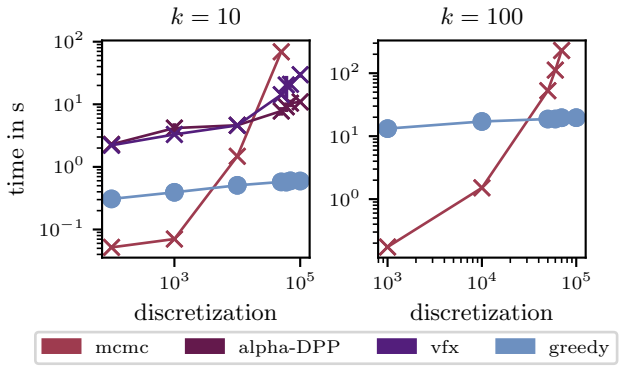

Figure 5: Runtime comparison of the greedy algorithm with state-of-the-art baselines. Discussion in text.

7 CONCLUSION

We introduced a greedy approximation for sampling from k-DPPs theoretically grounded in an analogy to greedy optimization of the Gaussian differntial entropy. We showed that approximation guarantees for greedy optimization of the entropy have a resembling interpretation in the sampling setting and tested the approximation in a BQ application, where we found the approximation error to be empirically negligible. We provided an efficient implementation for continuous domains – logarithmic in the size of the discretization N – for common kernels like the Matérn class. The algorithm described herein thus offers itself as a low-level routine for all applications that require such diverse points sets as part of a surrounding experimental design loop.

Acknowledgements

The authors thank the reviewers for useful comments and additional references. JG thanks Cheng Zhang and Marvin Pförtner for helpful discussions. The authors are grateful to Lucy Kuncheva and Joseph Courtney for (separately) pointing out a nontrivial typo in Eq. (20) in an earlier version of this manuscript, as well as to Simon Barthelmé for pointing out the approximate nature of the sampling scheme. This work was supported by Microsoft Research through its PhD Scholarship Programme. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting JG. PH and JG gratefully acknowledge financial support by the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. RG was supported by the National Science Foundation under award IIS-1845434

References

Masaki Adachi, Satoshi Hayakawa, Martin Jørgensen, Harald Oberhauser, and Michael A Osborne. Fast bayesian inference with batch bayesian quadrature via kernel recombination. *Advances in Neural Information Processing Systems*, 35:16533–16547, 2022.

Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte Carlo Markov chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes. In *Conference on Learning Theory*, pages 103–115. PMLR, 2016.

Rémi Bardenet and Adrien Hardy. Monte Carlo with determinantal point processes. *ArXiv e-print*, 1605.00361, May 2016.

Ayoub Belhadji, Rémi Bardenet, and Pierre Chainais. Kernel quadrature with DPPs. *Advances in Neural Information Processing Systems*, 32, 2019.

Daniele Calandriello, Michal Derezinski, and Michal Valko. Sampling from a k-DPP without looking at all items. *Advances in Neural Information Processing Systems*, 33:6889–6899, 2020.

Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. *Advances in Neural Information Processing Systems*, 31, 2018.

Yifan Chen, Ethan N Epperly, Joel A Tropp, and Robert J Webber. Randomly pivoted cholesky: Practical approximation of a kernel matrix with few entry evaluations. *arXiv preprint arXiv:2207.06503*, 2022.

Michal Derezinski, Daniele Calandriello, and Michal Valko. Exact sampling of determinantal point processes with sublinear time preprocessing. *Advances in neural information processing systems*, 32, 2019.

Josip Djolonga, Stefanie Jegelka, and Andreas Krause. Provable variational inference for constrained log-submodular models. *Advances in Neural Information Processing Systems*, 31, 2018.

Mohamed Elfeki, Camille Couprie, Morgane Riviere, and Mohamed Elhoseiny. Gdpp: Learning diverse generations using determinantal point processes. In *International conference on machine learning*, pages 1774–1783. PMLR, 2019.

Ethan N Epperly and Elvira Moreno. Kernel quadrature with randomly pivoted cholesky. *arXiv preprint arXiv:2306.03955*, 2023.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

Guillaume Gautier, Rémi Bardenet, and Michal Valko. On two ways to use determinantal point processes for Monte Carlo integration. *Advances in Neural Information Processing Systems*, 32, 2019a.

Guillaume Gautier, Guillermo Polito, Rémi Bardenet, and Michal Valko. DPPy: DPP Sampling with Python. *Journal of Machine Learning Research - Machine Learning Open Source Software (JMLR-MLOSS)*, 2019b. URL http://jmlr.org/papers/v20/19-179.html. Code at http://github.com/guilgautier/DPPy/ Documentation at http://dppy.readthedocs.io/.

Philipp Hennig and Christian J Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13(6), 2012.

John Ben Hough, Manjunath Krishnapur, Yuval Peres, and Bálint Virág. Determinantal processes and independence. *Probability Surveys*, 3:206–229, 2006.

John Ben Hough, Manjunath Krishnapur, Yuval Peres, and Bálint Virág. *Zeros of Gaussian analytic functions and determinantal point processes*, volume 51, University Lecture Series. American Mathematical Society Providence, RI, 2009.

Ferenc Huszár and David Duvenaud. Optimally-weighted herding is bayesian quadrature. *arXiv preprint arXiv:1204.1664*, 2012.

Byungkon Kang. Fast determinantal point process sampling with application to clustering. *Advances in Neural Information Processing Systems*, 26, 2013.

Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched Gaussian process bandit optimization via determinantal point processes. *Advances in neural information processing systems*, 29, 2016.

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9 (2), 2008.

Alex Kulesza and Ben Taskar. k-DPPs: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5:123–286, 2012.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Russell Lyons. Determinantal probability measures. *Publications Mathématiques de l'IHÉS*, 98:167–212, 2003.

Chao Ma, Sebastian Tschiatschek, Konstantina Palla, José Miguel Hernández-Lobato, Sebastian Nowozin, and Cheng Zhang. Eddi: Efficient dynamic discovery of high-value information with partial vae. *arXiv preprint arXiv:1809.11142*, 2018.

Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, pages 83–122, 1975.

Zelda Mariet and Suvrit Sra. Diversity networks: Neural network compression using determinantal point processes. *arXiv preprint arXiv:1511.05077*, 2015.

George Marsaglia and Wai Wan Tsang. The Ziggurat method for generating random variables. *Journal of Statistical Software*, 5(8):1–7, 2000.

Madan Lal Mehta. *Random Matrices*. Academic Press, 1991.

Elvis Nava, Mojmir Mutny, and Andreas Krause. Diversified sampling for batched Bayesian optimization with determinantal point processes. In *International Conference on Artificial Intelligence and Statistics*, pages 7031–7054. PMLR, 2022.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Andrei Paleyes, Maren Mahsereci, and Neil D. Lawrence. Emukit: A Python toolkit for decision making under uncertainty. *Proceedings of the Python in Science Conference*, 2023.

Alireza Rezaei and Shayan Oveis Gharan. A polynomial time MCMC method for sampling from continuous determinantal point processes. In *International Conference on Machine Learning*, pages 5438–5447. PMLR, 2019.

Dravyansh Sharma, Ashish Kapoor, and Amit Deshpande. On greedy maximization of entropy. In *International Conference on Machine Learning*, pages 1330–1338. PMLR, 2015.

Alexander Soshnikov. Determinantal random point fields. *Russian Mathematical Surveys*, 55(5):923–975, 2000.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved October 10, 2023, from http://www.sfu.ca/~ssurjano.

Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional Bayesian optimization via structural kernel learning. In *International Conference on Machine Learning*, pages 3656–3664. PMLR, 2017.

Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H Chi, and Jennifer Gillenwater. Practical diversified recommendations on youtube with determinantal point processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2165–2173, 2018.

Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] mathematical setting: Section 2.1; assumptions: finite domain (mentioned in text and at the beginning of each theorem) + analytically integrable kernel for the efficient implementation (mentioned in Section 3) algorithm: description in Section 3 and pseudocode in Appendix Section 2
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable] time: Section 3.1 and more details in Appendix Section 3, space: No, sample size: Not Applicable
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] Code is available at `https://github.com/JuliaGrosse/GreedykDPPSampling`, see README therein for dependencies etc.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] At the beginning of each theorem.
   (b) Complete proofs of all theoretical results. [Yes] Appendix Section 4
   (c) Clear explanations of any assumptions. [Yes] assumptions: finite domain (mentioned in text and at the beginning of each theorem) + analytically integrable kernel for the efficient implementation (mentioned in Section 3). We additionally included examples for analytically integrable kernels.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] Code is available at `https://github.com/JuliaGrosse/GreedykDPPSampling`, see README therein for how to generate the data and figures.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] Section 5, Section 6.1, Appendix Section 5
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] Section 6.1 and Caption of Figure 4
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] Appendix Section 5

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] `The Virtual Library of Simulation Experiments` ,`DPPy` and `emukit`: see references.
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [No]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Greedy Approximation for k-Determinantal Point Processes Supplementary Materials

The appendix contains a description of how the algorithm introduced in the main text for the square exponential kernel can be extended to other analytically integrable kernels in Section 1, a description of the greedy sampling algorithm in pseudocode in Section 2, a runtime analysis in Section 3, the proofs of the theoretical results for Section 4 of the main paper in 4, and the additional experimental details in Section 5.

1 OTHER ANALYTICAL KERNELS

While the Gaussian kernel is the most widely used kernel in machine learning, it has some shortcomings, primarily that it makes very strong smoothness assumptions that can lead to instability in interpolation models. But with some algebraic elbow grease, the scheme of Eq. 19 can be extended to many other popular kernels, assuming they factorize,

$$l(a, b) = \prod_{d}^{D} l(a_d, b_d), \tag{1}$$

and the indefinite integrals

$$\int l(a, a)\,\mathrm{d}a \qquad \text{and} \qquad \int l(a, b)l(a, c)\,\mathrm{d}a \tag{2}$$

are analytically solvable. For example, the above results are applicable to the Matérn class of kernels (Stein, 1999) (including the exponential kernel, which induces the Ornstein–Uhlenbeck process), noting that, assuming w.l.o.g. $x_0 < a < b < x_1$,

$$\int_{x_0}^{x_1} \exp\left(-|x-a|\right) \exp\left(-|x-b|\right)\,\mathrm{d}x = \frac{e^{-a-b}}{2}\left(e^{2a} - e^{2x_0}\right) + (b-a)e^{a-b} + \frac{e^{a+b}}{2}\left(e^{2x_1} - e^{2b}\right), \tag{3}$$

and using results such as (see e.g., Gradshteyn and Ryzhik, 2007, §2.322)

$$\int xe^{ax}\,\mathrm{d}x = e^{ax}\left(\frac{x}{a} - \frac{1}{a^2}\right), \tag{4}$$

$$\int x^2e^{ax}\,\mathrm{d}x = e^{ax}\left(\frac{x^2}{a} - \frac{2x}{a^2} + \frac{2}{a^3}\right), \ldots \tag{5}$$

2 PSEUDOCODE

Algorithm .1 Greedy sampling from approximate DPPs with analytic kernels, on $[0,1]^D$.

procedure DRAWFROMDPP(l, D, k)
  // initialize statistics of sample as empty
  $X \leftarrow \varnothing, m \leftarrow \varnothing, M \leftarrow \varnothing, L^{-1} \leftarrow \varnothing$
  for $i = 1, \ldots, k$ **do** // draw samples iteratively
    $x_i \leftarrow \varnothing$ // initialize current sample point
    // draw dimensions iteratively
    for $d = 1, \ldots, D$ **do**
      // construct function for Eq. (20)
      $\mathbb{V} \leftarrow \mathbb{V}$CONSTRUCT$(m, M, L^{-1})$
      // draw scaled unit random number
      $u \leftarrow \mathbb{V}(1) \cdot$RAND$(\cdot)$
      $I \leftarrow [0,1]$ // initialize search interval
      while $|I| > \varepsilon$ **do** // bisection search
        $\mu \leftarrow 1/2(I_0 + I_1)$ // interval midpoint
        $I \leftarrow$ **if** $(\mathbb{V}(\mu) < u)$
        then $[\mu, I_1]$ **else** $[I_0, \mu]$ // bisect
      end while
      $x_i \leftarrow [x_i, \mu]$ // store sampled element
    end for
    // update sample statistics (rank-1)
    $(X, m, M, L^{-1}) \leftarrow$ADD$(x_i \,|\, X, m, M, L^{-1})$
  end for
end procedure

3 RUNTIME

Let k be the number of points to select, D the dimensionality, N the size of the discretization (regular grid). Then the runtime of the greedy sampling algorithm is $\mathcal{O}(k^3 \log N)$:

- The Gaussian posterior is updated once in each of the k iterations in $\mathcal{O}(k^2)$ by using the matrix-inversion-lemma (see eg. the textbook on GPs by Rasmussen & Williams, A.3)

- In one iteration, along each of the D dimensions, a bisection search on a grid with discretization size $N^{(1/D)}$ is performed. This gives $\mathcal{O}(D \cdot log(N^{(1/D)})) = \mathcal{O}(log(N))$ steps in total for the bisection search. For k iterations these are $\mathcal{O}(k \cdot log(N))$ steps.

- Each step of the bisection search requires the calculation of Eq. (20). This expression contains a double sum over the previously observed points. There are at most k of them, so there are at most $\mathcal{O}(k^2)$ summands. Each summand consists of D factors and thereby costs $\mathcal{O}(D)$. This results in $\mathcal{O}(Dk^2)$ per step of the bisection search, $\mathcal{O}(Dk^2 \log(N))$ cost per iteration and $\mathcal{O}(Dk^3 \log(N))$ in total.

4 PROOFS

For (conditional) entropies and the expected values, we introduce the following notation:

For $\mathbf{S}_{1:i} \sim \pi$,

$$
\begin{aligned}
\mathcal{H}(\mathbf{S}_{1:i}) &= -\sum_{S_{1:i} \in E^k} \pi(S_{1:i}) \log \pi(S_{1:i}) \\
\mathcal{H}(\mathbf{S}_i | \mathbf{S}_{1:i-1} = S_{1:i-1}) &= -\sum_{S_i \in E} \pi(S_i | S_{1:i-1}) \log \pi(S_i | S_{1:i-1}) \\
\mathcal{H}(\mathbf{S}_i | \mathbf{S}_{1:i-1}) &= -\sum_{S_{1:i} \in E^i} \pi(S_{1:i}) \log \pi(S_i | S_{1:i-1}) \\
\mathbb{E}_{\mathbf{S}_{1:i}}[h(S_{1:i})] &= \sum_{S_{1:i} \in E^k} \pi(S_{1:i}) h(S_{1:i}) \\
\mathbb{E}_{\mathbf{S}_i | \mathbf{S}_{1:i-1} = S_{1:i-1}}[h(S_{1:i})] &= \sum_{S_i \in E} \pi(S_i | S_{1:i-1}) h(S_{1:i}) \\
H(\mathbf{S}_{1:i}) &= \mathbb{E}_{\mathbf{S}_{1:i}}[h(S_{1:i})] + \mathcal{H}(\mathbf{S}_{1:i})
\end{aligned}
$$

In general, $\mathcal{H}(\mathbf{S}_{1:i}) \neq \mathcal{H}(p_\pi)$ and thereby $H(\mathbf{S}_{1:i}) \neq H(p_\pi)$. The chain rule for the entropy is

$$
\mathcal{H}(\mathbf{S}_{1:i}) = \sum_{i=1}^{k} \mathcal{H}(\mathbf{S}_i | \mathbf{S}_{1:i-1})
$$

For reference, the two statements regarding submodularity and monotonicity:

Proposition 1. *Krause et al. (2008) Given any symmetric, positive semi-definite matrix $L \in \mathbb{R}^{N \times N}$, the function $h(X) = \log \det L_{XX}$ is submodular, i.e.*

$$
\forall X_1 \subseteq X_2 \text{ and } i \notin X_2, h(X_1 \cup \{i\}) - h(X_1) \geq h(X_2 \cup \{i\}) - h(X_2)
$$

Proposition 2. *Sharma et al. (2015) Given any symmetric $L \in \mathbb{R}^{N \times N}$ with the smallest eigenvalue $\lambda_{min}(L) \geq 1$, the function $h(X) = \log \det L_{XX}$ is monotone, i.e.*

$$
\forall X_1, X_2 \text{ with } X_1 \subseteq X_2 \subseteq X, h(X_1) \leq h(X_2)
$$

Lemma 1. *Consider two independent random variables $\boldsymbol{G}_{1:k} \sim \pi_{greedy}$ and $\boldsymbol{O}_{1:k} \sim \pi$ for an **arbitrary** policy π. In each iteration $i = 0, ..., k-1$, we have*

$$
H(\boldsymbol{O}_{1:k}) \leq \mathbb{E}_{\boldsymbol{G}_{1:i}}\Big[h(G_{1:i})\Big] + k\Big[\mathbb{E}_{\boldsymbol{G}_{1:i+1}}[h(G_{1:i+1})] - \mathbb{E}_{\boldsymbol{G}_{1:i}}[h(G_{1:i})] + \mathcal{H}(\boldsymbol{G}_{i+1} | \boldsymbol{G}_{1:i})\Big]
$$

Proof.

$$
\begin{aligned}
& & H(\mathbf{O}_{1:k}) \\
&\overset{\text{definition of } H}{=} & \mathbb{E}_{\mathbf{O}_{1:k}}[h(O_{1:k})] + \mathcal{H}(\mathbf{O}_{1:k}) \\
&\overset{h \text{ is monotone}}{\leq} & \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\left[h(O_{1:k} \cup G_{1:i})\right] + \mathcal{H}(\mathbf{O}_{1:k}) \\
&\overset{\text{telescoping sum}}{=} & \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\left[h(G_{1:i}) + \sum_{j=1}^{k} h(G_{1:i} \cup O_{1:j}) - h(G_{1:i} \cup O_{1:j-1})\right] + \mathcal{H}(\mathbf{O}_{1:k}) \\
&\overset{h \text{ is submodular}}{\leq} & \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\left[h(G_{1:i}) + \sum_{j=1}^{k} h(G_{1:i} \cup O_j) - h(G_{1:i})\right] + \mathcal{H}(\mathbf{O}_{1:k}) \\
&\overset{\text{entropy chain rule}}{\leq} & \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\left[h(G_{1:i}) + \sum_{j=1}^{k} h(G_{1:i} \cup O_j) - h(G_{1:i})\right] + \sum_{j=1}^{k} \mathcal{H}(\mathbf{O}_j|\mathbf{O}_{1:j-1}) \\
&\overset{\text{cond. can only decrease entropy}}{\leq} & \mathbb{E}_{\mathbf{G}_{1:i}}\left[h(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{O}_j}[h(G_{1:i} \cup O_j) - h(G_{1:i})] + \mathcal{H}(\mathbf{O}_j)\right] \\
&\overset{\text{summarize}}{=} & \mathbb{E}_{\mathbf{G}_{1:i}}\left[h(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{O}_j}[h(G_{1:i} \cup O_j) - h(G_{1:i})] + \mathcal{H}(\mathbf{O}_j)\right] \\
&\overset{\mathbf{G},\mathbf{O} \text{ indep.}}{=} & \mathbb{E}_{\mathbf{G}_{1:i}}\left[h(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{O}_j}[h(G_{1:i} \cup O_j) - h(G_{1:i})] + \mathcal{H}(\mathbf{O}_j|\mathbf{G}_{1:i} = G_{1:i})\right] \\
&\overset{\text{greedyness}}{\leq} & \mathbb{E}_{\mathbf{G}_{1:i}}\left[h(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{G}_{i+1}}[h(G_{1:i} \cup G_{i+1}) - h(G_{1:i})] + \mathcal{H}(\mathbf{G}_{i+1}|\mathbf{G}_{1:i} = G_{1:i})\right] \\
&\overset{\text{summarize}}{=} & \mathbb{E}_{\mathbf{G}_{1:i}}\left[h(G_{1:i})\right] + k\left[\mathbb{E}_{\mathbf{G}_{1:i+1}}[h(G_{1:i+1})] - \mathbb{E}_{\mathbf{G}_{1:i}}[h(G_{1:i})] + \mathcal{H}(\mathbf{G}_{i+1}|\mathbf{G}_{1:i})\right]
\end{aligned}
$$

Lemma 2. *Consider two independent random variables $\boldsymbol{G}_{1:k} \sim \pi_{greedy}$ and $\boldsymbol{O}_{1:k} \sim \pi$ for an **arbitrary** policy π. We have*

$$(1 - 1/e)H(\boldsymbol{O}_{1:k}) \leq H(\boldsymbol{G}_{1:k})$$

Proof. By rearranging the terms from Lemma 1

$$H(\mathbf{O}_{1:k}) \leq \mathbb{E}_{\mathbf{G}_{1:i}}\left[h(G_{1:i})\right] + k\left[\mathbb{E}_{\mathbf{G}_{1:i+1}}[h(G_{1:i+1})] - \mathbb{E}_{\mathbf{G}_{1:i}}[h(G_{1:i})] + \mathcal{H}(\mathbf{G}_{i+1}|\mathbf{G}_{1:i})\right],$$

we get

$$H(\mathbf{O}_{1:k}) - \mathbb{E}_{\mathbf{G}_{1:i+1}}[h(G_{1:i+1})] \leq \left(1 - \frac{1}{k}\right)\left[H(\mathbf{O}_{1:k}) - \mathbb{E}_{\mathbf{G}_{1:i}}[h(G_{1:i})]\right] + \mathcal{H}(\mathbf{G}_{i+1}|\mathbf{G}_{1:i})$$

By induction over i, we have

$$H(\mathbf{O}_{1:k}) - \mathbb{E}_{\mathbf{G}_{1:i}}[h(G_{1:i})] \leq \left(1 - \frac{1}{k}\right)^{i}\left[H(\mathbf{O}_{1:k}) - \mathbb{E}_{\mathbf{G}_{1:0}}[h(G_{1:0})]\right] + \sum_{i=1}^{k}\left(1 - \frac{1}{k}\right)^{i-1}\mathcal{H}(\mathbf{G}_i|\mathbf{G}_{1:i-1})$$

Because $\mathbb{E}_{\mathbf{G}_{1:0}}[h(G_{1:0})] = 0$:

$$H(\mathbf{O}_{1:k}) - \mathbb{E}_{\mathbf{G}_{1:i}}[h(G_{1:i})] \leq \left(1 - \frac{1}{k}\right)^i \Big[H(\mathbf{O}_{1:k})\Big] + \sum_{i=1}^{k} \left(1 - \frac{1}{k}\right)^{i-1} \mathcal{H}(\mathbf{G}_i|\mathbf{G}_{1:i-1})$$

Because $(1 - 1/k) < 1$ and the entropy chain rule:

$$H(\mathbf{O}_{1:k}) - \mathbb{E}_{\mathbf{G}_{1:i}}[h(G_{1:i})] \leq \left(1 - \frac{1}{k}\right)^i \Big[H(\mathbf{O}_{1:k})\Big] + \mathcal{H}(\mathbf{G}_{1:k})$$

Setting $i = k$ and using the known inequality $1 - x \leq e^{-x}$:

$$H(\mathbf{O}_{1:k}) - \mathbb{E}_{\mathbf{G}_{1:k}}[h(G_{1:k})] \leq (1/e)\Big[H(\mathbf{O}_{1:k})\Big] + \mathcal{H}(\mathbf{G}_{1:k})$$

Rearranging terms and using the definition of H, we get the desired result:

$$(1 - 1/e)H(\mathbf{O}_{1:k}) \leq H(\mathbf{G}_{1:k})$$

Lemma 3. *The optimal policy is not uniquely determined due to sampling ordered sequences instead of unordered sets. Let π_{opt} be the optimal policy, that samples all sequences corresponding to the same set equally often:*

$$\forall S \in \mathcal{X}\, \forall S_{1:k} \in perm(S) : \pi_{opt}(S_{1:k}) = \pi_{opt}(S_{1:k}) = \frac{1}{k!} p_{\pi_{opt}}(S)$$

For $\boldsymbol{O}_{1:k} \sim \pi_{opt}$, we have:

(1) $\mathbb{E}_{\boldsymbol{O}_{1:k}}[h(O_{1:k})] = \mathbb{E}_{O \sim p_{\pi_{opt}}}[h(O)]$

(2) $\mathcal{H}(\boldsymbol{O}_{1:k}) = \mathcal{H}(p_{\pi_{opt}}) + \log k!$

For the greedy policy $\boldsymbol{G}_{1:k} \sim \pi_{greedy}$, there is:

(3) $\mathbb{E}_{\boldsymbol{G}_{1:k}}[h(G_{1:k})] = \mathbb{E}_{G \sim p_{\pi_{greedy}}}[h(G)]$

(4) $\mathcal{H}(\boldsymbol{G}_{1:k}) \leq \mathcal{H}(p_{\pi_{greedy}}) + \log k!$

Proof. (3) h is a set function, i.e. the order does not matter for h and thereby also does not matter for expectations of h.

(2)

$$\begin{aligned}
& \mathcal{H}(O_{1:k}) \\
\overset{\text{definition of } \mathcal{H}}{=} & -\sum_{S_{1:k} \in E^k} \pi_{opt}(S_{1:k}) \log \pi_{opt}(S_{1:k}) \\
\overset{\text{definition of } \pi_{opt}}{=} & -\sum_{S_{1:k} \in E^k} \frac{1}{k!} p_{\pi_{opt}}(S) \log \frac{1}{k!} p_{\pi_{opt}}(S) \\
\overset{|\text{perm}(S)|=k!}{=} & -\sum_{S \in \mathcal{E}} p_{\pi_{opt}}(S) \log \frac{1}{k!} p_{\pi_{opt}}(S) \\
\overset{\text{summarize}}{=} & \mathcal{H}(p_{\pi_{opt}}) + \log k!
\end{aligned}$$

(3) h is a set function, i.e. the order does not matter for h and thereby also does not matter for expectations of h.C

(4) Consider a joint sample $\mathbf{S}_{1:k} \sim \pi_{greedy}$ and $S \sim p_{\pi_{greedy}}$, since S is fully determined by $S_{1:k}$:

$$\mathcal{H}(\mathbf{S}_{1:k}) = \mathcal{H}(\mathbf{S}_{1:k}, \mathbf{S}) = \mathcal{H}(\mathbf{S}_{1:k}|\mathbf{S}) + \mathcal{H}(\mathbf{S})$$

$\mathcal{H}(\mathbf{S}_{1:k}|\mathbf{S})$ is maximized by a uniform order over all permutations, i.e. $\mathcal{H}(\mathbf{S}_{1:k}|\mathbf{S}) \leq \log k!$

Theorem 1. *Let h be a submodular set function with $h(\varnothing) = 0$ and $\Delta_h(x|X) > (1/k)\log k!$ for all $X \subset \mathbb{X}$, $x \in \mathbb{X} \setminus X$. Assume $\mathbb{X}$ is finite. It holds*

$$(1 - 1/e)H(p_{\pi_{opt}}) \leq H(p_{\pi_{greedy}}),$$

with $H(p_\pi) = \mathbb{E}_{X\sim p_\pi}[h(X)] + \mathcal{H}(p_\pi)$ and π_{opt} being the optimal policy.

Proof. Define a new set function $m(S) := h(S) + l(S)$ with $l(S) = -\frac{|S|}{k}\log k!$. Due to the properties of h and l being a modular function, we still have monotony and submodularity for m as well as $m(\varnothing) = 0$ such that Lemma 3 applies to m, too. The greedy policy π_{greedy} and the optimal sampling policy with uniform order π_{opt} as defined in Lemma 3 the same for m and h. Using Lemma 2 and Lemma 3, we obtain the result:

$$\begin{aligned}
& H(p_{\pi_{greedy}}) \\
\overset{\text{definition of } H}{=} \quad & \mathbb{E}_{G\sim p_{\pi_{greedy}}}[h(G)] + \mathcal{H}(p_{\pi_{greedy}}) \\
\overset{\text{definition of } m}{=} \quad & \mathbb{E}_{G\sim p_{\pi_{greedy}}}[m(G)] + \mathcal{H}(p_{\pi_{greedy}}) + \log k! \\
\overset{\text{Lemma 3}}{\geq} \quad & \mathbb{E}_{\mathbf{G}_{1:k}}[m(G_{1:k})] + \mathcal{H}(\mathbf{G}_{1:k}) \\
\overset{\text{definition of } M}{=} \quad & M(\mathbf{G}_{1:k}) \\
\overset{\text{Lemma 2}}{\geq} \quad & (1 - 1/e)M(\mathbf{O}_{1:k}) \\
\overset{\text{definition of } M}{=} \quad & (1 - 1/e)\Big[\mathbb{E}_{\mathbf{O}_{1:k}}[m(O_{1:k})] + \mathcal{H}(\mathbf{O}_{1:k})\Big] \\
\overset{\text{Lemma 3}}{=} \quad & (1 - 1/e)\Big[\mathbb{E}_{O\sim p_{\pi_{opt}}}[m(O)] + \mathcal{H}(p_{\pi_{opt}}) + \log k!\Big] \\
\overset{\text{definition of } m}{=} \quad & (1 - 1/e)\Big[\mathbb{E}_{O\sim p_{\pi_{opt}}}[h(O)] + \mathcal{H}(p_{\pi_{opt}})\Big] \\
\overset{\text{definition of } H}{=} \quad & (1 - 1/e)H(p_{\pi_{opt}})
\end{aligned}$$

Corollary 1. *Running the algorithm $\pi_{greedy}(x_i \,|\, X_{1:i-1}) \propto \mathbb{v}_i(x)$ as introduced in Section 3 for k iterations on a finite grid is a $(1 - 1/e)$ approximation to the exact distribution $p_{k\text{-}DPP}$ of the corresponding fixed size DPP, in the sense of*

$$(1 - 1/e)H(p_{k\text{-}DPP}) \leq H(p_{\pi_{greedy}}),$$

with $H(p) = \mathbb{E}_{X\sim p}[h_\alpha(X)] + \mathcal{H}(P)$ and $h_\alpha(X) = \log\det\Big(\alpha L_{XX}\Big)$ for $\alpha = \frac{k!^{1/k}}{\lambda_{min}}$. Where $\lambda_{min} > 0$ is the smallest eigenvalue of the Kernel Gram matrix over the grid.

Proof. By proposition 1, the function h_α is monotone. By proposition 2, the function h is also monotone since scaling with the reciprocal of the smallest eigenvalue ensures that all eigenvalues are larger than 1. Additionally scaling with $(k!)^{1/k}$ ensures that the marginal gains are larger than $\log k!^{1/k} = \frac{1}{k}\log k!$. Thus the assumptions in Theorem 1 are met and the result follows.

4.1 Analysis in Terms of the Information Gain

For a Gaussian process regression model $f \sim \mathcal{GP}(\mu, \ell)$ with additional i.i.d noise $\epsilon \sim \mathcal{N}(0, \sigma I)$ on the observations, the information gain $g(\mathrm{X}) = \frac{1}{2} \log\det\left(\mathrm{I} + \sigma^{-2}\mathrm{L}_{\mathrm{XX}}\right)$ quantifies the reduction in uncertainty about f by observing $f_{\mathrm{X}} + \epsilon_{\mathrm{X}}$. The matrix I denotes the identity matrix.

We'll again drop the constant factor of $1/2$ and instead analyse $g(\mathrm{X}_{1:k}) = \log\det\left(\mathrm{I} + \sigma^{-2}\mathrm{L}_{\mathrm{X}_{1:k}}\right)$.

It has been pointed out before by e.g. Srinivas et al. (2009) that g is monotone and submodular and can be decomposed in terms of the posterior variances

$$g(\mathrm{X}_{1:k}) = \sum_{i=1}^{k} \log(1 + \sigma^2 \mathbb{v}_{\sigma_i}(x_i)),$$

where the $\mathbb{v}_{\sigma_i}(x) = l(x,x) - l(x, \mathrm{X}_{1:i})(\mathrm{L}_{\mathrm{X}_{1:i},\mathrm{X}_{1:i}} + \sigma^2 \mathrm{I}) l(\mathrm{X}_{1:i}, x)$.

In the optimization setting, the greedy choice is given by

$$x_i = \arg\max_x \log(1 + \sigma^2 \mathbb{v}_{\sigma_i}(x)), \tag{6}$$

so we again take the softargmax instead and sample greedily from

$$\pi_{greedy_g}(x|\mathrm{X}_{1:i}) \propto \exp(\log(1 + \sigma^2 \mathbb{v}_{\sigma_i}(x))) \propto 1 + \sigma^2 \mathbb{v}_{\sigma_i}(x) \tag{7}$$

By direct application of Lemma 2, we get:

Corollary 2. *Consider π_{greedy_g} as defined in Eq. (7) and $\pi_{k\text{-}DPP}$ for an optimal policy for sampling from a k-DPP. It holds*

$$(1 - 1/e) H_g\left(\pi_{k\text{-}DPP}\right) \leq H_g\left(\pi_{greedy_g}\right),$$

where $H_{Ig}(\pi) = \mathbb{E}_{X\sim\pi}\left(g(X)\right) + \mathcal{H}\left(\pi\right)$, where $g(X) = \log\det\left(I + \sigma^{-2} L_X\right)$ for $\sigma^2 > 0$.

The policy to sample greedily from information gain can be implemented as efficiently as the original policy for the differential entropy:

The posterior $\mathbb{v}_{\sigma_i}$ retains much of the structure of $\mathbb{v}_i$, so in the terms for the posterior (Eq. 16), as well as the unnormalized cumulative density in (Eq. 17) the matrix L merely has to be replaced with $\mathrm{L} + \sigma^2\mathrm{I}$ to obtain the corresponding terms for $\mathbb{v}_{\sigma_i}$ and $\mathbb{V}_{\sigma_i}$. Note that we do not sample proportionally to $\mathbb{v}_{\sigma_i}(x)$, but proportionally to $1 + \sigma^2\mathbb{v}_{\sigma_i}$. However, given the efficient computation of $\mathbb{V}_{\sigma_i}$ the unnormalized cumulative density for $1 + \sigma^2\mathbb{v}_{\sigma_i}$ can easily be obtained due to linearity of integration.

5 ADDITIONAL EXPERIMENTAL DETAILS

The experiments were performed on a desktop machine with a 4 GHz Quad-Core Intel Core i7 processor and 32 GB memory. Table 5 lists the scales of the squared-exponential kernels used for BQ. Since all methods share the same kernel hyperparameters, we assume that their choice only plays a subordinate role and obtained them by eyeballing the ground-truth scale from 2D plots of the integrands. Figure 1 shows additional results from the BQ experiment described in Section 5 of the main paper for more benchmark functions.

Figure 2 shows additional runtime results in three dimensions. We draw 100 samples with $k = 10$ and $k = 100$ points from a k-DPP with square-exponential kernel with lengthscale 0.01, respectively 0.001, on the interval $[0, 1]^3$. Runs taking longer than 100 seconds for $k = 10$ or 500 seconds for $k = 1000$ were stopped, i.e. they do not appear in the figure. For $k = 10$, the experiment was repeated 5 times and for $k = 100$, 3 times.

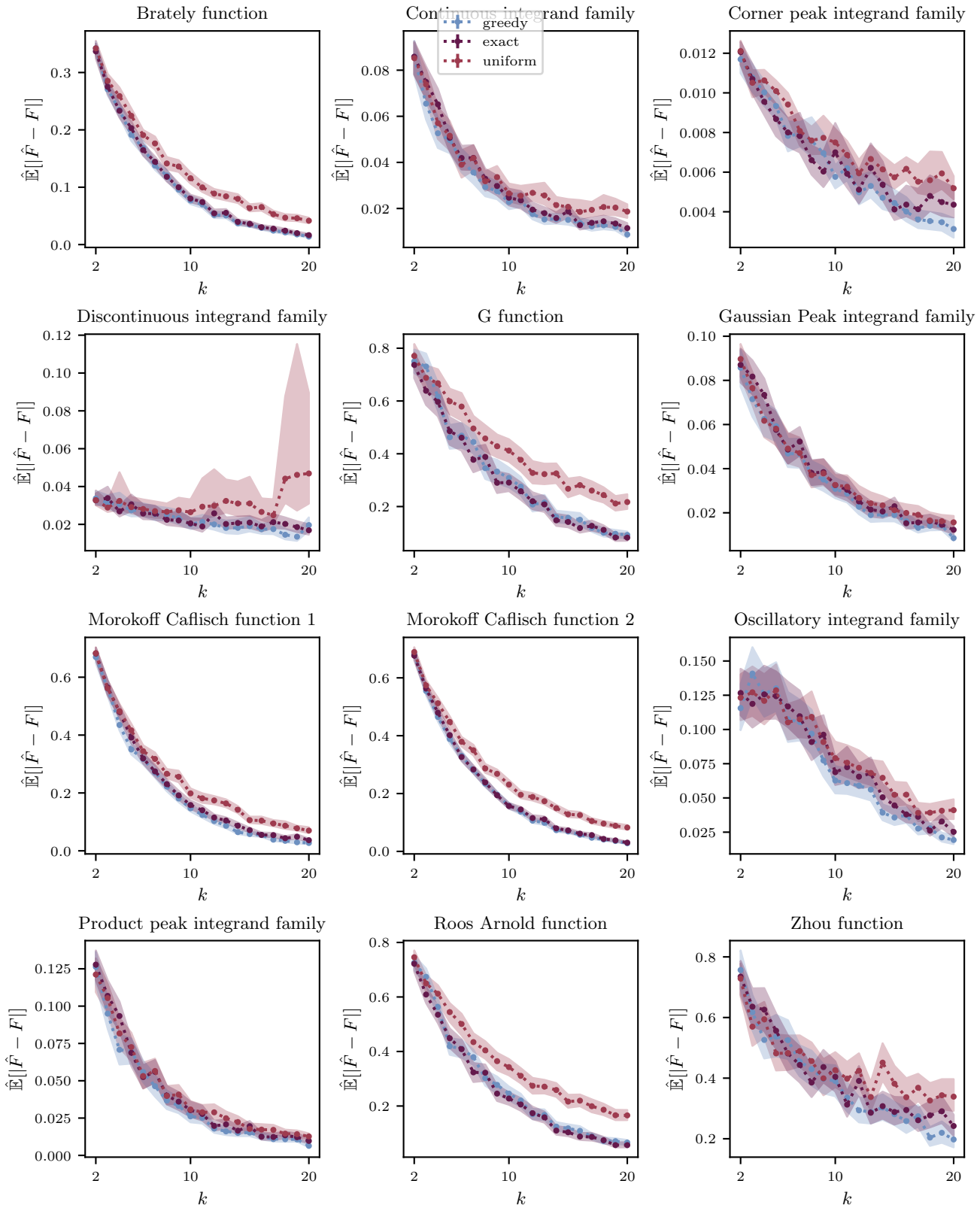

Figure 1: Results from Bayesian Quadrature with evaluation locations sampled from an exact k-DPP, the greedy approximation and uniformly sampled locations. The plots show the mean error over 100 samples and 95% confidence intervals for the mean error as shaded areas.

Table 1: Hyperparameters for the Bayesian Quadrature

Benchmark	**Scale**
Brately function	1
Continuous integrand family	1
Corner peak integrand family	1
Discontinuous integrand family	140
G function	3.5
Gaussian Peak integrand family	1
Morokoff Caflisch function 1	2.5
Morokoff Caflisch function 2	1
Oscillatroy integrand family	2
Product peak integrand family	700
Roos Arnold function	4
Zhou function	8

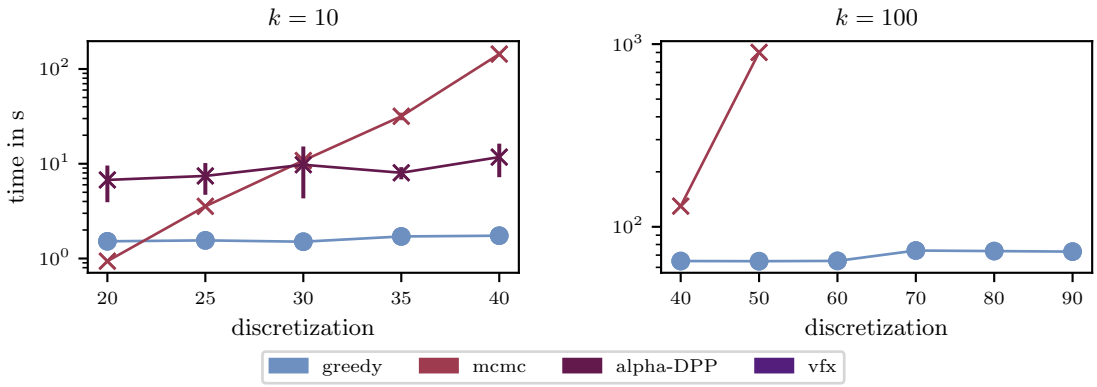

Figure 2: Runtime comparison of the greedy algorithm with state-of-the-art baselines in three dimensions.

References

Gradshteyn, I. and Ryzhik, I. (2007). *Table of Integrals, Series, and Products*. Academic Press, 7th edition.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2).

Sharma, D., Kapoor, A., and Deshpande, A. (2015). On greedy maximization of entropy. In *International Conference on Machine Learning*, pages 1330–1338. PMLR.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer S&B M.