Learning with Generalized Negative Dependence:

Probabilistic Models of Diversity for Machine Learning

by

Zelda E. Lawson Mariet

S.M., Massachusetts Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2019

Signature redacted

Certified by .....
Suvrit Sra
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Learning with Generalized Negative Dependence: Probabilistic Models of Diversity for Machine Learning

by

Zelda E. Lawson Mariet

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

This thesis establishes negative dependence as a powerful and computationally efficient framework to analyze machine learning problems that require a theoretical model of diversification. Examples of such problems include experimental design and model compression: subset-selection problems that require carefully balancing the *quality* of each selected element with the *diversity* of the subset as a whole. Negative dependence, which models the behavior of "repelling" random variables, provides a rich mathematical framework for the analysis of such problems.

Leveraging negative dependence theory for machine learning requires (a) scalable sampling and learning algorithms for negatively dependent measures, and (b) negatively dependent measures able to model the specific diversity requirements that arise in machine learning. These problems are the focus of this thesis.

The first part of this thesis develops scalable sampling and learning algorithms for determinantal point processes (DPPs), popular negatively dependent measures with many applications to machine learning. For scalable sampling, we introduce a theoretically-motivated generative deep neural network for DPP-like samples over arbitrary ground sets. To address the learning problem, we show that algorithms for maximum likelihood estimation (MLE) for DPPs are drastically sped up with Kronecker kernels, and that MLE can be further enriched by negative samples.

The second part of this thesis leverages negative dependence for core problems in machine learning. We begin by deriving a generalized form of volume sampling (GVS) based on elementary symmetric polynomials, and prove that the induced measures exhibit strong negative dependence properties. We then show that classical forms of optimal experimental design can be cast as optimization problems based on GVS, for which we derive randomized and greedy algorithms to obtain the associated designs. Finally, we introduce *exponentiated* strongly Rayleigh measures, which allow for simple tuning of the strength of repulsive forces between similar items while still enjoying fast sampling algorithms. The great flexibility of exponentiated strongly Rayleigh measures makes them an ideal tool for machine learning problems that benefit from negative dependence theory.

Thesis Supervisor: Suvrit Sra
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

I am profoundly indebted to my advisor, Suvrit Sra, who introduced me to the beauty of DPPs and negative dependence; his advice and guidance (both mathematical and not) were instrumental to every step of my PhD. I also thank my committee, Stefanie Jegelka and Tommi Jaakkola, for their advice and help throughout my grad school years.

My deepest thanks to Ahmed, for his steadfast support through the ups, downs, and sideways of research and life; to Andrea and Jane, the best friends and drinking buddies I could ask for; to Hsin-Yu, amazing roommate and actual angel. Within Sidney-Pacific: thank you to Aashka, Daniel, Dave, Dina, Elan, Eric, Fabián, Greg, Jasmine, Jenny, Jit, Joe, Mirna, Olivia, Nicholas, Rachael and Sydney for being fantastic friends and colleagues throughout officer and trustee life. Thank you also to Marzyeh, Sepehr, Reyu and Clément; to Sheila and Frank for being a family away from home; and to Aude, Flora, Hélène, Estelle, who have stuck with me throughout five years of jetlag and time differences.

Thank you to my labmates of the LogSS and LIS groups, especially Zi and Beomjoon, for many entertaining conversations throughout the years about life, the universe, and matrices, and to my labmate-slash-gym buddy Alex. I am also deeply thankful for the support of Leslie Pack Kaelbling and Tomas Lozano-Perez, who so kindly adopted me and taught me about POMDPs during my first advisor-less semester at MIT.

During my PhD, I was lucky to participate in several great internships and collaborations; thank you to Vitaly Kuznetsov, Alex Kulesza, Jennifer Gillenwater, Sergei Vassilvitskii, Jasper Snoek, Jamie Smith, Yaniv Ovadia, D Sculley, and the rest of the Google MAD science and Brain CAM teams, as well as Mike Gartrell, my collaborator in all things low-rank DPP.

Thank you to my old math teacher, Jean Rouquette, for encouraging my love of math.

And, most of all, thank you to my parents and to my sister Cassandra — without your love and support, this thesis never could have existed. Your love means everything to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Machine learning is part of most tools of modern life: automated personal assistants parse spoken requests for weather updates and reminders, complex recommender systems learn from a wealth of stored experience to suggest books, movies, and products tailored to our tastes, and probabilistic models predict anything from climate patterns to stock prices. Where previously machine learning existed only in the spaces of binary choices (Is this email spam? Does this scan show a cancerous mass?), we now rely on machine learning for complex predictions over large spaces with complicated structures. Often, such predictions involve many *simultaneous* decisions. For example, recommender systems rarely suggest a single option to a user. Rather, the user is presented with 5–10 options they might enjoy and makes the final decision.

Choosing a smaller set of items out of many raises the question of how interactions between chosen items should modify the quality of the selected set as a whole. For a book recommendation task, providing the user with recommendations that belong to different genres increases the probability of user engagement with at least one book. When suggesting websites relevant to the query "bow," showing options for violins, archery, and ribbons is more likely to produce a relevant result than three websites focused solely on archery.

These examples illustrate a common phenomenon within subset selection: *negative*

dependence. At a high level, negative dependence is the behavior exhibited when similarity between items (*e.g.*, similarity between interpretations of the term "bow" for the website query case) decreases their likelihood of being chosen simultaneously.

Such occurrences of negative dependence are particularly common in machine learning model design. To train a classification model, the training data must span all different classes that the model will be analyzing once deployed; the marginal gain of adding a data point from a previously seen class is much smaller compared to adding a point from a new class. To estimate a multidimensional hidden variable, experiments must be different enough to reduce uncertainty across each dimension of the hidden variable; running a set of similar experiments may reduce the time and cost of experiments, but will provide less information.

Given a large set of items, evaluating each item's quality sequentially comes at a cost that grows linearly in the number N of items. However, when moving to evaluate *sets* of items, such evaluations will in general require a combinatorial exploration over all 2^N possible sets. As such an exponential-cost exploration is not computationally feasible, simplifying assumptions and heuristics are applied to reduce the dimensionality of the search space. The strongest simplifying assumption models each item selection event as independent from the others — this, of course, cannot capture negative interactions between choices. Less drastic modeling assumptions allow for local, sparse interactions between items. Graphical models are a popular model choice for such settings. Typically, however, graphical models are restricted to modeling *positive* interactions in order to allow for fast sampling and inference.

Recently, specific models for negative dependence have gained prominence in machine learning. In particular, determinantal point processes (DPPs), which tractably model negative interactions in subset-selection, have seen many applications in recommender systems and automatic summarization — see *e.g.*, (Lin et al., 2012; Chao et al., 2015; Zhou et al., 2010; Wilhelm et al., 2018). Determinantal point processes have also been the focus of a significant theoretical analysis (Lyons, 2003; Hough et al., 2006; Borodin, 2009; Kulesza and Taskar, 2012; Decreusefond et al., 2015; Lavancier et al., 2015), which in turn has yielded an elegant theory of diverse subset selection.

However, the spectrum of negatively dependent measures is much broader than only determinantal point processes. In this thesis, we show that determinantal point processes and the broader class of negatively dependence measures have a key role to play in machine learning model design and optimization.

1.2 Thesis outline

The aim of this thesis is to motivate and develop the use of negatively dependent measures for problems in machine learning that require a model of diversity. In order to accomplish this goal, we must address two separate problems. First, we must scale up algorithms for negatively dependent measures, in order to broaden their applicability (Chapters 3 and 4). Then, we must generalize and develop the theory of negative dependence within the framework of machine learning; this will allow us to obtain fine-grained models for the negative dependence phenomena specific to machine learning (Chapters 5 and 6).

Specifically, the outline of this thesis is the following. Chapter 2 introduces the rich theory of negative dependence derived by Borcea et al. (2009), focusing on strongly Rayleigh (SR) measures and relevant results we will require for the rest of this thesis.

We then focus on the most popular negatively dependent measure: determinantal point processes. Chapters 3 and 4 present results for efficient sampling and learning algorithms for DPPs, allowing for a broader applications of DPPs to machine learning. In Chapter 3, we introduce deep generative networks to sample from DPPs with variable kernels. Chapter 4 focuses on the learning problem; we first introduce Kronecker-structured kernels for DPPs, which allow for efficient learning methods, then consider a novel learning paradigm that augments the standard Maximum Likelihood Estimation problem for DPPs with corrective negative samples.

In Chapter 5, we analyze Generalized Volume Sampling measures, *i.e.*, measures that extend volume sampling by leveraging the family of elementary symmetric polynomials. We identify key connections between generalized volume sampling and the Minimum Volume Covering Ellipsoid problem, a well-known and deeply studied op-

timization problem. We also show that standard models for optimal experimental design can be re-framed in the larger context of generalized volume sampling, and develop randomized and greedy algorithms to compute the associated designs.

Finally, Chapter 6 extends the class of strongly Rayleigh measures to the superclass of *exponentiated* strongly Rayleigh measures. We show that these measures inherit some of the negative dependence properties of SR measures, and allow for intuitive and efficient tuning of the strength of the similarity repulsion process. Experimental results on outlier detection and kernel reconstruction tasks confirm that the greater flexibility of exponentiated strongly Rayleigh measures makes them a valuable tool to approach fundamental problems in machine learning.

1.2.1 Publications contained in this thesis

This thesis covers results first derived in the following publications:

- Z. Mariet, Y. Ovadia, and J. Snoek. DPPNET: Approximating Determinantal Point Processes with deep networks. *Submitted to NeurIPS*, 2019b *Chapter 3*
- Z. Mariet and S. Sra. Kronecker Determinantal Point Processes. In *Advances in Neural Information Processing Systems*, 2016b *Chapter 4*
- Z. Mariet, M. Gartrell, and S. Sra. Learning Determinantal Point Processes by sampling inferred negatives. In *Proc. Int. Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019a *Chapter 4*
- Z. Mariet and S. Sra. Elementary symmetric polynomials for optimal experimental design. In *Advances in Neural Information Processing Systems*, 2017 *Chapter 5*
- Z. Mariet, S. Sra, and S. Jegelka. Exponentiated Strongly Rayleigh distributions. In *Advances in Neural Information Processing Systems*, 2018 *Chapter 6*

1.2.2 Additional related publications

The following papers contain results on negative dependence and its applications to machine learning, but are not covered in detail in this thesis.

- Z. Mariet and S. Sra. Fixed-point algorithms for learning Determinantal Point Processes. In *Proc. Int. Conference on Machine Learning (ICML)*, 2015
- Z. Mariet and S. Sra. Diversity networks: Neural network compression using Determinantal Point Processes. *Int. Conf. on Learning Representations (ICLR)*, 2016a
- J. A. Gillenwater, A. Kulesza, Z. Mariet, and S. Vassilvitskii. Maximizing Induced Cardinality Under a Determinantal Point Process. In *Advances in Neural Information Processing Systems*, 2018
- J. A. Gillenwater, A. Kulesza, Z. Mariet, and S. Vassilvitskii. Sublinear Sampling for Determinantal Point Processes. *Proc. Int. Conference on Machine Learning (ICML)*, 2019

1.3 Notation

We consider measures μ over a finite ground set $\mathcal{Y}$ of N items; without loss of generality we set $\mathcal{Y} = \{1, \dots, N\}$ and also use the notation $[N] \triangleq \{1, \dots, N\}$. We denote by $2^{\mathcal{Y}} = 2^{[N]}$ the powerset of $\mathcal{Y}$. Capital letters such as S and T denote subsets of $[N]$; we distinguish matrices from sets by writing them in bold, *e.g.*, $\boldsymbol{L}$ and $\boldsymbol{M}$. We write $\mathbf{0}$ (resp. $\mathbf{1}$) for the vector of all zeros (resp. all ones), where the dimension is clear from context.

We often consider submatrices indexed by subsets of $\mathcal{Y}[N]$. For $S, T \subseteq [N]$ and $\boldsymbol{M} \in \mathbb{R}^{N \times N}$, we write $\boldsymbol{M}_{S,T} := [\boldsymbol{M}_{ij}]_{i \in S,\ j \in T}$ for the $|S| \times |T|$ submatrix of $\boldsymbol{M}$. To lighten the notation, we write $\boldsymbol{M}_{:,T} = \boldsymbol{M}_{[N],T}$ and $\boldsymbol{M}_{S,:} = \boldsymbol{M}_{S,[N]}$. For principal submatrices, we additionally abbreviate $\boldsymbol{M}_{S,S}$ as $\boldsymbol{M}_S$.

Finally, we write $\mathbb{S}_N^+$ (resp. $\mathbb{S}_N^{++}$) for the set of positive semi-definite (resp. positive definite) matrices of size $N \times N$, and denote by $\succeq$ the usual Loewner order on $\mathbb{S}_N^+$: for $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{S}_N^+$, $\boldsymbol{A} \succeq \boldsymbol{B} \iff (\boldsymbol{A} - \boldsymbol{B}) \in \mathbb{S}_N^+$.

Chapter 2

A theory of negative dependence

Negative dependence characterizes the behavior of probability distributions over subsets of items. At a high level, events are negatively dependent if they are unlikely to occur simultaneously. As a simple example, consider N urns within which we have dropped a total of k balls in an *i.i.d.* fashion. Intuitively, the existence of an empty urn decreases the probability of other urns being empty; and in fact, one can verify that the events "urn i is empty" and "urn j is empty" are negatively dependent.

The necessity of a theoretically grounded model of negative dependence was recognized by Pemantle (2000), wherein Pemantle motivated the need for such theory and provided arguments suggesting its existence. However, the theory of negative dependence proved to be much more difficult to develop than the theory of positive dependence. Many different characterizations of negative dependence were suggested, yet identifying the interactions and overlaps between these characterizations turned out to be surprisingly complex.

A partial solution was offered by Borcea and Brändén (2009), who introduced the class of *strongly Rayleigh* measures: a class of distributions over subsets that inherits all previous characterizations of negative dependence. Since this contribution, strongly Rayleigh measures (and the related theory of real stable polynomials) have been fundamental to many areas of mathematics. Most famously, they were crucial to proving the Kadison-Singer conjecture (Marcus et al., 2015); they also served to prove a variant of the Kadison-Singer problem introduced in (Anari and Gharan, 2014), as

well as the Monotone Column Permanent conjecture (Brändén et al., 2011).

More recently, strongly Rayleigh measures have emerged as an elegant class well-suited to modeling many problems in machine learning model design and optimization. As this thesis aims to develop and apply the theory of negative dependence for machine learning, we begin with an overview of the theory of strongly Rayleigh distributions, focusing particularly on the properties that make them relevant for machine learning applications.

2.1 Characterizing negative dependence

Mathematically, negative dependence is defined by considering distributions over subsets of a ground set; in this thesis, we consider discrete ground sets $\mathcal{Y}$, which we identify with $[N]$. By definition, the probability measures μ we are interested in verify $\mu : 2^{[N]} \to [0, 1]$ and $\sum_{S \subseteq [N]} \mu(S) = 1$.

2.1.1 Preliminaries: distributions over subsets

Denote by X_i the binary random variable over $2^{[N]}$ defined for all subsets $S \subseteq [N]$ as $X_i(S) = 1$, if $i \in S$, and 0, otherwise. Equivalently, X_i is the i-th coordinate function. The indicator function χ_S of any subset S, defined as $\chi_S(T) \neq 0$ if and only if $S = T$, is given by

$$\chi_S(T) = \prod_{i \in S} X_i(T) \prod_{j \notin S} \big(1 - X_j(T)\big).$$

As χ_S is a multi-affine polynomial[1] in the X_i, it follows that any function f over $2^{[N]}$ can be written as a multi-affine polynomial in the X_i: first by writing $f(S) = \sum_{S \subseteq [N]} f(S)\chi_S$ and then by reordering the variables to obtain a formulation $f(S) = \tilde{f}\big(X_1(S), \ldots, X_N(S)\big) = \sum_{S \subseteq [N]} a_S \prod_{i \in S} X_i$.

As a consequence, any distribution μ over subsets of $[N]$ can be equivalently represented by a multi-affine polynomial with non-negative coefficients in N variables; this polynomial is called the *generating polynomial* of μ.

[1] Recall that a multivariate polynomial is multi-affine if it has degree at most one in each variable.

Definition 2.1.1 (Generating polynomial)**.** Let μ be a probability distribution over $2^{[N]}$. The generating polynomial of μ, denoted by g_μ, is the multi-affine polynomial over $\mathbb{C}^N$ defined by

$$g_\mu(z_1, \ldots, z_N) = \sum_{Y \subseteq [N]} \mu(Y) z^Y,$$

where we write $z^Y \triangleq \prod_{y \in Y} z_y$.

Following (Borcea and Brändén, 2009), when considering generating polynomials, we will use X_i for random variables and use lower-case x, y, z for variables in $\mathbb{C}^n$.

Remark 2.1.1. By construction, $g_\mu(\mathbf{1}) = 1$.

Proposition 2.1.1. *There is a one-to-one mapping between the set of distributions over $2^{[N]}$ and the set $P_N[z_1, \ldots, z_N]$ of multi-affine polynomials g in N variables with non-negative coefficients such that $g(\mathbf{1}) = 1$.*

Proof. Def. 2.1.1 provides an injective mapping from distributions over $2^{[N]}$ to the polynomials in $P_N[z_1, \ldots, z_N]$. Conversely, any multi-affine polynomial g with non-negative coefficients such that $g(\mathbf{1}) = 1$ defines a distribution over $2^{[N]}$. To see this, write $g(z) = \sum_{S \subseteq [N]} a_S z^S$ and define distribution μ_g by setting $\mu(S) = a_S \geq 0$ for all $S \subseteq [N]$. The condition $g(\mathbf{1}) = 1$ guarantees that μ is normalized. □

Most standard operations on distributions μ over $2^{[N]}$ translate directly into operations over the corresponding generating polynomial g_μ. In particular, the probability $\mu(S)$ and the marginal probability $\Pr_\mu(T \subseteq S)$ of observing T when sampling $S \sim \mu$ are given by the following operations:

$$\mu(S) = \partial^S g_\mu(\mathbf{0}) \quad \text{and} \quad \Pr_\mu(T \subseteq S) = \partial^T g_\mu(\mathbf{1}),$$

where we write ∂^S for the differential operator $\prod_{i \in S} \partial / \partial z_i$.

We will often need to consider distributions over subsets that only assign non-zero probabilities to subsets of a given size k; for this purpose, we recall here the definition of homogeneous polynomials and measures.

Definition 2.1.2 (Homogeneous polynomials and measures)**.** A polynomial $g \in \mathbb{R}[z_1, \ldots, z_N]$ is homogeneous of degree k if for all $t \in \mathbb{C}$ and $z \in \mathbb{C}^N$, $g(tz) = t^k g(z)$.

Similarly, a measure μ over $2^{[N]}$ is homogeneous of degree k if its generating polynomial is homogeneous of degree k, *i.e.*, if $\mu(S) \neq 0 \implies |S| = k$.

2.1.2 Various characterizations of negative dependence

Deriving the theory for negative dependence is a complex endeavor; as pointed out by Pemantle (2000), given to comparative simplicity of obtaining a theory of *positive* dependence, characterizing negative dependence is surprisingly difficult.

A weak characterization of negative dependence for a measure μ over $2^{[N]}$ is given by the pairwise negative correlation (p-NC) condition: μ is pairwise negatively correlated if for all $i, j \in [N]$,

$$\Pr_\mu(X_i)\Pr_\mu(X_j) \geq \Pr_\mu(X_i X_j), \tag{p-NC}$$

where we recall that X_i is the binary random variable indicating $i \in S$. In terms of generating polynomials, (p-NC) is equivalent to

$$\partial_i g_\mu(\mathbf{1})\partial_j g_\mu(\mathbf{1}) \geq \partial_i \partial_j g_\mu(\mathbf{1}).$$

The p-NC inequality simply states that elements i and j are more likely to be sampled independently under μ than simultaneously; this amounts to the weakest form of negative dependence.

A second characterization of negative dependence is given by the *negative lattice condition*, which requires for all $S, T \subseteq [N]$ that

$$\mu(S)\mu(T) \geq \mu(S \cup T)\mu(S \cap T). \tag{NLC}$$

In terms of μ's generating polynomial, (NLC) is equivalently stated as

$$\partial^S g_\mu(\mathbf{0})\partial^T g_\mu(\mathbf{0}) \geq \partial^{S \cup T} g_\mu(\mathbf{0})\partial^{S \cap T} g_\mu(\mathbf{0}).$$

The NLC condition is equivalent to the log-submodularity of μ.

Definition 2.1.3. A function f over $2^{[N]}$ is *submodular* if it verifies for every $S, T \subseteq [N]$, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. A function f over $2^{[N]}$ is said to be *log-submodular* if $\log(f)$ is well defined and submodular.

Proposition 2.1.2. *A distribution μ over $2^{[N]}$ verifies the NLC condition if and only if μ is log-submodular.*

Finally, a global characterization of negative dependence is given by the negative association (NA) property, stated as

$$\int F \, \mathrm{d}\mu \int G \, \mathrm{d}\mu \geq \int FG \, \mathrm{d}\mu, \tag{NA}$$

for all pairs of increasing functions F and G over $2^{[N]}$ where, crucially, F and G must depend on a *disjoint*[2] set of coordinates. Requiring that F and G depend on disjoint sets of coordinates is necessary to avoid trivial correlations such as the self-correlation between F and F that follows from the Cauchy-Schwarz inequality.

As alluded to earlier, characterizing negative dependence is more difficult than characterizing positive dependence. As a first complication, there is no straightforward chain of implications between the previous inequalities. The positive dependence equivalents to p-NC, NLC and NA (which are obtained by reversing the inequalities in all negative dependence characterizations, and which we refer to as p-PC, PLC and PA) are linked by the implications

$$\text{PA} \iff \text{PLC} \implies \text{p-PC},$$

where the first equivalence is given by the FKG theorem (Fortuin et al., 1971). However, there is no such chain of implications for negative dependence. Indeed, NLC does not imply negative association, nor is p-NC implied by the NLC condition.

[2]See (Borcea and Brändén, 2009) for a formal definition of set functions with disjoint sets of coordinates.

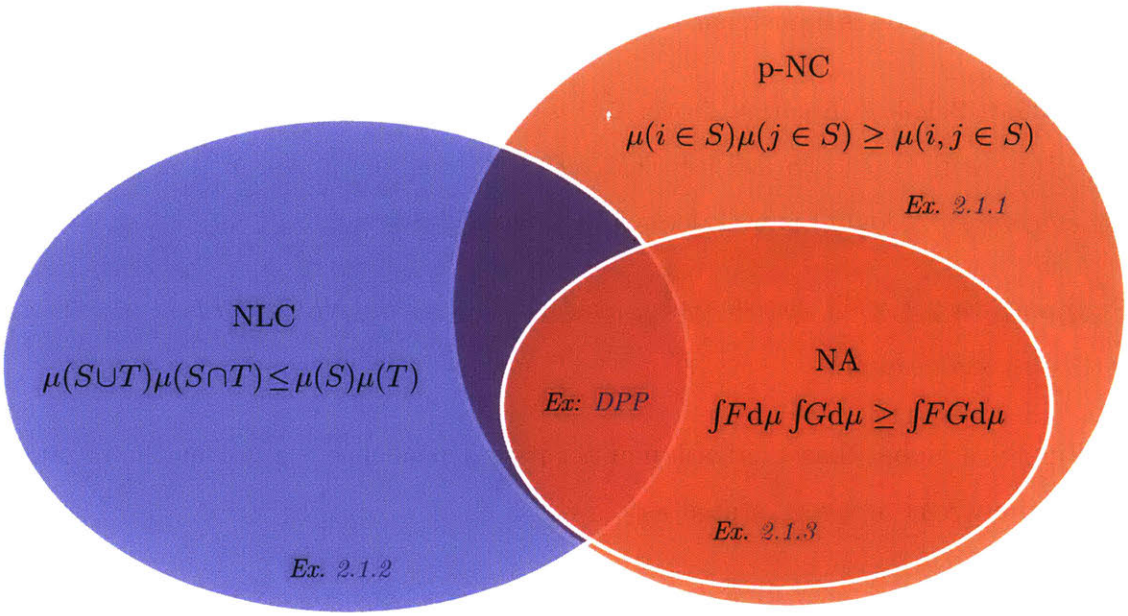


Figure 2-1: Venn diagram of implications between negative dependence properties

Example 2.1.1 (p-NC $\not\Longrightarrow$ NLC, NA)**.** The p-NC property is the weakest characterization of negative dependence; the following measure μ over $\{1, 2, 3\}$ is easily verified to be a p-NC measure that satisfies neither the NA nor NLC conditions:

$$\mu(1) = \mu(2) = \mu(3) = 0.3 \qquad \mu(123) = 0.1 \quad \text{(all other probabilities being zero)}.$$

Example 2.1.2 (NLC $\not\Longrightarrow$ p-NC, NA (Borcea and Brändén, 2009))**.** The measure μ over subsets of $\{1, \ldots, 4\}$ such that $\mu(\{1, 2\} = \mu(\{3, 4\}) = \frac{1}{2}$ trivially satisfies the NLC condition but $\mu(X_1 X_2) = \frac{1}{2}$ and $\mu(X_1)\mu(X_2) = \frac{1}{4}$. Hence, μ is not p-NC. Since NA $\Longrightarrow$ p-NC, it immediately follows that μ also does not satisfy NA.

Example 2.1.3 (NA $\not\Longrightarrow$ NLC (Borcea and Brändén, 2009))**.** Let e_k be the k-th elementary symmetric polynomial in n variables ($e_k(z) = \sum_{|S|=k} z^S$); the measure μ, with generating polynomial

$$g_\mu(z_1, z_2, z_3) \propto \tfrac{3}{2} + z_1 + z_2 + z_3 + z_1 z_2 + z_2 z_3 + z_1 z_3,$$

is NA but not NLC.

Finding a measure that simultaneously verifies the p-NC and NLC conditions while not being NA is more difficult; however, for $N \geq 4$ certain exponentiated DPPs (Chapter 6), which by construction satisfy the NLC inequality, can be shown to satisfy p-NC but break NA numerically.

A second setback to deriving a theory of negative dependence arises from the fact that the NA and NLC properties are not closed under desirable operations over probability distributions. Introduced in (Pemantle, 2000), these desirable closure properties for a characterization $\mathcal{P}$ of negative dependence are the following: for any measure μ over $2^{[N]}$ verifying $\mathcal{P}$,

- *Closure under projection*: given $A \subseteq [N]$, the projection of μ onto $\{0,1\}^A$ (*i.e.*, the measure obtained by integrating out the variables in $[N] \setminus A$) should satisfy $\mathcal{P}$. Equivalently, the measure with generating polynomial $g_\mu\big|_{z_i=1, i\in[N]\setminus A}$ should verify $\mathcal{P}$.

- *Closure under conditioning*: given $A \subseteq [N]$ and $\eta \in \{0,1\}^A$, the conditioned measure $\mu(\cdot \mid X_i = \eta(i),\ i \in A)$ should verify $\mathcal{P}$; equivalently, the measure with generating polynomial h in $N - |A|$ variables defined by $h(z_{i\notin A}) := g_\mu(z')$, where $z'_i = z_i$ if $i \notin A$ and $z_i = \eta(i)$ otherwise, should verify $\mathcal{P}$.

- *Closure under external fields*[3]: for a vector $a \in \mathbb{R}_+^N$, the measure with generating polynomial
$$(z_1, \ldots, z_N) \to \frac{g_\mu(a_1 z_1, \ldots, a_N z_N)}{g_\mu(a_1, \ldots, a_N)}$$
should verify $\mathcal{P}$.

Neither NA nor NLC satisfy these closure properties; for this reason, the following definitions are required.

Definition 2.1.4 ([Strong] hereditary negative lattice condition)**.** A measure μ over $2^{[N]}$ satisfies the *hereditary negative lattice condition* (h-NLC) if every projection of μ satisfies NLC. Furthermore, a measure μ over $2^{[N]}$ satisfies the *strong hereditary*

[3]This name is borrowed from the theory of Ising models.

negative lattice condition (h-NLC+) if any measure obtained from μ by imposing an external field satisfies h-NLC.

Definition 2.1.5 ([Strongly] conditionally negative associated measures)**.** A measure μ over $2^{[N]}$ is *conditionally negatively associated* (CNA) if any measure obtained by conditioning μ on some of the values of the variables is NA. Similarly, a measure μ is *strongly conditionally negatively associated* (CNA+) if any measure obtained by imposing external fields and projections on μ is CNA.

Proposition 2.1.3 (Pemantle (2000))**.** *We have the following implications: CNA* $\implies$ *h-NLC, and CNA+* $\implies$ *h-NLC+. Additionally, all of h-NLC, h-NLC+, NA, CNA, and CNA+ imply p-NC.*

2.2 Strongly Rayleigh measures

The crucial development in deriving a theory of negative dependence was derived by Borcea and Brändén (2009) through the introduction of the class of *strongly Rayleigh* (SR) measures. SR measures are characterized via the roots of their generating polynomials; for this reason, we begin by reproducing a few related results in multivariate polynomial theory below.

Definition 2.2.1 (Stable polynomial)**.** A polynomial $g \in \mathbb{C}[z_1, \ldots, z_N]$ is *stable* if

$$\mathfrak{Im}(z_i) > 0 \text{ for all } 1 \leq i \leq n \implies g(z_1, \ldots, z_N) \neq 0.$$

If g is stable and has all real coefficients, g is called *real stable.*

Real stable polynomials can be viewed as a generalization of real-rooted univariate polynomials to multivariate polynomials.

Proposition 2.2.1. *Let g be a real stable polynomial in N variables and consider the univariate polynomial $\tilde{g} : z \to g(z, \ldots, z)$; then $\tilde{g}$ is real-rooted.*

Proof. By contradiction, if $\tilde{g}$ has a root $z = a + ib$ with $b \neq 0$, one easily verifies that $a - ib$ is also a root for $\tilde{g}$. Hence, $\tilde{g}$ has a root z with $\mathfrak{Im}(z) > 0$, and so $z = (z, \ldots, z)$ is a root of g in the positive orthant $\{z \in \mathbb{C}^n \mid \forall i, \mathfrak{Im}(z_i) > 0\}$: g is not real stable. □

Multivariate stable polynomials have fundamental connections to many objects in polynomial theory and optimization, including hyperbolic polynomials and Lee-Yang polynomials; we refer the interested reader to works such as (Gårding, 1959; Güler, 1997; Choe et al., 2004; Borcea and Brändén, 2008; Brändén, 2007; Borcea and Brändén, 2009, 2010). They also admit the following characterization:

Proposition 2.2.2. *A polynomial $g \in \mathbb{R}[z_1, \ldots, z_n]$ is real stable if and only if for all $x \in \mathbb{R}^n_+$ and any $e \in \mathbb{R}^n$, the univariate polynomial $t \to g(e + tx)$ is real rooted.*

2.2.1 Definition and properties

Given the concept of stable polynomials, defining the class of strongly Rayleigh measure is straightforward: strongly Rayleigh measures are measures with real stable generating polynomials.

Definition 2.2.2 (Strongly Rayleigh measure)**.** A measure μ over $2^{[N]}$ is strongly Rayleigh if its generating polynomial g_μ is real stable.

Theorem 2.2.3 (Borcea and Brändén (2009))**.** *If a measure μ over $2^{[N]}$ is strongly Rayleigh, μ is CNA+. It follows from Proposition 2.1.3 that μ also verifies the h-NLC+ condition and is pairwise negatively correlated.*

Proving the real-stability of a polynomial via the location of its roots is often difficult; fortunately, an alternate characterization of real stability for multi-affine polynomials (and hence for strongly Rayleigh measures) is given by the next result.

Theorem 2.2.4 (Brändén (2007, Theorem 5.6))**.** *A multi-affine polynomial g over $\mathbb{C}^N$ is real stable if and only if the following inequality holds for any $1 \leq i < j \leq N$:*

$$\forall x \in \mathbb{R}^n, \quad \frac{\partial g}{\partial x_i}(x)\frac{\partial g}{\partial x_i}(x) - g(x)\frac{\partial^2 g(x)}{\partial x_i \partial x_j} \geq 0.$$

Remark 2.2.1. The above characterization of real stability only holds for multi-affine polynomials in $\mathbb{C}[z_1, \ldots, z_N]$.

Finally, note that although strongly Rayleigh measures are CNA+, CNA+ measures are not necessarily SR: for $N \geq 20$, Borcea et al. (2009, Section 7) construct measures over $2^{[N]}$ that are CNA+ but not SR.

2.2.2 Operations preserving real stability

In the previous section, we listed a sequence of operations that should ideally preserve a strong characterization of negative dependence. Fortunately, SR measures are stable under these operations, and more. As these operations will be crucial to proving the SR-ness of a variety of distributions over $2^{[N]}$, we reproduce them here.

Proposition 2.2.5 (Borcea et al. (2009, Prop. 3.1)). *Let g be a polynomial over complex variables $z_1, \ldots, z_N$ with degree d_i in variable z_i. If g is real stable, the following polynomials are real stable or constant equal to zero:*

(i) $\partial_i g(z_1, \ldots, z_N)$ *for* $1 \leq i \leq N$

(ii) $g(z_1, \ldots, z_{i-1}, \alpha z_i, z_{i+1}, \ldots, z_N)$ *for* $1 \leq i \leq N$ *and* $\alpha > 0$

(iii) $g(z_1, \ldots, z_{i-1}, \beta, z_{i+1}, \ldots, z_N)$ *for* $1 \leq i \leq N$ *and* $\beta > 0$

(iv) $z_1^{d_1} \cdots z_N^{d_N} g(\lambda_1 z_1^{-1}, \ldots, \lambda_N z_N^{-1})$ *for* $\pm(\lambda_1, \ldots, \lambda_N) \in \mathbb{R}_+^N$

(v) $g(z_1, \ldots, z_{i-1}, z_j, z_{i+1}, \ldots, z_N)$ *for* $i \neq j$.

Additionally, the two following operations, which restrict the distribution's support to sets of a fixed size, also preserve the strongly Rayleigh property.

Proposition 2.2.6 (Borcea et al. (2009, Thm. 4.2)). *Let μ be an SR measure over $2^{[N]}$. The symmetric homogenization of μ is SR and defined over $2^{[2N]}$ as*

$$\mu_{sh}(S) = \begin{cases} \mu(S \cap [N]) \binom{N}{|S \cap [N]|}^{-1} & \text{if } |S| = N \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Proposition 2.2.7 (Borcea et al. (2009, Corollary 4.18)). *Let μ be a SR measure over $2^{[N]}$. The p, q truncation of μ for $q - p \leq 1$ is also SR, and defined over $2^{[N]}$ as*

$$\mu_{p,q}(S) \propto \begin{cases} \mu(S) & \text{if } p \leq |S| \leq q \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Finally, the limit of a sequence of real stable polynomials is also real stable.

Proposition 2.2.8 (Borcea et al. (2009, Proposition 3.1)). *Let $\{f_j\}_{j=1}^{\infty}$ be a sequence of real stable polynomials in $\mathbb{R}[z_1, \ldots, z_N]$ and $f \not\equiv 0 \in \mathbb{R}[z_1, \ldots, z_N]$ be the limit, uniformly on compact subsets of $\mathbb{C}^N$, of $\{f_j\}_{j=1}^{\infty}$; then f is real stable.*

2.2.3 Examples

We now present a few classical distributions that are strongly Rayleigh and that have found applications in the larger scope of computer science.

The uniform distribution

Let μ be the distribution over all subsets of $2^{[N]}$ such that for all $S \subseteq [N]$, $\mu(S) = \frac{1}{2^N}$. The corresponding generating polynomial g_μ is

$$g_\mu(z_1, \ldots, z_N) = \frac{1}{2^N} \sum_{S \subseteq [N]} z^S = \frac{1}{2^N} \prod_{i=1}^{N} (1 + z_i),$$

which clearly admits no zeros in the space $\{z \in \mathbb{R}^N : \forall i, \mathfrak{Im}(z_i) > 0\}$.

Product measures

The generalization of the uniform distribution, which assigns different weights to each element of the ground set, is also strongly Rayleigh:

Proposition 2.2.9. *The measure over $2^{[N]}$ parametrized by scalars $q_i \in [0, 1], 1 \leq$*

$i \leq N$, *such that* $\mu(S) = \prod_{i \in S} q_i \prod_{j \notin S}(1 - q_j)$ *is SR and has generating polynomial*

$$g_\mu(z_1, \ldots, z_N) = \prod_{i=1}^{N}(q_i z_i + 1 - q_i).$$

Determinantal measures

Many real stable polynomials are based on determinants of symmetric matrices; the following result provides one such example. As many subsequent results will depend on the result of Prop. 2.2.10, we also include its proof below.

Proposition 2.2.10 (Borcea and Brändén (2009, Prop. 3.2)). *Let* $\boldsymbol{A}_0, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_N \in \mathbb{S}_n^+$. *The polynomial*

$$p : z \to \det(\boldsymbol{A}_0 + z_1 \boldsymbol{A}_1 + \ldots + z_N \boldsymbol{A}_N)$$

is either identically zero or real stable; hence as soon as p *is multi-affine, it defines a strongly Rayleigh polynomial.*

Proof. As positive definite matrices are a dense subset of positive semi-definite matrices, we can via Prop. 2.2.8 limit our analysis to the setting where the $\boldsymbol{A}_i$ are positive definite. Let $e \in \mathbb{R}^N$ and $x \in \mathbb{R}_+^N$. The polynomial $t \to p(e + tx) = \det(\boldsymbol{A}_0 + \sum_i e_i \boldsymbol{A}_i + t \sum_i x_i \boldsymbol{A}_i)$ can be rewritten as

$$t \to \det\Big(\sum\nolimits_i x_i \boldsymbol{A}_i\Big) \det\Big(t\boldsymbol{I} + \boldsymbol{M}^{-1/2}\big(\boldsymbol{A}_0 + \sum\nolimits_i e_i \boldsymbol{A}_i\big)\boldsymbol{M}^{-1/2}\Big),$$

where we write $\boldsymbol{M} = \sum_i x_i \boldsymbol{A}_i \in \mathbb{S}_n^{++}$. Then, since $\boldsymbol{M}^{-1/2}(\boldsymbol{A}_0 + \sum_i e_i \boldsymbol{A}_i)\boldsymbol{M}^{-1/2}$ is real symmetric, $t \to p(e + tx)$ is guaranteed to only have real zeros; invoking Prop. 2.2.2 concludes the proof. □

We will see in particular that the strong Rayleigh property for determinantal point processes (Chapters 3-4) and Generalized Volume Sampling measures (Chapter 5) is derived by applying various stability preserving operations to polynomials of the form $z \to \det(\boldsymbol{A}_0 + \sum_i z_i \boldsymbol{A}_i)$.

2.3 Tractable sampling

In order to use strongly Rayleigh measures in machine learning, we must be able to sample from such measures in a tractable fashion. Although specific SR measures such as determinantal point processes allow straightforward exact sampling algorithms (Chapter 3), we present here two general-purpose, *approximate* sampling algorithms for homogeneous (Algorithm 1) and non-homogeneous (Algorithm 2) SR measures.

These algorithms are Markov chain Monte Carlo (MCMC) methods that sample sets sequentially from the uniform distributions, and move from set S to the next set T based on the respective likelihoods of T and S under the target distribution μ.

2.3.1 Fast sampling for homogeneous measures

A straightforward MCMC chain for sampling from k-homogeneous strongly Rayleigh measures is described in Algorithm 1 (technically, Algorithm 1 describes a *lazy* chain: the probability of moving from one state to another is at most $\frac{1}{2}$).

Algorithm 1 MCMC sampling for a k-homogeneous SR measure μ

Input: SR proposal μ
Initialize S to a random subset of size k of $[n]$
while not mixed **do**
  Sample $i \in S$ and $j \in \bar{S}$ uniformly at random
  $T \leftarrow S \cup \{j\} \setminus \{i\}$
  $S \leftarrow T$ with probability $\frac{1}{2} \min\big(1, \mu(T)/\mu(S)\big)$
return S

The crucial metric used to analyzing samplers such as Algorithm 1 is the chain's *mixing time*. The mixing time upper bounds the number of iterations of the algorithm's inner loop required so that the distribution of the generated samples $\hat{\mu}$ is ϵ-close to the true target μ.

Definition 2.3.1 (Mixing time)**.** For a set S and $\epsilon > 0$, write $\hat{\mu}_{t,S}$ for the distribution over samples generated by the MCMC chain initialized at subset S followed by t

iterations of the chain's inner loop. The chain's mixing time $\tau_S(\epsilon)$ is given by

$$\tau_S(\epsilon) = \min_{t>0} \|\hat{\mu}_{t,S} - \mu\|_{TV} \leq \epsilon.$$

Algorithm 1 was proposed by Anari et al. (2016); their key result is the following.

Theorem 2.3.1 (Fast mixing MCMC chains for strongly Rayleigh measures). *Let μ be a k-homogeneous strongly Rayleigh measure over $2^{[N]}$. Algorithm 1 initialized with subset S such that $\mu(S) > 0$ has mixing time*

$$\tau_S(\epsilon) \leq 2kN \log\Big(\frac{1}{\epsilon\mu(S)}\Big).$$

2.3.2 Generalization to non-homogeneous SR measures

Fast sampling for homogeneous SR measures can be generalized to non-homogeneous measures by way of *symmetric homogenization* (Equation 2.1), as shown by Li et al. (2016d). This result relies on the following fact: to sample from a SR measure μ, we can sample from the symmetric homogenization μ_{sh} of μ which is N-homogeneous (using Algorithm 1), then return the subset $S' = S \cap [N]$. One easily verifies that this yields a subset of $[N]$ with the desired probability.

Li et al. (2016d) also introduce a more efficient way of applying the symmetrization trick; we retranscribe the proposed algorithm in Algorithm 2.

Theorem 2.3.2 (Fast sampling for SR measures (Li et al., 2016d)). *If μ is SR, Algorithm 2 mixes in time*

$$\tau_S(\epsilon) \leq 2N^2\Big(\log\binom{N}{|S|} + \log\frac{1}{\epsilon\mu(S)}\Big).$$

Theorems 2.3.1 and 2.3.2 are fundamental to the application of strongly Rayleigh measures to machine learning. For any SR measure μ, these theorems guarantee that — as long as we can efficiently compute the unnormalized value $\mu(S)$ — we can sample from μ efficiently.

Algorithm 2 Lazy MCMC sampling for non-homogeneous SR measures

Input: SR proposal μ
Initialize S to a random subset of size n of $[2n]$
$S \to T \cap [n]$
while not mixed **do**
    Draw $q \sim \text{Unif}[0,1]$
    Sample $i \in S$ and $j \in \bar{S}$ uniformly at random
    if $q \in \left[0, \frac{N-|S|}{2N^2}\right]$ **then**
        $S \leftarrow S \cup \{j\}$ with probability min $\left(1, \frac{\mu(S\cup\{t\})}{\mu(S)} \frac{|S|+1}{N-|S|}\right)$
    else if $q \in \left[\frac{N-|S|}{2N^2}, \frac{N-|S|}{2N}\right]$ **then**
        $S \leftarrow S \cup \{j\} \setminus \{i\}$ with probability min $\left(1, \frac{\mu(S\cup\{t\}\setminus\{s\})}{\mu(S)}\right)$
    else if $q \in \left[\frac{N-|S|}{2N}, \frac{|S|^2+N(N-|S|)}{2N^2}\right]$ **then**
        $S \leftarrow S \setminus \{i\}$ with probability min $\left(1, \frac{\mu(S\setminus\{s\})}{\mu(S)} \frac{|S|}{N-|S|+1}\right)$
    else
        Do nothing
return S

Chapter 3

Determinantal point processes — background and sampling

The class of Determinantal point processes (DPPs) is the class of strongly Rayleigh measures that has achieved the most popularity in machine learning. DPPs provide an elegant model to assign probabilities to the exponential number of subsets of a ground set of N items, while permitting polynomial time sampling and marginalization.

The antecedents of DPPs lie in statistical mechanics (Macchi, 1975); since the seminal work of (Kulesza and Taskar, 2012) they have made inroads into machine learning. Determinantal point processes have been applied to problems such as document and video summarization (Lin et al., 2012; Chao et al., 2015), sensor placement (Krause et al., 2008), recommender systems (Zhou et al., 2010), and object retrieval (Affandi et al., 2014). Recently, they have been used to compress layers in neural networks (Mariet and Sra, 2016a) and to provide sampling procedures for the Nyström method (Li et al., 2016c). The more general study of DPP properties has also garnered interest, see *e.g.*, (Hough et al., 2006; Kulesza and Taskar, 2011, 2012; Affandi et al., 2013; Decreusefond et al., 2015; Lavancier et al., 2015; Borodin, 2009; Lyons, 2003).

We begin this section by providing a summary of the previous key results on DPPs: their negative dependence properties, stability under conditioning, and exact sampling algorithms. However, despite polynomial-time exact sampling algorithms and approximate MCMC sampling schemes (Alg. 2), sampling from a DPP remains

a costly operation, especially when used jointly with other machine learning models such as neural networks. This, unfortunately, impedes the application of DPPs to modern machine learning problems, even when they are, theoretically, the right tool. This motivates our first contribution: we show that under the right conditions, DPPs can be efficiently approximated by generative networks. Such networks allow fast and parallelizable sampling algorithms, making DPPs both theoretically and practically well-suited to modern machine learning applications that require diverse sampling.

3.1 Background

As before, we consider $\mathcal{Y} = \{1, \ldots, N\}$ a ground set of N items.

Definition 3.1.1 (Determinantal point process)**.** A probability measure μ over $2^{\mathcal{Y}}$ is a determinantal point process if, for all $Y \subseteq \mathcal{Y}$ drawn according to μ, we have

$$\forall A \subseteq \mathcal{Y}, \qquad \Pr_{\mu}(A \subseteq Y) = \det(\boldsymbol{K}_A), \tag{3.1}$$

for some positive semi-definite matrix $\boldsymbol{K} \in \mathbb{S}_N^+$ such that $\boldsymbol{0} \preceq \boldsymbol{K} \preceq \boldsymbol{I}$ (the DPP's *marginal kernel*). We adopt the convention $\det(\boldsymbol{K}_\emptyset) = 1$.

If a DPP with marginal kernel $\boldsymbol{K}$ assigns nonzero probability to the empty set, the DPP can alternatively be parametrized by a positive definite matrix $\boldsymbol{L}$ such that

$$\mu(Y) = \det(\boldsymbol{L}_Y)/\det(\boldsymbol{L} + \boldsymbol{I}). \tag{3.2}$$

In this thesis, all DPPs we consider will parametrizable via the kernel $\boldsymbol{L}$. A brief manipulation shows that in this case, $\boldsymbol{K} = \boldsymbol{L}(\boldsymbol{L} + \boldsymbol{I})^{-1} = \boldsymbol{I} - (\boldsymbol{L} + \boldsymbol{I})^{-1}$.

Crucially, determinantal point processes are strongly Rayleigh measures; as such, they inherit all the characterizations of negative dependence introduced in Chapter 2.

Theorem 3.1.1. *Let $\boldsymbol{L} \succeq 0$ be a positive definite matrix in $\mathbb{R}^{N \times N}$. Then, the measure over $2^{[N]}$ defined by $\mu(S) \propto \det(\boldsymbol{L}_S)$ is strongly Rayleigh.*

Proof. The proof relies on the following identity, which can be easily proven by induction on $N - |A|$. Writing $\boldsymbol{Z} = \operatorname{diag}(z_1, \ldots, z_N)$ and $\bar{A} \triangleq [N] \setminus A$, we have

$$\forall A \subseteq [N], \sum_{A \subseteq Y \subseteq [N]} \det \boldsymbol{L}_Y z^Y = \det(\boldsymbol{L}\boldsymbol{Z} + \boldsymbol{I}_{\bar{A}}). \tag{3.3}$$

Evaluating (3.3) at $A = \emptyset$, we see that the generating polynomial of μ is of the form $\det(\boldsymbol{L}\boldsymbol{Z} + \boldsymbol{I})$. Hence, the generating polynomial for μ is given by

$$g_\mu(z_1, \ldots, z_n) \propto \det(\boldsymbol{L}\boldsymbol{Z} + \boldsymbol{I}) = \det(\boldsymbol{L}^{-1} + \boldsymbol{Z}) \det(\boldsymbol{L}^{-1}).$$

Hence, $\det(\boldsymbol{L}^{-1} + \boldsymbol{Z}) = 0$ if and only if one of the z_i is an eigenvalue of $\boldsymbol{L}^{-1}$; as $\boldsymbol{L} \succ 0$, its eigenvalues are all real, and so $g_\mu(z) \neq 0$ if $\mathfrak{Im}(z_i) > 0$ for all i. □

3.1.1 Key properties of determinantal point processes

Intuition behind the quality/diversity trade-off

That a DPP enables a trade-off between quality and diversity within sampled sets is apparent for sets of size 2. Let $\boldsymbol{K} \succeq 0$ be the marginal kernel of a DPP, and let $S \subseteq [N]$ be a set of size 2. By definition,

$$\Pr(S \subseteq Y) = \boldsymbol{K}_{ii}\boldsymbol{K}_{jj} - \boldsymbol{K}_{ij}^2,$$

and so this probability can be increased by increasing either the coefficients $\boldsymbol{K}_{ii}$ and $\boldsymbol{K}_{jj}$ (*point-wise quality* of items i and j) or by reducing the *similarity* between items i and j given by $\boldsymbol{K}_{ij} = \boldsymbol{K}_{ji}$.

More generally, the ability of a DPP trade-off quality with diversity is apparent through the geometric interpretation of determinants as volumes in d dimensional spaces. Let $\boldsymbol{\Phi}$ be a $\mathbb{R}^{n \times d}$ matrix such that $\boldsymbol{L} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ and let $S = \{i_1, \ldots, i_k\} \subseteq [N]$. Then we can rewrite the probability of sampling S under the DPP as

$$\mu(S) \propto \det(\boldsymbol{L}_S) = \det(\boldsymbol{\Phi}_{S,:}\boldsymbol{\Phi}_{S,:}^\top) = \operatorname{Vol}(\phi_{i_1}, \ldots, \phi_{i_k})^2,$$

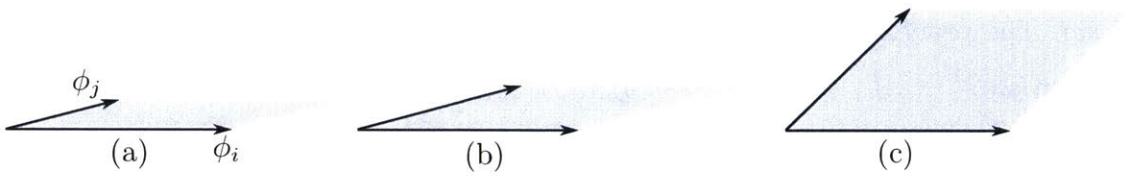


Figure 3-1: The volume spanned by vectors ϕ_i and ϕ_j can be increased either by elongating a vector (b) or by increasing the cosine distance between the vectors (c).

where $\text{Vol}(\phi_{i_1}, \ldots, \phi_{i_k})^2$ is the volume spanned by vectors $\phi_{i_1}, \ldots, \phi_{i_k}$; note that this volume may be zero due to dimensionality issues. Hence, the probability of S under the DPP with kernel $\boldsymbol{L}$ can be increased by increasing either the norm of the ϕ_i (*i.e.*, the point-wise *qualities* of each item) or the angle between the feature vectors (the *diversity* of the set), as illustrated in Fig. 3-1.

Operations over DPPs

Another ingredient to the success of DPPs lies in their robustness to standard operations over distributions such as restrictions, complements and conditioning. These operations for DPPs are covered in detail in (Kulesza and Taskar, 2012, Chapter 2.3); we summarize the results here.

Proposition 3.1.2 (Restriction). *If set Y is distributed according to a* DPP *with marginal kernel $\boldsymbol{K}$ and $A \subseteq \mathcal{Y}$, then $Y \cap A$ is distributed as a* DPP *with marginal kernel $\boldsymbol{K}_A$.*

Proposition 3.1.3 (Complement). *If Y is distributed according to a* DPP *with marginal kernel $\boldsymbol{K}$, then $\mathcal{Y} \setminus Y$ is distributed as a* DPP *with marginal kernel $\boldsymbol{I} - \boldsymbol{K}$.*

Proposition 3.1.4 (Conditioning). *If Y is distributed according to a* DPP *with kernel $\boldsymbol{L}$ and $A, B \subseteq \mathcal{Y}$ are such that $A \cap B = \emptyset$, then*

- $\Pr(Y = B \mid A \cap Y = \emptyset) \propto \det \boldsymbol{L}_Y$*: the event that no element of A appears in Y is distributed according to the* DPP *with kernel $\boldsymbol{L}_{\bar{A}}$.*
- $\Pr(Y = A \cup B \mid A \subseteq Y) \propto \det \boldsymbol{L}^A_Y$*: the event that no element of A appears in Y is distributed according to the* DPP *with kernel $\boldsymbol{L}^A := \left(\left[(\boldsymbol{L} + \boldsymbol{I}_{\bar{A}})^{-1}\right]_{\bar{A}}\right)^{-1} - \boldsymbol{I}$.*

The closure of DPPs to the above operations contributes to their success in machine learning. For example, recommender systems commonly leverage additional information, *e.g.*, when recommending items given other options the user has already expressed interest in (amounting to using a DPP conditioned on pre-selected items).

3.1.2 Sampling from DPPs and k-DPPs

In addition to the approximate sampler introduced earlier for all strongly Rayleigh measures, DPP sampling can be done exactly in $\mathcal{O}(N^3 + Nk^3)$ operations, where k is the size of the final subset; the algorithm is due to Hough et al. (2006). The dominant cost is the initial eigendecomposition of $\boldsymbol{L}$, which requires $\mathcal{O}(N^3)$ operations.

Algorithm 3 Exact sampling from a DPP

1: **Input:** eigendecomposition $(v_i, \lambda_i)_{i=1}^N$ of $\boldsymbol{L}$
2: $J, Y \leftarrow \emptyset$
3: **for** $i = 1$ **to** N **do**
4: $\quad J \leftarrow J \cup \{i\}$ with probability $\frac{\lambda_i}{\lambda_i + 1}$
5: $V \leftarrow \{v_i\}_{i \in J}$, $k \leftarrow |V|$
6: **for** $i = 1$ **to** k **do**
7: $\quad$ Select i from $\mathcal{Y}$ with probability $\Pr(i) = \frac{1}{|V|} \sum_{v \in V} v_i^2$
8: $\quad Y \leftarrow Y \cup \{i\}$
9: $\quad V \leftarrow V_\perp$, orthonormal basis to the subspace of V orthogonal to e_i.
10: **return** Y

Alg. 3 relies on the fact that any DPP can be expressed a mixture of *elementary* DPPs; we refer the reader to (Kulesza and Taskar, 2012). From Alg. 3, it is easy to see that the expected sample size under a DPP is the expected size of V (line 5).

Proposition 3.1.5 (Expected sample size). *The expected sample size of a* DPP *with kernel* $\boldsymbol{L}$ *and marginal kernel* $\boldsymbol{K}$ *is given by*

$$\mathbb{E}[|Y|] = \operatorname{tr}\left(\boldsymbol{L}(\boldsymbol{I} + \boldsymbol{L})^{-1}\right) = \operatorname{tr}(\boldsymbol{K}).$$

However, many applications require explicit control of the size of DPP samples; for this purpose, we need a measure that assigns probability 0 to any set of size different than a specific size k. This can be achieved with k-DPPs (Kulesza and Taskar, 2011).

Definition 3.1.2. A k-DPP is a measure over all subsets of $[N]$ defined by a kernel $\boldsymbol{L} \succeq 0 \in \mathbb{R}^{N \times N}$ and an integer $k \leq N$ that assigns to $S \subseteq [N]$ the probability

$$\Pr(S) = \det(\boldsymbol{L}_S)/e_k(\lambda_1, \ldots, \lambda_N),$$

if $|S| = k$, and 0, otherwise, where the λ_i are the eigenvalues of $\boldsymbol{L}$ and e_k is the k-th elementary symmetric polynomial, defined as $e_k(\lambda_1, \ldots, \lambda_N) = \sum_{|S|=k} \prod_{i \in S} \lambda_i$.

Since k-DPPs are not DPPs, they do not have marginal kernels. However, k-DPPs still admit an exact sampling algorithm; in fact, to sample from a k-DPP we only have to change how we choose the initial set of eigenvectors V (first 5 lines of Alg. 3).

Algorithm 4 Selecting eigenvectors to sample from a k-DPP

Input: eigendecomposition $(v_i, \lambda_i)_{i=1}^N$ of $\boldsymbol{L}$, desired set size $k \geq 0$.
$J \leftarrow \emptyset$
$\ell \leftarrow k$
for $i = N$ **to** 1 **do**
    if $\ell = 0$ **then**
        break
    if $u \sim U([0,1]) < \lambda_i \frac{e_{\ell-1}(\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_N)}{e_\ell(\lambda_1, \ldots, \lambda_N)}$ **then**
        $J \leftarrow J \cup \{i\}$
        $\ell \leftarrow \ell - 1$
$V \leftarrow \{v_i\}_{i \in J}$
return V

Introduced in (Kulesza and Taskar, 2011), Algorithm 4 samples the initial set of eigenvectors V. To sample from a k-DPP, one can simply run the second loop of Algorithm 3 with the set V chosen as in Algorithm 4.

Furthermore, when a low-rank decomposition of the kernel is already known, Algorithms 3 and 4 can be modified to take advantage of this pre-existing structure (Kulesza, 2012). Indeed, writing $\boldsymbol{L} = \boldsymbol{B}^\top \boldsymbol{B}$, with $\boldsymbol{B} \in \mathbb{R}^{d \times N}$ and $d < N$, the overall computational cost can be brought down to $\mathcal{O}(Ndk^2 + d^2k^3)$ (see Algorithm 5) by using the *dual kernel* $\boldsymbol{C} = \boldsymbol{B}\boldsymbol{B}^\top \in \mathbb{R}^{d \times d}$.

Algorithm 5 Sampling from a DPP with dual representation

Input: eigendecomposition $(v_i, \lambda_i)_{i=1}^N$ of $\boldsymbol{C} = \boldsymbol{B}\boldsymbol{B}^\top \in \mathbb{R}^{d \times d}$
$J, Y \leftarrow \emptyset$
for $i = 1$ **to** N **do**
    $J \leftarrow J \cup \{i\}$ with probability $\frac{\lambda_i}{\lambda_i + 1}$
$V \leftarrow \left\{ v_i / \sqrt{v_i^\top \boldsymbol{C} v_i} \right\}_{i \in J}$
while $|V| > 0$ **do**
    Select i from $\mathcal{Y}$ with probability $\Pr(i) = \frac{1}{|V|} \sum_{v \in V} (v^\top \boldsymbol{B}_i)^2$
    $Y \leftarrow Y \cup \{i\}$
    $v_0 \leftarrow$ any vector in V such that $\boldsymbol{B}_i^\top v_0 \neq 0$
    $V \leftarrow \left\{ v - \frac{v^\top \boldsymbol{B}_i}{v_0^\top \boldsymbol{B}_i} v_0 \mid v \in V \setminus \{v_0\} \right\}$
    $V \leftarrow V_\perp$, orthonormalization of V *w.r.t.* the dot product $\langle u, v \rangle = u^\top \boldsymbol{C} v$
return Y

In many cases, the decomposition $\boldsymbol{L} = \boldsymbol{B}^\top \boldsymbol{B}$ may be known for $\boldsymbol{B} \in \mathbb{R}^{d \times N}$ with $d \gg N$ (for example, when $\boldsymbol{L}$ is constructed using linear feature kernel with high-dimensional features). Fortunately, Gillenwater et al. (2012) showed that in these cases a low-dimensional, approximate decomposition of $\boldsymbol{L}$ can be constructed using random projections without significantly altering the induced DPP.

With this approximate representation, Algorithm 5 remains practical for large values of N and d, as long as the dual kernel $\boldsymbol{C}$ can be obtained efficiently.

3.2 Modeling DPPs with neural networks

Due to its $\mathcal{O}(N^3)$ cost, exact sampling for DPPs is impractical for large datasets; dual sampling (Algorithm 5) only applies when we have knowledge of a linear factorization of the DPP kernel $\boldsymbol{L}$. In some scenarios, even MCMC sampling can be prohibitively costly due to its inherently sequential nature.

Motivated by recent work on modeling set functions with neural networks (Zaheer et al., 2017; Cotter et al., 2018), we propose here to generate approximate samples via a generative network; this allows for simple parallelization while simultaneously benefiting from recent improvements in specialized architectures for neural network models (*e.g.*, parallelized matrix multiplications).

In the remainder of this chapter, we introduce DPPNETs: generative deep models that produce DPP-like samples for arbitrary ground sets and feature representations for the items in $\mathcal{Y}$. We show that a simple, carefully constructed neural network can generate approximate DPP samples with very little overhead, while maintaining some of the fundamental theoretical properties of DPP measures. Furthermore, we show that DPPNETs can be trivially employed to sample from a conditional DPP (*i.e.*, sampling S such that $A \subseteq S$ is predefined) and for greedy mode approximation. Material covered in this section is based on (Mariet et al., 2019b).

3.2.1 Generating DPP samples with deep models

We begin by considering which DPP properties we may leverage to simplify training, as well as certain DPP-specific properties we may want to maintain within the generative model. These properties guide our choice for neural network architecture.

Simple computation of marginal probabilities. Given a set Y sampled by a DPP with kernel $\boldsymbol{L}$ and $S \subseteq Y$, the probability $\Pr(\{i\} \cup S \subseteq Y \mid S \subseteq Y)$ of item $i \notin S$ also being sampled admits the closed form (where we write $\bar{S} := [N] \setminus S$):

$$\Pr(\{i\} \cup S \subseteq Y \mid S \subseteq Y) = 1 - [(\boldsymbol{L} + \boldsymbol{I}_{\bar{S}})^{-1}]_{ii}. \tag{3.4}$$

Although Eq. 3.4 requires an expensive matrix inversion, such costs can be offset during off-line training. These probabilities are the vector that DPPNET seeks to output. This signal has the advantage of providing information during every step of training, compared to other downstream possibilities (*e.g.*, NLL of generated sets).

Sequential sampling. When drawing samples from a DPP, the standard DPP sampling algorithm (Kulesza and Taskar, 2012, Alg. 1) generates the sample as a sequence, adding items one after the other until reaching a pre-determined size, similarly to Alg. 6. We take advantage of this by recording all intermediary subsets generated by the DPP when sampling training data: in practice, instead of training on n subsets of size k, we train on kn subsets of size $0, \ldots, k-1$. Thus, our model is much like an unrolled recurrent neural network (Elman, 1990; Jordan, 1990).

The sequential form of the exact sampling algorithm also makes it amenable to simple modifications that yield greedy sampling algorithms (Chen et al., 2018). This motivates a sequential sampling algorithm for DPPNET (Alg. 6), yielding a straightforward greedy form of DPPNET sampling without additional complexity.

Algorithm 6 Sampling and greedy mode for DPPNET

Input: Initial set S, target size k, feature matrix $\boldsymbol{\Phi}$
while $|S| < k$ **do**
  $v \leftarrow \textsc{DppNet}\,(S, \boldsymbol{\Phi})$
  if sampling **then**
    $i \sim \text{Multinomial}(v/\|v\|)$
  else if greedy mode **then**
    $i \leftarrow \operatorname{argmax} v$
  $S \leftarrow S \cup \{i\}$
return S

Closure under conditioning. Recall that DPPs are closed under conditioning: given $A \subseteq \mathcal{Y}$, the conditional distribution over $\mathcal{Y} \setminus A$ given by $\Pr(S = A \cup B \mid A \subseteq S)$ (for $B \cap A = \emptyset$) is also a DPP with kernel $\boldsymbol{L}^A = ([(\boldsymbol{L} + \boldsymbol{I}_{\bar{A}})^{-1}]_{\bar{A}})^{-1} - \boldsymbol{I}$. Although conditioning comes at the cost of an expensive matrix inversion, this property make DPPs well-suited to applications requiring diversity in conditioned sets, such as basket completion for recommender systems.

Standard deep generative models such as (Variational) Auto-Encoders (Kingma and Welling, 2014) (VAEs) and Generative Adversarial Networks (Goodfellow et al.,

2014) (GANs) would not enable conditioning operations during sampling — such operations would have to take place in the latent space. With the DPPNET architecture, we can sample a set via Alg. 6, which allows for trivial basket-completion type conditioning operations.

Summary. Given an input subset $S \subseteq \mathcal{Y}$, a DPPNET returns a probability vector over all items in $\mathcal{Y}$. The next item to add to S is sampled from the associated categorical distribution. During training, the DPPNET minimizes the distance between the generated probability vector and the true marginal probabilities under the target DPP. When generating training data from the true DPP, we use the additional information produced during exact sampling to obtain additional training data.

3.2.2 Sampling over fixed and varying ground sets

In simple settings, we wish to draw samples over a ground set with a fixed feature matrix representation $\mathbf{\Phi} \in \mathbb{R}^{N \times d}$. In this case, DPPNET's knowledge of $\mathbf{\Phi}$ can be obtained during training, and so DPPNET is a simple feed-forward network taking a partially sampled set S as input. However, in many cases the feature representation of the items $\mathbf{\Phi}$ will vary across time or contexts: for example, when $\mathbf{\Phi}$ represents a pool of products that are available for sale at a given time, or social media posts whose relevance that varies based on context. In such cases, DPPNET also takes as input the feature matrix $\mathbf{\Phi}$.

Naively adding $\mathbf{\Phi}$ as input to a stack of feed-forward connections would require deeper networks and larger layers, increasing learning and sampling time. Instead, we draw inspiration for the dot-product attention from (Vaswani et al., 2017). Intuitively, attention is a vector computed by the network that indicates relevant pars of the inputs. For DPPNET, this attention will *reweight* $\mathbf{\Phi}$ based on pre-sampled items.

Attention functions were introduced in (Bahdanau et al., 2015) for neural network-based translation. Intuitively, attention functions reweight inputs to a neural network, assigning higher weights to input segments that carry information relevant to the task at hand. In (Vaswani et al., 2017), the attention mechanism takes three matrices as

input, which can be viewed as a (1) matrix of keys $\boldsymbol{K}$, (2) a matrix of values $\boldsymbol{V}$, and (3) a query matrix $\boldsymbol{Q}$. The attention matrix $\boldsymbol{A}$, which reweights the values $\boldsymbol{V}$, is obtained as $\boldsymbol{A} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d})$ where d is the dimension of each query/key: the inner product acts as a proxy to the similarity between queries and keys.

Here, the feature representation of items in the input set S acts as the query $\boldsymbol{Q} = \boldsymbol{\Phi}_{S,:} \in \mathbb{R}^{k \times d}$. The features $\boldsymbol{\Phi} \in \mathbb{R}^{N \times d}$ of our ground set are both the keys and the values. In order for the attention mechanism to make sense in the framework of DPP modeling, we make two modifications to the original attention mechanism:

- We want our network to focus on items that are *dissimilar* to the query (input subset): for each item i in the input subset S, we compute its pairwise dissimilarity to each item $i \in \mathcal{Y}$ as the vector $d_i = 1 - \text{softmax}(\boldsymbol{Q}_i\boldsymbol{\Phi}^\top/\sqrt{d})$.

- We return a vector $a \in \mathbb{R}^N$ in the probability simplex such that $a_j \propto \prod_{i \in S} d_{ij}$. This allows us to have a fixed-size input to the network, and enforces the fact that similarity to a *single* item is enough to disqualify an element from the ground set.

Finally, our attention vector a is computed via the *inhibitive attention* mechanism

$$a' = \bigodot_{i \in S} \Big(1 - \text{softmax}(\boldsymbol{Q}\boldsymbol{\Phi}^\top/\sqrt{d})\Big), \qquad a = a'/\|a'\|_1, \tag{3.5}$$

where $\odot$ represents the row-wise multiplication operator; this vector can be computed in $\mathcal{O}(kDN)$ time. The attention component finally feeds the element-wise multiplication of each row of $\boldsymbol{V}$ with the attention vector a to the feed-forward block, effectively reweighting each ground set item by its dissimilarity to the pre-selected items in S.

Remark 3.2.1. Dual sampling (Alg. 5) is efficient only when the kernel is constructed as $\boldsymbol{L} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$; for non-linear kernels, a low-rank decomposition of $\boldsymbol{L}$ must first be obtained, which in the worst case requires $\mathcal{O}(N^3)$ operations. In comparison, DPPNET can be trained on non-linear DPP kernels, while only requiring $\boldsymbol{\Phi}$ as input.

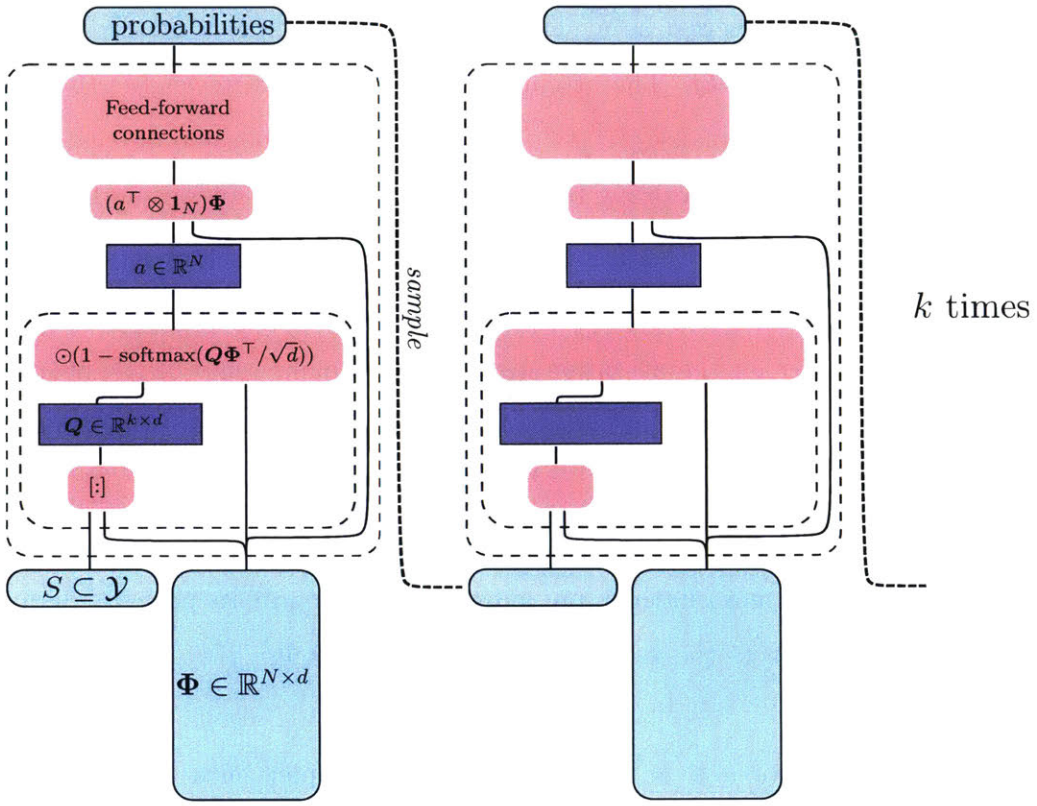


Figure 3-2: Transformer network architecture (variable feature matrix $\mathbf{\Phi}$).

3.2.3 Preserving log-submodularity

We now investigate whether any neural network approximation of determinantal point processes can inherit some of their negative dependence properties. Although the SR property is delicate and maintained by only few operations, log-submodularity, also a fundamental property in discrete optimization, see *e.g.*, (Zagoruyko and Komodakis, 2016; Buchbinder et al., 2012; Grötschel et al., 1981), is more robust. We show here that (log) submodularity can be inherited by a generative model.

Theorem 3.2.1. *Let $\mathcal{P}$ be a strictly submodular distribution over subsets of $\mathcal{Y}$, and $\mathcal{Q}$ be a function over the same space such that*

$$D_{\text{TV}}(\mathcal{P}, \mathcal{Q}) \leq \min_{S \neq T \notin \{\emptyset, \mathcal{Y}\}} \frac{1}{4} \left[\mathcal{P}(S) + \mathcal{P}(T) - \mathcal{P}(S \cup T) - \mathcal{P}(S \cap T))\right], \tag{3.6}$$

where D_{TV} indicates the total variational distance. Then $\mathcal{Q}$ is also submodular.

Proof. Let $S, T \subseteq \mathcal{Y}$ be subsets such that $S \neq T$ and $S, T \neq \emptyset, \mathcal{Y}$ (the submodular inequality is a trivial equality for $S = T$, $S = \emptyset$ and $S = \mathcal{Y}$, for any distribution). We must prove $\mathcal{Q}(S) + \mathcal{Q}(T) \geq \mathcal{Q}(S \cup T) + \mathcal{Q}(S \cap T)$. Let ϵ be defined by

$$\epsilon := \min_{S \neq T \notin \{\emptyset, \mathcal{Y}\}} \mathcal{P}(S) + \mathcal{P}(T) - \mathcal{P}(S \cup T) - \mathcal{P}(S \cap T)).$$

From the strict submodularity hypothesis, we know $\epsilon > 0$. By definition of ϵ,

$$\mathcal{P}(S) + \mathcal{P}(T) - \mathcal{P}(S \cup T) - \mathcal{P}(S \cap T)) \geq \epsilon.$$

From (3.6), we know that $\max_{S \subseteq \mathcal{Y}} |\mathcal{P}(S) - \mathcal{Q}(S)| \leq \epsilon/4$. It follows that

$$\mathcal{Q}(S) + \mathcal{Q}(T) - \mathcal{Q}(S \cup T) + \mathcal{Q}(S \cap T) \geq \mathcal{P}(S) + \mathcal{P}(T) - \mathcal{P}(S \cup T) - \mathcal{P}(S \cap T) - \epsilon \quad \geq 0,$$

which proves the submodularity of $\mathcal{Q}$. □

From Thm. 3.2.1 and the equivalence of norms in finite dimensional spaces, we can furthermore state the following result.

Corollary 3.2.1.1. *Let $\mathcal{P}_{\boldsymbol{L}}$ be a strictly log-submodular* DPP *over $\mathcal{Y}$ and* DPPNET *be a network with a loss function of the form $\|p - q\|$ where $\|\cdot\|$ is a norm and $p \in \mathbb{R}^{2^N}$ (resp. q) is the probability vector assigned by the* DPP *(resp. the* DPPNET*) to each subset of $\mathcal{Y}$. Let $\alpha = \max_{\|x\|_\infty = 1} \|x\|^{-1}$. If* DPPNET *converges to a loss smaller than*

$$\min_{S \neq T \neq \emptyset, \mathcal{Y}} \frac{1}{4\alpha} \left[\mathcal{P}(S) + \mathcal{P}(T) - \mathcal{P}(S \cup T) - \mathcal{P}(S \cap T))\right],$$

its generative distribution is log-submodular.

Remark 3.2.2. Cor. 3.2.1.1 is generalizable to the KL divergence loss $D_{\mathrm{KL}}(\mathcal{P} \| \mathcal{Q})$ via Pinsker's inequality.

Remark 3.2.3. Cor. 3.2.1.1 also holds for (log)-supermodular functions.

Checking numerically whether the conditions for Corollary 3.2.1.1 apply during training is NP-hard: the results of this section are purely theoretical. However, The-

orem 3.2.1 and Corollary 3.2.1.1 provide an additional justification for the use of probabilities in the objective function of DPPNET.

3.2.4 Experimental results

To evaluate DPPNET, we look at its performance both as a proxy for a DPP over a fixed ground set (Section 3.2.4) and as a tool for generating diverse subsets of varying ground sets (Section 3.2.4). Our models are trained with TensorFlow, using the Adam optimizer. Hyper-parameters are tuned to maximize the normalized log-likelihood of generated subsets. We compare DPPNET to DPPs and two additional baselines:

- UNIF: Uniform sampling over the ground set.
- HCP: Matérn hard core point processes. Points are sampled from a Poisson distribution, then thinned out to remove points within a distance $r < 0.2$ (chosen by cross-validation) from each other.
- k-MEDOIDS: k-medoids clustering (Hastie et al., 2001, 14.3.10) applied to the ground set; the distance between points is the one also used by DPP kernel.

We use the negative log-likelihood (NLL) of a subset under a fixed DPP over the ground set to evaluate the subsets obtained by all methods. This choice is motivated by the following considerations: DPPs have become a standard way of measuring and enforcing diversity over subsets of data in machine learning, and b) to the extent of our knowledge, there is no other standard method to benchmark the diversity of a selected subset that depends on specific dataset encodings.

Sampling over the unit square

We begin by analyzing the performance of a DPPNET trained on a DPP over a fixed ground set: the unit square. Our choice of the unit square is motivated by the need for diverse sampling methods on the unit hypercube, *e.g.*, quasi-Monte Carlo methods, Latin hypercube sampling (McKay et al., 1979) and low discrepancy sequences.

The ground set consists of the 100 points lying at the intersections of the 10×10 grid on the unit square. The DPP is defined by setting its kernel $\boldsymbol{L}$ to the exponentiated quadratic form $\boldsymbol{L}_{ij} = \exp(-\|x_i - x_j\|_2^2/2)$. As the ground set is fixed, the associated neural networks do not use the inhibitive attention mechanism and consists only of a feed-forward architecture.

We report the performance of the different sampling methods in Fig. 3-3. Visually (Fig. 3-3a) and quantitatively (Fig. 3-3b), DPPNET improves significantly over all other baselines. Furthermore, greedily sampling the mode from the DPPNET achieves a better NLL than DPP samples themselves. Note that a direct greedy approximation of the DPP would be significantly costlier. Numerical results are reported in Table 3.1.

Table 3.1: NLL under $\mathcal{P}_{\boldsymbol{L}}$ for sets of size $k = 20$ sampled over the unit square. DPPNET achieves comparable performance to the DPP, outperforming the other baselines.

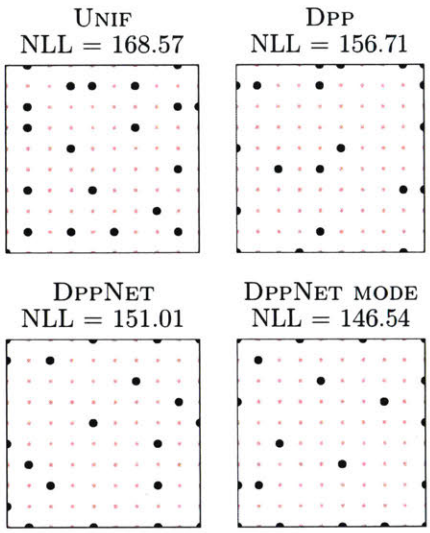
DPP	UNIFORM	HCP	k-MEDOIDS	DPPNET
154.95 ± 2.93	180.53 ± 9.56	163.40 ± 5.87	169.37 ± 6.41	153.44 ± 2.07

Sampling over variable ground sets

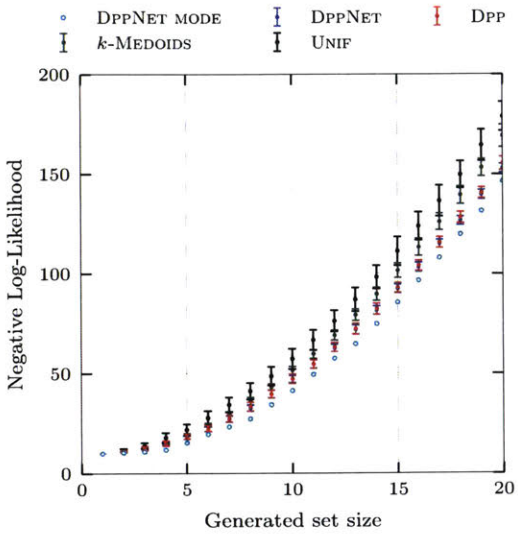
We evaluate the performance of DPPNETs on varying ground set sizes through the MNIST (LeCun and Cortes, 2010), CelebA (Liu et al., 2015), and MovieLens (Harper and Konstan, 2015) datasets. To obtain the feature representations of each point, we use a Variational Auto-Encoder (Kingma and Welling, 2014) for MNIST and CelebA[1] (generating features of length 32). For MovieLens, we obtain feature vectors of length 10 for each movie by applying non-negative matrix factorization to the rating matrix.

During training, the networks take as input feature matrices drawn from the training portion of the associated dataset (MNIST, CelebA, MovieLens). At test time, the feature matrices $\boldsymbol{\Phi}$ are drawn from the associated test portions.

[1]Details on the encoders are provided in Appendix A.2.1.



(a) Sampling subsets of size 20 over the unit square.



(b) Normalized log-likelihood of samples drawn from all methods as a function of the sampled set size.

Figure 3-3: Sampling on the unit square with a DPPNET (1 hidden layer with 841 neurons) trained on a single DPP kernel. Visually, DPPNET gives similar results to the full DPP (left). As evaluated by DPP NLL, the DPPNET's mode achieves superior performance to the full DPP, and DPPNET sampling overlaps completely with DPP sampling (right).

The DPPNET is trained based on samples from DPPs with a linear kernel for MovieLens and with an exponentiated quadratic kernel for the image datasets. Bandwidths were set to $\beta = 0.0025$ for MNIST and $\beta = 0.1$ for CelebA in order to obtain a DPP average sample size ≈ 20: recall that for a DPP with kernel $\boldsymbol{L}$, the expected sample size is given by Proposition 3.1.5.

For MNIST, Fig. 3-4 shows images selected by the baselines and the DPPNET, chosen among 100 digits with either random labels or all identical labels; visually, DPPNET and DPP samples provide a wider coverage of writing styles. The NLL of samples from DPPNET decay significantly, whereas the DPPNET mode continues to maintain competitive performance with DPP samples. For this reason, all subsequent experiments use only the DPPNET mode when working with varying ground sets.

Numerical results for MNIST are reported in Table 3.2; additionally to the previous baselines, we also considered learning a feed-forward neural network without

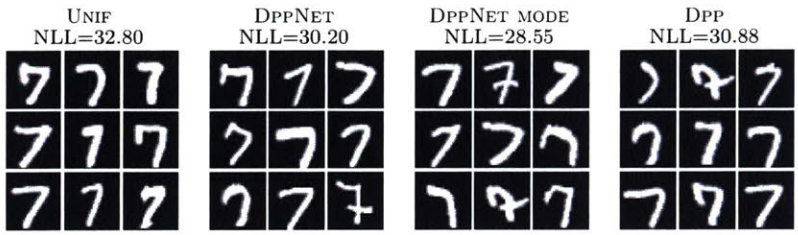


Figure 3-4: Digits sampled from MNIST based on their encodings. Only images labeled as "7" are provided to the various samplers; however, the DPPNET itself (3 layers of 365 neurons) is trained on all types of digits, and outperforms all baselines.

Table 3.2: NLL (mean ± standard error) under the true DPP of samples drawn uniformly, according to the mode of the DPPNET, and from the DPP itself. We sample subsets of size 20; for each class of digits we build 25 feature matrices $\mathbf{\Phi}$ from encodings of those digits, and for each feature matrix we draw 25 different samples. Bolded numbers indicate the best-performing (non-DPP) sampling method.

	0	1	2	3	4
DPP	52.2 ± 0.1	60.5 ± 0.1	49.8 ± 0.0	50.7 ± 0.1	51.0 ± 0.1
UNIF	54.9 ± 0.1	65.1 ± 0.1	51.5 ± 0.1	52.9 ± 0.1	53.3 ± 0.1
MEDOIDS	55.1 ± 0.1	65.0 ± 0.1	51.5 ± 0.0	52.9 ± 0.1	53.1 ± 0.1
DPPNET MODE	**53.6 ± 0.3**	**63.6 ± 0.4**	**50.8 ± 0.2**	**51.4 ± 0.3**	**51.6 ± 0.4**

	5	6	7	8	9	All
DPP	50.4 ± 0.1	51.6 ± 0.1	51.5 ± 0.1	50.9 ± 0.1	52.7 ± 0.1	49.2 ± 0.1
UNIF	52.4 ± 0.1	54.6 ± 0.1	55.1 ± 0.1	53.3 ± 0.1	56.2 ± 0.1	51.6 ± 0.1
MEDOIDS	52.4 ± 0.0	54.4 ± 0.1	55.1 ± 0.1	53.2 ± 0.1	56.1 ± 0.1	51.0 ± 0.1
DPPNET MODE	**51.8 ± 0.3**	**52.8 ± 0.3**	**52.7 ± 0.4**	**50.9 ± 0.3**	**55.0 ± 0.4**	48.6 ± 0.2

attention. After hyper-parameter tuning, we found that the best architecture for this model consisted in 6 layers of 585 neurons each, showing that inhibitive attention is necessary to reduce the number of parameters in the model.

Strikingly, DPPNET performs significantly better than other baselines even on feature matrices drawn from a single class of digits (Table 3.2), despite distribution over input matrices during training being much less specialized. This implies that DPPNET sampling for dataset summarization may be leveraged to focus on sub-areas of datasets that are identified as areas of interest. Numerical results for CelebA and MovieLens are reported in Table 3.3, confirming the modeling ability of DPPNETs.

Table 3.3: NLLs on CelebA and MovieLens (mean ± standard error); 20 samples of size 20 were drawn across 20 different feature matrices each for a total of 100 samples per method; DPPNET is the non-DPP model achieves the best NLLs.

DATASET	KERNEL	DPP	UNIFORM	k-MEDOIDS	DPPNET Mode
CelebA	Exp. quadratic	49.04 ± 2.03	50.84 ± 1.53	51.18 ± 1.34	**49.28 ± 1.57**
MovieLens	Linear	84.29 ± 0.20	92.04 ± 0.17	88.90 ± 0.16	**80.21 ± 0.33**

DppNet for kernel reconstruction

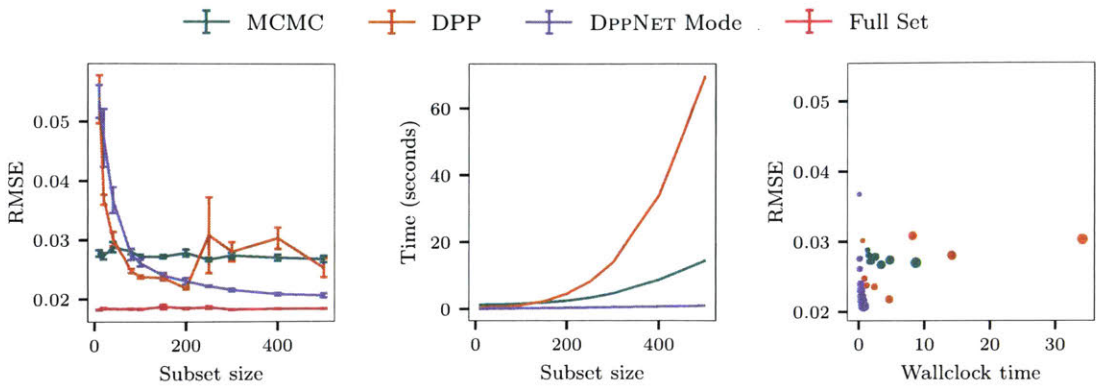
As a final experiment, we evaluate DPPNET's performance on a downstream task for which DPPs have been shown to be useful: kernel reconstruction using the Nyström method (Nyström, 1930; Williams and Seeger, 2001). Given a positive semidefinite matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, the Nyström method approximates $\mathbf{K}$ by $\hat{\mathbf{K}} = \mathbf{K}_{:,S}\mathbf{K}_{S,S}^{\dagger}\mathbf{K}_{S,:}$ where $\mathbf{K}^{\dagger}$ denotes the pseudoinverse of $\mathbf{K}$. The Nyström method is a popular method to scale up kernel methods and has found many applications in machine learning — see *e.g.*, (Bach and Jordan, 2002; Shen et al., 2009; Fowlkes et al., 2004; Talwalkar et al., 2013). The approximation quality directly depends on the choice of subset S.

Recently, DPPs have been shown to be a competitive approach for selecting S (Li et al., 2016c; Mariet et al., 2018). Following the approach of Li et al. (2016c), we evaluate the quality of the kernel reconstruction by learning a regression kernel $\mathbf{K}$ on a training set, and reporting the prediction error on the test set using the Nyström reconstructed kernel $\hat{\mathbf{K}}$.

Specifically, we apply Kernel Ridge Regression to the Ailerons dataset[2] also used in (Li et al., 2016c); we subsample 1,000 points from each dataset and use an RBF kernel with bandwidth β and regularization parameter λ chosen by 10-fold cross-validation. Results are averaged over 10 random subsets of data, using the swapchain sampler initialized with k-means++. Additionally to the full DPP, we also compare DPPNET to the MCMC sampling method with quadrature acceleration (Li et al., 2016c,b). Figure 3-5 reports the resulting Root Mean Squared Error (RMSE) on the

[2] `http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

withheld test set as well as the respective runtimes of each method.



(a) Test error (b) Wallclock time (c) Error vs. time

Figure 3-5: Results for the Nyström approximation experiments, comparing DppNet to the fast MCMC sampling method of Li et al. (2016c) according to root mean squared error (RMSE) and wallclock time. Subsets selected by DPPNet achieve comparable and lower RMSE than a DPP and the MCMC method respectively while being significantly faster. In (c), the relative size of the marker represents the size of the sampled subset.

Fig. 3-5 confirms that DppNet is a viable alternative to Dpps for kernel reconstructions. As expected, DppNet is significantly faster than even approximate MCMC sampling. Note finally that while all methods were run on CPU, DppNet is amenable to even further acceleration using GPUs.

3.3 Open problems

We close this chapter by mentioning some remaining open problems which we believe to be of interest, either for their theoretical ramifications or for potential applications.

- DppNets are *not* exchangeable: for a permutation σ of $[k]$, two sequences $i_1, \ldots, i_k$ and $\sigma(i_1), \ldots, \sigma(i_k)$ will not in general have the same probability under a DppNet. Although non-exchangeability can be an asset when sampling a ranking of items, exchangeability can be enforced by leveraging previous work (Zaheer et al., 2017), and is a path worth investigating for DppNets.

- The scaling of the DppNet's training time with the ground set size remains an

open question. However, standard tricks to enforce fixed-size ground sets such as sub-sampling from the dataset may be applied to DPPNETs. Similarly, if further are speedups necessary, sub-sampling from the ground set — a standard approach for DPP sampling over very large set sizes — can be combined with DPPNET sampling.

- In light of our results on dataset sampling, the question of whether encoders can be trained to produce encodings conducive to dataset summarization via DPPNETs seems of particular interest. Assuming knowledge of the relative diversity of a large quantity of subsets, an end-to-end training of the encoder and the DPPNET simultaneously may yield interesting results.

- Our analysis of DPPNETs extends further than DPPs: through Theorem 3.2.1 and Corollary 3.2.1.1, we have shown that under the right training conditions, generative models can inherit (log) submodularity, a crucial step in approximating submodular functions, with significant implications for discrete optimization and game theory. Understanding which additional properties of training distributions (including the larger class of negative dependence characterizations) may be conserved through careful training remains an open question which we believe to be significant to the community in general.

- Finally, the setup used to learn a DPP with neural networks can be generalized to learning any distribution over subsets, as long as marginal probabilities of adding an item to a subset can be computed efficiently. As optimizing and approximating set functions with intractable closed forms is a fundamental problem within combinatorial optimization, analyzing DPPNET-style approaches in this setting may prove to be a valuable research direction.

Chapter 4

Learning DPPs from data

In the previous chapter, we have seen that determinantal point processes provide an elegant framework for the efficient sampling of diverse, high-quality subsets from a ground set of items. The diversity and quality metrics for each application of DPPs are embedded in the chosen DPP's kernel; this kernel can either be constructed based on expert knowledge of the required notions of diversity and quality, or alternatively, constructed based on feature representations of items in the ground set.

In many cases, however, the right DPP kernel may not be known *a priori*. For example, consider the setting investigated by Wilhelm et al. (2018): using DPPs to provide video (YouTube) recommendations drawn from a very large pool of possibilities. There is no obvious choice of kernel that will yield high-quality recommendations; however, the authors have access to sets of videos enjoyed by various users. The task, then, is to *learn* a DPP kernel that will assign high probabilities to such desirable subsets; this kernel can then be used to provide better recommendations.

Clearly, the ability to learn a DPP kernel from data is key to applying determinantal point processes; this chapter focuses on this learning problem, addressing in particular how to learn a kernel efficiently, and how to incorporate knowledge of low-quality subsets into the learning procedure.

We begin by formally introducing the learning problem for DPPs and discussing its properties. To address the high-complexity cost of learning the DPP kernel, we introduce a matrix structure that enables fast determinant and inverse computations.

Then, in order to incorporate prior knowledge of "undesirable" subsets to which the model should assign low probabilities, we introduce Contrastive Estimation: a novel optimization problem that incorporates this additional information. We show that in practice, DPPs learned with Contrastive Estimation have a better modeling performance on recommendation tasks.

4.1 Learning the DPP kernel

The standard[1] formulation of the learning problem for determinantal point processes is the *maximum likelihood estimation* (MLE) problem: given subsets $S_1, \ldots, S_n \subseteq [N]$, we wish to learn a DPP kernel $\boldsymbol{L}$ that maximizes the probability of observing the S_i. Formally, the MLE problem assumes knowledge of the ground set $[N]$ and subsets $S_1, \ldots, S_n \subseteq [N]$ drawn in *i.i.d.* fashion, and is cast as the optimization problem

$$\text{Find } \boldsymbol{L} \in \operatorname*{argmax}_{\boldsymbol{L} \succeq 0} \phi(\boldsymbol{L}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \log\det(\boldsymbol{L}_{S_i}) - \log\det(\boldsymbol{I} + \boldsymbol{L}), \tag{4.1}$$

where $\phi(\boldsymbol{L})$ is the log-likelihood of observing $S_1, \ldots, S_n$ under a DPP with kernel $\boldsymbol{L}$. The optimization problem 4.1 is non-convex in $\boldsymbol{L}$.

Problem 4.1 can be approached using stochastic gradient ascent (SGA), and is commonly augmented with further constraints on $\boldsymbol{L}$: most commonly, requiring that $\boldsymbol{L}$ have a low rank. This choice imposes a clear limit on the modeling power of the DPP: any subset of size larger than the kernel rank has probability 0 under the DPP.

In the more general case where the kernel $\boldsymbol{L}$ is learned without further constraints, an Expectation-Maximization scheme was suggested by Gillenwater et al. (2014), and a fixed-point ("Picard") iteration, described by the simple update rule

$$\boldsymbol{L} \leftarrow \boldsymbol{L} + \boldsymbol{L}\nabla\phi(\boldsymbol{L})\boldsymbol{L}, \tag{4.2}$$

was introduced by Mariet and Sra (2015). Among other advantages, both EM and the

[1] We will see later a less common formulation of the DPP learning problem.

Picard are less likely than SGA to learn near-diagonal kernels, which cannot capture negative interactions (Gillenwater et al., 2014).

4.2 Learning DPPs with Kronecker kernels

Although the $\mathcal{O}(N^3)$ cost of exact sampling from a DPP has motivated a string of works on approximate sampling methods such as MCMC samplers (Kang, 2013; Li et al., 2016c), core-set based samplers (Li et al., 2015), and generative neural networks (Mariet et al., 2019b), the task of learning a DPP from data has received less attention. Methods in (Gillenwater et al., 2014; Mariet and Sra, 2015) cost $\mathcal{O}(N^3)$ per iteration, which is clearly unacceptable for realistic settings. This burden is partially alleviated in (Gartrell et al., 2017) and (Osogami et al., 2018), wherein the authors learn low-rank DPPs; this comes at the expense of being unable to sample subsets larger than the chosen rank.

Unsurprisingly, the $\mathcal{O}(N^3)$ cost stems from determinant computations and matrix inversions. To improve learning speed without sacrificing the ability to model large subset sizes, we choose to take advantage of a structure that significantly speeds up both these operations: Kronecker products.

Specifically, we consider DPPs with kernels that are the Kronecker product of smaller kernels $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \cdots \otimes \boldsymbol{L}_m$; each *sub-kernel* $\boldsymbol{L}_i$ is a smaller positive definite matrix. We will see that, additionally to decreasing the number of parameters required to specify the DPP from $\mathcal{O}(N^2)$ to $\mathcal{O}(N^{2/m})$, this structure also enables faster sampling and learning. The high-level comparison of Kronecker kernels to other structured and unstructured kernels is summarized in Table 4.1; the following material was first derived in (Mariet and Sra, 2016b).

Using Kronecker products to scale up matrix models is a popular and effective idea in several machine learning settings (Wu et al., 2005; Martens and Grosse, 2015; Flaxman et al., 2015; Zhang et al., 2015); however, we will see that its efficient execution for DPPs turns out to be more challenging.

MODEL	EXACT SAMPLING	STOCHASTIC LEARNING	
No structure	$\mathcal{O}(N^3 + Nk^3)$	SGD	$\mathcal{O}(N^3 + \kappa^3)$
		EM (Gillenwater et al., 2014)	$\mathcal{O}(N^3 + N\kappa^2)$
		PICARD (Mariet and Sra, 2015)	$\mathcal{O}(N^3 + \kappa^3)$
Low rank	$\mathcal{O}(Nd^2 + Ndk^2 + d^2k^3)$	SGA (Gartrell et al., 2017)	$\mathcal{O}(dN^2 + \kappa^3)$
		SGA (Osogami et al., 2018)	$\mathcal{O}(d^2N + \kappa^3)$
Kronecker	2 kernels: $\mathcal{O}(N^{3/2} + N\kappa^2)$ 3 kernels: $\mathcal{O}(N\kappa^2)$	KRK-PICARD	$\mathcal{O}(N^{3/2} + N\kappa^2)$

Table 4.1: Sampling and per-iteration MLE learning cost for DPPs; N is the size of the ground set, n the number of training subsets, κ the size of the largest training set, and d the dimension (if relevant) of the kernel.

Preliminaries: Kronecker products

We begin by recalling some basic properties of Kronecker products we will need in our analysis; we omit proofs of these well-known results.

The Kronecker (tensor) product of a matrix $\boldsymbol{A} \in \mathbb{R}^{p \times q}$ with $\boldsymbol{B} \in \mathbb{R}^{r \times s}$ is defined as the $pr \times qs$ block matrix $\boldsymbol{A} \otimes \boldsymbol{B} = [a_{ij}\boldsymbol{B}]_{i,j=1}^{p,q}$. We denote the $r \times s$ block $a_{ij}\boldsymbol{B}$ in $\boldsymbol{A} \otimes \boldsymbol{B}$ by $(\boldsymbol{A} \otimes \boldsymbol{B})_{(ij)}$ for any valid pair (i, j), and extend the notation to non-Kronecker product matrices to indicate the submatrix of size $r \times s$ at position (i, j).

Proposition 4.2.1. *Let $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ be matrices of sizes so that $\boldsymbol{AC}$ and $\boldsymbol{BD}$ are well-defined. Then,*

(i) If $\boldsymbol{A}, \boldsymbol{B} \succeq 0$ then $\boldsymbol{A} \otimes \boldsymbol{B} \succeq 0$.

(ii) If $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible then so is $\boldsymbol{A} \otimes \boldsymbol{B}$, with $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$.

(iii) $(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{AC}) \otimes (\boldsymbol{BD})$.

An important consequence of Prop. 4.2.1(*iii*) is the following corollary.

Corollary 4.2.1.1. *Let $\boldsymbol{A} = \boldsymbol{P}_A \boldsymbol{D}_A \boldsymbol{P}_A^\top$, $\boldsymbol{B} = \boldsymbol{P}_B \boldsymbol{D}_B \boldsymbol{P}_B^\top$ be the eigenvector decompositions of $\boldsymbol{A}$, $\boldsymbol{B}$. Then $\boldsymbol{A} \otimes \boldsymbol{B}$ diagonalizes as $(\boldsymbol{P}_A \otimes \boldsymbol{P}_B)(\boldsymbol{D}_A \otimes \boldsymbol{D}_B)(\boldsymbol{P}_A \otimes \boldsymbol{P}_B)^\top$.*

We will also require partial trace operators:

Definition 4.2.1. Let $\boldsymbol{A} \in \mathbb{R}^{N_1 N_2 \times N_1 N_2}$. The *partial traces* $\mathrm{Tr}_1(\boldsymbol{A})$ and $\mathrm{Tr}_2(\boldsymbol{A})$ are defined as follows:

$$\mathrm{Tr}_1(\boldsymbol{A}) := \left[\mathrm{Tr}(\boldsymbol{A}_{(ij)})\right]_{1 \le i,j \le N_1} \in \mathbb{R}^{N_1 \times N_1}, \qquad \mathrm{Tr}_2(\boldsymbol{A}) := \sum\nolimits_{i=1}^{N_1} \boldsymbol{A}_{(ii)} \in \mathbb{R}^{N_2 \times N_2}.$$

The action of partial traces is easy to visualize: indeed, $\mathrm{Tr}_1(\boldsymbol{A} \otimes \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B})\boldsymbol{A}$ and $\mathrm{Tr}_2(\boldsymbol{A} \otimes \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{A})\boldsymbol{B}$. For us, the most important property of partial trace operators is their positivity.

Proposition 4.2.2. Tr_1 *and* Tr_2 *are positive operators,* i.e., *for* $\boldsymbol{A} \succ 0$, $\mathrm{Tr}_1(\boldsymbol{A}) \succ 0$ *and* $\mathrm{Tr}_2(\boldsymbol{A}) \succ 0$.

4.2.1 Learning with alternating updates

We begin by considering the question of learning a Kronecker product kernel matrix from n observed subsets $S_1, \ldots, S_n$, where the kernel is a Kronecker product of two smaller kernels: $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \boldsymbol{L}_2$. For reasons that will become apparent, it suffices to two sub-kernels.

Recall that the MLE problem is given by

$$\max_{\boldsymbol{L} \succ 0} \phi(\boldsymbol{L}), \text{ where } \phi(\boldsymbol{L}) = \frac{1}{n} \sum_{i=1}^{n} \left(\log\det(\boldsymbol{L}_{S_i}) - \log\det(\boldsymbol{L} + \boldsymbol{I})\right). \tag{4.3}$$

Writing $\boldsymbol{U}_i$ as the indicator matrix for S_i of size $N \times |S_i|$ so that $\boldsymbol{L}_{S_i} = \boldsymbol{U}_i^\top \boldsymbol{L} \boldsymbol{U}_i$, the gradient of ϕ is easily seen to be

$$\boldsymbol{\Delta} := \nabla\phi(\boldsymbol{L}) = \frac{1}{n} \sum\nolimits_{i=1}^{n} \boldsymbol{U}_i \boldsymbol{L}_{S_i}^{-1} \boldsymbol{U}_i^\top - (\boldsymbol{L} + \boldsymbol{I})^{-1}. \tag{4.4}$$

Mariet and Sra (2015) derived an iterative method ("the Picard iteration") for computing an $\boldsymbol{L}$ that solves $\boldsymbol{\Delta} = 0$ by running the simple iteration $\boldsymbol{L} \leftarrow \boldsymbol{L} + \boldsymbol{L}\nabla\phi(\boldsymbol{L})\boldsymbol{L}$ (Eq. 4.2). Crucially, this iteration preserves the necessary condition $\boldsymbol{L} \succeq 0$, and moreover guarantees to monotonically increase the log-likelihood ϕ.

Unfortunately, these benefits accrue at a cost of $\mathcal{O}(N^3)$ per iteration; furthermore,

a direct application of the Picard iteration to a kernel $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \cdots \otimes \boldsymbol{L}_n$ cannot guarantee that the Kronecker structure will be preserved.

With the aim of obtaining an efficient algorithm to (locally) optimize (4.3), we follow (Mariet and Sra, 2015): we first rewrite ϕ as a function of $\boldsymbol{S} = \boldsymbol{L}^{-1}$, and re-arrange terms to write

$$\phi(\boldsymbol{S}) = \underbrace{\log\det(\boldsymbol{S})}_{f(\boldsymbol{S})} + \underbrace{\frac{1}{n}\sum\nolimits_{i=1}^{n} \log\det\left(\boldsymbol{U}_i^\top \boldsymbol{S}^{-1}\boldsymbol{U}_i\right) - \log\det(\boldsymbol{I}+\boldsymbol{S})}_{g(\boldsymbol{S})}. \tag{4.5}$$

One easily verifies that f is concave, while a short argument shows that g is convex (Mariet and Sra, 2015). An appeal to the convex-concave procedure (Yuille and Rangarajan, 2002) then shows that updating $\boldsymbol{S}$ by solving $\nabla f(\boldsymbol{S}^{(k+1)}) + \nabla g(\boldsymbol{S}^{(k)}) = 0$, which is exactly what (4.2) does (Mariet and Sra, 2015, Thm. 2.2), is guaranteed to monotonically increase ϕ.

But for KronDpp this idea does not apply so easily: due to the constraint $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \boldsymbol{L}_2$, the function

$$g_\otimes : (\boldsymbol{S}_1, \boldsymbol{S}_2) \to \tfrac{1}{n}\sum\nolimits_{i=1}^{n} \log\det\left(\boldsymbol{U}_i^\top (\boldsymbol{S}_1 \otimes \boldsymbol{S}_2)^{-1}\boldsymbol{U}_i\right) - \log\det(\boldsymbol{I} + \boldsymbol{S}_1 \otimes \boldsymbol{S}_2),$$

fails to be convex, precluding an easy generalization. Nevertheless, for fixed $\boldsymbol{S}_1$ or $\boldsymbol{S}_2$ the functions

$$\begin{cases} f_1 : \boldsymbol{S}_1 \mapsto f(\boldsymbol{S}_1 \otimes \boldsymbol{S}_2) \\ g_1 : \boldsymbol{S}_1 \mapsto g(\boldsymbol{S}_1 \otimes \boldsymbol{S}_2) \end{cases} \quad \text{and} \quad \begin{cases} f_2 : \boldsymbol{S}_2 \to f(\boldsymbol{S}_1 \otimes \boldsymbol{S}_2) \\ g_2 : \boldsymbol{S}_2 \to g(\boldsymbol{S}_1 \otimes \boldsymbol{S}_2) \end{cases}$$

are once again concave (f_i) or convex (g_i). Indeed, the map $\otimes : \boldsymbol{S}_1 \to \boldsymbol{S}_1 \otimes \boldsymbol{S}_2$ is linear and f is concave, and $f_1 = f \circ \otimes$ is also concave; similarly, f_2 is seen to be concave and g_1 and g_2 are convex. Hence, by generalizing the arguments of (Yuille and Rangarajan, 2002, Thm. 2) to our "block-coordinate" setting, updating via

$$\nabla f_i\big(\boldsymbol{S}_i^{(k+1)}\big) = -\nabla g_i\big(\boldsymbol{S}_i^{(k)}\big), \quad \text{for } i = 1, 2, \tag{4.6}$$

should increase the log-likelihood ϕ at each iteration.

We now prove that the updates defined in (4.6) exist, and that updating as per (4.6) ensure positive definiteness of the iterates as well as monotonic ascent.

Proposition 4.2.3 (Positive definite iterates). *For $\boldsymbol{S}_1 \succ 0$, $\boldsymbol{S}_2 \succ 0$, the solutions to (4.6) are given by the following expressions:*

$$\nabla f_1(\boldsymbol{X}) = -\nabla g_1(\boldsymbol{S}_1) \iff \boldsymbol{X}^{-1} = \mathrm{Tr}_1\big((\boldsymbol{I} \otimes \boldsymbol{S}_2)(\boldsymbol{L} + \boldsymbol{L}\Delta\boldsymbol{L})\big)/N_2$$
$$\nabla f_2(\boldsymbol{X}) = -\nabla g_2(\boldsymbol{S}_2) \iff \boldsymbol{X}^{-1} = \mathrm{Tr}_2\big((\boldsymbol{S}_1 \otimes \boldsymbol{I})(\boldsymbol{L} + \boldsymbol{L}\Delta\boldsymbol{L})\big)/N_1.$$

Moreover, these solutions are positive definite.

Proof. The high-level idea of the proof is the following: we know that $\boldsymbol{L} \succ 0 \implies \boldsymbol{L} + \boldsymbol{L}\Delta\boldsymbol{L} \geq 0$, because $\boldsymbol{L} - \boldsymbol{L}(\boldsymbol{I} + \boldsymbol{L})^{-1}\boldsymbol{L} \succ 0$. Since the partial trace operators are positive (Prop. 4.2.2), it follows that the solutions to (4.6) are also positive definite.

Specifically, write 'vect' for the operator that stacks columns of a matrix to form a vector; conversely, 'mat' takes a vector with k^2 coefficients and returns a $k \times k$ matrix.

Let $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \boldsymbol{L}_2$, $\boldsymbol{S}_1 = \boldsymbol{L}_1^{-1}, \boldsymbol{S}_2 = \boldsymbol{L}_2^{-1}$ and $\boldsymbol{S} = \boldsymbol{S}_1 \otimes \boldsymbol{S}_2 = \boldsymbol{L}^{-1}$. Let $\boldsymbol{E}_{ij}$ be the matrix with all 0s except for a 1 at position (i, j), its size being clear from context. We wish to solve

$$\nabla f_2(\boldsymbol{X}) = -\nabla g_2(\boldsymbol{S}_1) \quad \text{and } \nabla f_1(\boldsymbol{X}) = -\nabla g_1(\boldsymbol{S}_2). \tag{4.7}$$

It follows from the fact that $\log\det(\boldsymbol{S}_1 \otimes \boldsymbol{S}_2) = N_2 \log\det \boldsymbol{S}_1 + N_1 \log\det \boldsymbol{S}_2$ that $\nabla f_{\boldsymbol{S}_2}(\boldsymbol{X}) = N_2\boldsymbol{X}^{-1}$ and $\nabla f_{\boldsymbol{S}_1}(\boldsymbol{X}) = N_1\boldsymbol{X}^{-1}$. Moreover, we know that

$$\begin{aligned}
\nabla g(\boldsymbol{S}) &= -(\boldsymbol{I} + \boldsymbol{S})^{-1} - \boldsymbol{S}^{-1}\frac{1}{n}\sum\nolimits_i \boldsymbol{U}_i(\boldsymbol{U}_i^\top \boldsymbol{S}^{-1}\boldsymbol{U}_i)^{-1}\boldsymbol{U}_i\boldsymbol{S}^{-1} \\
&= -\boldsymbol{S}^{-1} - \boldsymbol{S}^{-1}\left(\frac{1}{n}\sum\nolimits_i \boldsymbol{U}_i(\boldsymbol{U}_i^\top \boldsymbol{S}^{-1}\boldsymbol{U}_i)^{-1}\boldsymbol{U}_i - (\boldsymbol{I} + \boldsymbol{S}^{-1})^{-1}\right)\boldsymbol{S}^{-1} \\
&= -(\boldsymbol{L} + \boldsymbol{L}\Delta\boldsymbol{L}).
\end{aligned}$$

The Jacobian of $\boldsymbol{S}_1 \to \boldsymbol{S}_1 \otimes \boldsymbol{S}_2$ is $\boldsymbol{J} = \Big(\operatorname{vect}(\boldsymbol{E}_{11} \otimes \boldsymbol{S}_2), \ldots, \operatorname{vect}(\boldsymbol{E}_{N_1 N_1} \otimes \boldsymbol{S}_2) \Big)$. Hence,

$$\begin{aligned}
\nabla f_1(\boldsymbol{X})_{ij} = -(\nabla g_1(\boldsymbol{S}_1))_{ij} &\iff N_2 \boldsymbol{X}_{ij}^{-1} = (\boldsymbol{J}^\top \operatorname{vect}(-\nabla g(\boldsymbol{S})))_{ij} \\
&\iff N_2 \boldsymbol{X}_{ij}^{-1} = \operatorname{vect}(\boldsymbol{E}_{ij} \otimes \boldsymbol{S}_2)^\top \operatorname{vect}(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}) \\
&\iff N_2 \boldsymbol{X}_{ij}^{-1} = \operatorname{Tr}((\boldsymbol{E}_{ij} \otimes \boldsymbol{S}_2)(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})) \\
&\iff N_2 \boldsymbol{X}_{ij}^{-1} = \operatorname{Tr}(\boldsymbol{S}_2(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})_{(ij)}) \\
&\iff N_2 \boldsymbol{X}_{ij}^{-1} = \operatorname{Tr}\left(((\boldsymbol{I} \otimes \boldsymbol{S}_2)(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}))_{(ij)}\right).
\end{aligned}$$

The last equivalence is obtained by manipulating indices. Thus, we have

$$\nabla f_2(\boldsymbol{X}) = -\nabla g_2(\boldsymbol{S}_1) \iff \boldsymbol{X}^{-1} = \frac{1}{N_2} \operatorname{Tr}_1 \left((\boldsymbol{I} \otimes \boldsymbol{S}_2)(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})\right).$$

Similarly, by setting $\boldsymbol{J}' = (\operatorname{vect}(\boldsymbol{S}_1 \otimes \boldsymbol{E}_{11}), \ldots, \operatorname{vect}(\boldsymbol{S}_1 \otimes \boldsymbol{E}_{N_1 N_1}))$, we have

$$\begin{aligned}
\nabla f_2(\boldsymbol{X})_{ij} = -(\nabla g_2(\boldsymbol{S}_2))_{ij} &\iff N_1 \boldsymbol{X}_{ij}^{-1} = (\boldsymbol{J}'^\top \operatorname{vect}(-\nabla g(\boldsymbol{S})))_{ij} \\
&\iff N_1 \boldsymbol{X}_{ij}^{-1} = \operatorname{vect}(\boldsymbol{S}_1 \otimes \boldsymbol{E}_{ij})^\top \operatorname{vect}(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}) \\
&\iff N_1 \boldsymbol{X}_{ij}^{-1} = \operatorname{Tr}((\boldsymbol{S}_1 \otimes \boldsymbol{E}_{ij})(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})) \\
&\iff N_1 \boldsymbol{X}_{ij}^{-1} = \Big(\textstyle\sum_{k,\ell=1}^{N_1} \boldsymbol{S}_{1k\ell} (\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})_{(\ell k)} \Big)_{ij} \\
&\iff N_1 \boldsymbol{X}_{ij}^{-1} = \Big(\textstyle\sum_{\ell=1}^{N_1} ((\boldsymbol{S}_1 \otimes \boldsymbol{I})(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}))_{(\ell\ell)} \Big)_{ij}.
\end{aligned}$$

Hence, we have the equivalence

$$\nabla f_{\boldsymbol{S}_1}(\boldsymbol{X}) = -\nabla g_{\boldsymbol{S}_1}(\boldsymbol{S}_2) \iff \boldsymbol{X}^{-1} = \frac{1}{N_1} \operatorname{Tr}_2 \left((\boldsymbol{S}_1 \otimes \boldsymbol{I})(\boldsymbol{L} + \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})\right),$$

which proves Prop. 4.2.3. □

We are now ready to establish that these updates ensure monotonic ascent in the log-likelihood.

Theorem 4.2.4 (Ascent). *Starting with* $\boldsymbol{L}_1^{(0)}, \boldsymbol{L}_2^{(0)} \in \mathbb{S}^{++}$, *updating according to* (4.6) *generates* $\boldsymbol{L}_1^{(k)}, \boldsymbol{L}_2^{(k)} \in \mathbb{S}^{++}$, *and the sequence* $\left\{\phi\big(\boldsymbol{L}_1^{(k)} \otimes \boldsymbol{L}_2^{(k)}\big)\right\}_{k \geq 0}$ *is non-decreasing.*

Proof. Updating according to (4.6) generates positive definite matrices $\boldsymbol{S}_i$, and hence positive definite subkernels $\boldsymbol{L}_i = \boldsymbol{S}_i$. Moreover, due to the convexity of g_1 and concavity of f_1, for matrices $\boldsymbol{A}, \boldsymbol{B} \succ 0$

$$f_1(\boldsymbol{B}) \leq f_1(\boldsymbol{A}) + \nabla f_1(\boldsymbol{A})^\top(\boldsymbol{B} - \boldsymbol{A}),$$
$$g_1(\boldsymbol{A}) \geq g_1(\boldsymbol{B}) + \nabla g_1(\boldsymbol{B})^\top(\boldsymbol{A} - \boldsymbol{B}).$$

Hence, $f_1(\boldsymbol{A}) + g_1(\boldsymbol{A}) \geq f_1(\boldsymbol{B}) + g_1(\boldsymbol{B}) + (\nabla f_1(\boldsymbol{A}) + \nabla g_1(\boldsymbol{B}))^\top(\boldsymbol{A} - \boldsymbol{B})$.

Thus, if $\boldsymbol{S}_1^{(k)}, \boldsymbol{S}_1^{(k+1)}$ verify (4.6), by setting $\boldsymbol{A} = \boldsymbol{S}_1^{(k+1)}$ and $\boldsymbol{B} = \boldsymbol{S}_1^{(k)}$ we have

$$\phi\big(\boldsymbol{L}_1^{(k+1)} \otimes \boldsymbol{L}_2^{(k)}\big) = f_1\big(\boldsymbol{S}_1^{(k+1)}\big) + g_1\big(\boldsymbol{S}_1^{(k+1)}\big) \geq f_1\big(\boldsymbol{S}_1^{(k)}\big) + g_1\big(\boldsymbol{S}_1^{(k)}\big) = \phi\big(\boldsymbol{L}_1^{(k)} \otimes \boldsymbol{L}_2^{(k)}\big).$$

The same reasoning holds for $\boldsymbol{L}_2$, which proves the theorem. □

As $\mathrm{Tr}_1\big((\boldsymbol{I} \otimes \boldsymbol{S}_2)\boldsymbol{L}\big) = N_2\boldsymbol{L}_1$ (and similarly for $\boldsymbol{L}_2$), update (4.6) is equivalent to

$$\boldsymbol{L}_1 \leftarrow \boldsymbol{L}_1 + \mathrm{Tr}_1\big((\boldsymbol{I} \otimes \boldsymbol{L}_2^{-1})(\boldsymbol{L}\Delta\boldsymbol{L})\big)/N_2, \qquad \boldsymbol{L}_2 \leftarrow \boldsymbol{L}_2 + \mathrm{Tr}_2\big((\boldsymbol{L}_1^{-1} \otimes \boldsymbol{I})(\boldsymbol{L}\Delta\boldsymbol{L})\big)/N_1.$$

The arguments above easily extend to the multiblock case: when learning $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \cdots \otimes \boldsymbol{L}_m$, we update $\boldsymbol{L}_k$ as

$$(\boldsymbol{L}_k)_{ij} \leftarrow (\boldsymbol{L}_k)_{ij} + \frac{N_k}{N_1 \cdots N_m} \mathrm{Tr}\left[(\boldsymbol{L}_1 \otimes \ldots \otimes \boldsymbol{L}_{k-1} \otimes \boldsymbol{E}_{ij} \otimes \boldsymbol{L}_{k+1} \otimes \ldots \otimes \boldsymbol{L}_m)(\boldsymbol{L}\Delta\boldsymbol{L})\right].$$

Generalization. As in (Mariet and Sra, 2015), we can generalize the updates to take an additional step-size parameter $a > 0$:

$$\boldsymbol{L}_1 \leftarrow \boldsymbol{L}_1 + a\,\mathrm{Tr}_1\big((\boldsymbol{I} \otimes \boldsymbol{L}_2^{-1})(\boldsymbol{L}\Delta\boldsymbol{L})\big)/N_2,$$
$$\boldsymbol{L}_2 \leftarrow \boldsymbol{L}_2 + a\,\mathrm{Tr}_2\big((\boldsymbol{L}_1^{-1} \otimes \boldsymbol{I})(\boldsymbol{L}\Delta\boldsymbol{L})\big)/N_1.$$

Experimentally, $a > 1$ (as long as the updates remain in $\mathbb{S}^+$) can provide faster convergence, although the monotonicity of the log-likelihood is no longer guaranteed.

From Theorem 4.2.4, we obtain Algorithm 7. However, from the above updates it is not obvious whether the Kronecker product saves us any computation. In particular,

it is not clear whether the updates can be implemented to run faster than $\mathcal{O}(N^3)$.

Algorithm 7 KrK-Picard iteration

Input: Matrices $\boldsymbol{L}_1, \boldsymbol{L}_2$, training set T, parameter a.
for $i = 1$ **to** maxIter **do**
  $\boldsymbol{L}_1 \leftarrow \boldsymbol{L}_1 + a \operatorname{Tr}_1\big((\boldsymbol{I} \otimes \boldsymbol{L}_2^{-1})(\boldsymbol{L}\Delta\boldsymbol{L})\big)/N_2$ // or update stochastically
  $\boldsymbol{L}_2 \leftarrow \boldsymbol{L}_2 + a \operatorname{Tr}_2\big((\boldsymbol{L}_1^{-1} \otimes \boldsymbol{I})(\boldsymbol{L}\Delta\boldsymbol{L})\big)/N_1$ // or update stochastically
return $(\boldsymbol{L}_1, \boldsymbol{L}_2)$

Theorem 4.2.5 (Complexity). *For $N_1 \approx N_2 \approx \sqrt{N}$, the updates in Algorithm 7 can be computed in $\mathcal{O}(n\kappa^3 + N^2)$ time and $\mathcal{O}(N^2)$ space, where κ is the size of the largest training subset. Furthermore, stochastic updates can be computed in $\mathcal{O}(N\kappa^2 + N^{3/2})$ time and $\mathcal{O}(N + \kappa^2)$ space.*

Proof. The updates to $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are obtained efficiently through different methods; hence, the proof to Thm. 4.2.5 is split into two sections. We write

$$\boldsymbol{\Theta} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{U}_i \boldsymbol{L}_{S_i}^{-1} \boldsymbol{U}_i^\top \quad (\text{or } \boldsymbol{\Theta} = \boldsymbol{U}_i \boldsymbol{L}_{S_i}^{-1} \boldsymbol{U}_i^\top \text{ for stochastic updates})$$

so that $\boldsymbol{\Delta} = \boldsymbol{\Theta} - (\boldsymbol{I} + \boldsymbol{L})^{-1}$. Recall that $(\boldsymbol{A} \otimes \boldsymbol{B})_{(ij)} = a_{ij}\boldsymbol{B}$.

Updating $\boldsymbol{L}_1$. We wish to compute $\boldsymbol{X} = \operatorname{Tr}_1\big((\boldsymbol{I} \otimes \boldsymbol{L}_2^{-1})(\boldsymbol{L}\Delta\boldsymbol{L})\big)$. We have

$$\begin{aligned}
\boldsymbol{X}_{ij} &= \operatorname{Tr}\left[((\boldsymbol{I} \otimes \boldsymbol{L}_2^{-1})(\boldsymbol{L}\Delta\boldsymbol{L}))_{(ij)}\right] \\
&= \operatorname{Tr}\left[\boldsymbol{L}_2^{-1}(\boldsymbol{L}\Delta\boldsymbol{L})_{(ij)}\right] \\
&= \operatorname{Tr}\left[\boldsymbol{L}_2^{-1}\sum\nolimits_{k,\ell=1}^{N_1} \boldsymbol{L}_{(ik)}\boldsymbol{\Delta}_{(k\ell)}\boldsymbol{L}_{(\ell j)}\right] \\
&= \sum\nolimits_{k,\ell=1}^{N_1} \boldsymbol{L}_{1ik}\boldsymbol{L}_{1\ell j} \operatorname{Tr}(\boldsymbol{L}_2^{-1}\boldsymbol{L}_2\boldsymbol{\Delta}_{(k\ell)}\boldsymbol{L}_2) \\
&= \sum\nolimits_{k,\ell=1}^{N_1} \boldsymbol{L}_{1ik}\boldsymbol{L}_{1\ell j} \underbrace{\operatorname{Tr}(\boldsymbol{\Theta}_{(k\ell)}\boldsymbol{L}_2)}_{\boldsymbol{A}_{k\ell}} - \underbrace{\operatorname{Tr}((\boldsymbol{I} + \boldsymbol{L})^{-1}_{(k\ell)}\boldsymbol{L}_2)}_{\boldsymbol{B}_{k\ell}} \\
&= (\boldsymbol{L}_1\boldsymbol{A}\boldsymbol{L}_1 - \boldsymbol{L}_1\boldsymbol{B}\boldsymbol{L}_1)_{ij}.
\end{aligned}$$

The $N_1 \times N_1$ matrix $\boldsymbol{A}$ can be computed in $\mathcal{O}(n\kappa^3 + N_1^2N_2^2)$ time simply by pre-computing $\boldsymbol{\Theta}$ in $\mathcal{O}(n\kappa^3)$ and then computing all N_1^2 traces in $\mathcal{O}(N_2^2)$ time. When

doing stochastic updates for which $\boldsymbol{\Theta}$ is sparse with only κ^2 non-zero coefficients, computing $\boldsymbol{A}$ can be done in $\mathcal{O}(N_1^2\kappa^2 + \kappa^3)$.

By diagonalizing $\boldsymbol{L}_1 = \boldsymbol{P}_1\boldsymbol{D}_1\boldsymbol{P}_1^\top$ and $\boldsymbol{L}_2 = \boldsymbol{P}_2\boldsymbol{D}_2\boldsymbol{P}_2^\top$, we have $(\boldsymbol{I}+\boldsymbol{L})^{-1} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top$ with $\boldsymbol{P} = \boldsymbol{P}_1 \otimes \boldsymbol{P}_2$ and $\boldsymbol{D} = (\boldsymbol{I} + \boldsymbol{D}_1 \otimes \boldsymbol{D}_2)^{-1}$. $\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{D}_1, \boldsymbol{D}_2$ and $\boldsymbol{D}$ can all be obtained in $\mathcal{O}(N_1^3 + N_2^3 + N_1N_2)$ as a consequence of Prop. 4.2.1. Then

$$\begin{aligned}
\boldsymbol{B}_{ij} &= \operatorname{Tr}((\boldsymbol{I} + \boldsymbol{L})^{-1}_{(ij)}\boldsymbol{L}_2) \\
&= \sum_k \operatorname{Tr}(\boldsymbol{P}_{(ik)}\boldsymbol{D}_{(kk)}\boldsymbol{P}^\top_{(kj)}\boldsymbol{L}_2) \\
&= \sum_k \boldsymbol{P}_{1ik}\boldsymbol{P}_{1jk} \operatorname{Tr}(\boldsymbol{P}_2\boldsymbol{D}_{(kk)}\boldsymbol{P}_2^\top\boldsymbol{P}_2\boldsymbol{D}_2\boldsymbol{P}_2^\top) \\
&= \sum_k \boldsymbol{P}_{1ik}\boldsymbol{P}_{1jk} \underbrace{\operatorname{Tr}(\boldsymbol{D}_{(kk)}\boldsymbol{D}_2)}_{\alpha_k}.
\end{aligned}$$

Let $\widehat{\boldsymbol{D}} = \operatorname{diag}(\alpha_1, \ldots, \alpha_{N_1})$, which can be computed in $\mathcal{O}(N_1N_2)$. Then $\boldsymbol{L}_1\boldsymbol{B}\boldsymbol{L}_1 = \boldsymbol{P}_1\boldsymbol{D}_1\widehat{\boldsymbol{D}}\boldsymbol{D}_1\boldsymbol{P}_1$ is computable in $\mathcal{O}(N_1^3 + N_2^3)$.

Overall, the update to $\boldsymbol{L}_1$ can be computed in $\mathcal{O}(n\kappa^3 + N_1^2N_2^2 + N_1^3 + N_2^3)$ time, or in $\mathcal{O}(N_1^2\kappa^2 + \kappa^3 + N_1^3 + N_2^3)$ time if the updates are stochastic. Moreover, if $\boldsymbol{\Theta}$ is sparse with only z non-zero coefficients (for stochastic updates $z = \kappa$), $\boldsymbol{A}$ can be computed in $\mathcal{O}(\kappa^2)$ space, leading to an overall $\mathcal{O}(z^2 + N_1^2 + N_2^2)$ memory cost.

Updating $\boldsymbol{L}_2$. We wish to compute $\boldsymbol{X} = \operatorname{Tr}_2\big[(\boldsymbol{L}_1^{-1} \otimes \boldsymbol{I})(\boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})\big]$ efficiently.

$$\begin{aligned}
\boldsymbol{X} &= \sum\nolimits_{i=1}^{N_1} \big((\boldsymbol{L}_1^{-1} \otimes \boldsymbol{I})(\boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L})\big)_{(ii)} \\
&= \sum\nolimits_{i=1}^{N_1} \big((\boldsymbol{I} \otimes \boldsymbol{L}_2)(\boldsymbol{\Theta} - (\boldsymbol{I} + \boldsymbol{L})^{-1})(\boldsymbol{L}_1 \otimes \boldsymbol{L}_2)\big)_{(ii)} \\
&= \sum\nolimits_{i,j=1}^{N_1} \boldsymbol{L}_{1ij}\boldsymbol{L}_2\boldsymbol{\Theta}_{(ij)}\boldsymbol{L}_2 - \sum_{i=1}^{N_1}((\boldsymbol{I} \otimes \boldsymbol{L}_2)(\boldsymbol{I} + \boldsymbol{L})^{-1}(\boldsymbol{L}_1 \otimes \boldsymbol{L}_2))_{(ii)} \\
&= \underbrace{\boldsymbol{L}_2 \sum\nolimits_{i,j=1}^{N_1} \boldsymbol{L}_{1ij}\boldsymbol{\Theta}_{(ij)}\boldsymbol{L}_2}_{A} - \underbrace{\sum\nolimits_{i=1}^{N_1}((\boldsymbol{I} \otimes \boldsymbol{L}_2)(\boldsymbol{I} + \boldsymbol{L})^{-1}(\boldsymbol{L}_1 \otimes \boldsymbol{L}_2))_{(ii)}}_{B}.
\end{aligned}$$

Matrix $\boldsymbol{A}$ can be computed in $\mathcal{O}(n\kappa^3 + N_1^2N_2^2 + N_2^3)$ time. For stochastic updates, we only require $\mathcal{O}(N_1^2\kappa^2 + \kappa^3 + N_2^3)$ time and $\mathcal{O}(N_2^2 + N_1^2 + \kappa^2)$ space due to $\boldsymbol{\Theta}$'s sparsity.

Regarding $\boldsymbol{B}$, as all matrices commute, we can write

$$(\boldsymbol{I} \otimes \boldsymbol{L}_2)(\boldsymbol{I} + \boldsymbol{L})^{-1}(\boldsymbol{L}_1 \otimes \boldsymbol{L}_2) = (\boldsymbol{P}_1 \otimes \boldsymbol{P}_2)\boldsymbol{\Lambda}(\boldsymbol{P}_1 \otimes \boldsymbol{P}_2),$$

where $\boldsymbol{\Lambda} = (\boldsymbol{I} \otimes \boldsymbol{D}_2)(\boldsymbol{I} + \boldsymbol{D}_1 \otimes \boldsymbol{D}_2)^{-1}(\boldsymbol{D}_1 \otimes \boldsymbol{D}_2)$ is diagonal and is obtained in $\mathcal{O}(N_1^3 + N_2^3 + N_1 N_2)$. Moreover,

$$\boldsymbol{B} = \sum\nolimits_{i=1}^{N_1} (\boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top)_{(ii)} = \boldsymbol{P}_2 \left(\sum\nolimits_{i,k=1}^{N_1} \boldsymbol{P}_{1ik} \boldsymbol{\Lambda}_{(kk)} \boldsymbol{P}_{1ik} \right) \boldsymbol{P}_2^\top,$$

which allows us to compute $\boldsymbol{B}$ in $\mathcal{O}(N_1^2 N_2 + N_2^3 + N_1^3)$ total.

Overall, we can obtain $\boldsymbol{X}$ in $\mathcal{O}(n\kappa^3 + N_1^2 N_2^2 + N_1^3 + N_2^3)$ or in $\mathcal{O}(N_1^2\kappa^2 + N_1^2 N_2 + N_1^3 + N_2^3)$ for stochastic updates, in which case only $\mathcal{O}(N_1^2 + N_2^2 + \kappa^2)$ space is necessary. □

The proof shows that by leveraging the properties of the Kronecker product, the maximum likelihood updates can be obtained without computing $\boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}$. This result is non-trivial: the components of $\boldsymbol{\Delta}$, $\frac{1}{n}\sum_i \boldsymbol{U}_i \boldsymbol{L}_{S_i}^{-1} \boldsymbol{U}_i^\top$ and $(\boldsymbol{I} + \boldsymbol{L})^{-1}$, must be considered separately for computational efficiency.

The runtime of Alg. 7 is a marked improvement over (Mariet and Sra, 2015), which runs in $\mathcal{O}(N^2)$ space and $\mathcal{O}(n\kappa^3 + N^3)$ time in the non-stochastic case, and in $\mathcal{O}(N^3)$ time for stochastic updates.

4.2.2 Learning with joint updates

An alternate possibility lies in updating $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ jointly via the following straightforward procedure: we first update $\boldsymbol{L} \leftarrow \boldsymbol{L} + \boldsymbol{L}\Delta\boldsymbol{L}$, then recover the Kronecker structure of the kernel by defining the updates $\boldsymbol{L}_1'$ and $\boldsymbol{L}_2'$ such that:

$$\begin{cases} (\boldsymbol{L}_1', \boldsymbol{L}_2') \text{ minimizes } \|\boldsymbol{L} + \boldsymbol{L}\Delta\boldsymbol{L} - \boldsymbol{L}_1' \otimes \boldsymbol{L}_2'\|_F^2 \\ \boldsymbol{L}_1' \succ 0, \boldsymbol{L}_2' \succ 0, \|\boldsymbol{L}_1'\| = \|\boldsymbol{L}_2'\|. \end{cases} \tag{4.8}$$

Theorem 4.2.6. *Solutions to* (4.8) *exist and can be computed from the first singular value and vectors of the matrix* $\left[\text{vect}((\boldsymbol{L}^{-1} + \boldsymbol{\Delta})_{(ij)})^\top\right]_{i,j=1}^{N_1}$.

Proof. In order to minimize the number of matrix multiplications, we equivalently (due to the properties of the Frobenius norm) find

$$\boldsymbol{X}, \boldsymbol{Y} \in \underset{\boldsymbol{X},\boldsymbol{Y}\succeq 0}{\operatorname{argmin}} \|\boldsymbol{L}^{-1} + \boldsymbol{\Delta} - \boldsymbol{X} \otimes \boldsymbol{Y}\|_F^2, \tag{4.9}$$

then let $\boldsymbol{L}_1' \leftarrow \boldsymbol{L}_1 \boldsymbol{X} \boldsymbol{L}_1$ and $\boldsymbol{L}_2' \leftarrow \boldsymbol{L}_2 \boldsymbol{Y} \boldsymbol{L}_2$. We will first require the following lemma.

Lemma 4.2.7. *Let* $\boldsymbol{L} \succ 0$. *Define* $\boldsymbol{R} := [\operatorname{vect}(\boldsymbol{L}_{(11)})^\top; \ldots; \operatorname{vect}(\boldsymbol{L}_{(N_1 N_1)})^\top]_{i,j=1}^{N_1} \in \mathbb{R}^{N_1 N_1 \times N_2 N_2}$. *Suppose that* $\boldsymbol{R}$ *has an eigengap between its largest singular value and the next, and let* u, v, σ *be the first singular vectors and value of* $\boldsymbol{R}$. *Let* $\boldsymbol{U} = \operatorname{mat}(u)$ *and* $\boldsymbol{V} = \operatorname{mat}(v)$. *Then* $\boldsymbol{U}$ *and* $\boldsymbol{V}$ *are either both positive definite or negative definite.*

Moreover, for any value $\alpha \neq 0$, *the pair* $(\alpha \boldsymbol{U}, \sigma/\alpha \boldsymbol{V})$ *minimizes* $\|\boldsymbol{L} - X \otimes Y\|_F^2$.

Lemma 4.2.7 is a consequence of (Loan and Pitsianis, 1993, Thm. 11), and shows that if $\boldsymbol{L}$ is initially positive definite, setting the sign of α based on whether $\boldsymbol{U}$ and $\boldsymbol{V}$ are positive or negative definite,[2] and updating

$$\begin{cases} \boldsymbol{L}_1 \leftarrow \alpha\, \boldsymbol{L}_1 \boldsymbol{U} \boldsymbol{L}_1 \\ \boldsymbol{L}_2 \leftarrow \sigma/\alpha\, \boldsymbol{L}_2 \boldsymbol{V} \boldsymbol{L}_2 \end{cases}$$

will maintain positive definite iterates. Given that if $\boldsymbol{L}_1 \succ 0$ and $\boldsymbol{L}_2 \succ 0$, we also have $\boldsymbol{L}_1 \otimes \boldsymbol{L}_2 \succ 0$, a simple induction then shows that by choosing an initial kernel estimate $\boldsymbol{L} \succ 0$, subsequent values of $\boldsymbol{L}$ will remain positive definite. Hence, by choosing α such that the new estimates $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ verify $\|\boldsymbol{L}_1\| = \|\boldsymbol{L}_2\|$, we verify all the conditions of Eq. 4.8. □

Theorem 4.2.7 leads to Algorithm 8: a straightforward iteration for learning matrices $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ based on the decomposition of the Picard estimate $\boldsymbol{L} + \boldsymbol{L}\nabla\phi(\boldsymbol{L})\boldsymbol{L}$ as a Kronecker product.

[2] This can easily be done simply by checking the sign of the first diagonal coefficient of $\boldsymbol{U}$, which will be positive if and only if $\boldsymbol{U} \succ 0$.

Algorithm 8 JOINT-PICARD iteration

Input: Matrices $\boldsymbol{L}_1, \boldsymbol{L}_2$, training set T, step-size $a \geq 1$.

for $i = 1$ **to** maxIter **do**

$\sigma, \boldsymbol{U}, \boldsymbol{V} \leftarrow \texttt{power\_method}(\boldsymbol{L}^{-1} + \Delta)$ to obtain the first singular value + vectors.

$\alpha \leftarrow \operatorname{sgn}(\boldsymbol{U}_{11})\sqrt{\sigma\|\boldsymbol{L}_2 V \boldsymbol{L}_2\| / \|\boldsymbol{L}_1 \boldsymbol{U} \boldsymbol{L}_1\|}$

$\boldsymbol{L}_1 \leftarrow \boldsymbol{L}_1 + a(\alpha\, \boldsymbol{L}_1 \boldsymbol{U} \boldsymbol{L}_1 - \boldsymbol{L}_1)$

$\boldsymbol{L}_2 \leftarrow \boldsymbol{L}_2 + a(\sigma/\alpha\, \boldsymbol{L}_2 V \boldsymbol{L}_2)$

return $(\boldsymbol{L}_1, \boldsymbol{L}_2)$

However, Alg. 8 produces no guaranteed increase in log-likelihood. An analysis similar that of Thm. 4.2.5 shows that updates can be obtained $\mathcal{O}\big(n\kappa^3 + \max(N_1, N_2)^4\big)$.

4.2.3 The knapsack problem for kernel learning

Although KRONDPPS have tractable learning algorithms, the memory requirements remain high for non-stochastic updates, as the matrix $\boldsymbol{\Theta} = \frac{1}{n}\sum_i \boldsymbol{U}_i \boldsymbol{L}_{S_i}^{-1} \boldsymbol{U}_i^\top$ needs to be stored, requiring $\mathcal{O}(N^2)$ memory.

However, if the training set $\{S_1, \ldots, S_n\}$ can be partitioned as $P_1 \cup \ldots \cup P_k$ with

$$\{S_1, \ldots, S_n\} = \cup_{k=1}^m P_k \quad s.t. \quad \forall k, |\cup_{S \in P_k} S| < z, \tag{4.10}$$

$\boldsymbol{\Theta}$ can be decomposed as $\frac{1}{n}\sum_{k=1}^m \boldsymbol{\Theta}_k$ with $\boldsymbol{\Theta}_k = \sum_{S_i \in P_k} \boldsymbol{U}_i \boldsymbol{L}_{S_i}^{-1} \boldsymbol{U}_i^\top$. Due to the bound in Eq. 4.10, each $\boldsymbol{\Theta}_k$ will be sparse, with only z^2 non-zero coefficients. We can then store each $\boldsymbol{\Theta}_k$ with minimal storage and update $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ in $\mathcal{O}(n\kappa^3 + mz^2 + N^{3/2})$ time and $\mathcal{O}(mz^2 + N)$ space.

Determining the existence of such a partition of size m is a variant of the NP-Hard Subset-Union Knapsack Problem (SUKP) (Goldschmidt et al., 1994) with m knapsacks and where the value of each item (*i.e.*, each S_i) is equal to 1: a solution to SUKP of value n with m knapsacks is equivalent to a solution to Eq. 4.10. However, an approximate partition can also be simply constructed via a greedy algorithm.

4.2.4 Sampling

Sampling exactly (Alg. 3) from a full DPP kernel costs $\mathcal{O}(N^3 + Nk^3)$ where k is the size of the sampled subset, with the bulk of the computation lying in the initial eigendecomposition of $\boldsymbol{L}$; the k orthonormalizations cost $\mathcal{O}(Nk^3)$.

It follows from Prop. 4.2.1.1 that for KRONDPPS, the eigenvalues λ_i can be obtained in $\mathcal{O}(N_1^3 + N_2^3)$, and the k eigenvectors in $\mathcal{O}(kN)$ operations. For $N_1 \approx N_2 \approx \sqrt{N}$, exact sampling thus only costs $\mathcal{O}(N^{3/2} + Nk^3)$. If $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \boldsymbol{L}_2 \otimes \boldsymbol{L}_3$, the same reasoning shows that exact sampling is linear in N, only requiring $\mathcal{O}(Nk^3)$ operations.

4.2.5 Experimental results

In order to validate our learning algorithm, we compare KRK-PICARD to JOINT-PICARD and to the Picard iteration (PICARD) on real and synthetic datasets.[3]

Synthetic tests

To enable a fair comparison between algorithms, we test them on synthetic data drawn from a full (non-Kronecker) ground-truth DPP kernel. The sub-kernels were initialized by $\boldsymbol{L}_i = \boldsymbol{X}^\top \boldsymbol{X}$, with $\boldsymbol{X}$'s coefficients drawn uniformly from $[0, \sqrt{2}]$; for PICARD, $\boldsymbol{L}$ was initialized with $\boldsymbol{L}_1 \otimes \boldsymbol{L}_2$.

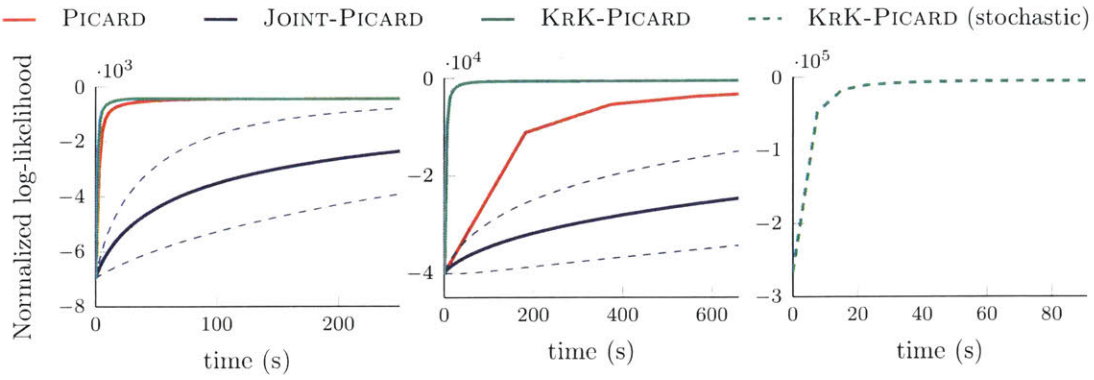
For Figures 4-1a and 4-1b, training data was generated by sampling 100 subsets from the true kernel with sizes uniformly distributed between 10 and 190.

To evaluate KRK-PICARD on matrices too large to fit in memory and with large κ, we drew samples from a $50 \cdot 10^3 \times 50 \cdot 10^3$ kernel of rank $1,000$ (on average $|S_i| \approx 1,000$), and learned the kernel stochastically (only KRK-PICARD could be run due to the memory requirements of other methods); the likelihood drastically improves in only two steps (Fig.4-1c).

As shown in Figures 4-1a and 4-1b, KRK-PICARD converges significantly faster than PICARD, especially for large values of N. However, although JOINT-PICARD also increases the log-likelihood at each iteration, it converges much slower and has a high

[3]Experiments were repeated 5 times and averaged, using MATLAB on a Linux Mint system with 16GB of RAM and an i7-4710HQ CPU @ 2.50GHz.

standard deviation, whereas the standard deviations for PICARD and KRK-PICARD are barely noticeable. For these reasons, we drop the comparison to JOINT-PICARD in the subsequent experiments.



(a) $N_1 = N_2 = 50$ (b) $N_1 = N_2 = 100$ (c) $N_1 = 100, N_2 = 500$

Figure 4-1: $a = 1$; the dotted lines indicated the standard deviation from the mean.

Small-scale real data: baby registries

We compared KRK-PICARD to PICARD and EM (Gillenwater et al., 2014) on the baby registry dataset, which has also been used to evaluate other DPP learning algorithms (Gillenwater et al., 2014; Mariet and Sra, 2015; Gartrell et al., 2017). The dataset contains 17 categories of baby-related products obtained from Amazon (see Appendix A.2.2 for a description of the dataset).

We learned kernels for the 6 largest categories ($N = 100$); in this case, PICARD is sufficiently efficient to be preferred to KRK-PICARD; this comparison serves only to evaluate the quality of the final kernel estimates.

The initial marginal kernel $\boldsymbol{K}$ for EM was sampled from a Wishart distribution with N degrees of freedom and an identity covariance matrix, then scaled by $1/N$; for PICARD, $\boldsymbol{L}$ was set to $\boldsymbol{K}(\boldsymbol{I} - \boldsymbol{K})^{-1}$; for KRK-PICARD, $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ were chosen (as in JOINT-PICARD) by minimizing $\|\boldsymbol{L} - \boldsymbol{L}_1 \otimes \boldsymbol{L}_2\|$. Convergence was determined when the objective change dipped below a threshold δ. As one EM iteration takes longer than one Picard iteration but increases the likelihood more, we set separate thresholds

for the different algorithms; specifically, $\delta_{\text{PIC}} = \delta_{\text{KRK}} = 10^{-4}$ and $\delta_{\text{EM}} = 10^{-5}$.

The final log-likelihoods are shown in Table 4.2; we set the step-sizes to their largest possible values, *i.e.*, $a_{\text{PIC}} = 1.3$ and $a_{\text{KRK}} = 1.8$. Table 4.2 shows that KRK-PICARD obtains comparable, albeit slightly worse log-likelihoods than PICARD and EM, which confirms that for tractable N, the better modeling capability of full kernels make them preferable to KRONDPPS.

Table 4.2: Final log-likelihoods for each large category of the baby registries dataset

(a) Training set

Category	EM	PICARD	KRK-PICARD
apparel	-10.1	-10.2	-10.7
bath	-8.6	-8.8	-9.1
bedding	-8.7	-8.8	-9.3
diaper	-10.5	-10.7	-11.1
feeding	-12.1	-12.1	-12.5
gear	-9.3	-9.3	-9.6

(b) Test set

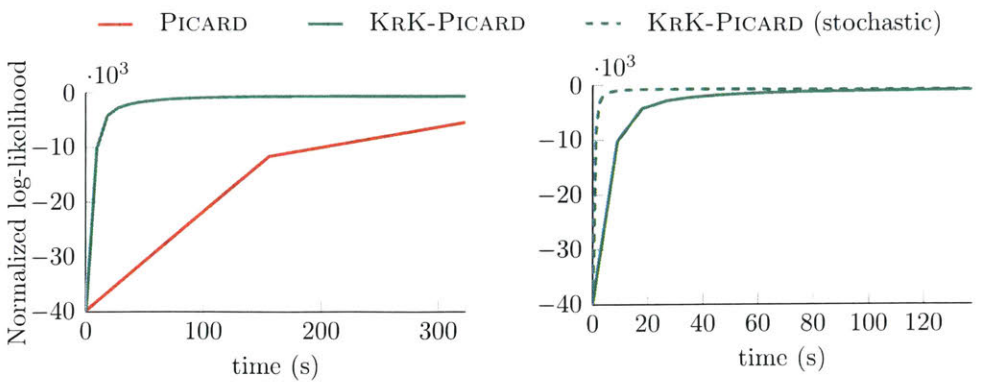
Category	EM	PICARD	KRK-PICARD
apparel	-10.1	-10.2	-10.7
bath	-8.6	-8.8	-9.1
bedding	-8.7	-8.8	-9.3
diaper	-10.6	-10.7	-11.2
feeding	-12.2	-12.2	-12.6
gear	-9.2	-9.2	-9.5

Large-scale real dataset: GENES

Finally, to evaluate KRK-PICARD on large matrices of real-world data, we train it on data from the GENES (Batmanghelich et al., 2014) dataset (which has also been used to evaluate DPPs in (Li et al., 2015; Batmanghelich et al., 2014)). This dataset consists in 10,000 genes, each represented by 331 features corresponding to the distance of a gene to hubs in the BioGRID gene interaction network.

We construct a ground truth Gaussian DPP kernel on the GENES dataset and use it to obtain 100 training samples with sizes uniformly distributed between 50 and 200 items. Similarly to the synthetic experiments, we initialized KRK-PICARD's kernel by setting $\boldsymbol{L}_i = \boldsymbol{X}_i^\top \boldsymbol{X}_i$ where $\boldsymbol{X}_i$ is a random matrix of size $N_1 \times N_1$; for PICARD, we set the initial kernel $\boldsymbol{L} = \boldsymbol{L}_1 \otimes \boldsymbol{L}_2$.

Figure 4-2 shows the performance of both algorithms. As with the synthetic experiments, KRK-PICARD converges much faster; stochastic updates increase its performance even more, as shown in Fig. 4-2b. Average runtimes and speed-up are given in Table 4.3: KRK-PICARD runs almost an order of magnitude faster than



(a) Non-stochastic learning

(b) Stochastic vs. non-stochastic

Figure 4-2: $n = 150$, $a = 1$.

Table 4.3: Average runtime, performance on the GENES dataset for $N_1 = N_2 = 100$

	PICARD	KRK-PICARD	KRK-PICARD (stoch.)
Average runtime	161.5 ± 17.7 s	8.9 ± 0.2 s	1.2 ± 0.02 s
NLL increase (1st iter.)	$(2.81 \pm 0.03) \cdot 10^4$	$(2.96 \pm 0.02) \cdot 10^4$	$(3.13 \pm 0.04) \cdot 10^4$

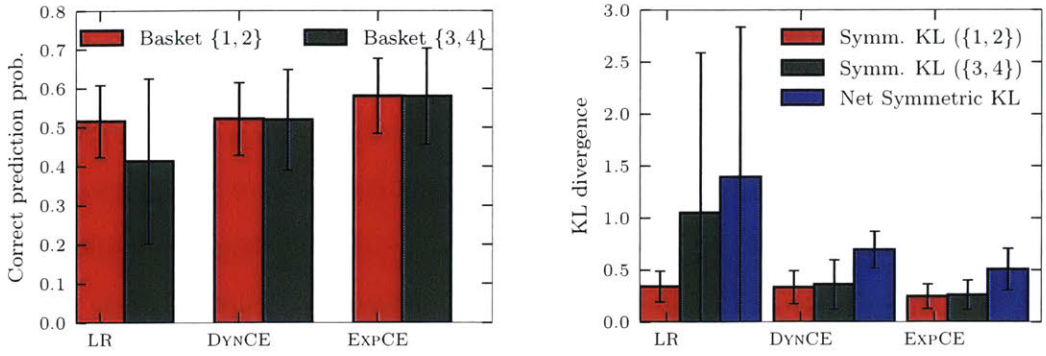
PICARD, and stochastic updates are more than two orders of magnitude faster, while providing slightly larger initial increases to the log-likelihood.

KRONDPPS also enable fast sampling, in $\mathcal{O}(N^{3/2} + Nk^3)$ operations when using two sub-kernels, and in $\mathcal{O}(Nk^3)$ when using three sub-kernels. This speedup allows for exact sampling at comparable or even better costs than previous algorithms for approximate sampling.

4.3 Learning with corrective negative information

We next consider the learning a (potentially unstructured) DPP kernel when information regarding "bad" subsets is available. Indeed, although fitting observed subsets well, maximum likelihood estimation for DPPs may also assign high likelihoods to unobserved subsets far from the underlying generative distribution (Chao et al., 2015). MLE-based DPP models may thus have modes corresponding to subsets that are close in likelihood, yet differ in how close they are to the true data distribution.

Such confusable modes reduce the quality of the learned model, hurting predic-



(a) Probability of making right prediction

(b) KL divergences

Figure 4-3: Results for experiments on a synthetic toy dataset. This toy dataset was generated by replicating the baskets {1, 2} and {3, 4} 1000 times each. We randomly select 80% of this dataset for training, and 20% for test. We train each model to convergence, and then compute the next-item predictive probabilities for each unique pair, along with the symmetric KL divergence (only over areas of shared support) between the predictive and empirical next-item distributions. Net symmetric KL divergence is computed by adding the symmetric KL divergences for each of the two unique baskets. Experiments were run 10 times, with $|\mathcal{A}^+|/|\mathcal{A}^-|$ set to the optimal value for each model; α is set to its optimal LR value. See Section Section 4.3.1 for details on the DYN and EXP negative sampling models. We see that the LR (low-rank DPP) model can assign relatively high predictive probabilities to modes that represent incorrect predictions, resulting in higher symmetric KL divergence compared to the true empirical distribution and a high variance, revealing a high sensitivity to initalization. The DYN and EXP methods we introduce in Section Section 4.3.1 reduce this confusable mode issue, resulting in predictive distributions that are closer to the true distribution and much smaller variances.

tions: this is the scenario illustrated in Figure 4-3.

Such concerns when learning generative models over huge sample spaces are not limited to the area of subset-selection: applications in image and text generation have been the driving force in developing techniques for generating high-quality samples. Among their innovations, a particularly successful technique uses generated samples as "negative samples" to train a discriminator, which in turn encourages generation of more realistic samples; this is the key idea behind the Generative Adversarial Nets (GANs) introduced in (Goodfellow et al., 2014).

These observations motivate us to investigate the use of DPP-generated samples with added perturbations as *negatives*, which we incorporate into the learning task to improve the modeling power of DPPs. Intuitively, negative samples are subsets that are far from the true data distribution, but to which the learned DPP model erro-

neously assigns high probability. As there is no closed form solution to generating such idealized negatives, we approximate them via an external "negative distribution".

Specifically, we introduce a DPP learning problem that incorporates samples from a negative distribution into traditional MLE. While the focus of this section lies on generating the negative distribution *jointly* with $\boldsymbol{L}$, we also investigate outside sources of negative information. Ultimately, our formulation leads to an optimization problem harder than the original DPP learning problem. However, we show that even approximate solutions greatly improve the performance of the DPP model when evaluated on concrete tasks: identifying the best item to add to a subset of chosen objects (*basket-completion*) and discriminating between held-out test data and randomly generated subsets. The following material is based on (Mariet et al., 2019a).

Related work

Aside from Tschiatschek et al. (2016); Djolonga et al. (2016), who learn a Facility LocatIon Diversity (FLID) distribution (as well as more complex FLIC and FLDC models) by contrasting it with a "negative" product distribution, little attention has been given to using negative samples to learn richer subset-selection models.

Nonetheless, leveraging negative information is a widely used in other applications. In object detection, negative mining corrects for the skewed simple-to-difficult negative distribution by training the model on its false positives (Sung, 1996; Canvet and Fleuret, 2014; Shrivastava et al., 2016). In language modeling, Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012), which tasks the model with distinguishing positive samples from generated negatives, was first applied in (Mnih and Teh, 2012) and has been instrumental in Word2Vec (Mikolov et al., 2013). Since then, variants using adaptive noise (Chen et al., 2017) have been introduced. NCE is also the method used by Tschiatschek et al. (2016) for subset-selection.

An alternate approach to negative samples within submodular language models was introduced as Contrastive Estimation in (Smith and Eisner, 2005a,b). Negative sampling is also used in GANs (Goodfellow et al., 2014), where a generator network competes with a discriminative network that distinguishes between positives and gen-

erated negatives. An adversarial approach to Contrastive Estimation has been recently introduced in (Bose et al., 2018), where ideas from GANs for discrete data are used to implement an adversarial negative sampler that augments a conventional negative sampler.

Motivated by the similarities between DPP learning and crucial structured prediction problems in other ML fields, we introduce an optimization problem that leverages negative information. We refer to this problem as Contrastive Estimation (CE) due to its ties to a notion discussed in (Smith and Eisner, 2005a).

4.3.1 Contrastive Estimation

Recall that in conventional DPP learning, we seek to maximize determinantal volumes of sets drawn *i.i.d.* from a source distribution μ (that we wish to model) by solving

$$\text{Find } \boldsymbol{L} \in \operatorname*{argmax}_{\boldsymbol{L} \succeq 0} \phi_{\text{MLE}}(\boldsymbol{L}) \triangleq \tfrac{1}{|\mathcal{A}^+|} \sum_{A \in \mathcal{A}^+} \log\det(\boldsymbol{L}_A) - \log\det(\boldsymbol{L} + \boldsymbol{I}). \tag{4.11}$$

To leverage information about *unlikely* samples, we augment problem (4.11) to incorporate additional information from a *negative* distribution ν, which we wish to have the DPP distribution move away from.

Definition 4.3.1 (Contrastive Estimation)**.** Given a training set of positive samples $\mathcal{A}^+$ on which ϕ_{MLE} is defined and a negative distribution ν over $2^{\mathcal{Y}}$, we call *Contrastive Estimation* the problem

$$\text{Find } \boldsymbol{L} \in \operatorname*{argmax}_{\boldsymbol{L} \succeq 0} \phi_{\text{CE}}(\boldsymbol{L}) \triangleq \ \phi_{\text{MLE}}(\boldsymbol{L}) - \mathbb{E}_{A \sim \nu}[\log \mathcal{P}_{\boldsymbol{L}}(A)], \tag{4.12}$$

where we write $\mathcal{P}_{\boldsymbol{L}}(A) \equiv \det(\boldsymbol{L}_A)/\det(\boldsymbol{L} + \boldsymbol{I})$.

The expectation in Eq. 4.12 is usually intractable. However, it can be approxi-

mated by sampling a set of samples $\mathcal{A}^-$ from the distribution ν: ϕ_{CE} then becomes[4]

$$\phi_{\text{CE}}(\boldsymbol{L}) = \frac{1}{|\mathcal{A}^+|} \sum_{A \in \mathcal{A}^+} \log \mathcal{P}_{\boldsymbol{L}}(A) - \frac{1}{|\mathcal{A}^-|} \sum_{A \in \mathcal{A}^-} \log \mathcal{P}_{\boldsymbol{L}}(A). \tag{4.13}$$

If $|\mathcal{A}^-| = 0$, the CE objective (4.12) reduces to ϕ_{MLE}. Conversely, ϕ_{MLE} can be viewed as a sample-based approximation of the value $\mathbb{E}_{A \sim \mu}[\log \mathcal{P}_{\boldsymbol{L}}(A)]$, where μ is the true distribution generating the samples in $\mathcal{A}^+$. Interestingly, another reformulation of (4.12) suggests an even broader class of DPP kernel learning: indeed, let y_A be $\frac{1}{|\mathcal{A}^+|}$ (resp. $-\frac{1}{|\mathcal{A}^-|}$) for $A \in \mathcal{A}^+$ (resp. $\mathcal{A}^-$), and define

$$\mathcal{A} = \{(y_A, A) : A \in \mathcal{A}^+\} \cup \{(y_A, A) : A \in \mathcal{A}^-\},$$

where the y_A should be viewed as belonging in $\{-1, 1\}$ with an additional normalization coefficient. Then, we can rewrite equation (4.13) in the following form

$$\phi_{\text{CE}}(\boldsymbol{L}) = \sum_{(y_A, A) \in \mathcal{A}} y_A \Big[\log \det \boldsymbol{L}_A - \det(\boldsymbol{L} + \boldsymbol{I}) \Big]. \tag{4.14}$$

Formulation (4.14) suggests the use of a broader scope of continuous labels y_A; we do not cover this variation in the present work, but note that (4.14) permits the use of *weighted* samples for learning.

Remark 4.3.1. Compared to the traditional Noise Contrastive Estimation (NCE) approach, which requires full knowledge of the negative distribution, CE does not suffer any such limitation: we only require an estimate of $\mathbb{E}_{\nu}[\log \mathcal{P}_{\boldsymbol{L}}(A)]$.

Remark 4.3.2. Eq. (4.12) can be made to go to $+\infty$ with pathological negative samples (*i.e.*, $\mathcal{P}_{\boldsymbol{L}}(A^-) = 0$); hence, choosing the negative distribution is a crucial concern for CE. In practice, we do not observe this pathological behavior (cf. Section 4.3.3).

Remark 4.3.3. CE is a non-convex optimization problem, and thus admits the same guarantees as DPP MLE learning when learned using Stochastic Gradient Ascent with decreasing step sizes; however, the convergence rate will depend on the choice of ν.

[4]With a slight abuse of notation, we write ϕ_{CE} despite the sample approximation to $\mathbb{E}_{A \sim \nu}[\cdot]$.

Indeed, to fully specify the CE problem one must first choose the negative distribution ν, or equivalently, choose a procedure to generate negative samples to obtain (4.13). We consider two classes of distributions ν with considerably different ramifications: dynamic and static negatives; their analysis is the focus of the next sections.

Dynamic negatives

In most applications leveraging negative information (e.g., negative mining, GANs), negative samples evolve over time based on the state of the learned model. We call any ν that depends on the state of the model a *dynamic negative distribution*: at iteration k of the learning procedure with kernel estimate $\boldsymbol{L}_k$, we use a ν parametrized by $\boldsymbol{L}_k$.

More specifically, we focus on the setting where negative samples themselves are generated by the current DPP, with the goal of reducing overfitting. Given a positive sample A^+, we generate a negative A^- by replacing $i \in A^+$ with j that yields a high probability $\mathcal{P}_{\boldsymbol{L}_k}(A^+ \setminus \{i\} \cup \{j\})$ (Alg. 9). We generate the samples probabilistically rather than via mode maximization so that a sample A^+ can lead to different A^- negatives when we generate more negatives than positives.

Algorithm 9 Generate dynamic negative

Input: Positive sample A^+, current kernel $\boldsymbol{L}_k$
Sample $i \in A^+$ prop. to its empirical probability in $\mathcal{A}^+$
$A^- := A^+ \setminus \{i\}$
Sample j w.p. proportional to $\mathcal{P}_{\boldsymbol{L}_k}(A^- \cup \{j\})$
$A^- \leftarrow A^- \cup \{j\}$
return A^-

As ν evolves along with $\boldsymbol{L}_k$, the second term of ϕ_{CE} acts as a moving target that must be continuously estimated during the learning procedure. For this reason, we choose to optimize ϕ_{CE} by a two-step procedure described in Alg. 10, similarly to an alternating maximization approach such as EM.

This approach bears strong similarities with GANs, in which both generator and discriminator evolve during training (dynamic negatives appear in Goodfellow (2014) as a theoretical tool to analyze differences between NCE and GANs).

Once the generated negative A^- has been used in an iteration of the optimization

Algorithm 10 Optimizing dynamic CE

Input: Positive samples $\mathcal{A}^+$, initial kernel $\boldsymbol{L}_0$, maxIter.
$k \leftarrow 1$
while $k++ <$ maxIter **and** not converged **do**
  $\mathcal{A}^- \leftarrow$ GENERATEDYNAMICNEGATIVES($\boldsymbol{L}_k$, $\mathcal{A}^+$)
  $\boldsymbol{L}_{k+1} \leftarrow$ OPTIMIZECE($\boldsymbol{L}_k, \mathcal{A}^+, \mathcal{A}^-$)
return $\boldsymbol{L}_k$

of ϕ_{CE}, it is less likely to be sampled again.[5] Crucially, such dynamic negatives also avoid the problem alluded to in Remark 4.3.2, since by construction they have a non-zero probability under $\mathcal{P}_{\boldsymbol{L}_k}$ at iteration k.

Static negatives

Conversely, we can simplify the optimization problem by considering a *static* negative distribution: ν does not depend on the current kernel estimate. A considerable theoretical advantage of static negatives lies in the simpler optimization problem: given a static negative distribution ν, the objective ϕ_{CE} does not evolve during training, and is amenable to a simple invocation of stochastic gradient descent (Bottou, 1998).

However, static negative distributions may suffer from the fundamental theoretical issue in Rem. 4.3.2, and hence careful attention must be paid to ensure that the learning algorithm does not converge to a spurious optimum that assigns a probability $\Pr_{\boldsymbol{L}}(A) = 0$ to $A \in \mathcal{A}^-$. In practice, we observed that the local nature of stochastic gradient ascent iterations was sufficient to avoid such behavior.

Let us now discuss two classical choices for fixed ν.

Product negatives. A common choice of negative distribution in other machine learning areas is the *product distribution*, which is the standard "noise" distribution used in NCE. It is defined by

$$\nu(A) = \prod_{i \in A} \hat{p}(i) \prod_{i \notin A} (1 - \hat{p}(i)), \tag{4.15}$$

[5] If A^- happens to be a false negative (*i.e.*, appears in $\mathcal{A}^+$), A^- will be comparatively sampled more frequently as a positive, and so will contribute on average as a positive sample. Additional precautions such as the ones mentioned in (Bose et al., 2018) can also be leveraged if necessary.

where $\hat{p}(i)$ is the empirical probability of $\{i\}$ in $\mathcal{A}^+$. Although (Mikolov et al., 2013) reports better results by raising the $\hat{p}$ to the power $\frac{3}{4}$, we did not observe any improvements when using exponentiated power distributions; for this reason, by *product negatives*, we always indicate the baseline distribution (4.15).

The product distribution is in practice a mismatch for DPPs, as it lacks the negative association property of DPPs that enables them to model the repulsive interactions between similar items.

Explicit negatives. Alternatively, we may have prior knowledge of a class of subsets that our model should *not* generate. For example, we might know that items i and j are negatively correlated and hence unlikely to co-occur. We may also learn via user feedback that some generated subsets are inaccurate. We refer to negatives obtained using such outside information as *explicit negatives*.

A fundamental advantage of explicit negatives is that they can incorporate prior knowledge and user feedback into the learning algorithm. The ability to incorporate such information, to our knowledge, is in itself a novel contribution to DPP learning.

Although such knowledge may be costly and/or only available at rare intervals, a form of continuous learning that would regularly update the state of our prior knowledge (and hence ν) would bring the explicit negative distribution into the realm of dynamic distributions, as described by Alg. 10.

4.3.2 Efficient learning and prediction

We now describe how the Contrastive Estimation problem for DPPs can be optimized efficiently. In order to efficiently generate dynamic negatives, which rely on DPP conditioning, we additionally generalize the dual transformation leveraged in (Osogami et al., 2018) to speed up basket-completion tasks with DPPs. This speed-up impacts the broader use of DPPs, outside of CE learning.

Optimizing ϕ_{CE}

We propose to optimize the CE problem by exploiting a low-rank factorization of the kernel, writing $\boldsymbol{L} = \boldsymbol{V}\boldsymbol{V}^\top$, where $\boldsymbol{V} \in \mathbb{R}^{N \times K}$ and $K \leq N$ is the rank of the kernel, which is fixed *a priori*.

This factorization ensures that the estimated kernel remains positive semi-definite, and enables us to leverage the low-rank computations derived in (Gartrell et al., 2017) and refined in (Osogami et al., 2018). Given the similar forms of the MLE and CE objectives, we use the stochastic gradient ascent algorithm introduced by (Gartrell et al., 2017) to optimize (4.12). In the case of dynamic negatives, we re-generate $\mathcal{A}^-$ after each gradient step; less frequent updates are also possible if the negative generation algorithm is very costly.

We furthermore augment ϕ_{CE} with a regularization term $R(\boldsymbol{V})$, defined as

$$R(\boldsymbol{V}) = \alpha \sum_{i=1}^{N} \frac{1}{\mu_i} \|v_i\|_2^2,$$

where μ_i counts the occurrences of i in the training set, v_i is the corresponding row vector of $\boldsymbol{V}$ and $\alpha > 0$ is a tunable hyperparameter. Note that this is the same regularization as introduced in (Gartrell et al., 2017). This regularization tempers the strength of $\|v_i\|_2$, a term interpretable as to the popularity of item i (Kulesza and Taskar, 2012; Gillenwater et al., 2014), based on its *empirical* popularity μ_i. Experimentally, we observed that adding $R(\boldsymbol{V})$ has a strong impact on the predictive quality of our model.

The reader may wonder if other approaches to DPP learning are also applicable to the CE problem.

Remark 4.3.4. The update rule for the fixed-point approach in (Mariet and Sra, 2015) does not admit a closed form solution for CE, rendering it impractical.

Indeed, letting $\beta = |\mathcal{A}^+| - |\mathcal{A}^-| \geq 0$ and writing $\boldsymbol{U}_A$ as the $N \times |A|$ indicator

matrix such that $\boldsymbol{L}_A = \boldsymbol{U}_A^\top \boldsymbol{L} \boldsymbol{U}_A$, we have

$$\phi(\boldsymbol{L}) \propto \underbrace{-\beta \log\det(\boldsymbol{I}+\boldsymbol{X}) + \sum_{A\in\mathcal{A}^+} \log\det(\boldsymbol{U}_A^\top \boldsymbol{X}^{-1} \boldsymbol{U}_A)}_{f \text{ convex}}$$
$$\underbrace{+\beta \log\det(\boldsymbol{X}) - \sum_{A\in\mathcal{A}^-} \log\det(\boldsymbol{U}_A^\top \boldsymbol{X}^{-1} \boldsymbol{U}_A)}_{g \text{ concave}},$$

where the convexity/concavity results follow immediately from (Mariet and Sra, 2015, Lemma 2.3). Then, the update rule $\nabla f(\boldsymbol{L}_{k+1}) = -\nabla g(\boldsymbol{L}_k)$ requires

$$\beta \boldsymbol{L}_{k+1} + \sum_{A\in\mathcal{A}^-} \boldsymbol{L}_{k+1}\boldsymbol{U}_A(\boldsymbol{U}_A^\top \boldsymbol{L}_{k+1}\boldsymbol{U}_A)^{-1}\boldsymbol{U}_A^\top \boldsymbol{L}_{k+1}$$
$$\leftarrow \beta(\boldsymbol{I}+\boldsymbol{L}_k^{-1})^{-1} + \sum_{A\in\mathcal{A}^+} \boldsymbol{L}_k\boldsymbol{U}_A(\boldsymbol{U}_A^\top \boldsymbol{L}_k\boldsymbol{U}_A)^{-1}\boldsymbol{U}_A^\top \boldsymbol{L}_k,$$

which cannot be evaluated due to the $\sum_{A\in\mathcal{A}^-}$ term.

The low-rank formulation allows us to apply CE (as well as NCE, as discussed in Section 4.3.3) to learn large datasets such as the Belgian retail supermarket dataset (described in Section 4.3.3) without prohibitive learning runtimes. We show below that by leveraging the idea described in (Osogami et al., 2018), the low-rank formulation can also lead to additional speed ups during prediction.

Efficient conditioning for predictions

Dynamic negatives rely upon conditioning a DPP on a chosen sample A (see Alg. 9: $\mathcal{P}_{\boldsymbol{L}_k}(A^- \cup \{j\})$ can be efficiently computed for all j by a preprocessing step that conditions $\boldsymbol{L}_k$ on set A^-). For this reason, we now describe how low-rank DPP conditioning can be significantly sped up.[6]

In (Gartrell et al., 2017), conditioning has a cost of $\mathcal{O}(K|\bar{A}|^2 + |A|^3)$, where $\bar{A} = \mathcal{Y} - A$. Since $|\mathcal{Y}| \gg |A|$ for many datasets, this represents a significant bottleneck for conditioning and computing next-item predictions for a set. We show here that this

[6]We include this section for completeness; these results are attributed to Mike Gartrell.

complexity can be brought down significantly.

Proposition 4.3.1. *Given $A \subseteq \{1, \ldots, N\}$ and a* DPP *of rank K parametrized by $\boldsymbol{V}$, where $\boldsymbol{L} = \boldsymbol{V}\boldsymbol{V}^\top$, we can derive the conditional marginal probabilities in the* DPP *parametrization $\boldsymbol{L}^A$ in only $\mathcal{O}(K^3 + |A|^3 + K^2|A|^2 + |\bar{A}|K^2)$ time.*

Proof. Let $\boldsymbol{V}$ be the low-rank parametrization of the DPP kernel ($\boldsymbol{L} = \boldsymbol{V}^\top\boldsymbol{V}$) and $A \subseteq \mathcal{Y}$. As in Gillenwater et al. (2014), we first compute the dual kernel $\boldsymbol{C} = \boldsymbol{B}^\top\boldsymbol{B}$, where $\boldsymbol{B} = \boldsymbol{V}^\top$. We then compute

$$\boldsymbol{C}^A = (\boldsymbol{B}^A)^\top\boldsymbol{B}^A = \boldsymbol{Z}^A\boldsymbol{C}\boldsymbol{Z}^A,$$

with $\boldsymbol{Z}^A = \boldsymbol{I} - \boldsymbol{B}_A(\boldsymbol{B}_A^\top\boldsymbol{B}_A)^{-1}\boldsymbol{B}_A^\top$, and where $\boldsymbol{C}^A$ is the DPP kernel conditioned on the event that all items in A are observed, and $\boldsymbol{B}_A$ is the restriction of $\boldsymbol{B}$ to the rows and columns indexed by A.

Computing $\boldsymbol{C}^A$ costs $\mathcal{O}(K^3 + |A|^3 + K^2|A|^2)$. Next, following Kulesza and Taskar (2012), we eigendecompose $\boldsymbol{C}^A$ to compute the conditional (marginal) probability P_i of every possible item i in $\bar{A}$:

$$P_i = \sum_{n=1}^{K} \frac{\lambda_n}{\lambda_n + 1}\left(\frac{1}{\sqrt{\lambda_n}}\boldsymbol{b}_i^A\hat{v}_n\right)^2,$$

where $\boldsymbol{b}_i^A$ indicates the column vector for item i in $\boldsymbol{B}^A$ and $(\lambda_n, \hat{v}_n)$ are the eigenvalues/vectors of $\boldsymbol{C}^A$.

The computational complexity for computing the eigendecomposition is $\mathcal{O}(K^3)$, and computing P_i for all items in $\bar{A}$ costs $\mathcal{O}(|\bar{A}|K^2)$. Therefore, we have an overall computational complexity of $\mathcal{O}(K^3 + |A|^3 + K^2|A|^2 + |\bar{A}|K^2)$ for computing next-item conditionals/predictions for the low-rank DPP using the dual kernel, which is significantly superior to the typical cost of $\mathcal{O}(K|\bar{A}|^2 + |A|^3)$. □

As in most cases $K \ll |\bar{A}|$, this represents a substantial improvement, allowing us condition in time essentially linear in the size of the item catalog.

4.3.3 Experiments

We run next-item prediction and AUC-based classification experiments on two recommendation datasets for DPP evaluation: the UK retail dataset (Chen, 2012), which after clipping all subsets to a maximum size[7] of 100, contains 4070 items and 20059 subsets, and the Belgian Retail Supermarket dataset,[8] which contains 88,163 subsets, of a total of 16,470 unique items (Brijs et al., 1999; Brijs, 2003). We compare the following Contrastive Estimation approaches:

- EXP: explicit negatives learned with CE. As to our knowledge there are no datasets with explicit negative information, we generate approximations of explicit negatives by removing one item from a positive sample and replacing it with the least likely item (Algorithm 11).
- DYN: dynamic negatives learned with CE.

As our work revolves around improving DPP performance, we focus on the two following baselines, which are targeted to learning DPP parametrizations from data:

- LR: the standard low-rank DPP stochastic gradient ascent algorithm (Gartrell et al., 2017).
- NCE: Noise Contrastive Estimation using product negatives. NCE learns a model by contrasting $\mathcal{A}^+$ with negatives drawn from a "noisy" distribution p_n, training the model to distinguish between sets drawn from μ and sets drawn from p_n.

NCE has gained popularity due to its ability to model distributions μ with intractable normalization coefficients, and has been shown to be a powerful technique to improve submodular recommendation models (Tschiatschek et al., 2016). NCE learns by maximizing the following conditional log-likelihood:

$$\phi_{\text{NCE}}(\boldsymbol{L}) = \sum_{A \in \mathcal{A}^+} \log P(A \in \mathcal{A}^+ \mid A) + \sum_{A \in \mathcal{A}^-} \log P(A \in \mathcal{A}^- \mid A). \tag{4.16}$$

[7]This allows us to use a low-rank matrix factorization for the DPP that scales well in terms of train and prediction time.

[8]`http://fimi.ua.ac.be/data/retail.pdf`

In our experiments, we learn the NCE objective with stochastic gradient ascent for our low-rank model, since $\nabla \log \Pr(A \in \mathcal{A}^* | A, \boldsymbol{V}\boldsymbol{V}^\top)$ is given by

$$\left(\epsilon^* - \left(1 + \frac{|\mathcal{A}^-|}{|\mathcal{A}^+|} \frac{p_n(A)}{\mathcal{P}_{\boldsymbol{V}\boldsymbol{V}^\top}(A)}\right)^{-1}\right) \nabla_{\boldsymbol{V}} \log \mathcal{P}_{\boldsymbol{V}\boldsymbol{V}^\top}(A). \tag{4.17}$$

where $\epsilon^* = 1$ if $\mathcal{A}^* = \mathcal{A}^+$ and 0 otherwise.

We approximate explicit negatives using Algorithm 11. This allows us to approximate true explicit negatives, as we use the empirical data to derive "implausible" sets. However, when using such negatives we have no guarantee that objective function will be well behaved, as opposed to the theoretically grounded dynamic negatives.

Algorithm 11 Approximate explicit negative generation

input: Positive sample A^+
Sample $i \neq j \in A^+$ w.p. $p_i \propto \widehat{P}(\{i\})$
Sample $k \notin A^+$ w.p. $p_k \propto 1 - \widehat{P}(\{i, k\})$.
return $(A^+ \setminus \{j\}) \cup \{k\}$

The performance of all methods are compared using standard recommender system metrics: Mean Percentile Rank (MPR). MPR is a recall-based metric that evaluates predictive power by measuring how well the model predicts the next item in a basket, and is a standard choice for recommender systems (Hu et al., 2008; Li et al., 2010).

Specifically, given a set A, let $p_{i,A} = \Pr(A \cup \{i\} \mid A)$. The percentile rank of an item i given a set A is defined as

$$\mathrm{PR}_{j,A} = \frac{\sum_{i' \notin A} \mathbb{1}(p_{i,A} \geq p_{i',A})}{|\mathcal{Y} \setminus A|} \times 100\%.$$

The MPR is then computed as

$$\mathrm{MPR} = \frac{1}{|\mathcal{T}|} \sum_{A \in \mathcal{T}} \mathrm{PR}_{i, A \setminus \{i\}},$$

where $\mathcal{T}$ is the set of test instances and i is a randomly selected element in each set A. An MPR of 50 is equivalent to random selection; a MPR of 100 indicates that the model perfectly predicts the held out item.

We also evaluate the discriminative power of each model using the AUC metric. For this task, we generate a set of negative subsets uniformly at random. For each positive subset A^+ in the test set, we generate a negative subset A^- of the same length by drawing $|A^+|$ samples uniformly at random, while ensuring that the same item is not drawn more than once for a subset.

We then compute the AUC for the model on these positive and negative subsets, where the score for each subset is the log-likelihood that the model assigns to the subset. This task measures the ability of the model to discriminate between positive subsets (ground-truth subsets) and randomly generated subsets.

In all experiments, 80% of subsets are used for training; the remaining 20% served as test; convergence is reached when the relative change in the validation log-likelihood is below a pre-determined threshold ϵ, set identically for all methods. All results are averaged over 5 learning trials.

Amazon registries

We conducted an experimental analysis on the largest 7 sub-datasets included in the Amazon Registry dataset. Given the small size of these datasets (the largest has 100 items), these experiments serve only to provide insight into the general behavior of the baselines and CE methods, and into the influence of hyperparameters on convergence.

Table 4.4: MPR, p@k, and AUC values for LR, and baseline improvement over LR for other methods. Positive values indicate the algorithm performs better than LR, and bold values indicate improvement over LR that lies outside the standard deviation. Experiments were run 5 times, with $|\mathcal{A}^+|/|\mathcal{A}^-| = \frac{1}{2}$; α is set to its optimal LR value.

		Improvement over LR		
Metric	LR	DYN	EXP	NCE
MPR	70.50	**0.92 ± 0.56**	**0.68 ± 0.62**	**0.86 ± 0.55**
p@1	9.96	0.67 ± 0.75	0.58 ± 0.76	0.20 ± 1.75
p@5	25.36	**1.04 ± 0.82**	**0.78 ± 0.67**	0.67 ± 1.09
p@10	36.50	**1.39 ± 0.85**	**1.13 ± 0.79**	0.97 ± 1.18
p@20	51.22	**1.38 ± 0.97**	**1.28 ± 1.11**	**1.35 ± 1.20**
AUC	0.630	**0.027 ± 0.017**	**0.026 ± 0.016**	0.009 ± 0.017

In Table 4.4, we compare the performance of the various algorithms with rank $K = 30$. The regularization strength α is set to its optimal value for the LR algorithm, and $|\mathcal{A}^-|/|\mathcal{A}^+| = 1/2$. This allows us to compare the LR algorithm to its "augmented" negative versions without hyper-parameter tuning. As PROD performs much worse than LR, it is not included in further experiments.

We evaluate the precision at k as

$$p@k = \frac{1}{|\mathcal{T}|} \sum_{A \in \mathcal{T}} \frac{1}{|A|} \sum_{i \in A} \mathbb{1}\big[\text{rank}(i \mid A \setminus \{i\}) \leq k\big].$$

Compared to traditional SGA methods, algorithms that use inferred negatives perform (PROD excepted) better across all metrics and datasets. DYN and EXP provide consistent improvements compared to the other methods, whereas NCE shows a higher variance and slightly worse performance. Improvements observed using DYN and EXP are larger than the loss in performance due to going from full-rank to low-rank kernels reported in (Gartrell et al., 2017).

We also found that CE was not sensitive to the α and $|\mathcal{A}^-|/|\mathcal{A}^+|$ hyperparameters. For this reason, in all further results, we set $|\mathcal{A}^-|/|\mathcal{A}^+| = .5$ and $\alpha = 1$. As in previous work on low-rank DPP learning (Gartrell et al., 2017), $\alpha = 1$ was found to be a reasonably optimal value, this ensures a fair comparison between all methods.

Table 4.5 reports the average time to convergence for each method. As generating the dynamic negatives has a high complexity due to DPP conditioning, DYN is 2.7x slower than EXP. LR is the fastest method, as it does not need to process any negatives. NCE is by far the most time-consuming.

Table 4.5: Runtime to convergence (s) on the feeding Amazon registry ($\alpha = 1$, $|\mathcal{A}^-|/|\mathcal{A}^+| = 0.5$, $K = 30$).

METHOD	LR	EXP	DYN	NCE
RUNTIME (S)	0.83 ± 0.54	2.69 ± 0.02	7.13 ± 0.28	27.59 ± 2.20

(a) UK dataset				(b) Belgian dataset		
		Improvement over LR			Improvement over LR	
Metric	LR	EXP	DYN	LR	EXP	DYN
MPR	80.07	**3.75 ± 0.16**	**3.74 ± 0.16**	79.42	**9.58 ± 0.15**	**9.64 ± 0.13**
AUC	0.57297	**0.41465 ± 0.01334**	**0.41467 ± 0.01339**	0.6162	**0.3705 ± 8.569e-5**	**0.3702 ± 4.447e-5**

Table 4.6: Results over the UK and Belgian datasets. Both explicit and dynamic CE obtain statistically significant improvements in MPR and AUC metrics, confirming that CE learning enhances recommender value of the model and its ability to distinguish data drawn from the target distribution from fake samples. The impact on precisions@k metrics is not reported as we did not observe statistically significant deviations from LowRank performance.

UK and Belgian Retail Datasets

Following (Gartrell et al., 2017), for both the UK and the Belgian dataset, we set the rank K of the kernel to be the size of the largest subset in the dataset (K=100 for the UK dataset, K=76 for the Belgian dataset): this optimizes memory costs while still modeling all ground-truth subsets. Based on our results on the smaller Amazon dataset, we set $|\mathcal{A}^-|/|\mathcal{A}^+| = 0.5$ and $\alpha = 1$.

Corroborating our timing results on the Amazon registry, we saw that one iteration of NCE required nearly 11 hours on the Belgian dataset (compared to 5 minutes for one iteration of CE). For this reason, we remove NCE as a baseline from all remaining experiments, as it is not feasible in the general case.

Tables 4.6 (a) and (b) summarize our results; the negative methods show significant MPR improvement over LR, with both DYN and EXP performing almost 10 points higher on the Belgian dataset, and 3 points higher on the UK dataset. This is a striking improvement, compounded by small standard deviations confirming that these results are robust to matrix initialization.

We also see a dramatic improvement over LR in AUC, with an improvement of approximately 0.41 for the UK dataset and 0.37 for the Belgian dataset, across both DYN and EXP methods. Both DYN and EXP perform quite well, with an AUC score of approximately 0.9864 or higher for both models. These results suggest that for larger datasets, CE is effective at improving the discriminative power of the DPP.

4.4 Open problems

- Kulesza (2012) conjectured that the general form of the maximum likelihood estimation problem is NP-hard; this question remains open.

- In (Mariet and Sra, 2015), the question of which range of values for the step-size parameter a will lead to the same guarantees of MLE increase and positive semi-definite estimates in the Picard iteration was raised, and remains open for the KrK-Picard extension.

- As is the case with any structured learning scheme, a Kronecker DPP allows for faster learning and sampling with lower memory costs with the trade-off of losing in expressive power — although experiments on small datasets suggest that this loss of expressivity is minimal, an in-depth investigation of this trade-off and how it compares to other structured kernels would be valuable in choosing one structure over another.

- Algorithm KrK-Picard is motivated by the necessity of preserving the Kronecker structure, while simultaneously maintaining the necessary convex and concave properties of the log-likelihood function $\phi(\boldsymbol{L}) = f(\boldsymbol{L}) + g(\boldsymbol{L})$. Although g is not convex in $\boldsymbol{L}_1 \otimes \boldsymbol{L}_2$, g is *geodesically* convex over the Riemannian manifold of positive definite matrices, which suggests that deriving an iteration that would take advantage of the intrinsic geometry of the problem may be a viable line of future work.

- Contrastive Estimation with dynamic negatives is a difficult optimization problem: the objective function is a moving target, evolving along with the current estimate. As such, the CE optimization problem is in of itself a theoretical problem worthy of independent study.

- The generating logic for both dynamic and static negatives may benefit from further investigation, for CE and other DPP learning schemes.

- In various works (Gillenwater et al., 2014; Mariet and Sra, 2015), the initialization of a DPP learning algorithm has been shown to significantly impact the final log-likelihood; a deeper study of initialization procedures for DPP learning algorithms, including KRK-PICARD, remains an important investigation area.

Chapter 5

Generalized volume sampling

We have seen that determinantal point processes are a powerful and intuitive model for diversity within subsets of a ground set. However, DPPs are far from the only measures able to tractably leverage negative dependence, nor indeed the only tractable negatively dependent measures with key implications within machine learning.

Suppose a chemist is planning on running expensive, time-consuming experiments to evaluate an underlying parameter $\theta \in \mathbb{R}^m$. Given experimental parameters represented by a vector $x \in \mathbb{R}^m$, they observe the outcome $y = x^\top \theta + \epsilon \in \mathbb{R}$, where ϵ is some random noise. As the experiments are costly to run, the chemist must, out of a total number N of possible experiments, choose a much smaller set of k experiments to run in order to reliably approximate θ. Equivalently, the chemist must select a small set of rows from the tall and thin experimental design matrix $\boldsymbol{X} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times m}$. Negative dependence is of fundamental importance in this scenario; as the chemist can only choose a few experiments, they must choose carefully so that the outcome of two different experiments provides them with complementary information.

This chapter focuses on *Generalized Volume Sampling (GVS) measures*: strongly Rayleigh measures that select k rows from a tall and thin matrix $X \in \mathbb{R}^{N \times m}$ ($m \leq k \leq N$), and which exactly characterize the negative dependence exhibited by the experimental design problem. A special case of GVS measure was introduced by Avron and Boutsidis (2013), using the standard volume of induced submatrices to define sampling probabilities. Although the wider class of GVS measures have not achieved

the popularity of DPPs in machine learning, they nonetheless have several key applications to a variety of fundamental optimization problems, and deserve attention in their own right.

The main contribution of this chapter lies in the introduction of the class of GVS measures. We begin defining these measures formally and deriving their negative dependence properties. Then, we focus on their use for the experimental design problem described above, deriving probabilistic and greedy algorithms to choose sets of experiments. Finally, we conclude by tying GVS measures and elementary symmetric polynomials to a generalized form of the Minimum Volume Covering Ellipsoid. All material in this chapter was first derived in (Mariet and Sra, 2017).

5.1 Background

Generalized volume sampling measures are defined based on elementary symmetric polynomials (ESPs), a fundamental family of polynomials that has been extensively studied in matrix theory, combinatorics, information theory, as well as several other areas — see *e.g.*, (Jozsa and Mitchison, 2015; Macdonald, 1998; Horn and Johnson, 1985; Bhatia, 2007; Jain, 2011; Bauschke et al., 2001).

Definition 5.1.1 (Elementary symmetric polynomial)**.** The ℓ-th elementary symmetric polynomial over $\mathbb{R}^m$ is defined as

$$e_\ell(x_1, \dots, x_m) \triangleq \sum_{I \subseteq [m], |I| = \ell} \prod_{j \in I} x_j, \tag{5.1}$$

Eq. (5.1) extends to matrices naturally: elementary symmetric polynomials are spectral functions, and so we will also write

$$e_\ell(\boldsymbol{M}) \triangleq e_\ell \circ \lambda(\boldsymbol{M}).$$

Remark 5.1.1. We have the following special identities for elementary symmetric polynomials: $e_0 \equiv 1, e_1 = \mathrm{Tr}, e_m = \det, e_{\ell > m} \equiv 0$.

Remark 5.1.2. Elementary symmetric polynomials also provide the partition function for k-DPPs (Definition 3.1.2).

The following proposition captures basic properties of ESPs that we will require in our analysis; the proofs can be found in standard works such as (Bhatia, 2007).

Proposition 5.1.1. *Let $\boldsymbol{M} \in \mathbb{R}^{m \times m}$ be symmetric and $1 \leq \ell \leq m$; also let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{S}_m^+$. We have the following properties:*

(i) If $\boldsymbol{A} \succeq \boldsymbol{B}$ in Löwner order, then $e_\ell(\boldsymbol{A}) \geq e_\ell(\boldsymbol{B})$;

(ii) If $\boldsymbol{M}$ is invertible, then $e_\ell(\boldsymbol{M}^{-1}) = \det(\boldsymbol{M}^{-1}) e_{m-\ell}(\boldsymbol{M})$;

(iii) $\nabla e_\ell(\lambda) = [e_{\ell-1}(\lambda^{(i)})]_{1 \leq i \leq m}$ where $\lambda^{(i)}$ is vector λ without its i-th component.

Elementary symmetric polynomials enjoy another representation that allows us to interpret them as "partial volumes": namely,

$$e_\ell(\boldsymbol{M}) = \sum\nolimits_{S \subseteq [m], |S| = \ell} \det(\boldsymbol{M}_S). \tag{5.2}$$

From this notion of partial volumes, we are now able to properly define Generalized Volume Sampling measures, that sample subsets of rows from a tall, thin matrix $\boldsymbol{X}$ based on their spanned partial volume.

Definition 5.1.2 (Generalized Volume Sampling)**.** A distribution μ over $2^{[N]}$ is a GVS measure if there exists $\boldsymbol{X} \in \mathbb{R}^{N \times m}$ and an integer $\ell \in \{0, \ldots, m\}$ such that

$$\mu(S) \propto e_\ell(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}). \tag{5.3}$$

Remark 5.1.3. Writing $x_1, \ldots, x_N \in \mathbb{R}^m$ the row vectors of $\boldsymbol{X}$, the probability defined in (5.3) is equivalent to

$$\mu(S) \propto e_\ell\Big(\sum\nolimits_{i \in S} x_i x_i^\top\Big). \tag{5.4}$$

The special case $\ell = m$ (*i.e.*, $e_\ell = \det$) has benefited from nascent interest in machine learning; first studied by Avron and Boutsidis (2013), polynomial-time sampling

algorithms were developed in (Li et al., 2017) for these measures, under the name of *Dual* Volume Sampling which informs our choice of Generalized Volume Sampling to name the larger class in Definition 5.1.2.

5.1.1 Negative dependence properties

That Dual Volume Sampling measures ($\ell = m$) are strongly Rayleigh was proven in (Li et al., 2017). To prove that *all* GVS measures are Strong Rayleigh, we follow the same steps as Li et al. (2017), making a few modifications to generalize to all elementary symmetric polynomials.

Proposition 5.1.2. *For $m \leq |S| \leq N$ and $\boldsymbol{L} = \boldsymbol{X}\boldsymbol{X}^\top$ with $\boldsymbol{X} \in \mathbb{R}^{N \times m}$, we have*

$$e_\ell(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) = e_\ell(\boldsymbol{L}_{S,S}).$$

Proof. Let $\boldsymbol{Y} = \mathrm{Diag}(y_1, \ldots, y_N)$ be a diagonal matrix. The generalized Cauchy-Binet identity yields

$$e_\ell(\boldsymbol{X}^\top \boldsymbol{Y} \boldsymbol{X}) = \sum_{\substack{T \subseteq [m] \\ |T| = \ell}} \det([\boldsymbol{X}^\top \boldsymbol{Y} \boldsymbol{X}]_{T,T}) = \sum_{\substack{T \subseteq [m] \\ |T| = \ell}} \sum_{\substack{K \subseteq [N] \\ |K| = \ell}} \det([\boldsymbol{X}^\top \boldsymbol{Y}]_{T,K}) \det(\boldsymbol{X}_{K,T}).$$

For $\boldsymbol{Y}$ the indicator matrix of set $S \subseteq [N]$, all terms with $K \not\subseteq S$ disappear, yielding

$$e_\ell(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) = \sum_{\substack{T \subseteq [m] \\ |T| = \ell}} \sum_{\substack{K \subseteq S \\ |K| = \ell}} \det(\boldsymbol{X}_{T,K}^\top) \det(\boldsymbol{X}_{K,T}) = \sum_{\substack{K \subseteq S \\ |K| = \ell}} \det([\boldsymbol{X}\boldsymbol{X}^\top]_{K,K}) = e_\ell([\boldsymbol{X}\boldsymbol{X}^\top]_{S,S}),$$

which concludes the proof. □

Theorem 5.1.3 (GVS measures are SR). *Let $\boldsymbol{X} \in \mathbb{R}^{N \times m}$, $k \in \{m, \ldots, N\}$ and $\ell \in \{1, \ldots, m\}$. The multi-affine polynomial g, defined over $\mathbb{C}^n$ as*

$$g(z_1, \ldots, z_N) = \sum_{|S| = k, S \subseteq [N]} e_\ell(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) z^S,$$

is real stable.

Proof. Let $\boldsymbol{L} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N\times N}$. Writing $\boldsymbol{W}$ for the diatonal matrix $\mathrm{Diag}(w_1, \ldots, w_N)$ and $\boldsymbol{Z} = \mathrm{Diag}(z_1, \ldots, z_N)$, we know from Prop. 2.2.10 that

$$g(z, w) = \det(\boldsymbol{L} + \boldsymbol{W} + \boldsymbol{Z}) = \sum_{S\subseteq[N]} \det(\boldsymbol{L}_S + \boldsymbol{W}_S) z^S = \sum_{S\subseteq[N]} \big(\sum_{T\subseteq S} \det(\boldsymbol{L}_T) w^T\big) z^S$$

is real stable in $2N$ variables; $g(z, w)$ can be expanded as

$$g(z, w) = \sum_{S\subseteq[N]} \big(\sum_{T\subseteq S} \det(\boldsymbol{L}_T) w^{S\setminus T}\big) z^{\bar{S}}.$$

Truncating $g(z, w)$ to degree $N - m$, which constrains $|T| = m$, then setting $w = (y, \ldots, y)$ yields

$$g(z, y) = \sum_{S\subseteq[N]} \big(\sum_{\substack{T\subseteq S \\ |T|=m}} \det(\boldsymbol{L}_T)\big) y^{|S|-m} z^{\bar{S}} = \sum_{S\subseteq[N]} e_\ell(\boldsymbol{L}_S) y^{|S|-m} z^{\bar{S}},$$

which once again remains real stable (Prop. 2.2.7). Finally, $k - m$ differentiations of $g(z, y)$ according to y followed by evaluating at $y = 0$ (both operations preserving stability as stated in Prop. 2.2.5) yields polynomial

$$h(z) = \sum_{S\subseteq[N], |S|=k} e_\ell(\boldsymbol{L}_S) z^{\bar{S}}.$$

By applying Prop. 2.2.5–*(iv)* we recover the generating polynomial for the GVS measure of degree ℓ (Prop. 5.1.2), concluding our proof. □

We now discuss a key applications of GVS measures to machine learning: optimal experimental design.

5.2 Optimal experimental design

Optimal experimental design (OED) develops the theory of selecting a small subset of experiments to perform in order evaluate a hidden variable as well as possible. As discussed earlier, OED has fundamental ties to negative dependence. Indeed,

to evaluate a multi-dimensional parameter θ in the fewest number of experiments possible, each experiment must provide new information about θ, making selecting similar experiments a sub-optimal choice.

5.2.1 Generalized Volume Sampling and optimal design

OED operates under the assumption that experiments are costly and cannot be run many times or even once without tremendous difficulty (Pukelsheim, 2006). OED has been applied in a large number of experimental settings (Sagnol, 2010; Cohn, 1994; Lasheras et al., 2010; Xygkis et al., 2016; Schein and Ungar, 2004), and has close ties to related machine learning problems such as outlier detection (Dolia et al., 2004; Jackson and Chen, 2004), active learning (He, 2010; Gu and Jin, 2013), and Gaussian process driven sensor placement (Krause et al., 2008) among others.

The OED problem is formulated as follows: there exist N experiments whose outcome y depends linearly on a hidden parameter $\theta \in \mathbb{R}^m$ according to (5.5):

$$y_i = x_i^\top \theta + \epsilon_i \quad 1 \leq i \leq n, \tag{5.5}$$

where $x_i \in \mathbb{R}^m$ represents the experiment parameters, and the ϵ_i are independent zero mean homoscedastic Gaussian noise vectors. Within this setting, OED's goal is to select a set S of k experiments that estimate θ with no bias and minimal variance.

The Gauss-Markov theorem guarantees that for a set S such that $\sum_{i \in S} x_i x_i^\top$ is invertible (in which case S is called a *feasible* set), the lowest variance for an unbiased estimate $\hat{\theta}$ verifies

$$\mathrm{Var}[\hat{\theta}] = \left(\sum\nolimits_{i \in S} x_i x_i^\top \right)^{-1}.$$

Of course, this variance is a positive semi-definite matrix, over the set of which there is no total order. To formally seek to minimize the variance, we require a suitable map Φ that maps (positive semi-definite) matrices to scalars. Hence, OED is cast as an optimization problem that seeks *an optimal design* S^* over the space $\Gamma_k = \{S \subseteq$

$[N], |S| \leq k, \sum_{i \in S} x_i x_i^\top$ is invertible$\}$ of feasible designs:

$$\min_{S \in \Gamma_k} f_\ell(S) \triangleq \Phi\Big(\big(\sum\nolimits_{i \in S} x_i x_i^\top\big)^{-1}\Big), \tag{5.6}$$

where Φ is a cost function that maps positive definite matrices to scalars.

The aim of (5.6) is thus to pick a subset S of k points out of N (typically $k \ll N$) that reduce the uncertainty as much as possible, measured according to the cost function Φ; the choice of Φ elicits different properties that a solution should satisfy, either statistical or structural.

The theory of OED branches into multiple variants of (5.6) depending on the choice of Φ, among which A-optimal design (Φ = trace) and D-optimal design (Φ = determinant) are probably the two most popular choices. Each of these choices has a wide range of applications as well as statistical, algorithmic, and other theoretical results. Due to the individual popularity of A- and D-optimal design, the theory surrounding these two sub-problems has diverged significantly.

As both the trace and the determinant are special cases of elementary symmetric polynomials, we generalize (5.6) to a broader optimization problem where Φ is obtained from an elementary symmetric polynomial; this allows us to consider A-optimal design and D-optimal design as special cases and treat the entire class of problems in a unified manner; we refer to this class of problems as *ESP-design*.

5.2.2 Problem definition

Let $\boldsymbol{X} \in \mathbb{R}^{N \times m}$ ($m \ll N$) be a matrix with full column rank, and $k \in \mathbb{N}$ satisfy $m \leq k \leq N$. For $\ell \in \{1, \ldots, m\}$, we introduce the *ESP-design* problem:

$$\min_{S \in \Gamma_k} \quad f_\ell(S) \triangleq \tfrac{1}{\ell} \log e_\ell\big((\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:})^{-1}\big). \tag{5.7}$$

We keep the $1/\ell$-factor in (5.7) to highlight the homogeneity (e_ℓ is a polynomial of degree ℓ) of our design criterion, as is advocated in (Pukelsheim, 2006, Ch. 6).

For $\ell = 1$, (5.7) yields A-optimal design, while for $\ell = m$, it yields D-optimal

design. For $1 < \ell < m$, ESP-design interpolates between these two extremes. Geometrically, we may view it as seeking an ellipsoid with the smallest average volume for ℓ-dimensional slices (taken across sets of size ℓ).

Alternatively, ESP-design can be also be interpreted as a regularized version of D-optimal design via Prop. 4.2.1-*(ii)*. In particular, for $\ell = m - 1$, we recover a form of regularized D-optimal design:

$$f_{m-1}(S) = \tfrac{1}{m-1}\big[\log\det\big((\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:})^{-1}\big) + \log\|\boldsymbol{X}_{S,:}\|_2^2\big].$$

Furthermore, using Prop. 5.1.1 *(ii)*, we can also rephrase the OED problem with $\Phi \equiv e_\ell$ as finding the mode of the product of GVS measures given by

$$S \to \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:})^{-1} e_{m-\ell}(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}).$$

Note that (5.7) is a hard combinatorial optimization problem (the case $\ell = m$, which also benefits from submodularity, is known to be NP-hard (Welch, 1982)), which precludes an exact solution. However, the objective function in its general ESP form enjoys remarkable properties; the following section focuses on the properties of the continuous relaxation of (5.7), which will be crucial to solving the discrete optimization problem.

5.2.3 Continuous optimization over elementary symmetric polynomials

In A-optimal design, a common approach to solving (5.7) consists in relaxing the constraint on S by allowing elements in the set to have fractional multiplicities. In the general case of ESP-design, this relaxation leads to the following continuous optimization problem:

$$\min_{z \in \Gamma_k^c} \tfrac{1}{\ell} \log e_\ell\Big((\boldsymbol{X}^\top \operatorname{Diag}(z)\boldsymbol{X})^{-1}\Big), \tag{5.8}$$

where by Γ_k^c we denote the set of vectors $\{z \in \mathbb{R}^n \mid 0 \leq z_i \leq 1\}$ such that $\boldsymbol{X}^\top \operatorname{Diag}(z)\boldsymbol{X}$ remains invertible and $\mathbf{1}^\top z \leq k$. The following proposition is a direct consequence of Proposition 4.2.1(i).

Proposition 5.2.1. *Let z^* be the optimal solution to* (5.8). *Then $\|z^*\|_1 = k$.*

The convexity of $\log \operatorname{tr}((\boldsymbol{X}^\top \operatorname{Diag}(z)\boldsymbol{X})^{-1})$ is a key ingredient to obtaining high-quality A-optimal designs (Wang et al., 2016a). The rest of this section focuses on establishing the convexity of f_ℓ for general ℓ on Γ_k^c, where by abuse of notation, f_ℓ also denotes the continuous relaxation in (5.8).

Although convexity of f_ℓ may be obtained by as a consequence of a classic result of Muir (1974), we present a new way of deriving it, based on a result on the geodesic log-convexity of functions of elementary symmetric polynomials.

We begin by deriving the geodesic convexity (see Appendix A.1.1) of $\log e_\ell$. Note that geodesic convexity of $\log e_\ell$ on positive definite matrices is strictly stronger than (and implies) the convexity of f_ℓ.

Theorem 5.2.2. *The function $\boldsymbol{X} \mapsto e_\ell(\boldsymbol{X}^s)$ for $s \in \{-1, 1\}$ is log-g-convex on positive definite matrices.*

Proof. By continuity, it suffices to prove midpoint log geodesic convexity; that is, it suffices to prove

$$e_\ell(\boldsymbol{P}\#\boldsymbol{Q}) \leq \sqrt{e_\ell(\boldsymbol{P})e_\ell(\boldsymbol{Q})}.$$

Since $(\boldsymbol{P}\#\boldsymbol{Q})^{-1} = \boldsymbol{P}^{-1}\#\boldsymbol{Q}^{-1}$, the same proof shows log geodesic convexity of $e_\ell(\boldsymbol{X}^{-1})$.

From multilinear algebra (*e.g.*, (Bhatia, 1997, Ch. 1)) we know that for an $N \times N$ matrix $\boldsymbol{P}$, there exists a matrix $\boldsymbol{W}$ such that $e_\ell(\boldsymbol{P}) = \operatorname{tr} \boldsymbol{W}^* \boldsymbol{P}^{\otimes N} \boldsymbol{W}$, where $\otimes$ denotes the Kronecker product. Lemma 2.23 in (Sra and Hosseini, 2015) shows that

$$(\boldsymbol{P}\#\boldsymbol{Q})^{\otimes N} = \boldsymbol{P}^{\otimes N}\#\boldsymbol{Q}^{\otimes N},$$

from which it follows that

$$e_\ell(\boldsymbol{P}\#\boldsymbol{Q}) = \operatorname{tr} \boldsymbol{W}^*(\boldsymbol{P}\#\boldsymbol{Q})^{\otimes N}\boldsymbol{W} = \operatorname{tr} \boldsymbol{W}^*[\boldsymbol{P}^{\otimes N}\#\boldsymbol{Q}^{\otimes N}]\boldsymbol{W}$$

$$\leq [\operatorname{tr} \boldsymbol{W}^* \boldsymbol{P}^{\otimes N} \boldsymbol{W}]^{1/2} [\operatorname{tr} \boldsymbol{W}^* \boldsymbol{Q}^{\otimes N} \boldsymbol{W}]^{1/2}$$
$$= [e_\ell(\boldsymbol{P}) e_\ell(\boldsymbol{Q})]^{1/2},$$

where the inequality follows from log geodesic convexity of the trace (Sra and Hosseini, 2015, Cor. 2.9). □

Theorem 5.2.2 immediately yields the following corollary:

Corollary 5.2.2.1. *The map $\boldsymbol{X} \mapsto \log e_\ell(\boldsymbol{X})$ is convex on positive definite matrices.*

As a corollary of the log-convexity of e_ℓ, we obtain that (5.8) is a convex optimization problem, which can therefore be solved using a variety of efficient algorithms. Projected gradient descent turns out to be particularly easy to apply because we only require projection onto the intersection of the cube $\mathbf{0} \leq z \leq \mathbf{1}$ and the plane $\{z \mid z^\top \mathbf{1} = k\}$ (as a consequence of Prop 5.2.1). Projection onto this intersection is a special case of the so-called continuous quadratic knapsack problem, which is a very well-studied problem and can be solved essentially in linear time (Cominetti et al., 2014; Davis et al., 2016).

Remark 5.2.1. The convex relaxation remains log-convex when points can be chosen with multiplicity, in which case the projection step is also significantly simpler, requiring only $z \geq \mathbf{0}$.

We conclude the analysis of the continuous relaxation by a bound on the support of its solution under some mild assumptions; this bound will serve as a first step to satisfying the set size constraint of the original ESP-design problem.

Theorem 5.2.3. *Let ϕ be the mapping from $\mathbb{R}^m$ to $\mathbb{R}^{m(m+1)/2}$ such that $\phi(x) = (\xi_{ij} x_i x_j))_{1 \leq i,j \leq m}$ with $\xi_{ij} = 1$ if $i = j$ and 2 otherwise. Let $\tilde{\phi}(x) = (\phi(x), 1)$ be the affine version of ϕ. If for any set of $m(m+1)/2$ distinct rows of $\boldsymbol{X}$, the mapping under $\tilde{\phi}$ is independent, then the support of the optimum z^* of (5.8) satisfies $\|z^*\|_0 \leq k + \frac{m(m+1)}{2}$.*

The proof is identical to that of (Wang et al., 2016a, Lemma 3.5), which shows such a result for $\ell = 1$.

Theorem 5.2.3 shows that the support of the solution to the continuous relaxation of (5.7) yields a subset S_0 of bounded size; by construction, S_0 is at least as good as an optimal solution to (5.7). To bring down the size of the continuous relaxation solution S_0 to satisfy the size constraint $|S| \leq k$, we now turn to combinatorial optimization.

5.2.4 Finding an optimal design

The combinatorial form of ESP-design admits an intuitive greedy approach despite not being a submodular optimization problem in general. In this greedy form, elements are removed one-by-one from a base set of experiments; greedy removal, as opposed to greedy addition, turns out to be much more practical.

Indeed, since f_ℓ is not defined for sets of size smaller than k, we cannot quantify the quality of intermediary (of size $< k$) sets in a greedy addition algorithm. This difficulty precludes analyses such as (Wang et al., 2016b; Smith and Thai, 2017) for optimizing non-submodular set functions by bounding their "curvature".

Algorithm 12 Greedy algorithm

Input: Matrix $\boldsymbol{X}$, budget k, initial set S_0

$S \leftarrow S_0$

while $|S| > k$ **do**

  Find $i \in S$ such that $S \setminus \{i\}$ is feasible and i minimizes $f_\ell(S \setminus \{i\})$

  $S \leftarrow S \setminus \{i\}$

return S

Instead, we focus on the greedy removal algorithm described in Algorithm 12; obtaining a bound on its performance relies on the following algorithm.

Lemma 5.2.4. *Let $\boldsymbol{X} \in \mathbb{R}^{N \times m} (N \geq m)$ be a matrix with full column rank, and let k be a budget $m \leq k \leq N$. Let S of size k be subset of $[N]$ drawn with probability $\mathcal{P} \propto \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:})$. Then*

$$_{S \sim \mathcal{P}} \left[e_\ell\Big(\left(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\right)^{-1} \Big) \right] \leq \prod_{i=1}^{\ell} \frac{N-m+i}{k-m+i} \cdot e_\ell\Big(\left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \Big), \tag{5.9}$$

with equality if $\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:} \succ 0$ *for all subsets* S *of size* k.

Some simple manipulations using Stirling's approximation show that the ratio $\prod_{i=1}^{\ell} \frac{N-m+i}{k-m+i}$ is upper bounded by $\left(1+\frac{\ell}{N-m}\right)^{N-k}\left(1+\frac{N-k}{k-m+\ell}\right)^{\ell}$.

Proof. The below calculations depend heavily on the Cauchy-Binet formula, of which we reproduce a special case here for $\boldsymbol{X} \in \mathbb{R}^{N\times m}$:

$$\det(\boldsymbol{X}^\top \boldsymbol{X}) = \sum\nolimits_{S\subseteq[N],|S|=m} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}). \tag{5.10}$$

We also use the representation (5.2). By definition we have

$$\mathbb{E}\left[e_\ell\left(\left(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\right)^{-1}\right)\right] = \frac{\sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) e_\ell\left(\left(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\right)^{-1}\right)}{\sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:})}.$$

For the denominator, we have

$$\begin{aligned}
\sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) &= \sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) \\
&\overset{(a)}{=} \sum_{S\subseteq[N],|S|=k} \sum_{T\subseteq S,|T|=m} \det(\boldsymbol{X}_{T,:}^\top \boldsymbol{X}_{T,:}) \\
&\overset{(b)}{=} \binom{N-m}{k-m} \sum_{T\subseteq S,|T|=m} \det(\boldsymbol{X}_{T,:}^\top \boldsymbol{X}_{T,:}) \\
&\overset{(c)}{=} \binom{N-m}{k-m} \det(\boldsymbol{X}^\top \boldsymbol{X}),
\end{aligned}$$

where (a) is obtained via the Cauchy-Binet formula (5.10), (b) by noticing that there are $\binom{N-m}{k-m}$ sets of size k that contain a set T of size m, and (c) by reapplying (5.10).

For the numerator, we first use the fact that $e_\ell(A^{-1}) = \frac{1}{\det A} e_{m-\ell}(A)$:

$$\begin{aligned}
&\sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) e_\ell\left(\left(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\right)^{-1}\right) \\
&\qquad \overset{(a)}{\leq} \sum_{S\subseteq[N],|S|=k} e_{m-\ell}(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) \\
&\qquad = \sum_{S\subseteq[N],|S|=k} \sum_{L\subseteq[m],|L|=m-\ell} (\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:})[L|L]
\end{aligned}$$

$$\overset{(b)}{=} \sum_{S\subseteq[N],|S|=k} \sum_{L\subseteq[m],|L|=m-\ell} \det((\boldsymbol{Y}_L)_{S,:}^\top (\boldsymbol{Y}_L)_{S,:})$$
$$\overset{(c)}{=} \sum_{S\subseteq[N],|S|=k} \sum_{L\subseteq[m],|L|=m-\ell} \sum_{T\subseteq S,|T|=m-\ell} \det((\boldsymbol{Y}_L)_{T,:}^\top (\boldsymbol{Y}_L)_{T,:}).$$

Here, (a) is just (5.2); we have equality if all subsets S of size k produce strictly positive definite matrices $\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}$. For (b), we note $\boldsymbol{Y}_L$ the submatrix of $\boldsymbol{X}$ with all rows but those in L removed; then, $(\boldsymbol{Y}_L)_{S,:}^\top (\boldsymbol{Y}_L)_{S,:} = [\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}]$ for all subsets S.

By counting subset occurrences and reapplying the Cauchy-Binet formula, we have

$$\sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) e_\ell\left(\left(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\right)^{-1}\right)$$
$$\leq \binom{N-m+\ell}{k-m+\ell} \sum_{L\subseteq[m],|L|=m-\ell} \sum_{T\in[n],|T|=m-\ell} \det((\boldsymbol{Y}_L)_{T,:}^\top (\boldsymbol{Y}_L)_{T,:})$$
$$\overset{(d)}{=} \binom{N-m+\ell}{k-m+\ell} \sum_{L\subseteq[m],|L|=m-\ell} \det((\boldsymbol{Y}_{L,:})^\top (\boldsymbol{Y}_{L,:}))$$
$$= \binom{N-m+\ell}{k-m+\ell} \sum_{L\subseteq[m],|L|=m-\ell} (\boldsymbol{X}^\top \boldsymbol{X})[L|L]$$
$$= \binom{N-m+\ell}{k-m+\ell} e_{m-\ell}(\boldsymbol{X}^\top \boldsymbol{X}).$$

Hence, we can finally bound the expected value of under the objective function of the sampled subset S as

$$\mathbb{E}\left[e_\ell\left(\left(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\right)^{-1}\right)\right] = \frac{\sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}) e_\ell\left(\left(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\right)^{-1}\right)}{\sum_{S\subseteq[N],|S|=k} \det(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:})}$$
$$= \frac{\binom{N-m+\ell}{k-m+\ell} e_{m-\ell}(\boldsymbol{X}^\top \boldsymbol{X})}{\binom{k-m}{N-m} \det(\boldsymbol{X}^\top \boldsymbol{X})}$$
$$= \left(\prod_{i=1}^{\ell} \frac{N-m+i}{k-m+i}\right) e_\ell((\boldsymbol{X}^\top \boldsymbol{X})^{-1}).$$

□

Lemma 5.2.4 extends a result from (Avron and Boutsidis, 2013, Lemma 3.9) on column-subset selection via volume sampling to all elementary symmetric polynomials.

In particular, it follows that removing one element (by volume sampling a set of size $N-1$) will in expectation decrease f by a multiplicative factor which is clearly also attained by a greedy minimization. From this argument, we obtain the following bound on the performance of Algorithm 12.

Theorem 5.2.5. *Algorithm 12 initialized with a set S_0 of size N_0 produces a set S^+ of size k such that*

$$e_\ell\Big(\big(\boldsymbol{X}_{S^+,:}^\top \boldsymbol{X}_{S^+,:}\big)^{-1} \Big) \leq \prod_{j=1}^{\ell} \frac{N_0 - m + j}{k - m + j} \cdot e_\ell\Big(\big(\boldsymbol{X}_{S_0,:}^\top \boldsymbol{X}_{S_0,:}\big)^{-1} \Big). \tag{5.11}$$

Proof. We recursively show that greedily removing j items constructs a set S (of size $N-j$) such that

$$e_\ell\left(\big(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\big)^{-1} \right) \leq \left(\prod_{i=1}^{\ell} \frac{N - m + i}{N - j - m + i} \right) e_\ell\left(\big(\boldsymbol{X}^\top \boldsymbol{X}\big)^{-1} \right). \tag{5.12}$$

(5.12) is trivially true for $j = 0$. Assume now that (5.12) holds for $j \geq 0$, and let S_j be the corresponding set of size $(N-j)$. Let now S_{j+1} be the set of size $|S_j| - 1$ that minimizes $e_\ell(\boldsymbol{X}_{S_{j+1},:}^\top \boldsymbol{X}_{S_{j+1},:})$.

From lemma 5.2.4, we know that for sets S of size $|S_j| - 1$ drawn according to dual volume sampling,

$$\mathbb{E}\left[e_\ell\left(\big(\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}\big)^{-1} \right) \right] \leq \left(\prod_{i=1}^{\ell} \frac{|S_j| - m + i}{(|S_j| - 1) - m + i} \right) e_\ell\left(\Big(\boldsymbol{X}_{S_j,:}^\top \boldsymbol{X}_{S_j,:}\Big)^{-1} \right).$$

In particular, the minimum of $e_\ell(\boldsymbol{X}_{S,:}^T \boldsymbol{X}_{S,:})$ over all sets of size $|S_j| - 1$ is upper bounded by the expectancy:

$$e_\ell(\boldsymbol{X}_{S_{j+1},:}^\top \boldsymbol{X}_{S_{j+1},\top}) \leq \left(\prod_{i=1}^{\ell} \frac{N - j - m + i}{N - j - 1 - m + i} \right) e_\ell\left(\Big(\boldsymbol{X}_{S_j,:}^\top \boldsymbol{X}_{S_j,:}\Big)^{-1} \right).$$

By recursion hypothesis applied to S_j, we then have

$$e_\ell(\boldsymbol{X}_{S_{j+1},:}^\top \boldsymbol{X}_{S_{j+1},:}) \leq \Big(\prod_{i=1}^{\ell} \frac{N - j - m + i}{N - j - 1 - m + i} \Big) e_\ell\Big(\Big(\boldsymbol{X}_{S_j,:}^\top \boldsymbol{X}_{S_j,:}\Big)^{-1} \Big)$$

$$\leq \Big(\prod_{i=1}^{\ell} \frac{N-j-m+i}{N-j-1-m+i}\Big)\Big(\prod_{i=1}^{\ell} \frac{n-m+i}{n-j-m+i}\Big) e_\ell\Big(\left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1}\Big)$$
$$\leq \Big(\prod_{i=1}^{\ell} \frac{N-m+i}{N-(j+1)-m+i}\Big) e_\ell\Big(\left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1}\Big),$$

which concludes the recursion. Then, constructing a set of size k amounts to setting $j = n - k$ in inequality (5.12), proving the inequality (5.11). □

As Wang et al. (2016a) note regarding A-optimal design, (5.11) provides a trivial optimality bound on the greedy algorithm when initialized with $\{1, \ldots, N\}$: denoting by S^* the optimal set, the greedy solution S^+ satisfies

$$E_\ell\Big(\left(\boldsymbol{X}_{S^+,:}^\top \boldsymbol{X}_{S^+,:}\right)^{-1}\Big)^{1/\ell} \leq \frac{N-m+\ell}{k-m+1} f([N])$$
$$\leq \frac{N-m+\ell}{k-m+1} E_\ell\Big(\left(\boldsymbol{X}_{S^*,:}^\top \boldsymbol{X}_{S^*,:}\right)^{-1}\Big)^{1/\ell}.$$

This naive initialization can be replaced by the support $\|z^*\|_0$ of the convex relaxation solution; in the setting described by Theorem 5.2.3, we obtain the following result:

Theorem 5.2.6. *Let $\tilde{\phi}$ be the mapping defined in 5.2.3, and assume that all choices of $m(m+1)/2$ distinct rows of $\boldsymbol{X}$ always have their mapping independent mappings for $\tilde{\phi}$. Then the outcome of the greedy algorithm initialized with the support of the solution to the continuous relaxation verifies*

$$f_\ell(S^+) \leq \log\left(\frac{k + m(m-1)/2 + \ell}{k-m+1}\right) + f_\ell(S^*).$$

Computing the ℓ-th elementary symmetric polynomial on a vector of size m can be done in $\mathcal{O}(m \log^2 \ell)$ using Fast Fourier Transform for polynomial multiplication, using the construction by Ben-Or (Shpilka and Wigderson, 2001). Hence, computing $f_\ell(S)$ requires $\mathcal{O}(nm^2)$ operations, where the cost is dominated by computing $\boldsymbol{X}_{S,:}^\top \boldsymbol{X}_{S,:}$.

Hence, by combining the continuous relaxation and the combinatorial optimization of Alg. 12, we obtain a greedy algorithm with performance guarantees that runs in $\mathcal{O}(m^2 N^3)$ time, plus the additional cost of solving the continuous relaxation.

5.3 Experiments: optimal design

In order to validate experimentally our analysis, we compared the following methods to solving the optimization problem (5.7):

- UNIF/UNIFFDV: k experiments sampled uniformly / with Fedorov exchange
- GREEDY/GREEDYFDV: greedy method (relaxed init.) / with Fedorov exchange
- SAMPLE: sampling (relaxed init.) items independently from the Bernoulli distribution represented by the solution to the continuous relaxation.

We also report the results for solution of the continuous relaxation (RELAX); the convex optimization was solved using projected gradient descent, the projection being done with the code from (Davis et al., 2016).

5.3.1 Synthetic experiments: optimization comparison

We generated the experimental matrix $\boldsymbol{X}$ by sampling N vectors of size m from the multivariate Gaussian distribution of mean 0 and sparse precision $\boldsymbol{\Sigma}^{-1}$ (density d ranging from 0.3 to 0.9). Due to the runtime of Fedorov methods, results are reported for only one run; results averaged over multiple iterations (as well as for other distributions over $\boldsymbol{X}$) are provided in Appendix A.2.3.

As shown in Fig. 5-1, the greedy algorithm applied to the convex relaxation's support outperforms sampling from the convex relaxation solution, and does as well as the usual Fedorov algorithm UNIFFDV; GREEDYFDV marginally improves upon the greedy algorithm and UNIFFDV. Strikingly, GREEDY provides designs of comparable quality to UNIFFDV; furthermore, as very few local exchanges improve upon its design, running the Fedorov algorithm with GREEDY initialization is much faster (Table 5.1); this is confirmed by Table 5.2, which shows the number of experiments in common for different algorithms: GREEDY and GREEDYFDV only differ on very few elements. As the budget k increases, the difference in performances between SAMPLE, GREEDY and the continuous relaxation decreases, and the simpler SAMPLE algorithm becomes

competitive. Table 5.3 reports the support of the continuous relaxation solution for ESP-design with $\ell = 10$.

Table 5.1: Runtimes (s) ($\ell = 10$, $d = 0.6$)

k	40	80	120	160	200
GREEDY	2.8 10^1	2.7 10^1	3.1 10^1	4.0 10^1	5.2 10^1
GREEDYFDV	6.6 10^1	2.2 10^2	3.2 10^2	1.2 10^2	1.3 10^2
UNIFFDV	1.6 10^3	4.1 10^3	6.0 10^3	6.2 10^3	4.7 10^3

Table 5.2: Common items between solutions ($\ell = 10$, $d = 0.6$)

k	40	80	120	160	200
|GREEDY ∩ UNIFFDV|	26	76	114	155	200
|GREEDY ∩ GREEDYFDV|	40	78	117	160	200
|UNIFFDV ∩ GREEDYFDV|	26	75	113	155	200

Table 5.3: Support size of the continuous relaxation $\|z^*\|_0$ ($\ell = 10$, $d = 0.6$)

k	40	80	120	160	200
$d = 0.3$	93 ± 3	117 ± 3	148 ± 2	181 ± 3	213 ± 2
$d = 0.6$	92 ± 7	117 ± 4	145 ± 4	180 ± 3	214 ± 4
$d = 0.9$	88 ± 3	116 ± 3	147 ± 4	179 ± 3	214 ± 1

5.3.2 Real data

We used the Concrete Compressive Strength dataset (Yeh, 1998) (with column normalization) from the UCI repository to evaluate ESP-design on real data; this dataset consists in 1030 possible experiments to model concrete compressive strength as a linear combination of 8 physical parameters. In Figure 5-2 (a), OED chose k experiments to run to estimate θ, and we report the normalized prediction error on the remaining $N - k$ experiments. The best choice of OED for this problem is of course A-optimal design, which shows the smallest predictive error. In Figure 5-2 (b), we report the fraction of non-zero entries in the design matrix $\boldsymbol{X}_{S,:}$; higher values of ℓ correspond to increasing sparsity.

This confirms that OED allows us to scale between the extremes of A-optimal design and D-optimal design to tune desirable side-effects of the design; for example, sparsity in a design matrix can indicate not needing to tune a potentially expensive experimental parameter, which is instead left at its default value.

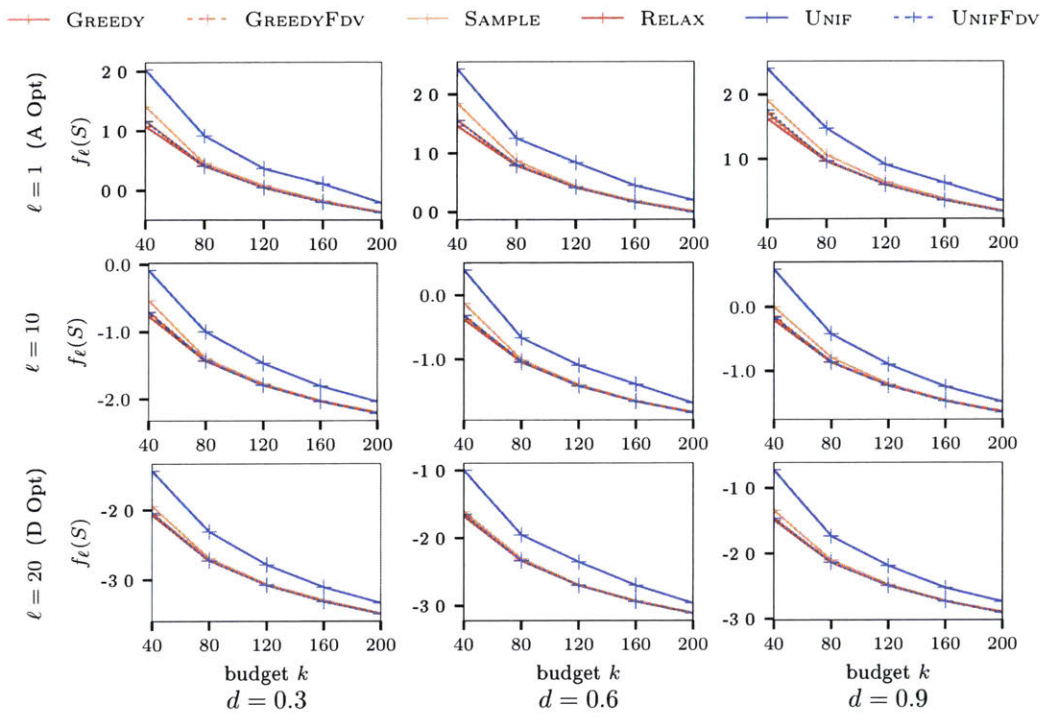
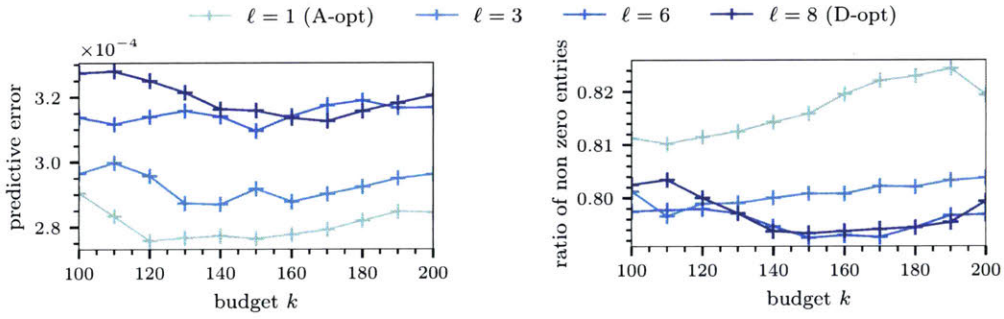


Figure 5-1: Synthetic experiments, $N = 500$, $m = 30$. The greedy algorithm performs as well as the classical Fedorov approach; as k increases, all designs except UNIF converge towards the continuous relaxation, making SAMPLE the best approach for large designs.



(a) MSE (b) Sparsity

Figure 5-2: Predicting concrete compressive strength via the greedy method; higher ℓ increases the sparsity of the design matrix $\boldsymbol{X}_{S,:}$, at the cost of marginally decreasing predictive performance.

5.4 Generalizing the MVCE problem

We close our discussion of GVS measures by discussing an important geometric problem related to ESP-design. Our motivation here is the dual to the continuous re-

laxation of D-optimal design: this is nothing but the well-known *Minimum Volume Covering Ellipsoid (MVCE)* problem, which is a problem of great interest to the optimization community (Todd, 2016).

Due to the importance of MVCE in optimization and statistics, investigating the dual form of the more general continuous ESP-design problem is a natural question which we investigate below.

5.4.1 GVS and Minimum Volume Covering Ellipsoids

Indeed, the dual of the natural convex relaxation to ESP-design generalizes the well-studied Minimum Volume Covering Ellipsoid (MVCE) optimization problem (Barnes, 1982; Rousseeuw and Leroy, 1987; Vandenberghe et al., 1998; Sun and Freund, 2004; Dolia et al., 2006; Todd, 2016), which we recover when $e_\ell \equiv \det$.

An ellipsoid $\mathcal{E}$ in $\mathbb{R}^m$ is defined by a matrix $\boldsymbol{Q} \succeq 0$ and a center c such that

$$\mathcal{E} = \{x \in \mathbb{R}^m \mid (x-c)^\top \boldsymbol{Q}(x-c) \leq 1\}.$$

The volume of an ellipsoid $\mathcal{E}$ is proportional[1] to $\det \boldsymbol{Q}^{-1/2}$. Hence, given N vectors $a_1, \ldots, a_N$, the Minimum Covering Ellipsoid (MVCE) problem is stated as follows: find $\boldsymbol{Q}_*, c_*$ such that

$$\boldsymbol{Q}_*, c_* \in \operatorname{argmin} \det \boldsymbol{Q}^{-1/2} \quad \text{with } \boldsymbol{Q} \succeq 0 \text{ and } \forall i, (a_i - c)^\top \boldsymbol{Q}(a_i - c) \leq 1.$$

In order to formulate MVCE as a convex problem, a standard transformation sets $\boldsymbol{M} = \boldsymbol{Q}^{1/2}$ and $z = \boldsymbol{M}c$, yielding the standard formulation of the MVCE problem:

$$(\boldsymbol{M}_*, z_*) \in \operatorname{argmax} \log\det \boldsymbol{M} \quad s.t. \quad \boldsymbol{M} \succeq 0 \text{ and } \forall i, (\boldsymbol{M}a_i - z)^\top (\boldsymbol{M}a_i - z) \leq 1. \tag{5.13}$$

It is well known that (5.13) is equivalent to the dual problem of continuous D-optimization relaxation, *i.e.*, the mode finding problem of a GVS measure with $\ell = m$.

[1]The proportionality factor is given by the volume of the unit ball in $\mathbb{R}^m$.

MVCE has numerous applications in outlier detection and robust statistics in general (Rousseeuw and Leroy, 1987; Rousseeuw and van Zomeren, 1990; Mount et al., 2000), image analysis (Comaniciu and Meer, 1997), clustering (Jolion et al., 1991), and robotics (Li et al., 2016a) among others. We believe that the dual problems presented herein may find similar applications.

5.4.2 Deriving the dual

With MVCE as our motivation, we develop the dual formulation of the continuous relation of ESP-design. We start by deriving $\nabla e_\ell(\boldsymbol{A})$, for which we recall that $e_\ell(\cdot)$ is a spectral function, whereby the spectral calculus of Lewis (1996) becomes applicable, saving us from intractable multilinear algebra (Jain, 2011). More precisely, say $U^\top \boldsymbol{\Lambda} \boldsymbol{U}$ is the eigendecomposition of $\boldsymbol{A}$, with $\boldsymbol{U}$ unitary. Then, as $e_\ell(\boldsymbol{A}) = e_\ell \circ \lambda(\boldsymbol{A})$, we have

$$\nabla e_\ell(\boldsymbol{A}) = \boldsymbol{U}^\top \operatorname{Diag}(\nabla e_\ell(\boldsymbol{\Lambda}))\boldsymbol{U} = \boldsymbol{U}^\top \operatorname{Diag}(e_{\ell-1}(\boldsymbol{\Lambda}^{-(i)}))\boldsymbol{U}.$$

We can now derive the dual of (5.8) as

$$\sup_{\boldsymbol{A}\succ 0, z\geq 0} \inf_{\mu\in\mathbb{R}, \boldsymbol{H}} -\tfrac{1}{\ell}\log e_\ell(\boldsymbol{A}) - \operatorname{tr}(\boldsymbol{H}(\boldsymbol{A}^{-1} - \boldsymbol{X}^\top \operatorname{Diag}(z)\boldsymbol{X})) - \mu(\mathbf{1}^\top z - k),$$

which admits as dual

$$\inf_{\mu\in\mathbb{R}, \boldsymbol{H}} \sup_{\boldsymbol{A}\succ 0, z\geq 0} \underbrace{-\tfrac{1}{\ell}\log e_\ell(\boldsymbol{A}) - \operatorname{tr}(\boldsymbol{H}\boldsymbol{A}^{-1})}_{g(\boldsymbol{A})} + \operatorname{tr}(\boldsymbol{H}\boldsymbol{X}^\top \operatorname{Diag}(z)\boldsymbol{X}) - \mu(\mathbf{1}^\top z - k). \quad (5.14)$$

It is easy to see that we must have $\boldsymbol{H} \succeq 0$: indeed, by contradiction, assume that there exists x such that $x^\top \boldsymbol{H} x < 0$ and $\|x\| = 1$. Then setting $\boldsymbol{A} = \boldsymbol{I} - \frac{t}{1+t} xx^\top$ has $g(\boldsymbol{A})$ go to infinity with t.

Before further analyzing (5.14), we first analyze $g(\boldsymbol{A})$.

Proposition 5.4.1. *Let function $g : \mathbb{S}^{++} \to \mathbb{R}$ be defined by* (5.14)*. Then, g attains its maximum on $\mathbb{S}^{++}$.*

Proof. This result follows from the fact that $g \to -\infty$ for $\|\boldsymbol{A}\| \to \infty$, $\boldsymbol{A} \to \partial\mathbb{S}^{++}$. □

Rewriting $\boldsymbol{A} = \boldsymbol{U}^\top \boldsymbol{\Lambda} \boldsymbol{U}$, we have

$$\nabla g(\boldsymbol{A}) = 0 \iff \boldsymbol{\Lambda} \operatorname{Diag}\left(e_{\ell-1}(\boldsymbol{\Lambda}_{(i)})\right)\boldsymbol{\Lambda} = e_\ell(\boldsymbol{\Lambda})\boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\top.$$

Let $a(\boldsymbol{H})$ be a matrix such that $\nabla g(a(\boldsymbol{H})) = 0$. Since f_ℓ is convex, $g(a(\boldsymbol{H})) = f_\ell^\star(-\boldsymbol{H})$ where $f_\ell^\star$ is the Fenchel conjugate of f_ℓ.

Furthermore, the previous equation shows that $\boldsymbol{H}$ and $a(\boldsymbol{H})$ are co-diagonalizable, with $\boldsymbol{\Lambda} \operatorname{Diag}(e_{\ell-1}(\boldsymbol{\Lambda}_{(i)}))\boldsymbol{\Lambda} = \operatorname{Diag}(h_1, \ldots, h_m)$. The eigenvalues of $a(\boldsymbol{H})$ must thus satisfy the system of equations

$$\lambda_i^2 e_{\ell-1}(\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_m) = h_i e_\ell(\lambda_1, \ldots, \lambda_m), \quad 1 \le i \le m.$$

Then,

$$\begin{aligned} e_\ell(\lambda_1, \ldots, \lambda_m) \operatorname{tr}(\boldsymbol{H} a(\boldsymbol{H})^{-1}) &= \operatorname{tr}\left(\boldsymbol{\Lambda} \operatorname{Diag}[e_{\ell-1}(\boldsymbol{\Lambda}^{-(i)})]\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-1}\right) \\ &= \sum_i \lambda_i e_{\ell-1}(\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_m) \\ &= \sum_i \sum_{J \subseteq [n], |J| = \ell, i \in J} \prod_{j \in J} \lambda_j. \end{aligned}$$

Each subset J of size ℓ appears ℓ times (once for each of its elements); finally

$$\operatorname{tr}(\boldsymbol{H}\,(a(\boldsymbol{H}))^{-1}) = \ell\; e_\ell(\lambda_1, \ldots, \lambda_m)^{-1} \sum_{J \subseteq [n], |J| = \ell} \prod_{j \in J} \lambda_j = \ell,$$

and hence

$$g(a(\boldsymbol{H})) = f_\ell^\star(-\boldsymbol{H}) = -\frac{1}{\ell} \log e_\ell(a(\boldsymbol{H})) - \ell.$$

Returning to the dual formulation (5.14), we can rewrite it as

$$\begin{aligned} &\inf_{\substack{\mu \in \mathbb{R}, \\ \boldsymbol{H} \in \mathbb{R}^{m \times m}}} \sup_{\boldsymbol{A} \succ 0, z \ge 0} -\frac{1}{\ell} \log e_\ell(\boldsymbol{A}) - \operatorname{tr}(\boldsymbol{H}\boldsymbol{A}^{-1}) + \operatorname{tr}(\boldsymbol{H}\boldsymbol{X}^\top \operatorname{Diag}(z)\boldsymbol{X}) - \mu(\mathbf{1}^\top z - k) \\ &\quad = \inf_{\mu \in \mathbb{R}, \boldsymbol{H} \succeq 0} \left[f_\ell^\star(-\boldsymbol{H}) + \sup_{z \ge 0} \operatorname{tr}(\boldsymbol{H}\boldsymbol{X}^\top \operatorname{Diag}(z)\boldsymbol{X}) - \mu(\mathbf{1}^\top z - k) \right] \end{aligned}$$

$$
\begin{aligned}
&= \inf_{\mu\in\mathbb{R},\boldsymbol{H}\succeq 0} \left[f_\ell^\star(-\boldsymbol{H}) + \sup_{z\geq 0} \sum_i z_i(x_i^\top \boldsymbol{H} x_i - \mu) + \mu k \right] \\
&= \inf_{\substack{x_i^\top \boldsymbol{H} x_i \leq \mu, \\ \boldsymbol{H}\succeq 0}} f_\ell^\star(-\boldsymbol{H}) + k\mu \\
&= \sup_{\substack{x_i^\top \boldsymbol{H} x_i \leq 1, \\ \boldsymbol{H}\succeq 0, \mu > 0}} -f_\ell^\star(-\mu\boldsymbol{H}) - k\mu \overset{\star}{=} \sup_{\substack{x_i^\top \boldsymbol{H} x_i \leq 1, \\ \boldsymbol{H}\succeq 0}} -f_\ell^\star(-\boldsymbol{H}) \\
&= \frac{1}{\ell} \sup_{\substack{x_i^\top \boldsymbol{H} x_i \leq 1, \\ \boldsymbol{H}\succeq 0}} \log e_\ell(a(\boldsymbol{H})) + \ell,
\end{aligned}
$$

where the eigenvalues $\boldsymbol{\Lambda}$ of $a(\boldsymbol{H})$ verify

$$
\lambda_i^2 \frac{e_{\ell-1}(\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_m)}{e_\ell(\lambda_1, \ldots, \lambda_m)} = h_i, \quad 1 \leq i \leq m. \tag{5.15}
$$

and $\overset{\star}{=}$ follows from $f_\ell^\star(-\mu\boldsymbol{H}) = \sup_{\boldsymbol{A}\succ 0} -\operatorname{tr}(\boldsymbol{H}\boldsymbol{A}) - f_\ell(\boldsymbol{A}/\mu) = f_\ell^\star(-\boldsymbol{H}) - \log\mu$.

In the general case, deriving $a(\boldsymbol{H})$ or even $e_\ell(a(\boldsymbol{H}))$ does not admit a closed form that we know of. However, we can gain some insight in the behavior of the dual by analyzing the behavior of the system of equations (5.15). Indeed, rewriting

$$
e_\ell(\lambda_1, \ldots, \lambda_m) = \lambda_i e_{\ell-1}(\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_m) + e_\ell(\lambda_1, \ldots, \lambda_{i-1}, 0, \lambda_{i+1}, \ldots, \lambda_m),
$$

we obtain the following formulations for (5.15):

$$
\lambda_i = h_i + \lambda_i \frac{e_\ell(\lambda_1, \ldots, 0, \ldots, \lambda_m)}{e_\ell(\lambda_1, \ldots, \lambda_m)} = h_i + g_i(h). \tag{5.16}
$$

Corollary 5.4.1.1. *Let g_i be defined by* (5.16). *Then for $\ell = m$, we have $g_i \equiv 0$ and so $a(\boldsymbol{H}) = \boldsymbol{H}$: we recover the MVCE problem.*

Corollary 5.4.1.2. *Let g_i be defined by* (5.16). *Then, for $\ell = 1$, we have $\lambda_i^2 / \operatorname{tr}(\boldsymbol{\Lambda}) = h_i$ and so $a(\boldsymbol{H}) = \operatorname{tr}(\boldsymbol{H}^{1/2})\boldsymbol{H}^{1/2}$.*

In general, we obtain an alternate form of the dual:

$$
\sup_{\substack{x_i^\top H x_i \leq 1, \\ H\succeq 0}} \log e_\ell\Big(h_i + g_i(h)\Big) \quad \text{such that the } \lambda_i \text{ satisfy (5.15)},
$$

where $g_i(h)$ is inversely tied to the impact a change away from 0 in the i-th coordinate has on e_ℓ; in the extreme case where $\ell = m$, only non-zero values of λ_i yield non-zero values of e_ℓ, and so the additional term g_i vanishes entirely from the dual formulation.

5.5 Open problems

- In deriving the Lagrangian dual of the optimization problem, we had to introduce the function $a(H)$; although $a(H)$ is known for $\ell = 1, m$, its form for other values of ℓ is unknown, making the dual form a purely theoretical object in the general case. Whether the closed form of a can be derived, or whether $E_\ell(a(H))$ can be obtained with only knowledge of H, remains an open problem.

- More practically, we showed experimentally (Fig. 5-2) that changing the degree ℓ of the elementary symmetric polynomial in the optimization problem (5.6) has straightforward consequences on the sparsity of the recovered designs; understanding this phenomenon theoretically, as well as investigating other properties that may scale similarly, are questions worthy of investigating.

Chapter 6

Exponentiated strongly Rayleigh measures

The previous chapters showed that strongly Rayleigh measures offer a tractable and theoretically grounded framework for models of diversity in machine learning. However, real-world problems cannot rely purely on negative dependence: consider once more a user querying the term "bow". We expect some amount of negative dependence, since a corresponding set of answers should allow for the diversity of interpretations of the term (archery, string instruments, etc.). However, answers that are irrelevant to the query — for example, bikes — reduce the quality of the answer set. In practice, we want the ability to temper the strength of the negative dependence.

Indeed, a practical challenge to using SR measures is that SR measures do not allow for intuitive tuning of the strength for the repulsive forces between similar items. This motivates us to consider a generalization of SR measures that enables an easy tuning of the relative strengths of quality and diversity considerations.

The closing contribution of this thesis is the introduction of *exponentiated strongly Rayleigh (ESR)* measures: distributions of the form $\nu(S) \propto \mu(S)^p$, where μ is an SR measure and $p > 0$ is a tunable parameter. A power $p > 1$ captures a sharper notion of diversity than μ; conversely, $p < 1$ allows for weaker diversity preferences. At the $p = 0$ extreme, ν is uniform, while for $p \to \infty$, ν concentrates at the mode of μ.

By introducing such a tunable parameter p that controls the strength of repulsive

forces between similar elements, we allow exponentiated strongly Rayleigh measures to capture a notion of diversity compatible with the particular requirements of machine learning subset selection problems.

ESR measures present an attractive generalization to SR measures, where a single parameter allows an intuitive regulation of desired strength of negative dependence. Interestingly, a few special cases of ESRs have been briefly noted in the literature (Zou and Adams, 2012; Kulesza and Taskar, 2012; Gillenwater et al., 2014), although only the guise of generalizations to DPPs and without noting any connection to SR measures. To our knowledge, there has been no previous discussion of ESR measures as a class. Nonetheless, they can benefit from the abundant existing theory for log-submodular models (Djolonga and Krause, 2014; Gotovos et al., 2015; Rebeschini and Karbasi, 2015; Djolonga et al., 2016), and isolated special cases have also been discussed in the literature. In particular, exponentiated DPPs (or E-DPPs) are mentioned in (Zou and Adams, 2012; Kulesza and Taskar, 2012), as well as in (Gillenwater et al., 2014) and (Anari and Oveis Gharan, 2017).

After analyzing the negative association properties of ESR measures, we derive general-purpose sampling algorithms that we further specialize to concrete cases. Subsequently, we evaluate ESR sampling procedures on outlier detection and kernel reconstruction, confirming that several machine learning problems benefit from the modeling power of ESR measures. This chapter is based on (Mariet et al., 2018).

6.1 Definition and negative dependence properties

We begin by formally introducing exponentiated strongly Rayleigh measures and analyzing their properties within the framework of negative dependence.

Definition 6.1.1 (Exponentiated SR measure)**.** A measure μ over $2^{[N]}$ is exponentiated strongly Rayleigh (ESR) if there exists an SR measure ν over $2^{[N]}$ and a power $p \geq 0$ such that $\mu(S) \propto \nu(S)^p$.

The parameter p serves to control the quality/diversity trade-off by sharpening ($p > 1$) or smoothing out ($p < 1$) the variations of the ground SR measure (see

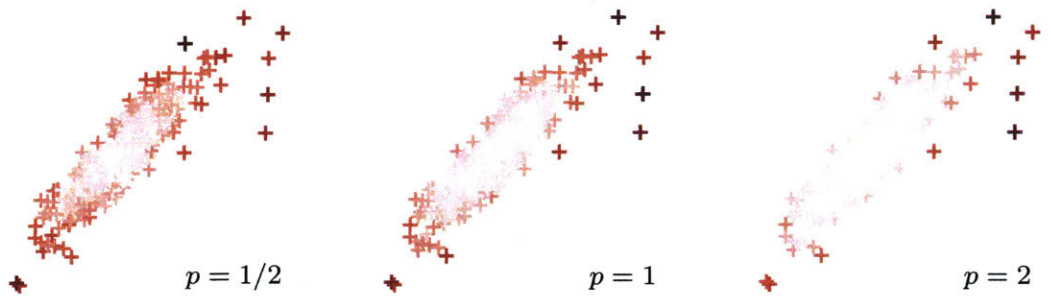


Figure 6-1: Anomaly detection by sampling with an exponentiated-DPP. 200 samples of size $k = 20$ were drawn from a E-DPP with Gaussian kernel; darker colors indicate higher sampling frequencies. As p increases, the points furthest from the mean accumulate all of the sampling probability mass.

Figure 6-1). A natural question is then to understand how this additional parameter impacts the negatively associated properties of the ESR. Recall that a fundamental property of SR measures lies in the fact that they are negatively associated: for two increasing functions F, G over $2^{[N]}$ that depend on a disjoint set of coordinates, an SR measure μ verifies the following inequality (Borcea et al., 2009, Theorem 4.9):

$$\mathbb{E}_\mu[F]\ \mathbb{E}_\mu[G] \geq \mathbb{E}_\mu[FG]. \tag{6.1}$$

Our first result states that the additional modularity enabled by the exponent parameter can break strong Rayleighness; as a consequence, we have no immediate guarantee that ESRs verify (6.1).

Proposition 6.1.1. *There exist ESR measures that are not SR.*

Proof. Recall (Borcea et al., 2009, Thm. 4.1) that a real bivariate affine polynomial $p(x, y)$ is stable if and only if

$$\partial_x p \partial_y p - p\, \partial_{xy} p \geq 0.$$

For an E-DPP with a kernel $\boldsymbol{L} \in \mathbb{R}^{2\times 2}$, this is exactly $l_{11}^p l_{22}^p \geq \det(\boldsymbol{L})^p$, which is clearly true for all $p \geq 0$ as $\boldsymbol{L}$ must be positive semi-definite.

However, stability does not hold in general. To obtain a counterexample, consider

the generating polynomial $p(x, y, z)$ for an E-DPP of dimension 3 (writing $d = \det(\boldsymbol{L})$ as a shorthand):

$$\begin{aligned} p(x, y, z) := d^p + l_{11}^p yz + l_{22}^p xz + l_{33}^p xy \\ + \det(\boldsymbol{L}[1,2])^p z + \det(\boldsymbol{L}[1,3])^p y + \det(\boldsymbol{L}[2,3])^p x + xyz. \end{aligned}$$

If $p(x, y, z)$ is SR, p must be stable through conditioning (Prop. 2.2.5); hence, $p(x, y, 1)$ must also be stable. Writing $d_{ij} = \det(\boldsymbol{L}_{\{i,j\}})$, this requires that

$$p(x, y, 1) = d^p + d_{12}^p + (d_{22}^p + d_{23}^p)x + (d_{11}^p + d_{13}^p)y + (d_{33}^p + 1)xy$$

be stable. Since $p(x, y, 1)$ is a real bivariate affine polynomial, we must then have

$$(d_{22}^p + d_{23}^p)(d_{11}^p + d_{13}^p) \geq (d^p + d_{12}^p)(d_{33}^p + 1).$$

Finally, one can verify that this last inequality is easily violated for several choices of (non-diagonal) positive semi-definite matrices. □

Conversely, some ESR measures remain SR for any p: if μ is a DPP parametrized by a block-diagonal kernel with 2×2 blocks, $\nu = \alpha\mu^p$ is also a DPP, and so SR and ESR by construction. The next theorem guarantees the existence of non-trivial ESR measures that are also SR.

Theorem 6.1.2. *There exists $\epsilon > 0$ such that $\forall p \in [1-\epsilon, 1+\epsilon], \forall N \in \mathbb{N}$, there exists a matrix $\boldsymbol{L} \in \mathbb{S}_N^+$, that is not block-diagonal with blocks of size 2×2, such that the* E-DPP *distribution defined by $\nu(S) \propto \det(\boldsymbol{L}[S,S])^p$ is SR.*

Proof. We write $d_S^i = \det(\boldsymbol{L}[S \cup \{i\}))^p$ and when possible $d_{ijk} = \det(\boldsymbol{L}[\{i,j,k\}])^p$.

Let $\boldsymbol{L} \succeq 0 \in \mathbb{R}^{n \times n}$. The associated E-DPP is SR if and only if $P_{ij}(z) \geq 0$ where $P_{ij}(z)$ is defined for any $1 \leq i \neq j \leq N$ as

$$P_r(z) = \frac{\partial f}{\partial z_i}(z)\frac{\partial f}{\partial z_j}(z) - f(z)\frac{\partial^2 f}{\partial z_i \partial z_j}(z)$$

$$= \Big(\sum_{S\in\mathcal{Y}'} d_S^i z^S\Big)\Big(\sum_{S\in\mathcal{Y}'} d_S^j z^S\Big) - \Big(\sum_{S\in\mathcal{Y}'} d_S^{ij} z^S\Big)\Big(\sum_{S\in\mathcal{Y}'} d_S z^S\Big),$$

where we write $\mathcal{Y}' = \{S \in [N], i \notin S, j \notin S\}$ and $z = (z_1, \ldots, z_n)$ where components z_i and z_j are removed. Now, choose $k \in [N] \setminus \{i, j\}$, and write $\tilde{z}$ for the vector z without component z_k. Write also $\mathcal{Y}'_k = \{S \in \mathcal{Y}', k \notin S\}$. Then,

$$\begin{aligned}
P_{ij}(z) =&(\sum_{k\notin S} d_S^i \tilde{z}^S + z_k \sum_{k\in S} d_S^i \tilde{z}^S)(\sum_{k\notin S} d_S^j \tilde{z}^S + z_k \sum_{k\in S} d_S^j \tilde{z}^S) \\
&- (\sum_{k\notin S} d_S^{ij} \tilde{z}^S + z_k \sum_{k\in S} d_S^{ij} \tilde{z}^S)(\sum_{k\notin S} d_S \tilde{z}^S + z_k \sum_{k\in S} d_S \tilde{z}^S) \\
=& \left(\sum_{S\in\mathcal{Y}'_k} d_S^{ik} \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S^{jk} \tilde{z}^S - \sum_{S\in\mathcal{Y}'_k} d_S^{ijk} \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S^k \tilde{z}^S\right) z_k^2 \\
&+ \left(\sum_{S\in\mathcal{Y}'_k} d_S^{ik} \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S^j \tilde{z}^S + \sum_{S\in\mathcal{Y}'_k} d_S^i \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S^{jk} \tilde{z}^S \right. \\
&\qquad \left. - \sum_{S\in\mathcal{Y}'_k} d_S^{ijk} \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S \tilde{z}^S - \sum_{S\in\mathcal{Y}'_k} d_S^{ij} \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S^k \tilde{z}^S\right) z_k \\
&+ \left(\sum_{S\in\mathcal{Y}'_k} d_S^i \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S^j \tilde{z}^S - \sum_{S\in\mathcal{Y}'_k} d_S^{ij} \tilde{z}^S \sum_{S\in\mathcal{Y}'_k} d_S \tilde{z}^S\right) \\
=& Az_k^2 + Bz_k + C.
\end{aligned}$$

Hence, $\boldsymbol{L}$ yields a strongly Rayleigh E-DPP measure if and only if the following inequalities hold for all i, j, k:

$$(B/2)^2 \leq AC; \quad A \geq 0; \quad C \geq 0. \tag{6.2}$$

When $n = 3$, (6.2) reduces to the arithmetic-geometric inequality:

$$\left(\frac{d_i d_{jk} + d_j d_{ik} - d_{ij} d_k - d_{ijk}}{2}\right)^2 \leq (d_i d_j - d_{ij})(d_{jk} d_{ik} - d_k d_{ijk}). \tag{6.3}$$

One can easily obtain 3×3 positive semi-definite matrices $\boldsymbol{L}$ that verify (6.3) strictly for $p = 1$; in particular, by continuity, there exists $\epsilon > 0$ such that the E-DPP

generated by the kernel $\boldsymbol{L}$ and power $p \in [1-\epsilon, 1+\epsilon]$ will still verify (6.3).

Then, as a block-diagonal matrix such that each diagonal block yields SR E-DPPs also yields a SR E-DPP, we can thus generate block-diagonal matrices of any size n such that the blocks are either $\boldsymbol{L}$ or 2×2 matrices, which all yield SR E-DPPs for $p \in [1-\epsilon, 1+\epsilon]$. □

Hence, ESRs are not guaranteed to be SR but may remain so. Due to their log-submodularity, they nonetheless will verify the so-called *negative lattice condition* $\mu(S \cap T)\mu(S \cup T) \leq \mu(S)\mu(T)$, and so retain negative dependence properties.

We now show that ESRs nonetheless have a fundamental advantage over standard log-submodular functions: although the intractability of their partition function precludes exact sampling algorithms, their closed form as the exponentiation of an SR measure can be leveraged to take advantage of the recent result (Li et al., 2016b) on fast-mixing Markov chains for SR measures.

6.2 Sampling from ESR measures

In the general case, the normalization term of an ESR is NP-hard to compute, precluding exact sampling algorithms. In this section, we propose instead two MCMC sampling algorithms whose key idea lies in exploiting the explicit relation ESR measures have to SR measures.

We begin by introducing the notion of r-closeness, which serves as a measure of the proximity between to distributions μ and ν over subsets. In practice, r-closeness will allow us to quantify how close an ESR measure is to being SR, and inform our bounds on mixing time.

Definition 6.2.1 (r-closeness)**.** Let μ, ν be measures over $2^{[N]}$ and let $p \geq 0$. We say that ν is *r-close* to μ (where we allow $r = \infty$) if we have for all $S \subseteq [N]$,

$$\nu(S) \neq 0 \text{ and } \mu(S) \neq 0 \implies r^{-1} \leq \nu(S)/\mu(S) \leq r.$$

We additionally write $r(\mu, \nu) = \min\{r \in \mathbb{R} \cup \{\infty\} : \nu \text{ is } r\text{-close to } \mu\}$.

Remark 6.2.1. r-closeness is a symmetric property.

Remark 6.2.2. If $r(\mu, \nu) < \infty$, ν is absolutely continuous with respect to μ: that is, $\mu(S) = 0 \implies \nu^p(S) = 0$.

The following result establishes that for any ESR measure ν, there exists an SR measure μ that is r-close to ν with $r < \infty$. This result is the cornerstone of the sampling algorithms we derive, as we show that we can use an r-close SR measure as proposal to efficiently sample from an ESR measure.

Proposition 6.2.1. *Let μ be a measure over $2^{[N]}$, and define ν to be the measure such that $\nu(S) \propto \mu(S)^p$. Then*

$$r(\mu, \nu) \leq \max_{S \in \text{supp}(\nu)} \left[\mu(S)^{-|p-1|}\right] < \infty.$$

Proof. Let μ, ν be as in the proposition statement, and consider $S \in \text{supp}(\nu)$. Recall that $\sum_T \nu(T) = 1$. By definition,

$$\frac{\nu(S)}{\mu(S)} = \frac{\mu(S)^p}{\mu(S) \sum_T \mu(T)^p}.$$

If $p \leq 1$, we have $\sum_T \mu(T)^p \geq 1$ and $\mu(S)^p \geq \mu(S)$, and so

$$(\min_S \mu(S))^{p-1} \leq \frac{\sum_T \mu(T)}{\sum_T \mu(T)^p} = \frac{1}{\sum_T \mu(T)^p} \leq \frac{\mu(S)^p}{\mu(S) \sum_T \mu(T)^p} \leq \mu(S)^{p-1} \leq \frac{1}{\min_S \mu(S)^{1-p}},$$

where the left inequality is obtained by noticing that $\frac{a+b}{c+d} \geq \min\left(\frac{a}{c}, \frac{b}{d}\right)$.

Similarly, for $p \geq 1$, we have $\sum_T \mu(T)^p \leq 1$ and $\mu(T)^p \leq \mu(T)$, and so

$$\min_S \mu(S)^{p-1} \leq \frac{\mu(S)^p}{\mu(S)} \leq \frac{\mu(S)^p}{\mu(S) \sum_T \mu(T)^p} \leq \frac{\sum_T \mu(T)}{\sum_T \mu(T)^p} \leq \max \mu(S)^{1-p} = \frac{1}{\min \mu(S)^{p-1}}.$$

□

In order to sample from an ESR distribution ν, we now generalize existing MCMC algorithms for SR measures; we bound the distance to stationarity of the the chain's

current state by comparing it to the distance to stationarity of a similar chain sampling from an SR measure μ, and leveraging the r-closeness $r(\mu, \nu)$.

6.2.1 Approximate samplers for ESR measures

Before investigating MCMC samplers, one may first wonder if rejection sampling might be sufficient: sample a set S from a proposal distribution μ, and accept with probability $\nu^p(S)/M\mu(S)$, where $M \geq \max_S \mu(S)/\nu^p(S)$. Unfortunately, the rejection sampling scaling factor M cannot be computed — although it can be bounded by $r(\mu, \nu^p)$ — leading us to prefer MCMC samplers (Andrieu et al., 2003).

We begin by analyzing the standard independent Metropolis-Hastings sampling algorithm (Metropolis et al., 1953; Hastings, 1970), using an SR measure μ as a proposal: we sample an initial set S from μ via a fast-mixing Markov chain, then iteratively swap from S to a new set S' with probability

$$\Pr(S \to S') = \min\left\{1, \frac{\nu(S')\mu(S)}{\nu(S)\mu(S')}\right\}.$$

Algorithm 13 Proposal-based sampling

Input: SR proposal μ, ESR measure ν and SR measure ρ s.t. $\nu = \alpha\rho^p$
Draw $S \sim \mu$
while not mixed **do**
  $S' \sim \mu$
  $S \leftarrow S'$ w.p. $\min\left\{1, \frac{\nu(S')\mu(S)}{\nu(S)\mu(S')}\right\} = \min\left\{1, \frac{\mu(S)}{\mu(S')}\left(\frac{\rho(S')}{\rho(S)}\right)^p\right\}$
return S

Algorithm 13 relies on the fact that we can compute $\nu(S')/\nu(S)$ as $(\rho(S')/\rho(S))^p$: we do not require knowledge of ν's partition function. This sampling method is valid as soon as ν is absolutely continuous with regards to the proposal μ; Proposition 6.2.1 guaranteed the existence of such measures.

If the ESR measure ν is k-homogeneous, we can instead sample from ν via Algorithm 14: we randomly sample $S \subseteq [N]$ and switch an element $u \in S$ for $v \notin S$ if this improves the probability of S.

The key to extending Algorithm 14 to non-homogeneous ESR measures is similar

Algorithm 14 Swap-chain sampling

Input: k-homogeneous ESR measure ν s.t. $\nu = \alpha\rho^p$ with ρ SR
Sample $S \sim \text{Unif}(N)$ such that $|S| = k$
while not mixed **do**
  Sample $u, v \in (S \times [N] \setminus S)$ uniformly at random
  $S \leftarrow S \cup \{u\} \setminus \{v\}$ w.p. $\min\left\{1, \frac{\nu(S\cup\{u\}\setminus\{v\})}{\nu(S)}\right\} = \min\left\{1, \left(\frac{\rho(S\cup\{u\}\setminus\{v\})}{\rho(S)}\right)^p\right\}$
return S

to the approach taken by Li et al. (2016b) for SR measures, and relies on leveraging the symmetric homogenization ν_{sh} of ν over $2^{[2n]}$, which we recall is defined by

$$\nu_{\text{sh}} : S \in 2^{[2n]} \to \begin{cases} \nu(S \cap [N]) \binom{n}{|S\cap[N]|}^{-1} & \text{if } |S| = n \\ 0 & \text{if } |S| \neq n. \end{cases}$$

If $\nu \propto \mu^p$, ν_{sh} is absolutely continuous with regards to μ_{sh}. A simple calculation further shows that $r(\mu_{\text{sh}}, \mu_{\text{sh}}) = r(\mu, \nu)$, and so to sample S from ν, it suffices to sample T of size n from ν_{sh} using Algorithm 14, and then output $S = T \cap [N]$.

Hence, although we cannot in the general case sample from an ESR measure exactly (unlike many SR measures), being able to evaluate an ESR measure's unnormalized density function allows us to leverage MCMC algorithms for approximate sampling. We now focus on bounding the mixing times of these algorithms.

6.2.2 Mixing time analysis

Writing $\nu'_{t,S}$ the distribution generated by a Markov chain sampler after t iterations and initialization set S, recall that the mixing time $\tau_S(\epsilon)$ measures the number of required iterations of the Markov chain so that $\nu'_{t,S}$ is close enough (in total variational distance) to the true ESR measure ν:

$$\tau_S(\epsilon) \triangleq \min\{t : \|\nu'_{t,S} - \nu\|_{\text{TV}} \leq \epsilon\}.$$

The mixing time of a chain depends on how close the distribution generating the initialization set S is to the target distribution μ. We now show this explicitly for the

two algorithms derived above, obtaining bounds on τ_S that directly depend on the r-closeness of the target ESR measure ν and an SR measure μ.[1]

For Alg. 13, mixing time explicitly depends on the quality of the proposal.

Theorem 6.2.2 (Algorithm 13 mixing time). *Let μ, ν be measures over $2^{[N]}$ such that μ is SR and ν is ESR. Sampling from ν via Algorithm 13 with μ as a proposal distribution has a mixing time $\tau(\epsilon)$ such that*

$$\tau_S(\epsilon) \leq 2r(\mu, \nu^p) \log \frac{1}{\epsilon}.$$

Proof. Alg. 13 has a state-independent proposal distribution μ, and hence its mixing time is governed by a ratio of probabilities: Cai (2000) showed that, after t iterations,

$$\max_{S,T} d_{\mathrm{TV}}(\nu_{(t)}(\cdot \mid S), \nu_{(t)}(\cdot \mid T)) = \left(1 - \frac{1}{\max_U \nu(U)/\mu(U)}\right)^t,$$

where $\nu_{(t)}(U \mid S)$ is the probability of being in state U after t iterations when starting from set S, and d_{TV} is the total variation distance.

Hence, following (Cai, 2000, Cor.1), we obtain

$$\tau_S(\epsilon) \leq 2 \max_U \frac{\nu(U)}{\mu(U)} \log \frac{1}{\epsilon} \leq 2r(\mu, \nu^p) \log \frac{1}{\epsilon}.$$

□

For the swapchain algorithm (Alg. 14), we derive a bound on the mixing time by comparing to a result by (Anari et al., 2016) which shows fast sampling for SR distributions over subsets of a fixed size.

Theorem 6.2.3 (Algorithm 14 mixing time). *Let ν be a k-homogeneous ESR measure over $2^{[N]}$. The mixing time for Alg. 14 is bounded in expectation by*

$$\tau_S(\epsilon) \leq \inf_{\mu \in SR} 2Nk\, r(\mu, \nu)^2 \log \tfrac{1}{\epsilon \nu(S)}.$$

[1] In (Anari et al., 2018), the authors independently showed that k-homogeneous ESR measures with $p < 1$ are *Strongly Log-Concave* (Anari et al., 2018, Theorem 1.7), and hence admit Markov Chains (Anari et al., 2018, Theorem 1.1) with $\tau_S(\epsilon) = \mathcal{O}(d \log \frac{1}{\epsilon \mu(S)})$.

Proof. This bound is based on a comparison method (Diaconis and Saloff-Coste, 1993), and relates the mixing time to the spectral gap. Let $1 = \mu_1 \geq \mu_2 \geq \ldots \geq -1$ be the eigenvalues of the state transition matrix of the chain. The *spectral gap* is $\gamma = 1 - \max\{|\mu|; \mu \text{ is an eigenvalue and } \mu \neq 1\}$. γ directly translates into a bound on the mixing time Diaconis and Stroock (1991):

$$\tau_S(\gamma) \leq \frac{1}{\gamma} \log\left(\tfrac{1}{\epsilon\nu(S)}\right).$$

The comparison method yields a bound on γ if we know a bound on $\tilde{\gamma}$ for a related chain with stationary distribution μ. Specifying (Diaconis and Saloff-Coste, 1993, Thm 2.1) to this case yields $\gamma \geq \tilde{\gamma}\alpha_1/\alpha_2$, where

$$\begin{aligned}
\alpha_1 &= \min_S \frac{\mu(S)}{\nu(S)} \geq \frac{1}{r(p)} \\
\alpha_2 &= \max_{T,U} \frac{\mu(T)}{\nu(T)} \frac{\min\{1, \mu(U)/\mu(T)\}}{\min\{1, \nu(U)/\nu(T)\}} \leq \max_T \frac{\mu(T)}{\nu(T)} \leq r(\mu, \nu).
\end{aligned}$$

Anari et al. (2016) show that $\tilde{\gamma} \geq \frac{1}{2Nk}$. Hence, we obtain

$$\tau_S(\gamma) \leq 2Nk \cdot r(\mu, \nu)^2 \cdot \log \tfrac{1}{\epsilon\nu(S)}.$$ □

The above bound depends on the *closest* SR distribution to the target measure ν. Combined with Prop. 6.2.1, Thm. 6.2.3 provides a simple upper bound to the mixing time of the swapchain algorithm.

Corollary 6.2.3.1 (Non-homogeneous swapchain mixing time)**.** *Let ν be a non-homogeneous ESR measure over $2^{[N]}$. The mixing time for the generalized swapchain sampler to sample from ν with initialization $S \subseteq [2N]$ is bounded in expectation by*

$$\tau_S(\epsilon) \leq \inf_{\mu \in \mathrm{SR}} 4N^2\, r(\mu, \nu)^2 \log \tfrac{1}{\epsilon\nu_{\mathrm{sh}}(S)}.$$

As a Markov chain's applicability strongly depends on its mixing time, finding an r-close SR distribution with small r is crucial to the applicability of ESR measures.

6.2.3 Some specific bounds

We now derive explicit mixing time bounds for ESR measures ν generated by two popular classes of SR measures: DPPs, in their usual form as well as their k-homogeneous form (k-DPPs), and Dual Volume Sampling (DVS). As Theorem 6.2.2 and Theorem 6.2.3 provide mixing time bounds that depend explicitly on $r(\mu, \nu)$, this section focuses on upper bounding $r(\mu, \nu)$. To the extent of our knowledge, the results below are the first for either of these two classes of ESR distributions.

Theorem 6.2.4 (E-DVS closeness bounds). *Let $n \geq k \geq m$ and let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ be a maximal-rank matrix. Let μ be the DVS distribution over $2^{[n]}$ for sets of size k:*

$$\mu : S \subseteq [N] \to \begin{cases} 0 & \text{if } |S| \neq k \\ \mu(S) \propto \det(\boldsymbol{X}[:, S]\boldsymbol{X}[:, S]^\top) & \text{if } |S| = k. \end{cases}$$

Let $p > 0$ and ν be the ESR measure induced by μ and p; let $\mathrm{MinVol}(\boldsymbol{X}, S)$ be the smallest non-zero minor of degree m of $\boldsymbol{X}[:, S]$. Then

$$r(\mu, \nu) \leq \binom{n-m}{k-m}^{|1-p|} \binom{k}{m}^{-|1-p|} \det(\boldsymbol{X}\boldsymbol{X}^\top)^{|1-p|} \mathrm{MinVol}(\boldsymbol{X}, S)^{-2|1-p|}.$$

Theorem 6.2.5 (E-DPP closeness bound). *Let μ be the distribution induced by a* DPP *with kernel $\boldsymbol{L} \succeq 0$ and ν be the E-*DPP *such that $\nu(S) \propto \det(\boldsymbol{L}[S])^p$. Let $\lambda_1 \leq \cdots \leq \lambda_N$ be the ordered eigenvalues of $\boldsymbol{L}$. Then,*

$$r(\mu, \nu^p) \leq \prod_{i=1}^{N} (1 + \lambda_i)^{|1-p|} \prod_{\lambda_i < 1} \lambda_i^{-|1-p|}.$$

Theorem 6.2.6 (E-k-DPP closeness bound). *Let μ be the distribution over $2^{[N]}$ induced by a k-*DPP *($k \leq N$) with kernel $\boldsymbol{L}$, and let ν be the induced ESR measure with power $p > 0$. Then*

$$r(\mu, \nu) \leq e_k(\lambda_1, \ldots, \lambda_N)^{|1-p|} \prod_{i=1}^{k} \lambda_i^{-|1-p|}.$$

where e_k the k-th elementary symmetric polynomial.

One easily shows that the values $r(\mu, \nu)$ we derive above for (k-) DPPs are loosely lower-bounded by $\kappa^{|1-p|}$, where κ is the condition number of the kernel matrix $\boldsymbol{L}$. However, it is possible to obtain a closer SR distribution to $\nu \propto \det(\boldsymbol{L})^p$ than the baseline choice of the DPP with kernel $\boldsymbol{L}$: indeed, as $\boldsymbol{L}$ is positive semi-definite, we can also consider a DPP parametrized by kernel $\boldsymbol{L}^p$.

For the rest of this section, we define μ as the SR measure corresponding to the DPP with kernel $\boldsymbol{L}^p$:

$$\mu(S) = \det(\boldsymbol{L}^p[S])/\det(\boldsymbol{I} + \boldsymbol{L}^p),$$

and ν as the ESR measure such that $\mu(S) \propto \det(\boldsymbol{L}[S])^p$, which remains absolutely continuous with regard to μ.

In this setting, upper bounding $r(\mu, \nu)$ proves to be significantly more difficult, and is the focus of the remainder of this section. We first recall a useful expansion of the determinant of principal submatrices, fundamental to deriving the bounds below and potentially of more general interest.

Lemma 6.2.7 (Shirai and Takahashi (2003, Lemma 2.9)). *Let $\boldsymbol{H}$ be an $N \times N$ Hermitian matrix with eigenvalues $\lambda_1, \ldots, \lambda_N$. There exists a $2^N \times 2^N$ symmetric doubly stochastic matrix $\boldsymbol{Q} = [\boldsymbol{Q}_{SJ}]$ indexed by subsets S, J of $[N]$ such that*

$$\det(\boldsymbol{H}[S]) = \sum\nolimits_{J \subseteq [N], |J|=|S|} \boldsymbol{Q}_{SJ} \prod\nolimits_{i \in J} \lambda_i.$$

$\boldsymbol{Q}$ can be chosen to depend only on the eigenvectors of $\boldsymbol{H}$ and to satisfy $\boldsymbol{Q}_{S \neq J} = 0$.

The above lemma allows us to bound the ratio $\det(\boldsymbol{L}^p[S])/\det(\boldsymbol{L}[S]^p)$ in terms of the *generalized condition number* of $\boldsymbol{L}$.

Definition 6.2.2 (Generalized condition number). Given a matrix $\boldsymbol{L} \in \mathbb{S}_N^{++}$ with eigenvalues $\lambda_1, \ldots, \lambda_N$, we define its generalized condition number of order k as

$$\kappa_k = (\lambda_1 \cdots \lambda_k)(\lambda_n \cdots \lambda_{N-k})^{-1}.$$

Note that κ_k is the usual condition number of the k-th exterior power $\boldsymbol{L}^{\wedge k}$ (and so, in particular, $\kappa_k \leq \kappa^k$).

Given the generalized conditioned number, Lemma 6.2.7 combined with the power-mean inequality (Specht, 1960) (Appendix A.1.2) suffices to bound the gap between volumes generated by E-DPPs and DPPs:

Theorem 6.2.8. *Let μ be the distribution induced by a* DPP *with kernel $\boldsymbol{L}^p$, and ν be the corresponding E-*DPP *such that $\nu \propto \det(\boldsymbol{L}[S])^p$. Then $r(\mu, \nu) \leq r(\kappa_{\lfloor n/2 \rfloor}, p)$ where $r(\kappa, p)$ is defined by*

$$r(\kappa, p) = \begin{cases} \left(\frac{p(\kappa-1)}{\kappa^p-1}\right)^p \left(\frac{(1-p)(\kappa-1)}{\kappa-\kappa^p}\right)^{1-p} & \text{for } 0 < p < 1 \\ \left(\frac{\kappa^p-1}{p(\kappa-1)}\right)^p \left(\frac{(p-1)(\kappa-1)}{\kappa^p-\kappa}\right)^{p-1} & \text{for } p > 1. \end{cases}$$

Proof. We show the result for general DPPs; the result for k-DPPs follows the same exact reasoning.

For $0 < p < 1$, it follows from Thm. A.1.2 that $\det(\boldsymbol{L}[S])^p \geq \det(\boldsymbol{L}^p[S])$. Hence,

$$Z_p = \sum_{S \subseteq [N]} \det(\boldsymbol{L}[S])^p \geq \sum_{S \subseteq [N]} \det(\boldsymbol{L}^p[S]) = \det(\boldsymbol{I} + \boldsymbol{L}^p),$$

which entails $\frac{\det(\boldsymbol{I}+\boldsymbol{L}^p)}{Z_p} \leq 1$ whereby it remains to bound $\frac{\det(\boldsymbol{L}[S])^p}{\det(\boldsymbol{L}^p[S])}$.

Let $S \subseteq [N]$ of size k, and let λ be the vector of $\boldsymbol{L}$'s eigenvalues. We write $\lambda^S = \prod_{i \in S} \lambda_i$, and denote by $\boldsymbol{\lambda}^{\wedge k}$ the $\binom{N}{k}$-vector $(\lambda^S)_{S \subseteq [N], |S|=k}$. Using Lemma 6.2.7, there exists $\boldsymbol{w} \in \mathbb{R}^{\binom{N}{k}}$ that sums to 1 such that

$$\frac{\det(\boldsymbol{L}[S])^p}{\det(\boldsymbol{L}^p[S])} = \frac{(\sum_{|S|=k} w_S \lambda^S)^p}{\sum_{|S|=k} w_S (\lambda^p)^S} = \left(\frac{M_1(\boldsymbol{w}; \boldsymbol{\lambda}^{\wedge k})}{M_p(\boldsymbol{w}; \boldsymbol{\lambda}^{\wedge k})}\right)^p \leq r(p)^p.$$

Where the last inequality follows from Thm. A.1.1. To lower bound $r_p(S)$, the same reasoning gives us $\frac{\det(\boldsymbol{L}[S]^p)}{\det(\boldsymbol{L}^p[S])} \geq 1$ and hence

$$r_p(S) \geq \frac{\det(\boldsymbol{I} + \boldsymbol{L}^p)}{Z_p} \geq \min_S \frac{\det \boldsymbol{L}^p[S]}{(\det \boldsymbol{L}[S])^p} \geq \left(\frac{M_p(\boldsymbol{w}'; \boldsymbol{\lambda}^{\wedge k})}{M_1(\boldsymbol{w}'; \boldsymbol{\lambda}^{\wedge k})}\right)^p \geq r(p)^{-p},$$

where the second inequality follows $\frac{a+b}{c+d} \geq \min(\frac{a}{c}, \frac{b}{d})$. The same reasoning yields the desired result for $p > 1$.

Finally, some algebra shows that for fixed p, r is an increasing function of κ, and so $r(\kappa_k, p)$ is upper bounded by $r(\kappa_{\lfloor N/2 \rfloor})$. □

Corollary 6.2.8.1. *Let μ be the distribution induced by a k-DPP with kernel $\boldsymbol{L}^p$, and ν be the ESR measure such that $\nu(S) \propto \det(\boldsymbol{L}[S])^p$. Then $r(\mu, \nu) \leq r(\kappa_k, p)$.*

As shown in Figure A-4 (Appendix A.2.4), the upper bound 6.2.8 grows slower than κ: this shows that $\mu(S) \propto \det(\boldsymbol{L}^p[S])$ is a closer SR distribution to an E-DPP with kernel $\boldsymbol{L}$ than the usual DPP with kernel $\boldsymbol{L}$, yielding finer mixing time bounds.

Note that the per-iteration complexity of both algorithms must also be taken into account when choosing a sampling procedure: for E-DPPs, despite Algorithm 13's smaller mixing time, Algorithm 14 is more efficient in cases when N large due to the comparative costs of each sampling round.

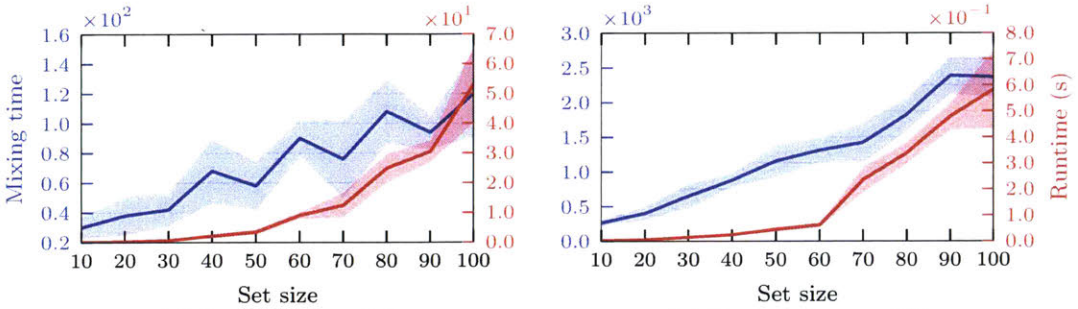
6.3 Experiments

To evaluate the empirical applications of ESR measures, we evaluate E-DPPs on a variety of machine learning task. In all cases where we use the proposal MCMC sampler (Alg. 13), we use the DPP with kernel $\boldsymbol{L}^p$ as a proposal.

6.3.1 Evaluating mixing time

We begin by empirically evaluating the mixing time of both algorithms. We measure mixing using the classical Potential Scale Reduction Factor (PSRF) metric (Brooks and Gelman, 1998). As the PSRF converges to 1, the chain mixes. In the following experiments, we report the mixing time (number of iterations) necessary to reach a PSRF of 1.05, as well as the runtime (in seconds) to convergence, averaged over 5 iterations; we use matrices with a fixed κ^k across all mixing time experiments.

The mixing time for proposal-based sampling is an order of magnitude smaller than swap-chain sampling; this is in line with the bounds we provide in Theorems 6.2.2



(a) Proposal, $r = 2.0$ (b) Swapchain, $r = 2.0$

Figure 6-2: Mixing and sampling time for E-k-DPPs as a function of the set size k. In both cases, the mixing time grows linearly with k; although the mixing time for the proposal algorithm is an order of magnitude smaller than for the swapchain algorithm, the latter samples faster due to the per-iteration cost of each transition step.

and 6.2.3. However, this does not translate into faster runtimes: indeed, the per-iteration complexity of proposal-based sampling is significantly higher than for the swapchain algorithm, as Alg. 13 samples from a DPP at each iteration. The evolution of mixing and wall clock times as a function of N is provided in Appendix A.2.4.

6.3.2 Anomaly detection

We now focus on applications for E-DPPs; we begin by evaluating the use of E-DPPs for outlier detection. As increasing p heightens the model's sensitivity to diversity, we expect $p > 1$ to provide better outlier detection. To our knowledge, this is the first application of DPPs to outlier detection, and so our goal for this experiment is not to improve upon state-of-the-art results, but to compare the performance of (E-)DPPs for various values of p to standard outlier detection algorithms.

Experimentally, we detect an outlier via the following approach: given a dataset of n points and an E-DPP with an RBF kernel built from the data (bandwidth $\beta = 100$), we sample $\frac{n}{5}$ subsets of size 50 and report as outliers points that appear at least $n\varepsilon$ times, where ε is a tunable parameter (hence, if we were doing uniform sampling, each point in the dataset would be sampled on average 10 times).

We detect outliers on three public datasets: the UCI Breast Cancer Wisconsin dataset (Street et al., 1993) modified as in (Goldstein and Uchida, 2016; Kriegel et al., 2009) as well as the Letter and Speech datasets from (Micenková et al., 2014).

We also report the performance of a selection of standard outlier detection algorithms whose reported performance in (Goldstein and Uchida, 2016) is competitive with other outlier detection algorithms: Local Outlier Factor (LOF) (Breunig et al., 2000), k-Nearest Neighbor (k-NN) (Angiulli and Pizzuti, 2002), Histogram-based Outlier Score (HBOS) (Goldstein and Dengel, 2012), Local Outlier Probability (LoOP) (Kriegel et al., 2009) and unweighted Cluster-Based Local Outlier Factor (uCBLOF) (Amer and Goldstein, 2012; Goldstein and Uchida, 2016).

Method	Cancer	Letter	Speech
E-DPP ($p = 0.5$)	0.952 ± 0.018	0.780 ± 0.013	0.455 ± 0.007
DPP ($p = 1$)	0.962 ± 0.004	0.820 ± 0.003	0.439 ± 0.011
E-DPP ($p = 2$)	0.965 ± 0.001	0.847 ± 0.002	0.445 ± 0.002
LOF*	0.982 ± 0.002	0.867 ± 0.027	0.504 ± 0.022
k-NN*	0.979 ± 0.001	0.872 ± 0.018	0.497 ± 0.010
HBOS*	0.983 ± 0.002	0.622 ± 0.007	0.471 ± 0.003
LoOP*	0.973 ± 0.012	0.907 ± 0.008	0.535 ± 0.034
uCBLOF*	0.950 ± 0.039	0.819 ± 0.023	0.469 ± 0.003

Table 6.1: AUC (mean + standard deviation) for E-DPPs and standard outlier detection algorithms. As expected, we see that a higher exponent leads to a stronger preference for diversity and hence a better outlier detection scheme. Only LoOP and LOF consistently outperform E-DPPs.

Results are reported in Table 6.1; as expected, we see that larger values of p ($p = 2$) are more sensitive to outliers, and provide better models for outlier detection.

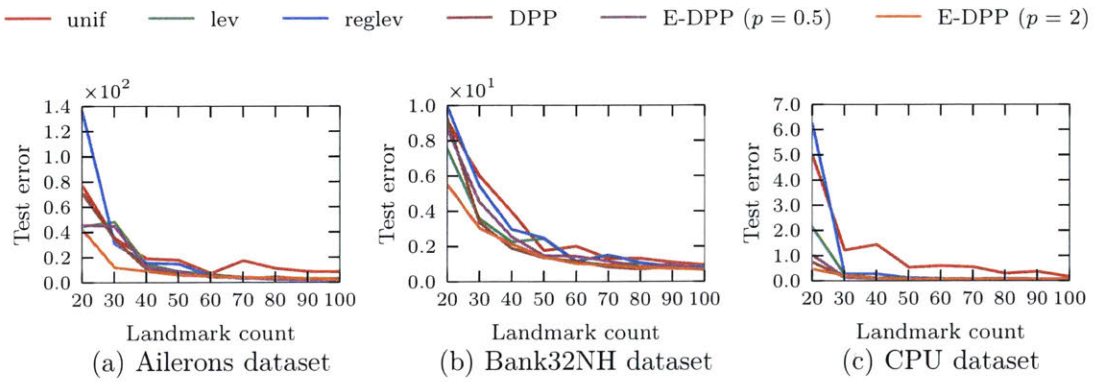
6.3.3 E-Dpps for the Nyström method

As a more standard application of DPPs, we now investigate the use of E-DPPs for kernel reconstruction via the Nyström method (Nyström, 1930; Williams and Seeger, 2001). Given a large kernel $\boldsymbol{K}$, recall that the Nyström method selects a subset C of columns ("landmarks") of $\boldsymbol{K}$ and approximates $\boldsymbol{K}$ as $\boldsymbol{K}_{:,C}\boldsymbol{K}_C^{\dagger}\boldsymbol{K}_{C,:}$. We show that E-DPPs improve upon the recent results of (Li et al., 2016c) for kernel reconstruction.

We apply Kernel Ridge Regression to 3 regression datasets: Ailerons, Bank32NH, and Machine CPU.[2] We subsample 4,000 points from each dataset (3,000 training and

[2] `http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

1,000 test) and use an RBF kernel and choose the bandwidth β and regularization parameter λ for each dataset by 10-fold cross-validation. Results are averaged over 3 random subsets of data, using the swapchain sampler initialized with k-means++ and run for 3000 iterations.



(a) Ailerons dataset (b) Bank32NH dataset (c) CPU dataset

Figure 6-3: Prediction error on regression datasets; we compare various E-DPP models to uniform sampling ("unif") as well as leverage and regularized leveraged sampling ("lev" and "reglev"). On all datasets, the E-DPPs achieve the lowest error, with the largest exponent $p = 2$ performing markedly better than other methods.

We evaluate the quality of the sampler via the prediction error on the held-out test set. Figure 6-3 reports the results. Consistently across all datasets, $p = 2$ outperforms all other samplers in terms of the prediction error, in particular when only sampling a few landmarks. Interestingly, we also see that the reconstruction error tends to be smaller when $p = \frac{1}{2}$ (see Appendix A.2.4).

6.4 Open problems

- We believe that further specifying the class of ESR measures that also remain SR will provide valuable insight into the study of negatively associated measures.

- Learning the function and the exponent jointly: given an SR measure μ and a collection of *i.i.d.* subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$, the MLE problem that finds the

best $p > 0$ to model $\mathcal{S}$ as being sampled from an ESR $\nu \propto \mu^p$ is convex:

$$\operatorname*{argmax}_{p>0} \frac{p}{m} \sum_{k=1}^{m} \log \mu(S_i) - \log \Big(\sum\nolimits_{S \subseteq [N]} \mu(S)^p \Big). \tag{6.4}$$

As such, standard convex optimization algorithms can be leveraged to select p, potentially after leaning a parametrization of μ. However, there may be advantages to learning p and μ jointly, an approach which remain to be investigated.

Chapter 7

Conclusion

With the goal of introducing negative dependence as a powerful framework to model machine learning problems that require diverse sampling, this thesis focused on two specific problems. First, we derived efficient algorithms to learn and sample from negatively dependent measures, expanding their range of applications to real-world settings previously too computationally expensive to consider.

Then, we focused on the theory of negatively dependent measures itself. We introduced GVS measures, strongly Rayleigh measures with key implications for several classical problems in machine learning and optimizations. Finally, we introduced exponentiated strongly Rayleigh measures: a class of negatively dependent measures parametrized by a scalar controlling the strength of repulsive forces between similar elements. With these contributions, we showed that negatively dependent measures can provide a fine-grained tool to model difficult machine learning problems.

We now conclude this thesis by re-framing it in the broader context of modern machine learning problems: how can we leverage the theory of negative dependence for principled machine learning model design and optimization?

7.1 High-level summary

This thesis has shown that the framework of negative dependence provides a unifying point of view for the broad class of machine learning optimization problems that —

at their core — seek diversity in their solutions. Within this framework, we derived theoretical results and general purpose algorithms for problems ranging from optimal design and recommender systems to outlier detection and kernel reconstruction.

The first part of this thesis focused on determinantal point processes, popular negatively dependent measures in machine learning. In Chapter 3 we showed that DPPs — as well as any other (log) super/submodular set function — can be approximated efficiently with neural networks. We obtained a DPP-specific neural architecture that efficiently approximates DPPs, both in sampling and mode-finding tasks. In Chapter 4, we scaled up DPP learning and sampling algorithms by leveraging Kronecker products, and introduced Contrastive Estimation, an efficient way of improving the DPP learning procedure by including information about unlikely subsets.

Chapter 5 introduced Generalized Volume Sampling measures: distributions over subsets based on elementary symmetric polynomials that extend volume sampling and remain strongly Rayleigh. We connected GVS measures to a generalization of classical problems in optimal experimental design, and developed randomized and greedy algorithms to solve the ensuing optimization problem. We obtained new results on the convexity properties of elementary symmetric polynomials, and derived a generalization of the classic Minimum Volume Covering Ellipsoid problem.

Finally, we introduced in Chapter 6 exponentiated strongly Rayleigh measures: a new class of negatively dependent measures that are well-suited to the requirements of machine learning applications. Indeed, exponentiated SR measures enjoy fast sampling and learning algorithms, while offering simple a simple tuning procedure for the strength of the measures' repulsion forces.

7.2 Open problems

Each chapter included a summary of open research problems that we believe to be worth investigating. We conclude here by discussing some over-arching open problems of import to the machine learning community.

Negative dependence theory. Pemantle (2000) and Borcea et al. (2009) provided a variety of open conjectures regarding the chain of implications between the various characterizations of negative dependence that one may wish to use. By introducing exponentiated strongly Rayleigh measures, we have added to this list; in particular by raising the question of which class of measures will remain strongly Rayleigh when exponentiated. Furthermore, recent work by Anari et al. (2018) considers exponentiated-DPPs with exponents $p \leq 1$ within the framework of *Complete Log-Concavity*, wherein the authors also derive weaker characterizations of negative dependence in this scenario, further confirming that the class of exponentiated strongly Rayleigh measures provides a valuable framework to develop and extend the rich theory of negative dependence.

Negative dependence for machine learning optimization. The scope of potential applications of negative dependence extends far beyond the examples discussed in this thesis. Zhang et al. (2017) showed that DPPs can provide variance reduction for minibatch SGD, a topic we believes warrants further study within the larger scope of negative dependence — indeed, minibatch SGD is in itself a particular example of negative dependence as we sample without replacement. Any improvements to the theory or speed of minibatch SGD will undeniably impact the fundamental underpinning of most machine learning applications. Furthermore, a less strict notion of negative dependence provides a framework to discuss trade-offs between exploration (diversity) and exploitation (quality), which occur — to mention only a few examples — in reinforcement learning and hyperparameter space exploration.

Bibliography

R. Affandi, A. Kulesza, E. Fox, and B. Taskar. Nyström approximation for large-scale Determinantal Point Processes. In *Proc. Int. Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

R. Affandi, E. Fox, R. Adams, and B. Taskar. Learning the parameters of Determinantal Point Process kernels. In *Proc. Int. Conference on Machine Learning (ICML)*, 2014.

M. Amer and M. Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *RapidMiner Community Meeting and Conference (RCOMM)*. Shaker Verlag GmbH, 2012.

N. Anari and S. O. Gharan. The Kadison-Singer problem for Strongly Rayleigh measures and applications to asymmetric TSP. *CoRR*, abs/1412.1143, 2014.

N. Anari and S. Oveis Gharan. A generalization of permanent inequalities and applications in counting and optimization. In *Symposium on Theory of Computing (STOC)*. ACM, 2017.

N. Anari, S. O. Gharan, and A. Rezaei. Monte Carlo Markov Chain algorithms for sampling strongly Rayleigh distributions and Determinantal Point Processes. In *Conference on Learning Theory (COLT)*, 2016.

N. Anari, K. Liu, S. O. Gharan, and C. Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. *CoRR*, abs/1811.01816, 2018.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1):5–43, Jan 2003.

F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, London, UK, UK, 2002. Springer-Verlag.

H. Avron and C. Boutsidis. Faster subset selection for matrices and applications. *SIAM J. Matrix Analysis Applications*, 34(4), 2013.

F. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3, 2002.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Int. Conf. on Learning Representations (ICLR)*, 2015.

E. R. Barnes. An algorithm for separating patterns by ellipsoids. *IBM Journal of Research and Development*, 26, 1982.

N. K. Batmanghelich, G. Quon, A. Kulesza, M. Kellis, P. Golland, and L. Bornn. Diversifying sparsity using variational Determinantal Point Processes. *CoRR*, abs/1411.6307, 2014.

H. H. Bauschke, O. Güler, A. S. Lewis, and H. S. Sendov. Hyperbolic polynomials and convex analysis. *Canad. J. Math.*, 53(3), 2001.

R. Bhatia. *Matrix Analysis.* Springer, 1997.

R. Bhatia. *Positive Definite Matrices.* Princeton University Press, 2007.

J. Borcea and P. Brändén. Applications of stable polynomials to mixed determinants: Johnson's conjectures, unimodality, and symmetrized Fischer products. *Duke Mathematical Journal*, 143(2), 2008.

J. Borcea and P. Brändén. The Lee-Yang and Pólya-Schur programs I Linear operators preserving stability. *Inventiones mathematicae*, 177(3), 2009.

J. Borcea and P. Brändén. Multivariate Pólya–Schur classification problems in the Weyl algebra. *Proceedings of the London Mathematical Society*, 101(1), 2010.

J. Borcea, P. Brändén, and T. Liggett. Negative dependence and the geometry of polynomials. *Journal of the American Mathematical Society*, 22(2), 2009.

A. Borodin. Determinantal Point Processes. *arXiv:0911.1153*, 2009.

A. Bose, H. Ling, and Y. Cao. Adversarial contrastive estimation. *arXiv preprint arXiv:1805.03642*, 2018.

L. Bottou. On-line learning and stochastic approximations. In *On-line Learning in Neural Networks*, New York, NY, USA, 1998.

P. Brändén. Polynomials with the half-plane property and matroid theory. *Advances in Mathematics*, 216(1), 2007.

P. Brändén, J. Haglund, M. Visontai, and D. G. Wagner. *Proof of the Monotone Column Permanent Conjecture*, pages 63–78. Springer Basel, 2011.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2), May 2000.

T. Brijs. Retail market basket data set. In *Workshop on Frequent Itemset Mining Implementations (FIMI03)*, 2003.

T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *SIGKDD*. ACM, 1999.

S. P. Brooks and A. Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7, 1998.

N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2012.

H. Cai. Exact bound for the convergence of metropolis chains. *Stochastic Analysis and Applications*, 18(1), 2000.

O. Canvet and F. Fleuret. Efficient Sample Mining for Object Detection. In *ACML*, JMLR: Workshop and Conference Proceedings, 2014.

W. Chao, B. Gong, K. Grauman, and F. Sha. Large-margin Determinantal Point Processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2015.

D. Chen. Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing and Customer Strategy Management*, 19(3), August 2012.

L. Chen, G. Zhang, and E. Zhou. Fast greedy MAP inference for Determinantal Point Process to improve recommendation diversity. In *Advances in Neural Information Processing Systems*, 2018.

L. Chen, F. Yuan, J. M. Jose, and W. Zhang. Improving negative sampling for word representation using self-embedded features. *CoRR*, abs/1710.09805, 2017.

Y.-B. Choe, J. G. Oxley, A. D. Sokal, and D. G. Wagner. Homogeneous multivariate polynomials with the half-plane property. *Advances in Applied Mathematics*, 32(1), 2004. Special Issue on the Tutte Polynomial.

D. A. Cohn. Neural network exploration using optimal experiment design. *Advances in Neural Information Processing Systems*, 1994.

D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1997.

R. Cominetti, W. F. Mascarenhas, and P. J. S. Silva. A Newton's method for the continuous quadratic knapsack problem. *Mathematical Programming Computation*, 6(2), 2014.

A. Cotter, M. R. Gupta, H. Jiang, J. Muller, T. Narayan, S. Wang, and T. Zhu. Interpretable set functions. *CoRR*, abs/1806.00050, 2018.

T. A. Davis, W. W. Hager, and J. T. Hungerford. An efficient hybrid algorithm for the separable convex quadratic knapsack problem. *ACM Trans. Math. Softw.*, 42 (3), May 2016.

L. Decreusefond, I. Flint, N. Privault, and G. L. Torrisi. Determinantal Point Processes, 2015.

P. Diaconis and L. Saloff-Coste. Comparison theorems for reversible Markov chains. *The Annals of Applied Probability*, 3(3), 1993.

P. Diaconis and D. Stroock. Geometric bounds for eingenvalues of Markov Chains. *Annals of Applied Probability*, 1991.

J. Djolonga and A. Krause. From MAP to marginals: Variational inference in Bayesian submodular models. In *Advances in Neural Information Processing Systems*, 2014.

J. Djolonga, S. Tschiatschek, and A. Krause. Variational inference in mixed probabilistic submodular models. In *Advances in Neural Information Processing Systems*, 2016.

A. N. Dolia, N. M. White, and C. J. Harris. D-optimality for minimum volume ellipsoid with outliers. In *In Proceedings of the Seventh International Conference on Signal/Image Processing and Pattern Recognition, (UkrOBRAZ2004*, 2004.

A. N. Dolia, T. De Bie, C. J. Harris, J. Shawe-Taylor, and D. M. Titterington. The minimum volume covering ellipsoid estimation in kernel-defined feature spaces. In *European Conference on Machine Learning*, 2006.

J. L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

S. Flaxman, A. Wilson, D. Neill, H. Nickisch, and A. Smola. Fast Kronecker inference in Gaussian processes with non-Gaussian likelihoods. In *Proc. Int. Conference on Machine Learning (ICML)*, 2015.

C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Comm. Math. Phys.*, 22(2), 1971.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, January 2004.

L. Gårding. An inequality for hyperbolic polynomials. *Journal of Mathematics and Mechanics*, 8(6), 1959.

M. Gartrell, U. Paquet, and N. Koenigstein. Low-rank factorization of determinantal point processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2017.

J. Gillenwater, A. Kulesza, E. Fox, and B. Taskar. Expectation-Maximization for learning Determinantal Point Processes. In *Advances in Neural Information Processing Systems*, 2014.

J. Gillenwater, A. Kulesza, and B. Taskar. Discovering diverse and salient threads in document collections. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 710–720, 2012.

J. A. Gillenwater, A. Kulesza, Z. Mariet, and S. Vassilvitskii. Maximizing Induced Cardinality Under a Determinantal Point Process. In *Advances in Neural Information Processing Systems*, 2018.

J. A. Gillenwater, A. Kulesza, Z. Mariet, and S. Vassilvitskii. Sublinear Sampling for Determinantal Point Processes. *Proc. Int. Conference on Machine Learning (ICML)*, 2019.

O. Goldschmidt, D. Nehme, and G. Yu. Note: On the set-union knapsack problem. *Naval Research Logistics*, 41, 1994.

M. Goldstein and A. Dengel. Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm. In *KI-2012: Poster and Demo Track. German Conference on Artificial Intelligence (KI-2012), 35th, September 24-27, Saarbrücken, Germany*, 9 2012.

M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS ONE*, 11(4), 04 2016.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

I. J. Goodfellow. On distinguishability criteria for estimating generative models, 2014.

A. Gotovos, S. Hassani, and A. Krause. Sampling from probabilistic submodular models. In *Advances in Neural Information Processing Systems*, 2015.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid algorithm and its consequences in combinatorial optimization. *Combinatorica*, 1, 1981.

Y. Gu and Z. Jin. Neighborhood preserving d-optimal design for active learning and its application to terrain classification. *Neural Computing and Applications*, 23(7), 2013.

O. Güler. Hyperbolic polynomials and interior point methods for convex programming. *Mathematics of Operations Research*, 22(2), 1997.

M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13(1), February 2012.

F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction.* Springer, 2001.

W. K. Hastings. Monte Carlo Sampling Methods using Markov chains and their applications. *Biometrika*, 57(1), 1970.

X. He. Laplacian regularized d-optimal design for active learning and its application to image retrieval. *IEEE Trans. Image Processing*, 19(1), 2010.

R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985.

J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág. Determinantal processes and independence. *Probability Surveys*, 3(206–229), 2006.

Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

D. A. Jackson and Y. Chen. Robust principal component analysis and outlier detection with ecological data. *Environmetrics*, 15(2), 2004.

T. Jain. Derivatives for antisymmetric tensor powers and perturbation bounds. *Linear Algebra and its Applications*, 435(5), 2011.

J.-M. Jolion, P. Meer, and S. Bataouche. Robust clustering with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8), Aug 1991.

M. I. Jordan. Artificial neural networks. chapter Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, 1990.

R. Jozsa and G. Mitchison. Symmetric polynomials in information theory: Entropy and subentropy. *Journal of Mathematical Physics*, 56(6), 2015.

B. Kang. Fast Determinantal Point Process sampling with application to clustering. In *Advances in Neural Information Processing Systems*, 2013.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

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

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. LoOP: Local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, New York, NY, USA, 2009. ACM.

A. Kulesza and B. Taskar. k-DPPs: Fixed-size Determinantal Point Processes. In *Proc. Int. Conference on Machine Learning (ICML)*, 2011.

A. Kulesza and B. Taskar. *Determinantal Point Processes for machine learning*, volume 5. Foundations and Trends in Machine Learning, 2012.

A. Kulesza. *Learning with Determinantal Point Processes.* PhD thesis, University of Pennsylvania, 2012.

F. S. Lasheras, J. V. Vilán, P. G. Nieto, and J. del Coz Díaz. The use of design of experiments to improve a neural network model in order to predict the thickness of the chromium layer in a hard chromium plating process. *Mathematical and Computer Modelling*, 52(78), 2010. Mathematical Models in Medicine, Business & Engineering 2009.

F. Lavancier, J. Møller, and E. Rubak. Determinantal Point Process models and statistical inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(4), 2015.

Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

A. S. Lewis. Derivatives of spectral functions. *Math. Oper. Res.*, 21(3), 1996.

A. Li, W. Luo, S. Nagavalli, N. Chakraborty, and K. Sycara. Handling state uncertainty in distributed information leader selection for robotic swarms. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016a.

C. Li, S. Sra, and S. Jegelka. Gaussian quadrature for matrix inverse forms with applications. In *Proc. Int. Conference on Machine Learning (ICML)*, 2016b.

C. Li, S. Jegelka, and S. Sra. Efficient sampling for k-Determinantal Point Processes. *CoRR*, abs/1509.01618, 2015.

C. Li, S. Jegelka, and S. Sra. Fast DPP sampling for Nyström with application to kernel methods. In *Proc. Int. Conference on Machine Learning (ICML)*, ICML'16. JMLR.org, 2016c.

C. Li, S. Jegelka, and S. Sra. Fast mixing Markov chains for strongly Rayleigh measures, DPPs, and constrained sampling. In *Advances in Neural Information Processing Systems*, 2016d.

C. Li, S. Jegelka, and S. Sra. Polynomial time algorithms for dual volume sampling. In *Advances in Neural Information Processing Systems*, 2017.

Y. Li, J. Hu, C. Zhai, and Y. Chen. Improving one-class collaborative filtering by incorporating rich user information. In *CIKM*, 2010.

Z. Lin, Y. Liu, G. Molteni, and D.Zhang. Spectral properties for a new composition of a matrix and a complex representation. *Electronic J. Linear Algebra*, 23, 2012.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, December 2015.

C. V. Loan and N. Pitsianis. Approximation with kronecker products. In *Linear Algebra for Large Scale and Real Time Applications*. Kluwer Publications, 1993.

R. Lyons. Determinantal probability measures. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 98(1), 2003.

O. Macchi. The coincidence approach to stochastic point processes. *Adv. Appl. Probab.*, 7, 1975.

I. G. Macdonald. *Symmetric functions and Hall polynomials*. Oxford university press, 1998.

A. Marcus, D. Spielman, and N. Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison–Singer problem. *Annals of Mathematics*, July 2015.

Z. Mariet and S. Sra. Fixed-point algorithms for learning Determinantal Point Processes. In *Proc. Int. Conference on Machine Learning (ICML)*, 2015.

Z. Mariet and S. Sra. Diversity networks: Neural network compression using Determinantal Point Processes. *Int. Conf. on Learning Representations (ICLR)*, 2016a.

Z. Mariet and S. Sra. Kronecker Determinantal Point Processes. In *Advances in Neural Information Processing Systems*, 2016b.

Z. Mariet and S. Sra. Elementary symmetric polynomials for optimal experimental design. In *Advances in Neural Information Processing Systems*, 2017.

Z. Mariet, S. Sra, and S. Jegelka. Exponentiated Strongly Rayleigh distributions. In *Advances in Neural Information Processing Systems*, 2018.

Z. Mariet, M. Gartrell, and S. Sra. Learning Determinantal Point Processes by sampling inferred negatives. In *Proc. Int. Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019a.

Z. Mariet, Y. Ovadia, and J. Snoek. DPPNET: Approximating Determinantal Point Processes with deep networks. *Submitted to NeurIPS*, 2019b.

J. Martens and R. B. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proc. Int. Conference on Machine Learning (ICML)*, 2015.

M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), 1953.

B. Micenková, B. McWilliams, and I. Assent. Learning outlier ensembles: The best of both worlds – supervised and unsupervised. In *Proceedings of the ACM SIGKDD 2014 Workshop on outlier Detection and Description under Data Diversity*, 2014.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proc. Int. Conference on Machine Learning (ICML)*, 2012.

D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. Quantile approximation for robust statistical estimation and k-enclosing problems, 2000.

W. W. Muir. Inequalities concerning the inverses of positive definite matrices. *Proceedings of the Edinburgh Mathematical Society*, 19(2), 1974.

E. Nyström. Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. *Acta Mathematica*, 54(1), 1930.

T. Osogami, R. Raymond, A. Goel, T. Shirai, and T. Maehara. Dynamic Determinantal Point Processes. In *Proc. AAAI Conference on Artificial Intelligence*, 2018.

R. Pemantle. Towards a theory of negative dependence. *Journal of Mathematical Physics*, 41, 2000.

F. Pukelsheim. *Optimal Design of Experiments.* Society for Industrial and Applied Mathematics, 2006.

P. Rebeschini and A. Karbasi. Fast mixing for discrete point processes. In *Conference on Learning Theory (COLT)*, 2015.

P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, Inc., New York, NY, USA, 1987.

P. J. Rousseeuw and B. C. van Zomeren. Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association*, 85(411), 1990.

G. Sagnol. *Optimal design of experiments with application to the inference of traffic matrices in large networks: second order cone programming and submodularity.* Theses, École Nationale Supérieure des Mines de Paris, December 2010.

A. Schein and L. Ungar. A-Optimality for Active Learning of Logistic Regression Classifiers, 2004.

H. Shen, S. Jegelka, and A. Gretton. Fast kernel-based independent component analysis. *IEEE Trans. on Signal Processing*, 57(9), 2009.

T. Shirai and Y. Takahashi. Random point fields associated with certain Fredholm determinants I: fermion, Poisson and boson point processes. *Journal of Functional Analysis*, 205(2), 2003.

A. Shpilka and A. Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *computational complexity*, 10(1), 2001.

A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

J. D. Smith and M. T. Thai. Breaking the bonds of submodularity: Empirical estimation of approximation ratios for monotone non-submodular greedy maximization. *CoRR*, abs/1702.07002, 2017.

N. A. Smith and J. Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *In Proc. of IJCAI Workshop on Grammatical Inference Applications*, 2005a.

N. A. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*, ACL '05, 2005b.

W. Specht. Zur Theorie der elementaren Mittel. *Math. Zeitschr.*, 74, 1960.

S. Sra and R. Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM J. Optimization (SIOPT)*, 25(1), 2015.

N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis, 1993.

P. Sun and R. M. Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5), 2004.

K. K. Sung. *Learning and Example Selection for Object and Pattern Detection.* PhD thesis, Massachusetts Institute of Technology, 1996.

A. Talwalkar, S. Kumar, M. Mohri, and H. Rowley. Large-scale svd and manifold learning. *J. Mach. Learn. Res.*, 14(1):3129–3152, January 2013.

M. Todd. *Minimum-Volume Ellipsoids.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

S. Tschiatschek, J. Djolonga, and A. Krause. Learning probabilistic submodular diversity models via noise contrastive estimation. In *Proc. Int. Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM J. Matrix Anal. Appl.*, 19(2), 1998.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Y. Wang, A. W. Yu, and A. Singh. On computationally tractable selection of experiments in regression models, 2016a.

Z. Wang, B. Moran, X. Wang, and Q. Pan. Approximation for maximizing monotone non-decreasing set functions with a greedy method. *J. Comb. Optim.*, 31(1), January 2016b.

W. J. Welch. Algorithmic complexity: three NP-hard problems in computational statistics. *Journal of Statistical Computation and Simulation*, 15(1), 1982.

M. Wilhelm, A. Ramanathan, A. Bonomo, S. Jain, E. Chi, and J. Gillenwater. Practical Diversified Recommendations on YouTube with Determinantal Point Processes. In *Proc. Conference on Information and Knowledge Management (CIKM)*, 2018.

C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, 2001.

G. Wu, Z. Zhang, and E. Y. Chang. Kronecker factorization for speeding up kernel machines. In *SIAM Data Mining (SDM)*, 2005.

T. C. Xygkis, G. N. Korres, and N. M. Manousakis. Fisher information based meter placement in distribution grids via the d-optimal experimental design. *IEEE Transactions on Smart Grid*, PP(99), 2016.

I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12), 1998.

A. L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In *Advances in Neural Information Processing Systems*, 2002.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, 2017.

C. Zhang, H. Kjellström, and S. Mandt. Stochastic learning on imbalanced data: Determinantal Point Processes for mini-batch diversification. *CoRR*, abs/1705.00607, 2017.

X. Zhang, F. X. Yu, R. Guo, S. Kumar, S. Wang, and S.-F. Chang. Fast orthogonal projection based on kronecker product. In *Int. Conference on Computer Vision (ICCV)*, 2015.

T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *PNAS*, 107(10), 2010.

J. Y. Zou and R. Adams. Priors for diversity in generative latent variable models. In *Advances in Neural Information Processing Systems*, 2012.

Appendix A

A.1 Mathematical tools

A.1.1 Geodesic convexity

Definition A.1.1 (geodesic-convexity). A function $f : \mathbb{S}^{++} \to \mathbb{R}$ defined on the Riemannian manifold $\mathbb{S}^{++}$ is called *geodesically convex* if it satisfies

$$f(\boldsymbol{P}\#_t\boldsymbol{Q}) \leq (1-t)f(\boldsymbol{P}) + tf(\boldsymbol{Q}), \qquad t \in [0,1], \text{ and } \boldsymbol{P}, \boldsymbol{Q} \succ 0.$$

where we use the notation $\boldsymbol{P}\#_t\boldsymbol{Q} := \boldsymbol{P}^{1/2}(\boldsymbol{P}^{-1/2}\boldsymbol{Q}\boldsymbol{P}^{-1/2})^t\boldsymbol{P}^{1/2}$ to denote the *geodesic* between $\boldsymbol{P}$ and $\boldsymbol{Q} \in \mathbb{S}^{++}$ under the Riemannian metric $g_{\boldsymbol{P}}(\boldsymbol{X}, Y) = \operatorname{tr}(\boldsymbol{P}^{-1}\boldsymbol{X}\boldsymbol{P}^{-1}Y)$.

A.1.2 Power-mean inequality

Theorem A.1.1 (Specht (1960)). *Let $x_i > 0$ and $w_i \geq 0$ for $1 \leq i \leq N$ such that $\sum_i w_i = 1$. Let $p < q \in \mathbb{R}$ such that $pq \neq 0$. Then, letting $\kappa = \frac{\max x_i}{\min x_i}$,*

$$1 \leq \frac{M_q(\boldsymbol{w};\boldsymbol{x})}{M_p(\boldsymbol{w};\boldsymbol{x})} \leq \left(\frac{q-p}{q}\frac{\kappa^q-1}{\kappa^q-\kappa^p}\right)^{\frac{1}{p}}\left(\frac{p}{q-p}\frac{\kappa^q-\kappa^p}{\kappa^p-1}\right)^{\frac{1}{q}}.$$

where the power mean $M_p(\boldsymbol{w};\boldsymbol{x})$ *is defined as*

$$M_p(m;x) := \left(\sum\nolimits_{i=1}^{N} w_i x_i^p\right)^{\frac{1}{p}}.$$

Theorem A.1.2. *Let $L \in \mathbb{P}^n$ be a positive definite matrix and $S \subseteq [n]$. Then,*

$$\det(L[S])^p \geq \det(L^p[S]), \qquad 0 \leq p \leq 1,$$
$$\det(L[S])^p \leq \det(L^p[S]), \qquad p \geq 1.$$

Proof. From Lemma 6.2.7, there exists a vector $\boldsymbol{w}$ in the probability simplex, of size $\binom{n}{|S|}$, such that

$$\det(L[S]) = \sum\nolimits_{J \subseteq [n], |J|=|S|} w_J \prod\nolimits_{i \in J} \lambda_i.$$

Since $t \to t^p$ is convex for $p \geq 1$, Jensen's inequality shows that

$$\det(L[S])^p \leq \sum\nolimits_{J \subseteq [n], |J|=|S|} w_J \prod\nolimits_{i \in J} \lambda_i^p = \det(L^p[S]),$$

where the latter equality follows due to L and L^p sharing the same eigenbasis. The same reasoning for $p < 1$ gives the other side of the inequality. □

A.2 Additional experimental results

A.2.1 Encoder details for DppNet

For the MNIST encodings, the VAE encoder consists of a 2d-convolutional layer with 64 filters of height and width 4 and strides of 2, followed by a 2d convolution layer with 128 filters (same height, width and strides), then by a dense layer of 1024 neurons. The encodings are of length 32.

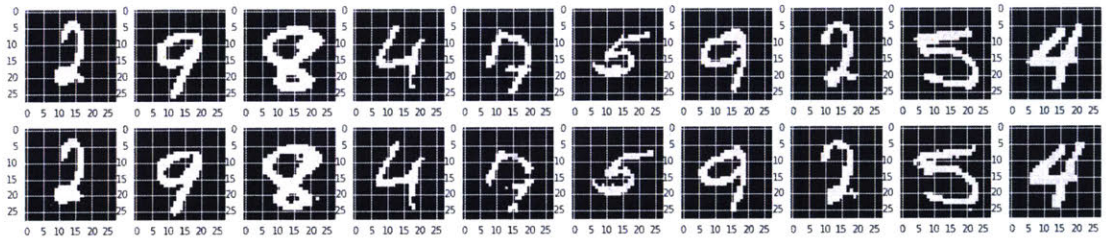


Figure A-1: Digits and VAE reconstructions from the MNIST training set

CelebA encodings were generated by a VAE using a Wide Residual Network en-

coder (Zagoruyko and Komodakis, 2016) with 10 layers and filter-multiplier $k = 4$, a latent space of 32 full-covariance Gaussians, and a deconvolutional decoder trained end-to-end using an ELBO loss. In detail, the decoder architecture consists of a 16K dense layer followed by a sequence of 4×4 convolutions with $[512, 256, 128, 64]$ filters interleaved with $2\times$ upsampling layers and a final 6×6 convolution with 3 output channels for each of 5 components in a mixture of quantized logistic distributions representing the decoded image.

A.2.2 Amazon Baby registries dataset

The Amazon baby registries dataset is a small, standard dataset used to evaluate the performance of DPPs.

Table A.1: Description of the Amazon Baby registries dataset.

REGISTRY	NUMBER OF ITEMS	NUMBER OF SETS
FURNITURE	32	1541
CARSEATS	34	2005
SAFETY	36	1654
STROLLERS	40	2132
MEDIA	58	2730
TOYS	62	4728
HEALTH	62	6598
BATH	100	6877
APPAREL	100	8102
BEDDING	100	8899
DIAPER	100	10,504
GEAR	100	8861
FEEDING	100	12,612

A.2.3 Elementary Symmetric Polynomials

To compare to (Wang et al., 2016a), we generated the experimental matrix X by sampling n vectors of size m from the multivariate Gaussian distribution of mean 0 and covariance $\Sigma = \text{Diag}(1^{-\alpha}, \ldots, m^{-\alpha})$ for various sizes of α and multiple budgets k, with $m = 50, n = 1000$; α controls hows *skewed* the distribution is.

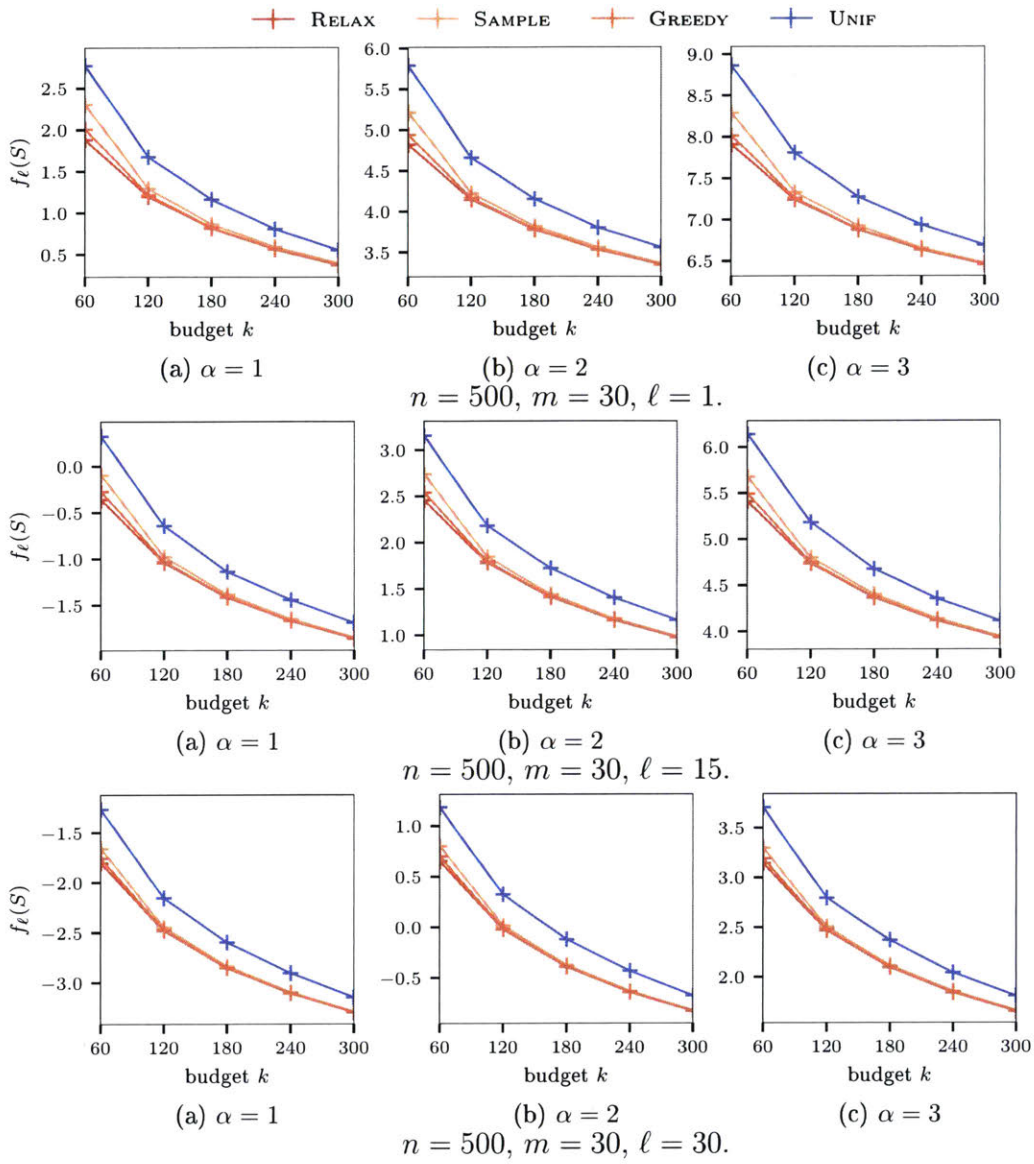
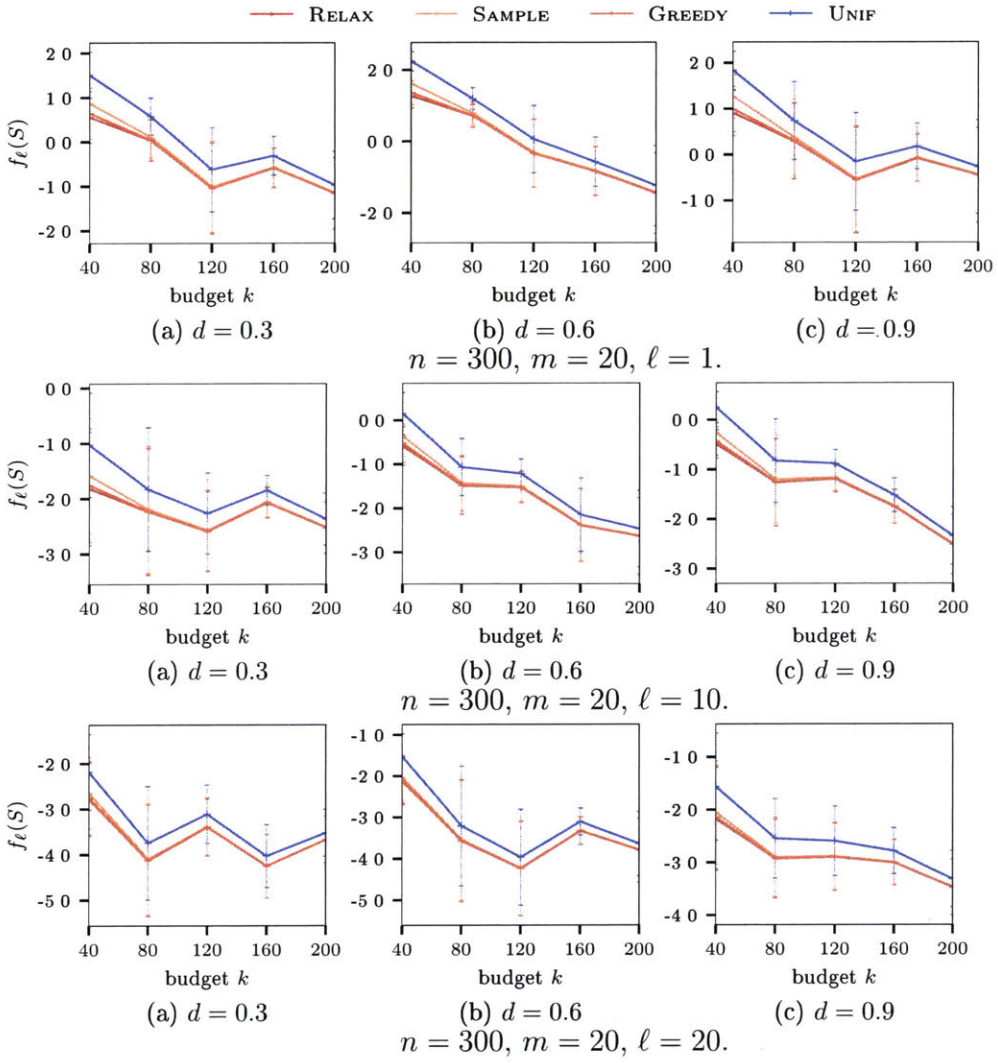


Figure A-2: Synthetic experiments with skewed covariance matrix

Results averaged over 5 runs are reported in Table A.2 and Figure A-2; standard deviations are too small to appear in Figure A-2. We also report more extensive results on synthetic experiments with a sparse precision matrix in Figure A-3.

Table A.2: $\|z\|_0$ for $n = 500$, $m = 30$, $\ell = 15$

	$k = 60$	$k = 120$	$k = 180$	$k = 240$	$k = 300$
$\alpha = 1$	167 ± 9	192 ± 6	241 ± 5	290 ± 4	335 ± 4
$\alpha = 2$	160 ± 4	187 ± 5	240 ± 2	284 ± 3	331 ± 6
$\alpha = 3$	160 ± 4	190 ± 3	237 ± 5	281 ± 4	333 ± 3



(a) $d = 0.3$ (b) $d = 0.6$ (c) $d = 0.9$

$n = 300$, $m = 20$, $\ell = 1$.

(a) $d = 0.3$ (b) $d = 0.6$ (c) $d = 0.9$

$n = 300$, $m = 20$, $\ell = 10$.

(a) $d = 0.3$ (b) $d = 0.6$ (c) $d = 0.9$

$n = 300$, $m = 20$, $\ell = 20$.

Figure A-3: Synthetic experiments with sparse precision matrix.

A.2.4 Exponentiated strongly Rayleigh measures

Figure A-4 presents the evolution of the closed-form upper-bound for the r-closeness term for E-DPPs, depending on the choice of proposal distributions.

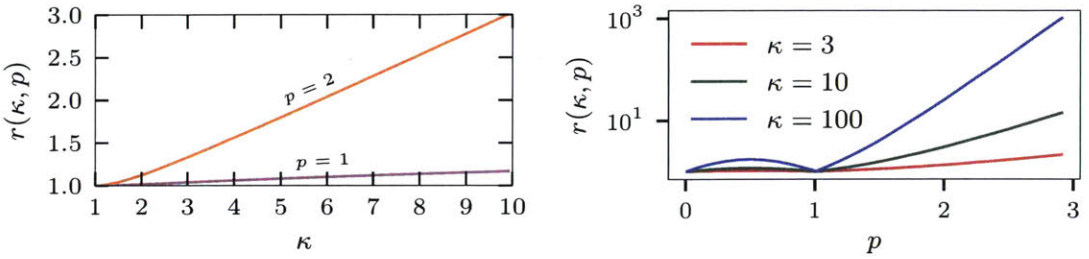
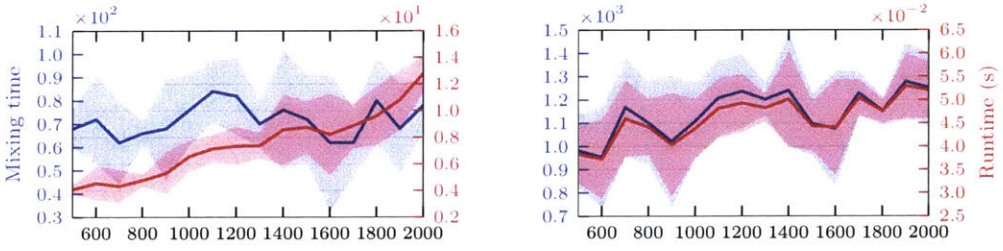


Figure A-4: Evolution of the upper bound for $r(p, \kappa)$ from Thm. 6.2.8, which measures the r-closeness between the E-DPP with kernel L and the DPP with kernel L^p.

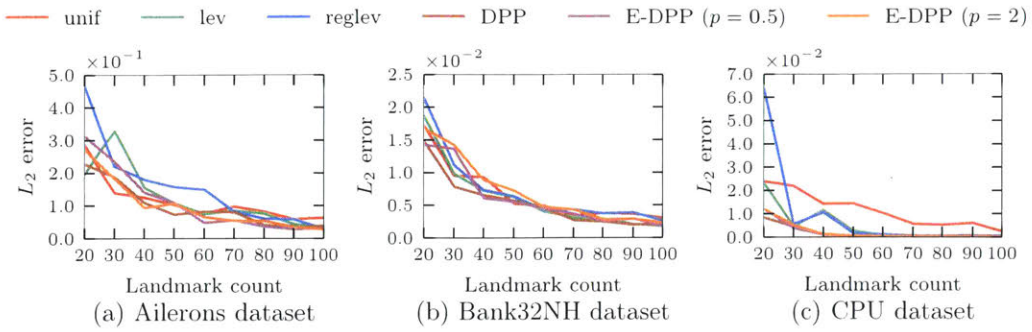
In Figure A-5, we show the empirical mixing time of the MCMC samplers for exponentiated DPPs when varying the size of the ground set size N.

Finally, additional results for the kernel reconstruction experiments are reported in Figure A-6 and A-7; these results evaluate the quality of the reconstructed kernel for different metrics.



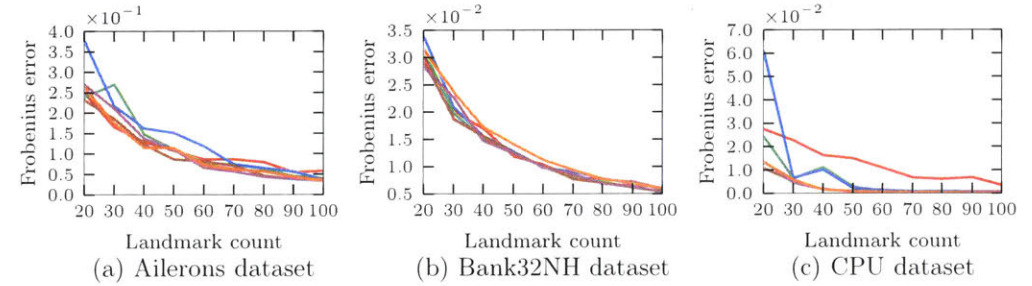
(a) Proposal, $r = 2.0$ (b) Swapchain, $r = 2.0$

Figure A-5: Influence of ground set size N on mixing time



(a) Ailerons dataset (b) Bank32NH dataset (c) CPU dataset

Figure A-6: L_2 reconstruction error



(a) Ailerons dataset (b) Bank32NH dataset (c) CPU dataset

Figure A-7: Frobenius norm reconstruction error