Submodular Point Processes with Applications to Machine Learning: Extended Version

Rishabh Iyer
University of Washington, Seattle
rkiyer@u.washington.edu

Jeff Bilmes
University of Washington, Seattle
bilmes@u.washington.edu

Abstract

We introduce a class of discrete point processes that we call the *Submodular Point Processes (SPPs)*. These processes are characterized via a submodular (or supermodular) function, and naturally model notions of *information, coverage* and *diversity*, as well as *cooperation*. Unlike Log-submodular and Log-supermodular distributions (Log-SPPs) such as determinantal point processes (DPPs), SPPs are themselves submodular (or supermodular). In this paper, we analyze the computational complexity of probabilistic inference in SPPs. We show that computing the partition function for SPPs (and Log-SPPs), requires exponential complexity in the worst case, and also provide algorithms which approximate SPPs up to polynomial factors. Moreover, for several subclasses of interesting submodular functions that occur in applications, we show how we can provide efficient closed form expressions for the partition functions, and thereby marginals and conditional distributions. We also show how SPPs are closed under mixtures, thus enabling maximum likelihood based strategies for learning mixtures of submodular functions. Finally, we argue how SPPs complement existing Log-SPP distributions, and are a natural model for several applications.

1 Introduction

Submodular functions provide a rich class of expressible models for a variety of machine learning problems. Submodular functions occur naturally for two purposes: In minimization problems, they model notions of cooperation, attractive potentials, and economies of scale [29, 24, 2] while in maximization problems, they model aspects of coverage, diversity, and information [40, 52]. A set function $f : 2^V \to \mathbb{R}$ is submodular if $\forall S, T \subseteq V,\ f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. An equivalent characterization is the "diminishing returns" property, which says that for $S \subseteq T$ and $j \notin T, f(S \cup j) - f(S) \geq f(T \cup j) - f(T)$. Submodular functions have properties that make their exact or approximate optimization efficient and often practical.

While significant research has gone into providing optimal and near optimal algorithms for various forms of submodular optimization problems [15, 23, 26, 46, 53], limited work has investigated submodular functions from a probabilistic perspective. Most research has focused on a special class of Log-Submodular and Log-Supermodular distributions, namely pairwise Markov Random Fields (also called Ising models) [16] and Determinantal Point Processes [37, 44]. Recently, [10] investigate the general class of Log-Submodular distributions, and provide algorithms for approximate probabilistic inference. In this paper, we make attempts to model submodular functions as probabilistic point processes, which we call the "Submodular Point Processes" (SPP). These distributions are defined via a non-negative submodular (supermodular) function as:

$$P(X) \propto f(X), \text{ for } X \subseteq V, \tag{1}$$

where f is a submodular (or supermodular) function defined so that when normalized, $P(X)$ is a valid distribution. A related but different class of distributions is the Log-submodular (or Log-supermodular) distributions, which we call Log-SPPs. These [10] are defined as:

$$P(X) \propto \exp(f(X)). \tag{2}$$

where f is submodular (or supermodular). Determinantal Point Processes (DPPs) [37, 44] are special cases of Log-submodular distributions, while Ising models are special cases of Log-supermodular distributions.

2 Submodular Point Processes

Discrete Point processes have in general widely been studied in mathematics and statistics [28]. Simply put[1], a finite discrete point process $\mathcal{P}$ over a set $V =$

Appearing in Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38. Copyright 2015 by the authors.

[1] In this paper, we use the term *point process* in a limited finite sense rather than the much more general sense of [28, 8].

$\{1, 2, \cdots, n\}$ is defined as a probability distribution over 2^V.

Existing Work: An important class of discrete point processes are the Log-Submodular and Log-Supermodular distributions: $\mathcal{P}(X) \propto \exp(f(X))$, where f is submodular (or supermodular). This class of distributions was recently studied in the general context in [10], where they provide algorithms for approximate probabilistic inference. A special case of these are the Determinantal Point Processes [37, 17, 44], which have gained much attention in machine learning: $\mathcal{P}(Y) = \det(L_A)$ for some positive semidefinite $n \times n$ matrix L, with L_A as the submatrix induced by the rows and columns in A. DPPs are Log-submodular, since $\log \mathcal{P}(Y) = \log \det(L_A)$ is a submodular set function in A. Another class of distributions, which also naturally occur in practice, are pairwise distributions induced via a Markov random field on a grid graph [16], which have been used extensively [2] in computer vision and in particular in image segmentation. The model in this context is $p(X) \propto \exp(-\phi(X))$ for a certain function ϕ. Under certain *regularity* assumptions, the function ϕ is submodular in X, and $p(X)$ is then Log-supermodular. Hence for this class of distributions, MAP inference can be efficiently solved in polynomial time by submodular function minimization [15]. Other inference problems like normalization and marginalization can be approximated efficiently in certain special cases [30]. A related class of models are log-linear models of the form $p(X) \propto \exp(-\lambda^\intercal T(X))$ for a linear function $\lambda^\intercal T(X)$ of statistics $T(X)$.

In this paper, we define a new class of probability distributions via a non-negative submodular (or supermodular) function that we call the submodular (or supermodular) point process. Given a non-negative submodular (or supermodular) function f, define $p(X) \propto f(X)$. We extensively study the properties of this class of distribution by characterizing the hardness and approximation factors of probabilistic inference, and also investigating several useful subclasses of SPPs which exhibit exact algorithms for inference.

Given a constraint set $X \in \mathcal{C} \subseteq 2^V$, the generalized partition function of SPPs and Log-SPPs can be defined, with $A \subseteq B$, as $Z^{\mathcal{C}}_f(A, B) = \sum_{A \subseteq X \subseteq B, X \in \mathcal{C}} f(X)$ for SPPs, and $Z^{\mathcal{C}}_{e^f}(A, B) = \sum_{A \subseteq X \subseteq B, X \in \mathcal{C}} \exp(f(X))$, for Log-SPPs. $\mathcal{C}$ represents a particular combinatorial structure of interest, and could be for example a size constraint (all sets of size k), matchings in a graphs, s-t cuts, etc. An SPP then becomes any distribution that is representable as follows: $\mathcal{P}^{\mathcal{C}}_f(Y) = \frac{f(Y)}{Z^{\mathcal{C}}_f(\emptyset, V)}$. Similarly, Log-SPPs are, $\mathcal{P}^{\mathcal{C}}_f(Y) = \frac{exp(f(Y))}{Z^{\mathcal{C}}_{e^f}(\emptyset, V)}$ [10]. A necessary condition for computing the partition function, along with marginals and conditionals, is that $Z^{\mathcal{C}}_f(A, B)$ can be computed exactly or approximately for any given sets $A \subseteq B$ and given constraints $\mathcal{C}$. In the current paper, we mostly restrict ourselves either to the unconstrained setting, $\mathcal{C} = 2^V$, or to cardinality constraints, $\mathcal{C} = \{X : |X| = k\}$. We denote the corresponding normalization quantities as $Z_f(A, B)$ for the unconstrained case, and $Z^k_f(A, B)$ for the cardinality constraints $\{X : |X| = k\}$. We later relax this, and consider more general matroid and knapsack constraints. The condition that $Z^{\mathcal{C}}_f(A, B)$ be computable in polynomial time for every set A, B seems very strong at first, since it involves a sum over exponential number of terms. In the following section, we show that we can approximately compute $Z^{\mathcal{C}}_f(A, B)$, for any submodular function and a number of useful constraints $\mathcal{C}$. We also show that we can compute the partition function **exactly** for several subclasses of submodular functions that often occur in applications. While we define these distributions via submodular functions (and for maximization applications), we show how they can be easily extended to supermodular functions for minimization problems.

SPPs may model aspects of coverage, diversity, and information (via a submodular function) or aspects of cooperation, attraction (via a supermodular function). Diversity and coverage is important in subset selection problems like document summarization [40, 39], speech data subset selection [43], and image summarization [52, 47], and diverse web and image search [20, 36]. Similarly, one often wishes to capture cooperation in problems like image segmentation and denoising [2, 29].

In this paper, we argue how SPPs and Log-SPPs have properties complementary to each other, and show how several subclasses of SPPs have efficiently computable partition functions.

The following are the main contributions of this paper: **1)** We investigate the hardness of computing the partition function for SPPs and Log-SPPs. In particular, we show that exact computation of the normalization constants for SPPs and Log-SPPs, could require exponential complexity in the worst case (independent of P v/s NP). **2)** We show that the Log Partition function of SPPs can be approximated within $O(\log n)$, and contrast this with the corresponding results of Log-SPPs [10], where the worst case factor is $O(n)$. **3)** We then investigate several subclasses of useful submodular functions and show how the partition function can be computed exactly for several of these subclasses. We show how these results for computing the normalization constant extend to computing marginals, conditional distributions, and other probabilistic inference problems. **4)** We investigate the problem of learning mixtures of submodular functions, and show how SPPs and Log-SPPs both provide natural maximum-likelihood learning strategies for this problem. **5)** Finally, we argue that while the SPPs are similar to the Log-SPP models from a modeling perspective, they have several key differences, thereby providing a complementary class of models. In particular, we argue that SPPs are natural for modeling submodular mixtures, and have quite different concentration-of-diversity properties than Log-SPPs. This paper is almost entirely theoretical, but we have implemented SPPs and offer some empirical observations in Section 5.

3 Probabilistic Inference

We here investigate the computation of the partition function for SPPs, the conditionals, and the marginals. We first investigate the hardness of probabilistic inference, and provide approximation algorithms for computing these for general SPPs. We contrast these with the corresponding guarantees and hardness results for Log-SPPs, and show how this problem is significantly harder in the context of Log-SPPs as opposed to SPPs. We then consider several subclasses of SPPs and show how the partition function can be computed either exactly, or up to a factor of $1 + \epsilon$, for these subclasses. Finally, we show how these result in algorithms for computing marginals, conditionals, sampling, and learning mixtures of submodular functions.

3.1 Hardness And Approximation Factors

In this section, we provide hardness results and worst case approximation factors for the general classes of SPPs and Log-SPPs. In the case of Log-SPPs, the worst case approximation factors are provided in [10]. In terms of hardness, the partition function computation was known to be $\#P$ hard [30]. In the current paper, we show that the partition function computation is provably exponential for both SPPs and Log-SPPs, in the worst case. We also provide the worst case approximation factor for SPPs, and show that the log-partition function can be approximated within a factor of $O(\log n)$, which is in contrast to the approximation factor for Log-SPPs shown in [10] and is $O(n)$.

Denote Z_f as the true partition function, and $\hat{Z}_f$ as the approximate partition function. We define the approximation factor of the Log-Partition function as $\alpha = |\log Z_f - \log \hat{Z}_f| = |\log \frac{Z_f}{\hat{Z}_f}|$.

The approximation factors for the general class were provided in [10] where they show that submodular sub- and super-gradients [26] provide lower and upper bounds on the partition function. In particular, the semigradients yield, in polynomial time, $\hat{Z}^u_f, \hat{Z}^l_f$ such that $\hat{Z}^l_f \leq Z_f \leq \hat{Z}^u_f$, where Z_f is the partition function of f. Here we offer a new result showing that computing the partition function has provably exponential cost.

Lemma 1. *There exists a submodular (or supermodular) function f, such that computing the partition function of $\mathcal{P}(X) \propto \exp(f(X))$ requires exponential complexity (independent of the $P \neq NP$ question).*

Proof. The information-theoretic proof uses a construction and argumentation similar to that in [18, 51].

Define two monotone submodular functions $h(X) = \min\{|X|, \alpha\}$ and $f^R(X) = \min\{\beta + |X \cap \bar{R}|, |X \cap R|, \alpha\}$, where $R \subseteq V$ is a random set of cardinality α. Let α and β be an integer such that $\alpha = x\sqrt{n}/5$ and $\beta = x^2/5$ for an $x^2 = \omega(\log n)$. Using a Chernoff bound, one can then show that any algorithm that uses a polynomial number of queries can distinguish h and f^R with probability only $n^{-\omega(1)}$, and therefore cannot reliably distinguish the functions with a polynomial number of queries [51]. Moreover, note that $h(X) \geq f^R(X), \forall X \subseteq V$. Moreover, $f^R(R) = \beta < \alpha = h(R)$, and hence $Z_h > Z_{f^R}$.

Note that $Z_h = \sum_{X \subseteq V} \exp(h(X))$ has a closed form expression as,

$$Z_h = \sum_{i=1}^{\alpha} \binom{n}{i} \exp(i) + \sum_{i=\alpha+1}^{n} \binom{n}{i} \exp(\alpha) \qquad (3)$$

Any poly-time algorithm, which can compute the partition function exactly, will distinguish f and h^R, since one can check if $Z_h > Z_{f^R}$ (since Z_h has a closed form expression). Hence the partition function cannot be exactly computed for the general class of submodular functions. The same proof technique for Log-supermodular functions, just replacing $\exp(f(X))$ with $\exp(-f(X))$. $\square$

Next, we study the hardness of SPPs.

Lemma 2. *There exists a submodular (or supermodular) function f, such that computing the partition function of $\mathcal{P}(X) \propto f(X)$ requires exponential complexity (independent of the $P \neq NP$ question).*

Proof. The proof of this Lemma follows with the same proof technique as the previous Lemma on the hardness of Log-SPPs. $\square$

Similar to Log-SPPs, we can use the sub and super-gradients to provide upper and lower bounds for the partition function. The sub/super gradients provide, for any $X \subseteq V$, modular functions $m^X_l(Y) + c_l$ and $m^X_u(Y) + c_u$ such that $m^X_l(Y) + c_l \leq f(Y) \leq m^X_u(Y) + c_u$ and that are tight at $Y = X$. Moreover, submodular functions also admit tighter approximations via non-modular functions. For example, the class of coverage functions (equivalently concave over modular functions) approximates the class of monotone submodular functions up to a factor of $O(\sqrt{n})$, which is the tightest possible bound for the general class of submodular functions. The main idea of the algorithm for computing an approximate partition function is to compute an approximation $\hat{f}(X)$ of $f(X)$, such that $\hat{f}(X) \leq f(X) \leq \alpha\hat{f}(X), \forall X \subseteq V$. Then, define

$$\hat{Z}_f = \sum_{X \subseteq V} \hat{f}(X). \qquad (4)$$

The following lemma shows that this approximation results in an approximation factor of $O(\log \alpha)$ for the log partition function:

Lemma 3. *Given a submodular function f, and an approximation $\hat{f}$, such that $\hat{f}(X) \leq f(X) \leq \alpha\hat{f}(X), \forall X \subseteq V$, it holds that $\hat{Z}_f \leq Z_f \leq \alpha\hat{Z}_f$. Moreover, $|\log \hat{Z}_f - \log Z_f| \leq \log \alpha$.*

Using the Lemma above, we can compute the approximation guarantees for the log partition function.

Theorem 1. *Given a submodular function f, there exists a poly-time algorithm which computes an approximation $\hat{Z}_f$ of the partition function Z_f of the distribution $\mathcal{P}(X) \propto f(X)$, such that $|\log \hat{Z}_f - \log Z_f| \leq O(\log n)$.*

Proof. The proof of the above result relies on the following facts, and Lemma 3. For a monotone submodular function, the sub and supergradients, approximate the submodular function up to a factor of $O(n)$ [25], implying a $O(\log n)$ approximation guarantee. Furthermore, a coverage function [9, 18] approximates a monotone submodular function within a factor of $O(\sqrt{n})$ [9, 18], which again provides a $O(\log n)$ approximation guarantee to the log-partition function.[2] We as shall see later, the partition function can exactly be computed for the coverage functions. Finally, general non-monotone submodular functions can be approximated within a factor of $O(n^2/4)$ by directed graph-cut functions. Since the partition function of directed graph-cut functions can also be exactly computed (see the next section), we can provide a multiplicative approximation factor of $O(n^2/4)$, which again provides an approximation factor of $O(\log n)$. $\square$

Note, k-SPPs (which are analogous to k-DPPs in [37] but for SPPs) have same approximation guarantee.

3.2 Subclasses Of SPPs

In this section, we investigate several subclasses of submodular functions, and show, surprisingly, how probabilistic inference is exact for certain of these functions, independent of the underlying tree-width [34] of the function. Note that the tree-width is, in general, the complexity parameter of exact inference for graphical models — a graphical model known to have tree-width k is such that inference is possible exponential in k, so for example inference on trees is very efficient. The section here indicates an analogous situation for SPPs, namely that certain traits may exist that allow for inference in SPPs to be done exactly in polynomial time.

3.2.1 Graph Based Submodular Functions

A number of submodular functions are graph based functions, defined on a graph $G = (V, E)$, with $|V| = n$ and E denoting the objects that interact. The submodular functions are typically parameterized by a kernel L which represents the pairwise interactions between objects. We denote $s_{ij} = L(i, j)$, which represents the similarity between item i and j. In the context of document summarization, this could represent the similarity between sentences. Similarly, in image summarization this would be the similarity between images. These matrices are often symmetric, where $s_{ij} = s_{ji}$, which is true in most applications so we assume this in the below. We also assume, with no loss of generality, that the similarities are normalized (i.e., $0 \leq s_{ij} \leq 1$).

[2]While the guarantee for the log-partition function is the same order, the multiplicative guarantee of the partition function is $O(\sqrt{n})$, which is tighter than the sub/supergradient approximations which is $O(n)$.

Facility Location and its generalizations: Given a similarity matrix $\{s_{ij}\}_{i,j \in V}$ the facility location function is $f(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$. This function has successfully been used in document summarization [39], image summarization [47] and data subset selection [43]. Denote $\mathcal{P}_{fac}(Y)$ as the corresponding point process, with, $\mathcal{P}_{fac}(Y) \propto \sum_{i \in V} \max_{j \in Y} s_{ij}$. This function is monotone submodular, since it models coverage. The normalization constants Z_{fac} of the facility location can be computed efficiently.

$$Z_{fac} = \sum_{i \in V} \sum_{l=1}^{n} 2^{l-1} s_{ij_i^l}, \tag{5}$$

$$Z_{fac}^k = \sum_{i \in V} \sum_{l=1}^{k} \binom{n-l}{k-1} s_{ij_i^{n-l+1}}, \tag{6}$$

We can also generalize this to the k-facility location case [42], where instead of a single max, we take the k-best maximum.

Graph Cut and Generalizations: This class of functions have been used extensively both in summarization problems (modeling coverage and diversity [43, 39]) as well in image segmentation and denoising (by capturing cooperation [2]). This general class can be defined as: $f(X) = M + \lambda \sum_{i \in V} \sum_{j \in X} s_{ij} - \mu \sum_{i,j \in X} s_{ij}$; $\mu = \lambda = 1, M = 0$ is the standard graph cut, and $\lambda = 0$ gives the redundancy penalty [43]. $M \geq 0$ is just a factor to ensure that $f(X) \geq 0$.

Notice that the similarity penalty models diversity in a manner very similar to the DPPs. Also note that the redundancy penalty can be used with any submodular function capturing coverage (like facility location or asymmetric graph cut) to define an objective for summarization. This has been used, for example, with the facility location and asymmetric graph cut [47, 43, 20]). Define,

$$\mathcal{P}_{gc}(X) \propto M + \lambda \sum_{i \in V} \sum_{j \in X} s_{ij} - \mu \sum_{i,j \in X} s_{ij}, \tag{7}$$

where λ, μ, M are appropriately chosen so that the objective is non-negative. This function is monotone for $\lambda > 2\mu$. This immediately provides an expression for Z_f and Z_f^k. Define $S = \sum_{i,j \in V} s_{ij}, S^d = \sum_{i \in V} s_{ii}$. Then,

$$Z_f = 2^n M + (2\lambda - \mu) 2^{n-2} S - 2^{n-2} \mu S^d,$$

$$Z_f^k = M \binom{n}{k} + \lambda \binom{n-1}{k-1} S - \mu \binom{n-2}{k-2} S - \mu \binom{n-2}{k-1} S^d$$

This class of point processes can thus be normalized in $O(n^2)$.

Saturated Coverage Function: The saturated coverage function, $f(X) = \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \alpha_i\}$, has successfully been used in document summarization [40]. Instead of average coverage (like the graph cut type functions), or the maximum coverage (which is the facility location), this function chooses a certain fraction of coverage for every item. We can define the corresponding point process $\mathcal{P}_{sc}(Y) \propto \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \alpha_i\}$. Unlike the graph-cut and facility location, the normalization constant for this one is hard to obtain in polynomial time, since it involves knapsack counting, which is #P complete [32]. Fortunately, it can be approximated to an arbitrary factor close to one, by using an fully polynomial time approximation scheme (FPTAS) for knapsack counting.

3.2.2 Coverage Functions

Set Cover: One can define a submodular function via "concepts", and assume that each object covers a set of concepts. Hence, given a set S, $\Gamma(S)$ denotes the set of concepts covered by S. Let V be the set of all items and W be the set of all concepts, so $\forall S \subseteq V, \Gamma(S) \subseteq W$. Given a modular function $c : 2^W \to \mathbb{R}_+$, the set cover function is defined as $f_{cov}(S) = c(\Gamma(S))$. This function simultaneously models aspects of coverage [38] in maximization, and the notion of complexity (like the size of the vocabulary in a speech corpus) in minimization problems [41]. We can also define an inverse map, Γ^{-1} such that for every $w \in W$, $\Gamma^{-1}(w)$ denotes the set of elements $v \in V$ such that $\Gamma(v) \ni w$. Since this is a monotone non-negative submodular function, we can define a distribution, $\mathcal{P}_{cov}(Y) \propto c(\Gamma(Y))$. The normalization factors Z_f and Z_f^k are:

$$Z_{cov} = \sum_{w \in W} c_w [2^n - 2^{n - |\Gamma^{-1}(w)|}], \tag{8}$$

$$Z_{cov}^k = \sum_{w \in W} c_w [\binom{n}{k} - \binom{n - |\Gamma^{-1}(w)|}{k}] \tag{9}$$

Probabilistic Coverage Functions: This is a generalization of the set cover function, which has been used in a number of models for summarization problems [11]. This provides a probabilistic notion to the set cover function, and is defined as $f(X) = \sum_{i \in \mathcal{U}} w_i [1 - \prod_{j \in X}(1 - p_{ij})]$ where $\mathcal{U}$ is some set (e.g., of features). The normalization factor of this class of functions can be obtained as $Z_f = \sum_{i \in \mathcal{U}} w_i [2^n - \prod_{j \in V}(2 - p_{ij})]$. One can similarly obtain Z_f^k.

3.2.3 Independent Distributions

Modular Functions: The simplest class of set functions is a modular function $f(X) = \sum_{i \in X} m_i$. The items in the set do not interact with each other. The normalization constant for this class of distributions is $Z_f = 2^{n-1} m(V)$ and $Z_f^k = \binom{n-1}{k-1} m(V)$.

Log-Modular & Product Bernoulli distributions: Log Modular distributions are defined with $f(X) = e^{-m(X)}$ which is supermodular and log-modular. The normalization constant is, $Z_f = \prod_{i \in V}[1 + e^{-m_j}]$. A related class of distributions is the Product of independent Bernoulli distribution, where we independently sample each $j \in V$ with a probability p_j. The resulting distribution is $f(X) = \prod_{i \in X} p_i \prod_{j \notin X}(1 - p_j)$ which is submodular, also log-modular, and is already a probability distribution (i.e., $Z_f = 1$).

3.2.4 Concave over modular Functions

A general class of submodular functions is sums of concave over modular. Given modular functions m_i and concave functions ψ_i, we can define a submodular function: $f_{CM}(X) = \sum_{i=1}^M w_i \psi(m_i(X))$. They appear in maximization problems as feature based functions, defined as $f(X) = \sum_{e \in \mathcal{F}} \psi(m_e(X))$ (where $|\mathcal{F}| = M$), and have been used in data subset selection applications [54, 33]. $m_e(j)$ captures how much item j covers feature $\mathcal{F}$. Another related function is $f(X) = \sum_{j=1}^M \psi(m_j(X \cap C_M))$, where $C_1, C_2, \cdots, C_M$ are clusters of similar items in the ground set V. This function simultaneously captures diversity in maximization problems [40], and notions of cooperation in minimization problems [29, 22]. Moreover, the saturated coverage function discussed above is also a special case of this class of functions.

Similar to the saturated coverage function, we expect that computing the exact normalization constant is #P complete. However, we can approximate it using ideas similar to the saturated coverage function. First, we restrict our attention to sums of piecewise linear concave over modular functions. These functions have finite number of breakpoints, and the function is modular within each piece. Hence, one can use knapsack counting within each component of the modular function, and approximately compute the normalization constant up to a factor of $1 + \epsilon$ [49]. Moreover, since it is possible to approximate any concave function with a truncation up to any desired factor [35], one can extend this result to general sums of concave over modular functions.

While the FPTAS for knapsack counting gives an FPTAS for sums of concave over modular functions, the resulting algorithm can be quite computationally expensive. A much simpler approximation can be used for functions which can be expressed as $f(X) = \sum_{i=1}^M [m_i(X)]^a$, where $a \in (0, 1]$. It is known that the function $\hat{f}(X) = \sum_{i=1}^M \sum_{j \in X} [m_i(j)]^a$ approximates f up to a factor of $O(|X|^{1-a})$ [25]. Since $\hat{f}$ is a modular function, and following Lemma 3, it is easy to see that the resulting approximation factor is $(1 - a)\log n$.

3.2.5 Sparse Pseudo-Boolean functions

For graphical models, in particular in computer vision, set functions are often written as polynomials [21]. Any set function can be written as a polynomial, $p_f(x) = \sum_{T \subseteq V} \alpha_T \prod_{i \in T} x_i$, where $x \in \{0,1\}^n$ is the characteristic vector of a set. In other words, $f(S) = \sum_{T \subseteq S} \alpha_T$. Submodular functions are a subclass of these polynomials. Often the polynomial is *sparse*, i.e., has few nonzero coefficients α_T. This is the case for graph cut like functions above and for the functions considered in [50, 21]. The partition function in this case is $Z_f = 2^n p_f(\mathbf{1}/2)$. The reason for this is that the pseudo-Boolean representation is exactly the multilinear extension of the submodular function corresponding to f. Furthermore, the multilinear extension $F(x) = \sum_{X \subseteq V} f(X) \prod_{i \in X} x_i \prod_{i \notin X}(1 - x_i)$ is closely related to the partition function $F(\mathbf{1}/2) = \sum_{X \subseteq V} f(X)/2^n = Z_f/2^n$.

3.2.6 Fourier Sparse Submodular Functions

A class of set functions introduced in [50] – given a set function f, its Fourier transform is $\hat{f}(B) = \frac{1}{2^n} \sum_{A \subseteq V} f(A)\psi_B(A)$, where $\psi_B(A) = (-1)^{|B \cap A|}$. Given $\hat{f}(B)$, the inverse Fourier transform recovers $f(A)$ as $f(A) = \sum_{B \subseteq V} \hat{f}(B)\psi_B(A)$. Fourier sparse submodular functions are functions where $\text{supp}(f) = \{B \subseteq V : \hat{f}(B) \neq 0\}$ is polynomial in n. In this case, we can evaluate the partition function as $Z_f = \sum_{B \in \text{supp}(f)} \hat{f}(B) \sum_{A \subseteq V} \psi_B(A)$ and since $\sum_{A \subseteq V} \psi_B(A) = \sum_{i=0}^{|B|} (-1)^i \binom{|B|}{i} 2^{n-|B|}$, we may evaluate the partition function in closed form.

3.3 Mixtures of Submodular Point Processes

Often it is desirable to consider not just one submodular function, but a mixture of many submodular component functions [42], i.e., $f(X) = \sum_{i=1}^m w_i f_i(X)$, where f_i's are component submodular functions (any of the aforementioned instances), and w_i's are weights. Given the normalization constants for the individual f_i's, we can easily obtain the resulting expressions for $Z_f = \sum_{i=1}^m w_i Z_{f_i}$ and $Z_f^k = \sum_{i=1}^m w_i Z_{f_i}^k$. Soon, we argue that SPPs are natural for handling mixtures.

3.4 Supermodular Point Processes

Most of the examples we investigated so far are submodular functions, occurring in maximization problems where we want to model notions of coverage and diversity. We may want to model similarity and cooperation for which supermodularity is natural. Given a submodular function f, we may easily define a supermodular point process via a supermodular function $M - f(X)$, where $M \geq \max_{X \subseteq V} f(X)$ is a positive number ensuring the distribution is non-negative. If f is a SPP as a submodular function, the $M - f$ is a SPP as a supermodular function.

3.5 Difference of Submodular Point Processes

Given a submodular SPP f and a supermodular SPP g, we can combine both to obtain $\lambda_1 f(X) + \lambda_2 g(X), \lambda_1, \lambda_2 \geq 0$, which is a difference of submodular functions. These functions occur naturally in problems where one wants to simultaneously model coverage/ diversity and attraction or cooperation [22, 31].

3.6 Extensions to more General Constraints

Applications sometimes require richer constraints, some common examples of which this section investigates.

Knapsack Constraints: Knapsack constraints are commonly used in, e.g., document summarization, where we require sentence sets satisfying a character or word budget constraint. Knapsack constraints involve counting problems which are often #P complete, easily seen by noting that the knapsack counting problem is exactly computing the partition function, under a knapsack constraint, for the distribution obtained by setting $f(X) = 1$. For a general submodular function, we can use the sub/supergradient based approximation, and the problem then requires computing the normalization constant of a modular function $m(X)$, subject to a knapsack constraint $w(X) \leq c$. This can be computed easily via knapsack counting [32]. Denote $\mathcal{N}_i = |\{X \subseteq V \backslash i : w(X) \leq c - w_i\}|$. The normalization constant is $Z_f = \sum_{i \in V} m_i \mathcal{N}_i$. Since each of the $\mathcal{N}_i$'s can be computed via knapsack counting up to a factor of $1 + \epsilon$, the normalization factor can also be computed up to a factor of $1 + \epsilon$.

Matroid Constraints: Matroid independent sets offer another class of constraints useful in applications. An important class of sampling algorithms use stratified sampling [6], where one partitions the set of items into strata, and independently samples within each block. Interestingly, the stratified sampling scheme can be seen as an product Bernoulli SPP under a partition matroid constraint. One can generalize this to other submodular functions under a partition matroid. In particular, denote a partition matroid as $\mathcal{C} = \{X : |X \cap P_i| \leq k_i, \forall i = 1, 2, \cdots, p\}$. It is easy to see that $Z^{\mathcal{C}}(A, B)$, can be expressed as $Z^{\mathcal{C}}(A, B) = \sum_{i=1}^p \sum_{k=1}^{k_i} Z^k(A \cap P_i, B \cap P_i)$. For the general class of matroid constraints, we can again use the sub/super gradient approximation $m(X)$, and compute the approximate partition function of $m(X) : X \in \mathcal{C}$. The partition function in this setting is $Z_f = \sum_{i \in V} m_i \mathcal{N}_i$, where $\mathcal{N}_i = |\{X \subseteq V \backslash i : X \cup \{i\}$ is independent in the matroid$\}|$.

3.7 Efficient Algorithms for Probabilistic Inference

We analyze the properties of SPPs regarding probabilistic inference and learning. We show how efficient computation of the generalized partition function leads to efficient probabilistic inference, sampling, and learning.

Normalization: Notice that the normalization factors can be easily obtained from the expressions of $Z_f^{\mathcal{C}}(A, B)$. In particular, $Z_f^{\mathcal{C}} = \sum_{X \subseteq V, X \in \mathcal{C}} f(X) = Z_f^{\mathcal{C}}(\emptyset, V)$.

Marginalization: We can similarly compute various marginals efficiently. It's not hard to see that,

$$\mathcal{P}_f^{\mathcal{C}}(A \subseteq Y) = \sum_{A \subseteq Y} P_f^{\mathcal{C}}(Y) = \frac{Z_f^{\mathcal{C}}(A, V)}{Z_f^{\mathcal{C}}}. \quad (10)$$

The singleton marginals $\mathcal{P}_f(i \in Y) = \frac{Z_f(i,V)}{Z_f(\emptyset,V)}$ provide interesting intuition about these functions. For example, consider the generalized graph-cut function. Denote the coverage of the object u as $S^u = \sum_{i \in V} s_{iu}$. The marginal probabilities $\mathcal{P}(i \in Y)$ takes a very intuitive form: $\mathcal{P}_{gc}(u \in Y) = \frac{1}{2} + \frac{(\lambda-\mu)S^u}{4M+(2\lambda-\mu)S-\mu S^d}$. Clearly we see that this probability is proportional to the coverage of the function (when $\lambda \geq \mu$). The marginal probability $\mathcal{P}_{gc}(\{i,j\} \in Y)$ is also similarly enlightening. We define here $S^{uv} = s_{uu} + s_{vv} + s_{uv}$. Then, $\mathcal{P}_{gc}(\{u,v\} \in Y) = \frac{1}{4} + \frac{(w-\lambda)(S^u+S^v)-\lambda S^{uv}}{8M+(4w-2\lambda)S-2\lambda S^d}$. The above expression clearly conveys the trade-off between coverage and diversity. In particular, the term $S^u + S^v$ captures the coverage of the objects u and v. Similarly S^{uv} captures the redundancy. The more diverse (less similar) u and v become, the greater the probability of them co-occurring.

Conditional Distributions: We can also define conditional models $\mathcal{P}_f(Y = C|E)$ given different events E. For example, we might be interested in the conditional distribution, given that $A \subseteq Y \subseteq B$. This can be easily characterized in an SPP:

$$\mathcal{P}_f^{\mathcal{C}}(Y = C|A \subseteq Y \subseteq B) = \frac{\mathcal{P}_f^{\mathcal{C}}(Y = C)}{\mathcal{P}_f^{\mathcal{C}}(A \subseteq Y \subseteq B)} = \frac{f(C)}{Z_f^{\mathcal{C}}(A, B)}$$

We can also extend this to compute marginals of these conditional distributions.

Expectations: Another interesting quantity we analyze is $\mathbb{E}(|Y|)$ and $\text{Var}(|Y|)$. Notice that $\mathcal{P}_f(|Y|) = \frac{Z_f^k}{Z_f}$. Hence, these expressions take the form:

$$\mathbb{E}(|Y|) = \sum_{k=1}^{n} k\frac{Z_f^k}{Z_f}, \ \text{Var}(|Y|) = \sum_{k=1}^{n} k^2\frac{Z_f^k}{Z_f} - \mathbb{E}^2(|Y|)$$

These expressions also take very intuitive forms. For example, consider the generalized graph cut functions. Then, $\mathbb{E}(|Y|) = \frac{n}{2} + \frac{(\lambda-\mu)S}{4M+(2\lambda-\mu)S-\mu S^d}$ and $\text{Var}(|Y|) = \frac{n}{4} - \frac{\mu(S-S^d)}{8M+2S(2\lambda-\mu)-2\mu S^d} - \frac{(\lambda-\mu)^2S^2}{(4M+(2\lambda-\mu)S-\mu S^d)^2}$. Hence, as $\lambda - \mu$ gets larger, more coverage occurs.

Sampling: The interesting structural properties of SPPs entail very simple algorithms for exact and approximate sampling a set Y distributed according to a SPP.

MAP Inference: Finally we consider the problem of finding the mode of the distribution $\mathcal{P}_f$. When f is submodular, the MAP inference problem corresponds to submodular maximization $\text{argmax}_{X \in \mathcal{C}} f(X)$. Though this problem is NP hard in general, a large class of optimization algorithms provide near optimal solutions. Many of these algorithms are combinatorial, like the greedy and local search based algorithms [26, 46, 3]. Another class of algorithms deal with the continuous extension of a submodular function, and use relaxation methods to round the continuous solution to a discrete set [5, 12, 4]. In general, the multilinear extension requires an exponential sum, and hence can only approximately be computed through sampling. Surprisingly, many of the submodular functions which characterize SPPs have an efficient form for the multilinear extension [27]. In fact, there is a close connection between the multilinear extension and the normalization factor – the normalization factor satisfies $Z_f = 2^n F(\mathbf{1}/2)$, where F is the multilinear extension of f. Similarly, when f is supermodular, the MAP inference problem in an SPP corresponding to submodular minimization, which is polynomial time in the unconstrained setting and admits bounded approximation factors in the constrained setting [26, 1].

3.8 Properties of SPPs

Submodularity: SPPs are submodular by definition.

Restriction: Restrictions of SPP's are also SPP's. Hence if a set Y is distributed as a submodular point process over 2^V, the sets $Y \cap T$ are also distributed as SPP's over 2^T. The reason for this is that the contraction of a submodular function is also submodular. Furthermore, since $Z_f(A, B)$ is computable in polynomial time, so is $Z_f(T \cap A, T \cap B)$.

Complements: A complement process of a SPP is also an SPP. This is closely related to the submodular dual $f^{\#}$ of a submodular function [15]. The main reason is that $f^c(X) = f(V \backslash X)$ is also submodular, for submodular f. Furthermore, if $Z_f(A, B)$ is computable in polynomial time, so is $Z_{f^c}(A, B)$.

Combinations of SPPs: If $\mathcal{P}_1$ and $\mathcal{P}_2$ are SPP's, so is $\lambda\mathcal{P}_1 + (1-\lambda)\mathcal{P}_2$ (with $0 \leq \lambda \leq 1$). Similarly if two submodular functions f_1, f_2 generate SPP's, so is the combination $\lambda f_1 + (1-\lambda)f_2$. This means that SPPs are closed under mixtures which, as we shall see, are an important characteristic for these distributions.

Log Submodularity: Not only are the SPP's submodular, but many of them are log submodular (like DPPs). In particular, it is known that given a monotone non-negative submodular function f, $\log f(X)$ and $\log(1 + f(X))$ is also monotone submodular.

4 Learning Mixtures of Submodular Functions

In many machine learning applications [39, 40, 42, 43, 47, 13, 52] the optimization objective utilizes a weighted combination of submodular components (as mentioned above). These components each capture certain aspects of the problem, and a weighted combination allows a learner, by choosing the weights, to decide which aspect, or combination thereof, is most important. In these settings, the goal is to learn the weights using a large margin learning approach [42, 48, 14, 52]. In this section, we show how one can learn mixtures of submodular functions via SPPs and Log-SPPs as well.

Learning using SPPs: We can define maximum likelihood learning for SPPs. We consider a supervised setting, where, as in [42, 37], we have training pairs $\{X^t, Y^t\}_{t=1}^T$, where $X^t \in \mathcal{X}$ is an input and $Y^t \in 2^{\mathcal{Y}(X_t)}$ corresponds to a chosen subset (which, e.g., could be a human generated summary, in the case of summarization). $\mathcal{Y}(X)$ stands for the ground set associated with input X. We can define the conditional probability as $\mathcal{P}(Y|X) \propto f(Y), \forall Y \subseteq \mathcal{Y}(X)$. We may define a simple mixture model, where $\forall i, w_i \geq 0$ and $\sum_{i=1}^m w_i = 1$, as:

$$\mathcal{P}^{mix}(Y|X, w) = \sum_{i=1}^m w_i \mathcal{P}_i(Y|X) = \sum_{i=1}^m w_i \frac{f_i(Y)}{Z_{f_i}(\emptyset, \mathcal{Y}(X))}$$

Maximum likelihood parameter estimation becomes:

$$\begin{aligned} w^* &= \underset{w: \forall i, w_i \geq 0, \sum_i w_i = 1}{\text{argmax}} \sum_{t=1}^T \log \mathcal{P}^{mix}(Y^t|X^t|w), \\ &= \underset{w: \forall i, w_i \geq 0, \sum_i w_i = 1}{\text{argmax}} \sum_{t=1}^T \log \sum_{i=1}^m w_i \frac{f_i(Y^t)}{Z_{f_i}(\emptyset, \mathcal{Y}(X^t))} \end{aligned}$$

This is a simple concave maximization subject to a simplex constraint, which can be easily done using one of a variety of convex optimization methods.

Learning with Log-SPPs: We assume the same setting: we have training samples $\{X^t, Y^t\}_{t=1}^T$, where $X^t \in \mathcal{X}$ is an input and $Y^t \in 2^{\mathcal{Y}(X)}$ is a subset. Define:

$$\forall Y \subseteq \mathcal{Y}(X), \;\; \mathcal{P}^{mix}(Y|X, w) \propto \exp(\sum_{i=1}^m w_i f_i(Y)),$$

where $\forall i, w_i \geq 0, \sum_{i=1}^m w_i = 1$. Define $Z(w, \mathbf{f}, \mathcal{X}^t) = \sum_{X \subseteq \mathcal{Y}(X^t)} \exp(\sum_{i=1}^m w_i f_i(X))$.

The ML estimation then becomes,

$$\begin{aligned} w^* &= \underset{w: \forall i, w_i \geq 0, \sum_i w_i = 1}{\text{argmax}} \sum_{t=1}^T \log \mathcal{P}^{mix}(Y^t|X^t, w), \\ &= \underset{w: \forall i, w_i \geq 0, \sum_i w_i = 1}{\text{argmax}} \sum_{t=1}^T \{\sum_{i=1}^m w_i f_i(Y^t) - \log Z(w, \mathbf{f}, \mathcal{X}^t)\} \end{aligned}$$

This is again a concave maximization problem. Unfortunately however, though $Z(w, \mathbf{f}, \mathcal{X}^t)$ is a concave function in w, evaluating it requires an exponential sum. Given an approximation $\hat{Z}(w, \mathbf{f}, \mathcal{X}^t)$, such that $|\log \hat{Z}(w, \mathbf{f}, \mathcal{X}^t) - \log Z(w, \mathbf{f}, \mathcal{X}^t)| \leq \alpha$, we can solve the optimization problem with the poly-time approximation $\hat{Z}(w, \mathbf{f}, \mathcal{X}^t)$ (computable, say using sub/supergradients [25, 10]). The ML estimation problem can be solved up to an additive approximation factor of α.

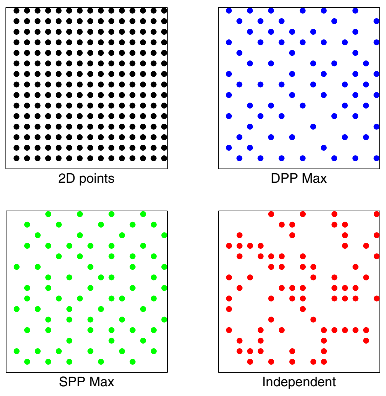

Figure 1: MAP estimates obtained with DPPs and SPPs for points on a 2D grid.

5 Simulation

Figure 1 shows the MAP estimates obtained with DPPs and SPPs. We see that the resulting subsets are similar, and capture diversity and coverage. The MAP estimates shows the result using the saturated coverage function, but most of the submodular functions discussed here capture the same intuition.

In looking at samples of both SPPs and Log-SPPs (e.g., DPPs), we noticed empirically a fundamental difference that is explained by their definitions. SPPs are directly proportional to the submodular function, and thus very often the probabilities themselves have a relatively low dynamic range. Log-Submodular distributions, like DPPs, on the other hand, have a high dynamic range since the probabilities are proportional to the exponential. Indeed, a large number of statistical and probabilistic models are defined via exponentials, and thus also have high dynamic range. This is particularly useful in sampling and inference, due to the high confidence in their decisions and concentration of their distributions. This property however can also sometimes be undesirable. For example, in the multi-class classification setting, classifiers built using low entropy distributions can be overconfident of their decisions (whether right or wrong), thereby motivating investigation of smoother transitions via linear models [7, 45].

6 Discussion

In this paper, we introduced a novel class of point processes, which we called the Submodular Point Processes (SPPs), which are distinct from the Log-Submodular and Log-Supermodular distributions (Log-SPPs) studied in literature [10]. While SPPs have properties analogous to Log-SPPs (like DPPs and Ising models), from a model perspective, there are significant differences.

One important distinction between SPPs and Log-SPPs is in handling mixtures. Most machine learning applications (like for example, summarization, and subset selection), do not inherently define single submodular functions, both most often, are modeled via a mixture of submodular functions. Handling and learning mixtures of submodular functions is very important, in considering models for submodular functions. SPPs are closed under mixtures (viz. positive conic combinations), and hence all the attractive properties for inference still hold. This is not true with Log-SPPs. In particular, given submodular f_1, f_2 which are both Log-SPPs, one particular characterization of a mixture distribution is $\mathcal{P} \propto \exp(w_1 f_1(X) + w_2 f_2(X))$. While this is still Log-Submodular, it may not have the nice properties of f_1 and f_2 (i.e all the inference quantities like the normalization factor, etc. might not any longer be computable). On the other hand, one could define a mixture distribution as $\mathcal{P} \propto w_1 \exp(f_1(X)) + w_2 \exp(f_2(X))$. While this retains the nice properties with respect to inference, it is no longer Log submodular. In particular, this means that MAP inference is no longer guaranteed. The same holds for Log-Supermodular distributions. When seen in the context of mixtures, the low dynamic range of SPPs also makes sense. Given two submodular functions f_1, f_2 which measure two different, and possibly complementary aspects of the application, we might not want any of the individual functions to be over-confident of its selection. Furthermore, the property of conic combinations also enables us to combine Submodular and Supermodular SPPs to obtain a difference of submodular point process, which is very important from a modeling perspective.

In this paper, we show several subclasses of SPPs which admit efficient computation of the partition function. Such exact computations are available only for DPPs and approximately for Ising models. An open question is, are there other classes of Log-SPPs which admit these characterizations? Another difference between SPPs and Log-SPPs is that Log-SPPs are closed under product distributions. One can model submodular functions as *priors* in this case. This does not hold for SPPs. On a whole, SPPs provide a new class of distributions, which are distinct from and complementary to Log-SPPs and other existing point processes used in applications. While the main contribution of this paper is the introduction of this new class, and hence is primarily theoretical, in future work, we plan to test these distributions in real world applications of summarization and data subset selection.

Acknowledgments: We thank the UW-MELODI group for discussions. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1162606, and by a Google, a Microsoft, and an Intel research award. Rishabh Iyer also acknowledges support from a Microsoft Research Ph.D Fellowship Award. This work was also supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] F. Bach. Learning with Submodular functions: A convex Optimization Perspective (updated version). *Arxiv*, 2013.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *TPAMI*, 26(9):1124–1137, 2004.

[3] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A tight (1/2) linear-time approximation to unconstrained submodular maximization. *In FOCS*, 2012.

[4] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák. Maximizing a monotone submodular function under a matroid constraint. *IPCO*, 2007.

[5] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *STOC*, 2011.

[6] W. G. Cochran. *Sampling techniques*. John Wiley & Sons, 2007.

[7] K. Crammer and A. Globerson. Discriminative learning vis semidefinite probabilities. In *Uncertainty in Artificial Intelligence (UAI)*, Cambridge, MA, July 2006. AUAI.

[8] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes: Volume I & II*. Springer Science & Business Media, 2007.

[9] N. R. Devanur, S. Dughmi, R. Schwartz, A. Sharma, and M. Singh. On the approximation of submodular functions. *arXiv preprint arXiv:1304.4948*, 2013.

[10] J. Djolonga and A. Krause. From MAP to Marginals: Variational Inference in Bayesian Submodular Models. In *Neural Information Processing Society (NIPS)*, Montreal, CA, December 2014.

[11] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In *KDD*, 2009.

[12] M. Feldman, J. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, 2011.

[13] A. Fix, T. Joachims, S. M. Park, and R. Zabih. Structured learning of sum-of-submodular higher order energy functions. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3104–3111. IEEE, 2013.

[14] A. Fix, T. Joachims, S. M. Park, and R. Zabih. Structured learning of sum-of-submodular higher order energy functions. In *International Conference on Computer Vision (ICCV)*, pages 3104–3111, 2013.

[15] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

[16] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6(6):721–741, 1984.

[17] J. Gillenwater, A. Kulesza, and B. Taskar. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems (NIPS) 25*, pages 2735–2743, 2012.

[18] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *SODA*, pages 535–544, 2009.

[19] P. Gopalan, A. Klivans, R. Meka, D. Stefankovic, S. Vempala, and E. Vigoda. An fptas for# knapsack and related counting problems. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 817–826. IEEE, 2011.

[20] J. He, H. Tong, Q. Mei, and B. Szymanski. Gender: A generic diversified ranking algorithm. In *Neural Information Processing Systems (NIPS)*, pages 1151–1159, 2012.

[21] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *CVPR*, 2009.

[22] R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *In UAI*, 2012.

[23] R. Iyer and J. Bilmes. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. In *NIPS*, 2013.

[24] R. Iyer and J. Bilmes. Near Optimal algorithms for constrained submodular programs with discounted cooperative costs. *NIPS Workshop on Discrete Optimization in Machine Learning (DISCML)*, 2014.

[25] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions . In *Neural Information Processing Society (NIPS)*, 2013.

[26] R. Iyer, S. Jegelka, and J. Bilmes. Fast Semidifferential based Submodular function optimization. In *ICML*, 2013.

[27] R. Iyer, S. Jegelka, and J. Bilmes. Fast Algorithms for Submodular Optimization based on Continuous Relaxations and Rounding. In *UAI*, 2014.

[28] M. Jacobsen. *Point process theory and applications*. Springer, 2006.

[29] S. Jegelka and J. A. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*, 2011.

[30] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the ising model. *SIAM Journal on computing*, 22(5):1087–1116, 1993.

[31] Y. Kawahara, R. Iyer, and J. Bilmes. On approximate non-submodular minimization via tree-structured supermodularity. In *18th International Conference on Artificial Intelligence and Statistics (AISTATS-2015)*, May 2015.

[32] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer Verlag, 2004.

[33] K. Kirchhoff and J. Bilmes. Submodularity for data selection in machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, October 2014.

[34] T. Kloks. *Treewidth: Computations and Approximations*. Springer-Verlag, 1994.

[35] P. Kohli, A. Osokin, and S. Jegelka. A principled deep random field for image segmentation. In *CVPR*, 2013.

[36] A. Kulesza and B. Taskar. k-DPPs: Fixed-size determinantal point processes. In *ICML*, 2011.

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

[38] H. Lin. *Submodularity in Natural Language Processing: Algorithms and Applications*. PhD thesis, University of Washington, Dept. of EE, 2012.

[39] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. *In NAACL*, 2010.

[40] H. Lin and J. Bilmes. A class of submodular functions for document summarization. *In ACL*, 2011.

[41] H. Lin and J. Bilmes. Optimal selection of limited vocabulary speech corpora. In *Interspeech*, 2011.

[42] H. Lin and J. Bilmes. Learning mixtures of submodular shells with application to document summarization. In *UAI*, 2012.

[43] H. Lin, J. Bilmes, and S. Xie. Graph-based submodular selection for extractive summarization. In *ASRU*, 2009.

[44] O. Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, pages 83–122, 1975.

[45] J. Malkin and J. Bilmes. Ratio semi-definite classifiers. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4113–4116. IEEE, 2008.

[46] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[47] I. Simon, N. Snavely, and S. Seitz. Scene summarization for online image collections. In *ICCV*, 2007.

[48] R. Sipos, P. Shivaswamy, and T. Joachims. Large-margin learning of submodular summarization models. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2012.

[49] D. Štefankovic, S. Vempala, and E. Vigoda. A deterministic polynomial-time approximation scheme for counting knapsack solutions. *SIAM Journal on Computing*, 41(2):356–366, 2012.

[50] P. Stobbe and A. Krause. Learning fourier sparse set functions. In *AISTATS*, 2012.

[51] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, pages 697–706, 2008.

[52] S. Tschiatschek, R. Iyer, H. Wei, and J. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Neural Information Processing Society (NIPS)*, Montreal, CA, December 2014.

[53] K. Wei, R. Iyer, and J. Bilmes. Fast multi-stage submodular maximization. In *ICML*, 2014.

[54] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes. Submodular subset selection for large-scale speech training data. *Proceedings of ICASSP, Florence, Italy*, 2014.

Supplementary Material

7 Generalized Partition Function for subclasses of submodular functions

7.1 Facility Location Function

Then the following theorem provides the generalized partition function of the Facility Location function:

Proposition 1. *The normalization constants of the facility location are:*

$$Z_f(A,B) = \sum_{i \in V} [s_{ij_i^0} 2^{|B\backslash A| - n_i} + \sum_{l=1}^{n_i} s_{ij_i^l} 2^{|B\backslash A| - n_i + l - 1}], Z_f^k(A,B) \qquad = \sum_{i \in V} [s_{ij_i^0} \binom{|B\backslash A| - n_i}{k - |A|} + \sum_{l=0}^{k_i - 1} s_{ij_i^{l+1}} \binom{|B\backslash A| - k_i + l}{k - |A|}]$$

where $j_i^0 = \text{argmax}_{l \in A}\, s_{il}$, n_i is the number of elements $l \in B\backslash A$, such that $s_{il} \geq s_{ij_i^0}$ and $j_i^1, j_i^2, \cdots, j_i^{k_i} \in B\backslash A$ such that $s_{ij_i^0} \leq s_{ij_i^1} \cdots \leq s_{ij_i^{k_i}}$.

Proof. We want to compute $\sum_{A \subseteq X \subseteq B} \sum_{i \in V} \max_{j \in X} s_{ij} = \sum_{i \in V} \sum_{A \subseteq X \subseteq B} \max_{j \in X} s_{ij}$. Now we start counting the number of times each term in the inner sum occurs. Notice that $s_{ij_i^0}$ will occur only for those $A \subseteq X \subseteq B$: $j_i^0 \in X, j_i^1, \cdots, j_i^{n_1} \notin X$, which is $2^{|B\backslash A| - n_i}$. Similarly in the case of computing $Z_f^k(A,B)$, we need the number of sets of size k satisfying this property. It's easy to see that this is $\binom{|B\backslash A| - n_i}{k - |A|}$. Now to obtain the contribution of $s_i j_i^l$, for $j_i^l \in B\backslash A$, this represents the number of sets $A \subseteq X \subseteq B : j_i^{l+1}, \cdots j_i^{n_i} \notin X, j_i^l \in X$. This is again $2^{|B\backslash A| - n_i + l - 1}$. The corresponding term in $Z_f^k(A,B)$ can similarly be obtained. Observe also that these are the only terms in the sum, which occur. ☐

7.2 Graph Cut Function

Proposition 2. *The generalized partition function corresponding to the graph-cut function is*

$$Z_{gc}(A,B) = 2^{|B\backslash A|}[M + \lambda \sum_{i \in V\backslash B, j \in A} s_{ij}] + \sum_{i \in V\backslash B, j \in B\backslash A} s_{ij} \lambda 2^{|B\backslash A| - 1} + (\lambda - \frac{\mu}{2})[\sum_{i \neq j \in B\backslash A} s_{ij} 2^{|B\backslash A| - 1}$$
$$+ \sum_{i \in A, j \in B\backslash A} s_{ij} 2^{|B\backslash A|}] + (\lambda - \mu)[\sum_{i,j \in A} s_{ij} 2^{|B\backslash A|} + 2^{|B\backslash A| - 1}\{\sum_{i \in A, j \in B\backslash A} s_{ij} + \sum_{i = j \in B\backslash A} s_{ij}\}]$$
$$Z_{sp}^k(A,B) = \lambda \binom{|B\backslash A|}{k - |A|} \sum_{i \in V\backslash B, j \in A} s_{ij} + \sum_{i \in V\backslash B, j \in B\backslash A} s_{ij} \binom{|B\backslash A| - 1}{k - |A|} + (\lambda - \frac{\mu}{2})[\sum_{i \neq j \in B\backslash A} s_{ij} \binom{|B\backslash A| - 1}{k - |A|} + \sum_{i \in A, j \in B\backslash A} s_{ij} 2 \binom{|B\backslash A|}{k - |A|}]$$
$$+ (\lambda - \mu)[\sum_{i,j \in A} s_{ij} \binom{|B\backslash A|}{k - |A|} + \sum_{i \in A, j \in B\backslash A} s_{ij} \binom{|B\backslash A| - 1}{k - |A|}] + M \binom{|B\backslash A|}{k - |A|} + \sum_{i = j \in B\backslash A} s_{ij} \binom{|B\backslash A| - 1}{k - |A|}$$

Proof. Similar to the facility location case, we can obtain this via counting arguments. In particular, consider the three terms of the expression individually. The normalization factor corresponding to M is $M2^{|B\backslash A|}$. The second term $\lambda \sum_{i \in V, j \in X} s_{ij}$ is a simple modular function, and the normalization function is $\sum_{i \in V} \lambda [\sum_{j \in A} s_{ij} 2^{|B\backslash A|} + \sum_{j \in B\backslash A} s_{ij} 2^{|B\backslash A| - 1}$. The last term is the most complicated $-\mu \sum_{i,j \in X} s_{ij}$. Here, we consider four cases, $i, j \in A$, $i \in A, j \in B\backslash A$, $i \in B\backslash A, j \in A$ and $i, j \in B\backslash A$. We expand each term, and get the resulting expressions. The same holds for the cardinality constrained case as well. ☐

7.3 Saturated Coverage Function

The following theorem provides the generalized partition function for saturated coverage function. It uses a fundamental result in knapsack counting.

Proposition 3. *The normalization terms for the Truncated Sum Point Processes can be obtained as:*

$$Z_{ts}(A,B) = \sum_{i \in V} [\{\mathcal{N}_i s_{ij} + \sum_{j \in B\backslash A} s_{ij} \mathcal{N}_{ij} + (2^{|B\backslash A|} - \mathcal{N}_i)\alpha_i\} I(\sum_{j \in A} s_{ij} \leq \alpha_i) + I(\sum_{j \in A} s_{ij} \geq \alpha_i) 2^{|B\backslash A|} \alpha_i]$$
$$Z_{ts}^k(A,B) = \sum_{i \in V} [\{\mathcal{N}_i^k s_{ij} + \sum_{j \in B\backslash A} s_{ij} \mathcal{N}_{ij}^k + (\binom{|B\backslash A|}{k - |A|} - \mathcal{N}_i^k)\alpha_i\} I(\sum_{j \in A} s_{ij} \leq \alpha_i) + I(\sum_{j \in A} s_{ij} \geq \alpha_i) \binom{|B\backslash A|}{k - |A|} \alpha_i]$$

where $\mathcal{N}_{ij}$ is the number of knapsack solutions of $\sum_{l \in Y\backslash j} s_{il} \leq \alpha_i - \sum_{l \in A \cup j} s_{il}$, $\mathcal{N}_i$ is the number of knapsack solutions of $\sum_{l \in Y} s_{il} \leq \alpha_i - \sum_{l \in A} s_{il}$ for $Y \subseteq B\backslash A$ and $\mathcal{N}_{ij}^k$ and $\mathcal{N}_i^k$ are the same as above, with additional constraints that $|Y| = k$.

Proof. We follow the same proof technique to show this result. First, notice that $\sum_{A \subseteq X \subseteq B} \sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \alpha_i\} = \sum_{i \in V} \sum_{A \subseteq X \subseteq B} \min\{\sum_{j \in X} s_{ij}, \alpha_i\}$, by exchanging the sums. Next, we consider two cases, for each i: whether $\sum_{j \in A} s_{ij} \geq \alpha_i$ or not. In the first case, the normalization factor is simple $2^{|B \backslash A|}\alpha_i$ since the function is already saturated for this i. If item i is not saturated at A, consider the coefficients for s_{ij}, for $j \in A$ and $j \in B \backslash A$. For $j \in A$, the number of times s_{ij} occurs is exactly $\mathcal{N}_i$. Similarly, for $j \in B \backslash A$, it occurs $\mathcal{N}_{ij}$ times. Finally, we also consider the number of times α_i occurs, which is exactly $2^{|B \backslash A|} - \mathcal{N}_i$. Similar arguments hold for the cardinality constrained case. □

The fundamental bases in computing the normalization terms, is computing the coefficients $\mathcal{N}_i, \mathcal{N}_{ij}$ and $\mathcal{N}_i^k, \mathcal{N}_{ij}^k$. This is the knapsack counting problem, which is #P complete in general, so we can hope to exactly compute these expressions. Fortunately, as shown in [49], the knapsack counting problem can be approximately solved to a fraction of $1 + \epsilon$. Formally, given a knapsack problem with weights $w_1, w_2, \cdots, w_n$, there exists a deterministic algorithm, which for any $\epsilon > 0$ finds $\hat{Z}$ such that $Z \leq \hat{Z} \leq (1 + \epsilon)Z$, where Z is the number of sets $X : w(X) \leq C$, for any real number C. The running time of this algorithm is $O(\frac{n^3}{\epsilon} \log \frac{n}{\epsilon})$. Observe that $\tilde{\mathcal{N}}_{ij}$ and $\tilde{\mathcal{N}}_i$ can be obtained in polynomial time $\forall i, j$ and hence we can compute $\tilde{Z}_{ts}(A, B)$ such that $Z_{ts}(A, B) \leq \tilde{Z}_{ts}(A, B) \leq (1 + \epsilon) Z_{ts}(A, B)$ in $O(\frac{n^5}{\epsilon} \log \frac{n}{\epsilon})$. However of computation $\mathcal{N}_{ij}^k$ and $\mathcal{N}_{ij}^k$ are little trickier. It is conceivable that the algorithms in [49] can be extended to this problem as well. However we can use the result from [19], where they show that there exists an FPTAS for solving a multidimensional knapsack counting problem. We can cast the problem of finding $\mathcal{N}_{ij}^k$ as a multidimensional knapsack problem, by finding the number of solutions of $\sum_{l \in Y \backslash j} s_{il} \leq \alpha_i - \sum_{l \in A \cup j} s_{il}$ and $|X| \leq k$. We however need the number of solutions such that $|X| = k$, and to do this, we run another round of the algorithm to find the number of knapsack solutions of $|X| \leq k - 1$. Both can be found in polynomial time, and hence taking the difference of the two we obtain an approximation of Z_{ts}^k within a relative error of $1 \pm \epsilon$ approximation.[3] The normalization factors Z_f and Z_f^k for the Truncated Sum Point Process can be given as:

$$Z_f = \sum_{i \in V} \{\sum_{j \in V} s_{ij} \mathcal{N}_{ij} + (2^n - \mathcal{N}_i)\alpha_i\}, \;\; Z_f^k = \sum_{i \in V} \{\sum_{j \in V} s_{ij} \mathcal{N}_{ij}^k + (\binom{n}{k} - \mathcal{N}_i^k)\alpha_i\} \tag{11}$$

7.4 Set Cover Function

Proposition 4. *The normalization terms corresponding to the coverage function are:*

$$Z_{cov}(A, B) = \sum_{w \in W} c_w [2^{|B \backslash A|} - 2^{|B \backslash A| - |B \cap \Gamma^{-1}(w)|} I(A \cap \Gamma^{-1}(w) = \emptyset)]$$

$$Z_{cov}^k(A, B) = \sum_{w \in W} c_w [\binom{|B \backslash A|}{k - |A|} - \binom{|B \backslash A| - |B \cap \Gamma^{-1}(w)|}{k - |A|} I(A \cap \Gamma^{-1}(w) = \emptyset)]$$

Proof. Again, we can obtain this with very similar counting arguments. For each concept $w \in W$, count the number of times it occurs in the sum. □

Notice that this normalization constant can be obtained via a single sweep through the elements in W. The normalization factors Z_f and Z_f^k are:

$$Z_{cov} = \sum_{w \in W} c_w \{2^n - 2^{n - |\Gamma^{-1}(w)|}\}, \; Z_{cov}^k = \sum_{w \in W} c_w \{\binom{n}{k} - \binom{n - |\Gamma^{-1}(w)|}{k}\} \tag{12}$$

7.5 Probabilistic Coverage Functions

Proposition 5. *The normalization constant for the probabilistic coverage functions is,*

$$Z_f(A, B) = \sum_{i \in \mathcal{U}} w_i \{2^{|B \backslash A|} - \prod_{j \in A} (1 - p_{ij}) \prod_{j \in B \backslash A} (2 - p_{ij})\}, \tag{13}$$

$$Z_f^k(A, B) = \sum_{i \in \mathcal{U}} w_i \{\binom{|B \backslash A|}{k} - \prod_{j \in A} (1 - p_{ij}) S_i^k(A, B)\}, \tag{14}$$

where $S_i^k(A, B) = \sum_{X \subseteq B \backslash A, |X| = k} \prod_{j \in X} (1 - p_{ij}$. Moreover, $S_i^k(A, B)$ can be computed recursively as,

$$S_i^k(A, B) = S_i^{k-1}(A, B \backslash l)(1 - p_{il}) + S_i^k(A, B \backslash l), \textit{for any } l \in B \backslash A. \tag{15}$$

Proof. The first part of the proof follows from the fact that $\sum_{X \subseteq V} \prod_{j \in X}(1 - m_j) = \prod_{j \in V}(1 - m_j)$. In the second part, the recursion follows by considering two cases, $l \in X$ and $l \notin X$. □

We can easily compute $S_i^k(A, B)$ by considering any order of the elements in $B \backslash A$.

8 Sampling from discrete distributions

In this section, we investigate an algorithm for sampling from a discrete distribution, given the generalized partition function.

Algorithm 1 Algorithm for Sampling from a Discrete Point Process $\mathcal{P}$

```
Input: A random ordering $v_1, v_2, \cdots, v_n$ of elements in $V$
Output: A set $Y \sim \mathcal{P}$
  $J = \emptyset, J^c = V, \delta = 0$.
  Choose random $x \in [0, 1]$.
  for $i = 1, 2, \cdots, n$ do
    if $\delta + \mathcal{P}(J \cup v_i \subseteq Y \subseteq V^c) > x$ then
      $J := J \cup v_i$
    else
      $J^c := J^c \backslash v_i$
      $\delta := \delta + \mathcal{P}(J \cup v_i \subseteq Y \subseteq V^c)$
    end if
  end for
  Return $J$
```

The sampling algorithm is shown in Algorithm 1. It is based on the simple discrete sampling procedure, where we divide the region of $(0, 1)$ into bins, the size of which is proportional to the probability. However since there are 2^n such bins, a naïve sampling procedure would be exponential. We however arrange the bins in a manner to exploit the fact that the conditionals and marginals can be computed in polynomial time. Given any ordering of elements $v_1, \cdots, v_n$, we arrange the bins so that all sets containing v_1 occur first. Within those sets containing v_1, the sets containing v_2 occur first and so on. Similarly, in the region of bins not containing v_1, we arrange the bins so that the sets containing v_2 are first and so on. Once we arrange the sets in this manner, we can easily find the bin containing the random point x through a binary search like trick. At the start of the algorithm, we check if $\mathcal{P}(v_1 \in Y) > x$. If this is true, we can conclude, based on our ordering that set corresponding to the bin containing x contains v_1. On the other hand, if it is not true, this means that the set does not contain v_1. If we assume that the latter is true, then we check if the set contains v_2. However due to the ordering, we need to add the offset $\delta = \mathcal{P}(v_1 \in Y)$, and hence we can check if $\delta + \mathcal{P}(v_2 \subseteq Y \subseteq V \backslash v_1) > x$. The algorithm continues in this manner, and notice that after $O(n)$ steps, we can obtain the set Y, which is distributed according to $\mathcal{P}$. Furthermore, since the computation of marginals and conditionals are relatively cheap in SPP's (for example $O(n^2)$ in the sum-penalty process), we can efficiently sample from these distributions in polynomial time.

A key quantity in the sampling algorithm is the conditional probability, which is easy to compute in the case of SPPs. In particular, notice that $\mathcal{P}(v_i \in Y | J \subseteq Y \subseteq J^c) = \frac{\mathcal{P}(v_i \in Y)}{\mathcal{P}(J \subseteq Y \subseteq J^c)} = \frac{\mathcal{P}(v_i \in Y)}{Z_f^C(J, J^c)}$.