Efficient Sampling for k-Determinantal Point Processes

Chengtao Li
MIT

Stefanie Jegelka
MIT

Suvrit Sra
MIT

Abstract

Determinantal Point Processes (DPPs) are elegant probabilistic models of repulsion and diversity over discrete sets of items. But their applicability to large sets is hindered by expensive cubic-complexity matrix operations for basic tasks such as sampling. In light of this, we propose a new method for approximate sampling from discrete k-DPPs. Our method takes advantage of the diversity property of subsets sampled from a DPP, and proceeds in two stages: first it constructs coresets for the ground set of items; thereafter, it efficiently samples subsets based on the constructed coresets. As opposed to previous approaches, our algorithm aims to minimize the total variation distance to the original distribution. Experiments on both synthetic and real datasets indicate that our sampling algorithm works efficiently on large data sets, and yields more accurate samples than previous approaches.

1 Introduction

Subset selection problems lie at the heart of many applications where a small subset of items must be selected to represent a larger population. Typically, the selected subsets are expected to fulfill various criteria such as sparsity, grouping, or diversity. Our focus is on *diversity*, a criterion that plays a key role in a variety of applications, such as gene network subsampling [9], document summarization [36], video summarization [23], content driven search [4], recommender systems [47], sensor placement [29], among many others [1, 5, 21, 30, 33, 43, 44].

Diverse subset selection amounts to sampling from the set of all subsets of a ground set according to a measure that places more mass on subsets with qualitatively different items. An elegant realization of this idea is given by Determinantal Point Processes (DPPs), which are probabilistic models that capture diversity by assigning subset probabilities proportional to (sub)determinants of a kernel matrix.

DPPs enjoy rising interest in machine learning [4, 23, 28, 30, 32, 34, 38]; a part of their appeal can be attributed to computational tractability of basic tasks such as computing partition functions, sampling, and extracting marginals [27, 33]. But despite being polynomial-time, these tasks remain infeasible for large data sets. DPP sampling, for example, relies on an eigendecomposition of the DPP kernel, whose cubic complexity is a huge impediment. Cubic preprocessing costs also impede wider use of the cardinality constrained variant k-DPP [32].

These drawbacks have triggered work on approximate sampling methods. Much work has been devoted to approximately sample from a DPP by first approximating its kernel via algorithms such as the Nyström method [3], Random Kitchen Sinks [2, 41], or matrix ridge approximations [45, 46], and then sampling based on this approximation. However, these methods are somewhat inappropriate for sampling because they aim to project the DPP kernel onto a lower dimensional space while minimizing a matrix norm, rather than minimizing an error measure sensitive to determinants. Alternative methods use a dual formulation [30], which however presupposes a decomposition $L = XX^\top$ of the DPP kernel, which may be unavailable and inefficient to compute in practice. Finally, MCMC [6, 10, 14, 28] offers a potentially attractive avenue different from the above approaches that all rely on the same spectral technique.

We pursue a yet different approach. While being similar to matrix approximation methods in exploiting redundancy in the data, in sharp contrast to methods that minimize matrix norms, we focus on minimizing the total variation distance between the original DPP and our approximation. As a result, our approximation models the true DPP probability distribution more faithfully, while permitting faster sampling. We make the following key contributions:

– An algorithm that constructs coresets for approximating a k-DPP by exploiting latent structure in the data. The construction, aimed at minimizing the total variation distance, takes $O(NM^3)$ time; linear in the number N of data points. The construction works as the

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

overhead of sampling algorithm and is much faster than standard cubic-time overhead that exploits eigendecomposition of kernel matrices. We also investigate conditions under which such an approximation is good.

– A sampling procedure that yields approximate k-DPP subsets using the constructed coresets. While most other sampling methods sample diverse subsets in $O(k^2N)$ time, the sampling time for our coreset-based algorithm is $O(k^2M)$, where $M \ll N$ is a user-specified parameter *independent of* N.

Our experiments indicate that our construction works well for a wide range of datasets, delivers more accurate approximations than the state-of-the-art, and is more efficient, especially when multiple samples are required.

Overview of our approach. Our sampling procedure runs in two stages. Its first stage constructs an approximate probability distribution close in total variation distance to the true k-DPP. The next stage efficiently samples from this approximate distribution.

Our approximation is motivated by the diversity sampling nature of DPPs: in a DPP most of the probability mass will be assigned to diverse subsets. This leaves room for exploiting redundancy. In particular, if the data possesses latent grouping structure, certain subsets will be much more likely to be sampled than others. For instance, if the data are tightly clustered, then any sample that draws two points from the same cluster will be very unlikely.

The key idea is to reduce the effective size of the ground set. We do this via the idea of coresets [17, 25], small subsets of the data that capture function values of interest almost as well as the full dataset. Here, the function of interest is a k-DPP distribution. Once a coreset is constructed, we can sample a subset of core points, and then, based on this subset, sample a subset of the ground set. For a coreset of size M, our sampling time is $O(k^2M)$, which is *independent of* N since we are using k-DPPs [32].

Related work. DPPs have been studied in statistical physics and probability [11, 12, 27]; they have witnessed rising interest in machine learning [21, 23, 30, 32–34, 38, 44]. Cardinality-conditioned DPP sampling is also referred to as "volume sampling", which has been used for matrix approximations [15, 16]. Several works address faster DPP sampling via matrix approximations [3, 15, 30, 37] or MCMC [10, 28]. Except for MCMC, even if we exclude preprocessing, known sampling methods still require $O(k^2N)$ time for a single sample; we reduce this to $O(k^2M)$. Finally, different lines of work address learning DPPs [4, 22, 31, 38] and MAP estimation [20].

Coresets have been applied to large-scale clustering [8, 18, 24, 26], PCA and CCA [18, 39], and segmentation of streaming data [42].

2 Setup and basic definitions

A determinantal point process $\mathrm{DPP}(L)$ is a distribution over all subsets of a ground set $\mathcal{Y}$ of cardinality N. It is determined by a positive semidefinite kernel $L \in \mathbb{R}^{N\times N}$. Let L_Y be the submatrix of L consisting of the entries L_{ij} with $i,j \in Y \subseteq \mathcal{Y}$. Then, the probability $P_L(Y)$ of observing $Y \subseteq \mathcal{Y}$ is proportional to $\det(L_Y)$; consequently, $P_L(Y) = \det(L_Y)/\det(L+I)$. Conditioning on sampling sets of fixed cardinality k, one obtains a k-DPP [32]:

$$\begin{aligned} P_{L,k}(Y) &:= P_L(Y \mid |Y| = k) \\ &= \det(L_Y) e_k(L)^{-1} [\![|Y| = k]\!], \end{aligned}$$

where $e_k(L)$ is the k-th coefficient of the characteristic polynomial $\det(\lambda I - L) = \sum_{k=0}^{N} (-1)^k e_k(L) \lambda^{N-k}$. We assume that $P_{L,k}(Y) > 0$ for all subsets $Y \subseteq \mathcal{Y}$ of cardinality k. To simplify notation, we also write $P_k \triangleq P_{L,k}$.

Our goal is to construct an approximation $\widehat{P}_k$ to P_k that is close in *total variation distance*

$$\|\widehat{P}_k - P_k\|_{\mathrm{tv}} := \tfrac{1}{2} \sum_{Y\subseteq\mathcal{Y}, |Y|=k} |\widehat{P}_k(Y) - P_k(Y)|, \quad (1)$$

and permits faster sampling than P_k. Broadly, we proceed as follows. First, we define a partition $\Pi = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_M\}$ of $\mathcal{Y}$ and extract a subset $\mathcal{C} \subset \mathcal{Y}$ of M core points, containing one point from each part. Then, for the set $\mathcal{C}$ we construct a special kernel $\widetilde{L}$ (as described in Section 3). When sampling, we first sample a set $\widetilde{Y} \sim \mathrm{DPP}_k(\widetilde{L})$ and then, for each $c \in \widetilde{Y}$ we uniformly sample one of its assigned points $y \in \mathcal{Y}_c$. These second-stage points y form our final sample. We denote the resulting distribution by $\widehat{P}_k = P_{\mathcal{C},k}$. Algorithm 1 formalizes the sampling procedure, which, after one eigendecomposition of the small matrix $\widetilde{L}$.

We begin by analyzing the effect of the partition on the approximation error, and then devise an algorithm to approximately minimize the error. We empirically evaluate our approach in Section 6.

3 Coreset sampling

Let $\Pi = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_M\}$ be a *partition* of $\mathcal{Y}$, i.e., $\cup_{i=1}^{M} \mathcal{Y}_i = \mathcal{Y}$ and $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$. We call $\mathcal{C} \subseteq \mathcal{Y}$ a *coreset* with respect to a partition Π if $|\mathcal{C} \cap \mathcal{Y}_i| = 1$ for $i \in [M]$. With a slight abuse of notation, we index each part $\mathcal{Y}_c \in \Pi$ by its core $c \in \mathcal{C} \cap \mathcal{Y}_c$. Based on the partition Π, we call a set $Y \subseteq \mathcal{Y}$ *singular*[1] with respect to $\Pi' \subseteq \Pi$, if for $\mathcal{Y}_i \in \Pi'$ we have $|Y \cap \mathcal{Y}_i| \leq 1$ and for $\mathcal{Y}_j \in \Pi \backslash \Pi'$ we have $|Y \cap \mathcal{Y}_j| = 0$. We say Y is k-singular if Y is singular and $|Y| = k$.

Given a partition Π and core $\mathcal{C}$, we construct a rescaled core kernel $\widetilde{L} \in \mathbb{R}^{M\times M}$ with entries $\widetilde{L}_{c,c'} = \sqrt{|\mathcal{Y}_c||\mathcal{Y}_{c'}|} L_{c,c'}$.

[1] In combinatorial language, Y is an independent set in the partition matroid defined by Π.

We then use this smaller matrix $\widetilde{L}$ and its eigendecomposition as an input to our two-stage sampling procedure in Algorithm 1, which we refer to as CoreDpp. The two stages are: (i) sample a k-subset from $\mathcal{C}$ according to $\mathrm{DPP}_k(\widetilde{L})$; and (ii) for each c, pick an element $y \in \mathcal{Y}_c$ uniformly at random. This algorithm uses only the much smaller matrix $\widetilde{L}$ and samples a subset from $\mathcal{Y}$ in $O(k^2M)$ time. When $M \ll N$ and we want many samples, it translates into a notable improvement over the $O(k^2N)$ time of sampling directly from $\mathrm{DPP}_k(L)$.

The following lemma shows that CoreDpp is equivalent to sampling from a k-Dpp where we replace each point in $\mathcal{Y}$ by its corresponding core point, and sample with the resulting induced kernel $L_{\mathcal{C}(\mathcal{Y})}$.

Lemma 1. CoreDpp *is equivalent to sampling from* $\mathrm{DPP}_k(L_{\mathcal{C}(\mathcal{Y})})$*, where in* $L_{\mathcal{C}(\mathcal{Y})}$ *we replace each element in* $\mathcal{Y}_c$ *by* c*, for all* $c \in \mathcal{C}$*.*

Proof. We denote the distribution induced by Algorithm 1 by $P_{\mathcal{C},k}$ and that induced by $\mathrm{DPP}_k(L_{\mathcal{C}(\mathcal{Y})})$ by P'_k.

First we claim that both sampling algorithms can only sample k-singular subsets. By construction, $P_{\mathcal{C},k}$ picks one or zero elements from each $\mathcal{Y}_c$. For P'_k, if Y is k-nonsingular, then there would be identical rows in $(L_{\mathcal{C}(\mathcal{Y})})_Y = L_{\mathcal{C}(Y)}$, resulting in $\det(L_{\mathcal{C}(Y)}) = 0$. Hence both $P_{\mathcal{C},k}$ and P'_k only assign nonzero probability to k-singular sets Y. As a result, we have

$$e_k(L_{\mathcal{C}(\mathcal{Y})}) = \sum_{C \text{ is } k\text{-singular}} \Big(\prod_{c\in C} |\mathcal{Y}_c|\Big) \det(L_C)$$
$$= \sum_{C\subseteq\mathcal{C},|C|=k} \Big(\prod_{c\in C} |\mathcal{Y}_c| \det(L_C)\Big) = \sum_{|C|=k} \det(\widetilde{L}_C) = e_k(\widetilde{L}).$$

For any $Y = \{y_1, \ldots, y_k\} \subseteq \mathcal{Y}$ that is k-singular, we have

$$P_{\mathcal{C},k}(Y) = \frac{\det(\widetilde{L}_{\mathcal{C}(Y)})}{e_k(\widetilde{L}) \prod_{i=1}^k |\mathcal{Y}_{\mathcal{C}(y_i)}|} \tag{2}$$
$$= \frac{(\prod_{i=1}^k |\mathcal{Y}_{\mathcal{C}(y_i)}|) \det(L_{\mathcal{C}(Y)})}{e_k(L_{\mathcal{C}(\mathcal{Y})}) \prod_{i=1}^k |\mathcal{Y}_{\mathcal{C}(y_i)}|} \tag{3}$$
$$= \frac{\det(L_{\mathcal{C}(Y)})}{e_k(L_{\mathcal{C}(\mathcal{Y})})} = P'_k(Y), \tag{4}$$

which shows that these two distributions are identical, i.e., sampling from $\mathrm{DPP}_k(\widetilde{L})$ followed by uniform sampling is equivalent to directly sampling from $\mathrm{DPP}_k(L_{\mathcal{C}(\mathcal{Y})})$. □

4 Partition, distortion and approximation error

Let us provide some insight on quantities that affect the distance $\|P_{\mathcal{C},k} - P_k\|_{\mathrm{tv}}$ when sampling with Algorithm 1. In a nutshell, this distance depends on three key quantities

Algorithm 1 CoreDpp Sampling

Input: core kernel $\widetilde{L} \in \mathbb{R}^{M\times M}$ and its eigendecomposition; partition Π; size k
sample $C \sim \mathrm{DPP}_k(\widetilde{L})$
sample $y_i \sim \mathrm{Uniform}(\mathcal{Y}_c)$ for $c \in C$
return $Y = \{y_1, \ldots, y_k\}$

(defined below): the probability of nonsingularity δ_Π, the distortion factor $1 + \varepsilon_\Pi$, and the normalization factor.

For a partition Π we define the *nonsingularity probability* δ_Π as the probability that a draw $Y \sim \mathrm{DPP}_k(L)$ is not singular with respect to any $\Pi' \subseteq \Pi$.

Given a coreset $\mathcal{C}$, we define the *distortion factor* $1 + \varepsilon_\Pi$ (for $\varepsilon_\Pi \geq 0$) as a partition-dependent quantity, so that for any $c \in \mathcal{C}$, for all $u, v \in \mathcal{Y}_c$, and for any $(k-1)$-singular set S with respect to $\Pi \setminus \mathcal{Y}_c$ the following bound holds:

$$\frac{\det(L_{S\cup\{u\}})}{\det(L_{S\cup\{v\}})} = \frac{L_{u,u} - L_{u,S}L_S^{-1}L_{S,u}}{L_{v,v} - L_{v,S}L_S^{-1}L_{S,v}} \leq 1 + \varepsilon_\Pi. \tag{5}$$

If ϕ is the feature map corresponding to the kernel L, then geometrically, the numerator of (5) is the length of the projection of $\phi(u)$ onto the orthogonal complement of $\mathrm{span}\{\phi(s) \mid s \in S\}$.

The *normalization factor* for a k-Dpp (L) is simply $e_k(L)$.

Given Π, $\mathcal{C}$ and the corresponding nonsingularity probability and distortion factors, we have the following bound:

Lemma 2. *Let* $Y \sim \mathrm{DPP}_k(L)$ *and* $\mathcal{C}(Y)$ *be the set where we replace each* $y \in Y$ *by its core* $c \in \mathcal{C}$*, i.e.,* $y \in \mathcal{Y}_c$*. With probability* $1 - \delta_\Pi$*, it holds that*

$$(1 + \varepsilon_\Pi)^{-k} \leq \frac{\det(L_{\mathcal{C}(Y)})}{\det(L_Y)} \leq (1 + \varepsilon_\Pi)^k. \tag{6}$$

Proof. Let $c \in \mathcal{C}$ and consider any $(k-1)$-singular set S with respect to $\Pi \setminus \mathcal{Y}_c$. Then, for any $v \in \mathcal{Y}_c$, using Schur complements and by the definition of ε_Π we see that

$$(1 + \varepsilon_\Pi)^{-1} \leq \frac{\det(L_{S\cup\{c\}})}{\det(L_{S\cup\{v\}})} = \frac{L_{c,c} - L_{c,S}L_S^{-1}L_{S,c}}{L_{v,v} - L_{v,S}L_S^{-1}L_{S,v}}$$
$$= \frac{||Q_{S^\perp}\phi(c)||^2}{||Q_{S^\perp}\phi(v)||^2} \leq (1 + \varepsilon_\Pi).$$

Here, $Q_{S^\perp}$ is the projection onto the orthogonal complement of $\mathrm{span}\{\phi(s) \mid s \in S\}$, and ϕ the feature map corresponding to the kernel L.

With a minor abuse of notation, we denote by $\mathcal{C}(y) = c$ the core point corresponding to y, i.e., $y \in \mathcal{Y}_c$. For any $Y = \{y_1, \ldots, y_k\}$, we then define the sets $Y_i = \{\mathcal{C}(y_1), \ldots, \mathcal{C}(y_i), y_{i+1}, \ldots, y_k\}$, where we gradually replace each point by its core point, with $Y_0 = Y$. If Y is

k-singular, then $\mathcal{C}(y_i) \neq \mathcal{C}(y_j)$ whenever $i \neq j$, and, for any $0 \leq i \leq k-1$, it holds that

$$(1+\varepsilon_\Pi)^{-1} \leq \frac{\det(L_{Y_{i+1}})}{\det(L_{Y_i})} \leq 1+\varepsilon_\Pi.$$

Hence we have

$$(1+\varepsilon_\Pi)^{-k} \leq \frac{\det(L_{\mathcal{C}(Y)})}{\det(L_Y)} = \prod_{i=0}^{k-1} \frac{\det(L_{Y_{i+1}})}{\det(L_{Y_i})} \leq (1+\varepsilon_\Pi)^k.$$

This bound holds when Y is k-singular, and, by definition of δ_Π, this happens with probability $1-\delta_\Pi$. □

Assuming ε_Π is small, Lemma 2 states that if replacing a single element in a given subset with another one in the same part does not cause much distortion, then replacing all elements in the subset with their corresponding cores will cause little distortion. This observation is key to our approximation: if we can construct such a partition and coreset, we can safely replace all elements with core points and then approximately sample with little distortion. More precisely, we then obtain the following result that bounds the variational error. Our subsequent construction aims to minimize this bound.

Theorem 3. *Let $P_k = \mathrm{DPP}_k(L)$ and let $P_{\mathcal{C},k}$ be the distribution induced by Algorithm 1. With the normalization factors $Z = e_k(L)$ and $Z_\mathcal{C} = e_k(\widetilde{L})$, the total variation distance between P_k and $P_{\mathcal{C},k}$ is bounded by*

$$\|P_k - P_{\mathcal{C},k}\|_{\mathrm{tv}} \leq |1 - \tfrac{Z_\mathcal{C}}{Z}| + k\varepsilon_\Pi + (1-k\varepsilon_\Pi)\delta_\Pi.$$

Proof. From the definition of Z and $Z_\mathcal{C}$ we know that $Z = \sum_{|Y|=k} \det(L_Y)$ and

$$\begin{aligned} Z_\mathcal{C} &= \sum_{|Y|=k} \det((L_{\mathcal{C}(\mathcal{Y})})_Y) = \sum_{|Y|=k} \det(L_{\mathcal{C}(Y)}) \\ &= \sum_{Y \text{ } k\text{-singular}} \det(L_{\mathcal{C}(Y)}). \end{aligned}$$

The last equality follows since, as argued above, $\det(L_{\mathcal{C}(Y)}) = 0$ for nonsingular Y. It follows that

$$\begin{aligned} \|P_k - P_{\mathcal{C},k}\|_{\mathrm{tv}} &= \sum_{|Y|=k} |P_k(Y) - P_{\mathcal{C},k}(Y)| \\ &= \sum_{Y \text{ } k\text{-singular}} |P_k(Y) - P_{\mathcal{C},k}(Y)| + \sum_{Y \text{ } k\text{-nonsingular}} P_k(Y). \end{aligned} \tag{7}$$

For the first term, we have

$$\begin{aligned} &\sum_{Y \text{ } k\text{-singular}} |P_k(Y) - P_{\mathcal{C},k}(Y)| \\ &= \sum_{Y \text{ } k\text{-singular}} \Big|\frac{\det(L_Y)}{Z} - \frac{\det(L_{\mathcal{C}(Y)})}{Z_\mathcal{C}}\Big| \\ &\leq \sum_{Y \text{ } k\text{-singular}} \Big|\frac{1}{Z}(\det(L_Y) - \det(L_{\mathcal{C}(Y)}))\Big| \\ &\quad + \sum_{Y \text{ } k\text{-singular}} \Big|\det(L_{\mathcal{C}(Y)})\Big(\frac{1}{Z} - \frac{1}{Z_\mathcal{C}}\Big)\Big| \\ &= \frac{1}{Z} \sum_{Y \text{ } k\text{-singular}} \det(L_Y)\Big|1 - \frac{\det(L_{\mathcal{C}(Y)})}{\det(L_Y)}\Big| + Z_\mathcal{C}\Big|\frac{1}{Z} - \frac{1}{Z_\mathcal{C}}\Big| \\ &\leq k\varepsilon_\Pi(1-\delta_\Pi) + \Big|1 - \frac{Z_\mathcal{C}}{Z}\Big|, \end{aligned}$$

where the first inequality uses the triangle inequality and the second inequality relies on Lemma 2. For the second term in (7), we use that, by definition of δ_Π,

$$\sum_{Y \text{ } k\text{-nonsingular}} P_k(Y) = \delta_\Pi.$$

Thus the total variation difference is bounded as

$$\begin{aligned} \|P_k - P_{\mathcal{C},k}\|_{\mathrm{tv}} &\leq \Big|1 - \frac{Z_\mathcal{C}}{Z}\Big| + k\varepsilon_\Pi(1-\delta_\Pi) + \delta_\Pi \\ &= \Big|1 - \frac{Z_\mathcal{C}}{Z}\Big| + k\varepsilon_\Pi + (1-k\varepsilon_\Pi)\delta_\Pi. \end{aligned}$$ □

In essence, if the probability of nonsingularity and the distortion factor are low, then it is possible to obtain a good coreset approximation. This holds, for example, if the data has intrinsic (grouping) structure. In the next subsection we provide further intuition on when we can achieve low error.

4.1 Sufficient conditions for a good bound

Theorem 3 depends on the data and the partition Π. Here, we aim to obtain some further intuition on the properties of Π that govern the bound. At the same time, these properties suggest sufficient conditions for a "good" coreset $\mathcal{C}$. For each $\mathcal{Y}_c$, we define the diameter

$$\rho_c := \max_{u,v \in \mathcal{Y}_c} \sqrt{L_{uu} + L_{vv} - 2L_{uv}}. \tag{8}$$

Next, define the minimum distance of any point $u \in \mathcal{Y}_c$ to the subspace spanned by the feature vectors of points in a "complementary" set S that is singular with respect to $\Pi \setminus \mathcal{Y}_c$:

$$d_c := \min_{S,u} \sqrt{\frac{\det(L_{S\cup\{u\}})}{\det(L_S)}} = \min_{S,u} \sqrt{L_{u,u} - L_{u,S}L_S^{-1}L_{S,u}}.$$

Lemma 4 connects these quantities with ε_Π; it essentially poses a separability condition on Π (i.e., Π needs to be "aligned" with the data) so that the bound on ε_Π holds.

Lemma 4. *If $d_c > \rho_c$ for all $c \in \mathcal{C}$, then*

$$\varepsilon_\Pi \leq \max_{c\in\mathcal{C}} \frac{(2d_c - \rho_c)\rho_c}{(d_c - \rho_c)^2}. \tag{9}$$

Proof. For any $c \in \mathcal{C}$ and any $u, v \in \mathcal{Y}_c$ and S $(k-1)$-singular with respect to $\Pi\backslash\mathcal{Y}_c$, we have

$$\frac{\det(L_{S\cup\{u\}})}{\det(L_{S\cup\{v\}})} = \frac{\det(L_S)(L_{u,u} - L_{u,S}L_S^{-1}L_{S,u})}{\det(L_S)(L_{v,v} - L_{v_i,S}L_S^{-1}L_{S,v_i})} = \frac{L_{u,u} - L_{u,S}L_S^{-1}L_{S,u}}{L_{v,v} - L_{v,S}L_S^{-1}L_{S,v}} = \frac{\|Q_{S^\perp}\phi(u)\|^2}{\|Q_{S^\perp}\phi(v)\|^2}.$$

Without loss of generality, we assume $\det(L_{S\cup\{u\}}) \geq \det(L_{S\cup\{v\}})$. By definition of ρ_c we know that

$$0 \leq \|Q_{S^\perp}\phi(u)\| - \|Q_{S^\perp}\phi(v)\| \leq \|Q_{S^\perp}(\phi(u) - \phi(v))\| \leq \|\phi(u) - \phi(v)\| \leq \rho_c.$$

Since $0 < \|Q_{S^\perp}\phi(v)\| \leq \|Q_{S^\perp}\phi(u)\| \leq \|\phi(u)\|$ by assumption, we have

$$\frac{\|Q_{S^\perp}\phi(u)\|^2}{\|Q_{S^\perp}\phi(v)\|^2} \leq \frac{\|Q_{S^\perp}\phi(u)\|^2}{(\|Q_{S^\perp}\phi(u)\| - \rho_c)^2} \leq \Big(\frac{\|\phi(u)\|}{\|\phi(u)\| - \rho_c}\Big)^2 \leq \Big(\frac{d_c}{(d_c - \rho_c)}\Big)^2.$$

Then, by definition of ε_Π, we have

$$1 + \varepsilon_\Pi \leq \max_c \frac{d_c^2}{(d_c - \rho_c)^2},$$

from which it follows that

$$\varepsilon_\Pi \leq \max_c \frac{(2d_c - \rho_c)\rho_c}{(d_c - \rho_c)^2}. \qquad \square$$

5 Efficient construction

Theorem 3 states an upper bound on the error induced by CoreDPP and relates the total variation distance to Π and $\mathcal{C}$. Next, we explore how to efficiently construct Π and $\mathcal{C}$ that approximately minimize the upper bound.

5.1 Constructing Π

Any set Y sampled via CoreDPP is, by construction, singular with respect to Π. In other words, CoreDPP assigns zero mass to any nonsingular set. Hence, we wish to construct a partition Π such that its nonsingular sets have low probability under $\text{DPP}_k(L)$. The optimal such partition minimizes the probability δ_Π of nonsingularity. A small δ_Π value also means that the parts of Π are dense and compact, i.e., the diameter ρ_c in Equation (8) is small.

Finding such a partition optimally is hard, so we resort to local search. Starting with a current partition Π, we reassign each y to a part $\mathcal{Y}_c$ to minimize δ_Π. If we assign y to $\mathcal{Y}_c$, then the probability of sampling a set Y that is singular with respect to the new partition Π is

$$\begin{aligned}\mathbb{P}[Y \sim \text{DPP}_k(L) \text{ is singular}] &= \frac{1}{Z}\sum_{Y\, k\text{-singular}} \det(L_Y) \\ &= \frac{1}{Z}\Big(\sum_{Y\, k\text{-sing.}, y\notin Y} \det(L_Y) + \sum_{Y\, k\text{-sing.}, y\in Y} \det(L_Y)\Big) \\ &= \frac{1}{Z}\Big(\text{const} + \sum_{Y'\ (k-1)\text{-sing. w.r.t } \Pi\setminus\mathcal{Y}_c} \det(L_{Y'\cup\{y\}})\Big) \\ &= \frac{1}{Z}\Big(\text{const} + L_{yy}s_{k-1}^{\Pi}(L^y_{\setminus c})\Big),\end{aligned}$$

where $s_k^\Pi(L) := \sum_{Y\ k\text{-sing.}} \det(L_Y)$. The matrix $L_{\setminus c}$ denotes L with rows $\mathcal{Y}_c$ and columns $\mathcal{Y}_c$ deleted, and $L^y = L - L_{\mathcal{Y},y}L_{y,\mathcal{Y}}$. For local search, we would hence compute $L_{yy}s_{k-1}^\Pi(L^y_{\setminus c})$ for each point y and core c, assign y to the highest-scoring c. Since this testing is still expensive, we introduce further speedups in Section 5.3.

5.2 Constructing $\mathcal{C}$

When constructing $\mathcal{C}$, we aim to minimize the upper bound on the total variation distance between P_k and $P_{\mathcal{C},k}$ stated in Theorem 3. Since δ_Π and ε_Π only depend on Π and not on $\mathcal{C}$, we here focus on minimizing $|1 - \frac{Z_\mathcal{C}}{Z}|$, i.e., bringing $Z_\mathcal{C}$ as close to Z as possible. To do so, we again employ local search and subsequently swap each $c \in \mathcal{C}$ with its best replacement $v \in \mathcal{Y}_c$. Let $\mathcal{C}^{c,v}$ be $\mathcal{C}$ with c replaced by v. We aim to find the best swap

$$\begin{aligned} v &= \text{argmin}_{v\in\mathcal{Y}_c}|Z - Z_{\mathcal{C}^{c,v}}| &(10)\\ &= \text{argmin}_{v\in\mathcal{Y}_c}|Z - e_k(L_{\mathcal{C}^{c,v}(\mathcal{Y})})|. &(11)\end{aligned}$$

Computing Z requires computing the coefficients $e_k(L)$, which takes a total of $O(N^3)$ time[2]. In the next section, we therefore consider a fast approximation.

5.3 Faster constructions and further speedups

Local search procedures for optimizing Π and $\mathcal{C}$ can be further accelerated by a sequence of relaxations that we found to work well in practice (see Section 6). We begin with the quantity $s_{k-1}^\Pi(L^y_{\setminus c})$ that involves summing over subdeterminants of the large matrix L. Assuming the initialization is not too bad, we can use the current $\mathcal{C}$ to approximate $\mathcal{Y}$. In particular, when re-assigning y, we substitute all other elements with their corresponding cores, resulting in the kernel $\widehat{L} = L_{\mathcal{C}(\mathcal{Y})}$. This changes our objective to finding the $c \in \mathcal{C}$ that maximizes $s_{k-1}^\Pi(\widehat{L}^y_{\setminus c})$. Key to a fast approximation is now Lemma 5, which follows from Lemma 1.

[2] In theory, this can be computed in $O(N^\omega \log(N))$ time [13], but the eigendecompositions and dynamic programming used in practice typically take cubic time.

Lemma 5. *For all $k \leq |\Pi|$, it holds that*

$$s_k^{\Pi}(L_{\mathcal{C}(\mathcal{Y})}) = e_k(L_{\mathcal{C}(\mathcal{Y})}) = e_k(\tilde{L}).$$

Proof.

$$s_k^{\Pi}(L_{\mathcal{C}(\mathcal{Y})}) = \sum_{Y \ k\text{-sing.}} \det((L_{\mathcal{C}(\mathcal{Y})})_Y) = \sum_{Y \ k\text{-sing.}} \det(L_{\mathcal{C}(Y)})$$
$$= \sum_{|Y|=k} \det(L_{\mathcal{C}(Y)}) = e_k(L_{\mathcal{C}(\mathcal{Y})}) = e_k(\widetilde{L});$$

the last equality was shown in the proof of Theorem 3. □

Computing the normalizer $e_k(\tilde{L})$ only needs $O(M^3)$ time. We refer to this acceleration as CoreDpp-z.

Second, when constructing $\mathcal{C}$, we observed that $Z_{\mathcal{C}}$ is commonly much smaller than Z. Hence, a fast approximation merely greedily increases $Z_{\mathcal{C}}$ without computing Z.

Third, we can be lazy in a number of updates: for example, we only consider changing cores for the part that changes. When a part $\mathcal{Y}_c$ receives a new member, we check whether to switch the current core c to the new member. This reduction keeps the core adjustment at time $O(M^3)$. Moreover, when re-assigning an element y to a different part $\mathcal{Y}_c$, it is usually sufficient to only check a few, say, ν parts with cores closest to y, and not all parts. The resulting time complexity for each element is $O(M^3)$.

With this collection of speedups, the approximate construction of Π and $\mathcal{C}$ takes $O(NM^3)$ for each iteration, which is linear in N, and hence a huge speedup over direct methods that require $O(N^3)$ preprocessing. The iterative algorithm is shown in Algorithm 2. The initialization also affects the algorithm performance, and in practice we find that kmeans++ as an initialization works well. Thus we use CoreDpp to refer to the algorithm that is initialized with kmeans++ and uses all the above accelerations. In practice, the algorithm converges very quickly, and most of the progress occurs in the first pass through the data. Hence, if desired, one can even use early stopping.

Algorithm 2 Iterative construction of Π and $\mathcal{C}$

Require: Π initial partition; $\mathcal{C}$ initial coreset; k the size of sampled subset; ν number of nearest neighbors taken into consideration
while not converged **do**
  for all $y \in \mathcal{Y}$ **do**
    $c \leftarrow$ group in which y lies currently: $y \in \mathcal{Y}_c$
    if $y \in \mathcal{C}$ **then**
      continue
    end if
    $G \leftarrow$ {groups of ν cores nearest to X_y}
    $g^* = \text{argmax}_{g \in G} s_{k-1}^{\Pi}(\widehat{L}_{\setminus g}^{y})$
    if $c \neq g^*$ **then**
      $\mathcal{Y}_c = \mathcal{Y}_c \setminus \{y\}$
      $\mathcal{Y}_{g^*} = \mathcal{Y}_{g^*} \cup \{y\}$
      if $e_k(L_{\mathcal{C}^{g^*,j}(\mathcal{Y})}) > e_k(L_{\mathcal{C}^{c,j}(\mathcal{Y})})$ **then**
        $\mathcal{C} \leftarrow \mathcal{C}^{g^*,y}$
      end if
    end if
  end for
  for all $g \in [M]$ **do**
    $j = \text{argmax}_{j \in \mathcal{Y}_g} e_k(L_{\mathcal{C}^{g,j}(\mathcal{Y})})$
    $\mathcal{C} = \mathcal{C}^{g,j}$
  end for
end while

6 Experiments

We next evaluate CoreDpp, and compare its efficiency and effectiveness against three competing approaches:

- Partitioning using k-means (with kmeans++ initialization [7]), with $\mathcal{C}$ chosen as the centers of the clusters; referred to as *K++* in the results.
- The adaptive, stochastic Nyström sampler of [3] (*NysStoch*). We used M dimensions for NysStoch, to use the same dimensionality as CoreDpp.
- The Metropolis-Hastings DPP sampler *MCDPP* [28]. We use the well-known Gelman and Rubin multiple sequence diagnostic [19] to empirically judge mixing.

In addition, we show results using different variants of CoreDpp: CoreDpp-z described in Section 5.3 and variants that are initialized either randomly (CoreDpp-r) or via kmeans++ (CoreDpp).

6.1 Synthetic Dataset

We first explore the effect of our fast approximate sampling on controllable synthetic data. The experiments here compare the accuracy of the faster CoreDpp from Section 5.3 to CoreDpp-z, CoreDpp-r and *K++*.

We generate an equal number of samples from each of *nClust* 30-dimensional Gaussians with means of varying length (ℓ_2-norm) and unit variance, and then rescale the samples to have the same length. As the length of the samples increases, ε_{Π} and δ_{Π} shrink. Finally, L is a linear kernel. Throughout this experiment we set $k = 4$ and $N = 60$ to be able to exactly compute $\|\widehat{P}_k - P_k\|_{\text{tv}}$. We extract $M = 10$ core points and use $\nu = 3$ neighboring cores. Recall from Section 5.3 that when considering the parts that one element should be assigned to, it is usually sufficient to only check ν parts with cores closest to y. Thus, $\nu = 3$ means we only consider re-assigning each element to its three closest parts.

Results. Fig. 1 shows the total variation distance $\|\widehat{P}_k - P_k\|_{\text{tv}}$ defined in Equation (1) for the partition and cores generated by *K++*, CoreDpp, CoreDpp-r and CoreDpp-z as *nClust* and the length vary. We see that in general, most approximations improve as ε_{Π} and δ_{Π} shrink. Remarkably, the CoreDpp variants achieve much

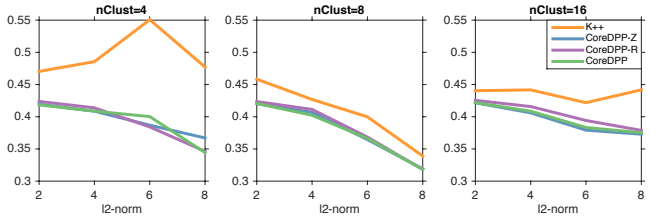

Figure 1: Total variation distances (error) on synthetic data with varying *nClust* and ℓ_2-norm.

lower error than *K++*. Moreover, the results suggest that the relaxations from Section 5.3 do not noticeably increase the error in practice. Also, CoreDpp-r performs comparable with CoreDpp, indicating that our algorithm is robust against initialization. Since, in addition, the CoreDpp construction makes most progress in the first pass through the data, and the kmeans++ initialization yields the best performance, we use only one pass of CoreDpp initialized with kmeans++ in the subsequent experiments.

6.2 Real Data

We apply CoreDpp to two larger real data sets:

1. MNIST [35]. MNIST consists of images of handwritten digits, each of dimension 28×28.
2. GENES [9]. This dataset consists of different genes. Each sample in GENES corresponds to a gene, and the features are shortest path distances to 330 different hubs in the BioGRID gene interaction network.

For our first set of experiments on both datasets, we use a subset of 2000 data points and an RBF kernel to construct L. To evaluate the effect of model parameters on performance, we vary M from 20 to 100 and k from 2 to 8 and fix $\nu = 2$ (see Section 6.1 for an explanation of the parameters). Larger-scale experiments on these datasets are reported in Section 6.3.

Performance Measure and Results. On these larger data sets, it becomes impossible to compute the total variation distance exactly. We therefore approximate it by uniform sampling and computing an empirical estimate.

The results in Figure 2 and Figure 3 indicate that the approximations improve as the number of parts M increases and k decreases. This is because increasing M increases the models' approximation power, and decreasing k leads to a simpler target probability distribution to approximate. In general, CoreDpp always achieves lower error than *K++*, and *NysStoch* performs poorly in terms of total variation distance to the original distribution. This phenomenon is perhaps not so surprising when recalling that the Nyström approximation minimizes a different type

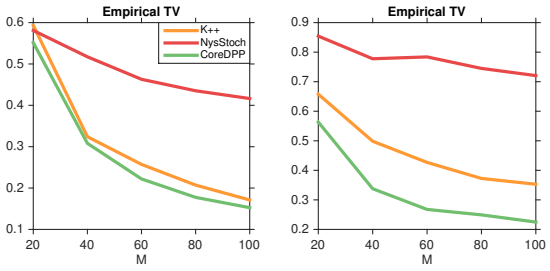

Figure 2: Approximate total variation distances (empirical estimate) on MNIST (left) and GENES (right) with M varying from 20 to 100 and fixed $k = 6$.

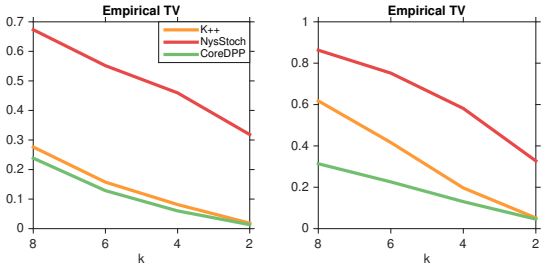

Figure 3: Approximate total variation distances (empirical estimate) on MNIST (left) and GENES (right) with k varying from 8 to 2 and fixed $M = 100$.

of error, a distance between the kernel *matrices*. These observations suggest to be careful when using matrix approximations to approximate L.

For an intuitive illustration, Figure 4 shows a core $\mathcal{C}$ constructed by CoreDpp, and the elements of one part $\mathcal{Y}_c$.

6.3 Running Time on Large Datasets

Lastly, we address running times for CoreDpp, *NysStoch* and the Markov chain k-DPP (*MCDPP* [28]). For the latter, we evaluate convergence via the Gelman and Rubin multiple sequence diagnostic [19]; we run 10 chains simultaneously and use the *CODA* [40] package to calculate the potential scale reduction factor (PSRF), and set the number of iterations to the point when PSRF drops below 1.1. Finally we run *MCDPP* again for this specific number of iterations.

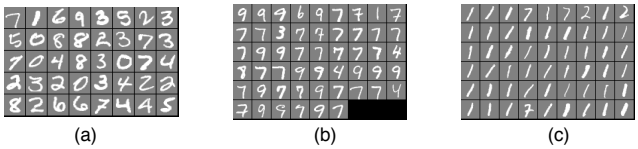

Figure 4: (a) Example coreset $\mathcal{C}$ of size 40, each figure is a core in the coreset constructed by our algorithm; (b,c) Two different parts corresponding to the first and second core.

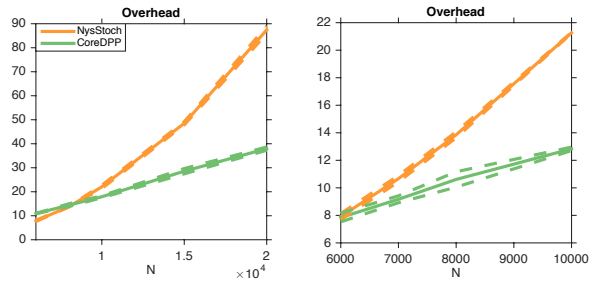

Figure 5: Overhead (setup) time in seconds with varying ground set size (N) on MNIST (left) and GENES (right).

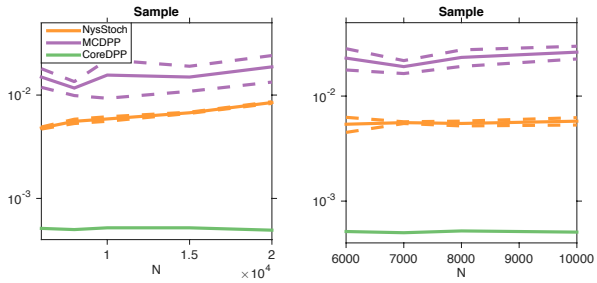

Figure 6: Average time for drawing one sample as the ground set size (N) varies on MNIST (left) and GENES (right). Note that the time axis is shown in log scale.

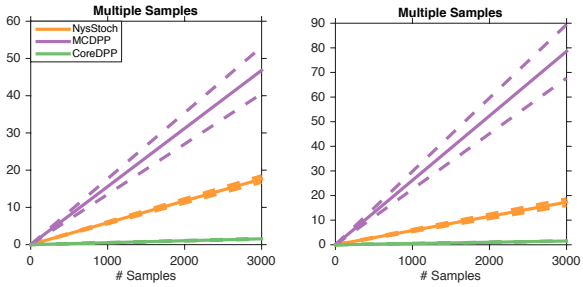

Figure 7: Average time for sampling different numbers of subsets with $N = 5000$, $M = 40$ and $k = 5$ on MNIST (left) and GENES (right).

For overhead time, i.e., time to set up the sampler that is spent once in the beginning, we compare against *NysStoch*: CoreDpp constructs the partition and $\widetilde{L}$, while *NysStoch* selects landmarks and constructs an approximation to the data. For sampling time, we compare against both *NysStoch* and *MCDPP*: CoreDpp uses Algorithm 1, and *NysStoch* uses the dual form of k-Dpp sampling [30]. We did not include the time for convergence diagnostics into the running time of *MCDPP*, giving it an advantage in terms of running time.

Overhead. Fig. 5 shows the overhead times as a function of N. For MNIST we vary N from 6,000 to 20,000 and for GENES we vary N from 6,000 to 10,000. These values of N are already quite large, given that the Dpp kernel is a dense RBF kernel matrix; this leads to increased running time for all compared methods. The construction time for *NysStoch* and CoreDpp is comparable for small-sized data, but *NysStoch* quickly becomes less competitive as the data gets larger. The construction time for CoreDpp is linear in N, with a mild slope. If multiple samples are sought, this construction can be performed offline as preprocessing as it is needed only once.

Sampling. Fig. 6 shows the time to draw one sample as a function of N, comparing CoreDpp against *NysStoch* and *MCDPP*. CoreDpp yields samples in time independent of N and is extremely efficient – it is orders of magnitude faster than *NysStoch* and *MCDPP*.

We also consider the time taken to sample a large number of subsets, and compare against both *NysStoch* and *MCDPP*—the sampling times for drawing approximately independent samples with *MCDPP* add up. Fig. 7 shows the results. As more samples are required, CoreDpp becomes increasingly efficient relative to the other methods.

7 Conclusion

In this paper, we proposed a fast, two-stage sampling method for sampling diverse subsets with k-Dpps. As opposed to other approaches, our algorithm directly aims at minimizing the total variation distance between the approximate and original probability distributions. Our experiments demonstrate the effectiveness and efficiency of our approach: not only does our construction have lower error in total variation distance compared with other methods, it also produces these more accurate samples efficiently, at comparable or faster speed than other methods.

Acknowledgements. This research was partially supposed by an NSF CAREER award 1553284 and a Google Research Award.

References

[1] R. H. Affandi, A. Kulesza, and E. B. Fox. Markov Determinantal Point Processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2012.

[2] R. H. Affandi, E. Fox, and B. Taskar. Approximate inference in continuous determinantal processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[3] R. H. Affandi, A. Kulesza, E. B. Fox, and B. Taskar. Nystrom Approximation for Large-Scale Determinantal Processes. In *Proc. Int. Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

[4] R. H. Affandi, E. B. Fox, R. P. Adams, and B. Taskar. Learning the Parameters of Determinantal Point Process Kernels. In *Int. Conference on Machine Learning (ICML)*, 2014.

[5] A. Agarwal, A. Choromanska, and K. Choromanski. Notes on using determinantal point processes for clustering with applications to text clustering. *arXiv preprint arXiv:1410.6975*, 2014.

[6] N. Anari, S. O. Gharan, and A. Rezaei. Monte Carlo Markov Chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes. *arXiv preprint arXiv:1602.05242*, 2016.

[7] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SIAM-ACM Symposium on Discrete Algorithms (SODA)*, 2007.

[8] O. Bachem, M. Lucic, and A. Krause. Coresets for nonparametric estimation-the case of dp-means. In *Int. Conference on Machine Learning (ICML)*, pages 209–217, 2015.

[9] N. K. Batmanghelich, G. Quon, A. Kulesza, M. Kellis, P. Golland, and L. Bornn. Diversifying sparsity using variational determinantal point processes. *arXiv preprint arXiv:1411.6307*, 2014.

[10] M.-A. Belabbas and P. J. Wolfe. Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, pages 369–374, 2009.

[11] A. Borodin. Determinantal point processes. *arXiv preprint arXiv:0911.1153*, 2009.

[12] A. Borodin and E. M. Rains. Eynard-Mehta Theorem, Schur Process, and Their Pfaffian Analogs. *Journal of Statistical Physics*, 121(3-4):291–317, 2005.

[13] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. Algebraic complexity theory. *Grundlehren der mathematischen Wissenschaften*, 315, 1997.

[14] L. Decreusefond, I. Flint, and K. C. Low. Perfect simulation of determinantal point processes. *arXiv preprint arXiv:1311.1027*, 2013.

[15] A. Deshpande and L. Rademacher. Efficient Volume Sampling for Row/Column Subset Selection. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.

[16] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *SIAM-ACM Symposium on Discrete Algorithms (SODA)*, 2006.

[17] D. Feldman, A. Fiat, and M. Sharir. Coresets for weighted facilities and their applications. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[18] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *SIAM-ACM Symposium on Discrete Algorithms (SODA)*, pages 1434–1453, 2013.

[19] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, pages 457–472, 1992.

[20] J. Gillenwater, A. Kulesza, and B. Taskar. Near-Optimal MAP Inference for Determinantal Point Processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[21] J. Gillenwater, A. Kulesza, and B. Taskar. Discovering diverse and salient threads in document collections. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.

[22] J. Gillenwater, E. Fox, A. Kulesza, and B. Taskar. Expectation-Maximization for Learning Determinantal Point Processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[23] B. Gong, W.-L. Chao, K. Grauman, and S. Fei. Diverse Sequential Subset Selection for Supervised Video Summarization. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[24] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 126–134, 2005.

[25] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Symposium on Theory of Computing (STOC)*, pages 291–300, 2004.

[26] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Symposium on Theory of Computing (STOC)*, pages 291–300, 2004.

[27] J. B. Hough, M. Krishnapur, Y. Peres, B. Virág, et al. Determinantal processes and independence. *Probability Surveys*, 2006.

[28] B. Kang. Fast Determinantal Point Process Sampling with Application to Clustering. *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[29] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[30] A. Kulesza and B. Taskar. Structured Determinantal Point Processes. *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[31] A. Kulesza and B. Taskar. Learning Determinantal Point Processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2011.

[32] A. Kulesza and B. Taskar. k-DPPs: Fixed-Size Determinantal Point Processes. In *Int. Conference on Machine Learning (ICML)*, 2011.

[33] A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.

[34] J. T. Kwok and R. P. Adams. Priors for diversity in generative latent variable models. In *Advances*

in Neural Information Processing Systems (NIPS), 2012.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[36] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Anal. Meeting of the Association for Computational Linguistics (ACL)*, 2011.

[37] A. Magen and A. Zouzias. Near optimal dimensionality reductions that preserve volumes. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, 2008.

[38] Z. Mariet and S. Sra. Learning Determinantal Point Processes. In *Int. Conference on Machine Learning (ICML)*, 2015.

[39] S. Paul. Core-sets for canonical correlation analysis. In *Int. Conference on Information and Knowledge Management (CIKM)*, pages 1887–1890, 2015.

[40] M. Plummer, N. Best, K. Cowles, and K. Vines. Coda: Convergence diagnosis and output analysis for mcmc. *R News*, 6(1):7–11, 2006.

[41] A. Rahimi and B. Recht. Random Features for Large-scale Kernel Machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[42] G. Rosman, M. Volkov, D. Feldman, J. W. Fisher III, and D. Rus. Coresets for k-segmentation of streaming data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 559–567, 2014.

[43] A. Shah and Z. Ghahramani. Determinantal clustering processes - a nonparametric bayesian approach to kernel based semi-supervised clustering. *arXiv preprint arXiv:1309.6862*, 2013.

[44] J. Snoek, R. Zemel, and R. P. Adams. A Determinantal Point Process Latent Variable Model for Inhibition in Neural Spiking Data. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[45] S. Wang, C. Zhang, H. Qian, and Z. Zhang. Using The Matrix Ridge Approximation to Speedup Determinantal Point Processes Sampling Algorithms. In *Proc. AAAI Conference on Artificial Intelligence*, 2014.

[46] Z. Zhang. The Matrix Ridge Approximation: Algorithms. *Machine Learning*, 97:227–258, 2014.

[47] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107 (10):4511–4515, 2010.