APPROXIMATE INFERENCE FOR DETERMINANTAL POINT PROCESSES

Jennifer Gillenwater

A DISSERTATION
in
Computer and Information Science

Presented to the Faculties of the University of Pennsylvania
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

2014

Ben Taskar, Associate Professor of Computer and Information Science
Supervisor of Dissertation

Emily Fox, Adjunct Professor of Computer and Information Science
Co-Supervisor of Dissertation

Lyle Ungar, Professor of Computer and Information Science
Graduate Group Chairperson

Dissertation Committee:
Michael Kearns, Professor of Computer and Information Science
Ali Jadbabaie, Professor of Electrical and Systems Engineering
Alexander Rakhlin, Assistant Professor of Statistics
Jeff Bilmes, Professor of Electrical Engineering, Univeristy of Washington

APPROXIMATE INFERENCE FOR DETERMINANTAL POINT PROCESSES

Acknowledgments

I would first like to thank my advisor, Ben Taskar: for his hundreds of "how's it going?" emails that catalyzed the best conversations of my graduate career; for his prescient advice that worked miracles on our experiments; for his uncanny ability to warp whatever time was left before a deadline into enough time to write a paper. I am deeply grateful to Emily Fox for so generously stepping in to fill Ben's shoes this past year. Further, I am indebted to my committee chair, Michael Kearns, and to my committee members, Ali Jadbabaie, Sasha Rakhlin, and Jeff Bilmes for their guidance and insights that helped to shape this document. I would also like to offer thanks to my friends for their great support: to Alex Kulesza for leading countless trips to the water cooler and sharing with me his dependably profound perceptions (DPPs); to Kuzman Ganchev and João Graça for getting me started at Penn with good PR work; to David Weiss for his energetic co-TAing; to Ben Sapp for running Ben's Union of Grad Students (BUGS); to Emily Pitler and Annie Louis for being my academic big sisters; to Partha Talukdar, Kayhan Batmanghelich, Katie Gibson, Alex Roederer, Andrew King, Luheng He, and many others for more than I can say here. Additionally, I owe a lot to my family: my brother, who keeps me informed about what is "cool"; my sister, who made it through my entire defense without napping; my dad, who houses Razzle, the World's Greatest Dog, and who is (really) the World's Greatest Dad. Finally, I want to thank Arjun, who has been my companion on so many recent adventures, and with whom I hope to share many more.

ABSTRACT

APPROXIMATE INFERENCE FOR DETERMINANTAL POINT PROCESSES

Jennifer Gillenwater

Ben Taskar

Emily Fox

In this thesis we explore a probabilistic model that is well-suited to a variety of subset selection tasks: the determinantal point process (DPP). DPPs were originally developed in the physics community to describe the repulsive interactions of fermions. More recently, they have been applied to machine learning problems such as search diversification and document summarization, which can be cast as subset selection tasks. A challenge, however, is scaling such DPP-based methods to the size of the datasets of interest to this community, and developing approximations for DPP inference tasks whose exact computation is prohibitively expensive.

A DPP defines a probability distribution over all subsets of a ground set of items. Consider the inference tasks common to probabilistic models, which include normalizing, marginalizing, conditioning, sampling, estimating the mode, and maximizing likelihood. For DPPs, exactly computing the quantities necessary for the first four of these tasks requires time cubic in the number of items or features of the items. In this thesis, we propose a means of making these four tasks tractable even in the realm where the number of items *and* the number of features is large. Specifically, we analyze the impact of randomly projecting the features down to a lower-dimensional space and show that the variational distance between the resulting DPP and the original is bounded. In addition to expanding the circumstances in which these first four tasks are tractable, we also tackle the other two tasks, the first of which is known to be NP-hard (with no PTAS) and the second of which is conjectured to be NP-hard. For mode estimation, we build on submodular maximization techniques to develop an algorithm with a multiplicative approximation guarantee. For likelihood maximization, we exploit the generative process associated with DPP sampling to derive an expectation-maximization (EM) algorithm. We experimentally verify the practicality of all the techniques that we develop, testing them on applications such as news and research summarization, political candidate comparison, and product recommendation.

Contents

List of Tables

List of Figures

List of Algorithms

1

Introduction

When there is a mismatch between the quantity of a resource available and consumer capacity, mechanisms for selecting a subset of the overlarge group are often invoked. For example: for a limited number of job openings, there may be an excessive number of applicants, or, for a camping expedition, there may be more supplies available than a camper can fit in his backpack. Moreover, as the amount of information readily available to us via electronic resources grows, we are increasingly facing this type of dilemma in the setting where the overlarge group is a form of information. For this reason, summarization mechanisms are becoming more and more important for paring information down to manageable portions that human (and machine) learners can easily digest. Subset selection is one elementary way of formalizing the summarization task.

In general, across a wide variety of practical applications, for the task of correcting disparities between resources and consumers, not only has subset selection been the fundamental approach, but also it happens that desirable subsets share two basic characteristics: a good subset is one whose individual items are all high-quality, but also all distinct. For instance, in the case of filling a set of job openings on a team, one might want to select applicants with high GPAs, but with diverse academic

majors, so as to have a variety of informed perspectives on the company's projects. The ability to balance the dual goals of quality and diversity is at the core of most effective subset selection methods.

1.1 Motivating subset selection applications

To give additional insight into situations where such subsets are desired, we list here examples of how common applications are typically cast in the subset selection mold:

- **Product recommendation** (McSherry, 2002; Gillenwater, Kulesza, Fox, and Taskar, 2014): For retailers with a large inventory, selecting a good subset of products to recommend to a given customer is important not only for boosting revenue, but also for saving the customer time. In this setting, a good subset should contain products that other customers have rated highly, but should also exhibit diversity—if a customer already has a carseat in their cart, they most likely will not buy an additional carseat, no matter how popular it is with other consumers. Figure 1.1 illustrates the type of product diversity that a determinantal point process (DPP) can achieve.

Figure 1.1: A set of 10 baby safety products selected using a DPP.

- **Document summarization** (Lin and Bilmes, 2012; Kulesza and Taskar, 2012): Given a set of documents consisting of a total of N sentences, consider the problem of selecting a subset of the sentences to summarize the core

content of the documents. There are many variations on this task, including some where the ground set consists of N structures, rather than just simple sentences; see for example the news threading application in Chapter 4.

- **Web search** (Kulesza and Taskar, 2011a): Given a large number of images or documents, consider the problem of selecting a subset that are relevant to a user query. Note that diversity is important in this setting as many queries are ambiguous (e.g. the search "jaguars" could refer to the cats, the cars, or the football team).

- **Social network marketing** (Hartline, Mirrokni, and Sundararajan, 2008): Consider a social network consisting of N people. Suppose a seller has a digital good that costs nothing to copy (unlimited supply). In this setting, a common marketing strategy is "influence-and-exploit": give the product to a subset of the people for free, then offer it to the rest of the network at a price. Intuitively, the people who get the product for free should be high-degree nodes in the network, but also spread out such that no one will be too many hops from a product owner.

- **Auction revenue maximization** (Dughmi, Roughgarden, and Sundararajan, 2009): Given a base set of bidders, a pool of N potential bidders, and a budget k, a common task is to recruit a subset of k additional bidders for the auction. This is sometimes called the "market expansion problem".

- **Sensor placement for Gaussian Processes** (Krause, Singh, and Guestrin, 2008): Suppose there are N possible locations for sensors (e.g. for measuring pressure, temperature, or pollution). For a given sensor budget, at most k sensors can be placed. Thus, it is important to be able to select a subset of the possible locations at which to actually place sensors, such that the measurements from these sensors will be as informative as possible about the space as a whole.

- **Image segmentation** (Kim, Xing, Fei-Fei, and Kanade, 2011): An initial step in most image-manipulation software is to segment a given image into k parts such that it is homogeneous within each part (e.g. separating an image into

"sky" and "earth" is a common task, with $k = 2$). The segmentation task can be formulated as the problem of identifying one pixel (or superpixel) to serve as the "center" for each segment. Thus, the segmentation task reduces to selecting a size-k subset of an image's N pixels.

- **Pose tracking** (Kulesza and Taskar, 2010): Consider the problem of identifying the pixels corresponding to each person in a video. For each video frame, a subset of the pixels must be chosen. Diversity is important in this setting because people tend to occupy disjoint locations in space, so the goodness of a subset tends to increase as its spatial diversity within a given frame increases.

- **Network routing** (Lee, Modiano, and Lee, 2010): Given the graph of a network, suppose that each link (edge) has some failure probability. Consider the problem of selecting a set of k paths from a source node to a target node such that if a message is sent along all of these paths, failure to deliver (at least one copy of) the message to the target is minimized. The best set of paths will be high-quality (all links on paths will have low failure probability), but also diverse (the failure of a link on one path should not hurt the other paths).

- **Motion summarization** (Affandi, Kulesza, Fox, and Taskar, 2013b): Given videos consisting of individuals performing a specific activity, consider the task of selecting a subset of the N video frames to summarize the activity. For example, the main motions involved in an activity such as basketball can be summarized by a few frames that are diverse in terms of the position of an athlete's limbs.

There are also several basic machine learning tasks that are sometimes cast as subset selection problems:

- **Clustering** (Elhamifar, Sapiro, and Vidal, 2012; Reichart and Korhonen, 2013; Shah and Ghahramani, 2013; Kang, 2013): Given N data points, select a subset of k points to serve as cluster centers or as representatives of the overall dataset.

- **Dimensionality reduction** (Guyon and Elisseeff, 2003): Given N features, select a subset of k features to which to assign non-zero weight in a model.

The complexity of the tasks mentioned in this section is increased by the frequent practical need for constraints on the selected subset, $Y \subseteq \{1, \ldots, N\}$. These range from relatively simple cardinality constraints ($|Y| \leq k$, $|Y| = k$), to intersections of multiple matroids. See Chapter 5 for additional discussion of subset constraints.

1.2 Expressing set-goodness as a determinant

Having established some motivation for solving subset selection problems where the end goal is a high-quality but diverse set, in this section we consider basic measures of quality and diversity that lead directly to the definition of a determinantal point process (DPP).

Given N items, let item i be represented by a D-dimensional feature vector $B_i \in \mathbb{R}^{D \times 1}$. For example, in the case of document summarization each sentence is an item, and D might be the size of the vocabulary. Each entry in B_i could be a count of the number of occurrences of the corresponding vocabulary word, normalized in some reasonable manner. One simple way to interpret B_i is to assume that its magnitude, $\|B_i\|_2$, represents the quality of item i, and that its dot product with another item's feature vector, $B_i^\top B_j$, corresponds to the similarity between these two items.

For sets of size 1, the goodness of a set can then reasonably be represented by the quality of its only item, or any monotonic transformation of that quality. For example, we could score each singleton set $\{i\}$ according to its quality squared, $\|B_i\|_2^2$. In geometric terms, this is the squared length (1-dimensional volume) of B_i. It is also (trivially) the determinant of the 1×1 matrix $B_i^\top B_i$.

As a measure of how good a set consisting of two items is, it is desirable to have an expression that not only rewards high quality, but also penalizes similarity. Extending the geometric intuition from the singleton setting, for $\{i, j\}$ the squared area (2-dimensional volume) of the parallelogram spanned by B_i and B_j would be a reasonable choice. See Figure 1.2 for an example. Theorem 1.1 establishes that the expression for this squared area is: $\|B_i\|_2^2 \|B_j\|_2^2 - (B_i^\top B_j)^2$. The first term here is the product of the items' squared qualities and the second term is the square of their similarity, so it is clear that this expression captures our stated goal of rewarding quality while penalizing similarity.

Theorem 1.1. *For vectors $B_i, B_j \in \mathbb{R}^{D \times 1}$, the area of the parallelogram with vertices* **0***,*

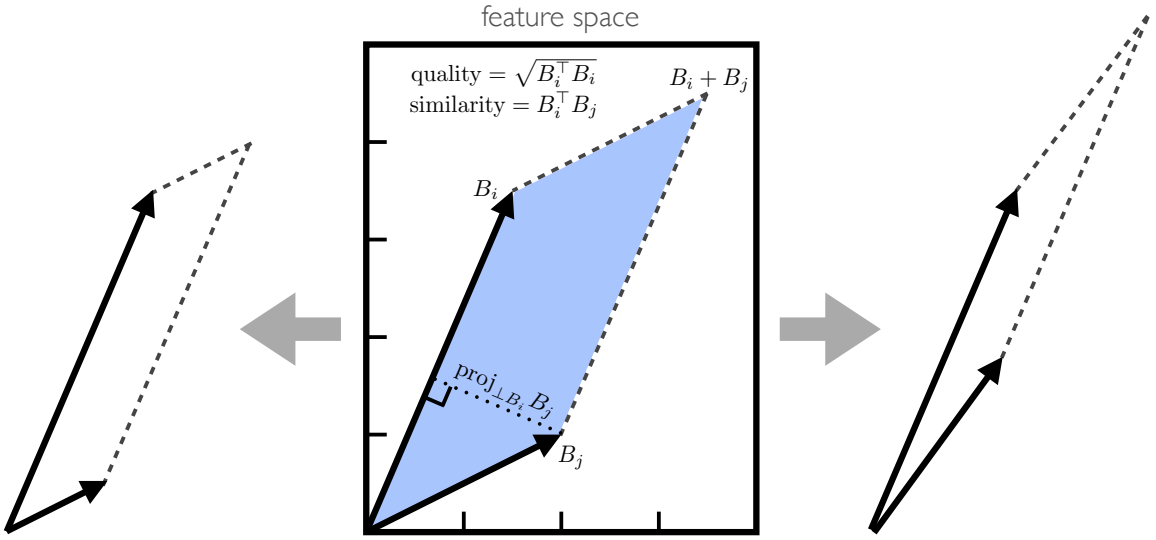


Figure 1.2: The area of the parallelogram outlined by B_i and B_j corresponds to the square root of the proposed set-goodness measure. **Left**: Reducing $\|B_j\|_2$ corresponds to reducing the quality of item j, hence making the set $\{i, j\}$ less desirable. Notice that the reduction in item quality produces a reduction in parallelogram area. **Right**: Reducing the angle between B_i and B_j corresponds to making the items less diverse, again making the set $\{i, j\}$ less desirable. The parallelogram area is similarly reduced in this case.

B_i, B_j, and $B_i + B_j$ is $\sqrt{\|B_i\|_2^2 \|B_j\|_2^2 - (B_i^\top B_j)^2}$.

Proof. See Section 2.1. □

In addition to being related to area, this set-goodness formula can be expressed as a determinant:

$$\det\left(\begin{array}{cc} \|B_i\|_2^2 & B_i^\top B_j \\ B_i^\top B_j & \|B_j\|_2^2 \end{array}\right) = \det\left(\left[\begin{array}{c} B_i^\top \\ B_j^\top \end{array}\right][B_i\ B_j]\right) = \|B_i\|_2^2\|B_j\|_2^2 - (B_i^\top B_j)^2\,. \quad (1.1)$$

The geometric and determinantal formulas for size-1 and size-2 sets extend in a fairly straightforward manner to sets of all sizes. Consider replacing length and area with k-dimensional volume. For a set of size 3, this is the canonical notion of volume. For larger sets, applying the standard "height $\times$ base" formula gives the recursive definition:

$$\text{vol}(B) = \|B_1\|_2 \text{vol}\left(\text{proj}_{\perp B_1}(B_{2:N})\right)\,, \quad (1.2)$$

where $\text{proj}_{\perp B_1}(\cdot)$ projects to the subspace perpendicular to B_1, and the subscript $B_{2:N}$ indexes all columns of B between 2 and N, inclusive. Theorem 1.2 establishes that the square of this geometric formula is equal to the natural extension of Equation (1.1), the determinantal formula, to larger size sets: $\det(B^\top B) = \text{vol}(B)^2$.

Theorem 1.2. *Given a $D \times N$ matrix B with $N \leq D$, consider the N-parallelotope whose vertices are linear combinations of the N columns of B with weights in $\{0, 1\}$: $V = \{\alpha_1 B_1 + \ldots + \alpha_N B_N \mid \alpha_i \in \{0, 1\} \ \forall i \in \{1, \ldots, N\}\}$. The volume of this N-parallelotope is $\sqrt{\det(B^\top B)}$.*

Proof. See Section 2.1. □

This basic extension of a natural set-goodness measure from size-1 and size-2 sets to sets of arbitrary size brings us immediately to the definition of a determinantal point process (DPP).

1.3 Definition of a DPP

The previous section informally defined a set-goodness score in terms of a feature matrix B. Here we give a more rigorous definition.

At a high level, stochastic processes are generalizations of random vectors. Discrete, finite stochastic processes are equivalent to random vectors: a sequence of N random variables $[Y_1, \ldots, Y_N]$, associated with a joint probability distribution $\mathcal{P}(Y_1 = y_1, \ldots, Y_N = y_N)$. A discrete, finite *point* process is a type of discrete, finite stochastic process where each variable is binary, $y_i \in \{0, 1\}$, indicating the occurrence or non-occurrence of some event (e.g. neuron spike, lightening strike, inclusion of a sentence in a summary). This thesis focuses on discrete, finite determinantal point processes (DPPs), which are point processes where the occurrence of one event corresponds with a decrease in the probability of similar events. That is, these processes exhibit repulsion between points, resulting in diversity.

Formally, let the points (sometimes referred to as events or items) under consideration be those in the set $\mathcal{Y} = \{1, 2, \ldots, N\}$. Let $2^\mathcal{Y}$ refer to the set of all subsets of $\mathcal{Y}$, which has magnitude 2^N. This includes the empty set, $\emptyset$, and the full set, $\mathcal{Y}$. We will frequently use $Y \subseteq \mathcal{Y}$ to denote one of these subsets, and $\boldsymbol{Y}$ to denote a random variable whose value can be any $Y \subseteq \mathcal{Y}$. According to Borodin and Rains

(2005), a random variable $\boldsymbol{Y}$ drawn according to a (discrete, finite) determinantal point process $\mathcal{P}_L$ has value Y with probability:

$$\mathcal{P}_L(\boldsymbol{Y} = Y) \propto \det(L_Y)\,, \tag{1.3}$$

for some positive semi-definite (PSD) matrix $L \in \mathbb{R}^{N \times N}$. The L_Y here denotes the restriction of L to rows and columns indexed by elements of Y: $L_Y \equiv [L_{ij}]_{i,j \in Y}$. It is assumed that $\det(L_\emptyset) = 1$. In what follows, we will use the shorthand $\mathcal{P}_L(Y)$ for $\mathcal{P}_L(\boldsymbol{Y} = Y)$ where the meaning is clear.

Note that the definition given by Equation (1.3) is similar to the geometrically motivated set-goodness score from Section 1.2, but with L in place of $B^\top B$. These definitions are in fact equivalent. To see this, note that $B^\top B$ can be any Gram matrix. The equivalence of the definitions then follows immediately from the equivalence of the class of Gram matrices and the class of PSD matrices: every PSD matrix can be written as the Gram matrix from some set of (potentially infinite dimensional) vectors, and every Gram matrix is PSD.

1.4 Motivating DPP inference tasks

Having established the formal definition of a DPP, it is now possible to discuss its associated inference problems and how each relates to the subset selection task. Common inference operations include MAP estimation, sampling, marginalizing, conditioning, likelihood maximization, and normalizing.

- **MAP estimation**: Finding the mode, or as it is sometimes referred to in conditional models, maximum a posteriori (MAP) estimation, is the problem of finding the highest-scoring set. This is the set for which the balanced measure of quality and diversity developed in Section 1.2 is highest. Clearly, this is the inference operation we would ultimately like to perform for the subset selection tasks from Section 1.1. Unfortunately, this problem corresponds to volume maximization, which is known to be NP-hard. Thus, for the problem of selecting the highest-scoring set under a DPP, we must rely on approximation algorithms.

- **Sampling**: Being able to sample from a DPP's distribution is important for a wide variety of tasks. These include estimating expected values and com-

bining DPPs with other probabilistic models to build novel generative stories. Sampling is also one very simple way of approximating a DPP's mode; since higher-quality, more diverse sets have greater probability mass, we are more likely to sample them. Even this simple MAP approximation technique can often yield better sets than non-DPP-based subset selection methods.

- **Marginalizing**: Sampling algorithms, as well as several other MAP estimation techniques, rely on the computation of marginal probabilities for efficiency. More concretely, one of the first steps in a basic DPP sampling algorithm is to estimate the marginal probability for each individual item $i \in \mathcal{Y}$. The first item for the sample set is then selected with probability proportional to these marginals.

- **Conditioning**: As with marginalization, being able to condition on the inclusion of an item $i \in Y$ is important to the efficiency of sampling algorithms. Independent of its use in sampling though, conditioning can easily be seen to have practical importance. Recall for a moment the product recommendation application from Section 1.1, and suppose that we are presented with a customer who already has several items in their cart. To recommend additional items, it makes sense to condition on the items already selected.

- **Likelihood maximization**: For some subset selection tasks appropriately-weighted feature vectors are readily available and can be used to form a feature matrix B, from which we can compute a DPP kernel $L = B^\top B$. However, in most cases it is not known up front how important each feature is. Instead, we often have access to indirect evidence of feature importance, in the form of examples of "good" subsets. Leveraging this information to learn feature weights, or to infer the entries of the kernel matrix L directly, can be done by maximizing the likelihood of the "good" subsets. The result is a kernel that can be used to find good subsets for related subset selection tasks.

- **Normalizing**: Being able to compute set goodness scores that all fall into the $[0, 1]$ range is useful for many probabilistic modeling tasks. For instance, normalization makes it easy to compare two different DPPs, which is necessary for likelihood maximization.

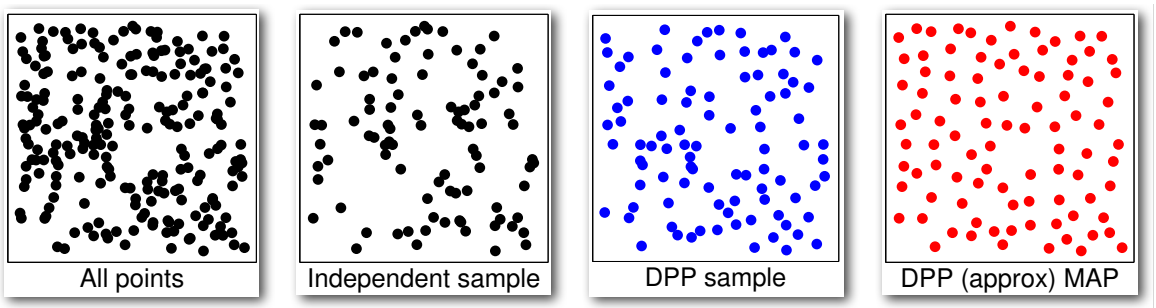


Figure 1.3: From left to right: A ground set of N points; sampling each point independently with probability $\frac{1}{2}$; sampling from a DPP with a Gaussian kernel; applying a DPP MAP approximation algorithm. The DPP sample exhibits greater diversity than the independent sample, but the DPP MAP approximation is the most diverse.

1.5 Thesis contributions

DPPs can typically be efficiently normalized, marginalized, conditioned, and sampled. Table 2.1 in the next chapter gives the exact time complexities, but roughly these operations require time cubic in the number of items, N, or features, D, of these items: $O(\min(N^3, D^3))$ time. Unfortunately, there are some practical settings where cubic time is too expensive. For example, recall the document summarization task from Section 1.1. If the documents we wish to summarize consist of all the New York Times articles from the past year, then the number of sentences N will be in the millions. Moreover, if we use the vocabulary of these articles as the feature set, then the number of features D will be (at least) in the tens of thousands. Exact normalization, marginalization, conditioning, and sampling algorithms cannot handle data of this size. Thus, in this thesis we explore approximations for the setting where both N and D are large.

Besides the quantities that can be computed exactly for moderate N and D, there are important inference operations for which the expressivity of the DPP causes us to pay a higher price. We focus on two of these in this thesis. The first, MAP estimation, is known to be NP-hard. Figure 1.3 illustrates for a simple example in two dimensions how the DPP MAP can significantly differ from a DPP sample. Given that sampling can sometimes be a poor approximation to the MAP, investigating other approximation methods is vital to making DPPs useful. A second more complex inference task we consider is likelihood maximization, which is conjectured to

be NP-hard. This means that to learn a DPP that puts as much weight as possible on observed "good" subsets, we must rely on local optimization techniques.

In this thesis, we seek to better address all three of these hard problems: the large-N, large-D setting, MAP estimation, and likelihood maximization. It might seem easier to skirt these issues by switching to a simpler model than the DPP—clearly, given all of the references in Section 1.1, there are many non-DPP-based subset selection methods to choose from. However, experimental results, such as those in Sections 4.4.4, 5.6, and 6.7, suggest that sticking with the DPP paradigm is often the better choice. The primary contributions of this document are summarized below.

- Chapter 2: We review the basic properties and representations of DPPs. For the core inference tasks, we survey algorithms and hardness results.
- Chapter 3: We discuss several useful DPP variants: k-DPPs, structured DPPs, Markov DPPs, and continuous DPPs. Several of these variants are relevant to the results in later chapters.
- Chapter 4: We analyze the impact of randomly projecting features down to a lower-dimensional space and establish a bound on the difference between the resulting DPP and the original. We illustrate the practicality of the projected DPP by applying it to select sets of document threads in large news and research collections.
- Chapter 5: We build on submodular maximization techniques to develop an algorithm for finding the DPP MAP problem with a multiplicative $\frac{1}{4}$-approximation guarantee. We validate the proposed algorithm on a political candidate comparison task.
- Chapter 6: We derive an expectation-maximization (EM) algorithm for learning the DPP kernel, motivated by the generative process that gives rise to DPP sampling algorithms. We show its superiority relative to projected gradient ascent on a product recommendation task.
- Chapter 7: We recap the innovations from the preceding chapters, summarizing the thesis contributions. Lastly, we discuss avenues for future work in the domain of approximate inference for DPPs.

2

DPP Basics

This chapter discusses the basic properties, algorithms, identities, and hardness results associated with DPPs. For additional background, references to mathematical surveys, the use of DPPs as models in physics (e.g. for fermions), theoretical point processes that are determinantal (e.g. edges in random spanning trees, non-intersecting random walks), proofs of many of the theorems stated in this section, and a list of related point processes (e.g. Poisson, hyperdeterminantal), see Kulesza (2012).

2.1 Geometric interpretation

Section 1.2 described the connection between determinants and volumes of parallelotopes. To sharpen those intuitions, in this section we provide proofs for the associated theorems. First, for the case of the 2×2 determinant, then for the more general $N \times N$ case.

Proof. (Of Theorem 1.1.) This is a straightforward application of the "base $\times$ height" formula for the area of a parallelogram. Let B_i serve as the base. Then:

$$\text{area} = \|B_i\|_2 \times \|\operatorname{proj}_{\perp B_i}(B_j)\|_2 \,, \tag{2.1}$$

where $\text{proj}_{\perp B_i}(\cdot)$ projects to the subspace perpendicular to B_i. Using the Pythagorean theorem to re-write this projection:

$$\|\text{proj}_{\perp B_i}(B_j)\|_2^2 + \|\text{proj}_{\|B_i}(B_j)\|_2^2 = \|B_j\|_2^2 \tag{2.2}$$

$$\|\text{proj}_{\perp B_i}(B_j)\|_2 = \sqrt{\|B_j\|_2^2 - \|\text{proj}_{\|B_i}(B_j)\|_2^2}\,. \tag{2.3}$$

Now, consider the basis formed by the vectors constituting the columns of a matrix A. According to Meyer (2000, Equation 5.13.3), the matrix $P_A = A(A^\top A)^{-1}A^\top$ projects to the vector space spanned by those columns. Applying this with B_i as A:

$$\text{proj}_{\|B_i}(B_j) = B_i(B_i^\top B_i)^{-1}B_i^\top B_j = \frac{B_i B_i^\top}{\|B_i\|_2^2}B_j\,. \tag{2.4}$$

Combining Equations (2.3) and (2.4), and squaring the area:

$$\text{area}^2 = \|B_i\|_2^2\left(\|B_j\|_2^2 - \left\|\frac{B_i B_i^\top}{\|B_i\|_2^2}B_j\right\|_2^2\right) \tag{2.5}$$

$$= \|B_i\|_2^2\|B_j\|_2^2 - \frac{\|B_i\|_2^2}{\|B_i\|_2^4}\|B_i B_i^\top B_j\|_2^2 \tag{2.6}$$

$$= \|B_i\|_2^2\|B_j\|_2^2 - \frac{1}{\|B_i\|_2^2}\|B_i\|_2^2(B_i^\top B_j)^2 \tag{2.7}$$

$$= \|B_i\|_2^2\|B_j\|_2^2 - (B_i^\top B_j)\,, \tag{2.8}$$

where Equations (2.6) and (2.7) follow from the fact that norms are homogeneous functions, which means that scalars such as $\|B_i^\top B_i\|_2$ and $B_i^\top B_j$ inside a norm are equivalent to powers of these scalars outside the norm. □

We now show the derivation of the more general determinant-volume relationship for N dimensions.

Proof. (Of Theorem 1.2.) We proceed by induction on N. For $N = 1$, volume is length. The length $||B_1||_2$ is trivially the square root of the 1×1 determinant $\det(B_1^\top B_1) = B_1^\top B_1$. Now, assuming the theorem is true for $N-1$, we will show that it holds for N.

Write $B_N = B_N^{\|} + B_N^{\perp}$, where $B_N^{\|}$ is in the span of $\{B_1, \ldots, B_{N-1}\}$ and $B_N^{\perp}$ is orthogonal to this subspace. Such a decomposition can be found by running the Gram-Schmidt process on the B_i. Let $\boldsymbol{w}$ be weights such that: $B_N^{\|} = w_1 B_1 + \ldots + w_{N-1}B_{N-1}$. Then define the corresponding elementary row-addition transformation

matrices, identity matrices with w_i added to entry (i, N): $T_{i,N}(w_i) = I + \mathbf{1}_i w_i \mathbf{1}_N^\top$, where $\mathbf{1}_i$ indicates an $N \times 1$ vector with zeros in all entries except for a one in the ith. Let $\tilde{B}$ be identical to B, but with B_N replaced by $B_N^\perp$. We can relate B to $\tilde{B}$ via the elementary transformation matrices: $B = \tilde{B} T_{1,N}(w_1) \dots T_{N-1,N}(w_{N-1})$. Taking the determinant of $B^\top B$, these transformations disappear; the determinant of a product decomposes into individual determinants, and the determinant of an elementary transformation matrix is 1. Thus, $\det(B^\top B) = \det(\tilde{B}^\top \tilde{B})$. Writing out the new product $\tilde{B}^\top \tilde{B}$:

$$\tilde{B}^\top \tilde{B} = \begin{pmatrix} B_{1:N-1}^\top B_{1:N-1} & B_{1:N-1}^\top B_N^\perp \\ {B_N^\perp}^\top B_{1:N-1} & {B_N^\perp}^\top B_N^\perp \end{pmatrix}. \tag{2.9}$$

The orthogonality of $B_N^\perp$ means that $B_{1:N-1}^\top B_N^\perp$ is all zeros, as is ${B_N^\perp}^\top B_{1:N-1}$. Thus the determinant is:

$$\det(\tilde{B}^\top \tilde{B}) = \det(B_{1:N-1}^\top B_{1:N-1}) {B_N^\perp}^\top B_N^\perp. \tag{2.10}$$

By the inductive hypothesis, the first term here is the squared volume of the $(N-1)$-parallelotope defined by $B_{1:N-1}$. Letting $B_{1:N-1}$ serve as the base of the N-parallelotope defined by B, the height component of the "base $\times$ height" volume formula is $||B_N^\perp||_2$. This is exactly the square root of the second term above, ${B_N^\perp}^\top B_N^\perp$. □

2.2 Inference

This section provides background on various common DPP inference tasks. The complexity of these tasks is summarized in Table 2.1.

2.2.1 Normalizing

The definition of a DPP given in Equation (1.3) omits the proportionality constant necessary to compute the exact value of $\mathcal{P}_L(Y)$. Naïvely, this constant seems hard to compute, as it is the sum over an exponential number of subsets: $\sum_{Y' : Y' \subseteq \mathcal{Y}} \det(L_{Y'})$. Fortunately though, due to the multilinearity of the determinant, this sum is in fact equivalent to a single determinant: $\det(L + I)$. This is a special case of Theorem 2.1 for $A = \emptyset$.

Task	Runtime
Normalizing	$O(\min\{N^\omega, D^\omega\})$
Marginalizing	$O(\min\{N^\omega, D^\omega + D^2k^2\})$
Conditioning	For exclusion: $O(N^\omega)$; For inclusion: $O(\min\{N^\omega, D^\omega + D^2k^2 + k^\omega\})$
Sampling	Given L's eigendecomposition: $O(Nk^2)$; else: $O(\min\{N^\omega + Nk^2, D^\omega + NDk^2 + D^2k^3, ND^2k\})$
Finding the mode	NP-hard, no PTAS
Maximizing likelihood	Conjectured to be NP-hard

Table 2.1: Complexity of basic inference tasks. The size of the DPP's ground set is N, and ω denotes the exponent of matrix multiplication. If each item in the ground set is associated with a feature vector (such as the B_i of Section 1.2), then D denotes the vector's length. We assume $D \leq N$. For marginalizing, conditioning, and sampling, k is the size of the set marginalized, conditioned on, or sampled, respectively.

Theorem 2.1. *For any $A \subseteq \mathcal{Y}$:*

$$\sum_{Y : A \subseteq Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I_{\bar{A}}) \,, \tag{2.11}$$

where $I_{\bar{A}}$ is the diagonal matrix with ones in the diagonal positions corresponding to elements of $\bar{A} = \mathcal{Y} \setminus A$, and zeros everywhere else.

Proof. See Kulesza (2012, Theorem 2.1). □

Typically, a single $N \times N$ determinant for a PSD matrix is computed by taking the matrix's Cholesky decomposition. This decomposition re-expresses a PSD matrix, such as L or $L + I$, as the product of a triangular matrix and its transpose: $L + I = TT^\top$. The determinant can then be computed as the square of the product of T's diagonal elements: $\sum_{i=1}^{N} T_{ii}^2$. Naïve algorithms can compute the Cholesky decomposition in $\frac{1}{3}N^3$ operations (multiplications). More nuanced algorithms exhibit improved performance for large N. For example, Bunch and Hopcroft (1974) show that the complexity of triangular factorization is identical to that of matrix multiplication. Building on the Strassen matrix multiplication algorithm, they obtain Cholesky decompositions in time $< 2.45N^{\log_2 7} \approx 2.45N^{2.807}$ (Bunch and Hopcroft, 1974, Final sentence of Section 4). This makes their algorithm more efficient than

the naïve approach for $N \approx 31,000$ and above. While there are matrix multiplication algorithms that are asymptotically faster than Strassen's, these are not used in practice. For instance, the Coopersmith-Winograd algorithm has a complexity of $O(N^{2.375})$, but the big-O notation hides much too large of a constant coefficient for this algorithm to be practical. In analyzing the complexity of algorithms presented in this thesis, we will use ω to denote the exponent of whatever matrix multiplication algorithm is used. For practical purposes though, think of ω as roughly 3.

2.2.2 Marginalizing

Just as the probability of a particular set, $\mathcal{P}_L(\boldsymbol{Y} = Y)$, is proportional to a sub-determinant of a kernel L, the probability of the inclusion of a set, $\mathcal{P}(Y \subseteq \boldsymbol{Y})$, depends on the sub-determinant of a kernel closely related to L.

Theorem 2.2. *For a DPP with kernel L, the matrix K defined by:*

$$K = L(L+I)^{-1} = I - (L+I)^{-1} \tag{2.12}$$

has minors satisfying:

$$\mathcal{P}(Y \subseteq \boldsymbol{Y}) = \det(K_Y)\,. \tag{2.13}$$

Proof. See Kulesza (2012, Theorem 2.2). □

We can also invert Equation (2.12) to express L in terms of K:

$$L = K(I-K)^{-1} = (I-K)^{-1} - I\,. \tag{2.14}$$

We will refer to the matrix K as the *marginal kernel*. From Equation (2.13), two properties of K are immediately obvious: first, since marginal probabilities must be non-negative, K must be PSD; second, since marginal probabilities must be < 1, $I - K$ must be PSD. An equivalent way of stating this second condition is to say that K's eigenvalues must be ≤ 1. Examining the eigendecomposition of K and L further clarifies their relationship. They share the same eigenvectors, and K squashes L's eigenvalues down to the $[0, 1]$ range:

$$L = V\Lambda V^\top = \sum_{i=1}^{N} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top\,, \qquad K = \sum_{i=1}^{N} \frac{\lambda_i}{1+\lambda_i} \boldsymbol{v}_i \boldsymbol{v}_i^\top\,, \tag{2.15}$$

where $\boldsymbol{v}_i$ is the ith column of the eigenvector matrix V.

Since K and L each contain all of the information needed to identify a DPP, we can use either one as the representation of a DPP. In fact, given K it is actually not even necessary to convert to L to obtain the probability of a particular subset. As shown by Kulesza (2012, Section 3.5), we can write:

$$\mathcal{P}_L(\boldsymbol{Y} = Y) = |\det(K - I_{\overline{Y}})|\,. \tag{2.16}$$

The redundancy of K and L comes with one caveat though: L does not exist if any of K's eigenvalues are exactly 1. This is clear from Equation (2.14), where the inverse is incomputable for K with an eigenvalue of 1; an eigenvalue of 1 for K would imply an eigenvalue of ∞ for L. As will be made clear by the sampling algorithms though, as long as some non-zero probability is assigned to the empty set, K's eigenvalues will be < 1.

In terms of the complexity of converting between L and K, if this is done using Equations (2.12) and (2.14) then the dominating operation is the inversion. The naïve algorithm for matrix inversion runs in time $2N^3$. Strassen's matrix multiplication algorithm runs in time $< 4.7N^{\log_2 7}$ (Bunch and Hopcroft, 1974, Paragraph 2 of the introduction) and can be used to compute a matrix inverse in time $< 6.84N^{\log_2 7}$ (Bunch and Hopcroft, 1974, Final sentence of Section 4). This approach is more efficient than the naïve one for $N \approx 600$ and above. If instead we convert from L to K by computing an eigendecomposition, this tends to be slightly more expensive. While asymptotically (as $N \to \infty$) it has the same complexity as matrix multiplication, the algorithms that achieve this complexity are not practical unless N is extremely large. In practice, we instead rely on algorithms such as Lanczos to convert L or K to a similar tridiagonal matrix, then apply divide-and-conquer algorithms such as those described in Gu and Eisenstat (1995) to compute the eigendecomposition of this matrix. (For real, symmetric matrices this is faster than relying on QR decomposition algorithms.) The overall complexity for computing the eigendecomposition of L or K is then $\approx 4N^3$.

2.2.3 Conditioning

Conditioning on the inclusion or exclusion of a subset is also an easy task for DPPs. In fact, the class of DPPs is closed under these conditioning operations, which means

that it is possible to write the resulting set probabilities as a DPP with some modified kernel L' such that $\mathcal{P}_{L'}(\boldsymbol{Y} = Y) = \det(L'_Y)/\det(L'+I)$. Formulas for these modified kernels are given in Table 2.2. We use L^A to denote the kernel conditioned on the inclusion of the set A and $L^{\neg A}$ to denote the kernel conditioned on the exclusion of A. The $(N - |A|) \times (N - |A|)$ matrix $L_{\bar{A}}$ is the restriction of L to the rows and columns indexed by elements in $\mathcal{Y} \setminus A$. The matrix $I_{\bar{A}}$ is the diagonal matrix with ones in the diagonal entries indexed by elements of $\mathcal{Y} \setminus A$ and zeros everywhere else. The $(N - |A|) \times |A|$ matrix $L_{\bar{A},A}$ consists of the $\overline{|A|}$ rows and the A columns of L.

For $|A| = k$, the complexity of computing these conditional kernels is:

- L^A, K^A: Equation (2.18) requires an $N \times N$ matrix inverse, which is an $O(N^\omega)$ operation. Equation (2.20) is dominated by its three-matrix product, an $O(N^2 k)$ operation. Formulas for K^A have the same complexity as the L^A ones.
- $L^{\neg A}$: Equation (2.19) simply requires copying $(N - k)^2$ elements of L.
- $K^{\neg A}$: Equation (2.22) requires an $(N - k) \times (N - k)$ matrix inverse, which is an $O((N - k)^\omega)$ operation.

The formulas in Table 2.2 can also be combined to produce kernels conditioned on both inclusion and exclusion. For instance, including the set A^{in} and excluding A^{out} the corresponding conditional kernel is:

$$L^{A^{\text{in}},\neg A^{\text{out}}} = ([L_{\overline{A^{\text{out}}}} + I_{\overline{A^{\text{in}}}}]_{\overline{A^{\text{in}}}})^{-1} - I\,. \tag{2.17}$$

Equations (2.18) and (2.21) are derived in Kulesza (2012, Equation 2.42, 2.45). Equation (2.19) follows immediately from the definition of a DPP, and Equation (2.22) from the application of the L-to-K conversion formula of Equation (2.12). We derive Equations (2.20) and (2.23) in Lemmas (2.4) and (2.5). These derivations rely upon the following identity.

Definition 2.3. Schur determinant identity: *For a $(p + q) \times (p + q)$ matrix M decomposed into blocks A, B, C, D that are respectively $p \times p$, $p \times q$, $q \times p$, and $q \times q$:*

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \tag{2.24}$$

	Inclusion: $A \subseteq \boldsymbol{Y}$	**Exclusion**: $A \cap \boldsymbol{Y} = \emptyset$
Prob.	L^A	$L^{\neg A}$
	$\left([(L + I_{\bar{A}})^{-1}]_{\bar{A}}\right)^{-1} - I$ (2.18) $L_{\bar{A}} - L_{\bar{A},A} L_A^{-1} L_{A,\bar{A}}$ (2.20)	$L_{\bar{A}}$ (2.19)
Marginal Prob.	K^A	$K^{\neg A}$
	$I - [(L + I_{\bar{A}})^{-1}]_{\bar{A}}$ (2.21) $K_{\bar{A}} - K_{\bar{A},A} K_A^{-1} K_{A,\bar{A}}$ (2.23)	$I - (L_{\bar{A}} + I)^{-1}$ (2.22)

Table 2.2: Formulas for computing DPP kernels conditioned on the inclusion or exclusion of the set $A \subseteq \mathcal{Y}$.

the determinant of M can be written in terms of the blocks:

$$\det(M) = \det(D) \det(A - BD^{-1}C)\,. \tag{2.25}$$

Given this identity, we can now derive Equations (2.20) and (2.23).

Lemma 2.4. *For a DPP with kernel L, the conditional kernel L^A with minors satisfying:*

$$\mathcal{P}(\boldsymbol{Y} = Y \cup A \mid A \subseteq \boldsymbol{Y}) = \frac{\det(L_Y^A)}{\det(L^A + I)} \tag{2.26}$$

on $Y \subseteq \mathcal{Y} \setminus A$, can be computed from L by the rank-$|A|$ update:

$$L^A = L_{\bar{A}} - L_{\bar{A},A} L_A^{-1} L_{A,\bar{A}}\,, \tag{2.27}$$

assuming that the inverse L_A^{-1} exists.

Proof. Conditioning on A means normalizing only by sets Y' that contain A:

$$\mathcal{P}_L(\boldsymbol{Y} = Y \cup A \mid A \subseteq \boldsymbol{Y}) = \frac{\det(L_{Y \cup A})}{\sum_{Y' : A \subseteq Y' \subseteq \mathcal{Y}} \det(L_{Y'})}\,. \tag{2.28}$$

Suppose, without loss of generality, that the items we are conditioning on are the last ones: $A = \{N - |A| + 1, \ldots, N\}$. Then L decomposes into blocks:

$$L = \begin{bmatrix} L_{\bar{A}} & L_{\bar{A},A} \\ L_{A,\bar{A}} & L_A \end{bmatrix} \tag{2.29}$$

and by Schur's identity we have that:

$$\det(L) = \det(L_A)\det\left(L_{\bar{A}} - L_{\bar{A},A}L_A^{-1}L_{A,\bar{A}}\right) . \tag{2.30}$$

Let L' denote the matrix $L_{\bar{A}} - L_{\bar{A},A}L_A^{-1}L_{A,\bar{A}}$. Then applying Schur's identity to the matrix $L_{B\cup A}$, where B is any set such that $A \cap B = \emptyset$, we have:

$$\det(L_{B\cup A}) = \det(L_A)\det\left(L_B - L_{B,A}L_A^{-1}L_{A,B}\right) = \det(L_A)\det\left(L'_B\right) . \tag{2.31}$$

This allows us to simplify Equation (2.28). The numerator can be written:

$$\det(L_{Y\cup A}) = \det(L_A)\det(L'_Y) . \tag{2.32}$$

The normalizing term can similarly be simplified:

$$\begin{aligned}
\sum_{Y':A\subseteq Y'\subseteq\mathcal{Y}} \det(L_{Y'}) &= \sum_{Y':A\subseteq Y'\subseteq\mathcal{Y}} \det(L_A)\det(L'_{Y'\setminus A}) && (2.33)\\
&= \det(L_A) \sum_{Y':A\subseteq Y'\subseteq\mathcal{Y}} \det(L'_{Y'\setminus A}) && (2.34)\\
&= \det(L_A) \sum_{Y':Y'\subseteq\mathcal{Y}\setminus A} \det(L'_{Y'}) && (2.35)\\
&= \det(L_A)\det(L'+I) , && (2.36)
\end{aligned}$$

where the final equality follows from Theorem 2.1. Plugging this back into Equation (2.28):

$$\mathcal{P}_L(\boldsymbol{Y} = Y \cup A \mid A \subseteq \boldsymbol{Y}) = \frac{\det(L_A)\det(L'_Y)}{\det(L_A)\det(L'+I)} = \frac{\det(L'_Y)}{\det(L'+I)} . \tag{2.37}$$

□

Lemma 2.5. *For a DPP with marginal kernel K, the conditional marginal kernel K^A with minors satisfying:*

$$\mathcal{P}(Y \subseteq \boldsymbol{Y} \mid A \subseteq \boldsymbol{Y}) = \det(K^A_Y) \tag{2.38}$$

on $Y \subseteq \mathcal{Y} \setminus A$, can be computed from K by the rank-$|A|$ update:

$$K^A = K_{\bar{A}} - K_{\bar{A},A}K_A^{-1}K_{A,\bar{A}} , \tag{2.39}$$

assuming that the inverse K_A^{-1} exists.

Proof. By the definition of conditional probability:

$$\mathcal{P}(Y \subseteq \boldsymbol{Y} \mid A \subseteq \boldsymbol{Y}) = \frac{\mathcal{P}(Y \subseteq \boldsymbol{Y}, A \subseteq \boldsymbol{Y})}{\mathcal{P}(A \subseteq \boldsymbol{Y})} = \frac{\det(K_{Y \cup A})}{\det(K_A)}\,. \tag{2.40}$$

Just as in Lemma 2.4, for any set B such that $A \cap B = \emptyset$, application of Schur's identity yields an expression for the determinant of $K_{B \cup A}$:

$$\det(K_{B \cup A}) = \det(K_A)\det(K'_B)\,, \tag{2.41}$$

where $K' = K_{\bar{A}} - K_{\bar{A},A} K_A^{-1} K_{A,\bar{A}}$. This means that Equation (2.40) simplifies to:

$$\mathcal{P}(Y \subseteq \boldsymbol{Y} \mid A \subseteq \boldsymbol{Y}) = \frac{\det(K_A)\det(K'_Y)}{\det(K_A)} = \det(K'_Y)\,. \tag{2.42}$$

□

2.2.4 Sampling

Sampling algorithms to draw $\boldsymbol{Y} \sim \mathcal{P}_L$ rely on the eigendecomposition of L (or K). They are based on the fact that any DPP can be written as a mixture of *elementary* DPPs, where the mixture weights are products of L's eigenvalues.

Definition 2.6. *A DPP is elementary if its marginal kernel's eigenvalues are all* $\in \{0, 1\}$.

An elementary DPP is simpler than a general DPP in that it only places probability mass on sets of a fixed size.

Lemma 2.7. *Under an elementary DPP with* k *non-zero eigenvalues, the probability of* $\boldsymbol{Y} = Y$ *is zero for all* Y *where* $|Y| \neq k$.

Proof. See Kulesza (2012, Lemma 2.3). □

Let V be a set of orthonormal vectors. Let V^J indicate selection of the columns of V that correspond to the indices in a set J. We will write $\mathcal{P}^{V^J}$ to denote an elementary DPP with marginal kernel K^{V^J}:

$$K^{V^J} = \sum_{j : j \in J} \boldsymbol{v}_j \boldsymbol{v}_j^\top = V^J (V^J)^\top\,, \qquad \mathcal{P}^{V^J}(Y \subseteq \boldsymbol{Y}) = \det(K_Y^{V^J})\,. \tag{2.43}$$

Given this notation, we can now express $\mathcal{P}_L$ as a mixture of elementary DPPs.

Lemma 2.8. *A DPP with kernel L that eigendecomposes as $\sum_{i=1}^{N} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$, is equal to the following mixture of elementary DPPs:*

$$\mathcal{P}_L(\boldsymbol{Y} = Y) = \sum_{J: J \subseteq \{1,\dots,N\}} \mathcal{P}^{V^J}(\boldsymbol{Y} = Y) \prod_{j: j \in J} \frac{\lambda_j}{1+\lambda_j} \prod_{j: j \notin J} \left(1 - \frac{\lambda_j}{1+\lambda_j}\right). \tag{2.44}$$

Proof. See Kulesza (2012, Lemma 2.2) for a proof that:

$$\mathcal{P}_L(\boldsymbol{Y} = Y) = \frac{1}{\det(L+I)} \sum_{J: J \subseteq \{1,\dots,N\}} \mathcal{P}^{V^J}(\boldsymbol{Y} = Y) \prod_{j: j \in J} \lambda_j \,. \tag{2.45}$$

Rewriting $\det(L+I) = \prod_{i=1}^{N}(\lambda_i + 1)$ and pushing this into the J summation:

$$\mathcal{P}_L(\boldsymbol{Y} = Y) = \sum_{J: J \subseteq \{1,\dots,N\}} \mathcal{P}^{V^J}(\boldsymbol{Y} = Y) \prod_{j: j \in J} \frac{\lambda_j}{1+\lambda_j} \prod_{j: j \notin J} \frac{1}{1+\lambda_j} \,. \tag{2.46}$$

Since $1 - \frac{\lambda_j}{1+\lambda_j} = \frac{1}{1+\lambda_j}$, the result is obtained. □

This mixture decomposition suggests a two-step sampling algorithm for DPPs. First, sample a subset of the eigenvectors, V^J, by including vector j with probability $\frac{\lambda_j}{1+\lambda_j}$. Second, sample a set $Y \subseteq \mathcal{Y}$ according to P^{V^J}. Algorithm 1, due to Hough, Krishnapur, Peres, and Virág (2006), fleshes out this two-step procedure. Assuming that the eigendecomposition of L is given as input, then the most expensive part of the algorithm is in the second step, where we modify V such that it is orthogonal to the indicator vector $\boldsymbol{e}_{y_i}$. This requires running the Gram-Schmidt process, an $O(Nk^2)$ operation. Overall, that makes Algorithm 1's runtime $O(Nk^3)$.

It is possible to improve this runtime by avoiding the V updating. More concretely, on each iteration we can lazily apply a simple conditioning formula to the diagonal of the elementary DPP's marginal kernel. Algorithm 2 outlines this approach. The first step, the selection of J, is the same as in Algorithm 1. The second step only requires $O(Nk^2)$ time though. To prove the correctness of Algorithm 2, we rely on the following corollary and lemma.

Corollary 2.9. *Given a marginal kernel K, the conditional marginal kernel $K^{\{i\}}$, with minors satisfying $\mathcal{P}(Y \subseteq \boldsymbol{Y} \mid i \in \boldsymbol{Y}) = \det(K_Y^{\{i\}})$, can be computed from K by the rank-1 update:*

$$K^{\{i\}} = K_{\overline{\{i\}}} - \frac{1}{K_{ii}} K_{\overline{\{i\}},i} K_{i,\overline{\{i\}}} \,, \tag{2.47}$$

Algorithm 1: $O(Nk^3)$ DPP Sampling

Input: eigendecomp. $V\Lambda V^\top$ of L
$J \leftarrow \emptyset$
for $j = 1, \ldots, N$ **do**
  $J \leftarrow J \cup \{j\}$ with prob. $\frac{\lambda_j}{1+\lambda_j}$
$V \leftarrow V_{:,J}$
$Y \leftarrow \emptyset$
while $|Y| < |J|$ **do**
  for $i = 1, \ldots, N$ **do**
    $z_i \leftarrow \sum_{\boldsymbol{v} \in V} \boldsymbol{v}(i)^2$
  Select y_i with $Pr(y_i) = \frac{z_{y_i}}{|J|-|Y|}$
  $Y \leftarrow Y \cup \{y_i\}$
  $j \leftarrow \arg\max_{j'} V_{y_i,j'}$
  $\boldsymbol{w} \leftarrow V_{:,j}$
  $V \leftarrow V_{:,\overline{\{j\}}}$
  $V \leftarrow V - \frac{1}{\boldsymbol{v}_j(y_i)} \boldsymbol{w} V_{y_i,:}$
  Gram-Schmidt(V)
Output: Y

Algorithm 2: $O(Nk^2)$ DPP Sampling

Input: eigendecomp. $V\Lambda V^\top$ of L
$J \leftarrow \emptyset$
for $j = 1, \ldots, N$ **do**
  $J \leftarrow J \cup \{j\}$ with prob. $\frac{\lambda_j}{1+\lambda_j}$
$V \leftarrow V_{:,J}$
$Y \leftarrow \emptyset$
for $i = 1, \ldots, N$ **do**
  $z_i \leftarrow \sum_{\boldsymbol{v} \in V} \boldsymbol{v}(i)^2$
while $|Y| < |J|$ **do**
  Select y_i with $Pr(y_i) = \frac{z_{y_i}}{|J|-|Y|}$
  $Y \leftarrow Y \cup \{y_i\}$
  $\boldsymbol{r}_{|Y|} \leftarrow V V_{y_i,:}^\top$
  for $j = 1, \ldots, |Y| - 1$ **do**
    $\boldsymbol{r}_{|Y|} \leftarrow \boldsymbol{r}_{|Y|} - \frac{r_j(y_i)}{r_j(y_j)} \boldsymbol{r}_j$
  $\boldsymbol{z} \leftarrow \boldsymbol{z} - \frac{1}{z_{y_i}} \boldsymbol{r}_{|Y|}^2$
Output: Y

Figure 2.1: Two DPP sampling algorithms, both based on the elementary DPP decomposition of L. The slower algorithm computes an orthonormal basis for the subspace V orthogonal to $\boldsymbol{e}_{y_i}$ for each point selected. The faster algorithm relies on lazy updates of the marginals based on a K-conditioning formula.

assuming $K_{ii} \neq 0$. The notation $K_{\overline{\{i\}},i}$ indicates the vector composed of the ith column of K, without its ith element.

Proof. This follows directly from Lemma 2.5 with $A = \{i\}$. □

Lemma 2.10. *Let $V \in \mathbb{R}^{N \times k}$ be the eigenvectors of an elementary DPP's marginal kernel: $K = VV^\top$. Let $Y \subseteq \mathcal{Y}$ be size-k and arbitrarily order its elements $[y_1, \ldots, y_k]$. Use Y_ℓ to denote the subset $\{y_1, \ldots, y_\ell\}$, with $Y_0 = \emptyset$. Then we can express the (s,t) entry of the conditional marginal kernel as follows:*

$$K^{Y_\ell}_{st} = K_{st} - \sum_{j=1}^{|Y_\ell|} \frac{K^{Y_{j-1}}_{sy_j} K^{Y_{j-1}}_{ty_j}}{K^{Y_{j-1}}_{y_j y_j}}, \tag{2.48}$$

where K^{Y_ℓ} is the marginal kernel conditioned on the inclusion of Y_ℓ.

Proof. We will proceed by induction on ℓ. For $\ell = 0$ the statement of the lemma reduces to $K^{\emptyset}_{st} = K_{st}$. This is trivially true, as the original marginal kernel is already conditioned on the inclusion of the empty set. For the inductive step, we assume the lemma holds for $\ell - 1$ and show that this implies it is also true for ℓ. From Corollary 2.9 we have the following expression for K^{Y_ℓ}:

$$K^{Y_\ell}_{st} = K^{Y_{\ell-1}}_{st} - \frac{K^{Y_{\ell-1}}_{sy_\ell} K^{Y_{\ell-1}}_{ty_\ell}}{K^{Y_{\ell-1}}_{y_\ell y_\ell}}. \tag{2.49}$$

Moving the $K^{Y_{\ell-1}}_{st}$ term to the lefthand side yields:

$$K^{Y_\ell}_{st} - K^{Y_{\ell-1}}_{st} = -\frac{K^{Y_{\ell-1}}_{sy_\ell} K^{Y_{\ell-1}}_{ty_\ell}}{K^{Y_{\ell-1}}_{y_\ell y_\ell}}. \tag{2.50}$$

The righthand side here is exactly the same as the difference between the ℓ and $\ell - 1$ cases in the statement of the lemma. □

We can now prove the correctness of Algorithm 2.

Theorem 2.11. *Given the eigendecomposition of a PSD matrix $L = \sum_{i=1}^{N} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$, Algorithm 2 samples $\boldsymbol{Y} \sim \mathcal{P}_L$.*

Proof. Lemma 2.8 establishes that the mixture weight for elementary DPP $\mathcal{P}^{V^J}$ is:

$$\prod_{j:j\in J} \frac{\lambda_j}{1+\lambda_j} \prod_{j:j\notin J} \left(1 - \frac{\lambda_j}{1+\lambda_j}\right). \tag{2.51}$$

The first loop of Algorithm 2 selects V^J with exactly this probability. Thus, the first loop of Algorithm 2 selects elementary DPP $\mathcal{P}^{V^J}$ with probability equal to its mixture component. Line 5 sets $V = V^J$, and all that remains to show is that the rest of the algorithm samples $\boldsymbol{Y} \sim \mathcal{P}^V$. In what follows, we will use K to refer to the marginal kernel of the selected elementary DPP.

As in Lemma 2.10, let Y_ℓ be the set $\{y_1, \ldots, y_\ell\}$. That is, the items selected during the first ℓ iterations of Line 9's while-loop. First, by induction on $|Y|$, we show that at Line 15 the variable $\boldsymbol{r}_{|Y|}$ is equal to $K_{:,y_{|Y|}}^{Y_{|Y|-1}}$. In other words, we show that $\boldsymbol{r}_{|Y|}$ is the $y_{|Y|}$th column of the marginal kernel conditioned on inclusion of $\{y_1, \ldots, y_{|Y|-1}\}$. For the base case, $|Y| = 1$, the result is immediate; Line 12 sets $\boldsymbol{r}_1$ to $K_{:,y_1}$, and the for-loop at Lines 13 and 14 does not change it. For the inductive case, $|Y| = \ell$, we assume $\boldsymbol{r}_j = K_{:,y_j}^{Y_{j-1}}$ for all $j \leq \ell - 1$. Line 12 sets $\boldsymbol{r}_\ell = K_{:,y_\ell}$ and Line 13's for-loop updates $\boldsymbol{r}_\ell$ exactly according to Equation (2.49) from Lemma 2.10. Thus, at Line 15 we have: $\boldsymbol{r}_\ell = K_{:,y_\ell}^{Y_{\ell-1}}$, as desired.

Given this invariant on $\boldsymbol{r}_{|Y|}$, we now show that at Line 10 the variable z_i is always equal to the marginal probability of selecting item i, conditioned on the current selection Y. That is, it is always the diagonal entry of the conditional marginal kernel: $z_i = K_{ii}^Y$. We again proceed by induction on $|Y|$. For the base case, $Y = \emptyset$, the marginal probability of an item i is K_{ii}. Since Line 8 initializes $z_i = V_{i,:}V_{i,:}^\top = K_{ii}$, the base case is trivially true. For the inductive case, $|Y| = \ell$, the inductive hypothesis says that $z_i = K_{ii}^{Y_{\ell-1}}$ at Line 10 during the ℓth execution of the while-loop. Then it is clear that Line 15 updates z_i by applying Equation (2.49) for $s = t$. Thus, we have $z_i = K_{ii}^{Y_\ell}$ at Line 10 during the next iteration of the while-loop, as desired.

Given this invariant on z_i, all that remains to show is that the probability of selecting element i to be the ℓth item in Y is $z_i/(k - \ell + 1)$, as in Line 10. According to Lemma 2.7, Y must have $|Y| = |J|$ at the end of the algorithm. Thus, when $\ell - 1$ items have been selected, there are still $k - \ell + 1$ ways in which item i could be added to the final selection: it could be added as element ℓ, or $\ell + 1$, etc. Thus, Line 10 correctly normalizes z_i to account for all of these possibilities. □

2.2.5 MAP estimation

Up to this point each of the inference tasks described—normalizing, marginalizing, conditioning, and sampling—can be performed in roughly $O(N^3)$ time. The task of finding the MAP (or mode) of a DPP is more difficult. In fact, it is known to be NP-hard. This was shown by Ko, Lee, and Queyranne (1995), whose work focuses on the equivalent problem of selecting a subset of Gaussian random variables such that the entropy of the selected set is as large as possible. They start by assuming that the covariance matrix Σ of the variables is known. Based on Σ, the entropy of any subset Y of the variables can be computed according to:

$$H(Y) = \frac{1}{2} \ln \left((2\pi e)^{|Y|} \det(\Sigma_Y) \right) . \tag{2.52}$$

For a fixed set size, $|Y| = k$, this is proportional to $\log\det(\Sigma_Y)$. Ko et al. (1995) reduce the NP-complete "stable set" problem to the problem of maximizing $\log\det(\Sigma_Y)$ subject to the cardinality constraint $|Y| = k$. Specifically, the stable set problem asks: given an N-vertex graph G and an integer $k < N$, decide whether G contains a stable set with k vertices. A stable set is defined as a subset S of G's nodes such that there are no edges between any of the nodes in S. The transformation of this graph problem into a determinant problem is accomplished by defining a PSD matrix based on the graph's edges:

$$\Sigma_{ij} = \begin{cases} 3N & \text{if } i = j\,, \\ 1 & \text{if } (i,j) \text{ is an edge in } G\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{2.53}$$

Ko et al. (1995) show that finding a size-k minor of value greater than $(1 - (3N)^{-2})(3N)^k$ corresponds to finding a stable set of size k. Thus, the problem of finding the largest size-k determinant of a PSD matrix is NP-hard, even if the matrix only has entries with values in $\{3N, 1, 0\}$. Ko et al. (1995) extend this reasoning to also show NP-hardness in the unconstrained setting, which is equivalent to the problem of finding the mode of a DPP with kernel Σ. They also experiment with a branch-and-bound search algorithm for finding the best subset. It relies primarily on the eigenvalue interlacing property to define an upper bound.

Definition 2.12. Eigenvalue interlacing property*: Let M be an $N \times N$ symmetric matrix. Denote the $r \times r$ leading principal submatrix of M as $M[r]$ and its eigenvalues as*

$\lambda_1(M[r]) \leq \lambda_2(M[r]) \leq \ldots \leq \lambda_r(M[r])$. *Then for any two matrices* $M[r]$ *and* $M[r+1]$, *and any index* $i \in \{1, 2, \ldots, r\}$, *the following inequality holds:*

$$\lambda_i(M[r+1]) \leq \lambda_i(M[r]) \leq \lambda_{i+1}(M[r+1])\,. \tag{2.54}$$

In experiments, the branch-and-bound method is shown to be capable of finding optimal subsets for problems of size up to $N = 75$, but takes a relatively long time to do so. The authors state: "In all of our experiments, the time spent by the heuristic is negligible [compared to the total runtime of the algorithm]." The methods we consider in Chapter 5 run in time comparable to Ko et al. (1995)'s "negligible" heuristics though.

The more recent work of Çivril and Magdon-Ismail (2009) strengthens the hardness results developed by Ko et al. (1995). In particular, they show that no PTAS exists for the problem of finding a maximum volume submatrix. That is, their proof precludes the existence of an algorithm that, given any error tolerance ϵ, produces a solution within a factor $1 - \epsilon$ of optimal. Kulesza (2012) adapts this proof to show that an approximation ratio of $\frac{8}{9} + \epsilon$ is NP-hard for the problem of finding the mode of a DPP.

Theorem 2.13. *Let* DPP-MODE *be the optimization problem of finding, for an* $N \times N$ *PSD matrix* L *indexed by elements of* $\mathcal{Y}$, *the maximum value of* $\det(L_Y)$ *over all* $Y \subseteq \mathcal{Y}$. *It is NP-hard to approximate* DPP-MODE *to a factor of* $\frac{8}{9} + \epsilon$.

Proof. See Kulesza (2012, Theorem 2.4). □

For the cardinality-constrained variant of the problem where $|Y| = k$, Çivril and Magdon-Ismail (2009) propose an approximation algorithm guaranteed to find a solution within a factor $O\left(\frac{1}{k!}\right)$ of optimal. This algorithm is exactly the greedy algorithm of Nemhauser, Wolsey, and Fisher (1978), for the function $\log\det$. In Chapter 5 we discuss this algorithm in more detail and compare it empirically with our own MAP estimation algorithms.

2.2.6 Likelihood maximization

Consider the problem of fitting a DPP to data. For probabilistic models, one standard way to find model parameters is to maximize the log-likelihood of the data. In

the most unrestricted setting, the model parameters for a DPP are the entries of L or K. Given data consisting of n example subsets, $\{Y_1, \ldots, Y_n\}$, where $Y_i \subseteq \mathcal{Y}$ for all i, the log-likelihood maximization problem is:

$$\max_{L} \sum_{i=1}^{n} [\log\det(L_{Y_i}) - \log\det(L+I)] \quad \text{s.t. } L \succeq 0\,. \tag{2.55}$$

Unfortunately, this objective is not concave. The function $f(M) = \log\det(M)$ is concave for PSD M, implying that the log-likelihood is a difference of two concave functions, but this does not make log-likelihood overall concave. Applying Equation (2.16) gives a form of the log-likelihood objective in terms of K:

$$\max_{K} \sum_{i=1}^{n} \log\left(|\det(K - I_{\overline{Y}_i})|\right) \quad \text{s.t. } K \succeq 0,\ I - K \succeq 0\,. \tag{2.56}$$

This looks simpler, but it is not concave either. The matrix $K - I_{\overline{Y}_i}$ can be non-PSD, and $\log\det$ is only concave when restricted to the PSD cone. No algorithm is currently known for efficiently finding a global optimum of DPP log-likelihood. In Chapter 6 though, we derive an expectation-maximization algorithm to find a *local* optimum.

More restricted likelihood maximization settings are also of interest. For instance, instead of allowing the entries of L or K to take on arbitrary values, suppose we have a fixed set of kernels $S_1, \ldots, S_r$ and require that L be a weighted sum of these. Maximizing likelihood under this constraint is conjectured to be NP-hard.

Conjecture 2.14. *Given a sequence of $N \times N$ PSD kernels $S_1, \ldots, S_r$, indexed by the elements of $\mathcal{Y}$, and a sequence of subsets $Y_1, \ldots, Y_n$ of $\mathcal{Y}$, finding $\boldsymbol{\theta} \in \mathbb{R}^r$, $\boldsymbol{\theta} \geq 0$ to maximize:*

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} [\log\det(S(\boldsymbol{\theta})_{Y_i}) - \log\det(S(\boldsymbol{\theta}) + I)] \quad \text{s.t. } S(\boldsymbol{\theta}) = \sum_{i=1}^{r} \theta_i S_i \tag{2.57}$$

is NP-hard.

Proof. Kulesza (2012, Conjecture 4.1) provides a partial reduction of the NP-complete exact 3-cover problem to this likelihood optimization. □

While restricting L to be a weighted sum of fixed kernels does not make likelihood maximization obviously tractable, there are other, tighter restrictions that do. We discuss these in Chapter 6.

2.2.7 Maximizing entropy

In addition to likelihood maximization, another common way of fitting a probabilistic model to data is to maximize its entropy, subject to certain feature-matching constraints. In the case of the exponential family of distributions, maximizing likelihood is equivalent to maximizing entropy, subject to the constraint that expected feature values under the model equal their empirical counts. While such a likelihood-entropy equivalence is not known to hold for DPPs, maximizing entropy is nevertheless a reasonable strategy for estimating DPP parameters, since it can be justified from the perspective of minimizing the prior knowledge put into the distribution. In terms of K, the entropy is:

$$H(K) = \sum_{Y:Y\subseteq\mathcal{Y}} |\det(K - I_{\overline{Y}})| \log(|\det(K - I_{\overline{Y}})|)\,. \tag{2.58}$$

Lyons (2003) conjectures that $H(K)$ is concave in K, and numerical simulation supports this. However, no proof is known.

Maximizing entropy to fit a DPP model is in some ways more difficult than maximizing likelihood. For instance, no efficient way of exactly computing the exponential-size sum in the entropy expression is known. While Section 2.2.1 exploits the multilinearity of the determinant to write $\sum_{Y:Y\subseteq\mathcal{Y}} \det(L_Y)$ as the single determinant $\det(L + I)$, entropy's multiplication by a log term breaks the linearity. Thus, in Chapter 6 of this thesis we focus on likelihood maximization and defer exploration of entropy maximization to future work.

2.2.8 Computing expectations

A more general inference task that also often seems to be difficult for DPPs is computing expected values. Since the distribution defined by a DPP has 2^N values, computing expectations naïvely involves an exponential-size sum. There are a few interesting quantities for which this sum simplifies though. For instance, the expected set size can be computed simply by taking the trace of the marginal kernel: $\mathbb{E}_{Y\sim\mathcal{P}_L}[|Y|] = \mathrm{tr}(K)$ (Kulesza, 2012, Equation 2.34). But for many other quantities no exact, efficient way of computing the expectation is known. For instance, the formula for entropy discussed in the previous section is an example of an expectation that seems difficult to compute. Specifically, it is equivalent to $\mathbb{E}_{Y\sim\mathcal{P}_L}[\log \mathcal{P}_L(Y)]$.

Likewise, it is not known how to compute the following expectation exactly and efficiently:

$$\mathbb{E}_{Y\sim\mathcal{P}_L}[\mathcal{P}_L(Y)] = \frac{1}{\det(L+I)^2}\sum_{Y:Y\subseteq\mathcal{Y}}\det(L_Y)^2\,. \tag{2.59}$$

Being able to exactly and efficiently compute this quantity would mean we could normalize a distribution that places probability mass proportional to $\det(L_Y)^2$ on a set Y. This distribution would be more peaked around high-quality, diverse sets than the corresponding DPP, which could make it a superior tool for subset selection. Unfortunately though, even the normalization constant for this distribution seems difficult to compute exactly. In fact, we can show definitively that one type of expectation similar to that of Equation (2.59) is #P-hard to compute.

Theorem 2.15. *Let L and M be $N \times N$ PSD matrices indexed by elements of $\mathcal{Y}$. Then the expected value:*

$$\mathbb{E}_{Y\sim\mathcal{P}_L}[\mathcal{P}_M(Y)] = \frac{1}{\det(L+I)\det(M+I)}\sum_{Y:Y\subseteq\mathcal{Y}}\det(L_Y)\det(M_Y) \tag{2.60}$$

is #P-hard to compute.

Proof. We reduce from the #P-complete problem IMPERFECT-MATCHINGS (Valiant, 1979, Section 4, Problem 6). The input to this problem is a bipartite graph $G = (U, V, E)$ with N edges E between the nodes in U and the nodes in V. The output is the number of matchings of any size.

The reduction is as follows. For the ith edge, let E_{i1} and E_{i2} denote the nodes from U and V, respectively. Let the $N \times N$ matrix L have entries $L_{ij} = \mathbb{1}(E_{i1} = E_{j1})$, indicating whether edges i and j share the same U-node. Similarly, let $M_{ij} = \mathbb{1}(E_{i2} = E_{j2})$, indicating whether edges i and j share the same V-node.

First, we show that $\det(L_Y)\det(M_Y) = 0$ whenever Y does not correspond to a matching. For any set $Y \subseteq \{1, \ldots, N\}$ with $i, j \in Y$, if E_i and E_j share an endpoint then either $L_{:,i} = L_{:,j}$ or $M_{:,i} = M_{:,j}$. That is, identical endpoints correspond to two identical columns in L or M. Since the determinant of a matrix whose columns are linearly dependent is zero, this means that $\det(L_Y)$ or $\det(L_M)$ is zero whenever Y contains indices of edges with common vertices.

Now we show that $\det(L_Y)\det(M_Y) = 1$ whenever Y does correspond to a matching. If the indices in Y correspond to edges that all have distinct U-nodes, then

$L_{ij} = 0$ for all $i, j \in Y$ where $i \neq j$. Thus, $L_Y = I$ and $\det(L_Y) = 1$. The same argument applies to M for distinct V-nodes.

The expectation from the statement of the theorem corresponds to a sum over all subsets of edges, and we have shown that each term in the sum contributes 1 for a matching and 0 for a non-matching. Multiplying the expectation by the inverse of the normalizing term, $\det(L + I) \det(M + I)$, yields a count of the number of matchings of any size. □

Despite the negative results for exactly computing expectations under DPPs, there is of course always the option of approximating. Since it is easy to sample from a DPP, any of these expectations can be approximated by drawing samples from $\mathcal{P}_L$.

2.3 Closure

The class of DPPs is closed under the operations of scaling, restricting, complementing, and conditioning. This means that for each of these operations it is possible to write the resulting set probabilities as a DPP with some modified kernel L' such that $\mathcal{P}_{L'}(\boldsymbol{Y} = Y) = \det(L'_Y)/\det(L' + I)$. For the conditioning operation, the corresponding L' can be found in Table 2.2. For the other operations, the modified kernels are given below; see Kulesza (2012, Section 2.3) for proofs. In each setting we assume that there is a variable $\boldsymbol{Y}$ distributed as a DPP with kernel L and marginal kernel K.

Scaled kernel: Scaling L by any non-negative constant γ results in a related PSD matrix γL, which is a valid DPP kernel. Similarly, scaling K by any $\gamma \in [0, 1]$ yields a PSD matrix with eigenvalues in $[0, 1]$, and thus another valid DPP kernel. (Note though that γL and γK almost always describe different DPPs.)

Restricted kernel: The restricted variable $\boldsymbol{Y} \cap A$, for $A \subseteq \mathcal{Y}$ is distributed as a DPP, with marginal kernel K_A. The corresponding non-marginal kernel can be found by applying Equation (2.14): $K_A(I - K_A)^{-1}$.

Complement kernel: The complement variable $\mathcal{Y} \setminus \boldsymbol{Y}$ is distributed as a DPP, with marginal kernel $\overline{K} = I - K$. With this complement identity, we can also now express the marginal probability of any partial assignment as a product of two

determinants:

$$\mathcal{P}(B \subseteq \boldsymbol{Y}, A \cap \boldsymbol{Y} = \emptyset) = \mathcal{P}(B \subseteq \boldsymbol{Y})\mathcal{P}(A \cap \boldsymbol{Y} = \emptyset \mid B \subseteq \boldsymbol{Y}) \tag{2.61}$$

$$= \det(K_B)\det(I - K_A^B)\,. \tag{2.62}$$

2.4 Dual representation

The normalizing, marginalizing, conditioning, and sampling operations discussed in Section 2.2 all require time polynomial in N. As N grows large, these operations can become prohibitively expensive. However, in the case where each item i is associated with a feature vector $B_i \in \mathbb{R}^{D\times 1}$, as in Section 1.2, a simple alternative exists for reducing runtime complexity. More concretely, we can replace N with D. In settings where $D \ll N$, this results in substantial savings. (In Chapter 4, we discuss the setting where *both* N and D are large.)

Let $L = B^\top B$ and consider the $D \times D$ matrix $C = BB^\top$, which is also PSD. We will refer to this matrix as the dual kernel. The eigendecompositions of L and C are closely related: their non-zero eigenvalues are identical and the eigenvectors of L convert to those of C via the linear map $B^\top$.

Proposition 2.16. *The eigendecomposition of C is:*

$$C = \hat{V}\Lambda\hat{V}^\top = \sum_{i=1}^{D} \lambda_i \hat{\boldsymbol{v}}_i \hat{\boldsymbol{v}}_i^\top \tag{2.63}$$

if and only if the eigendecomposition of L is:

$$L = \sum_{i=1}^{D} \lambda_i \left(\frac{1}{\sqrt{\lambda_i}} B^\top \hat{\boldsymbol{v}}_i\right)\left(\frac{1}{\sqrt{\lambda_i}} B^\top \hat{\boldsymbol{v}}_i\right)^\top . \tag{2.64}$$

Proof. See Kulesza (2012, Proposition 3.1). □

By exploiting this eigendecomposition relationship, we can rewrite many DPP inference operations in terms of C, replacing their N dependence with a D dependence. The following subsections give an overview of these reductions. Those that are not explained in detail here can be found in Kulesza (2012, Section 3.3).

2.4.1 Normalizing

DPP normalization can be transformed from an $O(N^\omega)$ operation to an $O(D^\omega)$ operation by applying the identity $\det(L + I) = \det(C + I)$.

2.4.2 Marginalizing

The eigendecomposition of C takes time $O(D^\omega)$ to compute. Exploiting Equation (2.15), which establishes that the eigendecompositions of L and the marginal kernel K are nearly identical, we can compute any single entry K_{ij} in $O(D^2)$ time, assuming the eigendecomposition of C is known. Extending this, for a set Y of size k, the $\frac{k(k+1)}{2}$ entries needed for the submatrix K_Y can be computed in $O(D^2k^2)$ time. This makes the operation of computing a size-k marginal $O(D^2k^2 + k^\omega)$.

2.4.3 Conditioning

Recall the conditioning equations from Table 2.2. Expressions such as the one for marginal exclusion, Equation (2.22), $K^{\neg A} = I - (L_{\bar{A}} + I)^{-1}$, are not particularly conducive to replacement of L by the dual kernel C. If $L_{\bar{A}}$ is full-rank, then $|\bar{A}|$ is $\leq D$ and there would be no savings from using C in place of L. If on the other hand $L_{\bar{A}}$ is not full-rank, then the addition of the identity matrix in this formula changes the eigendecomposition of L significantly in ways that cannot be easily translated to C; the transformation of the eigenvalues is trivial (the identity adds 1 to each), but the change in the eigenvectors formerly associated with zero eigenvalues is not easy to characterize.

For other conditioning formulas, such as the one for inclusion of a set A, Equation (2.20), $L^A = L_{\bar{A}} - L_{\bar{A},A}L_A^{-1}L_{A,\bar{A}}$, it is more clear how to leverage the dual kernel to improve computational complexity. In this case, the translation from L to L^A can be written as a linear map from B to a new matrix B^A. Lemma 2.17 gives the details. Given B^A, the conditional dual kernel C^A is simply $B^A(B^A)^\top$.

Lemma 2.17. *If $L = B^\top B$, then we can write the kernel conditioned on the inclusion of the set A as $L^A = (B^A)^\top B^A$, where:*

$$Z^A = I - B_A(B_A^\top B_A)^{-1}B_A^\top \tag{2.65}$$

$$B^A = Z^A B_{\bar{A}}. \tag{2.66}$$

Proof. Substituting B into the formula developed in Lemma 2.4:

$$L^A = L_{\bar{A}} - L_{\bar{A},A} L_A^{-1} L_{A,\bar{A}} \tag{2.67}$$

$$= B_{\bar{A}}^\top B_{\bar{A}} - B_{\bar{A}}^\top B_A \left(B_A^\top B_A\right)^{-1} B_A^\top B_{\bar{A}} \tag{2.68}$$

$$= B_{\bar{A}}^\top \left(I - B_A (B_A^\top B_A)^{-1} B_A^\top\right) B_{\bar{A}} \tag{2.69}$$

$$= B_{\bar{A}}^\top Z^A B_{\bar{A}}\,. \tag{2.70}$$

The matrix Z^A is a projection matrix, which implies that it is idempotent: $(Z^A)^2 = Z^A$. Thus, $L^A = B_{\bar{A}}^\top (Z^A)^2 B_{\bar{A}}$. Since the Z^A matrix is also clearly symmetric, we have the desired result: $L^A = (B^A)^\top B^A$. □

We could compute B^A and use it to get C^A via the product $B^A(B^A)^\top$. For small k and large N though, it is more efficient to compute C^A based on the fact that $B^A(B^A)^\top = Z^A C Z^A$. We still have to compute Z^A, which takes $O(D^2k^2 + k^\omega)$ time. But then, instead of an $O(D^2(N-k))$ matrix multiplication, only an $O(D^\omega)$ one is required. Overall, this means that C^A can be obtained in $O(D^\omega + D^2k^2 + k^\omega)$ time.

2.4.4 Sampling

Recall the two algorithms from Section 2.2.4. The slower one, Algorithm 1, has a complexity of $O(Nk^3)$, where k is the size of the sampled set. Kulesza (2012, Section 3.3.3) shows how to adapt this algorithm to use the dual kernel C in place of L, with a resulting complexity of $O(NDk^2 + D^2k^3)$. To better compare these two complexities, note that a sample from a DPP will not be larger than the rank of the DPP's kernel, implying $k \leq D$. This makes the $O(Nk^3)$ of the original algorithm better than the complexity of the dual algorithm. Still, the dual algorithm might be a better choice if one does not have the eigendecomposition of L pre-computed. Including the cost of the initial eigendecompositions in the runtime analysis, we have complexities of $O(N^\omega + Nk^3)$ for the original algorithm versus $O(D^\omega + NDk^2 + D^2k^3)$ for the dual version.

For the faster sampling algorithm from Section 2.2.4, Algorithm 2, the complexity without including the eigendecomposition is $O(Nk^2)$, but with the eigendecomposition it is $O(N^\omega + Nk^2)$. To create a dual variant of this algorithm, we can use

many of the same techniques as Kulesza (2012) applies to create a dual for Algorithm 1. We start by assuming that instead of an eigendecomposition of L, we have an eigendecomposition of C. The initial step of sampling the set J does not change, as C and L have identical non-zero eigenvalues. To compute the initial z_i, we borrow an $O(NDk)$ procedure that Kulesza (2012) uses to compute the same quantity for the dual version of Algorithm 1. Given these, all that remains to convert is Line 12 of Algorithm 2. This line, $\boldsymbol{r}_{|Y|} \leftarrow V V_{y_i,:}^\top$, computes $K_{:,y_i}$. From the dual marginalization section, Section 2.4.2, we know that it is possible to compute any single entry of K in $O(D^2)$ time, given C's eigendecomposition. The exact formula for $K_{:,y_i}$ is:

$$K_{:,y_i} = \sum_{i=1}^{D} \frac{\lambda_i}{\lambda_i + 1} \left(\frac{1}{\sqrt{\lambda_i}} B_{y_i}^\top \hat{\boldsymbol{v}}_i \right) \left(\frac{1}{\sqrt{\lambda_i}} B^\top \hat{\boldsymbol{v}}_i \right) . \tag{2.71}$$

Applying this, Line 12 can be re-written as an $O(ND^2)$ operation. This dominates the overall algorithm cost, yielding an $O(ND^2k)$ dual sampling algorithm. Algorithm 3 compiles the modifications described in this section to summarize the full sampling procedure.

Algorithm 3: Dual DPP Sampling

Input: B and eigendecomp. $\hat{V}\Lambda\hat{V}^\top$ of C
$J \leftarrow \emptyset$
for $j = 1, \ldots, D$ **do**
  $J \leftarrow J \cup \{j\}$ with prob. $\frac{\lambda_j}{1+\lambda_j}$
$\hat{V} \leftarrow \left\{ \frac{\hat{\boldsymbol{v}}_j}{\sqrt{\hat{\boldsymbol{v}}_j^\top C \hat{\boldsymbol{v}}_j}} \right\}_{j \in J}$
$Y \leftarrow \emptyset$
for $i = 1, \ldots, N$ **do**
  $z_i \leftarrow \sum_{\hat{\boldsymbol{v}} \in \hat{V}} \left(\hat{\boldsymbol{v}}^\top B_i \right)^2$
while $|Y| < |J|$ **do**
  Select y_i with $Pr(y_i) = \frac{z_{y_i}}{|J|-|Y|}$
  $Y \leftarrow Y \cup \{y_i\}$
  $\boldsymbol{r}_{|Y|} \leftarrow$ Equation (2.71)
  for $j = 1, \ldots, |Y| - 1$ **do**
    $\boldsymbol{r}_{|Y|} \leftarrow \boldsymbol{r}_{|Y|} - \frac{r_j(y_i)}{r_j(y_j)} \boldsymbol{r}_j$
  $\boldsymbol{z} \leftarrow \boldsymbol{z} - \frac{1}{z_{y_i}} \boldsymbol{r}_{|Y|}^2$
Output: Y

2.5 Quality-similarity decomposition

We introduce in this section a small amount of additional notation common to most practical DPP work. Specifically, it is standard to separate out notions of item quality and item similarity. Section 1.2 aliased these two by assuming that each item i is entirely characterized by a feature vector $B_i \in \mathbb{R}^{D \times 1}$. We can decompose B_i into two parts though: its magnitude $\|B_i\|_2$, which represents feature quality q_i, and its direction $\phi_i \in \mathbb{R}^{D \times 1}, \|\phi_i\|_2 = 1$, which models item similarity. Defining two $N \times N$ matrices to summarize this information, a diagonal quality matrix Q and a similarity matrix S, we have:

$$B_i = q_i \phi_i \,, \quad Q_{ii} = q_i \,, \quad S_{ij} = \phi_i^\top \phi_j \,, \quad L_{ij} = q_i \phi_i^\top \phi_j q_j = Q_{ii} S_{ij} Q_{jj} \,. \tag{2.72}$$

Given this decomposition, $L = QSQ$, we can re-write DPP probabilities as a product of a quality term and a diversity term:

$$\mathcal{P}_L(\boldsymbol{Y} = Y) = \left(\prod_{i:i \in Y} q_i^2 \right) \det(S_Y) \,. \tag{2.73}$$

For many of the real-world applications we discuss, this decomposition mirrors the form of the input data. For instance, in the case of image search, the quality of the ith image may depend on features such as contrast and sharpness that are not necessarily relevant to judging its similarity to other images. Hence, it is natural to separate out these quality features to generate a q_i score, then normalize the other features to create ϕ_i.

3

DPP Variants

This chapter discusses recent work that extends the basic DPP model discussed in Chapter 2 to create new models better suited to various practical settings. We examine cardinality-constrained and structured variants in detail here, as they will play a role in later chapters. We also touch on Markov DPPs and continuous DPPs, though the remainder of the thesis does not contain experimental results related to those two variants.

3.1 Cardinality-constrained DPPs

In practice, it is often useful to consider only subsets of a fixed size, k, rather than all 2^N subsets of $\mathcal{Y}$. This gives more control over the size of sets produced by DPP algorithms, which is important for two main reasons. First, for ease of use in applications where a cardinality constraint is standard. For example, document summarization systems are often limited to producing k-sentence summaries, so that the results are of a consistent, easily-digestible size. The second main reason that a model restricted to sets of a fixed size is needed is that it may inherently be a better fit for certain problems. For example, consider the problem of modeling the locations of spruce

trees on a particular plot of land. The resources of the land will most likely dictate that it can comfortably support approximately some fixed number of trees. Hence, a model that places substantial probability mass on much larger or much smaller numbers of trees will be a poor fit.

To address the need for models placing probability mass only on k-sets, Kulesza and Taskar (2011a) introduced k-DPPs. More concretely, a k-DPP with kernel L has probabilities:

$$\mathcal{P}_L^k(Y) = \frac{\det(L_Y)}{\sum_{\substack{Y':Y'\subseteq\mathcal{Y},\\ |Y'|=k}} \det(L_{Y'})}\,, \tag{3.1}$$

for $Y \subseteq \mathcal{Y}$ where $|Y| = k$, and $\mathcal{P}_L^k(Y) = 0$ otherwise. As with regular DPPs, many of the inference operations common to probabilistic models can be performed in polynomial time for k-DPPs. We require one additional definition to describe these k-based algorithms.

Definition 3.1. *The kth elementary symmetric polynomial on the values $\lambda_1, \ldots, \lambda_N$ is:*

$$e_k(\lambda_1, \ldots, \lambda_N) = \sum_{\substack{J:J\subseteq\{1,\ldots,N\},\\ |J|=k}} \prod_{j:j\in J} \lambda_j\,. \tag{3.2}$$

For a matrix L with eigenvalues $\lambda_1, \ldots, \lambda_N$, we will write $e_k(\lambda_1, \ldots, \lambda_N)$ as $e_k(L)$. Moreover, for the restriction to the first r eigenvalues $\lambda_1, \ldots, \lambda_r$, we will write $e_k^r(L)$. When the meaning is clear, we will also use e_k^r as shorthand for $e_k^r(L)$.

While the explicit formula in Definition 3.1 includes an exponential-size summation, it is possible to compute it efficiently due to the recurrence relation:

$$e_k^N = e_k^{N-1} + \lambda_N e_{k-1}^{N-1}\,. \tag{3.3}$$

Baker and Harwell (1996) exploit this relationship to create a summation algorithm that computes e_k^N and all of the lesser polynomials $e_1^N, \ldots, e_{k-1}^N$ in time $O(Nk)$ (Kulesza, 2012, Algorithm 7). All of the DPP inference methods can be modified to perform inference for k-DPPs by exploiting these elementary symmetric polynomials.

- **Normalizing**: The denominator in Equation (3.1) is equivalent to the elementary symmetric polynomial $e_k(L)$ (Kulesza, 2012, Proposition 5.1).

- **Marginalizing**: For regular DPPs, the marginal kernel K can be computed in $O(N^\omega)$ from L. Then each subsequent marginal for a set of size k requires just $O(k^\omega)$ time. Unfortunately, for k-DPPs no marginal kernel exists. However, it is still possible to get the marginal probability of a particular subset A as follows:

$$\mathcal{P}^k(A \subseteq \boldsymbol{Y}) = \frac{e^N_{k-|A|}(L^A)}{e^N_k(L)} \det(L_A) = e^N_{k-|A|}(L^A)\mathcal{P}^k_L(A) \tag{3.4}$$

  (Kulesza, 2012, Equation 5.29). Every marginal requires the computation of a unique conditional kernel L^A and its eigendecomposition. Thus, the cost of computing the marginal probability of a size-k set under a k-DPP is $O(N^2k + (N-k)^\omega)$. This can be somewhat improved for small sets by applying an alternative formula. For example, for singleton sets we have:

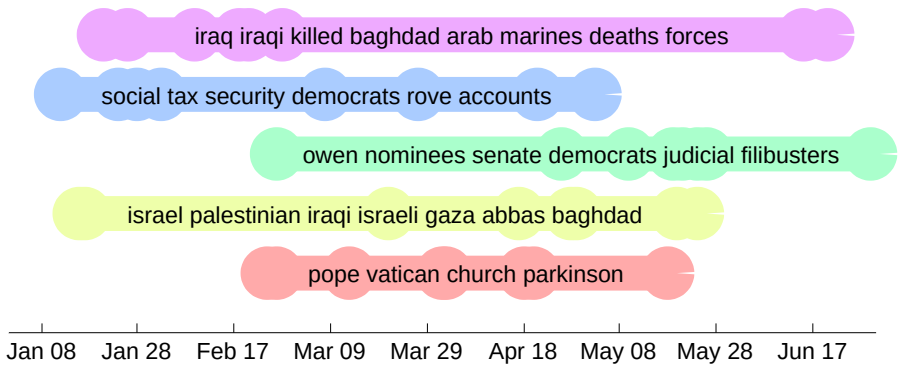
$$\mathcal{P}^k(i \in \boldsymbol{Y}) = \frac{1}{e^N_k(L)} \sum_{j=1}^{N} \lambda_j V_{ij}^2 e^{-j}_{k-1}(L)\,, \tag{3.5}$$

  where $e^{-j}_{k-1}(L) = e^{-j}_{k-1}(\lambda_1, \ldots, \lambda_{j-1}, \lambda_{j+1}, \ldots, \lambda_N)$ is the $(k-1)$-order elementary symmetric polynomial for all eigenvalues of L except λ_j (Kulesza, 2012, Equation 5.33). This formula requires $O(N^2k)$ time to compute, given the eigendecomposition of L. In fact, since the e^{-j}_{k-1} are the same for all i, we can compute all singleton marginals in $O(N^2k)$ time.

- **Conditioning**: Applying the same formulas as in the top half of Table 2.2 yields conditional kernels that can serve as k-DPP kernels.

- **Sampling**: Only the first part of the regular DPP sampling algorithms, the selection of the elementary DPP, has to change for k-DPPs. More concretely, instead of selecting from elementary DPPs of all sizes, it must change to select only from elementary DPPs of size k. The resulting algorithm (Kulesza, 2012, Algorithm 8) runs in time $O(Nk)$, assuming the eigendecomposition for L is given as input. This does not alter the overall complexity of the DPP sampling algorithms from Section 2.2.4, so the complexity of sampling from a k-DPP is the same as for sampling from a regular DPP.

- **Hard inference problems**: All of the operations that were shown or conjectured to be NP-hard for regular DPPs are similarly difficult for k-DPPs. (If



Feb 24: Parkinson's Disease Increases Risks to Pope
Feb 26: Pope's Health Raises Questions About His Ability to Lead
Mar 13: Pope Returns Home After 18 Days at Hospital
Apr 01: Pope's Condition Worsens as World Prepares for End of Papacy
Apr 02: Pope, Though Gravely Ill, Utters Thanks for Prayers
Apr 18: Europeans Fast Falling Away from Church
Apr 20: In Developing World, Choice [of Pope] Met with Skepticism
May 18: Pope Sends Message with Choice of Name

Figure 3.1: A set of five news threads. Above, the threads are shown on a timeline with the most salient words superimposed; below, the dates and headlines from the lowest thread are listed. The headlines indicate that the articles in this thread constitute a coherent news story.

switching to k-DPPs made any of these problems easy, then we could simply iterate over $k = 1, \ldots, N$ to solve the problem for regular DPPs.)

3.2 Structured DPPs

For some of the applications mentioned in the introduction, the number of items to choose from is exponential. For example, pose tracking, network routing, and motion summarization fall into this category, as do variants on the document summarization task. More concretely, consider a variant of the document summarization task that we will refer to as "news threading": given a collection of news articles from a certain time period, select k news "threads" where each thread covers one coherent news story and consists of R news articles describing the major events of that story. Figure 3.1 gives an example of this type of thread set.

If there are M news articles, then there are $O(M^R)$ possible news threads, and $O(M^{Rk})$ possible size-k subsets of threads. The inference tasks for the corresponding DPP can be prohibitively expensive even for moderate values of M, R, and k. The problem under consideration has some *structure* to it though, in that two threads can share a common sub-thread. To handle such settings, Kulesza and Taskar (2010) introduced structured DPPs. Essentially, their work shows that if we require that common components of threads share quality scores and similarity features, then we can efficiently handle a ground set of exponential size, $N = M^R$.

More concretely, let $\mathcal{Y}$ be the set of all structures and let $\boldsymbol{y} \in \mathcal{Y}$ denote a single structure with R parts $\{y_1, \ldots, y_R\}$. For the news threading example, $\boldsymbol{y}$ would be a single thread, the set of articles covering the major events of one news story, and the parts would be individual articles. Finally, define a set of factors F, where each factor $\alpha \in F$ represents some small subset of the parts of a structure. For instance, F could consist of singleton subsets where each α is an article, or F could consist of size-2 subsets where each α is a pair of articles. The key assumption we will make is that the model decomposes over these factors. Specifically, a structure $\boldsymbol{y}$ is characterized by quality score $q(\boldsymbol{y})$ and similarity features $\phi(\boldsymbol{y})$ that decompose as follows:

$$q(\boldsymbol{y}) = \prod_{\alpha \in F} q_\alpha(\boldsymbol{y}_\alpha)\,, \qquad \phi(\boldsymbol{y}) = \sum_{\alpha \in F} \phi_\alpha(\boldsymbol{y}_\alpha)\,. \tag{3.6}$$

These decompositions are very natural for many applications. Again, considering the news threading application, suppose that each α is a single article. If the similarity features for a single article $\phi_\alpha(\boldsymbol{y}_\alpha)$ are word counts, then the features for a thread are also word counts, since $\phi(\boldsymbol{y})$ is simply a sum over $\boldsymbol{y}$'s factors' features. Thus, $\phi(\boldsymbol{y})$ in this setting has a clear interpretation.

The factors F can be thought of as defining a graph consisting of two types of nodes, factor nodes and variable nodes, and an edge connecting each variable node to the factors in which it participates. As is the case for graphical models, structured DPP inference algorithms are efficient as long as F's graph has a low treewidth. That is, applying the junction tree algorithm (Lauritzen and Spiegelhalter, 1988) to convert it into a tree graph, the degree of the resulting tree factors is bounded by some small constant c. A small c is often quite sufficient for interesting models: for all of the experiments in subsequent sections we have $c = 2$.

Given the factor graph, all of the standard DPP inference methods can be con-

verted into structured DPP inference methods by exploiting the dual kernel C from Section 2.4 and the second-order message passing algorithm of Li and Eisner (2009). Second-order message passing allows us to compute C itself in $O(D^2M^cR)$ time (Kulesza, 2012, Section 6.3.1). Assuming C is given, the complexities of the various inference tasks are as listed below.

- **Normalizing**: By applying the $\det(C + I)$ equation from Section 2.4.1, structured DPPs can be normalized in $O(D^\omega)$ time.
- **Marginalizing**: By applying the dual kernel marginalization procedure described in Section 2.4.2, any size-k marginal of a structured DPP can be computed in $O(D^2k^2 + k^\omega)$ time. We can also compute the marginal of a particular part y_r in $O(D^2M^cR)$ time (Kulesza, 2012, Equation 6.49).
- **Conditioning**: By applying the dual kernel conditioning formula developed in Section 2.4.3, a structured DPP can be conditioned on the inclusion of a size-k set A in $O(D^\omega + D^2k^2 + k^\omega)$ time.
- **Sampling**: Assuming the eigendecomposition of C is known, a set of structures can be sampled from a structured DPP in $O(D^2k^3 + DM^cRk^2)$ time (Kulesza, 2012, Algorithm 11).

3.3 Markov DPPs

The simplest form of the subset selection task asks only for a single subset. Some applications though are better described as the sequential selection of *multiple* subsets. For example, consider another variant of the document summarization task: each day news outlets publish many articles, and from these we wish to select a few articles for a user to read. Automatically selecting a set of articles that is not only itself diverse, but that is also diverse compared to previously selected subsets is a task of practical importance: the additional between-day diversity allows for greater exploration of the article space. Affandi, Kulesza, and Fox (2012) address this type of problem by introducing Markov DPPs (M-DPPs).

More concretely, an M-DPP is a first-order, discrete-time, autoregressive point process that can be characterized by specifying a PSD matrix L, an initial distri-

bution, and a Markov transition distribution. For $\boldsymbol{Y}_t \subseteq \mathcal{Y}$, Affandi et al. (2012, Equation 13) define these distributions as:

$$\mathcal{P}(\boldsymbol{Y}_1 = Y_1) = \frac{\det(L_{Y_1})}{\det(L+I)}\,, \tag{3.7}$$

$$\mathcal{P}(\boldsymbol{Y}_t = Y_t \mid \boldsymbol{Y}_{t-1} = Y_{t-1}) = \frac{\det(M_{Y_t \cup Y_{t-1}})}{\det(M + I_{\overline{Y}_{t-1}})}\,, \tag{3.8}$$

where M is the matrix $L(I-L)^{-1}$. These particular distributions not only imply that the individual $\boldsymbol{Y}_t$ variables are DPP-distributed, but also that the set union variable $\boldsymbol{Z}_t \equiv \boldsymbol{Y}_t \cup \boldsymbol{Y}_{t-1}$ is DPP-distributed as well. Specifically, if the individual variables have kernel L and marginal kernel K, then $\boldsymbol{Z}_t$ has kernel $2M$ and marginal kernel $2K$. It is also possible to analogously define an M-k-DPP over sets of size k. The resulting union variable distribution is a $2k$-DPP, but does not yield $\boldsymbol{Y}_t$ that are exactly k-DPPs. Nevertheless, the diversity at the $2k$ level implies some diversity at the k level. Affandi et al. (2012) derive exact and efficient sampling procedures for M-DPPs and M-k-DPPs. These algorithms essentially combine DPP conditioning formulas with the standard DPP sampling formulas. The resulting M-k-DPP sampler can select T sets in time $O(TN^3 + TNk^2)$.

Experimentally, on the news summarization task described above, the M-k-DPP does well. It sacrifices a small amount of article quality to produce much more diverse sets than models that choose solely based on quality. Additionally, compared to a model that samples from an independent k-DPP at each time step, the M-k-DPP shows a substantial increase in between-step diversity.

For basic DPPs, learning item qualities by maximizing likelihood is a concave optimization problem (Kulesza and Taskar, 2011b), which we will discuss in more detail in Chapter 6. While the basic DPP sampling procedures translate well to M-DPPs, unfortunately even this most simple of learning procedures does not; the M-DPP log-likelihood objective is not concave for the quality-learning setting. Nevertheless, Affandi et al. (2012) demonstrate a promising heuristic for learning item qualities, inspired by standard online algorithms. Their procedure assumes that each day a user provides feedback by marking the articles that they see as either of interest or not of interest. From these labels, the quality scores for articles on subsequent days are adjusted: the parameters associated with quality features for articles of interest are increased, while those associated with the other articles are decreased.

3.4 Continuous DPPs

The extension of DPPs to the continuous realm is considered in Affandi, Fox, and Taskar (2013a), where they propose practical sampling algorithms for the resulting DPPs. Formally, the discrete DPP sampling algorithm applies to the continuous case, but computationally it is intractable to have N uncountable. For a continuous space $\Omega \subseteq \mathbb{R}^d$, L becomes an operator, $L : \Omega \times \Omega \to \mathbb{R}$. The probability density of a point configuration $A \subset \Omega$ is then: $\mathcal{P}_L(A) \propto \det(L_A)$, where L_A is the $|A| \times |A|$ matrix with entry $L(\boldsymbol{x}, \boldsymbol{y})$ for each $\boldsymbol{x}, \boldsymbol{y} \in A$.

Despite the fact that N is uncountable, Affandi et al. (2013a) show that it is possible to sample from $\mathcal{P}_L$ by adapting the dual sampling algorithm (Kulesza, 2012, Algorithm 3). The initial step in this algorithm only requires that the number of features, D, be small, putting no constraint on N. Many continuous kernels are by nature low-rank, which means that they already have a reasonable D. Many other high- or infinite-rank kernels can be transformed into low-rank kernels by using a Nyström approximation or random Fourier features. This type of approximation will be discussed in greater detail in Section 4.6.2 for discrete DPPs. For continuous DPPs, Appendix C in the supplement of Affandi et al. (2013a) lists common kernel types for which approximation is feasible.

Given that L (or its approximation) is low-rank, we can write it as: $L(\boldsymbol{x}, \boldsymbol{y}) = B(\boldsymbol{x})^* B(\boldsymbol{y})$, where we define the operator $B(\boldsymbol{x}) : \Omega \to \mathbb{C}^D$ and $B(\boldsymbol{x})^*$ indicates the complex conjugate transpose. Thus, the $D \times D$ dual kernel matrix C needed for the dual sampling algorithm can be computed by evaluating the following integral:

$$C = \int_{\Omega} B(\boldsymbol{x}) B(\boldsymbol{x})^* d\boldsymbol{x} \,. \tag{3.9}$$

Given C, we can compute its eigendecomposition and begin executing the dual sampling algorithm as usual. The rest of the algorithm only relies on N in the following step:

$$\text{Select } i \text{ from } \mathcal{Y} \text{ with } Pr(i) = \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} (\hat{\boldsymbol{v}}^\top B_i)^2 \,.$$

For the continuous case, this step becomes:

$$\text{Select } \boldsymbol{x} \text{ from } f(\boldsymbol{x}) = \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} |\hat{\boldsymbol{v}}^* B(\boldsymbol{x})|^2 \,.$$

Sampling directly from $f(\boldsymbol{x})$ is usually quite difficult, but often it is possible to use the inverse CDF method to generate a sample. That is, we can draw a number u uniformly from $[0, 1]$, then compute which $\boldsymbol{x}$ corresponds to value u in f's cumulative density function. Appendix C in the supplement of Affandi et al. (2013a) shows how to do this for Gaussian and polynomial kernels.

For continuous k-DPPs, rather than adapting the existing sampling algorithm, Affandi et al. (2013a) instead develop a Gibbs sampler. That is, they define k variables $\{\boldsymbol{x}_\ell\}_{\ell=1}^k$, arbitrarily assign each $\boldsymbol{x}_\ell$ to some value in Ω, then repeatedly re-sample each $\boldsymbol{x}_\ell$ from the conditional density $p(\boldsymbol{x}_\ell \mid \{\boldsymbol{x}_j\}_{j\neq\ell})$. The conditional density to sample from can be derived using Schur's determinantal identity (recall Definition 2.3). Letting R denote $\{x_j\}_{j\neq\ell}$, the conditional is as follows:

$$p(\boldsymbol{x}_\ell \mid \{\boldsymbol{x}_j\}_{j\neq\ell}) \propto L(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i,j\neq\ell} [L_R^{-1}]_{ij} L(\boldsymbol{x}_i, \boldsymbol{x}_\ell) L(\boldsymbol{x}_j, \boldsymbol{x}_\ell)\,. \tag{3.10}$$

This continuous k-DPP sampling method, as well as the other continuous DPP sampling method based on the dual kernel, both have complexity linear in d, the dimension of the space Ω.

From a practical standpoint, these continuous DPP sampling methods might help improve the results for problems where DPPs are used as priors for latent variable models. For example, Zou and Adams (2013) put a DPP prior on a topic model to encourage topics to have distinct parameter vectors (less topic overlap). This could more cleanly be done by employing a continuous DPP. While Affandi et al. (2013a) do not explore that particular task, one interesting application that they do experiment with is density estimation: given samples from some distribution, the goal is to model the distribution as best as possible with a mixture of a small number of Gaussians. The mixture weights and the means and variances of the Gaussians must be estimated to create such a model. Typically, the means are assumed to be independent, but if we instead assume that they are drawn from a DPP then this encourages diversity. For a variety of densities associated with real-world datasets (e.g. measurements of galaxy velocities, measurements of lake acidity), Affandi et al. (2013a) demonstrate that this strategy yields Gaussian mixtures that have fewer components with large mixture weights. In other words, use of a DPP prior on the means yields a more compact model of the data. Moreover, likelihood on heldout data is not negatively affected by the DPP assumption.

4

Dimensionality Reduction

Many of the core DPP inference tasks have a computational complexity of roughly $O(N^3)$, where N is the number of items in the ground set $\mathcal{Y}$. As discussed in Section 2.4, where dual kernels were defined, if it is known that the DPP kernel L is low-rank and decomposes as a product of a $D \times N$ matrix B with its transpose, as in $L = B^\top B$, then the computational complexity can often be reduced, with D substituting for N. While D does not entirely replace N for all inference tasks, it does reduce the cost to such an extent that computational complexities are at worst linear in N, rather than cubic. See Table 2.1 for exact complexities. Settings where N is so large that even a linear dependence is unacceptable often fall under the structured DPP umbrella, described in Section 3.2. For problems within the structured class, inference is tractable as long as the treewidth of the associated factor graph is not too large. This is similar to the tractability condition for graphical models.

In this chapter we consider the setting where both N and D are simultaneously large. This is a challenging realm because both the $N \times N$ primal kernel $L = B^\top B$ and the $D \times D$ dual kernel $C = B^\top B$ are intractable objects. Exact structured DPP inference is not feasible here, so development of an approximation scheme is vital for this DPP variant. Fortunately, we have recourse to dimensionality reduction

techniques.

The remainder of this chapter proves a bound on the change in a DPP's distribution when D features are randomly projected down to $d \ll D$ dimensions. This bound is then applied to several example problems, including a variant of the document summarization task. The majority of the information that this chapter conveys can also be found in Gillenwater, Kulesza, and Taskar (2012a). We conclude the chapter by surveying several recent papers that present alternative means of addressing the large-N, large-D setting.

4.1 Random projections

A classic result of Johnson and Lindenstrauss (1984) shows that high-dimensional points can be randomly projected onto a logarithmic number of dimensions while approximately preserving the distances between them. This result establishes that we can project a length-D vector B_i down to a much shorter length-d vector $\tilde{B}_i$, while approximately retaining the values $B_i^\top B_j$. A more recent result by Magen and Zouzias (2008) extends this idea to the preservation of *volumes* spanned by sets of points. That is, instead of just pairwise distances, the volume of the vectors' parallelotope is approximately preserved. Here, we use the relationship between determinants and volumes, established in Sections 1.2 and 2.1, to adapt the latter result. The primary adaptation necessary is to bound the change in the DPP's normalization term. As this term is a sum over an exponential number of volumes, it is not immediate that a good bound on change in volume implies a good bound on change in DPP probabilities.

We first state a variant of Magen and Zouzias (2008)'s result, which bounds the ratio of volumes before and after projection from D down to d dimensions.

Lemma 4.1. *Let B be a $D \times N$ matrix. Fix $k < N$ and $0 < \epsilon, \delta < \frac{1}{2}$, and set the projection dimension d to:*

$$d = \max\left\{\frac{2k}{\epsilon}, \frac{24}{\epsilon^2}\left(\frac{\log(3/\delta)}{\log N} + 1\right)(\log N + 1) + k - 1\right\}. \tag{4.1}$$

Let G be a $d \times D$ random projection matrix whose entries are randomly sampled from a

normal distribution with mean zero and variance $\frac{1}{d}$*:*

$$G_{ij} \sim \mathcal{N}\left(0, \frac{1}{d}\right). \tag{4.2}$$

Let B_Y *for* $Y \subseteq \{1, \ldots, N\}$ *denote the* $D \times |Y|$ *matrix formed by taking the columns of* B *corresponding to the indices in* Y*. Then for all* Y *with cardinality at most* k*, with probability at least* $1 - \delta$ *we have:*

$$(1-\epsilon)^{|Y|} \leq \frac{\text{vol}(GB_Y)}{\text{vol}(B_Y)} \leq (1+\epsilon)^{|Y|}, \tag{4.3}$$

where $\text{vol}(B_Y)$ *is the* k*-dimensional volume of the parallelotope spanned by the columns of* B_Y*.*

Practically, Lemma 4.1 says that for any set of N points, randomly projecting down to:

$$d = O\left(\max\left\{\frac{k}{\epsilon}, \frac{\log(1/\delta) + \log(N)}{\epsilon^2} + k\right\}\right) \tag{4.4}$$

dimensions approximately preserves all volumes with high probability. We can leverage this result to quantify the effectiveness of random projections for DPPs. Recall the following relationship between determinants and volumes:

$$\text{vol}(B_Y) = \sqrt{\det(B_Y^\top B_Y)}. \tag{4.5}$$

This equivalence implies that Lemma 4.1 is in fact a bound on determinants as well as on volumes. For DPPs we are interested in bounding not only the change in individual determinants, but also the total change in a sum of many determinants. This is necessary in order to handle the DPP normalization constant. The following lemma provides a bound on the change in a k-DPP's normalization constant when random projections are applied.

Lemma 4.2. *Under the same conditions as Lemma 4.1, with probability at least* $1 - \delta$ *we have:*

$$(1+2\epsilon)^{-2k} \leq \frac{\sum_{Y:|Y|=k} \det((GB_Y)^\top (GB_Y))}{\sum_{Y:|Y|=k} \det(B_Y^\top B_Y)} \leq (1+\epsilon)^{2k}. \tag{4.6}$$

Proof.

$$\sum_{Y:|Y|=k} \det((GB_Y)^\top (GB_Y)) = \sum_{Y:|Y|=k} \left(\mathrm{vol}(GB_Y)\right)^2 \tag{4.7}$$

$$\geq \sum_{Y:|Y|=k} \left(\mathrm{vol}(B_Y)(1-\epsilon)^k\right)^2 \tag{4.8}$$

$$= (1-\epsilon)^{2k} \sum_{Y:|Y|=k} \left(\mathrm{vol}(B_Y)\right)^2 \tag{4.9}$$

$$\geq (1+2\epsilon)^{-2k} \sum_{Y:|Y|=k} \det(B_Y^\top B_Y)\,. \tag{4.10}$$

The first inequality holds with probability at least $1-\delta$ by Lemma 4.1. The third follows from the fact that $(1-\epsilon)(1+2\epsilon) \geq 1$ (since $\epsilon < 1/2$), and thus, raising this expression to the $2k$ power, $(1-\epsilon)^{2k} \geq (1+2\epsilon)^{-2k}$. The upper bound follows directly from Lemma 4.1:

$$\sum_{Y:|Y|=k} \det((GB_Y)^\top (GB_Y)) \leq \sum_{Y:|Y|=k} \left(\mathrm{vol}(B_Y)(1-\epsilon)^k\right)^2 \tag{4.11}$$

$$= (1-\epsilon)^{2k} \sum_{Y:|Y|=k} \det(B_Y^\top B_Y)\,. \tag{4.12}$$

□

This bound on the DPP normalization constant can be exploited to yield a bound on a measure of distributional similarity. Formally, let $\mathcal{P}$ be the DPP's probability measure before the features B that define the DPP kernel $L = B^\top B$ are projected. Let $\tilde{\mathcal{P}}$ be the distribution after projection. Ideally, we want a bound on the total variational distance between $\mathcal{P}$ and $\tilde{\mathcal{P}}$. The formula for L_1 variational distance is:

$$\|\mathcal{P} - \tilde{\mathcal{P}}\|_1 = \sum_{Y:Y\subseteq\mathcal{Y}} |\mathcal{P}(Y) - \tilde{\mathcal{P}}(Y)|\,. \tag{4.13}$$

Theorem 4.3 bounds the k-constrained version of this quantity.

Theorem 4.3. *Let $\mathcal{P}^k$ be the k-DPP distribution associated with kernel $L = B^\top B$ for $B \in \mathbb{R}^{D\times N}$. Let d be as in Equation (4.1) and let G be a $d \times D$ random projection matrix, as defined in Lemma 4.1. Finally, let $\tilde{\mathcal{P}}^k$ be the k-DPP distribution associated with kernel $\tilde{L} = (GB)^\top (GB)$. Then for $0 < \epsilon, \delta < \frac{1}{2}$, with probability at least $1-\delta$ we have:*

$$\|\mathcal{P}^k - \tilde{\mathcal{P}}^k\|_1 \leq e^{6k\epsilon} - 1\,. \tag{4.14}$$

Note that $e^{6k\epsilon} - 1 \approx 6k\epsilon$ *when* $k\epsilon$ *is small.*

Proof. Starting from the definition of L_1 variational distance, we have:

$$\|\mathcal{P}^k - \tilde{\mathcal{P}}^k\|_1 = \sum_{Y:|Y|=k} |\mathcal{P}^k(Y) - \tilde{\mathcal{P}}^k(Y)| \tag{4.15}$$

$$= \sum_{|Y|=k} \mathcal{P}^k(Y) \left| 1 - \frac{\tilde{\mathcal{P}}^k(Y)}{\mathcal{P}^k(Y)} \right| \tag{4.16}$$

$$= \sum_{|Y|=k} \mathcal{P}^k(Y) \left| 1 - \frac{\det([GB_Y^\top][GB_Y])}{\det(B_Y^\top B_Y)} \frac{\sum_{Y:|Y'|=k} \det(B_{Y'}^\top B_{Y'})}{\sum_{Y:|Y'|=k} \det([GB_{Y'}^\top][GB_{Y'}])} \right|$$

$$\leq \left| 1 - (1+\epsilon)^{2k}(1+2\epsilon)^{2k} \right| \sum_{Y:|Y|=k} \mathcal{P}^k(Y) \tag{4.17}$$

$$= \left| 1 - (1+\epsilon)^{2k}(1+2\epsilon)^{2k} \right| \tag{4.18}$$

$$\leq \left| 1 - e^{2k\epsilon} e^{4k\epsilon} \right| \tag{4.19}$$

$$= e^{6k\epsilon} - 1\,. \tag{4.20}$$

The first inequality follows from Lemmas 4.1 and 4.2, which hold simultaneously with probability at least $1 - \delta$. The second inequality follows from $(1+a)^b \leq e^{ab}$ for $a, b \geq 0$. □

Given these bounds on random projections for DPPs, we can now apply them to handle the large-N, large-D setting. Specifically, combining dual DPP algorithms with the smaller number of features $d \ll D$ makes approximate inference possible for structured DPPs.

4.2 Threading k-SDPPs

To empirically verify the efficacy of random projections, we test this dimensionality reduction technique on several structured k-DPP applications. We will refer to structured k-DPPs as k-SDPPs. The inference techniques for SDPPs and k-DPPs, developed in Kulesza and Taskar (2010) and Kulesza and Taskar (2011a), respectively, are summarized in Sections 3.2 and 3.1. They combine seamlessly, such that the computational complexity of the SDPP sampling procedure is unchanged. Recall that, given the eigendecomposition of the dual kernel C, sampling an SDPP is an $O(D^2k^3 + DM^cRk^2)$ operation. The k here is the number of structures in the

sampled set, R is the number of parts in a structure, M is the number of values a part y_r can take on, and c is the maximum degree of any factor node. Also recall that the kernel C itself takes $O(D^2M^cR)$ time to compute, and $O(D^\omega)$ time to eigendecompose. With random projections, we can change the D factors in all of these expressions to $d \ll D$. Then, the only place that D will occur in our algorithms is in the random projection step itself. This projection takes $O(M^cRDd)$ time, as it corresponds to the multiplication of a $d \times D$ projection matrix G by a $D \times M^cR$ matrix that contains all of the feature values for the structured DPP factors.

In what follows, we will use the notation from Section 2.5, which decomposes a structure $\boldsymbol{y}$'s feature vector $B(\boldsymbol{y})$ into a quality score $q(\boldsymbol{y})$ and similarity features $\phi(\boldsymbol{y})$. With this notation, the probability of a set Y under a k-SDPP is:

$$\mathcal{P}_L^K(Y) = \frac{\left(\prod_{\boldsymbol{y}:\boldsymbol{y}\in Y} q(\boldsymbol{y})\right) \det\left(\phi(Y)^\top \phi(Y)\right)}{\sum_{\substack{Y':Y'\subseteq\mathcal{Y},\\ |Y'|=k}} \left(\prod_{\boldsymbol{y}:\boldsymbol{y}\in Y'} q(\boldsymbol{y})\right) \det\left(\phi(Y')^\top \phi(Y')\right)}, \tag{4.21}$$

where $\phi(Y)$ denotes the $D \times |Y|$ matrix consisting of columns $\phi(\boldsymbol{y})$ for $\boldsymbol{y} \in Y$. We assume that q and ϕ factor over the parts of a structure, as in Equation (3.6).

The structure on which our experiments operate is a *thread*—a singly-linked chain. In all cases, the threading application takes the form of finding diverse, high-quality paths (threads) in a directed graph. More concretely, suppose that we have an M-node graph. Let the ground set $\mathcal{Y}$ of our DPP consist of all length-R paths in the graph. Each $\boldsymbol{y} \in \mathcal{Y}$ is then a sequence $[y_1, \ldots, y_R]$ where each y_r is a graph node and nodes y_r, y_{r+1} are connected by an edge in the graph. For a complete graph, there are $N = M^R$ such possible threads. While we only consider the setting where the length R is identical for all threads, note that it is also possible to allow threads of variable length. This effect can be achieved by adding a single "dummy" node to the graph, with incoming edges from all other nodes and a single outgoing self-loop edge. Shorter threads will simply transition to this dummy node when they are complete.

Let the quality of a path in the M-node graph decompose based on the quality

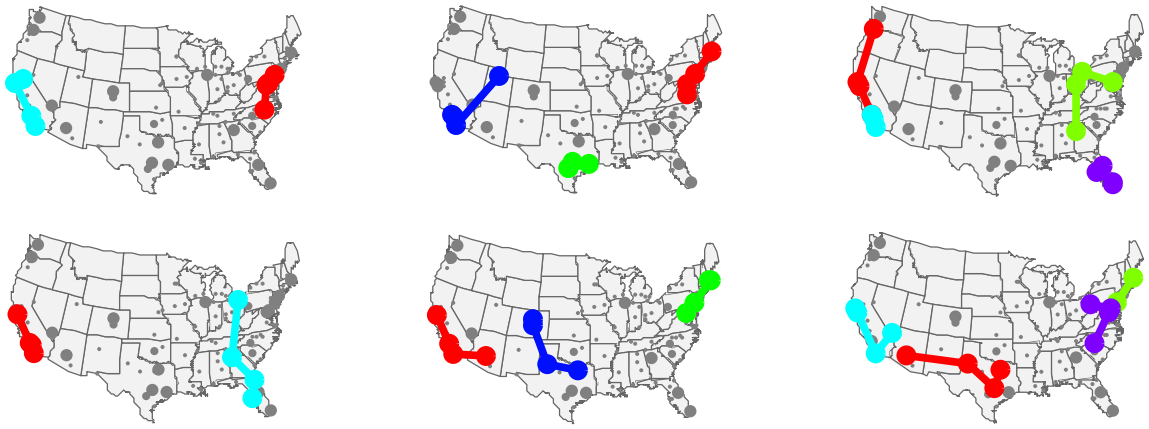


Figure 4.1: Each column shows two samples drawn from a k-SDPP; from left to right, $k = 2, 3, 4$. Circle size corresponds to city quality.

of individual nodes and edges, so that for a path $\boldsymbol{y}$ of length R we have:

$$q(\boldsymbol{y}) = \left(\prod_{r=1}^{R} q(y_r) \prod_{r=2}^{R} q(y_{r-1}, y_r) \right)^{\beta}, \tag{4.22}$$

where β is a hyperparameter controlling the dynamic range of the quality scores. Similarly, let the similarity features for a path in the graph decompose along the same lines:

$$\phi(\boldsymbol{y}) = \sum_{r=1}^{R} \phi(y_r) + \sum_{r=2}^{R} \phi(y_{r-1}, y_r)\,. \tag{4.23}$$

With these definitions for quality and similarity, the nodes in the corresponding factor graph have degrees in $\{1, 2\}$. This makes $c = 2$, resulting in a computational complexity of $O(D^2M^2R)$ for computing C, or $O(d^2M^2R)$ after applying random projections.

4.3 Toy example: geographical paths

We begin by demonstrating the performance of random projections on a small, synthetic threading task where the exact model is tractable, with $M = 200$ nodes and $D = 200$ similarity features. The goal in this case is to identify diverse, high-quality sets of travel routes between the 200 most populous U.S. cities. Each of these routes is a "thread", and the associated graph has one node per city. Every city is connected

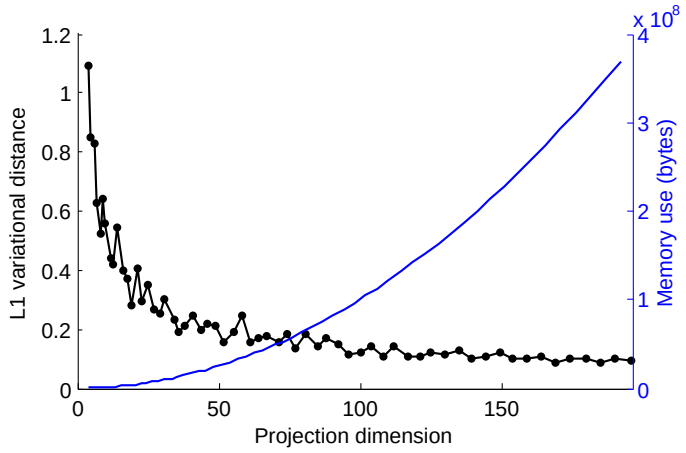


Figure 4.2: The effect of random projections. In black, on the left, we estimate the L_1 variational distance between the true and projected models. In blue, on the right, we plot the memory required for sampling. Runtime is proportional to memory use.

to every other city by an edge. In order to disallow paths that travel back and forth between the same cities though, we augment the nodes to include arrival direction. We then assign a quality score of zero to paths that return in the direction from which they came. This does not technically preclude paths with cycles of length > 2, but we find that it is in practice sufficient to prevent cycling.

For establishing path quality, we set the quality of a city $q(y_r)$ to the Google hit count of the city name, so that paths through popular cities are preferred. Pairwise factors $q(y_{r-1}, y_r)$ are defined to control path cycling, as described above.

We consider two routes to be diverse if they are well-separated geographically. To achieve this effect, we set the similarity features of a city $\phi(y_r)$ as the vector of inverse distances between y_r and each of the $M = 200$ cities. This makes $D = 200$, too. If cities y_i and y_j are distance a apart, then $\frac{1}{a}$ will be large for small distances, making the similarity score $\phi(y_i)^\top \phi(y_j)$ large. This clearly encourages paths to travel through diverse regions of the country. We do not define pairwise factors $\phi(y_{r-1}, y_r)$ for this application, but this does not change the overall computational complexity analysis from the previous section.

Given these definitions, the threading k-SDPP is fully specified and we can sample from it. Figure 4.1 shows sets sampled with path length $R = 4$ and various values of k. For $k = 2$, the model tends to choose one path along the east coast and one

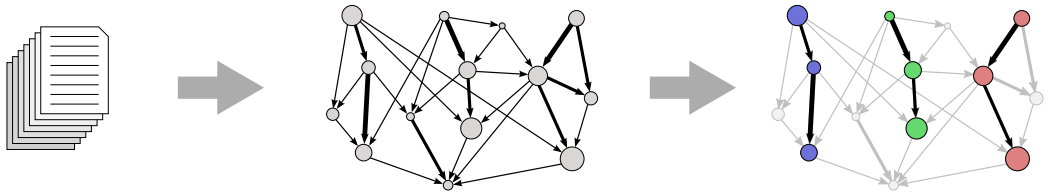


Figure 4.3: An illustration of document collection threading. We first build a graph from the collection, using measures of importance and relatedness to weight nodes (documents) and build edges (relationships). Then, from this graph, we extract a diverse, salient set of threads to represent the collection.

along the west. As k increases, a wider variety of configurations emerge. They continue, though, to emphasize popular cities and to remain relatively geographically diverse.

For this small, structured model we can easily investigate the effects of random projections. Figure 4.2 shows the L_1 variational distance between the original model and the projected model (estimated by sampling), as well as the actual memory required for a variety of projection dimensions d. As predicted by Theorem 4.3, fidelity to the true model increases rapidly with d, such that we only require a small number of dimensions to approximate the model well.

4.4 Threading document collections

We now describe the application of the threading structure plus random projections method to a more practical class of problems: summarizing large document collections. The increasing availability of large document collections has the potential to revolutionize our ability to understand the world. However, the scale and complexity of such collections frequently make it difficult to quickly grasp the important details and the relationships between them. As a result, automatic interfaces for data navigation, exploration, aggregation, and analysis are becoming increasingly valuable.

Consider a large graph, with documents as nodes, and edges indicating relationships between documents whose semantics depend on the exact task. The sets of threads we can obtain from a k-SDPP model over this graph could serve as a corpus summary. Figure 4.3 illustrates the document collection threading process.

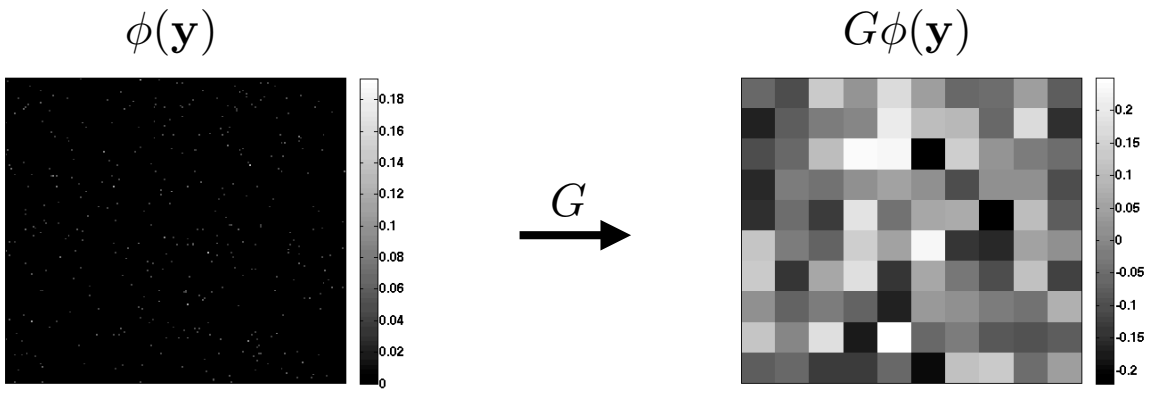


Figure 4.4: **Left**: The thousands of word features associated with a single document. (This is shown as a $\sqrt{D} \times \sqrt{D}$ matrix rather than a $D \times 1$ vector to fit this illustration on a single page.) Notice that most features have value zero, as any given article contains only a small fraction of the total number of vocabulary words. **Right**: The $d = 50$ features resulting from the application of a random projection matrix G.

If threads are high-quality, then they will represent salient aspects of the corpus. If a set of threads is diverse, then each individual thread will tend to be compact and hence coherent. For example, given a collection of academic papers, a thread set could identify the most significant lines of research in a citation graph by selecting chains of important papers. Or, given news articles connected chronologically, article threads could form timelines describing the major events from the most significant news stories. Top-tier news organizations like The New York Times and The Guardian regularly publish such timelines, but have so far been limited to creating them by hand. We explore these two applications, academic citation threads and news article timelines, in the following two subsections.

In these experiments we typically have at least $M = 30{,}000$ documents. If we make use of the most natural feature set, the vocabulary, then there are also tens of thousands of features, D. For $M, D > 30{,}000$, storing a single message for the message-passing routine involved in SDPP sampling would require over 200 terabytes of memory. Hence, this is a setting where random projections are essential for efficiency. We will project D down to $d = 50$ in our experiments. To give a better sense of the scale of this change, Figure 4.4 shows a typical word-based feature vector for a single document and the corresponding random projection for $d = 50$.

4.4.1 Related work

The Topic Detection and Tracking (TDT) program (Wayne, 2000) has led to some research in this direction. Several of TDT's core tasks, like link detection, topic detection, and topic tracking, can be seen as subroutines for the threading problem. Graph threading with k-SDPPs, however, addresses these tasks *jointly*, using a global probabilistic model with a tractable inference algorithm.

Other work in the topic tracking literature has addressed related tasks (Mei and Zhai, 2005; Blei and Lafferty, 2006; Leskovec, Backstrom, and Kleinberg, 2009; Ahmed and Xing, 2010). In particular, Blei and Lafferty (2006) introduced dynamic topic models (DTMs), which, assuming a division of documents into time slices, attempt to fit a generative model whose topics evolve over time. In each slice a DTM draws a set of topics from a Gaussian distribution whose mean is determined by the topics from the previous slice. In this sense, a DTM can be viewed as a mechanism for generating *topic* threads. These are related to document threads, but do not consist of actual items from the dataset. In our experiments we engineer a baseline for constructing document threads from DTM topic threads, but the topic-centric nature of DTMs means that they are not ideal for this task. The work of Ahmed and Xing (2010) generalizes DTMs to iDTMs (infinite DTMs) by allowing topics to span only a subset of time slices, and allowing an arbitrary number of topics. However, iDTMs still require placing documents into discrete epochs, and the issue of generating topic rather than document threads remains.

In the information retrieval community there has also been work on extracting temporal information from document collections. Swan and Jensen (2000) proposed a system for finding temporally clustered named entities in news text and presenting them on a timeline. Allan, Gupta, and Khandelwal (2001) introduced the task of *temporal summarization*, which takes a stream of news articles on a particular topic and tries to extract sentences describing important events as they occur. Yan, Wan, Otterbacher, Kong, Li, and Zhang (2011) evaluated methods for choosing sentences from temporally clustered documents that are relevant to a query. In contrast, graph threading seeks not to extract grouped entities or sentences, but instead to organize a subset of the objects (documents) themselves into threads, with topic identification as a side effect.

There has also been some prior work focused more directly on threading. Shahaf and Guestrin (2010) and Chieu and Lee (2004) proposed methods for selecting a *single* thread, while Shahaf, Guestrin, and Horvitz (2012) proposed *metro maps* as alternative structured representations of related news stories. Metro maps are effectively sets of non-chronological threads that are encouraged to intersect and thus create a "map" of events and topics. However, these approaches assume some prior knowledge about content. Shahaf and Guestrin (2010), for example, assume the thread endpoints are specified, and Chieu and Lee (2004) require a set of query words. These inputs make it possible to quickly pare down the document graph. In contrast, we apply graph threading to very large graphs and consider all possible threads.

4.4.2 Setup

We already specified the form of $q(\boldsymbol{y})$ and $\phi(\boldsymbol{y})$ in Equations (4.23) and (4.22), respectively, but it remains to establish definitions for their sub-functions: $q(y_r)$, $q(y_{r-1}, y_r)$, $\phi(y_r)$, and $\phi(y_{r-1}, y_r)$.

Pairwise node qualities, $q(y_{r-1}, y_r)$: The pairwise quality scores reflect the degree of textual similarity between the two documents that they couple. We exploit tf-idf vectors to define these scores. The tf-idf vectors were generated in the following manner: first, the text for all documents was tokenized; second, stop words and punctuation were discarded; third, for each remaining word w the *inverse document frequency* $\text{idf}(w)$ was computed. This $\text{idf}(w)$ is the negative logarithm of the fraction of documents that contain the word w. The *term frequency* $\text{tf}_y(w)$ is the number of times the word w appears in document y. Given these idfs and tfs, we have the following definition for entry w of the tf-idf vector for document y:

$$[\text{tf-idf}(y)]_w \propto \text{tf}_y(w)\text{idf}(w)\,. \tag{4.24}$$

By computing normalized cosine similarity (NCS) between the tf-idf vector of document y_{r-1} and document y_r, we arrive at the final formula for our pairwise quality

scores. Letting W indicate the set of all vocabulary words, the NCS score is:

$$\text{NCS}(y_{r-1}, y_r) = \frac{\sum_{w \in W} \text{tf}_{y_{r-1}}(w)\text{tf}_{y_r}(w)\text{idf}^2(w)}{\sqrt{\sum_{w \in W} \text{tf}^2_{y_{r-1}}(w)\text{idf}^2(w)}\sqrt{\sum_{w \in W} \text{tf}^2_{y_r}(w)\text{idf}^2(w)}}. \tag{4.25}$$

Individual node qualities, $q(y)$: For the quality score of each individual graph node, we use LexRank scores (Erkan and Radev, 2004), which are similar to node degrees. The LexRank score is the stationary distribution of a thresholded, binarized, row-normalized matrix of cosine similarities, plus a damping term, which we fix at 0.15. LexRank is a measure of salience; papers closely related to many other papers receive a higher LexRank score.

Individual node similarity features, $\phi(y)$: To get similarity features of each individual document, we again exploit its tf-idf vector. We could use this tf-idf vector directly as the similarity feature vector, but in preliminary experiments we found that a derivative measure of similarity was more effective. Specifically, we represent each document by the 1000 documents to which it is most similar according to NCS; this results in binary ϕ of dimension $D = M$ with exactly 1000 non-zeros. We scale $\phi(y)$ such that $\|\phi(y)\| = 1$. The dot product between the similarity features of two documents is thus proportional to the fraction of top-1000 documents they have in common. As described earlier, the final step is to randomly project this large feature set from $D \approx 30,000$ down to $d = 50$ dimensions.

Pairwise node similarity features, $\phi(y_{r-1}, y_r)$: We do not bother to define similarity features on pairs of nodes, but this is a simple extension that could be used to improve thread coherence. Note that adding these pairwise similarity features would not change the asymptotic runtime, as we already have pairwise quality features.

With these quality and similarity features in place, we are now ready to describe experiments on two document summarization tasks: generating academic citation threads and news article timelines.

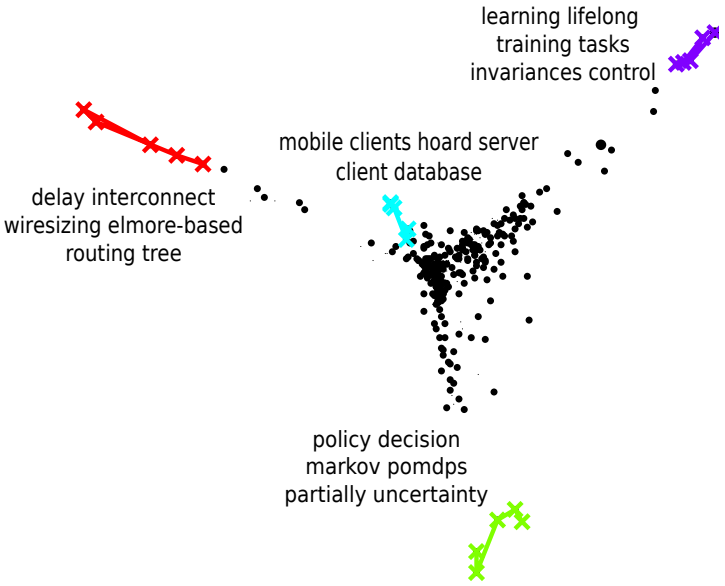
4.4.3 Academic citation data

The Cora dataset is a large collection of academic papers on computer science topics, including citation information (McCallum, Nigam, Rennie, and Seymore, 2000). We construct a directed graph with papers as nodes and citations as edges. After removing papers with missing metadata or zero outgoing citations, our graph contains $M = 28{,}155$ papers. The average out-degree is 3.26 citations per paper; 0.011% of the total possible edges are present. We apply the word filtering described in Section 4.4.2, additionally removing words that are too common, appearing in more than 10% of the papers, or too rare, appearing in only one paper. After this, there are 50,912 unique words remaining. We did not perform a quantitative evaluation for this dataset, but Figure 4.5 illustrates the behavior of the associated k-SDPP model when we project down to $d = 50$ dimensions. Samples from the model, like the one presented in the figure, offer not only some immediate intuition about the types of papers contained in the collection, but, upon examining individual threads, provide a succinct illustration of the content and development of sub-areas of computer science research.

4.4.4 News articles

For quantitative evaluation, we use newswire data. This dataset comprises over 200,000 articles from the New York Times, collected from 2005-2007 as part of the English Gigaword corpus (Graff and C., 2009). We split the articles into six-month time periods. Because the corpus contains a significant amount of noise in the form of articles that are short snippets, lists of numbers, and so on, we filer the results by discarding articles that are more than two standard deviations longer than the mean article, articles that contain less than 400 words, and articles whose fraction of non-alphabetic words is more than two standard deviations above the mean. On average, for each six-month period, we are left with $M = 34{,}504$ articles. We apply the word filtering described in the previous section, additionally removing words that are too common, appearing in more than 15% of the articles, or too rare, appearing in less than 20 articles. After filtering, there are a total of 36,356 unique words.

Next, we generate a graph for each time period with articles as nodes. As we do not have citations to define the graph edges, we instead use NCS, Equation (4.25), to



Thread: **learning lifelong training tasks invariances control**

1. Locally Weighted Learning for Control
2. Discovering Structure in Multiple Learning Tasks: The TC Algorithm
3. Learning One More Thing
4. Explanation Based Learning for Mobile Robot Perception
5. Learning Analytically and Inductively

Thread: **mobile clients hoard server client database**

1. A Database Architecture for Handling Mobile Clients
2. An Architecture for Mobile Databases
3. Database Server Organization for Handling Mobile Clients
4. Mobile Wireless Computing: Solutions and Challenges in Data Management
5. Energy Efficient Query Optimization

Figure 4.5: Example threads sampled from a 4-SDPP with thread length $R = 5$ on the Cora dataset. **Above**: We plot a subset of the Cora papers, projecting their tf-idf vectors to two dimensions by running PCA on the centroids of the threads. The documents of a thread are shown connected by colored edges. Displayed beside each thread are a few of the its highest tf-idf words. **Below**: Paper titles from two of the threads.

generate edge weights, and throw away edges with weight < 0.1. We also require that edges go forward in time; this enforces the chronological ordering of threads. The resulting graphs have an average of 0.32% of the total possible edges, and an average degree of 107. We set q and ϕ as described in Section 4.4.2, with one modification: we add a constant feature ρ to ϕ, which controls the overall degree of repulsion; large values of ρ make all documents more similar. We set ρ and the quality model hyperparameter β from Equation (4.22) to maximize a cosine similarity evaluation metric described below, using the data from the first six months of 2005 as a development set. Finally, we use random projections to reduce ϕ to $d = 50$ dimensions. For all of the following experiments, we use $k = 10$ and $R = 8$. All evaluation metrics we report are averaged over 100 random samples from the k-SDPP model for each six-month period.

Graph visualizations

At http://zoom.it/jOKV, the news graph for the first half of 2005 can be viewed interactively; placing the graph in this thesis would be impractical since the computational demands of rendering it, and the zooming depth required to explore it, would exceed the capabilities of modern document viewers. In this graph each node (dark circle) represents a news article, and is annotated with its headline. Node size corresponds to quality (LexRank score). Nodes are laid out chronologically, left-to-right, from January to June of 2005. The five colored paths indicate a set of threads sampled from the k-SDPP. Headlines of the articles in each thread are colored to match the thread. Due to the scale of the dataset, it is difficult to display all edges, so only 1% of the edges are shown. Edge thickness corresponds to document pair quality (NCS).

In Figure 4.6 we provide a view of a small subgraph of the full news graph for illustration purposes. It shows just the incoming and outgoing edges for a single node. In the digital version of this thesis, Figure 4.6 can be zoomed in order to read the headlines. As an alternative though, a zoomable version of this subgraph is also available at http://zoom.it/GUCR.

Baselines

We compare the k-SDPP model to two other thread selection methods.

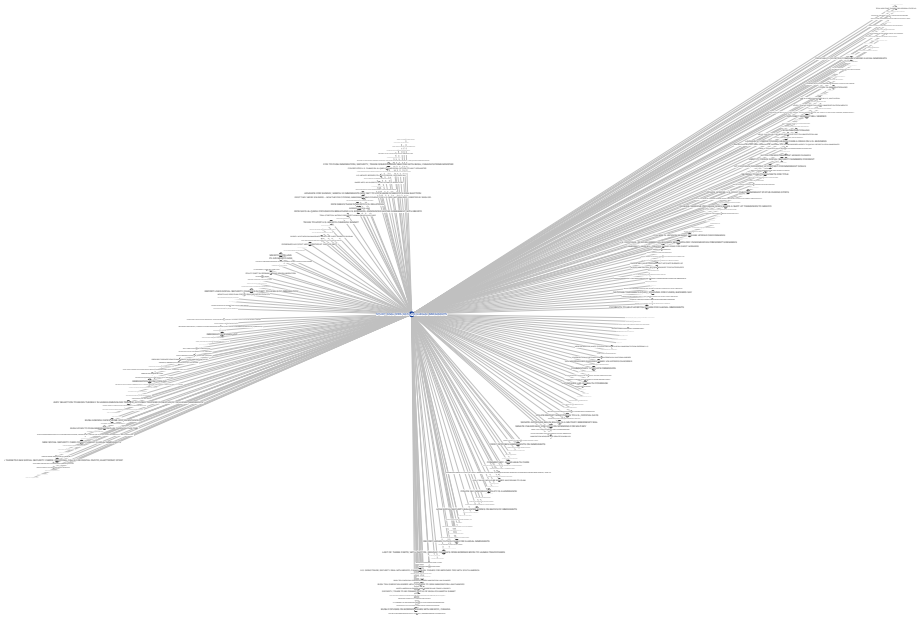
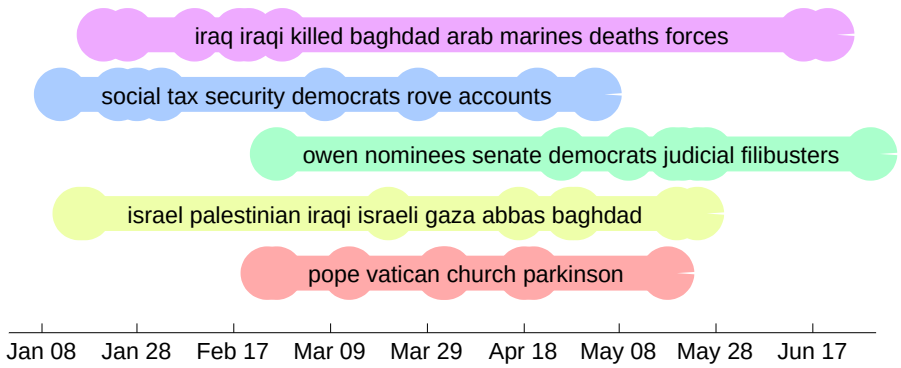


Figure 4.6: Snapshot of a single article node and all of its neighboring article nodes. See http://zoom.it/GUCR for the zoomable image.

k-means baseline: A simple baseline is to split each six-month period of articles into R equal time slices, then apply k-means clustering to each slice, using NCS to measure distance. We can then select the most central article from each cluster to form the basis of a set of threads. Suppose we match the k articles from time slice r one-to-one with those from slice $r-1$ by computing the pairing that maximizes the average NCS of the pairs, i.e., the coherence of the threads. Repeating this process $R-1$ times, the result is a set of k threads of length R, where no two threads contain the same article. Note though that because clustering is performed independently in each time slice, it is likely that the threads will sometimes exhibit discontinuities; the articles chosen in successive times slices may not always naturally align.

DTM baseline: A more sophisticated baseline is the dynamic topic model (Blei and Lafferty, 2006), which explicitly attempts to find topics that are smooth through time. We run the publicly available code from Blei and Lafferty (2006) to fit DTMs, with the number of topics set to k and with the data split into R equal time slices.

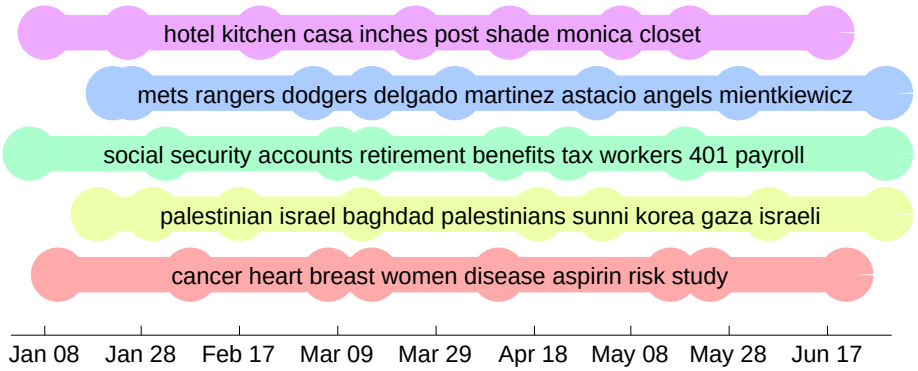


Feb 24: Parkinson's Disease Increases Risks to Pope
Feb 26: Pope's Health Raises Questions About His Ability to Lead
Mar 13: Pope Returns Home After 18 Days at Hospital
Apr 01: Pope's Condition Worsens as World Prepares for End of Papacy
Apr 02: Pope, Though Gravely Ill, Utters Thanks for Prayers
Apr 18: Europeans Fast Falling Away from Church
Apr 20: In Developing World, Choice [of Pope] Met with Skepticism
May 18: Pope Sends Message with Choice of Name

Figure 4.7: A set of five news threads generated by sampling from the k-SDPP for the first half of 2005. Above, the threads are shown on a timeline with the most salient words superimposed; below, the dates and headlines from the lowest thread are listed. The headlines indicate that this thread constitutes a coherent news story.

We then choose, for each topic at each time step, the document with the highest per-word probability of being generated by that topic. Documents from the same topic form a single thread.

Figure 4.7 shows threads sampled from the k-SDPP, and Figure 4.8 shows threads from the DTM baseline for the same time period. Qualitatively, the k-SDPP produces more consistent threads. The DTM threads, though topic-focused, are less coherent as a story. Furthermore, DTM threads span the entire time period, while our method has the flexibility to select threads covering only relevant spans. The quantitative results to which the remainder of this section is devoted underscore the empirical value of these characteristics.



Jan 11:	Study Backs Meat, Colon Tumor Link
Feb 07:	Patients—and Many Doctors—Still Don't Know How Often Women Get Heart Disease
Mar 07:	Aspirin Therapy Benefits Women, but Not in the Way It Aids Men
Mar 16:	Study Shows Radiation Therapy Doesn't Increase Heart Disease Risk for Breast Cancer Patients
Apr 11:	Personal Health: Women Struggle for Parity of the Heart
May 16:	Black Women More Likely to Die from Breast Cancer
May 24:	Studies Bolster Diet, Exercise for Breast Cancer Patients
Jun 21:	Another Reason Fish is Good for You

Figure 4.8: A set of five news threads generated by a dynamic topic model for the first half of 2005. Above, the threads are shown on a timeline with the most salient words superimposed; below, the dates and headlines from the lowest thread are listed. Topic models are not designed for document threading and often link together topically similar documents that do not constitute a coherent news narrative.

Comparison to human summaries

As an initial quantitative evaluation of our method, we compare the threads generated by the baselines and sampled from the k-SDPP to a set of human-generated news summaries. The human summaries are not threaded; they are flat, roughly daily news summaries found in the Agence France-Presse portion of the Gigaword corpus, distinguished by their "multi" type tag. The summaries generally focus on world news, which does not cover the entire spectrum of articles in our dataset. Nevertheless, they allow us to provide an extrinsic evaluation for this novel task without generating gold standard thread sets manually, which would be a difficult task for a corpus of this size. We compute four metrics:

- **Cosine similarity**: We concatenate the human summaries from each six-month period to obtain a target tf-idf vector. We also concatenate the set of threads to be evaluated to obtain a predicted tf-idf vector. For these vectors we compute the NCS (in percent) between the target and predicted vectors. The hyperparameters for all methods—such as the constant feature magnitude ρ for k-SDPPs and the parameter governing topic proportions for DTMs—were tuned to optimize cosine similarity on a development set consisting of the articles from the first six months of 2005.

- **ROUGE-1**, **2**, and **SU4**: ROUGE is an automatic evaluation metric for text summarization based on n-gram overlap statistics (Lin, 2004). We report three standard variants: unigram match, bigram match, and skip bigram with maximum gap length of 4.

Table 4.1 shows the results of these comparisons, averaged over all six half-year intervals. According to every metric, the k-SDPP threads more closely resemble human summaries.

Mechanical Turk evaluation

As a secondary quantitative evaluation, we employ Mechanical Turk, an online marketplace for efficiently completing tasks that require human judgment. We asked Turkers to read the headlines and first few sentences of each article in a timeline and

	Cosine Sim	ROUGE-1		ROUGE-2		ROUGE-SU4	
		F	Prec/ Rec	F	Prec / Rec	F	Prec/ Rec
k-means	29.9	16.5	17.3/15.8	0.695	0.725/0.669	3.76	3.94/3.60
DTM	27.0	14.7	15.5/14.0	0.750	0.813/0.698	3.44	3.63/3.28
k-SDPP	**33.2**	**17.2**	**17.7/16.7**	**0.892**	**0.917/0.870**	**3.98**	**4.11/3.87**

Table 4.1: Similarity of automatically generated timelines to human summaries. Bold entries are significantly higher than others in the column at 99% confidence, computed using bootstrapping (Hesterberg et al., 2003).

	Rating	Interlopers
k-means	2.73	0.71
DTM	3.19	1.10
k-SDPP	**3.31**	1.15

Table 4.2: Rating: average coherence score from 1 (worst) to 5 (best). Interlopers: average number of interloper articles identified (out of 2). Bold entries are significantly higher with 95% confidence.

then rate the overall narrative coherence of the timeline on a scale of 1 ("the articles are totally unrelated") to 5 ("the articles tell a single clear story"). Five separate Turkers rated each timeline. The average ratings are shown in Table 4.2. Note that k-means does particularly poorly in terms of coherence since it has no way to ensure that clusters are similar between time slices.

We also had Turkers evaluate threads implicitly by performing a second task. We inserted two "interloper" articles selected at random into timelines, and asked them to remove the two articles that they thought should be eliminated to "improve the flow of the timeline". A screenshot of the task is shown in Figure 4.9. Intuitively, the interlopers should be selected more often when the original timeline is coherent. The average number of interloper articles correctly identified is shown in Table 4.2.

Runtimes

Finally, we report in Table 4.3 the time required to produce a complete set of threads for each method. We assume tf-idf and feature values have been computed in advance (this process requires approximately 160 seconds), as these are common to all three threading methods. The time reported in the table is thus the time required

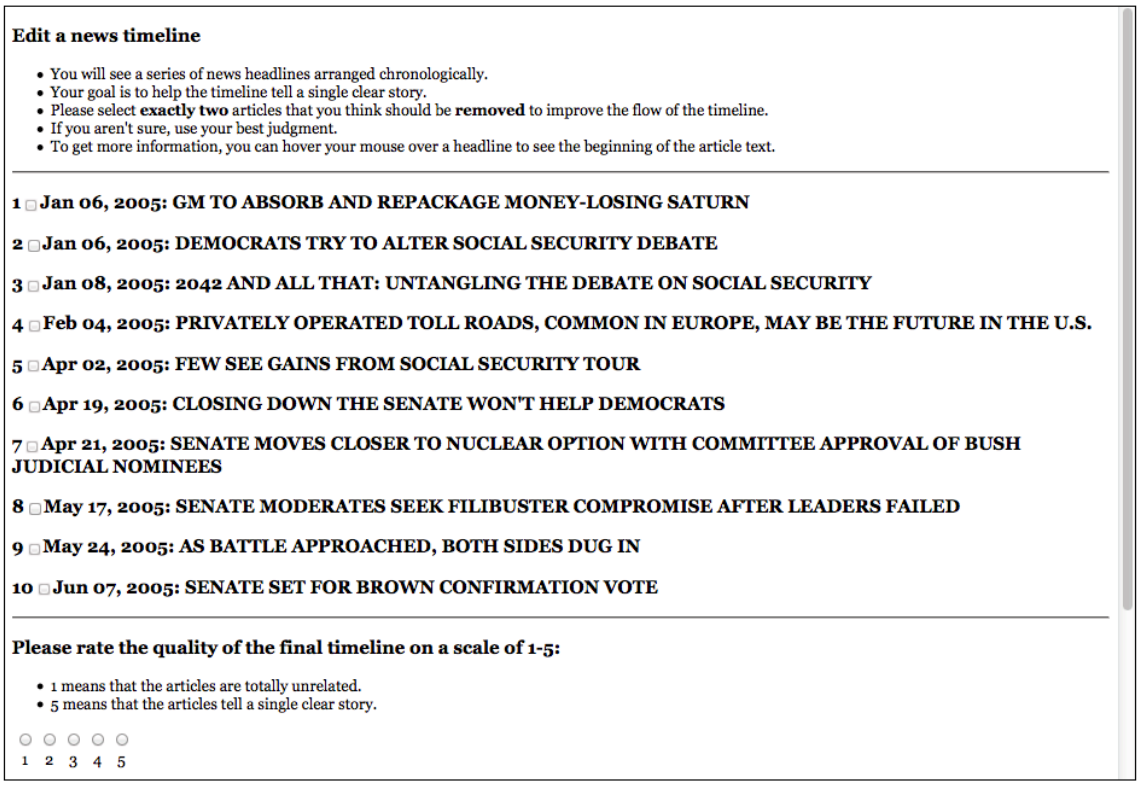


Figure 4.9: A screenshot of the Mechanical Turk task presented to annotators.

	Runtime
k-means	625.63
DTM	19,433.80
k-SDPP	**252.38**

Table 4.3: Time (in seconds) required to produce a complete set of news threads. The test machine has eight Intel Xeon E5450 cores and 32GB of memory.

for: clustering in k-means; model fitting for the DTM baseline; and random projections, computation of the dual kernel, and sampling for the k-SDPP. As can be seen from the runtimes in table, the k-SDPP method is about 2.5 times faster than k-means, and more than 75 times faster than the DTM.

4.5 Related random projections work

The $d \times D$ random projection matrix G of Theorem 4.3 has Gaussian entries $G_{ij} \sim \mathcal{N}\left(0, \frac{1}{d}\right)$. Many alternatives to this form of G have been studied. For example, Achlioptas (2002) showed that distances between points are approximately preserved when G is defined as follows:

$$G_{ij} = \sqrt{\frac{3}{d}} \begin{cases} +1 & \text{with probability } \frac{1}{6}\,, \\ 0 & \text{with probability } \frac{2}{3}\,, \\ -1 & \text{with probability } \frac{1}{6}\,. \end{cases} \tag{4.26}$$

This definition of G is relatively sparse, which means that its product with B (the $D \times N$ matrix that we are interested in projecting) can be computed more quickly than when the entries of G are drawn from a Gaussian. More precisely, since only one-third of this G is non-zero (in expectation), the product GB can be computed roughly 3 times faster.

More recent work by Ailon and Chazelle (2009) obtains a super-constant speedup by defining $G = RST$ for a $d \times D$ sparse matrix R, a $D \times D$ Walsh-Hadamard matrix S, and a $D \times D$ diagonal matrix T. The entries of these matrices are as follows:

$$R_{ij} = \begin{cases} \mathcal{N}\left(0, \frac{1}{q}\right) & \text{with probability } q\,, \\ 0 & \text{with probability } 1 - q\,, \end{cases} \tag{4.27}$$

$$S_{ij} = \frac{1}{D}(-1)^{\langle i-1, j-1 \rangle}\,, \tag{4.28}$$

$$D_{ii} = \begin{cases} +1 & \text{with probability } \frac{1}{2}\,, \\ -1 & \text{with probability } \frac{1}{2}\,, \end{cases} \tag{4.29}$$

where $\langle i, j \rangle$ is the modulo-2 dot product of the bit vectors expressing i and j in binary, and $q = \min\{\Theta((\log^2 N)/D), 1\}$ for approximately preserving pairwise dis-

tances under the ℓ_2-norm. This R matrix can be sparser than the G matrix of Achlioptas (2002). It will not necessarily preserve pairwise distances if applied directly to B, as it can interact poorly if B itself is also sparse, but the pre-conditioning of B by ST serves to ensure that such bad interactions are low-probability. Ailon and Chazelle (2009) show that the subsequent GB projection can be performed in $O(D \log D + \min\{D\epsilon^{-2} \log N, \epsilon^{-2} \log^3 N\})$ operations.

It would be advantageous, in terms of runtime, for DPP algorithms to use the G matrices of Achlioptas (2002) and Ailon and Chazelle (2009) rather than a Gaussian G. Unfortunately though, these projections are only known to preserve pairwise distances, not volumes. Thus, one possible avenue for future research would be to see if the pairwise distance preservation proofs can be extended to volumes. However, even if this is possible, note that the speedups achieved for computing the projection GB may be dwarfed by the time required for sampling from the resulting DPP. In our experiments on news data, the GB projection was not the most time-intensive operation.

A more practical direction, for future work on alternative forms of the projection matrix G, might be to search for a projection that tightens the existing volume-preservation bounds of Magen and Zouzias (2008). It seems somewhat unlikely though that any uninformed approach, where G does not take into account any information about B, will do a significantly better job of preserving volumes. One approach that *does* take various statistics of B into account when reducing the dimension from D to d is the Nyström approximation. As there is existing related work applying this approximation to DPPs, we will discuss it further in the following section.

4.6 Related DPP work

We conclude this chapter's discussion of the large-N, large-D setting by summarizing some recent related DPP work. The first of the two papers we survey in this section, Kang (2013), proposes several Markov chain Monte Carlo (MCMC) sampling methods for DPPs. Initially, one of these seemed especially promising for the large-N setting, as the mixing time stated in Kang (2013) is N-independent. Unfortunately though, this mixing time is incorrect. We show here that these MCMC

algorithms are not efficient for the general case (encompassing all positive definite kernel matrices), as it is impossible to prove fast mixing times without some stronger assumptions on the eigenvalues. We give examples illustrating this. The second paper we survey, Affandi et al. (2013b), focuses on reducing the number of items N by first selecting a few "landmark" items. In contrast to random projections, this approach can handle not just a large number of features, D, but even an infinite number. It also comes with approximation bounds that are potentially tighter than those of the random projections approach. However, at first glance it appears that this approach is not compatible with structured DPPs, due to its use of a pseudoinverse. Despite this apparent incompatibility, we extend Affandi et al. (2013b)'s work to show here one way that its guarantees can carry over to the setting where a few landmark *features* are sampled instead. This opens up the possibility of applying the method to a structured application.

4.6.1 MCMC sampling

Kang (2013) proposes a method for sampling from a DPP that avoids the initial eigendecomposition step. Specifically, the proposed algorithm falls into the class of Markov chain Monte Carlo (MCMC) methods. It proceeds as follows. Starting from a random set Y, sample an item i uniformly at random from $\mathcal{Y}$. If i is already in Y, remove it with probability:

$$\min\left\{1, \frac{\det(L_{Y\setminus\{i\}})}{\det(L_Y)}\right\}. \tag{4.30}$$

If i is not in Y, add it with probability:

$$\min\left\{1, \frac{\det(L_{Y\cup\{i\}})}{\det(L_Y)}\right\}. \tag{4.31}$$

Naïvely computing these transition probabilities would not result in an asymptotically faster algorithm than standard DPP sampling algorithms, such as Algorithm 2: computing any single $\det(L_Y)$ costs $O(|Y|^\omega)$, and $|Y|$ could be as large as N. However, by applying the Schur determinant identity (recall Definition 2.3), it is possible to compute each of these ratios in $O(|Y|^2)$ time.

Let $\tau_N(\epsilon)$ represent this Markov chain's mixing time. That is, after $\tau_N(\epsilon)$ steps, the chain produces a sample that is guaranteed to be from a distribution ϵ-close to

$\mathcal{P}_L$ in terms of total variation. This yields an algorithm that can generate a sample in $O(k^2\tau(\epsilon))$ time, where k is the average size of Y encountered as the Markov chain is mixing. If the mixing time is fast enough, then this procedure could generate an initial sample more quickly than the basic DPP sampling algorithms such as Algorithm 2. However, no proof of a fast mixing time is known, as the mixing time derivation given in Kang (2013) is incorrect. In fact, we can show that, absent additional assumptions beyond L being positive definite, the mixing time can be arbitrarily bad.

Example 4.4. *Consider the following positive definite matrix:*

$$L = \begin{bmatrix} a & a-\epsilon \\ a-\epsilon & a \end{bmatrix}. \tag{4.32}$$

The probability of the first singleton set is:

$$\mathcal{P}_L(\{1\}) = \frac{a}{1+2a+a^2-(a-\epsilon)^2}, \tag{4.33}$$

which is identical to the probability of $\{2\}$*. Yet, if the Markov chain starts at* $\{1\}$ *and proceeds as described by Equations (4.30) and (4.31), considering a single addition or deletion on each step, it can take arbitrarily long to reach* $\{2\}$*. To see this, notice that to get from* $\{1\}$ *to* $\{2\}$ *requires passing through either the state* $\{1,2\}$ *or the state* $\{\}$*.*

- *Through* $\{1,2\}$*: The set* $\{1,2\}$ *can have arbitrarily small probability relative to* $\{1\}$*, for small* ϵ *and large* a*:* $\frac{a^2-(a-\epsilon)^2}{a} \to 0$ *as* $a \to \infty$*.*
- *Through* $\{\}$*: The empty set is defined to have determinant value* 1*, which can be arbitrarily small compared to* a*:* $\frac{1}{a} \to 0$ *as* $a \to \infty$*.*

A simple fix to bound the mixing time for the above example would be to require L's condition number to be sufficiently small: bounding $\frac{\lambda_{\max}}{\lambda_{\min}} \leq b$ for some constant b. If the resulting mixing time were found to be short, then this would motivate using the MCMC scheme for sampling within this sub-class of L matrices. There remains the caveat though that, given the initial eigendecomposition, Algorithm 2 can always draw each successive sample in $O(Nk^2)$ time, where k is the size of the sampled set Y. This will almost certainly be faster than waiting for the Markov chain to mix. So, if many samples are desired, this MCMC scheme will likely not be the most efficient technique.

Extension to k-DPPs

Kang (2013) further proposes a k-DPP variant of the MCMC algorithm. This variant starts from a random size-k set Y, then samples an item i uniformly at random from Y and an item j uniformly at random from $\mathcal{Y} \setminus Y$. The element j then replaces i in Y with probability:

$$\min\left\{1, \frac{\det(L_{(Y\setminus\{i\})\cup\{j\}})}{\det(L_Y)}\right\}. \tag{4.34}$$

Naïvely this determinant ratio would take $O(k^3)$ time to compute, but the Schur determinant identity offers a means of computing it in $O(k^2)$ time. Thus, the overall sampling algorithm is $O(k^2\tau_{N,k}(\epsilon))$, where $\tau_{N,k}(\epsilon)$ is the mixing time. As for the other MCMC algorithm, no proof of a fast mixing time is known, since the mixing time derivation given in Kang (2013) is incorrect. This algorithm does allow for larger steps though, performing both an addition and a deletion in a single step, which gets around the issue illustrated by Example 4.4. However, with a slightly more complex example, we can again show that, absent additional assumptions beyond L being positive definite, the mixing time can be arbitrarily bad.

Example 4.5. *Consider the case where N is even and L is defined as follows:*

$$L_{ij} = \begin{cases} 1 & \text{if } i = j\,, \\ 1-\epsilon & \text{if } i \neq j \text{ and } i, j \geq N/2\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{4.35}$$

Intuitively, we can think of this matrix as consisting of two types of items. The first, which we will call set A, consists of $N/2-1$ distinct items that have no similarity to other items (zero off-diagonal). The second, which we will call set B, consists of $N/2+1$ nearly identical items (off-diagonal close to 1) that have no similarity to the items in A. Let the target set size be $k = N/2$. Then we can provide a large lower bound on mixing time due to the fact that any set that does not include A has essentially zero probability: such a set would have to include two items from B, whose similarity would make the determinant value near-zero. Thus, whenever the algorithm randomly selects one of the elements from A for removal, this move will essentially always be rejected, which results in far too high of a rejection rate.

More formally, let T represent the Markov chain's transition matrix. Let the two largest eigenvalues of T be denoted by α_1 and α_2, with α_1 being the largest. By definition, T is stochastic: $\sum_{Y':|Y'|=k} T_{Y,Y'} = 1$ for all Y. Thus, its largest eigenvalue is $\alpha_1 = 1$. Its second-largest eigenvalue, α_2, can be used to lower-bound the mixing time. Levin, Peres, and Wilmer (2008, Theorem 12.4) show that:

$$\tau_N(\epsilon) \geq \log\left(\frac{1}{2\epsilon}\right)\left(\frac{\alpha_2}{1-\alpha_2}\right). \tag{4.36}$$

This means that $\tau_N(\epsilon) \to \infty$ as $\alpha_2 \to 1$. We can show that α_2 is arbitrarily close to 1.

First, note that in the limit $\epsilon \to 0$, we have $\text{rank}(T) \to \frac{N}{2}$, which follows from the fact that any set including more than one item from B approaches determinant 0. Similarly, at $\epsilon = 0$ we have a simple formula for $\text{trace}(T)$: $\frac{N-2}{N}\left(\frac{N}{2}+1\right)$. This follows from the fact that T's diagonal entries are:

$$T_{Y,Y} = \begin{cases} \frac{N-2}{N} & \text{if } Y = A \cup B_i \text{ for some } i\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{4.37}$$

The first value is implied by the fact that anytime an item from A is selected for removal (happens with probability $\frac{N/2-1}{N/2}$), the removal move will be rejected. The second value (the zero) is implied by the fact that any size-k set not consisting of A and one item from B has determinant 0. Thus, the second eigenvalue can be lower bounded as follows:

$$\alpha_2 \geq \frac{\text{trace}(T) - \alpha_1}{\text{rank}(T) - 1} \tag{4.38}$$

$$= \left[\frac{N-2}{N}\left(\frac{N}{2}+1\right) - 1\right]\left[\frac{N}{2} - 1\right]^{-1} \tag{4.39}$$

$$= \frac{N+2}{N} - \frac{2}{N-2} = 1 + \frac{2}{N} - \frac{2}{N-2}\,. \tag{4.40}$$

The inequality follows from the fact that the trace is a sum of the eigenvalues and the rank is a count of the nonzero eigenvalues. The second line substitutes $\alpha_1 = 1$, and the third line simplifies. This expression implies that $\alpha_2 \to 1$ as $N \to \infty$.

Despite the negative result of the above example, the same simple fix proposed earlier for the non-cardinality constrained MCMC algorithm might also suffice to prove an interesting bound in the constrained setting. More concretely, requiring L to have a low condition number might suffice. Thus, there remains the possibility that MCMC algorithms could prove to be faster than the standard DPP sampling methods for some sub-class of kernel matrices.

4.6.2 Nyström approximation

The work most closely related to the random projections approach covered in this chapter is that of Affandi et al. (2013b). Their work also addresses the problem of approximating DPPs in the setting where N and D are large. In contrast to the random projections approach though, their method is applicable even when D is infinite. Considering the fact that a Gaussian kernel falls into this infinite-D class, with features of the form:

$$L(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|_2\right) \tag{4.41}$$

$$= \exp\left(-\frac{(\|\boldsymbol{x}\|_2^2 + \|\boldsymbol{x}'\|_2^2)}{2\sigma^2}\right) \sum_{i=0}^{\infty} \frac{1}{i!}\left(\frac{\boldsymbol{x}^\top \boldsymbol{x}'}{\sigma^2}\right)^i, \tag{4.42}$$

it is of practical importance to be able to handle this setting.

The approach Affandi et al. (2013b) explore is applying a Nyström approximation to project a DPP kernel to a low-dimensional space. A Nyström approximation involves selecting a small number ℓ of "landmark" points, $W \subseteq \mathcal{Y}$, and expressing L as the Gram matrix of the resulting $\ell \times N$ matrix:

$$B = (L_W^+)^{1/2} L_{W,:} \tag{4.43}$$

where L_W^+ is the pseudoinverse of L_W. This corresponds to approximating L with $\tilde{L}$:

$$L = \begin{pmatrix} L_W & L_{W,\overline{W}} \\ L_{\overline{W},W} & L_{\overline{W}} \end{pmatrix} \longrightarrow \tilde{L}(W) = \begin{pmatrix} L_W & L_{W,\overline{W}} \\ L_{\overline{W},W} & L_{\overline{W},W} L_W^+ L_{W,\overline{W}} \end{pmatrix}. \tag{4.44}$$

Notice that this approximation, with its pseudoinverse, may not be acceptable in the structured DPP setting. It is likely to break the product-of-qualities and sum-of-similarities decomposition requirements, Equation (3.6), needed for efficient structured DPP inference. We will show at the end of this section though how a Nyström approximation to the dual kernel C, can be effective in the structured setting.

Previous to the work of Affandi et al. (2013b), theoretical results bounding the Frobenius or spectral norm of the kernel error matrix, $E = L - \tilde{L}$, were known. For example, for the setting where landmarks are sampled with probability proportional to the squared diagonal entries of L, Drineas and Mahoney (2005) showed that

choosing at least $O(k/\epsilon^4)$ landmarks guarantees:

$$\|L - \tilde{L}\|_\beta \leq \|L - L_k\|_\beta + \epsilon \sum_{i=1}^{N} L_{ii}^2 \,, \tag{4.45}$$

in expectation, where L_k is the best rank-k approximation to L and $\beta = 2$ or $\beta = F$. More recent results such as those of Deshpande, Rademacher, Vempala, and Wang (2006) give additional bounds for the setting where landmarks are sampled adaptively, such that the choice of a landmark decreases the probability of choosing similar landmarks.

The main contribution of Affandi et al. (2013b) was to provide an upper bound on a *distributional* measure of kernel similarity, the L_1 variational distance from Equation (4.13). A simple example helps put into perspective the difference between bounds on matrix norms and bounds on variational distances (Affandi et al., 2013b, Example 1). Specifically, it is possible for the kernel error matrix to have a small norm, but still be far from the correct DPP distribution.

Example 4.6. *Let L be an $N \times N$ diagonal matrix with value α in all entries except the last, to which we assign value ϵ: $L_{NN} = \epsilon$. Let $\tilde{L}$ be identical to L, except in the last entry: $\tilde{L}_{NN} = 0$. Then $\|L - \tilde{L}\|_F = \|L - \tilde{L}\|_2 = \epsilon$, but for any size-$k$ set A that includes index N, the difference in determinants is:* $\det(L_A) - \det(\tilde{L}_A) = \epsilon\alpha^{k-1}$.

A slightly more involved example suffices to show that even for cases where $\|L - \tilde{L}\|_F$ and $\|L - \tilde{L}\|_2$ tend to zero as $N \to \infty$, it is possible to have variational distance $\|\mathcal{P} - \tilde{\mathcal{P}}\|_1$ approach a limit > 0 (Affandi et al., 2013b, Example 2).

Despite these negative results, Affandi et al. (2013b) are able to provide bounds on the L_1 variational distance for both DPPs and k-DPPs. Their proofs rely on applying Weyl's inequality for the eigenvalues of PSD matrices to the error matrix $E = L - \tilde{L}$. The final bounds depend only on the eigenvalues of L, the eigenvalues of the set A under consideration, and the spectral norm of the error matrix. More

precisely, for a set A under the k-DPP $\mathcal{P}^k$:

$$|\mathcal{P}^k(A) - \tilde{\mathcal{P}}^k(A)| \leq \mathcal{P}^k(A) \max\left\{ \left[\frac{e_k(L)}{e_k(\hat{\lambda}_1, \ldots, \hat{\lambda}_N)} - 1 \right], \left[1 - \frac{\prod_{i=1}^k \hat{\lambda}_i^A}{\prod_{i=1}^k \lambda_i} \right] \right\}, \quad (4.46)$$

$$\text{where } \operatorname{rank}(L) = m, \quad \operatorname{rank}(\tilde{L}) = r, \quad (4.47)$$

$$\hat{\lambda}_i = \max\{\lambda_{i+(m-r)}, \lambda_i - \|L - \tilde{L}\|_2\}, \text{ and} \quad (4.48)$$

$$\hat{\lambda}_i^A = \max\{\lambda_i^A - \|L - \tilde{L}\|_2, 0\}. \quad (4.49)$$

The λ_i here are the eigenvalues of L and the λ_i^A are the eigenvalues of L_A. These bounds are tight for the diagonal matrix examples mentioned above.

Affandi et al. (2013b) test the Nyström approximation method on a motion capture summarization task. The inputs are recordings of human subjects performing motions characteristic of a particular activity (e.g. dancing, playing basketball). The goal is to select a small number of video frames to summarize each activity. Each video has an average of $N \approx 24,000$ frames, so constructing an $N \times N$ kernel would be intractable. Constructing the $D \times D$ dual kernel C would be feasible, as the feature vector for a video frame simply consists of $D = 62$ values listing the locations of the motion capture sensors. With this C, it would be tractable to directly apply the sampling algorithms from Section 2.4. However, Affandi et al. (2013b) argue that a Gaussian kernel is a more natural fit for problems where items have inherent geometric relationships, such as this one. Thus, they define as the target kernel L a matrix where frames with feature vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ are related as in Equation (4.42). This kernel is then approximated by sampling landmarks with probability proportional to the diagonal of the error matrix, E_{ii}^2, for $E = L - \tilde{L}$. In a study where each human judge was shown 10 frames sampled independently at random and 10 frames sampled from a k-DPP with kernel $\tilde{L}$, the latter was voted the better overall activity summary 67.3% of the time.

Affandi et al. (2013b) also investigate using random Fourier features (RFFs) as an alternative to the Nyström approximation. For several datasets from the UCI repository (uci), empirical tests demonstrate that the Nyström approximation is the better approach. In their work on continuous DPPs though, Affandi et al. (2013a) found that RFFs can have an advantage when the eigenspectrum decays slowly.

Suppose that instead of trying to reduce the rank of the L matrix by selecting ℓ landmark items, we instead reduced it by selecting d landmark *features*. That is, we could replace the dual kernel $C = BB^\top$ by its Nyström approximation, call it $\dot{C}$, instead of replacing L by $\tilde{L}$. This $\dot{C}$ though is not sufficient information for running the standard dual DPP inference algorithms, such as Algorithm 3. These algorithms require that $\dot{C}$ be related to a primal kernel, call it $\dot{L}$, via a known $d \times N$ matrix $\dot{B}$, as described in Proposition 2.16: $\dot{L} = \dot{B}^\top \dot{B}, \dot{C} = \dot{B}\dot{B}^\top$. Finding such a $\dot{B}$ seems difficult; we cannot hope to obtain $\dot{B}$ by a Cholesky decomposition of $\dot{C}$, as this only yields $D \times D$ matrices. However, consider taking the eigendecomposition $\dot{C} = \dot{V}\dot{\Lambda}\dot{V}^\top$, where $\dot{V} \in \mathbb{R}^{D\times d}$ and $\dot{\Lambda} \in \mathbb{R}^{d\times d}$, and using it in combination with the original B to construct the following matrix: $\ddot{B} = \dot{V}^\top B$. This matrix is $d \times N$, as desired. Moreover, as Lemma 4.7 shows, the corresponding $\ddot{L} = \ddot{B}^\top \ddot{B}$ matrix has the desirable property that its error matrix is PSD. This property is in fact all that is needed to extend the proofs from Affandi et al. (2013b) from $\tilde{L}$ to $\ddot{L}$.

Lemma 4.7. *Let $B \in \mathbb{R}^{D\times N}$ be a rank-D matrix, and define $C = BB^\top$ and $L = B^\top B$. Let $\dot{C}$ be the Nyström approximation of C constructed from d landmark features constituting the set W. Let $\dot{V} \in \mathbb{R}^{D\times d}, \dot{\Lambda} \in \mathbb{R}^{d\times d}$ represent its eigendecomposition. Then for $\ddot{B} = \dot{V}^\top B$ and $\ddot{L} = \ddot{B}^\top \ddot{B}$, the following error matrix:*

$$\ddot{E} \equiv L - \ddot{L} \tag{4.50}$$

is positive semi-definite with rank $\operatorname{rank}(\ddot{E}) = \operatorname{rank}(L) - \operatorname{rank}(\ddot{L})$.

Proof. Expanding according to the definition of L and $\ddot{L}$:

$$\ddot{E} \equiv L - \ddot{L} = B^\top B - B^\top \dot{V}\dot{V}^\top B = B^\top (I - \dot{V}\dot{V}^\top) B\,. \tag{4.51}$$

Let $Z = I - \dot{V}\dot{V}^\top$. This matrix is idempotent:

$$Z^2 = I - 2\dot{V}\dot{V}^\top + \dot{V}\dot{V}^\top\dot{V}\dot{V}^\top = I - 2\dot{V}\dot{V}^\top + \dot{V}I\dot{V}^\top = I - \dot{V}\dot{V}^\top = Z\,. \tag{4.52}$$

This, combined with it symmetry, implies that it is a projection matrix. These properties also imply that Z is PSD:

$$\boldsymbol{x}^\top Z\boldsymbol{x} = \boldsymbol{x}^\top Z^2\boldsymbol{x} = \boldsymbol{x}^\top Z^\top Z\boldsymbol{x} = \|Z\boldsymbol{x}\|_2^2 \geq 0\,. \tag{4.53}$$

Thus, there must exist a decomposition $Z = T^\top T$ for some matrix $T \in \mathbb{R}^{D \times D}$. This means we can re-express $\ddot{E}$ as:

$$\ddot{E} = B^\top Z B = B^\top T^\top T B = Q^\top Q\,, \tag{4.54}$$

for $Q = TB$. Thus, $\ddot{E}$ is also PSD. Turning now to the question of rank, first note that the rank of $\ddot{E}$ reduces to the rank of Q, which is $\min(\text{rank}(T), \text{rank}(B))$. The statement of the lemma assumes $\text{rank}(B) = D$. The rank of T is identical to the rank of $Z = T^\top T$. Recalling that Z is a projection matrix, we have: $\text{rank}(Z) + \text{rank}(I - Z) = D$ (Yanai, Takeuchi, and Takane, 2011, Equation 2.11). Since $I - Z = \dot{V}\dot{V}^\top$, this matrix has rank equal to that of $\dot{V}$. Accordingly, Z (and hence T) has rank $D - \text{rank}(\dot{V})$. Thus, we have $\text{rank}(\ddot{E}) = D - \text{rank}(\dot{V})$. The matrices $L = B^\top B$ and $\ddot{L} = B^\top \dot{V}\dot{V}^\top B$ have ranks equal to $\text{rank}(B)$ and $\text{rank}(\dot{V})$, respectively, completing the proof. □

The properties established in Lemma 4.7 for $\ddot{E}$ are sufficient to translate the results from Affandi et al. (2013b) for $\tilde{L}$ to $\ddot{L}$. For example, the k-DPP result from Equation (4.46) holds for $\tilde{L}$ replaced by $\ddot{L}$. This by itself is not necessarily a good bound though. To see why, first note that the properties of the error matrix established in Lemma 4.7 hold for *any* low-dimensional projection of B (not just $\ddot{B}$), and certainly some such projections will not preserve L well. The missing piece of the puzzle here is that bounds such as Equation (4.45), capping the value of $\|L - \tilde{L}\|_2$ at $\|L - L_k\|_2$ plus a small error term, where L_k is the best rank-k approximation, do not necessarily hold for $\ddot{L}$. We suspect that such bounds can be proven for the particular projection $\ddot{B} = \dot{V}^\top B$, but leave the proof of such bounds on $\|L - \ddot{L}\|_2$ to future work.

Algorithm 4 summarizes the procedure for sampling from L via $\ddot{L}$. The $\hat{\Lambda}^+$ notation indicates the matrix pseudoinverse (inversion of all non-zero eigenvalues). Line 4's construction of $\dot{C}$'s eigendecomposition from C_W's eigendecomposition is a standard feature of the Nyström approximation (Li, Kwok, and Lu, 2010, Algorithm 2).

The runtime of Algorithm 4 is linear in D. The eigendecomposition of C_W takes $O(d^3)$ time, and the subsequent construction of $\dot{V}$ takes $O(Dd^2)$ time. The multiplication to create $\ddot{B}$ takes $O(NDd)$ time, and to create $\ddot{C}$ takes $O(Nd^2)$ time. Its eigendecomposition is an $O(d^3)$ operation. The `DualDPPSample` step takes $O(Nd^2k)$

time, for a size-k sample set Y. Thus, overall the Nyström-based dual DPP sampling procedure takes $O(Dd^2 + NDd + Nd^2k)$ time.

Algorithm 4: Nyström-based Dual DPP Sampling

Input: B and chosen landmark feature indices $W = \{i_1, \ldots, i_d\}$
$C_W \leftarrow$ principal submatrix of C, indexed by W
$\hat{V}, \hat{\Lambda} \leftarrow$ eigendecomposition of C_W
$\dot{V} \leftarrow \sqrt{\frac{d}{D}} C_{:,W} \hat{V} \hat{\Lambda}^{+}$
$\ddot{B} \leftarrow \dot{V}^{\top} B$
$\ddot{C} \leftarrow \ddot{B}\ddot{B}^{\top}$
$\ddot{V}, \ddot{\Lambda} \leftarrow$ eigendecomposition of $\ddot{C}$
$Y \leftarrow$ `DualDPPSample`$(\ddot{B}, \ddot{V}, \ddot{\Lambda})$ (Algorithm 3)
Output: Y

Extension to structured DPPs

The previous section provides a translation of Affandi et al. (2013b)'s proofs from selection of landmark items to selection of landmark features. We can build off of this to address the structured DPP setting. First, note that selection of landmark items is not an option in this setting. Structured DPPs rely on product-of-qualities and sum-of-similarities decomposition rules, summarized in Equation (3.6), for efficient inference. Unfortunately, even if L satisfies these rules, the Nyström approximation $\tilde{L}$ based on selecting landmark items W can violate them. The simple example given below illustrates this point.

Example 4.8. *Consider length-2 structures where the features for a single structure $\boldsymbol{y}$ decompose as:*

$$B_{\boldsymbol{y}} = q(\boldsymbol{y})\phi(\boldsymbol{y}) = q(\boldsymbol{y}_1)q(\boldsymbol{y}_2)(\phi(\boldsymbol{y}_1) + \phi(\boldsymbol{y}_2)) \,. \tag{4.55}$$

The matrix L then has entries:

$$L_{\boldsymbol{y},\boldsymbol{y}'} = q(\boldsymbol{y}_1)q(\boldsymbol{y}_2)q(\boldsymbol{y}'_1)q(\boldsymbol{y}'_2)(\phi(\boldsymbol{y}_1) + \phi(\boldsymbol{y}_2))^{\top}(\phi(\boldsymbol{y}'_1) + \phi(\boldsymbol{y}'_2)) \,. \tag{4.56}$$

Further suppose that there are just two possible structures (say, two possible instantiations for $\boldsymbol{y}_1$ and one for $\boldsymbol{y}_2$). Let the landmark set W consist of a single item, call it $\boldsymbol{y}$. Let the

other item be represented by $\boldsymbol{y}'$. Then the Nyström approximation for L has entries:

$$\tilde{L}_{\boldsymbol{y},\boldsymbol{y}} = L_{\boldsymbol{y},\boldsymbol{y}}, \quad \tilde{L}_{\boldsymbol{y},\boldsymbol{y}'} = L_{\boldsymbol{y},\boldsymbol{y}'}, \quad \tilde{L}_{\boldsymbol{y}',\boldsymbol{y}} = L_{\boldsymbol{y}',\boldsymbol{y}}, \tag{4.57}$$

$$\tilde{L}_{\boldsymbol{y}',\boldsymbol{y}'} = L_{\boldsymbol{y}',\boldsymbol{y}} L^{+}_{\boldsymbol{y},\boldsymbol{y}} L_{\boldsymbol{y},\boldsymbol{y}'} = \frac{(L_{\boldsymbol{y}',\boldsymbol{y}})^2}{L_{\boldsymbol{y},\boldsymbol{y}}} \tag{4.58}$$

$$= q(\boldsymbol{y}'_1)^2 q(\boldsymbol{y}'_2)^2 g(\boldsymbol{y}) [(\phi(\boldsymbol{y}_1) + \phi(\boldsymbol{y}_2))^\top (\phi(\boldsymbol{y}'_1) + \phi(\boldsymbol{y}'_2))]^2 . \tag{4.59}$$

where $g(\boldsymbol{y})$ is a function that is constant with respect to $\boldsymbol{y}'$. While this final entry $L_{\boldsymbol{y}',\boldsymbol{y}}$ may decompose as a product over part qualities and a sum over part similarity features for some *definition of part qualities and part similarity features, it clearly does not decompose according to the same definitions used for the other entries in the $\tilde{L}$ matrix.*

Since selecting landmark items can violate the decompositions necessary for efficient structured inference, suppose that we instead consider the selection of landmark features, as in the previous section. For structured DPPs, N is of exponential size, so in practice the $D \times N$ matrix B is never explicitly constructed. Instead, an $D \times M^c R$ matrix, consisting of features for all of the possible components, is used: for a structured DPP with R parts, each of which takes a value from a finite set of M possibilities, if we assume the degree of any factor in its factor graph is bounded by c, then there are at most $M^c R$ components. Let H denote this matrix, and consider substituting H for B in Line 5 of Algorithm 4. The multiplication by $\dot{V}^\top$ there is a linear transformation of the original D features, just as the random projections of Theorem 4.3 are linear. Thus, transforming H is equivalent to transforming B itself; the algorithm remains unchanged. The subsequent computation of $\ddot{C}$ and the `DualDPPSample` can be done efficiently using standard structured DPP inference algorithms, such as those summarized in Section 3.2.

The runtime of Algorithm 4 remains linear in D with these structured modifications. The computation of C_W requires $O(M^c R d^2)$ time in the structured case, but the time required for its eigendecomposition ($O(d^3)$) and the construction of $\dot{V}$ ($O(Dd^2)$) remain unchanged. The construction of $\ddot{H}$ requires time $O(M^c RDd)$, and creating $\ddot{C}$ takes $O(M^c R d^2)$ time. The `DualDPPSample` runtime becomes $O(d^2 k^3 + M^c R d k^2)$ for a size-k sample set Y (Kulesza, 2012, Algorithm 11). Thus, given that $D \geq d \geq k$, the overall structured Nyström-based sampling algorithm requires $O(M^c RDd^2 + d^2 k^3)$ time.

5

MAP estimation

Maximum a posteriori (MAP) estimation is the problem of finding the highest-probability set under a DPP:

$$Y = \operatorname*{arg\,max}_{Y' : Y' \subseteq \mathcal{Y}} \det(L_{Y'}) \,. \tag{5.1}$$

We have seen in the preceding chapters that it is possible to efficiently perform many DPP inference tasks, such as normalization and sampling. In contrast, as mentioned in Section 2.2.5, the DPP MAP task is NP-hard, and has been shown to have no PTAS (Ko et al., 1995; Çivril and Magdon-Ismail, 2009). This is especially unfortunate, as the MAP solution is exactly the set that would best satisfy many standard subset selection problems, such as those described in Section 1.1. That is, assuming we have a DPP kernel L that is a good match to a given subset selection task (in that its minors accurately represent the goodness of the sets), finding the largest minor would yield the best set. Regrettably, the hardness of this problem dictates that we must be satisfied with approximate solutions.

Most work applying DPPs to subset selection tasks has thus far approximated the MAP by sampling. In its simplest form this amounts to drawing a single sample, or repeatedly sampling and returning the highest-scoring sample. In the case

where the DPP kernel L may not be a perfect match to the task at hand, Kulesza (2012, Page 79) proposed minimum Bayes risk decoding. This is a more sophisticated sampling-based MAP approximation technique: we sample repeatedly, but rather than returning the sample Y with maximum minor $\det(L_Y)$, we hedge bets by returning the sample that is most similar to the other samples. More precisely, given a pool of R samples and some (potentially task-specific) measure of set similarity g, this method computes:

$$Y^{\text{MBR}} = \operatorname*{arg\,max}_{Y^{r'}:r'\in\{1,\ldots,R\}} \frac{1}{R}\sum_{r=1}^{R} g(Y^r, Y^{r'})\,. \tag{5.2}$$

In this chapter though, we will move away from sampling-based MAP approximations, and we will also assume that the DPP kernel L is indeed a good match for the task at hand; Chapter 6 considers several ways to construct L such that this is approximately true.

The approach we take to MAP estimation is to exploit *submodularity*. There is a large body of work on approximately maximizing submodular functions, and we can leverage this because the function $f(Y) = \log\det(L_Y)$ is submodular. We will discuss related submodular maximization work in more detail later in the chapter, but give a brief overview here. A greedy algorithm of Nemhauser et al. (1978) offers an approximation guarantee of $1-\frac{1}{e}$ for monotone submodular functions, but does not apply for general DPPs, as they are non-monotone. A more recent greedy algorithm by Buchbinder, Feldman, Naor, and Schwartz (2012) does give an approximation guarantee for the non-monotone setting, but unfortunately, it does not perform well in practice for the log det objective, even compared to the Nemhauser et al. (1978) algorithm; see experiments in Section 5.6. Thus, instead of developing combinatorial greedy algorithms, we focus on *continuous* techniques for submodular function maximization. In other words, techniques that represent a set Y by its characteristic vector $\boldsymbol{x} \in \{0,1\}^N$, with $x_i = \mathbb{1}(i \in Y)$. These techniques relax $\boldsymbol{x}$ such that its entries can be anywhere in the 0 to 1 range, $x_i \in [0,1]$, and replace $f(Y)$ with a related function $F(\boldsymbol{x})$. They optimize this function to get $\hat{\boldsymbol{x}}$, and then round $\hat{\boldsymbol{x}}$ to get a corresponding set $\hat{Y}$. We present a novel continuous relaxation $\tilde{F}$, which, in contrast to the standard relaxation F used for general submodular functions, can be evaluated and differentiated exactly and efficiently for log det.

Ultimately, we obtain a practical MAP algorithm with a $\frac{1}{4}$-approximation guarantee. The continuous nature of the algorithm allows it to extend neatly to MAP inference under complex polytope constraints, although we do not have approximation guarantees for this setting. Nevertheless, this extension opens the door to combining DPPs with other models such as Markov random fields and weighted matchings. In Section 5.6, we demonstrate that our approach outperforms several standard submodular maximization methods on both synthetic and real-world data. The majority of the information that this chapter conveys can also be found in Gillenwater, Kulesza, and Taskar (2012b).

5.1 Definition of submodularity

As always, let $\mathcal{Y}$ denote a ground set consisting of N items: $\mathcal{Y} = \{1, \ldots, N\}$. A set function $f : 2^{\mathcal{Y}} \rightarrow \mathbb{R}$ maps each subset of this ground set $\mathcal{Y}$ to a real number. A set function qualifies as *submodular* if it obeys the law of diminishing returns.

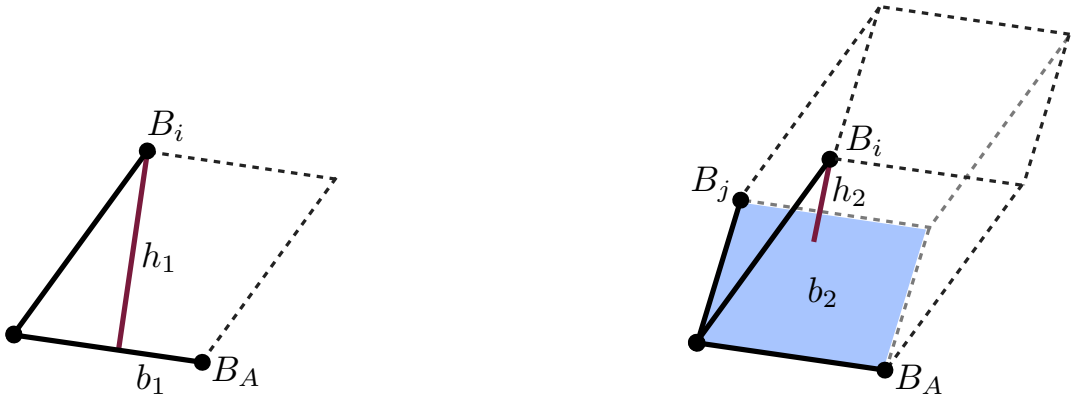
Definition 5.1. Submodularity by diminishing returns: *The function $f : 2^{\mathcal{Y}} \rightarrow \mathbb{R}$ is submodular if and only if $\forall A, B \subseteq \mathcal{Y}$ such that $A \subseteq B$ and $i \in \mathcal{Y} \setminus B$, it is the case that $f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A)$.*

In words: a new element contributes more value when added to set A than when added to any superset of A. Diminishing returns is equivalent to several other simple conditions.

Definition 5.2. Submodularity by set combination: *The function $f : 2^{\mathcal{Y}} \rightarrow \mathbb{R}$ is submodular if and only if $\forall A, B \subseteq \mathcal{Y}$ it is the case that $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$.*

Definition 5.3. Submodularity by second-order differences: *The function $f : 2^{\mathcal{Y}} \rightarrow \mathbb{R}$ is submodular if and only if $\forall A \subseteq \mathcal{Y}$ and $i, j \in \mathcal{Y} \setminus A$, it is the case that $f(A \cup \{i, j\}) - f(A \cup \{j\}) \leq f(A \cup \{i\}) - f(A)$.*

Note that while Definition 5.3 seems weaker than Definition 5.1, as it corresponds to restricting to $B = A \cup \{j\}$, it is nonetheless equivalent (Bach, 2013, Propositions 2.2 and 2.3).



$$\frac{\mathrm{vol}(B_{A\cup\{i\}})}{\mathrm{vol}(B_A)} = \frac{b_1 h_1}{b_1} = h_1 \qquad \geq \qquad h_2 = \frac{b_2 h_2}{b_2} = \frac{\mathrm{vol}(B_{A\cup\{i,j\}})}{\mathrm{vol}(B_{A\cup\{j\}})}$$

Figure 5.1: Illustration of Theorem 5.4's argument for $|A| = 1$. **Left**: Adding element i to A changes B_A's volume by the height h_1. **Right**: Adding element i to $A \cup \{j\}$ changes $A \cup \{j\}$'s volume by a lesser factor, h_2. The inequality follows from the fact that "height" is defined as the shortest distance from B_i to the subspace it is joining. The space associated with $A \cup \{j\}$ includes and extends the space associated with A, implying that the shortest distance from B_i to the former can only be smaller than the shortest distance from B_i to the latter.

5.2 Log-submodularity of det

As previously mentioned, the function $f(Y) = \log\det(L_Y)$ is submodular. In other words, det is a log-submodular function. Given that entropy is submodular, one simple proof that log det is submodular is that, as seen in Equation (2.52), the entropy of a Gaussian is proportional to log det. We can also more directly prove the log-submodularity of det by making a geometric argument. Theorem 5.4 does exactly this. Figure 5.1 illustrates the theorem's argument for the simple case of a size-1 base set.

Theorem 5.4. *For a $D \times N$ matrix B with columns indexed by the elements of $\mathcal{Y}$, the function $f : 2^{\mathcal{Y}} \to \mathbb{R}$ defined by:*

$$f(Y) = \log\det(B_Y^\top B_Y) \tag{5.3}$$

is submodular.

Proof. By the second-order differences definition of submodularity, Definition 5.3, f is submodular if, for all sets $A \subseteq \mathcal{Y}$ and all $i, j \notin A$:

$$f(A \cup \{i\}) - f(A) \geq f(A \cup \{i,j\}) - f(A \cup \{j\}) \,. \tag{5.4}$$

Recall from Theorem 1.2 that the parallelotope defined by B_Y's columns has volume $\text{vol}(B_Y) = \sqrt{\det(B_Y^\top B_Y)}$. Thus, we can write $f(Y) = 2 \log \text{vol}(B_Y)$. Then the submodularity condition is equivalent to:

$$\frac{\text{vol}(B_{A\cup\{i\}})}{\text{vol}(B_A)} \geq \frac{\text{vol}(B_{A\cup\{i,j\}})}{\text{vol}(B_{A\cup\{j\}})} \,. \tag{5.5}$$

To establish the correctness of this inequality, start by writing $B_j = B_j^{\parallel} + B_j^{\perp}$, where $B_j^{\parallel}$ is in the span of $\{B_\ell\}_{\ell \in A} \cup B_i$ and $B_j^{\perp}$ is orthogonal to this subspace. Such a decomposition can be found by running the Gram-Schmidt process. As in the proof of Theorem 1.2, the parallel component can be shown to have no effect on the volume, such that the numerator on the righthand side of Equation (5.5) can be written:

$$\text{vol}(B_{A\cup\{i,j\}}) = \text{vol}(B_{A\cup\{i\}}) \text{vol}(B_j^{\perp}) \,. \tag{5.6}$$

Similarly, we can rewrite the denominator of Equation (5.5)'s righthand side by applying the standard "base $\times$ height" formula:

$$\text{vol}(B_{A\cup\{j\}}) = \text{vol}(B_A) \text{vol}\left(\text{proj}_{\perp B_A}(B_j)\right) \,. \tag{5.7}$$

Now, notice that the fraction:

$$\frac{\text{vol}\left(B_j^{\perp}\right)}{\text{vol}\left(\text{proj}_{\perp B_A}(B_j)\right)} \tag{5.8}$$

is ≤ 1, since $B_j^{\perp}$ is the component of B_j orthogonal to a subspace that *includes* B_A. Putting together Equations (5.6) and (5.7) along with this observation, we have:

$$\frac{\text{vol}(B_{A\cup\{i,j\}})}{\text{vol}(B_{A\cup\{j\}})} = \frac{\text{vol}(B_{A\cup\{i\}}) \text{vol}(B_j^{\perp})}{\text{vol}(B_A) \text{vol}\left(\text{proj}_{\perp B_A}(B_j)\right)} \leq \frac{\text{vol}(B_{A\cup\{i\}})}{\text{vol}(B_A)} \,. \tag{5.9}$$

□

Having given some intuition about the log-submodularity of det, we now review relevant literature on submodular maximization.

5.3 Submodular maximization

While submodular minimization is a convex optimization problem, submodular maximization is NP-hard. In fact, submodular maximization generalizes the NP-hard max-cut problem—the problem of dividing the nodes in a graph into two sets such that the number of edges between the sets is maximized. For max-cut there exists a 0.878-approximation algorithm (Goemans and Williamson, 1995), but this algorithm does not easily extend to all submodular functions. The max-cut problem is significantly harder than many other submodular problems though. In fact, max-cut and many of the hardest submodular problems can be characterized as *non-monotone*.

Definition 5.5. Monotonicity: *The submodular function $f : 2^{\mathcal{Y}} \to \mathbb{R}$ is monotone if and only if $\forall A, B \subseteq \mathcal{Y}$, such that $A \subseteq B$, it is the case that $f(A) \leq f(B)$.*

In words: adding elements can never decrease the value of a monotone submodular function. This monotonicity property can be used to divide the class of submodular functions, and also the associated maximization algorithms.

5.3.1 Monotone f

Algorithm 5: greedy

Input: submodular f, ground set $\mathcal{Y}$, limit k
$Y \leftarrow \emptyset, U \leftarrow \mathcal{Y}$
while $|Y| < k$ **do**
  $i^* \leftarrow \arg\max_{i \in U} f(Y \cup \{i\})$
  if $f(Y \cup \{i^*\}) < f(Y)$ **then**
    break
  $Y \leftarrow Y \cup \{i^*\}$
  $U \leftarrow U \setminus \{i^*\}$
return Y

Even restricting to monotone f, the maximization problem is not always easy. In fact, the addition of a simple cardinality constraint such as $|Y| \leq k$ makes the problem NP-hard. One common approximation algorithm for addressing this set-

ting is that of Nemhauser et al. (1978), shown here as Algorithm 5, GREEDY. This algorithm starts from the empty set and greedily adds the item that produces the maximum marginal gain each iteration, until the k limit is reached. Note that for monotone f the if-condition that results in early stopping will never evaluate to "true". The condition is included here only because in later experiments we apply this algorithm to non-monotone f.

As with *all* of the algorithms that we will consider, this one only has an approximation guarantee when f, in addition to being submodular, is non-negative: $f(Y) \geq 0 \; \forall Y$. Without this property, an exponential number of calls to an f-oracle is needed to even decide whether the maximum value is positive or zero (Chekuri, Vondrák, and Zenklusen, 2011, Footnote 4). For the submodular function of interest in the case of DPPs, $f(Y) = \log\det(L_Y)$, the requirement that the DPP kernel L be PSD only implies that $\det(L_Y)$ is non-negative; the log of this expression will be negative whenever $\det(L_Y) < 1$. We can attempt to circumvent this issue by adding a sufficiently large constant η such that:

$$\tilde{f}(Y) = f(Y) + \eta = \log\det(L_Y) + \eta \geq 0\,. \tag{5.10}$$

However, this can damage approximation guarantees. To see how, define η to be:

$$\eta = -\min_{Y:Y\subseteq\mathcal{Y}} \log\det(L_Y)\,, \tag{5.11}$$

the smallest amount that we can add to f such that it is non-negative. If an algorithm with an α-approximation guarantee is run, then, with respect to the true best set Y^*, the set $\hat{Y}$ that it returns has value:

$$\tilde{f}(\hat{Y}) = \log\det(L_{\hat{Y}}) + \eta \geq \alpha(\log\det(L_{Y^*}) + \eta) = \alpha\tilde{f}(Y^*)\,, \tag{5.12}$$

$$\text{which implies:}\quad \log\det(L_{\hat{Y}}) \geq \alpha\log\det(L_{Y^*}) - (1-\alpha)\eta\,. \tag{5.13}$$

The larger the offset η, the less meaningful this statement is. Thus, adding a constant η is not an ideal solution to the problem of negative f values. We could instead make the objective non-negative by replacing L with $L + I$ (essentially marginals of the corresponding DPP), but this is usually a worse solution. These marginals, $\det(K_Y)$, can be misleading: when there are sets of many sizes, small sets with inferior scores can overwhelm less-numerous sets that have larger scores, thus leading the algorithm down the wrong path.

Ultimately, in practice, submodular maximization algorithms such as GREEDY can still do relatively well with typical DPP kernels L, even though these can have $\log\det(L_Y)$ significantly < 0. The continuous approximation algorithm we develop later in this chapter also requires f non-negative for its $\frac{1}{4}$-approximation guarantee, but similarly performs well in practice with f that violate the non-negativity constraint.

One additional property—normalization—is necessary before we can state the approximation guarantee for the GREEDY algorithm. The set function $f : 2^{\mathcal{Y}} \to \mathbb{R}$ is normalized if and only if $f(\emptyset) = 0$. Notice that the submodular function of interest in the case of DPPs satisfies this definition:

$$f(\emptyset) = \log\det(L_\emptyset) = \log(1) = 0\,. \tag{5.14}$$

Nemhauser et al. (1978)'s algorithm provides a $(1 - \frac{1}{e})$-approximation guarantee for *normalized, non-negative, monotone* submodular functions. There is no guarantee, however, when using this algorithm with a *non-monotone* function. Our function of interest, $\log\det$, unfortunately falls into the non-monotone class; adding an element i to the set Y may decrease the det value. Recalling that det is a measure of a set's quality and diversity, it is intuitively clear that a decrease can occur for several reasons: if the new item's magnitude is small, or if it is too similar to any other included item.

5.3.2 NON-MONOTONE f

For non-monotone submodular functions, the maximization problem is harder to solve. In fact, Feige, Mirrokni, and Vondrák (2007) prove that for this setting attaining better than a $\frac{1}{2}$-approximation is hard in the value oracle sense. That is, according to Theorem 4.5 of their work, it would require an exponential number of f-value queries, $e^{\epsilon^2 N/16}$, for any $\epsilon > 0$.

Proof sketch: Feige et al. (2007) construct an f such that for most sets its value is that of the cut function of a complete graph with edge weights of 1 (max value: $\frac{1}{4}N^2$). But for a few sets f has the value of a cut function on a complete bipartite graph with edge weights of 2 (max value: $\frac{1}{2}N^2$). They show that any algorithm would

have to query f an exponential number of times to be assured of finding the bipartite portion of f.

More recently, Dobzinski and Vondrák (2012) translated this result from the value oracle context to complexity theory. Theorem 3.1 of their work states that better than a $\frac{1}{2}$-approximation implies NP = RP.

Proof sketch: Using the same f as Feige et al. (2007), Dobzinski and Vondrák (2012) show how to represent f compactly and explicitly such that distinguishing between the two types of graphs implies deciding Unique-SAT. Unique-SAT is the problem of deciding whether a Boolean formula has exactly one satisfying assignment. There is a randomized polynomial-time (RP) reduction from SAT to Unique-SAT (Valiant and Vazirani, 1986, Theorem 1.1), so Unique-SAT cannot be solved in polynomial time unless NP = RP.

In light of this, the best that can be hoped for, without additional assumptions on f, is a $\frac{1}{2}$-approximation. Buchbinder et al. (2012) present a surprisingly simple algorithm that achieves exactly this. The algorithm is shown as Algorithm 6 here, and we will refer to it as RANDOMIZED-SYMMETRIC-GREEDY. This algorithm is "symmetric" in that it starts from both $f(\emptyset)$ and $f(\mathcal{Y})$, then simultaneously greedily builds up a set X while paring down a set Y until they meet somewhere in the middle. This is somewhat like optimizing both f and its complement, $\bar{f}(Y) = f(\mathcal{Y} \setminus Y)$. (Note that the complement of any submodular function is itself submodular.) A deterministic version, Algorithm 7, SYMMETRIC-GREEDY, yields a $\frac{1}{3}$-approximation. The two algorithms are very similar, with the main difference being that for Algorithm 6 the decision of whether or not to include an item is softened.

In addition to its tight approximation guarantee, RANDOMIZED-SYMMETRIC-GREEDY is also advantageous compared to many other non-monotone submodular maximization algorithms because of its simplicity, which makes it trivial to code, and its relatively low time complexity. It makes $O(N)$ calls to the f-oracle, and the constant hidden by the big-O notation here is at most 4, even for a naïve implementation. However, as we show in Section 6.7, in practice this algorithm does not perform very competitively on the log det problem.

Algorithm 6:
RANDOMIZED-SYMMETRIC-GREEDY

Input: submodular f, ground set $\mathcal{Y}$
$X_0 \leftarrow \emptyset, Y_0 \leftarrow \mathcal{Y}$
for $i = 1$ **to** N **do**
  $a_i \leftarrow f(X_{i-1} \cup \{i\}) - f(X_{i-1})$
  $b_i \leftarrow f(Y_{i-1} \setminus \{i\}) - f(Y_{i-1})$
  if $a_i \geq b_i$ **then**
    $X_i \leftarrow X_{i-1} \cup \{i\}, Y_i \leftarrow Y_{i-1}$
  else
    $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} \setminus \{i\}$
return X_n

Algorithm 7:
SYMMETRIC-GREEDY

Input: submodular f, ground set $\mathcal{Y}$
$X_0 \leftarrow \emptyset, Y_0 \leftarrow \mathcal{Y}$
for $i = 1$ **to** N **do**
  $a_i \leftarrow f(X_{i-1} \cup \{i\}) - f(X_{i-1})$
  $b_i \leftarrow f(Y_{i-1} \setminus \{i\}) - f(Y_{i-1})$
  $a'_i \leftarrow \max\{a_i, 0\}, b'_i \leftarrow \max\{b_i, 0\}$
  with probability $\frac{a'_i}{a'_i + b'_i}$ **do:**
    $X_i \leftarrow X_{i-1} \cup \{i\}, Y_i \leftarrow Y_{i-1}$
  else do:
    $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} \setminus \{i\}$
return X_n

Figure 5.2: Non-monotone submodular maximization algorithms from Buchbinder et al. (2012).

5.3.3 Constrained f

Buchbinder et al. (2012)'s algorithms provide a tight approximation guarantee for the unconstrained submodular maximization setting, where any $Y \subseteq \mathcal{Y}$ is an acceptable solution. Settings where additional constraints are placed on Y are also of interest though. In our discussion of monotone submodular functions we already mentioned the cardinality constraint $|Y| \leq k$, and Buchbinder et al. (2012)'s non-monotone algorithms have been extended to handle this type of constraint (Buchbinder, Feldman, Naor, and Schwartz, 2014). This adaptation relies on continuous optimization techniques, as do several other algorithms that handle more complex constraints.

As described at the beginning of this chapter, continuous techniques are those that represent a set Y by its characteristic vector $\boldsymbol{x} \in \{0,1\}^N$, with $x_i = \mathbb{1}(i \in Y)$, and then relax $\boldsymbol{x}$ such that its entries can be anywhere in the 0 to 1 range: $x_i \in [0,1]$. While some combinatorial approaches have been developed to handle knapsack and matroid constraints (Gupta, Roth, Schoenebeck, and Talwar, 2010; Gharan and Vondrák, 2011), the most flexible constraint-compatible algorithms rely on continuous techniques. The work of Chekuri et al. (2011) falls into this category, and the

algorithm we derive in this chapter builds on that work. Chekuri et al. (2011) describe methods for handling not only knapsack constraints and matroid constraints, but also more general polytope constraints.

5.4 Polytope constraints

Before describing our algorithm, we formally define the polytope constraints that Chekuri et al. (2011) can handle and give a few motivating practical examples of constrained problems. Most generally, a polytope is a closed shape defined by flat sides. We can formally define it in a recursive manner. A 0-polytope is a vertex. A 1-polytope is a line segment bounded by two 0-polytopes. A 2-polytope is a plane with 1-polytope boundaries. An N-polytope is an N-dimensional surface with $(N-1)$-polytope boundaries.

For submodular maximization, the relevant polytopes P are those that are N-dimensional and occupy some portion of the space $[0,1]^N$. For such polytopes, each dimension of a point $\boldsymbol{x} \in P$ is then interpretable as the probability x_i that $i \in Y$. This defines a clear link between a continuous function $F(\boldsymbol{x})$ and its discrete submodular counterpart $f(Y)$. For efficient submodular maximization algorithms, a simple restriction on this class of polytopes is necessary: solvability.

Definition 5.6. Solvable polytope: *A polytope $P \subseteq [0,1]^N$ is solvable if for any linear objective function $g(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{x}$, it is possible to efficiently find the best $\boldsymbol{x}$ in P:*

$$\boldsymbol{x} = \arg\max_{\boldsymbol{x}' \in P} g(\boldsymbol{x}') \,. \tag{5.15}$$

Vondrák (2008) proposes a "continuous greedy" algorithm that is a $(1 - \frac{1}{e})$-approximation for the problem $\max_{\boldsymbol{x} \in P} F(\boldsymbol{x})$, when f is monotone and P is solvable. Unfortunately, these guarantees do not extend to non-monotone f. In fact, an additional restriction on the polytopes is necessary for *any* guarantees in the non-monotone case; according to Vondrák (2009), no constant factor approximation is possible unless the polytope is down-monotone.

Definition 5.7. Down-monotone polytope: *A polytope $P \subseteq [0,1]^N$ is down-monotone if for all $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^N$, we have that $\boldsymbol{y} \leq \boldsymbol{x}$ and $\boldsymbol{x} \in P$ implies $\boldsymbol{y} \in P$. (Inequalities apply element-wise: $\boldsymbol{y} \leq \boldsymbol{x}$ implies $y_i \leq x_i$ for all i.)*

Fortunately, the class of solvable, down-monotone polytopes still includes a wide variety of useful constraint types, including knapsack polytopes and matroid polytopes. We now define these polytopes more formally.

Definition 5.8. Knapsack polytope: *A constraint of the form $\sum_{i=1}^{N} a_i x_i \leq b$ for non-negative values a_i, x_i, b is called a knapsack constraint. The convex hull of all the feasible solutions is called the knapsack polytope:* $\operatorname{conv}\left(\boldsymbol{x} \mid \sum_{i=1}^{N} a_i x_i \leq b\right)$.

Definition 5.9. Basis matroid polytope: *Let $\boldsymbol{e}_i \in \mathbb{R}^N$ denote the standard unit vector with a 1 in entry i and zeros elsewhere. For a set Y, let its associated incidence vector be $\boldsymbol{e}_Y = \sum_{i \in Y} \boldsymbol{e}_i$. Represent the bases $\boldsymbol{Z}$ of a matroid by their incidence vectors. Then the polytope $P_{\boldsymbol{Z}} = \operatorname{conv}(\boldsymbol{e}_Z \mid Z \in \boldsymbol{Z})$, is called the basis matroid polytope.*

In our experiments, we make use of one type of matroid in particular: the matching matroid. This matroid is very useful in practice for making comparisons. For example, Figure 5.3 illustrates how log det maximization subject to a matching matroid polytope constraint might be effective for an image comparison task.

Definition 5.10. Matching matroid: *Given a graph $G = (V, E)$, let $\mathcal{I}$ be the family of all $Y \subseteq E$ that can be covered by a matching. (A set of edges is a matching if no two edges share a vertex.) Then $(E, \mathcal{I})$ is a matroid. The bases of the matroid correspond to subsets of E that are maximum matchings in G.*

5.5 Softmax extension

Continuous submodular maximization methods (Chekuri et al., 2011; Feldman, Naor, and Schwartz, 2011) are based on the following submodularity definition.

Definition 5.11. Continuous submodularity: *A function $G : [0, 1]^N \to \mathbb{R}$ is submodular if, for all $\boldsymbol{x}, \boldsymbol{y} \in [0, 1]^N$:*

$$G(\boldsymbol{x} \vee \boldsymbol{y}) + G(\boldsymbol{x} \wedge \boldsymbol{y}) \leq G(\boldsymbol{x}) + G(\boldsymbol{y}) \,, \tag{5.16}$$

where $(\boldsymbol{x} \vee \boldsymbol{y})_i = \max\{x_i, y_i\}$ and $(\boldsymbol{x} \wedge \boldsymbol{y})_i = \min\{x_i, y_i\}$.

The standard continuous submodular function that maximization algorithms optimize is called the *multilinear* extension. This extension was first introduced by

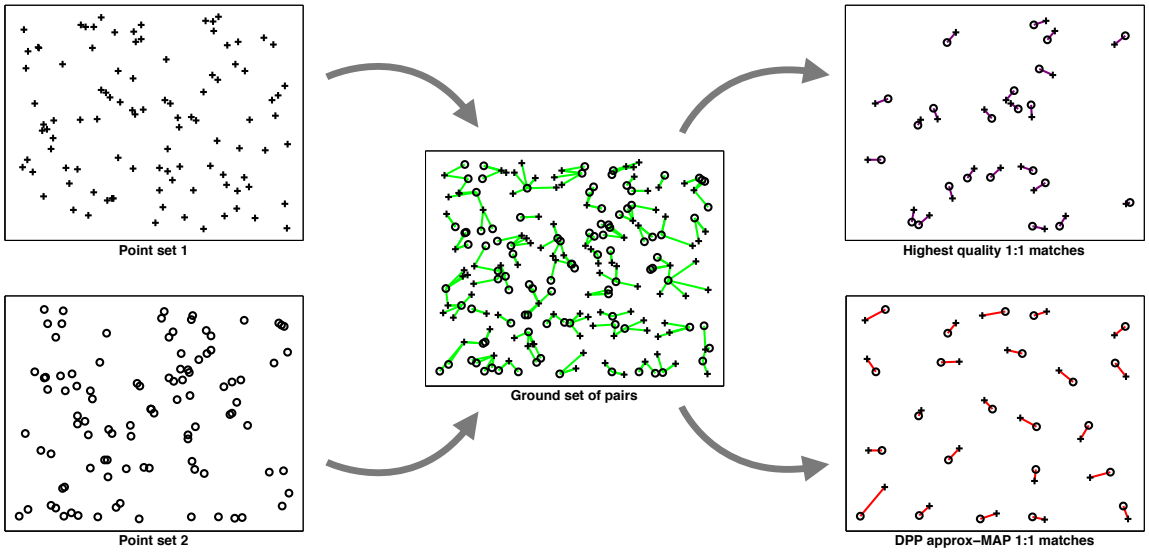


Figure 5.3: Consider the task of comparing two images, where we start by running a key point detector on each image. This is visualized on the left by two point sets. To match the key points in one image to those in the other, it is desirable to pick point pairs where the constituent points are as close to each other as possible. The center image here indicates with green lines some pairs that are fairly close (good quality matches). To improve the accuracy of our image comparison, we could impose the 1-to-1 matching constraint that says no key point in image 1 can map to more than one key point in image 2 and vice versa. If just the top-k highest quality pairs obeying this constraint are selected, the matching looks like that on the top right. If instead a DPP MAP approximation algorithm is run, then the result is more diverse, with only slightly longer edges. Whereas the highest-quality method may indicate two images are very similar even if there are large regions of disparity, the DPP MAP method, with its better coverage, is less likely to make this mistake.

Calinescu, Chekuri, Pál, and Vondrák (2007). For a vector $\boldsymbol{x} \in [0,1]^N$, consider a random subset $R(\boldsymbol{x})$ that contains element i of $\mathcal{Y}$ with probability x_i. The multilinear extension is the expected value of f on $R(\boldsymbol{x})$: $F(\boldsymbol{x}) = \mathbb{E}[f(R(\boldsymbol{x}))]$. More concretely, the multilinear extension of f is $F : [0,1]^N \to \mathbb{R}$ with value:

$$F(\boldsymbol{x}) = \sum_{Y : Y \subseteq \mathcal{Y}} f(Y) \prod_{i : i \in Y} x_i \prod_{i : i \notin Y} (1 - x_i) \,. \tag{5.17}$$

Notice that explicitly computing F according to this formula would require summing over 2^N sets $Y \subseteq \mathcal{Y}$. For some f though, such as graph cuts, F can be computed efficiently. For all other f, by randomly sampling sets Y according to the probabilities in $\boldsymbol{x}$, $F(\boldsymbol{x})$ can be estimated arbitrarily well; Calinescu et al. (2007) state that for any f, a polynomial number of samples yields a $(1 - 1/\text{poly}(N))$-approximation to $F(\boldsymbol{x})$. Nevertheless, the need to sample tends to make algorithms based on the multilinear extension much slower than greedy, combinatorial algorithms. For the function of interest in the case of DPPs, $f(Y) = \log\det(L_Y)$, sampling to estimate F makes for a much slower algorithm than GREEDY or SYMMETRIC-GREEDY.

With a small change to the multilinear extension though, we arrive at an objective $\tilde{F}$ that can be computed exactly and efficiently for $\log\det$. The resulting algorithm runs more quickly than GREEDY or SYMMETRIC-GREEDY for large N. Moreover, the modified objective retains critical properties of the multilinear extension, which allow us to build off of proofs in Chekuri et al. (2011), ultimately resulting in a constant-factor approximation guarantee for optimizing $\tilde{F}$. For the special case of $f(Y) = \log\det(L_Y)$, consider the following modification to the multilinear extension:

$$\tilde{F}(\boldsymbol{x}) = \log \sum_{Y : Y \subseteq \mathcal{Y}} \exp(f(Y)) \prod_{i : i \in Y} x_i \prod_{i : i \notin Y} (1 - x_i) \,. \tag{5.18}$$

We refer to this as the *softmax* extension.

Figure 5.4 provides a visual comparison of the objectives from Equations (5.17) and (5.18) for a toy example. Consider the case where the ground set consists of just the two items pictured on the top left in Figure 5.4. At the integer points, circled in the left image, the two objective functions agree with each other and their value is exactly the $\log\det$ of the set specified by $\boldsymbol{x}$. For example, notice that the value of the objectives at $\boldsymbol{x} = [1,0]$ is smaller than at $\boldsymbol{x} = [0,1]$ since vector B_1 has smaller magnitude than B_2.

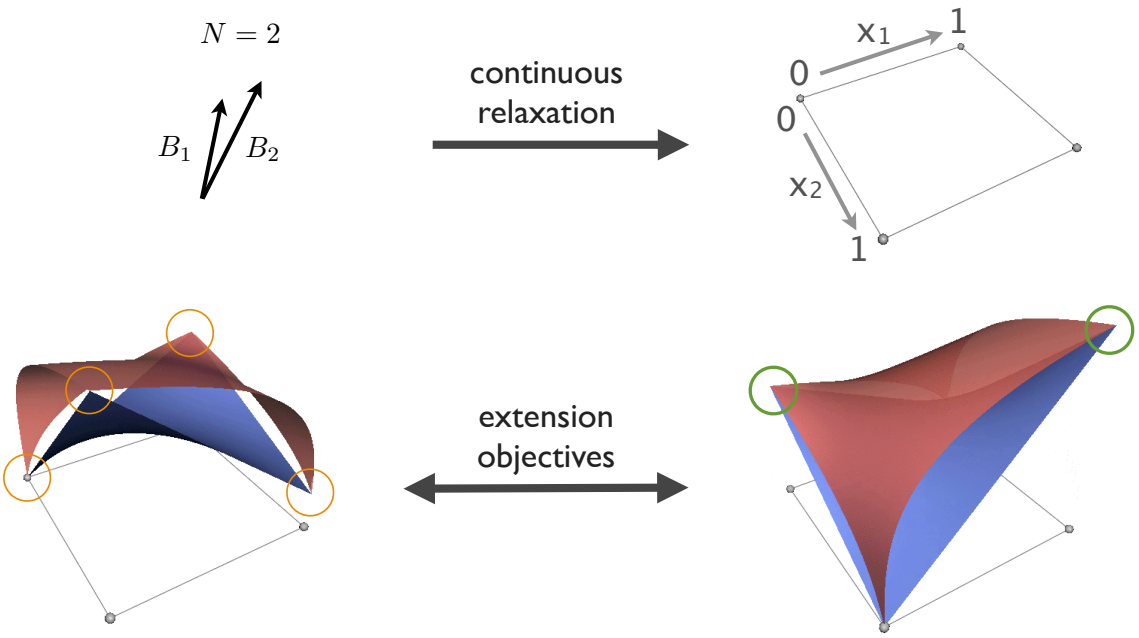


Figure 5.4: **Top left**: Ground set consisting of two vectors. **Top right**: Domain of the optimization variables; x_i = probability of including vector i. **Bottom left**: Softmax (top, red) and multilinear (bottom, blue) objectives plotted on the domain from the top right. Integer points are circled. **Bottom right**: Plot rotated $90°$ clockwise. The points $\boldsymbol{x} = [1, 0]$ and $\boldsymbol{x} = [0, 1]$ are circled.

In addition to the similarities observed in Figure 5.4, the softmax extension and multilinear extension are related in that they both involve a sum over exponentially many sets Y. Yet, we can show that the softmax extension is efficiently computable. Theorem 5.12 contains the details.

Theorem 5.12. *For a positive semidefinite matrix L and $\boldsymbol{x} \in [0,1]^N$, we have:*

$$\exp(\tilde{F}(\boldsymbol{x})) = \sum_{Y:Y\subseteq\mathcal{Y}} \det(L_Y) \prod_{i:i\in Y} x_i \prod_{i:i\notin Y} (1-x_i) \tag{5.19}$$

$$= \det(\operatorname{diag}(\boldsymbol{x})(L-I)+I)\,, \tag{5.20}$$

where $\operatorname{diag}(\boldsymbol{x})$ *indicates a diagonal matrix with entry* (i,i) *equal to* x_i.

Proof. Assume momentarily that $x_i < 1,\ \forall i$. Then:

$$\exp(\tilde{F}(\boldsymbol{x})) = \prod_{i=1}^{N}(1-x_i) \sum_{Y:Y\subseteq\mathcal{Y}} \det(L_Y) \prod_{i:i\in Y} \frac{x_i}{1-x_i} \tag{5.21}$$

$$= \prod_{i=1}^{N}(1-x_i) \sum_{Y:Y\subseteq\mathcal{Y}} \det\left(\left[\operatorname{diag}(\boldsymbol{x})\operatorname{diag}(1-\boldsymbol{x})^{-1}L\right]_Y\right) \tag{5.22}$$

$$= \prod_{i=1}^{N}(1-x_i) \det(\operatorname{diag}(\boldsymbol{x})\operatorname{diag}(1-\boldsymbol{x})^{-1}L + I) \tag{5.23}$$

$$= \det(\operatorname{diag}(\boldsymbol{x})L + \operatorname{diag}(1-\boldsymbol{x})) \tag{5.24}$$

$$= \det(\operatorname{diag}(\boldsymbol{x})(L-I)+I)\,. \tag{5.25}$$

The second and fourth equalities follow from the multilinearity of the determinant, and the third follows from DPP normalization, Theorem 2.1. Since Equation (5.25) is a polynomial in $\boldsymbol{x}$, by continuity the formula holds when some $x_i = 1$. □

Given that the softmax extension can be written as a single $N \times N$ determinant, it can be computed in $O(N^\omega)$ time. The derivative of a determinant can be found by applying rules from Petersen and Pedersen (2012). This results in an expression for the derivative of the softmax extension that can also be computed in $O(N^\omega)$ time. Corollary 5.13 provides the details.

Corollary 5.13. *For $\tilde{F}(\boldsymbol{x}) = \log\det(\operatorname{diag}(\boldsymbol{x})(L-I)+I)$, we have:*

$$\frac{\partial \tilde{F}(\boldsymbol{x})}{\partial x_i} = \operatorname{tr}((\operatorname{diag}(\boldsymbol{x})(L-I)+I)^{-1}(L-I)_i) \tag{5.26}$$

$$= \left[(\operatorname{diag}(\boldsymbol{x})(L-I)+I)^{-1}(L-I)\right]_{ii}\,, \tag{5.27}$$

where $(L - I)_i$ denotes the matrix obtained by zeroing all except the ith row of $L - I$.

5.5.1 Softmax maximization algorithms

Continuous submodular maximization algorithms for the multilinear extension typically follow the objective's gradient to find a local maximum. The algorithm we give here for optimizing the softmax extension behaves in an analogous manner. When the optimization polytope P is simple—for instance, the unit cube $[0, 1]^N$—methods such as L-BFGS (Nocedal and Wright, 2006, Section 9.1) can be employed to rapidly find a local maximum of the softmax extension. Alternatively, in situations where we are able to efficiently project onto the polytope P, we can apply projected gradient methods.

In the general case, however, we assume only that the polytope is solvable and down-monotone, as in Definitions 5.6 and 5.7. In such a setting, we rely on the conditional gradient algorithm, also known as the Frank-Wolfe algorithm (Bertsekas, 1999; Frank and Wolfe, 1956). Algorithm 8, COND-GRAD, describes the procedure. Intuitively, at each step the gradient gives a linear approximation for the softmax objective function at the current point $\boldsymbol{x}$. Since we assume the polytope P is solvable, we can find the point $\boldsymbol{y} \in P$ that maximizes this linear function. We then move from the current point $\boldsymbol{x}$ to a convex combination of $\boldsymbol{x}$ and $\boldsymbol{y}$. This ensures that we move in an increasing direction, while remaining in the polytope P, which is sufficient to efficiently find a local maximum of the softmax extension over P.

It remains to show that this local maximum, like the multilinear extension's local maxima, comes with approximation guarantees. Technically, to obtain these guarantees we must run COND-GRAD twice, once on the polytope P and once on a modification of P that depends on the solution found by the first run: $P \cap \{\boldsymbol{y} \mid \boldsymbol{y} \leq 1 - \boldsymbol{x}\}$. This second step is a bit reminiscent of the use of $f(\mathcal{Y})$ in Algorithm 7, SYMMETRIC-GREEDY, which can be thought of as optimizing the complement function $\bar{f}(Y) = f(\mathcal{Y} \setminus Y)$. In practice this second run of COND-GRAD can usually be omitted with minimal loss (if any). Algorithm 9, SOFTMAX-OPT, outlines the full softmax optimization procedure, which is the same as the one proposed by Chekuri et al. (2011) for the multilinear extension, F.

Algorithm 8: COND-GRAD

Input: function $\tilde{F}$, polytope P
$\boldsymbol{x} \leftarrow \mathbf{0}$
while not converged **do**
  $\boldsymbol{y} \leftarrow \arg\max_{\boldsymbol{y}' \in P} \nabla \tilde{F}(\boldsymbol{x})^\top \boldsymbol{y}'$
  $\alpha \leftarrow \arg\max_{\alpha' \in [0,1]} \tilde{F}(\alpha' \boldsymbol{x} + (1-\alpha')\boldsymbol{y})$
  $\boldsymbol{x} \leftarrow \alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}$
Output: $\boldsymbol{x}$

Algorithm 9: SOFTMAX-OPT

Input: kernel L, polytope P
Let $\tilde{F}(\boldsymbol{x}) = \log\det(\mathrm{diag}(\boldsymbol{x})(L-I)+I)$
$\boldsymbol{x} \leftarrow$ LOCAL-OPT$(\tilde{F}, P)$
$\boldsymbol{y} \leftarrow$ LOCAL-OPT$(\tilde{F}, P \cap \{\boldsymbol{y}' \mid \boldsymbol{y}' \leq \mathbf{1} - \boldsymbol{x}\})$

Output: $\begin{cases} \boldsymbol{x} & \text{if } \tilde{F}(\boldsymbol{x}) > \tilde{F}(\boldsymbol{y}) \\ \boldsymbol{y} & \text{otherwise} \end{cases}$

5.5.2 Softmax approximation bound

In this section, we show that SOFTMAX-OPT is a $\frac{1}{4}$-approximation for the problem of maximizing $\tilde{F}$ over the polytope $P = [0,1]^N$. We build on proofs from Chekuri et al. (2011). First, we show that, as is the case for the multilinear extension, the softmax extension exhibits partial concavity. Lemma 5.14 establishes that the second derivative of $\tilde{F}$ is negative in certain cases and Corollary 5.15 states the implied concavity condition.

Lemma 5.14. *For $\boldsymbol{u}, \boldsymbol{v} \geq \mathbf{0}$, we have:*

$$\frac{\partial^2}{\partial s \partial t} \tilde{F}(\boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v}) \leq 0 \tag{5.28}$$

wherever $\mathbf{0} < \boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v} < \mathbf{1}$.

Proof. We begin by rewriting $\tilde{F}$ in a symmetric form:

$$\begin{aligned}
\tilde{F}(\boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v}) &= \log\det(\mathrm{diag}(\boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v})(L-I)+I) && (5.29)\\
&= \log\det(\mathrm{diag}(\boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v})) + && (5.30)\\
&\quad \log\det(L - I + \mathrm{diag}(\boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v})^{-1}) && (5.31)\\
&= \log\det(D) + \log\det(M)\,, && (5.32)
\end{aligned}$$

where $D = \mathrm{diag}(\boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v})$ and $M = L - I + D^{-1}$. Note that $D, M \succ 0$, since $\mathbf{0} < \boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v} < \mathbf{1}$. Then, applying standard matrix derivative rules, we have:

$$\frac{\partial}{\partial t} \tilde{F}(\boldsymbol{x} + s\boldsymbol{u} + t\boldsymbol{v}) = \mathrm{tr}(D^{-1}\mathrm{diag}(\boldsymbol{v}) - M^{-1}D^{-2}\mathrm{diag}(\boldsymbol{v}))\,. \tag{5.33}$$

Taking the second derivative with respect to s:

$$\frac{\partial^2}{\partial s \partial t}\tilde{F}(\boldsymbol{x}+s\boldsymbol{u}+t\boldsymbol{v}) = \text{tr}(-D^{-2}\text{diag}(\boldsymbol{v})\text{diag}(\boldsymbol{u}) + 2M^{-1}D^{-3}\text{diag}(\boldsymbol{v})\text{diag}(\boldsymbol{u}) - M^{-1}D^{-2}\text{diag}(\boldsymbol{u})M^{-1}D^{-2}\text{diag}(\boldsymbol{v}))\,. \tag{5.34}$$

Since diagonal matrices commute and $\text{tr}(AB) = \text{tr}(BA)$, the above is equal to $-\text{tr}(SS^\top) \leq 0$, where:

$$S = D^{-1}\text{diag}(\sqrt{\boldsymbol{v}})\text{diag}(\sqrt{\boldsymbol{u}}) - D^{-1}\text{diag}(\sqrt{\boldsymbol{v}})M^{-1}\text{diag}(\sqrt{\boldsymbol{u}})D^{-1}\,. \tag{5.35}$$

Note that S is well-defined, since $\boldsymbol{u}, \boldsymbol{v} \geq \boldsymbol{0}$. The matrix $SS^\top$ has squared entries on its diagonal, hence we are guaranteed that its trace is non-negative. □

Given this proof of negative Hessian entries, we immediately have the following corollary. The inequality here applies elementwise: $v_i \geq 0\ \forall i$. To better visualize the meaning of this corollary, Figure 5.5 shows cross-sections of the softmax and multilinear extensions in an all-positive direction and a non-all-positive direction. Only in the former do we observe concavity.

Corollary 5.15. *$\tilde{F}(\boldsymbol{x}+t\boldsymbol{v})$ is concave along any direction $\boldsymbol{v} \geq \boldsymbol{0}$ (equivalently, $\boldsymbol{v} \leq \boldsymbol{0}$).*

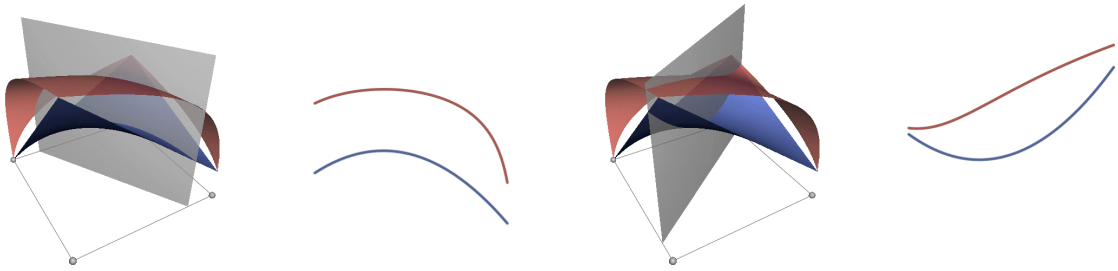


Figure 5.5: The softmax (top, red) and multilinear (bottom, blue) objectives. **Left**: Example all-positive, concave cross-section of the objectives. **Right**: Example non-all-positive, non-concave cross-section of the objectives.

Corollary 5.15 tells us that a local optimum $\boldsymbol{x}$ of $\tilde{F}$ has certain global properties—namely, that $\tilde{F}(\boldsymbol{x}) \geq \tilde{F}(\boldsymbol{y})$ whenever $\boldsymbol{y} \leq \boldsymbol{x}$ or $\boldsymbol{y} \geq \boldsymbol{x}$. Applying this fact, we can derive a lower-bound on $\tilde{F}(\boldsymbol{x})$, just as Chekuri et al. (2011) did for $F(\boldsymbol{x})$. Lemma 5.16 gives the details.

Lemma 5.16. *If $\boldsymbol{x}$ is a local optimum of $\tilde{F}$, then for any $\boldsymbol{y} \in [0,1]^N$:*

$$2\tilde{F}(\boldsymbol{x}) \geq \tilde{F}(\boldsymbol{x} \vee \boldsymbol{y}) + \tilde{F}(\boldsymbol{x} \wedge \boldsymbol{y})\,, \tag{5.36}$$

where $(\boldsymbol{x} \vee \boldsymbol{y})_i = \max(x_i, y_i)$ and $(\boldsymbol{x} \wedge \boldsymbol{y})_i = \min(x_i, y_i)$.

Proof. By definition, $\boldsymbol{x} \vee \boldsymbol{y} - \boldsymbol{x} \geq 0$ and $\boldsymbol{x} \wedge \boldsymbol{y} - \boldsymbol{x} \leq 0$. By Corollary 5.15 and the first order definition of concavity:

$$\nabla\tilde{F}(\boldsymbol{x})^\top(\boldsymbol{x} \vee \boldsymbol{y} - \boldsymbol{x}) \geq \tilde{F}(\boldsymbol{x} \vee \boldsymbol{y}) - \tilde{F}(\boldsymbol{x}) \tag{5.37}$$

$$\nabla\tilde{F}(\boldsymbol{x})^\top(\boldsymbol{x} \wedge \boldsymbol{y} - \boldsymbol{x}) \geq \tilde{F}(\boldsymbol{x} \wedge \boldsymbol{y}) - \tilde{F}(\boldsymbol{x})\,. \tag{5.38}$$

Adding the two equations gives the desired result, given that, at a local optimum $\boldsymbol{x}$, we know $\nabla\tilde{F}(\boldsymbol{x})^\top(\boldsymbol{z} - \boldsymbol{x}) \leq \mathbf{0}$ for any $\boldsymbol{z} \in P$ that is all-negative or all-positive. □

Continuing to follow along the same proof path as Chekuri et al. (2011), we now define a surrogate function $\tilde{F}^*$. Let $\mathcal{X}_i \subseteq [0,1]$ be a subset of the unit interval representing $x_i = |\mathcal{X}_i|$, where $|\mathcal{X}_i|$ denotes the measure of $\mathcal{X}_i$. (Note that this representation is overcomplete, since there are in general many subsets of $[0,1]$ with measure x_i.) Then $\tilde{F}^*$ is defined on $\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N)$ by:

$$\tilde{F}^*(\mathcal{X}) = \tilde{F}(\boldsymbol{x}), \quad \boldsymbol{x} = (|\mathcal{X}_1|, |\mathcal{X}_2|, \ldots, |\mathcal{X}_N|)\,. \tag{5.39}$$

Lemma 5.17. *$\tilde{F}^*$ is submodular.*

Proof. We first show that for $\boldsymbol{x} \leq \boldsymbol{y}$ and $\boldsymbol{a} \geq \mathbf{0}$, we have:

$$\tilde{F}(\boldsymbol{x} + \boldsymbol{a}) - \tilde{F}(\boldsymbol{x}) \geq \tilde{F}(\boldsymbol{y} + \boldsymbol{a}) - \tilde{F}(\boldsymbol{y})\,. \tag{5.40}$$

By the fundamental theorem of calculus:

$$\tilde{F}(\boldsymbol{x} + \boldsymbol{a}) - \tilde{F}(\boldsymbol{x}) = \int_0^1 \frac{\partial}{\partial t}\tilde{F}(\boldsymbol{x} + t\boldsymbol{a})\; dt\,, \tag{5.41}$$

and by a second application:

$$\begin{aligned}(\tilde{F}(\boldsymbol{y} + \boldsymbol{a}) - \tilde{F}(\boldsymbol{y})) - (\tilde{F}(\boldsymbol{x} + \boldsymbol{a}) - \tilde{F}(\boldsymbol{x})) = \\ \int_0^1 \int_0^1 \frac{\partial^2}{\partial s \partial t}\tilde{F}(\boldsymbol{x} + s(\boldsymbol{y} - \boldsymbol{x}) + t\boldsymbol{a})\; dt\; ds\,.\end{aligned} \tag{5.42}$$

Since $\boldsymbol{y} - \boldsymbol{x} \geq \mathbf{0}$, Corollary 5.15 allows us to conclude that the second derivatives are non-positive. Now, for $\mathcal{X} \subseteq \mathcal{Y}$ and $\mathcal{A} \cap \mathcal{Y} = \emptyset$ where $\mathcal{X}, \mathcal{Y}, \mathcal{A}$ represent $\boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{a}$, respectively, we have:

$$\begin{aligned} \tilde{F}^*(\mathcal{X} \cup \mathcal{A}) - \tilde{F}^*(\mathcal{X}) &= \tilde{F}(\boldsymbol{x} + \boldsymbol{a}) - \tilde{F}(\boldsymbol{x}) && (5.43) \\ &\geq \tilde{F}(\boldsymbol{y} + \boldsymbol{a}) - \tilde{F}(\boldsymbol{y}) && (5.44) \\ &= \tilde{F}^*(\mathcal{Y} \cup \mathcal{A}) - \tilde{F}^*(\mathcal{Y})\,. && (5.45) \end{aligned}$$

□

Lemmas 5.16 and 5.17 suffice to prove the following theorem, which appears for the multilinear extension in Chekuri et al. (2011), and here bounds the approximation ratio of Algorithm 9, SOFTMAX-OPT. We omit the proof since it is unchanged from Chekuri et al. (2011).

Theorem 5.18. *Let $\tilde{F}(\boldsymbol{x})$ be the softmax extension of a non-negative submodular function $f(Y) = \log\det(L_Y)$, let $\mathrm{OPT} = \max_{\boldsymbol{x}' \in P} \tilde{F}(\boldsymbol{x}')$, and let $\boldsymbol{x}$ and $\boldsymbol{y}$ be local optima of $\tilde{F}$ in the polytope P and the polytope $P \cap \{\boldsymbol{y}' \mid \boldsymbol{y}' \leq \mathbf{1} - \boldsymbol{x}\}$, respectively. Then:*

$$\max\left\{\tilde{F}(\boldsymbol{x}), \tilde{F}(\boldsymbol{y})\right\} \geq \frac{1}{4}\mathrm{OPT} \geq \frac{1}{4}\max_{Y \in P} \log\det(L_Y)\,. \tag{5.46}$$

Note that the softmax extension is an upper bound on the multilinear extension, and thus Equation (5.46) is at least as tight as the corresponding result in Chekuri et al. (2011). We state the following corollary to address the possible existence of negative f values.

Corollary 5.19. *Algorithm 9 yields a $\frac{1}{4}$-approximation to the DPP MAP problem whenever $\log\det(L_Y) \geq 0$ for all Y. In general, the objective value obtained by Algorithm 9 (SOFTMAX-OPT) is bounded below by $\frac{1}{4}\mathrm{OPT} - \frac{3}{4}\eta$, where $\eta = -\min_Y \log\det(L_Y)$.*

In practice, filtering of near-duplicates and especially low-quality items can be used to keep η from getting too large; however, in our empirical tests η did not seem to have a significant effect on approximation quality.

5.5.3 ROUNDING

To convert a $\frac{1}{4}$-approximation with respect to $\tilde{F}$ into an approximation guarantee for its discrete counterpart f, the continuous variable that is output by SOFTMAX-OPT, $\boldsymbol{x} \in [0,1]^N$, must be converted into a set $Y \subseteq \mathcal{Y}$. When the polytope P is

unconstrained, as in $P = [0,1]^N$, it is easy to show that the results of Algorithm 9 are integral (or can be rounded without loss). Theorem 5.20 addresses this case.

Theorem 5.20. *If $P = [0,1]^N$, then for any local optimum $\boldsymbol{x}$ of $\tilde{F}$, either $\boldsymbol{x}$ is integral or at least one fractional coordinate x_i can be set to 0 or 1 without lowering the objective.*

Proof. By multilinearity of the determinant, the expression:

$$\tilde{F}(\boldsymbol{x}) = \log\det(\operatorname{diag}(\boldsymbol{x})(L - I) + I) \tag{5.47}$$

is linear in each coordinate x_i if all the other coordinates $\boldsymbol{x}_{-i}$ are held fixed. That is:

$$\tilde{F}(x_i, \boldsymbol{x}_{-i}) = \log(a_i x_i + b_i)\,, \tag{5.48}$$

where a_i and b_i are constants. Suppose that coordinate i is fractional ($0 < x_i < 1$) at a local optimum. Then the gradient with respect to x_i must be zero, since the polytope constraint is not active. This gradient is:

$$\frac{\partial \tilde{F}(\boldsymbol{x})}{\partial x_i} = \frac{a_i}{a_i x_i + b_i}\,, \tag{5.49}$$

which can only be zero if $a_i = 0$. Hence, setting x_i to 0 or 1 does not affect the objective value. □

More generally, the polytope P can be a complex geometric object, and in this case we are not guaranteed that SOFTMAX-OPT will return an integer solution. Thus, its output needs to be rounded. For the multilinear extension, in the simple case where P corresponds to the discrete cardinality constraint $|Y| \leq k$, rounding can be done without loss in expected f-value (Calinescu, Chekuri, Pál, and Vondrák, 2011, Section 2.4) by using pipage rounding techniques (Ageev and Sviridenko, 2004).

For general polytopes, more complex mechanisms are necessary. To this end, Chekuri et al. (2011) contribute a class of rounding algorithms that they call "contention resolution schemes" (CR schemes). Ultimately, given a continuous solution $\boldsymbol{x} \in P$, these schemes start from a random subset $R(\boldsymbol{x})$ that contains element i of $\mathcal{Y}$ with probability x_i. Then, elements are removed from $R(\boldsymbol{x})$ until it is feasible according to the original constraints on S. The key is to design the removal algorithm in such a way that we are guaranteed that element i appears in the final set

with probability at least cx_i, for some sufficiently large $c > 0$. Chekuri et al. (2011) obtain an optimal CR scheme for the case of a single matroid constraint, and also provide CR schemes for knapsack constraints and combinations of such constraints.

Unfortunately, none of these rounding results, neither the pipage rounding result for cardinality constraints nor the CR schemes for more complex constraints, translate directly from the multilinear extension to the softmax. Pipage rounding may result in loss of expected f-value when applied to the softmax objective, as softmax does not satisfy the ϵ-concavity condition that is necessary for lossless pipage rounding. In fact we can give a trivial example that shows no constant-factor guarantee is possible for rounding softmax in the cardinality-constrained setting.

Example 5.21. *Suppose we have the constraint $|Y| \leq 1$, which translates to the polytope constraint $\sum_{i=1}^{N} x_i \leq 1$. Let L be an $N \times N$ diagonal matrix with all diagonal entries having value $\alpha \geq N^N$. Then the value $\boldsymbol{x}$ such that $x_i = \frac{1}{N}$ for all i is in the constraint polytope, and has softmax value:*

$$\tilde{F}(\boldsymbol{x}) = \log \sum_{Y:Y\subseteq\mathcal{Y}} \frac{1}{N^N} \det(L_Y) \tag{5.50}$$

$$\geq \log \sum_{Y:Y=\mathcal{Y}} \frac{1}{N^N} \det(L_Y) \tag{5.51}$$

$$= \log\left(\frac{1}{N^N}\alpha^N\right) \tag{5.52}$$

$$\geq \log(\alpha^{N-1})\,. \tag{5.53}$$

In contrast, given the $|Y| \leq 1$ constraint, the maximum f-value is $\log(\alpha)$. Thus, the best approximation factor any rounding algorithm could achieve is $\frac{1}{N-1}$.

Perhaps there is some characterization of pathological cases, such as the one from this example, such that if we restrict the space of kernels L to avoid them, then it is possible to find constant-factor rounding guarantees. However, additional similarities between the multilinear and the softmax would have to be derived to prove this. Thus, we defer the development of such rounding schemes to future work. In our experiments, we apply pipage rounding and threshold rounding (rounding all coordinates up or down using a single threshold), and these seem to work well in practice, despite their lack of formal guarantees.

5.6 Experiments

To illustrate the softmax optimization method, we compare it to GREEDY (Algorithm 5), and to SYMMETRIC-GREEDY (Algorithm 7). Recall that the former has no guarantees for non-monotone f such as log det, while the latter is a $\frac{1}{3}$-approximation in the unconstrained setting. We also test with RANDOMIZED-SYMMETRIC-GREEDY, which has a $\frac{1}{2}$ approximation guarantee, but its results are not as good as those of its deterministic counterpart, so we do not bother to report them here. (Its results might improve if we ran it many times instead of just once; but in that case, for a fair comparison, we should also allow the softmax algorithm many random restarts.)

While a naïve implementation of the arg max in line 4 of GREEDY requires evaluating the objective for each item in U, we exploit the fact that DPPs are closed under conditioning to compute all necessary values with only two matrix inversions. We report baseline runtimes using this optimized greedy algorithm, which is about 10 times faster than the naïve version at $N = 200$. The code and data for all experiments is publicly available here: http://www.seas.upenn.edu/~jengi/dpp-map.html.

5.6.1 Synthetic data

As a first test, we seek the MAP set for DPPs whose kernels are drawn randomly from a Wishart distribution. Specifically, we choose $L = B^\top B$, where $B \in \mathbb{R}^{N \times N}$ has entries drawn independently from the standard normal distribution, $b_{ij} \sim \mathcal{N}(0, 1)$. This results in $L \sim \mathcal{W}_N(N, I)$, a Wishart distribution with N degrees of freedom and an identity covariance matrix. This distribution has several desirable properties: (1) in terms of eigenvectors, it spreads its mass uniformly over all unitary matrices (James, 1964), and (2) the probability density of eigenvalues $\lambda_1, \ldots, \lambda_N$ is:

$$\exp\left(-\sum_{i=1}^{N} \lambda_i\right) \prod_{i=1}^{N} \frac{\prod_{j=i+1}^{N} (\lambda_i - \lambda_j)^2}{((N-i)!)^2}, \tag{5.54}$$

the first term of which deters the eigenvalues from being too large, and the second term of which encourages the eigenvalues to be well-separated. Property (1) implies that we will see a variety of eigenvectors, which play an important role in the structure of a DPP. Property (2) implies that interactions between these eigenvectors will be

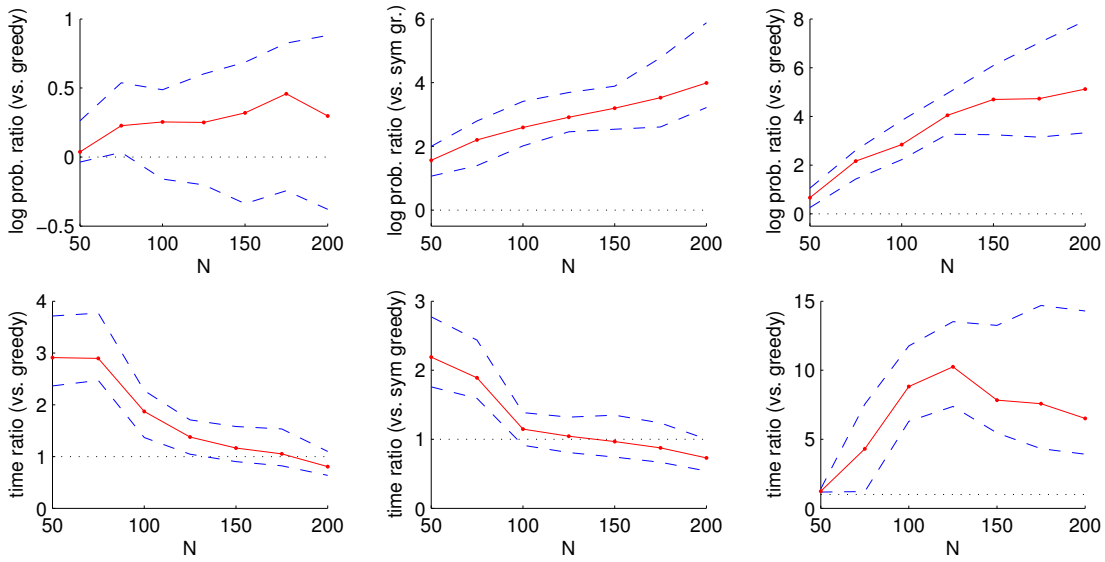


Figure 5.6: Median and quartile log probability ratios (top) and running time ratios (bottom) for 100 random trials. SOFTMAX-OPT versus: (left) GREEDY on unconstrained problems, (center) SYMMETRIC-GREEDY on unconstrained problems, (right) GREEDY on constrained problems. Dotted black lines indicate equal performance.

important, as no one eigenvalue is likely to dominate. Combined, these properties suggest that samples should encompass a wide range of DPPs.

For these tests, we let N vary in the range $[50, 200]$, since most prior work with (non-structured) DPPs in real-world scenarios has typically operated in this range. Figure 5.6 (top left and center) shows performance results on these random kernels in the unconstrained setting. SOFTMAX-OPT generally outperforms GREEDY, and the performance gap tends to grow with the size of the ground set, N. Moreover, Figure 5.6 (bottom left) illustrates that our method is of comparable efficiency at medium N, and becomes more efficient as N grows. Despite the fact that the symmetric greedy algorithm (Buchbinder et al., 2012) has an improved approximation guarantee of $\frac{1}{3}$, essentially the same analysis applies to it as well (Figure 5.6, center).

For the constrained setting, we test a matching matroid constraint, as described in Definition 5.10. To this end, we first generate two separate random matrices $B^{(1)}$ and $B^{(2)}$, then select random pairs of columns $(B_i^{(1)}, B_j^{(2)})$. Averaging $(B_i^{(1)} + B_j^{(2)})/2$ creates one column of the final B matrix that we use to construct the DPP kernel: $L = B^\top B$. Each item in $\mathcal{Y}$ is thus associated with a pair (i, j). We impose the constraint that if the item corresponding to the (i, j) pair is selected, then no other

item with first element i or second element j can be included; i.e. the pairs cannot overlap. Since exact duplicate pairs produce identical rows in L, they are never both selected and can be pruned ahead of time. This means that our constraints are of a graphical form that allows us to apply pipage rounding to the possibly fractional result (though with no guarantees that the rounded solution will be approximately optimal). The constrained variant of the GREEDY algorithm, shown in Algorithm 10, simply has an additional step each iteration to prune from its "unused" set U the items disallowed by the constraints.

Figure 5.6 (right) summarizes the performance of our algorithm in a constrained setting. Interestingly, we see even greater gains over greedy in this setting. Enforcing the constraints precludes using fast methods like L-BFGS though, so our optimization procedure is in this case somewhat slower than greedy.

5.6.2 Political candidate comparison

Finally, we demonstrate our approach using real-world data. Consider a variant of the document summarization task where we wish to summarize two bodies of work in a manner that compares them. For instance, we might want to compare the opinions of various authors on a range of topics—or even to compare the statements made at different points in time by the same author, e.g. a politician believed to have changed positions on various issues. For this task, we might first create a pool of pairs of sentences $\mathcal{Y}$, where one item in each pair comes from each body of work. We could then select a subset of these pairs $Y \subseteq \mathcal{Y}$ such that they cover the two individual sources well (are diverse), but also provide a clean comparison (sentences within a single selected pair are highly related). This problem can easily be cast as log det maximization, subject to a matching matroid constraint.

In this vein, we extract all the statements made by the eight main contenders during the 2012 U.S. Republican primary debates: Bachmann, Cain, Gingrich, Huntsman, Paul, Perry, Romney, and Santorum. Each pair of candidates (a, b) constitutes one instance of our task. The task output is a set of statement pairs where the first statement in each pair comes from candidate a and the second from candidate b. The goal of optimization is to find a set that is diverse (contains many topics, such as healthcare, foreign policy, immigration, etc.) but where both statements in each

Algorithm 10: CONSTRAINED-GREEDY

Input: kernel L, polytope P
$Y \leftarrow \emptyset$, $U \leftarrow \mathcal{Y}$
while U is not empty **do**
  $i^* \leftarrow \arg\max_{i \in U} \log\det(L_{Y \cup \{i\}})$
  if $\log\det(L_{Y \cup \{i^*\}}) < \log\det(L_Y)$ **then**
    break
  $Y \leftarrow Y \cup \{i^*\}$
  $U \leftarrow \{i \mid i \notin Y, \boldsymbol{e}_{Y \cup \{i\}} \in P\}$
Output: Y

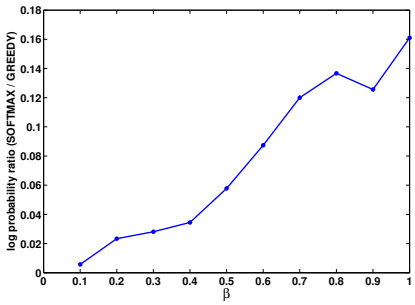


Figure 5.7: Plot of the log ratio of SOFTMAX-OPT's det value to CONSTRAINED-GREEDY's det value for ten settings of match weight β.

pair are topically similar.

Before formulating a DPP objective for this task, we do some pre-processing. We filter short statements, leaving us with an average of 179 quotes per candidate (min = 93, max = 332). We parse the quotes, keeping only nouns, and further filter nouns by document frequency, keeping only those that occur in at least 10% of the quotes. Then we generate a feature matrix Φ where Φ_{ts} is the number of times term t appears in quote s. This matrix is normalized so that $\|\Phi_s\|_2 = 1$, where Φ_s is the sth column.

For a given pair of candidates (a, b) we compute the quality of each possible quote pair $(s_i^{(a)}, s_j^{(b)})$ as the dot product of their columns in Φ. This is essentially a cosine similarity score, like that in Equation (4.25). We use $\mathbf{r}$ to denote the resulting quality scores. While the model will naturally ignore low-quality pairs, for efficiency we throw away such pairs in pre-processing. For each of candidate a's quotes $s_i^{(a)}$ we keep a pair with quote:

$$j = \arg\max_{j'} \; r(s_i^{(a)}, s_{j'}^{(b)}) \tag{5.55}$$

from candidate b, and vice-versa. The quality scores of the unpruned quotes are re-normalized to span the $[0, 1]$ range. To create a similarity feature vector describing each pair, we simply add the corresponding pair of single-quote feature vectors and re-normalize, forming a new Φ matrix.

Our task is to select some high-quality representative subset of the unpruned quote pairs. We formulate this as a DPP objective by creating a kernel from the quality and similarity features in the manner described by Equation (2.72). More

precisely, the kernel we use is $L = QSQ$, where S_{ij} is a measurement of similarity between quote pairs i and j, and Q is a diagonal matrix with Q_{ii} representing the match quality of pair i. We set $S = \Phi^\top \Phi$ and $\text{diag}(Q) = \sqrt{\exp(\beta \mathbf{r})}$, where β is a hyperparameter. Larger β values place more emphasis on picking high-quality pairs than on making the overall set diverse.

To help limit the number of pairs selected when optimizing the objective, we add some constraints. While we could impose the simple constraint that no selected pairs can have a shared quotation, here we impose slightly more restrictive constraints by first applying quote clustering. For each candidate we cluster their quotes using k-means on the word feature vectors and impose the constraint that no more than one quote per cluster can be selected. After running SOFTMAX-OPT with the polytope version of these constraints, we round the final solution using the threshold rounding scheme described in Section 5.5.3.

Figure 5.7 shows the result of optimizing this constrained objective, averaged over all 56 candidate pairs. For all settings of β we outperform greedy. In general, we observe that our algorithm is most improved compared to greedy when the constraints are in play. When β is small the constraints are less relevant, since the model has an intrinsic preference for smaller sets. On the other hand, when β is very large the algorithms must choose as many pairs as possible in order to maximize their score; in this case the constraints play a more important role.

5.7 Model combination

In addition to theoretical guarantees and the empirical advantages we demonstrate in Section 5.6, the proposed approach to the DPP MAP problem offers a great deal of flexibility. Since the general framework of continuous optimization is widely used in machine learning, this technique is a step towards making DPPs easy to combine with other models. For instance, if P is the local polytope for a Markov random field, then, augmenting the softmax objective with the (linear) log-likelihood of the MRF—additive linear objective terms do not affect the lemmas proved above—we can approximately compute the MAP configuration of the DPP-MRF product model. We might in this way model diverse objects placed in a sequence, or fit to an underlying signal like an image. Studies of these possibilities are left to future work.

6

Likelihood maximization

A DPP is compactly parameterized by a PSD kernel matrix L, or, equivalently, the corresponding marginal kernel K. Thus, with just $\frac{N(N+1)}{2}$ parameters (assuming the kernel is symmetric), a DPP assigns scores to 2^N sets. The attractive computational properties that DPPs offer—exact and efficient normalization, marginalization, conditioning, and sampling—arise in part from this compactness of the parameterization. However, even this seemingly simple parameterization poses a challenge for other inference tasks. We saw an example of this in the previous chapter, where we discussed the NP-hard problem of finding a DPP's mode. There, we assumed knowledge of a kernel L that was a good fit for the task at hand. In practice, finding such an L can also be difficult.

To fit a DPP to a given task, we would like to learn the entries of its kernel matrix by maximizing the log-likelihood of available data. This is non-trivial, as it involves solving a non-convex optimization problem. Even in more restricted settings the learning task is likely NP-hard. For example, often we have access to many features that we know are relevant for a task, and are only unsure of the relative importance of these features. As mentioned in Section 2.2.6, finding likelihood-maximizing weights even for a simple additive combination of these features is conjectured to be

NP-hard.

In this chapter we first briefly consider alternative methods of learning a DPP kernel using non-likelihood based methods. These bear a resemblance to approaches for learning distance metrics or kernels for support vector machines (SVMs). Then, we describe two very restrictive parameterizations for which the log-likelihood function is in fact concave. The first of these involves learning only a single weight for each row of the kernel matrix (Kulesza and Taskar, 2011b). This amounts to learning what makes an item high-quality, but does not address the issue of what makes two items similar. The second of these restrictive parameterizations involves learning weights for a linear combination of DPPs with fixed kernel matrices (Kulesza and Taskar, 2011a). We follow the description of these concave optimization problems with a more complete characterization of what makes certain less restrictive parameterizations a challenge.

The remainder of the chapter focuses on learning in these less restrictive settings. First, with parametric kernels such as Gaussians (Affandi, Fox, Adams, and Taskar, 2014), then with non-parametric kernels—kernels where the entries can be arbitrary as long as the overall matrix is PSD. In particular, for this last setting we describe an expectation-maximization (EM) algorithm for optimizing a lower bound on log-likelihood. Testing this method on a real-world product recommendation task, we observe relative gains of up to 16.5% in test log-likelihood compared to the naive approach of maximizing likelihood by projected gradient ascent. The majority of the information that this chapter conveys regarding the EM algorithm and the product recommendation experiments can also be found in Gillenwater et al. (2014).

6.1 Alternatives to maximizing likelihood

Before delving into likelihood maximization techniques, we briefly suggest here several non-likelihood based alternatives for DPP learning, and explain why these methods are not quite as promising.

On the surface, the problem of learning a DPP kernel is similar to the problem of learning an SVM kernel or learning a distance metric. In all of these cases we have similar PSD constraints, and wish to set L_{ij} so as to capture the similarity of two elements (or two features). Upon deeper reflection though, significant differences

become apparent. Most importantly, basic SVM kernel learning and distance metric learning can be cast as convex optimization problems.

For SVMs, in the binary classification setting we have labels $y \in \{-1, 1\}$. Stacking these into a column vector $\boldsymbol{y}$ produces have a target kernel: $\hat{L} = \boldsymbol{y}\boldsymbol{y}^\top$. This kernel has $\hat{L}_{ij} = 1$ when $y_i = y_j$, and $\hat{L}_{ij} = -1$ when $y_i \neq y_j$. One basic kernel learning method then is to maximize the alignment between L and $\hat{L}$:

$$\max_{L:L\succeq 0} \frac{\mathrm{tr}(L^\top \hat{L})}{\|L\|_F \|\hat{L}\|_F}\,, \tag{6.1}$$

where the F subscript indicates the Frobenius norm. This can be cast as a semi-definite programming problem (Lanckriet, Cristianini, Bartlett, Ghaoui, and Jordan, 2004, Section 4.7).

For distance metric learning, the setup is often similar. The goal is to learn a kernel L that defines a distance metric on a given feature space by assigning weights to features and feature pairs. In the simplest case, judgments of the form "i and j are similar" or "i and j are dissimilar" are given as input. Let $\mathcal{S}$ be the set of similar item pairs, and let $\mathcal{D}$ be the set of dissimilar item pairs. Then the learning task can be formulated as the following convex optimization problem:

$$\min_{L:L\succeq 0} \sum_{(i,j)\in\mathcal{S}} \|B_i - B_j\|_L^2 \qquad \text{s.t.} \sum_{(i,j)\in\mathcal{D}} \|B_i - B_j\|_L^2 \geq 1\,, \tag{6.2}$$

where B_i is the feature vector associated with the ith item, and $\|A\|_L^2 = A^\top L A$ (Xing, Ng, Jordan, and Russell, 2002).

Suppose we try to mimic these SVM kernel and distance metric learning methods for DPP kernel learning. Let our training data be of the form $Y_1, \ldots, Y_T$ such that $Y_t \subseteq \mathcal{Y}$. Then we can try to construct a target matrix $\hat{L}$ from these sets. For DPPs, it is actually more natural to do this for the marginal kernel K, since data marginals are easy to compute. The following sums yield the data marginals:

$$m_i = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(i \in Y_t) \tag{6.3}$$

$$m_{ij} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(\{i,j\} \subseteq Y_t) \tag{6.4}$$

$$m_{ijk} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(\{i,j,k\} \subseteq Y_t)\,. \tag{6.5}$$

The first of these translates directly into the diagonal of a target marginal kernel; recalling from Equation (2.13) that diagonal elements of K are equal to size-1 marginals, our diagonal target is $K_{ii} = m_i$. Similarly, the size-2 marginals are equal to $\det(K_{\{i,j\}})$, which we can use to solve for the *magnitude* of the off-diagonal elements:

$$m_{ij} = \det(K_{\{i,j\}}) = K_{ii}K_{jj} - K_{ij}^2 \quad \Longrightarrow \quad |K_{ij}| = \sqrt{K_{ii}K_{jj} - m_{ij}}\,. \tag{6.6}$$

The size-3 marginals provide the *signs* of these off-diagonal elements. Writing out one of these determinants we have:

$$m_{ijk} = \det(K_{\{i,j,k\}}) = K_{ii}K_{jj}K_{kk} + 2K_{ij}K_{jk}K_{ik} - K_{ii}K_{jk}^2 - K_{jj}K_{ik}^2 - K_{kk}K_{ij}^2\,. \tag{6.7}$$

The first term here is known (from the size-1 marginals), and the last three terms are also known, as they depend only on the magnitude of the off-diagonal elements. Thus, the value of m_{ijk} tells us the sign of the $2K_{ij}K_{jk}K_{ik}$ term. Assuming that there are no zero off-diagonal elements, we can leverage this information, aggregated across all $O(N^3)$ size-3 sets, to get all of K's off-diagonal signs. Rising, Kulesza, and Taskar (2014) provide an $O(N^3)$ algorithm that solves this aggregation problem. Finally, given K, the kernel L is recoverable via Equation (2.14).

There are several obvious practical issues with this low-order marginal matching scheme. First, if the data is not actually drawn from a DPP ($Y_t \sim \mathcal{P}_L$), or if there are insufficient samples to accurately estimate the m_{ijk}, then we immediately run into problems with the above formulas. For instance, suppose that in all the sample sets Y_t, every time that item i appears, we also see item j. Then for some constant c we have: $m_i = m_j = m_{ij} = c$. The makes the square root in Equation (6.6) imaginary for any $c < 1$. Clearly, this data is not consistent with any DPP kernel.

Of course, the SVM kernel and distance metric learners do not attempt to directly set kernel entries to match the data; rather, a more nuanced objective is optimized, while enforcing a PSD constraint. Following this example, suppose we ask that K match the marginals as closely as possible while remaining a valid DPP kernel. For

instance, we might try to solve:

$$\min_{\substack{K:K\succeq 0,\\ I-K\succeq 0}} \sum_{i=1}^{N} \|K_{ii} - m_i\|_2^2 + \sum_{i=1}^{N}\sum_{j=1}^{N} \|\det(K_{\{i,j\}}) - m_{ij}\|_2^2 + \tag{6.8}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} \|\det(K_{\{i,j,k\}}) - m_{ijk}\|_2^2 \,. \tag{6.9}$$

Unfortunately, this objective is not necessarily convex. It is a degree-6 polynomial, and minimizing an arbitrary polynomial, even subject to only polynomial constraints, is NP-hard. There do exist techniques for optimizing semi-definite relaxations of polynomials, such as those summarized in Lasserre (2009, Chapter 5), but these are somewhat complex. We leave further exploration of this to future work.

Finally, as Section 2.2.7 touched upon, optimizing an objective based on entropy is also an alternative to likelihood maximization. Entropy maximization has to be subject to reasonable data-based constraints. We could try to maximize entropy plus Equation (6.9), but Equation (6.9) is a formidable objective itself. Alternatively, given features, we could maximize entropy subject to constraints requiring expected feature counts under the model to match their empirical counts. Depending on how the matching was formulated, this might be a more feasible problem. However, there is no known way of computing the DPP entropy of a fixed kernel exactly and efficiently. So no matter how basic the constraints, the maximum entropy approach would suffer from the difficulty that its objective value must be approximated. This might be worth the added effort though, if this objective, as conjectured by Lyons (2003), is indeed concave in K. Again, we leave exploration of this alternative to future work.

6.2 Feature representation

Before discussing likelihood-based learning methods, we introduce here some notation for situations in which a single, fixed $\mathcal{Y}$ is inadequate. For example, consider the simple document summarization task where we have a cluster of articles with sentences $\mathcal{Y}$, and wish to select a few sentences to constitute a summary. Suppose we are given T sample summaries $Y_1, \ldots, Y_T \subseteq \mathcal{Y}$, and somehow learn a kernel L from these. Then applying the MAP approximation techniques from the previous

chapter to this L, we would anticipate that the resulting subset Y would constitute a better summary than the T given ones. This is the type of single, fixed $\mathcal{Y}$ scenario we restrict to for the EM learning method in Section 6.6.3. However, in many practical scenarios it is inadequate.

Suppose that instead of having many sample summaries for a *single* document cluster, we have one or two sample summaries for *many* clusters. Further suppose that our goal is not to come up with better summaries for *these* clusters, but to extract knowledge of what makes a good summary in order craft summaries for *additional* (test) document clusters. This requires that each sentence be associated with some feature vector and that we learn weights on features, rather than item-specific parameters. More concretely, let $\mathcal{X}$ represent the input space (e.g. all articles). Let our training sample be $\{X^{(t)}, Y^{(t)}\}_{t=1}^{T}$, drawn i.i.d. from a distribution D over pairs: $(X, Y) \in \mathcal{X} \times 2^{\mathcal{Y}(X)}$. For features (e.g. tf-idf vectors) that can be computed for any item (e.g. any sentence), let θ represent corresponding feature weights. Then the log-likelihood of the data is:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \log \mathcal{P}_\theta(Y^{(t)} \mid X^{(t)}) \tag{6.10}$$

$$= \sum_{t=1}^{T} \left[\log \det(L_{Y^{(t)}}(X^{(t)}; \theta)) - \log \det(L(X^{(t)}; \theta) + I)\right] . \tag{6.11}$$

The goal here is to find θ that model the distribution D well, and as such can be used to identify high-probability Y when presented with an X unseen in the training data. The next few learning methods we consider (Kulesza and Taskar, 2011b,a; Affandi et al., 2014) apply to this more general, multi-ground-set learning problem.

6.3 Concave likelihood-based objectives

We turn now to the consideration of two methods that optimize restrictive parameterizations with concave likelihood-based objectives.

The first of these is from Kulesza and Taskar (2011b). The authors assume that a good similarity kernel, $S = \Phi^\top \Phi$, is already known. All that remains then is to learn item qualities. Kulesza and Taskar (2011b) propose the following log-linear

parametric form for quality:

$$q_i(X;\theta) = \exp\left(\frac{1}{2}\theta^\top \boldsymbol{f}_i(X)\right), \tag{6.12}$$

where $\boldsymbol{f}_i(X) \in \mathbb{R}^{m\times 1}$ is the quality feature vector for item i. These $\boldsymbol{f}_i$ could be the same as the similarity features ϕ_i, but that is not necessary. Kulesza and Taskar (2011b) prove that the resulting log-likelihood objective, Equation (6.11), is concave in θ. Moreover, they show that the gradient is just the difference between the empirical feature counts and the expected feature counts under the model. For a single (X, Y) this is:

$$\nabla\mathcal{L}(\theta) = \sum_{i:i\in Y} \boldsymbol{f}_i(X) - \sum_{i=1}^{N} K_{ii}(X;\theta)\boldsymbol{f}_i(X), \tag{6.13}$$

where K_{ii} is the marginal probability of element i, from the diagonal of the marginal kernel. The most expensive part of this computation is eigendecomposing $L(X;\theta)$ to get $K(X;\theta)$, which takes $O(N^\omega)$ time. Thus, we can efficiently compute and follow this gradient to find globally optimal θ.

Kulesza and Taskar (2011b) experiment with this quality learning framework on a document summarization task similar to the one described in the previous section. They explore a variety of quality features, such as sentence length, position of a sentence in its source article, and counts of personal pronouns. The likelihood-learned model is compared to several baselines. For establishing the importance of a DPP-tailored learning framework, the most significant of these baselines is one that relies on logistic regression. In short, as an alternative to optimizing likelihood, Kulesza and Taskar (2011b) consider setting θ via logistic regression. To do so, they transform the training data such that each sentence is labeled as "included" as "not-included", depending on whether it is in the oracle summary or not. Logistic regression is then used to learn θ to minimize classification error. As expected, incorporating these θ into a DPP model at test time does not produce summaries that score as well (according to ROUGE metrics such as those seen in Table 4.1). Despite having the advantage of the similarity model S at *test* time, the fact that logistic regression *training* does not take S into account results in inferior θ.

A second restrictive parameterization with a concave likelihood-based objective was proposed by Kulesza and Taskar (2011a). In this work, instead of draws from a

distribution D over (X, Y) pairs, the training data is of a different nature; labels come in the form of pairs of sets, where one set is preferred over the other: $\{Y_t^+, Y_t^-\}_{t=1}^T$. The sets are assumed to have the same size: $|Y_t^+| = |Y_t^-| = k$, and the goal is to learn a weighted sum of fixed k-DPP models. That is, assuming that a set of "expert" kernels $L^{(1)}, \ldots, L^{(D)}$ is given, the goal is to learn parameters θ that weight the associated DPPs:

$$\mathcal{P}_\theta^k = \sum_{d=1}^{D} \theta_d \mathcal{P}_{L^{(d)}}^k \qquad \text{s.t. } \sum_{d=1}^{D} \theta_d = 1\,. \tag{6.14}$$

Note that this is different from learning a single kernel $L(\theta) = \sum_{d=1}^{D} \theta_d L^{(d)}$, which is likely a much harder problem. The sum-of-DPPs learning is done by maximizing the likelihood difference between the preferred sets and the dis-preferred:

$$\max_{\theta:\theta^\top \mathbf{1}=1} \sum_{t=1}^{T} \mathcal{P}_\theta^k(Y_t^+) - \mathcal{P}_\theta^k(Y_t^-)\,. \tag{6.15}$$

Kulesza and Taskar (2011a) show that this objective is concave, and possesses an efficiently computable gradient. They test the learned model on an image search task similar to the one described in Section 1.1. As one of the baselines, each individual DPP with expert kernel $L^{(d)}$ is checked to see which has the highest training data accuracy. The highest-accuracy one is then compared to the sum-of-experts model on the test data. The performance difference is statistically significant in favor of the learned sum-of-experts.

The two methods discussed in this section are advantageous in that their simplicity makes for an easy (concave) optimization problem. However, the first method fails to address the need for learning similarity feature weights, which frequently arises in practical applications. The second method partially addresses the similarity learning question, but not by learning a single DPP model. Furthermore, it requires storing a potentially large set of kernel matrices, and the final model distribution is no longer a DPP. This means that many of the attractive computational and probabilistic properties associated with DPPs are lost. The following section considers more general parameterizations that lead to non-concave objectives, but that address similarity learning in the context of crafting a single DPP model.

6.4 Non-concave likelihood-based objectives

As mentioned in Section 2.2.6, some parameterizations make likelihood maximization more difficult: to learn weights such that L that is a weighted sum of fixed kernels, $L^{(1)}, \ldots, L^{(D)}$, is conjectured to be NP-hard. Log-likelihood is also non-concave in the setting where L is restricted to be a parametric kernel such as a Gaussian. Additionally, non-concavity is present if the kernel entries are completely unrestricted (except for a PSD constraint).

Kulesza (2012, Section 4.3.1) considers the possibility that non-concavity arises simply from an identifiability problem. That is, for any DPP described by kernel L and any diagonal matrix D, one can show that the kernel DLD^{-1} describes the same DPP (same probabilities). Even restricting to D for which this product matrix is symmetric, there are up to 2^{N-1} distinct kernels that all describe the same DPP; there are exactly this many if L has no zeros. Lack of identifiability makes it impossible to have a *strictly* concave objective, as that would require a single L to be better than all the rest. More generally, it suggests that a concave objective is unlikely to be found unless we first eliminate the identifiability issue. In fact, Kulesza (2012, Proposition 4.2) shows that only trivial objectives, whose maximizers are diagonal matrices, can be concave in such a setting. Kulesza (2012, Theorem 4.1) further provides an exact characterization of all kernels that give rise to the same DPP: D-similarity.

Definition 6.1. D-similarity: *Two $N \times N$ symmetric matrices L and M are D-similar if $L = DMD^{-1}$ for some $N \times N$ diagonal matrix D with entries $D_{ii} \in \{-1, 1\}$ for $i \in \{1, \ldots, N\}$.*

With this definition in hand, we can restrict parameterizations such that there is only a single candidate L for each DPP. Kulesza (2012) considers several methods for doing this, but none of them results in a concave objective. For example, suppose we require that all of L's entries be non-negative. This takes care of the identifiability problem. It also rules out some valid DPPs, but this restriction would probably be worth that loss if it resulted in a concave objective. Unfortunately, the resulting objective:

$$\max_{\substack{L: L \succeq 0, \\ L_{ij} \geq 0}} \sum_{t=1}^{T} \log\det(L_{Y_t}) - \log\det(L + I) \tag{6.16}$$

is still a difference of two concave terms, which is non-concave overall. Thus, it seems that identifiability is only one component of the hardness of DPP learning. Still, restricting parameterizations to avoid the identifiability problem may be a good practical tool to use for smoothing the search space. The EM method of Section 6.6.3 does not use this trick, but it could easily be incorporated.

6.5 MCMC approach for parametric kernels

In this section, we summarize the methods that Affandi et al. (2014) develop for handling non-concavity of log-likelihood in the setting where L is restricted to be a parametric kernel, such as a Gaussian. Note that requiring L to take on such parameterizations represents a restriction of the search space (though not a significant enough restriction to make log-likelihood concave). For example, lost expressivity is obvious in the case of a Gaussian kernel—it is stationary, so its kernel values only depend on differences between feature vectors, $B_i - B_j$, rather than absolute feature values.

Prior to Affandi et al. (2014), the only attempt to learn such parametric DPP kernels was the work of Lavancier, Møller, and Rubak (2012, Section 6). They applied the Nelder-Mead method to optimize likelihood. This method has no convergence guarantees though, and can even fail to find the optimum of a strictly convex function (McKinnon, 1998). Its advantage is that it doesn't require gradient computations. This method may be of value in the continuous DPP setting, described in Section 3.4, but for discrete DPPs computing the gradient is not difficult; Affandi et al. (2014, Appendix A) derive formulas for log-likelihood gradients in the special cases of Gaussian kernels and polynomial kernels.

Affandi et al. (2014) focus on developing Bayesian methods for selecting L's parameters. Rather than trying to *maximize* likelihood, Affandi et al. (2014) propose *sampling* from the posterior distribution over kernel parameters. Let $\mathcal{P}(\theta)$ represent a parameter prior. Then the posterior over θ, given example sets $Y_1, \ldots, Y_T$, is:

$$\mathcal{P}(\theta \mid Y_1, \ldots Y_T) \propto \mathcal{P}(\theta) \prod_{t=1}^{T} \frac{\det(L_{Y_t}(\theta))}{\det(L(\theta) + I)} \,. \tag{6.17}$$

Computing the normalizer for this distribution may be difficult, so rather than directly sampling θ, Affandi et al. (2014) rely on Markov chain Monte Carlo (MCMC)

methods. The first one they explore is random-walk Metropolis Hastings. To speed computation of the acceptance ratio, which requires computing $\det(L(\theta) + I)$, they further propose approximating $L(\theta)$ by truncating its eigendecomposition to the top-k largest eigenvalues (and associated eigenvectors). This is important for making the method applicable to continuous DPPs, whose eigendecompositions may be an infinite sum. The second MCMC method Affandi et al. (2014) apply is slice sampling. This type of sampling does not require tuning a proposal distribution, and hence can in some cases converge more quickly than Metropolis Hastings. No mixing time bounds are proven, but as a test of chain convergence Affandi et al. (2014) show that it is simple to check whether the model moments match the empirical data moments well. If the feature vector for the ith item is ϕ_i, then in the discrete case we can write the mth model moment as:

$$\mathbb{E}_\theta[\phi^m] = \sum_{i=1}^{N} \phi_i^m K_{ii}(\theta)\,, \tag{6.18}$$

where $K(\theta)$ is the marginal kernel. If these are close to the data moments:

$$\mathbb{E}[\phi^m] = \sum_{t=1}^{T} \sum_{i:i\in Y_t} \phi_i^m\,, \tag{6.19}$$

then the chain has most likely converged.

Affandi et al. (2014) test these MCMC methods on two practical applications. The first is diabetic neuropathy prediction: skin tissue imaging reveals that nerve fibers become more clustered as diabetes progresses, so the task in this case is to classify images of nerve fibers as belonging to *mildly* diabetic versus *severely* diabetic patients. To accomplish this, Affandi et al. (2014) first separate the data into these two classes and then sample two sets of parameters θ_1, θ_2 to define a separate Gaussian kernel for each. The dataset available is small, but in a leave-one-out cross-validation test, all images were found to have higher likelihood under the correct θ_i. The second application Affandi et al. (2014) test re-purposes the data from Kulesza and Taskar (2011a)'s image search task. Again assuming Gaussian kernels, for each image search query two sets of parameters are learned, θ_1, θ_2. The first is learned from example sets Y_t consisting of the top six image results from Google, and the second is learned from example sets that are partially human-annotated. The resulting parameters indicate some potential discrepancies in the features that Google considers important for

diversity versus those that humans consider important. For example, color seems to be more important to people for queries such as "cars" and "dogs" than the Google search engine currently accounts for.

6.6 EM APPROACH FOR UNRESTRICTED KERNELS

In this section, we describe an expectation-maximization (EM) scheme for optimizing likelihood in the completely unrestricted setting where L's entries can take on any values (as long as the overall L is PSD). In particular, the algorithm we derive exploits L's eigendecomposition and negates the need for the projection step that is required to maintain positive semi-definiteness of L when applying naïve gradient ascent.

We will assume that the training data consists of n example subsets, $\{Y_1, \ldots, Y_n\}$, where $Y_i \subseteq \{1, \ldots, N\}$ for all i. Our goal is to maximize the likelihood of these example sets by learning a kernel L that assigns them high probability. To simplify some of the math, we will work with the marginal kernel K instead of L, but Equation (2.14) can be used to obtain L from K at any point. We first describe in Section 6.6.1 a naive optimization procedure: projected gradient ascent on the entries of the marginal kernel K, which will serve as a baseline in our experiments. We then develop an EM method: Section 6.6.2 changes variables from kernel entries to eigenvalues and eigenvectors (introducing a hidden variable in the process), Section 6.6.3 applies Jensen's inequality to lower-bound the objective, and Sections 6.6.4-6.6.6 outline a coordinate ascent procedure on this lower bound. We conclude by testing the resulting EM algorithm on a real-world product recommendation task. Section 6.7 describes these empirical results.

6.6.1 PROJECTED GRADIENT ASCENT

Recalling Equation (2.16), the log-likelihood can be expressed in terms of the marginal kernel K as follows:

$$\mathcal{L}(K) = \sum_{i=1}^{n} \log\left(|\det(K - I_{\overline{Y}_i})|\right) . \tag{6.20}$$

The associated optimization problem is:

$$\max_{\substack{K:K\succeq 0,\\ I-K\succeq 0}} \mathcal{L}(K)\,, \tag{6.21}$$

where the first constraint ensures that K is PSD and the second puts an upper limit of 1 on its eigenvalues. The partial derivative of $\mathcal{L}$ with respect to K is easy to compute by applying a standard matrix derivative rule (Petersen and Pedersen, 2012, Equation 57):

$$\frac{\partial \mathcal{L}(K)}{\partial K} = \sum_{i=1}^{n} (K - I_{\overline{Y}_i})^{-1}\,. \tag{6.22}$$

Thus, projected gradient ascent (Levitin and Polyak, 1966) is a viable, simple optimization technique. Algorithm 11 outlines this method, which we refer to as K-Ascent (KA). The initial K supplied as input to the algorithm can be any PSD matrix with eigenvalues ≤ 1. The first part of the projection step, $\max(\boldsymbol{\lambda}, 0)$, chooses the closest (in Frobenius norm) PSD matrix to Q (Henrion and Malick, 2011, Equation 1). The second part, $\min(\boldsymbol{\lambda}, 1)$, caps the eigenvalues at 1. Notice that only the eigenvalues have to be projected; K remains symmetric after the gradient step, so its eigenvectors are already guaranteed to be real.

Unfortunately, the eigenvalue projection can take us to a poor local optima. To see this, consider the case where the starting kernel K is a poor fit to the data. In this case, a large initial step size η will probably be accepted; even though such a step will likely result in the truncation of many eigenvalues at 0, the resulting matrix will still be an improvement over the poor initial K. However, with many zero eigenvalues, the new K will be near-diagonal, and, unfortunately, Equation (6.22) dictates that if the current K is diagonal, then its gradient is as well. Thus, the KA algorithm cannot easily move to any highly non-diagonal matrix. It is possible that employing more complex step-size selection mechanisms could alleviate this problem, but the EM algorithm we develop in the next section will negate the need for these.

The EM algorithm we develop also has an advantage in terms of asymptotic runtime. The computational complexity of KA is dominated by the matrix inverses of the $\mathcal{L}$ derivative, each of which requires $O(N^\omega)$ operations, and by the eigendecomposition needed for the projection, also $O(N^\omega)$. The overall runtime of KA, assuming T_1 iterations until convergence and an average of T_2 iterations to find a step size, is $O(T_1 n N^\omega + T_1 T_2 N^\omega)$. As we will show in the following sections, the overall runtime

Algorithm 11: K-Ascent (KA)

Input: K, $\{Y_1, \ldots, Y_n\}$, c
repeat
  $G \leftarrow \frac{\partial \mathcal{L}(K)}{\partial K}$ (Eq. 6.22)
  $\eta \leftarrow 1$
  repeat
    $Q \leftarrow K + \eta G$
    Eigendecompose Q into $V, \boldsymbol{\lambda}$
    $\boldsymbol{\lambda} \leftarrow \min(\max(\boldsymbol{\lambda}, 0), 1)$
    $Q \leftarrow V \operatorname{diag}(\boldsymbol{\lambda}) V^\top$
    $\eta \leftarrow \frac{\eta}{2}$
  until $\mathcal{L}(Q) > \mathcal{L}(K)$
  $\delta \leftarrow \mathcal{L}(Q) - \mathcal{L}(K)$
  $K \leftarrow Q$
until $\delta < c$
Output: K

Algorithm 12: Expectation-Maximization (EM)

Input: K, $\{Y_1, \ldots, Y_n\}$, c
Eigendecompose K into $V, \boldsymbol{\lambda}$
repeat
  for $j = 1, \ldots, N$ **do**
    $\boldsymbol{\lambda}'_j \leftarrow \frac{1}{n} \sum_{i=1}^{n} p_K(j \in J \mid Y_i)$
  $G \leftarrow \frac{\partial F(V, \boldsymbol{\lambda}')}{\partial V}$ (Eq. 6.61)
  $\eta \leftarrow 1$
  repeat
    $V' \leftarrow V \exp[\eta \left(V^\top G - G^\top V\right)]$
    $\eta \leftarrow \frac{\eta}{2}$
  until $\mathcal{L}(V', \boldsymbol{\lambda}') > \mathcal{L}(V, \boldsymbol{\lambda}')$
  $\delta \leftarrow F(V', \boldsymbol{\lambda}') - F(V, \boldsymbol{\lambda})$
  $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}', \quad V \leftarrow V', \quad \eta \leftarrow 2\eta$
until $\delta < c$
Output: K

Figure 6.1: **Left**: Projected gradient ascent on the entries of K. **Right**: Expectation-maximization on the eigendecomposition of K.

of the EM algorithm is $O(T_1 n N k^2 + T_1 T_2 N^\omega)$, which can be substantially better than KA's runtime for $k \ll N$.

6.6.2 Eigendecomposing

As we have seen in previous chapters, eigendecomposition is key to many core DPP algorithms such as sampling and marginalization. This is because the eigendecomposition provides an alternative view of the DPP as a generative process, which often leads to more efficient algorithms. For instance, sampling a set Y can be broken down into a two-step process, the first of which involves generating a hidden variable $J \subseteq \{1, \ldots, N\}$ that codes for a particular set of K's eigenvectors. We review this process below, then exploit it to develop an EM optimization scheme.

Suppose $K = V \Lambda V^\top$ is an eigendecomposition of K. Let V^J denote the submatrix of V containing only the columns corresponding to the indices in a set $J \subseteq \{1, \ldots, N\}$. Consider the corresponding marginal kernel, with all selected eigen-

values set to 1:

$$K^{V^J} = \sum_{j:j\in J} \boldsymbol{v}_j \boldsymbol{v}_j^\top = V^J (V^J)^\top . \tag{6.23}$$

Recall that any such kernel whose eigenvalues are all 1 is called an *elementary* DPP. According to Hough et al. (2006, Theorem 7), a DPP with marginal kernel K is a mixture of all 2^N possible elementary DPPs:

$$\mathcal{P}(Y) = \sum_{J:J\subseteq\{1,\ldots,N\}} \mathcal{P}^{V^J}(Y) \prod_{j:j\in J} \lambda_j \prod_{j:j\notin J} (1-\lambda_j) , \tag{6.24}$$

$$\text{where } \mathcal{P}^{V^J}(Y) = \mathbb{1}(|Y| = |J|) \det(K_Y^{V^J}) . \tag{6.25}$$

This perspective is the basis for the sampling method of Algorithm 1, where first a set J is chosen according to its mixture weight in Equation (6.25), and then P^{V^J} is sampled from. In this sense, the index set J is an intermediate hidden variable in the process for generating a sample Y.

We can exploit this hidden variable J to develop an EM algorithm for learning K. Re-writing the data log-likelihood to make the hidden variable explicit:

$$\mathcal{L}(K) = \mathcal{L}(\Lambda, V) = \sum_{i=1}^{n} \log \left(\sum_J p_K(J, Y_i) \right) \tag{6.26}$$

$$= \sum_{i=1}^{n} \log \left(\sum_J p_K(Y_i \mid J) p_K(J) \right) , \tag{6.27}$$

$$\text{where } p_K(J) = \prod_{j:j\in J} \lambda_j \prod_{j:j\notin J} (1-\lambda_j) , \tag{6.28}$$

$$\text{and } p_K(Y_i \mid J) = \mathbb{1}(|Y_i| = |J|) \det([V^J (V^J)^\top]_{Y_i}) . \tag{6.29}$$

These equations follow directly from Equations (6.23) and (6.25).

6.6.3 Lower bounding the objective

We now introduce an auxiliary distribution, $q(J \mid Y_i)$, and deploy it with Jensen's inequality to lower-bound the likelihood objective. This is a standard technique for developing EM schemes for dealing with hidden variables (Neal and Hinton, 1998).

Proceeding in this direction:

$$\mathcal{L}(V,\Lambda) = \sum_{i=1}^{n} \log\left(\sum_{J} q(J \mid Y_i)\frac{p_K(J,Y_i)}{q(J \mid Y_i)}\right) \geq \tag{6.30}$$

$$\sum_{i=1}^{n}\sum_{J} q(J \mid Y_i)\log\left(\frac{p_K(J,Y_i)}{q(J \mid Y_i)}\right) \equiv F(q,V,\Lambda)\,. \tag{6.31}$$

The function $F(q,V,\Lambda)$ can be expressed in either of the following two forms:

$$F(q,V,\Lambda) = \sum_{i=1}^{n} -\mathbf{KL}(q(J \mid Y_i) \parallel p_K(J \mid Y_i)) + \mathcal{L}(V,\Lambda) \tag{6.32}$$

$$= \sum_{i=1}^{n} \mathbb{E}_q[\log p_K(J,Y_i)] + H(q)\,, \tag{6.33}$$

where H is entropy. Consider optimizing this new objective by coordinate ascent. From Equation (6.32) it is clear that, holding V,Λ constant, F is concave in q. This follows from the concavity of **KL** divergence. Holding q constant in Equation (6.33) yields the following function:

$$F(V,\Lambda) = \sum_{i=1}^{n}\sum_{J} q(J \mid Y_i)\left[\log p_K(J) + \log p_K(Y_i \mid J)\right]\,. \tag{6.34}$$

This expression is concave in λ_j, since log is concave. However, the optimization problem is not concave in V due to the non-convex $V^\top V = I$ constraint. We describe in Section 6.6.6 one way to handle this.

To summarize, coordinate ascent on $F(q,V,\Lambda)$ alternates the following "expectation" and "maximization" steps; the first is concave in q, and the second is concave in the eigenvalues:

$$\text{E-step:}\quad \min_{q} \sum_{i=1}^{n} \mathbf{KL}(q(J \mid Y_i) \parallel p_K(J \mid Y_i))\,, \tag{6.35}$$

$$\text{M-step:}\quad \max_{V,\Lambda} \sum_{i=1}^{n} \mathbb{E}_q[\log p_K(J,Y_i)] \;\text{ s.t. }\; \mathbf{0} \leq \boldsymbol{\lambda} \leq \mathbf{1}, V^\top V = I\,. \tag{6.36}$$

6.6.4 E-step

The E-step is easily solved by setting $q(J \mid Y_i) = p_K(J \mid Y_i)$, which minimizes the KL divergence. Interestingly, we can show that this distribution is itself a conditional

DPP, and hence can be compactly described by an $N \times N$ kernel matrix. Thus, to complete the E-step, we simply need to construct this kernel. Lemma 6.2 gives an explicit formula.

Lemma 6.2. *At the completion of the E-step, $q(J \mid Y_i)$ with $|Y_i| = k$ is a k-DPP with (non-marginal) kernel Q^{Y_i}:*

$$Q^{Y_i} = RZ^{Y_i}R, \text{ and } q(J \mid Y_i) \propto \mathbb{1}(|Y_i| = |J|) \det(Q^{Y_i}_J), \text{ where} \tag{6.37}$$

$$U = V^\top, \quad Z^{Y_i} = U^{Y_i}(U^{Y_i})^\top, \text{ and } R = \operatorname{diag}\left(\sqrt{\boldsymbol{\lambda}/(\mathbf{1} - \boldsymbol{\lambda})}\right). \tag{6.38}$$

Proof. Since the E-step is an unconstrained KL divergence minimization, we have:

$$q(J \mid Y_i) = p_K(J \mid Y_i) = \frac{p_K(J, Y_i)}{p_K(Y_i)} \propto p_K(J, Y_i) = p_K(J)p_K(Y_i \mid J), \tag{6.39}$$

where the proportionality follows because Y_i is held constant in the conditional q distribution. Recalling Equation (6.29), notice that $p_K(Y_i \mid J)$ can be re-expressed as follows:

$$p_K(Y_i \mid J) = \mathbb{1}(|Y_i| = |J|) \det([V^J(V^J)^\top]_{Y_i}) \tag{6.40}$$

$$= \mathbb{1}(|Y_i| = |J|) \det([U^{Y_i}(U^{Y_i})^\top]_J). \tag{6.41}$$

This follows from the identity $\det(AA^\top) = \det(A^\top A)$, which applies for any full-rank square matrix A. The subsequent swapping of J and Y_i, once $V^\top$ is re-written as U, does not change the indexed submatrix. Plugging this back into Equation (6.39):

$$q(J \mid Y_i) \propto p_K(J)\mathbb{1}(|Y_i| = |J|) \det([U^{Y_i}(U^{Y_i})^\top]_J) \tag{6.42}$$

$$= p_K(J)\mathbb{1}(|Y_i| = |J|)P^{U^{Y_i}}(J), \tag{6.43}$$

where $P^{U^{Y_i}}$ represents an elementary DPP, just as in Equation (6.23), but over J rather than over Y. Multiplying this expression by a term that is constant for all J maintains proportionality and allows us to simplify the the $p_K(J)$ term. Taking the definition of $p_K(J)$ from Equation (6.29):

$$q(J \mid Y_i) \propto \left(\prod_{j=1}^{N} \frac{1}{1 - \lambda_j}\right) \mathbb{1}(|Y_i| = |J|)P^{U^{Y_i}}(J) \prod_{j:j\in J} \lambda_j \prod_{j:j\notin J} (1 - \lambda_j) \tag{6.44}$$

$$= \mathbb{1}(|Y_i| = |J|)P^{U^{Y_i}}(J) \prod_{j:j\in J} \frac{\lambda_j}{1 - \lambda_j}. \tag{6.45}$$

Having eliminated all dependence on $j \notin J$, it is now possible to express $q(J \mid Y_i)$ as the J principal minor of a PSD kernel matrix (see Q^{Y_i} in the statement of the lemma). Thus, q is a k-DPP. □

6.6.5 M-step eigenvalue updates

The M-step update for the eigenvalues is a closed-form expression with no need for projection. Taking the derivative of Equation (6.34) with respect to λ_j, setting it equal to zero, and solving for λ_j:

$$\lambda_j = \frac{1}{n} \sum_{i=1}^{n} \sum_{J: j \in J} q(J \mid Y_i) \,. \tag{6.46}$$

The exponential-sized sum here is impractical, but we can eliminate it. Recall from Lemma 6.2 that $q(J \mid Y_i)$ is a k-DPP with kernel Q^{Y_i}. Thus, we can use k-DPP marginalization algorithms to efficiently compute the sum over J. Specifically, the exponential-size sum over J from Equation (6.46) can be reduced to the computation of an eigendecomposition and several elementary symmetric polynomials on the resulting eigenvalues. Let $e_{k-1}^{-j}(Q^{Y_i})$ be the $(k-1)$-order elementary symmetric polynomial over all eigenvalues of Q^{Y_i} except for the jth one. Let $\hat{V}$ represent the eigenvectors of Q^{Y_i}, with $\hat{v}_r(j)$ indicating the jth element of the rth eigenvector. Then, by direct application of Equation (3.5), q's singleton marginals are:

$$\sum_{J: j \in J} q(J \mid Y_i) = q(j \in J \mid Y_i) = \frac{1}{e_{|Y_i|}^{N}(Q^{Y_i})} \sum_{r=1}^{N} \hat{v}_r(j)^2 \hat{\lambda}_r e_{|Y_i|-1}^{-r}(Q^{Y_i}) \,. \tag{6.47}$$

As previously noted, elementary symmetric polynomials can be efficiently computed using Baker and Harwell (1996)'s "summation algorithm".

We can further reduce the complexity of this formula by noting that rank of the $N \times N$ matrix $Q^{Y_i} = RZ^{Y_i}R$ is at most $|Y_i|$. Because Q^{Y_i} only has $|Y_i|$ non-zero eigenvalues, it is the case that, for all r:

$$\hat{\lambda}_r e_{|Y_i|-1}^{-r}(Q^{Y_i}) = e_{|Y_i|}^{N}(Q^{Y_i}) \,. \tag{6.48}$$

Recalling that the eigenvectors and eigenvalues of Q^{Y_i} are denoted $\hat{V}, \hat{\Lambda}$, the computation of the singleton marginals of q that are necessary for the M-step eigenvalue

updates can be written as follows:

$$q(j \in J \mid Y_i) = \frac{1}{e^N_{|Y_i|}(Q^{Y_i})} \sum_{r=1}^{N} \hat{v}_r(j)^2 \hat{\lambda}_r e^{-r}_{|Y_i|-1}(Q^{Y_i}) = \sum_{r=1}^{|Y_i|} \hat{v}_r(j)^2 \,. \tag{6.49}$$

This simplified formula is dominated by the $O(N^\omega)$ cost of the eigendecomposition required to find $\hat{V}$. This cost can be further reduced, to $O(Nk^2)$, by eigendecomposing a related matrix instead of Q^{Y_i}. Specifically, consider the $|Y_i| \times |Y_i|$ matrix $H^{Y_i} = V_{Y_i} R^2 V_{Y_i}^\top$. Let $\tilde{V}$ and $\tilde{\Lambda}$ be the eigenvectors and eigenvalues of H^{Y_i}. This $\tilde{\Lambda}$ is identical to the non-zero eigenvalues of Q^{Y_i}, $\hat{\Lambda}$, and its eigenvectors are related as follows:

$$\hat{V} = R V_{Y_i}^\top \tilde{V} \text{diag}\left(\frac{\mathbf{1}}{\sqrt{\tilde{\boldsymbol{\lambda}}}}\right) . \tag{6.50}$$

Getting $\hat{V}$ via Equation (6.50) is an $O(N|Y_i|^2)$ operation, given the eigendecomposition of H^{Y_i}. Since this eigendecomposition is an $O(|Y_i|^\omega)$ operation, it is dominated by the $O(N|Y_i|^2)$. To compute Equation (6.49) for all j requires only $O(Nk)$ time, given $\hat{V}$. Thus, letting $k = \max_i |Y_i|$, the size of the largest example set, the overall complexity of the eigenvalue updates is $O(nNk^2)$.

Note that this eigenvalue update is self-normalizing, so explicit enforcement of the $0 \leq \lambda_j \leq 1$ constraint is unnecessary. There is one small caveat: the Q^{Y_i} matrix will be infinite if any λ_j is exactly equal to 1 (due to R in Equation (6.38)). In practice, we simply tighten the constraint on $\boldsymbol{\lambda}$ to keep it slightly below 1.

6.6.6 M-step eigenvector updates

Turning now to the M-step update for the eigenvectors, notice that Equation (6.34)'s $p_K(J)$ term does not depend on the eigenvectors, so we only have to be concerned with the $p_K(Y_i \mid J)$ term when computing the eigenvector derivatives. Recall that this term is defined as follows:

$$p_K(Y_i \mid J) = \mathbb{1}(|Y_i| = |J|) \det\left(\left[V^J (V^J)^\top\right]_{Y_i}\right) . \tag{6.51}$$

Thus, applying standard matrix derivative rules such as Petersen and Pedersen (2012, Equation 55), the gradient of the M-step objective with respect to entry (a, b) of V is:

$$\frac{\partial F(V, \Lambda)}{\partial [V]_{ab}} = \sum_{i=1}^{n} \sum_{J} q(J \mid Y_i) \mathbb{1}(a \in Y_i \wedge b \in J) 2 [(W^J_{Y_i})^{-1}]_{g_{Y_i}(a)} \cdot \boldsymbol{v}_b(Y_i) \,, \tag{6.52}$$

where $W_{Y_i}^J = [V^J (V^J)^T]_{Y_i}$ and the subscript $g_{Y_i}(a)$ indicates the index of a in Y_i. The $[(W_{Y_i}^J)^{-1}]_{g_{Y_i}(a)}$ indicates the corresponding row in $W_{Y_i}^J$, and $\boldsymbol{v}_b(Y_i)$ is eigenvector b restricted to Y_i. Based on this, we can more simply express the derivative with respect to the entire V matrix:

$$\frac{\partial F(V,\Lambda)}{\partial V} = \sum_{i=1}^{n} \sum_{J} 2q(J \mid Y_i)(\dot{W}_{Y_i}^J)^{-1}\dot{V}\,, \tag{6.53}$$

where the $\dot{V} = \text{diag}(\mathbf{1}_{Y_i})V\text{diag}(\mathbf{1}_J)$ is equal to V with the rows $\ell \notin Y$ and the columns $j \notin J$ zeroed. Similarly, the other half of the expression represents $(W_{Y_i}^J)^{-1}$ sorted such that $g_{Y_i}(\ell) = \ell$ and expanded with zero rows for all $\ell \notin Y_i$ and zero columns for all $\ell \notin Y_i$. The exponential-size sum over J could be approximated by drawing a sufficient number of samples from q, but in practice that proves somewhat slow. It turns out that it is possible, by exploiting the relationship between Z and V, to perform the first gradient step on V without needing to sample q.

Exact computation of the first gradient

Recall that Z^{Y_i} is defined to be $U^{Y_i}(U^{Y_i})^\top$, where $U = V^\top$. The $p_K(Y_i \mid J)$ portion of the M-step objective, Equation (6.51), can be re-written in terms of Z^{Y_i}:

$$p_K(Y_i \mid J) = \mathbb{1}(|Y_i| = |J|)\det\left(Z_J^{Y_i}\right)\,. \tag{6.54}$$

Taking the gradient of the M-step objective with respect to Z^{Y_i}:

$$\frac{\partial F(V,\Lambda)}{\partial Z^{Y_i}} = \sum_{J} q(J \mid Y_i)(Z_J^{Y_i})^{-1}\,. \tag{6.55}$$

Plugging in the k-DPP form of $q(J \mid Y_i)$ derived in Lemma 6.2 of the E-step section:

$$\frac{\partial F(V,\Lambda)}{\partial Z^{Y_i}} = \frac{1}{e_{|Y_i|}^N(Q^{Y_i})} \sum_{J:|J|=|Y_i|} \det(Q_J^{Y_i})(Z_J^{Y_i})^{-1}\,. \tag{6.56}$$

Recall from Section 3.1 the identity used to normalize a k-DPP, and consider taking its derivative with respect to Z^{Y_i}:

$$\sum_{J:|J|=k} \det(Q_J^{Y_i}) = e_k^N(Q^{Y_i}) \quad \underset{\text{derivative wrt } Z^{Y_i}}{\Longrightarrow} \quad \sum_{J:|J|=k} \det(Q_J^{Y_i})(Z_J^{Y_i})^{-1} = \frac{\partial e_k^N(Q^{Y_i})}{\partial Z^{Y_i}}\,. \tag{6.57}$$

Note that this relationship is only true at the start of the M-step, before V (and hence Z) undergoes any gradient updates; a gradient step for V would mean that Q^{Y_i}, which remains fixed during the M-step, could no longer can be expressed as $RZ^{Y_i}R$. Thus, the formula we develop in this section is only valid for the first gradient step. Plugging Equation (6.57) back into Equation (6.56):

$$\frac{\partial F(V,\Lambda)}{\partial Z^{Y_i}} = \frac{1}{e^N_{|Y_i|}(Q^{Y_i})} \frac{\partial e^N_{|Y_i|}(Q^{Y_i})}{\partial Z^{Y_i}} \,. \tag{6.58}$$

Multiplying this by the derivative of Z^{Y_i} with respect to V and summing over i gives the final form of the gradient with respect to V. Thus, we can compute the value of the first gradient on V exactly in polynomial time.

Faster computation of the first gradient

Recall from Section 6.6.5 that the rank of the $N \times N$ matrix $Q^{Y_i} = RZ^{Y_i}R$ is at most $|Y_i|$ and that its non-zero eigenvalues are identical to those of the $|Y_i| \times |Y_i|$ matrix $H^{Y_i} = V_{Y_i} R^2 V_{Y_i}^\top$. Since the elementary symmetric polynomial e^N_k depends only on the eigenvalues of its argument, this means H^{Y_i} can substitute for Q^{Y_i} in Equation (6.58), if we change variables back from Z to V:

$$\frac{\partial F(V,\Lambda)}{\partial V} = \sum_{i=1}^{n} \frac{1}{e^N_{|Y_i|}(H^{Y_i})} \frac{\partial e^N_{|Y_i|}(H^{Y_i})}{\partial V} \,, \tag{6.59}$$

where the i-th term in the sum is assumed to index into the Y_i rows of the V derivative. Further, because H is only size $|Y_i| \times |Y_i|$:

$$e^N_{|Y_i|}(H^{Y_i}) = e^{|Y_i|}_{|Y_i|}(H^{Y_i}) = \det(H^{Y_i}) \,. \tag{6.60}$$

Plugging this back into Equation (6.59) and applying standard matrix derivative rules:

$$\frac{\partial F(V,\Lambda)}{\partial V} = \sum_{i=1}^{n} \frac{1}{\det(H^{Y_i})} \frac{\partial \det(H^{Y_i})}{\partial V} = \sum_{i=1}^{n} 2(H^{Y_i})^{-1} V_{Y_i} R^2 \,. \tag{6.61}$$

Thus, the initial M-step derivative with respect to V can be more efficiently computed via the above equation. Specifically, the matrix H^{Y_i} can be computed in time $O(N|Y_i|^2)$, since R is a diagonal matrix. It can be inverted in time $O(|Y_i|^\omega)$, which is dominated by $O(N|Y_i|^2)$. Thus, letting $k = \max_i |Y_i|$, the size of the largest example set, the overall complexity of computing the eigenvector gradient in Equation (6.61) is $O(nNk^2)$.

Equation (6.61) is only valid for the first gradient step, so in practice we do not bother to fully optimize V in each M-step; we simply take a single gradient step on V. Ideally we would repeatedly evaluate the M-step objective, Equation (6.34), with various step sizes to find the optimal one. However, the M-step objective is intractable to evaluate exactly, as it is an expectation with respect to an exponential-size distribution. In practice, we solve this issue by performing an E-step for each trial step size. That is, we update q's distribution to match the updated V and Λ that define p_K, and then determine if the current step size is good by checking for improvement in the likelihood $\mathcal{L}$.

There is also the issue of enforcing the non-convex constraint $V^\top V = I$. We could project V to ensure this constraint, but, as previously discussed for eigen*values*, projection steps often lead to poor local optima. Thankfully, for the particular constraint associated with V, more sophisticated update techniques exist: the constraint $V^\top V = I$ corresponds to optimization over a Stiefel manifold, so the algorithm from Edelman, Arias, and Smith (1998, Page 326) can be employed. In practice, we simplify this algorithm by neglecting second-order information (the Hessian) and using the fact that the V in our application is full-rank. With these simplifications, the following multiplicative update is all that is needed:

$$V \leftarrow V \exp\left[\eta\left(V^\top \frac{\partial \mathcal{L}}{\partial V} - \left(\frac{\partial \mathcal{L}}{\partial V}\right)^\top V\right)\right], \tag{6.62}$$

where exp denotes the matrix exponential and η is the step size. Algorithm 12 summarizes the overall EM method. As previously mentioned, assuming T_1 iterations until convergence and an average of T_2 iterations to find a step size, its overall runtime is $O(T_1 nNk^2 + T_1 T_2 N^\omega)$. The first term in this complexity comes from the eigenvalue updates, Equation (6.49), and the eigenvector derivative computation, Equation (6.61). The second term comes from repeatedly computing the Stiefel manifold update of V, Equation (6.62), during the step size search.

6.7 Experiments

We test the proposed EM learning method (Algorithm 12) by comparing it to K-Ascent (KA, Algorithm 11). Code and data for all experiments can be downloaded here: https://code.google.com/p/em-for-dpps. Both EM and KA require a starting marginal kernel $\tilde{K}$. Note that neither EM nor KA can deal well with starting from a kernel with too many zeros. For example, starting from a diagonal kernel, both gradients, Equations (6.22) and (6.61), will be diagonal, resulting in no modeling of diversity. Thus, the two initialization options that we explore have non-trivial off-diagonals. The first of these options is relatively naive, while the other incorporates statistics from the data.

For the first initialization type, we use a Wishart distribution with N degrees of freedom and an identity covariance matrix to draw $\tilde{L} \sim \mathcal{W}_N(N, I)$. The Wishart distribution is relatively unassuming: in terms of eigenvectors, it spreads its mass uniformly over all unitary matrices (James, 1964). We make just one simple modification to its output to make it a better fit for practical data: we re-scale the resulting matrix by $1/N$ so that the corresponding DPP will place a non-trivial amount of probability mass on small sets. (The Wishart's mean is NI, so it tends to over-emphasize larger sets unless we re-scale.) We then convert $\tilde{L}$ to $\tilde{K}$ via Equation (2.12).

For the second initialization type, we employ a form of moment matching. Let m_i and m_{ij} represent the normalized frequencies of single items and pairs of items in the training data:

$$m_i = \frac{1}{n}\sum_{\ell=1}^{n} \mathbb{1}(i \in Y_\ell), \quad m_{ij} = \frac{1}{n}\sum_{\ell=1}^{n} \mathbb{1}(i \in Y_\ell \wedge j \in Y_\ell)\,. \tag{6.63}$$

Recalling the definition of the marginal kernel from Equation (2.13), we attempt to match the first and second order moments by setting $\tilde{K}$ according to:

$$\tilde{K}_{ii} = m_i, \quad \tilde{K}_{ij} = \sqrt{\max\left(\tilde{K}_{ii}\tilde{K}_{jj} - m_{ij}, 0\right)}\,. \tag{6.64}$$

To ensure a valid starting kernel, we then project $\tilde{K}$ by clipping its eigenvalues at 0 and 1.

Category	N	# of Regs
feeding	100	13300
gear	100	11776
diaper	100	11731
bedding	100	11459
apparel	100	10479
bath	100	10179
toys	62	7051
health	62	9839
media	58	4132
strollers	40	5175
safety	36	6224
carseats	34	5296
furniture	32	4965

Table 6.1: Size of the post-filtering ground set for each product category, and the associated number of sub-registries (subsets of $\{1, \ldots, N\}$).

6.7.1 Baby registry tests

Consider a product recommendation task, where the ground set comprises N products that can be added to a particular category (e.g., toys or safety) in a baby registry. A very simple recommendation system might suggest products that are popular with other consumers. However, this does not account for negative interactions; if a consumer has already chosen a carseat, they most likely will not choose an additional carseat, no matter how popular it is with other consumers. DPPs are ideal for capturing such negative interactions. A learned DPP could be used to populate an initial, basic registry, as well as to provide live updates of product recommendations as a consumer builds their registry.

To test our DPP learning algorithms, we collected a dataset consisting of $29{,}632$ baby registries from Amazon.com, filtering out those listing fewer than 5 or more than 100 products. Amazon characterizes each product in a baby registry as belonging to one of 18 categories, such as "toys" and"safety". For each registry, we created sub-registries by splitting it according to these categories. (A registry with 5 toy items and 10 safety items produces two sub-registries.) For each category, we then filtered down to its top 100 most frequent items, and removed any product that did not oc-

cur in at least 100 sub-registries. We discarded categories with $N < 25$ or fewer than $2N$ remaining (non-empty) sub-registries for training. The resulting 13 categories have an average inventory size of $N = 71$ products and an average number of sub-registries $n = 8{,}585$. We used 70% of the data for training and 30% for testing. Note that categories such as "carseats" contain more diverse items than just their namesake; for instance, "carseats" also contains items such as seat back kick protectors and rear-facing baby view mirrors. See Table 6.1 for more dataset details.

Figure 6.2 shows the relative test log-likelihood differences of EM and KA when starting from a Wishart initialization. These numbers are the medians from 25 trials (draws from the Wishart). Quartiles for this experiment and all subsequent ones can be found in Table 6.2. EM gains an average of 3.7%, but has a much greater advantage for some categories than for others. Speculating that EM has more of an advantage when the off-diagonal components of K are truly important—when products exhibit strong negative interactions—we created a matrix M for each category with the true data marginals from Equation (6.63) as its entries. We then checked the value of $d = \frac{1}{N}\frac{||M||_F}{||\text{diag}(M)||_2}$. This value correlates well with the relative gains for EM: the 4 categories for which EM has the largest gains (safety, furniture, carseats, and strollers) all exhibit $d > 0.025$, while categories such as feeding and gear have $d < 0.012$. Investigating further, we found that, as foreshadowed in Section 6.6.1, KA performs particularly poorly in the high-d setting because of its projection step—projection can result in KA learning a near-diagonal matrix.

If instead of the Wishart initialization we use the moments-matching initializer, this alleviates KA's projection problem, as it provides a starting point closer to the true kernel. With this initializer, KA and EM have comparable test log-likelihoods (average EM gain of 0.4%). However, the moments-matching initializer is not a perfect fix for the KA algorithm in all settings. For instance, consider a data-poor setting, where for each category we have only $n = 2N$ training examples. In this case, even with the moments-matching initializer EM has a significant edge over KA, as shown in Figure 6.2: EM gains an average of 4.5%, with a maximum gain of 16.5% for the safety category.

To give a concrete example of the advantages of EM training, Figure 6.3 shows a greedy approximation (Algorithm 5) to the most-likely ten-item registry in the category "safety", according to a Wishart-initialized EM model. The corresponding

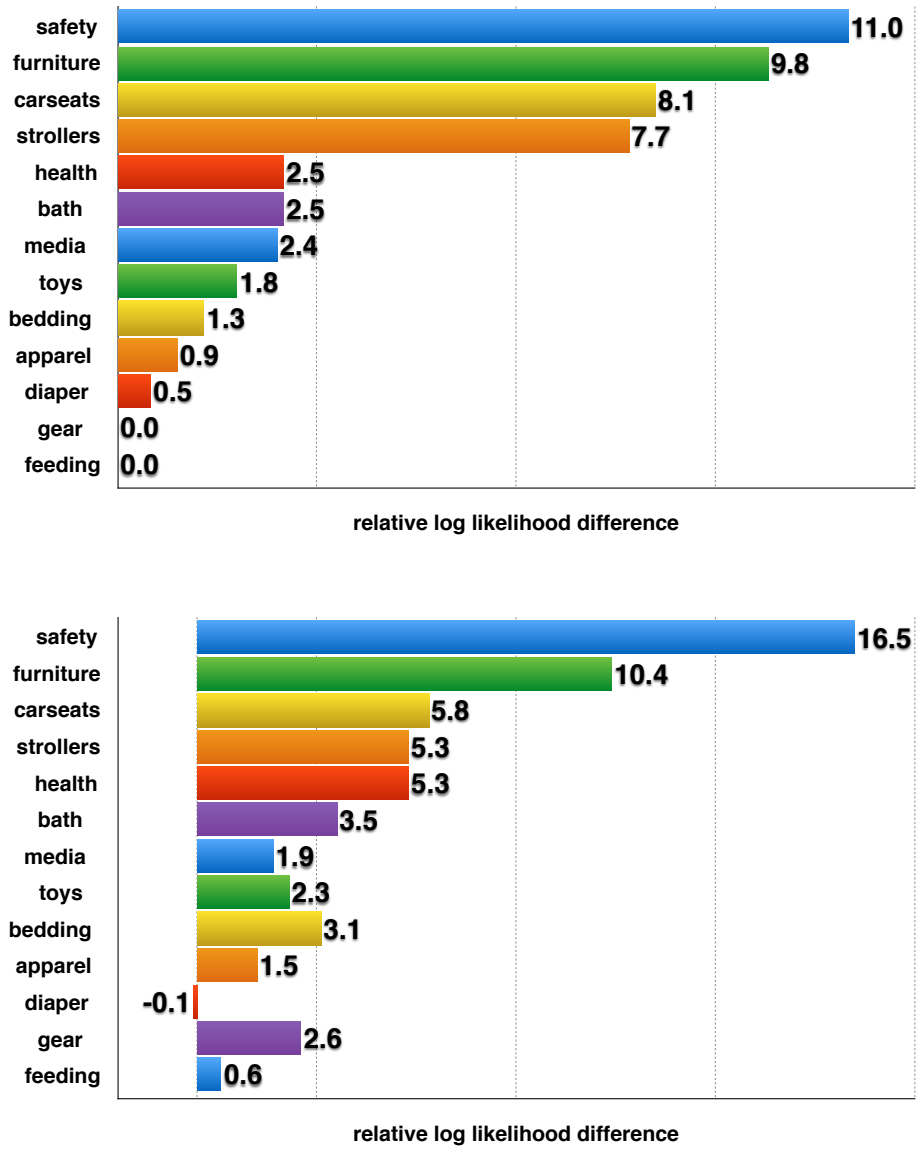


Figure 6.2: Relative test log-likelihood differences, $100\frac{(\text{EM}-\text{KA})}{|\text{KA}|}$, using: Wishart initialization in the full-data setting (top), and moments-matching initialization in the data-poor setting (bottom).

Category	**Wishart**			**Moments** (all data)	**Moments** (less data)		
safety	(10.88)	11.05	(11.12)	-0.13	(10.19)	16.53	(19.46)
furniture	(9.80)	9.89	(10.07)	0.23	(8.00)	10.47	(13.57)
carseats	(8.06)	8.16	(8.31)	0.61	(3.40)	5.85	(8.28)
strollers	(7.66)	7.77	(7.88)	-0.07	(2.51)	5.35	(7.41)
health	(2.50)	2.54	(2.58)	1.37	(2.67)	5.36	(6.03)
bath	(2.50)	2.54	(2.59)	-0.24	(2.22)	3.56	(4.23)
media	(2.37)	2.42	(2.49)	-0.17	(0.44)	1.93	(2.77)
toys	(1.76)	1.80	(1.83)	0.13	(1.01)	2.39	(4.30)
bedding	(0.42)	1.34	(1.44)	2.81	(2.44)	3.19	(3.70)
apparel	(0.88)	0.92	(0.93)	0.53	(0.78)	1.59	(2.23)
diaper	(0.50)	0.58	(1.02)	-0.47	(-0.87)	-0.19	(1.26)
gear	(0.03)	0.05	(0.07)	0.86	(1.36)	2.63	(3.22)
feeding	(-0.11)	-0.09	(-0.07)	-0.03	(-1.32)	0.61	(1.22)
average		3.76		0.41		4.55	

Table 6.2: Relative test log-likelihood differences, $100\frac{(\text{EM}-\text{KA})}{|\text{KA}|}$, for three cases: a Wishart initialization, a moments-matching initialization, and a moments-matching initialization in a low-data setting (only $n = 2N$ examples in the training set). For the first and third settings there is some variability: in the first setting, because the starting matrix drawn from the Wishart can vary; in the third setting, because the training examples (drawn from the full training set used in the other two settings) can vary. Thus, for these two settings the numbers in parentheses give the first and third quartiles over 25 trials.

Figure 6.3: A high-probability set of size $k = 10$ selected using an EM model for the "safety" category.

KA selection differs from Figure 6.3 in that it replaces the lens filters and the head support with two additional baby monitors: "Motorola MBP36 Remote Wireless Video Baby Monitor", and "Summer Infant Baby Touch Digital Color Video Monitor". It seems unlikely that many consumers would select three different brands of video monitor.

Having established that EM is more robust than KA, we conclude with an analysis of runtimes. Figure 6.4 shows the ratio of KA's runtime to EM's for each category. As discussed earlier, EM is asymptotically faster than KA, and we see this borne out in practice even for the moderate values of N and n that occur in our registries dataset: on average, EM is 2.1 times faster than KA.

In summary, we have explored in this section learning DPPs in a setting where the kernel K is not assumed to have fixed values or a restrictive parametric form. By exploiting K's eigendecomposition, we were able to develop a novel EM learning algorithm. On a product recommendation task, we have shown EM to be faster and more robust than the naive approach of maximizing likelihood by projected gradient. In other applications for which modeling negative interactions between items is important, we anticipate that EM will similarly have a significant advantage.

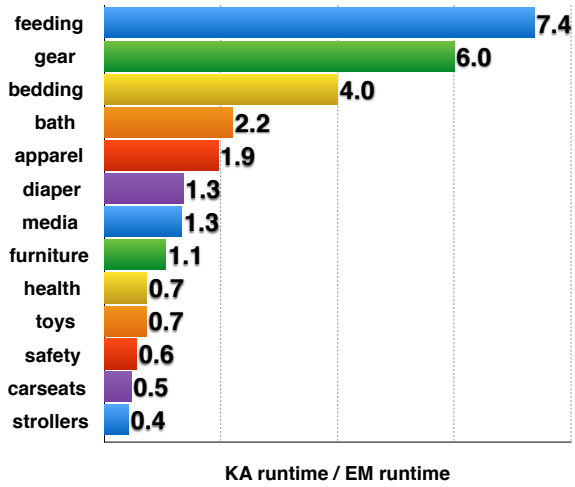


Figure 6.4: Ratios of KA runtime to EM runtime.

6.7.2 Exponentiated gradient

Besides the EM method developed in the preceding sections, another learning method that improves on the projected gradient ascent of KA is exponentiated gradient. Tsuda, Rätsch, and Warmuth (2005) provide several exponeniated gradient update schemes for positive semi-definite matrices, and we briefly discuss one of these here.

One way of viewing the updates of the gradient ascent method is as a solution to the following minimization problem:

$$K' = \arg\min_{K'} \|K' - K\|_F^2 - \eta \mathcal{L}(K') \,. \tag{6.65}$$

This minimization balances the goal of maximizing likelihood with the goal of minimizing change in K, where the learning rate η quantifies the relative importances of these two goals. Taking the derivative, setting it equal to zero, and solving for K':

$$K' = K + \frac{\eta}{2} \frac{\partial \mathcal{L}(K')}{\partial K'} \,. \tag{6.66}$$

This is identical to the gradient update once we make the approximation $\frac{\partial \mathcal{L}(K')}{\partial K'} \approx \frac{\partial \mathcal{L}(K)}{\partial K}$. Tsuda et al. (2005) develop an exponentiated gradient update rule by replacing the Frobenius norm in Equation (6.65) with a different measure of similarity between K and K'. Specifically, Tsuda et al. (2005) use the von Neumann divergence:

$$\mathrm{tr}(K' \log K' - K' \log K - K' + K) \,. \tag{6.67}$$

The solution to the resulting optimization problem (again approximating $\frac{\partial \mathcal{L}(K')}{\partial K'}$ with $\frac{\partial \mathcal{L}(K)}{\partial K}$) is:

$$K' = \exp\left[\log K + \eta \frac{\partial \mathcal{L}(K)}{\partial K}\right] . \tag{6.68}$$

Notice that the matrix exponential here implies that K' will have non-negative eigenvalues. Thus, the only projection necessary to enforce the DPP kernel learning constraints is to cap the eigenvalues at 1. This eliminates the type of problems we observed with the naïve projected gradient method, KA, where too many eigenvalues were truncated at zero. If we apply this exponentiated update rule to the problem of learning DPP kernels for the baby registry data, its performance is nearly identical to that of EM: with Wishart initialization the relative gain in log-likelihood of EM over exponentiated gradient averages 0.57%, and for the moments-matching initialization it averages -0.18%.

7

Conclusion

In this thesis we have developed a set of approximate inference tools for determinantal point processes (DPPs), and demonstrated their efficacy on a variety of real-world applications.

In Chapter 4, we brought standard dimensionality reduction techniques to bear on the problem of DPP inference, in the setting where both the number of items in the ground set and the number of features of those items are large. We established that, when employing random projections to reduce the number of features of a DPP, the variational distance between the original DPP and the projected one decays exponentially as the number of projection dimensions increases. This granted theoretical justification to the use of random projections with DPPs, and motivated our use of them in combination with structured DPPs. The resulting partnership allowed us to tackle a novel summarization task for large document corpora.

In Chapter 5, we considered the NP-hard problem of finding the DPP MAP configuration. Observing that det is a log-submodular function, we leveraged existing submodular maximization algorithms. Introducing a modification to the standard submodular multilinear extension, we obtained an alternative extension whose value and gradient can be computed efficiently for det, while maintaining the par-

tial concavity necessary for proving a $\frac{1}{4}$-approximation guarantee for the associated algorithm. We verified the practicality of the algorithm, and illustrated its ability to accommodate interesting constraints on the selected subset, by testing on a political candidate comparison task.

In Chapter 6, we made a foray into the domain of DPP learning. To address the problem of learning the full DPP kernel, likely an NP-hard task, we proposed a novel algorithm for locating local optima of the likelihood objective. More precisely, by decomposing the likelihood objective into eigenvalue and eigenvector components and then lower-bounding it using Jensen's inequality, we derived an expectation-maximization (EM) scheme. Our experiments on a practical product recommendation task indicate that this EM algorithm is more robust than a naïve projected gradient approach.

7.1 Future work

Given the relatively recent appearance of DPPs on the machine learning front, there remain substantial opportunities for innovation. As illustrated in this thesis, some core DPP inference tasks are NP-hard, and other tasks' complexities are cubic in the size of the ground set, which is impractical for the large datasets that are of interest to the machine learning community. Thus, many of the opportunities for innovation lie in the realm of approximate inference. We briefly touch here on some such possibilities for future work.

In the area of dimensionality reduction, there are numerous more sophisticated techniques than random projections that one could try to adapt for use with DPPs. While the computational complexity of the vanilla versions of most methods, such as principal component analysis, is too high for use in settings with a large number of items and features, finding ways to approximate these techniques for use with DPPs would be of interest. For instance, the dual and structured versions of the Nyström approximation outlined in Sections 4.6.2 and 4.6.2 might be a good starting point for future work.

For the problem of finding the DPP MAP set, the question of how hard it is to approximate the MAP under complex constraint sets (matroids, knapsacks, etc.) remains open. While the submodular maximization literature gives a starting point

for this problem, there are likely algorithms that do not generalize to all submodular functions, yet work well with DPPs. Exploiting those inference operations that can be performed efficiently for DPPs, such as normalization and marginalization, is a good starting point for developing such algorithms.

In terms of learning DPPs, there are as yet a vast number of unexplored options: likelihood maximization via other local search techniques such as the difference of convex functions method, entropy maximization subject to moment-matching constraints, incorporation of alternative forms of labeled data, and learning in an online setting, to name a few.

Beyond dimensionality reduction and the MAP and learning problems, there are several other prominent avenues open for inquiry. For instance, there is the question of how to do structured DPP inference on loopy graphs. Additionally, there is the question of how to modify DPP inference algorithms when DPPs are combined with other data models, such as graphical models. As we seek to apply DPPs to more and more complex tasks, such questions become increasingly important.

Bibliography

University of California at Irvine Machine Learning Repository. http://archive.ics.uci.edu/ml/. [Cited on page 76.]

D. Achlioptas. Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins. *Journal of Computer and System Sciences*, 2002. [Cited on pages 68 and 69.]

R. Affandi, A. Kulesza, and E. Fox. Markov Determinantal Point Processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012. [Cited on pages 42 and 43.]

R. Affandi, E. Fox, and B. Taskar. Approximate Inference in Continuous Determinantal Point Processes. In *Neural Information Processing Systems (NIPS)*, 2013a. [Cited on pages 44, 45, and 76.]

R. Affandi, A. Kulesza, E. Fox, and B. Taskar. Nyström Approximation for Large-Scale Determinantal Processes. In *Conference on Artificial Intelligence and Statistics (AIStats)*, 2013b. [Cited on pages 4, 70, 74, 75, 76, 77, 78, and 79.]

R. Affandi, E. Fox, R. Adams, and B. Taskar. Learning the Parameters of Determinantal Point Process Kernels. In *International Conference on Machine Learning (ICML)*, 2014. [Cited on pages 110, 114, 118, and 119.]

A. Ageev and M. Sviridenko. Pipage Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee. *Journal of Combinatorial Optimization*, 8(3), 2004. [Cited on page 102.]

A. Ahmed and E. Xing. Timeline: A Dynamic Hierarchical Dirichlet Process Model for Recovering Birth/Death and Evolution of Topics in Text Stream. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010. [Cited on page 56.]

N. Ailon and B. Chazelle. The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM Journal on Computing (SICOMP)*, 2009. [Cited on pages 68 and 69.]

J. Allan, R. Gupta, and V. Khandelwal. Temporal Summaries of New Topics. In *Conference of the Special Interest Group on Information Retrieval (SIGIR)*, 2001. [Cited on page 56.]

F. Bach. Learning with Submodular Functions: A Convex Optimization Perspective. Technical Report HAL 00645271-v2, Institut national de recherche en informatique et en automatique (INRIA), 2013. [Cited on page 83.]

F. Baker and M. Harwell. Computing Elementary Symmetric Functions and Their Derivatives: A Didactic. *Applied Psychological Measurement*, 20:169–192, 1996. [Cited on pages 38 and 126.]

D. Bertsekas. *Nonlinear Programming*. 1999. ISBN 9781886529144. [Cited on page 97.]

D. Blei and J. Lafferty. Dynamic Topic Models. In *International Conference on Machine Learning (ICML)*, 2006. [Cited on pages 56 and 62.]

A. Borodin and E. Rains. Eynard-Mehta Theorem, Schur Process, and Their Pfaffian Analogs. *Journal of Statistical Physics*, 121:291–317, 2005. [Cited on page 7.]

N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization. In *Foundations of Computer Science (FOCS)*, 2012. [Cited on pages 82, 89, 90, and 105.]

N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. Submodular Maximization with Cardinality Constraints. In *Symposium on Discrete Algorithms (SODA)*, 2014. [Cited on page 90.]

J. Bunch and J. Hopcroft. Triangular Factorization and Inversion by Fast Matrix Multiplication. *Mathematics of Computation*, 28(125):231–236, 1974. [Cited on pages 15 and 17.]

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a Submodular Set Function Subject to a Matroid Constraint. In *Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2007. [Cited on page 94.]

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM Journal on Computing (SICOMP)*, 40:1740–1766, 2011. [Cited on page 102.]

A. Çivril and M. Magdon-Ismail. On Selecting a Maximum Volume Sub-Matrix of a Matrix and Related Problems. *Theoretical Computer Science*, 410(47-49):4801–4811, 2009. [Cited on pages 27 and 81.]

C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes. In *Symposium on the Theory of Computing (STOC)*, 2011. [Cited on pages 87, 90, 91, 92, 94, 97, 98, 99, 100, 101, 102, and 103.]

H. Chieu and Y. Lee. Query Based Event Extraction along a Timeline. In *Conference of the Special Interest Group on Information Retrieval (SIGIR)*, 2004. [Cited on page 57.]

A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix Approximation and Projective Clustering via Volume Sampling. *Theory of Computing*, 2:225–247, 2006. [Cited on page 75.]

S. Dobzinski and J. Vondrák. From Query Complexity to Computational Complexity. In *Symposium on the Theory of Computing (STOC)*, 2012. [Cited on page 89.]

P. Drineas and M. Mahoney. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Journal of Machine Learning Research (JMLR)*, 6:2153–2175, 2005. [Cited on page 74.]

S. Dughmi, T. Roughgarden, and M. Sundararajan. Revenue Submodularity. In *Electronic Commerce*, 2009. [Cited on page 3.]

A. Edelman, T. Arias, and S. Smith. The Geometry of Algorithms with Orthogonality Constraints. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 1998. [Cited on page 130.]

E. Elhamifar, G. Sapiro, and R. Vidal. Finding Exemplars from Pairwise Dissimilarities via Simultaneous Sparse Recovery. In *Neural Information Processing Systems (NIPS)*, 2012. [Cited on page 4.]

G. Erkan and D. Radev. LexRank: Graph-Based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479, 2004. [Cited on page 58.]

U. Feige, V. Mirrokni, and J. Vondrák. Maximizing Non-Monotone Submodular Functions. In *Foundations of Computer Science (FOCS)*, 2007. [Cited on pages 88 and 89.]

M. Feldman, J. Naor, and R. Schwartz. Nonmonotone Submodular Maximization via a Structural Continuous Greedy Algorithm. *Automata, Languages, and Programming*, 2011. [Cited on page 92.]

M. Frank and P. Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, 3(1-2), 1956. [Cited on page 97.]

S. Gharan and J. Vondrák. Submodular Maximization by Simulated Annealing. In *Symposium on Discrete Algorithms (SODA)*, 2011. [Cited on page 90.]

J. Gillenwater, A. Kulesza, and B. Taskar. Discovering Diverse and Salient Threads in Document Collections. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2012a. [Cited on page 47.]

J. Gillenwater, A. Kulesza, and B. Taskar. Near-Optimal MAP Inference for Determinantal Point Processes. In *Neural Information Processing Systems (NIPS)*, 2012b. [Cited on page 83.]

J. Gillenwater, A. Kulesza, E. Fox, and B. Taskar. Expectation-Maximization for Learning Determinantal Point Processes. In *Neural Information Processing Systems (NIPS)*, 2014. [Cited on pages 2 and 110.]

M. Goemans and D. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the Association for Computing Machinery (JACM)*, 42:1115–1145, 1995. [Cited on page 86.]

D. Graff and Cieri C. English Gigaword, 2009. [Cited on page 59.]

M. Gu and S. Eisenstat. A Divide-and-Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 16(1):172–191, 1995. [Cited on page 17.]

A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained Nonmonotone Submodular Maximization: Offline and Secretary Algorithms. In *Conference on Web and Internet Economics (WINE)*, 2010. [Cited on page 90.]

I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research (JMLR)*, 2003. [Cited on page 4.]

J. Hartline, V. Mirrokni, and M. Sundararajan. Optimal Marketing Strategies over Social Networks. In *World Wide Web Conference (WWW)*, 2008. [Cited on page 3.]

D. Henrion and J. Malick. Projection Methods for Conic Feasibility Problems. *Optimization Methods and Software*, 26(1):23–46, 2011. [Cited on page 121.]

T. Hesterberg, S. Monaghan, D. Moore, A. Clipson, and R. Epstein. *Bootstrap Methods and Permutation Tests*. 2003. ISBN 9780716759980. [Cited on page 66.]

J. Hough, M. Krishnapur, Y. Peres, and B. Virág. Determinantal Processes and Independence. *Probability Surveys*, 3, 2006. [Cited on pages 22 and 123.]

A. James. Distributions of Matrix Variates and Latent Roots Derived from Normal Samples. *Annals of Mathematical Statistics*, 35(2):475–501, 1964. [Cited on pages 104 and 131.]

W. Johnson and J. Lindenstrauss. Extensions of Lipschitz Mappings into a Hilbert Space. *Contemporary Mathematics*, 26:189–206, 1984. [Cited on page 47.]

B. Kang. Fast Determinantal Point Process Sampling with Application to Clustering. In *Neural Information Processing Systems (NIPS)*, 2013. [Cited on pages 4, 69, 70, 71, and 72.]

G. Kim, E. Xing, L. Fei-Fei, and T. Kanade. Distributed Cosegmentation via Submodular Optimization on Anisotropic Diffusion. In *International Conference on Computer Vision (ICCV)*, 2011. [Cited on page 3.]

C. Ko, J. Lee, and M. Queyranne. An Exact Algorithm for Maximum Entropy Sampling. *Operations Research*, 43(4):684–691, 1995. [Cited on pages 26, 27, and 81.]

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 (JMLR)*, 9:235–284, 2008. [Cited on page 3.]

A. Kulesza. *Learning with Determinantal Point Processes*. PhD thesis, University of Pennsylvania, 2012. [Cited on pages 12, 15, 16, 17, 18, 21, 22, 27, 28, 29, 31, 32, 34, 35, 38, 39, 42, 44, 80, 82, and 117.]

A. Kulesza and B. Taskar. Structured Determinantal Point Processes. In *Neural Information Processing Systems (NIPS)*, 2010. [Cited on pages 4, 41, and 50.]

A. Kulesza and B. Taskar. k-DPPs: Fixed-Size Determinantal Point Processes. In *International Conference on Machine Learning (ICML)*, 2011a. [Cited on pages 3, 38, 50, 110, 114, 115, 116, and 119.]

A. Kulesza and B. Taskar. Learning Determinantal Point Processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011b. [Cited on pages 43, 110, 114, and 115.]

A. Kulesza and B. Taskar. Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*, 5(2-3), 2012. [Cited on page 2.]

G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan. Learning the Kernel Matrix with Semidefinite Programming. *Journal of Machine Learning Research (JMLR)*, 5:27–72, 2004. [Cited on page 111.]

J. Lasserre. *Moments, Positive Polynomials, and Their Applications*. 2009. ISBN 9781848164468. [Cited on page 113.]

S. Lauritzen and D. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society*, page 157–224, 1988. [Cited on page 41.]

F. Lavancier, J. Møller, and E. Rubak. Statistical Aspects of Determinantal Point Processes. Technical Report R-2012-02, Department of Mathematical Sciences, Aalborg University, 2012. [Cited on page 118.]

H. Lee, E. Modiano, and K. Lee. Diverse Routing in Networks With Probabilistic Failures. *IEEE Transactions on Networking*, 18(6), 2010. [Cited on page 4.]

J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the Dynamics of the News Cycle. In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2009. [Cited on page 56.]

D. Levin, Y. Peres, and E. Wilmer. *Markov Chains and Mixing Times*. 2008. ISBN 9780821886274. [Cited on page 73.]

E. Levitin and B. Polyak. Constrained Minimization Methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966. [Cited on page 121.]

M. Li, J. Kwok, and B. Lu. Making Large-Scale Nyström Approximation Possible. In *International Conference on Machine Learning (ICML)*, 2010. [Cited on page 78.]

Z. Li and J. Eisner. First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2009. [Cited on page 42.]

C. Lin. ROUGE: A package for automatic evaluation of summaries. In *Workshop on Text Summarization (WAS)*, 2004. [Cited on page 65.]

H. Lin and J. Bilmes. Learning Mixtures of Submodular Shells with Application to Document Summarization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012. [Cited on page 2.]

R. Lyons. Determinantal Probability Measures. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 98(1):167–212, 2003. [Cited on pages 29 and 113.]

A. Magen and A. Zouzias. Near Optimal Dimensionality Reductions that Preserve Volumes. *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques (APPROX / RANDOM)*, 5171:523–534, 2008. [Cited on pages 47 and 69.]

A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval Journal*, 3:127–163, 2000. [Cited on page 59.]

K. McKinnon. Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization (SIOPT)*, 9(1):148–158, 1998. [Cited on page 118.]

D. McSherry. Diversity-Conscious Retrieval. In *Advances in Case-Based Reasoning*, 2002. [Cited on page 2.]

W. Mei and C. Zhai. Discovering Evolutionary Theme Patterns From Text: An Exploration of Temporal Text Mining. In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2005. [Cited on page 56.]

C. Meyer. *Matrix Analysis and Applied Linear Algebra*. 2000. ISBN 9780898714548. [Cited on page 13.]

R. Neal and G. Hinton. A New View of the EM Algorithm that Justifies Incremental, Sparse and Other Variants. *Learning in Graphical Models*, 1998. [Cited on page 123.]

G. Nemhauser, L. Wolsey, and M. Fisher. An Analysis of Approximations for Maximizing Submodular Set Functions I. *Mathematical Programming*, 14(1), 1978. [Cited on pages 27, 82, 87, and 88.]

J. Nocedal and S. Wright. *Numerical Optimization*. 2006. ISBN 9780387303031. [Cited on page 97.]

K. Petersen and M. Pedersen. The Matrix Cookbook. Technical report, Technical University of Denmark, 2012. [Cited on pages 96, 121, and 127.]

R. Reichart and A. Korhonen. Improved Lexical Acquisition through DPP-based Verb Clustering. In *Conference of the Association for Computational Linguistics (ACL)*, 2013. [Cited on page 4.]

J. Rising, A. Kulesza, and B. Taskar. An Efficient Algorithm for the Symmetric Principal Minor Assignment Problem. *Linear Algebra and its Applications*, 2014. [Cited on page 112.]

A. Shah and Z. Ghahramani. Determinantal Clustering Process — A Nonparametric Bayesian Approach to Kernel Based Semi-Supervised Clustering. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013. [Cited on page 4.]

D. Shahaf and C. Guestrin. Connecting the Dots Between News Articles. In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2010. [Cited on page 57.]

D. Shahaf, C. Guestrin, and E. Horvitz. Trains of Thought: Generating Information Maps. In *World Wide Web Conference (WWW)*, 2012. [Cited on page 57.]

R. Swan and D. Jensen. TimeMines: Constructing Timelines with Statistical Models of Word Usage. In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2000. [Cited on page 56.]

K. Tsuda, G. Rätsch, and M. Warmuth. Matrix Exponentiated Gradient Updates for On-line Learning and Bregman Projection. *Journal of Machine Learning Research (JMLR)*, 6:995–1018, 2005. [Cited on page 137.]

L. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing (SICOMP)*, 8:410–421, 1979. [Cited on page 30.]

L. Valiant and V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theoretical Computer Science*, 47:85–93, 1986. [Cited on page 89.]

J. Vondrák. Optimal Approximation for the Submodular Welfare Problem in the Value Oracle Model. In *Symposium on the Theory of Computing (STOC)*, 2008. [Cited on page 91.]

J. Vondrák. Symmetry and Approximability of Submodular Maximization Problems. In *Foundations of Computer Science (FOCS)*, 2009. [Cited on page 91.]

C. Wayne. Multilingual Topic Detection and Tracking: Successful Research Enabled by Corpora and Evaluation. In *Language Resources and Evaluation Conference (LREC)*, 2000. [Cited on page 56.]

E. Xing, A. Ng, M. Jordan, and S. Russell. Distance Metric Learning with Application to Clustering with Side-Information. In *Neural Information Processing Systems (NIPS)*, 2002. [Cited on page 111.]

R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang. Evolutionary Timeline Summarization: A Balanced Optimization Framework via Iterative Substitution. In *Conference of the Special Interest Group on Information Retrieval (SIGIR)*, 2011. [Cited on page 56.]

H. Yanai, K. Takeuchi, and Y. Takane. *Projection Matrices, Generalized Inverse Matrices, and Singular Value Decomposition*. 2011. ISBN 9781441998873. [Cited on page 78.]

J. Zou and R. Adams. Priors for Diversity in Generative Latent Variable Models. In *Neural Information Processing Systems (NIPS)*, 2013. [Cited on page 45.]