Foundations and Trends® in
Machine Learning
Vol. 5, Nos. 2–3 (2012) 123–286

DOI: 10.1561/2200000044

now
the essence of knowledge

Determinantal Point Processes for Machine Learning

By Alex Kulesza and Ben Taskar

Contents

Foundations and Trends® in
Machine Learning
Vol. 5, Nos. 2–3 (2012) 123–286

DOI: 10.1561/2200000044

now
the essence of knowledge

Determinantal Point Processes for Machine Learning

Alex Kulesza[1] and Ben Taskar[2]

[1] *University of Michigan, USA, kulesza@umich.edu*
[2] *University of Pennsylvania, USA, taskar@cis.upenn.edu*

Abstract

Determinantal point processes (DPPs) are elegant probabilistic models of repulsion that arise in quantum physics and random matrix theory. In contrast to traditional structured models like Markov random fields, which become intractable and hard to approximate in the presence of negative correlations, DPPs offer efficient and exact algorithms for sampling, marginalization, conditioning, and other inference tasks. We provide a gentle introduction to DPPs, focusing on the intuitions, algorithms, and extensions that are most relevant to the machine learning community, and show how DPPs can be applied to real-world applications like finding diverse sets of high-quality search results, building informative summaries by selecting diverse sentences from documents, modeling nonoverlapping human poses in images or video, and automatically building timelines of important news stories.

1

Introduction

Probabilistic modeling and learning techniques have become indispensable tools for analyzing data, discovering patterns, and making predictions in a variety of real-world settings. In recent years, the widespread availability of both data and processing capacity has led to new applications and methods involving more complex, structured output spaces, where the goal is to simultaneously make a large number of interrelated decisions. Unfortunately, the introduction of structure typically involves a combinatorial explosion of output possibilities, making inference computationally impractical without further assumptions.

A popular compromise is to employ graphical models, which are tractable when the graph encoding local interactions between variables is a tree. For loopy graphs, inference can often be approximated in the special case when the interactions between variables are positive and neighboring nodes tend to have the same labels. However, dealing with global, negative interactions in graphical models remain intractable, and heuristic methods often fail in practice.

Determinantal point processes (DPPs) offer a promising and complementary approach. Arising in quantum physics and random matrix

theory, DPPs are elegant probabilistic models of global, negative correlations, and offer efficient algorithms for sampling, marginalization, conditioning, and other inference tasks. While they have been studied extensively by mathematicians, giving rise to a deep and beautiful theory, DPPs are relatively new in machine learning. We aim to provide a comprehensible introduction to DPPs, focusing on the intuitions, algorithms, and extensions that are most relevant to our community.

1.1 Diversity

A DPP is a distribution over subsets of a fixed ground set, for instance, sets of search results selected from a large database. Equivalently, a DPP over a ground set of N items can be seen as modeling a binary characteristic vector of length N. The essential characteristic of a DPP is that these binary variables are negatively correlated; that is, the inclusion of one item makes the inclusion of other items less likely. The strengths of these negative correlations are derived from a kernel matrix that defines a global measure of similarity between pairs of items, so that more similar items are less likely to co-occur. As a result, DPPs assign higher probability to sets of items that are *diverse*; for example, a DPP will prefer search results that cover multiple distinct aspects of a user's query, rather than focusing on the most popular or salient one.

This focus on diversity places DPPs alongside a number of recently developed techniques for working with diverse sets, particularly in the information retrieval community [23, 26, 121, 122, 140, 158, 159]. However, unlike these methods, DPPs are fully probabilistic, opening the door to a wider variety of potential applications, without compromising algorithmic tractability.

The general concept of diversity can take on a number of forms depending on context and application. Including multiple kinds of search results might be seen as *covering* or *summarizing* relevant interpretations of the query or its associated topics; see Figure 1.1. Alternatively, items inhabiting a continuous space may exhibit diversity as a result of *repulsion*, as in Figure 1.2. In fact, certain repulsive quantum particles are known to be precisely described by a DPP; however, a DPP can also serve as a model for general repulsive phenomena, such

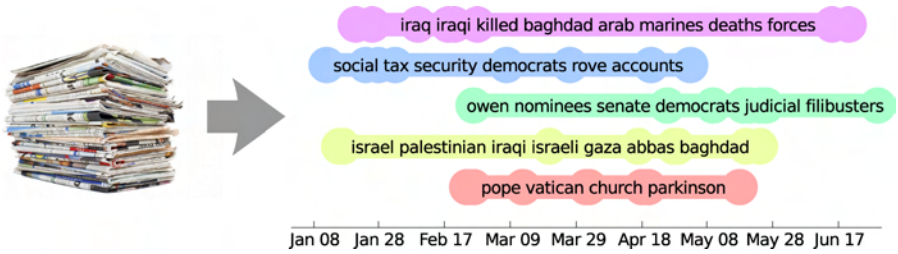


Fig. 1.1 Diversity is used to generate a set of summary timelines describing the most important events from a large news corpus.

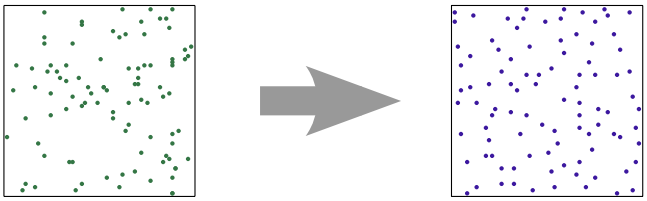


Fig. 1.2 On the left, points are sampled randomly; on the right, repulsion between points leads to the selection of a diverse set of locations.

Fig. 1.3 On the left, the output of a human pose detector is noisy and uncertain; on the right, applying diversity as a filter leads to a clean, separated set of predictions.

as the locations of trees in a forest, which appear diverse due to physical and resource constraints. Finally, diversity can be used as a *filtering* prior when multiple selections must be based on a single detector or scoring metric. For instance, in Figure 1.3 a weak pose detector favors large clusters of poses that are nearly identical, but filtering through a DPP ensures that the final predictions are well separated.

Throughout this survey we demonstrate applications for DPPs in a variety of settings, including:

- The DUC 2003/2004 text summarization task, where we form extractive summaries of news articles by choosing diverse subsets of sentences (Section 4.2.1);

- An image search task, where we model human judgments of diversity for image sets returned by Google Image Search (Section 5.3),
- A multiple pose estimation task, where we improve the detection of human poses in images from television shows by incorporating a bias toward nonoverlapping predictions (Section 6.4), and
- A news threading task, where we automatically extract timelines of important news stories from a large corpus by balancing intra-timeline coherence with inter-timeline diversity (Section 6.6.4).

1.2 Outline

In this monograph we present general mathematical background on DPPs along with a range of modeling extensions, efficient algorithms, and theoretical results that aim to enable practical modeling and learning. The material is organized as follows.

Section 2: Determinantal Point Processes. We begin with an introduction to determinantal point processes tailored to the interests of the machine learning community. We focus on discrete DPPs, emphasizing intuitions and including new, simplified proofs for some theoretical results. We provide descriptions of known efficient inference algorithms and characterize their computational properties.

Section 3: Representation and Algorithms. We describe a decomposition of the DPP that makes explicit its fundamental trade-off between quality and diversity. We compare the expressive power of DPPs and MRFs, characterizing the trade-offs in terms of modeling power and computational efficiency. We also introduce a dual representation for DPPs, showing how it can be used to perform efficient inference over large ground sets. When the data are high-dimensional and dual inference is still too slow, we show that random projections can be used to maintain a provably close approximation to the original model while greatly reducing computational requirements.

Section 4: Learning. We derive an efficient algorithm for learning the parameters of a quality model when the diversity model is held fixed. We employ this learning algorithm to perform extractive summarization of news text.

Section 5: k-DPPs. We present an extension of DPPs that allows for explicit control over the number of items selected by the model. We show not only that this extension solves an important practical problem, but also that it increases expressive power: a k-DPP can capture distributions that a standard DPP cannot. The extension to k-DPPs necessitates new algorithms for efficient inference based on recursions for the elementary symmetric polynomials. We validate the new model experimentally on an image search task.

Section 6: Structured DPPs. We extend DPPs to model diverse sets of structured items, such as sequences or trees, where there are combinatorially many possible configurations. In this setting the number of possible subsets is doubly exponential, presenting a daunting computational challenge. However, we show that a factorization of the quality and diversity models together with the dual representation for DPPs makes efficient inference possible using second-order message passing. We demonstrate structured DPPs on a toy geographical paths problem, a still-image multiple pose estimation task, and two high-dimensional text threading tasks.

2

Determinantal Point Processes

Determinantal point processes (DPPs) were first identified as a class by Macchi [98], who called them "fermion processes" because they give the distributions of fermion systems at thermal equilibrium. The Pauli exclusion principle states that no two fermions can occupy the same quantum state; as a consequence fermions exhibit what is known as the "antibunching" effect. This repulsion is described precisely by a DPP.

In fact, years before Macchi gave them a general treatment, specific DPPs appeared in major results in random matrix theory [40, 41, 42, 58, 104], where they continue to play an important role [36, 75]. Recently, DPPs have attracted a flurry of attention in the mathematics community [13, 14, 15, 16, 21, 72, 73, 74, 116, 117, 131], and much progress has been made in understanding their formal combinatorial and probabilistic properties. The term "determinantal" was first used by Borodin and Olshanski [14], and has since become accepted as standard. Many good mathematical surveys are now available [12, 68, 97, 132, 133, 137, 145].

We begin with an overview of the aspects of DPPs most relevant to the machine learning community, emphasizing intuitions, algorithms, and computational properties.

2.1 Definition

A point process $\mathcal{P}$ on a ground set $\mathcal{Y}$ is a probability measure over "point patterns" or "point configurations" of $\mathcal{Y}$, which are finite subsets of $\mathcal{Y}$. For instance, $\mathcal{Y}$ could be a continuous time interval during which a scientist records the output of a brain electrode, with $\mathcal{P}(\{y_1, y_2, y_3\})$ characterizing the likelihood of seeing neural spikes at times y_1, y_2, and y_3. Depending on the experiment, the spikes might tend to cluster together, or they might occur independently, or they might tend to spread out in time. $\mathcal{P}$ captures these correlations.

For the remainder of this monograph, we will focus on discrete, finite point processes, where we assume without loss of generality that $\mathcal{Y} = \{1, 2, \ldots, N\}$; in this setting we sometimes refer to elements of $\mathcal{Y}$ as *items*. Much of our discussion extends to the continuous case, but the discrete setting is computationally simpler and often more appropriate for real-world data — e.g., in practice, the electrode voltage will only be sampled at discrete intervals. The distinction will become even more apparent when we apply our methods to $\mathcal{Y}$ with no natural continuous interpretation, such as the set of documents in a corpus.

In the discrete case, a point process is simply a probability measure on $2^{\mathcal{Y}}$, the set of all subsets of $\mathcal{Y}$. A sample from $\mathcal{P}$ might be the empty set, the entirety of $\mathcal{Y}$, or anything in between. $\mathcal{P}$ is called a *determinantal point process* if, when $\boldsymbol{Y}$ is a random subset drawn according to $\mathcal{P}$, we have, for every $A \subseteq \mathcal{Y}$,

$$\mathcal{P}(A \subseteq \boldsymbol{Y}) = \det(K_A) \tag{2.1}$$

for some real, symmetric $N \times N$ matrix K indexed by the elements of $\mathcal{Y}$.[1] Here, $K_A \equiv [K_{ij}]_{i,j \in A}$ denotes the restriction of K to the entries indexed by elements of A, and we adopt $\det(K_\emptyset) = 1$. Note that normalization is unnecessary here, since we are defining marginal probabilities that need not sum to 1.

Since $\mathcal{P}$ is a probability measure, all principal minors $\det(K_A)$ of K must be nonnegative, and thus K itself must be positive semidefinite. It is possible to show in the same way that the eigenvalues of K are

[1] In general, K need not be symmetric. However, in the interest of simplicity, we proceed with this assumption; it is not a significant limitation for our purposes.

bounded above by one using Equation (2.27), which we introduce later. These requirements turn out to be sufficient: any K, $0 \preceq K \preceq I$, defines a DPP. This will be a consequence of Theorem 2.3.

We refer to K as the *marginal kernel* since it contains all the information needed to compute the probability of any subset A being included in $\boldsymbol{Y}$. A few simple observations follow from Equation (2.1). If $A = \{i\}$ is a singleton, then we have

$$\mathcal{P}(i \in \boldsymbol{Y}) = K_{ii}. \tag{2.2}$$

That is, the diagonal of K gives the marginal probabilities of inclusion for individual elements of $\mathcal{Y}$. Diagonal entries close to 1 correspond to elements of $\mathcal{Y}$ that are almost always selected by the DPP. Furthermore, if $A = \{i, j\}$ is a two-element set, then

$$\begin{aligned} \mathcal{P}(i, j \in \boldsymbol{Y}) &= \begin{vmatrix} K_{ii} & K_{ij} \\ K_{ji} & K_{jj} \end{vmatrix} && (2.3) \\ &= K_{ii}K_{jj} - K_{ij}K_{ji} && (2.4) \\ &= \mathcal{P}(i \in \boldsymbol{Y})\mathcal{P}(j \in \boldsymbol{Y}) - K_{ij}^2. && (2.5) \end{aligned}$$

Thus, the off-diagonal elements determine the negative correlations between pairs of elements: large values of K_{ij} imply that i and j tend not to co-occur.

Equation (2.5) demonstrates why DPPs are "diversifying". If we think of the entries of the marginal kernel as measurements of similarity between pairs of elements in $\mathcal{Y}$, then highly similar elements are unlikely to appear together. If $K_{ij} = \sqrt{K_{ii}K_{jj}}$, then i and j are "perfectly similar" and do not appear together almost surely. Conversely, when K is diagonal there are no correlations and the elements appear independently. Note that DPPs cannot represent distributions where elements are *more* likely to co-occur than if they were independent: correlations are always nonpositive.

Figure 2.1 shows the difference between sampling a set of points in the plane using a DPP (with K_{ij} inversely related to the distance between points i and j), which leads to a relatively uniformly spread set with good coverage, and sampling points independently, which results in random clumping.

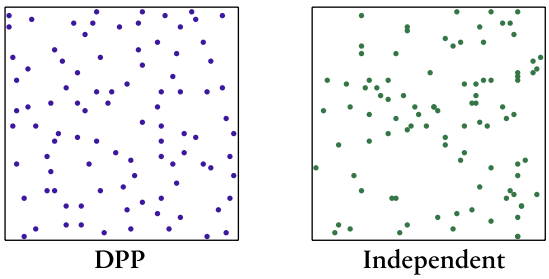


Fig. 2.1 A set of points in the plane drawn from a DPP (left), and the same number of points sampled independently using a Poisson point process (right).

2.1.1 Examples

In this monograph, our focus is on using DPPs to model real-world data. However, many theoretical point processes turn out to be exactly determinantal, which is one of the main reasons they have received so much recent attention. In this section we briefly describe a few examples; some of them are quite remarkable on their own, and as a whole they offer some intuition about the types of distributions that are realizable by DPPs. Technical details for each example can be found in the accompanying reference.

Descents in random sequences [13] Given a sequence of N random numbers drawn uniformly and independently from a finite set (say, the digits 0–9), the locations in the sequence where the current number is less than the previous number form a subset of $\{2, 3, \ldots, N\}$. This subset is distributed as a determinantal point process. Intuitively, if the current number is less than the previous number, it is probably not too large, thus it becomes less likely that the next number will be smaller yet. In this sense, the positions of decreases repel one another.

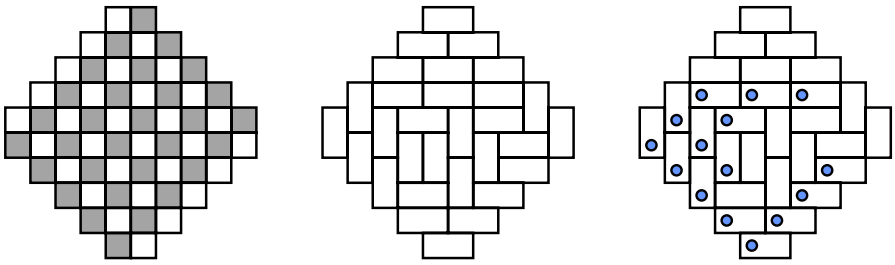
Nonintersecting random walks [73] Consider a set of k independent, simple, symmetric random walks of length T on the integers. That is, each walk is a sequence $x_1, x_2, \ldots, x_T$ where $x_i - x_{i+1}$ is either -1 or $+1$ with equal probability. If we let the walks begin at positions $x_1^1, x_1^2, \ldots, x_1^k$ and condition on the fact that they end at positions

$x_T^1, x_T^2, \ldots, x_T^k$ and do not intersect, then the positions $x_t^1, x_t^2, \ldots, x_t^k$ at any time t are a subset of $\mathbb{Z}$ and distributed according to a DPP. Intuitively, if the random walks do not intersect, then at any time step they are likely to be far apart.

Edges in random spanning trees [21] Let G be an arbitrary finite graph with N edges, and let T be a random spanning tree chosen uniformly from the set of all the spanning trees of G. The edges in T form a subset of the edges of G that is distributed as a DPP. The marginal kernel in this case is the transfer–impedance matrix, whose entry $K_{e_1 e_2}$ is the expected signed number of traversals of edge e_2 when a random walk begins at one endpoint of e_1 and ends at the other (the graph edges are first oriented arbitrarily). Thus, edges that are in some sense "nearby" in G are similar according to K, and as a result less likely to participate in a single uniformly chosen spanning tree. As this example demonstrates, some DPPs assign zero probability to sets whose cardinality is not equal to a particular k; in this case, k is the number of nodes in the graph minus one — the number of edges in any spanning tree. We will return to this issue in Section 5.

Eigenvalues of random matrices [58, 104] Let M be a random matrix obtained by drawing every entry independently from the complex normal distribution. This is the complex Ginibre ensemble. The eigenvalues of M, which form a finite subset of the complex plane, are distributed according to a DPP. If a Hermitian matrix is generated in the corresponding way, drawing each diagonal entry from the normal distribution and each pair of off-diagonal entries from the complex normal distribution, then we obtain the Gaussian unitary ensemble, and the eigenvalues are now a DPP-distributed subset of the real line.

Aztec diamond tilings [74] The Aztec diamond is a diamond-shaped union of lattice squares, as depicted in Figure 2.2(a). (Half of the squares have been colored gray in a checkerboard pattern.) A domino tiling is a perfect cover of the Aztec diamond using 2×1 rectangles, as in Figure 2.2(b). Suppose that we draw a tiling uniformly at random from among all possible tilings. (The number of tilings is



(a) Aztec diamond (checkered) (b) Domino tiling (c) DPP

Fig. 2.2 Aztec diamonds.

known to be exponential in the width of the diamond.) We can identify this tiling with the subset of the squares that are (a) painted gray in the checkerboard pattern and (b) covered by the left half of a horizontal tile or the bottom half of a vertical tile (see Figure 2.2(c)). This subset is distributed as a DPP.

2.2 L-ensembles

For the purposes of modeling real data, it is useful to slightly restrict the class of DPPs by focusing on *L-ensembles*. First introduced by Borodin and Rains [15], an L-ensemble defines a DPP not through the marginal kernel K, but through a real, symmetric matrix L indexed by the elements of $\mathcal{Y}$:

$$\mathcal{P}_L(\boldsymbol{Y} = Y) \propto \det(L_Y). \tag{2.6}$$

Whereas Equation (2.1) gave the marginal probabilities of inclusion for subsets A, Equation (2.6) directly specifies the atomic probabilities for every possible instantiation of $\boldsymbol{Y}$. As for K, it is easy to see that L must be positive semidefinite. However, since Equation (2.6) is only a statement of proportionality, the eigenvalues of L need not be less than one; any positive semidefinite L defines an L-ensemble. The required normalization constant can be given in closed form due to the fact that $\sum_{Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I)$, where I is the $N \times N$ identity matrix. This is a special case of the following theorem.

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

$$\sum_{A \subseteq Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I_{\bar{A}}), \tag{2.7}$$

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

Proof. Suppose that $A = \mathcal{Y}$; then Equation (2.7) holds trivially. Now suppose inductively that the theorem holds whenever $\bar{A}$ has cardinality less than k. Given A such that $|\bar{A}| = k > 0$, let i be an element of $\mathcal{Y}$ where $i \in \bar{A}$. Splitting blockwise according to the partition $\mathcal{Y} = \{i\} \cup \mathcal{Y} - \{i\}$, we can write

$$L + I_{\bar{A}} = \begin{pmatrix} L_{ii} + 1 & L_{i\bar{i}} \\ L_{\bar{i}i} & L_{\mathcal{Y}-\{i\}} + I_{\mathcal{Y}-\{i\}-A} \end{pmatrix}, \tag{2.8}$$

where $L_{\bar{i}i}$ is the subcolumn of the i-th column of L whose rows correspond to $\bar{i}$, and similarly for $L_{i\bar{i}}$. By multilinearity of the determinant, then,

$$\det(L + I_{\bar{A}}) = \begin{vmatrix} L_{ii} & L_{i\bar{i}} \\ L_{\bar{i}i} & L_{\mathcal{Y}-\{i\}} + I_{\mathcal{Y}-\{i\}-A} \end{vmatrix} + \begin{vmatrix} 1 & \mathbf{0} \\ L_{\bar{i}i} & L_{\mathcal{Y}-\{i\}} + I_{\mathcal{Y}-\{i\}-A} \end{vmatrix} \tag{2.9}$$

$$= \det(L + I_{\overline{A \cup \{i\}}}) + \det(L_{\mathcal{Y}-\{i\}} + I_{\mathcal{Y}-\{i\}-A}). \tag{2.10}$$

We can now apply the inductive hypothesis separately to each term, giving

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

$$= \sum_{A \subseteq Y \subseteq \mathcal{Y}} \det(L_Y), \tag{2.12}$$

where we observe that every Y either contains i and is included only in the first sum, or does not contain i and is included only in the second sum. □

Thus we have

$$\mathcal{P}_L(\boldsymbol{Y} = Y) = \frac{\det(L_Y)}{\det(L+I)}. \tag{2.13}$$

As a shorthand, we will write $\mathcal{P}_L(Y)$ instead of $\mathcal{P}_L(\boldsymbol{Y} = Y)$ when the meaning is clear.

We can write a version of Equation (2.5) for L-ensembles, showing that if L is a measure of similarity then diversity is preferred:

$$\mathcal{P}_L(\{i,j\}) \propto \mathcal{P}_L(\{i\})\mathcal{P}_L(\{j\}) - \left(\frac{L_{ij}}{\det(L+I)}\right)^2. \tag{2.14}$$

In this case we are reasoning about the full contents of $\boldsymbol{Y}$ rather than its marginals, but the intuition is essentially the same. Furthermore, we have the following result of [98].

Theorem 2.2. An L-ensemble is a DPP, and its marginal kernel is

$$K = L(L+I)^{-1} = I - (L+I)^{-1}. \tag{2.15}$$

Proof. Using Theorem 2.1, the marginal probability of a set A is

$$\mathcal{P}_L(A \subseteq \boldsymbol{Y}) = \frac{\sum_{A \subseteq Y \subseteq \mathcal{Y}} \det(L_Y)}{\sum_{Y \subseteq \mathcal{Y}} \det(L_Y)} \tag{2.16}$$

$$= \frac{\det(L + I_{\bar{A}})}{\det(L+I)} \tag{2.17}$$

$$= \det((L + I_{\bar{A}})(L+I)^{-1}). \tag{2.18}$$

We can use the fact that $L(L+I)^{-1} = I - (L+I)^{-1}$ to simplify and obtain

$$\mathcal{P}_L(A \subseteq \boldsymbol{Y}) = \det(I_{\bar{A}}(L+I)^{-1} + I - (L+I)^{-1}) \tag{2.19}$$

$$= \det(I - I_A(L+I)^{-1}) \tag{2.20}$$

$$= \det(I_{\bar{A}} + I_A K), \tag{2.21}$$

where we let $K = I - (L+I)^{-1}$. Now, we observe that left multiplication by I_A zeros out all the rows of a matrix except those corresponding to A. Therefore we can split blockwise using the partition $\mathcal{Y} = \bar{A} \cup A$

to get

$$\det(I_{\bar{A}} + I_A K) = \begin{vmatrix} I_{|\bar{A}|\times|\bar{A}|} & \mathbf{0} \\ K_{A\bar{A}} & K_A \end{vmatrix} \tag{2.22}$$

$$= \det(K_A). \tag{2.23}$$

□

Note that K can be computed from an eigendecomposition of $L = \sum_{n=1}^{N} \lambda_n \boldsymbol{v}_n \boldsymbol{v}_n^\top$ by a simple rescaling of eigenvalues:

$$K = \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + 1} \boldsymbol{v}_n \boldsymbol{v}_n^\top. \tag{2.24}$$

Conversely, we can ask when a DPP with marginal kernel K is also an L-ensemble. By inverting Equation (2.15), we have

$$L = K(I - K)^{-1}, \tag{2.25}$$

and again the computation can be performed by eigendecomposition. However, while the inverse in Equation (2.15) always exists due to the positive coefficient on the identity matrix, the inverse in Equation (2.25) may not. In particular, when any eigenvalue of K achieves the upper bound of 1, the DPP is not an L-ensemble. We will see later that the existence of the inverse in Equation (2.25) is equivalent to $\mathcal{P}$ giving nonzero probability to the empty set. (This is somewhat analogous to the positive probability assumption in the Hammersley–Clifford theorem for Markov random fields.) This is not a major restriction, for two reasons. First, when modeling real data we must typically allocate some nonzero probability for rare or noisy events, so when cardinality is one of the aspects we wish to model, the condition is not unreasonable. Second, we will show in Section 5 how to control the cardinality of samples drawn from the DPP, thus sidestepping the representational limits of L-ensembles.

Modulo the restriction described above, K and L offer alternative representations of DPPs. Under both representations, subsets that have higher diversity, as measured by the corresponding kernel, have higher likelihood. However, while K gives marginal probabilities, L-ensembles

directly model the atomic probabilities of observing each subset of $\mathcal{Y}$, which offers an appealing target for optimization. Furthermore, L need only be positive semidefinite, while the eigenvalues of K are bounded above. For these reasons we will focus our modeling efforts on DPPs represented as L-ensembles.

2.2.1 Geometry

Determinants have an intuitive geometric interpretation. Let B be a $D \times N$ matrix such that $L = B^\top B$. (Such a B can always be found for $D \leq N$ when L is positive semidefinite.) Denote the columns of B by B_i for $i = 1, 2, \ldots, N$. Then:

$$\mathcal{P}_L(Y) \propto \det(L_Y) = \mathrm{Vol}^2(\{B_i\}_{i \in Y}), \tag{2.26}$$

where the right-hand side is the squared $|Y|$-dimensional volume of the parallelepiped spanned by the columns of B corresponding to elements in Y.

Intuitively, we can think of the columns of B as feature vectors describing the elements of $\mathcal{Y}$. Then the kernel L measures similarity using dot products between feature vectors, and Equation (2.26) says that the probability assigned by a DPP to a set Y is related to the volume spanned by its associated feature vectors. This is illustrated in Figure 2.3.

From this intuition we can verify several important DPP properties. Diverse sets are more probable because their feature vectors are more orthogonal, and hence span larger volumes. Items with parallel feature vectors are selected together with probability zero, since their feature vectors define a degenerate parallelepiped. All else being equal, items with large-magnitude feature vectors are more likely to appear, because they multiply the spanned volumes for sets containing them.

We will revisit these intuitions in Section 3.1, where we decompose the kernel L so as to separately model the direction and magnitude of the vectors B_i.

2.3 Properties

In this section we review several useful properties of DPPs.

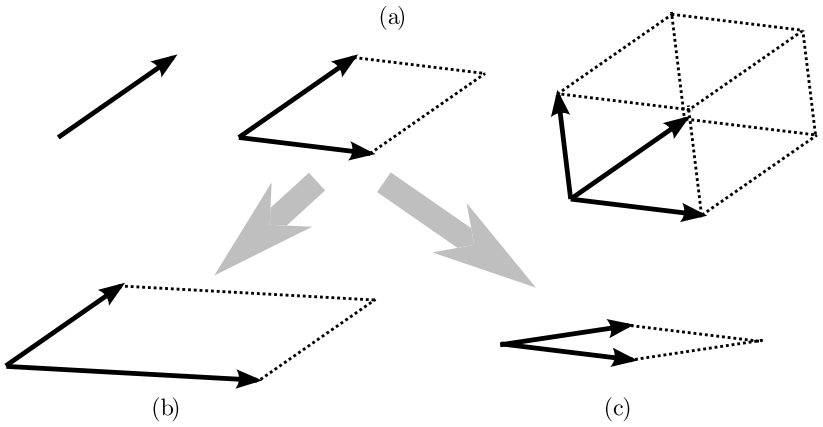


Fig. 2.3 A geometric view of DPPs: each vector corresponds to an element of $\mathcal{Y}$. (a) The probability of a subset Y is the square of the volume spanned by its associated feature vectors. (b) As the magnitude of an item's feature vector increases, so do the probabilities of sets containing that item. (c) As the similarity between two items increases, the probabilities of sets containing both of them decrease.

Restriction If $\boldsymbol{Y}$ is distributed as a DPP with marginal kernel K, then $\boldsymbol{Y} \cap A$, where $A \subseteq \mathcal{Y}$, is also distributed as a DPP, with marginal kernel K_A.

Complement If $\boldsymbol{Y}$ is distributed as a DPP with marginal kernel K, then $\mathcal{Y} - \boldsymbol{Y}$ is also distributed as a DPP, with marginal kernel $\bar{K} = I - K$. In particular, we have

$$\mathcal{P}(A \cap \boldsymbol{Y} = \emptyset) = \det(\bar{K}_A) = \det(I - K_A), \tag{2.27}$$

where I indicates the identity matrix of appropriate size. It may seem counterintuitive that the complement of a diversifying process should also encourage diversity. However, it is easy to see that

$$\begin{aligned}
\mathcal{P}(i, j \notin \boldsymbol{Y}) &= 1 - \mathcal{P}(i \in \boldsymbol{Y}) - \mathcal{P}(j \in \boldsymbol{Y}) + \mathcal{P}(i, j \in \boldsymbol{Y}) && (2.28)\\
&\leq 1 - \mathcal{P}(i \in \boldsymbol{Y}) - \mathcal{P}(j \in \boldsymbol{Y}) \\
&\quad + \mathcal{P}(i \in \boldsymbol{Y})\mathcal{P}(j \in \boldsymbol{Y}) && (2.29)\\
&= \mathcal{P}(i \notin \boldsymbol{Y}) + \mathcal{P}(j \notin \boldsymbol{Y}) - 1 \\
&\quad + (1 - \mathcal{P}(i \notin \boldsymbol{Y}))(1 - \mathcal{P}(j \notin \boldsymbol{Y})) && (2.30)\\
&= \mathcal{P}(i \notin \boldsymbol{Y})\mathcal{P}(j \notin \boldsymbol{Y}), && (2.31)
\end{aligned}$$

where the inequality follows from Equation (2.5).

Domination If $K \preceq K'$, that is, $K' - K$ is positive semidefinite, then for all $A \subseteq \mathcal{Y}$ we have

$$\det(K_A) \leq \det(K'_A). \tag{2.32}$$

In other words, the DPP defined by K' is larger than the one defined by K in the sense that it assigns higher marginal probabilities to every set A. An analogous result fails to hold for L due to the normalization constant.

Scaling If $K = \gamma K'$ for some $0 \leq \gamma < 1$, then for all $A \subseteq \mathcal{Y}$ we have

$$\det(K_A) = \gamma^{|A|} \det(K'_A). \tag{2.33}$$

It is easy to see that K defines the distribution obtained by taking a random set distributed according to the DPP with marginal kernel K', and then independently deleting each of its elements with probability $1 - \gamma$.

Cardinality Let $\lambda_1, \lambda_2, \ldots, \lambda_N$ be the eigenvalues of L. Then $|\boldsymbol{Y}|$ is distributed as the number of successes in N Bernoulli trials, where trial n succeeds with probability $\frac{\lambda_n}{\lambda_n+1}$. This fact follows from Theorem 2.3, which we prove in the next section. One immediate consequence is that $|\boldsymbol{Y}|$ cannot be larger than $\operatorname{rank}(L)$. More generally, the expected cardinality of $\boldsymbol{Y}$ is

$$\mathbb{E}[|\boldsymbol{Y}|] = \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + 1} = \operatorname{tr}(K), \tag{2.34}$$

and the variance is

$$\operatorname{Var}(|\boldsymbol{Y}|) = \sum_{n=1}^{N} \frac{\lambda_n}{(\lambda_n + 1)^2}. \tag{2.35}$$

Note that, by Equation (2.15), $\frac{\lambda_1}{\lambda_1+1}, \frac{\lambda_2}{\lambda_2+1}, \ldots, \frac{\lambda_N}{\lambda_N+1}$ are the eigenvalues of K. Figure 2.4 shows a plot of the function $f(\lambda) = \frac{\lambda}{\lambda+1}$. It is easy to see from this why the class of L-ensembles does not include DPPs where the empty set has probability zero — at least one of the Bernoulli trials would need to always succeed, and in turn one or more of the eigenvalues of L would be infinite.

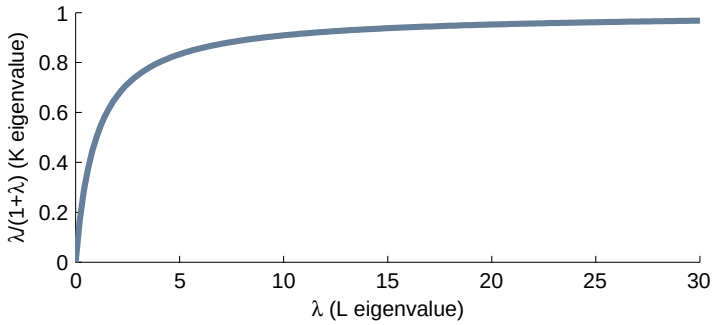


Fig. 2.4 The mapping between eigenvalues of L and K.

In some instances, the sum of Bernoullis may be an appropriate model for uncertain cardinality in real-world data, for instance when identifying objects in images where the number of objects is unknown in advance. In other situations, it may be more practical to fix the cardinality of $\boldsymbol{Y}$ up front, for instance when a set of exactly ten search results is desired, or to replace the sum of Bernoullis with an alternative cardinality model. We show how these goals can be can be achieved in Section 5.

2.4 Inference

One of the primary advantages of DPPs is that, although the number of possible realizations of $\boldsymbol{Y}$ is exponential in N, many types of inference can be performed in polynomial time. In this section we review the inference questions that can (and cannot) be answered efficiently. We also discuss the empirical practicality of the associated computations and algorithms, estimating the largest values of N that can be handled at interactive speeds (within 2–3 seconds) as well as under more generous time and memory constraints. The reference machine used for estimating real-world performance has eight Intel Xeon E5450 3Ghz cores and 32GB of memory.

2.4.1 Normalization

As we have already seen, the partition function, despite being a sum over 2^N terms, can be written in closed form as $\det(L + I)$.

Determinants of $N \times N$ matrices can be computed through matrix decomposition in $O(N^3)$ time, or reduced to matrix multiplication for better asymptotic performance. The Coppersmith–Winograd algorithm, for example, can be used to compute determinants in about $O(N^{2.376})$ time. Going forward, we will use ω to denote the exponent of whatever matrix multiplication algorithm is used.

Practically speaking, modern computers can calculate determinants up to $N \approx 5{,}000$ at interactive speeds, or up to $N \approx 40{,}000$ in about 5 minutes. When N grows much larger, the memory required simply to store the matrix becomes limiting. (Sparse storage of larger matrices is possible, but computing determinants remains prohibitively expensive unless the level of sparsity is extreme.)

2.4.2 Marginalization

The marginal probability of any set of items A can be computed using the marginal kernel as in Equation (2.1). From Equation (2.27) we can also compute the marginal probability that none of the elements in A appear. (We will see below how marginal probabilities of arbitrary configurations can be computed using conditional DPPs.)

If the DPP is specified as an L-ensemble, then the computational bottleneck for marginalization is the computation of K. The dominant operation is the matrix inversion, which requires at least $O(N^\omega)$ time by reduction to multiplication, or $O(N^3)$ using Gauss–Jordan elimination or various matrix decompositions, such as the eigendecomposition method proposed in Section 2.2. Since an eigendecomposition of the kernel will be central to sampling, the latter approach is often the most practical when working with DPPs.

Matrices up to $N \approx 2{,}000$ can be inverted at interactive speeds, and problems up to $N \approx 20{,}000$ can be completed in about 10 minutes.

2.4.3 Conditioning

It is easy to condition a DPP on the event that none of the elements in a set A appear. For $B \subseteq \mathcal{Y}$ not intersecting with A

we have

$$\mathcal{P}_L(\boldsymbol{Y} = B \mid A \cap \boldsymbol{Y} = \emptyset) = \frac{\mathcal{P}_L(\boldsymbol{Y} = B)}{\mathcal{P}_L(A \cap \boldsymbol{Y} = \emptyset)} \tag{2.36}$$

$$= \frac{\det(L_B)}{\sum_{B':B' \cap A = \emptyset} \det(L_{B'})} \tag{2.37}$$

$$= \frac{\det(L_B)}{\det(L_{\bar{A}} + I)}, \tag{2.38}$$

where $L_{\bar{A}}$ is the restriction of L to the rows and columns indexed by elements in $\mathcal{Y} - A$. In other words, the conditional distribution (over subsets of $\mathcal{Y} - A$) is itself a DPP, and its kernel $L_{\bar{A}}$ is obtained by simply dropping the rows and columns of L that correspond to elements in A.

We can also condition a DPP on the event that *all* of the elements in a set A are observed. For B not intersecting with A we have

$$\mathcal{P}_L(\boldsymbol{Y} = A \cup B \mid A \subseteq \boldsymbol{Y}) = \frac{\mathcal{P}_L(\boldsymbol{Y} = A \cup B)}{\mathcal{P}_L(A \subseteq \boldsymbol{Y})} \tag{2.39}$$

$$= \frac{\det(L_{A \cup B})}{\sum_{B':B' \cap A = \emptyset} \det(L_{A \cup B'})} \tag{2.40}$$

$$= \frac{\det(L_{A \cup B})}{\det(L + I_{\bar{A}})}, \tag{2.41}$$

where $I_{\bar{A}}$ is the matrix with ones in the diagonal entries indexed by elements of $\mathcal{Y} - A$ and zeros everywhere else. Though it is not immediately obvious, Borodin and Rains [15] showed that this conditional distribution (over subsets of $\mathcal{Y} - A$) is again a DPP, with a kernel given by

$$L^A = \left(\left[(L + I_{\bar{A}})^{-1} \right]_{\bar{A}} \right)^{-1} - I. \tag{2.42}$$

(Following the $N \times N$ inversion, the matrix is restricted to rows and columns indexed by elements in $\mathcal{Y} - A$, then inverted again.) It is easy to show that the inverses exist if and only if the probability of A appearing is nonzero.

Combining Equations (2.38) and (2.41), we can write the conditional DPP given an arbitrary combination of appearing and nonappearing

elements:

$$\mathcal{P}_L(\boldsymbol{Y} = A^{\text{in}} \cup B \mid A^{\text{in}} \subseteq \boldsymbol{Y}, A^{\text{out}} \cap \boldsymbol{Y} = \emptyset) = \frac{\det(L_{A^{\text{in}} \cup B})}{\det(L_{\bar{A}^{\text{out}}} + I_{\bar{A}^{\text{in}}})}. \tag{2.43}$$

The corresponding kernel is

$$L^{A^{\text{in}}}_{\bar{A}^{\text{out}}} = ([(L_{\bar{A}^{\text{out}}} + I_{\bar{A}^{\text{in}}})^{-1}]_{\bar{A}^{\text{in}}})^{-1} - I. \tag{2.44}$$

Thus, the class of DPPs is closed under most natural conditioning operations.

General marginals These formulas also allow us to compute arbitrary marginals. For example, applying Equation (2.15) to Equation (2.42) yields the marginal kernel for the conditional DPP given the appearance of A:

$$K^A = I - \left[(L + I_{\bar{A}})^{-1}\right]_{\bar{A}}. \tag{2.45}$$

Thus we have

$$\mathcal{P}(B \subseteq \boldsymbol{Y} | A \subseteq \boldsymbol{Y}) = \det(K^A_B). \tag{2.46}$$

(Note that K^A is indexed by elements of $\mathcal{Y} - A$, so this is only defined when A and B are disjoint.) Using Equation (2.27) for the complement of a DPP, we can now compute the marginal probability of any partial assignment, i.e.,

$$\mathcal{P}(A \subseteq \boldsymbol{Y}, B \cap \boldsymbol{Y} = \emptyset) = \mathcal{P}(A \subseteq \boldsymbol{Y})\mathcal{P}(B \cap \boldsymbol{Y} = \emptyset | A \subseteq \boldsymbol{Y}) \tag{2.47}$$

$$= \det(K_A)\det(I - K^A_B). \tag{2.48}$$

Computing conditional DPP kernels in general is asymptotically as expensive as the dominant matrix inversion, although in some cases (conditioning only on nonappearance), the inversion is not necessary. In any case, conditioning is at most a small constant factor more expensive than marginalization.

2.4.4 Sampling

Algorithm 1, due to Hough et al. [68], gives an efficient algorithm for sampling a configuration Y from a DPP. The input to the algorithm

Algorithm 1 Sampling from a DPP

Input: eigendecomposition $\{(\boldsymbol{v}_n, \lambda_n)\}_{n=1}^N$ of L
$J \leftarrow \emptyset$
for $n = 1, 2, \ldots, N$ **do**
  $J \leftarrow J \cup \{n\}$ with prob. $\frac{\lambda_n}{\lambda_n + 1}$
end for
$V \leftarrow \{\boldsymbol{v}_n\}_{n \in J}$
$Y \leftarrow \emptyset$
while $|V| > 0$ **do**
  Select i from $\mathcal{Y}$ with $\Pr(i) = \frac{1}{|V|} \sum_{\boldsymbol{v} \in V} (\boldsymbol{v}^\top \boldsymbol{e}_i)^2$
  $Y \leftarrow Y \cup i$
  $V \leftarrow V_\perp$, an orthonormal basis for the subspace of V orthogonal to $\boldsymbol{e}_i$
end while
Output: Y

is an eigendecomposition of the DPP kernel L. Note that $\boldsymbol{e}_i$ is the i-th standard basis N-vector, which is all zeros except for a one in the i-th position. We will prove the following theorem.

Theorem 2.3. Let $L = \sum_{n=1}^N \lambda_n \boldsymbol{v}_n \boldsymbol{v}_n^\top$ be an orthonormal eigendecomposition of a positive semidefinite matrix L. Then Algorithm 1 samples $\boldsymbol{Y} \sim \mathcal{P}_L$.

Algorithm 1 has two main loops, corresponding to two phases of sampling. In the first phase, a subset of the eigenvectors is selected at random, where the probability of selecting each eigenvector depends on its associated eigenvalue. In the second phase, a sample Y is produced based on the selected vectors. Note that on each iteration of the second loop, the cardinality of Y increases by one and the dimension of V is reduced by one. Since the initial dimension of V is simply the number of selected eigenvectors ($|J|$), Theorem 2.3 has the previously stated corollary that the cardinality of a random sample is distributed as a sum of Bernoulli variables.

To prove Theorem 2.3 we will first show that a DPP can be expressed as a mixture of simpler, *elementary* DPPs. We will then show that the first phase chooses an elementary DPP according to its mixing coefficient, while the second phase samples from the elementary DPP chosen in phase one.

Definition 2.1. A DPP is called **elementary** if every eigenvalue of its marginal kernel is in $\{0,1\}$. We write $\mathcal{P}^V$, where V is a set of orthonormal vectors, to denote an elementary DPP with marginal kernel $K^V = \sum_{\boldsymbol{v}\in V} \boldsymbol{v}\boldsymbol{v}^\top$.

We introduce the term "elementary" here; Hough et al. [68] refer to elementary DPPs as determinantal *projection* processes, since K^V is an orthonormal projection matrix to the subspace spanned by V. Note that, due to Equation (2.25), elementary DPPs are not generally L-ensembles. We start with a technical lemma.

Lemma 2.4. Let W_n for $n = 1, 2, \ldots, N$ be an arbitrary sequence of $k \times k$ rank-one matrices, and let $(W_n)_i$ denote the i-th column of W_n. Let $W_J = \sum_{n\in J} W_n$. Then

$$\det(W_J) = \sum_{\substack{n_1,n_2,\ldots,n_k\in J,\\ \text{distinct}}} \det([(W_{n_1})_1 (W_{n_2})_2 \ldots (W_{n_k})_k]). \tag{2.49}$$

Proof. Expanding on the first column of W_J using the multilinearity of the determinant,

$$\det(W_J) = \sum_{n\in J} \det([(W_n)_1 (W_J)_2 \ldots (W_J)_k]), \tag{2.50}$$

and, applying the same operation inductively to all columns,

$$\det(W_J) = \sum_{n_1,n_2,\ldots,n_k\in J} \det([(W_{n_1})_1 (W_{n_2})_2 \ldots (W_{n_k})_k]). \tag{2.51}$$

Since W_n has rank one, the determinant of any matrix containing two or more columns of W_n is zero; thus the terms in the sum vanish unless $n_1, n_2, \ldots, n_k$ are distinct. □

Lemma 2.5. A DPP with kernel $L = \sum_{n=1}^{N} \lambda_n \boldsymbol{v}_n \boldsymbol{v}_n^\top$ is a mixture of elementary DPPs:

$$\mathcal{P}_L = \frac{1}{\det(L+I)} \sum_{J \subseteq \{1,2,\ldots,N\}} \mathcal{P}^{V_J} \prod_{n \in J} \lambda_n, \tag{2.52}$$

where V_J denotes the set $\{\boldsymbol{v}_n\}_{n \in J}$.

Proof. Consider an arbitrary set A, with $k = |A|$. Let $W_n = [\boldsymbol{v}_n \boldsymbol{v}_n^\top]_A$ for $n = 1, 2, \ldots, N$; note that all of the W_n have rank one. From the definition of K^{V_J}, the mixture distribution on the right-hand side of Equation (2.52) gives the following expression for the marginal probability of A:

$$\frac{1}{\det(L+I)} \sum_{J \subseteq \{1,2,\ldots,N\}} \det\left(\sum_{n \in J} W_n\right) \prod_{n \in J} \lambda_n. \tag{2.53}$$

Applying Lemma 2.4, this is equal to

$$\frac{1}{\det(L+I)} \sum_{J \subseteq \{1,2,\ldots,N\}} \sum_{\substack{n_1,\ldots,n_k \in J, \\ \text{distinct}}} \det([(W_{n_1})_1 \ldots (W_{n_k})_k]) \prod_{n \in J} \lambda_n \tag{2.54}$$

$$= \frac{1}{\det(L+I)} \sum_{\substack{n_1,\ldots,n_k = 1, \\ \text{distinct}}}^{N} \times \det([(W_{n_1})_1 \ldots (W_{n_k})_k]) \sum_{J \supseteq \{n_1,\ldots,n_k\}} \prod_{n \in J} \lambda_n \tag{2.55}$$

$$= \frac{1}{\det(L+I)} \sum_{\substack{n_1,\ldots,n_k = 1, \\ \text{distinct}}}^{N} \det([(W_{n_1})_1 \ldots (W_{n_k})_k]) \times \frac{\lambda_{n_1}}{\lambda_{n_1}+1} \cdots \frac{\lambda_{n_k}}{\lambda_{n_k}+1} \prod_{n=1}^{N} (\lambda_n + 1) \tag{2.56}$$

$$= \sum_{\substack{n_1,\ldots,n_k = 1, \\ \text{distinct}}}^{N} \det\left(\left[\frac{\lambda_{n_1}}{\lambda_{n_1}+1}(W_{n_1})_1 \ldots \frac{\lambda_{n_k}}{\lambda_{n_k}+1}(W_{n_k})_k\right]\right), \tag{2.57}$$

using the fact that $\det(L + I) = \prod_{n=1}^{N}(\lambda_n + 1)$. Applying Lemma 2.4 in reverse and then the definition of K in terms of the eigendecomposition of L, we have that the marginal probability of A given by the mixture is

$$\det\left(\sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + 1} W_n\right) = \det(K_A). \tag{2.58}$$

Since the two distributions agree on all marginals, they are equal. □

Next, we show that elementary DPPs have fixed cardinality.

Lemma 2.6. If $\boldsymbol{Y}$ is drawn according to an elementary DPP $\mathcal{P}^V$, then $|\boldsymbol{Y}| = |V|$ with probability one.

Proof. Since K^V has rank $|V|$, $\mathcal{P}^V(Y \subseteq \boldsymbol{Y}) = 0$ whenever $|Y| > |V|$, so $|\boldsymbol{Y}| \leq |V|$. But we also have

$$E[|\boldsymbol{Y}|] = E\left[\sum_{n=1}^{N} \mathbb{I}(n \in \boldsymbol{Y})\right] \tag{2.59}$$

$$= \sum_{n=1}^{N} E\left[\mathbb{I}(n \in \boldsymbol{Y})\right] \tag{2.60}$$

$$= \sum_{n=1}^{N} K_{nn} = \operatorname{tr}(K) = |V|. \tag{2.61}$$

Thus $|\boldsymbol{Y}| = |V|$ almost surely. □

We can now prove the theorem.

Proof of Theorem 2.3. Lemma 2.5 says that the mixture weight of $\mathcal{P}^{V_J}$ is given by the product of the eigenvalues λ_n corresponding to the eigenvectors $\boldsymbol{v}_n \in V_J$, normalized by $\det(L + I) = \prod_{n=1}^{N}(\lambda_n + 1)$. This shows that the first loop of Algorithm 1 selects an elementary DPP $\mathcal{P}^V$ with probability equal to its mixture component. All that remains is to show that the second loop samples $\boldsymbol{Y} \sim \mathcal{P}^V$.

Let B represent the matrix whose rows are the eigenvectors in V, so that $K^V = B^\top B$. Using the geometric interpretation of determinants introduced in Section 2.2.1, $\det(K^V_Y)$ is equal to the squared

volume of the parallelepiped spanned by $\{B_i\}_{i\in Y}$. Note that since V is an orthonormal set, B_i is just the projection of $\boldsymbol{e}_i$ onto the subspace spanned by V.

Let $k = |V|$. By Lemma 2.6 and symmetry, we can consider without loss of generality a single $Y = \{1, 2, \ldots, k\}$. Using the fact that any vector both in the span of V and perpendicular to $\boldsymbol{e}_i$ is also perpendicular to the projection of $\boldsymbol{e}_i$ onto the span of V, by the base $\times$ height formula for the volume of a parallelepiped we have

$$\mathrm{Vol}\left(\{B_i\}_{i\in Y}\right) = \|B_1\|\mathrm{Vol}\left(\{\mathrm{Proj}_{\perp \boldsymbol{e}_1} B_i\}_{i=2}^{k}\right), \tag{2.62}$$

where $\mathrm{Proj}_{\perp \boldsymbol{e}_1}$ is the projection operator onto the subspace orthogonal to $\boldsymbol{e}_1$. Proceeding inductively,

$$\mathrm{Vol}\left(\{B_i\}_{i\in Y}\right) = \|B_1\|\|\mathrm{Proj}_{\perp \boldsymbol{e}_1} B_2\| \cdots \|\mathrm{Proj}_{\perp \boldsymbol{e}_1,\ldots,\boldsymbol{e}_{k-1}} B_k\|. \tag{2.63}$$

Assume that, as iteration j of the second loop in Algorithm 1 begins, we have already selected $y_1 = 1, y_2 = 2, \ldots, y_{j-1} = j-1$. Then V in the algorithm has been updated to an orthonormal basis for the subspace of the original V perpendicular to $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{j-1}$, and the probability of choosing $y_j = j$ is exactly

$$\frac{1}{|V|}\sum_{\boldsymbol{v}\in V} (\boldsymbol{v}^\top \boldsymbol{e}_j)^2 = \frac{1}{k-j+1}\|\mathrm{Proj}_{\perp \boldsymbol{e}_1,\ldots,\boldsymbol{e}_{j-1}} B_j\|^2. \tag{2.64}$$

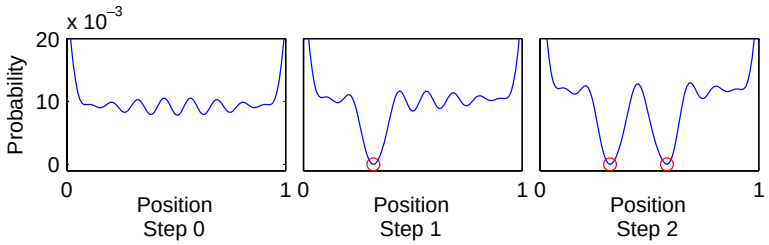
Therefore, the probability of selecting the sequence $1, 2, \ldots, k$ is

$$\frac{1}{k!}\|B_1\|^2\|\mathrm{Proj}_{\perp \boldsymbol{e}_1} B_2\|^2 \cdots \|\mathrm{Proj}_{\perp \boldsymbol{e}_1,\ldots,\boldsymbol{e}_{k-1}} B_k\|^2 = \frac{1}{k!}\mathrm{Vol}^2\left(\{B_i\}_{i\in Y}\right). \tag{2.65}$$

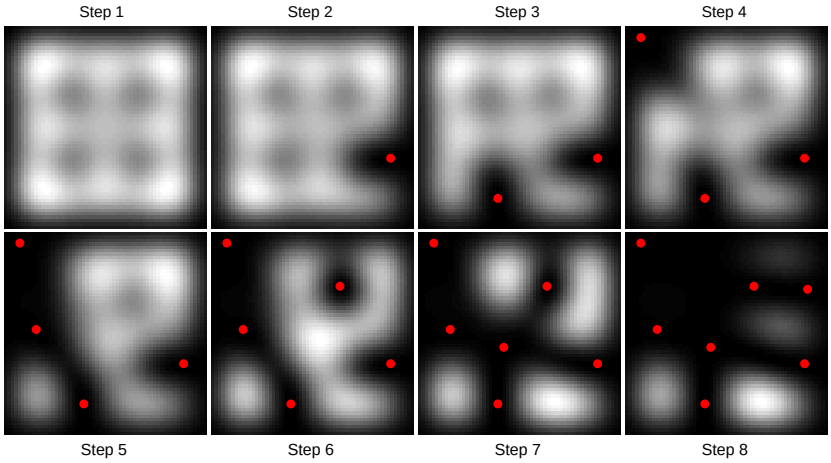
Since volume is symmetric, the argument holds identically for all of the $k!$ orderings of Y. Thus the total probability that Algorithm 1 selects Y is $\det(K_Y^V)$. □

Corollary 2.7. Algorithm 1 generates Y in uniformly random order.

Discussion To get a feel for the sampling algorithm, it is useful to visualize the distributions used to select i at each iteration, and to see how they are influenced by previously chosen items. Figure 2.5(a) shows



(a) Sampling points on an interval



(b) Sampling points in the plane

Fig. 2.5 Sampling DPP over one-dimensional (top) and two-dimensional (bottom) particle positions. Red circles indicate already selected positions. On the bottom, lighter color corresponds to higher probability. The DPP naturally reduces the probabilities for positions that are similar to those already selected.

this progression for a simple DPP where $\mathcal{Y}$ is a finely sampled grid of points in $[0,1]$, and the kernel is such that points are more similar the closer together they are. Initially, the eigenvectors V give rise to a fairly uniform distribution over points in $\mathcal{Y}$, but as each successive point is selected and V is updated, the distribution shifts to avoid points near those already chosen. Figure 2.5(b) shows a similar progression for a DPP over points in the unit square.

The sampling algorithm also offers an interesting analogy to clustering. If we think of the eigenvectors of L as representing soft clusters, and the eigenvalues as representing their strengths — the way we do

for the eigenvectors and eigenvalues of the Laplacian matrix in spectral clustering — then a DPP can be seen as performing a clustering of the elements in $\mathcal{Y}$, selecting a random subset of clusters based on their strength, and then choosing one element per selected cluster. Of course, the elements are not chosen independently and cannot be identified with specific clusters; instead, the second loop of Algorithm 1 coordinates the choices in a particular way, accounting for overlap between the eigenvectors.

Algorithm 1 runs in time $O(Nk^3)$, where $k = |V|$ is the number of eigenvectors selected in phase one. The most expensive operation is the $O(Nk^2)$ Gram–Schmidt orthonormalization required to compute $V_\perp$. If k is large, this can be reasonably expensive, but for most applications we do not want high-cardinality DPPs. (And if we want very high-cardinality DPPs, we can potentially save time by using Equation (2.27) to sample the complement instead.) In practice, the initial eigendecomposition of L is often the computational bottleneck, requiring $O(N^3)$ time. Modern multicore machines can compute eigendecompositions up to $N \approx 1{,}000$ at interactive speeds of a few seconds, or larger problems up to $N \approx 10{,}000$ in around 10 minutes. In some instances, it may be cheaper to compute only the top k eigenvectors; since phase one tends to choose eigenvectors with large eigenvalues anyway, this can be a reasonable approximation when the kernel is expected to be low rank. Note that when multiple samples are desired, the eigendecomposition needs to be performed only once.

Deshpande and Rademacher [35] recently proposed a $(1-\epsilon)$-approximate algorithm for sampling that runs in time $O(N^2 \log N \frac{k^2}{\epsilon^2} + N \log^\omega N \frac{k^{2\omega+1}}{\epsilon^{2\omega}} \log(\frac{k}{\epsilon} \log N))$ when L is already decomposed as a Gram matrix, $L = B^\top B$. When B is known but an eigendecomposition is not (and N is sufficiently large), this may be significantly faster than the exact algorithm.

2.4.5 Finding the Mode

Finding the mode of a DPP — that is, finding the set $Y \subseteq \mathcal{Y}$ that maximizes $\mathcal{P}_L(Y)$ — is NP-hard. In conditional models, this problem is sometimes referred to as maximum *a posteriori* (or MAP) inference, and

it is also NP-hard for most general structured models such as Markov random fields. Hardness was first shown for DPPs by Ko et al. [77], who studied the closely-related maximum entropy sampling problem: the entropy of a set of jointly Gaussian random variables is given (up to constants) by the log-determinant of their covariance matrix; thus finding the maximum entropy subset of those variables requires finding the principal covariance submatrix with maximum determinant. Here, we adapt the argument of Çivril and Magdon-Ismail [28], who studied the problem of finding maximum-volume submatrices.

Theorem 2.8. Let DPP-MODE be the optimization problem of finding, for a positive semidefinite $N \times N$ input matrix L indexed by elements of $\mathcal{Y}$, the maximum value of $\det(L_Y)$ over all $Y \subseteq \mathcal{Y}$. DPP-MODE is NP-hard, and furthermore it is NP-hard even to approximate DPP-MODE to a factor of $\frac{8}{9} + \epsilon$.

Proof. We reduce from EXACT 3-COVER (X3C). An instance of X3C is a set S and a collection C of three-element subsets of S; the problem is to decide whether there is a subcollection $C' \subseteq C$ such that every element of S appears exactly once in C' (that is, C' is an exact 3-cover). X3C is known to be NP-complete.

The reduction is as follows. Let $\mathcal{Y} = \{1, 2, \ldots, |C|\}$, and let B be an $|S| \times |C|$ matrix where $B_{si} = \frac{1}{\sqrt{3}}$ if C_i contains $s \in S$ and zero otherwise. Define $L = \gamma B^\top B$, where $1 < \gamma \leq \frac{9}{8}$. Note that the diagonal of L is constant and equal to γ, and an off-diagonal entry L_{ij} is zero if and only if C_i and C_j do not intersect. L is positive semidefinite by construction, and the reduction requires only polynomial time. Let $k = \frac{|S|}{3}$. We will show that the maximum value of $\det(L_Y)$ is greater than γ^{k-1} if and only if C contains an exact 3-cover of S.

($\leftarrow$) If $C' \subseteq C$ is an exact 3-cover of S, then it must contain exactly k 3-sets. Letting Y be the set of indices in C', we have $L_Y = \gamma I$, and thus its determinant is $\gamma^k > \gamma^{k-1}$.

($\rightarrow$) Suppose there is no 3-cover of S in C. Let Y be an arbitrary subset of $\mathcal{Y}$. If $|Y| < k$, then

$$\det(L_Y) \leq \prod_{i \in Y} L_{ii} = \gamma^{|Y|} \leq \gamma^{k-1}. \tag{2.66}$$

Now suppose $|Y| \geq k$, and assume without loss of generality that $Y = \{1, 2, \ldots, |Y|\}$. We have $L_Y = \gamma B_Y^\top B_Y$, and

$$\det(L_Y) = \gamma^{|Y|} \text{Vol}^2(\{B_i\}_{i \in Y}). \tag{2.67}$$

By the base $\times$ height formula,

$$\text{Vol}\left(\{B_i\}_{i \in Y}\right) = \|B_1\| \|\text{Proj}_{\perp B_1} B_2\| \cdots \|\text{Proj}_{\perp B_1, \ldots, B_{|Y|-1}} B_{|Y|}\|. \tag{2.68}$$

Note that, since the columns of B are normalized, each term in the product is at most one. Furthermore, at least $|Y| - k + 1$ of the terms must be strictly less than one, because otherwise there would be k orthogonal columns, which would correspond to a 3-cover. By the construction of B, if two columns B_i and B_j are not orthogonal then C_i and C_j overlap in at least one of the three elements, so we have

$$\begin{aligned} \|\text{Proj}_{\perp B_j} B_i\| &= \|B_i - (B_i^\top B_j) B_j\| && (2.69) \\ &\leq \left\| B_i - \frac{1}{3} B_j \right\| && (2.70) \\ &\leq \sqrt{\frac{8}{9}}. && (2.71) \end{aligned}$$

Therefore,

$$\begin{aligned} \det(L_Y) &\leq \gamma^{|Y|} \left(\frac{8}{9}\right)^{|Y|-k+1} && (2.72) \\ &\leq \gamma^{k-1}, && (2.73) \end{aligned}$$

since $\gamma \leq \frac{9}{8}$.

We have shown that the existence of a 3-cover implies that the optimal value of DPP-MODE is at least γ^k, while the optimal value cannot be more than γ^{k-1} if there is no 3-cover. Thus any algorithm that can approximate DPP-MODE to better than a factor of $\frac{1}{\gamma}$ can be used to solve X3C in polynomial time. We can choose $\gamma = \frac{9}{8}$ to show that an approximation ratio of $\frac{8}{9} + \epsilon$ is NP-hard. □

Since there are only $|C|$ possible cardinalities for Y, Theorem 2.8 shows that DPP-MODE is NP-hard even under cardinality constraints.

Ko et al. [77] propose an exact, exponential branch-and-bound algorithm for finding the mode using greedy heuristics to build

candidate sets; they tested their algorithm on problems up to $N = 75$, successfully finding optimal solutions in up to about an hour. Modern computers are likely a few orders of magnitude faster; however, this algorithm is still probably impractical for applications with large N. Çivril and Magdon-Ismail [28] propose an efficient greedy algorithm for finding a set of size k, and prove that it achieves an approximation ratio of $O(\frac{1}{k!})$. While this guarantee is relatively poor for all but very small k, in practice the results may be useful nonetheless.

Submodularity $\mathcal{P}_L$ is *log-submodular*; that is,

$$\log \mathcal{P}_L(Y \cup \{i\}) - \log \mathcal{P}_L(Y) \geq \log \mathcal{P}_L(Y' \cup \{i\}) - \log \mathcal{P}_L(Y') \quad (2.74)$$

whenever $Y \subseteq Y' \subseteq \mathcal{Y} - \{i\}$. Intuitively, adding elements to Y yields diminishing returns as Y gets larger. (This is easy to show by a volume argument.) Submodular functions can be minimized in polynomial time [127], and many results exist for approximately maximizing *monotone* submodular functions, which have the special property that supersets always have higher function values than their subsets [46, 53, 110]. In Section 4.2.1 we will discuss how these kinds of greedy algorithms can be adapted for DPPs. However, in general $\mathcal{P}_L$ is highly nonmonotone, since the addition of even a single element can decrease the probability to zero.

Recently, Feige et al. [47] showed that even nonmonotone submodular functions can be approximately maximized in polynomial time using a local search algorithm, and a growing body of research has focused on extending this result in a variety of ways [25, 48, 49, 56, 90, 153]. In our recent work we showed how the computational structure of DPPs gives rise to a particularly efficient variant of these methods [81].

2.5 Related Processes

Historically, a wide variety of point process models have been proposed and applied to applications involving diverse subsets, particularly in settings where the items can be seen as points in a physical space and diversity is taken to mean some sort of "spreading" behavior. However, DPPs are essentially unique among this class in having efficient and exact algorithms for probabilistic inference, which is why

they are particularly appealing models for machine learning applications. In this section we briefly survey the wider world of point processes and discuss the computational properties of alternative models; we will focus on point processes that lead to what is variously described as diversity, repulsion, (over)dispersion, regularity, order, and inhibition.

2.5.1 Poisson Point Processes

Perhaps the most fundamental point process is the Poisson point process, which is depicted on the right side of Figure 2.1 [32]. While defined for continuous $\mathcal{Y}$, in the discrete setting the Poisson point process can be simulated by flipping a coin independently for each item, and including those items for which the coin comes up heads. Formally,

$$\mathcal{P}(\boldsymbol{Y} = Y) = \prod_{i \in Y} p_i \prod_{i \notin Y} (1 - p_i), \tag{2.75}$$

where $p_i \in [0, 1]$ is the bias of the i-th coin. The process is called *stationary* when p_i does not depend on i; in a spatial setting this means that no region has higher density than any other region.

A random set $\boldsymbol{Y}$ distributed as a Poisson point process has the property that whenever A and B are disjoint subsets of $\mathcal{Y}$, the random variables $\boldsymbol{Y} \cap A$ and $\boldsymbol{Y} \cap B$ are independent; that is, the points in $\boldsymbol{Y}$ are not correlated. It is easy to see that DPPs generalize Poisson point processes by choosing the marginal kernel K with $K_{ii} = p_i$ and $K_{ij} = 0, i \neq j$. This implies that inference for Poisson point processes is at least as efficient as for DPPs; in fact, it is more efficient, since for instance it is easy to compute the most likely configuration. However, since Poisson point processes do not model correlations between variables, they are rather uninteresting for most real-world applications.

Addressing this weakness, various procedural modifications of the Poisson process have been proposed in order to introduce correlations between items. While such constructions can be simple and intuitive, leading to straightforward sampling algorithms, they tend to make general statistical inference difficult.

Matérn repulsive processes Matérn [100, 101] proposed a set of techniques for thinning Poisson point processes in order to induce a type of repulsion when the items are embedded in a Euclidean space. Type I process is obtained from a Poisson set $\boldsymbol{Y}$ by removing all items in $\boldsymbol{Y}$ that lie within some radius of another item in $\boldsymbol{Y}$. That is, if two items are close to each other, they are both removed; as a result all items in the final process are spaced at least a fixed distance apart. Type II Matérn repulsive process, designed to achieve the same minimum distance property while keeping more items, begins by independently assigning each item in $\boldsymbol{Y}$ a uniformly random "time" in $[0, 1]$. Then, any item within a given radius of a point having a smaller time value is removed. Under this construction, when two items are close to each other only the later one is removed. Still, an item may be removed due to its proximity with an earlier item that was itself removed. This leads to Type III process, which proceeds dynamically, eliminating items in time order whenever an earlier point which has not been removed lies within the radius.

Inference for the Matérn processes is computationally daunting. First- and second-order moments can be computed for Types I and II, but in those cases computing the likelihood of a set Y is seemingly intractable [106]. Recent work by Huber and Wolpert [69] shows that it is possible to compute likelihood for certain restricted Type III processes, but computing moments cannot be done in closed form. In the general case, likelihood for Type III processes must be estimated using an expensive Markov chain Monte Carlo algorithm.

The Matérn processes are called "hard-core" because they strictly enforce a minimum radius between selected items. While this property leads to one kind of diversity, it is somewhat limited, and due to the procedural definition it is difficult to characterize the side effects of the thinning process in a general way. Stoyan and Stoyan [138] considered an extension where the radius is itself chosen randomly, which may be more natural for certain settings, but it does not alleviate the computational issues.

Random sequential adsorption The Matérn repulsive processes are related in spirit to the random sequential adsorption (RSA) model,

which has been used in physics and chemistry to model particles that bind to two-dimensional surfaces, e.g., proteins on a cell membrane [45, 51, 66, 123, 142, 143]. RSA is described generatively as follows. Initially, the surface is empty; iteratively, particles arrive and bind uniformly at random to a location from among all locations that are not within a given radius of any previously bound particle. When no such locations remain (the "jamming limit"), the process is complete.

Like the Matérn processes, RSA is a hard-core model, designed primarily to capture packing distributions, with much of the theoretical analysis focused on the achievable density. If the set of locations is further restricted at each step to those found in an initially selected Poisson set $\boldsymbol{Y}$, then it is equivalent to a Matérn Type III process [69]; it therefore shares the same computational burdens.

2.5.2 Gibbs and Markov Point Processes

While manipulating the Poisson process procedurally has some intuitive appeal, it seems plausible that a more holistically defined process might be easier to work with, both analytically and algorithmically. The Gibbs point process provides such an approach, offering a general framework for incorporating correlations among selected items [33, 107, 108, 120, 124, 125, 148]. The Gibbs probability of a set Y is given by

$$\mathcal{P}(\boldsymbol{Y} = Y) \propto \exp(-U(Y)), \tag{2.76}$$

where U is an energy function. Of course, this definition is fully general without further constraints on U. A typical assumption is that U decomposes over subsets of items in Y; for instance

$$\exp(-U(Y)) = \prod_{A \subseteq Y, |A| \leq k} \psi_{|A|}(A) \tag{2.77}$$

for some small constant order k and potential functions ψ. In practice, the most common case is $k = 2$, which is sometimes called a pairwise interaction point process [39]:

$$\mathcal{P}(\boldsymbol{Y} = Y) \propto \prod_{i \in Y} \psi_1(i) \prod_{i,j \subseteq Y} \psi_2(i,j). \tag{2.78}$$

In spatial settings, a Gibbs point process whose potential functions are identically 1 whenever their input arguments do not lie within a ball

of fixed radius — that is, whose energy function can be decomposed into only local terms — is called a Markov point process. A number of specific Markov point processes have become well known.

Pairwise Markov processes Strauss [139] introduced a simple pairwise Markov point process for spatial data in which the potential function $\psi_2(i,j)$ is piecewise constant, taking the value 1 whenever i and j are at least a fixed radius apart, and the constant value γ otherwise. When $\gamma > 1$, the resulting process prefers clustered items. (Note that $\gamma > 1$ is only possible in the discrete case; in the continuous setting the distribution becomes nonintegrable.) We are more interested in the case $0 < \gamma < 1$, where configurations in which selected items are near one another are discounted. When $\gamma = 0$, the resulting process becomes hard-core, but in general the Strauss process is "soft-core", preferring but not requiring diversity.

The Strauss process is typical of pairwise Markov processes in that its potential function $\psi_2(i,j) = \psi(|i-j|)$ depends only on the distance between its arguments. A variety of alternative definitions for $\psi(\cdot)$ have been proposed [114, 125]. For instance,

$$\psi(r) = 1 - \exp(-(r/\sigma)^2) \tag{2.79}$$

$$\psi(r) = \exp(-(\sigma/r)^n), \quad n > 2 \tag{2.80}$$

$$\psi(r) = \min(r/\sigma, 1), \tag{2.81}$$

where σ controls the degree of repulsion in each case. Each definition leads to a point process with a slightly different concept of diversity.

Area-interaction point processes Baddeley and Van Lieshout [3] proposed a non-pairwise spatial Markov point process called the *area-interaction* model, where $U(Y)$ is given by $\log\gamma$ times the total area contained in the union of discs of fixed radius centered at all of the items in Y. When $\gamma > 1$, we have $\log\gamma > 0$ and the process prefers sets whose discs cover as little area as possible, i.e., whose items are clustered. When $0 < \gamma < 1$, $\log\gamma$ becomes negative, so the process prefers "diverse" sets covering as much area as possible.

If none of the selected items fall within twice the disc radius of each other, then $\exp(-U(Y))$ can be decomposed into potential functions

over single items, since the total area is simply the sum of the individual discs. Similarly, if each disc intersects with at most one other disc, the area-interaction process can be written as a pairwise interaction model. However, in general, an unbounded number of items might appear in a given disc; as a result the area-interaction process is an infinite-order Gibbs process. Since items only interact when they are near one another, however, local potential functions are sufficient and the process is Markov.

Computational issues Markov point processes have many intuitive properties. In fact, it is not difficult to see that, for discrete ground sets $\mathcal{Y}$, the Markov point process is equivalent to a Markov random field (MRF) on binary variables corresponding to the elements of $\mathcal{Y}$. In Section 3.2.2 we will return to this equivalence in order to discuss the relative expressive possibilities of DPPs and MRFs. For now, however, we simply observe that, as for MRFs with negative correlations, repulsive Markov point processes are computationally intractable. Even computing the normalizing constant for Equation (2.76) is NP-hard in the cases outlined above [32, 107].

On the other hand, quite a bit of attention has been paid to approximate inference algorithms for Markov point processes, employing pseudolikelihood [8, 10, 71, 124], Markov chain Monte Carlo methods [6, 9, 63, 125], and other approximations [38, 115]. Nonetheless, in general these methods are slow and/or inexact, and closed-form expressions for moments and densities rarely exist [108]. In this sense the DPP is unique.

2.5.3 Generalizations of Determinants

The determinant of a $k \times k$ matrix K can be written as a polynomial of degree k in the entries of K; in particular,

$$\det(K) = \sum_{\pi} \operatorname{sgn}(\pi) \prod_{i=1}^{k} K_{i,\pi(i)}, \tag{2.82}$$

where the sum is over all permutations π on $1, 2, \ldots, k$, and sgn is the permutation sign function. In a DPP, of course, when K is (a submatrix of) the marginal kernel Equation (2.82) gives the appearance

probability of the k items indexing K. A natural question is whether generalizations of this formula give rise to alternative point processes of interest.

Immanantal point processes In fact, Equation (2.82) is a special case of the more general *matrix immanant*, where the sgn function is replaced by χ, the irreducible representation-theoretic character of the symmetric group on k items corresponding to a particular partition of $1, 2, \ldots, k$. When the partition has k parts, that is, each element is in its own part, $\chi(\pi) = \operatorname{sgn}(\pi)$ and we recover the determinant. When the partition has a single part, $\chi(\pi) = 1$ and the result is the permanent of K. The associated *permanental process* was first described alongside DPPs by Macchi [98], who referred to it as the "boson process." Bosons do not obey the Pauli exclusion principle, and the permanental process is in some ways the opposite of a DPP, preferring sets of points that are more tightly clustered, or less diverse, than if they were independent. Several recent papers have considered its properties in some detail [68, 103]. Furthermore, [37] considered the point processes induced by general immanants, showing that they are well defined and in some sense "interpolate" between determinantal and permanental processes.

Computationally, obtaining the permanent of a matrix is #P-complete [147], making the permanental process difficult to work with in practice. Complexity results for immanants are less definitive, with certain classes of immanants apparently hard to compute [19, 20], while some upper bounds on complexity are known [5, 65], and at least one nontrivial case is efficiently computable [62]. It is not clear whether the latter result provides enough leverage to perform inference beyond computing marginals.

α-determinantal point processes An alternative generalization of Equation (2.82) is given by the so-called α-determinant, where $\operatorname{sgn}(\pi)$ is replaced by $\alpha^{k-\nu(\pi)}$, with $\nu(\pi)$ counting the number of cycles in π [68, 152]. When $\alpha = -1$ the determinant is recovered, and when $\alpha = +1$ we have again the permanent. Relatively little is known for other values of α, although Shirai and Takahashi [133] conjecture that

the associated process exists when $0 \leq \alpha \leq 2$ but not when $\alpha > 2$. Whether α-determinantal processes have useful properties for modeling or computational advantages remains an open question.

Hyperdeterminantal point processes A third possible generalization of Equation (2.82) is the hyperdeterminant originally proposed by Cayley [24] and discussed in the context of point processes by Evans and Gottlieb [44]. Whereas the standard determinant operates on a two-dimensional matrix with entries indexed by pairs of items, the hyperdeterminant operates on higher-dimensional kernel matrices indexed by sets of items. The hyperdeterminant potentially offers additional modeling power, and Evans and Gottlieb [44] show that some useful properties of DPPs are preserved in this setting. However, so far relatively little is known about these processes.

2.5.4 Quasirandom Processes

Monte Carlo methods rely on draws of random points in order to approximate quantities of interest; randomness guarantees that, regardless of the function being studied, the estimates will be accurate in expectation and converge in the limit. However, in practice we get to observe only a finite set of values drawn from the random source. If, by chance, this set is "bad", the resulting estimate may be poor. This concern has led to the development of so-called *quasirandom* sets, which are in fact deterministically generated, but can be substituted for random sets in some instances to obtain improved convergence guarantees [112, 135].

In contrast with pseudorandom generators, which attempt to mimic randomness by satisfying statistical tests that ensure unpredictability, quasirandom sets are not designed to appear random, and their elements are not (even approximately) independent. Instead, they are designed to have low *discrepancy*; roughly speaking, low-discrepancy sets are "diverse" in that they cover the sample space evenly. Consider a finite subset Y of $[0,1]^D$, with elements $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_D^{(i)})$ for $i = 1, 2, \ldots, k$. Let $S_{\boldsymbol{x}} = [0, x_1) \times [0, x_2) \times \cdots \times [0, x_D)$ denote the box defined by the origin and the point $\boldsymbol{x}$. The discrepancy of Y is defined

as follows.

$$\mathrm{disc}(Y) = \max_{\boldsymbol{x} \in Y} \left| \frac{|Y \cap S_{\boldsymbol{x}}|}{k} - \mathrm{Vol}(S_{\boldsymbol{x}}) \right|. \tag{2.83}$$

That is, the discrepancy measures how the empirical density $|Y \cap S_{\boldsymbol{x}}|/k$ differs from the uniform density $\mathrm{Vol}(S_{\boldsymbol{x}})$ over the boxes $S_{\boldsymbol{x}}$. Quasirandom sets with low discrepancy cover the unit cube with more uniform density than do pseudorandom sets, analogously to Figure 2.1.

This deterministic uniformity property makes quasirandom sets useful for Monte Carlo estimation via (among other results) the Koksma-Hlawka inequality [67, 112]. For a function f with bounded variation $V(f)$ on the unit cube, the inequality states that

$$\left| \frac{1}{k} \sum_{\boldsymbol{x} \in Y} f(\boldsymbol{x}) - \int_{[0,1]^D} f(\boldsymbol{x}) d\boldsymbol{x} \right| \leq V(f)\mathrm{disc}(Y). \tag{2.84}$$

Thus, low-discrepancy sets lead to accurate quasi-Monte Carlo estimates. In contrast to typical Monte Carlo guarantees, the Koksma-Hlawka inequality is deterministic. Moreover, since the rate of convergence for standard stochastic Monte Carlo methods is $k^{-1/2}$, this result is an (asymptotic) improvement when the discrepancy diminishes faster than $k^{-1/2}$.

In fact, it is possible to construct quasirandom sequences where the discrepancy of the first k elements is $O((\log k)^D/k)$; the first such sequence was proposed by [64]. The Sobol sequence [134], introduced later, offers improved uniformity properties and can be generated efficiently [18].

It seems plausible that, due to their uniformity characteristics, low-discrepancy sets could be used as computationally efficient but nonprobabilistic tools for working with data exhibiting diversity. An algorithm generating quasirandom sets could be seen as an efficient prediction procedure if made to depend somehow on input data and a set of learned parameters. However, to our knowledge no work has yet addressed this possibility.

3

Representation and Algorithms

Determinantal point processes come with a deep and beautiful theory, and, as we have seen, exactly characterize many theoretical processes. However, they are also promising models for real-world data that exhibit diversity, and we are interested in making such applications as intuitive, practical, and computationally efficient as possible. In this section, we present a variety of fundamental techniques and algorithms that serve these goals and form the basis of the extensions we discuss later.

We begin by describing a decomposition of the DPP kernel that offers an intuitive trade-off between a unary model of quality over the items in the ground set and a global model of diversity. The geometric intuitions from Section 2 extend naturally to this decomposition. Splitting the model into quality and diversity components then allows us to make a comparative study of *expressiveness* — that is, the range of distributions that the model can describe. We compare the expressive powers of DPPs and negative-interaction Markov random fields, showing that the two models are incomparable in general but exhibit qualitatively similar characteristics, despite the computational advantages offered by DPPs.

Next, we turn to the challenges imposed by large datasets, which are common in practice. We first address the case where N, the number of items in the ground set, is very large. In this setting, the superlinear number of operations required for most DPP inference algorithms can be prohibitively expensive. However, by introducing a dual representation of a DPP we show that efficient DPP inference remains possible when the kernel is low-rank. When the kernel is not low-rank, we prove that a simple approximation based on random projections dramatically speeds inference while guaranteeing that the deviation from the original distribution is bounded. These techniques will be especially useful in Section 6, when we consider exponentially large N.

Finally, we discuss some alternative formulas for the likelihood of a set Y in terms of the marginal kernel K. Compared to the L-ensemble formula in Equation (2.13), these may be analytically more convenient, since they do not involve ratios or arbitrary principal minors.

3.1 Quality versus Diversity

An important practical concern for modeling is interpretability; that is, practitioners should be able to understand the parameters of the model in an intuitive way. While the entries of the DPP kernel are not totally opaque in that they can be seen as measures of similarity — reflecting our primary qualitative characterization of DPPs as diversifying processes — in most practical situations we want diversity to be balanced against some underlying preferences for different items in $\mathcal{Y}$. In this section, we propose a decomposition of the DPP that more directly illustrates the tension between diversity and a per-item measure of quality.

In Section 2 we observed that the DPP kernel L can be written as a Gram matrix, $L = B^\top B$, where the columns of B are vectors representing items in the set $\mathcal{Y}$. We now take this one step further, writing each column B_i as the product of a *quality* term $q_i \in \mathbb{R}^+$ and a vector of normalized *diversity features* $\phi_i \in \mathbb{R}^D$, $\|\phi_i\| = 1$. (While $D = N$ is sufficient to decompose any DPP, we keep D arbitrary since in practice we may wish to use high-dimensional feature vectors.) The entries of

the kernel can now be written as

$$L_{ij} = q_i \phi_i^\top \phi_j q_j. \tag{3.1}$$

We can think of $q_i \in \mathbb{R}^+$ as measuring the intrinsic "goodness" of an item i, and $\phi_i^\top \phi_j \in [-1, 1]$ as a signed measure of similarity between items i and j. We use the following shorthand for similarity:

$$S_{ij} \equiv \phi_i^\top \phi_j = \frac{L_{ij}}{\sqrt{L_{ii} L_{jj}}}. \tag{3.2}$$

This decomposition of L has two main advantages. First, it implicitly enforces the constraint that L must be positive semidefinite, which can potentially simplify learning (see Section 4). Second, it allows us to independently model quality and diversity, and then combine them into a unified model. In particular, we have:

$$\mathcal{P}_L(Y) \propto \left(\prod_{i \in Y} q_i^2 \right) \det(S_Y), \tag{3.3}$$

where the first term increases with the quality of the selected items and the second term increases with the diversity of the selected items. We will refer to q as the *quality model* and S or ϕ as the *diversity model*. Without the diversity model, we would choose high-quality items, but we would tend to choose similar high-quality items over and over. Without the quality model, we would get a very diverse set, but we might fail to include the most important items in $\mathcal{Y}$, focusing instead on low-quality outliers. By combining the two models we can achieve a more balanced result.

Returning to the geometric intuitions from Section 2.2.1, the determinant of L_Y is equal to the squared volume of the parallelepiped spanned by the vectors $q_i \phi_i$ for $i \in Y$. The magnitude of the vector representing item i is q_i, and its direction is ϕ_i. Figure 3.1 (reproduced from the previous section) now makes clear how DPPs decomposed in this way naturally balance the two objectives of high quality and high diversity. Going forward, we will nearly always assume that our models are decomposed into quality and diversity components. This provides us not only with a natural and intuitive setup for real-world applications, but also a useful perspective for comparing DPPs with existing models, which we turn to next.

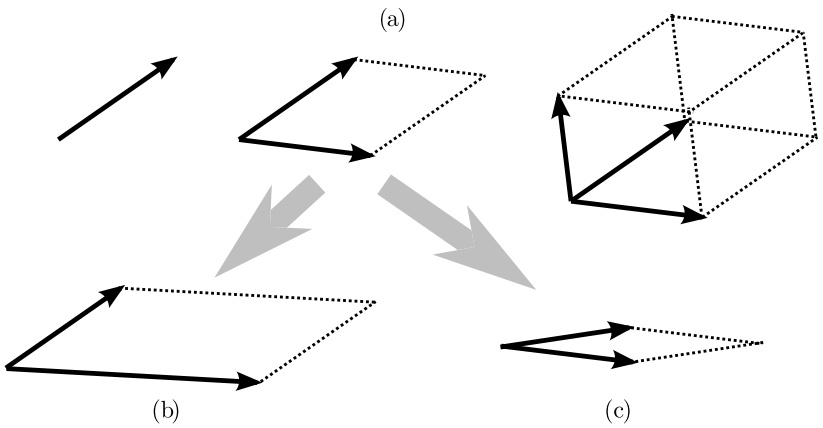


Fig. 3.1 Revisiting DPP geometry: (a) The probability of a subset Y is the square of the volume spanned by $q_i\phi_i$ for $i \in Y$. (b) As item i's quality q_i increases, so do the probabilities of sets containing item i. (c) As two items i and j become more similar, $\phi_i^\top \phi_j$ increases and the probabilities of sets containing both i and j decrease.

3.2 Expressive Power

Many probabilistic models are known and widely used within the machine learning community. A natural question, therefore, is what advantages DPPs offer that standard models do not. We have seen already how a large variety of inference tasks, like sampling and conditioning, can be performed efficiently for DPPs; however, efficiency is essentially a prerequisite for any practical model. What makes DPPs particularly unique is the marriage of these computational advantages with the ability to express *global, negative* interactions between modeling variables; this repulsive domain is notoriously intractable using traditional approaches like graphical models [17, 70, 82, 109, 146, 156, 157]. In this section we elaborate on the expressive powers of DPPs and compare them with those of Markov random fields, which we take as representative graphical models.

3.2.1 Markov Random Fields

A Markov random field (MRF) is an undirected graphical model defined by a graph G whose nodes $1, 2, \ldots, N$ represent random variables. For

our purposes, we will consider binary MRFs, where each output variable takes a value from $\{0, 1\}$. We use y_i to denote a value of the i-th output variable, bold $\boldsymbol{y}_c$ to denote an assignment to a set of variables c, and $\boldsymbol{y}$ for an assignment to all of the output variables. The graph edges E encode direct dependence relationships between variables; for example, there might be edges between similar elements i and j to represent the fact that they tend not to co-occur. MRFs are often referred to as *conditional random fields* when they are parameterized to depend on input data, and especially when G is a chain [88].

An MRF defines a joint probability distribution over the output variables that factorize across the cliques $\mathcal{C}$ of G:

$$\mathcal{P}(\boldsymbol{y}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{y}_c). \tag{3.4}$$

Here each ψ_c is a potential function that assigns a nonnegative value to every possible assignment $\boldsymbol{y}_c$ of the clique c, and Z is the normalization constant $\sum_{\boldsymbol{y}'} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{y}'_c)$. Note that, for a binary MRF, we can think of $\boldsymbol{y}$ as the characteristic vector for a subset Y of $\mathcal{Y} = \{1, 2, \ldots, N\}$. Then the MRF is equivalently the distribution of a random subset $\boldsymbol{Y}$, where $\mathcal{P}(\boldsymbol{Y} = Y)$ is equivalent to $\mathcal{P}(\boldsymbol{y})$.

The Hammersley–Clifford theorem states that $\mathcal{P}(\boldsymbol{y})$ defined in Equation (3.4) is always Markov with respect to G; that is, each variable is conditionally independent of all other variables, given its neighbors in G. The converse also holds: any distribution that is Markov with respect to G, as long as it is strictly positive, can be decomposed over the cliques of G as in Equation (3.4) [61]. MRFs therefore offer an intuitive way to model problem structure. Given domain knowledge about the nature of the ways in which outputs interact, a practitioner can construct a graph that encodes a precise set of conditional independence relations. (Because the number of unique assignments to a clique c is exponential in $|c|$, computational constraints generally limit us to small cliques.)

For comparison with DPPs, we will focus on *pairwise* MRFs, where the largest cliques with interesting potential functions are the edges; that is, $\psi_c(\boldsymbol{y}_c) = 1$ for all cliques c where $|c| > 2$. The pairwise

distribution is

$$\mathcal{P}(\boldsymbol{y}) = \frac{1}{Z}\prod_{i=1}^{N}\psi_i(y_i)\prod_{ij\in E}\psi_{ij}(y_i, y_j). \tag{3.5}$$

We refer to $\psi_i(y_i)$ as node potentials and $\psi_{ij}(y_i, y_j)$ as edge potentials.

MRFs are very general models — in fact, if the cliques are unbounded in size, they are fully general — but inference is only tractable in certain special cases. Cooper [30] showed that general probabilistic inference (conditioning and marginalization) in MRFs is NP-hard, and this was later extended by Dagum and Luby [31], who showed that inference is NP-hard even to approximate. Shimony [130] proved that the MAP inference problem (finding the mode of an MRF) is also NP-hard, and Abdelbar and Hedetniemi [1] showed that the MAP problem is likewise hard to approximate. In contrast, we showed in Section 2 that DPPs offer efficient exact probabilistic inference; furthermore, although the MAP problem for DPPs is NP-hard, it can be approximated to a constant factor under cardinality constraints in polynomial time.

The first tractable subclass of MRFs was identified by Pearl [119], who showed that belief propagation can be used to perform inference in polynomial time when G is a tree. More recently, certain types of inference in binary MRFs with *associative* (or submodular) potentials ψ have been shown to be tractable [17, 79, 146]. Inference in nonbinary associative MRFs is NP-hard, but can be efficiently approximated to a constant factor depending on the size of the largest clique [146]. Intuitively, an edge potential is called associative if it encourages the endpoint nodes take the same value (e.g., to be both in or both out of the set Y). More formally, associative potentials are at least one whenever the variables they depend on are all equal, and exactly one otherwise.

We can rewrite the pairwise, binary MRF of Equation (3.5) in a canonical log-linear form:

$$\mathcal{P}(\boldsymbol{y}) \propto \exp\left(\sum_i w_i y_i + \sum_{ij\in E} w_{ij} y_i y_j\right). \tag{3.6}$$

Here we have eliminated redundancies by forcing $\psi_i(0) = 1$, $\psi_{ij}(0,0) = \psi_{ij}(0,1) = \psi_{ij}(1,0) = 1$, and setting $w_i = \log \psi_i(1)$, $w_{ij} = \log \psi_{ij}(1,1)$. This parameterization is sometimes called the fully visible Boltzmann machine. Under this representation, the MRF is associative whenever $w_{ij} \geq 0$ for all $ij \in E$.

We have seen that inference in MRFs is tractable when we restrict the graph to a tree or require the potentials to encourage agreement. However, the repulsive potentials necessary to build MRFs exhibiting diversity are the opposite of associative potentials (since they imply $w_{ij} < 0$), and lead to intractable inference for general graphs. Indeed, such negative potentials can create "frustrated cycles", which have been used both as illustrations of common MRF inference algorithm failures [82] and as targets for improving those algorithms [136]. A wide array of (informally) approximate inference algorithms have been proposed to mitigate tractability problems, but none to our knowledge effectively and reliably handles the case where potentials exhibit strong repulsion.

3.2.2 Comparing DPPs and MRFs

Despite the computational issues outlined in the previous section, MRFs are popular models and, importantly, intuitive for practitioners, both because they are familiar and because their potential functions directly model simple, local relationships. We argue that DPPs have a similarly intuitive interpretation using the decomposition in Section 3.1. Here, we compare the distributions realizable by DPPs and MRFs to see whether the tractability of DPPs comes at a large expressive cost.

Consider a DPP over $\mathcal{Y} = \{1, 2, \ldots, N\}$ with $N \times N$ kernel matrix L decomposed as in Section 3.1; we have

$$\mathcal{P}_L(Y) \propto \det(L_Y) = \left(\prod_{i \in Y} q_i^2 \right) \det(S_Y). \tag{3.7}$$

The most closely related MRF is a pairwise, complete graph on N binary nodes with negative interaction terms. We let $y_i = \mathbb{I}(i \in Y)$ be indicator variables for the set Y, and write the MRF in the log-linear

form of Equation (3.6):

$$\mathcal{P}_{\mathrm{MRF}}(Y) \propto \exp\left(\sum_i w_i y_i + \sum_{i<j} w_{ij} y_i y_j\right), \tag{3.8}$$

where $w_{ij} \leq 0$.

Both of these models can capture negative correlations between indicator variables y_i. Both models also have $\frac{N(N+1)}{2}$ parameters: the DPP has quality scores q_i and similarity measures S_{ij}, and the MRF has node log-potentials w_i and edge log-potentials w_{ij}. The key representational difference is that, while w_{ij} are individually constrained to be nonpositive, the positive semidefinite constraint on the DPP kernel is global. One consequence is that, as a side effect, the MRF can actually capture certain limited positive correlations; for example, a 3-node MRF with $w_{12}, w_{13} < 0$ and $w_{23} = 0$ induces a positive correlation between nodes two and three by virtue of their mutual disagreement with node one. As we have seen in Section 2, the semidefinite constraint prevents the DPP from forming any positive correlations.

More generally, semidefiniteness means that the DPP diversity feature vectors must satisfy the triangle inequality, leading to

$$\sqrt{1 - S_{ij}} + \sqrt{1 - S_{jk}} \geq \sqrt{1 - S_{ik}} \tag{3.9}$$

for all $i, j, k \in \mathcal{Y}$ since $\|\phi_i - \phi_j\| \propto \sqrt{1 - S_{ij}}$. The similarity measure therefore obeys a type of transitivity, with large S_{ij} and S_{jk} implying large S_{ik}.

Equation (3.9) is not, by itself, sufficient to guarantee that L is positive semidefinite, since S must also be realizable using unit length feature vectors. However, rather than trying to develop further intuition algebraically, we turn to visualization. While it is difficult to depict the feasible distributions of DPPs and MRFs in high dimensions, we can get a feel for their differences even with a small number of elements N.

When $N = 2$, it is easy to show that the two models are equivalent, in the sense that they can both represent any distribution with negative correlations:

$$\mathcal{P}(y_1 = 1)\mathcal{P}(y_2 = 1) \geq \mathcal{P}(y_1 = 1, y_2 = 1). \tag{3.10}$$

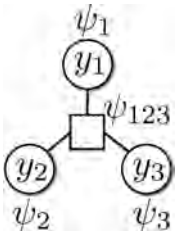


Fig. 3.2 A factor graph representation of a 3-item MRF or DPP.

When $N = 3$, differences start to become apparent. In this setting both models have six parameters: for the DPP they are $(q_1, q_2, q_3, S_{12}, S_{13}, S_{23})$, and for the MRF they are $(w_1, w_2, w_3, w_{12}, w_{13}, w_{23})$. To place the two models on equal footing, we represent each as the product of unnormalized per-item potentials ψ_1, ψ_2, ψ_3 and a single unnormalized ternary potential ψ_{123}. This representation corresponds to a factor graph with three nodes and a single, ternary factor (see Figure 3.2). The probability of a set Y is then given by

$$\mathcal{P}(Y) \propto \psi_1(y_1)\psi_2(y_2)\psi_3(y_3)\psi_{123}(y_1, y_2, y_3). \tag{3.11}$$

For the DPP, the node potentials are $\psi_i^{\mathrm{DPP}}(y_i) = q_i^{2y_i}$, and for the MRF they are $\psi_i^{\mathrm{MRF}}(y_i) = e^{w_i y_i}$. The ternary factors are

$$\psi_{123}^{\mathrm{DPP}}(y_1, y_2, y_3) = \det(S_Y), \tag{3.12}$$

$$\psi_{123}^{\mathrm{MRF}}(y_1, y_2, y_3) = \exp\left(\sum_{i<j} w_{ij} y_i y_j\right). \tag{3.13}$$

Since both models allow setting the node potentials arbitrarily, we focus now on the ternary factor. Table 3.1 shows the values of $\psi_{123}^{\mathrm{DPP}}$ and $\psi_{123}^{\mathrm{MRF}}$ for all subsets $Y \subseteq \mathcal{Y}$. The last four entries are determined, respectively, by the three edge parameters of the MRF and three similarity parameters S_{ij} of the DPP, so the sets of realizable ternary factors form three-dimensional manifolds in four-dimensional space. We attempt to visualize these manifolds by showing two-dimensional slices in three-dimensional space for various values of $\psi_{123}(1, 1, 1)$ (the last row of Table 3.1).

Figure 3.3(a) depicts four such slices of the realizable DPP distributions, and Figure 3.3(b) shows the same slices of the realizable MRF

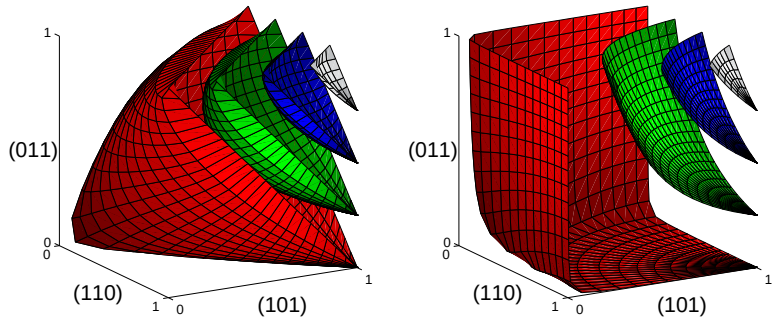
Table 3.1. Values of ternary factors for 3-item MRFs and DPPs.

Y	$y_1y_2y_3$	ψ_{123}^{MRF}	ψ_{123}^{DPP}
{}	000	1	1
{1}	100	1	1
{2}	010	1	1
{3}	001	1	1
{1,2}	110	$e^{w_{12}}$	$1 - S_{12}^2$
{1,3}	101	$e^{w_{13}}$	$1 - S_{13}^2$
{2,3}	011	$e^{w_{23}}$	$1 - S_{23}^2$
{1,2,3}	111	$e^{w_{12}+w_{13}+w_{23}}$	$1 + 2S_{12}S_{13}S_{23} - S_{12}^2 - S_{13}^2 - S_{23}^2$

distributions. Points closer to the origin (on the lower left) correspond to "more repulsive" distributions, where the three elements of $\mathcal{Y}$ are less likely to appear together. When $\psi_{123}(1,1,1)$ is large (gray surfaces), negative correlations are weak and the two models give rise to qualitatively similar distributions. As the value of the $\psi_{123}(1,1,1)$ shrinks to zero (red surfaces), the two models become quite different. MRFs, for example, can describe distributions where the first item is strongly anticorrelated with both of the others, but the second and third are not anticorrelated with each other. The transitive nature of the DPP makes this impossible.

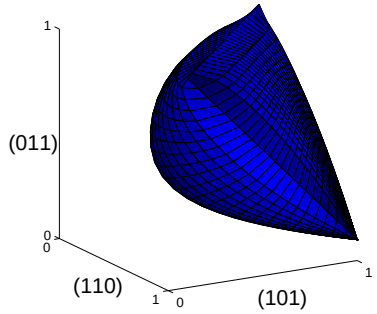
To improve visibility, we have constrained $S_{12}, S_{13}, S_{23} \geq 0$ in Figure 3.3(a). Figure 3.3(c) shows a single slice without this constraint; allowing negative similarity makes it possible to achieve strong three-way repulsion with less pairwise repulsion, closing the surface away from the origin. The corresponding MRF slice is shown in Figure 3.3(d), and the two are overlaid in Figures 3.3(e) and 3.3(f). Even though there are relatively strong interactions in these plots ($\psi_{123}(1,1,1) = 0.1$), the models remain roughly comparable in terms of the distributions they can express.

As N gets larger, we conjecture that the story is essentially the same. DPPs are primarily constrained by a notion of transitivity on the similarity measure; thus it would be difficult to use a DPP to model, for example, data where items repel "distant" items rather than similar items — if i is far from j and j is far from k we cannot necessarily conclude that i is far from k. One way of looking at this is that repulsion of distant items induces positive correlations between the selected items, which a DPP cannot represent.

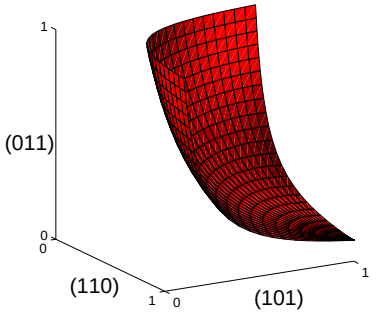


(a) DPP slices (partial)

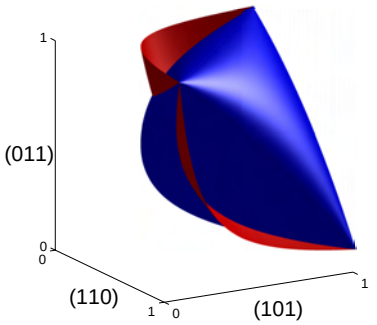
(b) MRF slices



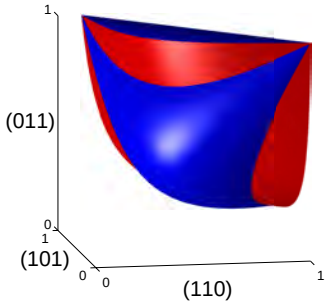
(c) DPP slice (full) for $\psi_{123}(1,1,1) = 0.1$



(d) MRF slice for $\psi_{123}(1,1,1) = 0.1$



(e) Overlay



(f) Overlay (rotated)

Fig. 3.3 (a,b) Realizable values of $\psi_{123}(1,1,0)$, $\psi_{123}(1,0,1)$, and $\psi_{123}(0,1,1)$ in a 3-factor when $\psi_{123}(1,1,1) = 0.001$ (red), 0.25 (green), 0.5 (blue), and 0.75 (gray). (c,d) Surfaces for $\psi_{123}(1,1,1) = 0.1$, allowing negative similarity for the DPP. (e,f) DPP (blue) and MRF (red) surfaces superimposed.

MRFs, on the other hand, are constrained by their local nature and do not effectively model data that are "globally" diverse. For instance, a pairwise MRF we cannot exclude a set of three or more items without excluding some pair of those items. More generally, an MRF assumes that repulsion does not depend on (too much) context, so it cannot express that, say, there can be only a certain number of selected items overall. The DPP can naturally implement this kind of restriction though the rank of the kernel.

3.3 Dual Representation

The standard inference algorithms for DPPs rely on manipulating the kernel L through inversion, eigendecomposition, and so on. However, in situations where N is large we may not be able to work efficiently with L — in some cases we may not even have the memory to write it down. In this section, instead, we develop a dual representation of a DPP that shares many important properties with the kernel L but is often much smaller. Afterward, we will show how this dual representation can be applied to perform efficient inference.

Let B be the $D \times N$ matrix whose columns are given by $B_i = q_i \phi_i$, so that $L = B^\top B$. Consider now the matrix

$$C = BB^\top. \tag{3.14}$$

By construction, C is symmetric and positive semidefinite. In contrast to L, which is too expensive to work with when N is large, C is only $D \times D$, where D is the dimension of the diversity feature function ϕ. In many practical situations, D is fixed by the model designer, while N may grow without bound as new items become available; for instance, a search engine may continually add to its database of links. Furthermore, we have the following result.

Proposition 3.1. The nonzero eigenvalues of C and L are identical, and the corresponding eigenvectors are related by the matrix B. That is,

$$C = \sum_{n=1}^{D} \lambda_n \hat{\boldsymbol{v}}_n \hat{\boldsymbol{v}}_n^\top \tag{3.15}$$

is an eigendecomposition of C if and only if

$$L = \sum_{n=1}^{D} \lambda_n \left(\frac{1}{\sqrt{\lambda_n}} B^\top \hat{\boldsymbol{v}}_n \right) \left(\frac{1}{\sqrt{\lambda_n}} B^\top \hat{\boldsymbol{v}}_n \right)^\top \tag{3.16}$$

is an eigendecomposition of L.

Proof. In the forward direction, we assume that $\{(\lambda_n, \hat{\boldsymbol{v}}_n)\}_{n=1}^D$ is an eigendecomposition of C. We have

$$\sum_{n=1}^{D} \lambda_n \left(\frac{1}{\sqrt{\lambda_n}} B^\top \hat{\boldsymbol{v}}_n \right) \left(\frac{1}{\sqrt{\lambda_n}} B^\top \hat{\boldsymbol{v}}_n \right)^\top = B^\top \left(\sum_{n=1}^{D} \hat{\boldsymbol{v}}_n \hat{\boldsymbol{v}}_n^\top \right) B \tag{3.17}$$

$$= B^\top B = L, \tag{3.18}$$

since $\hat{\boldsymbol{v}}_n$ are orthonormal by assumption. Furthermore, for any n we have

$$\left\| \frac{1}{\sqrt{\lambda_n}} B^\top \hat{\boldsymbol{v}}_n \right\|^2 = \frac{1}{\lambda_n} (B^\top \hat{\boldsymbol{v}}_n)^\top (B^\top \hat{\boldsymbol{v}}_n) \tag{3.19}$$

$$= \frac{1}{\lambda_n} \hat{\boldsymbol{v}}_n^\top C \hat{\boldsymbol{v}}_n \tag{3.20}$$

$$= \frac{1}{\lambda_n} \lambda_n \|\hat{\boldsymbol{v}}_n\| \tag{3.21}$$

$$= 1, \tag{3.22}$$

using the fact that $C\hat{\boldsymbol{v}}_n = \lambda_n \hat{\boldsymbol{v}}_n$ since $\hat{\boldsymbol{v}}_n$ is an eigenvector of C. Finally, for any distinct $1 \leq a, b \leq D$, we have

$$\left(\frac{1}{\sqrt{\lambda_a}} B^\top \hat{\boldsymbol{v}}_a \right)^\top \left(\frac{1}{\sqrt{\lambda_b}} B^\top \hat{\boldsymbol{v}}_b \right) = \frac{1}{\sqrt{\lambda_a \lambda_b}} \hat{\boldsymbol{v}}_a^\top C \hat{\boldsymbol{v}}_b \tag{3.23}$$

$$= \frac{\sqrt{\lambda_b}}{\sqrt{\lambda_a}} \hat{\boldsymbol{v}}_a^\top \hat{\boldsymbol{v}}_b \tag{3.24}$$

$$= 0. \tag{3.25}$$

Thus $\left\{ \left(\lambda_n, \frac{1}{\sqrt{\lambda_n}} B^\top \hat{\boldsymbol{v}}_n \right) \right\}_{n=1}^D$ is an eigendecomposition of L. In the other direction, an analogous argument applies once we observe that, since $L = B^\top B$, L has rank at most D and therefore at most D nonzero eigenvalues. □

Proposition 3.1 shows that C contains quite a bit of information about L. In fact, C is sufficient to perform nearly all forms of DPP inference efficiently, including normalization and marginalization in constant time with respect to N, and sampling in time linear in N.

3.3.1 Normalization

Recall that the normalization constant for a DPP is given by $\det(L + I)$. If $\lambda_1, \lambda_2, \ldots, \lambda_N$ are the eigenvalues of L, then the normalization constant is equal to $\prod_{n=1}^{N}(\lambda_n + 1)$, since the determinant is the product of the eigenvalues of its argument. By Proposition 3.1, the nonzero eigenvalues of L are also the eigenvalues of the dual representation C. Thus, we have

$$\det(L + I) = \prod_{n=1}^{D}(\lambda_n + 1) = \det(C + I). \tag{3.26}$$

Computing the determinant of $C + I$ requires $O(D^\omega)$ time.

3.3.2 Marginalization

Standard DPP marginalization makes use of the marginal kernel K, which is of course as large as L. However, the dual representation C can be used to compute the entries of K. We first eigendecompose the dual representation as $C = \sum_{n=1}^{D} \lambda_n \hat{\boldsymbol{v}}_n \hat{\boldsymbol{v}}_n^\top$, which requires $O(D^\omega)$ time. Then, we can use the definition of K in terms of the eigendecomposition of L as well as Proposition 3.1 to compute

$$K_{ii} = \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} (B_i^\top \hat{\boldsymbol{v}}_n)^2 \tag{3.27}$$

$$= q_i^2 \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} (\phi_i^\top \hat{\boldsymbol{v}}_n)^2. \tag{3.28}$$

That is, the diagonal entries of K are computable from the dot products between the diversity features ϕ_i and the eigenvectors of C; we can therefore compute the marginal probability of a single item $i \in \mathcal{Y}$ from an eigendecomposition of C in $O(D^2)$ time. Similarly, given two items

i and j we have

$$K_{ij} = \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} (B_i^\top \hat{\boldsymbol{v}}_n)(B_j^\top \hat{\boldsymbol{v}}_n) \tag{3.29}$$

$$= q_i q_j \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} (\phi_i^\top \hat{\boldsymbol{v}}_n)(\phi_j^\top \hat{\boldsymbol{v}}_n), \tag{3.30}$$

so we can compute arbitrary entries of K in $O(D^2)$ time. This allows us to compute, for example, pairwise marginals $\mathcal{P}(i, j \in \boldsymbol{Y}) = K_{ii}K_{jj} - K_{ij}^2$. More generally, for a set $A \in \mathcal{Y}$, $|A| = k$, we need to compute $\frac{k(k+1)}{2}$ entries of K to obtain K_A, and taking the determinant then yields $\mathcal{P}(A \subseteq \boldsymbol{Y})$. The process requires only $O(D^2k^2 + k^\omega)$ time; for small sets $|A|$ this is just quadratic in the dimension of ϕ.

3.3.3 Sampling

Recall the DPP sampling algorithm, which is reproduced for convenience in Algorithm 2. We will show that this algorithm can be implemented in a tractable manner by using the dual representation C. The main idea is to represent V, the orthonormal set of vectors in $\mathbb{R}^N$, as a set $\hat{V}$ of vectors in $\mathbb{R}^D$, with the mapping

$$V = \{B^\top \hat{\boldsymbol{v}} \mid \hat{\boldsymbol{v}} \in \hat{V}\}. \tag{3.31}$$

Note that, when $\hat{V}$ contains eigenvectors of C, this is (up to scale) the relationship established by Proposition 3.1 between eigenvectors $\hat{\boldsymbol{v}}$ of C and eigenvectors $\boldsymbol{v}$ of L.

The mapping in Equation (3.31) has several useful properties. If $\boldsymbol{v}_1 = B^\top \hat{\boldsymbol{v}}_1$ and $\boldsymbol{v}_2 = B^\top \hat{\boldsymbol{v}}_2$, then $\boldsymbol{v}_1 + \boldsymbol{v}_2 = B^\top(\hat{\boldsymbol{v}}_1 + \hat{\boldsymbol{v}}_2)$, and likewise for any arbitrary linear combination. In other words, we can perform implicit scaling and addition of the vectors in V using only their preimages in $\hat{V}$. Additionally, we have

$$\boldsymbol{v}_1^\top \boldsymbol{v}_2 = (B^\top \hat{\boldsymbol{v}}_1)^\top (B^\top \hat{\boldsymbol{v}}_2) \tag{3.32}$$

$$= \hat{\boldsymbol{v}}_1^\top C \hat{\boldsymbol{v}}_2, \tag{3.33}$$

so we can compute dot products of vectors in V in $O(D^2)$ time. This means that, for instance, we can implicitly normalize $\boldsymbol{v} = B^\top \hat{\boldsymbol{v}}$ by updating $\hat{\boldsymbol{v}} \leftarrow \frac{\hat{\boldsymbol{v}}}{\hat{\boldsymbol{v}}^\top C \hat{\boldsymbol{v}}}$.

Algorithm 2 Sampling from a DPP

Input: eigendecomposition $\{(\boldsymbol{v}_n, \lambda_n)\}_{n=1}^N$ of L
$J \leftarrow \emptyset$
for $n = 1, 2, \ldots, N$ **do**
  $J \leftarrow J \cup \{n\}$ with prob. $\frac{\lambda_n}{\lambda_n + 1}$
end for
$V \leftarrow \{\boldsymbol{v}_n\}_{n \in J}$
$Y \leftarrow \emptyset$
while $|V| > 0$ **do**
  Select i from $\mathcal{Y}$ with $\Pr(i) = \frac{1}{|V|} \sum_{\boldsymbol{v} \in V} (\boldsymbol{v}^\top \boldsymbol{e}_i)^2$
  $Y \leftarrow Y \cup i$
  $V \leftarrow V_\perp$, an orthonormal basis for the subspace of V orthogonal to $\boldsymbol{e}_i$
end while
Output: Y

We now show how these operations allow us to efficiently implement key parts of the sampling algorithm. Because the nonzero eigenvalues of L and C are equal, the first loop of the algorithm, where we choose in index set J, remains unchanged. Rather than using J to construct orthonormal V directly, however, we instead build the set $\hat{V}$ by adding $\frac{\hat{\boldsymbol{v}}_n}{\hat{\boldsymbol{v}}_n^\top C \hat{\boldsymbol{v}}_n}$ for every $n \in J$.

In the last phase of the loop, we need to find an orthonormal basis $V_\perp$ for the subspace of V orthogonal to a given $\boldsymbol{e}_i$. This requires two steps. In the first, we subtract a multiple of one of the vectors in V from all of the other vectors so that they are zero in the i-th component, leaving us with a set of vectors spanning the subspace of V orthogonal to $\boldsymbol{e}_i$. In order to do this we must be able to compute the i-th component of a vector $\boldsymbol{v} \in V$; since $\boldsymbol{v} = B^\top \hat{\boldsymbol{v}}$, this is easily done by computing the i-th column of B, and then taking the dot product with $\hat{\boldsymbol{v}}$. This takes only $O(D)$ time. In the second step, we use the Gram–Schmidt process to convert the resulting vectors into an orthonormal set. This requires a series of dot products, sums, and scalings of vectors in V; however, as previously argued all of these operations can be performed implicitly. Therefore the mapping in Equation (3.31) allows us to implement

the final line of the second loop using only tractable computations on vectors in $\hat{V}$.

All that remains, then, is to efficiently choose an item i according to the distribution

$$\Pr(i) = \frac{1}{|V|} \sum_{\boldsymbol{v} \in V} (\boldsymbol{v}^\top \boldsymbol{e}_i)^2 \tag{3.34}$$

$$= \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} ((B^\top \hat{\boldsymbol{v}})^\top \boldsymbol{e}_i)^2 \tag{3.35}$$

in the first line of the while loop. Simplifying, we have

$$\Pr(i) = \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} (\hat{\boldsymbol{v}}^\top B_i)^2. \tag{3.36}$$

Thus the required distribution can be computed in time $O(NDk)$, where $k = |\hat{V}|$. The complete dual sampling algorithm is given in Algorithm 3; the overall runtime is $O(NDk^2 + D^2k^3)$.

3.4 Random Projections

As we have seen, dual DPPs allow us to deal with settings where N is too large to work efficiently with L by shifting the computational focus to the dual kernel C, which is only $D \times D$. This is an effective approach when $D \ll N$. Of course, in some cases D might also be unmanageably large, for instance when the diversity features are given by word counts in natural language settings, or high-resolution image features in vision.

To address this problem, we describe a method for reducing the dimension of the diversity features while maintaining a close approximation to the original DPP model. Our approach is based on applying random projections, an extremely simple technique that nonetheless provides an array of theoretical guarantees, particularly with respect to preserving distances between points [151]. A classic result of Johnson and Lindenstrauss [76], for instance, shows that high-dimensional points can be randomly projected onto a logarithmic number of dimensions while approximately preserving the distances between them. More recently, Magen and Zouzias [99] extended this idea to the preservation of volumes spanned by sets of points. Here, we apply the connection

Algorithm 3 Sampling from a DPP (dual representation)

Input: eigendecomposition $\{(\hat{\boldsymbol{v}}_n, \lambda_n)\}_{n=1}^N$ of C
$J \leftarrow \emptyset$
for $n = 1, 2, \ldots, N$ **do**
  $J \leftarrow J \cup \{n\}$ with prob. $\frac{\lambda_n}{\lambda_n + 1}$
end for
$V \leftarrow \left\{ \frac{\hat{\boldsymbol{v}}_n}{\hat{\boldsymbol{v}}^\top C \hat{\boldsymbol{v}}} \right\}_{n \in J}$
$Y \leftarrow \emptyset$
while $|V| > 0$ **do**
  Select i from $\mathcal{Y}$ with $\Pr(i) = \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} (\hat{\boldsymbol{v}}^\top B_i)^2$
  $Y \leftarrow Y \cup i$
  Let $\hat{\boldsymbol{v}}_0$ be a vector in $\hat{V}$ with $B_i^\top \hat{\boldsymbol{v}}_0 \neq 0$
  Update $\hat{V} \leftarrow \left\{ \hat{\boldsymbol{v}} - \frac{\hat{\boldsymbol{v}}^\top B_i}{\hat{\boldsymbol{v}}_0^\top B_i} \hat{\boldsymbol{v}}_0 \mid \hat{\boldsymbol{v}} \in \hat{V} - \{\hat{\boldsymbol{v}}_0\} \right\}$
  Orthonormalize $\hat{V}$ with respect to the dot product $\langle \hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2 \rangle = \hat{\boldsymbol{v}}_1^\top C \hat{\boldsymbol{v}}_2$
end while
Output: Y

between DPPs and spanned volumes to show that random projections allow us to reduce the dimensionality of ϕ, dramatically speeding up inference, while maintaining a provably close approximation to the original, high-dimensional model. We begin by stating a variant of Magen and Zouzias' result.

Lemma 3.2. (Adapted from Magen and Zouzias [99]) Let X be a $D \times N$ matrix. Fix $k < N$ and $0 < \epsilon, \delta < 1/2$, and set the projection dimension

$$d = \max\left\{ \frac{2k}{\epsilon}, \frac{24}{\epsilon^2} \left(\frac{\log(3/\delta)}{\log N} + 1 \right) (\log N + 1) + k - 1 \right\}. \tag{3.37}$$

Let G be a $d \times D$ random projection matrix whose entries are independently sampled from $\mathcal{N}(0, \frac{1}{d})$, and let X_Y, where $Y \subseteq \{1, 2, \ldots, N\}$, denote the $D \times |Y|$ matrix formed by taking the columns of X corresponding to indices in Y. Then with probability at least $1 - \delta$ we have,

for all Y with cardinality at most k:

$$(1-\epsilon)^{|Y|} \leq \frac{\mathrm{Vol}(GX_Y)}{\mathrm{Vol}(X_Y)} \leq (1+\epsilon)^{|Y|}, \tag{3.38}$$

where $\mathrm{Vol}(X_Y)$ is the k-dimensional volume of the parallelepiped spanned by the columns of X_Y.

Lemma 3.2 says that, with high probability, randomly projecting to

$$d = O(\max\{k/\epsilon, (\log(1/\delta) + \log N)/\epsilon^2\}) \tag{3.39}$$

dimensions is sufficient to approximately preserve all volumes spanned by k columns of X. We can use this result to bound the effectiveness of random projections for DPPs.

In order to obtain a result that is independent of D, we will restrict ourselves to the portion of the distribution pertaining to subsets Y with cardinality at most a constant k. This restriction is intuitively reasonable for any application where we use DPPs to model sets of relatively small size compared to N, which is common in practice. However, formally it may seem a bit strange, since it implies conditioning the DPP on cardinality. In Section 5 we will show that this kind of conditioning is actually very practical and efficient, and Theorem 3.3, which we prove shortly, will apply directly to the k-DPPs of Section 5 without any additional work.

For now, we will seek to approximate the distribution $\mathcal{P}^{\leq k}(Y) = \mathcal{P}(\boldsymbol{Y} = Y \mid |\boldsymbol{Y}| \leq k)$, which is simply the original DPP conditioned on the cardinality of the modeled subset:

$$\mathcal{P}^{\leq k}(Y) = \frac{\left(\prod_{i \in Y} q_i^2\right) \det(\phi(Y)^\top \phi(Y))}{\sum_{|Y'| \leq k} \left(\prod_{i \in Y} q_i^2\right) \det(\phi(Y)^\top \phi(Y))}, \tag{3.40}$$

where $\phi(Y)$ denotes the $D \times |Y|$ matrix formed from columns ϕ_i for $i \in Y$. Our main result follows.

Theorem 3.3. Let $\mathcal{P}^{\leq k}(Y)$ be the cardinality-conditioned DPP distribution defined by quality model q and D-dimensional diversity feature function ϕ, and let

$$\tilde{\mathcal{P}}^{\leq k}(Y) \propto \left(\prod_{i \in Y} q_i^2\right) \det([G\phi(Y)]^\top [G\phi(Y)]) \tag{3.41}$$

be the cardinality-conditioned DPP distribution obtained by projecting ϕ with G. Then for projection dimension d as in Equation (3.37), we have

$$\|\mathcal{P}^{\leq k} - \tilde{\mathcal{P}}^{\leq k}\|_1 \leq e^{6k\epsilon} - 1 \tag{3.42}$$

with probability at least $1 - \delta$. Note that $e^{6k\epsilon} - 1 \approx 6k\epsilon$ when $k\epsilon$ is small.

The theorem says that for d logarithmic in N and linear in k, the L_1 variational distance between the original DPP and the randomly projected version is bounded. In order to use Lemma 3.2, which bounds volumes of parallelepipeds, to prove this bound on determinants, we will make use of the following relationship:

$$\mathrm{Vol}(X_Y) = \sqrt{\det(X_Y^\top X_Y)}. \tag{3.43}$$

In order to handle the conditional DPP normalization constant

$$\sum_{|Y|\leq k} \left(\prod_{i \in Y} q_i^2 \right) \det(\phi(Y)^\top \phi(Y)), \tag{3.44}$$

we also must adapt Lemma 3.2 to *sums* of determinants. Finally, for technical reasons we will change the symmetry of the upper and lower bounds from the sign of ϵ to the sign of the exponent. The following lemma gives the details.

Lemma 3.4. Under the definitions and assumptions of Lemma 3.2, we have, with probability at least $1 - \delta$,

$$(1 + 2\epsilon)^{-2k} \leq \frac{\sum_{|Y|\leq k} \det((GX_Y)^\top (GX_Y))}{\sum_{|Y|\leq k} \det(X_Y^\top X_Y)} \leq (1 + \epsilon)^{2k}. \tag{3.45}$$

Proof.

$$\sum_{|Y|\leq k} \det((GX_Y)^\top (GX_Y)) = \sum_{|Y|\leq k} \mathrm{Vol}^2(GX_Y) \tag{3.46}$$

$$\geq \sum_{|Y|\leq k} (\mathrm{Vol}(X_Y)(1 - \epsilon)^{|Y|})^2 \tag{3.47}$$

$$\geq (1-\epsilon)^{2k} \sum_{|Y|\leq k} \mathrm{Vol}^2(X_Y) \tag{3.48}$$

$$\geq (1+2\epsilon)^{-2k} \sum_{|Y|\leq k} \det(X_Y^\top X_Y), \tag{3.49}$$

where the first inequality holds with probability at least $1-\delta$ by Lemma 3.2, and the third follows from the fact that $(1-\epsilon)(1+2\epsilon) \geq 1$ (since $\epsilon < 1/2$), thus $(1-\epsilon)^{2k} \geq (1+2\epsilon)^{-2k}$. The upper bound follows directly:

$$\sum_{|Y|\leq k} (\mathrm{Vol}(GX_Y))^2 \leq \sum_{|Y|\leq k} (\mathrm{Vol}(X_Y)(1+\epsilon)^{|Y|})^2 \tag{3.50}$$

$$\leq (1+\epsilon)^{2k} \sum_{|Y|\leq k} \det(X_Y^\top X_Y). \tag{3.51}$$

We can now prove Theorem 3.3. □

Proof of Theorem 3.3. Recall the matrix B, whose columns are given by $B_i = q_i \phi_i$. We have

$$\begin{aligned}
&\|P^{\leq k} - \tilde{P}^{\leq k}\|_1 \\
&\quad = \sum_{|Y|\leq k} |P^{\leq k}(Y) - \tilde{P}^{\leq k}(Y)| && (3.52)\\
&\quad = \sum_{|Y|\leq k} P^{\leq k}(Y) \left| 1 - \frac{\tilde{P}^{\leq k}(Y)}{P^{\leq k}(Y)} \right| && (3.53)\\
&\quad = \sum_{|Y|\leq k} P^{\leq k}(Y) \\
&\qquad \times \left| 1 - \frac{\det([GB_Y^\top][GB_Y])}{\det(B_Y^\top B_Y)} \frac{\sum_{|Y'|\leq k} \det(B_{Y'}^\top B_{Y'})}{\sum_{|Y'|\leq k} \det([GB_{Y'}^\top][GB_{Y'}])} \right| \\
&\quad \leq \left| 1 - (1+\epsilon)^{2k}(1+2\epsilon)^{2k} \right| \sum_{|Y|\leq k} P^{\leq k}(Y) && (3.54)\\
&\quad \leq e^{6k\epsilon} - 1, && (3.55)
\end{aligned}$$

where the first inequality follows from Lemma 3.2 and Lemma 3.4, which hold simultaneously with probability at least $1 - \delta$, and the second follows from $(1 + a)^b \leq e^{ab}$ for $a, b \geq 0$. □

By combining the dual representation with random projections, we can deal simultaneously with very large N and very large D. In fact, in Section 6 we will show that N can even be exponentially large if certain structural assumptions are met. These techniques vastly expand the range of problems to which DPPs can be practically applied.

3.5 Alternative Likelihood Formulas

Recall that, in an L-ensemble DPP, the likelihood of a particular set $Y \subseteq \mathcal{Y}$ is given by

$$\mathcal{P}_L(Y) = \frac{\det(L_Y)}{\det(L + I)}. \tag{3.56}$$

This expression has some nice intuitive properties in terms of volumes, and, ignoring the normalization in the denominator, takes a simple and concise form. However, as a ratio of determinants on matrices of differing dimension, it may not always be analytically convenient. Minors can be difficult to reason about directly, and ratios complicate calculations like derivatives. Moreover, we might want the likelihood in terms of the marginal kernel $K = L(L + I)^{-1} = I - (L + I)^{-1}$, but simply plugging in these identities yields a expression that is somewhat unwieldy.

As alternatives, we will derive some additional formulas that, depending on context, may have useful advantages. Our starting point will be the observation, used previously in the proof of Theorem 2.2, that minors can be written in terms of full matrices and diagonal indicator matrices; specifically, for positive semidefinite L,

$$\begin{aligned} \det(L_Y) &= \det(I_Y L + I_{\bar{Y}}) && (3.57)\\ &= (-1)^{|\bar{Y}|} \det(I_Y L - I_{\bar{Y}}) && (3.58)\\ &= |\det(I_Y L - I_{\bar{Y}})|, && (3.59) \end{aligned}$$

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

These identities can be easily shown by examining the matrices blockwise under the partition $\mathcal{Y} = Y \cup \bar{Y}$, as in the proof of Theorem 2.2.

Applying Equation (3.57) to Equation (3.56), we get

$$\mathcal{P}_L(Y) = \frac{\det(I_Y L + I_{\bar{Y}})}{\det(L+I)} \tag{3.60}$$

$$= \det((I_Y L + I_{\bar{Y}})(L+I)^{-1}) \tag{3.61}$$

$$= \det(I_Y L(L+I)^{-1} + I_{\bar{Y}}(L+I)^{-1}). \tag{3.62}$$

Already, this expression, which is a single determinant of an $N \times N$ matrix, is in some ways easier to work with. We can also more easily write the likelihood in terms of K:

$$\mathcal{P}_L(Y) = \det(I_Y K + I_{\bar{Y}}(I - K)). \tag{3.63}$$

Recall from Equation (2.27) that $I - K$ is the marginal kernel of the complement DPP; thus, in an informal sense we can read Equation (3.63) as combining the marginal probability that Y is selected with the marginal probability that $\bar{Y}$ is not selected.

We can also make a similar derivation using Equation (3.58):

$$\mathcal{P}_L(Y) = (-1)^{|\bar{Y}|}\frac{\det(I_Y L - I_{\bar{Y}})}{\det(L+I)} \tag{3.64}$$

$$= (-1)^{|\bar{Y}|}\det((I_Y L - I_{\bar{Y}})(L+I)^{-1}) \tag{3.65}$$

$$= (-1)^{|\bar{Y}|}\det(I_Y L(L+I)^{-1} - I_{\bar{Y}}(L+I)^{-1}) \tag{3.66}$$

$$= (-1)^{|\bar{Y}|}\det(I_Y K - I_{\bar{Y}}(I-K)) \tag{3.67}$$

$$= (-1)^{|\bar{Y}|}\det(K - I_{\bar{Y}}) \tag{3.68}$$

$$= |\det(K - I_{\bar{Y}})|. \tag{3.69}$$

Note that Equation (3.63) involves asymmetric matrix products, but Equation (3.69) does not; on the other hand, $K - I_{\bar{Y}}$ is (in general) indefinite.

4

Learning

We have seen that determinantal point process offer appealing modeling intuitions and practical algorithms, capturing geometric notions of diversity and permitting computationally efficient inference in a variety of settings. However, to accurately model real-world data we must first somehow determine appropriate values of the model parameters. While an expert could conceivably design an appropriate DPP kernel from prior knowledge, in general, especially when dealing with large datasets, we would like to have an automated method for learning a DPP.

We first discuss how to parameterize DPPs conditioned on input data. We then define what we mean by learning, and, using the quality versus diversity decomposition introduced in Section 3.1, we show how a parameterized quality model can be learned efficiently from a training set.

4.1 Conditional DPPs

Suppose we want to use a DPP to model the seats in an auditorium chosen by students attending a class. (Perhaps we think students tend

to spread out.) In this context each meeting of the class is a new sample from the empirical distribution over subsets of the (fixed) seats, so we merely need to collect enough samples and we should be able to fit our model, as desired.

For many problems, however, the notion of a single fixed base set $\mathcal{Y}$ is inadequate. For instance, consider extractive document summarization, where the goal is to choose a subset of the sentences in a news article that together form a good summary of the entire article. In this setting $\mathcal{Y}$ is the set of sentences in the news article being summarized, thus $\mathcal{Y}$ is not fixed in advance but instead depends on context. One way to deal with this problem is to model the summary for each article as its own DPP with a separate kernel matrix. This approach certainly affords great flexibility, but if we have only a single sample summary for each article, there is little hope of getting good parameter estimates. Even more importantly, we have learned nothing that can be applied to generate summaries of unseen articles at test time, which was presumably our goal in the first place.

Alternatively, we could let $\mathcal{Y}$ be the set of all sentences appearing in *any* news article; this allows us to learn a single model for all of our data, but comes with obvious computational issues and does not address the other concerns, since sentences are rarely repeated.

To solve this problem, we need a DPP that depends parametrically on the input data; this will enable us to share information across training examples in a justifiable and effective way. We first introduce some notation. Let $\mathcal{X}$ be the input space; for example, $\mathcal{X}$ might be the space of news articles. Let $\mathcal{Y}(X)$ denote the ground set of items implied by an input $X \in \mathcal{X}$, e.g., the set of all sentences in news article X. We have the following definition.

Definition 4.1. A **conditional DPP** $\mathcal{P}(\boldsymbol{Y} = Y|X)$ is a conditional probabilistic model which assigns a probability to every possible subset $Y \subseteq \mathcal{Y}(X)$. The model takes the form of an L-ensemble:

$$\mathcal{P}(\boldsymbol{Y} = Y|X) \propto \det(L_Y(X)), \tag{4.1}$$

where $L(X)$ is a positive semidefinite $|\mathcal{Y}(X)| \times |\mathcal{Y}(X)|$ kernel matrix that depends on the input.

As discussed in Section 2, the normalization constant for a conditional DPP can be computed efficiently and is given by $\det(L(X) + I)$. Using the quality/diversity decomposition introduced in Section 3.1, we have

$$L_{ij}(X) = q_i(X)\phi_i(X)^\top \phi_j(X) q_j(X) \tag{4.2}$$

for suitable $q_i(X) \in \mathbb{R}^+$ and $\phi_i(X) \in \mathbb{R}^D$, $\|\phi_i(X)\| = 1$, which now depend on X.

In the following sections we will discuss application-specific parameterizations of the quality and diversity models q and ϕ in terms of the input. First, however, we review our learning setup.

4.1.1 Supervised Learning

The basic supervised learning problem is as follows. We receive a training data sample $\{(X^{(t)}, Y^{(t)})\}_{t=1}^T$ drawn independently and identically from a distribution D over pairs $(X, Y) \in \mathcal{X} \times 2^{\mathcal{Y}(X)}$, where $\mathcal{X}$ is an input space and $\mathcal{Y}(X)$ is the associated ground set for input X. We assume that the conditional DPP kernel $L(X;\theta)$ is parameterized in terms of a generic θ, and let

$$\mathcal{P}_\theta(Y|X) = \frac{\det(L_Y(X;\theta))}{\det(L(X;\theta) + I)} \tag{4.3}$$

denote the conditional probability of an output Y, given input X under parameter θ. The goal of learning is to choose appropriate θ based on the training sample so that we can make accurate predictions on unseen inputs.

While there are a variety of objective functions commonly used for learning, here we will focus on *maximum likelihood* learning (or maximum likelihood estimation, often abbreviated MLE), where the goal is to choose θ to maximize the conditional log-likelihood of the observed data:

$$\mathcal{L}(\theta) = \log \prod_{t=1}^T \mathcal{P}_\theta(Y^{(t)}|X^{(t)}) \tag{4.4}$$

$$= \sum_{t=1}^{T} \log \mathcal{P}_\theta(Y^{(t)}|X^{(t)}) \tag{4.5}$$

$$= \sum_{t=1}^{T} [\log\det(L_{Y^{(t)}}(X^{(t)};\theta)) - \log\det(L(X^{(t)};\theta) + I)]. \tag{4.6}$$

Optimizing $\mathcal{L}$ is consistent under mild assumptions; that is, if the training data are actually drawn from a conditional DPP with parameter θ^*, then the learned $\theta \to \theta^*$ as $T \to \infty$. Of course real data are unlikely to exactly follow any particular model, but in any case the maximum likelihood approach has the advantage of calibrating the DPP to produce reasonable probability estimates, since maximizing $\mathcal{L}$ can be seen as minimizing the log-loss on the training data.

To optimize the log-likelihood, we will use standard algorithms such as gradient ascent or L-BFGS [113]. These algorithms depend on the gradient $\nabla\mathcal{L}(\theta)$, which must exist and be computable, and they converge to the optimum whenever $\mathcal{L}(\theta)$ is concave in θ. Thus, our ability to optimize likelihood efficiently will depend fundamentally on these two properties.

4.2 Learning Quality

We begin by showing how to learn a parameterized quality model $q_i(X;\theta)$ when the diversity feature function $\phi_i(X)$ is held fixed [85]. This setup is somewhat analogous to support vector machines [149], where a kernel is fixed by the practitioner and then the per-example weights are automatically learned. Here, $\phi_i(X)$ can consist of any desired measurements (and could even be infinite-dimensional, as long as the resulting similarity matrix S is a proper kernel). We propose computing the quality scores using a log-linear model:

$$q_i(X;\theta) = \exp\left(\frac{1}{2}\theta^\top \boldsymbol{f}_i(X)\right), \tag{4.7}$$

where $\boldsymbol{f}_i(X) \in \mathbb{R}^m$ is a feature vector for item i and the parameter θ is now concretely an element of $\mathbb{R}^m$. Note that feature vectors $\boldsymbol{f}_i(X)$ are in general distinct from $\phi_i(X)$; the former are used for modeling quality, and will be "interpreted" by the parameters θ, while the latter

define the diversity model S, which is fixed in advance. We have

$$\mathcal{P}_\theta(Y|X) = \frac{\prod_{i\in Y}[\exp(\theta^\top \boldsymbol{f}_i(X))]\det(S_Y(X))}{\sum_{Y'\subseteq\mathcal{Y}(X)}\prod_{i\in Y'}[\exp(\theta^\top \boldsymbol{f}_i(X))]\det(S_{Y'}(X))}. \tag{4.8}$$

For ease of notation, going forward we will assume that the training set contains only a single instance (X,Y), and drop the instance index t. All of the following results extend easily to multiple training examples. First, we show that under this parameterization the log-likelihood function is concave in θ; then we will show that its gradient can be computed efficiently. With these results in hand we will be able to apply standard optimization techniques.

Proposition 4.1. $\mathcal{L}(\theta)$ is concave in θ.

Proof. We have

$$\begin{aligned}\mathcal{L}(\theta) &= \log \mathcal{P}_\theta(Y|X) &(4.9)\\ &= \theta^\top \sum_{i\in Y} \boldsymbol{f}_i(X) + \log\det(S_Y(X)) \\ &\quad - \log \sum_{Y'\subseteq\mathcal{Y}(X)} \exp\left(\theta^\top \sum_{i\in Y'} \boldsymbol{f}_i(X)\right)\det(S_{Y'}(X)). &(4.10)\end{aligned}$$

With respect to θ, the first term is linear, the second is constant, and the third is the composition of a concave function (negative log-sum-exp) and an affine function, so the overall expression is concave. □

We now derive the gradient $\nabla\mathcal{L}(\theta)$, using Equation (4.10) as a starting point.

$$\nabla\mathcal{L}(\theta) = \sum_{i\in Y}\boldsymbol{f}_i(X) - \nabla\left[\log\sum_{Y'\subseteq\mathcal{Y}(X)}\exp\left(\theta^\top\sum_{i\in Y'}\boldsymbol{f}_i(X)\right)\det(S_{Y'}(X))\right] \tag{4.11}$$

$$= \sum_{i \in Y} \boldsymbol{f}_i(X) - \sum_{Y' \subseteq \mathcal{Y}(X)} \frac{\exp\left(\theta^\top \sum_{i \in Y'} \boldsymbol{f}_i(X)\right) \det(S_{Y'}(X)) \sum_{i \in Y'} \boldsymbol{f}_i(X)}{\sum_{Y'} \exp\left(\theta^\top \sum_{i \in Y'} \boldsymbol{f}_i(X)\right) \det(S_{Y'}(X))} \tag{4.12}$$

$$= \sum_{i \in Y} \boldsymbol{f}_i(X) - \sum_{Y' \subseteq \mathcal{Y}(X)} \mathcal{P}_\theta(Y'|X) \sum_{i \in Y'} \boldsymbol{f}_i(X). \tag{4.13}$$

Thus, as in standard maximum entropy modeling, the gradient of the log-likelihood can be seen as the difference between the empirical feature counts and the expected feature counts under the model distribution. The difference here, of course, is that $\mathcal{P}_\theta$ is a DPP, which assigns higher probability to diverse sets. Compared with a standard independent model obtained by removing the diversity term from $\mathcal{P}_\theta$, Equation (4.13) actually emphasizes those training examples that are *not* diverse, since these are the examples on which the quality model must focus its attention in order to overcome the bias imposed by the determinant. In the experiments that follow we will see that this distinction is important in practice.

The sum over Y' in Equation (4.13) is exponential in $|\mathcal{Y}(X)|$; hence we cannot compute it directly. Instead, we can rewrite it by switching the order of summation:

$$\sum_{Y' \subseteq \mathcal{Y}(X)} \mathcal{P}_\theta(Y'|X) \sum_{i \in Y'} \boldsymbol{f}_i(X) = \sum_i \boldsymbol{f}_i(X) \sum_{Y' \supseteq \{i\}} \mathcal{P}_\theta(Y'|X). \tag{4.14}$$

Note that $\sum_{Y' \supseteq \{i\}} \mathcal{P}_\theta(Y'|X)$ is the marginal probability of item i appearing in a set sampled from the conditional DPP. That is, the expected feature counts are computable directly from the marginal probabilities. Recall that we can efficiently marginalize DPPs; in particular, per-item marginal probabilities are given by the diagonal of $K(X;\theta)$, the marginal kernel (which now depends on the input and the parameters). We can compute $K(X;\theta)$ from the kernel $L(X;\theta)$ using matrix inversion or eigendecomposition. Algorithm 4 shows how we can use these ideas to compute the gradient of $\mathcal{L}(\theta)$ efficiently.

In fact, note that we do not need all of $K(X;\theta)$, but only its diagonal. In Algorithm 4 we exploit this in the main loop, using only $O(N^2)$

Algorithm 4 Gradient of the log-likelihood

Input: instance (X, Y), parameters θ
Compute $L(X;\theta)$ as in Equation (4.2)
Eigendecompose $L(X;\theta) = \sum_{n=1}^{N} \lambda_n \boldsymbol{v}_n \boldsymbol{v}_n^\top$
for $i \in \mathcal{Y}(X)$ **do**
  $K_{ii} \leftarrow \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + 1} \boldsymbol{v}_{ni}^2$
end for
$\nabla \mathcal{L}(\theta) \leftarrow \sum_{i \in Y} \boldsymbol{f}_i(X) - \sum_i K_{ii} \boldsymbol{f}_i(X)$
Output: gradient $\nabla \mathcal{L}(\theta)$

multiplications rather than the $O(N^3)$ we would need to construct the entire marginal kernel. (In the dual representation, this can be improved further to $O(ND)$ multiplications.) Unfortunately, these savings are asymptotically irrelevant since we still need to eigendecompose $L(X;\theta)$, requiring about $O(N^3)$ time (or $O(D^3)$ time for the corresponding eigendecomposition in the dual). It is conceivable that a faster algorithm exists for computing the diagonal of $K(X;\theta)$ directly, along the lines of ideas recently proposed by [144] (which focus on sparse matrices); however, we are not currently aware of a useful improvement over Algorithm 4.

4.2.1 Experiments: Document Summarization

We demonstrate learning for the conditional DPP quality model on an extractive multi-document summarization task using news text. The basic goal is to generate a short piece of text that summarizes the most important information from a news story. In the *extractive* setting, the summary is constructed by stringing together sentences found in a cluster of relevant news articles. This selection problem is a balancing act: on the one hand, each selected sentence should be relevant, sharing significant information with the cluster as a whole; on the other, the selected sentences should be diverse as a group so that the summary is not repetitive and is as informative as possible, given its length [34, 111]. DPPs are a natural fit for this task, viewed through the decomposition of Section 3.1 [85].

As in Section 4.1, the input X will be a cluster of documents, and $\mathcal{Y}(X)$ a set of candidate sentences from those documents. In our experiments $\mathcal{Y}(X)$ contains all sentences from all articles in the cluster, although in general preprocessing could also be used to try to improve the candidate set [29]. We will learn a DPP to model good summaries Y for a given input X. Because DPPs model unordered sets while summaries are linear text, we construct a written summary from Y by placing the sentences it contains in the same order in which they appeared in the original documents. This policy is unlikely to give optimal results, but it is consistent with prior work [94] and seems to perform well. Furthermore, it is at least partially justified by the fact that modern automatic summary evaluation metrics like ROUGE, which we describe later, are mostly invariant to sentence order.

We experiment with data from the multidocument summarization task (Task 2) of the 2003 and 2004 Document Understanding Conference (DUC) [34]. The article clusters used for these tasks are taken from the NIST TDT collection. Each cluster contains approximately ten articles drawn from the AP and New York Times newswires, and covers a single topic over a short time span. The clusters have a mean length of approximately 250 sentences and 5800 words. The 2003 task, which we use for training, contains 30 clusters, and the 2004 task, which is our test set, contains 50 clusters. Each cluster comes with four reference human summaries (which are not necessarily formed by sentences from the original articles) for evaluation purposes. Summaries are required to be at most 665 characters in length, including spaces. Figure 4.1 depicts a sample cluster from the test set.

To measure performance on this task we follow the original evaluation and use ROUGE, an automatic evaluation metric for summarization [93]. ROUGE measures n-gram overlap statistics between the human references and the summary being scored, and combines them to produce various submetrics. ROUGE-1, for example, is a simple unigram recall measure that has been shown to correlate quite well with human judgments [93]. Here, we use ROUGE's unigram F-measure (which combines ROUGE-1 with a measure of precision) as our primary metric for development. We refer to this measure as ROUGE-1F. We also report ROUGE-1P and ROUGE-1R (precision

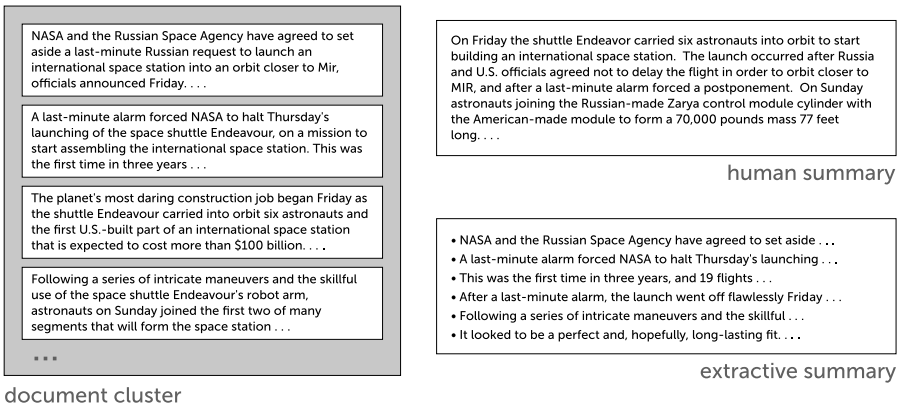


Fig. 4.1 A sample cluster from the DUC 2004 test set, with one of the four human reference summaries and an (artificial) extractive summary.

and recall, respectively) as well as ROUGE-2F and ROUGE-SU4F, which include bigram match statistics and have also been shown to correlate well with human judgments. Our implementation uses ROUGE version 1.5.5 with stemming turned on, but without stopword removal. These settings correspond to those used for the actual DUC competitions [34]; however, we use a more recent version of ROUGE.

Training data Recall that our learning setup requires a training sample of pairs (X, Y), where $Y \subseteq \mathcal{Y}(X)$. Unfortunately, while the human reference summaries provided with the DUC data are of high quality, they are not extractive, thus they do not serve as examples of summaries that we can actually model. To obtain high-quality extractive "oracle" summaries from the human summaries, we employ a simple greedy algorithm (Algorithm 5). On each round the sentence that achieves maximal unigram F-measure to the human references, normalized by length, is selected and added to the extractive summary. Since high F-measure requires high precision as well as recall, we then update the references by removing the words "covered" by the newly selected sentence and proceed to the next round.

We can measure the success of this approach by calculating ROUGE scores of our oracle summaries with respect to the human summaries. Table 4.1 shows the results for the DUC 2003 training set. For reference,

Algorithm 5 Constructing extractive training data

Input: article cluster X, human reference word counts H, character limit b
$U \leftarrow \mathcal{Y}(X)$
$Y \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  $i \leftarrow \arg\max_{i' \in U} \left(\frac{\text{ROUGE-1F}(\text{words}(i'), H)}{\sqrt{\text{length}(i')}} \right)$
  $Y \leftarrow Y \cup \{i\}$
  $H \leftarrow \max(H - \text{words}(i), 0)$
  $U \leftarrow U - (\{i\} \cup \{i' | \text{length}(Y) + \text{length}(i') > b\})$
end while
Output: extractive oracle summary Y

Table 4.1. ROUGE scores for the best automatic system from DUC 2003, our heuristically generated oracle extractive summaries, and human summaries.

System	ROUGE-1F	ROUGE-2F	ROUGE-SU4F
Machine	35.17	9.15	12.47
Oracle	46.59	16.18	19.52
Human	56.22	33.37	36.50

the table also includes the ROUGE scores of the best automatic system from the DUC competition in 2003 ("machine"), as well as the human references themselves ("human"). Note that, in the latter case, the human summary being evaluated is also one of the four references used to compute ROUGE; hence the scores are probably significantly higher than a human could achieve in practice. Furthermore, it has been shown that extractive summaries, even when generated optimally, are by nature limited in quality compared with unconstrained summaries [55]. Thus we believe that the oracle summaries make strong targets for training.

Features We next describe the feature functions that we use for this task. For diversity features $\phi_i(X)$, we generate standard normalized tf–idf vectors. We tokenize the input test, remove stop words and

punctuation, and apply a Porter stemmer.[1] Then, for each word w, the *term frequency* $\mathrm{tf}_i(w)$ of w in sentence i is defined as the number of times the word appears in the sentence, and the *inverse document frequency* $\mathrm{idf}(w)$ is the negative logarithm of the fraction of articles in the training set where w appears. A large value of $\mathrm{idf}(w)$ implies that w is relatively rare. Finally, the vector $\phi_i(X)$ has one element per word, and the value of the entry associated with word w is proportional to $\mathrm{tf}_i(w)\mathrm{idf}(w)$. The scale of $\phi_i(X)$ is set such that $\|\phi_i(X)\| = 1$.

Under this definition of ϕ, the similarity S_{ij} between sentences i and j is known as their *cosine similarity*:

$$S_{ij} = \frac{\sum_w \mathrm{tf}_i(w)\mathrm{tf}_j(w)\mathrm{idf}^2(w)}{\sqrt{\sum_w \mathrm{tf}_i^2(w)\mathrm{idf}^2(w)}\sqrt{\sum_w \mathrm{tf}_j^2(w)\mathrm{idf}^2(w)}} \in [0,1]. \tag{4.15}$$

Two sentences are cosine similar if they contain many of the same words, particularly words that are uncommon (and thus more likely to be salient).

We augment $\phi_i(X)$ with an additional constant feature taking the value $\rho \geq 0$, which is a hyperparameter. This has the effect of making all sentences more similar to one another, increasing repulsion. We set ρ to optimize ROUGE-1F score on the training set; in our experiments, the best choice was $\rho = 0.7$.

We use the very standard cosine distance as our similarity metric because we need to be confident that it is sensible; it will remain fixed throughout the experiments. On the other hand, weights for the quality features are learned, so we can use a variety of intuitive measures and rely on training to find an appropriate combination. The quality features we use are listed below. For some of the features, we make use of cosine distances; these are computed using the same tf–idf vectors as the diversity features. When a feature is intrinsically real-valued, we produce a series of binary features by binning. The bin boundaries are determined either globally or locally. Global bins are evenly spaced quantiles of the feature values across all sentences in the training set, while local bins are quantiles of the feature values in the current cluster only.

[1] Code for this preprocessing pipeline was provided by Hui Lin and Jeff Bilmes.

- **Constant**: A constant feature allows the model to bias toward summaries with a greater or smaller number of sentences.
- **Length**: We bin the length of the sentence (in characters) into five global bins.
- **Document position**: We compute the position of the sentence in its original document and generate binary features indicating positions 1–5, plus a sixth binary feature indicating all other positions. We expect that, for newswire text, sentences that appear earlier in an article are more likely to be useful for summarization.
- **Mean cluster similarity**: For each sentence we compute the average cosine distance to all other sentences in the cluster. This feature attempts to measure how well the sentence reflects the salient words occurring most frequently in the cluster. We use the raw score, five global bins, and ten local bins.
- **LexRank**: We compute continuous LexRank scores by finding the principal eigenvector of the row-normalized cosine similarity matrix. (See Erkan and Radev [43] for details.) This provides an alternative measure of centrality. We use the raw score, five global bins, and five local bins.
- **Personal pronouns**: We count the number of personal pronouns ("he", "her", "themselves", etc.) appearing in each sentence. Sentences with many pronouns may be poor for summarization since they omit important entity names.

In total we have 40 quality features; including ρ our model has 41 parameters.

Inference At test time, we need to take the learned parameters θ and use them to predict a summary Y for a previously unseen document cluster X. One option is to sample from the conditional distribution, which can be done exactly and efficiently, as described in Section 2.4.4. However, sampling occasionally produces low-probability predictions. We obtain better performance on this task by applying two alternative inference techniques.

Algorithm 6 Approximately computing the MAP summary

Input: document cluster X, parameter θ, character limit b
$U \leftarrow \mathcal{Y}(X)$
$Y \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  $i \leftarrow \arg\max_{i' \in U} \left(\frac{\mathcal{P}_\theta(Y \cup \{i\}|X) - \mathcal{P}_\theta(Y|X)}{\text{length}(i)} \right)$
  $Y \leftarrow Y \cup \{i\}$
  $U \leftarrow U - (\{i\} \cup \{i' | \text{length}(Y) + \text{length}(i') > b\})$
end while
Output: summary Y

Greedy MAP approximation. One common approach to prediction in probabilistic models is maximum *a posteriori* (MAP) decoding, which selects the highest probability configuration. For practical reasons, and because the primary metrics for evaluation were recall-based, the DUC evaluations imposed a length limit of 665 characters, including spaces, on all summaries. In order to compare with prior work we also apply this limit in our tests. Thus, our goal is to find the most likely summary, subject to a budget constraint:

$$Y^{\text{MAP}} = \arg\max_{Y} \; \mathcal{P}_\theta(Y|X) \quad \text{s.t.} \; \sum_{i \in Y} \text{length}(i) \leq b, \tag{4.16}$$

where length(i) is the number of characters in sentence i, and $b = 665$ is the limit on the total length. As discussed in Section 2.4.5, computing Y^{MAP} exactly is NP-hard, but, recalling that the optimization in Equation (4.16) is submodular, we can approximate it through a simple greedy algorithm (Algorithm 6).

Algorithm 6 is closely related to those given by Krause and Guestrin [80] and especially Lin and Bilmes [94]. As discussed in Section 2.4.5, algorithms of this type have formal approximation guarantees for monotone submodular problems. Our MAP problem is not generally monotone; nonetheless, Algorithm 6 seems to work well in practice, and is very fast (see Table 4.2).

Minimum Bayes risk decoding. The second inference technique we consider is minimum Bayes risk (MBR) decoding. First proposed by Goel and Byrne [59] for automatic speech recognition, MBR decoding has also been used successfully for word alignment and machine translation [86, 87]. The idea is to choose a prediction that minimizes a particular application-specific loss function under uncertainty about the evaluation target. In our setting we use ROUGE-1F as a (negative) loss function, so we have

$$Y^{\text{MBR}} = \arg\max_{Y} \; \mathbb{E}[\text{ROUGE-1F}(Y, \boldsymbol{Y}^*)], \tag{4.17}$$

where the expectation is over realizations of $\boldsymbol{Y}^*$, the true summary against which we are evaluated. Of course, the distribution of $\boldsymbol{Y}^*$ is unknown, but we can assume that our trained model $\mathcal{P}_\theta(\cdot|X)$ gives a reasonable approximation. Since there are exponentially many possible summaries, we cannot expect to perform an exact search for Y^{MBR}; however, we can approximate it through sampling, which is efficient.

Combining these approximations, we have the following inference rule:

$$\tilde{Y}^{\text{MBR}} = \arg\max_{Y^{r'},\; r' \in \{1,2,\ldots,R\}} \frac{1}{R} \sum_{r=1}^{R} \text{ROUGE-1F}(Y^{r'}, Y^{r}), \tag{4.18}$$

where $Y^1, Y^2, \ldots, Y^R$ are samples drawn from $\mathcal{P}_\theta(\cdot|X)$. In order to satisfy the length constraint imposed by the evaluation, we consider only samples with length between 660 and 680 characters (rejecting those that fall outside this range), and crop $\tilde{Y}^{\text{MBR}}$ to the limit of 665 bytes if necessary. The choice of R is a trade-off between fast running time and quality of inference. In the following section, we report results for $R = 100, 1000$, and 5000; Table 4.2 shows the average time required to produce a summary under each setting. Note that MBR decoding is easily parallelizable, but the results in Table 4.2 are for a single processor. Since MBR decoding is randomized, we report all results averaged over 100 trials.

Results We train our model with a standard L-BFGS optimization algorithm. We place a zero-mean Gaussian prior on the parameters θ,

Table 4.2. The average time required to produce a summary for a single cluster from the DUC 2004 test set (without parallelization).

System	Time (s)
DPP-GREEDY	0.15
DPP-MBR100	1.30
DPP-MBR1000	16.91
DPP-MBR5000	196.86

with variance set to optimize ROUGE-1F on a development subset of the 2003 data. We learn parameters θ on the DUC 2003 corpus, and test them using DUC 2004 data. We generate predictions from the trained DPP using the two inference algorithms described in the previous section, and compare their performance to a variety of baseline systems.

Our first and simplest baseline merely returns the first 665 bytes of the cluster text. Since the clusters consist of news articles, this is not an entirely unreasonable summary in many cases. We refer to this baseline as BEGIN.

We also compare against an alternative DPP-based model with identical similarity measure and quality features, but where the quality model has been trained using standard logistic regression. To learn this baseline, each sentence is treated as a unique instance to be classified as included or not included, with labels derived from our training oracle. Thus, it has the advantages of a DPP at test time, but does not take into account the diversity model while training; comparing with this baseline allows us to isolate the contribution of learning the model parameters in context. Note that MBR inference is impractical for this model because its training does not properly calibrate for overall summary length, so nearly all samples are either too long or too short. Thus, we report only the results obtained from greedy inference. We refer to this model as LR+DPP.

Next, we employ as baselines a range of previously proposed methods for multidocument summarization. Perhaps the simplest and most popular is Maximum Marginal Relevance (MMR), which uses a greedy selection process [23]. MMR relies on a similarity measure between sentences, for which we use the cosine distance measure S,

and a measure of relevance for each sentence, for which we use the same logistic regression-trained quality model as above. Sentences are chosen iteratively according to

$$\arg\max_{i \in \mathcal{Y}(X)} \left[\alpha q_i(X) - (1-\alpha) \max_{j \in Y} S_{ij} \right], \tag{4.19}$$

where Y is the set of sentences already selected (initially empty), $q_i(X)$ is the learned quality score, and S_{ij} is the cosine similarity between sentences i and j. The trade-off α is optimized on a development set, and sentences are added until the budget is full. We refer to this baseline as LR+MMR.

We also compare against the three highestscoring systems that actually competed in the DUC 2004 competition — peers 65, 104, and 35 — as well as the submodular graph-based approach recently described by Lin and Bilmes [94], which we refer to as SUBMOD1, and the improved submodular learning approach proposed by [95], which we denote SUBMOD2. We produced our own implementation of SUBMOD1, but rely on previously reported numbers for SUBMOD2, which include only ROUGE-1 scores.

Table 4.3 shows the results for all methods on the DUC 2004 test corpus. Scores for the actual DUC competitors differ slightly from the originally reported results because we use an updated version of the ROUGE package. Bold entries highlight the best performance in each column; in the case of MBR inference, which is stochastic, the improvements are significant at 99% confidence. The DPP models outperform the baselines in most cases; furthermore, there is a significant boost in performance due to the use of DPP maximum likelihood training in place of logistic regression. MBR inference performs best, assuming that we take sufficiently many samples; on the other hand, greedy inference runs more quickly than DPP-MBR100 and produces superior results. Relative to most other methods, the DPP model with MBR inference seems to more strongly emphasize recall. Note that MBR inference was performed with respect to ROUGE-1F, but could also be run to optimize other metrics if desired.

Feature contributions. In Table 4.4 we report the performance of DPP-GREEDY when different groups of features from Section 4.2.1 are

Table 4.3. ROUGE scores on the DUC 2004 test set.

System	ROUGE-1F	ROUGE-1P	ROUGE-1R	ROUGE-2F	ROUGE-SU4F
BEGIN	32.08	31.53	32.69	6.52	10.37
LR+MMR	37.58	37.15	38.05	9.05	13.06
LR+DPP	37.96	37.67	38.31	8.88	13.13
PEER 35	37.54	37.69	37.45	8.37	12.90
PEER 104	37.12	36.79	37.48	8.49	12.81
PEER 65	37.87	37.58	38.20	9.13	13.19
SUBMOD1	38.73	38.40	39.11	8.86	13.11
SUBMOD2	39.78	39.16	40.43	—	—
DPP-GREEDY	38.96	38.82	39.15	**9.86**	13.83
DPP-MBR100	38.83	38.06	39.67	8.85	13.38
DPP-MBR1000	39.79	38.96	40.69	9.29	13.87
DPP-MBR5000	**40.33**	**39.43**	**41.31**	9.54	**14.13**

Table 4.4. ROUGE scores for DPP-GREEDY with features removed.

Features	ROUGE-1F	ROUGE-1P	ROUGE-1R
All	38.96	38.82	39.15
All but length	37.38	37.08	37.72
All but position	36.34	35.99	36.72
All but similarity	38.14	37.97	38.35
All but LexRank	38.10	37.92	38.34
All but pronouns	38.80	38.67	38.98
All but similarity, LexRank	36.06	35.84	36.32

removed, in order to estimate their relative contributions. Length and position appear to be quite important; however, although individually similarity and LexRank scores have only a modest impact on performance, when both are omitted the drop is significant. This suggests, intuitively, that these two groups convey similar information — both are essentially measures of centrality — but that this information is important to achieving strong performance.

5

k-DPPs

A determinantal point process assigns a probability to every subset of the ground set $\mathcal{Y}$. This means that, with some probability, a sample from the process will be empty; with some probability, it will be all of $\mathcal{Y}$. In many cases this is not desirable. For instance, we might want to use a DPP to model the positions of basketball players on a court, under the assumption that a team tends to spread out for better coverage. In this setting, we know that with very high probability each team will have exactly five players on the court. Thus, if our model gives some probability of seeing zero or fifty players, it is not likely to be a good fit.

We showed in Section 2.4.4 that there exist elementary DPPs having fixed cardinality k; however, this is achieved only by focusing exclusively (and equally) on k-specific "aspects" of the data, as represented by eigenvectors of the kernel. Thus, for DPPs, the notions of *size* and *content* are fundamentally intertwined. We cannot change one without affecting the other. This is a serious limitation on the types of distributions that can be expressed; for instance, a DPP cannot even capture the uniform distribution over sets of cardinality k.

More generally, even for applications where the number of items is unknown, the size model imposed by a DPP may not be a good

fit. We have seen that the cardinality of a DPP sample has a simple distribution: it is the number of successes in a series of Bernoulli trials. But while this distribution characterizes certain types of data, other cases might look very different. For example, picnickers may tend to stake out diverse positions in a park, but on warm weekend days there might be hundreds of people, and on a rainy Tuesday night there are likely to be none. This bimodal distribution is quite unlike the sum of Bernoulli variables imposed by DPPs.

Perhaps most importantly, in some cases we do not even want to model cardinality at all, but instead offer it as a parameter. For example, a search engine might need to deliver ten diverse results to its desktop users, but only five to its mobile users. This ability to control the size of a DPP "on the fly" can be crucial in real-world applications.

In this section we introduce k-DPPs, which address the issues described above by conditioning a DPP on the cardinality of the random set $\boldsymbol{Y}$. This simple change effectively divorces the DPP content model, with its intuitive diversifying properties, from the DPP size model, which is not always appropriate. We can then use the DPP content model with a size model of our choosing, or simply set the desired size based on context. The result is a significantly more expressive modeling approach (which can even have limited positive correlations) and increased control.

We begin by defining k-DPPs. The conditionalization they require, though simple in theory, necessitates new algorithms for inference problems like normalization and sampling. Naively, these tasks require exponential time, but we show that through recursions for computing elementary symmetric polynomials we can solve them exactly in polynomial time. Finally, we demonstrate the use of k-DPPs on an image search problem, where the goal is to show users diverse sets of images that correspond to their query.

5.1 Definition

A k-DPP on a discrete set $\mathcal{Y} = \{1, 2, \ldots, N\}$ is a distribution over all subsets $Y \subseteq \mathcal{Y}$ with cardinality k [84]. In contrast to the standard DPP,

which models both the size and content of a random subset $\boldsymbol{Y}$, a k-DPP is concerned only with the content of a random k-set. Thus, a k-DPP is obtained by conditioning a standard DPP on the event that the set $\boldsymbol{Y}$ has cardinality k. Formally, the k-DPP $\mathcal{P}_L^k$ gives probabilities

$$P_L^k(Y) = \frac{\det(L_Y)}{\sum_{|Y'|=k} \det(L_{Y'})}, \tag{5.1}$$

where $|Y| = k$ and L is a positive semidefinite kernel. Compared to the standard DPP, the only changes are the restriction on Y and the normalization constant. While in a DPP every k-set Y competes with all other subsets of $\mathcal{Y}$, in a k-DPP it competes only with sets of the same cardinality. This subtle change has significant implications.

For instance, consider the seemingly simple distribution that is uniform over all sets $Y \subseteq \mathcal{Y}$ with cardinality k. If we attempt to build a DPP capturing this distribution we quickly run into difficulties. In particular, the marginal probability of any single item is $\frac{k}{N}$, so the marginal kernel K, if it exists, must have $\frac{k}{N}$ on the diagonal. Likewise, the marginal probability of any pair of items is $\frac{k(k-1)}{N(N-1)}$, and so by symmetry the off-diagonal entries of K must be equal to a constant. As a result, any valid marginal kernel has to be the sum of a constant matrix and a multiple of the identity matrix. Since a constant matrix has at most one nonzero eigenvalue and the identity matrix is full rank, it is easy to show that, except in the special cases $k = 0, 1, N-1$, the resulting kernel has full rank. But we know that a full-rank kernel implies that the probability of seeing all N items together is nonzero. Thus the desired process cannot be a DPP unless $k = 0, 1, N-1$, or N. On the other hand, a k-DPP with the identity matrix as its kernel gives the distribution we are looking for. This improved expressive power can be quite valuable in practice.

5.1.1 Alternative Models of Size

Since a k-DPP is conditioned on cardinality, k must come from somewhere outside of the model. In many cases, k may be fixed according to application needs, or perhaps changed on the fly by users or depending on context. This flexibility and control is one of the major practical

advantages of k-DPPs. Alternatively, in situations where we wish to model size as well as content, a k-DPP can be combined with a size model $\mathcal{P}_{\text{size}}$ that assigns a probability to every possible $k \in \{1, 2, \ldots, N\}$:

$$\mathcal{P}(Y) = \mathcal{P}_{\text{size}}(|Y|)\mathcal{P}_L^{|Y|}(Y). \tag{5.2}$$

Since the k-DPP is a proper conditional model, the distribution $\mathcal{P}$ is well defined. By choosing $\mathcal{P}_{\text{size}}$ appropriate to the task at hand, we can effectively take advantage of the diversifying properties of DPPs in situations where the DPP size model is a poor fit.

As a side effect, this approach actually enables us to use k-DPPs to build models with both negative and positive correlations. For instance, if $\mathcal{P}_{\text{size}}$ indicates that there are likely to be either hundreds of picnickers in the park (on a nice day) or, otherwise, just a few, then knowing that there are 50 picnickers today implies that there are likely to be even more. Thus, k-DPPs can yield more expressive models than DPPs in this sense as well.

5.2 Inference

Of course, increasing the expressive power of the DPP causes us to wonder whether, in doing so, we might have lost some of the convenient computational properties that made DPPs useful in the first place. Naively, this seems to be the case; for instance, while the normalizing constant for a DPP can be written in closed form, the sum in Equation (5.1) is exponential and seems hard to simplify. In this section, we will show how k-DPP inference can in fact be performed efficiently, using recursions for computing the elementary symmetric polynomials.

5.2.1 Normalization

Recall that the k-th elementary symmetric polynomial on $\lambda_1, \lambda_2 \ldots, \lambda_N$ is given by

$$e_k(\lambda_1, \lambda_2, \ldots, \lambda_N) = \sum_{\substack{J \subseteq \{1,2,\ldots,N\} \\ |J|=k}} \prod_{n \in J} \lambda_n. \tag{5.3}$$

For instance,

$$e_1(\lambda_1, \lambda_2, \lambda_3) = \lambda_1 + \lambda_2 + \lambda_3 \tag{5.4}$$

$$e_2(\lambda_1, \lambda_2, \lambda_3) = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 \tag{5.5}$$

$$e_3(\lambda_1, \lambda_2, \lambda_3) = \lambda_1\lambda_2\lambda_3. \tag{5.6}$$

Proposition 5.1. The normalization constant for a k-DPP is

$$Z_k = \sum_{|Y'|=k} \det(L_{Y'}) = e_k(\lambda_1, \lambda_2, \ldots, \lambda_N), \tag{5.7}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_N$ are the eigenvalues of L.

Proof. One way to see this is to examine the characteristic polynomial of L, $\det(L - \lambda I)$ [54]. We can also show it directly using properties of DPPs. Recalling that

$$\sum_{Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I), \tag{5.8}$$

we have

$$\sum_{|Y'|=k} \det(L_{Y'}) = \det(L + I) \sum_{|Y'|=k} \mathcal{P}_L(Y'), \tag{5.9}$$

where $\mathcal{P}_L$ is the DPP with kernel L. Applying Lemma 2.5, which expresses any DPP as a mixture of elementary DPPs, we have

$$\det(L + I) \sum_{|Y'|=k} \mathcal{P}_L(Y') = \sum_{|Y'|=k} \sum_{J \subseteq \{1,2,\ldots,N\}} \mathcal{P}^{V_J}(Y') \prod_{n \in J} \lambda_n \tag{5.10}$$

$$= \sum_{|J|=k} \sum_{|Y'|=k} \mathcal{P}^{V_J}(Y') \prod_{n \in J} \lambda_n \tag{5.11}$$

$$= \sum_{|J|=k} \prod_{n \in J} \lambda_n, \tag{5.12}$$

where we use Lemma 2.6 in the last two steps to conclude that $\mathcal{P}^{V_J}(Y') = 0$ unless $|J| = |Y'|$. (Recall that V_J is the set of eigenvectors of L associated with λ_n for $n \in J$.) □

Algorithm 7 Computing the elementary symmetric polynomials

Input: k, eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_N$
$e_0^n \leftarrow 1 \quad \forall\, n \in \{0, 1, 2, \ldots, N\}$
$e_l^0 \leftarrow 0 \quad \forall\, l \in \{1, 2, \ldots, k\}$
for $l = 1, 2, \ldots k$ **do**
  for $n = 1, 2, \ldots, N$ **do**
    $e_l^n \leftarrow e_l^{n-1} + \lambda_n e_{l-1}^{n-1}$
  end for
end for
Output: $e_k(\lambda_1, \lambda_2, \ldots, \lambda_N) = e_k^N$

To compute the k-th elementary symmetric polynomial, we can use the recursive algorithm given in Algorithm 7, which is based on the observation that every set of k eigenvalues either omits λ_N, in which case we must choose k of the remaining eigenvalues, or includes λ_N, in which case we get a factor of λ_N and choose only $k-1$ of the remaining eigenvalues. Formally, letting e_k^N be a shorthand for $e_k(\lambda_1, \lambda_2, \ldots, \lambda_N)$, we have

$$e_k^N = e_k^{N-1} + \lambda_N e_{k-1}^{N-1}. \tag{5.13}$$

Note that a variety of recursions for computing elementary symmetric polynomials exist, including Newton's identities, the Difference Algorithm, and the Summation Algorithm [4]. Algorithm 7 is essentially the Summation Algorithm, which is both asymptotically faster and numerically more stable than the other two, since it uses only sums and does not rely on precise cancellation of large numbers.

Algorithm 7 runs in time $O(Nk)$. Strictly speaking, the inner loop need only iterate up to $N - k + l$ in order to obtain e_k^N at the end; however, by going up to N we compute all of the preceding elementary symmetric polynomials e_l^N along the way. Thus, by running Algorithm 7 with $k = N$ we can compute the normalizers for k-DPPs of every size in time $O(N^2)$. This can be useful when k is not known in advance.

5.2.2 Sampling

Since a k-DPP is just a DPP conditioned on size, we could sample a k-DPP by repeatedly sampling the corresponding DPP and rejecting

the samples until we obtain one of size k. To make this more efficient, recall from Section 2.4.4 that the standard DPP sampling algorithm proceeds in two phases. First, a subset V of the eigenvectors of L is selected at random, and then a set of cardinality $|V|$ is sampled based on those eigenvectors. Since the size of a sample is fixed in the first phase, we could reject the samples before the second phase even begins, waiting until we have $|V| = k$. However, rejection sampling is likely to be slow. It would be better to directly sample a set V conditioned on the fact that its cardinality is k. In this section we show how sampling k eigenvectors can be done efficiently, yielding a sampling algorithm for k-DPPs that is asymptotically as fast as sampling standard DPPs.

We can formalize the intuition above by rewriting the k-DPP distribution in terms of the corresponding DPP:

$$\mathcal{P}_L^k(Y) = \frac{1}{e_k^N} \det(L + I)\mathcal{P}_L(Y) \tag{5.14}$$

whenever $|Y| = k$, where we replace the DPP normalization constant with the k-DPP normalization constant using Proposition 5.1. Applying Lemma 2.5 and Lemma 2.6 to decompose the DPP into elementary parts yields

$$\mathcal{P}_L^k(Y) = \frac{1}{e_k^N} \sum_{|J|=k} \mathcal{P}^{V_J}(Y) \prod_{n \in J} \lambda_n. \tag{5.15}$$

Therefore, a k-DPP is also a mixture of elementary DPPs, but it only gives nonzero weight to those of dimension k. Since the second phase of DPP sampling provides a means for sampling from any given elementary DPP, we can sample from a k-DPP if we can sample index sets J according to the corresponding mixture components. Like normalization, this is naively an exponential task, but we can do it efficiently using the recursive properties of elementary symmetric polynomials.

Theorem 5.2. Let $\boldsymbol{J}$ be the desired random variable, so that $\Pr(\boldsymbol{J} = J) = \frac{1}{e_k^N} \prod_{n \in J} \lambda_n$ when $|J| = k$, and zero otherwise. Then Algorithm 8 yields a sample for $\boldsymbol{J}$.

Algorithm 8 Sampling k eigenvectors

Input: k, eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$
compute e_l^n for $l = 0, 1, \ldots, k$ and $n = 0, 1, \ldots, N$ (Algorithm 7)
$J \leftarrow \emptyset$
$l \leftarrow k$
for $n = N, \ldots, 2, 1$ **do**
  if $l = 0$ **then**
    break
  end if
  if $u \sim U[0,1] < \lambda_n \frac{e_{l-1}^{n-1}}{e_l^n}$ **then**
    $J \leftarrow J \cup \{n\}$
    $l \leftarrow l - 1$
  end if
end for
Output: J

Proof. If $k = 0$, then Algorithm 8 returns immediately at the first iteration of the loop with $J = \emptyset$, which is the only possible value of $\boldsymbol{J}$.

If $N = 1$ and $k = 1$, then $\boldsymbol{J}$ must contain the single index 1. We have $e_1^1 = \lambda_1$ and $e_0^0 = 1$, thus $\lambda_1 \frac{e_0^0}{e_1^1} = 1$, and Algorithm 8 returns $J = \{1\}$ with probability 1.

We proceed by induction and compute the probability that Algorithm 8 returns J for $N > 1$ and $1 \leq k \leq N$. By inductive hypothesis, if an iteration of the loop in Algorithm 8 begins with $n < N$ and $0 \leq l \leq n$, then the remainder of the algorithm adds to J a set of elements J' with probability

$$\frac{1}{e_l^n} \prod_{n' \in J'} \lambda_{n'} \tag{5.16}$$

if $|J'| = l$, and zero otherwise.

Now suppose that J contains N, $J = J' \cup \{N\}$. Then N must be added to J in the first iteration of the loop, which occurs with probability $\lambda_N \frac{e_{k-1}^{N-1}}{e_k^N}$. The second iteration then begins with $n = N - 1$ and $l = k - 1$. If l is zero, we have the immediate base case; otherwise we

have $1 \leq l \leq n$. By the inductive hypothesis, the remainder of the algorithm selects J' with probability

$$\frac{1}{e_{k-1}^{N-1}} \prod_{n \in J'} \lambda_n \tag{5.17}$$

if $|J'| = k - 1$, and zero otherwise. Thus Algorithm 8 returns J with probability

$$\left(\lambda_N \frac{e_{k-1}^{N-1}}{e_k^N}\right) \frac{1}{e_{k-1}^{N-1}} \prod_{n \in J'} \lambda_n = \frac{1}{e_k^N} \prod_{n \in J} \lambda_n \tag{5.18}$$

if $|J| = k$, and zero otherwise.

On the other hand, if J does not contain N, then the first iteration must add nothing to J; this happens with probability

$$1 - \lambda_N \frac{e_{k-1}^{N-1}}{e_k^N} = \frac{e_k^{N-1}}{e_k^N}, \tag{5.19}$$

where we use the fact that $e_k^N - \lambda_N e_{k-1}^{N-1} = e_k^{N-1}$. The second iteration then begins with $n = N - 1$ and $l = k$. We observe that if $N - 1 < k$, then Equation (5.19) is equal to zero, since $e_l^n = 0$ whenever $l > n$. Thus almost surely the second iteration begins with $k \leq n$, and we can apply the inductive hypothesis. This guarantees that the remainder of the algorithm chooses J with probability

$$\frac{1}{e_k^{N-1}} \prod_{n \in J} \lambda_n \tag{5.20}$$

whenever $|J| = k$. The overall probability that Algorithm 8 returns J is therefore

$$\left(\frac{e_k^{N-1}}{e_k^N}\right) \frac{1}{e_k^{N-1}} \prod_{n \in J} \lambda_n = \frac{1}{e_k^N} \prod_{n \in J} \lambda_n \tag{5.21}$$

if $|J| = k$, and zero otherwise. □

Algorithm 8 precomputes the values of $e_1^1, \ldots, e_k^N$, which requires $O(Nk)$ time using Algorithm 7. The loop then iterates at most N times and requires only a constant number of operations, so Algorithm 8

runs in $O(Nk)$ time overall. By Equation (5.15), selecting J with Algorithm 8 and then sampling from the elementary DPP $\mathcal{P}^{V_J}$ generates a sample from the k-DPP. As shown in Section 2.4.4, sampling an elementary DPP can be done in $O(Nk^3)$ time (see the second loop of Algorithm 1), so sampling k-DPPs is $O(Nk^3)$ overall, assuming that we have an eigendecomposition of the kernel in advance. This is no more expensive than sampling a standard DPP.

5.2.3 Marginalization

Since k-DPPs are not DPPs, they do not in general have marginal kernels. However, we can still use their connection to DPPs to compute the marginal probability of a set A, $|A| \leq k$:

$$\mathcal{P}_L^k(A \subseteq \boldsymbol{Y}) = \sum_{\substack{|Y'|=k-|A| \\ A \cap Y' = \emptyset}} \mathcal{P}_L^k(Y' \cup A) \tag{5.22}$$

$$= \frac{\det(L+I)}{Z_k} \sum_{\substack{|Y'|=k-|A| \\ A \cap Y' = \emptyset}} \mathcal{P}_L(Y' \cup A) \tag{5.23}$$

$$= \frac{\det(L+I)}{Z_k} \sum_{\substack{|Y'|=k-|A| \\ A \cap Y' = \emptyset}} \mathcal{P}_L(\boldsymbol{Y} = Y' \cup A | A \subseteq \boldsymbol{Y}) \mathcal{P}_L(A \subseteq \boldsymbol{Y}) \tag{5.24}$$

$$= \frac{Z^A_{k-|A|}}{Z_k} \frac{\det(L+I)}{\det(L^A+I)} \mathcal{P}_L(A \subseteq \boldsymbol{Y}), \tag{5.25}$$

where L^A is the kernel, given in Equation (2.42), of the DPP conditioned on the inclusion of A, and

$$Z^A_{k-|A|} = \det(L^A+I) \sum_{\substack{|Y'|=k-|A| \\ A \cap Y' = \emptyset}} \mathcal{P}_L(\boldsymbol{Y} = Y' \cup A | A \subseteq \boldsymbol{Y}) \tag{5.26}$$

$$= \sum_{\substack{|Y'|=k-|A| \\ A \cap Y' = \emptyset}} \det(L^A_{Y'}) \tag{5.27}$$

is the normalization constant for the $(k-|A|)$-DPP with kernel L^A. That is, the marginal probabilities for a k-DPP are just the marginal probabilities for a DPP with the same kernel, but with an appropriate

change of normalizing constants. We can simplify Equation (5.25) by observing that

$$\frac{\det(L_A)}{\det(L+I)} = \frac{\mathcal{P}_L(A \subseteq \boldsymbol{Y})}{\det(L^A+I)}, \tag{5.28}$$

since the left-hand side is the probability (under the DPP with kernel L) that A occurs by itself, and the right-hand side is the marginal probability of A multiplied by the probability of observing nothing else conditioned on observing A: $1/\det(L^A+I)$. Thus we have

$$\mathcal{P}^k_L(A \subseteq \boldsymbol{Y}) = \frac{Z^A_{k-|A|}}{Z_k}\det(L_A) = Z^A_{k-|A|}\mathcal{P}^k_L(A). \tag{5.29}$$

That is, the marginal probability of A is the probability of observing exactly A times the normalization constant when conditioning on A. Note that a version of this formula also holds for standard DPPs, but there it can be rewritten in terms of the marginal kernel.

Singleton marginals Equations (5.25) and (5.29) are general but require computing large determinants and elementary symmetric polynomials, regardless of the size of A. Moreover, those quantities (for example, $\det(L^A+I)$) must be recomputed for each unique A whose marginal probability is desired. Thus, finding the marginal probabilities of many small sets is expensive compared to a standard DPP, where we need only small minors of K. However, we can derive a more efficient approach in the special but useful case where we want to know all of the singleton marginals for a k-DPP — for instance, in order to implement quality learning as described in Section 4.2.

We start by using Equation (5.15) to write the marginal probability of an item i in terms of a combination of elementary DPPs:

$$\mathcal{P}^k_L(i \in \boldsymbol{Y}) = \frac{1}{e^N_k}\sum_{|J|=k}\mathcal{P}^{V_J}(i \in \boldsymbol{Y})\prod_{n' \in J}\lambda_{n'}. \tag{5.30}$$

Because the marginal kernel of the elementary DPP $\mathcal{P}^{V_J}$ is given by $\sum_{n \in J}\boldsymbol{v}_n\boldsymbol{v}_n^\top$, we have

$$\mathcal{P}^k_L(i \in \boldsymbol{Y}) = \frac{1}{e^N_k}\sum_{|J|=k}\left(\sum_{n \in J}(\boldsymbol{v}_n^\top \boldsymbol{e}_i)^2\right)\prod_{n' \in J}\lambda_{n'} \tag{5.31}$$

$$= \frac{1}{e_k^N} \sum_{n=1}^{N} (\boldsymbol{v}_n^\top \boldsymbol{e}_i)^2 \sum_{J \supseteq \{n\}, |J|=k} \prod_{n' \in J} \lambda_{n'} \tag{5.32}$$

$$= \sum_{n=1}^{N} (\boldsymbol{v}_n^\top \boldsymbol{e}_i)^2 \lambda_n \frac{e_{k-1}^{-n}}{e_k^N}, \tag{5.33}$$

where $e_{k-1}^{-n} = e_{k-1}(\lambda_1, \lambda_2, \ldots, \lambda_{n-1}, \lambda_{n+1}, \ldots, \lambda_N)$ denotes the $(k-1)$-order elementary symmetric polynomial for all eigenvalues of L except λ_n. Note that $\lambda_n e_{k-1}^{-n} / e_k^N$ is exactly the marginal probability that $n \in J$ when J is chosen using Algorithm 8; in other words, the marginal probability of item i is the sum of the contributions $(\boldsymbol{v}_n^\top \boldsymbol{e}_i)^2$ made by each eigenvector scaled by the respective probabilities that the eigenvectors are selected. The contributions are easily computed from the eigendecomposition of L, thus we need only e_k^N and e_{k-1}^{-n} for each value of n in order to calculate the marginals for all items in $O(N^2)$ time, or $O(ND)$ time if the rank of L is $D < N$.

Algorithm 7 computes $e_{k-1}^{N-1} = e_{k-1}^{-N}$ in the process of obtaining e_k^N, so naively we could run Algorithm 7 N times, repeatedly reordering the eigenvectors so that each takes a turn at the last position. To compute all of the required polynomials in this fashion would require $O(N^2 k)$ time. However, we can improve this (for small k) to $O(N \log(N) k^2)$; to do so we will make use of a binary tree on N leaves. Each node of the tree corresponds to a set of eigenvalues of L; the leaves represent single eigenvalues, and an interior node of the tree represents the set of eigenvalues corresponding to its descendant leaves. (See Figure 5.1.) We will associate with each node the set of elementary symmetric polynomials $e_1(\Lambda), e_2(\Lambda), \ldots, e_k(\Lambda)$, where Λ is the set of eigenvalues represented by the node.

These polynomials can be computed directly for leaf nodes in constant time, and the polynomials of an interior node can be computed given those of its children using a simple recursion:

$$e_k(\Lambda_1 \cup \Lambda_2) = \sum_{l=0}^{k} e_l(\Lambda_1) e_{k-l}(\Lambda_2). \tag{5.34}$$

Thus, we can compute the polynomials for the entire tree in $O(N \log(N) k^2)$ time; this is sufficient to obtain e_k^N at the root node.

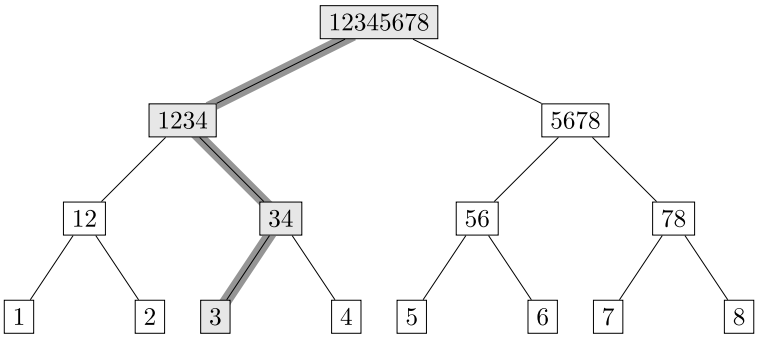


Fig. 5.1 Binary tree with $N = 8$ leaves; interior nodes represent their descendant leaves. Removing a path from leaf n to the root leaves $\log N$ subtrees that can be combined to compute e_{k-1}^{-n}.

However, if we now remove a leaf node corresponding to eigenvalue n, we invalidate the polynomials along the path from the leaf to the root; see Figure 5.1. This leaves $\log N$ disjoint subtrees which together represent all of the eigenvalues of L, leaving out λ_n. We can now apply Equation (5.34) $\log N$ times to the roots of these trees in order to obtain e_{k-1}^{-n} in $O(\log(N)k^2)$ time. If we do this for each value of n, the total additional time required is $O(N\log(N)k^2)$.

The algorithm described above thus takes $O(N\log(N)k^2)$ time to produce the necessary elementary symmetric polynomials, which in turn allow us to compute all of the singleton marginals. This is a dramatic improvement over applying Equation (5.25) to each item separately.

5.2.4 Conditioning

Suppose we want to condition a k-DPP on the inclusion of a particular set A. For $|A| + |B| = k$ we have

$$\begin{aligned}
\mathcal{P}_L^k(\boldsymbol{Y} = A \cup B | A \subseteq \boldsymbol{Y}) &\propto \mathcal{P}_L^k(\boldsymbol{Y} = A \cup B) && (5.35)\\
&\propto \mathcal{P}_L(\boldsymbol{Y} = A \cup B) && (5.36)\\
&\propto \mathcal{P}_L(\boldsymbol{Y} = A \cup B | A \subseteq \boldsymbol{Y}) && (5.37)\\
&\propto \det(L_B^A). && (5.38)
\end{aligned}$$

Thus the conditional k-DPP is a $k - |A|$-DPP whose kernel is the same as that of the associated conditional DPP. The normalization constant is $Z^A_{k-|A|}$. We can condition on excluding A in the same manner.

5.2.5 Finding the Mode

Unfortunately, although k-DPPs offer the efficient versions of DPP inference algorithms presented above, finding the most likely set Y remains intractable. It is easy to see that the reduction from Section 2.4.5 still applies, since the cardinality of the Y corresponding to an exact 3-cover, if it exists, is known. In practice we can utilize greedy approximations, like we did for standard DPPs in Section 4.2.1.

5.3 Experiments: Image Search

We demonstrate the use of k-DPPs on an image search task [84]. The motivation is as follows. Suppose that we run an image search engine, where our primary goal is to deliver the most relevant possible images to our users. Unfortunately, the query strings those users provide are often ambiguous. For instance, a user searching for "philadelphia" might be looking for pictures of the city skyline, street-level shots of buildings, or perhaps iconic sights like the Liberty Bell or the Love sculpture. Furthermore, even if we know the user is looking for a skyline photograph, he or she might specifically want a daytime or nighttime shot, a particular angle, and so on. In general, we cannot expect users to provide enough information in a textual query to identify the best image with any certainty.

For this reason search engines typically provide a small array of results, and we argue that, to maximize the probability of the user being happy with at least one image, the results should be relevant to the query but also diverse with respect to one another. That is, if we want to maximize the proportion of users searching "philadelphia" who are satisfied by our response, each image we return should satisfy a large but distinct subset of those users, thus maximizing our overall coverage. Since we want diverse results but also require control over the number of results we provide, a k-DPP is a natural fit.

5.3.1 Learning Setup

Of course, we do not actually run a search engine and do not have real users. Thus, in order to be able to evaluate our model using real human feedback, we define the task in a manner that allows us to obtain inexpensive human supervision via Amazon Mechanical Turk. We do this by establishing a simple binary decision problem, where the goal is to choose, given two possible sets of image search results, the set that is more diverse. Formally, our labeled training data comprises comparative pairs of image sets $\{(Y_t^+, Y_t^-)\}_{t=1}^T$, where set Y_t^+ is preferred over set Y_t^-, $|Y_t^+| = |Y_t^-| = k$. We can measure performance on this classification task using the zero–one loss, which is zero whenever we choose the correct set from a given pair, and one otherwise.

For this task we employ a simple method for learning a combination of k-DPPs that is convex and seems to work well in practice. Given a set $L_1, L_2, \ldots, L_D$ of "expert" kernel matrices, which are fixed in advance, define the combination model

$$\mathcal{P}_\theta^k = \sum_{l=1}^{D} \theta_l \mathcal{P}_{L_l}^k, \tag{5.39}$$

where $\sum_{l=1}^D \theta_l = 1$. Note that this is a combination of distributions, rather than a combination of kernels. We will learn θ to optimize a logistic loss measure on the binary task:

$$\begin{aligned} \min_\theta \quad & \mathcal{L}(\theta) = \sum_{t=1}^{T} \log(1 + e^{-\gamma[\mathcal{P}_\theta^k(Y_t^+) - \mathcal{P}_\theta^k(Y_t^-)]}) \\ \text{s.t.} \quad & \sum_{l=1}^{D} \theta_l = 1, \end{aligned} \tag{5.40}$$

where γ is a hyperparameter that controls how aggressively we penalize mistakes. Intuitively, the idea is to find a combination of k-DPPs where the positive sets Y_t^+ receive higher probability than the corresponding negative sets Y_t^-. By using the logistic loss (Figure 5.2), which acts like a smooth hinge loss, we focus on making fewer mistakes.

Because Equation (5.40) is convex in θ (it is the composition of the convex logistic loss function with a linear function of θ), we can

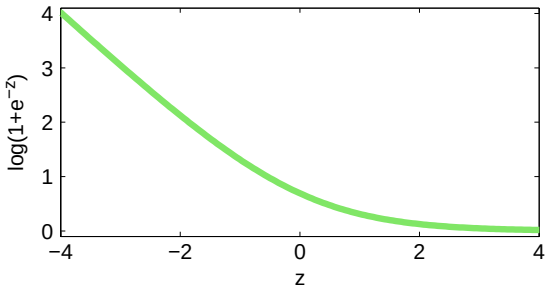


Fig. 5.2 The logistic loss function.

optimize it efficiently using projected gradient descent, where we alternate taking gradient steps and projecting on the constraint $\sum_{l=1}^{D} \theta_l = 1$. The gradient is given by

$$\nabla \mathcal{L} = \sum_{t=1}^{T} \frac{e^{\theta^\top \delta^t}}{1 + e^{\theta^\top \delta^t}} \delta^t, \tag{5.41}$$

where δ^t is a vector with entries

$$\delta_l^t = -\gamma [\mathcal{P}_{L_l}^k(Y_t^+) - \mathcal{P}_{L_l}^k(Y_t^-)]. \tag{5.42}$$

Projection onto the simplex is achieved using standard algorithms [7].

5.3.2 Data

We create datasets for three broad image search categories, using 8–12 hand-selected queries for each category. (See Table 5.1.) For each query, we retrieve the top 64 results from Google Image Search, restricting the search to JPEG files that pass the strictest level of Safe Search filtering. Of those 64 results, we eliminate any that are no longer available for download. On average this leaves us with 63.0 images per query, with a range of 59–64.

We then use the downloaded images to generate 960 training instances for each category, spread evenly across the different queries. In order to compare k-DPPs directly against baseline heuristic methods that do not model probabilities of full sets, we generate only instances where Y_t^+ and Y_t^- differ by a single element. That is, the classification

Table 5.1. Queries used for data collection.

Cars	Cities	Dogs
Chrysler	Baltimore	Beagle
Ford	Barcelona	Bernese
Honda	London	Blue Heeler
Mercedes	Los Angeles	Cocker Spaniel
Mitsubishi	Miami	Collie
Nissan	New York City	Great Dane
Porsche	Paris	Labrador
Toyota	Philadelphia	Pomeranian
	San Francisco	Poodle
	Shanghai	Pug
	Tokyo	Schnauzer
	Toronto	Shih Tzu

problem is effectively to choose which of two candidate images i_t^+, i_t^i is a less redundant addition to a given partial result set Y_t:

$$Y_t^+ = Y_t \cup \{i_t^+\} \quad Y_t^- = Y_t \cup \{i_t^-\}. \tag{5.43}$$

In our experiments Y_t contains five images, so $k = |Y_t^+| = |Y_t^-| = 6$. We sample partial result sets using a k-DPP with a SIFT-based kernel (details below) to encourage diversity. The candidates are then selected uniformly at random from the remaining images, except for 10% of instances that are reserved for measuring the performance of our human judges. For those instances, one of the candidates is a duplicate image chosen uniformly at random from the partial result set, making it the obviously more redundant choice. The other candidate is chosen as usual.

In order to decide which candidate actually results in the more diverse set, we collect human diversity judgments using Amazon's Mechanical Turk. Annotators are drawn from the general pool of Turk workers, and are able to label as many instances as they wish. Annotators are paid $0.01 USD for each instance that they label. For practical reasons, we present the images to the annotators at reduced scale; the larger dimension of an image is always 250 pixels. The annotators are instructed to choose the candidate that they feel is "less similar" to the images in the partial result set. We do not offer any specific guidance on how to judge similarity, since dealing with uncertainty in human users

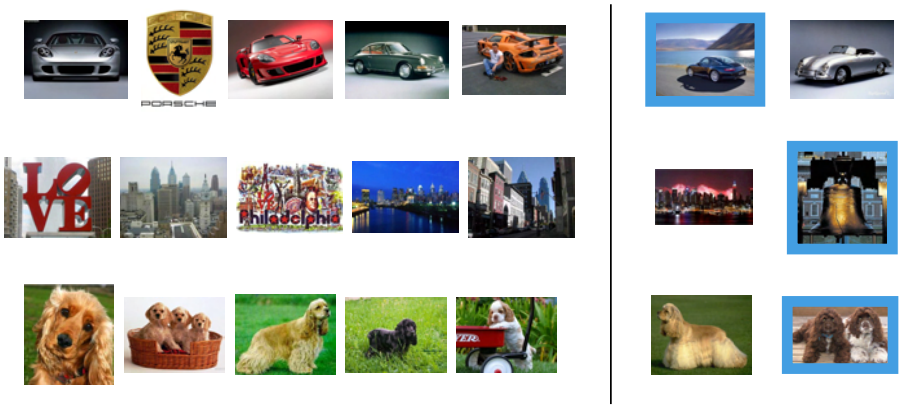


Fig. 5.3 Sample labeling instances from each search category. The five images on the left form the partial result set and the two candidates are shown on the right. The candidate receiving the majority of annotator votes has a blue border.

is central to the task. The candidate images are presented in random order. Figure 5.3 shows a sample instance from each category.

Overall, we find that workers choose the correct image for 80.8% of the calibration instances (that is, they choose the one not belonging to the partial result set). This suggests only moderate levels of noise due to misunderstanding, inattention or robot workers. However, for non-calibration instances the task is inherently difficult and subjective. To keep noise in check, we have each instance labeled by five independent judges, and keep only those instances where four or more judges agree. In the end this leaves us with 408–482 labeled instances per category, or about half of the original instances.

5.3.3 Kernels

We define a set of 55 "expert" similarity kernels for the collected images, which form the building blocks of our combination model and baseline methods. Each kernel $L^{\boldsymbol{f}}$ is the Gram matrix of some feature function $\boldsymbol{f}$; that is, $L^{\boldsymbol{f}}_{ij} = \boldsymbol{f}(i) \cdot \boldsymbol{f}(j)$ for images i and j. We therefore specify the kernels through the feature functions used to generate them. All of our feature functions are normalized so that $\|\boldsymbol{f}(i)\|^2 = 1$ for all i; this ensures that no image is *a priori* more likely than any other. Implicitly,

thinking in terms of the decomposition in Section 3.1, we are assuming that all of the images in our set are equally *relevant* in order to isolate the modeling of diversity. This assumption is at least partly justified by the fact that our images come from actual Google searches, and are thus presumably relevant to the query.

We use the following feature functions, which derive from standard image processing and feature extraction methods:

- **Color** (2 variants): Each pixel is assigned a coordinate in three-dimensional Lab color space. The colors are then sorted into axis-aligned bins, producing a histogram of either 8 or 64 dimensions.
- **SIFT** (2 variants): The images are processed with the `vlfeat` toolbox to obtain sets of 128-dimensional SIFT descriptors [96, 150]. The descriptors for a given category are combined, subsampled to a set of 25,000, and then clustered using k-means into either 256 or 512 clusters. The feature vector for an image is the normalized histogram of the nearest clusters to the descriptors in the image.
- **GIST**: The images are processed using code from Oliva and Torralba [118] to yield 960-dimensional GIST feature vectors characterizing properties like "openness," "roughness," "naturalness," and so on.

In addition to the five feature functions described above, we include another five that are identical but focus only on the center of the image, defined as the centered rectangle with dimensions half those of the original image. This gives our first ten kernels. We then create 45 pairwise combination kernels by concatenating every possible pair of the ten basic feature vectors. This technique produces kernels that synthesize more than one source of information, offering greater flexibility.

Finally, we augment our kernels by adding a constant hyperparameter ρ to each entry. ρ acts a knob for controlling the overall preference for diversity; as ρ increases, all images appear more similar, thus increasing repulsion. In our experiments, ρ is chosen independently for each method and each category to optimize performance on the training set.

5.3.4 Methods

We test four different methods. Two use k-DPPs and two are derived from Maximum Marginal Relevance (MMR) [23]. For each approach, we test both the single best expert kernel on the training data and a learned combination of kernels. All methods were tuned separately for each of the three query categories. On each run a random 25% of the labeled examples are reserved for testing, and the remaining 75% form the training set used for setting hyperparameters and training. Recall that Y_t is the five-image partial result set for instance t, and let $C_t = \{i_t^+, i_t^-\}$ denote the set of two candidates images, where i_t^+ is the candidate preferred by the human judges.

Best k-DPP Given a single kernel L, the k-DPP prediction is

$$k\text{DPP}_t = \arg\max_{i \in C_t} \mathcal{P}_L^6(Y_t \cup \{i\}). \tag{5.44}$$

We select the kernel with the best zero–one accuracy on the training set, and apply it to the test set.

Mixture of k-DPPs We apply our learning method to the full set of 55 kernels, optimizing Equation (5.40) on the training set to obtain a 55-dimensional mixture vector θ. We set γ to minimize the zero–one training loss. We then take the learned θ and apply it to making predictions on the test set:

$$k\text{DPPmix}_t = \arg\max_{i \in C_t} \sum_{l=1}^{55} \theta_l \mathcal{P}_{L_l}^6(Y_t \cup \{i\}). \tag{5.45}$$

Best MMR Recall that MMR is a standard, heuristic technique for generating diverse sets of search results. The idea is to build a set iteratively by adding on each round a result that maximizes a weighted combination of relevance (with respect to the query) and diversity, measured as the maximum similarity with any of the previously selected results. (See Section 4.2.1 for more details about MMR.) For our experiments, we assume relevance is uniform; hence we merely need to decide which of the two candidates has the smaller maximum similarity with the partial result set. Thus, for a given kernel L, the MMR

prediction is

$$\mathrm{MMR}_t = \arg\min_{i \in C_t} \left[\max_{j \in Y_t} L_{ij} \right]. \tag{5.46}$$

As for the k-DPP, we select the single best kernel on the training set, and apply it to the test set.

Mixture MMR We can also attempt to learn a mixture of similar kernels for MMR. We use the same training approach as for k-DPPs, but replace the probability score $P^k_\theta(Y_y \cup \{i\})$ with the negative cost

$$-c_\theta(Y_t, i) = -\max_{j \in Y_t} \sum_{l=1}^{D} \theta_l [L_l]_{ij}, \tag{5.47}$$

which is just the negative similarity of item i to the set Y_t under the combined kernel metric. Significantly, this substitution makes the optimization nonsmooth and nonconvex, unlike the k-DPP optimization. In practice this means that the global optimum is not easily found. However, even a local optimum may provide advantages over the single best kernel. In our experiments we use the local optimum found by projected gradient descent starting from the uniform kernel combination.

5.3.5 Results

Table 5.2 shows the mean zero–one accuracy of each method for each query category, averaged over 100 random train/test splits. Statistical significance is computed by bootstrapping. Regardless of whether we learn a mixture, k-DPPs outperform MMR on two of the three categories, significant at 99% confidence. In all cases, the learned mixture of k-DPPs achieves the best performance. Note that, because the decision being made for each instance is binary, 50% is equivalent to random performance. Thus, the numbers in Table 5.2 suggest that this is a rather difficult task, a conclusion supported by the rates of noise exhibited by the human judges. However, the changes in performance due to learning and the use of k-DPPs are more obviously significant when measured as improvements above this baseline level. For example, in the cars category our mixture of k-DPPs performs 14.58 percentage

Table 5.2. Percentage of real-world image search examples judged the same way as the majority of human annotators. Bold results are significantly higher than others in the same row with 99% confidence.

Category	Best MMR	Best k-DPP	Mixture MMR	Mixture k-DPP
CARS	55.95	57.98	59.59	**64.58**
CITIES	56.48	56.31	60.99	61.29
DOGS	56.23	57.70	57.39	**59.84**

Fig. 5.4 Samples from the k-DPP mixture model.

points better than random versus 9.59 points for MMR with a mixture of kernels. Figure 5.4 shows some actual samples drawn using the k-DPP sampling algorithm.

Table 5.3. Kernels receiving the highest average weights for each category (shown in parentheses). Ampersands indicate kernels generated from pairs of feature functions.

CARS	color-8-center & sift-256	(0.13)
	color-8-center & sift-512	(0.11)
	color-8-center	(0.07)
CITIES	sift-512-center	(0.85)
	gist	(0.08)
	color-8-center & gist	(0.03)
DOGS	color-8-center	(0.39)
	color-8-center & sift-512	(0.21)
	color-8-center & sift-256	(0.20)

Table 5.3 shows, for the k-DPP mixture model, the kernels receiving the highest weights for each search category (on average over 100 train/test splits). Combined-feature kernels appear to be useful, and the three categories exhibit significant differences in what annotators deem diverse, as we might expect.

We can also return to our original motivation and try to measure how well each method "covers" the space of likely user intentions. Since we do not have access to real users who are searching for the queries in our dataset, we instead simulate them by imagining that each is looking for a particular target image drawn randomly from the images in our collection. For instance, given the query "philadelphia" we might draw a target image of the Love sculpture, and then evaluate each method on whether it selects an image of the Love sculpture, i.e., whether it satisfies that virtual user. More generally, we will simply record the maximum similarity of any image in the result set to the target image. We expect better methods to show higher similarity when averaged over a large number of such users.

We consider only the mixture models here, since they perform best. For each virtual user, we sample a ten-image result set Y_{DPP} using the mixture k-DPP, and select a second ten-image result set Y_{MMR} using the mixture MMR. For MMR, the first image is selected uniformly at random, since they are assumed to be uniformly relevant. Subsequent selections are deterministic. Given a target image i drawn uniformly at

Table 5.4. The percentage of virtual users whose desired image is more similar to the k-DPP results than the MMR results. Above 50 indicates better k-DPP performance; below 50 indicates better MMR performance. The results for the 55 individual expert kernels are averaged in the first column.

Category	Single kernel (average)	Uniform mixture	MMR mixture
CARS	57.58	68.31	58.15
CITIES	59.00	64.76	62.32
DOGS	57.78	62.12	57.86

random, we then compute similarities

$$s_{\mathrm{DPP}}(i) = \max_{j \in Y_{\mathrm{DPP}}} L_{ij} \quad s_{\mathrm{MMR}}(i) = \max_{j \in Y_{\mathrm{MMR}}} L_{ij} \tag{5.48}$$

for a particular similarity kernel L. We report the fraction of the time that $s_{\mathrm{DPP}}(i) > s_{\mathrm{MMR}}(i)$; that is, the fraction of the time that our virtual user would be better served by the k-DPP model. Because we have no gold standard kernel L for measuring similarity, we try several possibilities, including all 55 expert kernels, a uniform combination of the expert kernels, and the combination learned by MMR. (Note that the mixture k-DPP does not learn a kernel combination, hence there is no corresponding mixture to try here.) Table 5.4 shows the results, averaged across all of the virtual users (i.e., all the images in our collection). Even when using the mixture learned to optimize MMR itself, the k-DPP does a better job of covering the space of possible user intentions. All results in Table 5.4 are significantly higher than 50% at 99% confidence.

6

Structured DPPs

We have seen in the preceding sections that DPPs offer polynomial-time inference and learning with respect to N, the number of items in the ground set $\mathcal{Y}$. This is important since DPPs model an exponential number of subsets $Y \subseteq \mathcal{Y}$, so naive algorithms would be intractable. And yet, we can imagine DPP applications for which even linear time is too slow. For example, suppose that after modeling the positions of basketball players, as proposed in the previous section, we wanted to take our analysis one step further. An obvious extension is to realize that a player does not simply occupy a single position, but instead moves around the court over time. Thus, we might want to model not just diverse sets of positions on the court, but diverse sets of *paths* around the court during a game. While we could reasonably discretize the possible court positions to a manageable number M, the number of paths over, say, 100 time steps would be M^{100}, making it almost certainly impossible to enumerate them all, let alone build an $M^{100} \times M^{100}$ kernel matrix.

However, in this combinatorial setting we can take advantage of the fact that, even though there are exponentially many paths, they are *structured*; that is, every path is built from a small number of the

same basic components. This kind of structure has frequently been exploited in machine learning, for example, to find the best translation of a sentence, or to compute the marginals of a Markov random field. In such cases structure allows us to factor computations over exponentially many possibilities in an efficient way. And yet, the situation for structured DPPs is even worse: when the number of items in $\mathcal{Y}$ is exponential, we are actually modeling a distribution over the *doubly exponential* number of subsets of an exponential $\mathcal{Y}$. If there are M^{100} possible paths, there are $2^{M^{100}}$ subsets of paths, and a DPP assigns a probability to every one. This poses an extreme computational challenge.

In order to develop efficient structured DPPs (SDPPs), we will therefore need to combine the dynamic programming techniques used for standard structured prediction with the algorithms that make DPP inference efficient. We will show how this can be done by applying the dual DPP representation from Section 3.3, which shares spectral properties with the kernel L but is manageable in size, and the use of second-order message passing, where the usual sum-product or min-sum semiring is replaced with a special structure that computes quadratic quantities over a factor graph [92]. In the end, we will demonstrate that it is possible to normalize and sample from an SDPP in polynomial time.

Structured DPPs open up a large variety of new possibilities for applications; they allow us to model diverse sets of essentially any structured objects. For instance, we could find not only the best translation but a diverse set of high-quality translations for a sentence, perhaps to aid a human translator. Or, we could study the distinct proteins coded by a gene under alternative RNA splicings, using the diversifying properties of DPPs to cover the large space of possibilities with a small representative set. Later, we will apply SDPPs to three real-world tasks: identifying multiple human poses in images, where there are combinatorially many possible poses, and we assume that the poses are diverse in that they tend not to overlap; identifying salient lines of research in a corpus of computer science publications, where the structures are citation chains of important papers, and we want to find a small number of chains that cover the major topic in the corpus; and building threads from news text, where the goal is to extract from a

large corpus of articles the most significant news stories, and for each story present a sequence of articles covering the major developments of that story through time.

We begin by defining SDPPs and stating the structural assumptions that are necessary to make inference efficient; we then show how these assumptions give rise to polynomial-time algorithms using second-order message passing. We discuss how sometimes even these polynomial algorithms can be too slow in practice, but demonstrate that by applying the technique of random projections (Section 3.4) we can dramatically speed up computation and reduce memory use while maintaining a close approximation to the original model [83]. Finally, we show how SDPPs can be applied to the experimental settings described above, yielding improved results compared with a variety of standard and heuristic baseline approaches.

6.1 Factorization

In Section 2.4 we saw that DPPs remain tractable on modern computers for N up to around 10,000. This is no small feat, given that the number of subsets of 10,000 items is roughly the number of particles in the observable universe to the 40th power. Of course, this is not magic but simply a consequence of a certain type of *structure*; that is, we can perform inference with DPPs because the probabilities of these subsets are expressed as combinations of only a relatively small set of $O(N^2)$ parameters. In order to make the jump now to ground sets $\mathcal{Y}$ that are exponentially large, we will need to make a similar assumption about the structure of $\mathcal{Y}$ itself. Thus, a structured DPP (SDPP) is a DPP in which the ground set $\mathcal{Y}$ is given implicitly by combinations of a set of *parts*. For instance, the parts could be positions on the court, and an element of $\mathcal{Y}$ a sequence of those positions. Or the parts could be rules of a context-free grammar, and then an element of $\mathcal{Y}$ might be a complete parse of a sentence. This assumption of structure will give us the algorithmic leverage we need to efficiently work with a distribution over a doubly exponential number of possibilities.

Because elements of $\mathcal{Y}$ are now structures, we will no longer think of $\mathcal{Y} = \{1, 2, \ldots, N\}$; instead, each element $\boldsymbol{y} \in \mathcal{Y}$ is a structure given

by a sequence of R parts $(y_1, y_2, \ldots, y_R)$, each of which takes a value from a finite set of M possibilities. For example, if $\boldsymbol{y}$ is the path of a basketball player, then R is the number of time steps at which the player's position is recorded, and y_r is the player's discretized position at time r. We will use $\boldsymbol{y}_i$ to denote the i-th structure in $\mathcal{Y}$ under an arbitrary ordering; thus $\mathcal{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$, where $N = M^R$. The parts of $\boldsymbol{y}_i$ are denoted y_{ir}.

An immediate challenge is that the kernel L, which has N^2 entries, can no longer be written down explicitly. We therefore define its entries using the quality/diversity decomposition presented in Section 3.1. Recall that this decomposition gives the entries of L as follows:

$$L_{ij} = q(\boldsymbol{y}_i)\phi(\boldsymbol{y}_i)^\top \phi(\boldsymbol{y}_j) q(\boldsymbol{y}_j), \tag{6.1}$$

where $q(\boldsymbol{y}_i)$ is a nonnegative measure of the quality of structure $\boldsymbol{y}_i$, and $\phi(\boldsymbol{y}_i)$ is a D-dimensional vector of diversity features so that $\phi(\boldsymbol{y}_i)^\top \phi(\boldsymbol{y}_j)$ is a measure of the similarity between structures $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$. We cannot afford to specify q and ϕ for every possible structure, but we can use the assumption that structures are built from parts to define a factorization, analogous to the factorization over cliques that give rise to Markov random fields.

Specifically, we assume that the model decomposes over a set of *factors* F, where a factor $\alpha \in F$ is a small subset of the parts of a structure. (Keeping the factors small will ensure that the model is tractable.) We denote by $\boldsymbol{y}_\alpha$ the collection of parts of $\boldsymbol{y}$ that are included in factor α; then the factorization assumption is that the quality score decomposes multiplicatively over parts, and the diversity features decompose additively:

$$q(\boldsymbol{y}) = \prod_{\alpha \in F} q_\alpha(\boldsymbol{y}_\alpha) \tag{6.2}$$

$$\phi(\boldsymbol{y}) = \sum_{\alpha \in F} \phi_\alpha(\boldsymbol{y}_\alpha). \tag{6.3}$$

We argue that these are quite natural factorizations. For instance, in our player tracking example we might have a positional factor for each time r, allowing the quality model to prefer paths that go through certain high-traffic areas, and a transitional factor for each pair of times

$(r-1, r)$, allowing the quality model to enforce the smoothness of a path over time. More generally, if the parts correspond to cliques in a graph, then the quality scores can be given by a standard log-linear Markov random field (MRF), which defines a multiplicative distribution over structures that give labelings of the graph. Thus, while in Section 3.2 we compared DPPs and MRFs as alternative models for the same binary labeling problems, SDPPs can also be seen as an extension to MRFs, allowing us to take a model of individual structures and use it as a quality measure for modeling diverse sets of structures.

Diversity features, on the other hand, decompose additively, so we can think of them as global feature functions defined by summing local features, again as done in standard structured prediction. For example, $\phi_r(y_r)$ could track the coarse-level position of a player at time r, so that paths passing through similar positions at similar times are less likely to co-occur. Note that, in contrast to the unstructured case, we do not generally have $\|\phi(\boldsymbol{y})\| = 1$, since there is no way to enforce such a constraint under the factorization in Equation (6.3). Instead, we simply set the factor features $\phi_\alpha(\boldsymbol{y}_\alpha)$ to have unit norm for all α and all possible values of $\boldsymbol{y}_\alpha$. This slightly biases the model toward structures that have the same (or similar) features at every factor, since such structures maximize $\|\phi\|$. However, the effect of this bias seems to be minor in practice.

As for unstructured DPPs, the quality and diversity models combine to produce balanced, high-quality, diverse results. However, in the structured case the contribution of the diversity model can be especially significant due to the combinatorial nature of the items in $\mathcal{Y}$. For instance, imagine taking a particular high-quality path and perturbing it slightly, say by shifting the position at each time step by a small random amount. This process results in a new and distinct path, but is unlikely to have a significant effect on the overall quality: the path remains smooth and goes through roughly the same positions. Of course, this is not unique to the structured case; we can have similar high-quality items in any DPP. What makes the problem especially serious here is that there is a combinatorial number of such slightly perturbed paths; the introduction of structure dramatically increases not only the number of items in $\mathcal{Y}$, but also the number of subtle

variations that we might want to suppress. Furthermore, factored distributions over structures are often very peaked due to the geometric combination of quality scores across many factors, so variations of the most likely structure can be much more probable than any real alternative. For these reasons independent samples from an MRF can often look nearly identical; a sample from an SDPP, on the other hand, is much more likely to contain a truly diverse set of structures.

6.1.1 Synthetic Example: Particle Tracking

Before describing the technical details needed to make SDPPs computationally efficient, we first develop some intuition by studying the results of the model as applied to a synthetic motion tracking task, where the goal is to follow a collection of particles as they travel in a one-dimensional space over time. This is essentially a simplified version of our player tracking example, but with the motion restricted to a line. We will assume that a path $\boldsymbol{y}$ has 50 parts, where each part $y_r \in \{1, 2, \ldots, 50\}$ is the particle's position at time step r discretized into one of 50 locations. The total number of possible trajectories in this setting is 50^{50}, and we will be modeling $2^{50^{50}}$ possible sets of trajectories. We define positional and transitional factors

$$F = \{\{r\} \mid r = 1, 2, \ldots, 50\} \cup \{\{r-1, r\} \mid r = 2, 3, \ldots, 50\}. \tag{6.4}$$

While a real tracking problem would involve quality scores $q(\boldsymbol{y})$ that depend on some observations — for example, measurements over time from a set of physical sensors, or perhaps a video feed from a basketball game — for simplicity we determine the quality of a trajectory here using only its starting position and a measure of smoothness over time. Specifically, we have

$$q(\boldsymbol{y}) = q_1(y_1) \prod_{r=2}^{50} q(y_{r-1}, y_r), \tag{6.5}$$

where the initial quality score $q_1(y_1)$ is given by a smooth trimodal function with a primary mode at position 25 and secondary modes at positions 10 and 40, depicted by the blue curves on the left side of Figure 6.1, and the quality scores for all other positional factors are

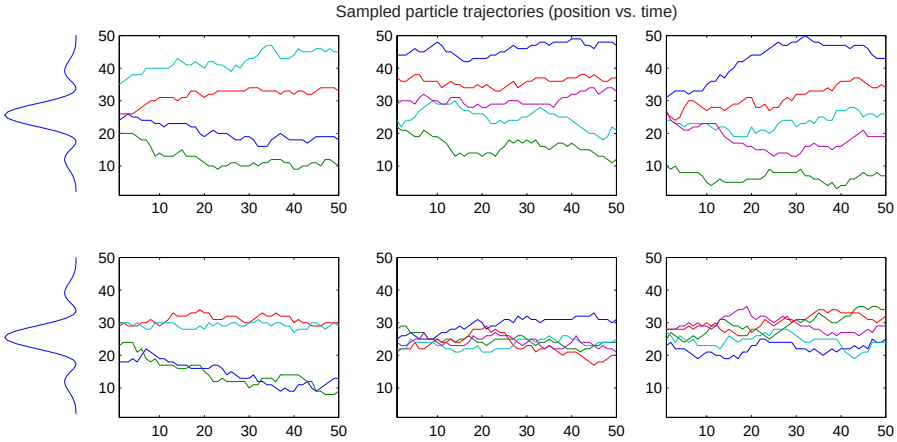


Fig. 6.1 Sets of particle trajectories sampled from an SDPP (top row) and independently using only quality scores (bottom row). The curves to the left indicate quality scores for the initial positions of the particles.

fixed to one and have no effect. The transition quality is the same at all time steps, and given by $q(y_{r-1}, y_r) = f_{\mathcal{N}}(y_{r-1} - y_r)$, where $f_{\mathcal{N}}$ is the density function of the normal distribution; that is, the quality of a transition is maximized when the particle does not change location, and decreases as the particle moves further and further from its previous location. In essence, high-quality paths start near the central position and move smoothly through time.

We want trajectories to be considered similar if they travel through similar positions, so we define a 50-dimensional diversity feature vector as follows:

$$\phi(\boldsymbol{y}) = \sum_{r=1}^{50} \phi_r(y_r) \tag{6.6}$$

$$\phi_{rl}(y_r) \propto f_{\mathcal{N}}(l - y_r), \quad l = 1, 2, \ldots, 50. \tag{6.7}$$

Intuitively, feature l is activated when the trajectory passes near position l, so trajectories passing through nearby positions will activate the same features and thus appear similar in the diversity model. Note that for simplicity, the time at which a particle reaches a given position has no effect on the diversity features. The diversity features for the transitional factors are zero and have no effect.

We use the quality and diversity models specified above to define our SDPP. In order to obtain good results for visualization, we scale the kernel so that the expected number of trajectories in a sample from the SDPP is five. We then apply the algorithms developed later to draw samples from the model. The first row of Figure 6.1 shows the results, and for comparison each corresponding panel on the second row shows an equal number of trajectories sampled independently, with probabilities proportional to their quality scores. As evident from the figure, trajectories sampled independently tend to cluster in the middle region due to the strong preference for this starting position. The SDPP samples, however, are more diverse, tending to cover more of the space while still respecting the quality scores — they are still smooth, and still tend to start near the center.

6.2 Second-order Message Passing

The central computational challenge for SDPPs is the fact that $N = M^R$ is exponentially large, making the usual inference algorithms intractable. However, we showed in Section 3.3 that DPP inference can be recast in terms of a smaller dual representation C; recall that, if B is the $D \times N$ matrix whose columns are given by $B_{\boldsymbol{y}_i} = q(\boldsymbol{y}_i)\phi(\boldsymbol{y}_i)$, then $L = B^\top B$ and

$$C = BB^\top \tag{6.8}$$

$$= \sum_{\boldsymbol{y} \in \mathcal{Y}} q^2(\boldsymbol{y})\phi(\boldsymbol{y})\phi(\boldsymbol{y})^\top. \tag{6.9}$$

Of course, for the dual representation to be of any use we must be able to efficiently compute C. If we think of $q_\alpha^2(\boldsymbol{y}_\alpha)$ as the factor potentials of a graphical model $p(\boldsymbol{y}) \propto \prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_\alpha)$, then computing C is equivalent to computing second moments of the diversity features under p (up to normalization). Since the diversity features factor additively, C is quadratic in the local diversity features $\phi_\alpha(\boldsymbol{y}_\alpha)$. Thus, we could naively calculate C by computing the pairwise marginals $p(\boldsymbol{y}_\alpha, \boldsymbol{y}_{\alpha'})$ for all realizations of the factors α, α' and, by linearity of expectations, adding up their contributions:

$$C \propto \sum_{\alpha, \alpha'} \sum_{\boldsymbol{y}_\alpha, \boldsymbol{y}_{\alpha'}} p(\boldsymbol{y}_\alpha, \boldsymbol{y}_{\alpha'})\phi_\alpha(\boldsymbol{y}_\alpha)\phi_{\alpha'}(\boldsymbol{y}_{\alpha'})^\top, \tag{6.10}$$

where the proportionality is due to the normalizing constant of $p(\boldsymbol{y})$. However, this sum is quadratic in the number of factors and their possible realizations, and can therefore be expensive when structures are large.

Instead, we can substitute the factorization from Equation (6.3) into Equation (6.9) to obtain

$$C = \sum_{\boldsymbol{y} \in \mathcal{Y}} \left(\prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} \phi_\alpha(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} \phi_\alpha(\boldsymbol{y}_\alpha) \right)^\top . \tag{6.11}$$

It turns out that this expression is computable in linear time using a second-order message passing algorithm.

Second-order message passing was first introduced by Li and Eisner [92]. The main idea is to compute second-order statistics over a graphical model by using the standard belief propagation message passing algorithm, but with a special semiring in place of the usual sum-product or max-product. This substitution makes it possible to compute quantities of the form

$$\sum_{\boldsymbol{y} \in \mathcal{Y}} \left(\prod_{\alpha \in F} p_\alpha(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} a_\alpha(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} b_\alpha(\boldsymbol{y}_\alpha) \right), \tag{6.12}$$

where p_α are nonnegative and a_α and b_α are arbitrary functions. Note that we can think of p_α as defining a multiplicatively decomposed function

$$p(\boldsymbol{y}) = \prod_{\alpha \in F} p_\alpha(\boldsymbol{y}_\alpha), \tag{6.13}$$

and a_α and b_α as defining corresponding additively decomposed functions a and b.

We begin by defining the notion of a factor graph, which provides the structure for all message passing algorithms. We then describe standard belief propagation on factor graphs, and show how it can be defined in a general way using semirings. Finally we demonstrate that belief propagation using the semiring proposed by Li and Eisner [92] computes quantities of the form in Equation (6.12).

6.2.1 Factor Graphs

Message passing operates on *factor graphs*. A factor graph is an undirected bipartite graph with two types of vertices: *variable* nodes and *factor* nodes. Variable nodes correspond to the parts of the structure being modeled; for the SDPP setup described above, a factor graph contains R variable nodes, each associated with a distinct part r. Similarly, each factor node corresponds to a distinct factor $\alpha \in F$. Every edge in the graph connects a variable node to a factor node, and an edge exists between variable node r and factor node α if and only if $r \in \alpha$. Thus, the factor graph encodes the relationships between parts and factors. Figure 6.2 shows an example factor graph for the tracking problem from Section 6.1.1.

It is obvious that the computation of Equation (6.12) cannot be efficient when factors are allowed to be arbitrary, since in the limit a factor could contain all parts and we could assign arbitrary values to every configuration $\boldsymbol{y}$. Thus we will assume that the degree of the factor nodes is bounded by a constant c. (In Figure 6.2, as well as all of the experiments we run, we have $c = 2$.) Furthermore, message-passing algorithms are efficient whenever the factor graph has low treewidth, or, roughly, when only small sets of nodes need to be merged to obtain a tree. Going forward we will assume that the factor graph is a tree, since any low-treewidth factor graph can be converted into an equivalent factor tree with bounded factors using the junction tree algorithm [89].

6.2.2 Belief Propagation

We now describe the basic belief propagation algorithm, first introduced by Pearl [119]. Suppose each factor has an associated real-valued

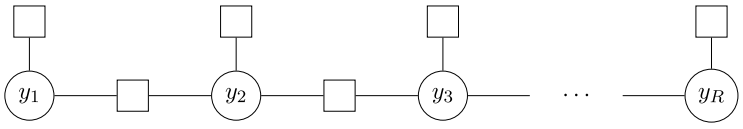


Fig. 6.2 A sample factor graph for the tracking problem. Variable nodes are circular and factor nodes are square. Positional factors that depend only on a single part appear in the top row; binary transitional factors appear between parts in the second row.

weight function $w_\alpha(\boldsymbol{y}_\alpha)$, giving rise to the multiplicatively decomposed global weight function

$$w(\boldsymbol{y}) = \prod_{\alpha \in F} w_\alpha(\boldsymbol{y}_\alpha). \tag{6.14}$$

Then the goal of belief propagation is to efficiently compute sums of $w(\boldsymbol{y})$ over combinatorially large sets of structures $\boldsymbol{y}$.

We will refer to a structure $\boldsymbol{y}$ as an *assignment* to the variable nodes of the factor graph, since it defines a value y_r for every part. Likewise we can think of $\boldsymbol{y}_\alpha$ as an assignment to the variable nodes adjacent to α, and y_r as an assignment to a single variable node r. We use the notation $\boldsymbol{y}_\alpha \sim y_r$ to indicate that $\boldsymbol{y}_\alpha$ is consistent with y_r, in the sense that it assigns the same value to variable node r. Finally, denote by $F(r)$ the set of factors in which variable r participates.

The belief propagation algorithm defines recursive message functions m to be passed along edges of the factor graph; the formula for the message depends on whether it is traveling from a variable node to a factor node, or vice versa:

- From a variable r to a factor α:

$$m_{r \to \alpha}(y_r) = \prod_{\alpha' \in F(r) - \{\alpha\}} m_{\alpha' \to r}(y_r) \tag{6.15}$$

- From a factor α to a variable r:

$$m_{\alpha \to r}(y_r) = \sum_{\boldsymbol{y}_\alpha \sim y_r} \left[w_\alpha(\boldsymbol{y}_\alpha) \prod_{r' \in \alpha - \{r\}} m_{r' \to \alpha}(y_{r'}) \right] \tag{6.16}$$

Intuitively, an outgoing message summarizes all of the messages arriving at the source node, excluding the one coming from the target node. Messages from factor nodes additionally incorporate information about the local weight function.

Belief propagation passes these messages in two phases based on an arbitrary orientation of the factor tree. In the first phase, called the forward pass, messages are passed upward from the leaves to the root. In the second phase, or backward pass, the messages are passed downward, from the root to the leaves. Upon completion of the second

phase one message has been passed in each direction along every edge in the factor graph, and it is possible to prove using an inductive argument that, for every y_r,

$$\prod_{\alpha \in F(r)} m_{\alpha \to t}(y_r) = \sum_{\boldsymbol{y} \sim y_r} \prod_{\alpha \in F} w_\alpha(\boldsymbol{y}_\alpha). \tag{6.17}$$

If we think of the w_α as potential functions, then Equation (6.17) gives the (unnormalized) marginal probability of the assignment y_r under a Markov random field.

Note that the algorithm passes two messages per edge in the factor graph, and each message requires considering at most M^c assignments, therefore its running time is $O(M^c R)$. The sum on the right-hand side of Equation (6.17), however, is exponential in the number of parts. Thus belief propagation offers an efficient means of computing certain combinatorial quantities that would naively require exponential time.

6.2.3 Semirings

In fact, the belief propagation algorithm can be easily generalized to operate over an arbitrary semiring, thus allowing the same basic algorithm to perform a variety of useful computations. Recall that a semiring $\langle W, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$ comprises a set of elements W, an addition operator $\oplus$, a multiplication operator $\otimes$, an additive identity $\mathbf{0}$, and a multiplicative identity $\mathbf{1}$ satisfying the following requirements for all $a, b, c \in W$:

- Addition is associative and commutative, with identity $\mathbf{0}$:

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c \tag{6.18}$$

$$a \oplus b = b \oplus a \tag{6.19}$$

$$a \oplus \mathbf{0} = a \tag{6.20}$$

- Multiplication is associative, with identity $\mathbf{1}$:

$$a \otimes (b \otimes c) = (a \otimes b) \otimes c \tag{6.21}$$

$$a \otimes \mathbf{1} = \mathbf{1} \otimes a = a \tag{6.22}$$

- Multiplication distributes over addition:

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \tag{6.23}$$

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c) \tag{6.24}$$

- $\mathbf{0}$ is absorbing under multiplication:

$$a \otimes \mathbf{0} = \mathbf{0} \otimes a = \mathbf{0} \tag{6.25}$$

Obviously these requirements are met when $W = \mathbb{R}$ and multiplication and addition are the usual arithmetic operations; this is, the standard sum-product semiring. We also have, for example, the max-product semiring, where $W = [0, \infty)$, addition is given by the maximum operator with identity element 0, and multiplication is as before.

We can rewrite the messages defined by belief propagation in terms of these more general operations. For $w_\alpha(\boldsymbol{y}_\alpha) \in W$, we have

$$m_{r \to \alpha}(y_r) = \bigotimes_{\alpha' \in F(r) - \{\alpha\}} m_{\alpha' \to r}(y_r) \tag{6.26}$$

$$m_{\alpha \to r}(y_r) = \bigoplus_{\boldsymbol{y}_\alpha \sim y_r} \left[w_\alpha(\boldsymbol{y}_\alpha) \otimes \bigotimes_{r' \in \alpha - \{r\}} m_{r' \to \alpha}(y_{r'}) \right]. \tag{6.27}$$

As before, we can pass messages forward and then backward through the factor tree. Because the properties of semirings are sufficient to preserve the inductive argument, we then have the following analog of Equation (6.17):

$$\bigotimes_{\alpha \in F(r)} m_{\alpha \to r}(y_r) = \bigoplus_{\boldsymbol{y} \sim y_r} \bigotimes_{\alpha \in F} w_\alpha(\boldsymbol{y}_\alpha). \tag{6.28}$$

We have seen that Equation (6.28) computes marginal probabilities under the sum-product semiring, but other semirings give rise to useful results as well. Under the max-product semiring, for instance, Equation (6.28) is the so-called max-marginal — the maximum unnormalized probability of any single assignment $\boldsymbol{y}$ consistent with y_r. In the next section we take this one step further, and show how a carefully designed semiring will allow us to sum second-order quantities across exponentially many structures $\boldsymbol{y}$.

6.2.4 Second-order Semiring

Li and Eisner [92] proposed the following second-order semiring over four-tuples $(q, \phi, \psi, c) \in W = \mathbb{R}^4$:

$$(q_1, \phi_1, \psi_1, c_1) \oplus (q_2, \phi_2, \psi_2, c_2) = (q_1 + q_2,\ \phi_1 + \phi_2,\ \psi_1 + \psi_2,\ c_1 + c_2) \tag{6.29}$$

$$\begin{aligned}(q_1, \phi_1, \psi_1, c_1) \otimes (q_2, \phi_2, \psi_2, c_2) = (&q_1 q_2,\ q_1\phi_2 + q_2\phi_1,\ q_1\psi_2 + q_2\psi_1,\\ &q_1 c_2 + q_2 c_1 + \phi_1\psi_2 + \phi_2\psi_1)\end{aligned} \tag{6.30}$$

$$\mathbf{0} = (0, 0, 0, 0) \tag{6.31}$$

$$\mathbf{1} = (1, 0, 0, 0) \tag{6.32}$$

It is easy to verify that the semiring properties hold for these operations. Now, suppose that the weight function for a factor α is given by

$$\begin{aligned}w_\alpha(\boldsymbol{y}_\alpha) = (&p_\alpha(\boldsymbol{y}_\alpha),\ p_\alpha(\boldsymbol{y}_\alpha) a_\alpha(\boldsymbol{y}_\alpha),\ p_\alpha(\boldsymbol{y}_\alpha) b_\alpha(\boldsymbol{y}_\alpha),\\ &p_\alpha(\boldsymbol{y}_\alpha) a_\alpha(\boldsymbol{y}_\alpha) b_\alpha(\boldsymbol{y}_\alpha)),\end{aligned} \tag{6.33}$$

where p_α, a_α, and b_α are as before. Then $w_\alpha(\boldsymbol{y}_\alpha) \in W$, and we can get some intuition about the multiplication operator by observing that the fourth component of $w_\alpha(\boldsymbol{y}_\alpha) \otimes w_{\alpha'}(\boldsymbol{y}_{\alpha'})$ is

$$\begin{aligned}&p_\alpha(\boldsymbol{y}_\alpha)[p_{\alpha'}(\boldsymbol{y}_{\alpha'}) a_{\alpha'}(\boldsymbol{y}_{\alpha'}) b_{\alpha'}(\boldsymbol{y}_{\alpha'})] + p_{\alpha'}(\boldsymbol{y}_{\alpha'})[p_\alpha(\boldsymbol{y}_\alpha) a_\alpha(\boldsymbol{y}_\alpha) b_\alpha(\boldsymbol{y}_\alpha)]\\ &\quad + [p_\alpha(\boldsymbol{y}_\alpha) a_\alpha(\boldsymbol{y}_\alpha)][p_{\alpha'}(\boldsymbol{y}_{\alpha'}) b_{\alpha'}(\boldsymbol{y}_{\alpha'})]\\ &\quad + [p_{\alpha'}(\boldsymbol{y}_{\alpha'}) a_{\alpha'}(\boldsymbol{y}_{\alpha'})][p_\alpha(\boldsymbol{y}_\alpha) b_\alpha(\boldsymbol{y}_\alpha)] \qquad (6.34)\\ &= p_\alpha(\boldsymbol{y}_\alpha) p_{\alpha'}(\boldsymbol{y}_{\alpha'})[a_\alpha(\boldsymbol{y}_\alpha) + a_{\alpha'}(\boldsymbol{y}_{\alpha'})][b_\alpha(\boldsymbol{y}_\alpha) + b_{\alpha'}(\boldsymbol{y}_{\alpha'})]. \qquad (6.35)\end{aligned}$$

In other words, multiplication in the second-order semiring combines the values of p multiplicatively and the values of a and b additively, leaving the result in the fourth component. It is not hard to extend this argument inductively and show that the fourth component of $\bigotimes_{\alpha \in F} w_\alpha(\boldsymbol{y}_\alpha)$ is given in general by

$$\left(\prod_{\alpha \in F} p_\alpha(\boldsymbol{y}_\alpha)\right)\left(\sum_{\alpha \in F} a_\alpha(\boldsymbol{y}_\alpha)\right)\left(\sum_{\alpha \in F} b_\alpha(\boldsymbol{y}_\alpha)\right). \tag{6.36}$$

Thus, by Equation (6.28) and the definition of $\oplus$, belief propagation with the second-order semiring yields messages that satisfy

$$\left[\bigotimes_{\alpha \in F(r)} m_{\alpha \to r}(y_r)\right]_4 = \sum_{\boldsymbol{y} \sim y_r} \left(\prod_{\alpha \in F} p_\alpha(\boldsymbol{y}_\alpha)\right) \left(\sum_{\alpha \in F} a_\alpha(\boldsymbol{y}_\alpha)\right) \left(\sum_{\alpha \in F} b_\alpha(\boldsymbol{y}_\alpha)\right). \tag{6.37}$$

Note that multiplication and addition remain constant-time operations in the second-order semiring, thus belief propagation can still be performed in time linear in the number of factors. In the following section we will show that the dual representation C, as well as related quantities needed to perform inference in SDPPs, takes the form of Equation (6.37); thus second-order message passing will be an important tool for efficient SDPP inference.

6.3 Inference

The factorization proposed in Equation (6.3) gives a concise definition of a structured DPP for an exponentially large $\mathcal{Y}$; remarkably, under suitable conditions it also gives rise to tractable algorithms for normalizing the SDPP, computing marginals, and sampling. The only restrictions necessary for efficiency are the ones we inherit from belief propagation: the factors must be of bounded size so that we can enumerate all of their possible configurations, and together they must form a low-treewidth graph on the parts of the structure. These are precisely the same conditions needed for efficient graphical model inference [78], which is generalized by inference in SDPPs.

6.3.1 Computing C

As we saw in Section 3.3, the dual representation C is sufficient to normalize and marginalize an SDPP in time constant in N. Recall from Equation (6.11) that the dual representation of an SDPP can be written as

$$C = \sum_{\boldsymbol{y} \in \mathcal{Y}} \left(\prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_\alpha)\right) \left(\sum_{\alpha \in F} \phi_\alpha(\boldsymbol{y}_\alpha)\right) \left(\sum_{\alpha \in F} \phi_\alpha(\boldsymbol{y}_\alpha)\right)^\top, \tag{6.38}$$

which is of the form required to apply second-order message passing. Specifically, we can compute for each pair of diversity features (a, b) the value of

$$\sum_{\boldsymbol{y} \in \mathcal{Y}} \left(\prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} \phi_{\alpha a}(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} \phi_{\alpha b}(\boldsymbol{y}_\alpha) \right) \tag{6.39}$$

by summing Equation (6.37) over the possible assignments y_r, and then simply assemble the results into the matrix C. Since there are $\frac{D(D+1)}{2}$ unique entries in C and message passing runs in time $O(M^c R)$, computing C in this fashion requires $O(D^2 M^c R)$ time.

We can make several practical optimizations to this algorithm, though they will not affect the asymptotic performance. First, we note that the full set of messages at *any* variable node r is sufficient to compute Equation (6.39). Thus, during message passing we need only perform the forward pass; at that point, the messages at the root node are complete and we can obtain the quantity we need. This speeds up the algorithm by a factor of two. Second, rather than running message passing D^2 times, we can run it only once using a vectorized second-order semiring. This has no effect on the total number of operations, but can result in significantly faster performance due to vector optimizations in modern processors. The vectorized second-order semiring is over four-tuples (q, ϕ, ψ, C) where $q \in \mathbb{R}$, $\phi, \psi \in \mathbb{R}^D$, and $C \in \mathbb{R}^{D \times D}$, and uses the following operations:

$$(q_1, \phi_1, \psi_1, C_1) \oplus (q_2, \phi_2, \psi_2, C_2) = (q_1 + q_2,\, \phi_1 + \phi_2,\, \psi_1 + \psi_2,\, C_1 + C_2) \tag{6.40}$$

$$\begin{aligned}(q_1, \phi_1, \psi_1, C_1) \otimes (q_2, \phi_2, \psi_2, C_2) = (&q_1 q_2,\; q_1\phi_2 + q_2\phi_1,\; q_1\psi_2 + q_2\psi_1,\\ &q_1 C_2 + q_2 C_1 + \phi_1\psi_2^\top + \phi_2\psi_1^\top)\end{aligned} \tag{6.41}$$

$$\mathbf{0} = (0, \mathbf{0}, \mathbf{0}, \mathbf{0}) \tag{6.42}$$

$$\mathbf{1} = (1, \mathbf{0}, \mathbf{0}, \mathbf{0}). \tag{6.43}$$

It is easy to verify that computations in this vectorized semiring are identical to those obtained by repeated use of the scalar semiring.

Given C, we can now normalize and compute marginals for an SDPP using the formulas in Section 3.3; for instance

$$K_{ii} = \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} (B_i^\top \hat{\boldsymbol{v}}_n)^2 \tag{6.44}$$

$$= q^2(\boldsymbol{y}_i) \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} (\phi(\boldsymbol{y}_i)^\top \hat{\boldsymbol{v}}_n)^2, \tag{6.45}$$

where $C = \sum_{n=1}^{D} \lambda_n \hat{\boldsymbol{v}}_n \hat{\boldsymbol{v}}_n^\top$ is an eigendecomposition of C.

Part marginals The introduction of structure offers an alternative type of marginal probability, this time not of structures $\boldsymbol{y} \in \mathcal{Y}$ but of single part assignments. More precisely, we can ask how many of the structures in a sample from the SDPP can be expected to make the assignment $\hat{y}_r$ to part r:

$$\mu_r(\hat{y}_r) = \mathbb{E}\left[\sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbb{I}(\boldsymbol{y} \in \boldsymbol{Y} \wedge y_r = \hat{y}_r)\right] \tag{6.46}$$

$$= \sum_{\boldsymbol{y} \sim \hat{y}_r} \mathcal{P}_L(\boldsymbol{y} \in \boldsymbol{Y}). \tag{6.47}$$

The sum is exponential, but we can compute it efficiently using second-order message passing. We apply Equation (6.44) to get

$$\sum_{\boldsymbol{y} \sim \hat{y}_r} \mathcal{P}_L(\boldsymbol{y} \in \boldsymbol{Y}) = \sum_{\boldsymbol{y} \sim \hat{y}_r} q^2(\boldsymbol{y}) \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} (\phi(\boldsymbol{y})^\top \hat{\boldsymbol{v}}_n)^2 \tag{6.48}$$

$$= \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} \sum_{\boldsymbol{y} \sim \hat{y}_r} q^2(\boldsymbol{y}) (\phi(\boldsymbol{y})^\top \hat{\boldsymbol{v}}_n)^2 \tag{6.49}$$

$$= \sum_{n=1}^{D} \frac{\lambda_n}{\lambda_n + 1} \sum_{\boldsymbol{y} \sim \hat{y}_r} \left(\prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_\alpha)\right) \left(\sum_{\alpha \in F} \phi_\alpha(\boldsymbol{y}_\alpha)^\top \hat{\boldsymbol{v}}_n\right)^2. \tag{6.50}$$

The result is a sum of D terms, each of which takes the form of Equation (6.37), and therefore is efficiently computable by message passing.

The desired part marginal probability simply requires D separate applications of belief propagation, one per eigenvector $\hat{\boldsymbol{v}}_n$, for a total runtime of $O(D^2M^cR)$. (It is also possible to vectorize this computation and use a single run of belief propagation.) Note that if we require the marginal for only a single part $\mu_r(\hat{y}_r)$, we can run just the forward pass if we root the factor tree at part node r. However, by running both passes we obtain everything we need to compute the part marginals for any r and $\hat{y}_r$; the asymptotic time required to compute all part marginals is the same as the time required to compute just one.

6.3.2 Sampling

While the dual representation provides useful algorithms for normalization and marginals, the dual sampling algorithm is linear in N; for SDPPs, this is too slow to be useful. In order to make SDPP sampling practical, we need to be able to efficiently choose a structure $\boldsymbol{y}_i$ according to the distribution

$$\Pr(\boldsymbol{y}_i) = \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} (\hat{\boldsymbol{v}}^\top B_i)^2 \tag{6.51}$$

in the first line of the while loop in Algorithm 3. We can use the definition of B to obtain

$$\Pr(\boldsymbol{y}_i) = \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} q^2(\boldsymbol{y}_i)(\hat{\boldsymbol{v}}^\top \phi(\boldsymbol{y}_i))^2 \tag{6.52}$$

$$= \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} \left(\prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_{i\alpha}) \right) \left(\sum_{\alpha \in F} \hat{\boldsymbol{v}}^\top \phi_\alpha(\boldsymbol{y}_{i\alpha}) \right)^2 . \tag{6.53}$$

Thus, the desired distribution has the familiar form of Equation (6.37). For instance, the marginal probability of part r taking the assignment $\hat{y}_r$ is given by

$$\frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} \sum_{\boldsymbol{y} \sim \hat{y}_r} \left(\prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} \hat{\boldsymbol{v}}^\top \phi_\alpha(\boldsymbol{y}_\alpha) \right)^2 , \tag{6.54}$$

which we can compute with $k = |\hat{V}|$ runs of belief propagation (or a single vectorized run), taking only $O(DM^cRk)$ time. More generally,

the message-passing computation of these marginals offers an efficient algorithm for sampling individual full structures $\boldsymbol{y}_i$. We will first show a naive method based on iterated computation of conditional marginals, and then use it to derive a more efficient algorithm by integrating the sampling of parts into the message-passing process.

Single structure sampling Returning to the factor graph used for belief propagation (see Section 6.2.1), we can force a part r' to take a certain assignment $y_{r'}$ by adding a new singleton factor containing only r', and setting its weight function to $\mathbf{1}$ for $y_{r'}$ and $\mathbf{0}$ otherwise. (In practice, we do not need to actually create a new factor; we can simply set outgoing messages from variable r' to $\mathbf{0}$ for all but the desired assignment $y_{r'}$.) It is easy to see that Equation (6.28) becomes

$$\bigotimes_{\alpha \in F(r)} m_{\alpha \to r}(y_r) = \bigoplus_{\boldsymbol{y} \sim y_r, y_{r'}} \bigotimes_{\alpha \in F} w_\alpha(\boldsymbol{y}_\alpha), \tag{6.55}$$

where the sum is now doubly constrained, since any assignment $\boldsymbol{y}$ that is not consistent with $y_{r'}$ introduces a $\mathbf{0}$ into the product. If $\bigotimes_{\alpha \in F} w_\alpha(\boldsymbol{y}_\alpha)$ gives rise to a probability measure over structures $\boldsymbol{y}$, then Equation (6.55) can be seen as the unnormalized *conditional* marginal probability of the assignment y_r given $y_{r'}$. For example, using the second-order semiring with $p = q^2$ and $a = b = \hat{\boldsymbol{v}}^\top \phi$, we have

$$\left[\bigotimes_{\alpha \in F(r)} m_{\alpha \to r}(y_r) \right]_4 = \sum_{\boldsymbol{y} \sim y_r, y_{r'}} \left(\prod_{\alpha \in F} q_\alpha^2(\boldsymbol{y}_\alpha) \right) \left(\sum_{\alpha \in F} \hat{\boldsymbol{v}}^\top \phi_\alpha(\boldsymbol{y}_\alpha) \right)^2 . \tag{6.56}$$

Summing these values for all $\hat{\boldsymbol{v}} \in \hat{V}$ and normalizing the result yields the conditional distribution of y_r given fixed assignment $y_{r'}$ under Equation (6.53). Going forward we will assume for simplicity that $\hat{V}$ contains a single vector $\hat{\boldsymbol{v}}$; however, the general case is easily handled by maintaining $|\hat{V}|$ messages in parallel or by vectorizing the computation.

The observation that we can compute conditional probabilities with certain assignments held fixed gives rise to a naive algorithm for sampling a structure according to $\Pr(\boldsymbol{y}_i)$ in Equation (6.53), shown in Algorithm 9. While polynomial, Algorithm 9 requires running belief propagation R times, which might be prohibitively expensive for large

Algorithm 9 Sampling a structure (naive)

Input: factored q and ϕ, $\hat{\boldsymbol{v}}$
$S \leftarrow \emptyset$
for $r = 1, 2, \ldots, R$ **do**
  Run second-order belief propagation with:
  - $p = q^2$
  - $a = b = \hat{\boldsymbol{v}}^\top \phi$
  - assignments in S held fixed

  Sample y_r according to $\Pr(y_r|S) \propto \left[\bigotimes_{\alpha \in F(r)} m_{\alpha \to r}(y_r)\right]_4$
  $S \leftarrow S \cup \{y_r\}$
end for
Output: $\boldsymbol{y}$ constructed from S

structures. We can do better by weaving the sampling steps into a single run of belief propagation. We discuss first how this can be done for linear factor graphs, where the intuition is simpler, and then extend it to general factor trees.

Linear graphs Suppose that the factor graph is a linear chain arranged from left to right. Each node in the graph has at most two neighbors — one to the left and one to the right. Assume the belief propagation forward pass proceeds from left to right, and the backward pass from right to left. To send a message to the right, a node needs only to receive its message from the left. Conversely, to send a message to the left, only the message from the right is needed. Thus, the forward and backward passes can be performed independently.

Consider now the execution of Algorithm 9 on this factor graph. Assume the variable nodes are numbered in decreasing order from left to right, so the variable sampled in the first iteration is the rightmost variable node. Observe that on iteration r, we do not actually need to run belief propagation to completion; we need only the messages incoming to variable node r, since those suffice to compute the (conditional) marginals for part r. To obtain those messages, we must compute all of the forward messages sent from the left of variable r, and the backward messages from the right. Call this set of messages $\boldsymbol{m}(r)$.

Note that $\boldsymbol{m}(1)$ is just a full, unconstrained forward pass, which can be computed in time $O(DM^cR)$. Now compare $\boldsymbol{m}(r)$ to $\boldsymbol{m}(r-1)$. Between iteration $r-1$ and r, the only change to S is that variable $r-1$, to the right of variable r, has been assigned. Therefore, the forward messages in $\boldsymbol{m}(r)$, which come from the left, do not need to be recomputed, as they are a subset of the forward messages in $\boldsymbol{m}(r-1)$. Likewise, the backward messages sent from the right of variable $r-1$ are unchanged, so they do not need to be recomputed. The only new messages in $\boldsymbol{m}(r)$ are those backward messages traveling from $r-1$ to r. These can be computed, using $\boldsymbol{m}(r-1)$ and the sampled assignment y_{r-1}, in constant time. See Figure 6.3 for an illustration of this process.

Thus, rather than restarting belief propagation on each loop of Algorithm 9, we have shown that we need only compute a small number of additional messages. In essence we have threaded the sampling of parts r into the backward pass. After completing the forward pass, we sample y_1; we then compute the backward messages from y_1 to y_2, sample y_2, and so on. When the backward pass is complete, we sample the final assignment y_R and are finished. Since the initial forward pass takes $O(DM^cR)$ time and each of the $O(R)$ subsequent iterations takes at most $O(DM^c)$ time, we can sample from $\Pr(\boldsymbol{y}_i)$ over a linear graph in $O(DM^cR)$ time.

Trees The algorithm described above for linear graphs can be generalized to arbitrary factor trees. For standard graphical model

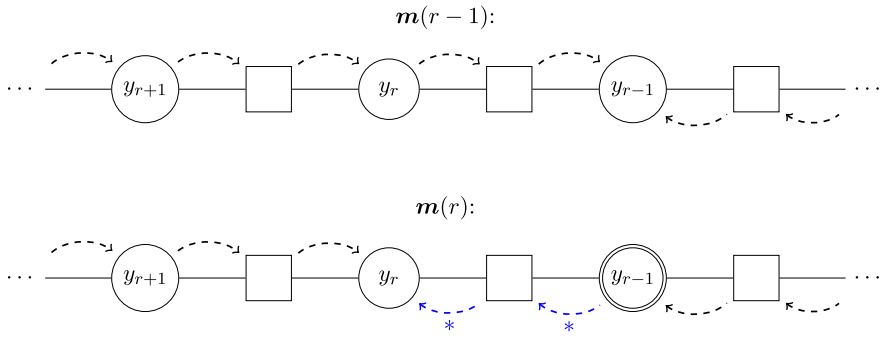


Fig. 6.3 Messages on a linear chain. Only the starred messages need to be computed to obtain $\boldsymbol{m}(r)$ from $\boldsymbol{m}(r-1)$. The double circle indicates that assignment y_{r-1} has been fixed for computing $\boldsymbol{m}(r)$.

sampling using the sum-product semiring, the generalization is straightforward — we can simply pass messages up to the root and then sample on the backward pass from the root to the leaves. However, for arbitrary semirings this algorithm is incorrect, since an assignment to one node can affect the messages arriving at its siblings even when the parent's assignment is fixed.

Let $m_{b \to a}(\cdot|S)$ be the message function sent from node b to node a during a run of belief propagation where the assignments in S have been held fixed. Imagine that we re-root the factor tree with a as the root; then define $T_a(b)$ to be the subtree rooted at b (see Figure 6.4). Several useful observations follow.

Lemma 6.1. If b_1 and b_2 are distinct neighbors of a, then $T_a(b_1)$ and $T_a(b_2)$ are disjoint.

Proof. The claim is immediate, since the underlying graph is a tree. □

Lemma 6.2. $m_{b \to a}(\cdot|S)$ can be computed given only the messages $m_{c \to b}(\cdot|S)$ for all neighbors $c \neq a$ of b and either the weight function w_b (if b is a factor node) or the assignment to b in S (if b is a variable node and such an assignment exists).

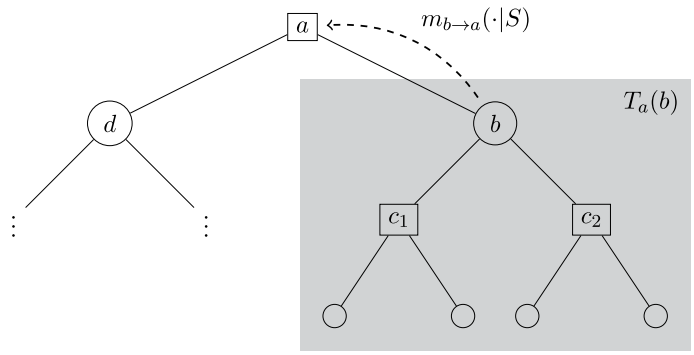


Fig. 6.4 Notation for factor trees, including $m_{b \to a}(\cdot|S)$ and $T_a(b)$ when a is a (square) factor node and b is a (round) variable node. The same definitions apply when a is a variable and b is a factor.

Proof. Follows from the message definitions in Equations (6.26) and (6.27). □

Lemma 6.3. $m_{b\to a}(\cdot|S)$ depends only on the assignments in S that give values to variables in $T_a(b)$.

Proof. If b is a leaf (that is, its only neighbor is a), the lemma holds trivially. If b is not a leaf, then assume inductively that incoming messages $m_{c\to b}(\cdot|S)$, $c \neq a$, depend only on assignments to variables in $T_b(c)$. By Lemma 6.2, the message $m_{b\to a}(\cdot|S)$ depends only on those messages and (possibly) the assignment to b in S. Since b and $T_b(c)$ are subgraphs of $T_a(b)$, the claim follows. □

To sample a structure, we begin by initializing $S_0 = \emptyset$ and setting messages $\hat{m}_{b\to a} = m_{b\to a}(\cdot|S_0)$ for all neighbor pairs (a, b). This can be done in $O(DM^cR)$ time via belief propagation.

Now we walk the graph, sampling assignments and updating the current messages $\hat{m}_{b\to a}$ as we go. Step t from node b to a proceeds in three parts as follows:

1. Check whether b is a variable node without an assignment in S_{t-1}. If so, sample an assignment y_b using the current incoming messages $\hat{m}_{c\to b}$, and set $S_t = S_{t-1} \cup \{y_b\}$. Otherwise set $S_t = S_{t-1}$.
2. Recompute and update $\hat{m}_{b\to a}$ using the current messages and Equations (6.26) and (6.27), taking into account any assignment to b in S_t.
3. Advance to node a.

This simple algorithm has the following useful invariant.

Theorem 6.4. Following step t from b to a, for every neighbor d of a we have

$$\hat{m}_{d\to a} = m_{d\to a}(\cdot|S_t). \tag{6.57}$$

Proof. By design, the theorem holds at the outset of the walk. Suppose inductively that the claim is true for steps $1, 2, \ldots, t-1$. Let t' be the most recent step prior to t at which we visited a, or 0 if step t was our first visit to a. Since the graph is a tree, we know that between steps t' and t the walk remained entirely within $T_a(b)$. Hence the only assignments in $S_t - S_{t'}$ are to variables in $T_a(b)$. As a result, for all neighbors $d \neq b$ of a we have $\hat{m}_{d\to a} = m_{d\to a}(\cdot|S_{t'}) = m_{d\to a}(\cdot|S_t)$ by the inductive hypothesis, Lemma 6.1, and Lemma 6.3.

It remains to show that $\hat{m}_{b\to a} = m_{b\to a}(\cdot|S_i)$. For all neighbors $c \neq a$ of b, we know that $\hat{m}_{c\to b} = m_{c\to b}(\cdot|S_{i-1}) = m_{c\to b}(\cdot|S_t)$ due to the inductive hypothesis and Lemma 6.3 (since b is not in $T_b(c)$). By Lemma 6.2, then, we have $\hat{m}_{b\to a} = m_{b\to a}(\cdot|S_t)$. □

Theorem 6.4 guarantees that whenever we sample an assignment for the current variable node in the first part of step t, we sample from the conditional marginal distribution $\Pr(y_b|S_{t-1})$. Therefore, we can sample a complete structure from the distribution $\Pr(\boldsymbol{y})$ if we walk the entire tree. This can be done, for example, by starting at the root and proceeding in depth-first order. Such a walk takes $O(R)$ steps, and each step requires computing only a single message. Thus, allowing now for $k = |\hat{V}| > 1$, we can sample a structure in time $O(DM^cRk)$, a significant improvement over Algorithm 9. The procedure is summarized in Algorithm 10.

Algorithm 10 is the final piece of machinery needed to replicate the DPP sampling algorithm using the dual representation. The full SDPP sampling process is given in Algorithm 11 and runs in time $O(D^2k^3 + DM^cRk^2)$, where k is the number of eigenvectors selected in the first loop. As in standard DPP sampling, the asymptotically most expensive operation is the orthonormalization; here we require $O(D^2)$ time to compute each of the $O(k^2)$ dot products.

6.4 Experiments: Pose Estimation

To demonstrate that SDPPs effectively model characteristics of real-world data, we apply them to a multiple-person pose estimation task [83]. Our input will be a still image depicting multiple people,

Algorithm 10 Sampling a structure

Input: factored q and ϕ, $\hat{\boldsymbol{v}}$
$S \leftarrow \emptyset$
Initialize $\hat{m}_{a\to b}$ using second-order belief propagation with $p = q^2$, $a = b = \hat{\boldsymbol{v}}^\top \phi$
Let $a_1, a_2, \ldots, a_T$ be a traversal of the factor tree
for $t = 1, 2, \ldots, T$ **do**
  if a_t is a variable node r with no assignment in S **then**
    Sample y_r according to $\Pr(y_r) \propto \left[\bigotimes_{\alpha \in F(r)} \hat{m}_{\alpha\to r}(y_r)\right]_4$
    $S \leftarrow S \cup \{y_r\}$
  end if
  if $t < T$ **then**
    Update $\hat{m}_{a_t \to a_{t+1}}$ using Equations (6.26) and (6.27), fixing assignments in S
  end if
end for
Output: $\boldsymbol{y}$ constructed from S

and our goal is to simultaneously identify the poses — the positions of the torsos, heads, and left and right arms — of all the people in the image. A pose $\boldsymbol{y}$ is therefore a structure with four parts, in this case literally body parts. To form a complete structure, each part r is assigned a position/orientation pair y_r. Our quality model will be based on "part detectors" trained to estimate the likelihood of a particular body part at a particular location and orientation; thus we will focus on identifying poses that correspond well to the image itself. Our similarity model, on the other hand, will focus on the location of a pose within the image. Since the part detectors often have uncertainty about the precise location of a part, there may be many variations of a single pose that outscore the poses of all the other, less detectable people. An independent model would thus be likely to choose many similar poses. By encouraging the model to choose a spatially diverse set of poses, we hope to improve the chance that the model predicts a single pose for each person.

Algorithm 11 Sampling from an SDPP

Input: eigendecomposition $\{(\hat{\boldsymbol{v}}_n, \lambda_n)\}_{n=1}^{D}$ of C
$J \leftarrow \emptyset$
for $n = 1, 2, \ldots, N$ **do**
  $J \leftarrow J \cup \{n\}$ with prob. $\frac{\lambda_n}{\lambda_n + 1}$
end for
$\hat{V} \leftarrow \{\frac{\hat{\boldsymbol{v}}_n}{\hat{\boldsymbol{v}}_n^\top C \hat{\boldsymbol{v}}_n}\}_{n \in J}$
$Y \leftarrow \emptyset$
while $|\hat{V}| > 0$ **do**
  Select $\boldsymbol{y}_i$ from $\mathcal{Y}$ with $\Pr(\boldsymbol{y}_i) = \frac{1}{|\hat{V}|} \sum_{\hat{\boldsymbol{v}} \in \hat{V}} ((B^\top \hat{\boldsymbol{v}})^\top \boldsymbol{e}_i)^2$ (Algorithm 10)
  $Y \leftarrow Y \cup \boldsymbol{y}_i$
  $\hat{V} \leftarrow \hat{V}_\perp$, where $\{B^\top \hat{\boldsymbol{v}} \mid \hat{\boldsymbol{v}} \in \hat{V}_\perp\}$ is an orthonormal basis for the subspace of V
    orthogonal to $\boldsymbol{e}_i$
end while
Output: Y

Our dataset consists of 73 still frames taken from various TV shows, each approximately 720 by 540 pixels in size [126].[1] As much as possible, the selected frames contain three or more people at similar scale, all facing the camera and without serious occlusions. Sample images from the dataset are shown in Figure 6.6. Each person in each image is annotated by hand; each of the four parts (head, torso, right arm, and left arm) is labeled with the pixel location of a reference point (e.g., the shoulder) and an orientation selected from among 24 discretized angles.

6.4.1 Factorized Model

There are approximately 75,000 possible values for each part, so there are about $4^{75{,}000}$ possible poses, and thus we cannot reasonably use a standard DPP for this problem. Instead, we build a factorized SDPP. Our factors are given by the standard pictorial structure model [50, 52],

[1] The images and code were obtained from `http://www.vision.grasp.upenn.edu/video`.

treating each pose as a two-level tree with the torso as the root and the head and arms as leaves. Each node (body part) has a singleton factor and each edge has a corresponding pairwise factor.

Our quality function derives from the model proposed by Sapp et al. [126], and is given by

$$q(\boldsymbol{y}) = \gamma \left(\prod_{r=1}^{R} q_r(y_r) \prod_{(r,r') \in E} q_{r,r'}(y_r, y_{r'}) \right)^{\beta}, \tag{6.58}$$

where E is the set of edges in the part tree, γ is a scale parameter that will control the expected number of poses in an SDPP sample, and β is a sharpness parameter that controls the dynamic range of the quality scores. We set the values of the hyperparameters γ and β using a held-out training set, as discussed below. The per-part quality scores $q_r(y_r)$ are provided by the customized part detectors trained by Sapp et al. [126] on similar images; they assign a value to every proposed location and orientation y_r of part r. The pairwise quality scores $q_{r,r'}(y_r, y_{r'})$ are defined according to a Gaussian "spring" that encourages, for example, the left arm to begin near the left shoulder of the torso. Full details of the model are provided in Sapp et al. [126].

In order to encourage the model not to choose overlapping poses, our diversity features reflect the locations of the constituent parts:

$$\phi(\boldsymbol{y}) = \sum_{r=1}^{R} \phi_r(y_r), \tag{6.59}$$

where each $\phi_r(y_r) \in \mathbb{R}^{32}$. There are no diversity features on the edge factors. The local features are based on a 8×4 grid of reference points $x_1, x_2, \ldots, x_{32}$ spaced evenly across the image; the l-th feature is

$$\phi_{rl}(y_r) \propto f_{\mathcal{N}}\left(\frac{\mathrm{dist}(y_r, x_l)}{\sigma} \right). \tag{6.60}$$

Here $f_{\mathcal{N}}$ is again the standard normal density function, and $\mathrm{dist}(y_r, x_l)$ is the Euclidean distance between the position of part r (ignoring orientation) and the reference point x_l. Poses that occupy the same part of the image will be near the same reference points, and thus their feature vectors ϕ will be more closely aligned. The parameter σ controls the

width of the kernel; larger values of σ make poses at a given distance appear more similar. We set σ on a held-out training set.

6.4.2 Methods

We compare samples from the SDPP defined above to those from two baseline methods. The first, which we call the independent model, draws poses independently according to the distribution obtained by normalizing the quality scores, which is essentially the graphical model used by Sapp et al. [126]. For this model the number of poses to be sampled must be supplied by the user, so to create a level playing field we choose the number of poses in an SDPP sample Y. Since this approach does not incorporate a notion of diversity (or any correlations between selected poses whatsoever), we expect that we will frequently see multiple poses that correspond to the same person.

The second baseline is a simple non-maximum suppression model [22], which incorporates a heuristic for encouraging diversity. The first pose is drawn from the normalized quality model in the same manner as for the independent method. Subsequent poses, however, are constrained so that they cannot overlap with the previously selected poses, but otherwise drawn according to the quality model. We consider poses overlapping if they cover any of the same pixels when rendered. Again, the number of poses must be provided as an argument, so we use the number of poses from a sample of the SDPP. While the non-max approach can no longer generate multiple poses in the same location, it achieves this using a hard, heuristic constraint. Thus, we might expect to perform poorly when multiple people actually do overlap in the image, for example if one stands behind the other.

The SDPP, on the other hand, generates samples that prefer, but do not require poses to be spatially diverse. That is, strong visual information in the image can override our prior assumptions about the separation of distinct poses. We split our data randomly into a training set of 13 images and a test set of 60 images. Using the training set, we select values for γ, β, and σ that optimize overall F_1 score at radius 100 (see below), as well as distinct optimal values of β for the baselines. (γ and σ are irrelevant for the baselines.) We then use each model to

sample ten sets of poses for each test image, for a total of 600 samples per model.

6.4.3 Results

For each sample from each of the three tested methods, we compute measures of precision and recall as well as an F_1 score. In our tests, precision is measured as the fraction of predicted parts for which both endpoints are within a given radius of the endpoints of an expert-labeled part of the same type (head, left arm, and so on). We report results across a range of radii. Correspondingly, recall is the fraction of expert-labeled parts with endpoints within a given radius of a predicted part of the same type. Since the SDPP model encourages diversity, we expect to see improvements in recall at the expense of precision, compared to the independent model. F_1 score is the harmonic mean of precision and recall. We compute all metrics separately for each sample, and then average the results across samples and images in the test set.

The results are shown in Figure 6.5(a). At tight tolerances, when the radius of acceptance is small, the SDPP performs comparably to the independent and non-max samples, perhaps because the quality scores are simply unreliable at this resolution, thus diversity has little effect. As the radius increases, however, the SDPP obtains better results, significantly outperforming both baselines. Figure 6.5(b) shows the

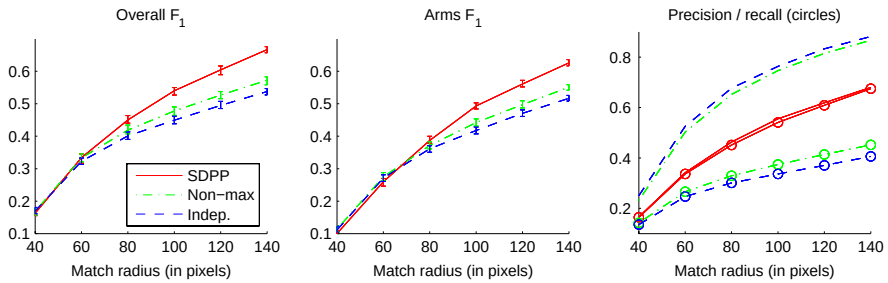


Fig. 6.5 Results for pose estimation. The horizontal axis gives the acceptance radius used to determine whether two parts are successfully matched. 95% confidence intervals are shown. (a) Overall F_1 scores. (b) Arm F_1 scores. (c) Overall precision/recall curves (recall is identified by circles).

curves for just the arm parts, which tend to be more difficult to locate accurately and exhibit greater variance in orientation. Figure 6.5(c) shows the precision/recall obtained by each model. As expected, the SDPP model achieves its improved F_1 score by increasing recall at the cost of precision.

For illustration, we show the SDPP sampling process for some sample images from the test set in Figure 6.6. The SDPP part marginals are visualized as a "cloud", where brighter colors correspond to higher probability. From left to right, we can see how the marginals change as poses are selected during the main loop of Algorithm 11. As we saw for simple synthetic examples in Figure 2.5(a), the SDPP discounts but does not entirely preclude poses that are similar to those already selected.

6.5 Random Projections for SDPPs

It is quite remarkable that we can perform polynomial-time inference for SDPPs given their extreme combinatorial nature. Even so, in some cases the algorithms presented in Section 6.3 may not be fast enough. Eigendecomposing the dual representation C, for instance, requires $O(D^3)$ time, while normalization, marginalization, and sampling, even when an eigendecomposition has been precomputed, scale quadratically in D, both in terms of time and memory. In practice, this limits us to relatively low-dimensional diversity features ϕ; for example, in our pose estimation experiments we built ϕ from a fairly coarse grid of 32 points mainly for reasons of efficiency. As we move to textual data, this will become an even bigger problem, since there is no natural low-dimensional analog for feature vectors based on, say, word occurrences. In the following section we will see data where natural vectors ϕ have dimension $D \approx 30{,}000$; without dimensionality reduction, storing even a single belief propagation message would require over 200 terabytes of memory.

To address this problem, we will make use of the random projection technique described in Section 3.4, reducing the dimension of the diversity features without sacrificing the accuracy of the model. Because Theorem 3.3 depends on a cardinality condition, we will focus on

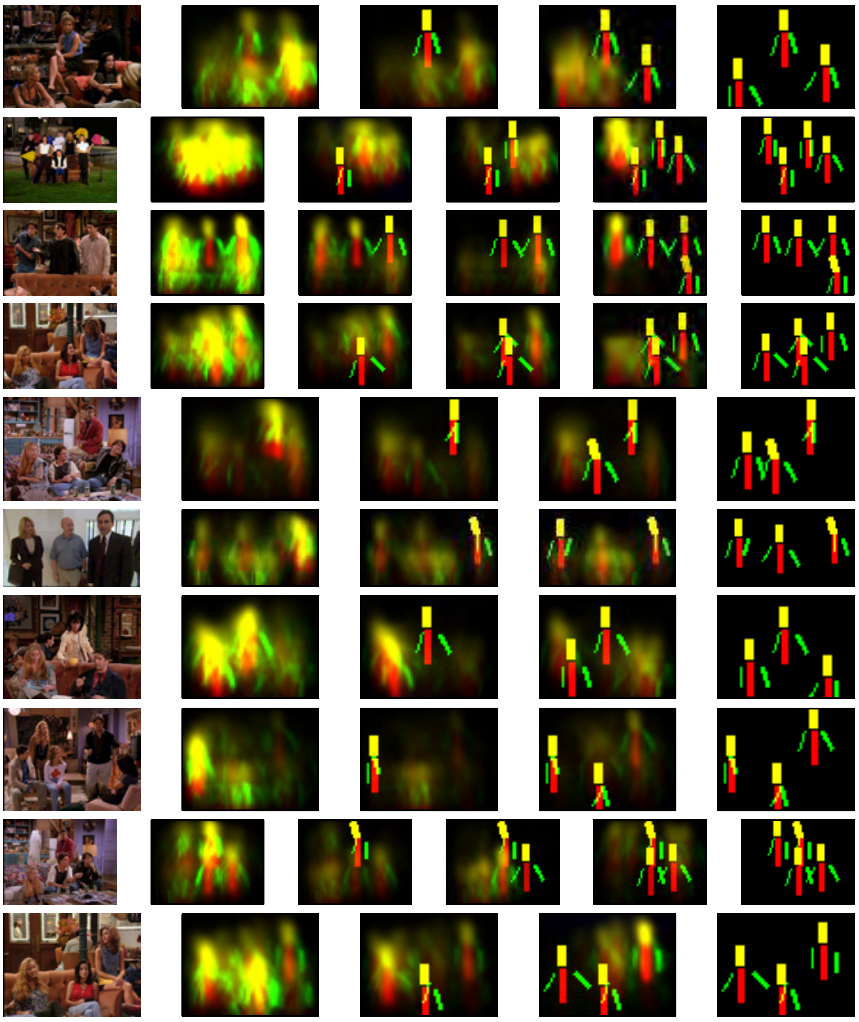


Fig. 6.6 Structured marginals for the pose estimation task, visualized as clouds, on successive steps of the sampling algorithm. Already selected poses are superimposed. Input images are shown on the left.

k-SDPPs. As described in Section 5, a k-DPP is simply a DPP conditioned on the cardinality of the modeled subset $\boldsymbol{Y}$:

$$\mathcal{P}^k(Y) = \frac{\left(\prod_{\boldsymbol{y}\in Y} q^2(\boldsymbol{y})\right) \det(\phi(Y)^\top \phi(Y))}{\sum_{|Y'|=k} \left(\prod_{\boldsymbol{y}\in Y} q^2(\boldsymbol{y})\right) \det(\phi(Y)^\top \phi(Y))}, \tag{6.61}$$

where $\phi(Y)$ denotes the $D \times |Y|$ matrix formed from columns $\phi(\boldsymbol{y})$ for $\boldsymbol{y} \in Y$. When q and ϕ factor over parts of a structure, as in Section 6.1, we will refer to this distribution as a k-SDPP. We note in passing that the algorithms for normalization and sampling in Section 5 apply equally well to k-SDPPs, since they depend mainly on the eigenvalues of L, which we can obtain from C.

Recall that Theorem 3.3 requires projection dimension

$$d = O(\max\{k/\epsilon, (\log(1/\delta) + \log N)/\epsilon^2\}). \tag{6.62}$$

In the structured setting, $N = M^R$, thus d must be logarithmic in the number of labels and linear in the number of parts. Under this condition, we have, with probability at least $1 - \delta$,

$$\|\mathcal{P}^k - \tilde{\mathcal{P}}^k\|_1 \leq e^{6k\epsilon} - 1, \tag{6.63}$$

where $\tilde{\mathcal{P}}^k(Y)$ is the projected k-SDPP.

6.5.1 Toy Example: Geographical Paths

In order to empirically study the effects of random projections, we test them on a simple toy application where D is small enough that the exact model is tractable. The goal is to identify diverse, high-quality sets of travel routes between U.S. cities, where diversity is with respect to geographical location, and quality is optimized by short paths visiting the most populous or most touristy cities. Such sets of paths could be used, for example, by a politician to plan campaign routes, or by a traveler organizing a series of vacations.

We model the problem as a k-SDPP over path structures having $R = 4$ parts, where each part is a stop along the path and can take any of $M = 200$ city values. The quality and diversity functions are factored, with a singleton factor for every individual stop and pairwise factors for consecutive pairs of stops. The quality of a singleton factor is based on the Google hit count for the assigned city, so that paths stopping in popular cities are preferred. The quality of a pair of consecutive stops is based on the distance between the assigned cities, so that short paths are preferred. In order to disallow paths that travel back and forth between the same cities, we augment the stop assignments to

include arrival direction, and assign a quality score of zero to paths that return in the direction from which they came. The diversity features are only defined on the singleton factors; for a given city assignment y_r, $\phi_r(y_r)$ is just the vector of inverse distances between y_r and all of the 200 cities. As a result, paths passing through the same or nearby cities appear similar, and the model prefers paths that travel through different regions of the country. We have $D = 200$.

Figure 6.7 shows sets of paths sampled from the k-SDPP for various values of k. For $k = 2$, the model tends to choose one path along the east coast and another along the west coast. As k increases, a variety of configurations emerge; however, they continue to emphasize popular cities and the different paths remain geographically diverse.

We can now investigate the effects of random projections on this model. Figure 6.8 shows the L_1 variational distance between the original model and the projected model (estimated by sampling), as well as the memory required to sample a set of paths for a variety of projection dimensions d. As predicted by Theorem 3.3, only a relatively small number of projection dimensions are needed to obtain a close approximation to the original model. Past $d \approx 25$, the rate of improvement due to increased dimension falls off dramatically; meanwhile, the required memory and running time start to become significant. Figure 6.8 suggests that aggressive use of random projections, like

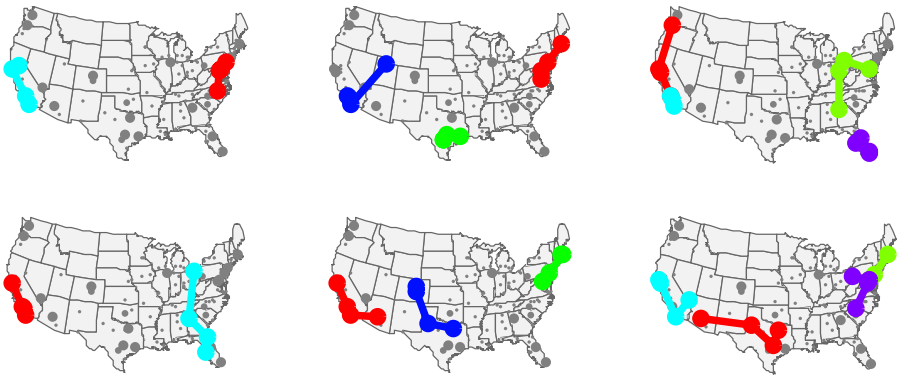


Fig. 6.7 Each column shows two samples drawn from a k-SDPP; from left to right, $k = 2, 3, 4$. Circle size corresponds to city quality.

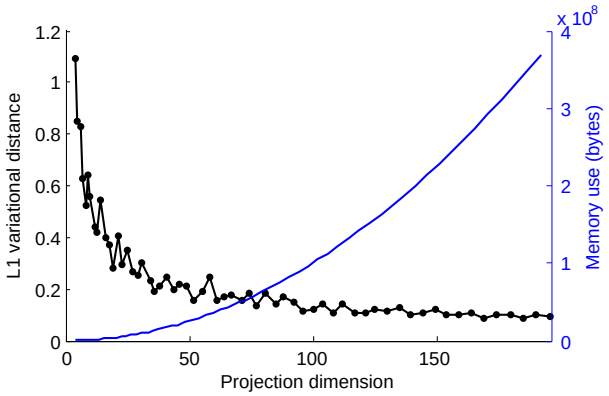


Fig. 6.8 The effect of random projections. In black, on the left, we estimate the L_1 variational distance between the original and projected models. In blue, on the right, we plot the memory required for sampling, which is also proportional to running time.

those we employ in the following section, is not only theoretically but also empirically justified.

6.6 Experiments: Threading Graphs

In this section we put together many of the techniques introduced in this monograph in order to complete a novel task that we refer to as *graph threading* [57]. The goal is to extract from a large directed graph a set of diverse, salient *threads*, or singly connected chains of nodes. Depending on the construction of the graph, such threads can have various semantics. For example, given a corpus of academic literature, high-quality threads in the citation graph might correspond to chronological chains of important papers, each building on the work of the last. Thus, graph threading could be used to identify a set of significant lines of research. Or, given a collection of news articles from a certain time period, where each article is a node connected to previous, related articles, we might want to display the most significant news stories from that period, and for each story provide a thread that contains a timeline of its major events. We experiment on data from these two domains in the following sections. Other possibilities might include discovering trends on social media sites, for example, where users can post image or video responses to each other, or mining blog entries for

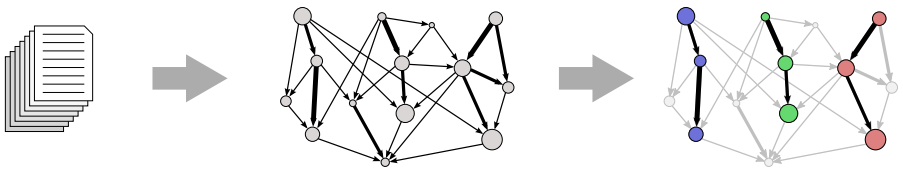


Fig. 6.9 An illustration of graph threading applied to a document collection. We first build a graph from the collection, using measures of importance and relatedness to weight nodes (documents) and build edges (relationships). Then, from this graph, we extract a diverse, salient set of threads to represent the collection.

important conversations through trackback links. Figure 6.9 gives an overview of the graph threading task for document collections.

Generally speaking, graph threading offers a means of gleaning insights from collections of interrelated objects — for instance, people, documents, images, events, locations, and so on — that are too large and noisy for manual examination. In contrast to tools like search, which require the user to specify a query based on prior knowledge, a set of threads provide an immediate, concise, high-level summary of the collection, not just identifying a set of important objects but also conveying the relationships between them. As the availability of such datasets continues to grow, this kind of automated analysis will be key in helping us to efficiently and effectively navigate and understand the information they contain.

6.6.1 Related Work

Research from to the Topic Detection and Tracking (TDT) program [154] has led to useful methods for tasks like link detection, topic detection, and topic tracking that can be seen as subroutines for graph threading on text collections. 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 [11, 91, 105]. In particular, Blei and Lafferty [11] proposed dynamic topic models (DTMs), which, given a division of text documents into time slices, attempt to fit a generative model where topics evolve over time, and documents are drawn from the topics available

at the time slice during which they were published. The evolving topics found by a DTM can be seen as threads of a sort, but in contrast to graph threading they are not composed of actual items in the dataset (in this case, documents). In Section 6.6.4 we will return to this distinction when we compare k-SDPP threading with a DTM baseline.

The information retrieval community has produced other methods for extracting temporal information from document collections. Swan and Jensen [141] proposed a system for finding temporally clustered named entities in news text and presenting them on a timeline. Allan et al. [2] introduced the task of *temporal summarization*, which takes as input a stream of news articles related to a particular topic, and then seeks to extract sentences describing important events as they occur. Yan et al. [155] 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.

Some prior work has also focused more directly on threading. Shahaf and Guestrin [128] and Chieu and Lee [27] proposed methods for selecting individual threads, while Shahaf et al. [129] recently 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, in doing so, generate a map of events and topics. However, these approaches assume some prior knowledge about content. Shahaf and Guestrin [128], for example, assume that the thread endpoints are specified, and Chieu and Lee [27] require a set of query words. Likewise, because they build metro maps individually, Shahaf et al. [129] implicitly assume that the collection is filtered to a single topic, perhaps from a user query. These inputs make it possible to quickly pare down the document graph. In contrast, we will apply graph threading to very large graphs, and consider all possible threads.

6.6.2 Setup

In order to be as useful as possible, the threads we extract from a data graph need to be both high quality, reflecting the most important parts

of the collection, and diverse, so that they cover distinct aspects of the data. In addition, we would like to be able to directly control both the length and the number of threads that we return, since different contexts might necessitate different settings. Finally, to be practical our method must be efficient in both time and memory use. k-SDPPs with random projections allow us to simultaneously achieve all of these goals.

Given a directed graph on M vertices with edge set E and a real-valued weight function $w(\cdot)$ on nodes and edges, define the weight of a thread $\boldsymbol{y} = (y_1, y_2, \ldots, y_R)$, $(y_r, y_{r+1}) \in E$ by

$$w(\boldsymbol{y}) = \sum_{r=1}^{R} w(y_r) + \sum_{r=2}^{R} w(y_{r-1}, y_r). \tag{6.64}$$

We can use w to define a simple log-linear quality model for our k-SDPP:

$$q(\boldsymbol{y}) = \exp(\beta w(\boldsymbol{y})) \tag{6.65}$$

$$= \left(\prod_{r=1}^{R} \exp(w(y_r)) \prod_{r=2}^{R} \exp(w(y_{r-1}, y_r)) \right)^{\beta}, \tag{6.66}$$

where β is a hyperparameter controlling the dynamic range of the quality scores. We fix the value of β on a validation set in our experiments.

Likewise, let ϕ be a feature function from nodes in the graph to $\mathbb{R}^D$, then the diversity feature function on threads is

$$\phi(\boldsymbol{y}) = \sum_{r=1}^{R} \phi(y_r). \tag{6.67}$$

In some cases it might also be convenient to have diversity features on edges of the graph as well as nodes. If so, they can be accommodated without much difficulty; however, for simplicity we proceed with the setup above.

We assume that R, k, and the projection dimension d are provided; the first two depend on application context, and the third, as discussed in Section 6.5, is a trade-off between computational efficiency and faithfulness to the original model. To generate diverse thread samples, we

first project the diversity features ϕ by a random $d \times D$ matrix G whose entries are drawn independently and identically from $\mathcal{N}(0, \frac{1}{d})$. We then apply second-order message passing to compute the dual representation C, as in Section 6.3.1. After eigendecomposing C, which is only $d \times d$ due to the projection, we can run the first phase of the k-DPP sampling algorithm from Section 5.2.2 to choose a set $\hat{V}$ of eigenvectors, and finally complete the SDPP sampling algorithm in Section 6.3.2 to obtain a set of k threads Y. We now apply this model to two datasets; one is a citation graph of computer science papers, and the other is a large corpus of news text.

6.6.3 Academic Citation Data

The Cora dataset comprises a large collection of approximately 200,000 academic papers on computer science topics, including citation information [102]. We construct a directed graph with papers as nodes and citations as edges, and then remove papers with missing metadata or zero outgoing citations, leaving us with 28,155 papers. The average out-degree is 3.26 citations per paper, and 0.011% of the total possible edges are present in the graph.

To obtain useful threads, we set edge weights to reflect the degree of textual similarity between the citing and the cited paper, and node weights to correspond with a measure of paper "importance". Specifically, the weight of edge (a, b) is given by the cosine similarity metric, which for two documents a and b is the dot product of their normalized tf–idf vectors, as defined in Section 4.2.1:

$$\text{cos-sim}(a,b) = \frac{\sum_{w \in W} \text{tf}_a(w)\text{tf}_b(w)idf^2(w)}{\sqrt{\sum_{w \in W} \text{tf}_a^2(w)idf^2(w)}\sqrt{\sum_{w \in W} \text{tf}_b^2(w)idf^2(w)}}, \tag{6.68}$$

Here W is a subset of the words found in the documents. We select W by filtering according to document frequency; that is, we remove words that are too common, appearing in more than 10% of papers, or too rare, appearing in only one paper. After filtering, there are 50,912 unique words.

The node weights are given by the LexRank score of each paper [43]. The LexRank score is the stationary distribution of the thresholded, binarized, row-normalized matrix of cosine similarities, plus a damping term, which we fix to 0.15. LexRank is a measure of centrality, so papers that are closely related to many other papers will receive a higher score.

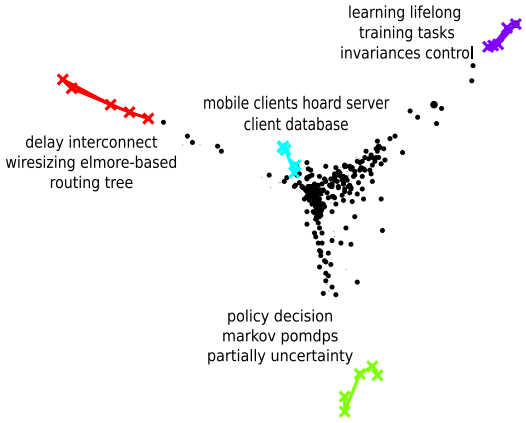
Finally, we design the diversity feature function ϕ to encourage topical diversity. Here we apply cosine similarity again, representing a document by the 1,000 documents to which it is most similar. This results in binary ϕ of dimension $D = M = 28,155$ with exactly 1,000 nonzeros; $\phi_l(y_r) = 1$ implies that l is one of the 1,000 most similar documents to y_r. Correspondingly, the dot product between the diversity features of two documents is proportional to the fraction of top-1,000 documents they have in common. In order to make k-SDPP inference efficient, we project ϕ down to $d = 50$ dimensions.

Figure 6.10 illustrates the behavior of the model when we set $k = 4$ and $R = 5$. Samples from the model, like the one presented in the figure, not only offer some immediate intuition about the types of papers contained in the collection but also, upon examining individual threads, provide a succinct illustration of the content and development of each area. Furthermore, the sampled threads cover distinct topics, standing apart visually in Figure 6.10 and exhibiting diverse salient terms.

6.6.4 News Articles

Our news dataset comprises over 200,000 articles from the New York Times, collected from 2005 to 2007 as part of the English Gigaword corpus [60]. We split the articles into six groups, with six months' worth of articles in each group. 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 filter the results by discarding articles more than two standard deviations longer than the mean article, articles less than 400 words, and articles whose fraction of nonalphabetic words is more than two standard deviations above the mean. On average, for each six-month period we are left with 34,504 articles.

For each time period, we generate a graph with articles as nodes. As for the citations dataset, we use cosine similarity with define edge



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

Fig. 6.10 Sampled threads 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, and then highlight the thread selections in color. Displayed beside each thread are the words in the thread with highest tf–idf score. Below, we show the titles of the papers in two of the threads.

weights. The subset of words W used to compute cosine similarity contains all words that appear in at least 20 articles and at most 15% of the articles. Across the six time periods, this results in an average of 36,356 unique words. We include in our graph only those edges with cosine similarity of at least 0.1; furthermore, we require that edges

go forward in time to enforce 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. As before, we use LexRank for node weights, and the top-1000 similar documents to define a binary feature function ϕ. We add a constant feature ρ to ϕ, which controls the overall degree of repulsion; large values of ρ make all documents more similar to one another. We set ρ and the quality model hyperparameters to maximize a cosine similarity evaluation metric (see Section 6.6.4), using the data from the first half of 2005 as a development set. Finally, we randomly project the diversity features from $D \approx 34,500$ 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 model.

Graph visualizations In order to convey the scale and content of the graphs built from news data, we provide some high-resolution renderings. Figure 6.11 shows the graph neighborhood of a single article node from our development set. Each node represents an article and is annotated with the corresponding headline; the size of each node reflects its weight, as does the thickness of an edge. The horizontal position of a node corresponds to the time at which the article was published, from left to right; the vertical positions are optimized for readability. In the digital version of this monograph, Figure 6.11 can be zoomed in order to read the headlines; in hardcopy, however, it is likely to be illegible. As an alternative, an online, zoomable version of the figure is available at http://zoom.it/GUCR.

Visualizing the entire graph is quite challenging since it contains tens of thousands of nodes and millions of edges; placing such a figure in the monograph would be impractical since the computational demands of rendering it and the zooming depth required to explore it would exceed the abilities of modern document viewers. Instead, we provide an online, zoomable version based upon a high-resolution (540 megapixel) rendering, available at http://zoom.it/jOKV. Even at this level of detail, only 1% of the edges are displayed; otherwise they become visually indistinct. As in Figure 6.11, each node represents an article and is sized according to its weight and overlaid with its

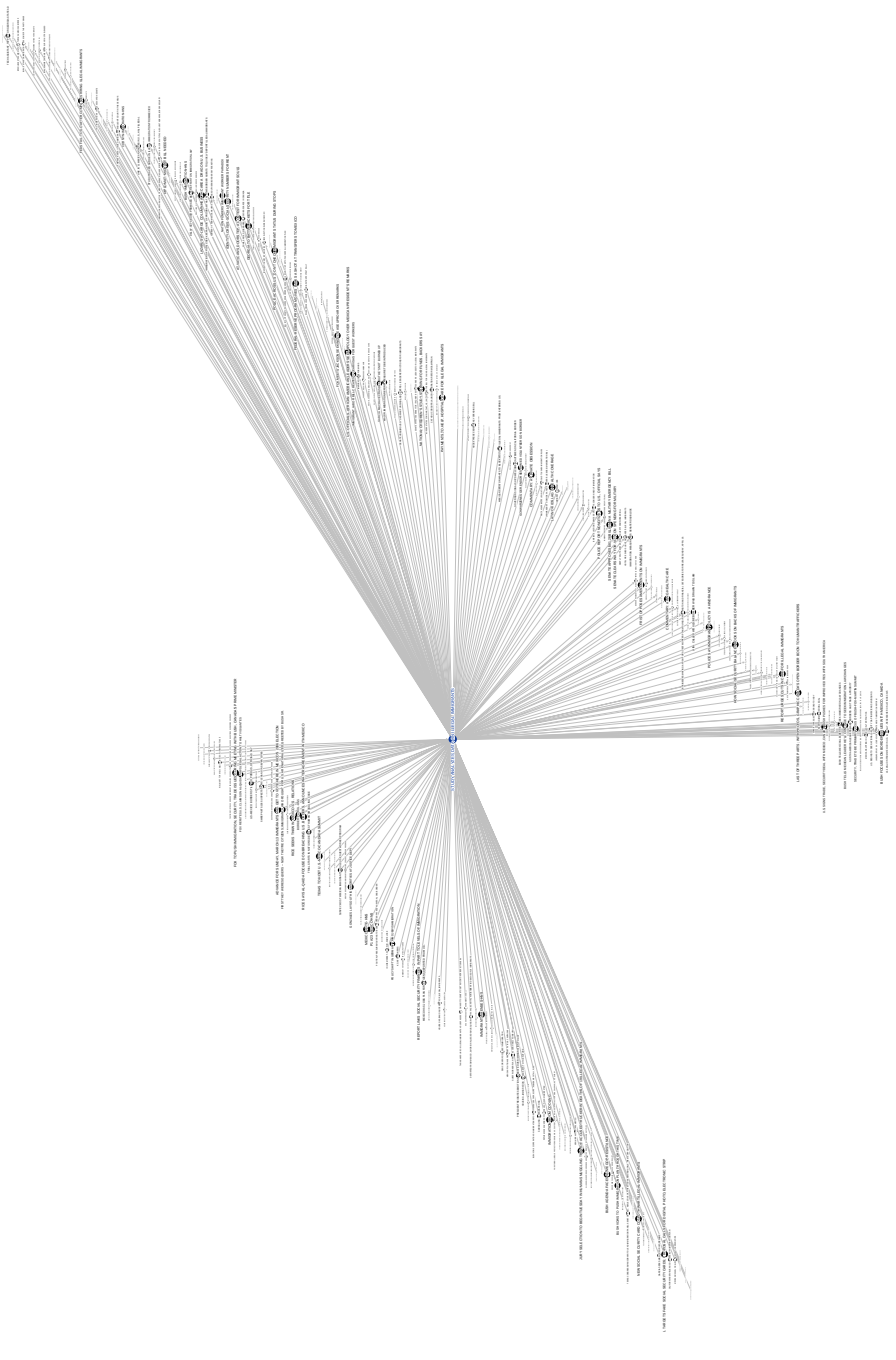


Fig. 6.11 Visualization of a single article node and all of its neighboring article nodes.

headline. The horizontal position corresponds to time, ranging from January 2005 (on the left) to June 2005 (on the right). The vertical positions are determined by similarity with a set of threads sampled from the k-SDPP, which are rendered in color.

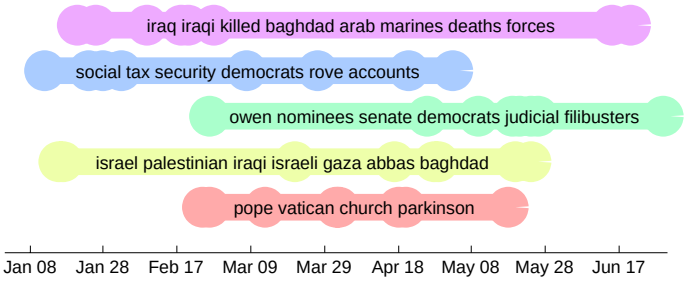
Baselines We will compare the k-SDPP model to two natural baselines.

k-means baseline. A simple method for this task is to split each six-month period of articles into R equal-sized time slices, and then apply k-means clustering to each slice, using cosine similarity at the clustering metric. We can then select the most central article from each cluster to form the basis of a set of threads. The k articles chosen from time slice r are matched one-to-one with those from slice $r-1$ by computing the pairing that maximizes the average cosine similarity of the pairs — that is, the coherence of the threads. Repeating this process for all r yields a set of k threads of length R, where no two threads will contain the same article. However, because clustering is performed independently for each time slice, it is likely that the threads will sometimes exhibit discontinuities when the articles chosen at successive time slices do not naturally align.

DTM baseline. A natural extension, then, is the dynamic topic model (DTM) of Blei and Lafferty [11], which explicitly attempts to find topics that are smooth through time. We use publicly available code[2] to fit DTMs with the number of topics set to k and with the data split into R equal time slices. We set the hyperparameters to maximize the cosine similarity metric (see Section 6.6.4) on our development set. 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 6.12 shows some of the threads sampled randomly from the k-SDPP for our development set, and Figure 6.13 shows the same for threads produced by the DTM baseline. An obvious distinction is that topic model threads always span nearly the entire time period, selecting one article per time slice as required by the form of the model, while the

[2] http://code.google.com/p/princeton-statistical-learning/



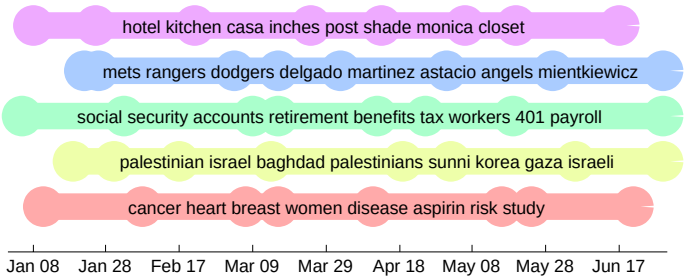
Thread: **pope vatican church parkinson**

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

Fig. 6.12 A set of five news threads randomly sampled from a 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 a single thread are listed.

DPP can select threads covering only the relevant span. Furthermore, the headlines in the figures suggest that the k-SDPP produces more tightly focused, narrative threads due to its use of the data graph, while the DTM threads, though topically related, tend not to describe a single continuous news story. This distinction, which results from the fact that topic models are not designed with threading in mind, and so do not take advantage of the explicit relation information given by the graph, means that k-SDPP threads often form a significantly more coherent representation of the news collection.

Comparison to human summaries We provide a quantitative evaluation of the threads generated by our baselines and sampled from the k-SDPP by comparing them with a set of human-generated news summaries. The human summaries are not threaded; they are flat, approximately daily news summaries found in the Agence France-Presse portion of the Gigaword corpus, distinguished by their "multi" type tag. The summaries generally cover world news, which is only a



Thread: **cancer heart breast women disease aspirin risk study**

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

Fig. 6.13 A set of five news threads generated by the 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 a single thread are listed.

subset of the contents of our dataset. Nonetheless, they allow us to provide an extrinsic evaluation for this novel task without generating gold standard timelines manually which is a difficult task, given the size of the corpus. We compute four metrics:

- **Cosine similarity.** We concatenate the human summaries over each six-month period to obtain a target tf–idf vector, concatenate the set of threads to be evaluated to obtain a predicted tf–idf vector, and then compute the cosine similarity (in percent) between the target and predicted vectors. All hyperparameters are chosen to optimize this metric on a validation set.
- **ROUGE-1, 2, and SU4.** As described in Section 4.2.1, ROUGE is an automatic evaluation metric for text summarization based on n-gram overlap statistics [93]. We report three standard variants.

Table 6.1. Similarity of automatically generated timelines to human summaries. Bold entries are significantly higher than others in the column at 99% confidence, verified using bootstrapping.

		ROUGE-1		ROUGE-2		ROUGE-SU4	
System	Cos-sim	F	Prec/Rec	F	Prec/Rec	F	Prec/Rec
k-means	29.9	16.5	17.3/15.8	0.695	0.73/0.67	3.76	3.94/3.60
DTM	27.0	14.7	15.5/14.0	0.750	0.81/0.70	3.44	3.63/3.28
k-SDPP	**33.2**	**17.2**	**17.7/16.7**	**0.892**	**0.92/0.87**	**3.98**	**4.11/3.87**

Table 6.1 shows the results of these comparisons, averaged over all six half-year intervals. Under each metric, the k-SDPP produces threads that more closely resemble human summaries.

Mechanical Turk evaluation An important distinction between the baselines and the k-SDPP is that the former are *topic*-oriented, choosing articles that relate to broad subject areas, while the k-SDPP approach is *story*-oriented, chaining together articles with direct individual relationships. An example of this distinction can be seen in Figures 6.12 and 6.13.

To obtain a large-scale evaluation of this type of thread coherence, we employ Mechanical Turk, on online marketplace for inexpensively and efficiently completing tasks requiring human judgment. We asked Turkers to read the headlines and first few sentences of each article in a timeline and 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 the left column of Table 6.2; the k-SDPP timelines are rated as significantly more coherent, while k-means does poorly since it has no way to ensure that clusters are similar between time slices.

In addition, we asked Turkers to evaluate threads implicitly by performing a second task. (This also had the side benefit of ensuring that Turkers were engaged in the rating task and did not enter random decisions.) We displayed timelines into which two additional "interloper" articles selected at random had been inserted, and asked users to remove the two articles that they thought should be removed

Table 6.2. 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.

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

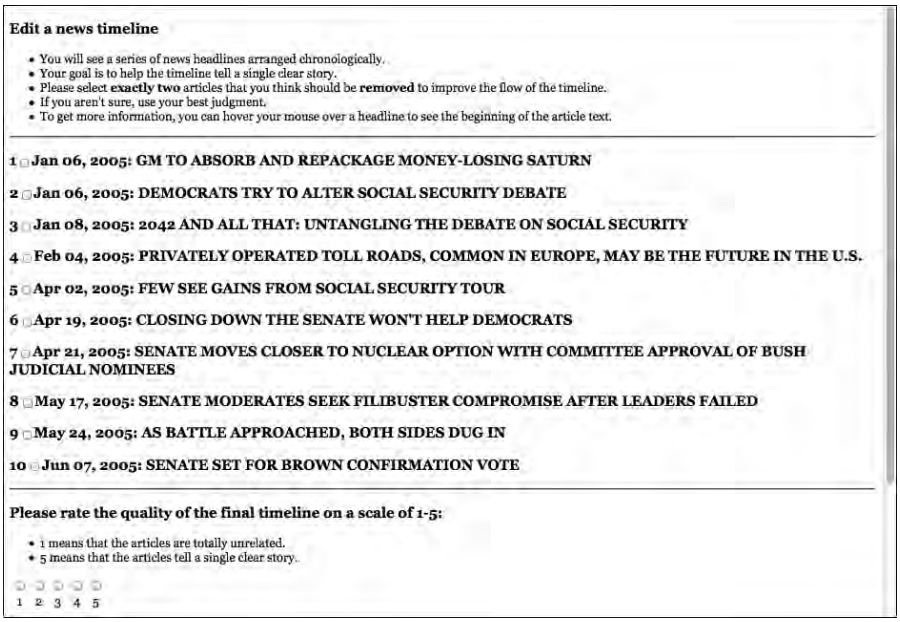


Fig. 6.14 A screenshot of the Mechanical Turk task presented to annotators.

to improve the flow of the timeline. A screenshot of the task is provided in Figure 6.14. Intuitively, the true interlopers should be selected more often when the original timeline is coherent. The average number of interloper articles correctly identified is shown in the right column of Table 6.2.

Runtime Finally, assuming that tf–idf and feature values have been computed in advance (this process requires approximately 160 seconds), we report in Table 6.3 the time required to produce a set of threads

Table 6.3. Running time for the tested methods.

System	Runtime (s)
k-means	625.63
DTM	19,433.80
k-SDPP	**252.38**

on the development set. This measurement includes clustering for the k-means baseline, model fitting for the DTM baseline, and random projections, computation of the covariance matrix, and sampling for the k-SDPP. The tests were run on a machine with eight Intel Xeon E5450 cores and 32G of memory. Thanks to the use of random projections, the k-SDPP is not only the most faithful to human news summaries, but also the fastest by a large margin.

7

Conclusion

We believe that DPPs offer exciting new possibilities for a wide range of practical applications. Unlike heuristic diversification techniques, DPPs provide coherent probabilistic semantics, and yet they do not suffer from the computational issues that plague existing models when negative correlations arise. Before concluding, we briefly mention two open technical questions, as well as some possible directions for future research.

7.1 Open Question: Concavity of Entropy

The Shannon entropy of the DPP with marginal kernel K is given by

$$H(K) = -\sum_{Y \subseteq \mathcal{Y}} \mathcal{P}(Y) \log \mathcal{P}(Y). \tag{7.1}$$

Conjecture 1 (Lyons [97]). $H(K)$ is concave in K.

While numerical simulation strongly suggests that the conjecture is true, to our knowledge no proof currently exists.

7.2 Open Question: Higher-order Sums

In order to calculate, for example, the Hellinger distance between a pair of DPPs, it would be useful to be able to compute quantities of the form

$$\sum_{Y \subseteq \mathcal{Y}} \det(L_Y)^p \tag{7.2}$$

for $p > 1$. To our knowledge it is not currently known whether it is possible to compute these quantities efficiently.

7.3 Research Directions

A variety of interesting machine learning questions remain for future research.

- Would DPPs based on Hermitian or asymmetric kernels offer worthwhile modeling advantages?
- Is there a simple characterization of the conditional independence relations encoded by a DPP?
- Can we perform DPP inference under more complicated constraints on allowable sets? (For instance, if the items correspond to edges in a graph, we might only consider sets that comprise a valid matching.)
- How can we learn the similarity kernel for a DPP (in addition to the quality model) from labeled training data?
- How can we efficiently (perhaps approximately) work with SDPPs over loopy factor graphs?
- Can SDPPs be used to diversify n-best lists and improve reranking performance, for instance in parsing or machine translation?

References

[1] A. Abdelbar and S. Hedetniemi, "Approximating maps for belief networks is NP-hard and other theorems," *Artificial Intelligence*, vol. 102, no. 1, pp. 21–38, 1998.

[2] J. Allan, R. Gupta, and V. Khandelwal, "Temporal Summaries of New Topics," in *Proceedings of the Annual Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[3] A. J. Baddeley and M. N. M. Van Lieshout, "Area-interaction point processes," *Annals of the Institute of Statistical Mathematics*, vol. 47, no. 4, pp. 601–619, 1995.

[4] F. B. Baker and M. R. Harwell, "Computing elementary symmetric functions and their derivatives: A didactic," *Applied Psychological Measurement*, vol. 20, no. 2, p. 169, 1996.

[5] A. I. Barvinok, "Computational complexity of immanents and representations of the full linear group," *Functional Analysis and Its Applications*, vol. 24, no. 2, pp. 144–145, 1990.

[6] K. Berthelsen and J. Møller, "Bayesian analysis of Markov point processes," *Case Studies in Spatial Point Process Modeling*, pp. 85–97, 2006.

[7] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[8] J. Besag, "Some methods of statistical analysis for spatial data," *Bulletin of the International Statistical Institute*, vol. 47, no. 2, pp. 77–92, 1977.

[9] J. Besag and P. Green, "Spatial statistics and Bayesian computation," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 25–37, 1993.

[10] J. Besag, R. Milne, and S. Zachary, "Point process limits of lattice processes," *Journal of Applied Probability*, pp. 210–216, 1982.

[11] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 113–120, 2006.
[12] A. Borodin, "Determinantal point processes," URL http://arxiv.org/abs/0911.1153, 2009.
[13] A. Borodin, P. Diaconis, and J. Fulman, "On adding a list of numbers (and other one-dependent determinantal processes)," *American Mathematical Society*, vol. 47, no. 4, pp. 639–670, 2010.
[14] A. Borodin and G. Olshanski, "Distributions on partitions, point processes, and the hypergeometric kernel," *Communications in Mathematical Physics*, vol. 211, no. 2, pp. 335–358, 2000.
[15] A. Borodin and E. Rains, "Eynard-mehta theorem, schur process, and their pfaffian analogs," *Journal of Statistical Physics*, vol. 121, pp. 291–317, 2005. ISSN 0022-4715. 10.1007/s10955-005-7583-z.
[16] A. Borodin and A. Soshnikov, "Janossy densities. i. determinantal ensembles," *Journal of Statistical Physics*, vol. 113, no. 3, pp. 595–610, 2003.
[17] E. Boros and P. L. Hammer, "Pseudo-Boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1-3, pp. 155–225, 2002.
[18] P. Bratley and B. Fox, "Algorithm 659: Implementing Sobol's quasirandom sequence generator," *ACM Transactions on Mathematical Software (TOMS)*, no. 1, pp. 88–100, 1988.
[19] J. L. Brylinski and R. Brylinski, "Complexity and completeness of immanants," Arxiv preprint cs/0301024, 2003.
[20] P. Bürgisser, "The computational complexity of immanants," *SIAM Journal on Computing*, vol. 30, p. 1023, 2000.
[21] R. Burton and R. Pemantle, "Local characteristics, entropy and limit theorems for spanning trees and domino tilings via transfer-impedances," *The Annals of Probability*, pp. 1329–1371, 1993.
[22] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 679–698, 1986.
[23] J. Carbonell and J. Goldstein, "The use of MMR, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the Annual Conference on Research and Development in Information Retrieval (SIGIR)*, 1998.
[24] A. Cayley, "On the theory of determinants," *Transaction of the Cambridge Philosophical Society*, vol. 8, no. 1843, pp. 1–16, 1843.
[25] C. Chekuri, J. Vondrák, and R. Zenklusen, "Submodular function maximization via the multilinear relaxation and contention resolution schemes," Arxiv preprint arXiv:1105.4593, 2011.
[26] H. Chen and D. R. Karger, "Less is more: Probabilistic models for retrieving fewer relevant documents," in *Proceedings of the Annual Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 429–436, 2006.
[27] H. Chieu and Y. Lee, "Query based event extraction along a timeline," in *Proceedings of the Annual Conference on Research and Development in Information Retrieval (SIGIR)*, 2004.

[28] A. Çivril and M. Magdon-Ismail, "On selecting a maximum volume sub-matrix of a matrix and related problems," *Theoretical Computer Science*, vol. 410, no. 47–49, pp. 4801–4811, 2009.

[29] J. M. Conroy, J. Schlesinger, J. Goldstein, and D. P. Oleary, "Left-brain/right-brain multi-document summarization," in *Proceedings of the Document Understanding Conference (DUC)*, 2004.

[30] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, vol. 42, no. 2–3, pp. 393–405, 1990.

[31] P. Dagum and M. Luby, "Approximating probabilistic inference in Bayesian belief networks is NP-hard," *Artificial Intelligence*, vol. 60, no. 1, pp. 141–153, 1993.

[32] D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Springer, 2003.

[33] D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes: General Theory and Structure*, vol. 2. Springer Verlag, 2008.

[34] H. T. Dang, "Overview of DUC 2005," in *Procedings of the Document Understanding Conference (DUC)*, 2005.

[35] A. Deshpande and L. Rademacher, "Efficient volume sampling for row/column subset selection," in *2010 IEEE Annual Symposium on Foundations of Computer Science*, pp. 329–338, 2010.

[36] P. Diaconis, "Patterns in eigenvalues: The 70th Josiah Willard Gibbs lecture," *Bulletin-American Mathematical Society*, vol. 40, no. 2, pp. 155–178, 2003.

[37] P. Diaconis and S. N. Evans, "Immanants and finite point processes," *Journal of Combinatorial Theory, Series A*, vol. 91, no. 1-2, pp. 305–321, 2000.

[38] P. J. Diggle, T. Fiksel, P. Grabarnik, Y. Ogata, D. Stoyan, and M. Tanemura, "On parameter estimation for pairwise interaction point processes," *International Statistical Review/Revue Internationale de Statistique*, pp. 99–117, 1994.

[39] P. J. Diggle, D. J. Gates, and A. Stibbard, "A nonparametric estimator for pairwise-interaction point processes," *Biometrika*, vol. 74, no. 4, pp. 763–770, 1987.

[40] F. J. Dyson, "Statistical theory of the energy levels of complex systems. i," *Journal of Mathematical Physics*, vol. 3, pp. 140–156, 1962.

[41] F. J. Dyson, "Statistical theory of the energy levels of complex systems. ii," *Journal of Mathematical Physics*, vol. 3, no. 157–165, 1962.

[42] F. J. Dyson, "Statistical theory of the energy levels of complex systems. iii," *Journal of Mathematical Physics*, vol. 3, pp. 166–175, 1962.

[43] G. Erkan and D. R. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, no. 1, pp. 457–479, 2004. ISSN 1076-9757.

[44] S. N. Evans and A. Gottlieb, "Hyperdeterminantal point processes," *Metrika*, vol. 69, no. 2, pp. 85–99, 2009.

[45] J. Feder, "Random sequential adsorption," *Journal of Theoretical Biology*, vol. 87, no. 2, pp. 237–254, 1980.

[46] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.
[47] U. Feige, V. S. Mirrokni, and J. Vondrak, "Maximizing non-monotone submodular functions," in *Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pp. 461–471, 2007.
[48] M. Feldman, J. Naor, and R. Schwartz, "Nonmonotone submodular maximization via a structural continuous greedy algorithm," *Automata, Languages and Programming*, pp. 342–353, 2011.
[49] M. Feldman, J. S. Naor, and R. Schwartz, "A unified continuous greedy algorithm for submodular maximization," in *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 570–579, 2011.
[50] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005. ISSN 0920-5691.
[51] L. Finegold and J. T. Donnell, "Maximum density of random placing of membrane particles," *Nature*, 1979.
[52] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. 100, no. 22, 1973.
[53] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions — II," *Polyhedral Combinatorics*, pp. 73–87, 1978.
[54] I. M. Gel'fand, *Lectures on Linear Algebra*. Dover, 1989. ISBN 0486660826.
[55] P. E. Genest, G. Lapalme, and M. Yousfi-Monod, "Hextac: The creation of a manual extractive run," in *Proceedings of the Text Analysis Conference (TAC)*, Gaithersburg, Maryland, USA, 2010.
[56] S. O. Gharan and J. Vondrák, "Submodular maximization by simulated annealing," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1098–1116, 2011.
[57] J. Gillenwater, A. Kulesza, and B. Taskar, "Discovering diverse and salient threads in document collections," in *Proceedings of the 2012 Conference on Empirical Methods in Machine Learning*, 2012.
[58] J. Ginibre, "Statistical ensembles of complex, quaternion, and real matrices," *Journal of Mathematical Physics*, vol. 6, p. 440, 1965.
[59] V. Goel and W. Byrne, "Minimum Bayes-risk automatic speech recognition," *Computer Speech & Language*, vol. 14, no. 2, pp. 115–135, 2000.
[60] D. Graff and C. Cieri, "English Gigaword," 2009.
[61] G. R. Grimmett, "A theorem about random fields," *Bulletin of the London Mathematical Society*, vol. 5, no. 13, pp. 81–84, 1973.
[62] R. Grone and R. Merris, "An algorithm for the second immanant," *Mathematics of Comp*, vol. 43, pp. 589–591, 1984.
[63] O. Häggström, M. C. N. M. Van Lieshout, and J. Møller, "Characterization results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes," *Bernoulli*, vol. 5, no. 4, pp. 641–658, 1999.
[64] J. H. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, no. 1, pp. 84–90, 1960.

[65] W. Hartmann, "On the complexity of immanants," *Linear and Multilinear Algebra*, vol. 18, no. 2, pp. 127–140, 1985.

[66] E. L. Hinrichsen, J. Feder, and T. Jøssang, "Geometry of random sequential adsorption," *Journal of Statistical Physics*, vol. 44, no. 5, pp. 793–827, 1986.

[67] E. Hlawka, "Funktionen von beschränkter variatiou in der theorie der gleichverteilung," *Annali di Matematica Pura ed Applicata*, vol. 54, no. 1, pp. 325–333, 1961.

[68] J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág, "Determinantal processes and independence," *Probability Surveys*, vol. 3, pp. 206–229, 2006.

[69] M. L. Huber and R. L. Wolpert, "Likelihood-based inference for matérn type-iii repulsive point processes," *Advances in Applied Probability*, vol. 41, no. 4, pp. 958–977, 2009.

[70] H. Ishikawa, "Exact optimization for Markov random fields with convex priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1333–1336, 2003.

[71] J. L. Jensen and J. Moller, "Pseudolikelihood for exponential family models of spatial point processes," *The Annals of Applied Probability*, vol. 1, no. 3, pp. 445–461, 1991.

[72] K. Johansson, "Non-intersecting paths, random tilings and random matrices," *Probability Theory and Related Fields*, vol. 123, no. 2, pp. 225–280, 2002.

[73] K. Johansson, "Determinantal processes with number variance saturation," *Communications in Mathematical Physics*, vol. 252, no. 1, pp. 111–148, 2004.

[74] K. Johansson, "The arctic circle boundary and the airy process," *The Annals of Probability*, vol. 33, no. 1, pp. 1–30, 2005.

[75] K. Johansson, "Random matrices and determinantal processes," Arxiv preprint math-ph/0510038, 2005.

[76] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Mathematics*, vol. 26, no. 189–206, pp. 1–1, 1984.

[77] C. W. Ko, J. Lee, and M. Queyranne, "An exact algorithm for maximum entropy sampling," *Operations Research*, vol. 43, no. 4, pp. 684–691, 1995. ISSN 0030-364X.

[78] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[79] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 147–159, 2004.

[80] A. Krause and C. Guestrin, "A note on the budgeted maximization of submodular functions," Technical Report No. CMU-CALD, vol. 5, p. 103, 2005.

[81] A. Kulesza, J. Gillenwater, and B. Taskar, "Near-optimal map inference for determinantal point processes," in *Proceedings of the Neural Information Processing Systems*, 2012.

[82] A. Kulesza and F. Pereira, "Structured learning with approximate inference," *Advances in neural information processing systems*, vol. 20, pp. 785–792, 2008.

[83] A. Kulesza and B. Taskar, "Structured determinantal point processes," in *Proceedings of the Neural Information Processing Systems*, 2010.

[84] A. Kulesza and B. Taskar, "k-DPPs: Fixed-size determinantal point processes," in *Proceedings of the International Conference on Machine Learning*, 2011.

[85] A. Kulesza and B. Taskar, "Learning determinantal point processes," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2011.

[86] S. Kumar and W. Byrne, "Minimum Bayes-risk word alignments of bilingual texts," in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*-vol. 10, pp. 140–147, Association for Computational Linguistics, 2002.

[87] S. Kumar and W. Byrne, "Minimum Bayes-risk decoding for statistical machine translation," in *Proceedings of HLT-NAACL*, pp. 169–176, 2004.

[88] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the International Conference on Machine Learning*, pp. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[89] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 157–224, 1988.

[90] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 323–332, 2009.

[91] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, 2009.

[92] Z. Li and J. Eisner, "First-and second-order expectation semirings with applications to minimum-risk training on translation forests," in *Proceedings of the Conference on Empirical Methods in National Language Processing (EMNLP)*, 2009.

[93] C. Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proceedings of the Workshop on Text Summarization Branches out (WAS 2004)*, pp. 25–26, 2004.

[94] H. Lin and J. Bilmes, "Multi-document summarization via budgeted maximization of submodular functions," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL/HLT)*, 2010.

[95] H. Lin and J. Bilmes, "Learning mixtures of submodular shells with application to document summarization," in *Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, USA, AUAI, July 2012.

[96] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1999.

[97] R. Lyons, "Determinantal probability measures," *Publications Mathématiques de l'IHÉS*, vol. 98, no. 1, pp. 167–212, 2003.

[98] O. Macchi, "The coincidence approach to stochastic point processes," *Advances in Applied Probability*, vol. 7, no. 1, pp. 83–122, 1975.

[99] A. Magen and A. Zouzias, "Near optimal dimensionality reductions that preserve volumes," *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pp. 523–534, 2008.

[100] B. Matérn, "Spatial variation. Stochastic models and their application to some problems in forest surveys and other sampling investigations," *Meddelanden fran statens Skogsforskningsinstitut*, vol. 49, no. 5, 1960.

[101] B. Matérn, *Spatial Variation*. Springer-Verlag, 1986.

[102] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval Journal*, vol. 3, pp. 127–163, 2000.

[103] P. McCullagh and J. Møller, "The permanental process," *Advances in Applied Probability*, pp. 873–888, 2006.

[104] M. L. Mehta and M. Gaudin, "On the density of eigenvalues of a random matrix," *Nuclear Physics*, vol. 18, no. 0, pp. 420–427, 1960. ISSN 0029-5582. doi: 10.1016/0029-5582(60)90414-4.

[105] W. Mei and C. Zhai, "Discovering evolutionary theme patterns from text: An exploration of temporal text mining," in *Proceedings of the SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, 2005.

[106] J. Møller, M. L. Huber, and R. L. Wolpert, "Perfect simulation and moment properties for the matérn type III process," *Stochastic Processes and Their Applications*, vol. 120, no. 11, pp. 2142–2158, 2010.

[107] J. Møller and R. P. Waagepetersen, *Statistical Inference and Simulation for Spatial Point Processes*, vol. 100. CRC Press, 2004.

[108] J. Møller and R. P. Waagepetersen, "Modern statistics for spatial point processes," *Scandinavian Journal of Statistics*, vol. 34, no. 4, pp. 643–684, 2007.

[109] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 467–475, Morgan Kaufmann Publishers Inc., 1999.

[110] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, no. 1, pp. 265–294, 1978.

[111] A. Nenkova, L. Vanderwende, and K. McKeown, "A compositional context sensitive multi-document summarizer: Exploring the factors that influence summarization," in *Proceedings of the Annual Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[112] H. Niederreiter, *Quasi-Monte Carlo Methods*. Wiley Online Library, 1992.

[113] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.

[114] Y. Ogata and M. Tanemura, "Likelihood analysis of spatial point patterns," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 496–518, 1984.

[115] Y. Ogata and M. Tanemura, "Estimation of interaction potentials of marked spatial point patterns through the maximum likelihood method," *Biometrics*, pp. 421–433, 1985.

[116] A. Okounkov, "Infinite wedge and random partitions," *Selecta Mathematica, New Series*, vol. 7, no. 1, pp. 57–81, 2001.

[117] A. Okounkov and N. Reshetikhin, "Correlation function of Schur process with application to local geometry of a random 3-dimensional young diagram," *Journal of the American Mathematical Society*, vol. 16, no. 3, pp. 581–604, 2003.

[118] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in Brain Research*, vol. 155, pp. 23–36, 2006. ISSN 0079-6123.

[119] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the AAAI National Conference on AI*, pp. 133–136, 1982.

[120] C. J. Preston, *Random Fields*. Springer-Verlag New York, 1976.

[121] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[122] K. Raman, P. Shivaswamy, and T. Joachims, "Learning to diversify from implicit feedback," in *WSDM Workshop on Diversity in Document Retrieval*, 2012.

[123] J. J. Ramsden, "Review of new experimental techniques for investigating random sequential adsorption," *Journal of Statistical Physics*, vol. 73, no. 5, pp. 853–877, 1993.

[124] B. D. Ripley, *Statistical Inference for Spatial Processes*. Cambridge University Press, 1991.

[125] B. D. Ripley and F. P. Kelly, "Markov point processes," *Journal of the London Mathematical Society*, vol. 2, no. 1, p. 188, 1977.

[126] B. Sapp, C. Jordan, and B. Taskar, "Adaptive pose priors for pictorial structures," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'10)*, 2010.

[127] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *Journal of Combinatorial Theory, Series B*, vol. 80, no. 2, pp. 346–355, 2000.

[128] D. Shahaf and C. Guestrin, "Connecting the dots between news articles," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, 2010.

[129] D. Shahaf, C. Guestrin, and E. Horvitz, "Trains of thought: Generating information maps," in *Proceedings of the International Conference on World Wide Web*, 2012.

[130] S. E. Shimony, "Finding maps for belief networks is NP-hard," *Artificial Intelligence*, vol. 68, no. 2, pp. 399–410, 1994.

[131] T. Shirai and Y. Takahashi, "Fermion process and Fredholm determinant," in *Proceedings of the ISAAC Congress*, vol. 1, pp. 15–23, Kluwer Academic Publishers, 2000.

[132] T. Shirai and Y. Takahashi, "Random point fields associated with certain fredholm determinants ii: Fermion shifts and their ergodic and gibbs properties," *The Annals of Probability*, vol. 31, no. 3, pp. 1533–1564, 2003.

[133] T. Shirai and Y. Takahashi, "Random point fields associated with certain Fredholm determinants i: fermion, poisson and boson point processes," *Journal of Functional Analysis*, vol. 205, no. 2, pp. 414–463, 2003.

[134] I. M. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 7, no. 4, pp. 784–802, 1967.

[135] I. M. Sobol, "On quasi-Monte Carlo integrations," *Mathematics and Computers in Simulation*, vol. 47, no. 2, pp. 103–112, 1998.

[136] D. Sontag and T. Jaakkola, "New outer bounds on the marginal polytope," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1393–1400, 2007.

[137] A. Soshnikov, "Determinantal random point fields," *Russian Mathematical Surveys*, vol. 55, p. 923, 2000.

[138] D. Stoyan and H. Stoyan, "On one of matérn's hard-core point process models," *Mathematische Nachrichten*, vol. 122, no. 1, pp. 205–214, 1985.

[139] D. Strauss, "A model for clustering," *Biometrika*, vol. 62, no. 2, pp. 467–475, 1975.

[140] A. Swaminathan, C. V. Mathew, and D. Kirovski, "Essential pages," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pp. 173–182, 2009.

[141] R. Swan and D. Jensen, "TimeMines: Constructing timelines with statistical models of word usage," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, 2000.

[142] R. H. Swendsen, "Dynamics of random sequential adsorption," *Physical Review A*, vol. 24, no. 1, p. 504, 1981.

[143] M. Tanemura, "On random complete packing by discs," *Annals of the Institute of Statistical Mathematics*, vol. 31, no. 1, pp. 351–365, 1979.

[144] J. M. Tang and Y. Saad, "A probing method for computing the diagonal of a matrix inverse," *Numerical Linear Algebra with Applications*, 2011.

[145] T. Tao, "Determinantal processes," http://terrytao.wordpress.com/2009/08/23/determinantal-processes/, August 2009.

[146] B. Taskar, V. Chatalbashev, and D. Koller, "Learning associative Markov networks," in *Proceedings of the International Conference on Machine Learning*, p. 102, 2004.

[147] L. G. Valiant, "The complexity of computing the permanent," *Theoretical Computer Science*, vol. 8, no. 2, pp. 189–201, 1979.

[148] M. N. M. Van Lieshout, "Markov point processes and their applications," *Recherche*, vol. 67, p. 02, 2000.

[149] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 2000.

[150] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[151] S. S. Vempala, *The Random Projection Method*, vol. 65. American Mathematical Society, 2004.

[152] D. Vere-Jones, "Alpha-permanents and their applications to multivariate gamma, negative binomial and ordinary binomial distributions," *New Zealand Journal of Mathematics*, vol. 26, pp. 125–149, 1997.

[153] J. Vondrák, C. Chekuri, and R. Zenklusen, "Submodular function maximization via the multilinear relaxation and contention resolution schemes," in *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pp. 783–792, 2011.

[154] C. Wayne, "Multilingual topic detection and tracking: Successful research enabled by Corpora and evaluation," in *Proceedings of the International Language Resources and Evaluation (LREC)*, 2000.

[155] R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang, "Evolutionary timeline summarization: A balanced optimization framewo rk via iterative substitution," in *Proceedings of the Annual Conference on Research and Development in Information Retrieval (SIGIR)*, 2011.

[156] C. Yanover, T. Meltzer, and Y. Weiss, "Linear programming relaxations and belief propagation — an empirical study," *The Journal of Machine Learning Research*, vol. 7, pp. 1887–1907, 2006.

[157] C. Yanover and Y. Weiss, "Approximate inference and protein folding," *Advances in Neural Information Processing Systems*, vol. 15, pp. 1457–1464, 2002.

[158] Y. Yue and T. Joachims, "Predicting diverse subsets using structural SVMs," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[159] C. X. Zhai, W. W. Cohen, and J. Lafferty, "Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval," in *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 10–17, 2003.