Greedy Submodular Sampling and Its Relation to DPPs

by

Rahel Fischer

A thesis submitted to attain the degree
Bachelor of Science in
Cognitive Science

at the
Eberhard Karl University of Tübingen
2023

Date of Final Oral Exam: 28/02/2023
Supervisor: Julia Grosse
The thesis was approved by Philipp Hennig, Professor for Machine Learning.

Fischer, Rahel:
Greedy Submodular Sampling and Its Relation to DPPs
Bachelor Thesis Cognitive Science
Eberhard Karl University of Tübingen

Greedy Submodular Sampling and Its Relation to DPPs

A Cognitive Science Bachelor Thesis

Rahel Fischer supervised by **Julia Grosse**

Abstract

In machine learning, it is often required to generate samples that are diverse and informative without applying a deterministic procedure. Taking this as a quality criterion, determinantal point processes (DPPs) generate optimal samples. Yet, it is not always possible to apply DPPs in a reasonable run time - especially when it comes to huge domains. Hennig and Garnett (2018) proposed a greedy sampling algorithm for drawing DPP samples in a fast way. However, it turned out that this algorithm does not exactly draw DPP samples. In this thesis, we try to find lower bound guarantees about the relation between greedy sampling and DPPs by transferring guarantees from other procedures with known relations to both DPPs and greedy sampling. The primary approach transfers the $\left(1 - \frac{1}{e}\right)$-approximation guarantee of a greedy entropy maximization algorithm (Sharma et al., 2015) into the sampling setting. This approach requires monotonicity of entropy which can be ensured by modifications in greedy sampling. However, these modifications make the validity and the meaning of the guarantee more difficult to interpret.
We firstly give an introduction on the theoretical concepts that are relevant for this thesis, then introduce the greedy sampling algorithm according to Hennig and Garnett (2018) and point out why it does not draw exact DPP samples. In the final step, we analyze different approaches to find guarantees of samples drawn by greedy sampling and conclude our results.

Acknowledgements

My thanks go to all those who enabled and facilitated this bachelor thesis.
In the first place, of course, a whole lot of thanks go to Julia. I am so grateful for her patience despite all my (repeatedly asked) questions, her encouragement during the whole process, all the insights I got from her, her openness to discuss mathematical questions with me and take time to develop our approaches together - all in all: her caring, friendly, competent and invested supervision. Here, I also want to point out that she spent lots of energy and time on finding the proof that turned out to be a core of our reasoning.
Also, I want to thank Philipp who did not just offer me a bachelor thesis and accompanied Julia and me with technical help but also esteemed our work and thoughts even if they were not thought-out yet.
However, I could neither ask curious questions nor make use of good supervision if I would not have gotten so far in my studies. Therefore, I want to thank my parents who not only supported me during my studies by carrying me all the way through but also laid a great foundation for my entire life and are responsible for so many good things I have experienced. Also, I am thankful for great fellow students who provide an environment of good community for as well working, learning, discussing and thinking as just having a good time. Besides that, there were so many people who knew about this bachelor thesis (siblings, friends, flat mates, not to forget the two office mates of Julia) and took an interest in it - that was really heartening.
Taking all together: credits of this work are shared among many - what a blessing! Thank you!

Contents

Introduction

In machine learning, algorithms are often supposed to learn a function (f.e. a classification) on basis of a "small" data set to then predict the function value for new data. Obviously, this becomes more and more complex and expensive with growing training data sets. Therefore, some machine learning algorithms sample a subset from the big training data set for saving a lot of time and power. Doing so, it is clearly a fundamental question of how these samples should be generated. DPPs provide a way to find subsets that are both very informative but also vary between different samples.
Yet, when it comes to larger domains, DPPs take increasingly long to draw samples. Therefore, there are many approaches to approximate the mode of the DPP - the optimal sample (Bıyık et al., 2019; Çivril & Magdon-Ismail, 2009; Gillenwater et al., 2012). Other approaches approximate a space as good as possible (Kanagawa & Hennig, 2019) or reduce the size of the domain randomly to then draw an informative set (stochastic greedy algorithm suggested by Mirzasoleiman et al. (2015)). However, deterministic approaches do not offer a variety between samples and stochastic greedy algorithms can hardly guarantee to generate exact DPP samples.
In contrast, Calandriello et al. (2020) found a way to fast generate exact k-DPP samples (for which the sample size k is known in advance) without looking at all items. Similarly, Hennig and Garnett (2018) introduced a greedy sampling algorithm whose run time is not dominated by the domain size whereas it draws well-distributed samples.
However, this greedy sampling algorithm does not exactly draw DPP samples, mainly, because it was assumed that the probability of a sample to be drawn is the same, no matter in what order the points are selected. This assumption does not hold for greedy sampling and thus the exactness of this algorithm is not given anymore. Therefore, we try to show here how close greedy sampling is to DPPs. To do so, we find procedures of which guarantees about the relation to DPPs are known and try to show equivalence to greedy sampling such that we can apply the known guarantees to greedy sampling as well.
In this thesis, we first give an introduction to DPPs and mathematical concepts that will be used later (Chap. 1). In Section 2.1.4, the greedy sampling algorithm will be introduced and in Section 2.1 it will be shown why it does not exactly draw DPP samples. Chapter 3 contains ideas on how to find assertions about how close greedy sampling is to DPPs. Two of the approaches that are shown there (originating from the Cholesky-based DPP sampling method and from a weak greedy maximum fill distance algorithm (Kanagawa & Hennig, 2019)) do not succeed because the procedures are not found to be equivalent to greedy sampling. In contrast, the approach originating from a greedy maximization of entropy algorithm leads to new insights. Provided that entropy is a monotone submodular function, it is known that finding the maximum of entropy has a greedy approach that

finds a $\left(1-\frac{1}{e}\right)$-approximation of the maximum which is the mode of the DPP. However, because we are not interested in approximating one sample but a whole distribution, we show that a probabilistic version of the greedy maximization algorithm also guarantees a $\left(1-\frac{1}{e}\right)$-approximation of the DPP distribution.
Still, it turns out that entropy is not always monotone such that some guarantees apply to greedy maximization algorithms only with a scaled similarity measurement and space. This leads to questions regarding the general meaningfulness of ratio guarantees concerning entropy that will be discussed in Subsection 3.2.2.

Chapter 1

Theoretical Background

1.1 Determinantal Point Processes (DPPs)

Intuitively, DPPs are stochastic point processes that generate samples that are as well informative as rich in variety. They were introduced in order to model fermion distributions (Macchi, 1975). Meanwhile, they have found various applications in other fields as machine learning. This section will give a brief summary of DPPs emphasizing aspects that are relevant for this thesis.

1.1.1 Why are we interested in DPPs?

In some applications, it is convenient to have both: diversity within one selected subset but also varying selections of subsets of domains. In the case of a news feed, for instance, we usually aim to have different topics (intra-variety) but also a "fresh" news feed each time the user reloads it (inter-variety). Though deterministic approaches might have a higher intra-variety, they can not generate different results given the same input (user and their interests, current news) and thus show no inter-variety.

Beyond that it is sometimes convenient to have varying samples, it is also an NP-hard problem to determine the subset of points that span the greatest volume together (Çivril & Magdon-Ismail, 2009). With that, DPPs also provide reasonably diverse sets of points in

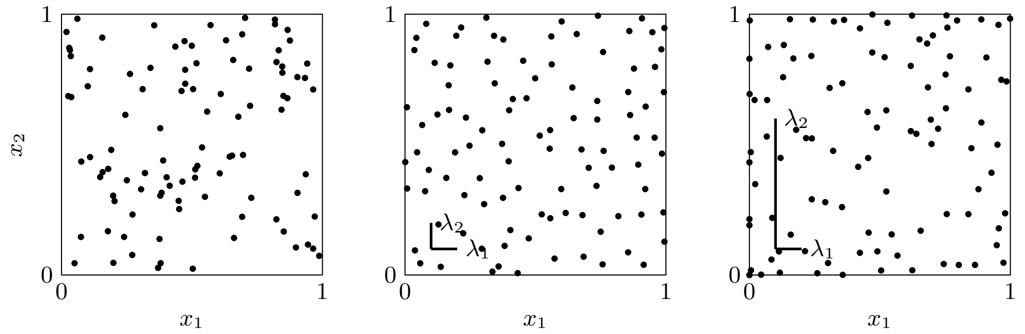

Figure 1.1: Fig.1 from Hennig and Garnett (2018). Different distributions of one hundred points depending on the sampling procedure. Left: uniform random. Center and right: RBF-kernel DPPs of different length scales - here denoted by λ.

polynomial time. In machine learning, DPPs can be applied in the exploration approaches of algorithms. Because they choose points that push each other off, exploration is more effective than when exploring randomly (see Fig. 1.1). In classification tasks, for instance, the differentiation can be learned based on human feedback on DPP-generated samples (Bıyık et al., 2019). Due to the use of the DPPs, the human feedback is gained on different but each diverse samples and is thus very informative.

1.1.2 What are DPPs?

Let $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a positive semi-definite kernel function. That means that there is a scalar product space $(U, \langle ., . \rangle)$ and a mapping $\Phi : \mathbb{X} \to U$ into that space with $k(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle \ \forall x_1, x_2 \in \mathbb{X}$. The kernel function, therefore, indicates the similarity (scalar product) of two points of $\mathbb{X}$ after they were first mapped into a scalar product space by the feature function Φ. Often, Φ is not explicitly calculated but a relevant step to ensure both that $\mathbb{X}$ does not have to be a scalar product space itself and that a scalar product/similarity can be calculated.
Given a kernel function, one can calculate a kernel matrix $K_{\mathbb{X}\mathbb{X}}$ with $K_{\mathbb{X}\mathbb{X}_{x_1,x_2}} = k(x_1, x_2)$ for all $x_1, x_2 \in \mathbb{X}$ that describes the similarities between all $x_1, x_2 \in \mathbb{X}$. Respectively, for a finite subset $X \subset \mathbb{X}$ K_{XX} is a principal submatrix of $K_{\mathbb{X}\mathbb{X}}$ describing the similarities for all $x_1, x_2 \in X$.
A DPP now draws a sample S in proportion to the variety of points in S which is contained in the kernel matrix:

$$p(X = S) = \frac{\det(K_{SS})}{Z} \propto \det(K_{SS})$$

That means: the greater the volume of K_{SS} (and with that the diversity of points), the more likely it is that the DPP would draw S. With that property, DPPs provide a procedure that preferentially draws samples that have a strong variety within one sample but also offer a reasonable variety between samples as they generate their samples in a non-deterministic way. This proportional relation is characterized by the proportionality factor (here, we also call it normalization constant) Z that is the sum of the (unnormalized) probabilities over all possible subsets $X \subset \mathbb{X}$:

$$Z = \sum_{X \subset \mathbb{X}} \det(K_{XX})$$

In the literature, one can find DPPs defined with both a correlation kernel (here denoted as $K_{\mathbb{X}\mathbb{X}}$) and a likelihood kernel (here denoted as $L_{\mathbb{X}\mathbb{X}}$). These kernels can (except in the case that $K_{\mathbb{X}\mathbb{X}}$ is orthogonal) be transformed into each other:

$$L_{\mathbb{X}\mathbb{X}} = K_{\mathbb{X}\mathbb{X}}(I - K_{\mathbb{X}\mathbb{X}})^{-1} \text{ and } L_{\mathbb{X}\mathbb{X}} = I - (L_{\mathbb{X}\mathbb{X}} + I)^{-1}$$

with I being the identity matrix with size $|\mathbb{X}| \times |\mathbb{X}|$ for $|\mathbb{X}| < \infty$ (Kulesza & Taskar, 2012). DPPs conducted on basis of a likelihood kernel are also called L-ensembles. Using either of the two kernels can lead to different algebraic traits, and applications but also interpretations of the kernels. However, we will focus on correlation kernels in this thesis.

In the following, we will look further into different types of DPPs, their properties, and their sampling methods.

1.1.3 Finite and Continuous DPPs

When talking about characteristics of different DPPs, it also plays a role from which domain a certain kernel maps into $\mathbb{R}$. More precisely, one can differentiate between DPPs whose domain is finite (finite DPPs) or continuous (continuous DPPs). In both cases, however, the domain is bounded. Both types come with different restrictions and enablements for DPP sampling procedures. Therefore, we will shortly go over some properties and definitions of these two types of DPPs before describing sampling procedures.

Finite DPPs

Let $\mathbb{X} = \{1, 2, ..., N\} = [N]$ be a finite (and thus discrete) domain and $K_{\mathbb{X}\mathbb{X}} \in \mathbb{R}^{N \times N}$ the corresponding kernel matrix. Just as given above, we say for subsets $S \subset \mathbb{X}$ that they follow a DPP distribution with kernel $K_{\mathbb{X}\mathbb{X}}$ if $p(S) \propto \det(K_{SS})$.
One of the most basic characteristics of any sample is its size. For most DPPs (exception: projections, see Subsec. 1.1.4), the number of points itself is not determined before sampling. Still, there is a formula describing the sample size for a sample S:

$$|S| = \sum_{n=1}^{N} Ber(\lambda_n)$$

where λ_n ($n \in \{1, ..., N\}$) stands for the eigenvalues of $K_{\mathbb{X}\mathbb{X}}$ and $Ber(\lambda) \in \{0, 1\}$ for a Bernoulli variable with success probability λ. One can directly derive the expected sample size (Hough et al., 2006):

$$\mathbb{E}[|S|] = trace(K_{\mathbb{X}\mathbb{X}}) = \sum_{n=1}^{N} \lambda_n$$

Continuous DPPs

Defining a continuous DPP can differ in complexity. For our purposes, it is sufficient to use the same general definition with a different domain. Let $\mathbb{X}$ be a continuous bounded set and $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ the corresponding kernel function. As given earlier: for a finite subset $X \subset \mathbb{X}$ K_{XX} denotes the kernel matrix for X. A subset $S \subset \mathbb{X}$ now follows a DPP distribution of kernel $K_{\mathbb{X}\mathbb{X}}$ if $p(S) \propto \det(K_{SS})$.
Just as for finite DPPs the size of a sample drawn by a continuous DPP is not known before sampling. It is given by the infinite sum

$$|S| = \sum_{n=1}^{\infty} Ber(\lambda_n)$$

where λ_n ($n > 1$) again stand for the eigenvalues of $K_{\mathbb{X}\mathbb{X}}$ and $Ber(\lambda) \in \{0, 1\}$ for a Bernoulli variable with probability λ. The expected value is given by

$$|S| = \sum_{n=1}^{\infty} \lambda_n,$$

respectively (Lavancier et al., 2015).
Continuous DPPs are less explored than finite DPPs. When talking about sampling algorithms, the great majority of literature covers finite DPPs. That is intuitive as one could argue that numbers that a computer can use are always instances on a grid and thus never really continuous. However, in this thesis, we will talk about algorithms with a very fine underlying grid as continuous.

So far, we differentiated between DPPs according to their domain. In the next two sections, the characteristic of interest is the kernel. Firstly, we will have a closer look at orthogonal kernels.

1.1.4 Orthogonal Projection DPPs

A DPP is fundamentally characterized by its kernel function. Projection DPPs are DPPs whose kernel function is orthogonal. There are several properties that come with that orthogonality, especially when talking about the eigenvalue decomposition. All the eigenvalues $\lambda_1, ...\lambda_n$ of any orthogonal kernel matrix are either 0 or 1. Therefore, it is always known in advance how many points a sample S generated by a projection DPP will sample because $|S| = \sum_{n=1}^{\infty} Ber(\lambda_n)$ is the same as the number of eigenvalues that are unequal to 0 which again is the same as the kernel matrix' rank. That number can be easily determined once the eigenvalue decomposition of the kernel matrix is done.

Projection DPPs can be used as components to mix other non-orthogonal kernels. Though they have some handy properties, the most popular kernel function is not orthogonal. It will be introduced in the following section.

1.1.5 The RBF Kernel

As stated above, the kernel function characterizes a particular DPP. In this thesis, we will mainly talk about the radial basis function (RBF, aka. Gaussian, squared-exponential) correlation kernel

$$k_{RBF}(x_1, x_2) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}, (x_1, x_2) \mapsto \exp\left(-\frac{1}{2}\sum_{d=1}^{D} \frac{(x_1 - x_2)_d^2}{\sigma^2}\right)$$

for a fixed parameter $\sigma > 0$ that is called the length scale.
The RBF kernel became prevalent for DPPs and machine learning in general - possibly because it combines several convenient qualities: Firstly, it is stationary, that is, that it describes the similarity of two points independently from their absolute position within the domain. That gives us the insight that the kernel function only considers the local relation between the points rather than the global relation of these points to the whole domain. The RBF kernel is even homogeneous (isotropic) which means that it does only depend on the norm of the difference between two points (Rasmussen & Williams, 2006, Ch. 4). That property allows making assumptions that make the application in machine learning easier.
Secondly, the functions which an RBF kernel fits are smooth functions. This can be both helpful and restricting depending on whether we assume the function that should

be learned to be smooth or not. Despite the smoothness property, the RBF kernel is still universal, which means, informally speaking, that it could theoretically approximate any differentiable bounded function arbitrarily well. In other words, its scalar proudct space is dense in $f : X \to \mathbb{R}$ (Micchelli et al., 2006). However, because of the smoothness property, it might not be possible to approximate an unsmooth function with a limited amount of data.

Though the RBF kernel has several convenient characteristics, there are also some shortcomings. Most notably for us, it is not orthogonal. This is relevant because there are some handy features that only projection DPPs have which we then can not transfer onto RBF DPPs.

1.1.6 Not Quite a DPP: k-DPPs

Especially when applying DPPs to real-world tasks, it can be practical to have certainty about the final sample size. Therefore Kulesza and Taskar (2011) proposed the concept of k-DPPs that determines the sample size beforehand. A k-DPP is defined by

$$p_{\mathrm{k}}(X = S) = \frac{\det(K_{SS})}{\sum_{|S'|=\mathrm{k}} \det(K_{S'S'})} = \frac{\det(K_{SS})}{Z}$$

where $|S| = \mathrm{k}$ and $K_{\mathbb{X}\mathbb{X}}$ stands for the kernel matrix and $Z = \sum_{|S'|=k} \det(K_{S'S'})$ is a normalization constant .
A naive way to transform a DPP to a k-DPP is to introduce a rejection condition such that only samples that by chance have a maximal size k will be accepted. Researchers, however, have found many more favorable approaches (Kulesza & Taskar, 2011) that run in $O(N\mathrm{k}^2)$, assuming an eigendecomposition of the kernel matrix $L_{\mathbb{X}\mathbb{X}}$ (in this paper they used a likelihood kernel). Because the samples drawn by a k-DPP have a constant size, they are not identically distributed as non-restricted DPPs and thus are not exactly DPPs, though they are often traded as restricted DPPs. In fact, this thesis will mainly focus on k-DPPs. Thus, from now on we will talk about DPPs when actually talking about k-DPPs for simplicity.

1.1.7 Exact Sampling Algorithms

All algorithms that implement DPP sampling draw samples that follow the exact same distribution. However, there are different approaches to fulfill this task of which three will be introduced in the following. Therefore we assume a finite DPP for a start and consider the kernel matrix $K_{\mathbb{X}\mathbb{X}}$ to be given.

Classic: Spectral Method

The spectral method was introduced by Hough et al. (2006) and is the standard method for DPP sampling. Its key is that any kernel matrix is a mixture of several projection matrices.
Firstly, the orthonormal eigendecomposition (λ_i, v_i) of the kernel matrix $K_{\mathbb{X}\mathbb{X}}$ is calculated (λ_i denote the eigenvalues and v_i the eigenvectors for $i \in \{1, ...N\}$). To do so, assume

$K_{\mathbb{X}\mathbb{X}}$ to be a real symmetric matrix. Its eigenvectors form an orthonormal basis and all its eigenvalues are between 0 and 1. The spectral decomposition is then given by

$$K(x, y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(y),$$

λ_i, ..., λ_n denoting its eigenvalues, n the total number of eigenvalues, ϕ the feature function (see Sec. 1.1).
Gautier et al. (2019) now describe a two-step procedure to draw exact samples from that: in step one m of the n eigenvalue indices $\{i_1, ..., i_m\}$ $m \leq n$, are randomly selected using Bernoulli trials. Out of that, there is a new kernel matrix $\tilde{K}$ generated with

$$\tilde{K}(x, y) = \sum_{l=1}^{m} \phi_{i_l}(x)\phi_{i_l}(y).$$

$\tilde{K}$ now is an orthogonal component of the mixture of projections constituting K.
Step 2 is simply the sampling step in which any exact procedure (for instance Algorithm 18 from Hough et al. (2006)) can be used to draw a sample $\{x_1, ..., x_m\} \sim DPP(\tilde{K})$ which now also follows $DPP(K)$ (Gautier et al., 2019). Side note: here, the number of points is known in advance because $\tilde{K}$ is a projection DPP (see Subsec. 1.1.4).
It is evidently tricky to do exactly the same in the continuous case because $K_{\mathbb{X}\mathbb{X}}$ would then be an infinite matrix and would possibly have infinitely many eigenvalues. Still, Lavancier et al. (2015) present an implementation of this method for continuous domains. Though this method is exact and popular, it is not the fastest. The eigendecomposition needs $O(N^3)$ in the finite setting. The duration of the steps that follow (containing Gram-Schmidt orthonormalization) depends on the sample size $|S|$ of the drawn sample S and is in $O(N|S|^2)$ which is still in $O(N^3)$. All in all, the first term dominates the run time. Especially for large amounts of data computing the eigenvalues can be very expensive which then makes the spectral method overall expensive (Launay et al., 2020). Altogether, it still is an exact procedure that is reasonably fast for not too huge data sets.

State-of-the-Art: Intermediate Sampling

The intermediate sampling method (Dereziński, 2019) constitutes a newer and faster DPP sampling method. Because the domain's dimension will play a role when talking about this method, we now emphasize this by talking about a data matrix rather than about a domain.
Given a data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$, the main idea is to first select poly(D) points S following an intermediate distribution *intermediate*($\mathbf{X}$), N denoting the size of the domain and D the domain's dimension. Out of that intermediate sample S a new sample X is drawn that follows the DPP distribution target(S). Because the intermediate distribution was chosen appropriately, X now follows target($\mathbf{X}$).
Evidently, a crucial point for this procedure is the design of *intermediate*($\mathbf{X}$). To guarantee that *intermediate*($\mathbf{X}$) does not distort the target distribution Dereziński (2019) make use of the ridge leverage score introduced by Alaoui and Mahoney (2015). It is, intuitively speaking, a measurement that balances low-rank approximation and overfitting for a matrix approximation (McCurdy, 2018). Thus, the matrix of the intermediate sample S is

an approximation of the large data matrix $\mathbf{X}$. Still, it is not the same as a DPP sample as the points chosen so far do not push each other off. To exactly draw a DPP sample, the rejection sampling step on S is applied following the DPP distribution.
Derezinski et al. (2019) of course, show more detail about guarantees, formulae, and premises regarding $intermediate(\mathbf{X})$. In their paper, the authors did not include continuous domains. For a continuous domain (or a fine grid), this procedure might take long to determine the ridge leverage scores but in general, it is a procedure that can handle great domains.
Talking about run time, this method performs remarkably better than the spectral method. The preprocessing step (determining the intermediate distribution) runs in $O(N\log(N) + \text{poly}(D))$ time. The sampling step runs in poly(D) time. With that, the intermediate sampling method could substantially speed up using the idea of reducing the possibly large data matrix/domain.

Stochastically Intuitive: Cholesky-Based Method

When thinking about the Cholesky-based method (CBM) presented by Launay et al. (2020) it might be good to imagine a binary probability tree. Initially, consider two empty sets of points A_{in} and A_{out} that will in the end contain all the points from $\mathbb{X}$. Now we iteratively go through every single point x in $\mathbb{X}$ and randomly decide whether we want to include this point (add it to A_{in}) or not (add it to A_{out}). That iterative binary decision is what can be well shown with a binary tree. To decide about in- or excluding a point, the probability for a point x_i to be included $p(x_i \in S)$ depends on the preceding choices of points:

$$p(x_i \in S) = p(x_i \in S | A_{in}(i-1) \subset S \text{ and } A_{out}(i-1) \cap S = \varnothing\ )$$

with $p(x_1 \in S) = 0.5$ and $p(x_i S) = 1 - p(x_i \in S)$. After iterating through the whole domain the selected points form a DPP sample (Gautier et al., 2019). In each step we use Cholesky decomposition and thus can use matrix multiplication for determining each p. The main advantage of this method is that we do not have to compute an eigendecomposition which can be costly for large $\mathbb{X}$. Apparently, this method is not possible to be applied to continuous domains because its key feature is iterating through the whole domain.
The order of computational complexity is, just as the one of the spectral method $O(N^3)$. However, Launay et al. (2020) showed in their tests that the spectral method outplays the CBM. The crucial point that makes this method expensive is probably the Cholesky decomposition which is conducted in each iteration.

1.2 Submodularity

In this thesis, we are interested in DPPs and other processes that maximize the volume (determinant) of a matrix (for more detail, see Sec. 1.1). Taking the log-determinant from a matrix and increasing the matrix' numbers of rows and columns, the log-determinant changes in a submodular way (Gillenwater et al., 2012). This can be helpful for finding guarantees and comparing different approaches to the determinant maximization task. Therefore, we will shortly introduce the concept of submodularity.

Formally, a function $f : \mathcal{P}(M) \to \mathbb{R}$ is called submodular if

$$f(A \cup \{x\}) - f(A) \geq f(B \cup x) - f(B)$$

holds for $A \subset B \subset M$ with $x \in M$ and $x \notin B$, $\mathcal{P}(M)$ denoting the power set of M (Gillenwater et al., 2012). For monotone functions, this formalizes, intuitively speaking, that the benefit regarding f of a single element in a set diminishes when the set's size increases.
When making use of the submodularity of a function f, it is often also required to be monotone increasing. That means that $A \subset B$ implies that $f(A) \leq f(B)$. This is because if f was monotone decreasing ($A \subset B \Rightarrow f(A) \geq f(B)$), it was submodular anyways such that submodularity is no helpful information about f. To gain an intuition about submodularity, see also Figure 1.2.
In the following section, we will look into two functions on sets that are submodular.

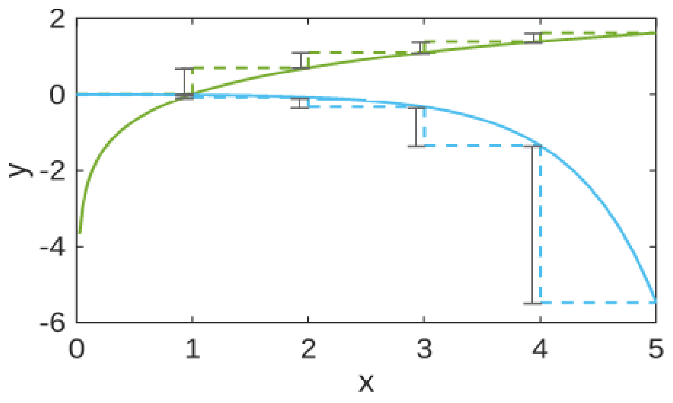

Figure 1.2: Two submodular functions. Blue: monotone decreasing. Green: monotone increasing. In both cases, the change of the function value decreases. However, in the case of the blue function, its absolute value increases. Thus, submodularity is implicated by the monotone decreasing property and hence not informative.

1.3 Entropy and Information Gain

In many contexts, entropy is seen as a measurement of randomness in a system. Here, we need a slightly different perspective on entropy and have to be sensitive that the interpretation of entropy is not quite the same for discrete and continuous random variables. In general, entropy is defined for a random variable. Because we will later apply the entropy in contexts of sampling, we will here first introduce the concept in that original vocabulary and later transfer it to our setting - sampling procedures. In that setting, entropy is a quality measurement for samples drawn by a certain policy π which can be thought of as a probability distribution.
In the following, we will introduce two types of entropy. On the way, there will occur logarithm terms for which we will not use a specific basis but talk about any basis > 1. That is because the basis only changes the unit of entropy but no property that is relevant for us. However, it was first introduced using 2 as basis (Shannon, 1948).

1.3.1 Shannon Entropy

The Shannon entropy was originally introduced in information theory to measure the average information content of a single sign in a word - a measure that is always positive. For a random variable $\mathbf{Y}$ it defines the average information content of a single event Y. Here, an underlying assumption is that the less likely an event is to happen the more informative it is, and the greater its entropy is (Shannon, 1948).
Let $\mathbf{Y}$ be a discrete random variable with n finite possible events $\mathbb{Y} = \{Y_1, ..., Y_n\}$. For each event Y let $p(Y)$ be the probability for that event to happen. Proceeding from that the information content for a single event can be computed with

$$I(Y) = -\log(p(Y))$$

which formalizes that a higher probability of an event makes it less informative.
The entropy of one event (how much information (on average) one event contains) is then defined as

$$H_1(\mathbf{Y}) = \mathbb{E}[I] = \sum_{Y \in \mathbb{Y}} p(Y) I(Y) = -\sum_{Y \in \mathbb{Y}} p(Y) \log(p(Y)).$$

Now assume that m different events occurred in a sequence S and assume that we have a given probability for such a sequence $p(S)$. The entropy (the average information content of a sequence S of m events) then is

$$H_m(\mathbf{Y}) = -\sum_{S \in \mathbb{Y}^m} p(S) \log(p(S)).$$

Because Shannon entropy is based on the probability of a single event, it can not simply be transferred to continuous domains because then every single event Y would be selected with probability $p(Y) = 0$. With that, the information content $I(Y)$ of a single event would virtually be ∞ and the whole concept of entropy would lose its validity. Therefore, there is another concept needed for continuous domains.

1.3.2 Differential Entropy

Differential entropy is nearly the continuous equivalent of Shannon entropy. However, it is harder to interpret differential entropy (for instance it can also be negative) - it is not exactly equivalent to Shannon entropy. Let $\mathbf{Y}$ be a continuous multivariate Gaussian distributed random variable with infinitely many single events $Y \in \mathbb{Y}$. Just as in the discrete case, a probability is assigned to every single event using a probability density function $p(Y) : \mathbb{Y} \rightarrow [0, 1]$. Then, the entropy of $\mathbf{Y}$ is

$$h(\mathbf{Y}) = \int_{-\infty}^{\infty} p(Y) \log(p(Y)) \; dY = \mathbb{E}[-\log(p(Y)].$$

Because $\mathbf{Y}$ is a multivariate Gaussian distributed variable, its differential entropy can be calculated as

$$h(\mathbf{Y}) = \frac{1}{2} \ln \det(2\pi e \Sigma)$$

where Σ is a covariance matrix.

There are two properties of differential entropy that are mathematically relevant. Firstly, it is shift-invariant, which means $h(c + \mathbf{Y}) = h(\mathbf{Y})$. This can be practical if one is interested in measuring entropy without looking at absolute values but rather at relations. Secondly, if $\mathbf{Y}$ is scaled with a scalar $A \in \mathbb{R}$, entropy changes with an additive term: $h(A\mathbf{Y}) = h(\mathbf{Y}) + \log(|A|)$. This will become important in later chapters.

It is not yet obvious how the concept of entropy is related to our context. That is why the next paragraph will translate entropy into the sampling setting for the cases we will use later in this thesis.

1.3.3 Transfer in Sampling Setting

In general, we are interested in using entropy as a quality criterion for a sampling policy π. The random variable in our case is the outcome of a sampling step. $\mathbb{X}$ is the set of possible events - the domain of which points can be drawn. What was called "sequence" in the sections above, is now a "sample" - several points that were drawn in one sampling procedure. π determines a probability as well for any single point ($p(x)$, $x \in \mathbb{X}$) as for any sample ($p(X)$, $X \subset \mathbb{X}$) to be drawn.
For our purposes, Shannon entropy will be relevant not for determining the average information content of a single point but rather for the average information content of a sample. This works, because we are interested in sampling procedures where the sample size is determined beforehand (k-DPPs, for instance) and $\mathbb{X}$ is finite. Thus, we can apply the formula for an event sequence/sample with length/size k:

$$H_{\mathrm{k}}(\mathbf{X}) = - \sum_{S \in \mathbb{X}^{\mathrm{k}}} p(S) \log(p(S)).$$

Because this is a measurement that tells us about the informativeness of a sample and not a point within a sample, we will term this type of entropy "inter-entropy".

In contrast, we will use differential entropy to make assertions about the informativeness of a single point in a single sample S generated by π. $\mathbf{X}$ is now again a multivariate k-dimensional Gaussian distributed random variable. Therefore, its entropy is determined by

$$h(\mathbf{X}) = \frac{1}{2} \ln \det(2\pi e K_{SS}) \propto \log(\det(K_{SS})).$$

Because this expression reveals something about a single point within one sample, not about a whole sample, we will term this type of entropy "intra-entropy". The formal main difference between inter- and intra-entropy is not that we used for one Shannon and for the other differential entropy but that intra-entropy is normalized by k contrary to inter-entropy.

Though we described several perspectives/types of entropy, there are properties that entropy generally has. Two are very shortly described in the following paragraph.

1.3.4 Properties of Entropy

Looking at (both, Shannon and differential) entropy of samples selected by policy π while increasing the number of points π should select (denoted by m) we want to emphasize two properties. Firstly, entropy is submodular (Sharma et al., 2015). Speaking intuitively, when considering a single sample, the benefit of adding a new point decreases with growing set size. With that, the average over all information contents of samples with size m, thus the inter-entropy $H_m(\pi)$ or $h_m(\pi)$, is submodular with respect to growing m. Secondly, entropy is not necessarily monotone (Sharma et al., 2015). This will play a role in later chapters of this thesis.

Chapter 2

Greedy Sampling - a Fast and Almost Exact DPP Sampling Approach

2.1 The Greedy Sampling Algorithm According to Hennig and Garnett (2018)

The algorithm suggested by Hennig and Garnett (2018) is an approach to finding a very fast algorithm that would (for certain continuous domains) generate exact DPP samples. This is desirable because the known exact sampling algorithms take a long run time - especially for continuous domains (see Subsec. 1.1.7). Fast algorithms do usually not provide an exact DPP sampling procedure or do not approach exact DPP sampling in the first place but try to approximate the mode of a DPP (Bıyık et al., 2019; Çivril & Magdon-Ismail, 2009; Gillenwater et al., 2012) or find other ways to select points in an informative way (Djolonga et al., 2018; Mirzasoleiman et al., 2015).
The new approach underlying this algorithm is based on a new perspective on DPPs. Hennig and Garnett (2018) suggest that a DPP would arise if one followed an elementary active learning strategy that chooses evaluation points to learn a function on the domain as exactly as possible. To make use of that, their algorithm uses the concepts from Gaussian processes. In this thesis, we will use the term greedy sampling to refer to that algorithm that will be introduced in more detail in the following.

2.1.1 How Greedy Sampling Works in the One-Dimensional Case

Let $f : \mathbb{X} \to \mathbb{R}$ be a function that an algorithm is aiming to learn on the domain $\mathbb{X}$. To do so, it iteratively chooses a set of evaluation points $X \subset \mathbb{X}$. In each iteration it selects a point $x \in \mathbb{X}$ solely depending on how uncertain it is about $f(x)$. The more uncertain it is, the more likely it will select and finally evaluate x. That procedure constitutes the greediness of this sampling algorithm.
In the first place, however, to have an uncertainty about $f(x)$, it has to keep track of belief over function values $f(x)$ for all $x \in \mathbb{X}$. The uncertainty about these function values for any x is formally the variance.

Therefore, it assumes a Gaussian process prior $f \sim GP(\mu, k)$ with an arbitrary a priori mean function $\mu : \mathbb{X} \to \mathbb{R}$ and the a priori covariance function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ that is precisely the kernel function. Once a certain x is selected, it is evaluated such that new information on f is gained and a current covariance cov_i can be computed. According to the updated variance, which is $cov_i(x, x)$ for $x \in \mathbb{X}$, the algorithm can again select a point according to the current uncertainty and thus iteratively draw a sample of informative points.

More technically, let $K_{\mathbb{X}\mathbb{X}} \in \mathbb{R}^{|\mathbb{X}| \times |\mathbb{X}|}$ be a (possibly infinitely large) covariance kernel matrix. The entry at row s and column t (here denoted by $K_{\mathbb{X}\mathbb{X}}(x_s, x_t)$) indicates the a priori covariance $k(x_s, x_t),\ s, t \in [1, |\mathbb{X}|]$. The entries on the diagonal contain the a priori variance $k(x, x)$ for all $x \in \mathbb{X}$ respectively. In our case, k is the RBF kernel function. Therefore $k(x, x) = K_{\mathbb{X}\mathbb{X}}(x)$ is the same for all $x \in \mathbb{X}$ (homogeneous kernel, see Subsec. 1.1.5).
Moreover, let $K_{X_i X_i}$ be a principal submatrix of $K_{\mathbb{X}\mathbb{X}}$ that only includes columns and rows of points that were selected in iteration i. It is also a kernel matrix but just for a subset of $\mathbb{X}$. In the beginning ($i = 0$), it is empty.

Furthermore, let $V \in \mathbb{R}^{|\mathbb{X}|}$ be a (possibly infinitely long) vector that indicates the variance for each $x \in \mathbb{X}$. In contrast to the kernel matrix $K_{\mathbb{X}\mathbb{X}}$, it changes with every iteration. For a start (iteration index $i = 0$), $V_0(x)$ is the same for all s, because it exactly contains the diagonal of $K_{\mathbb{X}\mathbb{X}}$. Later ($i \in \{1, .., \mathrm{k}\}$, k denoting the final number of points), the entries in V_i change with every iteration because certain points are evaluated and thus more is known, uncertainty is reduced which leads to different variances among different points.
This is formalized by the formula:

$$V_i(x) = \begin{cases} k(x, x) & \text{for i} = 0 \\ K_{xx} - K_{xX_{1:i-1}} K^{-1}_{X_{1:i-1}X_{1:i-1}} K_{X_{1:i-1}x} & \text{for i} \in \{1, .., \mathrm{k}\} \end{cases}$$

Because this algorithm aims for reducing uncertainty in every iteration, the choice of points depends on the variance function. More precisely, the variance function constitutes an unnormalized probability density function: the less is known about a point x, the stronger its variance is, and the more likely it is sampled. This relation is even proportional:

$$p(x \mid X_{i-1}) \propto V_i(x)$$

To ensure that proportionality in point selection, Hennig and Garnett (2018) use the integrated $V_i(x)$ (the cumulative probability density function $P(x)$) for the next steps. Now, the highest (and last, because $V_i(x) > 0\ \forall x \in \mathbb{X}$) value of the cumulative probability density function $\max_{x \in \mathbb{X}} P(x) = z$ is found so that $u \in]0, z]$ can be generated randomly. Lastly, their algorithm finds the point x for which $P(x) = u$ holds by running a bisection search. This point now is chosen to be the next one that is included in X. This procedure is also illustrated in Figure 2.1.
We assume that the final number of points k is known before, and stop after iteration k. The algorithm then outputs a sample of k different points that (in expectation) display rather strong entropy.

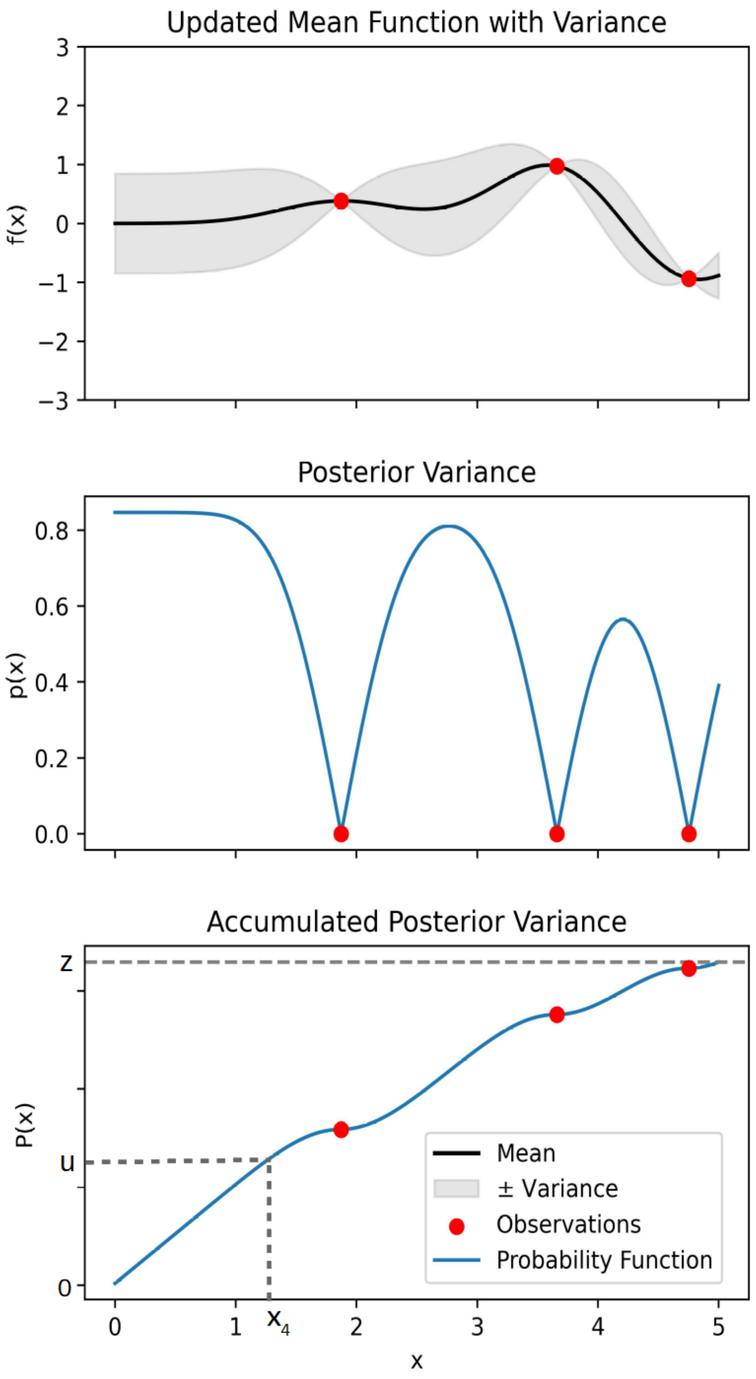

Figure 2.1: Greedy sampling after a third iteration. Top: so far learned/expected function f (GP posterior) with variance bounds. Center: posterior variance which is also the probability density function characterizing the next choice of points. Bottom: accumulative probability density function. $u \in [0, Z]$ is randomly generated and determines the point x_4 selected in iteration 4 proportionally to the slope of $P(x)$ which is $p(x)$. Plot drawn using Pedregosa et al. (2011).

2.1.2 Greedy Sampling Extended to d Dimensions

A strength of the Greedy Sampling algorithm proposed by Hennig and Garnett (2018) is that it is not limited to the one-dimensional case. Though we will now shortly introduce the two-dimensional case, this way of extending the algorithm can, in theory, be done to $d \leq \infty$ dimensions.
The kernel function is now defined for a tuple of vectors and the variance function can be thought of as a landscape.
Firstly, if we sample out of $\mathbb{X} = \mathbb{R}^2$, the algorithm tries to learn a function on an area, a landscape. The kernel function is now defined for a tuple of vectors $((x_1, x_2)^T, (y_1, y_2)^T) \in \mathbb{X} \times \mathbb{X}$ and the variance function can be thought of as a landscape. In the beginning, it looks like a plane that is parallel to both axes and whose height is determined by V_0.
The sampling step is conducted analogously to the one-dimensional case with the difference that firstly the whole landscape of V_i is projected onto one dimension. This function is then integrated into a cumulative distribution. Again, the maximum value of this function is denoted with z_1, u_1 is sampled out of $]0, z_1]$ and x_{i_1} is found such that $P(x_{i_1} \mid X_{i-1}) = z_1$.
Given x_{i_1}, x_{i_2} is now determined by making a cross-section of the V_i-landscape at position x_{i_1} - we consider $V_i(x_{i_1}, x_{i_2})$ with a fixed x_{i_1}. From here, it is proceeded just as in the one-dimensional case:

$$\int_{x_{i_2} \in \mathbb{X}} V_i(x_{i_1}, x_{i_2}) \; dx_{i_2} = P(x_{i_2} \mid X_i, x_{i_1} \in X_{i+1}),$$

Again, $\max_{x_{i_2} \in \mathbb{X}} P(x_{i_2} \mid X_i, x_{i_1} \in X_{i+1}) = z_2$. u is sampled randomly such that $u \in]0, z_2]$. x_{i_2} is found by the condition $P(x_{i_2} \mid X_i, x_{i_1} \in X_{i+1}) = u$. That determines the next point (x_{i_1}, x_{i_2}) and V_{i+1} can be calculated for the next step.

With that, it is said how greedy sampling works. However, it is not yet clear how it is linked to DPPs. That will be examined in the next Subsection.

2.1.3 How It Is Connected to DPPs

The key is that V_0 is completely determined by the Gaussian kernel function k. Because the Gaussian kernel is homogeneous, $V_0(x)$ is the same for any $x \in \mathbb{X}$. For $i > 0$ V_i is computed as given above. According to Hennig and Garnett (2018), the probability for a sample $\{x_1, ..., x_k\}$ to be drawn is then

$$p(x_1, ..., x_k) = Z \prod_{i=1}^{N} p(x_i \mid X_{1:i-1}) = Z \prod_{i=1}^{N} \frac{1}{N - i + i} V_i(x_i).$$

In the case of an orthogonal kernel, this is the same distribution as for a (projection) DPP (Hough et al., 2006).

2.1.4 How Long It Takes

Across the entire algorithm, the step dominating the run time is to compute V_i in every iteration i. It requires the matrix inverse $K^{-1}_{X_{1:i-1}X_{1:i-1}}$ while $X_{1:i-1}$ is growing with i. Because we know that k is the final sample size, this can be done in $O(\mathrm{k}^3)$. All the other steps in one iteration can either be conducted in $O(1)$ (sampling u) or in $O(\log(b))$ with b being the upper bound of $\mathbb{X}$ (interval bisection search for $x : P(x) = u$). Depending on the implementation and accuracy, computing $p(x)$ and $P(x)$ can be done in less than $O(\mathrm{k}^3)$.

Taking that together, if the algorithm of Hennig and Garnett (2018) drew exact DPP samples it would be a very fast ($O(\mathrm{k}^3)$) solution. However, the equation for the probability given in Subsection 2.1.3 does not hold in the case of the Gaussian kernel. The reason for that will be examined in the following section.

2.2 The Algorithm According to Hennig and Garnett (2018) Is not Exactly a k-DPP

2.2.1 The Reason: The Chain Rule

Though the resulting samples generated by the algorithm from Hennig and Garnett (2018) are similar to DPP samples they are not exactly the same.
The principle issue is that the equation $p(x_1, ..., x_k) = Z \prod_{i=1}^{N} p(x_i \mid X_{1:i-1})$ only holds for projection DPPs (which is also the context in which it was used by Hough et al. (2006) on which Hennig and Garnett (2018) refer back to). This is because of the requirements of the probability chain rule. Given two (it can of course be extended to n) random variables - say $\mathbf{x_1}$ and $\mathbf{x_2}$ (first and second point) - the rule says:

$$\begin{aligned} &p(\mathbf{x_1} = x_1, \mathbf{x_2} = x_2) \\ &= p(\mathbf{x_1} = x_1 \mid \mathbf{x_2} = x_2)p(\mathbf{x_2} = x_2) \\ &= p(\mathbf{x_2} = x_1 \mid \mathbf{x_1} = x_1)p(\mathbf{x_1} = x_1) \\ &= p(\mathbf{x_2} = x_2, \mathbf{x_1} = x_1) \end{aligned}$$

Transferring that onto our scenario means that for applying the chain rule correctly, the order in which the points are selected must not matter for the joint probability.

To see on an intuitive level why this is not given in the case of greedy sampling, assume a domain with three elements with equal distance out of which the algorithm should choose 2 points (k = 2). Let the three elements be ordered such that we denote them with x_{left}, x_{middle}, and x_{right} (see Fig. 2.2). In the beginning, every point has the same probability to be chosen because of the homogeneous Gaussian kernel ($V_0(x_{left}) = V_0(x_{middle}) = V_0(x_{right})$). However, after the first step, variance differs depending on the first step: If x_{right} was chosen firstly ($X_1 = \{x_{right}\}$, left side in Fig. 2.2), the variance for its neighbor x_{middle} is less than for x_{left}. Because the variances are proportional to the probabilities for the next points to be chosen, the probabilities for these

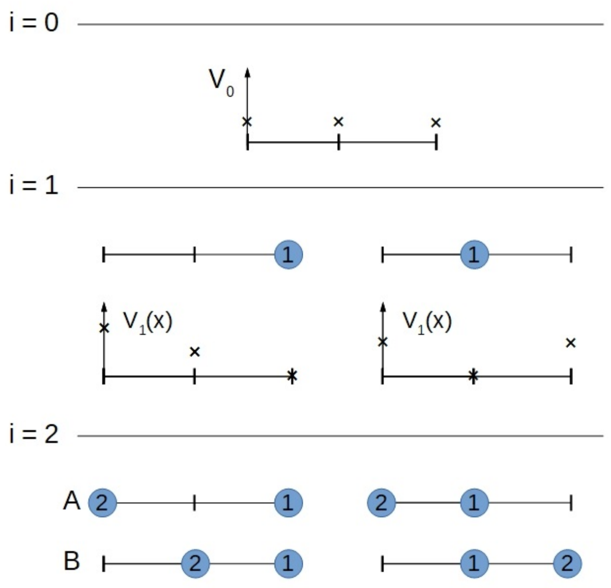

Figure 2.2: Scheme illustrating that for greedy sampling the order of selecting the elements is relevant. The horizontal line represents the domain in every plot. The numbers in circles indicate the iteration in which the element at the circles' position was drawn. $i = \{0, 1, 2\}$ indicates the iteration.
Left: $x_1 = x_{right}$. Right: $x_1 = x_{middle}$. That leads to different variance distributions over the remaining points. Hence, $p(x_1 = x_{right}, x_2 = x_{middle}) \neq p(x_1 = x_{middle}, x_2 = x_{right})$.

points to be taken are different.

If x_{middle} was chosen at first ($X_1 = \{x_{middle}\}$, right side in Fig. 2.2), the variance for x_{left} and x_{right} is the same - so are their probabilities to be chosen in the next iteration:

$$\begin{aligned} p(x_2 = & x_{right} \,|X_1 = \{x_{middle}\}) \\ p(x_2 = & \; x_{left} \,|X_1 = \{x_{middle}\}) \end{aligned}$$

Because there are no other events possible and the sum of the probabilities of all possible events must be 1, both probabilities, given $X_1 = \{x_{middle}\}$, are 0.5. With that, it can also be said that

$$\begin{aligned} p(x_2 = & x_{middle} \,|X_1 = \{x_{right}\}) \\ \neq p(x_2 = & \; x_{left} \,|X_1 = \{x_{right}\}) \\ \Rightarrow p(x_2 = & x_{middle} \,|X_1 = \{x_{right}\}) \\ \neq 0.5 = p(x_2 = & \; x_{right} \,|X_1 = \{x_{middle}\}) \end{aligned}$$

This shows that order is relevant for greedy sampling and that we can thus not apply the chain rule and that in general a k-DPP and greedy sampling do not provide the same samples in general.

2.2.2 How To Make the Chain Rule Applicable

Although it comes with challenges to use the chain rule for DPPs in a valid way, there are several approaches to realize that. Firstly, one could simply use a projection kernel. The orthogonal property of these kernels fulfills the condition, that the order of selection of points is indeed irrelevant (Gautier, 2019).
However, sometimes non-orthogonal kernels are needed. In that case, there is also a way to make use of the chain rule which is used in the spectral method (see Subsec. 1.1.7). Although the algorithm described by Gautier et al. (2019) is a correct and helpful procedure to make it possible to apply the chain rule even for non-orthogonal kernels, it requires an eigenvalue decomposition. With that, its run time is again cubically depending on the domain size. Especially in the case of a continuous domain this is a problem (one has to choose a grid trading off the accuracy of results and the size/run time).
To draw a conclusion, the initial problem to find an exact and fast DPP sampler is yet not solved. Therefore, we will discuss in the next chapter whether we can find guarantees for the algorithm from Hennig and Garnett (2018) and look for different approaches to find informative samples.

Chapter 3

Our Approaches for Finding Gaurantees for Greedy Sampling

The greedy sampling algorithm proposed by Hennig and Garnett (2018) does not exactly draw DPP samples. Yet, it might be possible to find statements about how much it deviates from the optimal DPP solution. Here, we point out three major approaches for finding such statements. The core idea is always to take a procedure of with known relations to a DPP (how close it is to a DPP) and then figure out, how it is connected to the greedy sampling algorithm. Therefore, we will now look at procedures about which we know the relation to DPPs and which seem like having similar key characteristics as greedy sampling to finally derive a connection between DPPs and greedy sampling. Because there are many procedures that one could take into account, Figure 3.1 provides a graphical scheme to receive an overview of the different approaches we analyzed in this thesis. Ultimately, the only approach that provides guarantees for greedy sampling originates from a greedy entropy maximization algorithm, these guarantees are hard to interpret. In the following sections, we will present three approaches we considered in more detail.

3.1 On the Cholesky-Based Method

The Cholesky-based method (CBM) is an exact DPP sampling procedure (see Subsec. 1.1.7). Therefore, knowing the relation to greedy sampling was very helpful as it was an immediate relation to a DPP.

The cause why one could think in the first place, that the CBM and greedy sampling are somewhat significantly connected, are structural and probabilistic similarities: First of all, both procedures work iteratively. Both of them sample the next point according to the already sampled points using the chain rule.

Having said this, there are great differences: The CBM does for instance not iterate according to the probabilities of the points in $\mathbb{X}$ but iterates through the whole domain. With that, it also rejects points such that for x_i it is not only relevant which points have been taken, but also which have been rejected so far. Finally, the product determining $p(X)$ contains every point in $\mathbb{X}$. With that, the procedure is order-invariant and thus the chain rule is applicable. Because greedy sampling is introduced for continuous domains,

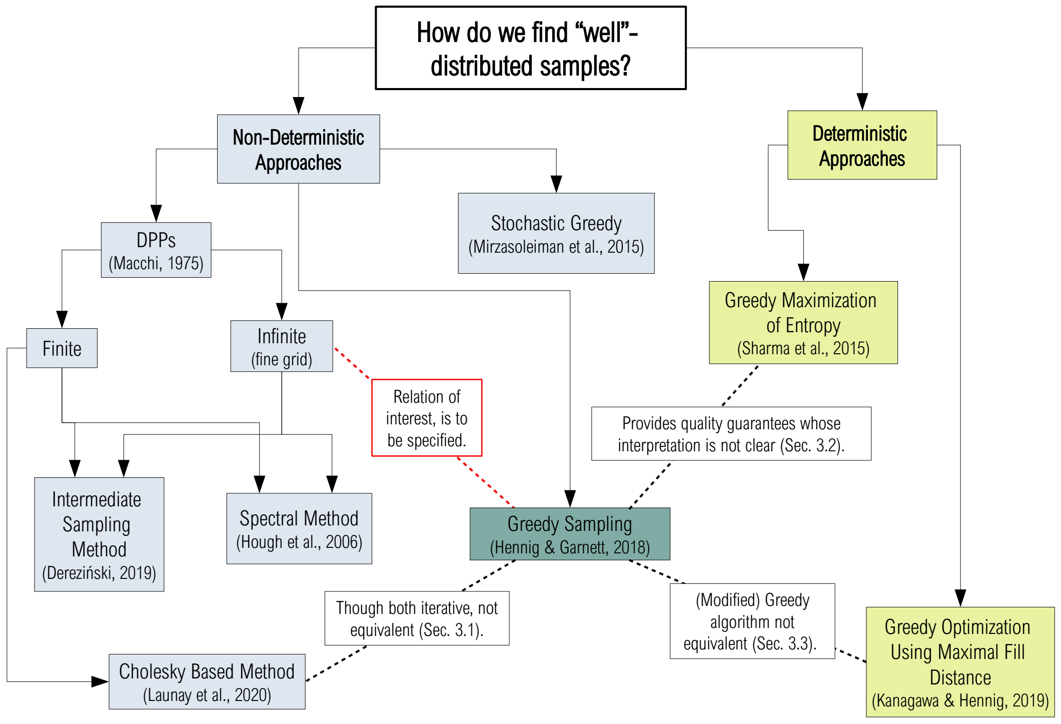
How do we find "well"-distributed samples?
Non-Deterministic Approaches
Deterministic Approaches
Stochastic Greedy (Mirzasoleiman et al., 2015)
DPPs (Macchi, 1975)
Greedy Maximization of Entropy (Sharma et al., 2015)
Finite
Infinite (fine grid)
Relation of interest, is to be specified.
Provides quality guarantees whose interpretation is not clear (Sec. 3.2).
Intermediate Sampling Method (Dereziński, 2019)
Spectral Method (Hough et al., 2006)
Greedy Sampling (Hennig & Garnett, 2018)
Though both iterative, not equivalent (Sec. 3.1).
(Modified) Greedy algorithm not equivalent (Sec. 3.3).
Greedy Optimization Using Maximal Fill Distance (Kanagawa & Hennig, 2019)
Cholesky Based Method (Launay et al., 2020)

Figure 3.1: Different approaches for generating informative samples that are considered here. The relation between an (infinite) DPP and greedy sampling has to be derived from already known relations. Three of them are analyzed in this thesis with the results summarized in the white boxes.

it can not simply iterate through the whole domain apart from the fact that it is a core characteristic that greedy sampling chooses the next point exactly by the current variance distribution.
With that, greedy sampling can not simply be transferred into a CBM. Apart from that, we know that applying the chain rule is not valid for greedy sampling (see Subsec. 2.2.1). Altogether, the differences between the Cholesky-based method and greedy sampling are so fundamental that it is not possible to derive guarantees from the CBM for greedy sampling.

So far, we have tried to find out, how far the greedy sampling algorithm is away from an exact k-DPP on a structural level. In the next two approaches, in contrast, we will try to find out how far the outcome of greedy sampling is away from an approximating procedure.

3.2 Guarantees for Greedy sampling's Entropy

As introduced in Section 1.3, we use entropy as a quality measurement of a sample. In this approach, we try to show that there is a guarantee for the relation between DPP entropy and greedy sampling entropy.

3.2.1 The Proof Idea

For this approach, the submodular property of entropy (see Subsec. 1.3.4) is central. From Nemhauser et al. (1978) we know that there is a guarantee about how well a greedy algorithm approximates a monotone submodular function $f(x)$ in relation to the optimal solution:

$$f_{greedy}(x) \geq \left(1 - \frac{1}{e}\right) f_{optimal}(x).$$

This guarantee can be applied to our DPP setting when talking about the DPP-sample with the maximal entropy, the mode of the DPP. Then, a greedy maximization of entropy algorithm (Sharma et al., 2015) provides a sample with entropy that is at least $(1 - \frac{1}{e}) f_{DPP}(x_{mode})$. However, we are not interested in a single sample but in a distribution of samples - in a sampling setting.
Hence, we aim for a proof that shows that the guarantee proposed by Nemhauser et al. (1978) is transferable to the sampling setting. To do so, we will now derive, what exactly we want to prove.

Let π_{DPP} and π be policies to select points. π_{DPP} denotes a DPP, π denotes the policy whose relation to the DPP we are interested in. We can write π_{DPP} in the form of a probability distribution (see Sec. 1.1):

$$\pi_{DPP}(X) = \frac{\det(K_{XX})}{Z} = \frac{\exp(\log(\det(K_{XX})))}{Z}$$

Because we are interested in the relation between π and π_{DPP}, we consider the Kullback-Leibler divergence that compares two distributions:

Start with the definition of the Kullback-Leibler divergence:

$$D_{KL}(\pi \mid \pi_{DPP}) = \sum_{X \in \mathbb{X}^k} \pi(X) \log\left(\frac{\pi(X)}{\pi_{DPP}(X)}\right)$$

Split the fraction:

$$= \sum_{X \in \mathbb{X}^k} \pi(X) \log(\pi(X)) - \sum_{X \in \mathbb{X}^k} \pi(X) \log(\pi_{DPP}(X))$$

Use the definition of the Shannon entropy:

$$= -H_{k_{X \sim \pi}}[X] - \sum_{X \in \mathbb{X}^k} \pi(X) \log(\pi_{DPP}(X))$$

Use the definition of a DPP and split the fraction:

$$= -H_{k_{X\sim\pi}}[X] - \sum_{X\in\mathbb{X}^k} \pi(X)\log(\det(x)) + \sum_{X\in\mathbb{X}^k} \pi(X)\log(Z)$$

$$= -H_{k_{X\sim\pi}}[X] - \mathbb{E}_{X\sim\pi}[\log(\det(K_{XX}))] + \log(Z)$$

The Kullback-Leibler divergence can only be positive. Thus:

$$0 \leq -H_{k_{X\sim\pi}}[X] - \mathbb{E}_{X\sim\pi}[\log(\det(K_{XX}))] + \log(Z)$$

Switch the signs:

$$0 \geq \quad H_{k_{X\sim\pi}}[X] + \mathbb{E}_{X\sim\pi}[\log(\det(K_{XX}))] - \log(Z)$$

Rearrange the terms:

$$\log(Z) \geq \quad H_{k_{X\sim\pi}}[X] + \mathbb{E}_{X\sim\pi}[\log(\det(K_{XX}))]$$

If π was the same as p (the strongest statement, in our case), D_{KL} was 0 and hence, $\log(Z) = H_{k_{X\sim\pi}}[X] + \mathbb{E}_{X\sim\pi}[\log(\det(K_{XX}))]$. However, because we know that in our case π (greedy sampling) is no DPP, we try to show lower bound guarantees for the term $H_{k_{X\sim\pi}}[X] + \mathbb{E}_{X\sim\pi}[\log(\det(K_{XX}))] =: F(X_\pi)$. The idea is to find a lower bound following the proof by Nemhauser et al. (1978). We conducted a proof that realizes that transfer that can be found attached (Greedy Approximation for Monotone Submodular Soft-Maximization). It states in Lemma 2 that

$$F(X_{greedy_{\mathrm{k}}}) \leq \left(1 - \frac{1}{e}\right) F(X_{DPP_{\mathrm{k}}})$$

with $X_{greedy_{\mathrm{k}}} \sim \pi_{greedy}$ denoting a k-sized sample selected by greedy sampling and $X_{DPP_{\mathrm{k}}} \sim \pi_{DPP}$ a k-DPP sample.
Though that final statement testifies exactly the required property, the proof comes up with some challenges. Firstly, the proof defines a way of understanding greediness that is not shown to be equivalent to the understanding in the greedy sampling that is used here. In fact, the third section in the attached proof document outlines the proof idea such that we account that equivalence to be plausible.
Secondly, the proof does not exactly show that relation for policies that generate sets but lists. This is relevant because when talking about entropy (which is to some extent average information content), the order in which points are selected also contains information. For now, we will proceed considering lists. Because our proof also includes the extension to sets, we will also consider that setting later in this Subsection.
Thirdly, Nemhauser et al. (1978) assume that the function that is to be maximized is monotone (nondecreasing). That property is also required for $\log(\det(K_{XX}))$ in our proof though we did not yet show that it is fulfilled. That is why the next subsection will investigate how the assumptions the proof makes can be satisfied and what possible modifications do to the guarantee and its meaning.

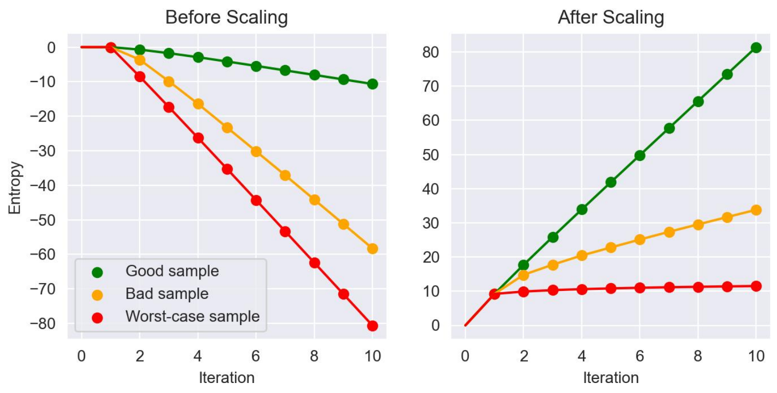

Figure 3.2: Entropies of different samples with size k = 10 before and after scaling. Good sample: evenly distributed points. Bad sample: neighbor points. Worst-case sample: the same point selected ten times. $\mathbb{X}$ is a set of 50 evenly distributed number within $[-4, 4]$. Noise $\Lambda = 0.0001$ was added onto the diagonal.

3.2.2 Monotonicity of Entropy

As stated in Subsection 1.3.4, entropy is not necessarily monotone increasing with respect to sample size. Therefore, we will now point out two ways to guarantee monotonicity of $\log(\det(K_{XX}))$ and make the proof still work.

Ensure Monotonicity by Scaling Up the Kernel Matrix

A central insight for this approach is that

$$\det(A) = \prod_{i=1}^{n} \lambda_i$$

for $A \in \mathbb{R}^{n \times n}$. Because the definition of an eigenvalue λ for an eigenvector v is

$$Av = \lambda v$$

it holds for a scaled matrix

$$\alpha A v = \alpha \lambda v$$

meaning that eigenvalues are scaled in the same way as the matrix itself.

Therefore, whenever there exists a minimal eigenvalue λ_{min} of a kernel matrix K_{XX}, it is possible to guarantee monotonicity for the $\det(K_{XX})$ by scaling K_{XX} with λ_{min}^{-1}. Sharma et al. (2015) used the same trick in their paper to guarantee the monotonicity of entropy.

However, K_{XX} must be scaled once in a sampling procedure, because updating $\alpha_{monotone}$ or shorter α_m in each iteration would lead to hardly interpretable guarantees in the end. Thus, we suggest a way to once find the smallest possible eigenvalue and use its inverse for the whole sampling procedure.

For finding the smallest possible eigenvalue we need to find the worst-case sample X_{worst} and determine λ_{min} for $K_{X_{worst}X_{worst}}$. Intuitively, the worst sample that can be selected when aiming for diversity is a sample of k times the same point. Because this is possible neither in DPP nor greedy sampling, we draw on a trick introducing an arbitrary small noise term Λ that we add on the diagonal in the kernel matrix $K_{\mathbb{X}\mathbb{X}}$. That introduces a little additive difference on the so far determined variance. The main effect of that is that $K_{x,x}$ becomes greater than 0, no matter whether x was already selected or not. Then, it is even possible to draw the worst sample containing x k times.
Given that, it is easy to determine $K_{X_{worst}X_{worst}} \in \mathbb{R}^{\mathrm{k}\times\mathrm{k}}$:

$$K_{X_{worst}X_{worst}} = \begin{pmatrix} \Lambda & 1 & ... & 1 \\ 1 & \Lambda & ... & 1 \\ ... & ... & ... & ... \\ 1 & 1 & ... & \Lambda \end{pmatrix}$$

Because we know $K_{X_{worst}X_{worst}}$ so exactly, deriving $\alpha_m = \lambda_k^{-1} = \lambda_{min}^{-1}$ is easy. Then it holds that $\log(\det(\alpha_m K_{XX}))$ is monotone decreasing with respect to the sample size until it reaches k. To be exact, we have not shown that the smallest eigenvalue of $K_{X_{worst}X_{worst}}$ is indeed the smallest eigenvalue from all possible sets. However, even if the smallest possible eigenvalue had to be computed differently, let $\alpha_m = \lambda_{min}^{-1}$ for the smallest possible eigenvalue. Then, $\det(K_{X_iX_i})$ receives a new factor in every iteration that is greater than 1 which makes $\log(\det(K_{X_iX_i}))$ receive a positive additive term in each iteration. Then, with respect to i $\log(\det(K_{X_iX_i}))$ is a monotone function.

The question that remains open, is, whether this way of ensuring monotonicity solely changes the eigenvalues or also the distributions such that the guarantees can not be applied to the original context
First of all: the noise term added to the matrix of course changes the probabilities of points and samples to be selected. However, it can be chosen arbitrarily small such that one should be aware of its potential effect but it should not play such a big role in practice. Furthermore, scaling up the matrix does, at least for a k-DPP and greedy sampling not change the point selections: the k-DPP maximizes the determinant of K_{XX} that grows linearly with α_m. Hence, the selection of points is the same. For "real" DPPs the eigenvalues are bounded by 1 (Kulesza & Taskar, 2012) which is also necessary for the computation of the expected sample size (see Subsec. 1.1.3 or 1.1.3).

The greedy sampling algorithm selects x_i proportionally to its variance that is in the scaled case computed with

$$V_i(x) = \alpha_m K_{xx} - \alpha_m K_{xX_{1:i-1}} \alpha_m K^{-1}_{X_{1:i-1}X_{1:i-1}} \alpha_m K_{X_{1:i-1}x}$$

which is

$$V_i(x) = \alpha_m K_{xx} - \alpha_m K_{xX_{1:i-1}} K^{-1}_{X_{1:i-1}X_{1:i-1}} K_{X_{1:i-1}x}$$

such that $V_i(x)$ also depends linearly on alpha. Because α_m is the same for every x, the proportionality factor increases with α_m but the point selection is the same as for an unscaled kernel matrix. In conclusion, the two procedures we are focusing on, choose the same points as in the non-scaled case.

With that, the proof that extends guarantees on entropy in the deterministic samples to the sampling setting works if we scale the kernel matrix. Yet, it is not clear whether the guarantee is also still meaningful. Mathematically, it looks a bit different:

$$\log(\det(\alpha_m(K_{X_{GS}X_{GS}} + \Lambda)) \geq (1 - \frac{1}{e}) \log(\det(\alpha_m(K_{X_{DPP}X_{DPP}} + \Lambda)))$$
$$\log(\det(K_{X_{GS}X_{GS}} + \Lambda) + \log(\alpha_m)) \geq (1 - \frac{1}{e}) \log(\det(K_{X_{DPP}X_{DPP}} + \Lambda)) + (1 - \frac{1}{e}) \log(\alpha_m)$$
$$\log(\det(K_{X_{GS}X_{GS}} + \Lambda)) \geq (1 - \frac{1}{e}) \log(\det(K_{X_{DPP}X_{DPP}} + \Lambda)) - \frac{1}{e} \log(\alpha_m)$$

Depending on k and Λ $\log(\alpha_m)$ could be such a big term that the guarantee is not a strong statement anymore. However, if $\alpha_m \in [0,1]$ (that happens if $\lambda_{min} > 1$) then $\log(\alpha_m) < 0$ such that the guarantee becomes even stronger. Either way, if the proof holds, we had a lower bound of the entropy of greedy sampling samples in the first place. One more advantage of this guarantee is that we exactly know α_m and could hence interpret the guarantee for a certain case better.
On that point, it is questionable why the guarantees change with scalars of the kernel matrix whereas the point selection is not influenced by α_m. Why should the unscaled entropy of the samples change then? Questions regarding comparing entropies with different scalars are addressed in the next subsection.

Ratio Scala for Entropies

In the proceeding section, it was shown that for one policy π there can be found different entropy guarantees depending on the scalar in front of the kernel matrix K_{XX} characterizing π. However, scaling the kernel matrix does not change the point selection. Thus, the guarantees for entropy we considered so far are either not meaningful in general, or there has to be found a scaling level that makes interpretations possible.
Talking in levels of measurement, the guarantees we are interested in, require a ratio scale which, in turn, requires an absolute zero. The choice of a meaningful absolute zero is (just as in our case) arguable. Here, a factor α_0 such that $\log(\det(\alpha_0 K_{XX})))$ is plausible to be an absolute zero is needed. We therefore suggest α_0 to be the factor such that $\log(\det(\alpha_0 K_{X_{worst}X_{worst}})) = 0$.

$$\begin{aligned}
& \log(\det(\alpha_0 K_{X_{worst}X_{worst}})) = 0 \\
\Leftrightarrow\ & \det(\alpha_0 K_{X_{worst}X_{worst}}) && = 1 \\
\Leftrightarrow\ & \alpha^{\mathrm{k}} \det({}_0 K_{X_{worst}X_{worst}}) && = 1 \\
\Leftrightarrow\ & \alpha^{\mathrm{k}} \prod_{i=1}^{\mathrm{k}} \lambda_i && = 1 \\
\Leftrightarrow\ & \alpha^{\mathrm{k}} && = \frac{1}{\prod_{i=1}^{\mathrm{k}} \lambda_i} \\
\Leftrightarrow\ & \alpha && = \sqrt[\mathrm{k}]{\frac{1}{\prod_{i=1}^{\mathrm{k}} \lambda_i}}
\end{aligned}$$

However, one could argue that this choice of α_0 is not reasonable. For instance, for an empty set $X_{\{\}} = \{\}$, the $\log(\det(K_{X_{\{\}}}))$ is 0, just as $\log(\det(K_{X_{worst}X_{worst}}))$ though X_{worst} obviously contains more information than $X_{\{\}}$. Thus, α_0 might actually be better chosen greater. In turn, this argument does not hold because the required comparison should be interpretable for samples with size $\mathrm{k} \neq 0$. Arguing with the empty set is hence not valid. Still, α_0 could be chosen differently. For instance, one could use the reference to "normal" DPPs whose eigenvalues are always between 0 and 1 and find α_0 such that the eigenvalues of K_X are between 0 and 1 for any sample X with $|X| = \mathrm{k}$. Then, however, the log-determinant of the kernel matrix would not be monotone anymore and, with increasing k possibly even become negative which might not be a good property for α_0.

Extension for Sets

As stated above, the $\left(1 - \frac{1}{e}\right)$-ratio guarantee could also be shown if we sample sets instead of lists. However, our proof only works if not only the monotonicity of entropy but even a slope of $\frac{1}{k}\log(k!)$ has to be guaranteed. To ensure that, the kernel matrix has to be scaled up by $\alpha_{slope} = \frac{k!^{\frac{1}{k}}}{\lambda_{min}}$ with λ_{min} being the smallest possible eigenvalue. Again, the questions of the interpretability of the ratio guarantee and the back transfer have to be clarified.

In conclusion, scaling the kernel matrix can ensure the required properties of entropy but also comes with new questions. Therefore, we will now look into another approach to treat the lack of monotonicity of entropy using a different concept which is a priori monotone.

Use Information Gain Instead

The idea behind using information gain is to find another function F that has a similar meaning to entropy but more convenient mathematical properties. Just like entropy, information gain indicates something about the informativeness of points in a sample but is both submodular and monotone increasing. Note that we only ensure monotonicity and not a certain slope which is required for making the proof for sets valid. Therefore, with this new approach, we can only validate the proof guaranteeing a certain quality of lists drawn by greedy sampling.
According to Srinivas et al. (2009), information gain is defined as the reduction of uncertainty about a function f the algorithm aims to learn by sampling and evaluating a new point:

$$IG_i(y_{X_i}, f) = h(y_{X_i}) - h(y_{X_i} \mid f).$$

Here, y_{X_i} denotes what is know about the function in iteration i: $y_{X_i} = f_{X_i} + \epsilon_{X_i}$ where $f_{X_i} = [f(x)]_{x \in X_i}$ and ϵ_{X_i} is an error following a Gaussian distribution $N(0, \sigma^2 I)$. h denotes the differential entropy. Because all y_{X_i} together form a multidimensional Gaussian distribution, the formula to compute information gain is

$$IG_i(y_{X_i}, f) = \frac{1}{2} \log(\det(I + \sigma^{-2} K_{X_i X_i}))$$

where I stands for the identity matrix and $K_{X_i X_i}$ for the kernel matrix of sample X_i (Srinivas et al., 2009).
Looking at the formula for information gain reveals that information gain also has mathematical similarities to entropy because it also contains $\log(\det(K_{XX}))$ with a σ-scaled kernel matrix. However, the noise Term I is a relevant difference that affects the point selection of a greedy maximization algorithm for information gain. Thus, statements that hold for such a greedy (sampling) algorithm might are not generally transferable on greedy (sampling) maximization of entropy. Therefore, deriving guarantees for information gain maximization might not be of any help to find guarantees for the greedy sampling algorithm we consider here.

Altogether, the approach to finding a guarantee for the entropy of samples generated by greedy sampling leads to some guarantees that might actually not be interpretable. However, we considered one more approach to transfer known guarantees on greedy sampling.

3.3 On a Weak Greedy Max-Dist Algorithm

In their paper, Kanagawa and Hennig (2019) introduced the concept of weak adaptivity in the context of adaptive Bayesian quadrature. They suggest an algorithm that is, just as greedy sampling, iterative and that tries to select points in a reproducing kernel Hilbert space (RKHS) such that the very same is covered by the selected points as good as possible. Seeing it from a different perspective, that is just the same as aiming for the most diverse sample.
Kanagawa and Hennig (2019) provide guarantees about the distance between the spanned space of all points selected so far and the point that is most far away. If we find a connection between both concepts like distance and the farthermost point and the procedures of the two algorithms it might be possible to transfer insights from Kanagawa and Hennig (2019) to our setting. Therefore, we will now describe the algorithm for maximum fill distance and afterward, look into parallels to greedy sampling.

3.3.1 Weak Greedy Algorithm

Let $\mathbb{H} \subset \mathcal{H}$ be a subset of the RKHS $\mathcal{H}$. Moreover, let $\mathbb{X}$ be the domain from which the algorithm picks points. It aims to choose points such as their projection into $\mathbb{H}$ cover the very same as good as possible. To do that, the concept of a subspace S_n for a sample of points in the RKHS $H_n = \{h_1, ..., h_n\}$ with

$$S_n := \text{span}(H_n)$$

is used (Buffa et al., 2012; DeVore et al., 2013; Hennig & Garnett, 2018). Based on that the distance between a point h and a subspace S_n is defined as

$$\text{dist}(h, S_n) := \inf_{g \in S_n} \|h - g\|$$

which is exactly the distance between h and its orthogonal projection onto the subspace H_n. With that, the worst-case error e_n can be defined by

$$e_n(\mathbb{H}) := \sup_{h \in \mathbb{H}} \text{dist}(h, S_n).$$

The algorithm aims for minimizing e_n which is the same as approximating $\mathbb{H}$ well.
To do so, in each iteration $i > 0$ the next point $h_{i+1} \in \mathbb{H}$ is chosen such that

$$\text{dist}(h_{i+1}, S_i) \geq \gamma \sup_{h \in \mathbb{H}} \text{dist}(h, S_i)$$

where S_i stands for the span of the so far generated points. For the start, h_1 is chosen such that $\|h_1\| \geq \gamma \sup_{h \in \mathbb{H}}$. $\gamma \in]0, 1]$ is what makes this algorithm weak greedy. For $\gamma = 1$ the algorithm is a pure greedy algorithm. For $\gamma \in]0, 1[$ in contrast, the algorithm is "satisfied" faster and not as strict about finding the current optimal next point.
Kanagawa and Hennig (2019) finally (after showing/considering several properties of their setting) find guarantees for the worst-case error after n iterations. In the next paragraph, we will try to link their procedure to the algorithm by Hennig and Garnett (2018).

3.3.2 Connection to Greedy sampling

Roughly speaking, in the case of greedy sampling, the algorithm aims to learn a function $f(x)$ of which it becomes more certain with each iteration. However, it will never learn the whole function so well that it is sure about every function value $f(x) \forall x \in \mathbb{X}$. On its way, it samples points according to how uncertain it is about its function value. The weak greedy algorithm from Kanagawa and Hennig (2019) aims to find a subset $H \in \mathbb{X}$ such that $\mathbb{H}$ is approximated as well as possible. Similarly to the setting of greedy sampling, it will not exactly describe $\mathbb{H}$ in the end but it chooses the next point by looking at its projections' distance to the projection of the currently selected points which can be seen as a type of uncertainty about $\mathbb{H}$. By regulating the factor γ, it can be decided how greedy the algorithm is.
Even if there might be key similarities, there is the major difference between these two approaches that the weak greedy algorithm is deterministic, unlike greedy sampling. Therefore, we suggest a modification to make the weak greedy algorithm non-deterministic.

Non-Deterministic Modification of the Weak Greedy Algorithm

For making the transfer of assertions on the deterministic approach onto greedy sampling possible we need to add a sampling step in the deterministic algorithm. In greedy sampling, the step that introduces uncertainty is generating a random number $u \in [0,1]$ that determines the next point x_i with the condition $P(x_i) = u$. Our idea is to sample γ (or more exactly γ^2 (see Subsec. 3.3.3) uniformly which determines how greedy the algorithm is in each iteration and thus, what point is chosen next.
In Kanagawa and Hennig (2019), the greedy selection of the next point is chosen with

$$\begin{aligned} \operatorname{dist}(h_{i+1}, S_i) &\geq \gamma \sup_{h \in \mathbb{H}} \operatorname{dist}(h, S_i) \\ \Leftrightarrow \operatorname{dist}(h_{i+1}, S_i)^2 &\geq \gamma^2 \sup_{h \in \mathbb{H}} \operatorname{dist}^2(h, S_i) \end{aligned}$$

because dist is always positive and can be calculated with

$$\operatorname{dist}^2(h_x, S_i) = q^2(x) V_i(x).$$

V_i again denotes the GP posterior variance function given by

$$V_i(x) = v(x,x) = k(x,x) - K_{X_i x} K_{X_i X_i} K_{x X_i}.$$

$q^2(x)$ denotes a proposal density that we set 1 for our purposes. Taking that together, the algorithm chooses the next point such that

$$V_i(x_{i+1}) \geq \gamma^2 \sup_{x \in \mathbb{X}} V_i(x).$$

So far, the selection of points seems very similar. Because we know that γ^2 is sampled randomly, we can use guarantees containing γ^2 using the expectation value for γ^2 instead. Having said this, it remains to be shown that selecting u and γ^2 randomly end up in the same selection of points. We found this to be unsuccessful, unfortunately. In the next subsection, we will describe why sampling γ^2 and u uniformly do not amount to the same thing.

3.3.3 Why Is the Weak Greedy Algorithm not Similar Enough?

For the next point x_i, the weak greedy algorithm selects a point whose variance is at least $\gamma^2 \sup_{x\in\mathbb{X}} V_i(x)$. Let us consider only the worst-case in which x_i is the point whose variance is exactly $\gamma^2 \sup_{x\in\mathbb{X}} V_i(x)$ (that is only possible if $\mathbb{X}$ is dense) which we assume for now. Then, γ^2 can be expressed as

$$\gamma^2 = \frac{V_i(x_i)}{\sup_{x\in\mathbb{X}} V_i(x)}$$

which can be seen as a measure of quality for x_i which is always in $]0,1]$. For $\gamma^2 = 1$, x_i is the best possible point. For γ^2 that is close to zero x_i is not very distant and does not come with much new information.
A difference between γ^2 and u is that u is not connected to a certain meaning. One can not predict according to its value, whether x_i will be more or less informative. We solely know that the choice of x_i will in the end be proportional to its current variance. However, that alone does not mean that these two procedures are not equivalent - if in the modified weak greedy algorithm this proportionality was ensured in a different way, equivalence could still be shown. What is the core of the problem is that γ^2 determines $p(x_i)$ - a probability density function value (if the values would be normalized such that their sum/integral equals one). u, in contrast, determines $P(x_i)$ - a cumulative distribution function value. Thus, to find equivalence, it might be necessary to integrate deviate on one or the other side which later also has to be transferred onto the guarantee such that it is still interpretable.
We did not follow these further thoughts in this thesis. However, maybe there is a potential to find an actual exact equivalence between these two procedures that reveals new information on greedy sampling.

Chapter 4

Conclusion

Altogether, we considered mainly two ways of generating diverse samples of size k in this thesis. Regarding entropy, k-DPPs constitute the optimal solution because they maximize the determinant of the sample's kernel matrix while providing diversity between samples at the same time. Another approach is the greedy sampling algorithm introduced by Hennig and Garnett (2018). It is faster - especially for large domains - and generates both informative but also diverse samples. However, it is not quite as good as a real k-DPP - roughly speaking, because its choice of points is not order-invariant. Here, we tried several ways of finding out how the relation between DPPs and greedy sampling can be exactly described.
Finally, we found a guarantee derived from a greedy maximization of entropy algorithm (Sharma et al., 2015). To be exact, the proof presented here only shows that for lists. Yet, to apply this list-specific guarantee the kernel matrix must be scaled up with $\alpha_m = \lambda_{min}^{-1}$ such that the logarithmic determinant of the kernel matrix of any sample is monotone increasing with respect to the sample size. Considering sets, an even greater scaling factor was necessary. Scaling up the kernel matrix makes it harder to interpret the guarantees which leads to the question of under what conditions such relational statements can be interpreted in the first place. That question should also be asked for guarantees in the existing literature. To make a ratio guarantee meaningful a ratio scale is needed, which again needs an absolute zero. We suggest as an absolute zero a α_0-scaled kernel matrix with $\alpha_0 = \frac{1}{det(K_{X_{worst}})}$ such that the determinant of any samples' kernel matrix (any principal submatrix) is positive. Yet, it is not clear how to transfer guarantees on a α_m-scaled kernel matrix to a guarantee on a α_m-scaled kernel matrix. In their paper, Sharma et al. (2015) included such a transfer "back" to an unscaled matrix that might be helpful to deal with this challenge in future work.
The second approach attempted to establish an equivalence between a weak greedy algorithm maximizing fill-distance (Kanagawa & Hennig, 2019) and greedy sampling. The idea was that covering a space and finding diverse, well-distributed samples could be aims that are similar enough such that guarantees on one can be transferred to the other. However, there was no link providing equivalence between these two procedures found.
Djolonga et al. (2018) derive different guarantees on log-submodular greedy maximization that we could not apply because there were requirements on the function of interest that we could not show to be true for entropy.

Drawing an overall conclusion, we found different ways to approach greedy sampling and its relation to DPPs. One way provided guarantees of whose meaningfulness we are not sure yet. Nevertheless, knowing more about this relation would be helpful as greedy sampling offers informative and fast samples. Thus, thoughts to clarify how close greedy sampling and DPPs are both needed and well invested.

Bibliography

Alaoui, A., & Mahoney, M. W. (2015). Fast randomized kernel ridge regression with statistical guarantees. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf

Bıyık, E., Wang, K., Anari, N., & Sadigh, D. (2019). Batch active learning using determinantal point processes. https://doi.org/10.48550/ARXIV.1906.07975

Buffa, A., Maday, Y., Patera, A. T., Prud'homme, C., & Turinici, G. (2012). A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM: Mathematical Modelling and Numerical Analysis*, *46*(3), 595–603. https://doi.org/10.1051/m2an/2011056

Calandriello, D., Derezinski, M., & Valko, M. (2020). Sampling from a k-dpp without looking at all items. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 6889–6899). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/4d410063822cd9be28f86701c0bc3a31-Paper.pdf

Çivril, A., & Magdon-Ismail, M. (2009). On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, *410*(47), 4801–4811. https://doi.org/https://doi.org/10.1016/j.tcs.2009.06.018

Derezinski, M., Calandriello, D., & Valko, M. (2019). Exact sampling of determinantal point processes with sublinear time preprocessing. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/fa3060edb66e6ff4507886f9912e1ab9-Paper.pdf

Dereziński, M. (2019). Fast determinantal point processes via distortion-free intermediate sampling. In A. Beygelzimer & D. Hsu (Eds.), *Proceedings of the thirty-second conference on learning theory* (pp. 1029–1049). PMLR. https://proceedings.mlr.press/v99/derezinski19a.html

DeVore, R., Petrova, G., & Wojtaszczyk, P. (2013). Greedy algorithms for reduced bases in banach spaces. *Constructive Approximation*, *37*(3), 455–466.

Djolonga, J., Jegelka, S., & Krause, A. (2018). Provable variational inference for constrained log-submodular models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/0c0a7566915f4f24853fc4192689aa7e-Paper.pdf

Gautier, G. (2019). *Exact sampling for k-dpps*. Retrieved January 19, 2023, from https://dppy.readthedocs.io/en/latest/finite_dpps/exact_sampling.html#finite-dpps-exact-sampling-k-dpps

Gautier, G., Polito, G., Bardenet, R., & Valko, M. (2019). Dppy: Dpp sampling with python. *J. Mach. Learn. Res.*, *20*, 180–1.

Gillenwater, J., Kulesza, A., & Taskar, B. (2012). Near-optimal map inference for determinantal point processes. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/6c8dba7d0df1c4a79dd07646be9a26c8-Paper.pdf

Hennig, P., & Garnett, R. (2018, April 17). Exact sampling from determinantal point processes. Retrieved November 1, 2022, from http://arxiv.org/abs/1609.06840

Hough, J. B., Krishnapur, M., Peres, Y., & Virág, B. (2006). Determinantal processes and independence. *Probability Surveys*, *3*. https://doi.org/10.1214/154957806000000078

Kanagawa, M., & Hennig, P. (2019). Convergence guarantees for adaptive bayesian quadrature methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/165a59f7cf3b5c4396ba65953d679f17-Paper.pdf

Kulesza, A., & Taskar, B. (2011). K-dpps: Fixed-size determinantal point processes. *ICML*, 1193–1200. https://icml.cc/2011/papers/611_icmlpaper.pdf

Kulesza, A., & Taskar, B. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, *5*(2), 123–286. https://doi.org/10.1561/2200000044

Launay, C., Galerne, B., & Desolneux, A. (2020). Exact sampling of determinantal point processes without eigendecomposition. *Journal of Applied Probability*, *57*(4), 1198–1221. https://doi.org/10.1017/jpr.2020.56

Lavancier, F., Møller, J., & Rubak, E. (2015). Determinantal point process models and statistical inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *77*(4), 853–877.

Macchi, O. (1975). The coincidence approach to stochastic point processes. *Advances in Applied Probability*, *7*(1), 83–122. https://doi.org/10.2307/1425855

McCurdy, S. (2018). Ridge regression and provable deterministic ridge leverage score sampling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/e1d5be1c7f2f456670de3d53c7b54f4a-Paper.pdf

Micchelli, C. A., Xu, Y., & Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, *7*(12).

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrak, J., & Krause, A. (2015). Lazier than lazy greedy. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9486

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, *14*(1), 265–294.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning* [OCLC: ocm61285753]. MIT Press.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Sharma, D., Kapoor, A., & Deshpande, A. (2015). On greedy maximization of entropy. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning* (pp. 1330–1338). PMLR. https://proceedings.mlr.press/v37/sharma15.html

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design [Publisher: arXiv Version Number: 4]. https://doi.org/10.48550/ARXIV.0912.3995

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

R. Fischer

Tübingen, 01.02.2023 Rahel Fischer

Greedy Approximation for Montone Submodular Soft-Maximization

Abstract

Finding the maximum of a monotone submodular set function is NP-hard, but the greedy approach is known to be an $(1-1/e)$-approximation. Here, we consider a greedy approach to sample from the softmax of monotone submodular set function and show that a probabilistic version of the $(1-1/e)$-approximation guarantee also holds in this setting. We apply the derived guarantees to an approximate sampling algorithm in [HG16] for finite, fixed sized determinantal point processes.

1 Montone Submodular Maximization

Let E be a finite set of items, and for $S \subseteq E$, let $f(S)$ give the value of subset S. Suppose that $f(\emptyset) = 0$ and that f is monotone and submodular, i.e.

- For any S and T with $S \subseteq T \subseteq E$, $f(S) \leq f(T)$
- For any $S, T \subseteq E$, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$

Let $\mathcal{E} = \{S \subseteq E \| S| = k\}$ be the set of all k-element subsets of E. The monotone submodular maximization problem consists in finding $O \in \mathcal{E}$ that maximizes the function f:

$$O = \operatorname*{argmax}_{S \in \mathcal{E}} f(S)$$

The problem is known to be NP-hard. A policy for this optimization problem is a function $\pi : \mathcal{P}(E) \mapsto E$ that maps the elements $S_{1:t-1}$ selected in steps $t = 1, .., k$ to the next choice s_t: $\pi(S_{1:t-1}) = s_t$. We use the slicing notation $S_{i:j}$ to denote the elements selected in steps $i, i+1..., j$. For $j < i$, $S_{i:j} = \emptyset$. The most common choice for a policy is the greedy strategy that maximizes the marginal gain $\Delta_f(s_t | S_{1:t-1}) := f(S_{1:t-1} \cup \{s_t\}) - f(S_{1:t-1})$ in each step t:

$$\pi_{greedy}(S_{1:t-1}) = \operatorname*{argmax}_{s_t \in E \setminus S_{1:t-1}} \Delta_f(S_{1:t-1} \cup \{s_t\})$$

for $t = 1, ..., k$. [NWF78] show that the subset G returned by greedy algorithm π_{greedy} achieves a function values that is at least $(1 - 1/e)$ the optimal one:

$$f(G) \geq (1 - 1/e) f(O).$$

Additionally, [Fei98] show that there is no polynomial time algorithm in general that can do better than this, unless $P = NP$.

2 Monotone Submodular Softmaximization

Instead of the "hard" maximization problem from above, we know consider a soft version of the problem. There, we do not want to select a single optimal subset, but sample subsets such that better subsets appear with larger probability. Formally, we consider a random variable $\mathbf{O}$ over the discrete probability space $(\mathcal{E}, 2^{\mathcal{E}}, P)$ with probability mass function P given by

$$P_{opt}(\mathbf{O} = \{e_1, ..., e_k\}) = \frac{\exp f(\{e_1, ..., e_k\})}{\sum_{S \in \mathcal{E}} \exp f(S)}$$

and known as the soft(arg)max. Instead of deterministic strategies, we now consider stochastic policies. Since we want to allow for sequential sampling strategies, we define a policy π over a product probability space $(E^k, 2^{E^k}, \pi)$:

$$\pi(\mathbf{S}_{1:k} = (e_1, ..., e_k)) = \pi(\mathbf{S}_1 = e_1) \prod_{i=2}^{k} \pi(\mathbf{S}_i = e_i | \mathbf{S}_{1:i-1} = (e_1, ..., e_{i-1}))$$

Every sequential policy π induces a random variable $\mathbf{S}$ over the discrete probability space $(\mathcal{E}, 2^{\mathcal{E}}, P_\pi)$, where P_π is obtained by summing over all permutations perm(S) of the elements in a set S:

$$P_\pi(\mathbf{S} = S) = \sum_{S' \in \text{perm}(S)} \pi(\mathbf{S}_{1:k} = S')$$

We say that a policy π_{opt} is optimal if and only if $P_{\pi_{opt}} \equiv P_{opt}$. For an optimal policy, there is a corresponding variational characterization given by

$$\pi_{opt} = \arg\max_{\pi} \mathbb{E}_{S \sim P_\pi} f(S) + \mathcal{H}(P_\pi)$$

where $\mathcal{H}(p_\pi) := -\sum_{S \in \mathcal{E}} p_\pi(S) \log p_\pi(S)$ is the Shannon entropy and $\mathbb{E}_{S \sim p_\pi} f(S)$ is the expected value of the objective value f of the sampled subsets. To see this, consider the Kullback-Leibler divergence between P_π and P_{opt}

$$\begin{aligned} & D_{KL}(P_\pi || P_{opt}) \\ = & \sum_{S \in \mathcal{E}} P_\pi(S) \log\left(\frac{P_\pi(S)}{P_{opt}(S)}\right) \\ = & \sum_{S \in \mathcal{E}} P_\pi(S) \log\Big(P_\pi(S)\Big) - \sum_{S \in \mathcal{E}} P_\pi(S) \log\Big(P_{opt}(S)\Big) \\ = & \sum_{S \in \mathcal{E}} P_\pi(S) \log\Big(P_\pi(S)\Big) - \sum_{S \in \mathcal{E}} P_\pi(S) \log\left(\frac{\exp(f(S))}{Z}\right) \qquad Z := \sum_{S \in \mathcal{E}} \exp f(S) \\ = & -\mathcal{H}(P_\pi) - \mathbb{E}_{S \sim P_\pi} f(S) + \log Z \end{aligned}$$

and note that π_{opt} achieves the miniminum $D_{KL}(P_{\pi_{opt}} || P_{opt}) = 0$ per definition of optimality.
Since $F(\pi) := \mathbb{E}_{S \sim P_\pi} f(S) + \mathcal{H}(P_\pi)$ is maximized by the optimal policy, we will derive approximation guarantees regarding F and not regarding f as in the previous section.
For (conditional) entropies and the expected values, we introduce the following notation:
For $\mathbf{S}_{1:k} \sim \pi$,

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

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

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

2.1 Greedy sampling

The greedy policy π_{greedy} is defined by the following rule: For all $S_{1:i} \in E^i$:

$$\pi_{greedy}(\mathbf{S}_i = S_i|\mathbf{S}_{1:i-1} = S_{1:i-1})) = \underset{\pi(\mathbf{S}_i=S_i|\mathbf{S}_{1:i-1}=S_{i:i-1})}{\operatorname{argmax}} \mathbb{E}_{\mathbf{S}_i|\mathbf{S}_{1:i-i}=S_{1:i-1}}[f(S_{1:i}) - f(S_{1:i-1})] + \mathcal{H}(\mathbf{S}_i|\mathbf{S}_{1:i-1} = S_{1:i-1})$$

2.2 Theoretical analysis

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

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

Proof of Lemma 1:

$$F(\mathbf{O}_{1:k})$$

$$\overset{\text{definition of } F}{=} \mathbb{E}_{\mathbf{O}_{1:k}}[f(O_{1:k})] + \mathcal{H}(\mathbf{O}_{1:k})$$

$$\overset{f \text{ is monotone}}{\leq} \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\Big[f(O_{1:k} \cup G_{1:i})\Big] + \mathcal{H}(\mathbf{O}_{1:k})$$

$$\overset{\text{telescoping sum}}{=} \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\Big[f(G_{1:i}) + \sum_{j=1}^{k} f(G_{1:i} \cup O_{1:j}) - f(G_{1:i} \cup O_{1:j-1})\Big] + \mathcal{H}(\mathbf{O}_{1:k})$$

$$\overset{f \text{ is submodular}}{\leq} \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\Big[f(G_{1:i}) + \sum_{j=1}^{k} f(G_{1:i} \cup O_j) - f(G_{1:i})\Big] + \mathcal{H}(\mathbf{O}_{1:k})$$

$$\overset{\text{entropy chain rule}}{\leq} \mathbb{E}_{\mathbf{O}_{1:k}\mathbf{G}_{1:i}}\Big[f(G_{1:i}) + \sum_{j=1}^{k} f(G_{1:i} \cup O_j) - f(G_{1:i})\Big] + \sum_{j=1}^{k} \mathcal{H}(\mathbf{O}_j|\mathbf{O}_{1:j-1})$$

$$\overset{\text{cond. can only decrease entropy}}{\leq} \mathbb{E}_{\mathbf{G}_{1:i}}\Big[f(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{O}_j}[f(G_{1:i} \cup O_j) - f(G_{1:i})] + \mathcal{H}(\mathbf{O}_j)\Big]$$

$$\overset{\text{summarize}}{=} \mathbb{E}_{\mathbf{G}_{1:i}}\Big[f(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{O}_j}[f(G_{1:i} \cup O_j) - f(G_{1:i})] + \mathcal{H}(\mathbf{O}_j)\Big]$$

$$\overset{\mathbf{G},\mathbf{O} \text{ indep.}}{=} \mathbb{E}_{\mathbf{G}_{1:i}}\Big[f(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{O}_j}[f(G_{1:i} \cup O_j) - f(G_{1:i})] + \mathcal{H}(\mathbf{O}_j|\mathbf{G}_{1:i} = G_{1:i})\Big]$$

$$\overset{\text{greedyness}}{\leq} \mathbb{E}_{\mathbf{G}_{1:i}}\Big[f(G_{1:i}) + \sum_{j=1}^{k} \mathbb{E}_{\mathbf{G}_{i+1}}[f(G_{1:i} \cup G_{i+1}) - f(G_{1:i})] + \mathcal{H}(\mathbf{G}_{i+1}|\mathbf{G}_{1:i} = G_{1:i})\Big]$$

$$\overset{\text{summarize}}{=} \mathbb{E}_{\mathbf{G}_{1:i}}\Big[f(G_{1:i})\Big] + k\Big[\mathbb{E}_{\mathbf{G}_{1:i+1}}[f(G_{1:i+1})] - \mathbb{E}_{\mathbf{G}_{1:i}}[f(G_{1:i})] + \mathcal{H}(\mathbf{G}_{i+1}|\mathbf{G}_{1:i})\Big]$$

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

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

Proof of Lemma 2:
By rearranging the terms from Lemma 1

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

we get

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

By induction over i, we have

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

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

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

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

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

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

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

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

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

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:

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

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

(1) $\mathbb{E}_{\mathbf{O}_{1:k}}[f(O_{1:k})] = \mathbb{E}_{O\sim P_{\pi_{opt}}}[f(O)]$

(2) $\mathcal{H}(\mathbf{O}_{1:k}) = \mathcal{H}(P_{opt}) + \log k!$

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

(3) $\mathbb{E}_{\mathbf{G}_{1:k}}[f(G_{1:k})] = \mathbb{E}_{G\sim P_{\pi_{greedy}}}[f(G)]$

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

Proof Lemma 3:

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

(2)

$$\begin{aligned}
&\mathcal{H}(O_{1:k})\\
\overset{\text{definition of } \mathcal{H}}{=}\ & -\sum_{S_{1:k}\in E^k} \pi_{opt}(S_{1:k}) \log \pi_{opt}(S_{1:k})\\
\overset{\text{definition of } \pi_{opt_2}}{=}\ & -\sum_{S_{1:k}\in E^k} \frac{1}{k!} P_{opt}(S) \log \frac{1}{k!} P_{opt}(S)\\
\overset{|\text{perm}(S)|=k!}{=}\ & -\sum_{S\in\mathcal{E}} P_{opt}(S) \log \frac{1}{k!} P_{opt}(S)\\
\overset{\text{summarize}}{=}\ & \mathcal{H}(P_{opt}) + \log k!
\end{aligned}$$

(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. $\mathcal{H}(\mathbf{S}_{1:k}|\mathbf{S}) \le \log k!$

Theorem 1:
Let f be a monotone and submodular set function with $f(\emptyset) = 0$ and $\Delta_f(e|S) > (1/k)\log k!$ for all $S \subset E$, $e \in E \setminus S$. It holds

$$(1 - 1/e)F(P_{opt}) \le F(P_{\pi_{greedy}}).$$

Proof Theorem 1:
Define a new set function $m(S) := f(S) + l(S)$ with $l(S) = -\frac{|S|}{k}\log k!$. Due to the properties of f and l being a modular function, we still have monotony and submodularity for m as well as $m(\emptyset) = 0$ 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 f. Using Lemma 2 and Lemma 3, we obtain the result:

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

Remark 1: In order for a quantitative comparison of two values to be meaningful, requirements on the level of measurement [NL86] have to be fulfilled. In our case with a guarantee on the ratio, the compared quantity has to be measured on a ratio scale, meaning that there is a "natural" zero point, as well as a "natural" interpretation of the difference of two values. Since the greedy as well as the optimal algorithm are invariant with respect to shifting f by a constant term there is a degree of freedom that allows one to choose a meaningful scaling for f.

3 Application: Sampling from a k Determinantal Point Process

A k-DPP on a finite set $E = \{e_1, ..., e_n\}$ (e.g. a grid) is given by

$$P_{kDPP}(S = \{S_1, ..., S_k\}) = \frac{\det K_{SS}}{\sum_{S \in \mathcal{E}} \det K_{SS}},$$

where K is a positive-semidefinite kernel matrix. To see, that this is an instance of Softmaximization as described in Section 2, choose $f(S) = \log\det K_{SS}$. The log-determinant of a Gram matrix corresponds to the differential entropy of the Gaussian distribution (up to constants) and is known to be submodular [KSG08].
The approximate sampling algorithm $\pi_{greedyDPP}$ suggested in [HG16] consists in sampling the next point S_i proportionally to the posterior variance:

$$\pi_{greedyDPP}(S_i|S_{1:i-1}) \propto \mathbb{V}_i(S_i) = \begin{cases} K_{S_iS_i} & \text{if } i = 1 \\ K_{S_iS_i} - K_{S_iS_{1:i-1}} K_{S_{1:i-1}S_{1:i-1}} K_{S_{1:i-1}S_i} & \text{otherwise} \end{cases}$$

Corollary 1:
Running the algorithm $\pi_{greedyDPP}$ in [HG16] for k iterations on a finite grid is a $(1 - 1/e)$ approximation for sampling from a k-DPP.

Proof Corollary 1:
First, we show that $\pi_{greedyDPP}$ is an instantiation of the greedy sampling algorithm defined in Section 2. It is well known that the differential entropy $\frac{1}{2}\log 2\pi e \det K_{SS}$ of a Gaussian random variable can be expressed as the sum over the predictive variances $\mathbb{V}_i(S_i)$:

$$\frac{1}{2}\log 2\pi e \det K_{SS} = \frac{1}{2}\sum_{i=1}^{k} \log(2\pi e \mathbb{V}_i(S_i)))$$

Removing constants, we get:

$$f(S) = \log\det K_{SS} = \sum_{i=1}^{k} \log \mathbb{V}_i(S_i)$$

For the marginal gain, this means:

$$\Delta_f(S_i|S_{1:i}) = \log\det K_{S_{1:i}S_{1:i}} - \log\det K_{S_{1:i-1}S_{1:i-1}} = \log \mathbb{V}_i(S_i).$$

This reveals that the policy $\pi_{greedyDPP}$ is the softargmax of the marginal gain $\Delta_f(S_i|S_{1:i})$:

$$\pi_{greedyDPP}(S_i|S_{1:i-1}) = \frac{\exp \Delta_f(S_i|S_{1:i})}{\sum_{S_j \in E} \exp \Delta_f(S_j|S_{1:i})}$$

Or equivalently,

$$\pi_{greedyDPP}(S_i|S_{1:i-1}) = \arg\max_{\pi} \mathbb{E}_\pi(\Delta_f(S_i|S_{1:i})) + \mathcal{H}(\pi)$$

by using the variational representation. In order to apply Theorem 1, it has to hold $\Delta_f(S_i|S_{1:i}) > (1/k)\log k!$. This can always be achieved by scaling K by a factor of $\alpha = \frac{k!^{1/k}}{\lambda_{min}}$, where λ_{min} is the smallest eigenvalue of K_{EE}. Note that scaling by α neither changes $\pi_{greedyDPP}$ nor P_{kDPP}. Theorem 1 gives the desired result.

Remark 2.
The scaling of f and thereby of F may seem arbitrary, but it has the following interpretation: The "natural" zero point is the empty set. Since all marginal gains are positive, no other set can underscore the value of the empty set. The unit difference between two values corresponds to ...

References

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[HG16] Philipp Hennig and Roman Garnett. Exact sampling from determinantal point processes. *arXiv preprint arXiv:1609.06840*, 2016.

[KSG08] 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.

[NL86] Louis Narens and R Duncan Luce. Measurement: The theory of numerical assignments. *Psychological Bulletin*, 99(2):166, 1986.

[NWF78] 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.