Submodular Point Processes

Rishabh Iyer
Department of Electrical Engineering
University of Washington
rkiyer@u.washington.edu

Jeff Bilmes
Department of Electrical Engineering
University of Washington
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. 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, while in maximization problems, they model aspects of coverage, diversity, and information. A set function $f : 2^V \rightarrow \mathbb{R}$ is submodular if $\forall S, T \subseteq V$, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. An equivalent characterization, which many real world models naturally exhibit, 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 [7, 13, 15, 29], 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) [8] and Determinantal Point Processes [20, 27]. Recently, [5] 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: $P(X) \propto f(X)$, for $X \subseteq V$, 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 [5] are defined as: $P(X) \propto \exp(f(X))$. where f is submodular (or supermodular). Determinantal Point Processes (DPPs) [20, 27] are special cases of Log-submodular distributions, while Ising models are special cases of Log-supermodular distributions.

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 $O(\log n)$. **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. **4)** Finally, we argue that while the SPPs are similar to the Log-SPP models from a modeling perspective, they have several key differences from these models, thereby providing a complementary class of models. In particular, we argue that SPPs form a class of models natural for modeling submodular mixtures.

2 Probabilistic Inference

We here investigate the computation of the partition function for SPPs, and correspondingly, the conditionals and marginals. The partition function corresponding to SPPs is $Z_f = \sum_{X \subseteq V} f(X)$. Similarly, the partition function for Log-SPPs can be defined as $Z_f = \sum_{X \subseteq V} \exp(f(X))$. We first investigate the hardness of probabilistic inference (in particular, 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.

2.1 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 [5]. In terms of hardness, the partition function computation was known to be $\#P$ hard [17]. 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 [5] 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 [5] where they show that submodular sub- and super-gradients [15] 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. Furthermore, [5] also provides approximation guarantees, which in the worst case is $O(n)$, and depends on the function valuation. 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 $P = NP$ question).*

Proof in [1]. 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 in [1].*

Similar to Log-SPPs, we can use the sub and super-gradients to provide upper and lower bounds for the partition function. Note that the sub/super gradients provide modular functions $m_l(X) + c_l$ and $m_u(X) + c_u$ such that $m_l(X) + c_l \leq f(X) \leq m_u(X) + c_u$. 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

$$\hat{Z}_f = \sum_{X \subseteq V} \hat{f}(X). \tag{1}$$

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 $O(n)$ [14], implying a $O(\log n)$ approximation guarantee. Furthermore, a coverage function [4, 9] approximates a monotone submodular function within a factor of $O(\sqrt{n})$ [4, 9], which again provides a $O(\log n)$ approximation guarantee to the log-partition function.[1] 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 $O(\log n)$. □

2.2 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 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. In the interest of space, we just provide expressions for Z_f and Z_f^k, and defer the exact expressions and corresponding proofs for computing the generalized partition function $Z_f(A, B)$ and $Z_f^k(A, B)$ to the supplement [1].

2.2.1 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., $0 \leq s_{ij} \leq 1$).

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 [21], image summarization [30] and data subset selection [25]. 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: $Z_{fac} = \sum_{i \in V} \sum_{l=1}^{n} 2^{l-1} s_{ij_i^l}$.

where j_i^l is as defined in [1]. We can also generalize this to the k-facility location case [24], 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 [25, 21]) as well in image segmentation and denoising (by capturing cooperation [2]). This general class can be defined as: $f(X) =$

[1] 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 $O(n)$.

$M + \lambda \sum_{i \in V} \sum_{j \in X} s_{ij} - \mu \sum_{i,j \in X} s_{ij}$; $\mu = \lambda = 1, M = 0$ is the standard graph cut, and $\lambda = 0$ gives the redundancy penalty [25]. $M \geq 0$ 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 the facility location or asymmetric graph cut etc.) to define an objective for summarization. This has been used, for example, with the facility location and asymmetric graph cut [30, 25, 10]). Define, $\mathcal{P}_{gc}(X) \propto M + \lambda \sum_{i \in V} \sum_{j \in X} s_{ij} - \mu \sum_{i,j \in X} s_{ij}$, where λ, μ, M are appropriately chosen so that the objective is non-negative. This function is monotone for $\lambda > 2\mu$. Define $S = \sum_{i,j \in V} s_{ij}, S^d = \sum_{i \in V} s_{ii}$. The normalization constants for these processes have a simple expression: $Z_f = 2^n M + (2\lambda - \mu) 2^{n-2} S - 2^{n-2} \mu S^d$ This class of point processes can 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 [22]. 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 [18]. Fortunately, it can be approximated to an arbitrary factor close to one, by using an fully polynomial time approximation scheme (FPTAS) for knapsack counting. In the interest of space, we defer the formal result to the supplement [1].

2.2.2 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 $c : 2^W \rightarrow \mathbb{R}_+$, the set cover function is defined as $f_{cov}(S) = c(\Gamma(S))$. This function simultaneously models aspects of coverage [26] in maximization, and the notion of complexity (like the size of the vocabulary in a speech corpus) in minimization problems [23]. 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 $\Gamma(v) = w$. 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 is: $Z_{cov} = \sum_{w \in W} c_w [2^n - 2^{n - |\Gamma^{-1}(w)|}]$.

Probabilistic Coverage Functions: This is a generalization of the set cover function, which has been used in a number of models for summarization problems [6]. 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 $Z_f = \sum_{i \in \mathcal{U}} w_i [2^n - \prod_{j \in V} (2 - p_{ij})]$.

2.2.3 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 $Z_f = 2^{n-1} m(V)$.

Log-Modular distributions & Poisson Processes: A related class of distributions is $f(X) = e^{-m(X)}$ 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 Poisson 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., $Z_f = 1$).

2.2.4 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 [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 [22], and notions of cooperation in minimization problems [16, 12]. 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 $1+\epsilon$ [31]. Moreover, since it is possible to approximate any concave function with a truncation up to any desired factor [19], 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 $O(|X|^{1-a})$ [14]. Since $\hat{f}$ is a modular function, and following Lemma 3, it is easy to see that the resulting approximation factor is $(1-a)\log n$.

2.2.5 Sparse Pseudo-Boolean functions

For graphical models, in particular in computer vision, set functions are often written as polynomials [11]. 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 [32, 11]. 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 $F(\mathbf{1}/2) = \sum_{X \subseteq V} f(X)/2^n = Z_f/2^n$.

2.2.6 Fourier Sparse Submodular Functions

A class of set functions introduced in [32] – 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 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 (like DPPs and Ising Models). SPPs have properties analogous to DPPs (when defined via submodular functions), and Ising models (when defined via supermodular functions), in that they both capture notions of coverage, diversity and cooperation.

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 [3, 28].

Another very important distinction between SPPs and Log-SPPs is 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. Correspondingly, handling and learning mixtures of submodular functions is very important, in considering models for submodular functions. SPPs are closed when taking mixtures, since a mixture of base SPPs is also a SPP, 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 overconfident of its selection.

On a whole, SPPs provide a new class of distributions, which are distinct from and complementary to 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.

References

[1] Anthors. Submodular Point Processes: Extended Version. 2015.

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

[4] Nikhil R Devanur, Shaddin Dughmi, Roy Schwartz, Ankit Sharma, and Mohit Singh. On the approximation of submodular functions. *arXiv preprint arXiv:1304.4948*, 2013.

[5] Josip Djolonga and Andreas Krause. From map to marginals: Variational inference in bayesian submodular models. In *Neural Information Processing Systems (NIPS)*, 2014.

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

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

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

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

[10] 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.

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

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

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

[14] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions . In *NIPS*, 2013.

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

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

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

[18] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer Verlag, 2004.

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

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

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

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

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

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

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

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

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

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

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

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

[31] 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.

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

[33] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. *Proceedings of ICASSP, Florence, Italy*, 2014.